Oracle® WebCenter Framework

Developer's Guide

10*g* (10.1.3.2.0)

B31074-05

January 2008

ORACLE®

Oracle WebCenter Framework Developer's Guide, 10*g* (10.1.3.2.0)

B31074-05

Primary Author: Peter Lubbers

Contributing Authors: Frank Rovitto, Vanessa Wang, Joan Carter, Lalithashree Rajesh, Promila Chitkara, Rosie Harvey, Marcie Caccamo, Sue Highmoor

Contributors: Alison Macmillan, Barry Hiern, Candace Fender, Chris Broadbent, Dragos Harabor, Hamsa Gopalan, Harry Wong, Harshivl Shah, Ian Penning, James Owen, Janet Blowney, Jeff Tang, Jeni Ferns, Karthic Lakshmanan, Lei Oh, Ling Cheng, Madhup Gulati, Madhwa Chintarevula, Mahima Subbaraman, Manish Devgan, Medini Kakade, Mick Andrew, Oliver Ricordel, Paul Encarnacion, Paul Lin, Paul Spencer, Peter Moskovits, Philipp Weckerle, Preeti Yarashi, Ravi Baranwal, Seshan Kannan, Stewart Wilson, Sue Vickers, Vicki Chun, Vineet Duggal, Yueh-hong Lin, Alistar Wilson

Contents

Part II Building a WebCenter Application

3 Preparing Your Development Environment

4 Populating Pages

5 Integrating Content

6 Integrating Oracle WebCenter Wiki

10 Securing Your WebCenter Application

11 Working Productively in Teams

13 Monitoring Your WebCenter Application

Part IV Building Portlets

14 Understanding Portlets

15 Portlet Technologies Matrix

16 Creating Portlets with OmniPortlet

17 Creating Content-Based Portlets with Web Clipping

18 Creating Java Portlets

19 Enhancing Java Portlets

Part V Appendixes

A Reuse of OracleAS Portal Components

B Additional Portlet Configuration

C Files for WebCenter Applications

D Manually Packaging and Deploying PDK Portlet Producers

E Administering Oracle WebCenter Wiki

F Node Type Definitions for Oracle WebCenter Adapters

G Troubleshooting WebCenter Applications

Index

List of Examples

List of Figures

List of Tables

Preface

This guide provides in-depth information for all of the following tasks:

- How to plan, build, deploy, and manage a WebCenter application with Oracle WebCenter Suite.
- How to administer, monitor, and maintain a WebCenter application and all of its associated components with Oracle WebCenter Suite.
- How to plan, build, deploy, and manage portlets for an application.

> **Note:** For the portable document format (PDF) version of this manual, when a URL breaks onto two lines, the full URL data is not sent to the browser when you click it. To get to the correct target of any URL included in the PDF, copy and paste the URL into your browser's address field. In the HTML version of this manual, you can click a link to directly display its target in your browser.

Audience

This guide is written for all of the following users:

- The Oracle Application Server Portal (OracleAS Portal) application developer, who wants to build a WebCenter application.
- The OracleAS Portal site administrator, who wants to maintain the deployed WebCenter application.
- The component developer, who wants to build portlets.

This guide assumes that the audience has already read the *Oracle Application Development Framework Developer's Guide* and is familiar with the following concepts:

- Java
- Oracle JDeveloper
- Java Server Faces
- Oracle Application Development Framework (Oracle ADF) (purpose, basic architecture, basic development skills)
- Oracle ADF Faces components
- Oracle Application Server and Oracle Containers for J2EE (OC4J)

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documents

For more information, see the following documents in the Oracle Application Server 10*g* (10.1.3.2.0) documentation set or on Oracle Technology Network (OTN) at `http://www.oracle.com/technology/index.html`.

- *Oracle WebCenter Framework Tutorial*
- *Oracle WebCenter Framework Error Messages Guide*
- *Oracle Application Development Framework Developer's Guide*
- *Oracle Application Server Portal Developer's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Introduction to Oracle WebCenter Suite

Part I contains the following chapters:

1

Understanding Oracle WebCenter Suite

This chapter introduces you to Oracle WebCenter Suite and helps you understand how you can use it to enhance your service-oriented applications. With Oracle WebCenter Suite, you get services that you can integrate with your application to afford your users improved communication, content management capabilities, customization, and advanced search support. More important, you get a development framework that provides essential capabilities, such as the ability to consume portlets and content in a Java Server Faces application, declarative security, and life cycle management tools.

In this chapter, you will discover answers to the following key questions:

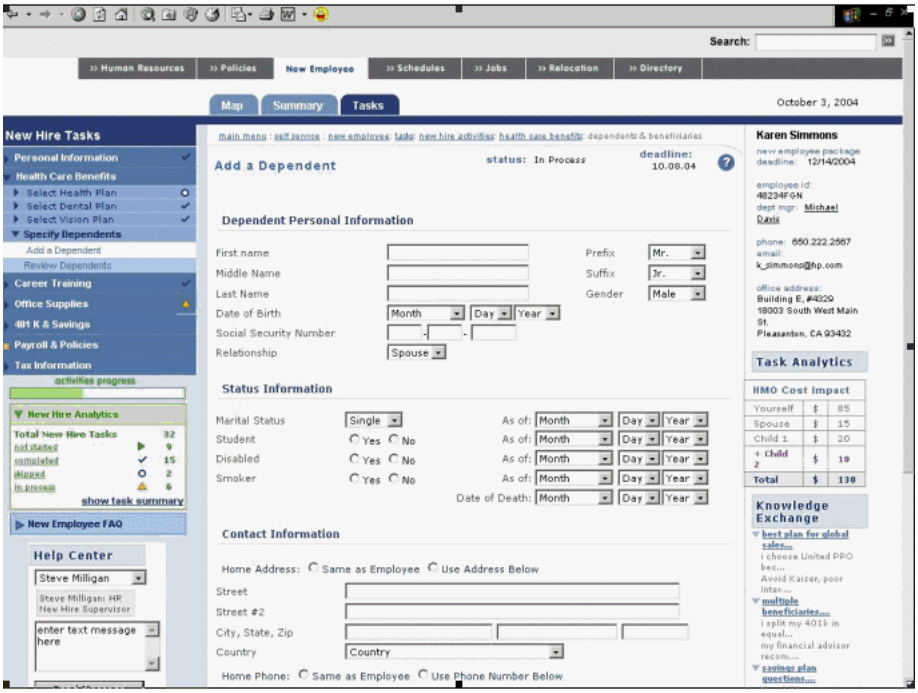
- What is Oracle WebCenter Suite?
- What Will You Learn in This Developer's Guide?

After you read this chapter, you'll be ready to start building your own Java EE 5 application.

1.1 What is Oracle WebCenter Suite?

As key technologies like Wiki, RSS, and blogs change the landscape of the Internet by empowering individuals across the globe, user demand for applications that simplify transactions becomes more pronounced. One way to simplify transactions is to provide everything the user needs to support a given task within the application itself. Consider the example shown in Figure 1–1.

Figure 1–1 Sample Application

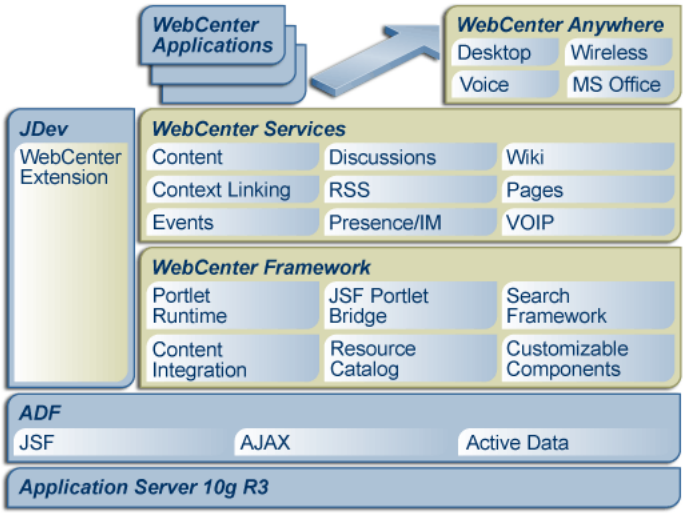


In this example, a user who is new to the company is working with an application that enables him to add dependents to his company insurance policy. Notice that the transaction itself is surrounded by additional context that helps the user, including the following:

- New Hires Tasks, in the upper left corner, provide an activity guide that shows where the user is in the larger process of becoming acclimated to his new company. The user's next task is also identified. This type of process orchestration helps the user step through the entire multistep flow quickly and easily.
- Task and process analytics let users know where they are in the process and how decisions are affecting them. In this case, the Task Analytics on the right show the total cost effect of the benefit choices made so far.
- The Help Center on the bottom left provides an up to date FAQ for quick access to typical questions and a direct chat link to the help center where the user can ask additional questions not addressed by the FAQ. Again, no need for the user to leave the context of the transaction to get help.
- Knowledge Exchange, on the bottom right, provides documentation relevant to the current task. These documents, stored in the corporate repository, give detailed advice on the different beneficiary and dependent scenarios applicable to the user.

Until Oracle WebCenter Suite, building this kind of application was a rather tedious process. To gain access to the beneficiary scenarios, for example, used to involve creating a portlet to gain a view into the JCR 1.0 Java Content Repository (JSR 170)--if the application programming interface (API) required to do so was available. Oracle WebCenter Suite reduces the front-end labor historically required to bring necessary business components to the user by capitalizing on the notion of Service Oriented Architecture (SOA). Thanks to Oracle WebCenter Suite's commitment to SOA, as well as to the JCR 1.0 Java Content Repository (JSR 170) and other industry standards, you get a wide range of plug-and-play products, tools, and services that make it easy to build the applications your users need. Figure 1–2 shows what Oracle WebCenter Suite provides[1]:

Figure 1–2 Oracle WebCenter Suite



Let's examine these building blocks in more detail.

1.1.1 Oracle WebCenter Framework

Oracle WebCenter Framework augments the Java Server Faces (JSF) environment by providing additional integration and run-time customization options. In essence, it integrates capabilities historically included in Oracle Application Server Portal (OracleAS Portal) products directly into the "fabric" of the JSF environment. This eliminates artificial barriers for the user and provides the foundation for developing the kinds of context-rich applications depicted in Figure 1–1.

1.1.1.1 Building and Consuming Portlets

Portlets help you bring data from the Web, database, and so on, into your application. Using Oracle JDeveloper, you can create your own standards-based portlets to be consumed by any JSR 168 or WSRP-compatible portal. The Oracle Application Server Portal Developer Kit (PDK) has been enhanced to support extended portlet capabilities as defined by WSRP 2.0 within the structure of the Java Portlet Standards APIs. From a WebCenter application, you can consume JSR 168, WSRP 1.0, WSRP 2.0 or Oracle PDK-Java portlets all within the same application, or even within the same page.

Several prebuilt portlets are available for use through a preconfigured Oracle Containers for J2EE (OC4J) that is automatically available to you through Oracle JDeveloper. Two such portlets, OmniPortlet and Web Clipping, help empower users to gather their own data, while the Rich Text portlet enables users to publish their own announcements and bulletins. You can make these portlets available to users by dropping them on your page, or you can use them yourself to create the specific portlets your users will need.

- OmniPortlet: A portlet that enables users to easily publish data from a variety of sources, using a variety of layouts. Users can base an OmniPortlet on almost any kind of data source, such as spreadsheets (character-separated values), XML, Web Services, and even application data from an existing Web page. Once the data has

[1] Some components shown are not available in the initial release of Oracle WebCenter Suite: Presence/IM, Discussions, and Wiki. This chapter provides a description of components relevant to this release of Oracle WebCenter Framework.

been obtained, they can format it using layouts such as bulleted lists, charts, HTML, and so on.

As a developer, you might want to use this tool to gather and format the data for your users--for example, to create an employee directory--then place it on your page for user consumption. Once you do so, the portlet becomes available through Oracle JDeveloper's Component Palette for others to use in their applications.

- Web Clipping: An extremely easy-to-use wizard that requires no technical expertise whatsoever. Users simply locate the Web content they want to "clip", then use the wizard to grab it and display it within their application. If the Web content on the original site is updated, then so is the user's associated Web Clipping.
- Rich Text portlet: A tool that enables users to publish their own announcements and broadcasts. When you place a Rich Text portlet on a page, during run time, authorized users can access all the rich-text editing tools needed to insert, update, and format display text.

1.1.1.2 Customizable Components

WebCenter Framework provides new JSF components that enable developers to make any of their applications customizable. These new components act as containers into which developers can drop another Faces view component or a portlet. With these capabilities in place, administrators can customize virtually any JSF page by minimizing or maximizing, hiding or showing, or moving any component on the page.

1.1.1.3 Content Integration

Suppose you have data in a content management system such as Oracle Content DB, OracleAS Portal—or even on your file system—that you want to make available to your application. WebCenter Framework provides you with the JCR adapters you must access that content. Using Oracle JDeveloper, you can then build JCR data controls to grab the content and drop it onto your page in a variety of display modes. WebCenter Framework also comes with Oracle Drive, through which you can represent the contents of your OracleAS Portal repository as a tree-like structure, right on your desktop.

1.1.1.4 Securing Your Application

With the Oracle ADF extensions provided in WebCenter Framework, you can define security for an entire application, a page within the application, or for individual actions provided by customizable components.

In many cases, it is desirable to leverage existing applications that have their own authentication mechanism, such as e-mail. WebCenter Framework provides the means to embed those applications through the use of the External Application wizard. See Chapter 10, "Securing Your WebCenter Application" for more information.

1.1.1.5 Managing Your Application Throughout the Life Cycle

WebCenter Framework reduces the time required to build, deploy, and migrate your applications through the use of several tools as follows:

- Development framework: Oracle JDeveloper and Oracle ADF provide the tools and framework you must build and update your application. Adding portlets, content, and customization capabilities to your WebCenter application is simply a matter of dragging and dropping the appropriate objects in either a source or WYSIWYG environment. To simplify the test and debug phases, WebCenter

Framework includes a deployment profile (WebCenter Application WAR) that packages and migrates your portlet customizations, content, and page customizations to any J2EE container (such as the standalone OC4J provided) so you can test and debug your application before deploying it to a production server.

- Enterprise deployment: When you're ready to deploy your application to a production environment, Oracle WebCenter Framework's Predeployment Tool packages and migrates your portlet customizations to your production location, changes the pointer to your content repository, and ensures that the application points to your production Metadata Services location. When the Predeployment Tool completes its work, you get a target EAR file that you can then deploy to the final location using Enterprise Manager.
- Standards-based administration: Browser-based tools enable administrators to deploy, configure, and manage WebCenter applications. In addition, tools built on industry standards-based JMX methods offer administrators granular control and monitoring mechanisms for health status, performance, and popularity. Tools for obtaining historical performance and status reporting over time (within a single Oracle Application Server context) are also provided. WebCenter application metrics are delivered using the familiar Application Server Control monitoring and management interface.

1.1.2 Oracle WebCenter Services

Oracle WebCenter Services offer a variety of content management, search, and communication services, including the following:

- Oracle Content Database (Oracle Content DB), the default content repository for Oracle WebCenter Services. Oracle Content DB is a full-fledged content management system that enables users to manage content through the Web or from desktop applications. A rich library of ready-to-use Web services is provided to content-enable your enterprise in a service-oriented environment. With Oracle Content DB, you can do the following:
  - Improve the productivity of individuals and teams with secure access to the right content in the context of business processes
  - Reduce risk associated with content, including information loss and legal discovery
  - Facilitate adaptability of business processes
  - Reduce Information Technology (IT) and administrative costs through content consolidation

  Oracle Content DB bridges the gap between limited capability file servers and the specialized, expensive, and complex content management applications that are so widely available.
- Oracle Secure Enterprise Search is a crawler-based service that can search a multitude of sources, structured and unstructured, in a variety of file formats, indexed or real-time. With Oracle Secure Enterprise Search, you can reduce the time spent finding relevant documents on your company's information repositories.
- Communication Services, which help you better connect people and facilitate communication. These services include the following:
  - Instant Messaging: Lets users freely exchange ideas through audio and video feeds, file exchange, and a range of other capabilities.

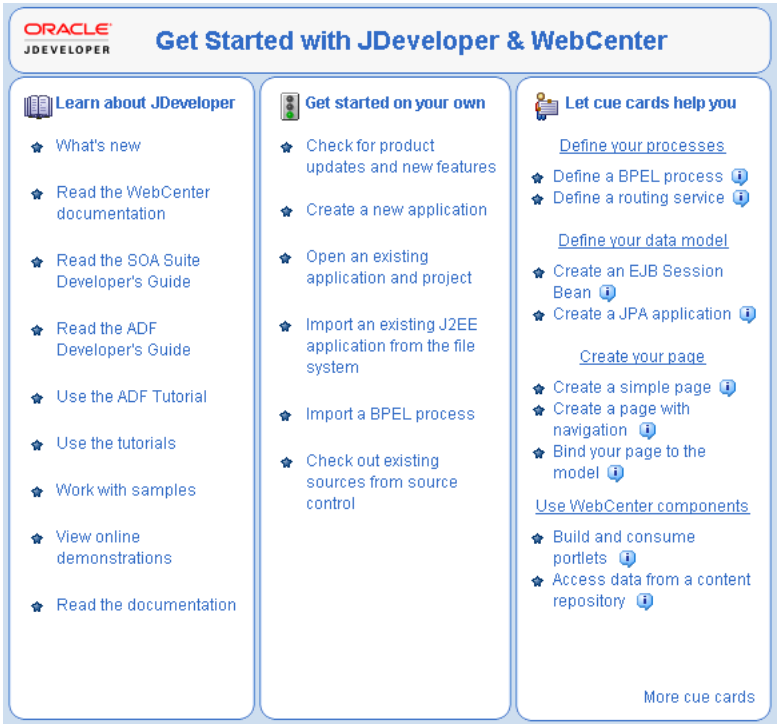
  - Presence Server: *Presence* provides information about a person's availability to every person or application that subscribes to that person's status. Chats and other real-time services can be initiated from the associated user interface.
  - Discussion forum: An interactive message board for sharing information, questions, and comments.
- Wiki is server software that enables users to freely edit and create Web page content using a Web browser. This ease of interaction and operation makes Wiki an effective tool for collaborative communication.

1.1.3 Oracle JDeveloper

Oracle JDeveloper is an integrated development environment (IDE) for building service oriented applications using the latest industry standards for Java, XML, Web services, and SQL. Oracle JDeveloper supports the complete software development life cycle, with integrated features for modeling, coding, debugging, testing, profiling, tuning, and deploying applications. Oracle JDeveloper's visual and declarative approach and Oracle ADF work together to simplify application development and to reduce mundane coding tasks. For example, code for many standard user interface widgets, such as buttons, list of values, and navigation bars, are prepackaged for you. All you have to do is select the appropriate widget from the Component Palette and drop it into your application.

As you work through this guide, you will become more familiar with Oracle JDeveloper and the advantages it offers. For more information about Oracle JDeveloper, access one of the many educational aids from the Oracle JDeveloper Start Page (Figure 1–3), accessible from Oracle JDeveloper's Help menu.

Figure 1–3 Oracle JDeveloper Start Page



1.2 What Will You Learn in This Developer's Guide?

In this Developer's Guide, you will learn about the following:

- Designing your WebCenter application.
- Preparing your development environment.
- Populating pages with Oracle WebCenter Framework.
- Integrating content from content repositories.
- Deploying applications.
- Securing applications.
- Monitoring applications.
- Choosing a portlet technology that suits your requirements.
- Building portlets with various technologies.
- Troubleshooting your WebCenter application.

2

Planning Your WebCenter Application

This chapter helps you determine the type of WebCenter application to build and explains the major considerations for building it. Before you begin building your WebCenter application, you should review this chapter carefully to determine what options are available to you and what kinds of issues you must consider before you start building.

2.1 Introduction to WebCenter Applications

Oracle WebCenter Framework provides you with a set of features (for example, portlets, customization, and content integration) that simplify the process of building a WebCenter application with Oracle ADF and deploying it.

A WebCenter application is an application that employs some or all of the following elements:

- Customizable pages
- Content (from content repositories by way of Java Content Repository data controls)
- Portlets
- Customizable components
- Collaboration tools (for example, discussion forums and instant messaging)
- Skins
- Security

After the WebCenter application has been built and tested, you must still deploy it for your end users. Once deployed and running, users will begin to access the WebCenter application and administrators to maintain it. The sections that follow provide an overview of the componentry of WebCenter applications.

2.1.1 About Customizable Pages

At its core, Oracle WebCenter Framework provides you with the functionality to extend your JSF pages with all of the following:

- **Customizable components** to rearrange content within an area or to hide or show child components.

- **Portlets** to display, personalize, and reuse dynamic content.
- **Content** retrieved from content repositories by way of Java Content Repository (JCR) data controls.

By combining these elements, you provide your users with easy access to content and collaboration capabilities. Optionally, you can also enable customization of these various page elements to make the WebCenter application more flexible. Note that administrators can customize the page for the end user, but end users cannot personalize pages. End users can only personalize portlets, assuming the portlets implement personalization and the application implements user authentication.

> **See Also:** Chapter 4, "Populating Pages" for more information about extending pages with these capabilities.

2.1.2 About Customizable Components

Customizable components enable administrators to manipulate their view according to their requirements. For example, they may choose to hide a certain component altogether or move it up to the top of the page. By adding customizable components to a page, you make the page customizable.

Customizable components include content container components called `ShowDetailFrame` and layout components called `PanelCustomizable`. You can use these components to enable administrators to rearrange, hide, or display pieces of content, and to be able to define the behavior of content on a page.

Oracle WebCenter Framework provides two customizable components:

- A `ShowDetailFrame` component that places a chrome around the components it contains, thereby enabling the administrator to minimize, maximize, or move that content.
- A `PanelCustomizable` component that enables the administrator to hide or show the child components that it contains.

In a typical scenario, you might have several `ShowDetailFrames` that each contain another component (for example, Oracle ADF Faces tables and `objectImage`). You could then wrap a `PanelCustomizable` component around these `ShowDetailFrames`, thereby making them children of the `PanelCustomizable`. In this scenario, each `ShowDetailFrame` would have its own chrome for minimize, maximize, and move. Furthermore, all of the `ShowDetailFrames` would be surrounded by chrome that enables you to hide or show the child `ShowDetailFrames`. Note that the `PanelCustomizable` provides the ability to move the contents, without it, the `ShowDetailFrame` could only be minimized or maximized.

> **See Also:** Section 4.4, "Using Customizable Components" for more information.

2.1.3 About Portlets

Oracle WebCenter Framework supports WSRP 1.0, WSRP 2.0, JSR 168, and Oracle PDK-Java for portlets. Note that WSRP 2.0 features require some Oracle extensions in this initial release of Oracle WebCenter Framework. You should also be aware that all portlets run remotely from the application in the Oracle WebCenter Framework environment, meaning there are no local portlets. You must always deploy the producer and register it with the application before consuming its portlets.

You can use the portlets that Oracle or third parties provide you, or you can create your own portlets programmatically. The prebuilt portlets that the Oracle WebCenter Framework provides include the following:

- The Rich Text portlet offers browser-based, rich text editing at run time.
- Web Clipping is a browser-based, declarative tool that enables you to integrate any Web application with your WebCenter application.
- OmniPortlet is a declarative portlet-building tool that enables you to build portlets against a variety of data sources, including XML files, character-separated value files (for example, spreadsheets), Web Services, databases, and Web pages.

Packaged applications will also often come with their own set of portlets that enable you to access particular data or functions of the application. Assuming that they were built with compatible technology (WSRP, JSR 168, or PDK-Java), you can include these portlets in your WebCenter application as well.

You can link portlets such that parameters are passed between portlets and Faces components, and between portlets and the page. In this fashion, you can create a context-sensitive application, where the data displayed by the portlets changes depending upon the page context.

See Also:

- Chapter 14, "Understanding Portlets" for information about portlets and how you might use them.
- Chapter 15, "Portlet Technologies Matrix" for information about the different ways you might create portlets.
- Chapter 4, "Populating Pages" for information about consuming portlets on pages and linking them.

2.1.4 About Content (Through JCR Data Controls)

To browse and query content repositories from your WebCenter application, you must bind the data from the repository to your application. With JCR data controls in Oracle JDeveloper and Oracle ADF, you can connect to content repositories and display their content within a WebCenter application. This functionality is based upon JSR 227, which provides a standard way of binding data from different sources to a Java user interface.

For example, you could create a data control that selects content from Oracle Content Database (Oracle Content DB), Oracle Application Server Portal (OracleAS Portal), or a file system. Once the data control is created, you can drop it onto a JSP document as a table. If you must retrieve data from a content repository other than Oracle Content DB, OracleAS Portal, or the file system, then you can create your own JCR adapter. From the Content Repository Configuration page of the Create Data Control Wizard, you choose the content repository from which you want to retrieve data.

See Also: Chapter 5, "Integrating Content" for more information about content integration.

2.1.5 About Skins

A *skin* in Oracle ADF Faces is a global style sheet that is set in one place for an entire application. Instead of having to style each component, or having to insert a style sheet on each page, you can create one skin for the entire application. Every component automatically uses the styles as described by the skin. Any changes to the skin are

picked up at run time, no code change is required. Skins are based on the Cascading Style Sheet specification and use CSS 3.0 syntax.

> **See Also:** Chapter 9, "Defining and Applying Styles to Core Customizable Components" for more information about skins and how to use them.

2.1.6 About Security

The security model of a WebCenter application can encompass a wide variety of areas. You must consider the needs of your particular environment and choose which aspects of security you ought to apply to your application as follows:

- You may want to require user login and control access to certain areas or functions based on user roles. You can define what actions are permissible, based upon roles, at a granular level (for example, pages, data iterators, attributes, and methods). You can also control whether a view component is visible to a user based upon permissions on other pages and components.
- Your WebCenter application may need to negotiate the security systems of external applications or content repositories to fetch data or content.
- You may want to configure secure identity propagation between the application and remote portlets.

> **See Also:** Chapter 10, "Securing Your WebCenter Application" for more information about options for securing your WebCenter application.

2.1.7 About Life Cycle

After you have created and tested your WebCenter application in the design time environment (Oracle JDeveloper), you must deploy it to your production system. Once deployed, you must then maintain the system. For example, you will want to monitor performance and availability, edit or refresh portlet producers, undeploy applications, and perhaps migrate customization data. Inevitably, you will also want to further enhance the application, stage it again, and then redeploy it to your production system. Using the life cycle tool and Grid Control Console, you can easily perform these tasks on your production WebCenter application.

> **See Also:**
>
> - Chapter 12, "Deploying Your WebCenter Application"
> - Chapter 13, "Monitoring Your WebCenter Application"

2.2 Design Questions to Consider Before You Start

When you come to design your WebCenter application, you must consider the needs of your audience. In particular, it's important to think about what features and capabilities your WebCenter application end users, administrators, and developers most need. The following list of common questions can help you work through this planning process. Before you begin to actually build a WebCenter application, you should think carefully about the answers to all of the following questions:

- **User Considerations**
  - **How many users are there?** If your WebCenter application must serve a large number of users, then you will must take that into account when you define

your deployment environment.You must choose a topology that can support the number of users you expect to have accessing your WebCenter application.

See Also: *Oracle Application Server Enterprise Deployment Guide* to help you decide which Oracle Application Server configuration will best support your WebCenter application.

– **Do users need to personalize their portlets?** Portlets can optionally include a personalization mode (Edit mode) that enables an authenticated user to personalize the portlet. The Edit mode might enable the user to enter things such as the portlet's title or a parameter that affects the content of the portlet. For example, you could implement an Edit mode that enables the user to enter ticker symbols for a stock portlet. Once the user's changes are applied, the portlet will display the prices for their chosen ticker symbols. Note that, if you choose to implement personalization, then you must also implement some form of security for the application consuming the portlet. The Personalize option only appears to authenticated users.

See Also: Chapter 14, "Understanding Portlets" to learn more about portlets.

– **What types of pages or layouts will best serve your users?** With Oracle WebCenter Framework, you can lay out content in a variety of ways. You can create a three column page with a navigation bar on the top or side. Or you can create more of a dashboard appearance. You can also enable users to rearrange objects on the page. You must consider these options up front and decide the best way to lay out your content on the page.

See Also: Chapter 4, "Populating Pages" for more information about how you might go about building your pages.

■ **Site Administrator Considerations**

Note: It's critical that the WebCenter application administrator and the developers communicate when the WebCenter application is under construction. At design time, developers must make many choices that will determine what the administrator can do to the application at run time. For example, if the developers choose not to implement skins, then the administrator will have no control over the look and feel of the application. Hence, the administrator and the developers should ensure that they are in sync at design time.

– **Do administrators need to customize pages and portlets for users or user groups?** You may want to enable your WebCenter application administrator to customize pages and portlets to provide default views of the WebCenter application for users.

* **WebCenter application look and feel.** By implementing skins for your WebCenter application, you enable administrators to choose the look of the pages in the WebCenter application.
* **Customizable Pages.** The administrator can customize the layout of the page.

* **Portlets.** Portlets can optionally include a customization mode (Edit Defaults mode) that enables the administrator to customize the portlet. The administrator's customization is treated as the default view of the portlet for all other users.
* **Portlet Title.** If the administrator must customize the portlet's title, then the developer must not use the `text` attribute of the `<portlet>` tag.

See Also:

Chapter 9, "Defining and Applying Styles to Core Customizable Components" for information about how to implement skins.

Chapter 4, "Populating Pages" to learn more about customizable components.

Chapter 14, "Understanding Portlets" to learn more about portlets.

- **Developer Considerations**
  - **Do you have portlets that contain <form> tags?** The HTML generated by a JSF page includes a `<form>` tag. If you place a portlet on a JSF page that contains its own `<form>` tag as well, then the JSF page will break. To avoid this `<form>` tag conflict, an inline frame (IFRAME) surrounds portlets by default. You can control the use of IFRAMEs for portlets by setting the `renderPortletInIFrame` attribute of the adfp:portlet tag. The recommended setting for this attribute is `auto`, which places IFRAMEs around portlets in which a `<form>` tag is detected. If you choose `false`, then it becomes your responsibility to ensure that the portlet is free of `<form>` tags.
  - **Do developers need to integrate content from content repositories (for example, OracleAS Portal)?** If you have content located in content repositories that you want to expose in the WebCenter application, then you must use JCR adapters and data controls.

    See Also: Chapter 5, "Integrating Content" for more information about including content through JCR data controls and adapters.

  - **Will the visible content of the WebCenter application vary depending upon the identity of the user?** If you plan to have content in your WebCenter application that is not for everyone, then you must set up a security model with login and user roles that enable you to control access. Components, portlets, and pages can display or not display depending upon a user's identity. If the content is open to everyone (for example, a Human Resources WebCenter application with content for all employees), then you might not need to implement access control.

    See Also: Chapter 10, "Securing Your WebCenter Application" for more information about implementing a security model for your WebCenter application.

  - **Does your WebCenter application need to provide access to external applications?** In many cases, you might have external applications that you want to surface in your WebCenter application. For example, you might have e-mail, human resources, or financial applications that you would like users to be able to view from within the WebCenter application. If you must provide access to such external applications from your WebCenter application, then you must consider the effect on your security model.

See Also: Chapter 10, "Securing Your WebCenter Application" for more information about accessing external applications from your WebCenter application.

– **Does your WebCenter application need external application portlets from OracleAS Portal?** If you have legacy external application portlets that were built for OracleAS Portal, then you can reuse those portlets in a WebCenter application. This feature enables users to view these portlets from their WebCenter application rather than having to open each application separately or go back to OracleAS Portal. When building such portlets, though, you must remember that they typically must authenticate themselves to the external application before retrieving and displaying any data. Such authentication requires a credential vault, where the WebCenter application can store the credentials necessary for logging into the external application. Oracle WebCenter Framework provides choices in Oracle JDeveloper for incorporating portlets based on external applications.

See Also: Section 10.7, "Accessing External Applications Requiring Credentials" to learn more about external application portlets.

– **Will the WebCenter application be developed by a team of developers?** If you are doing team development on your WebCenter application, then you must take into account some additional design considerations. For example, when doing team development, you must be much more aware of which files various Oracle JDeveloper actions will touch.

See Also: Chapter 11, "Working Productively in Teams" to learn more about team development.

2.3 Using the Service Request Demo

The example used in this guide is based on the Oracle ADF application shown in the *Oracle Application Development Framework Developer's Guide*, and shows you how to add portal and WebCenter Services capabilities to an existing Oracle ADF application. This section covers the following two topics:

- Section 2.3.1, "Introduction to the Oracle ADF Service Request Demo"
- Section 2.3.2, "Setting Up the Oracle ADF Service Request Demo"

2.3.1 Introduction to the Oracle ADF Service Request Demo

The Service Request demo (SRDemo) application is a sample customer relationship management application that lets customers of a household appliance servicing company attempt to resolve service issues over the web. The application, which consists of sixteen Web pages, manages the customer-generated service request through the following flow:

1. A customer logs in and submits a service request.
2. A manager logs in and assigns the request to a technician.
3. The technician logs in and reviews their assigned requests, then provides a solution or solicits more information from the customer.
4. The customer returns to the site and checks their service request and either closes the request or provides further information.

5. While a request is open, managers can review an existing request for a technician and if necessary reassign it to another technician.

Additionally, technicians can identify products in their area of expertise. Managers then use this information to assign service requests.

After the user logs in, they see only the application functionality that fits their role as a customer, manager, or technician.

See the chapter titled *Introduction to the Oracle ADF Service Request Demo* in the *Oracle Application Development Framework Developer's Guide*,for more information about the demo.

About the Scenario

The SRDemo takes an existing Oracle ADF application for tracking customer service requests and adds portal capabilities to the application without altering the existing application. This service request application enables customers, technicians, and managers view information about service requests all from the same interface. The three roles are as follows:

- Customer (a customer of the My Acme Corporation)
- Technician (a technician who handles service requests for the My Acme Corporation)
- Manager (a manager who administers the Web site and runs a team of technicians for the My Acme Corporation)

Customer

The customer logs in to the application and can view current announcements, his existing service requests, and details about these requests. He can also view information about the products he has purchased, as well as a list of his current contracts with the company providing the services. He can also submit feedback on existing service requests.

Technician

The technician logs in to the application and views the service requests assigned to him, and can update existing service requests.

Manager

The manager logs in to the application and, at run time, can update the announcements that the customer views. He can also modify the page at run time using content in the content repository. For example, if a new service is now available to customers, then the manager can add information about this new service at run time. The manager can also review the feedback the customer has returned and add his own notes. He can also view site statistics on a dashboard page, which shows the current service request volume, the most active customers, and so on, as well as customize this dashboard page. The manager also has access to general site administration from a single page, where he can change the look and feel by switching skins and customize the login.

2.3.2 Setting Up the Oracle ADF Service Request Demo

To view the SRDemo and perform some of the example tasks in this guide, you must download and install the starter files. To do so, perform the following steps:

1. Download the `WebCenterSRDemo.zip` file, located on this page:

   `http://www.oracle.com/technology/products/webcenter/documentation.h`

tml

2. Extract the ZIP file to your `c:\` drive, and follow the instructions in the `Install.html` file located at the top level of the directory.

Part II

Building a WebCenter Application

Part II contains the following chapters:

3

Preparing Your Development Environment

You can simplify the creation of WebCenter applications and projects by following a few preparatory procedures. Oracle WebCenter Framework provides a template that prepopulates the application with projects optimally scaled for the creation of a WebCenter application. Additionally, with a few simple steps, you can front-load portlets useful in both development and run time environments.

It is not required that you use the WebCenter application template. If you prefer, you can create your own WebCenter application by manually scoping the application technologies and creating the relevant projects.

This chapter steps through the process of applying a WebCenter application template. It provides additional information about creating the framework for a WebCenter application without using the template. It includes information about obtaining and installing a preconfigured Oracle Containers for J2EE (OC4J) and accessing its portlets.

This chapter contains the following sections:

3.1 Creating a WebCenter Application

When you create a WebCenter application and its projects, you may want to specify project technology scopes that support the creation of portlets and Java Content Repository (JCR) data controls and enable the WebCenter application to consume them. You can automatically scope the application and its projects by selecting a template, or you can manually scope the application and its projects by creating them yourself.

> **Note:** Technology scopes control what options display by default in the Oracle JDeveloper User Interface (UI). In a feature-rich IDE like Oracle JDeveloper, this is useful in eliminating the task of choosing from among many different options when developing a particular type of application.

This section describes both approaches. It includes the following subsections:

- Section 3.1.2, "What Happens When You Use the WebCenter Application Template"
- Section 3.1.3, "Manually Creating a WebCenter Application and Projects"
- Section 3.1.4, "Importing a WAR File to Create a WebCenter Application Project"

3.1.1 Creating a WebCenter Application Using a Template

The easiest way to ensure that you properly define an application and its projects with the appropriate technology scope is to apply an application template. An application template provides a way to partition the application into projects that reflect a logical separation of the overall work. The WebCenter Suite provides a template optimally configured for the creation and consumption of portlets.

The WebCenter Application template consists of a project for the data model (Model); a project for creating portlets (Portlets); and a project for consuming portlets, components, and data controls (ViewController). The WebCenter Application template folds all of these projects into one application for simplicity, but you are welcome to arrange your applications and projects the way that fits best.

For example, if you are both the portlet developer and the application developer, then it is more likely that you will use the WebCenter Application template as is, out of the box. If you are either the portlet developer or the application developer—but not both—then it is more likely that each role will have a separate application—one for the WebCenter application and one for the portlets the application consumes—unless your organization uses some kind of source control tool for its development projects.

To create an application using the WebCenter Application template, perform the following steps:

1. From the Applications Navigator, right-click the Applications node and choose **New Application** from the context menu.
2. In the Create Application dialog box, enter a name for the application in the **Application Name** field.
3. In the **Directory Name** field, enter a path to the directory where the application should be stored, or accept the default path.

   For example:

   ```
   C:\jdev\mywork\myapplication
   ```

   Optionally, click the **Browse** button to navigate to the desired directory.
4. If required, in the **Application Package Prefix** field, enter a prefix to use for packages created within this application.
5. From the **Application Template** list, select **WebCenter Application [Portlet, Content Repository, JSF]**.
6. Click **OK**.

The template prepopulates the application with three projects:

- **Model**, scoped for data modeling and content sourcing
- **Portlets**, scoped for portlet creation
- **View Controller**, scoped for creation of WebCenter application pages and page elements and for registering portlet producers

These projects are scoped to provide options throughout Oracle JDeveloper that are appropriate for WebCenter application development. For more information, see Section 3.1.2, "What Happens When You Use the WebCenter Application Template".

3.1.2 What Happens When You Use the WebCenter Application Template

Using the WebCenter application template generates default projects with unique technology scopes, libraries, and default files. It limits the types of options that are exposed in the Oracle JDeveloper user interface to those appropriate to the project type.

This section provides an overview of what occurs when you use the **WebCenter Application [Portlet, Content Repository, JSF]** template. It includes the following subsections:

- Section 3.1.2.1, "Template Projects, Technology Scopes, and Libraries"
- Section 3.1.2.2, "WebCenter Application Template Default Files and Folders"

3.1.2.1 Template Projects, Technology Scopes, and Libraries

By default, the template **WebCenter Application [Portlet, Content Repository, JSF]** names its three projects *Model*, *Portlets*, *ViewController*. If you prefer, you can rename the projects before you create the application, through the **Manage Templates** command from the **Tools** menu or after you create the application, through the **Rename** command from the **File** menu.

When you use the WebCenter Application template, Oracle JDeveloper does the following for you:

- Creates *Model*, *Portlets*, and *ViewController* projects, each with their own relevant technology scopes and libraries (Table 3–1).

Table 3–1 WebCenter Application Template Projects

Project	Purpose	Technology Scopes	Libraries
Model	Use this project to define data controls that pull in content from selected data repositories.	Java JavaBeans Content Repository	None
Portlets	Use this project to build portlets.	Java JSP and Servlets Portlet	None
ViewController	Use this project to build WebCenter application pages that consume portlets and contain your other WebCenter application components. You can also use the ViewController project as the starting point for registering portlet producers. Note, however, that portlet producers are created at the application level irrespective of the starting point from which they were created. For more information, see Section 4.3.1, "Registering Portlet Producers".	HTML Java JSF JSP and Servlets WebCenter View	**JSP Tag Libraries:** JSF Core JSF HTML **Other Libraries:** JSF Commons Beansutils Commons Digester Commons Logging Commons Collections JSTL

When you work in each project, the New Gallery is filtered to show only relevant technologies under each of its nodes. For example, select **Portlets** in the New Gallery while the *ViewController* project is selected, you'll see options for registering portlet producers. Select **Portlets** in the New Gallery when the *Portlets* project is selected, you'll see options for creating portlets.

- Creates a starter application deployment descriptor `web.xml` file with default settings. You'll find the `web.xml` file in the Applications Navigator under the ViewController project's `WEB-INF` folder. For more information about `web.xml`, see Section 3.1.2.2.2, "The WebCenter Application Template Default web.xml File".
- Creates an empty `faces-config.xml` file in the ViewController project's `WEB-INF` folder. The `faces-config.xml` file is a JSF configuration file where JSF application resources are registered, such as custom validators and managed beans, and all page-to-page navigation rules are defined. For more information about `faces-config.xml`, see Section 3.1.2.2.3, "The WebCenter Application Template Default faces-config.xml File".
- Adds `jsf-impl.jar` in the ViewController project's `WEB-INF\lib` folder. This JAR file is the JSF reference implementation that Oracle JDeveloper includes by default.

3.1.2.2 WebCenter Application Template Default Files and Folders

The WebCenter Application template instantly creates a folder hierarchy with an accompanying set of default files. This section provides detailed information about these in the following subsections:

- Section 3.1.2.2.1, "The WebCenter Application Default Folder Hierarchy"
- Section 3.1.2.2.2, "The WebCenter Application Template Default web.xml File"
- Section 3.1.2.2.3, "The WebCenter Application Template Default faces-config.xml File"

3.1.2.2.1 The WebCenter Application Default Folder Hierarchy Because there are several possible views of the Oracle JDeveloper Applications Navigator, the hierarchy of folders in a WebCenter application may differ when comparing folders in the Applications Navigator with the same folders in your file system. For example, Figure 3–1 illustrates one view of a newly created application in the Oracle JDeveloper Applications Navigator. Figure 3–2 illustrates the folder hierarchy of the same application in a Windows file system.

Figure 3–1 Application from WebCenter Application Template in Oracle JDeveloper

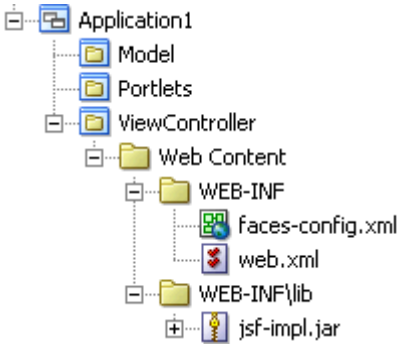
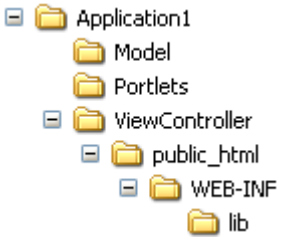


Figure 3–2 Application from WebCenter Application Template in a Windows File System



3.1.2.2.2 The WebCenter Application Template Default web.xml File Part of WebCenter application configuration is determined by the content of its J2EE application deployment descriptor file: `web.xml`. The `web.xml` file defines everything about the application that a server must know, with the exception of the context root path, which is assigned when the application is deployed.

Typical run time settings include initialization parameters, custom tag library location, and security settings.

Example 3–1 illustrates the default `web.xml` file provided through the WebCenter Application template.

Example 3–1 Default web.xml File Provided Through the WebCenter Application Template

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee">
   <description>Empty web.xml file for Web Application</description>
   <servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>/faces/*</url-pattern>
   </servlet-mapping>
   <session-config>
      <session-timeout>35</session-timeout>
   </session-config>
   <mime-mapping>
      <extension>html</extension>
      <mime-type>text/html</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>txt</extension>
      <mime-type>text/plain</mime-type>
   </mime-mapping>
</web-app>
```

The JSF servlet and servlet mapping configuration settings are automatically added to the default `web.xml` file when a WebCenter application is first created.

The `<servlet></servlet>` tags provide information about the JSF servlet, `javax.faces.webapp.FacesServlet`. This servlet manages the request processing life cycle for Web applications that use JSF to construct the user interface.

The configuration setting maps the JSF servlet to a symbolic name—in Example 3–1, *Faces Servlet*.

The `<servlet-mapping></servlet-mapping>` tags map the URL pattern to the JSF servlet's symbolic name. You can use either a path prefix or an extension suffix pattern.

By default, Oracle JDeveloper uses the path prefix `/faces/*`. This means that when a URL, for example, `http://localhost:8080/SRDemo/faces/index.jsp`, is issued, the URL activates the JSF servlet, which strips off the `faces` prefix and loads the file `/SRDemo/index.jsp`.

Note: If you prefer to use the extension `jsf` for Web pages instead of `jsp` or `jspx`, then you must add a servlet mapping in `web.xml` that invokes the JSP servlet for files with the extension `jsf`. Then you must set the `javax.faces.DEFAULT_SUFFIX` context parameter to `jsf`, for example:

```
<context-param>
    <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
    <param-value>.jsf</param-value>
</context-param>
```

To edit `web.xml` in Oracle JDeveloper, right-click `web.xml` in the Applications Navigator and choose **Properties** from the context menu. This opens `web.xml` in the Web Application Deployment Descriptor editor. If you're familiar with configuration element names, then you can also use the XML editor to modify `web.xml`.

Note: If you use the Oracle ADF data controls to build data-bound Web pages, then Oracle JDeveloper adds the ADF binding filter and a servlet context parameter for the application binding container in `web.xml`.

For reference information about the configuration elements you can use in the `web.xml` file, see Appendix A in the *Oracle Application Development Framework Developer's Guide.*

3.1.2.2.3 The WebCenter Application Template Default faces-config.xml File

Use the JSF configuration file to register a WebCenter application's resources, such as custom validators and managed beans, and to define all page-to-page navigation rules. While an application can have any JSF configuration file name, typically the file name is `faces-config.xml`.

Small applications usually have one `faces-config.xml` file. Larger applications can have multiple `faces-config.xml` files. For example:

- Create individual JSF configuration files for separate areas of your application.
- Create separate JSF configuration files for each library containing custom components or renderers.

For additional information about creating multiple `faces-config.xml` files, see *Oracle Application Development Framework Developer's Guide.*

Example 3–2 illustrates the default `faces-config.xml` file provided through the WebCenter Application template.

Example 3–2 Default faces-config.xml File Provided Through the WebCenter Application Template

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE faces-config PUBLIC
    "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
    "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
<faces-config xmlns="http://java.sun.com/JSF/Configuration">

</faces-config>
```

To edit the `faces-config.xml` file, double-click it in the Applications Navigator. By default, the file is opened in the Editor window in *Diagram* mode, as indicated by the active **Diagram** tab at the bottom of the Editor window. When creating or modifying JSF navigation rules, Diagram mode enables you to visually create and manage page flows. For more information, see *Oracle Application Development Framework Developer's Guide.*

To create or modify configuration elements other than navigation rules, use the Editor's Overview mode. Enter this mode by clicking the **Overview** tab at the bottom of the Editor window.

JSF enables multiple `<application>` elements in a single `faces-config.xml` file. When you use the JSF Configuration Editor, you can edit only the first instance. For any other `<application>` elements, you must edit the file directly using the XML editor. To use the XML editor, open the `faces-config.xml` file and go to the **Source** tab in the Editor window.

For reference information about the configuration elements you can use in the `faces-config.xml` file, see Appendix A in the *Oracle Application Development Framework Developer's Guide.*

3.1.3 Manually Creating a WebCenter Application and Projects

If you choose to build a WebCenter application without using a template, then you may find it useful to specify the application projects' technology scopes and libraries. The technology scopes limit UI options to those appropriate to the type of application project you are building. The libraries provide the components and elements useful in constructing application content.

The easiest way to understand what to specify for a WebCenter application is to review the out-of-the-box template's characteristics. You can then mimic these in your own WebCenter applications or templates.

This section steps you through the process of manually creating a WebCenter application and projects like those provided with the template. It contains the following subsections:

- Section 3.1.3.1, "Manually Creating a WebCenter Application"
- Section 3.1.3.2, "Manually Creating WebCenter Application Projects"

3.1.3.1 Manually Creating a WebCenter Application

Very little is required to manually create a WebCenter application. The technology scoping occurred at the project level. This means that the primary difference between manually creating a WebCenter application and creating one using a template is that you do not choose a template when you create the application.

To manually create a WebCenter application, perform the following steps:

1. From the Applications Navigator, right-click the Applications node and choose **New Application** from the context menu.
2. In the Create Application dialog box, enter a name for the application in the **Application Name** field.
3. In the **Directory Name** field, enter a path to the directory where the application should be stored, or accept the default path.

   For example:

   ```
   C:\jdev\mywork\myapplication
   ```

   Optionally, click the **Browse** button to navigate to the desired directory.
4. If required, in the **Application Package Prefix** field, enter a prefix to use for packages created within this application.
5. From the **Application Template** list, select **No Template [All Technologies]**.
6. Click **OK**.
7. Click **Cancel** in the Create Project dialog box.

3.1.3.2 Manually Creating WebCenter Application Projects

This section steps you through manual creation of three projects—each optimized for a particular use (see Section 3.1.1). It contains the following subsections:

- Section 3.1.3.2.1, "Creating a Project Optimized for Defining the Application Data Model"
- Section 3.1.3.2.2, "Creating a Project Optimized for Building Portlets"
- Section 3.1.3.2.3, "Creating a View Project for Consuming Content"

3.1.3.2.1 Creating a Project Optimized for Defining the Application Data Model

Use data controls to bind data from a content repository to a project's user interface components. To define data controls for a project, the project must be scoped to enable for their creation and use.

To create a project optimized for defining an application data model, perform the following steps:

1. In the Applications Navigator, right-click the WebCenter application and select **New Project** from the context menu.
2. In the New Gallery, expand the **General** node under **Categories** and select **Projects**.
3. Under **Items**, select **Empty Project**.
4. Click **OK**.
5. In the Create Project dialog box, enter a name for the project.
6. In the **Directory Name** field, specify the directory path for storing the project, or accept the default path.

   Optionally, click the **Browse** button to navigate to the desired directory.
7. Click **OK** to create the project.
8. In the Applications Navigator, right-click the project and select **Project Properties** from the context menu.
9. In the Project Properties dialog box, select the **Technology Scope** node.

10. If you want to include repository content in your application, then in the **Available Technologies** column select **Content Repository** and click the **Add (ALT-D)** icon to move it to the **Selected Technologies** column.

    You can also select other technologies, such as Java and JavaBeans, at this step.

11. Click **OK**.

> **Note:** See Table 3–1 for a summary of the technologies and libraries included in the Model project.

3.1.3.2.2 Creating a Project Optimized for Building Portlets If you want to create portlets as part of your application development effort, then you'll want to create a project that is scoped for portlets. Use portlets to provide controlled display of and access to local and remote content sources. Once scoped, the options available in Oracle JDeveloper are tailored to creating portlets. For example, the Web Tier node is displayed in the New Gallery, and portlet creation is enableed, but portlet producer registration is not.

To create a project optimized for building portlets, perform the following steps:

1. In the Applications Navigator, right-click the WebCenter application and select **New Project** from the context menu.
2. In the New Gallery, expand the **General** node under **Categories** and select **Projects**.
3. Under **Items**, select **Empty Project**.
4. Click **OK**.
5. In the Create Project dialog box, enter a name for the project.
6. In the **Directory Name** field, specify the directory path for storing the project, or accept the default path.

   Optionally, click the **Browse** button to navigate to the desired directory.
7. Click **OK** to create the project.
8. In the Applications Navigator, right-click the project and select **Project Properties** from the context menu.
9. In the Project Properties dialog box, select the **Technology Scope** node.
10. In the **Available Technologies** column, select **Portlet** and click the **Add (ALT-D)** icon to move it to the **Selected Technologies** column.
11. Click **OK**.

> **Note:** See Table 3–1 for a summary of the technologies and libraries included in the Portlets project.

3.1.3.2.3 Creating a View Project for Consuming Content A project optimized for consuming content includes a means of creating application pages, adding user interface components and binding them to data controls, and registering portlet producers. Once the consuming project is scoped, the options available in Oracle JDeveloper are tailored to creating application pages, consuming content, binding data controls to user interface components, and registering portlet producers. For example, the Web Tier node is displayed in the New Gallery, and portlet producer registration is enabled, but portlet creation is not.

Note: Portlet producers are created at the application level irrespective of the starting point from which they were created. For more information, see Section 4.3.1, "Registering Portlet Producers".

To create a project optimized for consuming portlets, perform the following steps:

1. In the Applications Navigator, right-click the WebCenter application and select **New Project** from the context menu.
2. In the New Gallery, expand the **General** node under **Categories** and select **Projects**.
3. Under **Items**, select **Empty Project**.
4. Click **OK**.
5. In the Create Project dialog box, enter a name for the project.
6. In the **Directory Name** field, specify the directory path for storing the project, or accept the default path.

   Optionally, click the **Browse** button to navigate to the desired directory.
7. Click **OK** to create the project.
8. In the Applications Navigator, right-click the project and select **Project Properties** from the context menu.
9. In the Project Properties dialog box, select the **Technology Scope** node.
10. If you will be using portlets in your pages, then in the **Available Technologies** column select **WebCenter View** and click the **Add (ALT-D)** icon to move it to the **Selected Technologies** column.
11. Click **OK**.

Note: See Table 3–1 for a summary of the technologies and libraries included in the ViewController project.

3.1.4 Importing a WAR File to Create a WebCenter Application Project

This procedure is included in the event that you have an existing WAR file that you want to import into your WebCenter application through Oracle JDeveloper.

To import a WAR file into Oracle JDeveloper, perform the following steps:

1. Right-click your application in the Applications Navigator, and choose **New Project** from the context menu.
2. In the New Gallery, expand the **General** node under **Categories** and select **Projects**.
3. Under **Items**, double-click **Project from War File**.
4. Follow the wizard instructions to complete creating the project.
5. Right-click the newly created project in the Applications Navigator, and select **Project Properties** from the context menu.
6. In the Project Properties dialog box, select **Libraries**, and click the **Add Library** button.
7. In the Add Library dialog box, select **Portlet Development** and click **OK**.

This adds the Portlet Development library to the project. This library is required for the newly created project to compile successfully.

8. Click **OK** to close the Project Properties dialog box.

3.2 Using the Preconfigured OC4J

Installation of Oracle WebCenter Framework includes a preconfigured OC4J application server. This OC4J contains preconfigured portlet producers and several useful prebuilt portlets.

This section discusses the preconfigured OC4J, including how to start and stop it, and describes some of the preconfigured portlet producers and prebuilt portlets it provides. It contains the following subsections:

- Section 3.2.1, "What You Should Know About the Preconfigured OC4J"
- Section 3.2.2, "Starting and Stopping the Preconfigured OC4J"
- Section 3.2.3, "The Preconfigured OC4J Readme File"
- Section 3.2.4, "What You Should Know About Preconfigured Portlet Producers"

3.2.1 What You Should Know About the Preconfigured OC4J

OC4J provides a complete Java 2 Enterprise Edition (J2EE) 1.4-compliant environment. OC4J is written entirely in Java and executes on the Java Virtual Machine (JVM) of the standard Java Development Kit (JDK). You can run OC4J on the standard JDK provided with your operating system or the one provided with Oracle JDeveloper.

You can use the preconfigured OC4J as a platform for pretesting WebCenter application deployments on your local computer by establishing an application server connection to it from Oracle JDeveloper.

Note: For more information, see Section 12.2.6.2.1, "Defining Standalone OC4J Connection Details".

The preconfigured OC4J differs from the embedded OC4J included with Oracle JDeveloper. The embedded OC4J is the server that is used when you run application pages within Oracle JDeveloper. It is literally *embedded* in Oracle JDeveloper's IDE.

The preconfigured OC4J is a separate tool that you can use, for example, to create test deployments. You will find more information about the preconfigured OC4J in this section. For more information about the embedded OC4J, see Section 12.2.6.1, "Deploying to Embedded OC4J".

For detailed information about OC4J, see *Oracle Containers for J2EE Configuration and Administration Guide*.

3.2.2 Starting and Stopping the Preconfigured OC4J

Icons for starting and stopping the preconfigured OC4J display in the Oracle JDeveloper toolbar.

- To start the preconfigured OC4J, click the **Start WebCenter Preconfigured OC4J** icon (Figure 3–3).

Figure 3–3 Start WebCenter Preconfigured OC4J Icon

- To stop the preconfigured OC4J, click the **Stop WebCenter Preconfigured OC4J** icon (Figure 3–4).

Figure 3–4 Stop WebCenter Preconfigured OC4J Icon

The first time you start the preconfigured OC4J, a dialog box tells you that the WebCenter preconfigured OC4J is not installed in the current user directory. This dialog box offers you the option of installing it in the current user directory. Select **Yes**.

Alternatively, you can select the **Start WebCenter Preconfigured OC4J** and **Stop WebCenter Preconfigured OC4J** options from the Tools menu to start and stop preconfigured OC4J.

There are a couple of ways to determine if the preconfigured OC4J is running:

- From the **View** menu, select **Log**, and look for the following entry:

  ```
  Oracle Containers for J2EE 10g (10.1.3.1.0)  initialized
  ```

- Access the preconfigured OC4J test page from a browser:

  ```
  http://localhost:6688
  ```

> **Note:** Sometimes the OC4J is not accessible (for example, if a user tries to restart the OC4J too quickly, before it has successfully shut down). In this case, you may have to manually shut down or stop the java process.

3.2.3 The Preconfigured OC4J Readme File

When you first start the preconfigured OC4J, its `readme.html` file displays in the Oracle JDeveloper editor. The `readme.html` file contains valuable information about how to use the preconfigured OC4J and what it provides. Under the heading "Index Page," you will find the `index page` link. The index page contains a list of links to preconfigured portlet producers. You can test each of the portlet producers by clicking the links on the index page.

Once you exit the preconfigured OC4J readme file, you can return to it by selecting **WebCenter Preconfigured OC4J Readme** from the Oracle JDeveloper **Help** menu.

3.2.4 What You Should Know About Preconfigured Portlet Producers

The preconfigured OC4J provides a variety of ready-to-use portlets that you can add to your application pages. Simply register the producers contained in the preconfigured OC4J with your WebCenter application, and then select the producers' portlets from Oracle JDeveloper's Component Palette.

This section provides a brief description of the preconfigured OC4J producers and some of the portlets they provide. It contains the following subsections:

- Section 3.2.4.1, "Preconfigured OC4J Portlet Producers and Portlets"

- Section 3.2.4.2, "The PDK-Java Sample Portlet Producer and Portlets"
- Section 3.2.4.3, "The WSRP Sample Portlet Producers and Portlets"

For information about registering portlet producers, see Section 4.3.1, "Registering Portlet Producers". For information about adding portlets to pages, see Section 4.3.2, "Adding Portlets to a Page".

3.2.4.1 Preconfigured OC4J Portlet Producers and Portlets

PortalTools provides access to the *design-time at run time* portlets OmniPortlet and the Web Clipping portlet. Design-time at run time means that users define portlet content after the portlet is placed on an application page and the page is run. This concept is explained more fully in Section 15.7, "Portlet Creation Style".

To access OmniPortlet and Web Clipping portlet producers, perform the following steps:

1. Start the preconfigured OC4J (see Section 3.2.2).
2. From the **Help** menu, select **WebCenter Preconfigured OC4J Readme**.
3. In the Readme file, scroll down to the section headed **Index Page** and click the **index page** link in the second paragraph.

   > **Note:** Once the preconfigured OC4J is running, another way to access the index page is to point your browser to `http://localhost:6688`.

4. Go to the heading **Preconfigured Portlet Producers**, and click the **PortalTools Welcome Page** link under **PortalTools Portlet Producers**.

   This opens the PortalTools Welcome page.
5. On the PortalTools Welcome page, copy the URL of the Web Clipping Producer link, the OmniPortlet Producer link, or the Sample Portlet Producer link, and use it as the producer URL in the Oracle PDK-Java Portlet Producer Registration Wizard.

   > **Note:** For information about registering a portlet producer, see Section 4.3.1, "Registering Portlet Producers".

Once you have registered one of these producers, its portlets become available on Oracle JDeveloper's Component Palette. On the Component Palette, select a producer name to list its portlets, then drag a portlet from the palette onto a WebCenter application page. (For information about adding portlets to pages, see Section 4.3.2, "Adding Portlets to a Page".)

The PortalTools Welcome page contains producer URLs for three producers:

- The **Web Clipping producer** provides the Web Clipping portlet, which is a browser-based declarative tool that enables dynamic reuse of content from another Web source. When the source changes, the content in the Web Clipping portlet also changes. With the Web Clipping portlet, you use a Web browser to navigate to the Web page that contains the desired content. Using Web Clipping Studio, which is accessed through the portlet, drill down through a visual rendering of the target page to choose the desired content. For detailed information about the Web Clipping portlet, see Chapter 17, "Creating Content-Based Portlets with Web Clipping".

- The **OmniPortlet producer** provides OmniPortlet, which is a declarative portlet-building tool that enables you to build portlets against a variety of data sources, including XML files, character-separated value files (for example, spreadsheets), Web Services, databases, Web pages, and SAP data sources. OmniPortlet users can also choose a prebuilt layout for the data. Prebuilt layouts include tabular, news, bullet, form, chart, or HTML. For information about OmniPortlet, see Chapter 16, "Creating Portlets with OmniPortlet".
- The **Sample Portlet Producer** is useful for illustrating the potential of an OmniPortlet. The sample producer is for demonstration and should not be used to create real-use portlet instances.

3.2.4.2 The PDK-Java Sample Portlet Producer and Portlets

To access PDK-Java sample portlet producers, perform the following steps:

1. Start the preconfigured OC4J (see Section 3.2.2).
2. From the **Help** menu, select **WebCenter Preconfigured OC4J Readme**.
3. In the Readme file, scroll down to the section headed **Index Page** and click the **index page** link in the second paragraph.

   Note: Once the preconfigured OC4J is running, another way to access the index page is to point your browser to `http://localhost:6688`.

4. Go to the heading **Preconfigured Portlet Producers**, and copy the link location for the **PDK-Java Sample Producer** or **PDK-Java Struts Sample Producer** link under **PDK-Java Portlet producers**.

   Use the copied link as the producer URL in the Oracle PDK-Java Portlet Producer Registration Wizard.

   Note: For information about registering a portlet producer, see Section 4.3.1, "Registering Portlet Producers".

Once you register either of these producers, its portlets display on Oracle JDeveloper's Component Palette. From here, you can drag and drop a variety of sample portlets onto your WebCenter application pages. Use the PDK-Java sample portlets to familiarize yourself with the types of functionality that are available through PDK-Java portlets.

3.2.4.3 The WSRP Sample Portlet Producers and Portlets

To access WSRP sample portlet producers, perform the following steps:

1. Start the preconfigured OC4J (see Section 3.2.2).
2. From the **Help** menu, select **WebCenter Preconfigured OC4J Readme**.
3. In the Readme file, scroll down to the section headed **Index Page**, and click the **index page** link in the second paragraph.

   Note: Once the preconfigured OC4J is running, another way to access the index page is to point your browser to `http://localhost:6688`.

4. Go to the heading **Preconfigured Portlet Producers**, and click the link for the **Rich Text Portlet Producer** or **Sample Portlets** under **WSRP Portlet Producers**.

   Both links open different WSRP Producer Test Pages—one for different WSRP producer versions of the Rich Text portlet, the other for different WSRP producer versions of sample portlets.

5. Copy the Web Services Description Language (WSDL) URL for one of the WSRP Producers—either WSRP v1 WSDL or WSRP v2 WSDL.

   Use the copied link as the producer URL in the WSRP Producer Registration Wizard.

> **Note:** For information about registering a portlet producer, see Section 4.3.1, "Registering Portlet Producers".

Once you register a producer, its portlets display on Oracle JDeveloper's Component Palette. The WSRP WSDL URLs on the Rich Text Portlet Producer page include the Rich Text portlet, which offers browser-based rich text editing at run time. For information about the Rich Text portlet, see Section 14.3.1, "Rich Text Portlet".

3.3 Enabling Oracle SOA Suite or a Standalone OC4J for WebCenter Applications

If you have Oracle SOA Suite 10.1.3.1.0 or a standalone OC4J 10.1.3.1.0, then you can update it to create, deploy, and run Oracle WebCenter Suite 10.1.3.2.0 applications. Patches to enable you to run WebCenter applications will be released in the future. Watch for more information about Meta*Link* and the Oracle Technology Network:

`http://www.oracle.com/technology/products/webcenter/index.html`

4

Populating Pages

This chapter explains how to add components to the pages of your WebCenter application. Note that this chapter does not cover Oracle JDeveloper or Oracle ADF page creation basics. It covers only those aspects of page creation that are specific to WebCenter application pages. Therefore, you must familiarize yourself with the information covered in*Oracle Application Development Framework Developer's Guide* before reading this chapter.

4.1 Introduction to Page Content

Oracle WebCenter Framework provides functionality for placing a wide variety of content on an application page. Content types include portlets, customizable components, and content integration tools. Optionally, you can enable customization of these components to make the application more flexible. Oracle WebCenter Framework also provides for the linking of these components so that they operate cohesively, making the application easier to understand and use.

This section describes the Oracle WebCenter Framework components you can add to your WebCenter application pages. It includes the following subsections:

> **Note:** You are not limited to adding just these components to your WebCenter applications. You can add other components as well. This list encompasses only Oracle WebCenter Framework-specific components. For information about other useful components you might want to use, see *Oracle Application Development Framework Developer's Guide*.

4.1.1 Portlet Overview

Oracle WebCenter Framework enables you to consume a portlet by registering its producer with an application. After you register the producer, its portlets appear on the Oracle JDeveloper Component Palette under the registered producer's name. You can drag portlets from the Component Palette and drop them onto a page as you would any other component.

Your application can consume portlets that you build as well as portlets that you receive from a third party, such as a packaged-application vendor.

Many options are associated with portlet consumption: you can choose to place portlets straight onto a page or nest them in a customizable component; you can adjust many attributes of the portlet tag (`adfp:portlet`); you can wire portlets together.

This chapter provides information about consuming portlets and the options that accompany their consumption. It includes the following sections:

- Section 4.3, "Consuming Portlets"
- Section 4.4, "Using Customizable Components"
- Section 4.5, "Contextually Linking Components"

Note: To learn more about creating portlets, see the following chapters:

- Chapter 14, "Understanding Portlets"
- Chapter 15, "Portlet Technologies Matrix"
- Chapter 16, "Creating Portlets with OmniPortlet"
- Chapter 17, "Creating Content-Based Portlets with Web Clipping"
- Chapter 18, "Creating Java Portlets"

4.1.2 Customizable Components Overview

Customizable components provide the ability to control the design time at run time behavior of the application. These components enable users to manipulate their view of the content according to their requirements. For example, one user may choose to hide a certain piece of content altogether, while another moves it up to the top of the page. By adding customizable components to a page, you make the page customizable for users.

Oracle WebCenter Framework provides two core customizable components:

- `ShowDetailFrame`
- `PanelCustomizable`

A `ShowDetailFrame` surrounds one Oracle ADF component and can provide a border and a chrome bar with menu actions, for example, to minimize the content. A `ShowDetailFrame` component enables you to do the following:

- Maximize or restore the display of the child component
- Provide a chrome or border for the component
- Provide an actions menu to perform specific actions on the child component

A `PanelCustomizable` component offers horizontal and vertical layout capabilities to a group of Oracle ADF components, including customizable components. Similar to

a ShowDetailFrame component, a PanelCustomizable component can also display a chrome to provide menu actions. A PanelCustomizable component enables you to do the following:

- Maximize or restore the display of child components
- Show or hide child components
- Move or rearrange child components within the PanelCustomizable component

To leverage all features of customizable components, you can add ShowDetailFrame and PanelCustomizable components in the following manner:

- Use a ShowDetailFrame component to render a border or chrome around its child component and thereby provide User Interface (UI) controls to customize the display of the child component. For example, options to move, minimize, or maximize the display of content.

  The placement hierarchy is as follows:

```
ShowDetailFrame
    ADF Faces component
```

- Wrap a ShowDetailFrame component within a PanelCustomizable component to provide the ability to show or hide child components.

  The placement hierarchy is as follows:

```
PanelCustomizable
   ShowDetailFrame Child1
          ADF Faces component
   ShowDetailFrame Child2
          ADF Faces component
   Portlet
   Portlet
```

> **Note:** Portlets provide the portlet chrome with display options similar to ShowDetailFrame components. Therefore, it is not necessary to include portlets within ShowDetailFrame components.

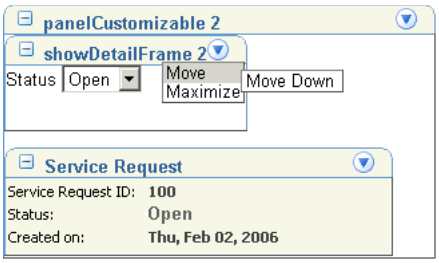
4.1.2.1 Defining Appearance and Customization Characteristics

Use the ShowDetailFrame tag to include a ShowDetailFrame component on the page. A list of options, available as a list on the ShowDetailFrame header as shown in Figure 4–1, enables users to control the display of a child component.

Figure 4–1 shows the following component placement hierarchy:

```
PanelCustomizable
   ShowDetailFrame Child1
          ADF Faces component
   Portlet
```

Figure 4–1 Actions Available with a ShowDetailFrame Component



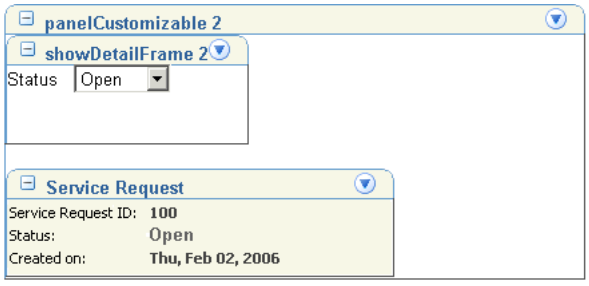
You can maximize and minimize content by using the options available under the Actions menu on the header. You can add your own UI controls to further customize the display by using facets of the `cust:showDetailFrame` tag.

4.1.2.2 Nesting Customizable Components

If you place `ShowDetailFrame` components inside a `PanelCustomizable` component, then in addition to maximizing and minimizing the display, you can also move the child components up and down or to the right and left.

Figure 4–2 shows the nesting of a `PanelCustomizable` and a `ShowDetailFrame` component with child components.

Figure 4–2 Nesting of Customizable Components



Placing Content and Portlets Inside of Customizable Components

You can include Oracle ADF Faces components as child components in your `PanelCustomizable` and `ShowDetailFrame` components. However, it is sufficient if you include portlets within `PanelCustomizable` components *only*. It is not necessary to include portlets in `ShowDetailFrame` components as portlets provide the portlet chrome with display options similar to `ShowDetailFrame` components. You do not get any additional benefits by including portlets in `ShowDetailFrame` components.

The procedure for adding child components in `PanelCustomizable` and `ShowDetailFrame` components is similar to adding Oracle ADF Faces components on a page. Before adding a child component, ensure that the `PanelCustomizable` or `ShowDetailFrame` component is selected in the Structure pane in Oracle JDeveloper. See Section 4.3.2, "Adding Portlets to a Page" for details.

4.1.3 JCR Data Control Overview

Java Content Repository (JCR) data controls expose file and folder content on a page. In this way, they enable you to integrate content from a content repository into your WebCenter application. You create a data control that selects content from a particular content repository (for example, Oracle Content DB or a file system). Once the data

control is created, you can drop it onto a JSP document in the desired format, for example, a table. For more information about JCR data controls and integrating content, see Chapter 5, "Integrating Content".

4.2 Building WebCenter Application-Enabled Pages in Oracle JDeveloper with Oracle ADF

For detailed information about creating pages in Oracle JDeveloper with Oracle ADF, see *Oracle Application Development Framework Developer's Guide*. There you will find all of the information that you need on Oracle ADF Faces and building basic and complex pages. The section that follows describes some of the special requirements of WebCenter application pages.

Requirements for Pages

The following list provides rules and guidance to apply when you are creating your WebCenter application pages in Oracle JDeveloper with Oracle ADF:

- Create WebCenter application pages as JSP *documents* (`jspx`) rather than JSP *pages* (`jsp`). In order for page customizations to be stored, the page must be represented in XML (`jspx`). Hence, by choosing to create a JSP document, you ensure that customizations are always possible for the WebCenter application page.

  If you are not planning to enable customization, then you could choose to create a JSP page, but this choice might cause problems if you later decide that you must enable customization.

- Make sure that you have the following libraries in the Tag Libraries page of the Create JSF JSP Wizard.

  – **ADF Portlet Components**

    This library is required when you plan to place portlets on application pages.

  – **Customizable Components Core**

    This library is required when you plan to enable page customizations, such as the ability to move, hide, and show portlets, and to enable these customizations on any Oracle ADF component.

4.3 Consuming Portlets

This section steps you through the processes of making portlets available to an application (that is, registering portlet producers), adding portlets to a page, and removing them from a page. It contains the following subsections:

- Section 4.3.1, "Registering Portlet Producers"
- Section 4.3.2, "Adding Portlets to a Page"
- Section 4.3.3, "Setting Attribute Values for the adfp:portlet Tag"
- Section 4.3.4, "Copying Portlets"
- Section 4.3.5, "Deleting Portlets from Application Pages"

For information about obtaining prebuilt portlets through Oracle, see Section 3.2, "Using the Preconfigured OC4J". For information about using Oracle JDeveloper's portlet creation wizards, see Chapter 18, "Creating Java Portlets". For more information about portlets, see Chapter 14, "Understanding Portlets".

4.3.1 Registering Portlet Producers

Before you can add a portlet to a WebCenter application, you must register the portlet's producer with the application. Oracle JDeveloper provides two producer registration wizards: the WSRP Producer Registration Wizard and the PDK-Java Producer Registration Wizard. This section describes how to use these wizards. It contains the following subsections:

- Section 4.3.1.1, "Registering WSRP Portlet Producers"
- Section 4.3.1.2, "Registering PDK-Java Portlet Producers"
- Section 4.3.1.3, "Editing Portlet Producer Registration Settings"
- Section 4.3.1.4, "Testing a Portlet Producer Connection"
- Section 4.3.1.5, "Refreshing a Portlet Producer"
- Section 4.3.1.6, "Deregistering a Portlet Producer"

Note: For additional information about producers, see Section 15.4, "Deployment Type".

4.3.1.1 Registering WSRP Portlet Producers

When you register a WSRP portlet producer, you provide basic information that describes the producer's operational parameters. This information is used by the portlet-consuming application to communicate with the producer and with the portlets through the producer.

Oracle WebCenter Framework supports both WSRP 1.0 and WSRP 2.0 producers. WSRP 2.0 support is for a preliminary (that is, preproduction) version of WSRP 2.0. Preliminary support is provided to accommodate the fact that the WSRP 2.0 standard was not finalized when Oracle WebCenter Framework was released.

This emerging standard, among others, provides support for inter-portlet communication and export or import of portlet customizations. This means that you can leverage the benefits of WSRP 2.0 while building standards-based JSR 168 portlets. To take advantage of more advanced features of WSRP 2.0, use the Oracle-specific `oracle-portlet.xml` metadata extensions.

To take advantage of the benefits WSRP 2.0 provides, JSR 168 portlets exposed through WSRP 2.0 must be deployed to Oracle Container for J2EE (OC4J). Because the new version of the portlet application programming interface (API) (JSR 286) will support WSRP 2.0 capabilities, when JSR 286 becomes publicly available, vendor-specific extensions will no longer be required.

The Producer Registration Wizard is the entry point for registering both WSRP 1.0 and 2.0 producers. When registration is successful, the newly registered producer displays in Oracle JDeveloper on the Component Palette, from which you can select portlets for placement on your application (`jspx`) page. Additionally, the producer is listed under the Portlet Producers node in the Applications Navigator.

To register a WSRP portlet producer, perform the following steps:

1. In the Applications Navigator, right-click the application under which to create the producer and select **New** from the context menu.
2. In New Gallery under **Categories**, expand the Web Tier node and select **Portlets**.
3. In New Gallery under **Items**, select **WSRP Producer Registration**.
4. Click **OK**.

5. On the Welcome page, click **Next**.

   Optionally, before clicking Next, select **Skip this Page Next Time** to forgo display of the Welcome page on subsequent uses of this wizard. The Welcome page may not display if the option to skip was selected on earlier use of the wizard.

6. In the **Name** field, enter a name for the producer.

   Give the producer a name that is unique among producers.

   > **Note:** Producers given the same name display only once on the Component Palette. So, if you register two producers and give them each the name *MyProducer*, then the name *MyProducer* displays only once on the Component Palette, and both producers' portlets are listed under the one instance. This makes it difficult to determine which portlets come from which producer.
   >
   > If you find yourself in this situation, then consider editing one of these producers and giving it a unique name. See Section 4.3.1.3, "Editing Portlet Producer Registration Settings".

7. Click **Next**.

8. In the **URL Endpoint** field, enter the producer's URL.

   The syntax will vary according to your WSRP implementation:

   ```
   http://<host>:<port>/<context-root>/portlets/wsrp1?WSDL
   http://<host>:<port>/<context-root>/portlets/wsrp2?WSDL
   http://<host>:<port>/<context-root>/portlets?WSDL (WSRP 1.0 for backward
   compatibility)
   ```

   Where:

   - `host` is the server to which your producer has been deployed.
   - `port` is the HTTP Listener port number.
   - `context-root` is the context root.
   - `portlets[/wsrp(1|2)]?WSDL` is static text. The text entered here depends on how the producer is deployed.

   For example:

   ```
   http://myhost:7778/MyPortletApp/porlets/wsrp2?WSDL
   ```

   You can access the producer test page through the URL:

   ```
   http://host:port/context-root/info
   ```

   > **Note:** While the URL Endpoint is editable, use the edit feature only to update the host name, port, or IP address. Do not use the edit feature to point to a new producer. Switching from one producer to another (even when the portlets are identical) is not supported by the WSRP specification.

9. If the application will use an HTTP proxy to connect to the producer, perform the following steps:

Note: The proxy fields on this panel default to the proxy preferences set in Oracle JDeveloper Preferences (From the **Tools** menu, select **Preferences**, then select **Web Browser and Proxy**.)

a. Select **Use proxy for contacting the WSRP Portlet Producer**.

   Select this check box if the application will use an HTTP proxy in contacting the producer. The proxy is required in cases where the consumer application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed for communication with the producer.

b. In the **Proxy Host** field, enter the URL of the proxy.

c. In the **Proxy Port** field, enter the port number of the proxy.

10. Click **Next**.

11. In the **Default Execution Timeout (Seconds)** field, enter the number of seconds to enable for a portlet to render in the WebCenter application before it times out.

    Some producers can define additional registration properties. In such cases, the properties are displayed in a table on the wizard's Registration Details panel. Users can enter values for these additional properties in the table. These properties are producer-specific and are used only at registration time. That is, they collect information that consumer applications send to producers at registration time; the producers store this information against the consumers and use it subsequently.

    At this point, you can click **Finish** to complete registration. Continue if you plan to require authentication whenever the producer (and consequently its portlet) is accessed, or if you plan to map user categories defined for the producer's portlets with J2EE security roles defined for the application.

12. Click **Next**.

13. From the Token Profile list, select the type of token profile to use for authentication with the WSRP Producer.

    Choose from the following:

    - **Username Token**—A Web service consumer can supply a Username Token as a means of identifying the requestor by user name to authenticate that identity to the Web service producer.
    - **SAML Token**—SAML (Security Assertion Markup Language) is an XML-based approach for passing security tokens defining authentication and authorization rights. An attesting entity (that already has trust relationship with the receiver) vouches for the verification of the subject by method called sender-vouches.
    - **None**—No token. If None is selected, then no WS-Security header is attached to the SOAP message. In this case, the Finish button is enabled to complete producer registration.

14. In the **Default User** field, enter a user name to assert to the remote producer when the user has not authenticated to the WebCenter application.

    When unauthenticated, the identity *anonymous* is associated with the application user. The value *anonymous* may be inappropriate for the remote producer, so you may need to specify an alternative identity here. Keep in mind though, that in this case, the WebCenter application has not authenticated the user so the default user you specify should be a lowly privileged user in the remote producer. If the user

has authenticated to the application, then the user's identity is asserted rather than the default user.

15. In the **Issuer Name** field, enter the name of the issuer of the SAML Token, for example `www.oracle.com`.

    This field appears only when SAML Token is selected from the Token Profile list. The issuer name is the attesting entity vouches for the verification of the subject.

16. Under **XML Signature**, select the means by which the signing certificate is referenced within the WS-Security KeyInfo.

    An XML Signature is used to digitally sign the security token and the SOAP message body to provide authenticity of the SOAP message. Use this panel to specify the means by which the signing certificate is referenced within the WS-Security KeyInfo.

    Select either **Binary Security Token** or **Subject Key Identifier**.

17. Click **Next**.

18. In the **Store Path** field, provide the full path to the Key Store that contains the certificate and the private key that is used for signing some parts (security token and SOAP message body) of the SOAP message.

    Optionally, click **Browse** to navigate to and select the file. The selected file could be a key store created with the Java keytool, or it could be an Oracle Wallet.

19. In the **Store Password** field, provide the password to the Key Store that was set when the Key Store was created.

    The store password must be correct for the **Store Type** field and the **Signature Key Alias** list to populate.

    If an incorrect password is entered, then an error message appears stating that the password is invalid and must be corrected. After you correct the password, press the **Tab** key to move to another active field (for example, the **Store Path** field). This ensures that the Store Type field and the Signature Key Alias list are properly populated.

    The **Store Type** value is read from the Key Store and is never editable. Applicable values include **JKS** (Java Key Store) or **Oracle Wallet**.

20. From the **Signature Key Alias** list, select the signature key alias.

    The Signature Key Alias list populates automatically when the correct password is entered in the **Store Password** field. The Signature Key Alias is the identifier for the certificate associated with the private key that is used for signing. The key aliases found in the specified key store are available in the list. Select the one to be used for signing.

21. In the **Signature Key Password** field, specify the password for accessing the key identified by the alias specified in Signature Key Alias.

22. Click **Finish** to complete registration of the WSRP portlet producer.

The next step depends on whether the producer being registered declares user categories:

- If the producer does not declare user categories, then the registration process is complete.
- If the producer declares user categories, then click **Yes** and see "Mapping a Producer's Declared User Categories to an Application's Defined J2EE Security Roles". Click **No** to decline this opportunity and complete producer registration.

Note: Most registration settings can be edited. For example, if you decline to map the producer's user categories with J2EE security roles now, then you can reenter the wizard later and provide mapping information about the User Categories tab. For additional information, see Section 4.3.1.3, "Editing Portlet Producer Registration Settings".

Mapping a Producer's Declared User Categories to an Application's Defined J2EE Security Roles

The user categories the producer declares come from the portlets it contains. For example, if the producer contains one or more JSR 168 portlets created with the Standards-based Java Portlet (JSR 168) Wizard, then any security roles added during portlet creation are included in the user categories the producer declares. J2EE Security Roles can be specified through the WebCenter application's `web.xml` file properties.

This procedure continues forward from the previous procedure.

To map producer-declared user categories with application-defined J2EE security roles, perform the following steps:

1. In the mapping dialog box, click in the Application J2EE Security Role column.

   This dialog box is accessed during producer registration, by completing registration and clicking **Yes** when given the option to continue with mapping (see Section 4.3.1.1). It is accessible as well when you edit producer registration settings (see Section 4.3.1.3).

2. From the resulting list, select the security role to map to the Producer User Category.
3. Repeat steps 1 and 2 for each user category.
4. Click **OK** when all user categories are mapped.

4.3.1.2 Registering PDK-Java Portlet Producers

When you register an Oracle PDK-Java portlet producer, you provide basic information that describes the producer's operational parameters. This information is consumed by the WebCenter application for use in communicating with the producer and with portlets through the producer.

When registration is successful, the newly registered producer displays in Oracle JDeveloper on the Component Palette, from which you can now select portlets for placement on your application (`jspx`) page. Additionally, the producer is listed under the Portlet Producers node in the Applications Navigator.

To register an Oracle PDK-Java Portlet Producer, perform the following steps:

1. In the Applications Navigator, right-click the application under which to create the producer and select **New** from the context menu.
2. In New Gallery under **Categories**, expand the Web Tier node and select **Portlets**.
3. In New Gallery under **Items**, select **Oracle PDK-Java Producer Registration**.
4. Click **OK**.
5. On the Welcome page, click **Next**.

   Optionally, before clicking Next, select **Skip this Page Next Time** to forgo display of the Welcome page on subsequent uses of this wizard. The Welcome page may not display if the option to skip was selected on earlier use of the wizard.

6. In the **Name** field, enter a name for the producer.

   Give the producer a name that is unique among producers.

   > **Note:** Producers given the same name display only once on the Component Palette. So, if you register two producers and give them each the name *MyProducer*, then the name *MyProducer* displays only once on the Component Palette, and both producers' portlets are listed under the one instance. This makes it difficult to determine which portlets come from which producer.
   >
   > If you find yourself in this situation, then consider editing one of these producers and giving it a unique name. See Section 4.3.1.3, "Editing Portlet Producer Registration Settings".

7. Click **Next**.

8. In the **URL Endpoint** field, enter the producer's URL using the following syntax:

   ```
   http://<host>:<port>/<context-root>/providers
   ```

   Where:

   - `host` is the server to which your producer has been deployed.
   - `port` is the port to which the server is listening for HTTP requests.
   - `context-root` is the Web application's context root.
   - `providers` is static text. The text entered here depends on how the producer is deployed.

   For example:

   ```
   http://myHost:7778/myEnterprisePortlets/providers
   ```

9. In the **Service ID** field, enter a unique identifier for this producer.

   PDK-Java enables you to deploy multiple producers under a single adapter servlet. The producers are identified by their unique service IDs. A service ID is required only when a service ID or producer name is not appended to the URL endpoint. For example the following URL endpoint requires the service ID, `sample`:

   ```
   http://domain.us.company.com:<port_number>/axyz/providers
   ```

   However, the following URL endpoint, does not require a service ID:

   ```
   http://domain.us.company.com:<port_number>/axyz/providers/sample
   ```

10. If the application will use an HTTP proxy to connect to the producer, perform the following steps:

    > **Note:** The proxy fields on this panel default to the proxy preferences set in Oracle JDeveloper Preferences (From the **Tools** menu, select **Preferences**, then select **Web Browser and Proxy**.)

    a. Select **Use proxy for contacting the PDK Portlet Producer**.

Select this check box if the application will use an HTTP proxy in contacting the producer. The proxy is required in cases where the consumer application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed for communication with the producer.

b. In the **Proxy Host** field, enter the URL of the proxy host.

c. In the **Proxy Port** field, enter the port number of the proxy host.

11. Select **Associate producer with external application**, then select the application, in the event this producer must provide authentication to an external application.

    For more information, see Section 10.7, "Accessing External Applications Requiring Credentials".

12. Select **Enable Producer Sessions** to enable portlet producer sessions.

    Use this option to enable sessions between the producer and the OC4J server. For sessionless communication between the producer and the server, do not select this option.

    When sessions are enabled, the server maintains session-specific information, such as user name. Message authentication uses sessions, so if the shared key is set, then this option should also be selected.

13. Click **Next**.

14. In the **Default Execution Timeout (Seconds)** field, enter the number of seconds to enable for a portlet to render in the WebCenter application before it times out.

15. In the **Subscriber ID** field, enter a string to identify the consumer of the producer being registered.

    When a producer is registered, a call is made to the producer. During the call, the consumer passes the value for Subscriber ID to the producer. If the producer does not see the expected value for Subscriber ID, then it might reject the registration call.

16. In the **Shared Key** field, enter a shared key to use for producers that are set up to handle encryption.

    The shared key is used by the encryption algorithm to generate a message signature for message authentication. Note that producer registration will fail if the producer is set up with a shared key and you enter an incorrect shared key here. The shared key can contain between 10 and 20 alphanumeric characters.

17. Click **Finish** to complete registration of the PDK-Java portlet producer.

18. Click **OK**, to close the success message.

4.3.1.3 Editing Portlet Producer Registration Settings

Both the WSRP and PDK-Java portlet producer registration wizards enable you to access and revise many of the values you entered when you registered the producer.

Note: Once you have completed your edits, consider testing the producer connection to be sure connection information is valid. For more information, see Section 4.3.1.4, "Testing a Portlet Producer Connection".

To edit a WSRP or Oracle PDK-Java Portlet Producer, perform the following steps:

1. In the Applications Navigator, navigate to the producer:

```
Applications
   <ApplicationName>
      Portlet Producers
```

2. Right-click the producer to be edited, and select **Edit** from the context menu.

> **Note:** You can also double-click the producer in the Applications Navigator to open the producer in edit mode.

3. Click the tabs at the top of the wizard to bring different setting panels forward:
   - For information about WSRP Producer settings, see Section 4.3.1.1, "Registering WSRP Portlet Producers".
   - For information about PDK-Java Producer settings, see Section 4.3.1.2, "Registering PDK-Java Portlet Producers".

4.3.1.4 Testing a Portlet Producer Connection

The connection testing feature provides a means of testing the validity of a portlet producer connection.

To test a portlet producer connection, perform the following steps:

1. In the Applications Navigator, navigate to the producer:

```
Applications
   <ApplicationName>
      Portlet Producers
         <ProducerName>
```

2. Right-click the producer to be edited, and select **Test Producer Connection** from the context menu.

   A progress bar appears while the test is underway. A success or failure dialog box displays when the test is complete. Click **OK** to close this dialog box.

   If the failure dialog box displays, then consider reediting producer registration details and retesting the producer connection. Additionally, make sure that the producer is available. For example, if the producer is provided through the preconfigured Oracle Containers for J2EE (OC4J), then make sure the preconfigured OC4J is running, and then retest the connection.

4.3.1.5 Refreshing a Portlet Producer

When you refresh a portlet producer, the portlets from that producer are also refreshed. This means that newly added portlets and any updates to existing portlets become available to any applications that are consuming portlets from this producer.

> **Note:** When a portlet is removed from a producer, be sure to manually delete the portlet from all application pages on which it has been placed. For more information, see Section 4.3.5, "Deleting Portlets from Application Pages".

To refresh a WSRP or Oracle PDK-Java Portlet Producer, perform the following steps:

1. In the Applications Navigator, navigate to the producer:

```
Applications
   <ApplicationName>
      Portlet Producers
```

2. Right-click the producer to be edited, and select **Refresh** from the context menu.
3. In the Refresh Portlet Producer dialog box, click **Yes**.
4. Click **OK** to close the Success dialog box.

4.3.1.6 Deregistering a Portlet Producer

When you deregister a producer, registration data is removed on both the WebCenter application end and the remote producer end. On the application end the producer connection is deleted. On the producer end, portlet instances are deleted (though not the portlets themselves).

Though portlet instances are removed on the remote producer end, they are not also removed on the application end. Therefore, when you deregister a portlet producer, you must also remove any portlets the producer provides from your application pages. Additionally, if the portlet included parameters, then any associated page variables must be removed from any affected application page's Page Definition file.

> **Note:** For information about deleting portlets and relevant page variables, see Section 4.3.5, "Deleting Portlets from Application Pages".

To deregister a portlet producer, perform the following steps:

1. In the Applications Navigator, navigate to the producer:

```
Applications
   <ApplicationName>
      Portlet Producers
         <producer_name>
```

2. Right-click the producer to be deregistered, and select **Deregister** from the context menu.
3. In the Portlet Producer Deregister dialog box, click **Yes**.

   Instead of the Portlet Producer Deregister dialog box, you may see a Connection Error dialog box. This dialog displays when a connection cannot be made to the producer.

   Connection failure can occur when the producer is not available or the producer's connection details are incorrectly specified in the WebCenter application. Your options are to click **Yes** or **No**:

   - Click **Yes** to continue with deregistration. When you click Yes, registration data is removed on the WebCenter application end but remains untouched on the remote producer end.
   - Click **No** to cancel deregistration. Should you cancel, you can try again after verifying that the producer is available and the connection details are valid in the Producer Registration wizard (see Section 4.3.1.3, "Editing Portlet Producer Registration Settings").
4. Click **OK** to close the resulting success message.

4.3.2 Adding Portlets to a Page

Placing a portlet on a WebCenter application page is a simple matter of dragging-and-dropping, though there are a few preparatory steps you must take before you can take this simple action. These include, creating a WebCenter application (see Section 3.1, "Creating a WebCenter Application"), creating an application page (see Section 4.2, "Building WebCenter Application-Enabled Pages in Oracle JDeveloper with Oracle ADF"), and registering the portlet's producer with the application (see Section 4.3.1, "Registering Portlet Producers").

> **Note:** For information about portlet parameters, see Section 4.5, "Contextually Linking Components".

When you examine the source of the application page on which you have placed a portlet, you will see an `adfp:portlet` tag. This is the tag that binds the portlet to the page. It carries with it a string of attributes you can revise to further control the behavior and appearance of the portlet. For more information, see Section 4.3.3, "Setting Attribute Values for the adfp:portlet Tag".

> **Note:** Some of the portlets you plan to consume may come from applications that handle their own authentication. In such cases, you must register the application as an external application and identify it to the portlet producer that will provide it. For more information, see Chapter 10, "Securing Your WebCenter Application".
>
> Some of the portlets you plan to consume may come from producers that are Secure Sockets Layer (SSL) enabled. When you try to access an SSL-enabled producer, a Security Alert dialog box may pop up, prompting you to view the producer's security certificate and add it to the list of trusted certificates. The Security Alert dialog box is displayed only if the producer uses a security certificate issued by a certificate authority that is not widely accepted. To consume portlets from such a producer, you must first add the producer's security certificate to the keystore. See Section 10.8, "Registering Custom Certificates with the Keystore" for the steps to be performed.

To add a portlet to a page, perform the following steps:

1. In the Applications Navigator, right-click the application page (`jspx` file) to which you will add the portlet, and select **Open** from the context menu.

   You can find the `jspx` file at the following location in the Applications Navigator:

   ```
   Applications
      <ApplicationName>
         <ProjectName>
            Web Content
              <ApplicationPage>.jspx
   ```

2. In the Component Palette, select the portlet producer that contains the portlet you will add to the application page.

   Under the selected producer, the Component Palette lists all portlets contained by that producer.

3. Select a portlet, and drag it over the `h:form` element in the Oracle JDeveloper Structure pane.

Alternatively, drag the portlet from the Component Palette directly onto the page in the editor.

The portlet should be nested somewhere in an `h:form` element. It need not necessarily be directly wrapped in `h:form` tags. If you add the component outside an `h:form` element, then Oracle JDeveloper asks whether you want the form element to wrapped around it. Select this option.

If the application page includes one or more core customizable components, then this may influence where the portlet is placed. For example, in the Structure pane, a portlet placed on a page with a `cust:panelCustomizable` tag, would be placed as illustrated in Example 4–1:

Example 4–1 Hierarchical Placement of the adfp:portlet Tag

```
h:form
   cust:panelCustomizable
      adfp:portlet
```

For information about the core customizable tags, `cust:panelCustomizable` and `cust:showDetailFrame`, see Section 4.4, "Using Customizable Components".

When you add a portlet to a page, an `adfp:portlet` tag is added to the page source. The `adfp:portlet` tag includes a number of attributes accessible through the Oracle JDeveloper Property Inspector. For information about attributes of the `adfp:portlet` tag, see Section 4.3.3, "Setting Attribute Values for the adfp:portlet Tag".

Note: When you drop an instance of OmniPortlet onto your page, open the Property Inspector and ensure that the **AllModesSharedScreen**, under the Display Mode category, is set to `false`, the default value. Setting this property to `true` may prevent you from editing certain sections of your OmniPortlet in the OmniPortlet wizard.

Once you place a portlet on a page, right-click the page and select **Run** from the context menu. This displays the page and runs the portlet in your default browser using Oracle JDeveloper's embedded OC4J. Different portlets may require additional run time configuration. Notably, the content of an OmniPortlet or Web Clipping portlet instance is defined at run time. For more information about OmniPortlet, see Chapter 16, "Creating Portlets with OmniPortlet". For more information about the Web Clipping portlet, see Chapter 17, "Creating Content-Based Portlets with Web Clipping". For more information about portlets generally, see Chapter 14, "Understanding Portlets" and Chapter 15, "Portlet Technologies Matrix".

When running a portlet that has an Edit mode (in a WebCenter application, this renders as a Personalize command on the portlet's Actions menu), the Personalize option displays in the portlet's Actions menu only to authenticated users (that is, users who have logged in). Anonymous or public users do not see the option to personalize the portlet. Some form of security must be implemented for the portlet-consuming application before users can personalize their view of a portlet. If you are a developer creating portlets and pages, then you may want to test your portlet's Edit mode without creating a complete security model for your application. See Section 10.6, "Configuring Basic Authentication for Testing Portlet Personalization" for an explanation of how to add security to enable testing of portlet personalization (that is, for testing your portlet's Edit mode).

4.3.3 Setting Attribute Values for the adfp:portlet Tag

On application source pages, each portlet is represented by an `adfp:portlet` tag, which includes a set of required and optional attributes. Required attributes, `value` and `portletType`, are provided automatically by the framework, and must not be altered. Optional attribute values are relevant when support for the attribute is built into the portlet. For example, you can set `isAboutModeAvailable` to `true`, but if no About mode has been defined for the portlet, then the attribute setting does not affect the portlet.

Portlets also support a set of style-related attributes, which are discussed more fully in Section 9.4, "Defining Styles Through the Property Inspector".

Set attribute values at design time either through the Oracle JDeveloper Property Inspector or in the source as attributes of the `adfp:portlet` tag, shown in Example 4–2.

Example 4–2 adfp:portlet Tag

```
<adfp:portlet value="#{bindings.portlet1}"
   portletType="/oracle/adf/portlet/WsrpPortletProducer1/applicationPortlets/
      E0default_b452f828_010a_1000_8002_82235f57eaa8"
   allModesSharedScreen="true"
   isMaximizable="true"
   isMinimizable="true"/>
```

The `adfp:portlet` tag uses four types of attributes, which this section lists and describes. It contains the following subsections:

- Section 4.3.3.1, "General Attributes of the adfp:portlet Tag"
- Section 4.3.3.2, "Actions Attributes of the adfp:portlet Tag"
- Section 4.3.3.4, "Core Attributes of the adfp:portlet Tag"
- Section 4.3.3.5, "Display Mode Attributes of the adfp:portlet Tag"

4.3.3.1 General Attributes of the adfp:portlet Tag

Table 4–1 describes the general attributes of the adfp:portlet tag.

Table 4–1 General Attributes of the adfp:portlet Tag

Attribute	Value	Description
`height`	A number expressed in pixels or as a percentage of the available area: ■ For pixels, enter *n*px, for example, 300px. ■ For percentage, enter *n*%, for example, 50%.	The height of area to enable for portlet display. If the actual portlet height is larger than the `height` value entered here, then a scrollbar appears, provided `displayScrollBar` is set to `auto` or `true`. If `displayScrollBar` is set to `false`, and actual portlet height exceeds the value expressed for the `height` attribute, then the `height` attribute value is considered and the portlet content is truncated.

`icon`	The URI to an image. For example:
	`icon="coffee.png"`
	In Oracle JDeveloper, click the Edit icon in the value column to locate and select the required image. The URI provided in this example is stored at the document root; therefore, a full path is not required. An image that is not stored at the document root requires a full path, for example:
	`icon="C:\portal\images\box_b.gif"`
	Or:
	`icon="http://source-pc/imgs/art.gif"`
		A URI specifying the location of an icon to display to the left of the portlet title in the portlet header.

`id`	A text string to use as the portlet's unique identifier. For example: `id="newsBrief"`	The unique identifier of the component.
`text`	A text string to use as the portlet's header title. For example: `text="Announcements …"`	The portlet header title. Use this to supply a display title on the portlet. The attribute value specified in the `<adfp:portlet>` tag takes precedence over any title specified elsewhere (for example, in the portlet mark-up). If no value is specified, then the portlet extracts its title from the portlet mark-up (response). If neither the `text` attribute nor the portlet mark-up is available, then the title is extracted from the portlet definition. **Note:** Supplying a value to the `text` attribute at design-time prevents customization and personalization of the portlet title at run time.

Table 4–1 (Cont.) General Attributes of the adfp:portlet Tag

Attribute	Value	Description
`width`	A number expressed in pixels or as a percentage of available area: ■ For pixels, enter *n*px, for example, 300px. ■ For percentage, enter *n*%, for example, 50%.	The width of the portlet. If the actual portlet width is larger than the `width` value entered here, then a scrollbar appears, provided `displayScrollBar` is set to `auto` or `true`. If `displayScrollBar` is set to `false`, and actual portlet width exceeds the value expressed for the `width` attribute, then the `width` attribute value is considered and the portlet content is truncated.
`binding`	The name of the managed bean. For example: `binding="#{frameActionsBean.Binding}"` In Oracle JDeveloper, click the Edit icon in the value column to select a managed bean and specify the relevant managed bean property.	A binding reference to store the component instance. The binding reference binds an instance of the portlet to a managed bean property. Managed beans are any JavaBeans used by the application that are registered in the JSF `faces-config.xml` file.
`partialTriggers`	The ID of one or more components that trigger a partial update. For example: `partialTriggers="_id1 componentID5"`	IDs of components that trigger a partial update. The portlet listens on the specified trigger components. If one of the trigger components receives a trigger event that causes it to update in some way, then the portlet also requests to be updated. In the `partialTriggers` tag, components are separated from each other by a space.
`portletType`	This required attribute value is provided by the framework by default.	`portletType` is used to differentiate between different portlets in the design-time environment. The value is provided programmatically by the framework and should not be revised or removed.
`shortDesc`	A text string to use as a brief description of the portlet. For example: `shortDesc="Portlet for entering display text in place."`	A short description of the portlet.
`value`	This required attribute value is provided by default.	The portlet gets its portlet container (producer) support from the reference to the portlet binding in the page definition. This attribute value is provided programmatically and should not be revised or removed.
`submitUrlParamters`	`false`	Portlet links that point to the page on which the portlet is situated force a page to submit itself rather than just reload with the link URL. By default, the parameters in this URL are not made available to the page. Rather, they are available only inside the portlet initiating the request. Setting `submitUrlParameters` to `true` makes these URL parameters available on the container page as well.

4.3.3.2 Actions Attributes of the adfp:portlet Tag

Actions attributes control the rendering of mode-switching UI actions, such as entering edit mode. The ability to render a portlet in a particular mode depends on the modes supported by the portlet and the user authorization. For example, if the `isCustomizeModeAvailable` attribute is set to `true`, but the action is not supported in the portlet, then the attribute setting will not affect the portlet.

Actions attributes, described in Table 4–2, are value binding expressions that evaluates to `true` or `false`:

- `true` means the portlet is enabled to render in the named mode.
- `false` means the portlet is not enabled to render in the named mode.

Table 4–2 Actions Attributes of the adfp:portlet Tag

Attribute	Default	Description
`isAboutModeAvailable`	`true`	In a WebCenter application, renders an **About** command on the portlet's Actions menu. Users select About to invoke the portlet's About mode.
`isConfigModeAvailable`	`true`	In a WebCenter application, renders a **Configure** command on a JSR 168 portlet's Actions menu. Users select Configure to open the portlet's Configuration settings.
`isCustomizeModeAvailable`	`true`	In a WebCenter application, renders a Customize command on the portlet's Actions menu. Customize mode enables site administrators to edit a portlet's default personalization data.
`isDetailModeAvailable`	`true`	In a WebCenter application, renders a **Details** command on the portlet's Actions menu. Users select Details to open the portlet's Details page. This attribute maps to the *Show details page* mode in Oracle PDK-Java portlets. It has no application in standards-based (JSR 168) Java portlets.
`isHelpModeAvailable`	`true`	In a WebCenter application, renders a **Help** command on the portlet's Actions menu. Users select Help to open the portlet's Help page.
`isMaximizable`	`true`	In a WebCenter application, renders a **Maximize** command on the portlets Actions menu. Users select Maximize to expand the portlet. When the component is maximized, the **Restore** command displays on the Actions menu to return the portlet to its default display mode. The Maximize option applies only to portlets placed inside a `PanelCustomizable` component. When users select Maximize, the portlet expands to the width of its parent `PanelCustomizable`, displacing all other components placed in the `PanelCustomizable`. There is a difference in the way the Maximize and Restore actions work at design-time and at run time. For more information, see Section 4.3.3.3, "What You Should Know About Maximize, Minimize, Restore, and Move".
`isMinimizable`	`true`	In a WebCenter application, renders a **Minimize** icon on the portlet header. Users click the icon to collapse the portlet like a window shade. Users restore the portlet by clicking the icon again. There is a difference in the way the Minimize action behaves at design-time and at run time. For more information, see Section 4.3.3.3, "What You Should Know About Maximize, Minimize, Restore, and Move".

Table 4–2 (Cont.) Actions Attributes of the adfp:portlet Tag

Attribute	Default	Description
`isMovable`	`true`	In a WebCenter application, renders a **Move** command on the portlet's Actions menu. Users select the command, then the subcommand **Move Up**, **Move Down**, **Move Left**, or **Move Right**, depending on the portlet's current position related to the other portlets on the page. There is a difference in the way the Move action behaves at design-time and at run time. For more information, see Section 4.3.3.3, "What You Should Know About Maximize, Minimize, Restore, and Move".
`isNormalModeAvailable`	`true`	In a WebCenter application, renders a **Refresh** command on the portlet's Actions menu. Users select the Refresh command and the portlet refreshes (that is, redraws) independent of any other content on the page (also known as a partial-page refresh).
`isPersonalizeModeAvailable`	`true`	In a WebCenter application, renders a **Personalize** command on the portlet's Actions menu. Users select Personalize to alter their personal view of the portlet. This mode is equivalent to the Edit mode selection in the Standards-based Java Portlet (JSR168) Wizard. The Personalize command displays on the Actions menu only to authenticated users (that is, users who are logged in). It does not display to Public or unauthenticated users. You must implement some form of application security for users to be able to personalize their portlet views. If you are a developer creating portlets, and you want to test the Personalize mode without creating a complete security model for your application, then see Section 10.6, "Configuring Basic Authentication for Testing Portlet Personalization". **Note:** A typical personalization setting is Portlet Title. You can set Portlet Title at design-time, by providing a value for the `text` attribute of the `adfp:portlet` tag. Consider however that supplying a value to the `text` attribute at design-time prevents personalization and customization of the portlet title at run time.
`isPreviewModeAvailable`	`false`	Provides a means of previewing portlet content. This mode has no particular application in WebCenter applications, but it is used in Oracle Application Server Portal's (OracleAS Portal) Portlet Repository, where it renders as a magnifying glass icon, which users click to preview a portlet.
`isPrintModeAvailable`	`true`	In a WebCenter application, renders a **Print** command on a JSR 168 portlet's Actions menu that displays a printer-friendly version of the portlet.
`isSeededInteractionAvailable`	`true`	Makes the portlet's seeded interactions, Maximize, Move, and Minimize, available for exposure to users. When the attribute is set to `true`, seeded interactions are available for display (note, however, that display of each seeded interaction is controlled by its own attribute setting). When the attribute is set to `false`, seeded interactions are not available for display. Custom interactions are not affected by this attribute setting.

4.3.3.3 What You Should Know About Maximize, Minimize, Restore, and Move

To accommodate the needs of the development environment, the behavior of the actions *Minimize*, *Maximize*, *Restore*, and *Move* for `ShowDetailFrame` and `portlet` components differs between design-time and run time. At design-time, these actions persist in a given OC4J session, but do not persist over sessions (*session* means the time

between starting and stopping the OC4J). At run time, these actions persist both during a given OC4J session and across sessions.

This difference has been introduced to enable an automatic reset of an application page at design-time.

If persisting across sessions is not required at run time, then a simple modification to the application's `web.xml` file can turn it off. Go to the following parameter setting in the application's `web.xml` file (Example 4–3):

Example 4–3 Persistence Setting in the Application's web.xml File

```
<context-param>
   <param-name>oracle.adf.view.faces.CHANGE_PERSISTENCE</param-name>
   <param-value>oracle.adfinternal.view.faces.change.HybridChangeManager</param-va
lue>
</context-param>
```

Replace it with the following (Example 4–4):

Example 4–4 Turning Run-Time Persistence Off in the Application's web.xml File

```
<context-param>
   <param-name>oracle.adf.view.faces.CHANGE_PERSISTENCE</param-name>
   <param-value>oracle.adf.view.faces.change.SessionChangeManager</param-value>
</context-param>
```

If security has been implemented on the application, then the *Minimize*, *Maximize*, *Restore*, and *Move* actions display only to users with Customize privileges. They do not display to users with Personalize privileges. Customize users can test the effect of these actions by following these steps at design-time:

- Run the application page using Oracle JDeveloper's embedded OC4J.
- Log-in as the administrator.
- Maximize a portlet. Move portlets around. Make whatever changes you want using the relevant actions commands.
- Log-out, then log-in as a user and check the effects of your actions.

4.3.3.4 Core Attributes of the adfp:portlet Tag

Table 4–3 describes the core attributes of the adfp:portlet tag.

Table 4–3 Core Attributes of the adfp:portlet Tag

Attribute	Value	Description
`contentInlineStyle`	The name of a CSS style For example, to turn off the portlet border, add the following to the `adfp:portlet` tag: `contentInlineStyle="border-style:none;"`	The CSS style to apply to the portlet. Expand this node to specify styles for specific style elements on the specific portlet instance. Values entered here take precedence over styles included in a CSS or skin on the specific portlet instance. For more information, see Section 9.4, "Defining Styles Through the Property Inspector".
`inlineStyle`	The name of a CSS style	The CSS style to apply to the portlet. Expand this node to specify styles for specific style elements on the specific portlet instance. Values entered here take precedence over styles included in a CSS or skin on the specific portlet instance. For more information, see Section 9.4, "Defining Styles Through the Property Inspector".
`rendered`	`true/false`	Specifies whether the portlet is rendered. When set to `false`, no output is rendered. The default value is `true`.

4.3.3.5 Display Mode Attributes of the adfp:portlet Tag

Table 4–4 describes the display mode attributes of the adfp:portlet tag.

Table 4–4 Display Mode Attributes of the adfp:portlet Tag

Attribute	Value	Description
`allModesSharedScreen`	`true/false`	Determines whether a change in portlet mode renders the new mode on a new page, other than the page on which the portlet resides (`false`). ■ `true` means all portlet modes are displayed inline. One mode is swapped out for another on the same page. In other words, this attribute enables all portlet modes to display without leaving the context of a given page. ■ `false` means all portlet modes, except View (JSR 168) or Show (PDK-Java), are rendered each on their own page. The default mode is useful for such portlets as OmniPortlet and the Web Clipping portlet, which require that modes other than Show mode display on pages other than the page on which the portlet resides. The default value is `false`.
`background`	`light/medium/dark`	Provides a means of applying a different look and feel to each portlet on an application page. The default skins, Oracle, Minimal, and Simple, and custom skins include three versions of style selectors: light, medium, and dark. Depending on which value is specified for the background property, the skin will apply the relevant style or icon selector version. For more information, see Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components".
`displayHeader`	`true/false`	Indicates whether the portlet's header is displayed. ■ `true` means the header is displayed. Consequently, header-based icons and links are displayed. ■ `false` means the header is not displayed, and icons and links normally displayed in the header are hidden. If `isSeededInteractionAvailable` is set to `true`, then the user can access portlet menus and icons by rolling the mouse over the portlet. A fade-in/fade-out toolbar appears, from which users can select Actions menu options. The default value is `true`.

Table 4–4 (Cont.) Display Mode Attributes of the adfp:portlet Tag

Attribute	Value	Description
expansionMode	maximized/minimized/normal	The default state of the portlet. Select from: ▪ minimized—Portlet's default display mode is collapsed (minimized). ▪ maximized—This option applies only to portlets placed inside a PanelCustomizable component. Portlet's default display mode is expanded to the width of its parent PanelCustomizable, displacing all other components placed in the PanelCustomizable. ▪ normal—This is the default value for the expansionMode attribute. Portlet's default display mode is normal. That is, it is neither collapsed nor expanded to the width of the page.
displayScrollBar	true/false/auto	Display a scrollbar if content does not fit the width and height specified (auto). A value of true always renders a scroll bar. A value of false never renders a scroll bar. If the portlet exceeds the values expressed for width or height, then these values are ignored and the portlet renders in its actual size. The default is auto.
renderPortletInIFrame	true/false/auto	A value binding expression that evaluates to true/false/auto: ▪ true means the portlet is rendered in an iframe. ▪ auto means the portlet tag checks the portlet response and decides if an iframe is required. It bases its decision on the existence of FORM or any links inside the portlet response. ▪ false Although this value is available for selection, it is not supported. HTML mark-up from a portlet that is not rendered in an iframe may interfere with other components on the Oracle ADF page. The default is auto. For additional information, see Section 4.3.3.6, "iframes and form Tags".

4.3.3.6 iframes and form Tags

The HTML source generated by a JavaServer Faces page contains a form tag. Because nested form tags are not enabled in HTML, portlets cannot open new forms on the same page.

The best way to circumvent this issue is to render your portlets inside of an iframe. You achieve this by setting the renderPortletInIFrame attribute to true or auto. A value of true always uses an iframe. A value of auto causes Oracle WebCenter Framework to detect whether the portlet contains a form element and, if it does, then render it within an iframe.

For example, the Upload Portlet that comes with the preconfigured OC4J's WSRP Sample Producer requires an `iframe` to run correctly. If you do not place it in an `iframe` by setting `renderPortletInIFrame` to `true` or `auto`, then the portlet does not work.

Note: If you render a portlet within an `iframe`, then manipulating `window.location` may give unexpected results. If your portlet uses `window.location`, then you should ensure that your JavaScript is robust enough to handle the case where the portlet renders itself inside of an `iframe`.

Alternatively, you could rewrite your portlet such that you eliminate the form elements in the portlet's mark-up, but this approach is cumbersome and may not be feasible in all cases.

4.3.4 Copying Portlets

When you copy portlets, the portlets and their copies must reside within the same application. For example, you can copy a portlet from one page in an application to another page in the same application, from one place on a page to another place on the same page, or from one project to another project within the same application. The copies are references to the same portlet instance. This means customizations or personalizations made to any instance of the portlet (original or copy) affect all the other instances.

Copying a portlet is more than a matter of copying and pasting the portlet view tag. It involves copying portlet-related entries from the application page's source. It may also involve copying portlet-related entries from the page definition file as well as removing duplicate portlet binding information or creating a new method in the copied portlet's binding bean.

When a portlet is copied, the target page must be an Oracle ADF Faces page. Any preexisting code on the target page must reflect that. This is quite easy to accomplish. When Oracle JDeveloper creates a new JSF page, it contains pure JSF tags. The first time you drop an Oracle ADF Faces component onto the page, tags are automatically updated to be Oracle ADF Faces tags. For example, an `<html>` tag becomes `<afh:html>`, `<head>` and `<title="title">` tags become `<afh:head title="title">`, and so on. Therefore, a simple way to ensure the conversion of the target page to an Oracle ADF Faces page is to place any Oracle ADF Faces component on the target page. This will perform any required code conversion for you automatically.

This section describes how to copy portlets from one application page to another as well as how to copy a portlet from one part of a page to another part of the same page. It includes the following subsections:

- Section 4.3.4.1, "Copying and Placing a Portlet on the Same Page"
- Section 4.3.4.2, "Copying Portlets from One Application Page to Another"

4.3.4.1 Copying and Placing a Portlet on the Same Page

Because all of the page's resources are available to both portlet instances when you copy a portlet to the same page, there is no need to copy portlet-related information from the page's Page Definition file. It is just a matter of copying and pasting the portlet's view tag, and assigning a unique identifier to the copy.

To copy and place a portlet on the same page, perform the following steps:

1. In Oracle JDeveloper, go to the Source view of the page that hosts the portlet to be copied.
2. Copy the portlet tag (Example 4–5).

Example 4–5 Code Fragment to be Copied When Copying a Portlet

```
<f:view>
   <afh:html binding="#{backing_portlet_page.html1}" id="html1">
      <afh:head title="portlet_page" binding="#{backing_portlet_page.head1}"
         id="head1">
         <meta http-equiv="Content-Type"
            content="text/html;charset=windows-1252"/>
      </afh:head>
      <afh:body binding="#{backing_portlet_page.body1}" id="body1">
         <h:form binding="#{backing_portlet_page.form1}" id="form1">
            <adfp:portlet value="#{bindings.portlet1}"
               portletType="/oracle/adf/portlet/
               pdksampleproducer_1153245807295/applicationPortlets/
               Portlet2_82d49b79_010c_1000_8006_82235ffc4e2b"
               binding="#{backing_portlet_page.portlet1}"
               id="portlet1"
               isCustomModesAvailable="true"/>
         </h:form>
      </afh:body>
   </afh:html>
</f:view>
```

3. Paste the copied code fragment into the page's Source view.
4. Provide a unique value for the copy's `ID` attribute (Example 4–6).

Example 4–6 Changing the Portlet ID

```
<adfp:portlet value="#{bindings.portlet1}"
   portletType="/oracle/adf/portlet/
   pdksampleproducer_1153245807295/applicationPortlets/
   Portlet2_82d49b79_010c_1000_8006_82235ffc4e2b"
   binding="#{backing_portlet_page.portlet1}"
   id="portlet2"
   isCustomModesAvailable="true"/>
```

> **Note:** On a given page, each portlet must have a unique ID.

5. In the page source, if the copied portlet's `adfp:portlet` tag has a binding attribute, for example:

   ```
   binding="#{backing_untitled2.portlet1}"
   ```

   Then either remove this binding, or create a new method in the binding bean by opening the managed bean class for this managed bean and defining the new method in the `faces-config.xml` file.

   For example, if portlet1 is copied, and the pasted copy becomes portet2, in the `faces-config.xml` file, as shown in Example 4–7.

Example 4–7 Creating a New Method for a Managed Bean in faces-config.xml

```
.
private PortletBase portlet2;
public void setPortlet2(PortletBase portet2)  {
     this.portlet2 = portlet2;
}
.
public PortletBase getPortlet2()  {
     return portlet2;
}
```

4.3.4.2 Copying Portlets from One Application Page to Another

When you copy a portlet from one page to another in an application, portlet-related code must also be copied from the source page's Page Definition file. This section describes the steps related to both copying from one application page to another and from one application project to another.

To copy a portlet from one application page or project to another, perform the following steps:

1. In Oracle JDeveloper, go to the Source view of the page that hosts the portlet to be copied.
2. Copy the portlet tag (Example 4–8).

   If the target page does not contain Oracle ADF Faces components, then make sure the container objects—that is, any tags the portlet tag is nested in—use Oracle ADF tags:

Example 4–8 Source Page Code Fragment to Be Copied When Copying a Portlet

```
<f:view>
   <afh:html binding="#{backing_portlet_page.html1}" id="html1">
      <afh:head title="portlet_page" binding="#{backing_portlet_page.head1}"
         id="head1">
         <meta http-equiv="Content-Type"
            content="text/html;charset=windows-1252"/>
      </afh:head>
      <afh:body binding="#{backing_portlet_page.body1}" id="body1">
         <h:form binding="#{backing_portlet_page.form1}" id="form1">
            <adfp:portlet value="#{bindings.portlet1}"
               portletType="/oracle/adf/portlet/
               pdksampleproducer_1153245807295/applicationPortlets/
               Portlet2_82d49b79_010c_1000_8006_82235ffc4e2b"
               binding="#{backing_portlet_page.portlet1}"
               id="portlet1"
               isCustomModesAvailable="true"/>
         </h:form>
      </afh:body>
   </afh:html>
</f:view>
```

> **Note:** Portlets can reside only on Oracle ADF Faces pages. See the introductory paragraph to this procedure for more information.

3. Go to the application page to which to copy the portlet (the target page).
4. Paste the copied code into the target page's Source view.

5. Go to the page definition file of the page from which the portlet was copied (the source page).

   Right-click the source page, and select **Go to Page Definition** from the context menu.

6. Copy the portlet binding from the source page's page definition file (Example 4–9).

Example 4–9 Code Fragment to Be Copied From a Page Definition File

```
<portlet id="portlet1"
      portletInstance="/oracle/adf/portlet/
      pdksampleproducer_1153245807295/applicationPortlets/Portlet2_82d49b79_
      010c_1000_8006_82235ffc4e2b"
      class="oracle.adf.model.portlet.binding.PortletBinding"
      xmlns="http://xmlns.oracle.com/portlet/bindings"/>
```

> **Note:** When the portlet being copied includes parameters, be sure to include the copied portlet's portlet parameters as well as the page variables linked to the portlet parameters in the copy.

7. Go the to page definition file of the target page.

   Create a page definition file if necessary. Do this by right-clicking the target page and selecting **Go to Page Definition**. You will be prompted to create a page definition file if none exists.

8. Paste the portlet binding you copied from the source (as well as relevant portlet parameters and the page variables associated with those parameters).

4.3.5 Deleting Portlets from Application Pages

When you delete a portlet from an application page, if the portlet had parameters, then you should also delete page variables associated with those parameters from the application page's Page Definition file.

To delete a portlet from a page and related page variables from a Page Definition file, perform the following steps:

1. In the Applications Navigator, navigate to the relevant application page (`jspx` file), and open it:

```
Applications
   <ApplicationName>
      <ProjectName>
         WEB Content
           <ApplicationPage>.jspx
```

2. In Design view, right-click the portlet to delete and select **Delete** from the context menu.

   This deletes the portlet from the page and the portlet binding from the page's Page Definition file.

3. If the portlet included variables, then right-click the `jspx` file in the Editor and select **Go to Page Definition** from the context menu.

   The page definition file opens in the Editor pane.

4. Locate the page variables associated with the deleted portlet, and delete them from the page definition file.

   For example, if portlet1 is deleted you would delete the highlighted variables in Example 4–10:

Example 4–10 Deleting Portlet-Related Page Variables from a Page Definition File

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
      version="10.1.3.38.97" id="untitled1PageDef"
      Package="project1.pageDefs">
   <parameters/>
   <executables>
      <variableIterator id="variables">
            <variable Name="portlet1_param1" Type="java.lang.Object"/>
            <variable Name="portlet1_param2" Type="java.lang.Object"/>
            <variable Name="portlet2_param1" Type="java.lang.Object"/>
            <variable Name="portlet2_param2" Type="java.lang.Object"/>
      </variableIterator>
      <portlet id="portlet2" portletInstance="/oracle/adf/portlet/
            PdkPortletProducer2_1154100666247/applicationPortlets/
            Portlet1_b5e49696_010c_1000_8008_8c5707ef9c4f"
            class="oracle.adf.model.portlet.binding.PortletBinding"
            xmlns="http://xmlns.oracle.com/portlet/bindings">
         <parameters>
              <parameter name="param1" pageVariable="portlet2_param1"/>
              <parameter name="param2" pageVariable="portlet2_param2"/>
         </parameters>
      </portlet>
   </executables>
   <bindings/>
</pageDefinition>
```

5. From the File menu, select **Save All**.

4.4 Using Customizable Components

Customizable components are JSF components that provide the ability to customize any JSF page. By using these components, you can minimize or maximize, hide or show, or move any component on the page. You add and modify customizable components in Oracle JDeveloper through the Component Palette and Property Inspector.

- Section 4.4.1, "Adding Customizable Components"
- Section 4.4.2, "Dragging and Dropping Components onto a Page"
- Section 4.4.3, "Changing the Look and Feel of Customizable Components"
- Section 4.4.4, "Implementing Security for Customizable Components"

4.4.1 Adding Customizable Components

This section describes the steps involved in adding customizable components to the page, and including portlets and content within a `PanelCustomizable` or `ShowDetailFrame` component.

4.4.1.1 Adding a PanelCustomizable Component

To add a `PanelCustomizable` component to a JSF JSP (`*.jspx` file), perform the following steps:

1. In the Oracle JDeveloper Applications Navigator, right-click the `*.jspx` file, and select **Open** from the context menu.

   The `*.jspx` file is located in the Applications Navigator under:

   ```
   Applications
      <ApplicationName>
         <ProjectName>
            Web Content
   ```

2. In the Structure pane, select the component inside which you want to add the `PanelCustomizable` component.
3. In the Component Palette, select **Customizable Components Core**.
4. Click **PanelCustomizable**. The `PanelCustomizable` component is displayed in the Page Editor. In the Structure pane, this component is placed under the component you had selected in step 2.
5. In the Property Inspector, set the attributes for this component as required.

   Table 4–5 describes the attributes of a `PanelCustomizable` component.

Table 4–5 Attributes of a PanelCustomizable Component

Attribute	Value	Description
General Attributes:		
Background	`light/medium/dark` default: `light`	Working in conjunction with the skin CSS, provides a means of applying a different look and feel for this `PanelCustomizable` instance.
DisplayHeader	`true/false` default: `false`	Indicates whether the header of the `PanelCustomizable` is displayed. **Note**: One function of the header is to provide a place for users to interact with the customizations options a `PanelCustomizable` enables through the Actions menu. You can still enable these interactions when `DisplayHeader` is set to `False` by setting the `isSeededInteractionAvailable` property to `True`. This will cause the Actions menu, usually displayed on the header, to appear when the mouse moves over the component.
ExpansionMode	`maximized/minimized/normal` default: `normal`	The default state of the `PanelCustomizable`. In the minimize mode, only the header is displayed. In the maximize mode, the component occupies the entire space allocated to its root parent (whichever occurs highest in the hierarchy). The root parent could be either a `ShowDetailFrame` or a `PanelCustomizable` component.

Table 4–5 (Cont.) Attributes of a PanelCustomizable Component

Attribute	Value	Description
Height	A number expressed in pixels (px) or as a percentage (%) of the available area. For example: ■ `500px` ■ `50%`	Specifies the height the `PanelCustomizable` should have. This attribute setting is optional. If no value is specified for this attribute, then the layout wraps to the height of the child component. **Note**: If you have specified `DisplayHeader` to be `false`, then it is recommended that you specify a fixed height for the component, for example `500px`. If you do not set the `Height` attribute, then you may not be able to access the toolbar containing the Actions menu when all child components are hidden.

Icon	Enter the URI to an image. For example:
	`icon="coffee.png"`
	The URI provided in this example is stored at the document root; therefore, a full path is not required. An image that is not stored at the document root requires a full path, for example:
	`icon="c:\portal\images\box_b.gif"`
	Or:
	`icon="http://source-pc/images/accessability.gif"`
		If you decide to add an icon on the header of the `PanelCustomizable` component, then this specifies the path where the image for the icon is stored.

Id	Enter a text string to use as the `PanelCustomizable`'s unique identifier. For example: `id="weather"`	The unique identifier for the component on the page.
Layout	`horizontal/vertical` default: `vertical`	Specifies whether the children of the `PanelCustomizable` must be laid out vertically or horizontally. If you specify `vertical`, then the child components are displayed one below the other and can be moved either up or down within the layout. If you choose `horizontal`, then the child components are displayed adjacent to each other and can be moved either to the left or right within the layout.
Text	Enter a text string to use as the `PanelCustomizable`'s header title. For example: `text="Forecast for the Day"`	A title for the `PanelCustomizable` component.

Table 4–5 (Cont.) Attributes of a PanelCustomizable Component

Attribute	Value	Description
Width	A number expressed in pixels (px) or as a percentage (%) of the available area. For example: ■ `500px` ■ `50%`	Specifies the width the `PanelCustomizable` should have. **Note**: If you have specified `DisplayHeader` to be `false`, then it is recommended that you specify a fixed width for the component, for example `500px`. If you do not set the `Width` attribute, then you may not be able to access the toolbar containing the Actions menu when all child components are hidden.
DisclosureListener	Specify a method reference of type javax.faces.el.MethodBinding.	A method reference to a disclosure listener.
DisplayScrollbar	`true/false/auto` default: `auto`	Specifies whether scrollbars should be rendered for the `PanelCustomizable` content area always (`true`), never (`false`), or only when the `PanelCustomizable` content is larger that specified width and height (`auto`).
ShortDesc	Enter a text string.	A short description of this component.
Core Attributes:		
ContentInlineStyle	The name of a CSS style.	The CSS style to apply to the `PanelCustomizable` content area. Manually enter any style in compliance with CSS version 2.0 or later.
InlineStyle	The name of a CSS style.	The CSS style to apply to the whole `PanelCustomizable`. Manually enter any style in compliance with CSS version 2.0 or later, or expand this node to specify style elements.
Rendered	`true/false` default: `true`	Specifies whether the component will be rendered or not.
StyleClass	The name of a CSS style class.	The CSS style class for this component.
Actions Attributes:		
IsEditable	`true/false` default: `false`	Specify whether an Edit command is rendered on the Actions menu for editing the child component. **Note**: If you specify `true`, then a corresponding `editAction` facet must also be specified.
IsHelpAvailable	`true/false` default: `false`	Specifies whether a Help command is rendered on the Actions menu for accessing Help on the child component. **Note**: If you specify true, then a corresponding `helpAction` facet must also be specified.
IsMaximizable	`true/false` default: `true`	Renders a Maximize command on the `PanelCustomizable's` Actions menu so that the child component can take advantage of the entire `PanelCustomizable` area for display.
IsMinimizable	`true/false` default: `true`	Renders a Minimize icon on the `PanelCustomizable` header that collapses and restores the `PanelCustomizable`.

Table 4–5 (Cont.) Attributes of a PanelCustomizable Component

Attribute	Value	Description
IsMovable	`true/false` default: `true`	Renders a Move command on the `PanelCustomizable` Actions menu.
IsSeededInteraction Available	`true/false` default: `false`	Makes the `PanelCustomizable`'s seeded interactions, Maximize, Minimize, and Move, available for exposure to users. **Note**: Display of each seeded interaction is controlled by its own attribute setting.
IsShowContentEnabled	`true/false` default: `true`	Makes available a control in the Actions menu to users enabling them to show and hide the children of the `PanelCustomizable`.

The Bind option available when setting these attributes enables you to bind a `PanelCustomizable` instance to a managed bean property.

6. Save your work.

> **Note:** To wrap an existing component in a `PanelCustomizable` component, right-click the component in the Oracle JDeveloper Structure window, select **Surround With** from the context menu. Select **Customizable Components Core** from the list in the Surround With dialog box. Select `PanelCustomizable` from the list of components.

4.4.1.2 Adding a ShowDetailFrame Component

To add a `ShowDetailFrame` component to a JSF JSP (`*.jspx` file), perform the following steps:

1. In the Oracle JDeveloper Applications Navigator, right-click the `*.jspx` file, and select **Open** from the context menu.

   The `*.jspx` file is located in the Applications Navigator under:

   ```
   Applications
      <ApplicationName>
         <ProjectName>
            Web Content
   ```

2. In the Structure pane, select the component inside which you want to add the `ShowDetailFrame` component.
3. In the Component Palette, select **Customizable Components Core**.
4. Click **ShowDetailFrame**. The `ShowDetailFrame` component is displayed in the Page Editor. In the Structure pane, this component is placed under the component you had selected in step 2.
5. In the Property Inspector, set the attribute values for this component as required.

   Table 4–6 describes the attributes of a `ShowDetailFrame` component.

Table 4–6 Attributes of a ShowDetailFrame Component

Attribute	Type	Description
General Attributes:		
Background	`light/medium/dark` default: `light`	Working in conjunction with the skin CSS, provides a means of applying a different look and feel for this `ShowDetailFrame` instance.
DisplayHeader	`true/false` default: `true`	Indicates whether the header of the `ShowDetailFrame` is displayed. **Note**: If you have exposed some actions on the component, and if the header display is turned off, then a toolbar is displayed when you move the mouse over the component area. The toolbar contains a drop down icon, which displays a menu of available options. This toolbar will appear only if there are actions available on the component. The toolbar display is also affected by the `isSeededInteractionAvailable` attribute. As the default value for `isSeededInteractionAvailable` is `false`, the toolbar is not displayed. It can be displayed by setting `isSeededInteractionAvailable` to `true`.
ExpansionMode	`maximized/minimized/normal` default: `normal`	The default state of the `ShowDetailFrame`. In the minimize mode, only the header is displayed. In the maximize mode, the component occupies the entire space allocated to its root parent, which occurs highest in the hierarchy. The root parent could be either a `ShowDetailFrame` or a `PanelCustomizable` component.

Icon	Enter the URI to an image. For example:
	`icon="coffee.png"`
	The URI provided in this example is stored at the document root; therefore, a full path is not required. An image that is not stored at the document root requires a full path, for example:
	`icon="c:\portal\images\box_b.gif"`
	Or:
	`icon="http://source-pc/images/accessability.gif"`
		If you decide to add an icon on the header of the `ShowDetailFrame` component, then this specifies the path where the image for the icon is stored.

Table 4–6 (Cont.) Attributes of a ShowDetailFrame Component

Attribute	Type	Description
Id	Enter a text string to use as the ShowDetailFrame's unique identifier. For example: `id="weather"`	A unique identifier for the component on the page.
Text	Enter a text string to use as the ShowDetailFrame's header title. For example: `text="Forecast for the Day"`	A title for the `ShowDetailFrame` component.
AttributeChangeListener	Specify a method reference of type javax.faces.el.MethodBinding.	A method reference to an attribute change listener. An event is fired when an attribute value changes.
DisclosureListener	Specify a method reference of type javax.faces.el.MethodBinding.	A method reference to a disclosure listener. A disclosure event is fired when the disclosure state changes.
ShortDesc	Enter a text string.	A short description of this component.
Core Attributes:		
ContentInlineStyle	The name of a CSS style.	The CSS style to apply to the `ShowDetailFrame` content area. Manually enter any style in compliance with CSS version 2.0 or later.
InlineStyle	The name of a CSS style.	The CSS style to apply to the whole `ShowDetailFrame`. Manually enter any style in compliance with CSS version 2.0 or later, or expand this node to specify style elements. Use the CCC styles in this attribute to set a width and height for the `ShowDetailFrame`, or to enable scrollbars on the `ShowDetailFrame`.
Rendered	`true/false` default: `true`	Specifies whether the component will be rendered or not.
StyleClass	The name of a CSS style class.	The CSS style class for this component.
Actions Attributes:		
IsEditable	`true/false` default: `false`	Specify whether an Edit command is rendered on the menu for editing the child component. **Note**: If you specify `true`, then a corresponding `editAction` facet must also be specified.
IsHelpAvailable	`true/false` default: `false`	Specify whether a Help command is rendered on the menu for accessing help on the child component. **Note**: If you specify `true`, then a corresponding `helpAction` facet must also be specified.

Table 4–6 (Cont.) Attributes of a ShowDetailFrame Component

Attribute	Type	Description
IsMaximizable	`true/false` default: `true`	For `ShowDetailFrame` components placed inside a `PanelCustomizable` component, renders a Maximize command on the `ShowDetailFrame`'s Actions menu so that the child component can take advantage of the entire `PanelCustomizable` area for display.
IsMinimizable	`true/false` default: `true`	Renders a Minimize icon on the `ShowDetailFrame` header that collapses and restores the `ShowDetailFrame`.
IsMovable	`true/false` default: `true`	For `ShowDetailFrame` components placed inside a `PanelCustomizable` component, renders a Move command on the `ShowDetailFrame` Actions menu.
IsSeededInteractionAvailable	`true/false` default: `true`	Makes the `ShowDetailFrame`'s seeded interactions, Maximize, Minimize, and Move, available for exposure to users. **Note**: Display of each seeded interaction is controlled by its own attribute setting.

The Bind option available when setting these attributes enables you to bind a `ShowDetailFrame` instance to a managed bean property.

6. Save your work.

> **Note:** To wrap an existing component in a `ShowDetailFrame` component, right-click the component in the Oracle JDeveloper Structure window, select **Surround With** from the context menu. Select **Customizable Components Core** from the list in the Surround With dialog box. Select `ShowDetailFrame` from the list of components.

4.4.1.3 Adding ShowDetailFrame Facets

Use `ShowDetailFrame` facets to define and display custom actions on the `ShowDetailFrame` component. Table 4–7 describes the facets that provide additional hooks to display custom actions supported by the `ShowDetailFrame` component.

Table 4–7 ShowDetailFrame Facets

Facet	Description
editAction	Custom edit action facet.
helpAction	Custom help action facet.
titleBarAction	Used if an action is to be associated with title of the `ShowDetailFrame` component.
additionalActions	Used if some additional actions are to be added to the list of actions available in the `ShowDetailFrame` component.

Oracle JDeveloper displays all facets available to the `ShowDetailFrame` component in the Structure window. However, only those that contain UI components appear activated.

To add a `ShowDetailFrame` facet, perform the following steps:

1. Right-click a `ShowDetailFrame` component in the Structure window, and select **Facets - Show Detail Frame**.
2. Click the arrow to the right of this option.
3. From the list of supported facets, select the facet you want to add.

   The `f:facet` element for that facet is inserted in the page.

> **Note:** A checkmark next to a facet name means the `f:facet` element for that facet is already inserted in the page, but it may or may not contain a child component.

4.4.2 Dragging and Dropping Components onto a Page

The Component Palette in the Oracle JDeveloper user interface lists all the available component tag libraries. You can select a library and drag and drop components from within that library onto a `*.jspx` page.

See the *Oracle Application Development Framework Developer's Guide* for more information about dragging and dropping components onto a page.

4.4.3 Changing the Look and Feel of Customizable Components

You can change the look and feel of all customizable components by changing the Customizable Components selector CSS definition in the skin CSS file. Different Style selectors and Icon selectors are available for skinning customizable components. See Chapter 9, "Defining and Applying Styles to Core Customizable Components" for the list of available selectors, information about Oracle ADF Faces skins, creating custom skins, and configuring an application to use a skin.

4.4.4 Implementing Security for Customizable Components

You can use `PanelCustomizable` and `ShowDetailFrame` components to perform various actions on child components, such as minimize, maximize, move, and so on. Actions on customizable components are not secured by default. It is advisable not to expose all actions to all types of users, and therefore, recommended that you define security for these components appropriately.

To implement security for actions on customizable components at various levels, see Section 10.9.2, "Customizable Components Security".

4.5 Contextually Linking Components

One way to make your WebCenter application more interactive is by linking related components such that their contents are synchronized based upon the context. For example, suppose you have two stock portlets on a page, one provides data about a stock's price while the other provides headline news items for a stock. Both portlets are based upon the stock ticker symbol, hence it would make sense that, when the ticker symbol is changed in the stock price portlet, the stock headlines portlet picks up that change and refreshes itself with headlines pertaining to the same ticker symbol.

You can implement this kind of synchronization through parameters. Table 4–8 summarizes the types of components you may tie together with this type of contextual behavior.

Table 4–8 Components You Can Synchronize

Component Initiating Parameter	Component Reading Parameter Value	Comments
Oracle ADF Faces components	Oracle ADF Faces components	For more information about linking Faces components together, see *Oracle Application Development Framework Developer's Guide*.
	WSRP portlets	Faces components, such as options or lists, can be used to change the displayed content of portlets.
	PDK-Java portlets	Faces components, such as options or lists, can be used to change the displayed content of portlets.
WSRP 2.0 portlets[1]	Oracle ADF Faces components	Standards-based portlets can pass parameters to Faces components.
	WSRP 2.0 portlets	You can pass parameters between standards-based portlets.
	PDK-Java portlets	You can pass parameters from a standards-based portlet to a PDK-Java-based portlet.
PDK-Java portlets	Oracle ADF Faces components	PDK-Java portlets can pass parameters to Faces components.
	WSRP portlets	You can pass parameters from a PDK-Java portlet to a standards-based portlet.
	PDK-Java portlets	You can pass parameters between PDK-Java-based portlets.
Oracle ADF Faces page variable/parameter	Oracle ADF Faces components	You can set parameters at the page level and have them picked up by Faces components on the page.
	WSRP portlets	Standards-based portlets can pick up page parameters.
	PDK-Java portlets	PDK-Java portlets can pick up page parameters.

[1] Note that support for navigational parameters exists only in WSRP 2.0. WSRP 1.0 does not have support for navigational parameters.

4.5.1 Linking Portlets to Pages

If you define page variables, then portlets on that page can then read the page variables at run time and change their behavior accordingly. By having page variables that can be linked to portlet parameters, you enable any portlets on the page to adapt to the context. For example, the page could contain a customer identifier variable. Any related portlets on the page could use this value to determine what they ought to display. A customer details portlet could take this value and determine which customer's information to display.

Figure 4–3 shows a portlet that gets a customer identifier number (320) from a page parameter.

Figure 4–3 Example of Portlet with Page Parameter

How to Link Portlets to Pages

If you defined public parameters when creating your portlet, then page variables are created for them when you drop the portlet onto a page.

To see how parameters are implemented when you drop a portlet onto a page, do the following:

> **Note:** The scenario that follows uses OmniPortlet, but the basic principles regarding parameters and page variables would be true for any portlet.

1. Open the page by double-clicking it in the Applications Navigator. The page opens for editing and you should see its structure in the Structure pane.
2. Add an OmniPortlet portlet to the page. See Section 4.3.2, "Adding Portlets to a Page". Note that the OmniPortlet uses a PDK-Java producer and must be registered with the application. See Section 4.3.1.2, "Registering PDK-Java Portlet Producers" for more information about registering PDK-Java producers.
3. Right-click anywhere in the tree displayed in the Structure pane, and choose **Go to Page Definition** in the context menu. The page definition tree should now appear in the Structure pane and the page definition XML should open for editing.
4. In your page definition XML, you should see the five default OmniPortlet page variables inside the `<variableIterator>` tag. You should then see references to the page variables from inside of the `<portlet>` tag. Example 4–11 shows a sample page definition after an OmniPortlet was first dropped onto the page.

Example 4–11 PageDef.xml File with OmniPortlet

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="10.1.3.39.0" id="SRDemoDashboardPageDef"
                Package="view.pageDefs">
  <parameters/>
  <executables>
    <variableIterator id="variables">
      <variable Name="portlet1_Param1" Type="java.lang.Object"/>
      <variable Name="portlet1_Param2" Type="java.lang.Object"/>
      <variable Name="portlet1_Param3" Type="java.lang.Object"/>
      <variable Name="portlet1_Param4" Type="java.lang.Object"/>
      <variable Name="portlet1_Param5" Type="java.lang.Object"/>
    </variableIterator>
    <portlet id="portlet1"
      portletInstance="/oracle/adf/portlet/OmniPortletProducer_1154018261057/
        ap/Portlet100_b0da57c9_010c_1000_8003_82235f50a408"
```

```
      class="oracle.adf.model.portlet.binding.PortletBinding"
            xmlns="http://xmlns.oracle.com/portlet/bindings">
      <parameters>
        <parameter name="Param1" pageVariable="portlet1_Param1"/>
        <parameter name="Param2" pageVariable="portlet1_Param2"/>
        <parameter name="Param3" pageVariable="portlet1_Param3"/>
        <parameter name="Param4" pageVariable="portlet1_Param4"/>
        <parameter name="Param5" pageVariable="portlet1_Param5"/>
      </parameters>
    </portlet>
  </executables>
</pageDefinition>
```

5. In order for the page to take a parameter value through its URL, you must add a page level parameter. In the Structure pane, right-click the **parameters** node and select **Insert inside parameters**, **parameter** from the context menu. The Insert Parameter dialog box is displayed.
6. Enter an **ID** of `custID` and a **value** of `${param.customerID}`. Click **OK**. You should now see something similar to the following in the XML of your page definition.

   ```
   <parameters>
       <parameter id="custID" value="${param.customerID}"/>
   </parameters>
   ```

7. Right-click your page in the Applications Navigator and choose **Run** from the context menu.
8. When the page appears, click **Define** in OmniPortlet.
9. Choose **SQL** and click **Next**.
10. In the Statement area, enter the following SELECT statement. Note the reference to `Param1` in the WHERE clause.

    ```
    select * from USERS
    where USER_ID = ##Param1##
    ```

11. In the Connection section, choose or create a connection to a database that contains the SRDemo schema.
12. In Portlet Parameters, enter `320` as the **Default Value** for `Param1` and select **Personalizable**.
13. Click **Next**.
14. You do not need a filter in this case, so click **Next**.
15. In **Header Text**, enter:

    ```
    Select a customer from the Most Active Customers List to view the customer
    details.
    ```

16. Choose **HTML** for **Layout Style** and click **Next**.
17. Delete the default HTML in the **Non-Repeating Heading Section**.
18. Replace the default HTML in the Repeating Section with the following:

    ```
    <TABLE BORDER='0' WIDTH="100%">
    <TR CLASS='PortletText1'>
      <TD CLASS='PortletHeading1'>ID</TD>
      <TD>##USER_ID##</TD>
    </TR>
    ```

```
<TR CLASS='PortletText1'>
  <TD CLASS='PortletHeading1'>Email</TD>
  <TD>##EMAIL##</TD>
</TR>
<TR CLASS='PortletText1'>
  <TD CLASS='PortletHeading1'>First Name</TD>
  <TD>##FIRST_NAME##</TD>
</TR>
<TR CLASS='PortletText1'>
  <TD CLASS='PortletHeading1'>Last Name</TD>
  <TD>##LAST_NAME##</TD>
</TR>
<TR CLASS='PortletText1'>
  <TD CLASS='PortletHeading1'>Street</TD>
  <TD>##STREET_ADDRESS##</TD>
</TR>
<TR CLASS='PortletText1'>
  <TD CLASS='PortletHeading1'>City</TD>
  <TD>##CITY##</TD>
</TR>
<TR CLASS='PortletText1'>
  <TD CLASS='PortletHeading1'>State</TD>
  <TD>##STATE_PROVINCE##</TD>
</TR>
```

19. Leave the default **Non-Repeating Footer Section** as is and click **Finish**.
20. Return to Oracle JDeveloper and click the page variable, `portlet1_Param1`, in the Structure pane. Its properties should now be displayed in the Property Inspector.
21. In the Property Inspector, click next to the **DefaultValue** property. A button for editing the value of the property should appear. Click this Edit button.
22. In the DefaultValue dialog box, expand the **ADF Bindings** node and then the **bindings** node beneath it. You should see a binding for `portlet1_Param1`.
23. You can move variables into the Expression using the right arrow (>) and apply operators using the buttons. The expression you create should look something like the following:

    ```
    ${(bindings.custID == null || bindings.custID == '') ? 320 : bindings.custID }
    ```

24. Click **OK**. You should now see something similar to Example 4–12 in your page definition XML.

Example 4–12 <variable>

```
<executables>
  <variableIterator id="variables">
    <variable Name="portlet1_Param1" Type="java.lang.Object"
        DefaultValue="${(bindings.custID == null ||
        bindings.custID == '') ? 320 : bindings.custID }"/>
  </variableIterator>
...
```

25. Now you must set the value of the parameter to test your portlet on the page. You can set the value through a page URL. For example:

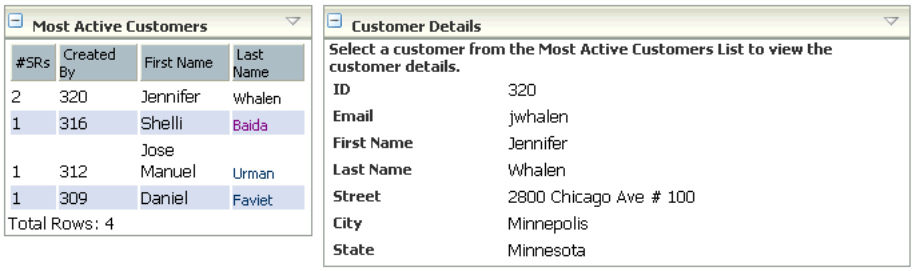
    ```
    http://pc1.com:8988/SRDemo/faces/app/management/SRDashboard.jspx?customerID=321
    ```

The passing of this URL causes the page to refresh with a value for `customerID` of 321. The OmniPortlet on the page should be refreshed with data that reflects the changed value.

4.5.2 Linking Portlets

In many cases, it's useful to tie portlets together such that when a parameter in one portlet changes, it causes the other portlet to refresh with a new value as well. For example, Figure 4–4 illustrates two portlets. When a user clicks on the Last Name from the Most Active Customers portlet on the left, that customer's ID is provided to the Customer Details portlet on the right, which in turn refreshes with that customer's detailed data.

Figure 4–4 Example of Portlet to Portlet Communication



How to Link Portlets

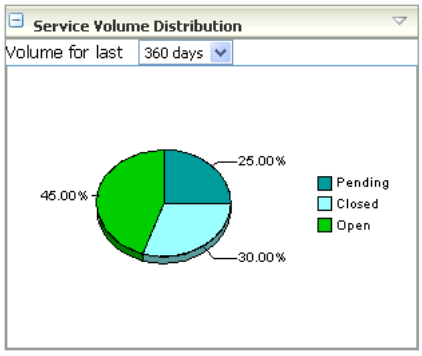
Linking portlets works in much the same fashion as linking a portlet to a page. The example in "How to Link Portlets to Pages" illustrates how a value is passed from a page parameter in the URL to a page variable, and then the page variable's value is passed into a portlet's parameter. You could easily have multiple portlets on the page that read the page variable's value in this way. Furthermore, because the page parameter can be changed from the page URL, a portlet could set the value of the page parameter and in turn the page variable, which would then affect the other portlets on the page.

4.5.3 Linking Faces Component to Portlets

A very powerful pairing of components is a Faces component with a portlet. This kind of synchronization comes in handy if you envision an interactive Faces component, such as a list or option, that relates to one or more portlets on a page. For example, suppose you have some sort of a dashboard page and you want users to be able to pick from a predefined set of choices to determine what the portlets on the page display. The Faces component (list or option selection) could pass a value to one or more portlets on the page. The portlets in turn could use this value to determine what content to display.

In Figure 4–5, the list called *Volume for last* is a Faces component, and the pie chart is generated by OmniPortlet. The value from the list is passed to OmniPortlet, which in turn takes the new value and refreshes itself with the updated chart.

Figure 4–5 Example of Faces Component to Portlet Communication



How to Link Faces Components to Portlets

To link your Faces component to a portlet, do the following:

1. Open the page by double-clicking it in the Applications Navigator. The page opens for editing and you should see its structure in the Structure pane.
2. Add the desired Faces component to the page using the Oracle JDeveloper user interface gestures. Example 4–13 illustrates a `selectOneChoice` created through the user interface within a `ShowDetailFrame` within a `PanelCustomizable`. The `selectOneChoice` provides a list called `dayPicker` from which a user can select the span of time (in days) over which an OmniPortlet (`portlet1`) pie chart will graph data. Note that in this example, the get and set functionality of the day picker was automatically created in the backing class. Setting `autoSubmit` to `true` forces the value of the variable to change whenever an item is selected from the list. Notice also how the OmniPortlet specification, which appears just below the `dayPicker`, references `dayPicker`.

Example 4–13 selectOneChoice

```
<cust:panelCustomizable text="Service Requests Volume"
                        displayHeader="false"
                        expansionMode="normal" isMovable="true"
                        isSeededInteractionAvailable="true"
                        id="panelCustomizable2" layout="horizontal">
  <cust:showDetailFrame id="showDetailFrame1"
                        text="Service Volume Distribution"
                        displayHeader="true"
                        isSeededInteractionAvailable="true">
    <af:selectOneChoice id="dayPicker" label="Volume for last"
      value="360"
      binding="#{backing_app_management_SRDashboard.SelectOneChoice}"
       autoSubmit="true">
       <af:selectItem label="1 day" value="1"/>
       <af:selectItem label="2 days" value="2"/>
       <af:selectItem label="3 days" value="3"/>
       <af:selectItem label="5 days" value="5"/>
       <af:selectItem label="10 days" value="10"/>
       <af:selectItem label="30 days" value="30"/>
       <af:selectItem label="60 days" value="60"/>
       <af:selectItem label="90 days" value="90"/>
       <af:selectItem label="180 days" value="180"/>
       <af:selectItem label="360 days" value="360"/>
    </af:selectOneChoice>
    <adfp:portlet value="#{bindings.portlet1}"
       portletType="/oracle/adf/portlet/OmniPortlet_Producer/applicationPortlets/
```

```
        Portlet100_f20eef2c_010a_1000_8003_a9fe020295ca"
      partialTriggers="dayPicker"
      displayScrollBar="False" displayHeader="False"
      isMinimizable="True"
      renderPortletInIFrame="False"/>
  </cust:showDetailFrame>
  <adfp:portlet value="#{bindings.portlet3}"
     portletType="/oracle/adf/portlet/OmniPortlet_Producer/applicationPortlets/
       Portlet100_f20f676b_010a_1000_8005_a9fe020295ca"
     renderPortletInIFrame="False"/>
  <adfp:portlet value="#{bindings.portlet2}"
     portletType="/oracle/adf/portlet/OmniPortlet_Producer/applicationPortlets/
       Portlet100_f20f50cc_010a_1000_8004_a9fe020295ca"
     renderPortletInIFrame="False"/>
</cust:panelCustomizable>
```

3. Once you have set up the Faces component and referenced it from the desired portlet on the page, you must add the parameter to the page definition. Right-click the JSP root in the Structure pane and choose **Go to Page Definition** in the context menu. The page definition XML opens for editing.

4. In the `<executables>` section, you must specify the parameter to be passed from `dayPicker` to `portlet1`. Example 4–14 illustrates the default code in the page definition.

Example 4–14 portlet1 Parameters

```
<executables>
<variableIterator id="variables">
   <variable Name="OmniPortlet2_1_Param1" Type="java.lang.Object"
   DefaultValue="<parameter name="Param1" value="${(backing_app_management_
   SRDashboard.dayPicker.value == null) ? 360 : backing_app_management_
   SRDashboard.dayPicker.value}"/>
   <variable Name="OmniPortlet2_1_Param2" Type="java.lang.Object"/>
   <variable Name="OmniPortlet2_1_Param3" Type="java.lang.Object"/>
   <variable Name="OmniPortlet2_1_Param4" Type="java.lang.Object"/>
   <variable Name="OmniPortlet2_1_Param5" Type="java.lang.Object"/>
</variableIterator>
```

5 Integrating Content

This chapter describes the procedures to configure data controls based on Java Content Repository (JCR), such as Oracle Content Database (Oracle Content DB), Oracle Application Server Portal (OracleAS Portal), IBM Lotus Domino, Microsoft SharePoint, EMC Documentum, and other content systems such as Stellent Content Server and Business Intelligence (BI) Publisher. This chapter also discusses, through examples, how to use these data controls to integrate and publish decentralized content in your WebCenter application. Read the following sections to understand how you can use the content integration capabilities of your WebCenter application:

5.1 Introduction to Content Integration Capabilities of Oracle WebCenter Suite

The content integration capabilities of Oracle WebCenter Suite enable you to integrate decentralized content located across multiple JCR 1.0 (JSR 170) Level 1-compliant repositories, JCR adapters, JCR APIs and other content systems, such as Stellent Content Server and BI Publisher.

Section 5.2, "Configuring Content Data Controls for JCR Adapters" provides an overview of content data controls that are based on JCR 1.0 repositories, such as OracleAS Portal, Oracle Content DB, IBM Lotus Domino, Microsoft SharePoint, EMC Documentum, and discusses the procedures to configure these data controls. Section 5.3, "Using JCR Data Controls: Examples" contains examples to show how to use JCR data controls.

Stellent Content Server is a scalable, secure, and centralized Web-based repository that enables you to manage all phases of the content life cycle: from creation and approval to publishing, searching, expiration, and archival or disposition. Stellent Content Server lets everyone in your organization contribute content using desktop applications. It enables you to efficiently manage business content through its rich library services. You can access Stellent Content Server securely through a Web browser. See Section 5.4, "Configuring Data Controls Based on Stellent Content Server"

for procedures to configure Stellent-based data controls and examples on how to use these data controls.

Oracle BI Publisher offers efficient and scalable reporting solutions for complex, distributed environments. Oracle BI Publisher enables you to assimilate data from multiple data sources into a single output document. It lets you generate and deliver information in suitable formats for different business groups. Oracle BI Publisher report formats can be designed using widely-used tools, such as Microsoft Word or Adobe Acrobat. The BI Publisher reports can be delivered through printer, e-mail, fax, or WebDav. See Section 5.6, "Integrating Oracle Business Intelligence Publisher" for procedures to integrate and store BI Publisher reports in WebCenter application.

5.2 Configuring Content Data Controls for JCR Adapters

The JCR data controls enable you to connect to and read from the file system, OracleAS Portal, Oracle Content DB, Microsoft SharePoint, and EMC Documentum as well as JCR 1.0 repositories. The JCR data controls enable you to publish content from any JCR 1.0 repository.

The JCR data controls discussed in the following sections provide you with easy-to-use methods that you can drag and drop onto JSF JSP pages to publish content as URLs, files, and folders:

- Section 5.2.1, "Understanding Content Data Controls"
- Section 5.2.2, "Configuring a Content Data Control Based on the File System Adapter"
- Section 5.2.3, "Configuring a Content Data Control Based on the OracleAS Portal Adapter"
- Section 5.2.4, "Configuring a Content Data Control Based on the Oracle Content DB Adapter"
- Section 5.2.5, "Configuring a Content Data Control Based on Oracle Content DB Version 10.2"
- Section 5.2.6, "Configuring a Content Data Control Based on Oracle WebCenter Adapter for IBM Lotus Domino"
- Section 5.2.7, "Configuring a Content Data Control Based on Oracle WebCenter Adapter for Microsoft SharePoint"
- Section 5.2.8, "Configuring a Content Data Control Based on Oracle WebCenter Adapter for EMC Documentum"
- Section 5.2.9, "Editing Content Data Controls"
- Section 5.2.10, "Applying Oracle ADF Security on JCR Data Controls"

> **See Also:** Section 12.2.8, "Deploying a Content Integration Application" to learn how to deploy content integration applications

5.2.1 Understanding Content Data Controls

A data control is a container for all the data objects, collections, methods, and operations used to create User Interface (UI) components within your WebCenter application. In your WebCenter application, you can create content repository data controls that connect to different repositories through the following adapters:

- File System: The File System adapter is used to add content located on your operating system's file system to JSF JSP pages.

Note: The File System adapter is intended for development purposes only.

- OracleAS Portal: The OracleAS Portal adapter is used to integrate content from a content repository located in the Oracle Application Server Portal schema.
- Oracle Content DB: The Oracle Content DB adapter is used to integrate content from a content repository located in a database and managed by Oracle Content DB.
- Oracle WebCenter Adapter for IBM Lotus Domino: This adapter is used to integrate content from IBM Lotus Domino repository.
- Oracle WebCenter Adapter for Microsoft SharePoint: This adapter is used to integrate content from Microsoft SharePoint 2003 repository.
- Oracle WebCenter Adapter for EMC Documentum: This adapter is used to integrate content from EMC Documentum repository.

Each type of data control contains common methods and parameters to publish content as links, tables, files, and folders, and to add search and advanced search capabilities for your content. While the methods are similar across all types of data controls, the method attributes are of the following types:

- Default attributes: The default attributes are common across File System, OracleAS Portal, Oracle Content DB, JCR Lotus Domino, JCR SharePoint, and JCR Documentum data controls.
- Extended attributes: The extended attributes are created with OracleAS Portal, Oracle Content DB, JCR Lotus Domino, JCR SharePoint, and JCR Documentum data controls and are unique to each.
- Custom attributes: The custom attributes are requirement-specific and can be added easily.

Content Data Control Methods, Parameters, and Default Attributes

The following sections describe methods, parameters, and default attributes that are common across File System, OracleAS Portal, Oracle Content DB, JCR Lotus Domino, JCR SharePoint, and JCR Documentum data controls:

- The search Method
- The advancedSearch Method
- The getURI Method
- The getItems Method
- The getAttributes Method

The search Method

The `search` method enables you to create a simple search by name pattern or keyword.

Table 5–1 describes the parameters of the `search` method.

Table 5–1 Parameters of the search Method

Parameter	Description
`path`	Starting path of the search.

Table 5–1 (Cont.) Parameters of the search Method

Parameter	Description
`isRecursive`	Specifies whether only the specified folder (`=false`) or the whole tree starting at the specified path (`=true`) should be searched.
`keyword`	Search keyword for full text search.
`namePattern`	Pattern search on name. Use the `%` wildcard to match any number of characters and the _ wildcard to match one character.

Table 5–2 describes attributes of the `search` method.

Table 5–2 Attributes of the search Parameters

Parameter	Description
`name`	Describes the name of the returned file or folder.
`path`	Describes the location of the returned file or folder within the content repository. The path is relative to the starting folder specified during the creation of the data control.
`URI`	The direct access URL of a file or folder.
`primaryType`	Describes whether the returned object is a file or folder.

The advancedSearch Method

The `advancedSearch` method enables you to perform an advanced search by creating a set of search criteria out of any available attribute.

Table 5–3 describes the parameters of the `advancedSearch` method.

Table 5–3 Parameters of the advancedSearch Method

Parameter	Description
`path`	Starting path of the search.
`isRecursive`	Specifies whether only the specified folder (`=false`) or the whole tree starting at the specified path (`=true`) should be searched.
`keyword`	Search keyword for full text search.
`namePattern`	Pattern search on name. Use the `%` wildcard to match any number of characters and the _ wildcard to match one character.
`matchAny`	Specifies whether all predicates (`=false`) or any predicate (`=true`) should be matched.
`predicates`	A collection of `SimplePredicate` which consists of attribute, comparator, and value.
`type`	Specifies what should be returned: only files, only folders, or any object.

The `advancedSearch` method returns the attributes described in Table 5–4.

Table 5–4 Attributes of the advancedSearch Method Parameters

Parameter	Description
`name`	Describes the name of the returned file or folder.
`path`	Describes the location of the returned file or folder within the content repository. The path is relative to the starting folder specified during the creation of the data control.

Table 5–4 (Cont.) Attributes of the advancedSearch Method Parameters

Parameter	Description
URI	The direct access URL of a file or folder.
primaryType	Describes whether the returned object is a file or folder.

The getItems Method

The getItems method returns the files and folders stored starting at a particular location in the repository. This method enables you to publish content in forms, tables, and hierarchical trees. Using this method, you can also create navigation lists and buttons.

Table 5–5 describes the parameters of the getItems method.

Table 5–5 Parameters of the getItems Method

Parameter	Description
path	Defines starting point for getItems relative to the base path defined in the data control.
type	Specifies what should be returned: only files, only folders, or any object.

The getItems method returns the attributes described in Table 5–6.

Table 5–6 Parameters of the getItems Method

Parameter	Description
name	Describes the name of the returned file or folder.
path	Describes the location of the returned file or folder within the content repository. The path is relative to the starting folder specified during the creation of the data control.
URI	The direct access URL of a file or folder.
primaryType	Describes whether the returned object is a file or folder.

The getAttributes Method

The getAttributes method returns the list of attributes and their values for a given file or folder.

Table 5–7 describes the path parameter of the getAttributes method.

Table 5–7 Parameters of the getAttributes Method

Parameter	Description
path	Describes the path to the respective object.

Table 5–8 describes the attributes that the getAttributes method returns.

Table 5–8 Parameters of the getAttributes Method

Parameter	Description
name	Name of the attribute.
value	Value of the given item.

The getURI Method

The `getURI` method returns the direct access URL to a file.

Table 5–9 describes the path parameter of the `getURI` method. You can use this method to create links to content and to inline content in your page.

Table 5–9 Parameters of the getURI Method

Parameter	Description
`path`	Describes the path to the respective object.

The `getURI` method returns the `URI` attribute.

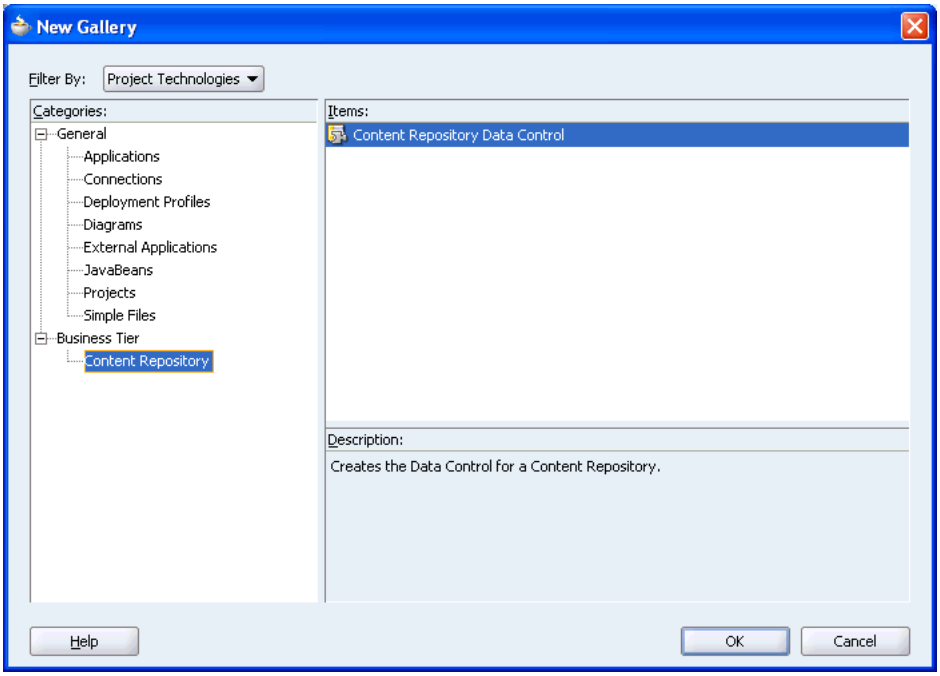
5.2.2 Configuring a Content Data Control Based on the File System Adapter

This section describes the procedure to configure a content data control based on the File System adapter that can access and publish content stored on your computer.

To configure a content data control based on the File System adapter, perform the following steps:

1. In Oracle JDeveloper, under the Applications Navigator, select your application that you created as described in Section 3.1, "Creating a WebCenter Application". Then, select **Model** and from the **File** menu select **New**. The New Gallery dialog box is displayed.
2. Under Categories, expand the **Business Tier** node and select **Content Repository**. Then, under Items, select the **Content Repository Data Control**, as shown in Figure 5–1, and click **OK**. The Create Data Control dialog box is displayed.

Figure 5–1 New Gallery Dialog Box



3. In the Create Data Control dialog box, click **Next** to skip the Welcome page.
4. On Step 1, enter a name for the data control, for example, `SRFileSystem`, and then click **Next**.

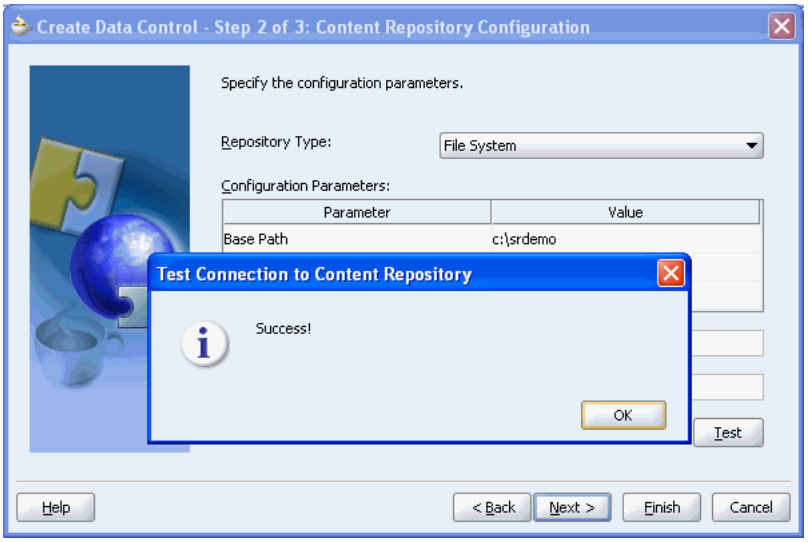
Note: When you work on a UNIX system, the File System adapter inherits the case-sensitive file name characteristic of UNIX systems. So, on UNIX systems, you must ensure that references to files follow the same case as that used in the original file names. For example, suppose the `Test.html` file was created on a Microsoft Windows system. When you reference this file on a Linux system, you must ensure that you use `Test.html`, and not test.html or `TEST.html`.

5. On Step 2, select **File System** from the **Repository Type** box.
6. In the **Base Path** field, enter the path to the folder in which your content is placed, for example, `C:\SRContent`.

Note: This location will be used as the root (`"/"`) for this data control in Section 5.3, "Using JCR Data Controls: Examples".

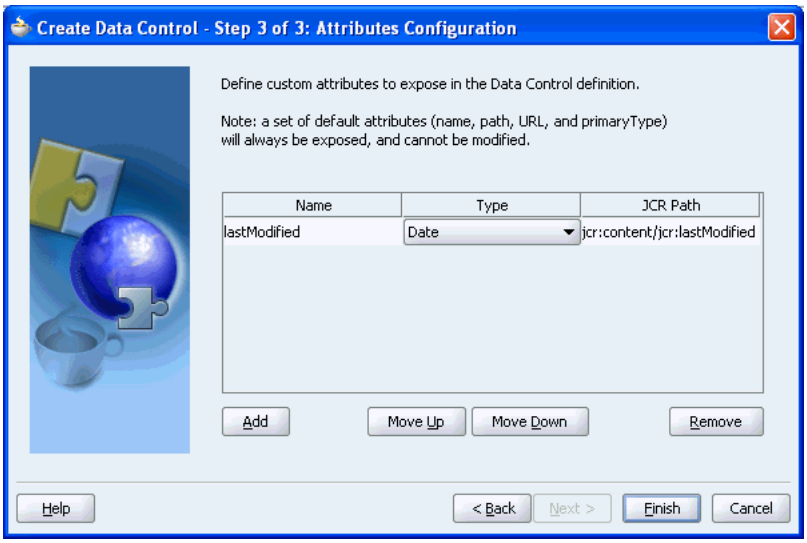
7. Click the **Test** button to check whether you have entered the connection details correctly. You should see a `Success!` message, as shown in Figure 5–2.

Figure 5–2 File System Connection



8. If you get an error message, click **OK**. Then edit the **Base Path** to specify the full path and press Enter. Then, click **Test** again. If the test is successful, click **OK** to close the message box.
9. Click **Next**. The Attributes Configuration page is displayed, as shown in Figure 5–3. This page contains the `lastModified` extended attribute, which is used to show the date when the object was last modified.

Figure 5–3 Attributes Configuration

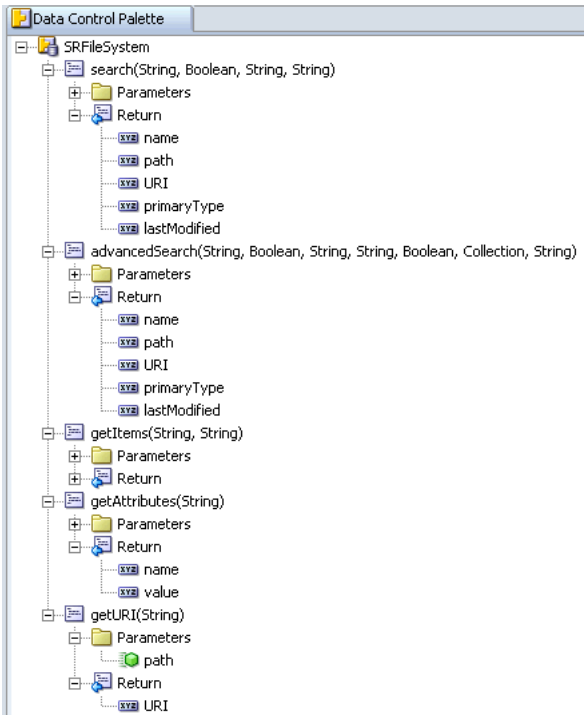


10. Click **Finish** to complete the creation of the File System data control.
11. To display the data control you created, select **Data Control Palette** from the **View** menu.

    Expand **SRFileSystem** in the Data Control Palette to see a hierarchical list of methods, parameters, and operators for the new data control, as shown in Figure 5–4.

Note: The default attributes are the same across File System, OracleAS Portal, and Oracle Content DB data controls. For more information, see Section 5.2.1, "Understanding Content Data Controls".

Figure 5–4 Data Control Palette - SRFileSystem



See Also: Section 5.3, "Using JCR Data Controls: Examples" for examples of how you can publish your content using these methods

5.2.3 Configuring a Content Data Control Based on the OracleAS Portal Adapter

This section describes the procedure to create a content data control based on the OracleAS Portal adapter that enables you to integrate content from OracleAS Portal with your JSF JSP page.

This section covers the following:

- Section 5.2.3.1, "What You Should Know About OracleAS Portal"
- Section 5.2.3.2, "Creating a Content Data Control Based on the OracleAS Portal Adapter"

5.2.3.1 What You Should Know About OracleAS Portal

The following should be considered while using the OracleAS Portal adapter:

- To use OracleAS Portal-based content data control functionality, you must install OracleAS Portal release 10.1.4. The OracleAS Portal must be up-to-date with the latest patches. Consult Oracle Application Server Release Notes for Microsoft Windows for release 10.1.3.2.0 to know the exact patch number.
- In a production environment, OracleAS Portal-based content data control supports only public users.
- The OracleAS Portal adapter uses a data source or a database connection and is dependent on connection pooling for good performance. So, to ensure high performance, the Oracle Containers for J2EE (OC4J) connection pooling must be configured in the embedded OC4J instance in your development environment, and in the OC4J instance to which you deploy your application. To configure connection pooling for the embedded OC4J, go to the **Tools** menu in Oracle

JDeveloper and select **Embedded OC4J Server Preferences**. Then, under Data Sources, select **jdev-connection-pool-***`your data source name`* and set Max Connections and Min Connections to `1`. If you do not see **jdev-connection-pool-*your data source name***, then click the **Refresh Now** button under Data Sources.

Note: In the production environment, value for minimum and maximum connection parameters should be set based on the system load.

For best performance and to minimize network latency, the portal repository database should be in the same subnet as the OC4J, with the deployed WebCenter application or as close as possible.

For the production environment, you can use Enterprise Manager to manually create a data source and connection pool. This requires the correct JNDI lookup name, which you can check in the data source configuration of the embedded OC4J instance in Oracle JDeveloper. For more information, see chapter titled "Data Sources" in *Oracle Containers for J2EE Services Guide*.

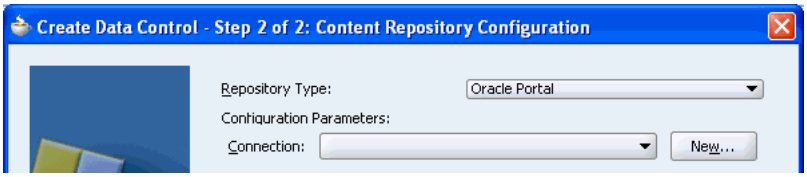
- Only `file`, `image`, `imagemap`, and `text` item types and custom types based on these item types are supported.
- The `portal:container` page type and its extensions are supported.
- Content is always exposed in the default language of the page group. For example, if there are three page groups with different default languages, then the content displays only for those default languages and not for any translations that exist.

5.2.3.2 Creating a Content Data Control Based on the OracleAS Portal Adapter

To create a content data control based on the OracleAS Portal adapter, perform the following steps:

1. In Oracle JDeveloper, go to the Applications Navigator. Under your application, select **Model**. Then, from the **File** menu select **New**. The New Gallery dialog box is displayed.
2. Under Categories, expand the **Business Tier** node and select **Content Repository**. Then, under Items, select the **Content Repository Data Control**, and click **OK**.
3. In the Create Data Control dialog box, click **Next** to skip the Welcome page.
4. On Step 1, enter a name for the data control, for example, `SRContentRepository`, and click **Next**.
5. On Step 2, select **Oracle Portal** from the **Repository Type** box, as shown in Figure 5–5.

Figure 5–5 Step 2 of Oracle Portal Configuration



6. Next to the Connection box, click **New**. The Create Database Connection - Welcome dialog box is displayed. Click **Next**.

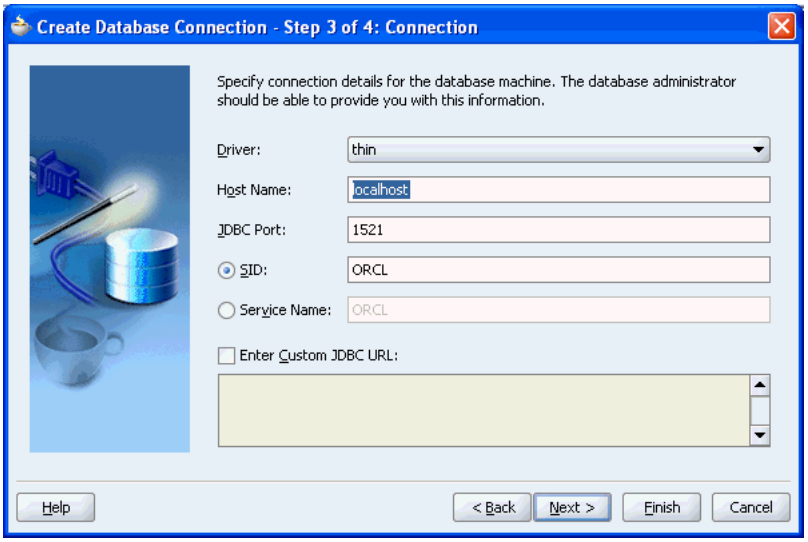
7. On Step 1, enter a name for the database connection, for example, `SRDatabase`, and click **Next**.
8. On Step 2, enter database user name and password of the OracleAS Portal schema, and click **Next**. By default, the user is PORTAL.

Table 5–10 Parameters for Creating OracleAS Portal-based Content Data Control

Parameters	Description
Driver	There are two types of drivers: thin and oci8. The thin driver can be used to connect to Oracle Database release 8*i* or later databases with TCP/IP network protocols. This driver is included in the default Oracle JDBC library for all projects. The oci8 driver is used when creating a Java application that runs against an Oracle Application Server. This is a thick driver optimized for the Oracle Database. It is a mix of Java and native code. This driver handles any database protocol (TCP, IPX, BEQ, and so on). It is recommended for applications that are run from the computer on which they are stored.
Host Name	Name of the computer running on the Oracle Application Server. Use an IP address or a host name that can be resolved by TCP/IP, for example, `myserver`. The default value is `localhost`.
JDBC Port	Value to identify the TCP/IP port.
SID	Unique system identifier of an Oracle database instance.
Service Name	The service name for an Oracle database instance.
Database URL	Database URL of the portal schema in the `jdbc` format: `jdbc:oracle:thin:@dbhost:dbport:dbsid`

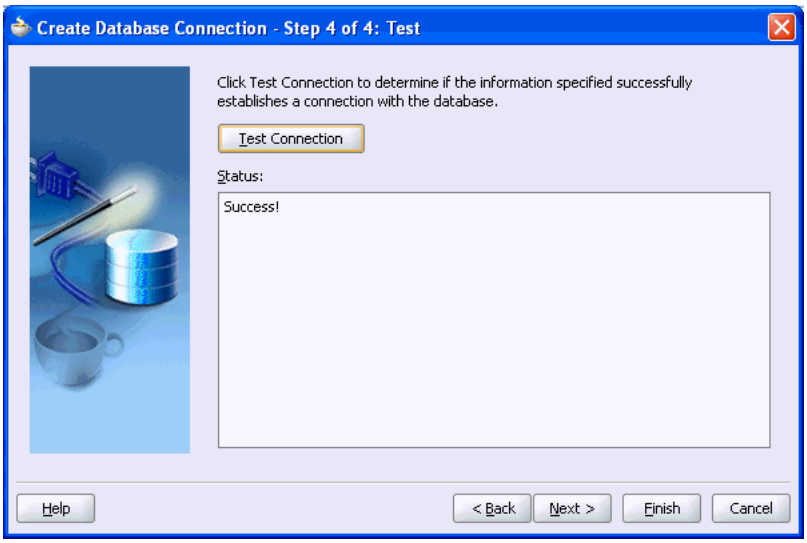
9. On Step 3, select a driver, enter the host name, JDBC port, SID, and service name, as shown in Figure 5–6. Alternatively, specify the database URL, as described in Table 5–10.

Figure 5–6 Create Database Connection - Step 3 of 4



10. On Step 4, click **Test**. If the values are valid, then the `Success!` message is displayed, as shown in Figure 5–7.

Figure 5–7 Successful Database Connection

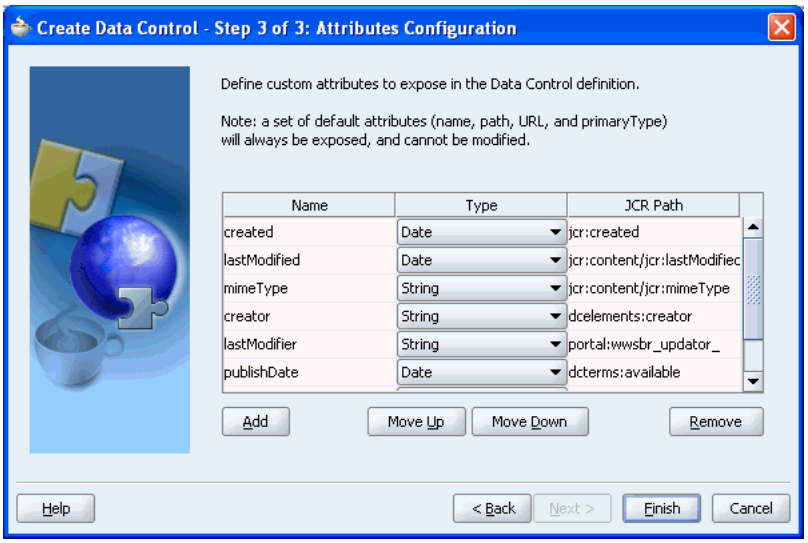


11. Click **Finish** to complete the database connection procedure. Step 2 of the Create Data Control dialog box is displayed.
12. Enter the OracleAS Portal SSO user name and password for logging in to your portal instance. The OracleAS Portal SSO user name and password are stored in the credential store. They are used by the data control to authenticate against the repository when the Use JAAS for Security checkbox is not selected.
13. Select the **Use JAAS for security** check box to transfer the identity from JAAS to the repository without passing the credentials. This option is useful for secured applications.
14. Click **Test**. The `Success!` message is displayed.

> **Note:** The test might fail if your OracleAS Portal instance is not up and running. A successful test is not necessary to create the data control.

15. Click **Next**. The Attributes Configuration page is displayed, as shown in Figure 5–8.

Figure 5–8 Attributes Configuration



The following are the extended attributes for OracleAS Portal:

- `lastModified`: Shows the date when the object was last modified.
- `mimeType`: Contains the `mimetype` of the corresponding document, or nothing for a folder.
- `creator`: Shows the ID of the user who created the object.
- `lastModifier`: Shows the ID of the user who last modified the object.
- `publishDate`: Shows the date when the object was published.
- `expirationPeriod`: Shows the date when the object will expire.
- `versionNumber`: Shows the version number of the object.

16. To add custom attributes, click **Add**. Then, enter a name for the attribute as it should appear in the Data Control Palette, select its type, and enter the JCR path.

    Note: To retrieve the JCR paths of item attributes, run the `getAttributes` method on the required items. Then, reenter the wizard to include those paths by selecting the `DataControls.dcx` file in the Applications Navigator, right-clicking the respective data control in the Structure Pane, and selecting **Edit** from the menu.

    You can remove a custom attribute by clicking **Remove** in the Attributes Configuration page.

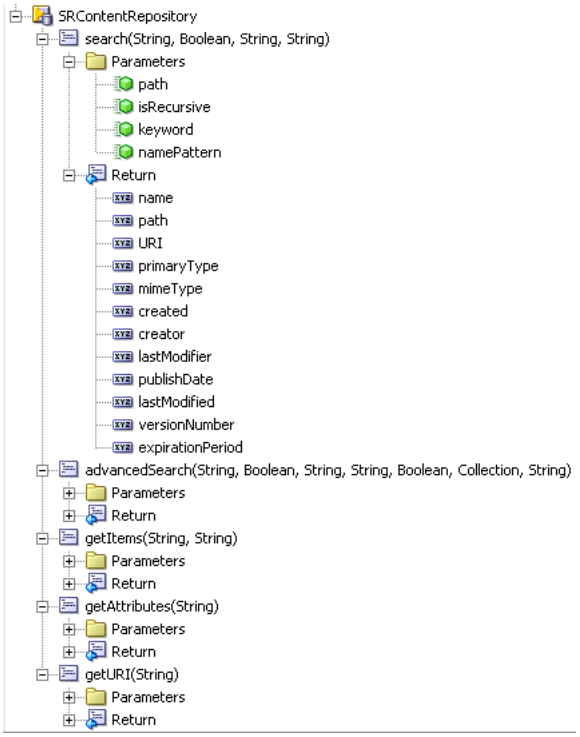
17. Click **Finish** to complete the data control configuration procedure.

18. To display the data control that you created, from the View menu, select **Data Control Palette**.

    Expand **SRContentRepository** in the Data Control Palette to see the hierarchical list of methods, parameters, and operations for the new data control, as shown in Figure 5–9.

Note: The default attributes are same across File System, OracleAS Portal, and Oracle Content DB data controls. See Section 5.2.1, "Understanding Content Data Controls" for more information.

Figure 5–9 Data Control Palette - OracleAS Portal



See Also: Section 5.3, "Using JCR Data Controls: Examples" for examples of how you can publish your content using these methods.

5.2.4 Configuring a Content Data Control Based on the Oracle Content DB Adapter

This section describes how to configure a content data control based on the Oracle Content DB adapter for Oracle Content DB release 10.1.3.2.0 that ships with Oracle Application Server 10.1.3.2.0. In addition, it also discusses procedures to enable cleartext (unencrypted plain text) authentication and WS-Security, and configure Secure Sockets layer (SSL). Enabling cleartext authentication over HTTP and configuring SSL are optional.

Note: To configure Oracle Content DB data control for Oracle Content DB version 10.2, see Section 5.2.5, "Configuring a Content Data Control Based on Oracle Content DB Version 10.2".

This section contains the following subsections:

- Section 5.2.4.1, "Configuring Keystores and Keys to Enable WS-Security"
- Section 5.2.4.2, "Creating a Content Data Control Based on the Oracle Content DB Adapter"
- Section 5.2.4.3, "Enabling Cleartext Authentication Over HTTP (Optional)"

- Section 5.2.4.4, "Configuring SSL (Optional)"

5.2.4.1 Configuring Keystores and Keys to Enable WS-Security

To enable Web Services Security (WS-Security) trusted authentication for Oracle Content DB release 10.1.3.2.0, you must configure keystores on the system that hosts Oracle Content DB as well as on the system that hosts the WebCenter application. Trusted authentication means users can securely identify themselves to other users and servers on a network without sending secret information like passwords, over the network. WS-Security establishes a trust relationship between your WebCenter application and Oracle Content DB so that your WebCenter application can pass user identity information to Oracle Content DB without knowing the user's credentials.

Preparing to Configure Keystores and Keys

The following are the prerequisites for configuring keystores and keys:

- You must configure Oracle ADF Security for your application before you set up a keystore. For more information about Oracle ADF Security, see Chapter 10, "Securing Your WebCenter Application".
- On Windows, you can use the keytool utility provided with Oracle JDeveloper under *JDEV_HOME*/jdk/bin to set up keystores. However, you can find the keytool utility in any JDK, such as the JDK shipped with Oracle Application Server.

> **Note:** The steps in the following sections generate self-certified dummy keys. It is recommended that you use real certificates in a production environment.

Configuring Keystores and Keys

To configure a keystore on the WebCenter application (client) side, perform the following steps:

1. In your development environment, go to *JDEV_HOME*/jdk/bin and open the command prompt.
2. Generate the client keystore by running the following keytool command:

   ```
   keytool -genkey -keyalg RSA -validity 5000 -alias Client private key alias
   -keystore client-keystore.jks
   -dname "cn=client" -keypass Private key password -storepass KeyStore password
   ```

3. To verify that the keys have been correctly created, run the following keytool command. This is an optional step, so you can skip it, if you want:

   ```
   keytool -list -keystore client-keystore.jks -storepass KeyStore password
   ```

4. To use the key, sign it by running the following keytool command:

   ```
   keytool -selfcert -validity 5000 -alias Client private key alias -keystore
   client-keystore.jks
   -keypass Private key password -storepass KeyStore password
   ```

5. Export the client public key by running the following keytool command:

   ```
   keytool -export -alias Client private key alias -keystore client-keystore.jks
   -file client.pubkey -keypass Private key password -storepass KeyStore password
   ```

To configure a keystore for the Oracle Content DB (server) side, perform the following steps:

1. In the same development environment, go to *JDEV_HOME*/jdk/bin and open the command prompt.
2. Generate the server keystore by running the following keytool command:

   ```
   keytool -genkey -keyalg RSA -validity 5000 -alias Server public key alias
   -keystore server-keystore.jks -dname "cn=server" -keypass Private server key
   password -storepass KeyStore password
   ```

3. To verify that the keys have been correctly created, run the following keytool command:

   ```
   keytool -list -keystore server-keystore.jks -keypass Server private key
   password -storepass KeyStore password
   ```

4. To use the key, sign it by running the following keytool command:

   ```
   keytool -selfcert -validity 5000 -alias Server public key alias -keystore
   server-keystore.jks
   -keypass Private server key password -storepass KeyStore password
   ```

5. Export the server public key to the server keystore by running the following keytool command:

   ```
   keytool -export -alias Server public key alias -keystore server-keystore.jks
   -file server.pubkey -keypass Server private key password -storepass KeyStore
   password
   ```

Verifying Signatures of Trusted Clients

To verify signatures of trusted clients, import the client public key into the server keystore, by performing the following steps:

1. In your development environment, go to *JDEV_HOME*/jdk/bin and open the command prompt.
2. To verify the signature of trusted clients, import the client's public key in to the server keystore by running the following keytool command:

   ```
   keytool -import -alias Client public key alias -file client.pubkey -keystore
   server-keystore.jks -keypass Private server key password -storepass KeyStore
   password
   ```

3. Import the server public key into the client keystore by running the following keytool command:

   ```
   keytool -import -alias Server public key alias -file server.pubkey -keystore
   client-keystore.jks -keypass Private key password -storepass KeyStore password
   ```

   When the tool prompts you if the key is self certified, you must enter Yes. Example 5–1 shows a sample output that is generated after this procedure is completed successfully.

Example 5–1 Sample Output Generated by the Keytool

```
[user@server]$ keytool -import -alias client -file client.pubkey
-keystore server-keystore.jks -keypass Server private key password -storepass
Keystore password
Owner: CN=client
Issuer: CN=client
```

```
Serial number: serial number, for example, 123a19cb
Valid from: Date, Year, and Time until: Date, Year, and Time
Certificate fingerprints:
        ...
Trust this certificate? [no]:  yes
Certificate was added to keystore.
```

Installing the Keystore on the Client

The `client-keystore.jks` file can be specified in either a relative path or an absolute path. If you use a relative path, then the keystore must be under `Web Content/WEB-INF/lib/`*`relative path`*. You specify the relative path to the keystore when you configure an Oracle Content DB-based content data control, described in Section 5.2.4.2, "Creating a Content Data Control Based on the Oracle Content DB Adapter".

If you use an absolute path, then the keystore is not deployed with your WebCenter application. Therefore, you must manually copy the keystore on the server where your WebCenter application is deployed and use the Predeployment tool to change the parameter if the absolute path you specified is not same on both computers. To learn the procedure to reconfigure keystore parameters using the Predeployment tool, see Section 12.2.2.2, "Predeploying WebCenter Applications and JCR Adapter-based Applications". It is recommended that you reconfigure the keystore and associated passwords when moving from a stage to a production environment. To learn how to reconfigure keystores and passwords, see chapter titled "Oracle Content DB Security" in *Oracle Content Database for Oracle WebCenter Suite Administrator's Guide*, and Section 12.5, "Updating Credentials in a Deployed Application".

Installing the Keystore on the Server

To install the keystore on the Oracle Content DB server, perform the following steps:

1. Copy the server keystore file `server-keystore.jks` in *`ORACLE_HOME`*`/content/settings`.
2. Set up the server keystore password that you created in Configuring Keystores and Keys, by running the `changepassword` script available in *`ORACLE_HOME`*`/content/bin/` with the `-k` flag.

   When executing the `changepassword` script for the first time, press the Enter key (without entering any value) for the old password.
3. Set up the private server key password that you created in Configuring Keystores and Keys, by running the `changepassword` script with the `-p` flag.

   When executing the `changepassword` script for the first time, press the Enter key (without entering any value) for the old password.
4. Restart Oracle Content DB.

5.2.4.2 Creating a Content Data Control Based on the Oracle Content DB Adapter

This section describes the procedure to create a content data control based on the Oracle Content DB adapter for Oracle Content DB release 10.1.3.2.0 that enables you to integrate content stored in an Oracle Database with your JSF JSP page. The Oracle Content DB data control contains methods and parameters discussed in Section 5.2.1, "Understanding Content Data Controls".

To create a content data control based on the Oracle Content DB adapter, perform the following steps:

1. In Oracle JDeveloper, go to the Applications Navigator. Under your application, select **Model**. Then, from the **File** menu select **New**. The New Gallery dialog box is displayed.
2. Under Categories, expand the **Business Tier** node and select **Content Repository**. Then, under Items, select **Content Repository Data Control**, and click **OK**.
3. In the Create Data Control dialog box, click **Next** to skip the Welcome page.
4. On Step 1, enter a name for the data control, for example, `SRContentDB`, and then click **Next**.
5. On Step 2, select **Oracle Content DB** from the **Repository Type** box.
6. Use WS-Security as the trusted authentication method and enter parameter values based on Table 5–11. You must set up keystores, as described in Section 5.2.4.1, "Configuring Keystores and Keys to Enable WS-Security", before configuring WS-Security trusted authentication method for Oracle Content DB data control.

> **Note:** The trusted client application does not need to provide a user credential because the server authenticates the trusted client application and assumes that the trusted client application has already verified the identity of the user.

Table 5–11 Parameters for Configuring WS-Security Trusted Authentication in Oracle Content DB Release 10.1.3.2.0-Based Content Data Control

Parameters	Description
Trusted Authentication Method	WS-Security is the trusted authentication method for servers release 10.1.3.2.0 and later.
KeyStore Type	The keystore type is usually `JKS` or `PKCS12`, but other formats may be supported if the appropriate provider is installed.
KeyStore Password	The password required to access the keystore. For more information, see Section 5.2.4.1, "Configuring Keystores and Keys to Enable WS-Security".
KeyStore File Location	The keystore location can be a relative path. For more information, see Installing the Keystore on the Client.
Server Public Key Alias	The public key of the server. For more information, see Section 5.2.4.1, "Configuring Keystores and Keys to Enable WS-Security". This key is used to encrypt messages sent to the server.
Server URL	URL of the server on which the data is located. It is a mandatory field. The format of server URL is: `http://`*`server`*`:`*`port`*`/content/ws`
Private Key Password	The client private key password required to retrieve the key from the keystore.
Private Key Alias	The client private key alias in the keystore, for example client, as discussed in Section 5.2.4.1, "Configuring Keystores and Keys to Enable WS-Security". The key is used to sign messages to the server. The public key corresponding to this private key must be imported in the server keystore.
Server Version Parameter	This parameter is not required if the version of the server is 10.1.3.2 or later.

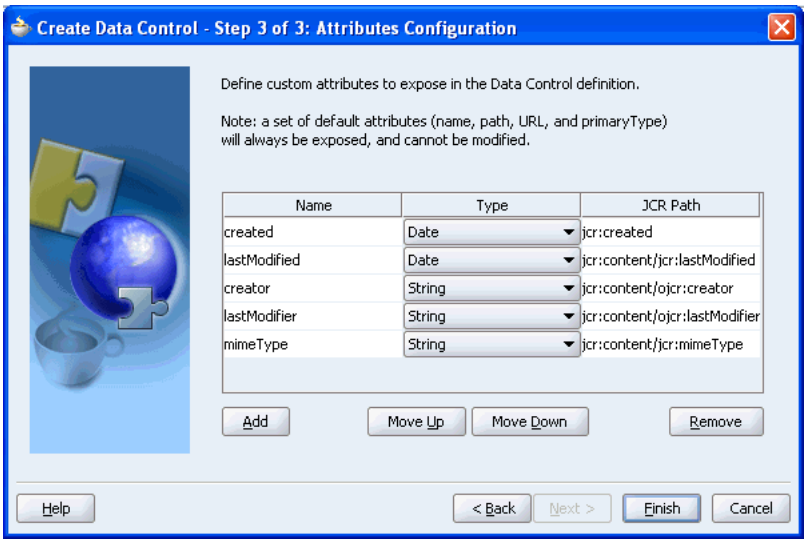
7. Select the **Use JAAS for security** check box to transfer the identity to the repository without passing the credentials and instead relying on the trust relationship between the WebCenter application and Oracle Content DB. To use JAAS with the Oracle Content DB release 10.1.3.2.0, you must enable WS-Security.

   Note: You cannot test your content data control if the Use JAAS for security check box is selected.

   The Oracle Content DB data control does not support access using lightweight user name and password.

8. Click **Next**. The Attributes Configuration page is displayed, as shown in Figure 5–10.

Figure 5–10 Attributes Configuration - Oracle Content DB



   The following are the extended attributes for Oracle Content DB:

   - `lastModified`: Shows the date when the object was last modified.
   - `creator`: Shows the ID of the user who created the object.
   - `lastModifier`: Shows the ID of the user who last modified the object.
   - `mimeType`: Contains the `mimetype` of the corresponding document, or nothing for a folder.

9. To add custom attributes, click **Add**. Then, enter a name for the attribute as it should appear in the Data Control Palette, select its type, and enter the JCR path.

   Note: To retrieve the JCR paths of item attributes, run the `getAttributes` method on the required items. Then reenter the wizard to include those paths by selecting the CPX file, right-clicking the respective data control in the Structure Pane, and selecting **Edit** from the menu.

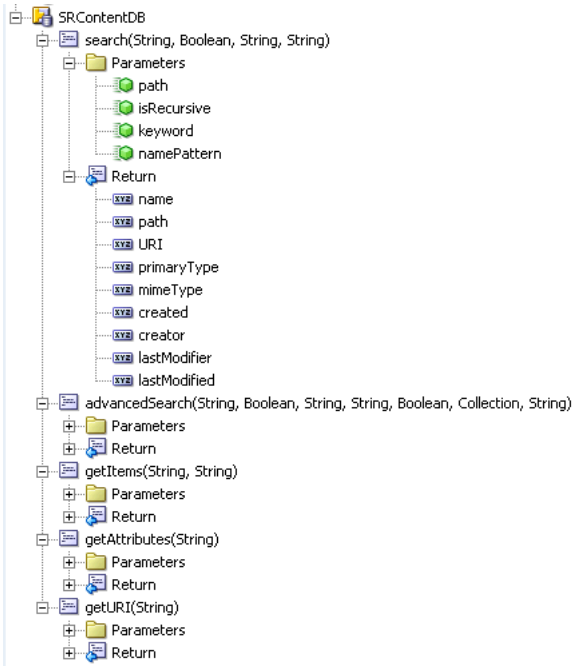
   You can remove a custom attribute by selecting the attribute and then clicking **Remove** in the Attributes Configuration page.

10. Click **Finish** to complete the data control configuration procedure.
11. To display the data control that you created, from the View menu, select **Data Control Palette**.

    Expand **SRContentDB** in the Data Control Palette to see the hierarchical list of methods, parameters, and operations for the new data control, as shown in Figure 5–11.

> **Note:** The default attributes are same across File System, OracleAS Portal, and Oracle Content DB data controls. See Section 5.2.1, "Understanding Content Data Controls" for more information.

Figure 5–11 Data Control Palette - Oracle Content DB



> **See Also:** Section 5.3, "Using JCR Data Controls: Examples" for examples of how you can publish your content using these methods.

5.2.4.3 Enabling Cleartext Authentication Over HTTP (Optional)

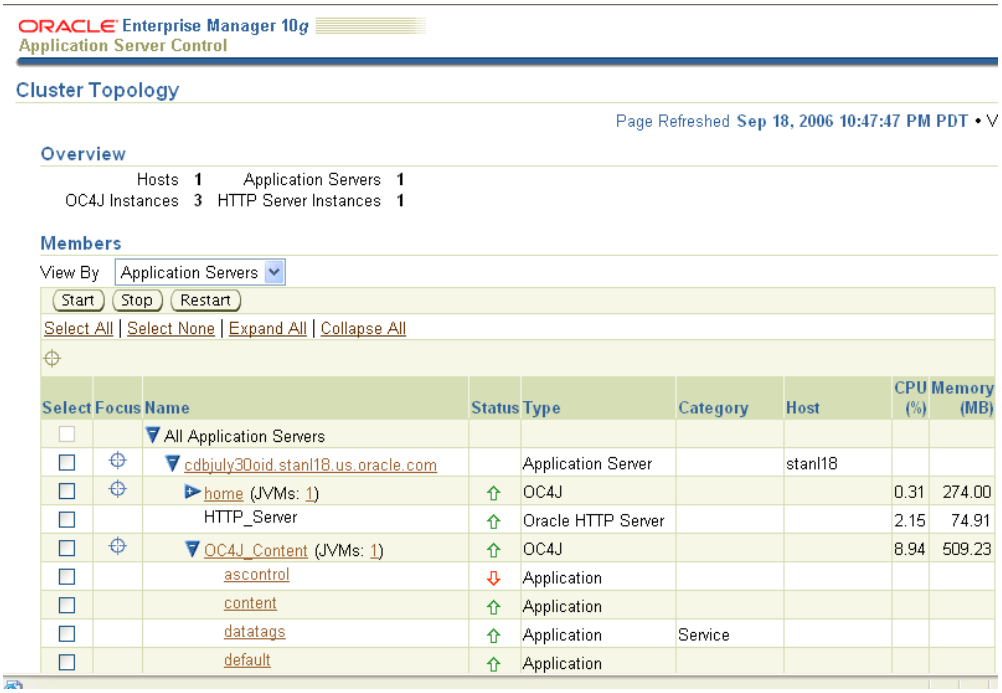
By default, the Oracle Content DB domain property `CleartextAuthenticationRequiresHttps` is set to true. If you want to access Oracle Content DB using simple credentials like user name and password over HTTP, then you must set this domain property to false. You do not have to set CleartextAuthenticationRequiresHttps to false in an environment where WS-Security is implemented over HTTP or HTTPS.

> **Caution:** If you enable cleartext authentication over HTTP, then your user name and password are transmitted in clear text. Change this domain property only in a development environment.

To enable set CleartextAuthenticationRequiresHttps to false, perform the following steps:

1. Log in to the Application Server Control Console. The Cluster Topology page is displayed, as shown in Figure 5–12.

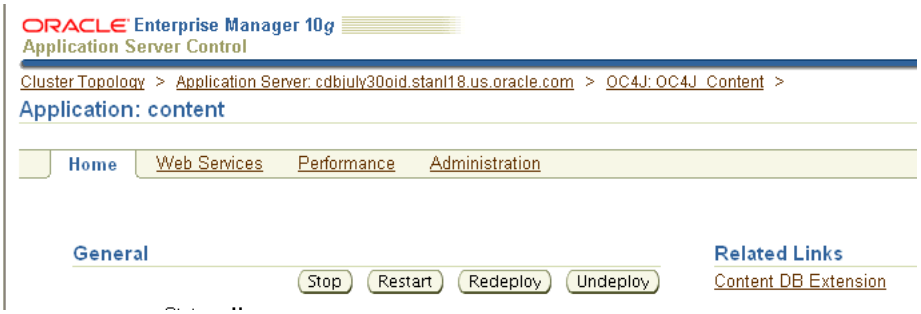
Figure 5–12 Cluster Topology Page



2. Under Members, expand any **OC4J_Content** instance, and then click **content**. The Application: content page is displayed, as shown in Figure 5–13.

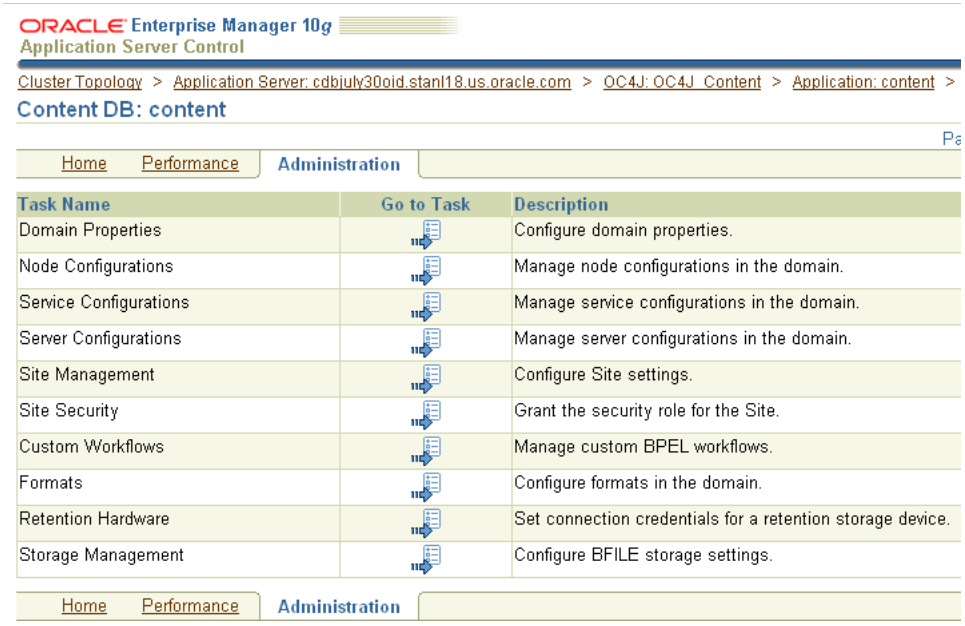
> **Note:** The OC4J_Content instance must be running when you expand it. To start the OC4J_Content instance, select it and click **Start**.

Figure 5–13 Application: content Page



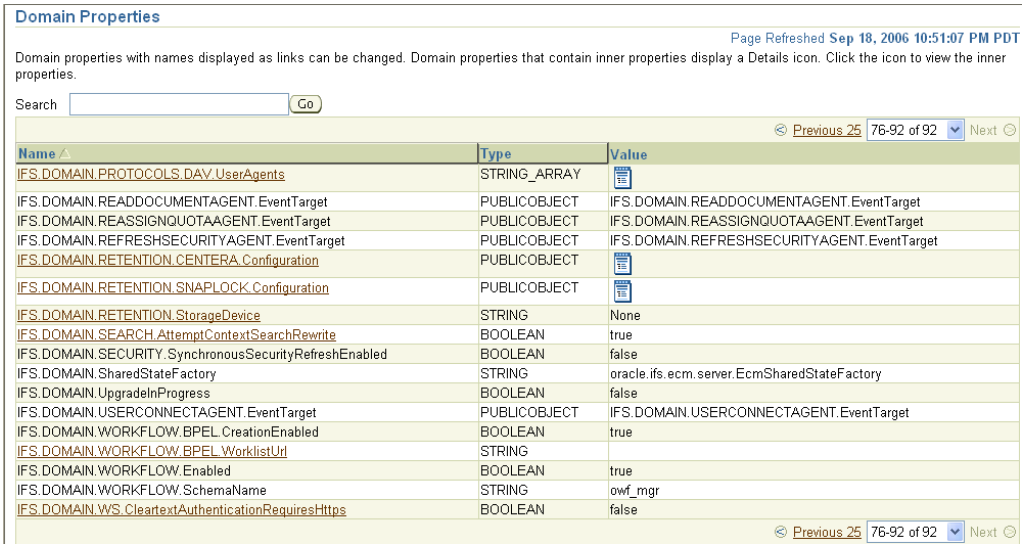
3. Under Related Links, click **Content DB Extension**, and then click the **Administration** tab. The Content DB: content page is displayed, as shown in Figure 5–14.

Figure 5–14 Content DB: content Page



4. Next to Domain Properties, click the **Go To Task** icon. The Domain Properties page is displayed.
5. To find the `IFS.DOMAIN.WS.CleartextAuthenticationRequiresHttps`, property, enter it in the Search field and click **Go**. The page containing IFS.DOMAIN.WS.CleartextAuthenticationRequiresHttps is displayed, as shown in Figure 5–15.

Figure 5–15 Domain Properties Page



6. Click **IFS.DOMAIN.WS.CleartextAuthenticationRequiresHttps**. The Edit IFS.DOMAIN.WS.CleartextAuthenticationRequiresHttps page is displayed.
7. Set the value to **false**, as shown in Figure 5–16, and click **OK**.

Figure 5–16 Edit IFS.DOMAIN.WS.CleartextAuthenticationRequiresHttps

8. Return to the Cluster Topology page and restart OC4J. If you have multiple Oracle Content DB middle tiers, then restart the OC4J_Content instance for each middle tier.

5.2.4.4 Configuring SSL (Optional)

Secure Sockets Layer (SSL) configuration is optional, but SSL must be configured if there is a need to protect the integrity and security of the data transmitted between WebCenter applications and the Oracle Content DB server. It is highly recommended if your applications and server communicate over the public Internet. However, if WebCenter applications and Oracle Content DB are on the same server, or in the same data center, then it is not necessary to configure SSL.

To configure SSL, see section titled SSL Configuration for Oracle Content DB in *Oracle Content Database for Oracle WebCenter Suite Administrator's Guide*.

> **Note:** If your Oracle Content DB instance uses HTTPS, then a Security Alert dialog box may pop up prompting you to view the security certificate and add it to the list of trusted certificates. The Security Alert dialog box is displayed only if the Oracle Content Database (Oracle Content DB) instance uses a security certificate issued by a certificate authority that is not widely accepted. See Section 10.8, "Registering Custom Certificates with the Keystore" for the steps to be performed.

5.2.5 Configuring a Content Data Control Based on Oracle Content DB Version 10.2

Oracle Content DB release 10.1.3.2.0 comes packaged with Oracle Application Server 10.1.3.2.0 by default. However, WebCenter Framework also supports Oracle Content DB version 10.2. Section 5.2.5.1, "Creating a Content Data Control Based on Oracle Content DB" describes the procedure to configure a content data control for a Oracle Content DB adapter based on Oracle Content DB version 10.2.

> **Note:** Oracle Content DB must be up and running when you create an Oracle Content DB data control.

The following sections include optional procedures:

- Section 5.2.5.2, "Enabling Cleartext Authentication Over HTTP (Optional)"
- Section 5.2.5.3, "Configuring SSL (Optional)"

5.2.5.1 Creating a Content Data Control Based on Oracle Content DB

This section describes the procedure to create an Oracle Content DB data control for Oracle Content DB version 10.2 that enables you to integrate content stored in an Oracle Database with your JSF JSP page. The Oracle Content DB data control

contains methods and parameters discussed in Section 5.2.1, "Understanding Content Data Controls".

To create an Oracle Content DB data control, perform the following steps:

1. In Oracle JDeveloper, go to the Applications Navigator. Under your application, select **Model**. Then, from the **File** menu select **New**. The New Gallery dialog box is displayed.
2. Under Categories, expand the **Business Tier** node and select **Content Repository**. Then, under Items, select **Content Repository Data Control**, and click **OK**.
3. In the Create Data Control dialog box, click **Next** to skip the Welcome page.
4. On step 1, enter a name for the data control, for example, `SRContentDB`, and then click **Next**.
5. On step 2, select **Oracle Content DB** from the **Repository Type** box.
6. Use Service-to-Service (S2S) trusted authentication method, and enter appropriate parameter values based on Table 5–12.

For information about setting up service-to-service authentication, see *Oracle Workspaces Web Services Application Developer's Guide* on the Oracle Collaboration Suite Documentation page on Oracle Technology Network (OTN) (`http://www.oracle.com/technology/documentation/collab.html`).

Table 5–12 Parameters for Configuring S2S Trusted Authentication in Content Data Control Based on Oracle Content DB ***version 10.2***

Parameters	Description
Trusted Authentication Method	S2S is the trusted authentication method for servers of version 10.2.
S2S Application Name	The trusted client application name. The format is usually LDAP distinguished name (DN).
S2S Application Password	The trusted client application password.
Server Version Parameters	The version of the server is 10.2.
Server URL	URL of the server on which the data is located. It is a mandatory field. The format of server URL is: `http://server:port/content/ws`

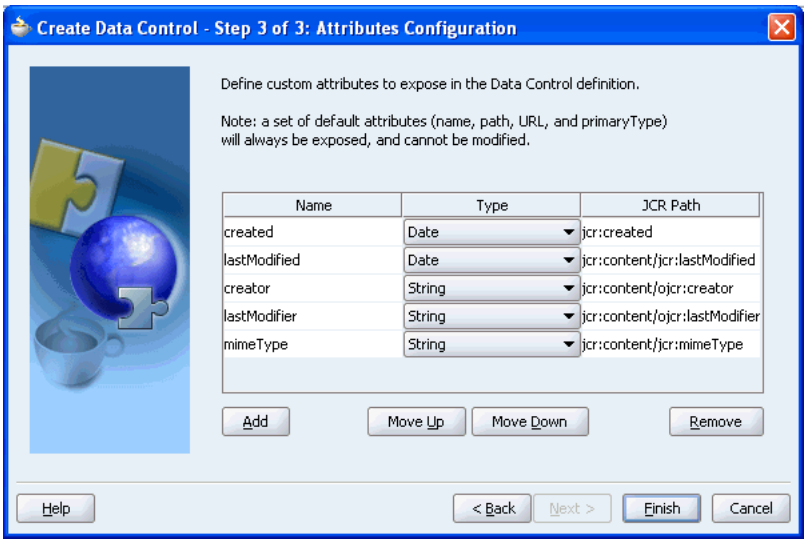
Note: The trusted client application does not need to provide a user credential because the server authenticates the trusted client application and assumes that the trusted client application has already verified the identity of the user.

7. Select the **Use JAAS for security** check box to transfer the identity to the repository without passing the credentials and instead relying on the trust relationship between the application and Oracle Content DB. To access 10.2 version of Oracle Content DB, you must set up S2S authentication.

Note: You cannot test your content data control if the Use JAAS for security check box is selected.

8. Click **Next**. The Attributes Configuration page is displayed, as shown in Figure 5–10.

Figure 5–17 Attributes Configuration - Oracle Content DB



The following are the extended attributes for Oracle Content DB:

- `lastModified`: Shows the date when the object was last modified.
- `creator`: Shows the ID of the user who created the object.
- `lastModifier`: Shows the ID of the user who last modified the object.
- `mimeType`: Contains the `mimetype` of the corresponding document, or nothing for a folder.

9. To add custom attributes, click **Add**. Then, enter a name for the attribute as it should appear in the Data Control Palette, select its type, and enter the JCR path.

> **Note:** To retrieve the JCR paths of item attributes, run the `getAttributes` method on the required items. Then, reenter the wizard to include those paths by selecting the `DataControls.dcx` file in the Applications Navigator, right-clicking the respective data control in the Structure Pane, and selecting **Edit** from the menu.

You can remove a custom attribute by clicking **Remove** in the Attributes Configuration page.

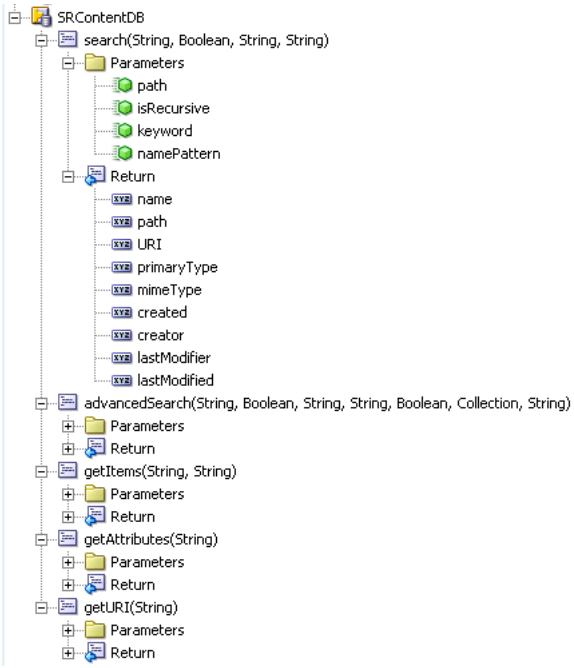
10. Click **Finish** to complete the data control configuration procedure.

11. To display the data control that you created, from the **View** menu, select **Data Control Palette**.

Expand **SRContentDB** in the Data Control Palette to see the hierarchical list of methods, parameters, and operations for the new data control, as shown in Figure 5–11.

Note: The default attributes are the same across File System, OracleAS Portal, and Oracle Content DB data controls. See Section 5.2.1, "Understanding Content Data Controls" for more information.

Figure 5–18 Data Control Palette - Oracle Content DB



See Also: Section 5.3, "Using JCR Data Controls: Examples" for examples of how you can publish your content using these methods.

5.2.5.2 Enabling Cleartext Authentication Over HTTP (Optional)

To enable cleartext authentication for Web services, see procedure in Section 5.2.4.3, "Enabling Cleartext Authentication Over HTTP (Optional)".

5.2.5.3 Configuring SSL (Optional)

To configure SSL, see procedure in Section 5.2.4.4, "Configuring SSL (Optional)".

5.2.6 Configuring a Content Data Control Based on Oracle WebCenter Adapter for IBM Lotus Domino

This section discusses the following:

- Section 5.2.6.1, "Overview of the Oracle WebCenter Adapter for IBM Lotus Domino"
- Section 5.2.6.2, "Platform Requirements for Oracle WebCenter Adapter for IBM Lotus Domino"
- Section 5.2.6.3, "What You Should Know About Oracle WebCenter Adapter for IBM Lotus Domino"
- Section 5.2.6.4, "Installing Oracle WebCenter Adapter for IBM Lotus Domino in Oracle JDeveloper"

- Section 5.2.6.5, "Installing Oracle WebCenter Adapter for IBM Lotus Domino on Oracle Application Server"
- Section 5.2.6.6, "Configuring a Data Control based on Oracle WebCenter Adapter for IBM Lotus Domino"
- Section 5.2.6.7, "Verifying the JCR Domino Adapter Library in the Model Project"

5.2.6.1 Overview of the Oracle WebCenter Adapter for IBM Lotus Domino

Oracle WebCenter adapter for Lotus Domino extracts and searches content in a IBM Lotus Domino server. The adapter accesses IBM Lotus Domino server using Java APIs. Java classes access Domino objects through CORBA by making remote (IIOP) calls. The following are the main purposes of Oracle WebCenter adapter for Lotus Domino:

- Read metadata and content of documents and views stored in a Domino database.
- Search for documents stored in a Domino database.

5.2.6.2 Platform Requirements for Oracle WebCenter Adapter for IBM Lotus Domino

Oracle WebCenter adapter for IBM Lotus Domino must be installed with a high speed network connection to the IBM Lotus Domino repository. The following are the platform requirements:

- The Oracle WebCenter adapter for IBM Lotus Domino supports IBM Lotus Domino Server version 6.5.x or 7.0.x.
- The IBM Domino database must be configured with full text indexing to support searches of its data.
- It is recommended that both client and server are deployed on the same LAN network.
- If a WAN network is used, then network throughput must be high and latency must be low to manage network traffic that the IIOP protocol generates.

  For information on network-related analysis and troubleshooting of Domino servers, see the vendor documentation for IBM Lotus Domino.

5.2.6.3 What You Should Know About Oracle WebCenter Adapter for IBM Lotus Domino

The Domino adapter provides access to the Documents and Views within a Domino database. Appendix F.1, "Node Type Definitions for the Oracle WebCenter Adapter for IBM Lotus Domino" describes how the data is mapped to JCR node types.

The adapter node type mapping reflects the structure of a Domino database. Although this mapping uses `nt:file` and `nt:folder` types, the Domino types need to be more specialized to support the Domino data. The generic data control is optimized for content repositories that follow a simple file and folder model. Therefore, it is possible that pages built using the generic data control will not achieve the desired results for a particular Domino database. A possible approach for more complex data is to build a custom data control tailored to a Domino database. A custom data control can be created from a Java bean that uses the JCR interface directly to access the required data. The generic data control can be of use in the development of a custom data control since it allows a developer to explore the structure of data within a Domino database, for example viewing the Items as a Tree, as described in Section 5.3.3, "Publishing Folder Content in a Tree" and viewing the attributes of a particular node using the `getAttributes` method.

Latest Patch Required

To use IBM Lotus Domino-based content data control functionality, Oracle JDeveloper and Oracle Application Server must be up-to-date with the latest patch. Consult Oracle Application Server Release Notes for Microsoft Windows for release 10.1.3.2.0 to know the exact patch number.

Importing the Shared Library for Oracle WebCenter Adapter for IBM Lotus Domino

If the shared library is not imported by default, then the application's `orion-application.xml` must be modified to import the Domino shared library, as described in Section 12.2.1.3, "Manually Creating and Editing the orion-application.xml File". Example 5–2 shows the `orion-application.xml` file to import the Domino shared library.

Example 5–2 orion-application.xml to Import the Domino Shared Library

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<orion-application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/orion-appli
cation-10_0.xsd">
  <library path="./adf"></library>
  <jazn location="/jazn-data.xml" provider="XML"/>
  <imported-shared-libraries>
    <import-shared-library name ="oracle.vcr.adf2domino">
  </imported-shared-libraries>
</orion-application>
```

5.2.6.4 Installing Oracle WebCenter Adapter for IBM Lotus Domino in Oracle JDeveloper

To install Oracle WebCenter adapter for IBM Lotus Domino on Oracle JDeveloper, perform the following steps:

1. Extract `/adapter/adf2domino.zip` that contains `oracle.vcr.adf2jcr.10.1.3.zip`, and `oracle.vcr.adf2domino.10.1.3.zip`.
2. Extract the shared library package `oracle.vcr.adf2jcr.10.1.3.zip` archive to *JDEV_HOME*`/jdev/extensions`. If you extracted this library package while installing Oracle WebCenter adapters for Microsoft SharePoint or EMC Documentum, then skip this step.
3. Extract the adapter package `oracle.vcr.adf2domino.10.1.3.zip` archive to *JDEV_HOME*`/jdev/extensions`.
4. Copy the `Notes.jar` archive from your IBM Lotus Notes Domino install to *JDEV_HOME*`/jdev/extensions/oracle.vcr.adf2domino.10.1.3`.
5. Restart Oracle JDeveloper to activate the Oracle WebCenter adapter for IBM Lotus Domino.

5.2.6.5 Installing Oracle WebCenter Adapter for IBM Lotus Domino on Oracle Application Server

The adapter is installed into the Oracle Application Server as a shared library. You can install Oracle WebCenter adapter for IBM Lotus Domino on Oracle Application Server by performing either of the following procedures:

- Installing Oracle WebCenter Adapter for IBM Lotus Domino on Oracle Application Server using admin_client.jar
- Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server using Application Server Control Console

Installing Oracle WebCenter Adapter for IBM Lotus Domino on Oracle Application Server using admin_client.jar

To install the adapter on Oracle Application Server, perform the following steps:

1. Extract `/adapter/adf2domino.zip` that contains `oracle.vcr.adf2jcr.10.1.3.zip`, and `oracle.vcr.adf2domino.10.1.3.zip`.
2. Extract the `oracle.vcr.adf2jcr.10.1.3.zip` archive to a temporary directory, for example, `mydomino`.
3. Extract the `oracle.vcr.adf2domino.10.1.3.zip` archive to the `mydomino` directory.
4. Copy the `Notes.jar` from your IBM Lotus Domino install into the `mydomino` directory.
5. To create the Domino shared library, run the command shown in Example 5–3 on Linux and Example 5–4 on Windows, from the `mydomino` directory.

Example 5–3 Syntax to Install Oracle WebCenter Adapter for IBM Lotus Domino on Linux

```
#In the following syntax, the format of DEPLOYER_URI_OC4J_INSTANCE should be
deployer:cluster:opmn://server.company.com:6004/home
#The opm request port may not be 6004. To find the port for an install, see
APPLICATION_SERVER_INSTALL_DIR/opmn/conf/opmn.xml configuration file.
java -jar ORACLE_HOME/j2ee/home/admin_client.jar \
DEPLOYER_URI_OC4J_INSTANCE username password \
-publishSharedLibrary -name oracle.vcr.adf2domino -version 10.1.3 \
-installCodeSources \
oracle.vcr.adf2domino.10.1.3.jar \
oracle.vcr.adf2domino.10.1.3/day-commons-ldapclient.jar \
oracle.vcr.adf2domino.10.1.3/crx-auth-ldap.jar \
oracle.vcr.adf2domino.10.1.3/crx-core.jar \
oracle.vcr.adf2domino.10.1.3/crx2domino.jar \
oracle.vcr.adf2domino.10.1.3/adf2domino.jar \
oracle.vcr.adf2domino.10.1.3/commons-pool.jar \
oracle.vcr.adf2domino.10.1.3/ehcache.jar \
oracle.vcr.adf2jcr.10.1.3/adf2crx.jar \
oracle.vcr.adf2jcr.10.1.3/crx-api.jar \
oracle.vcr.adf2jcr.10.1.3/concurrent.jar \
oracle.vcr.adf2jcr.10.1.3/crx-commons.jar \
oracle.vcr.adf2jcr.10.1.3/day-collections.jar\
oracle.vcr.adf2jcr.10.1.3/slf4j-jdk14.jar \
oracle.vcr.adf2jcr.10.1.3/lucene-core.jar \
oracle.vcr.adf2jcr.10.1.3/did.jar \
oracle.vcr.adf2jcr.10.1.3/day-commons-text.jar \
oracle.vcr.adf2jcr.10.1.3/day-commons-naming.jar \
Notes.jar \
-imports adf.oracle.domain oracle.xml apache.commons.logging
```

Example 5–4 Syntax to Install Oracle WebCenter Adapter for IBM Lotus Domino on Windows

```
java -jar ORACLE_HOME\j2ee\home\admin_client.jar
deployer:oc4j:opmn://server.company.com:6004/home oc4jadmin password
-publishSharedLibrary -name oracle.vcr.adf2domino -version 10.1.3
-installCodeSources oracle.vcr.adf2domino.10.1.3.jar
oracle.vcr.adf2domino.10.1.3.jar
oracle.vcr.adf2domino.10.1.3/day-commons-ldapclient.jar
oracle.vcr.adf2domino.10.1.3/crx-auth-ldap.jar
oracle.vcr.adf2domino.10.1.3/crx-core.jar
oracle.vcr.adf2domino.10.1.3/crx2domino.jar
oracle.vcr.adf2domino.10.1.3/adf2domino.jar
oracle.vcr.adf2domino.10.1.3/commons-pool.jar
oracle.vcr.adf2domino.10.1.3/ehcache.jar
oracle.vcr.adf2jcr.10.1.3/crx-api.jar oracle.vcr.adf2jcr.10.1.3/concurrent.jar
oracle.vcr.adf2jcr.10.1.3/crx-commons.jar
oracle.vcr.adf2jcr.10.1.3/day-collections.jar
oracle.vcr.adf2jcr.10.1.3/slf4j-jdk14.jar
oracle.vcr.adf2jcr.10.1.3/lucene-core.jar oracle.vcr.adf2jcr.10.1.3/did.jar
oracle.vcr.adf2jcr.10.1.3/day-commons-text.jar
oracle.vcr.adf2jcr.10.1.3/day-commons-naming.jar dfc.jar dfcBase.jar
```

5.2.6.6 Configuring a Data Control based on Oracle WebCenter Adapter for IBM Lotus Domino

Before configuring a data control based on Oracle WebCenter adapter for IBM Lotus Domino, you must perform the procedure in Section 5.2.6.4, "Installing Oracle WebCenter Adapter for IBM Lotus Domino in Oracle JDeveloper".

To configure the data control, perform the following steps:

1. In the Oracle JDeveloper, under Applications Navigator, select the **Model** project of your application. Then, select **New** from the **File** menu. The New Gallery is displayed.
2. Under Business Tier, select **Content Repository** and click **OK**.
3. Click **Next** to skip the Welcome page.
4. On Step 1: Data Control Name, enter a name for the new Domino adapter-based data control, and click **OK**.
5. On Step 2: Content Repository Configuration, select **JCR Domino Adapter** from the Repository Type box.
6. On Step 2: Content Repository Configuration, specify the configuration parameters as shown in Table 5–13.

Table 5–13 Configuration Parameters for the Data Control based on Oracle WebCenter Adapter for IBM Lotus Domino

Parameter	Description
Password	Password of an IBM Lotus Domino administrative user for the database.

Table 5–13 (Cont.) Configuration Parameters for the Data Control based on Oracle WebCenter Adapter for IBM Lotus Domino

Parameter	Description
Max. view entries	This is a mandatory parameter and accepts numeric value. This can be used to restrict the number of child nodes retrieved in a view. For example, for an IBM Lotus Notes email database, the inbox (view name (`$Inbox`)), may contain 1000 messages. If this parameter is set to `-1`, then all 1000 messages are retrieved corresponding to child nodes of the `/VIEWS/($Inbox)` node. If the configuration value is set to 0, then no child nodes are retrieved, and any other positive value limits the number of child nodes, for example, if the value is set to 5, then there will be only five child nodes of `/VIEWS/($Inbox)`.
Cache directory	By default, the adapter uses the current directory to cache information. Using this parameter you can specify a different directory for this cache location.
Host name	The name or IP address of the server where IBM Lotus Domino is installed.
Database name	The name of the database in the IBM Lotus Domino server, for example, `xyz.nsf`.
Username	The name of IBM Lotus Domino user. The user must have administrative rights, that is the user must have full access rights to the database specified by previous parameter. Note that the users are identified by their Lotus Domino User ID rather than Lotus Domino username.
Server name	The name of the IBM Lotus Domino server.

7. Select either the **Use JAAS for Security** check box, or supply a set of shared credentials in the Username and Password fields. If you specify shared credentials, then every connection to the Oracle WebCenter adapter for IBM Lotus Domino will use this same set of shared credentials. If you use JAAS for security, then the user identity will be used to log into the adapter. When configuring JAAS, for security reasons, it is strongly advised that the IBM Lotus Domino repository shares identity management with the application server. This is because the JCR adapter uses a trust model to permit logon to the Domino repository based on user identity alone.

> **Note:** To change shared credentials post deployment, use the Edit Data Control wizard and then redeploy the application.

8. Click **Finish** to complete the creation of the JCR data control. The data control is displayed under the Data Control Palette.

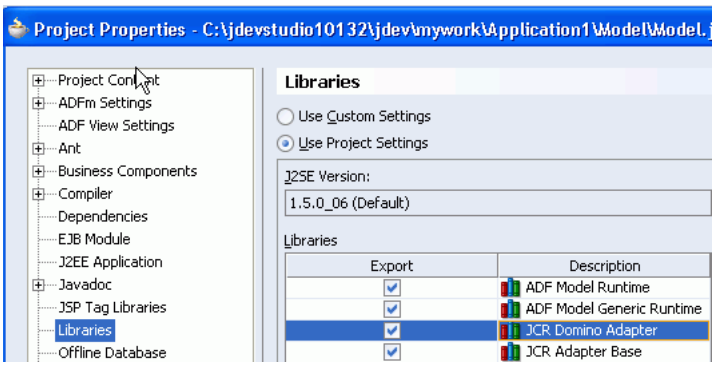
5.2.6.7 Verifying the JCR Domino Adapter Library in the Model Project

You can verify the JCR Domino adapter library after you have performed the procedure in Section 5.2.6.6, "Configuring a Data Control based on Oracle WebCenter Adapter for IBM Lotus Domino".

To verify whether the JCR Domino adapter has been added to the library, perform the following steps:

1. In the Oracle JDeveloper, right-click the **Model** project of your application under Applications Navigator, and select **Project Properties**. The Project Properties dialog box is displayed.
2. Select **Libraries**. The JCR Domino Adapter library is displayed under Libraries.

Figure 5–19 Project Properties Dialog Box



3. Click **OK** to close the Project Properties dialog box.

5.2.7 Configuring a Content Data Control Based on Oracle WebCenter Adapter for Microsoft SharePoint

This section discusses the following:

- Section 5.2.7.1, "Overview of Oracle WebCenter Adapter for Microsoft SharePoint"
- Section 5.2.7.2, "Platform Requirements"
- Section 5.2.7.3, "What You Should Know About Oracle WebCenter Adapter for Microsoft SharePoint"
- Section 5.2.7.4, "Installing Oracle WebCenter Adapter for Microsoft SharePoint in Oracle JDeveloper"
- Section 5.2.7.5, "Installing Oracle WebCenter Adapter for Microsoft SharePoint on Oracle Application Server"
- Section 5.2.7.6, "Installing Additional SharePoint Services on Microsoft SharePoint 2003 Server"
- Section 5.2.7.7, "Configuring a Data Control Based on Oracle WebCenter Adapter for Microsoft SharePoint"
- Section 5.2.7.8, "Verifying the Library of Oracle WebCenter Adapter for Microsoft SharePointin the Model Project"

5.2.7.1 Overview of Oracle WebCenter Adapter for Microsoft SharePoint

The Oracle WebCenter adapter for Microsoft SharePoint extracts and searches content within a Microsoft SharePoint 2003 repository. The adapter supports the various object types with a SharePoint 2003 repository. For information about the type support and mapping to JCR, see Section F.2, "Node Type Definitions for the Oracle WebCenter Adapter for Microsoft SharePoint".

The adapter accesses the repository using the Microsoft SharePoint SOAP interfaces, and additionally uses some JCR components installed on the SharePoint server for search and change notification support.

5.2.7.2 Platform Requirements

- The Oracle WebCenter adapter for Microsoft SharePoint requires Microsoft SharePoint Services 2003 with Microsoft SQL Server 2000.
- The Oracle WebCenter adapter for Microsoft SharePoint runs on the following:
  - Windows Server 2000, 2003
  - Red Hat Enterprise Linux Advanced Server 3 and 4
- To enable certain search features, additional configuration steps are performed in the SharePoint instance. The Full-text search needs to be enabled in the Administrative Tools. To enable search in specific language, the correct language resource package must be selected in the MS SQL server (for example, neutral). For more information, see MS SharePoint 2003 documentation.

5.2.7.3 What You Should Know About Oracle WebCenter Adapter for Microsoft SharePoint

This section covers the following:

- Latest Patch Required
- Importing the Shared Library of Oracle WebCenter Adapter for Microsoft SharePoint
- Application Listener for Change Notification
- Optimizing Performance of Oracle WebCenter Adapter for Microsoft SharePoint

Latest Patch Required

To use Microsoft SharePoint-based content data control functionality, Oracle JDeveloper and Oracle Application Server must be up-to-date with the latest patch. Consult Oracle Application Server Release Notes for Microsoft Windows for release 10.1.3.2.0 to know the exact patch number.

Importing the Shared Library of Oracle WebCenter Adapter for Microsoft SharePoint

If the shared library of Oracle WebCenter adapter for Microsoft SharePoint is not imported by default, then the application's `orion-application.xml` must be modified to import the SharePoint shared library, as described in Section 12.2.1.3, "Manually Creating and Editing the orion-application.xml File". Example 5–5 shows `orion-application.xml` file to import the SharePoint shared library.

Example 5–5 orion-application.xml to Import the SharePoint Shared Library

```
orion-application.xml:
<?xml version = '1.0' encoding = 'windows-1252'?>
<orion-application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/orion-appli
cation-10_0.xsd">
  <library path="./adf"></library>
  <jazn location="/jazn-data.xml" provider="XML"/>
  <imported-shared-libraries>
     <import-shared-library name ="oracle.vcr.adf2sharepoint">
  </imported-shared-libraries>
</orion-application>
```

Application Listener for Change Notification

The Oracle WebCenter adapter for Microsoft SharePoint includes a change listener servlet. The SharePoint Changes Service can send notifications of repository changes to this servlet. To enable notifications, the application's `web.xml` configuration file must include the servlet. Example 5–6 shows a sample `web.xml` entry for the servlet.

Example 5–6 web.xml

```
<servlet>
<servlet-name>SharepointChangeListenerServlet</servlet-name>
<servlet-class>oracle.vcr.adf2sharepoint.SharepointChangeListenerServlet</servlet-
class>
</servlet>
<servlet-mapping>
<servlet-name>SharepointChangeListenerServlet</servlet-name>
<url-pattern>/sharepoint-changes</url-pattern>
</servlet-mapping>
```

Note: The `url-pattern` must NOT be under a security constraint.

The SharePoint Changes service must also be configured to send notification to the application. See Section 5.2.7.6.2, "Installing SharePoint Changes Service on Microsoft SharePoint 2003 Server" for an explanation of the service and its configuration.

Optimizing Performance of Oracle WebCenter Adapter for Microsoft SharePoint

This section explains how you can optimize performance of the Oracle WebCenter adapter for Microsoft SharePoint on Windows and Linux platforms, as follows:

- Optimizing Performance of Oracle WebCenter Adapter for Microsoft SharePoint on Windows
- Optimizing Performance of Oracle WebCenter Adapter for Microsoft SharePoint on Linux

Optimizing Performance of Oracle WebCenter Adapter for Microsoft SharePoint on Windows

This section describes the settings to tune Windows 2000, Windows XP, and Windows 2003 operating systems to optimize the performance of the Oracle WebCenter adapter for Microsoft SharePoint.

Configure the following settings or variables according to your specific tuning needs:

- **TCPTimedWaitDelay**: The `TCPTimedWaitDelay` determines the time that must elapse before TCP/IP can release a closed connection and reuse its resources. This interval between closure and release is known as the `TIME_WAIT` state or twice the maximum segment lifetime (2MSL) state. During this time, reopening the connection to the client and server costs less than establishing a new connection. By reducing the value of this entry, TCP/IP can release closed connections faster and provide more resources for new connections. Adjust this parameter if the running application requires rapid release, the creation of new connections, or an adjustment because of a low throughput caused by multiple connections in the `TIME_WAIT` state. To set this parameter:

  1. Run the `regedit` command and access the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters` registry subkey.

2. Create a new `REG_DWORD` value named **TcpTimedWaitDelay**.
3. Set the value to decimal **30**, which is `Hex 0x0000001e`. This value sets the wait time to 30 seconds.

   The default value is `0xF0`, which sets the wait time to 240 seconds (4 minutes). The recommended value is minimum `0x1E`, which sets the wait time to 30 seconds.
4. Stop and restart the system.

- **MaxUserPort**: This determines the highest port number that TCP/IP can assign when an application requests an available user port from the system. To set this parameter:
  1. Run the `regedit` command and access the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TCPIP\Parameters` registry subkey.
  2. Create a new `REG_DWORD` value named **MaxUserPort**.
  3. Set this value to at least decimal **32768**.

     The default value is none. The recommended value is at least decimal `32768`.
  4. Stop and restart the system.

Optimizing Performance of Oracle WebCenter Adapter for Microsoft SharePoint on Linux

To optimize the performance of Oracle WebCenter adapter for Microsoft SharePoint on Linux, set the `timeout_timewait` parameter. To do so, run the following command:

```
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
```

> **Tip:** The `timeout_timewait` parameter on Linux and the `TCPTimedWaitDelay` parameter on Windows perform the same function. See Optimizing Performance of Oracle WebCenter Adapter for Microsoft SharePoint on Windows for an overview of the `TCPTimedWaitDelay` parameter.

5.2.7.4 Installing Oracle WebCenter Adapter for Microsoft SharePoint in Oracle JDeveloper

To install the Oracle WebCenter adapter for Microsoft SharePoint in Oracle JDeveloper, perform the following steps:

1. Extract `/adapter/adf2sharepoint.zip` that contains `oracle.vcr.adf2jcr.10.1.3.zip`, `oracle.vcr.adf2sharepoint.10.1.3.zip`, and `sharepoint-services.zip`.
2. Extract the shared library package `oracle.vcr.adf2jcr.10.1.3.zip` archive to *`JDEV_HOME`*`/jdev/extensions`. If you extracted this library package while installing Oracle WebCenter adapters for IBM Lotus Domino or EMC Documentum, then skip this step.
3. Extract the adapter package `oracle.vcr.adf2sharepoint.10.1.3.zip` archive to *`JDEV_HOME`*`/jdev/extensions`.
4. Restart Oracle JDeveloper to activate the Oracle WebCenter adapter for Microsoft SharePoint.

5.2.7.5 Installing Oracle WebCenter Adapter for Microsoft SharePoint on Oracle Application Server

The Oracle WebCenter adapter for Microsoft SharePoint is installed into the Oracle Application Server as a shared library. You can install the adapter by performing either of the following procedures:

- Installing Oracle WebCenter Adapter for Microsoft SharePoint on Oracle Application Server using admin_client.jar
- Installing Oracle WebCenter Adapter for Microsoft SharePoint on Oracle Application Server using Application Server Control Console

Installing Oracle WebCenter Adapter for Microsoft SharePoint on Oracle Application Server using admin_client.jar

To install the Oracle WebCenter adapter for Microsoft SharePoint on Oracle Application Server, perform the following steps:

1. Extract `/adapter/adf2sharepoint.zip` that contains `oracle.vcr.adf2jcr.10.1.3.zip`, `oracle.vcr.adf2sharepoint.10.1.3.zip`, and `sharepoint-services.zip`.
2. Extract the `oracle.vcr.adf2jcr.10.1.3.zip` archive to a temporary directory, for example, `mysharepoint`.
3. Extract the `oracle.vcr.adf2sharepoint.10.1.3.zip` archive to the `mysharepoint` directory.
4. To create the SharePoint shared library, run the command shown in Example 5–7 on Linux and Example 5–8 on Windows, from the `mysharepoint` directory:

Example 5–7 Syntax to Install the Oracle WebCenter Adapter for Microsoft SharePoint on Linux

```
#In the following syntax, the format of DEPLOYER_URI_OC4J_INSTANCE should be
deployer:cluster:opmn://server.company.com:6004/home
#The opm request port is by default 6004, but on actual install it may be
different. To find the port for an install, see APPLICATION_SERVER_INSTALL_
DIR/opmn/conf/opmn.xml configuration file.
java -jar ORACLE_HOME/j2ee/home/admin_client.jar DEPLOYER URI OC4J INSTANCE
username password -publishSharedLibrary -name\
oracle.vcr.adf2sharepoint -version 10.1.3 -installCodeSources\
oracle.vcr.adf2sharepoint.10.1.3.jar \
oracle.vcr.adf2sharepoint.10.1.3/wsdl4j.jar \
oracle.vcr.adf2sharepoint.10.1.3/axiom-api.jar \
oracle.vcr.adf2sharepoint.10.1.3/axis2-adb.jar \
oracle.vcr.adf2sharepoint.10.1.3/sharepoint-api.jar \
oracle.vcr.adf2sharepoint.10.1.3/crx-core.jar \
oracle.vcr.adf2sharepoint.10.1.3/crx2sharepoint.jar \
oracle.vcr.adf2sharepoint.10.1.3/commons-httpclient.jar \
oracle.vcr.adf2sharepoint.10.1.3/commons-lang.jar \
oracle.vcr.adf2sharepoint.10.1.3/stax-api.jar \
oracle.vcr.adf2sharepoint.10.1.3/adf2sharepoint.jar \
oracle.vcr.adf2sharepoint.10.1.3/axis2-kernel.jar \
oracle.vcr.adf2sharepoint.10.1.3/commons-codec.jar \
oracle.vcr.adf2sharepoint.10.1.3/axiom-impl.jar \
oracle.vcr.adf2sharepoint.10.1.3/sharepoint-axis-impl.jar \
oracle.vcr.adf2sharepoint.10.1.3/stax.jar \
oracle.vcr.adf2sharepoint.10.1.3/xbean.jar \
oracle.vcr.adf2sharepoint.10.1.3/policy.jar \
```

```
oracle.vcr.adf2jcr.10.1.3/adf2crx.jar \
oracle.vcr.adf2jcr.10.1.3/crx-api.jar \
oracle.vcr.adf2jcr.10.1.3/concurrent.jar \
oracle.vcr.adf2jcr.10.1.3/crx-commons.jar \
oracle.vcr.adf2jcr.10.1.3/day-collections.jar \
oracle.vcr.adf2jcr.10.1.3/slf4j-jdk14.jar \
oracle.vcr.adf2jcr.10.1.3/lucene-core.jar \
oracle.vcr.adf2jcr.10.1.3/did.jar \
oracle.vcr.adf2jcr.10.1.3/day-commons-text.jar \
oracle.vcr.adf2jcr.10.1.3/day-commons-naming.jar \
-imports adf.oracle.domain oracle.xml apache.commons.logging
```

Example 5–8 Syntax to Install the Oracle WebCenter Adapter for Microsoft SharePoint on Windows

```
java -jar ORACLE_HOME\j2ee\home\admin_client.jar
deployer:oc4j:opmn://server.company.com:6003/home oc4jadmin password
-publishSharedLibrary -name oracle.vcr.adf2sharepoint -version 10.1.3
oracle.vcr.adf2sharepoint.10.1.3.jar
oracle.vcr.adf2sharepoint.10.1.3/wsdl4j.jar
oracle.vcr.adf2sharepoint.10.1.3/axiom-api.jar
oracle.vcr.adf2sharepoint.10.1.3/axis2-adb.jar
oracle.vcr.adf2sharepoint.10.1.3/sharepoint-api.jar
oracle.vcr.adf2sharepoint.10.1.3/crx-core.jar
oracle.vcr.adf2sharepoint.10.1.3/crx2sharepoint.jar
oracle.vcr.adf2sharepoint.10.1.3/commons-httpclient.jar
oracle.vcr.adf2sharepoint.10.1.3/commons-lang.jar
oracle.vcr.adf2sharepoint.10.1.3/stax-api.jar
oracle.vcr.adf2sharepoint.10.1.3/adf2sharepoint.jar
oracle.vcr.adf2sharepoint.10.1.3/axis2-kernel.jar
oracle.vcr.adf2sharepoint.10.1.3/commons-codec.jar
oracle.vcr.adf2sharepoint.10.1.3/axiom-impl.jar
oracle.vcr.adf2sharepoint.10.1.3/sharepoint-axis-impl.jar
oracle.vcr.adf2sharepoint.10.1.3/stax.jar
oracle.vcr.adf2sharepoint.10.1.3/xbean.jar
oracle.vcr.adf2sharepoint.10.1.3/policy.jar
oracle.vcr.adf2jcr.10.1.3/adf2crx.jar
oracle.vcr.adf2jcr.10.1.3/crx-api.jar
oracle.vcr.adf2jcr.10.1.3/concurrent.jar
oracle.vcr.adf2jcr.10.1.3/crx-commons.jar
oracle.vcr.adf2jcr.10.1.3/day-collections.jar
oracle.vcr.adf2jcr.10.1.3/slf4j-jdk14.jar
oracle.vcr.adf2jcr.10.1.3/lucene-core.jar
oracle.vcr.adf2jcr.10.1.3/did.jar
oracle.vcr.adf2jcr.10.1.3/day-commons-text.jar
oracle.vcr.adf2jcr.10.1.3/day-commons-naming.jar
-imports adf.oracle.domain oracle.xml apache.commons.logging
```

The Oracle Application Server can be configured such that the adapter's shared library is imported by default by every application deployed to the server. This is done by updating the `system-application.xml` file. For information on system application, see chapter titled "Introduction to Oracle Containers for J2EE (OC4J)" of *Oracle Containers for J2EE Configuration and Administration Guide*. Such a configuration is only advisable if the contents of the SharePoint shared library will not conflict with applications deployed to the container. If the SharePoint adapter shared library is not imported by default, the application must include the import in its configuration. See Importing the Shared Library of Oracle WebCenter Adapter for Microsoft SharePoint for more information.

Installing Oracle WebCenter Adapter for Microsoft SharePoint on Oracle Application Server using Application Server Control Console

To install Oracle WebCenter adapter for Microsoft SharePoint using Application Server Control Console, see the procedure described in Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server using Application Server Control Console. See Installing Oracle WebCenter Adapter for Microsoft SharePoint on Oracle Application Server using admin_client.jar for the required JARs.

5.2.7.6 Installing Additional SharePoint Services on Microsoft SharePoint 2003 Server

In addition to installing the Oracle WebCenter adapter for Microsoft SharePoint on your Oracle JDeveloper or Oracle Application Server instance, you must install two additional services on the SharePoint server. These services support search and notification between the SharePoint server and JCR SharePoint adapter.

This section covers the following tasks:

- Section 5.2.7.6.1, "Installing Search Web Service On Microsoft SharePoint 2003 Server"
- Section 5.2.7.6.2, "Installing SharePoint Changes Service on Microsoft SharePoint 2003 Server"

5.2.7.6.1 Installing Search Web Service On Microsoft SharePoint 2003 Server Before configuring the data control based on Oracle WebCenter adapter for Microsoft SharePoint, you must install the search Web service on your Microsoft SharePoint 2003 server version 2.0.

> **Note:** See Section 5.2.7.2, "Platform Requirements" for server requirements for supporting full text and natural language search.

This section covers the following tasks:

- Installing the Search Web Service
- Configuring the SharePoint Environment
- Configuring the Search Web Service
- Executing Search SQL Scripts

Installing the Search Web Service

To install the search Web service, perform the following steps:

1. The search Web service is packaged as `sharepoint-services.zip`. Create a folder called `sharepoint-services` and extract this ZIP file. The following files are extracted:
   - `etc\Web.config`
   - `lib\search\SharepointDBSearchService.asmx`
   - `lib\search\bin\log4net.dll`
   - `lib\search\bin\SharePointSearchService.dll`
2. Create a new folder, for example, `SharePointSearchService`, on your local hard drive.

3. Copy the following search Web service files from the `sharepoint-services` folder to the `SharePointSearchService` folder. The directory structure should look like this:
   - `SharePointSearchService\SharePointDBSearchService.asmx`, the web service wrapper.
   - `SharePointSearchService\Web.config`, the web service configuration file.
   - `SharePointSearchService\bin\log4net.dll`, the logging library.
   - `SharePointSearchService\bin\SharePointSearchService.dll`, the web service library.
4. Open the **Internet Information Services (IIS) Manager** from Administrative Tools under Control Panel.
5. In the IIS Manager snap-in, open the Web site for which you would like to configure the search and add a new virtual folder, `SharePointSearchService`. To do this, perform the following steps:
   a. Right-click the Web site and select **New**, **Virtual Directory**.
   b. In the Virtual Directory Creation Wizard, specify `SharePointSearchService`, as the name for the service and click **Next**.
   c. Select the directory where you have copied the service files and click **Next**.
   d. Provide read and execute permission by selecting the **Read** and **Execute** options and click **Next** to complete the Virtual Directory Creation Wizard.

Configuring the SharePoint Environment

To configure the SharePoint environment to use the search Web service, perform the following steps:

1. Open the SharePoint Central Administration and click the **Configure virtual server settings** link in the Virtual Server Configuration section.
2. From the Virtual Server List, select the site for which you want to configure search.
3. Click the **Define managed paths** link in the Virtual Server Management section.
4. In the Add a New Path section, specify `/SharePointSearchService` in the Path field, and select the **Excluded path** option.
5. Click **OK**.
6. The new path, `/SharePointSearchService`, should now appear in the Excluded Paths section of this page.

To verify that the service was created successfully, use the following URL in your browser:

`http://<site>/SharePointSearchService/SharePointDBSearchService.asmx`

A Web service page is displayed with a list of operations supported by the Web service methods.

Configuring the Search Web Service

To configure the search Web service connection to the Microsoft SharePoint repository, you must change some settings of the copied `Web.config` file. To do this,

open the `Web.config` file and update the following parameters with details specific to your installation:

```
<appSettings>
  <add key="ConnectionString"  value="DataSource=w2k3-11\sharepointportal;Initial
Catalog=STS_W2K3-11_1040801166;Integrated Security=SSPI;" />
</appSettings>
```

To configure the connection to the SharePoint DB Server, use the connection strings in the `ConnectionString` parameter. These strings vary depending on the type of authentication used to access the MSSQL Server 2000 data:

- Using Integrated Windows Authentication to access site data

  `Data Source=<`*`DBINSTANCE`*`>;Initial Catalog=<`*`DBNAME`*`>;Integrated Security=SSPI;`

- Using SQL authentication to access site data

  `Data Source=<`*`DBINSTANCE`*`>;Initial Catalog=<`*`DBNAME`*`>;User Id=<`*`DBUSER`*`>;Password=<`*`DBUSERPASSWORD`*`>;`

where,

<*DBINSTANCE*> is an instance of the MSSQL Server 2000 with the SharePoint site's content, typically <*hostname*>/SHAREPOINT.

<*DBNAME*> is the name of the database with site content.

<*DBUSER*> is the SQL user name.

<*DBUSERPASSWORD*> is the SQL user password.

The name of the database can be retrieved from the SharePoint Central Administration page. Click the **Configure virtual server settings** link, select the server whose changes you want to listen to, and click the **Manage Content Databases** link. The database name is available in the Content Databases section.

The advantages of using Windows authentication is that no passwords are stored in the service configuration file. However, you must configure an application pool identity, which has sufficient rights to access the SharePoint site database, for the application pool to run as a Windows user. You can use **IIS Manager snap-in** to configure the application pool identity.

The SQL authentication does not require any additional steps to configure application pool identity, but in this case the SQL user password is stored in clear text which is not secure.

Executing Search SQL Scripts

Open any tool which can provide the script execution on the Microsoft SQL Server 2000. For example, you can use the standard Query Analyzer tool. Choose the database with the SharePoint Site data and run the `Functions.sql` script available in the `SearchWebService\SQLScripts` folder. This script creates all the necessary functions to provide the extended search feature.

Ensure that the script ran successfully.

5.2.7.6.2 Installing SharePoint Changes Service on Microsoft SharePoint 2003 Server Before configuring the data control based on the Oracle WebCenter adapter for Microsoft SharePoint, you must install changes service on your MS SharePoint 2003 server version 2.0. The Changes Service ensures that the JCR adapter's cache is notified when content in the SharePoint repository changes.

This section covers the following procedures:

- Installing the SharePoint Changes Service
- Configuring the SharePoint Changes Service

Installing the SharePoint Changes Service

To install the changes service, perform the following steps:

1. The SharePoint changes service is packaged as `sharepoint-services.zip`. Extract this ZIP file on your machine. This folder contains the following files:
   - `etc\SharePointChangeListener.exe.config`
   - `lib\observation\SharePointChangeListener.exe`
   - `lib\observation\log4net.dll`
   - `lib\observation\sql\Tables.sql`
   - `lib\observation\sql\Triggers.sql`
2. Create a new folder, for example, `SharePointChangesService`, on your local hard drive.
3. Copy the following Changes Service files from the sharepoint-services folder to the folder you just created:
   - `log4net.dll`, the logging library
   - `SharePointChangeListener.exe`, the service executable
   - `SharePointChangeListener.exe.config`, the service configuration file
4. Install the service with the `InstallUtil` utility and configure it
   - Open the command prompt and go to the Microsoft .NET installation directory, for example:

     MICROSOFT_NET_FRAMEWORK_PATH`\v1.1.4322\`

   - Start the command to register the service.

     `InstallUtil` *SERVICE_FOLDER_PATH*`\SharePointChangeListener.exe`

   where *MICROSOFT_NET_FRAMEWORK_PATH* is the path of the Microsoft .NET framework, which is typically `C:\WINDOWS\Microsoft.NET\Framework`. The *SERVICE_FOLDER_PATH* is the path to the change service folder, `SharePointChangesService`.
5. Verify that the changes service was installed successfully. To do this, perform the following steps:
   a. Look through the log files to confirm that the installation was successful.
   b. Ensure that the installed service appears in the list of the computer services.
6. Before starting the change service, you must configure it, as described in Configuring the SharePoint Changes Service.

Configuring the SharePoint Changes Service

To configure the SharePoint Changes Service you just installed, first configure the connection to your SharePoint repository and the list of JCR applications that must be notified of Repository changes. In addition, to add database objects to support the

change notification, run scripts against the SharePoint repository mechanism, as described in the following steps:

1. To configure the SharePoint repository connection, modify the application setting in the `SharePointChangeListener.exe.config` file:

```
<appSettings>
   <add key="Delay" value="10" /> - interval(in seconds) of scanning for
changes
   <add key="ConnectionString" value="connection_string" /> - connection string
to the SharePoint site database
</appSettings>
```

   To configure the connection to the SharePoint database server in the `ConnectionString` parameter, use the two types of connection strings available. These strings vary depending on the authentication method used to access the MSSQL Server 2000 data. The two types of connection strings are as follows:

   - Using Integrated Windows Authentication to access site data

     `Data Source=<DBINSTANCE>;Initial Catalog=<DBNAME>;Integrated Security=SSPI;`

   - Using SQL authentication to access site data

     `Data Source=<DBINSTANCE>;Initial Catalog=<DBNAME>;User Id=<DBUSER>;Password=<DBUSERPASSWORD>;`

   where,

   <*DBINSTANCE*> is an instance of the MSSQL Server 2000 with the SharePoint site's content, typically <*hostname*>/SHAREPOINT.

   <*DBNAME*> is the name of the database with site content.

   <*DBUSER*> is the SQL user name.

   <*DBUSERPASSWORD*> is the SQL user password.

   The name of the database can be retrieved from the SharePoint Central Administration page. Click the **Configure virtual server settings** link, select the server whose changes you want to listen to, and click the **Manage Content Databases** link. The database name is available in the Content Databases section.

   The advantages of using Windows authentication is that no passwords are stored in the service configuration file. However, you must configure an application pool identity for the application pool to run as a Windows user, which has sufficient rights to access the SharePoint site database. You can use **IIS Manager snap-in** to configure the application pool identity.

   The SQL authentication does not require any additional steps to configure application pool identity, but in this case the SQL user password is stored in clear text which is not secure.

2. To configure the list of JCR applications to be notified of changes, update the `<listeners>` element in the `SharePointChangeListener.exe.config` file:

```
<listeners>
    <listener>http://host:port/myapp/sharepoint-changes</listener>
</listeners>
```

Note: The SharePoint changes service must be stopped before any changes are made to its config file. When the service is stopped, the notifications registered (notification events) to the application are lost.

The configuration file of the JCR application must include the SharePoint changes servlet, Application Listener for Change Notification.

3. To add the database objects to support changes notification, run the supplied SQL scripts on the database with the stored SharePoint site data. The scripts can be found in the `lib\observation\sql` folder, as explained in Step 1. of Installing the SharePoint Changes Service. Open any tool which can provide script execution on the Microsoft SQL Server 2000. For example, you can use the standard Query Analyzer tool. Select the database with the SharePoint Site data and execute the SQL scripts on it in the following order:
   - `Tables.sql` - This script creates all necessary tables to provide event listening
   - `Triggers.sql` - This script creates all necessary listeners on the site content tables

   Ensure that the scripts ran successfully and that the table, `CRXMESSAGES`, has been successfully created in the SharePoint site database.
4. Start the SharePoint Changes Service. To do so, click **Start**, select **Control Panel**. In the Control Panel, open **Administrative Tools**, then open **Services**.

5.2.7.7 Configuring a Data Control Based on Oracle WebCenter Adapter for Microsoft SharePoint

To configure a data control based on Oracle WebCenter adapter for Microsoft SharePoint, perform the following steps:

1. In the Oracle JDeveloper, select the **Model** project of your application under Applications Navigator. Then, select **New** from the **File** menu. The New Gallery is displayed.
2. Under Business Tier, select **Content Repository**, and click **OK**.
3. Click **Next** to skip the Welcome page.
4. On Step 1: Data Control Name, enter a name for the new data control, and click **Next**.
5. On Step 2: Content Repository Configuration, select **JCR SharePoint Adapter** from the Repository Type box.
6. Then specify the configuration parameters as described in Table 5–14.

Table 5–14 Configuration Parameters for the Data Control Based on Oracle WebCenter Adapter for Microsoft SharePoint

Parameter	Description
Password	Password of the SharePoint admin user. To change the admin user password post deployment, perform the procedure given in Section 12.5, "Updating Credentials in a Deployed Application" and modify the value of `jcr_PASSWORD` parameter.

Table 5–14 (Cont.) Configuration Parameters for the Data Control Based on Oracle WebCenter Adapter for Microsoft SharePoint

Parameter	Description
Domain	Microsoft Windows domain of SharePoint Server.
Port	Port on which SharePoint Web Services are available.
Cache directory	Directory where the adapter stores cache and other data. It can be set to default or outside of the read-only installation location.
Host name	Host running SharePoint services.
Username	Microsoft Windows username for SharePoint. The user must have administrative rights, that is the user must have full access to the database specified by previous parameter. To change the username postdeployment, edit the `connections.xml` file available at `J2EE_HOME`/applications/`app name`/adf/META-INF/.

7. Select either the **Use JAAS for Security** check box, or supply a set of shared credentials in the Username and Password fields. If you specify shared credentials, then every connection to the Oracle WebCenter adapter for Microsoft SharePoint will use this same set of shared credentials. If you use JAAS for security, then the user identity will be used to log into the adapter. When configuring JAAS, for security reasons, it is strongly advised that the SharePoint repository shares identity management with the application server. This is because the JCR adapter uses a trust model to permit logon to the SharePoint repository based on user identity alone.

> **Note:** To change shared credentials post deployment, use the Edit Data Control wizard and then redeploy the application.

8. Click **Finish** to complete the creation of the data control. The data control is displayed under the Data Control Palette.

5.2.7.8 Verifying the Library of Oracle WebCenter Adapter for Microsoft SharePointin the Model Project

To verify if the Oracle WebCenter adapter for Microsoft SharePoint has been added to the library, perform the procedure described in Section 5.2.6.7, "Verifying the JCR Domino Adapter Library in the Model Project".

5.2.8 Configuring a Content Data Control Based on Oracle WebCenter Adapter for EMC Documentum

This section discusses the following:

- Section 5.2.8.1, "Overview of the Oracle WebCenter Adapter for EMC Documentum"
- Section 5.2.8.2, "Platform and DFC Requirements for the Oracle WebCenter Adapter for EMC Documentum"
- Section 5.2.8.3, "What You Should Know About the Oracle WebCenter Adapter for EMC Documentum"

- Section 5.2.8.4, "Installing Oracle WebCenter Adapter for EMC Documentum in Oracle JDeveloper"
- Section 5.2.8.5, "Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server"
- Section 5.2.8.6, "Configuring a Data Control Based on Oracle WebCenter Adapter for EMC Documentum"
- Section 5.2.8.7, "Verifying the Library of the Oracle WebCenter Adapter for EMC Documentum in the Model Project"

5.2.8.1 Overview of the Oracle WebCenter Adapter for EMC Documentum

The Oracle WebCenter adapter for EMC Documentum extracts content from an EMC Documentum repository and submits to the JCR Documentum adapter. The Oracle WebCenter adapter for EMC Documentum reads Documentum objects, documents, folders, users, and groups. It accesses the Documentum repository using the Documentum Foundation Classes (DFC). This adapter is used for the following purposes:

- **Read**: Allows to read users, groups, group memberships, and content stored in a Documentum repository.
- **Search**: Allows to search content stored in a Documentum repository.

The Oracle WebCenter adapter for EMC Documentum periodically checks for content updates, for example, modified, created, or deleted Documentum objects, in the Documentum repository. The adapter also checks for user and permission updates, such as users, groups, and group memberships, in the Documentum repository. The frequency of these checks is configurable.

The Oracle WebCenter adapter for EMC Documentum receives JCR internal query from WebCenter framework, and converts it to Documentum query language (DQL). Then, it converts Documentum resultset into JCR query results.

5.2.8.2 Platform and DFC Requirements for the Oracle WebCenter Adapter for EMC Documentum

Platform Requirements:

- EMC Documentum server version 5.3 or 5.2.5
- Platform-independent JAR files, such as `dfc.jar`, `dfcBase.jar`.
- Platform-specific library files:
  - Windows platform: `dmcl40.dll`
  - Unix/Linux platform: `libdmcl40.so`

Documentum Foundation Classes (DFC) Requirements:

DFC 5.0 files. DFC is an EMC Documentum client API library, and therefore needs to be obtained separately.

5.2.8.3 What You Should Know About the Oracle WebCenter Adapter for EMC Documentum

Consider the following points while you install the Oracle WebCenter adapter for EMC Documentum:

- **Latest Patch Required**: To use EMC Documentum-based content data control, Oracle JDeveloper and Oracle Application Server must be up-to-date with the

latest patch. Consult Oracle Application Server Release Notes for Microsoft Windows for release 10.1.3.2.0 to know the exact patch number.

- **Design Time Performance**: The Oracle WebCenter adapter for EMC Documentum is optimized for running in a server environment where a fast access time against a running connector outweighs a slower start-up time. A result of this is that the Test button in the JDeveloper Content Repository may be slow to complete the test.
- **Documentum Foundation Classes (DFC)**: The Oracle WebCenter adapter for EMC Documentum uses DFC to communicate with the Documentum server, therefore the JCR Documentum adapter is dependent on native libraries and configuration files found in the DFC install. DFC must be installed before any EMC Documentum client product can use it. DFC may also be installed independently.

  Installing other Documentum client applications may also install DFC, for example installing Documentum Desktop.
- **Documentum User**: The Oracle WebCenter adapter for EMC Documentum must be configured with the credentials of a Documentum user that is a super user and that has the `CONFIG AUDIT` extended privilege.
- **Shared Library Import for Oracle WebCenter Adapter for EMC Documentum**: If the shared library is not imported by default, then the application's `orion-application.xml` must be modified to import the Documentum shared library, as described in Section 12.2.1.3, "Manually Creating and Editing the orion-application.xml File". Example 5–9 shows `orion-application.xml` file to import the Documentum shared library.

Example 5–9 orion-application.xml to Import the Documentum Shared Library

```
orion-application.xml:
<?xml version = '1.0' encoding = 'windows-1252'?>
<orion-application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/orion-appli
cation-10_0.xsd">
  <library path="./adf"></library>
  <jazn location="/jazn-data.xml" provider="XML"/>
  <imported-shared-libraries>
    <import-shared-library name ="oracle.vcr.adf2documentum">
  </imported-shared-libraries>
</orion-application>
```

5.2.8.4 Installing Oracle WebCenter Adapter for EMC Documentum in Oracle JDeveloper

This section describes the following procedures:

- Installing Oracle WebCenter Adapter for EMC Documentum in Oracle JDeveloper on Windows
- Installing the Oracle WebCenter Adapter for EMC Documentum in Oracle JDeveloper on Linux

Installing Oracle WebCenter Adapter for EMC Documentum in Oracle JDeveloper on Windows

Before you install the Oracle WebCenter adapter for EMC Documentum on Windows, make sure that the following libraries are available in appropriate directories:

- dmcl.ini in Windows home.
- dmcl40.dll in the DFC directory, which must be included in the PATH environment variable.

To install the Oracle WebCenter adapter for EMC Documentum in Oracle JDeveloper, perform the following steps:

1. Extract /adapter/adf2documentum.zip that contains oracle.vcr.adf2jcr.10.1.3.zip and oracle.vcr.adf2documentum.10.1.3.zip.
2. Exit Oracle JDeveloper, if it is running.
3. Extract the oracle.vcr.adf2jcr.10.1.3.zip archive to *JDEV_HOME*/jdev/extensions. If you extracted this archive while installing SharePoint adapter, then skip this step.
4. Extract the adapter archive oracle.vcr.adf2documentum.10.1.3.zip to *JDEV_HOME*/jdev/extensions.
5. From Documentum DFC installation, copy the Documentum JARs, dfc.jar and dfcbase.jar to the *JDEV_HOME*/jdev/extensions/oracle.vcr.adf2documentum.10.1.3 directory.
6. Create a *JDEV_HOME*/jdev/extensions/oracle.vcr.adf2documentum.10.1.3/classes directory.
7. Copy the dfc.properties file from your Documentum client install into this classes directory. The dfc.properties file may be found in the config directory.
8. Ensure that the directories containing the Documentum DFC native libraries and the dmcl.ini configuration file are on the system path. The Documentum native library directory is the directory containing dmcl40.dll, dmcl40.pdb files. The dmcl.ini file should be in the Windows directory, for example, c:\windows. The Documentum installer is likely to have placed it in this location.
9. Restart Oracle JDeveloper to activate the Oracle WebCenter adapter for EMC Documentum.

Installing the Oracle WebCenter Adapter for EMC Documentum in Oracle JDeveloper on Linux

1. Extract /adapter/adf2documentum.zip that contains oracle.vcr.adf2jcr.10.1.3.zip and oracle.vcr.adf2documentum.10.1.3.zip.
2. Exit Oracle JDeveloper, if it is running.
3. Extract the oracle.vcr.adf2jcr.10.1.3.zip archive to *JDEV_HOME*/jdev/extensions. If you extracted this archive while installing Oracle WebCenter adapters for IBM Lotus Domino or Microsoft SharePoint, then skip this step.
4. Extract the shared library package oracle.vcr.adf2documentum.10.1.3.zip archive to *JDEV_HOME*/jdev/extensions.
5. Copy the dfc.jar and dfcbase.jar files to the *JDEV_HOME*/jdev/extensions/oracle.vcr.adf2documentum.10.1.3 directory.

6. Create a *JDEV_HOME*/jdev/extensions/oracle.vcr.adf2documentum.10.1.3/classes directory.
7. Copy the dfc.properties file from your Documentum client install into this classes directory. The dfc.properties file may be found in the config directory.
8. Define environment variable DMCL_CONFIG as the path to the dmcl.ini file, for example, setenv DMCL_CONFIG Documentum/dmcl.ini.
9. Add the full path to the directory containing the Documentum native library file, libdmcl40.so, to LD_LIBRARY_PATH and PATH. For example, setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/usr/dfc and setenv PATH $PATH:/usr/dfc.
10. Restart Oracle JDeveloper to activate the Oracle WebCenter adapter for EMC Documentum.

5.2.8.5 Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server

The Oracle WebCenter adapter for EMC Documentum is installed into the Oracle Application Server as a shared library. You can install the adapter using either of the following procedures:

- Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server on Windows
- Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server on Linux
- Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server using Application Server Control Console

Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server on Windows

This section explains how to install the Oracle WebCenter adapter for EMC Documentum on Oracle Application Server in Windows.

1. Extract /adapter/adf2documentum.zip that contains oracle.vcr.adf2jcr.10.1.3.zip and oracle.vcr.adf2documentum.10.1.3.zip.
2. Extract the oracle.vcr.adf2documentum.10.1.3.zip archive to a temporary working directory.
3. Extract the oracle.vcr.adf2jcr.10.1.3.zip archive to the same directory.
4. Copy the dfc.jar and dfcbase.jar files from your Documentum installation to the same directory.
5. Copy the dfc.properties file to the same directory.
6. Run the following command from the directory where you have copied dfc.properties file:

```
jar cf dfcProperties.jar dfc.properties
```

Note: Now the `dfc.properties` file is duplicated. The Oracle Application Server will use the one packaged in `dfcProperties.jar`.

7. Run the following command:

```
java -jar ORACLE_HOME/j2ee/home/admin_client.jar \
          DEPLOYER_URI_OC4J_INSTANCE username password \
          -publishSharedLibrary \
          -name oracle.vcr.adf2documentum -version 10.1.3 \
          -installCodeSources \
          oracle.vcr.adf2documentum.10.1.3.jar \
          oracle.vcr.adf2documentum.10.1.3/crx-core.jar \
          oracle.vcr.adf2documentum.10.1.3/crx2documentum.jar \
          oracle.vcr.adf2documentum.10.1.3/adf2documentum.jar \
          oracle.vcr.adf2documentum.10.1.3/commons-lang.jar \
          oracle.vcr.adf2documentum.10.1.3/log4j.jar \
          oracle.vcr.adf2jcr.10.1.3/adf2crx.jar \
          oracle.vcr.adf2jcr.10.1.3/crx-api.jar \
          oracle.vcr.adf2jcr.10.1.3/concurrent.jar \
          oracle.vcr.adf2jcr.10.1.3/crx-commons.jar \
          oracle.vcr.adf2jcr.10.1.3/day-collections.jar \
          oracle.vcr.adf2jcr.10.1.3/slf4j-jdk14.jar \
          oracle.vcr.adf2jcr.10.1.3/lucene-core.jar \
          oracle.vcr.adf2jcr.10.1.3/did.jar \
          oracle.vcr.adf2jcr.10.1.3/day-commons-text.jar \
          oracle.vcr.adf2jcr.10.1.3/day-commons-naming.jar \
          dfc.jar dfcbase.jar dfcProperties.jar \
          -imports adf.oracle.domain oracle.xml
```

8. Place the Documentum native library files, `dmcl40.dll` and `dmcl40.pdb`, in the `bin` subdirectory of your *ORACLE_HOME*.

Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server on Linux

This section explains how to install the Oracle WebCenter adapter for EMC Documentum on Oracle Application Server in Linux.

1. Extract `/adapter/adf2documentum.zip` that contains `oracle.vcr.adf2jcr.10.1.3.zip` and `oracle.vcr.adf2documentum.10.1.3.zip`.
2. Extract the `oracle.vcr.adf2documentum.10.1.3.zip` archive to a temporary working directory.
3. Extract the `oracle.vcr.adf2jcr.10.1.3.zip` archive to the same directory.
4. Copy the `dfc.jar` and `dfcbase.jar` files from your Documentum installation to the same directory.
5. Copy the `dfc.properties` file to the same directory.
6. Run the following command from the directory where you have copied `dfc.properties` file:

```
jar cf dfcProperties.jar dfc.properties
```

Note: Now the `dfc.properties` file is duplicated. The Oracle Application Server will use the one packaged in `dfcProperties.jar`.

7. Run the following command:

```
java -jar ORACLE_HOME/j2ee/home/admin_client.jar \
      DEPLOYER_URI_OC4J_INSTANCE username password \
      -publishSharedLibrary \
      -name oracle.vcr.adf2documentum -version 10.1.3 \
      -installCodeSources \
      oracle.vcr.adf2documentum.10.1.3.jar \
      oracle.vcr.adf2documentum.10.1.3/crx-core.jar \
      oracle.vcr.adf2documentum.10.1.3/crx2documentum.jar \
      oracle.vcr.adf2documentum.10.1.3/adf2documentum.jar \
      oracle.vcr.adf2documentum.10.1.3/commons-lang.jar \
      oracle.vcr.adf2documentum.10.1.3/log4j.jar \
      oracle.vcr.adf2jcr.10.1.3/adf2crx.jar \
      oracle.vcr.adf2jcr.10.1.3/crx-api.jar \
      oracle.vcr.adf2jcr.10.1.3/concurrent.jar \
      oracle.vcr.adf2jcr.10.1.3/crx-commons.jar \
      oracle.vcr.adf2jcr.10.1.3/day-collections.jar \
      oracle.vcr.adf2jcr.10.1.3/slf4j-jdk14.jar \
      oracle.vcr.adf2jcr.10.1.3/lucene-core.jar \
      oracle.vcr.adf2jcr.10.1.3/did.jar \
      oracle.vcr.adf2jcr.10.1.3/day-commons-text.jar \
      oracle.vcr.adf2jcr.10.1.3/day-commons-naming.jar \
      dfc.jar dfcbase.jar dfcProperties.jar \
      -imports adf.oracle.domain oracle.xml
```

8. Add Documentum environment variables `DMCL_CONFIG` and `LD_LIBRARY_PATH` to *ORACLE_HOME*`/opmn/config/opmn.xml` in the corresponding ias-component:

```
<ias-component id="default_group">
     <environment>
        <variable id="DMCL_CONFIG" value="/usr/dfc/dmcl.ini"/>
<!-- TThe value of DMCL_CONFIG is the full path to the dmcl.ini in the DFC
install. -->
        <variable id="LD_LIBRARY_PATH" value="$LD_LIBRARY_PATH:/usr/dfc"/>
<!-- The value of the LD_LIBRARY_PATH includes the full path to the directory
where the DFC native libraries are found. -->
     </environment>
  ...
</ias-component>
      ...
```

The Oracle Application Server can be configured such that the adapter shared library is imported by default by every application that is deployed to the server. This is done by updating the `system-application.xml` file. For information on system application, see chapter titled "Introduction to Oracle Containers for J2EE (OC4J)" of *Oracle Containers for J2EE Configuration and Administration Guide*. Such a configuration is only advisable if the contents of the Documentum shared library will not conflict with applications deployed to the container. If the shared library of Oracle WebCenter adapter for EMC Documentum is not imported by default, the application must include the import in its configuration. See Section 5.2.8.3, "What You Should Know About the Oracle WebCenter Adapter for EMC Documentum" for more information.

Installing the Oracle WebCenter Adapter for EMC Documentum on Oracle Application Server using Application Server Control Console

To install the Oracle WebCenter adapter for EMC Documentum on Oracle Application Server, perform the following steps:

1. Log into Application Server Control Console. The Cluster Topology page is displayed.
2. Under Members, click the home instance. The Home tab is displayed.
3. Select the **Administration** tab. Under Administration Tasks, as shown in Figure 5–20, click the **Go to Task** icon next to Shared Libraries. The Shared Library page is displayed, as shown in Figure 5–21.

Figure 5–20 Administration Tasks

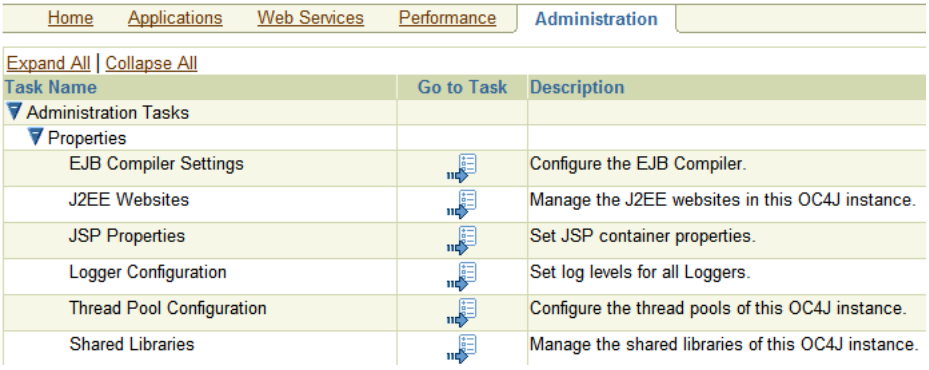
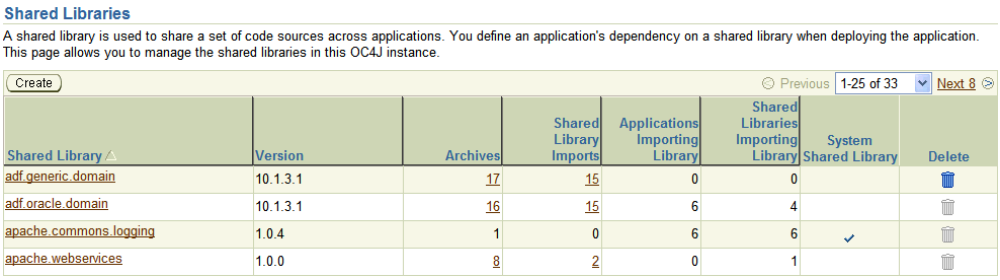
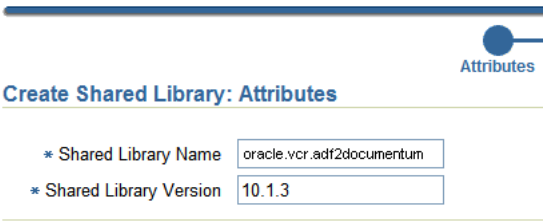


Figure 5–21 Shared Library



4. Click **Create**. The Create Shared Library: Attributes page is displayed, as shown in Figure 5–22.

Figure 5–22 Create Shared Library: Attributes



5. Enter the library name, **oracle.vcr.adf2documentum** and version, **10.1.3**, and click **Next**. The Create Shared Library: Add Archives page is displayed.
6. Click **Add**. The Add Archives: Add Archive page is displayed.

7. Specify the path to JARs of both `adf2jcr.10.1.3.jar` and `adf2documentum.10.1.3.jar`. For example, `adf2documentum.10.1.3.jar` includes the following JARs:
   - `oracle.vcr.adf2documentum.10.1.3.jar \`
   - `oracle.vcr.adf2documentum.10.1.3/crx-core.jar \`
   - `oracle.vcr.adf2documentum.10.1.3/crx2documentum.jar \`
   - `oracle.vcr.adf2documentum.10.1.3/adf2documentum.jar \`
   - `oracle.vcr.adf2documentum.10.1.3/commons-lang.jar \`
   - `oracle.vcr.adf2documentum.10.1.3/log4j.jar \`
8. Specify a minimum or maximum version to import. This is an optional step.
9. Click **Finish**. The Documentum shared library is added and is displayed under the Shared Library page.

5.2.8.6 Configuring a Data Control Based on Oracle WebCenter Adapter for EMC Documentum

Before configuring a data control based on Oracle WebCenter adapter for EMC Documentum, you must perform the procedure in Section 5.2.8.4, "Installing Oracle WebCenter Adapter for EMC Documentum in Oracle JDeveloper".

To configure a data control, perform the following steps:

1. In the Oracle JDeveloper, select the **Model** project of your application under Applications Navigator. Then, select **New** from the **File** menu. The New Gallery is displayed.
2. Under Business Tier, select **Content Repository**, and click **OK**.
3. Click **Next** to skip the Welcome page.
4. On Step 1: Data Control Name, enter a name for the new Documentum adapter-based data control, and click **Next**.
5. On Step 2: Content Repository Configuration, select **JCR Documentum Adapter** from the Repository Type box.
6. Then, specify the configuration parameters as described in Table 5–15.

Table 5–15 Configuration Parameters for the Data Control Based on Oracle WebCenter Adapter for EMC Documentum

Parameter	Description
Documentum version	Version of the adapter, either 5.2 or 5.3.
Root path	It is the Documentum subtree. It defaults to the user's default cabinet and is set to "/" to list all cabinets. It accepts any path to a `dm_folder`. Assign to the `rootPath` parameter, the value of the root of the subtree as the starting point of the view, for example, `/DocUser/FolderA/FolderA0`. The view will encompass the entire subtree starting from the `rootPath` root.
Document base	Documentum DocBase name, for example, `dctm`.

Table 5–15 (Cont.) Configuration Parameters for the Data Control Based on Oracle WebCenter Adapter for EMC Documentum

Parameter	Description
Allowed objects	This parameter is used to specify the types of objects that can be accessed through the Oracle WebCenter adapter for EMC Documentum. Add Documentum types with a comma-seperated list of the acceptable types, for example `dm_document`, `dm_acl` (no prefix required). The `*` symbol allows all. Blank or default removes the system types.
Denied objects	Deny Documentum types with a comma seperated list. The `*` symbol denies all, same as no allowedObject parameter.
Refresh interval	The `refreshIntervall` property contains the interval in seconds between checking changes in the Documentum repository. It defaults to 60 seconds. The Docbase's Audittrail checks for changes on Documentum objects.
User name	Username of the admin user to connect to the Oracle WebCenter adapter for EMC Documentum. To change the username postdeployment, edit the `connections.xml` file available at *`J2EE_HOME`*/applications/*`app name`*/adf/META-INF/.
password	Password of the admin user as required by Oracle WebCenter adapter for EMC Documentum to connect to the repository. To change the admin user password post deployment, perform the procedure given in Section 12.5, "Updating Credentials in a Deployed Application" and modify the value of `jcr_PASSWORD` parameter.
Cache directory	Path where java content repository is to be created. This is an optional field. However, it is required for Oracle Application Server to place the directory outside the hierarchy.

7. Select either the **Use JAAS for Security** check box, or supply a set of shared credentials in the Username and Password fields. If you specify shared credentials, then every connection to the adapter will use this same set of shared credentials. If you use JAAS for security, then the user identity will be used to log into the Oracle WebCenter adapter for EMC Documentum. When configuring JAAS, for security reasons, it is strongly advised that the Documentum repository shares identity management with the application server. This is because the adapter uses a trust model to permit logon to the Documentum repository based on user identity alone.

> **Note:** To change shared credentials post deployment, use the Edit Data Control wizard and then redeploy the application.

8. Click **Finish** to complete the creation of the data control. This data control is displayed under the Data Control Palette.

5.2.8.7 Verifying the Library of the Oracle WebCenter Adapter for EMC Documentum in the Model Project

To verify whether the Oracle WebCenter adapter for EMC Documentum has been added to the library, perform the procedure explained in Section 5.2.6.7, "Verifying the JCR Domino Adapter Library in the Model Project".

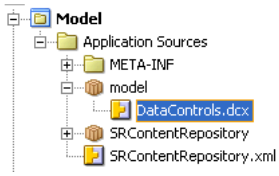
5.2.9 Editing Content Data Controls

This section describes a generic procedure to edit content data controls that you configured as described in Section 5.2, "Configuring Content Data Controls for JCR Adapters".

To edit a content data control, perform the following steps:

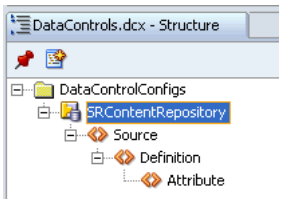
1. In Oracle JDeveloper, go to the application that contains your content data control.
2. Under your application, expand **Model** and **Application Sources**. Then select **DataControls.dcx**, as shown in Figure 5–23.

Figure 5–23 DataControls.dcx



3. In the Structure window, select the content data control you want to edit, as shown in Figure 5–24.

Figure 5–24 Content Data Control in the Structure Window



4. Right-click your content data control and select **Edit**, as shown in Figure 5–25. The Create Data Control dialog is displayed, as shown in Figure 5–25.

Figure 5–25 Context Menu

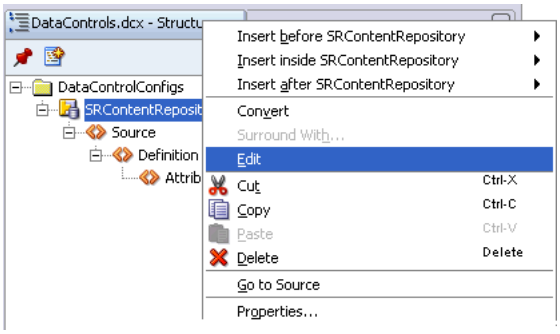
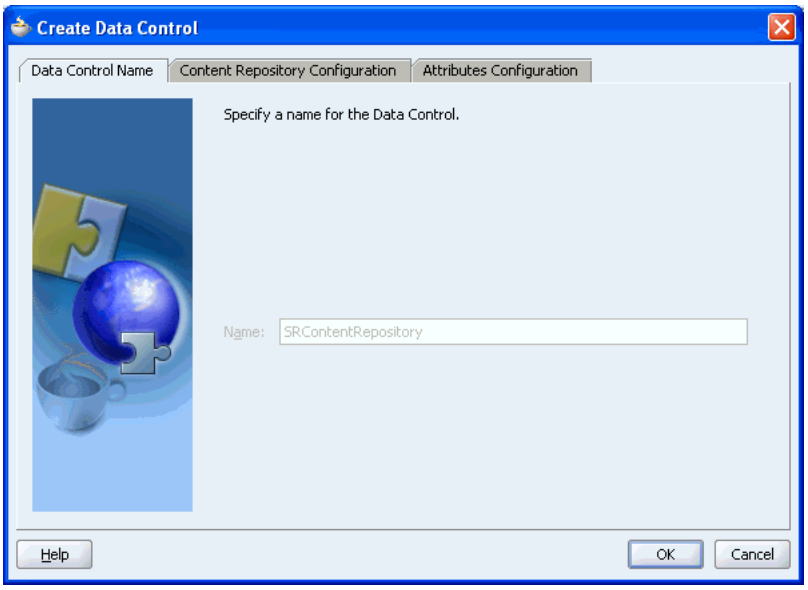


Figure 5–26 Create Data Control



5. To edit the repository path, use the Content Repository Configuration tab.
6. To add or delete custom attributes, use the Attributes Configuration tab.

5.2.10 Applying Oracle ADF Security on JCR Data Controls

You can enable security for the new bindings and executables created in the page definition. See Section 10.2.3.4, "Applying Security on JCR Data Controls" for details.

5.3 Using JCR Data Controls: Examples

In this section, you will use `getURI`, `getItems`, and `search` methods to add content to a page at design time, to publish content as a table and tree, and to add search capabilities for the integrated content.

Read the sample procedures in the following sections to use the data control methods created in Section 5.2, "Configuring Content Data Controls for JCR Adapters":

- Section 5.3.1, "Publishing Content As Links"
- Section 5.3.2, "Publishing Content in a Table"
- Section 5.3.3, "Publishing Folder Content in a Tree"
- Section 5.3.4, "Adding Search Capabilities to Content Repositories"
- Section 5.3.5, "Configuring Custom Attributes in Oracle Content DB"
- Section 5.3.6, "Creating Clickable Images Using Custom Attributes"

Note: If you are using an Oracle Content DB adapter-based data control and encounter a generic exception such as `JBO-29000: Unexpected exception caught: java.lang.RuntimeException, msg=Unknown exception` while running your JSPX page, then check OC4J and `opmn` log files corresponding to the Oracle Content DB instance. By default, application logs are located in:

ORACLE_HOME`/j2ee/OC4J_Content/application-deployments/Content/OC4J_Content_default_group_1/application.log`

ORACLE_HOME`/opmn/logs/default_group~OC4J_Content~default_group~1.log`

For more information about logs, see chapter titled "Monitoring Domain, Node, Service, and Server Performance" in *Oracle Content Database for Oracle WebCenter Suite Administrator's Guide*.

5.3.1 Publishing Content As Links

This section describes how to create hyperlinks to files stored in a file system and convert them into textual and image links. You use the ADF Go Link and the `getURI` method to create textual links and the ObjectImage of ADF Faces Core to create image links. The output of these procedures should be similar to Figure 5–27 and Figure 5–28.

Figure 5–27 ADF Go Link in a Browser

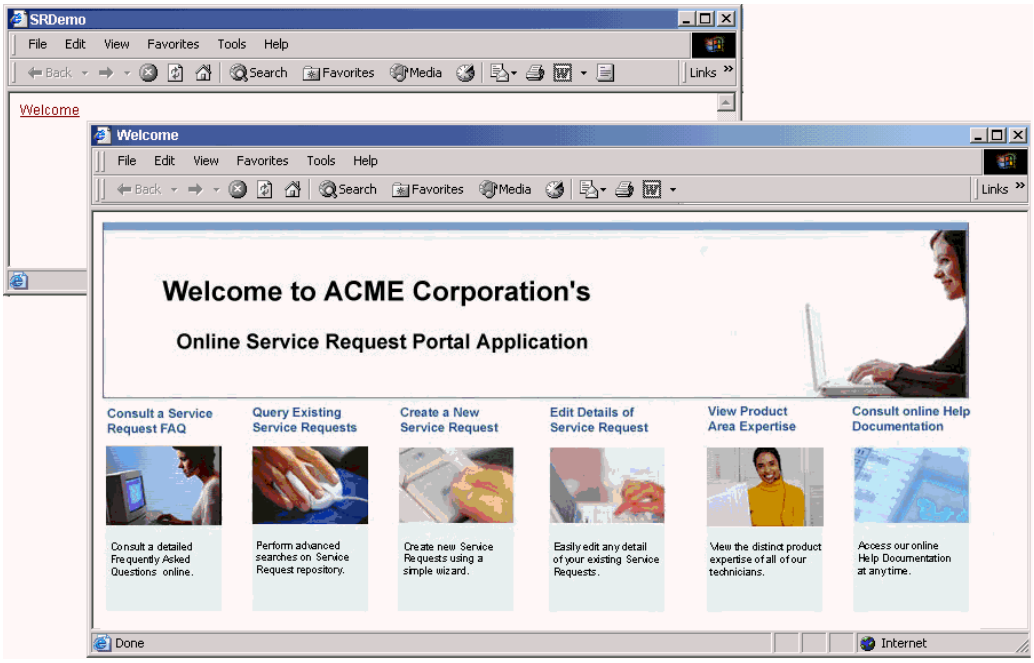
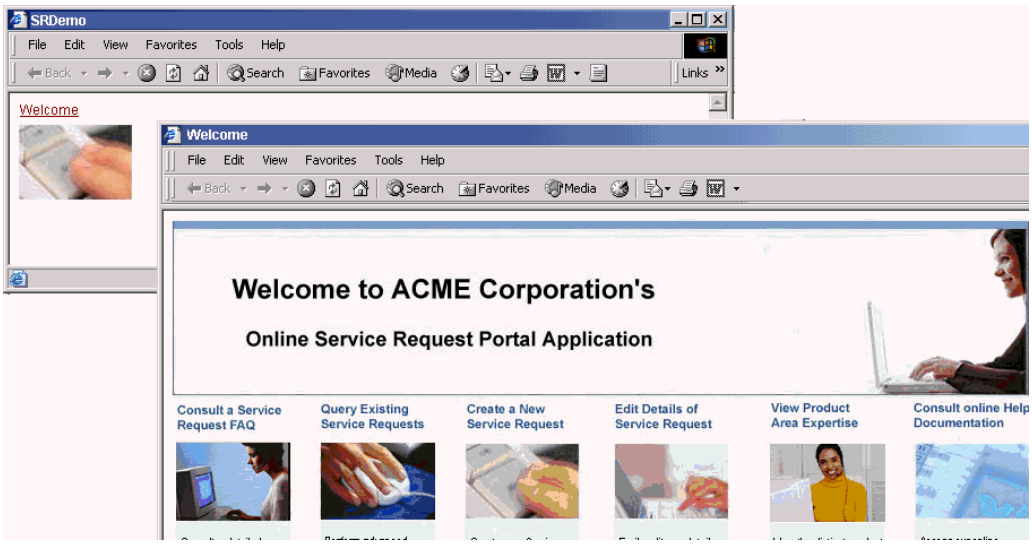


Figure 5–28 ADF Object Image Link in a Browser



This section includes the following procedures:

- Section 5.3.1.1, "Publishing Content As a Textual Link"
- Section 5.3.1.2, "Publishing Content As an Image Object"

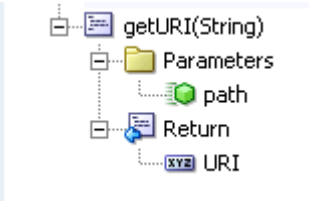
5.3.1.1 Publishing Content As a Textual Link

In this section, you will use the Oracle ADF Go Link option of the `getURI` method to publish your content as a textual link.

To publish content as a textual link, perform the following steps:

1. In the Applications Navigator, double-click a `.jspx` page to open it in the Visual Editor. In this example the page name is `MySRPage.jspx`.
2. In the Data Control Palette, under `SRFileSystem`, expand the **getURI (String)** method and expand **Return**. You should see the `URI` attribute, as shown in Figure 5–29.

Figure 5–29 SRFileSystem.getURI



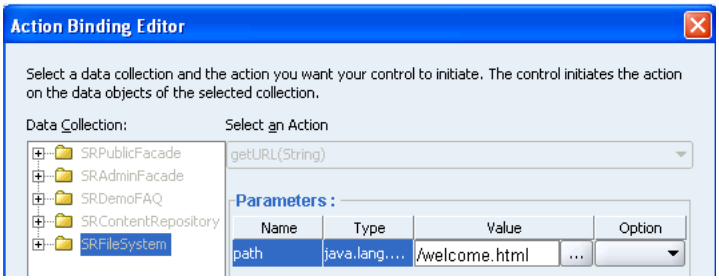
3. Select the **URI** attribute and drag and drop it on to the page, or in the Structure window under `h:form`. From the **Create** menu, select **ADF Go Link**, as shown in Figure 5–30. The Action Binding Editor dialog box is displayed.

Figure 5–30 Oracle JDeveloper Context Menu for the getURI method

Note: The Action Binding Editor dialog box displays only the first time that you drop the `getURI` string. To modify or delete the path that you specified the first time, edit the page definition by right-clicking the page and selecting Go to Page Definition.

4. In the **Value** field of the path parameter, enter the path of the file for which you want to create the link, as shown in Figure 5–31. Enter a slash (`/`).

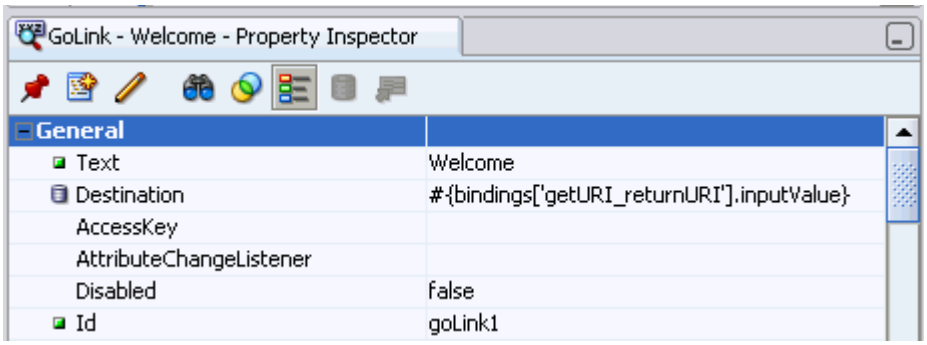
Figure 5–31 Action Binding Editor



Note: To grant edit, personalize, customize, and view permissions at the attribute level, see Section 5.2.10, "Applying Oracle ADF Security on JCR Data Controls".

5. Click **OK**.
6. Right-click the page and select **Run**. In your browser, you should see the URL for the page without any formatting.
7. By default, the link displays the text `goLink1`. In the Structure window, select **af:goLink - goLink - 1** and view the properties in the Property Inspector.
8. In the **Text** field, enter a name for the link, for example, `Welcome`, as shown in Figure 5–32.

Figure 5–32 GoLink Properties



9. Right-click your page and select **Run**. The page appears in your browser window with the new link.

10. Click the link to check that the correct file is displayed. Your browser should look like the one displayed in Figure 5–27.

In Section 5.3.1.2, "Publishing Content As an Image Object", you will extend this textual link into an image link.

Adding Textual Links to Display Different Content from a Repository

To add another link to your `.jspx` page, perform the following steps:

1. In the Applications Navigator, right-click the `.jspx` page, in which you created the ADF Go Link, and select **Go to Page Definition**.

2. In the `executables` element, add another `methodIterator` and change the `methodIterator id` and value of `Binds`, as shown in **Bold** in the following example:

```
<executables>
    <methodIterator id="getURIIter" Binds="getURI.result"
                    DataControl="SRContentRepository" RangeSize="10"
                    BeanClass="SRContentRepository.getURI_return"/>
    <methodIterator id="getURIIter1" Binds="getURI1.result"
                    DataControl="SRContentRepository" RangeSize="10"
                    BeanClass="SRContentRepository.getURI_return"/>
  </executables>
```

3. In the `Bindings` element, add another `methodAction` and change the `methodAction id`, `ReturnName`, and `NDValue`, as shown in **Bold** in the following example:

```
<methodAction id="getURI" InstanceName="SRContentRepository"
                  DataControl="SRContentRepository" MethodName="getURI"
                  RequiresUpdateModel="true" Action="999"
                  IsViewObjectMethod="false"

ReturnName="SRContentRepository.methodResults.SRContentRepository_getURI_
result">
      <NamedData NDName="path" NDValue="/SRContentRepository/welcome.html"
                 NDType="java.lang.String"/>
    </methodAction>
    <methodAction id="getURI1" InstanceName="SRContentRepository"
                  DataControl="SRContentRepository" MethodName="getURI"
                  RequiresUpdateModel="true" Action="999"
                  IsViewObjectMethod="false"
```

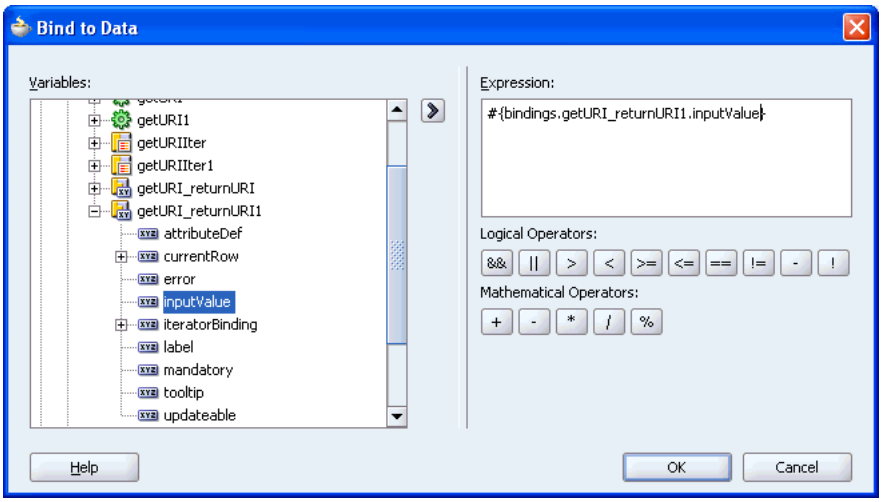
```
ReturnName="SRContentRepository.methodResults.SRContentRepository_getURI1_
result">
    <NamedData NDName="path" NDValue="/SRContentRepository/edit.jpg"
NDType="java.lang.String"/>
    </methodAction>
```

4. In the `Bindings` element, add another `attributeValues` tag to specify the new Id, as shown in **bold** in the following example:

```
<attributeValues id="getURI_returnURI" IterBinding="getURIIter">
      <AttrNames>
        <Item Value="URI"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="getURI_returnURI1" IterBinding="getURIIter1">
      <AttrNames>
        <Item Value="URI"/>
      </AttrNames>
    </attributeValues>
```

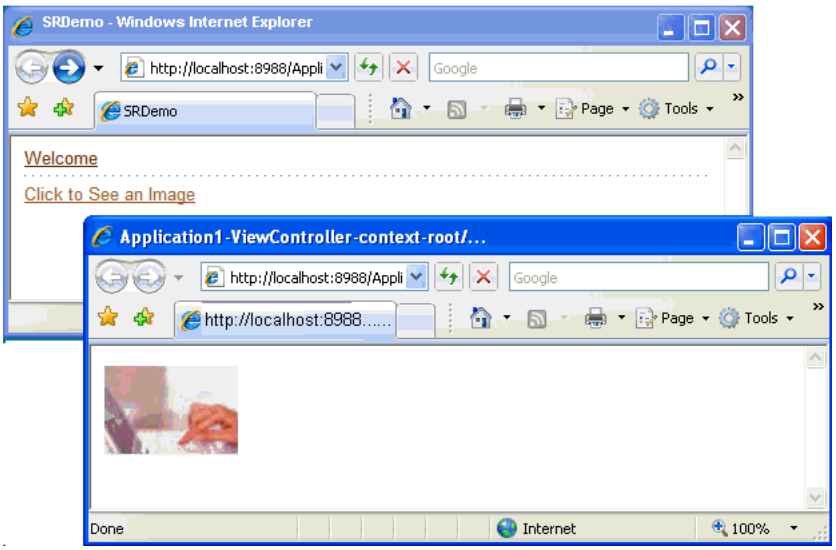
5. Follow steps 2 and 3 from the first procedure of Section 5.3.1.1, "Publishing Content As a Textual Link".
6. In the Structure window, double-click the new `goLink`, for example, `af:goLink2` to display the GoLink Properties window.
7. In the Text field, enter a display name for your new link.
8. In the Destination field, click **Bind** to display the Bind to Data dialog box. Then, expand **ADF Bindings**, **bindings**, and **getURI_returnURI1**.
9. Double-click **inputValue** variable to create the `#{bindings.getURI_returnURI1.inputValue}` expression, as shown in Figure 5–33, and click **OK**. This expression is based on new elements that you added in `executables` and `bindings` in steps 2 and 3.

Figure 5–33 Bind to Data



10. Run your page to see the output in a browser. The new link should display content from the new path (`NDValue`) that you specified in step 3. Figure 5–34 shows the new link in addition to the Welcome link that was added in the first procedure described in Section 5.3.1.1, "Publishing Content As a Textual Link".

Figure 5–34 New Textual Link



You can add as many links as required by following these steps.

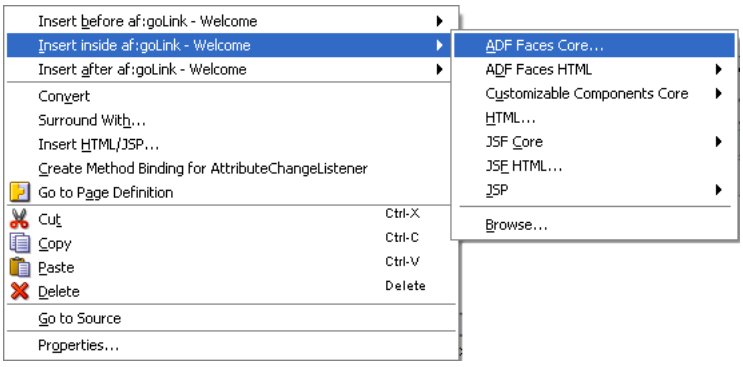
5.3.1.2 Publishing Content As an Image Object

In this section, you will use the ObjectImage option of ADF Faces Core to publish your content as a clickable image object. Clicking the image object displays your content.

To publish content as a clickable image object, perform the following steps:

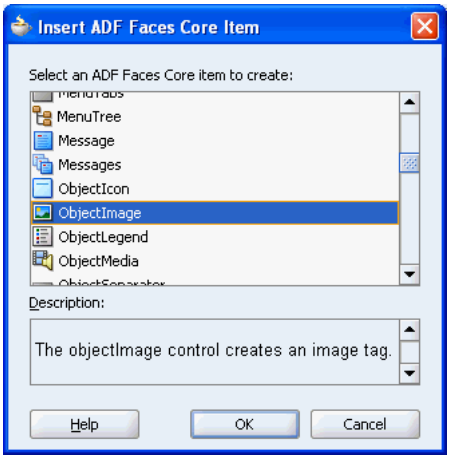
1. In the Applications Navigator, open the `.jspx` page, in which you created the ADF Go Link, by double-clicking it.
2. In the Structure window, right-click **af:goLink**, which you created in Section 5.3.1.1, "Publishing Content As a Textual Link", select **Insert Inside af:goLink**, and then select **ADF Faces Core**, as shown in Figure 5–35. The Insert ADF Faces Core Item dialog box is displayed.

Figure 5–35 Insert Inside af:goLink - ADF Faces Core



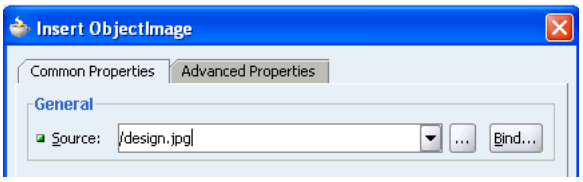
3. From the Select an ADF Faces Core item to create list, select **ObjectImage**, as shown in Figure 5–36.

Figure 5–36 Insert ADF Faces Core Item



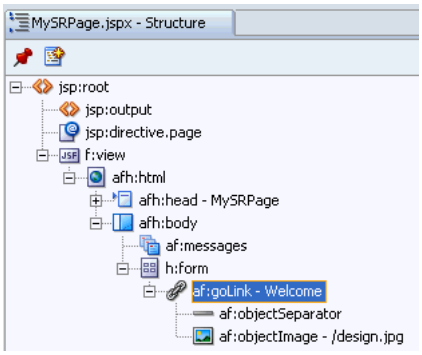
4. Click **OK**. The Insert ObjectImage dialog box is displayed.
5. Browse to the image file you want to display as the image link, for example, `design.jpg`, as shown in Figure 5–37.

Figure 5–37 Insert ObjectImage



6. Click **OK**. The Structure window should look like Figure 5–38.

Figure 5–38 Structure Window



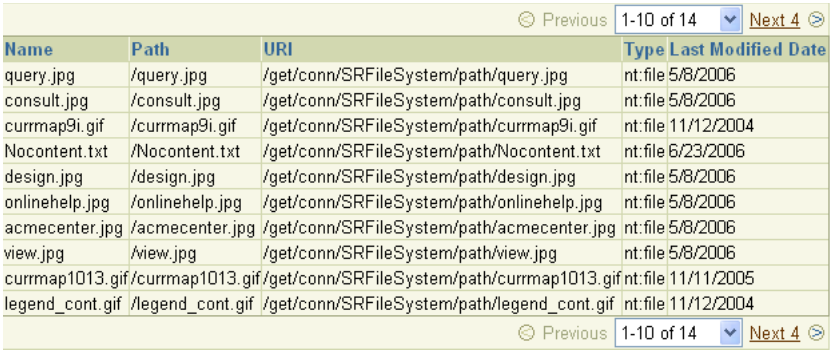
7. Right-click your page and select **Run**. The output should be similar to Figure 5–28.

5.3.2 Publishing Content in a Table

In this section, the `getItems` data control method is used to publish file and folder information in a table. This section describes the following procedures:

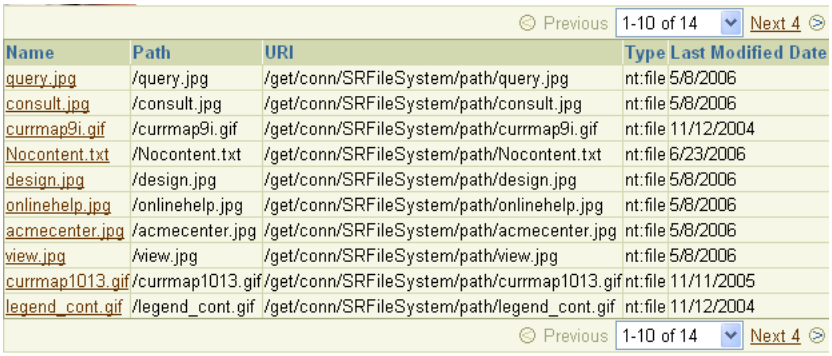
- Section 5.3.2.1, "Displaying Files and Folders in Read-Only Format": This procedure generates output similar to Figure 5–39.

Figure 5–39 Files and Folders Displayed in a Read-Only Table



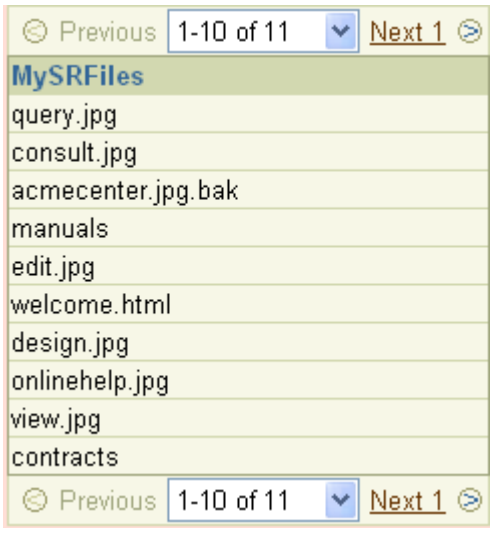
- Section 5.3.2.2, "Displaying the Name Attribute As a GoLink": This procedure generates output similar to Figure 5–40.

Figure 5–40 Folder Content Displayed as Hyperlinks



- Section 5.3.2.3, "Configuring a Table to Show Only a Single Column": This procedure generates output similar to Figure 5–41.

Figure 5–41 Files and Folders Displayed in a Single-Column Table



- Section 5.3.2.4, "Configuring a Table to Show Only Files": This procedure generates output similar to Figure 5–42.

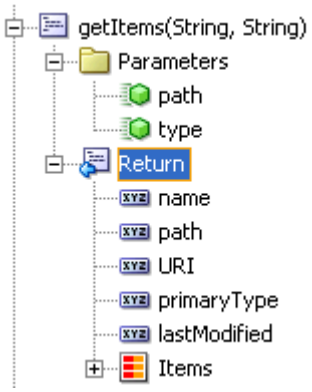
Figure 5–42 Files Displayed in a Single-Column Table

5.3.2.1 Displaying Files and Folders in Read-Only Format

To display folder content in a read-only table, perform the following steps:

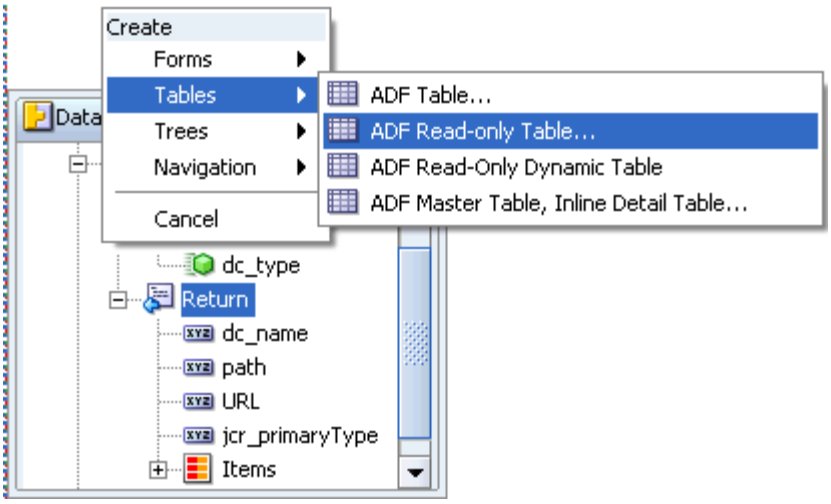
1. To open the page in the Visual Editor, double-click `MySRPage.jspx` in the Applications Navigator.
2. In the Data Control Palette, under `SRFileSystem`, expand the **getItems** method and the **Return** node, as shown in Figure 5–43.

Figure 5–43 SRFileSystem.getItems



3. To create a table that lists every file and folder available through this data control, drop the **Return** node on to the page or under `h:form` in the Structure window. The **Create** menu is displayed.
4. From the **Create** menu, select **Tables** and then select **ADF Read-only Table,** as shown in Figure 5–44. The Action Binding Editor dialog box is displayed.

Figure 5–44 Oracle JDeveloper Context Menu for getItems Method

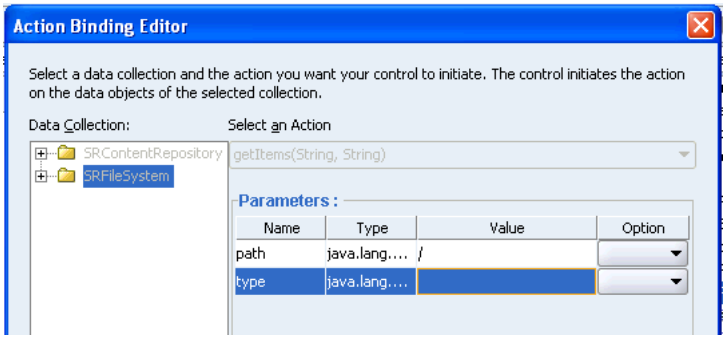


Note: The Action Binding Editor dialog box displays only the first time that you drop the return node. To modify or delete the path you specified the first time, edit the page definition by right-clicking the page and selecting Go to Page Definition.

5. Enter the path of the content directory as the `path` parameter, as shown in Figure 5–45. You must enter a slash (`/`).

   Leave the `type` parameter blank. This implies that the table displays both files and folders.

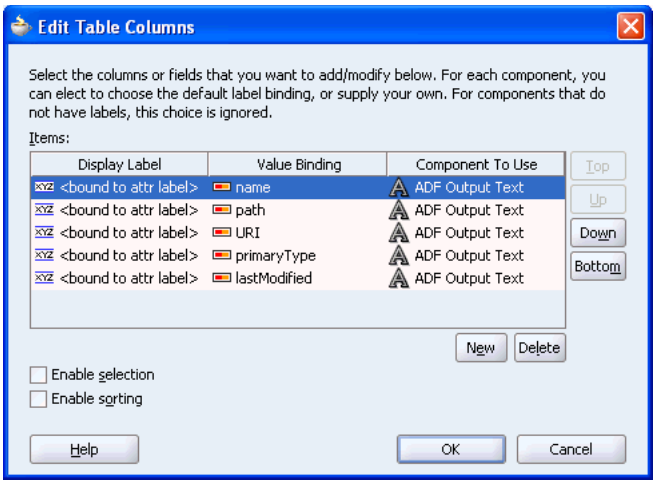
Figure 5–45 Action Binding Editor



Note: To grant edit, personalize, customize, and view permissions at the attribute level, See Section 5.2.10, "Applying Oracle ADF Security on JCR Data Controls".

6. Click **OK.** The Edit Table Columns dialog box is displayed, as shown in Figure 5–46.

Figure 5–46 Edit Table Columns



7. Click **<bound to attr label>** under Display Label and enter `Name`.

   Repeat this step for the **path**, **URI**, **primaryType**, and **lastModified** attributes. Enter new display labels such as `Path`, `URL`, and so on.

8. Click **OK.** You should now see a table on `MySRPage.jspx` that looks like Figure 5–47.

Figure 5–47 Read-Only Table for Publishing Folder Content

Name	Path	URI	Type	Last Modified Date
#{row.name}	#{row.path}	#{row.URI}	#{row.primaryType}	#{row.lastModified}
#{row.name}	#{row.path}	#{row.URI}	#{row.primaryType}	#{row.lastModified}
#{row.name}	#{row.path}	#{row.URI}	#{row.primaryType}	#{row.lastModified}

Note: You can turn the page caching on or off. To do so, open the page definition and expand **executables**, and select **methodIterator** in the Structure Pane. Then, in the Property Inspector, set CacheResults to **true** or **false**, as required.

---

9. Run the page in the browser to see the output. You should see a list of all files and folders available in your content directory, as shown in Figure 5–39.

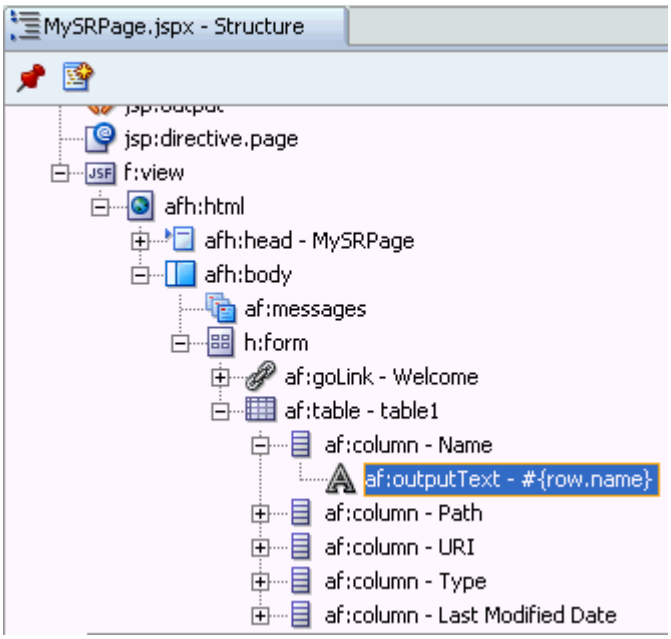
   By default, the table displays file or folder attributes as read-only text (`af:outputText`). The next section describes how to display the `Name` attribute (`name`) as a GoLink (`af:goLink`).

5.3.2.2 Displaying the Name Attribute As a GoLink

To display the `Name` attribute as a `GoLink`, perform the following steps:

1. In the Structure window (Figure 5–48), expand the first column of the table (`af:column - Name`) to show the default display format `af:outputText - #{row.name}`.

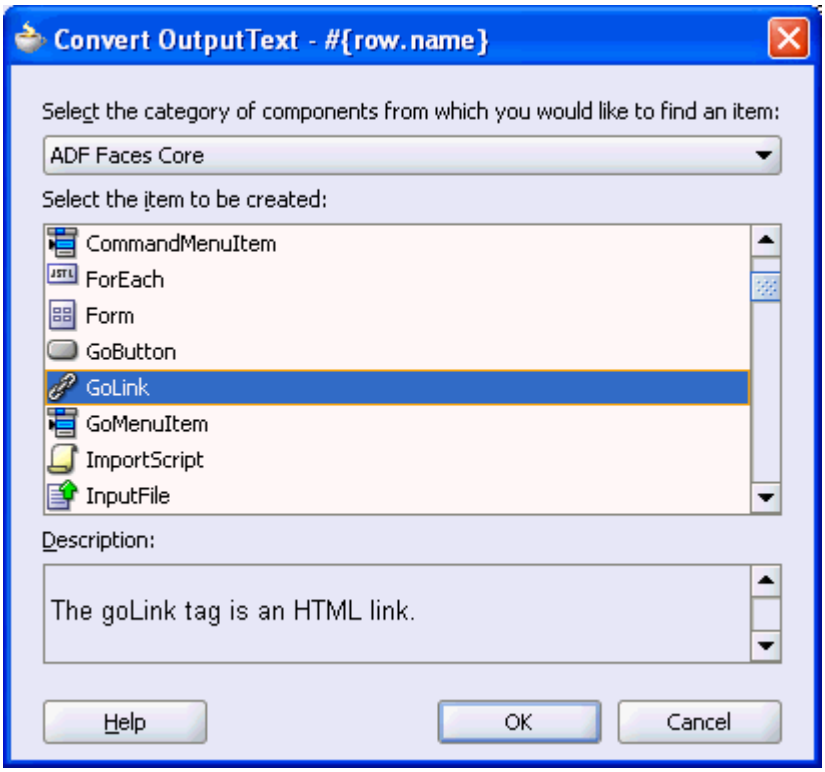
Figure 5–48 Default Formatting for the Name Column



2. Right-click **af:outputText - #{row.name}** and click **Convert**. The Convert OutputText dialog box is displayed.

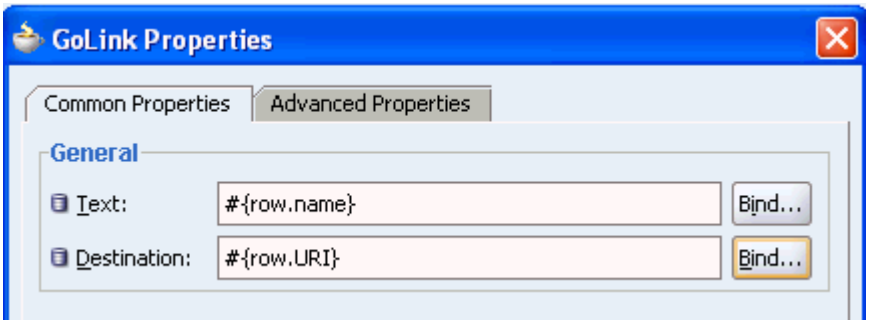
3. Select **ADF Faces Core** as the component category. In the Select the item to be created box select **GoLink**, as shown in Figure 5–49, then click **OK**. The Confirm Convert dialog box is displayed. Click **OK** to complete conversion.

Figure 5–49 Convert OutputText to a GoLink



4. In the Structure window, double-click **af:goLink - goLink - 1** to display the GoLink Properties dialog box.
5. To build the `#{row.name}` expression that displays the file or folder name, click **Bind** next to the **Text** field. The Bind to Data dialog box is displayed.
6. Expand **JSP Objects**, and then expand **row**. Click **OK**. Then, click **OK** in the GoLink Properties dialog box.
7. In the **Destination** field, click **Bind** to build `#{row.URI}`, an expression from Expression Language that passes the correct URL for the HTTP link, as shown in Figure 5–50.

Figure 5–50 GoLink Properties



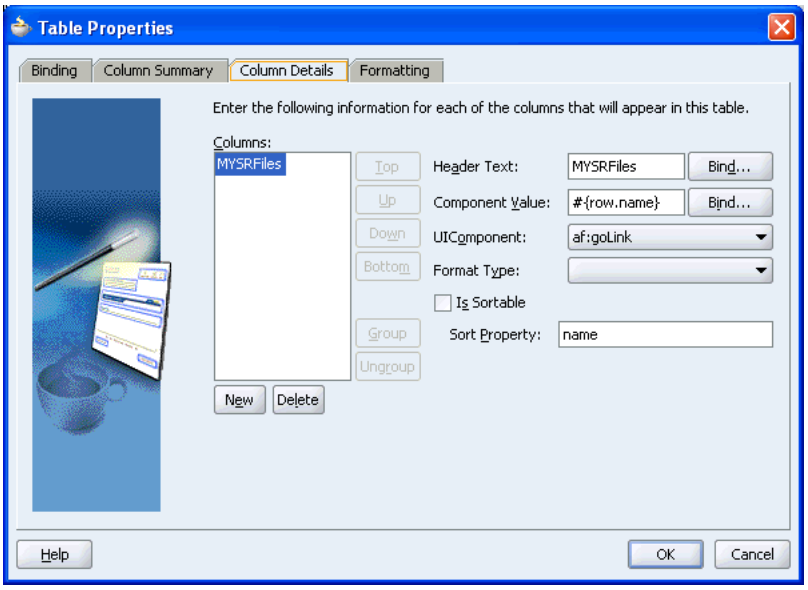
8. Right-click the page in the Applications Navigator and select **Run**. You should see a list of hyperlinked file and folder names like the one shown in Figure 5–40.
9. Click one of the file names. The file you pick should appear in a browser window.
10. Click the name of a folder. You should see an authorization error, because you cannot access folders through a direct URL. Folders can be accessed through a data control only.

5.3.2.3 Configuring a Table to Show Only a Single Column

To configure the table to show only the Name column, perform the following steps:

1. In the Structure window, double-click the **af:table -table** node to display the Properties dialog box.
2. Click the **Column Summary** tab. Use the **Delete** button to remove all but the Name column.
3. Click the **Column Details** tab. In the **Header Text** field enter a name, for example, `MySRFiles`, as shown in Figure 5–51, and click **OK**. The Name column is configured.

Figure 5–51 Table Properties - Column Detail Tab



4. Run the page to view the output, as shown in Figure 5–41.

In the next section, the Name column will be configured to show only files.

5.3.2.4 Configuring a Table to Show Only Files

The `type` attribute is used to configure a table to show only files.

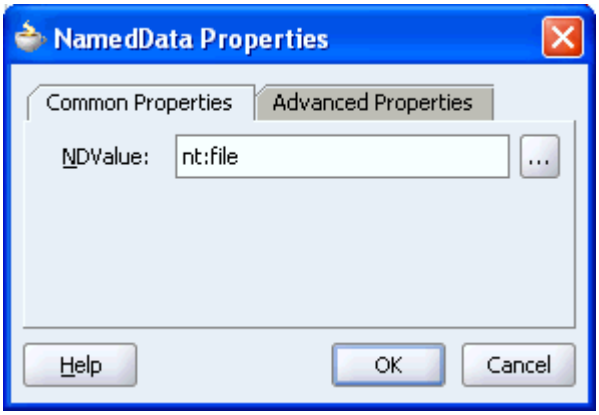
To configure the table to show files (not folders), perform the following steps:

1. Right-click your `.jspx` page and select **Go to Page Definition**. The page definition is displayed.

> **Note:** The `RangeSize` binding setting, which is used to control the number of items displayed on a page, is set to `10` by default in the page definition file. You can change it, as required, in the Property Inspector.

2. In the Structure window, expand **Bindings** and **getItems**, and double-click **type**.
3. The **type** options are `nt:file` and `nt:folder`. To specify the display of only files, enter `nt:file` in the **NDValue** field, as shown in Figure 5–52, and click **OK**.

Figure 5–52 Display Files Only



4. Now run the page. The list of files should look like Figure 5–42.

5.3.3 Publishing Folder Content in a Tree

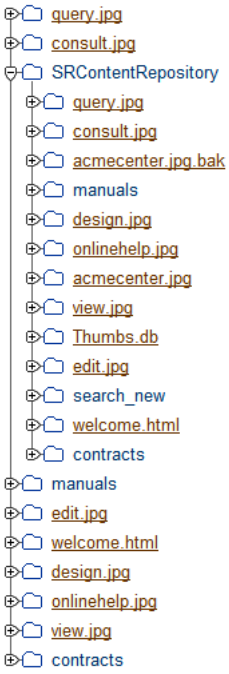
In this section, you will use the `getItems` method to publish content in a hierarchal tree format. This section describes the following procedures:

- Section 5.3.3.1, "Displaying Files and Folders in Read-Only Format": This procedure generates output similar to Figure 5–53.

Figure 5–53 Folder Content Displayed in a Tree

- Section 5.3.3.2, "Displaying File Names As Hyperlinks": This procedure generates output similar to Figure 5–54.

Figure 5–54 Tree with File Names as Hyperlinks

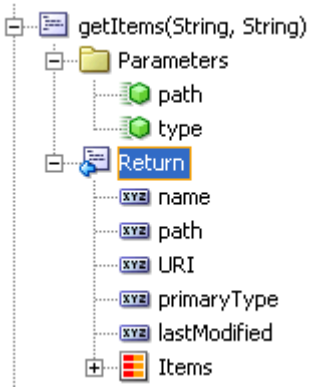


5.3.3.1 Displaying Files and Folders in Read-Only Format

To display your content in the tree format, perform the following steps:

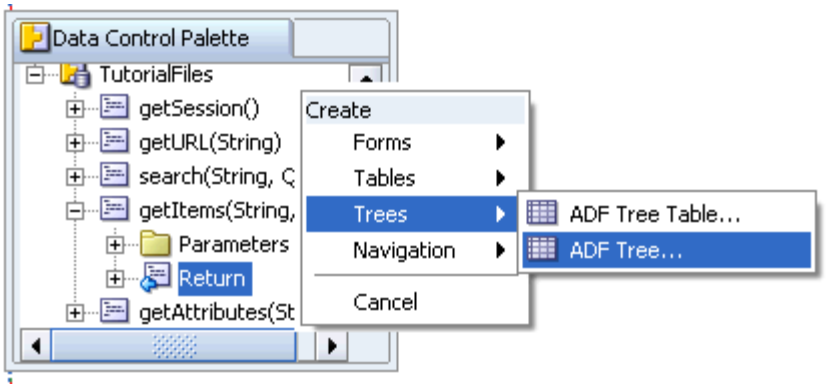
1. In the Applications Navigator, double-click your page to open it in the Visual Editor. In this example, the name of the page is `MySRPage.jspx`.
2. In the Data Control Palette, under `SRFileSystem`, expand the **getItems** method as shown in Figure 5–55.

Figure 5–55 SRFileSystem.getItems



3. Select the **Return** node and drag it onto the page. The **Create** menu is displayed.
4. Select **Trees** and then **ADF Tree**, as shown in Figure 5–56. The Action Binding Editor dialog box is displayed, as shown in Figure 5–57.

Figure 5–56 Oracle JDeveloper Create Menu for getItems

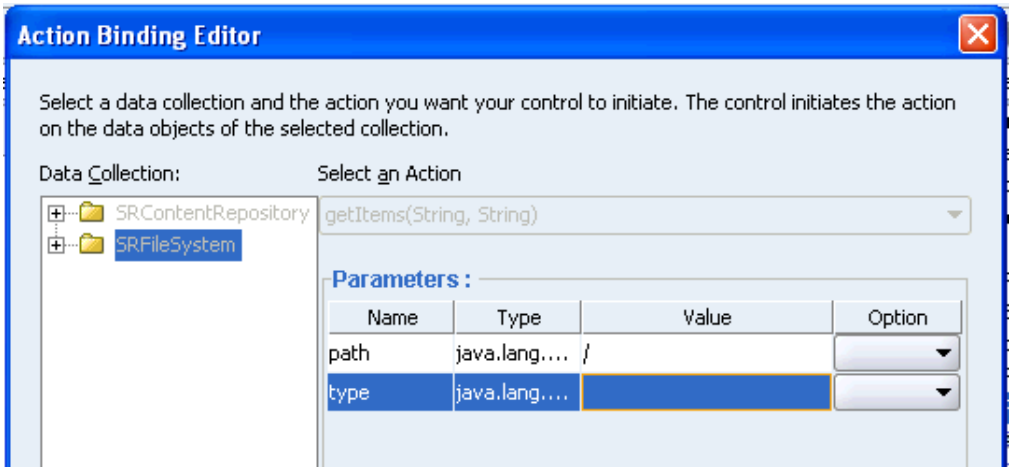


Note: The Action Binding Editor dialog box displays only the first time when you drop the return node. To modify or delete the path you specified the first time, edit the page definition by right-clicking the page and selecting Go to Page Definition.

5. To create a tree that displays everything under the base path, enter the slash (`/`) for the `path` parameter, as shown in Figure 5–57.

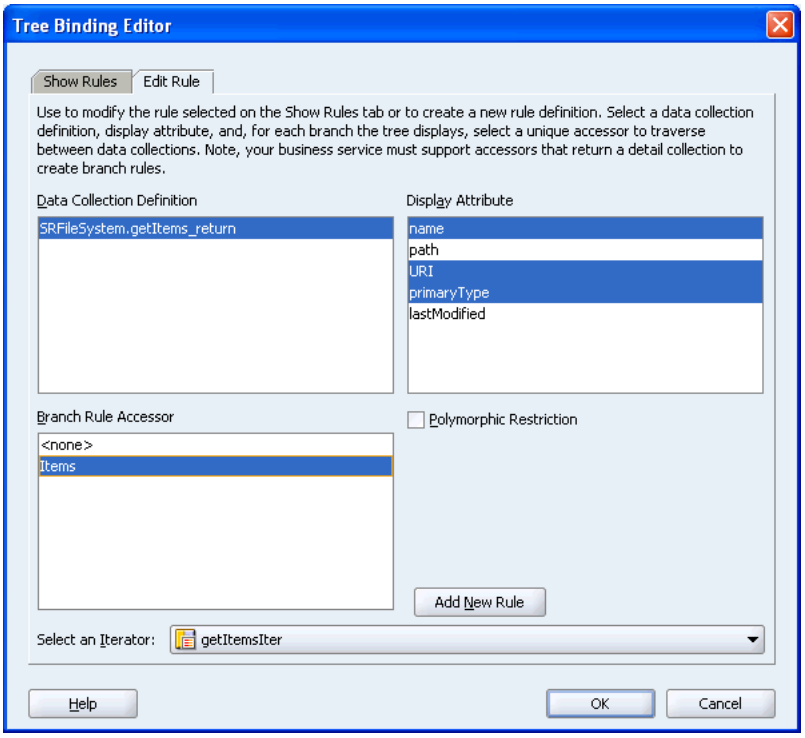
   Leave `type` parameter blank to show both files and folders.

Figure 5–57 Action Binding Editor



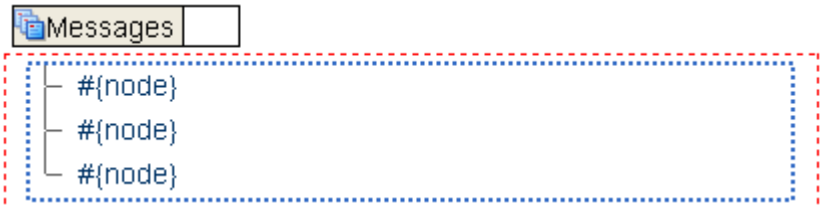
6. Click **OK**. The Tree Binding Editor dialog box is displayed.
7. Under Display Attribute, select **name**, **URI**, and **primary type**, as shown in Figure 5–58.

Figure 5–58 Tree Binding Editor



8. In the Branch Rule Accessor, select **Items**, and then click **Add New Rule.** You should see a message that a new rule is added to the Show Rules tab. Click **OK** to close the message.
9. Click **OK.** A tree displays on `MySRPage.jspx` that looks like Figure 5–59.

Figure 5–59 Tree for Navigating Folder Content



10. Run your page to display the results.

    When the page appears in your browser window, you should see a list of files and folders available through the `SRFileSystem` data control, as shown in Figure 5–53. Expand a branch to see the content in this subdirectory.

> **Note:** By default, the range size is `10`. To change the number of items displayed in the tree, edit the **RangeSize** property for the data control in the page definition file (`My_PagePageDef.xml`).

By default, the tree displays file and folder names as read-only text. You can add hyperlinks to the file names (not folders), but you cannot hide the folders as they are required for navigation through the tree. So, you can display the folder names as

read-only text. The next section describes the procedure to create hyperlinks to file names while keeping the folder names read-only.

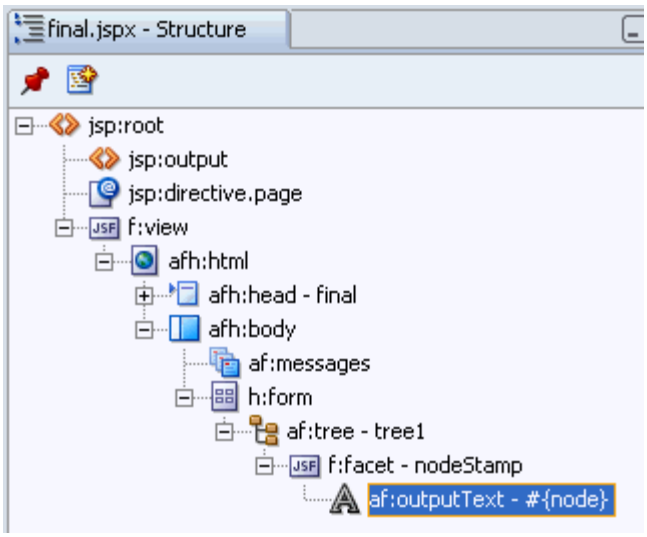
5.3.3.2 Displaying File Names As Hyperlinks

To create hyperlinks to file names and to keep folder names read-only, you need the `af:switcher` component with two facets: one for folders and one for files.

To use the Switcher component for folders and files, perform the following steps:

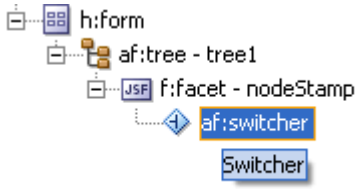
1. In the Structure window, navigate to `nodeStamp` to show the default display format `af:outputText - {#node}`, as shown in Figure 5–60.

Figure 5–60 Default Display Format for Trees



2. Right-click **af:outputText - #{node}** and click **Convert**. The Convert dialog box is displayed.
3. Select **ADF Faces Core** as the category of the component. Under the Select the item to be created box, select **Switcher**, and click **OK**. The Confirm Convert dialog box is displayed.
4. Click **OK** to complete conversion and display the Switcher in the Structure window, as shown in Figure 5–61.

Figure 5–61 Output Text Converted to a Switcher Component



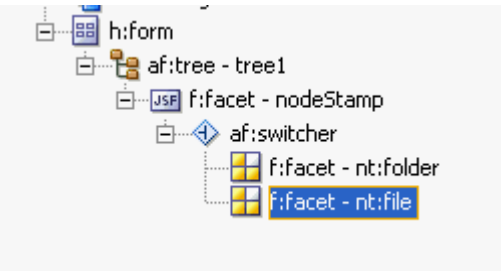
5. Double-click **af:switcher** to display the Switcher Properties dialog box.
6. In the **FacetName** field, enter the expression `#{node.primaryType}` and click **OK**.

7. Now insert two facets for the switcher. Right-click **af:switcher** and select **Insert Inside af:switcher**. From the **JSF Core** menu, select **Facet**. Name the first facet `nt:folder` and click **OK**. Folder names require no additional formatting, so you can display the node names as plain text.

   The facet is added, as shown in Figure 5–62.

   Repeat this step and add a second Facet named `nt:file`.

Figure 5–62 Switcher Component with Two Facets



8. Right-click **f:facet - nt:folder** and select **Insert Inside f:facet - nt:folder**. Select **ADF Faces Core** as the category, then select **OutputText**.
9. Double-click **af:outputText - outputText1** to display the Properties dialog box. In the **Value** field, enter the expression `#{node.name}`, and click **OK**.
10. Right-click **f:facet - nt:file** and select **Insert Inside f:facet - nt:file.** The Insert Item dialog box is displayed.
11. Select **ADF Faces Core** as the category of the component. Then, select **GoLink** from the Select the item to be created box, and click **OK**.
12. In the Structure window, double-click **af:goLink - goLink** to display the GoLink Properties dialog box.
13. In the **Text** field, enter the `#{node.name}` expression.
14. In the **Destination** field, enter the expression `#{node.URI}` and click **OK**.
15. Run the page to see a list of hyperlinked file names, as shown in Figure 5–54.

5.3.4 Adding Search Capabilities to Content Repositories

With help of two examples, this section describes the procedures to enable simple and advanced search functionality for the integrated content. The simple search enables users to search for content based on name or content fragments in specific locations. The advanced search enables users to search by attribute values of the content.

This section contains the following topics:

- Section 5.3.4.1, "What You Should to Know When Using Search Capabilities"
- Section 5.3.4.2, "Adding Simple Search Capabilities"
- Section 5.3.4.3, "Adding Advanced Search Capabilities"
- Section 5.3.4.4, "Adding Advanced Search Capabilities Using the Data Control based on Oracle WebCenter Adapter for EMC Documentum"

5.3.4.1 What You Should to Know When Using Search Capabilities

Consider the following points while adding search capabilities:

- The functionality or how certain operations work depends on the implementation of the adapter and the underlying repository. While read and query operations are similar, full text search works differently. For example, the file system adapter looks for the provided string in the binary content of the files, but OracleAS Portal and Oracle Content DB adapters perform full text index operation on their documents using a database functionality.
- The file system, OracleAS Portal, and Oracle Content DB adapters do not support search based on the `primaryType` attribute. The only supported way to search based on type is through the element (`*`, type) construct.
- If you use the OracleAS Portal adapter, then the behavior of search functionality varies depending on whether Oracle Text is set to On or Off. The Oracle Text can enable or disable full text search of items and their contents. It is advised that Oracle Text is set to On.
- The OracleAS Portal does not support translations and will only return content in the base language of a page group. Search across multiple page groups with different base languages is not supported.

5.3.4.2 Adding Simple Search Capabilities

To enable the search function, perform the following steps:

Steps 1 to 9 enable you to create a simple search that will enable users to use the input field to perform name-based search. This search will also enable users to use `%` for wildcard search. For instance, if `%.ppt%` is used as input, the search will return files with the `.ppt` extension.

1. In the Applications Navigator, double-click your `.jspx` page to open it.
2. In the Data Control Palette, expand the **search** node.
3. To enable users to execute a search by clicking a button, drag and drop the **search** node on your `.jspx` page. The **Create** menu is displayed. Select **Methods** and then select **ADF Command Button**. The Action Binding Editor dialog box is displayed.

   Note: The Action Binding Editor dialog box displays only the first time when you drop a node. To modify or delete the path that you specified the first time, edit the page definition by right-clicking the page and selecting Go to Page Definition.

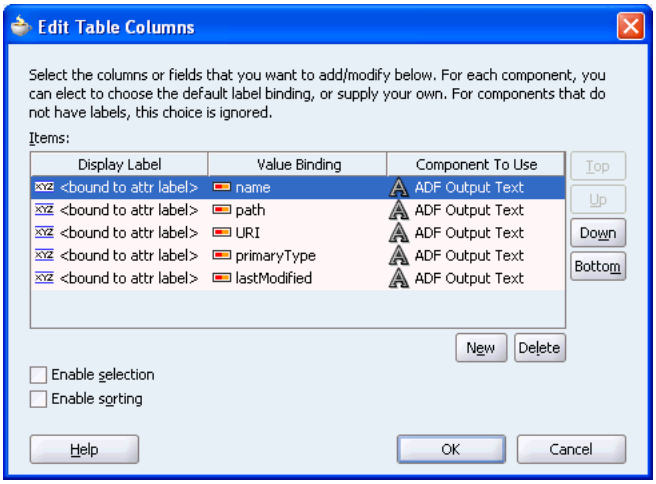
4. For the path parameter, enter `/` as the value, as shown in Figure 5–63.
5. To enable a search from the location specified in the path attribute and its subdirectories, enter `true` as the value for the `isRecursive` parameter, as shown in Figure 5–63, and click **OK**.

Figure 5–63 Action Binding Editor - search

6. To enable search by file name, expand **search** and **parameters** and drag and drop the **namePattern** attribute onto the page. The **Create** menu is displayed. Select **Texts** and then select **ADF Input Text**.

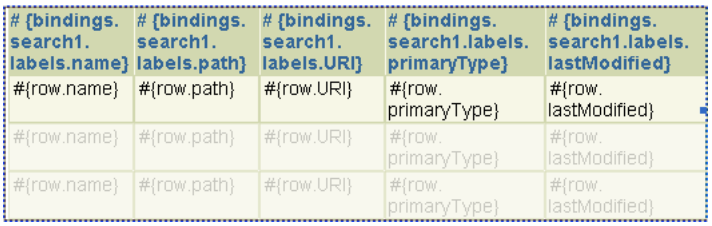
7. To enable the display of search results in a read-only table, drag and drop the **Return** node onto the page. The **Create** menu appears. Select **Tables** and then select **ADF Read-Only Table**. The Edit Table Columns dialog box is displayed, as shown in Figure 5–64.

Figure 5–64 Edit Table Columns



8. Click **OK**. A table similar to Figure 5–65 is displayed.

Figure 5–65 Table with Four Columns - search



9. Run this page in a browser and search for `%jpg%`. All `.jpg` files from your repository display, as shown in Figure 5–66.

Figure 5–66 Search Results for .jpg Files

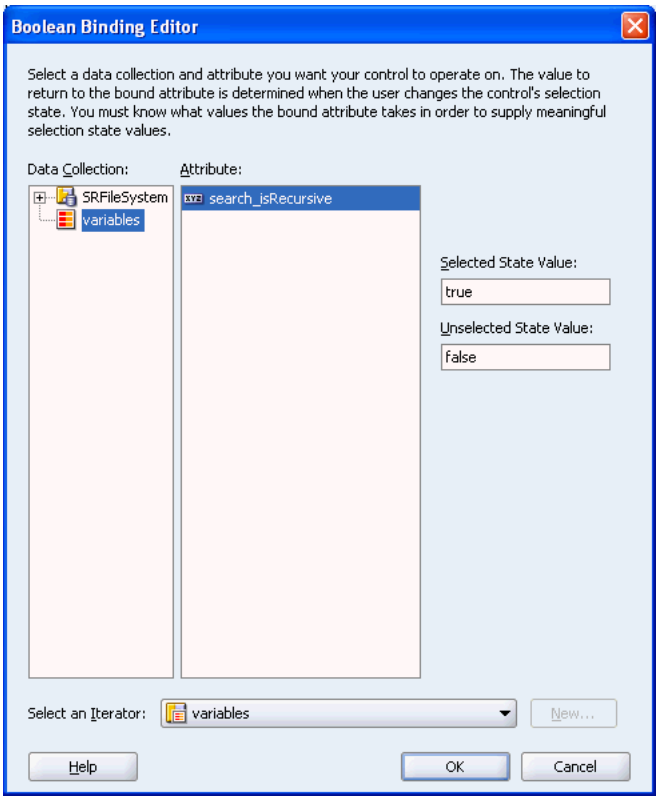
search %jpg%

Name	Path	URL	Type	Last Modified Date
query.jpg	/query.jpg	/get/conn/SRFileSystem/path/query.jpg	nt:file	5/8/2006
consult.jpg	/consult.jpg	/get/conn/SRFileSystem/path/consult.jpg	nt:file	5/8/2006
acmecenter.jpg.bak	/acmecenter.jpg.bak	/get/conn/SRFileSystem/path/acmecenter.jpg.bak	nt:file	4/24/2006
edit.jpg	/edit.jpg	/get/conn/SRFileSystem/path/edit.jpg	nt:file	5/8/2006
design.jpg	/design.jpg	/get/conn/SRFileSystem/path/design.jpg	nt:file	5/8/2006
onlinehelp.jpg	/onlinehelp.jpg	/get/conn/SRFileSystem/path/onlinehelp.jpg	nt:file	5/8/2006
view.jpg	/view.jpg	/get/conn/SRFileSystem/path/view.jpg	nt:file	5/8/2006
acmecenter.jpg	/acmecenter.jpg	/get/conn/SRFileSystem/path/acmecenter.jpg	nt:file	5/8/2006

Steps 10 to 12 extend the simple search such that users can enter a value for the path and can select the check box to determine whether the search should be performed at the folder tree level or should be limited to the specified location (`=path`).

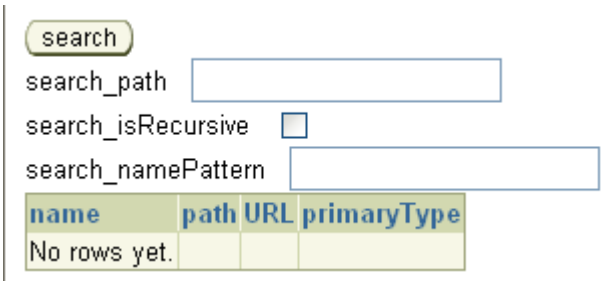
10. Drag and drop the **path** attribute onto the page. The **Create** menu is displayed. Select **Texts** and then select **ADF Input Text**.
11. To create a check box for users to select if the search should be performed from the location specified in the path attribute and its subdirectories, drag and drop the **isRecursive** parameter onto the page. The **Create** menu is displayed. Select **Single Selections** and then select **ADF Select Boolean Checkbox**. The Boolean Binding Editor dialog box is displayed.
12. Enter `true` in the **Selected State Value** field and `false` in the **Unselected State Value** field, as shown in Figure 5–67, and click **OK**.

Figure 5–67 Boolean Binding Editor



13. Run the page in a browser. The output should be similar to Figure 5–68.

Figure 5–68 Search Fields in a Browser



In the next section, this simple search will be extended into an advanced search.

5.3.4.3 Adding Advanced Search Capabilities

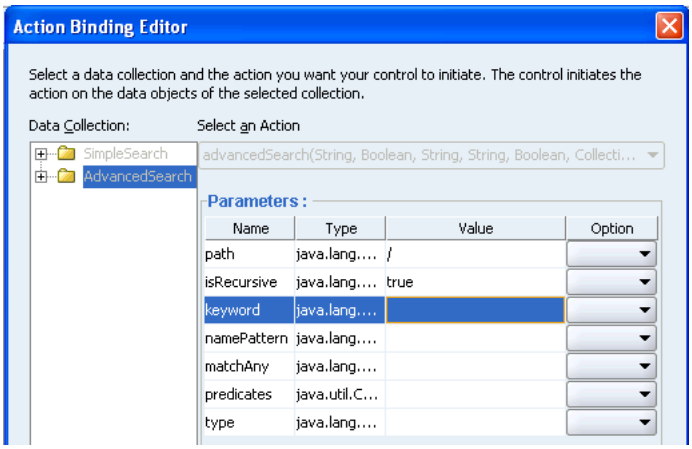
To enable the advanced search function, perform the following steps:

1. In the Applications Navigator, double-click the `.jspx` page in which you created the simple search function.
2. To enable users to execute an advanced search by clicking a button, from the Data Control Palette drop the **advancedSearch** node onto the page. The **Create** menu is displayed. Select **Methods** and then select **ADF Command Button**. The Action Binding Editor dialog box is displayed.

> **Note:** The Action Binding Editor dialog box displays only the first time when you drop a node. To modify or delete the path you specified the first time, edit the page definition by right-clicking the page and selecting Go to Page Definition.

3. For the path parameter, enter `/` as the value, as shown in Figure 5–69.
4. To enable a search from the location given in the path attribute and its subdirectories, enter `true` as the value for the `isRecursive` parameter, as shown in Figure 5–69, and click **OK**.

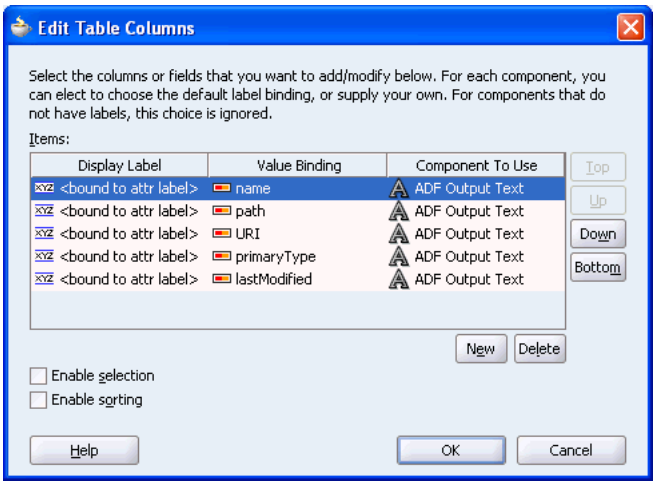
Figure 5–69 Action Binding Editor - advancedSearch



5. To enable the display of advanced search results in a read-only table, drag and drop the **Return** node of advancedSearch on your `.jspx` page. The **Create** menu

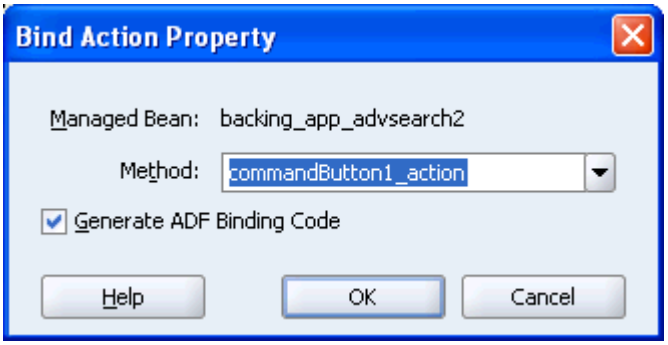
appears. Select **Table** and then select **ADF Read-only Table**. The Edit Table Columns dialog box is displayed, as shown in Figure 5–70.

Figure 5–70 Edit Table Columns



6. Change the display label, if you want, and click **OK**.
7. To create an input field for the parameter value, from the Component Palette, select **ADF Faces Core**, and then drag and drop an **SelectInputDate** control onto the page.
8. Double-click the **advancedSearch** button. The Bind Action Property dialog box is displayed. Ensure that the Generate ADF Binding Code check box is selected, as shown in Figure 5–71. This generates a skeleton in the backing bean where you will add the code to build the predicates.

Figure 5–71 Bind Action Property



9. Click **OK**. The `commandButton1_action()` method displays in the source mode.
10. To include necessary classes that create the code, include the following entries in the Source mode, if they do not exist already:

```
import oracle.vcr.datacontrol.search.Operator;
import oracle.vcr.datacontrol.search.Predicate;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.Calendar;
```

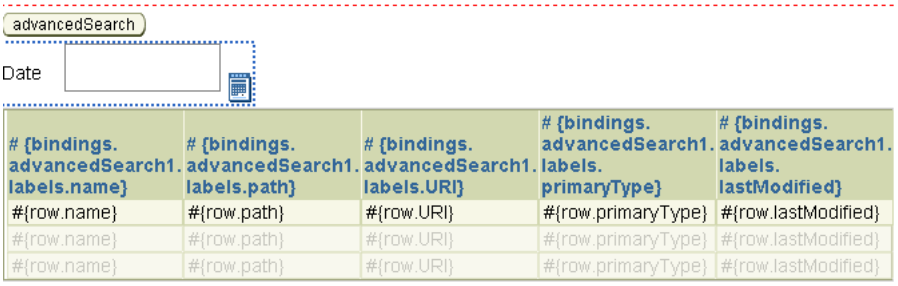
11. Now, modify the `commandButton1_action()` to look like the following:

```
public String commandButton1_action() {
        // Get the bindings.
        BindingContainer bindings = getBindings();
        // Set Date format
         Date value = (Date) selectInputDate1.getValue();
         Calendar cal = new GregorianCalendar();
         cal.setTime(value);
       // Create a predicate out of the value of your inputDate, the operation
and the attribute you want to search for.
        Predicate p = new Predicate("jcr:content/jcr:lastModified",
Operator.GREATER_THAN, cal);
        ArrayList al = new ArrayList(1);
        // Add this predicate to the list of predicates for the search.
       // You can have multiple predicates and use the "MatchAll" attribute to
define
        // whether they should be concatenated by a logical AND or OR.
        al.add(p);
        // Get the advancedSearch operation binding.
        OperationBinding operationBinding =
getBindings().getOperationBinding("advancedSearch");
        // Assign the list of predicates to the "predicates" attribute.
        operationBinding.getParamsMap().put("predicates", al);
        // Execute the operation.
        Object result = operationBinding.execute();
        return null;
    }
```

> **Note:** To retrieve the JCR paths of item attributes, run the `getAttributes` method on the required items.

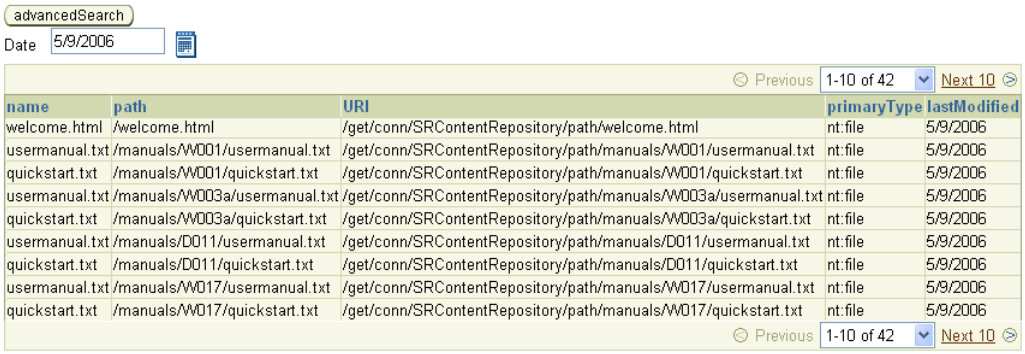
In the Design mode, the page should look like Figure 5–72.

Figure 5–72 Advanced Search Page in the Design Mode



12. Run the page in a browser and search for a document using its last modified date. Figure 5–73 shows search results based on the last modified date entered as the search criterion.

Figure 5–73 Advanced Search Results



5.3.4.4 Adding Advanced Search Capabilities Using the Data Control based on Oracle WebCenter Adapter for EMC Documentum

This section shows through an example how to add advanced search capabilities based on the Oracle WebCenter adapter for EMC Documentum.

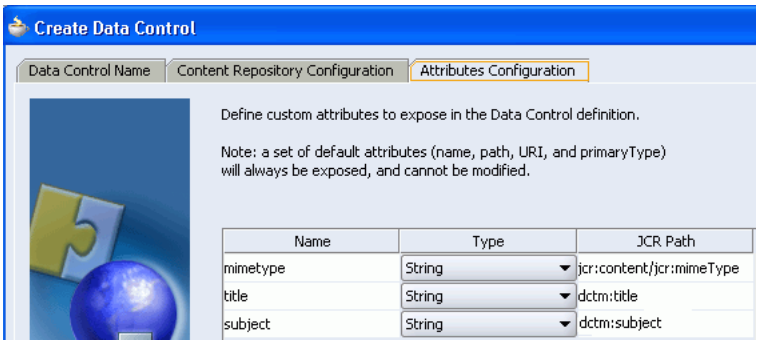
To add advanced search capabilities, perform the following steps:

1. Add custom attributes of `String` type to your data control.
   a. To add custom attributes, open the `DataControls.dcx` file under Applications Navigator. Then right-click the data control under the Structure pane and select **Edit**. The Create Data Control wizard is displayed.
   b. Under the Attributes Configuration tab, click **Add** to include the attributes listed in the following table:

Attribute Name	JCR Path
mimeType	jcr:content/jcr:mimeType
title	dctm:title
subject	dctm:subject

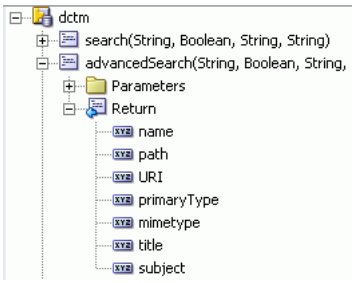
The Attributes Configuration tab looks like Figure 5–74.

Figure 5–74 Custom Attributes



The new attributes show under Data Control Palette, as shown in Figure 5–75.

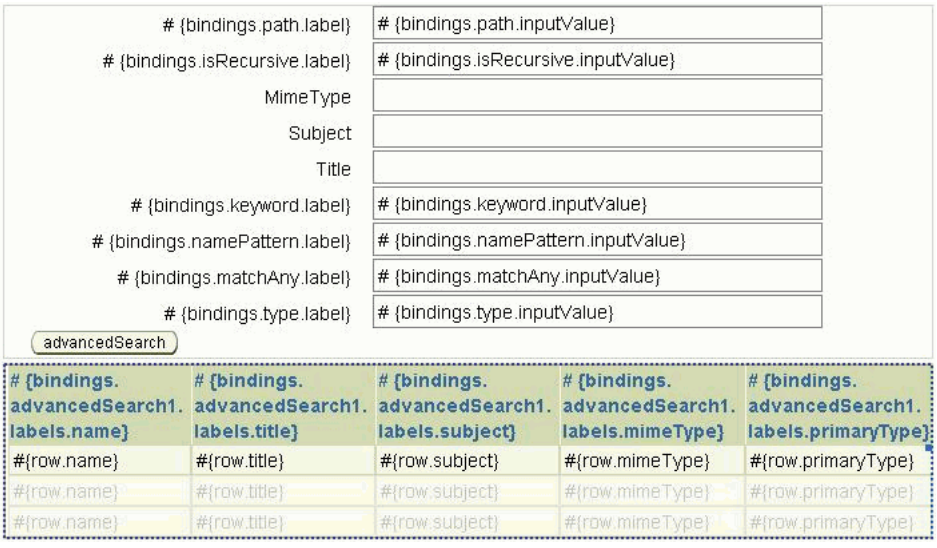
Figure 5–75 Custom Attributes under



2. Open your `.jspx` page.
3. From the Data Control Palette, drag and drop **advancedSearch** method onto the page. Select **Parameters** and then select **ADF Parameter Form**. The Edit Form Fields dialog box is displayed.
4. Select the `predicates` value binding and click **Delete**. This value binding is deleted because predicates value cannot be bound to a single input text control on the page. Predicates will be built separately in this procedure, and the value will be bound to the data control programmatically.
5. In the Data Control Palette, expand **advancedSearch** and **Return** nodes.
6. Under Components Palette, select **InputText** from the ADF Faces Core and drop it under `af:panelForm` in the Structure pane. Then, rename it to MimeType.

   Repeat this step to include `Title` and `Subject` as `InputText`.
7. From the Data Control Palette, drag and drop the **Return** node onto the page. From the Create menu, select **Tables** and then select **ADF Read-only Table**. The Edit Table Columns dialog box is displayed.
8. Click **OK**. The ADF Read-only Table is added in the design mode. The design mode looks like Figure 5–76.

Figure 5–76 advancedSearch in the Design View



9. In the design mode, double-click the **advancedSearch** button. The Bind Action Property dialog box is displayed.

10. Click **OK** to generate the method. The source mode is displayed with the method.
11. Replace the existing `commandButton1_action()` method with the example method shown in Figure 5–10.

Example 5–10 commandButton1_action() Method

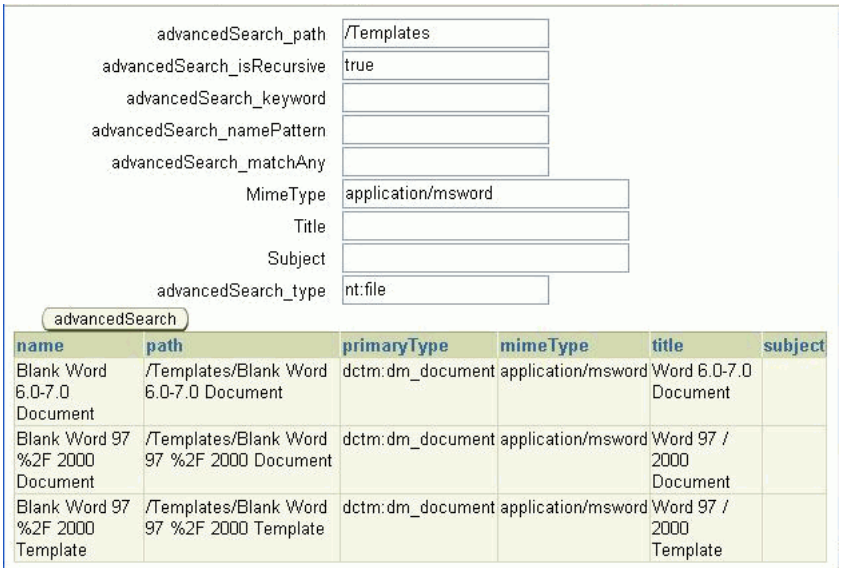
```
public String commandButton1_action() {
        // Add in predicates binding to bindings container
        ArrayList predicates = new ArrayList();
        String _mimeType = (String) mimeType.getValue();
        if ( _mimeType != null && ! "".equals(_mimeType)) {
          predicates.add(
           new Predicate("jcr:content/jcr:mimeType", Operator.LIKE, _mimeType));
        }
        String _title = (String) title.getValue();
        if ( _title != null  && ! "".equals(_title)) {
          predicates.add(
           new Predicate("dctm:title", Operator.LIKE, _title));
        }
        String _subject = (String) subject.getValue();
        if ( _subject != null && ! "".equals(_subject)) {
          predicates.add(
           new Predicate("dctm:subject", Operator.LIKE, _subject));
        }
        BindingContainer bindings = getBindings();
        OperationBinding operationBinding =
         bindings.getOperationBinding("advancedSearch");
        operationBinding.getParamsMap().put("predicates", predicates);
        Object result = operationBinding.execute();
        if (!operationBinding.getErrors().isEmpty()) {
        // Add error handling
              return null;
        }
        return null;
    }
```

12. Add the following imports to the class:

```
import java.util.ArrayList;
import oracle.vcr.datacontrol.search.Operator;
import oracle.vcr.datacontrol.search.Predicate;
```

13. Display the page in a browser by right-clicking the page and selecting **Run**.
14. Perform a search as shown in Figure 5–77.

Figure 5–77 advancedSearch in a Browser



5.3.5 Configuring Custom Attributes in Oracle Content DB

This section demonstrates, using examples, how to create attributes in Oracle Content DB and add them as custom attributes while creating the Oracle Content DB data control. This section discusses the following:

- Section 5.3.5.1, "Creating Attributes in Oracle Content DB"
- Section 5.3.5.2, "Adding Custom Attributes to the Oracle Content DB Data Control"

5.3.5.1 Creating Attributes in Oracle Content DB

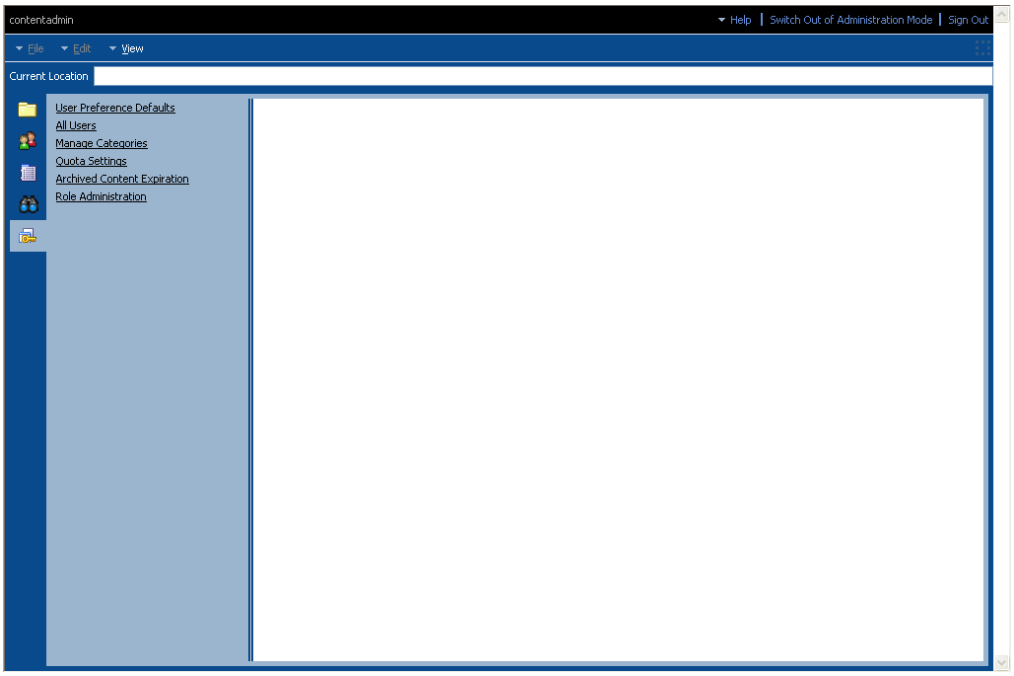
In this section you will create categories and then add attributes to them. These attributes will then be added as custom attributes when you create the Oracle Content DB data control.

> **See Also:** *Oracle Content Database for Oracle WebCenter Suite Application Administrator's Guide*

To create categories and attributes, perform the following steps:

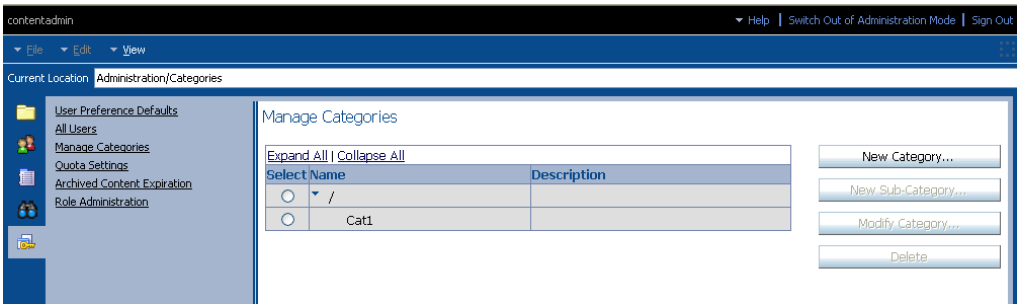
1. Log in to Oracle Content DB and switch to administration mode by clicking the **Switch to Administration Mode** button on the top-right corner of the window. The window shown in Figure 5–78 is displayed.

Figure 5–78 Administration Mode in Oracle Content DB



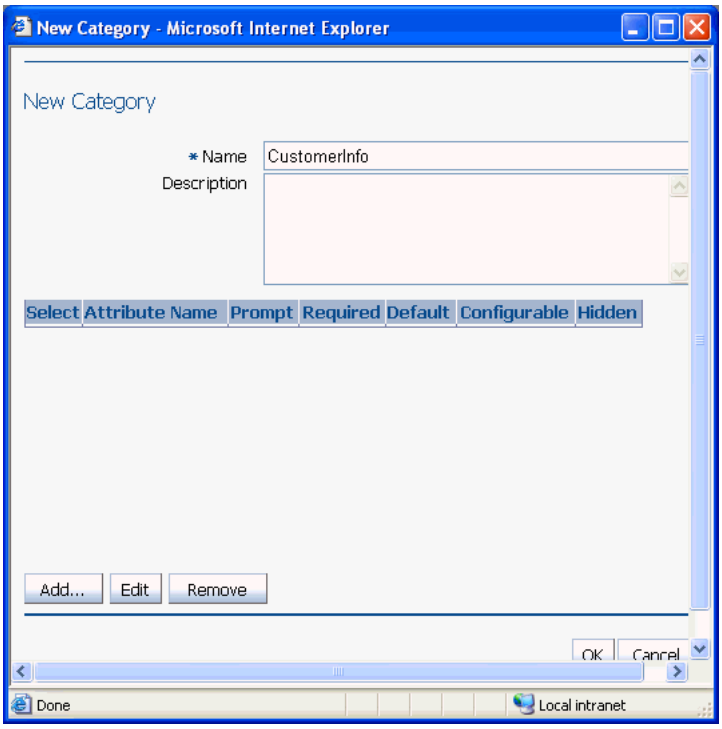
2. Click **Manage Categories** on the left pane. The Manage Categories window is displayed, as shown in Figure 5–79.

Figure 5–79 Manage Categories



3. Click **New Category**. The New Category window is displayed. Enter a name for the new category, for example, `CustomerInfo`, as shown in Figure 5–80, and click **Add**.

Figure 5–80 New Category



4. Click **Add**. The Add Attributes window is displayed. Enter a name for the new attribute, for example, `CustomerName`, as shown in Figure 5–81 and and select a datatype for the new attribute from the **Type** dropdown, and then click **OK**.

   You can add more attributes, for example, `Contact`, `CustomerID`, and so on, as shown in Figure 5–82.

Figure 5–81 Add Attribute

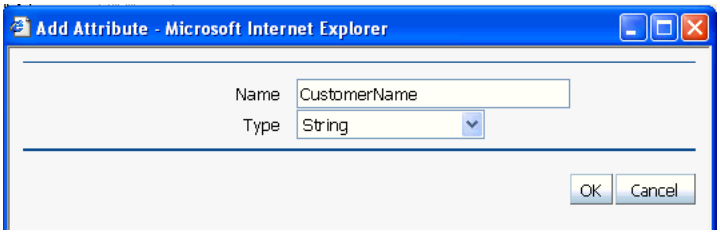
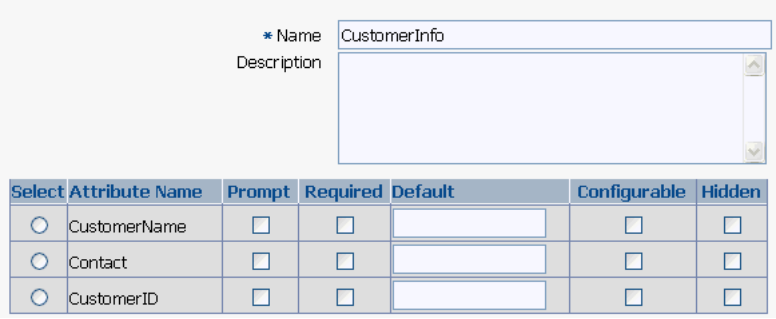
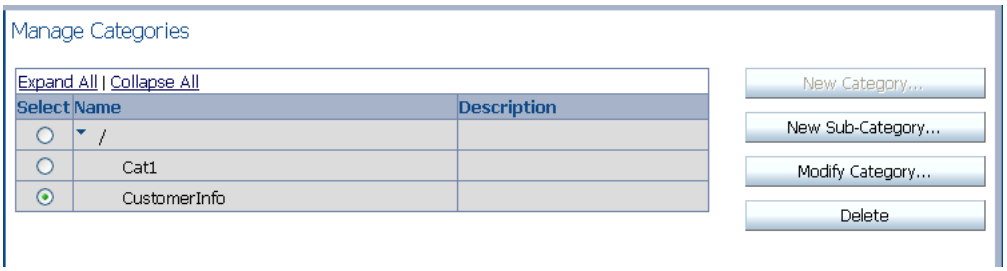


Figure 5–82 Attributes of the CustomerInfo Category



5. To add subcategories, click **Manage Categories** on the left pane. The Manage Categories window is displayed, as shown in Figure 5–83.

Figure 5–83 Manage Categories



6. Select the **CustomerInfo** option, as shown in Figure 5–83, and then click **New Sub-Category**. The New Category window is displayed.

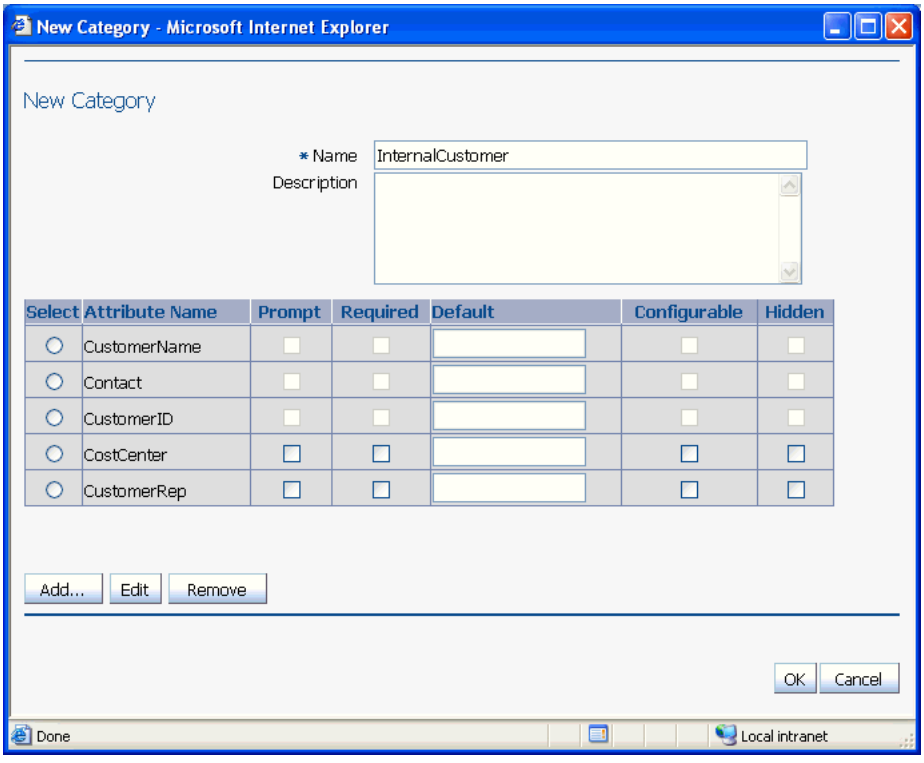
7. To add subcategories, click **Add**. Enter a name for the subcategory, for example, `InternalCustomer`, and click **OK**. The Add Attributes window is displayed.

   Subcategories inherit main properties from the parent but can have additional properties.

8. Enter a name for the new attribute, for example, `CustomerRep`, and click **OK**.

   You can add more attributes for the `InternalCustomer` subcategory, for example, `CostCenter`, as shown in Figure 5–84.

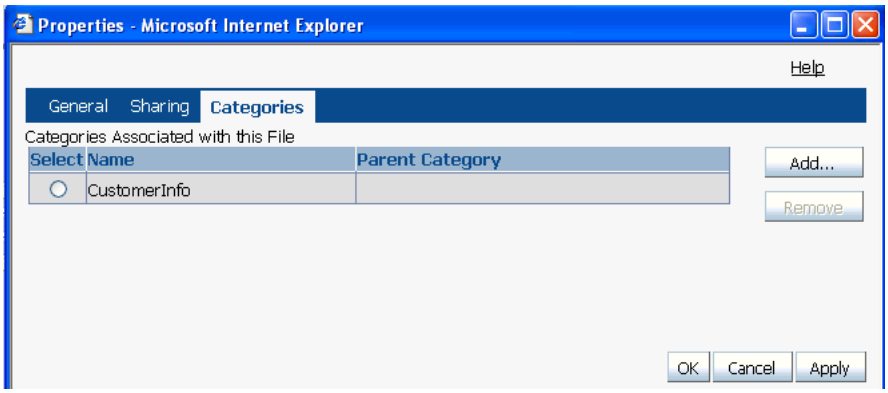
Figure 5–84 New Category



9. Now switch out of administration mode by clicking **Switch Out of Administration Mode**.

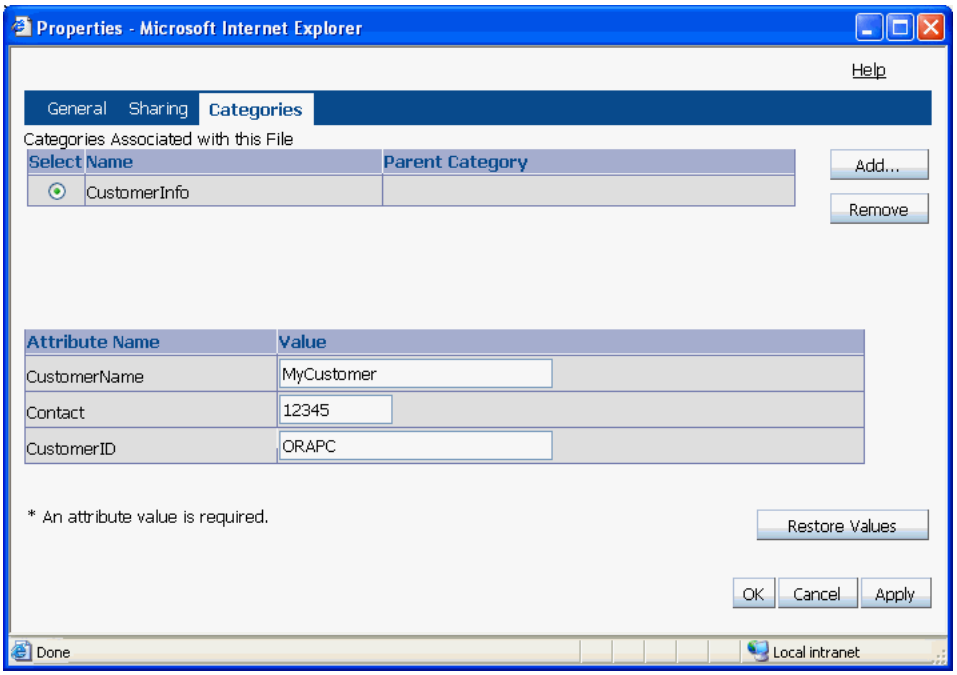
10. To assign a category to a file, navigate to the file by clicking the Folder icon on the left pane.
11. Right-click the file to which you want to assign the category, then select **Properties** from the menu. The Properties window is displayed, as shown in Figure 5–85.

Figure 5–85 Properties Window



12. Click the **Categories** tab and then click **Add**. The Add Categories window is displayed.
13. Select the new category option, for example, `CustomerInfo`, and click **OK**.
14. Now provide values for the new attributes, as shown in Figure 5–86.

Figure 5–86 Properties Window with New Attributes



15. Click **Apply** and then click **OK**.

5.3.5.2 Adding Custom Attributes to the Oracle Content DB Data Control

In this section, you will add attributes that you created in Section 5.3.5.1, "Creating Attributes in Oracle Content DB" as custom attributes to your Oracle Content DB data control.

To add custom attributes to your Oracle Content DB data control, perform the following steps:

1. Perform steps 1 to 9 from Section 5.2.4.2, "Creating a Content Data Control Based on the Oracle Content DB Adapter".
2. In the Attributes Configuration page, click **Add** three times to add three blank rows for the `CustomerName`, `CustomerID`, and `Contact` attributes that you created earlier.
3. Enter a name and JCR path for each attribute. The JCR path should be in the following format:

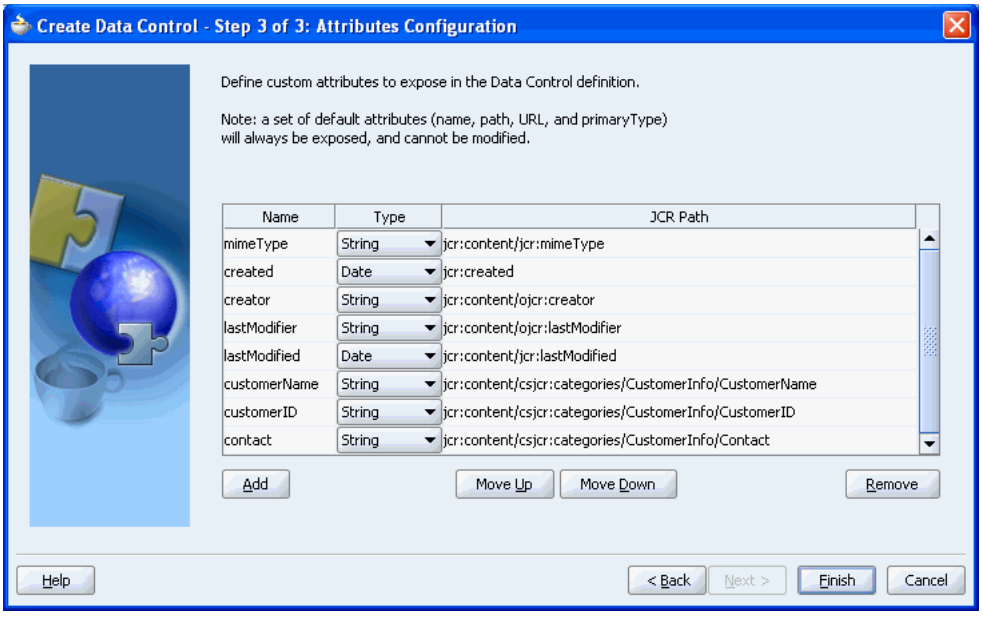
   `jcr:content/csjcr:categories/`*`CategoryName`*`/`*`AttributeName`*

The JCR paths for `CustomerName`, `CutomerID`, and `Contact` attributes are:

```
jcr:content/csjcr:categories/CustomerInfo/CustomerName
jcr:content/csjcr:categories/CustomerInfo/CustomerID
jcr:content/csjcr:categories/CustomerInfo/Contact
```

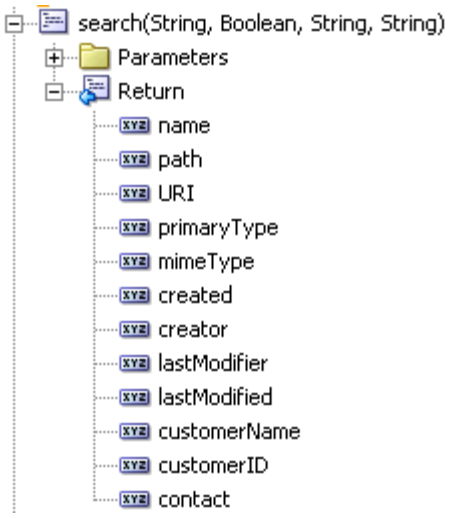
Figure 5–87 shows the `customerName`, `cutomerID`, and `contact` custom attributes in the Attributes Configuration page.

Figure 5–87 Attributes Configuration Page



4. Click **Finish** to complete the creation of the data control.
5. In the Data Control Palette, expand the Oracle Content DB data control that you created. You should see these custom attributes, as shown in Figure 5–88.

Figure 5–88 Custom Attributes



5.3.6 Creating Clickable Images Using Custom Attributes

Web applications often contain images, such as product pictures, that support the information provided on the page. These images frequently link to other parts of the Web site. One way of managing these images is to store them in your content management system and tag them with additional information using custom attributes.

This example shows how you can use these images and the information in the custom attributes to enhance the experience of your WebCenter application. The following steps show you how to do so using Oracle Content DB as the content repository. You can, however, follow the steps after creating the data control using any content repository, using the appropriate naming convention for the custom attribute.

This section contains the following:

- Section 5.3.6.1, "Creating a Custom Attribute in Oracle Content DB"
- Section 5.3.6.2, "Creating a Data Control based on Oracle Content DB Adapter"
- Section 5.3.6.3, "Creating a Clickable Image and Table of Clickable Images"

5.3.6.1 Creating a Custom Attribute in Oracle Content DB

In Oracle Content DB, you can create custom attributes for the images you want to add to your WebCenter application. In Oracle Content DB, you organize custom attributes into categories, then assign them to items or folders in the repository.

To create the category and custom attribute, perform the following steps:

1. Ensure that you have uploaded the image(s) you want to use with your WebCenter application to Oracle Content DB.
2. In Oracle Content DB, click **Switch to Administration Mode** to display the administration options.
3. Click **Manage Categories** to create or modify categories and their associated custom attributes.
4. Click **New Category** to display the New Category dialog box.

5. In the New Category dialog box, enter a name for the new category in the **Name** field. For the purposes of this example, enter `ImageAttributes`.
6. Click **Add** to add a new attribute to this category.
7. In the Add Attribute dialog box, in the **Name** field, enter `TargetURL` and choose **String** from the Type dropdown list.
8. Click **OK** to return to the New Category dialog box. The attributes you add display below the category name.

   Here, you can set options such as whether the user should be prompted for this attribute, whether the attribute is required, the default value, and whether the attribute is configurable or hidden. Once you have defined your attributes and categories, you can apply attributes to images in the content repository.
9. Click **OK** to return to the Administration page.
10. Click **Switch Out of Administration Mode** to view the File Management page. On the File Management page, you find the images to which you want to apply the custom attributes.
11. Right-click the image you want to modify, then choose **Properties** from the context menu.
12. Click the **Categories** tab.
13. Click **Add** to add the category you created earlier.

> **Note:** The list of categories is defined on the folder level. A folder can either inherit the list of categories from its parent folder or the administrator can manually define the list of categories.

14. Select the **ImageAttributes** category to display its custom attributes. Here, you can provide values for the attributes contained in the ImageAttributes category, such as the URL to which you want the image to link.
15. In the TargetURL field, enter a URL, such as `http://www.oracle.com/technology`.
16. Click **Apply**, then **OK** to apply the custom attribute to your image.

5.3.6.2 Creating a Data Control based on Oracle Content DB Adapter

After you have set up custom attributes for your images in Oracle Content DB, you must create a data control using the Oracle Content DB adapter so that you can add the images to your WebCenter application.

For information about creating a data control, see Section 5.2.4, "Configuring a Content Data Control Based on the Oracle Content DB Adapter".

To create the data control, perform the following steps:

1. In the Create Data Control Wizard, enter a name for your data control (for example, `Images`), then enter the connection information for your instance of Oracle Content DB.
2. On Step 3 of the Create Data Control Wizard, on the Attributes Configuration page, add the attribute for the image, `TargetURL`.

3. Custom attributes are addressed by path in the JCR node tree. In this example, the attribute carrying the necessary information is called **TargetURL** in the category **ImageAttributes**.

   In the JCR Path field for the TargetURL attribute, enter:

   ```
   jcr:content/csjcr:categories/ImageAttributes/TargetURL
   ```

   Note: If you do not know the exact syntax of this path for a particular attribute, create a simple JSPX page that uses the `getAttributes` method of the Data Control on an item where you know the attribute exists. The resulting table will show the attribute path and the value of the attributes for this item. You can then reenter the Create Data Control Wizard and add all the custom attributes by simply cutting and pasting the attribute path.

4. Click **Finish**.

5.3.6.3 Creating a Clickable Image and Table of Clickable Images

Once you have set up the custom attribute in Oracle Content DB and created the data control, you can add the image to your WebCenter application and enable the image to be clicked. You can also create a table of clickable images to publish all images from a folder.

Creating a Clickable Image

To create a clickable image, perform the following steps:

1. In Oracle JDeveloper, open your JSPX page.
2. In the Data Control Palette, expand that data control you created in Section 5.3.6.2, "Creating a Data Control based on Oracle Content DB Adapter" and expand the **search** method.
3. Under Return, click **URI**, then drag the URI attribute onto the page.
4. From the Create list, choose **Link,** then **Object Image** to display the Action Binding Editor.
5. In the Action Binding Editor, enter the input parameters for the search method. In our example, set the values for **Path** and **NamePattern**. These values should be the path where the image resides and the image filename.
6. Click **OK**.
7. In the Structure Pane, right-click the **ObjectImage** you created and click **Insert Before**, then **Browse**.
8. From the category dropdown list, choose **ADF Faces Core.**
9. Choose **GoLink** from the list of items, then click **OK** to insert the GoLink into the page before the ObjectImage.
10. In the Structure Pane, drag and drop **ObjectImage** below the new GoLink so that the ObjectImage is a child of the GoLink.
11. View the Page Definition for your JSPX page. To do so, right-click your JSPX page then choose **Go to Page Definition** from the context menu.
12. In the Structure Pane, right-click **Bindings**, choose **Insert Inside**, then select **attributeValues** from the context menu.

13. In the Attributes Binding Editor, expand the data control node (in this example, the **Images** node), then the **search** node, then select **Return** to view a list of available attributes.
14. In the Attribute list, select **TargetURL** and click **OK** to create the attribute binding.
15. Return to your JSPX page from the Page Definition.
16. In the Structure Pane, right-click **goLink** and choose **Properties** from the context menu.
17. In the Properties dialog box, delete the text in the **Text** field. The image in this case will be clickable, not the text.
18. Next to the Destination field, click **Bind**.
19. Click **ADF Bindings**, then **Bindings** to find a node for the attributeValue you just created.
20. Search for the AttributeValues you just created (TargetURL), then click the arrow to add the binding to the Expression.
21. Click **OK**.

    The resulting code will look something like the following:

```
<f:verbatim>
  <p>
    <af:goLink binding="#{backing_imageLink.goLink1}"
   id="goLink1"
 destination="#{bindings.getItems_returntargetURL.inputValue}">
      <af:objectImage binding="#{backing_imageLink.outputText1}"
                      id="outputText1"
                      source="#{bindings.getItems_returnURI.inputValue}"/>
    </af:goLink>
  </p>
</f:verbatim>
```

22. Run the page. When you click the image, the target URL you used for the attribute should display its contents.

Creating a Table of Clickable Images

You can also create a table of clickable images to publish all images from a folder in Oracle Content DB.

To create a table of clickable images, perform the following steps:

1. Create a new JSPX page in your WebCenter application.
2. From the Data Control palette, drag and drop the **Return** of the getItems method as an ADF Read-Only Table onto the page.

> **Note:** When you drag and drop the Return node onto your page, a context menu appears where you can choose **Create > Tables > ADF Read-Only Table**. Doing so displays the Action Binding Editor.

3. In the Action Binding Editor, next to the **path** parameter, specify the path to the location of the image, then click **OK**.
4. In the Edit Table Columns dialog box that displays, remove all columns except for the URI column, then click **OK** to generate the table.

5. In the Structure Pane, right-click **outputText**, then choose **Convert** to create an object image out of the selected item.
6. Open the Properties dialog box for the converted item and, next to the Source input field, click **Bind**.
7. In the Structure Pane, right-click **goLink** and choose **Properties** from the context menu.
8. In the Action Binding Editor, expand **JSP Objects > Row** and select the **URI** to create the binding expression, then click **OK**.
9. In the Structure Pane, right-click **objectImage** and choose **Insert Before > ADF Core** from the context menu.
10. From the list of available components, select **GoLink**, then click **OK**.
11. In the Structure Pane, drag and drop the objectImage to become a child of the goLink.
12. In the Properties dialog box, delete the value from the **Text** field. The image in this case will be clickable, not the text.
13. Next to the Destination field, click **Bind**.
14. In the Binding dialog box, expand **JSP Objects > Row** and select **TargetURL** to create the binding expression, then click **OK**.
15. In the Structure Pane for your page, right-click **ObjectImage** and choose **Properties** from the context menu.
16. On the Common Properties tab, next to the source attribute, click **Bind** to define the expression language for the image source.
17. Click **OK**, then run your page. You should see a table of all the images in the directory, and each image should link to its respective Web site.

Note: You can also search for items based on certain custom attributes and render those on your page. For example, you could display product images for certain categories of pages or filter images from mixed-type result sets. To do so, follow the steps in Creating a Table of Clickable Images but use the advancedSearch method instead of the getItems method and manually build a query predicate.

5.4 Configuring Data Controls Based on Stellent Content Server

In addition to Oracle Content DB, you can integrate other content management systems, such as Stellent. Stellent provides numerous ways for you to integrate its functionality into custom applications. These integration methods include a Web services API as well as a J2EE API (which is part of the Content Integration Suite). For more information about Stellent integration in custom applications, please see *Getting Started with the Stellent Developer's Kit (SDK)*. This guide provides the background information you will need and gives pointers to other, more detailed documents.

As WebCenter Framework is based on Oracle ADF, you can leverage Web services and custom data controls to simplify development, as described in the following sections:

- Section 5.4.1, "Configuring Web Service Data Controls Based on Stellent Content Server"

- Section 5.4.2, "Integrating Content from Stellent Content Server Using a Custom Data Control"

5.4.1 Configuring Web Service Data Controls Based on Stellent Content Server

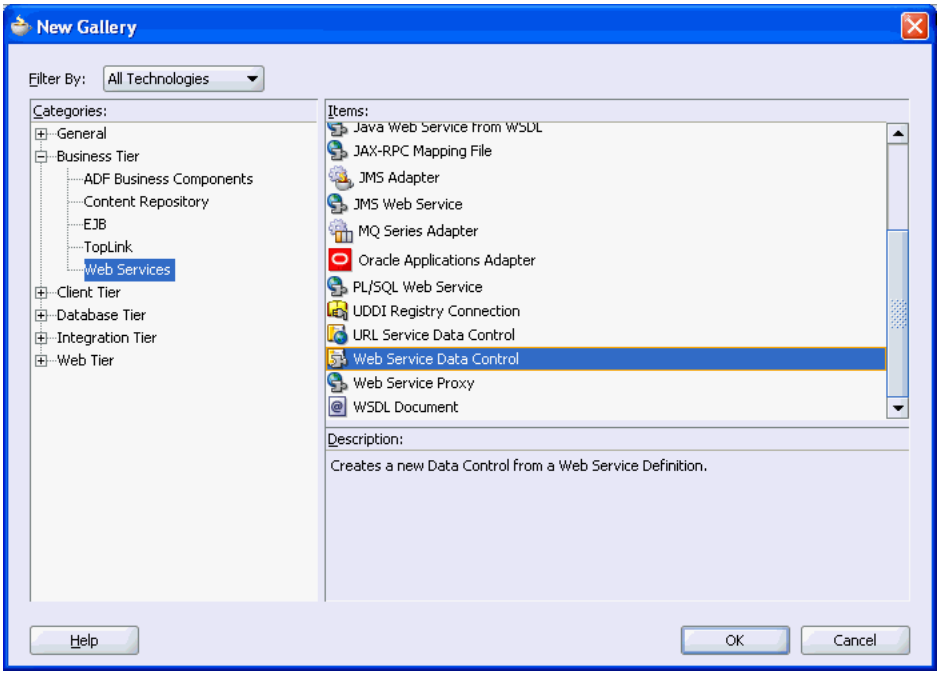
Stellent Content Server exposes its services as Web services. You retrieve the necessary information about the different services using the WSDL file, which comes packaged with the Stellent SOAP component. This WSDL file is available in the `<install_dir>/weblayout/groups/secure/wsdl/custom/` directory. For information on Stellent Web services support, see *Stellent Content Server - Using WSDL Generator and SOAP Manual*.

The Web Service data control enables you to integrate content from Stellent Content Server into your WebCenter application. In this section, we use the `search.wsdl` file to create a Web Service data control that will enable integration of content from Stellent Content Server into your WebCenter application.

To create a Stellent Content Server-based data control, perform the following steps:

1. Start Oracle JDeveloper, then open your WebCenter application and project.
2. Select the **Model** project, then select **New** from the **File** menu. The New Gallery window is displayed.
3. In the **Filter By** field, select **All Technologies**, if not selected already.
4. Expand **Business Tier** and select **Web Services**.
5. Under **Items**, select **Web Service Data Control**, as shown in Figure 5–89.

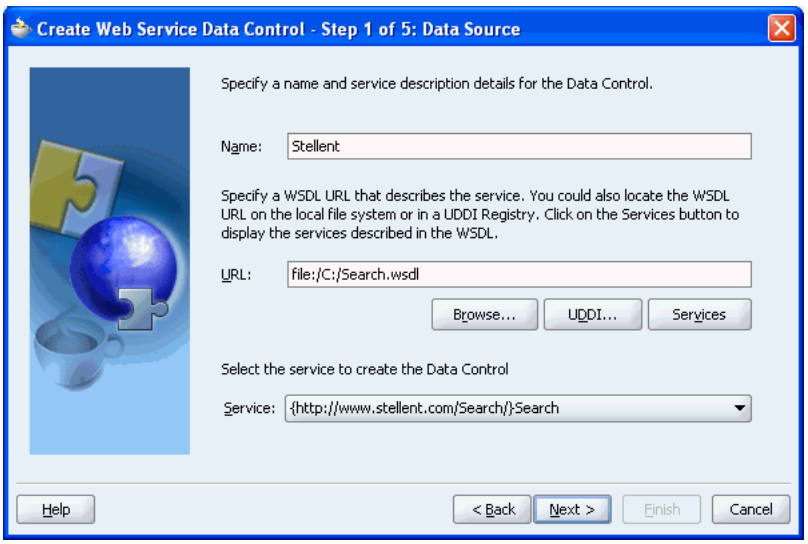
Figure 5–89 New Gallery - Web Service Data Control



6. Click **OK**. The Create Web Service Data Control wizard is displayed.
7. If the Welcome page is displayed, then click **Next** to skip it.

8. On the Data Source page of the wizard, enter a name for the data control in the **Name** field. Then, in the **URL** field, select your WSDL file, for example `search.wsdl`, by clicking **Browse**, as shown in Figure 5–90.

Figure 5–90 Data Source



9. Click **Next**. The Data Control Operations page of the wizard is displayed.

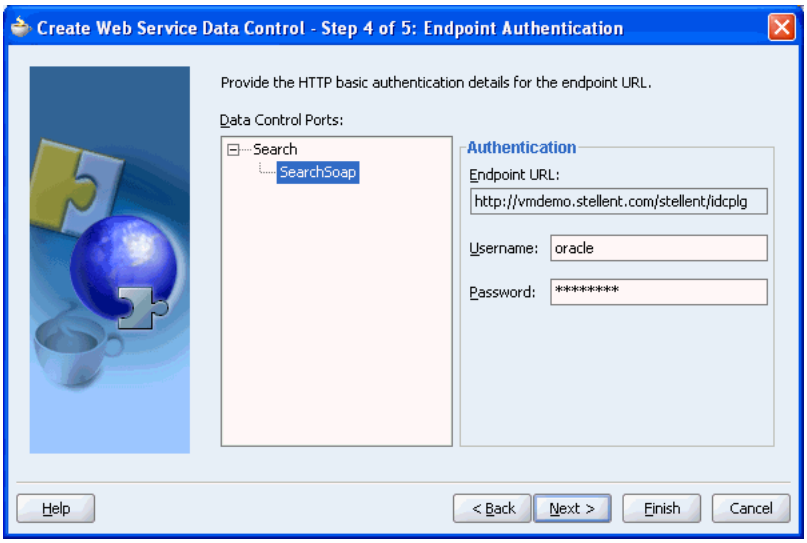
10. Select all available operations by clicking **Add All**, as shown in Figure 5–91, and click **Next**.

Figure 5–91 Data Control Operations

11. Click **Next** to skip the Response Format page of the wizard.

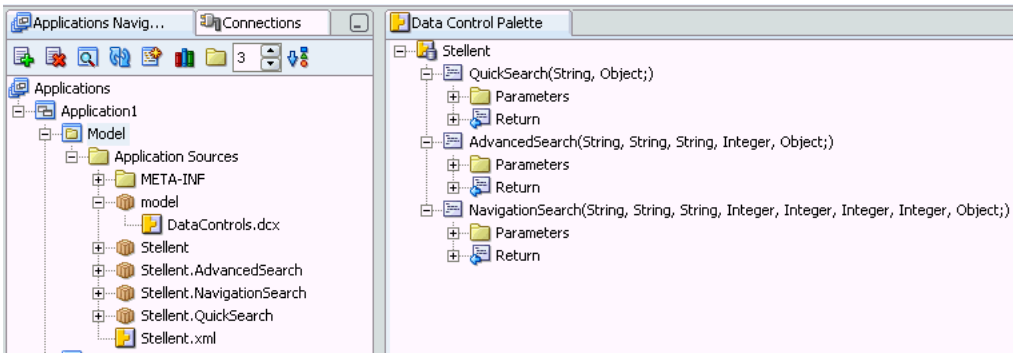
12. On the Endpoint Authentication page, select **SearchSoap**, and specify the username and password to log in to Stellent Content Server as shown in Figure 5–92.

Figure 5–92 Endpoint Authentication



13. Click **Finish**. The Stellent data control is created. Figure 5–93 shows `DataControls.dcx` and `Stellent.xml` files under the Applications Navigator and the Stellent data control under the Data Control Palette.

Figure 5–93 Stellent Data Control in the Applications Navigator and the Data Control Palette



Tip: To change the Endpoint URL or the user information, expand your **Model** project under the Applications Navigator. Under the **Model** project, expand the `model` package, then select the `DataControls.dcx` file. In the Structure pane, right-click the data control and select **Edit Web Service Connection**.

5.4.2 Integrating Content from Stellent Content Server Using a Custom Data Control

You can create a custom data control to integrate content from Stellent Content Server. A custom data control has an advantage over the Web Service data control, that is, it enables you to address complex scenarios, such as concatenating operations, adding business logic around the content operations, and so on.

Before you create your custom data control, you need to expose the relevant Web service proxy and custom Java class. The procedures to create Web service proxy, custom Java class, and custom data control are described in the following sections:

- Section 5.4.2.1, "Creating a Java Proxy"

- Section 5.4.2.2, "Creating the Custom Java Class"
- Section 5.4.2.3, "Creating a Custom Data Control"

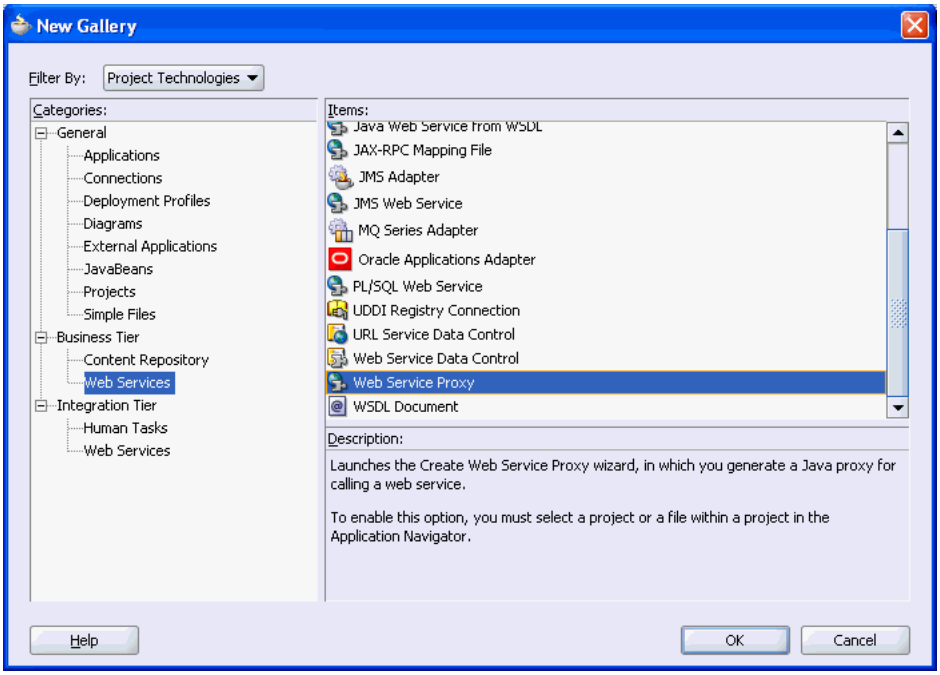
5.4.2.1 Creating a Java Proxy

The first step in the process of creating a custom data control is to create a Java proxy.

To create a Java Proxy, perform the following steps:

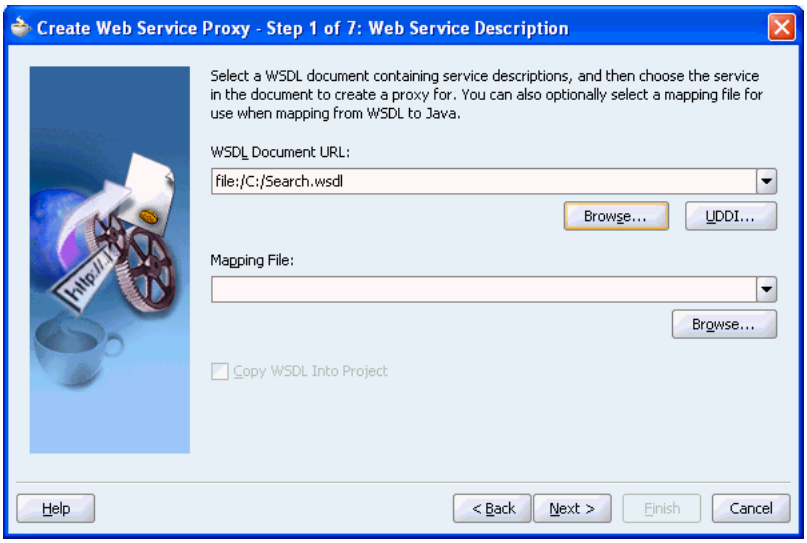
1. In Oracle JDeveloper, open your WebCenter application and project.
2. Select the **Model** project, then select **New** from the **File** menu. The New Gallery window is displayed.
3. Expand **Business Tier**, then select **Web Services**.
4. Under **Items**, select **Web Service Proxy**, as shown in Figure 5–94.

Figure 5–94 New Gallery - Web Service Proxy



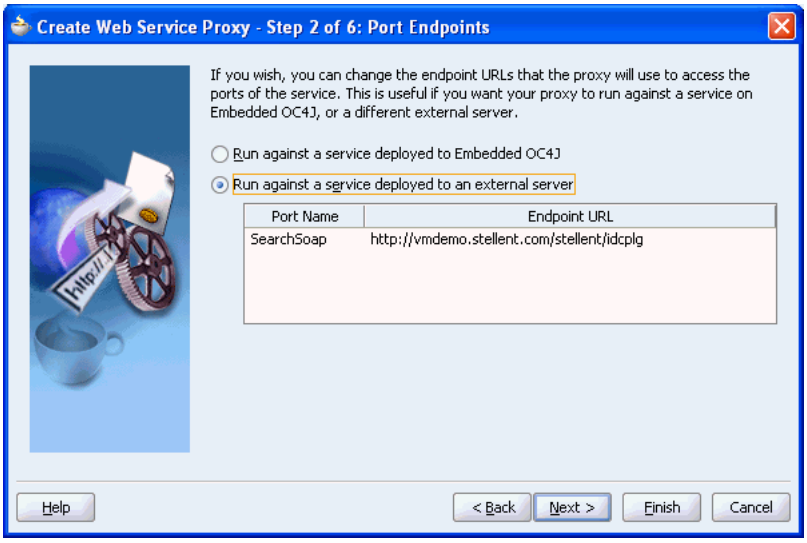
5. Click **OK**. The Create Web Service Proxy wizard is displayed.
6. If the Welcome page is displayed, then click **Next** to skip it.
7. On the Web Service Description page of the wizard, click **Browse** under WSDL Document URL and select the path to WSDL document, for example, `search.wsdl`, as shown in Figure 5–95.

Figure 5–95 Web Service Description



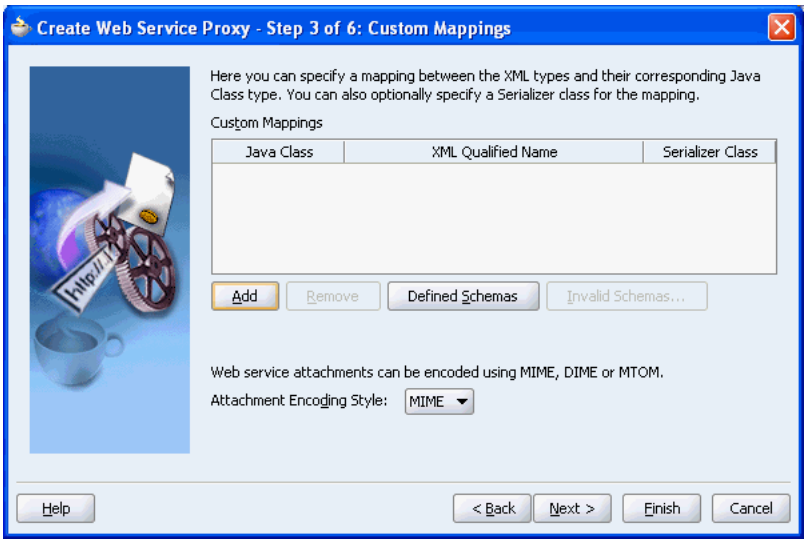
8. On the Port Endpoints page, as shown in Figure 5–96, you can modify the **Endpoint URL**, if necessary.

Figure 5–96 Port Endpoints



9. If required, perform the following optional steps:
   a. Click **Next** on the Port Endpoints page.
   b. On the Custom Mappings page, shown in Figure 5–97, specify mappings between XML types and their corresponding Java class types.

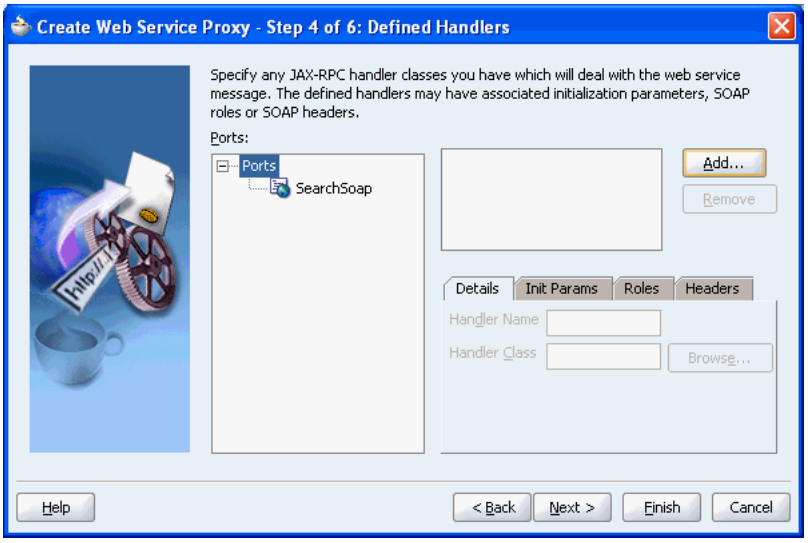
Figure 5–97 Custom Mappings



c. Click **Next**.

d. On the Defined Handlers page, shown in Figure 5–98, specify any handler class that you have to deal with the Web service message. You can also use this page to update any initialization parameters, SOAP roles, or SOAP headers.

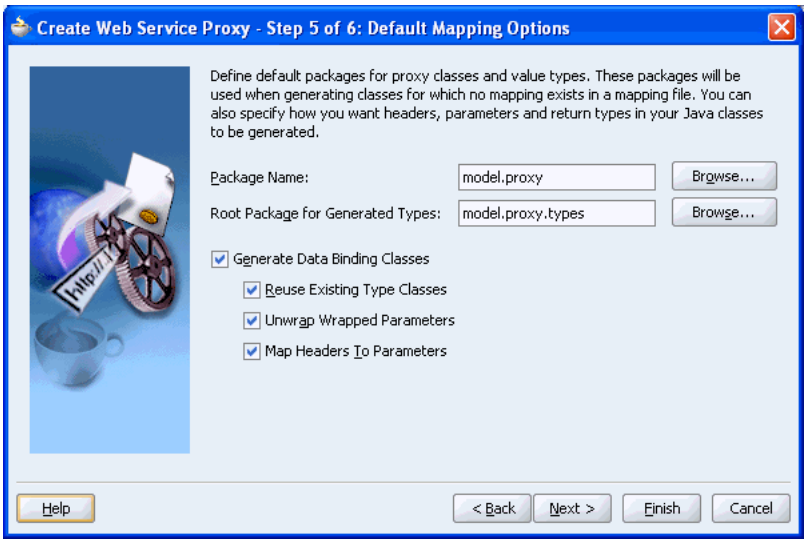
Figure 5–98 Defined Handlers



e. Click **Next**.

f. On the Default Mapping Options page, define default packages for proxy classes and value types as shown in Figure 5–99. These packages will be used when generating classes for which no mapping exists in a mapping file.

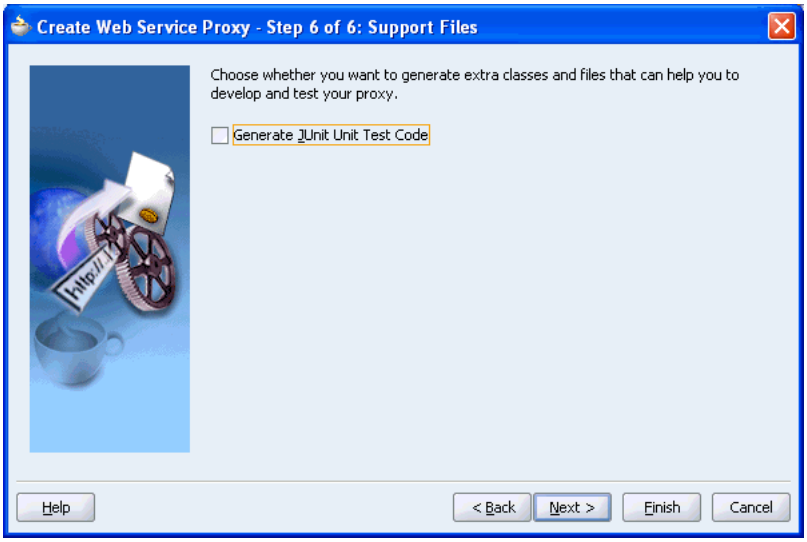
Figure 5–99 Default Mapping Options



g. Click **Next**.

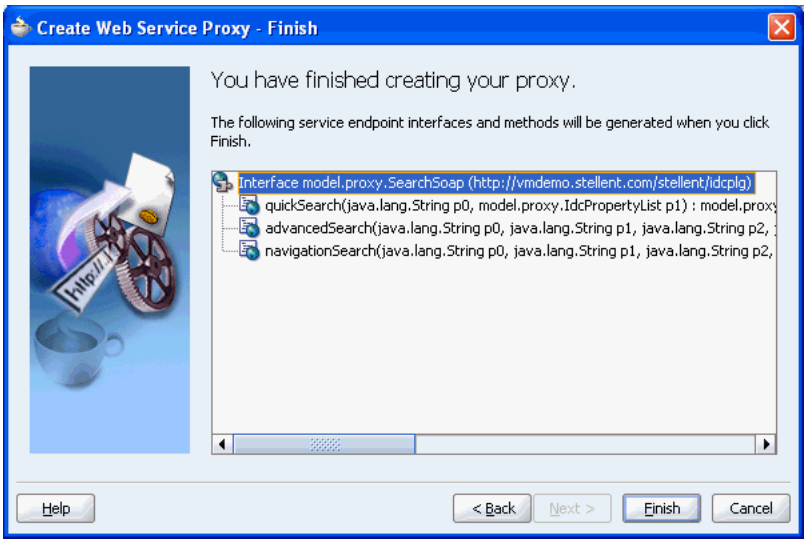
h. On the Support Files page, shown in Figure 5–100, specify whether you want to generate the extra classes and files to help you develop and test your proxy.

Figure 5–100 Support Files



i. Click **Next** to display the Finish page, which lists the service endpoint interfaces and methods that will be generated on finishing the wizard, as shown in Figure 5–101.

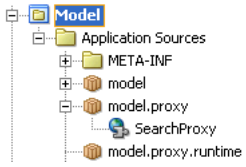
Figure 5–101 Finish Page



10. Click **Finish**.

    A set of classes for your Web service proxy are created under the **Model** project, as shown in Figure 5–102.

Figure 5–102 Web Service Proxy Classes



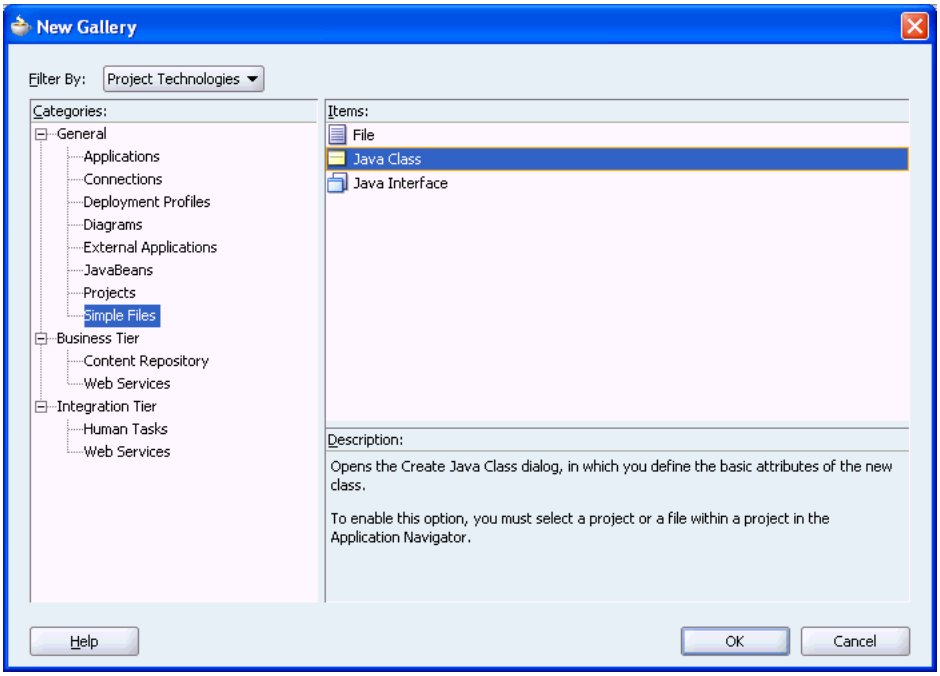
5.4.2.2 Creating the Custom Java Class

After creating the Web service proxy, the class that implements the business logic must be created.

To create a custom Java class, perform the following steps:

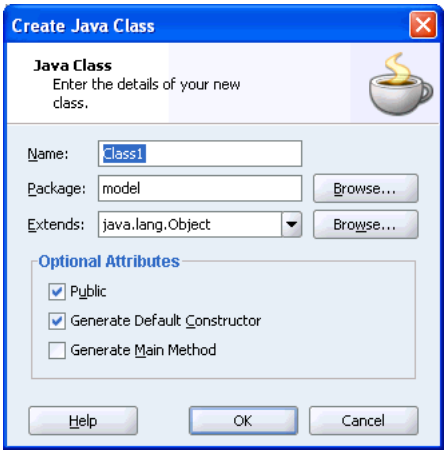
1. In Oracle JDeveloper, open your WebCenter application and project.
2. Select the **Model** project, then select **New** from the **File** menu. The New Gallery window is displayed.
3. Expand **General**, then select **Simple Files**.
4. Under **Items**, select **Java Class**, as shown in Figure 5–103.

Figure 5–103 New Gallery - Java Class



5. Click **OK**. The Create Java Class dialog box is displayed, as shown in Figure 5–104.

Figure 5–104 Create Java Class



6. Specify **Name** and **Package**, and click **OK**. The *class.java* file is opened in the design mode.

7. Add your business logic to the class. To ensure that required parameters can be passed and data returned, make sure you define input parameters and a valid `Return`. In Example 5–11, a search term is submitted and the result is returned. In this example, the username and password are hard-coded to simplify the code, but these could also be provided as parameters or derived through other methods from the application.

Example 5–11 Sample Java Class

```
package model;
import model.proxy.QuickSearchResult;
```

```
import model.proxy.SearchSoapClient;
public class StellentImpl {
    public QuickSearchResult StellentImpl(String queryTerm) {
        String docURL = "";
        QuickSearchResult result;
        result = new QuickSearchResult();
                try {
            SearchSoapClient myPort = new SearchSoapClient();
            myPort.setUsername("sysadmin");
            myPort.setPassword("idc");
            result = myPort.quickSearch(queryTerm, null);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result;

    }
}
```

8. After you have added your business logic, save the Java class.

5.4.2.3 Creating a Custom Data Control

Now that you have created a Java class with the custom code, you can create a custom data control from it.

To create a custom data control, go to the Applications Navigator and right-click the `.java` file that you created, as shown in Figure 5–105. Your custom data control is created and it is displayed under the Data Control Palette, as shown in Figure 5–106.

Figure 5–105 The Create Data Control Option

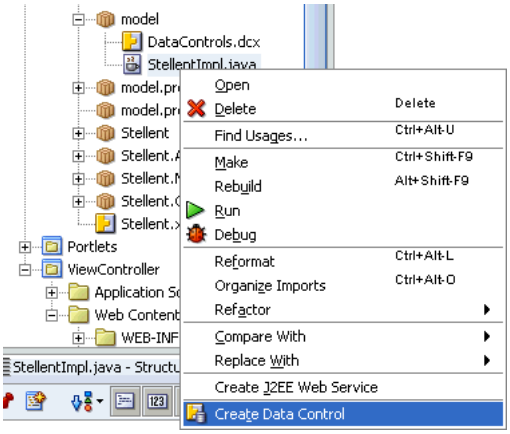
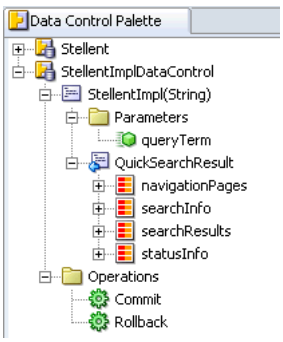


Figure 5–106 Custom Data Control in the Data Control Palette



5.5 Using Stellent Content Server-Based Data Controls: Examples

The following sections discuss through examples how to use Stellent Content Server-based Web service and custom data controls:

- Section 5.5.1, "Adding Simple Search Capabilities Using the Web Service Data Control"
- Section 5.5.2, "Adding Simple Search Capabilities Using the Custom Data Control"

5.5.1 Adding Simple Search Capabilities Using the Web Service Data Control

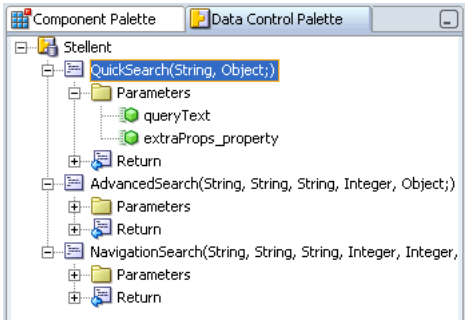
To add simple search capabilities that enable search in Stellent Content Server, perform the following steps:

1. If Oracle JDeveloper is not open, then start it and open the desired WebCenter application and project.
2. Expand **ViewController**, then double-click your JSPX page.

> **Note:** If your JSPX page does not expose UI components in a new managed bean, then create a new page with the **Automatically Expose UI Components in a New Managed Bean** option enabled. You can enable this option on the Component Binding page of the Create JSF JSP wizard. See Section 4.2, "Building WebCenter Application-Enabled Pages in Oracle JDeveloper with Oracle ADF" for more information.

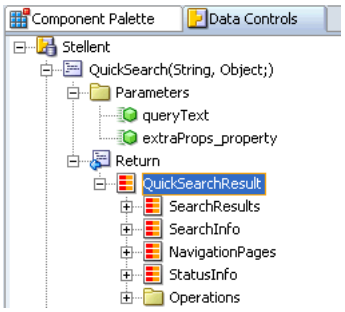
3. In the Data Control Palette, expand the **QuickSearch** node and its parameters, as shown in Figure 5–107.

Figure 5–107 Data Control Palette - QuickSearch



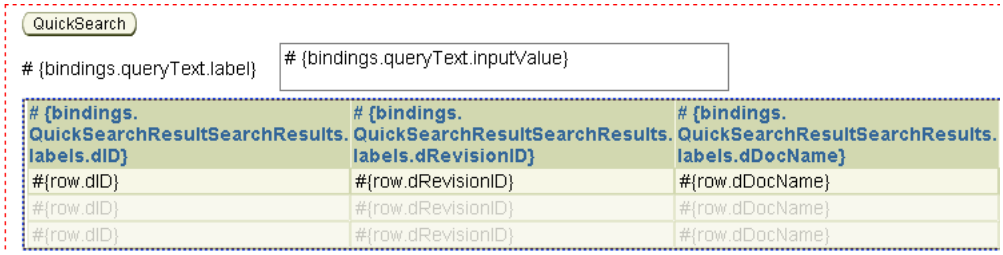
4. Drag and drop the **QuickSearch** node onto the page. From the **Create** menu, select **Methods**, then select **ADF Command Button**. The Action Binder Editor dialog box is displayed.
5. Leave the parameter fields blank, and click **OK**.
6. In the Data Control Palette, select the **queryText** attribute under **Parameters**, and drop it onto the page. From the **Create** menu, select **Texts**, then select **ADF Input Text w/Label**.
7. Expand the **Return** node and its child node **QuickSearchResult**, as shown in Figure 5–108.

Figure 5–108 The Return Node - QuickSearchResult



8. Under **QuickSearchResult**, select **SearchResults** and drop it onto the page. From the **Create** menu, select **Tables**, then select **ADF Read-only Table**. The Edit Table Columns dialog box is displayed.
9. In the Edit Table Columns dialog box, add or delete the columns as needed, and click **OK**. Your page in the design mode should look like Figure 5–109.

Figure 5–109 QuickSearch in the Design Mode

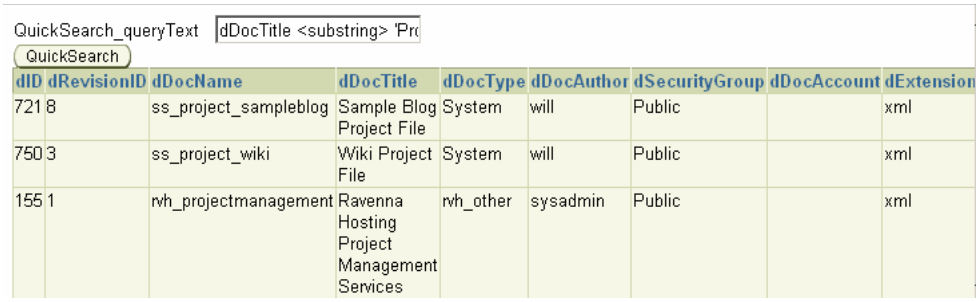


10. Run the page in a browser. The page should look like Figure 5–110.

Figure 5–110 QuickSearch in a Browser

11. Now submit a query, for example, `dDocTitle <substring> Project`, your search is executed and the results are displayed in the table, as shown in Figure 5–111. The results displayed in the table depend on the content of the Stellent Content Server.

Figure 5–111 The Search Results



5.5.2 Adding Simple Search Capabilities Using the Custom Data Control

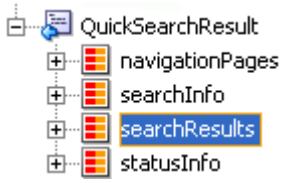
To add simple search capabilities to your JSPX page, perform the following steps:

1. If Oracle JDeveloper is not already open, then start it, and open the desired WebCenter application and project.
2. Expand **ViewController**, then double-click your JSPX page.
3. In the Data Control Palette, expand your custom data control and drop the method, for example, `StellentImp` onto the page. From the **Create** menu, select **Methods**, then select **ADF Command Button**. The Action Binding Editor is displayed.
4. Click **OK**.
5. Under **Parameters**, select **queryTerm**, as shown in Figure 5–112, and drop it onto the page. From the **Create** menu, select **Texts**, then select **ADF Input Text w/Label**.

Figure 5–112 queryTerm

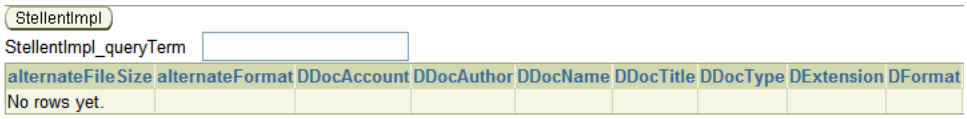
6. Under **QuickSearchResults**, select **searchResults**, as shown in Figure 5–113, and drop it on the page. From the **Create** menu, select **Tables**, then select **ADF Read-only Table**. The Edit Table Columns dialog box is displayed.

Figure 5–113 searchResults



7. Click **OK** and run the page in a browser. Your page should look like Figure 5–114.

Figure 5–114 Search Page in a Browser



8. Perform a search for `dDocTitle <substring> Project`, for example. Your page should show the expected results.

5.6 Integrating Oracle Business Intelligence Publisher

Oracle Business Intelligence Publisher reduces the high costs associated with the development, customization and maintenance of business documents; while increasing the efficiency of reports management. Utilizing a set of familiar desktop tools users can create and maintain their own report formats based on data extracts from diverse sources.

You can integrate Oracle Business Intelligence (BI) Publisher in either of the following ways:

- Integrate the BI Publisher application. BI Publisher is a self-contained application that manages a set of reports. After logging in, you are presented with a hierarchy of reports organized by category. Each of these reports can be independently included in your WebCenter application. See Section 5.6.1, "Integrating BI Publisher Reports into Your WebCenter Application" for more information.
- Integrate scheduled BI Publisher reports. BI Publisher runs scheduled reports and stores their output. These reports can subsequently be viewed by directly accessing the underlying document repository. See Section 5.6.2, "Storing BI Publisher Reports in the WebCenter Content Repository" for more information.

For detailed information about how to use BI Publisher, see *Oracle Business Intelligence Publisher User's Guide*.

5.6.1 Integrating BI Publisher Reports into Your WebCenter Application

The most convenient mechanism for integrating BI Publisher with your WebCenter application is the built-in Web Clipping portlet. This portlet enables you to select any part of a BI Publisher report and display it directly within your WebCenter application. It also handles aspects such as external application integration and parameterization of these reports.

1. Add a Web Clipping portlet to your page by following the instructions in Section 17.2.1, "Registering a Web Clipping Producer" and Section 17.2.2, "Adding a Web Clipping Portlet to a Page". Once you have the Web Clipping portlet, you must obtain the starting URL for BI Publisher.
2. Log in to BI Publisher and view the report that you want to include in your WebCenter application. For detailed information about how to use BI Publisher, see *Oracle Business Intelligence Publisher User's Guide*.

> **Note:** If you want to use the initial BI Publisher URL as your first page URL, you must turn on accessibility mode on the login page.

3. Cut and paste the URL of this report page as the start page for your Web Clipping portlet. Alternatively, you could construct this URL by following the instructions

in "Accessing Reports via a URL" in the *Oracle Business Intelligence Publisher User's Guide*.

4. Proceed through the Web Clipping process as described in Section 17.2.3, "Selecting a Section of a Web Page to Display in the Web Clipping Portlet". The result displays all or part of the BI Publisher report in your WebCenter application.
5. If you do not want your users to have to enter credential information when they view the page, you can register the Web Clipping producer used to display your reports as an external application. Follow the instructions in Section 17.3, "Integrating Authenticated Web Content Using Single Sign-On". You must choose the following options in the Register External Application Wizard:
   - For **Login URL**, enter a URL of the following form:

     `http://host:port/xmlpserver/login.jsp`

   - For **User Name/ID Field Name**, enter `id`.
   - For **Password Field ID**, enter `passwd`.
   - For **Authentication Method**, choose **POST**.

5.6.2 Storing BI Publisher Reports in the WebCenter Content Repository

Scheduled BI Publisher reports leverage WebDAV APIs to store the reports to the desired document repository. WebCenter Framework uses Oracle Content DB as it's out-of-the-box content repository. Oracle Content DB includes a WebDAV server that BI Publisher can leverage for persisting reports into Oracle Content DB.

6

Integrating Oracle WebCenter Wiki

This chapter explains how to integrate Oracle WebCenter Wiki into a WebCenter application. You will learn about wiki, the wiki syntax, and how to consume the Oracle WebCenter Wiki Web Service into your WebCenter applications.

To learn about the administration options for Oracle WebCenter Wiki, refer to Appendix E, "Administering Oracle WebCenter Wiki".

This chapter contains the following sections:

6.1 Introduction to Wiki

Oracle WebCenter Suite enables you to integrate many different services into your WebCenter application, including the Oracle WebCenter Wiki. A wiki is a type of Web site where users can browse available content, update, remove, and otherwise edit the content, sometimes without the need for registration. This ease of interaction and the variety of operations makes wiki an effective tool for collaborative authoring, where multiple people create written content together using the wiki markup language. For more information about wiki, see the Wikipedia at `http://en.wikipedia.org/wiki/Wiki`.

Oracle WebCenter Suite enables you to use Oracle WebCenter Wiki, which includes its own Web-based interface and a Web Service that enables you to embed the wiki into your WebCenter applications. Users of the Oracle WebCenter Suite can use wiki's Web-based interface to create and edit wiki pages. WebCenter application developers can embed some of the wiki functionality into their applications using the Web Service interface.

6.2 Setting Up Oracle WebCenter Wiki

This section describes how to install and configure wiki for your Oracle WebCenter Suite environment.

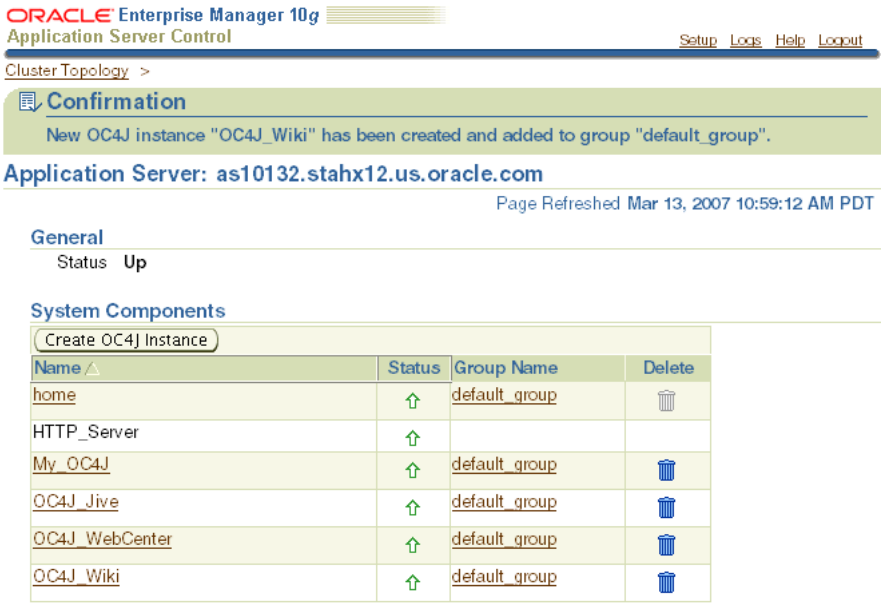
6.2.1 Installing Oracle WebCenter Wiki

To install the Oracle WebCenter Wiki, perform the following steps:

1. Log in to the Application Server Control as an administrator.

2. Click the Oracle Application Server instance to go to its application server page.
3. We recommend you create a new OC4J instance to contain Oracle WebCenter Wiki. Although not recommended, you can use an existing OC4J instance. To create a new OC4J instance, click **Create OC4J Instance** (Figure 6–1).

Figure 6–1 Application Server Page of Application Server Control

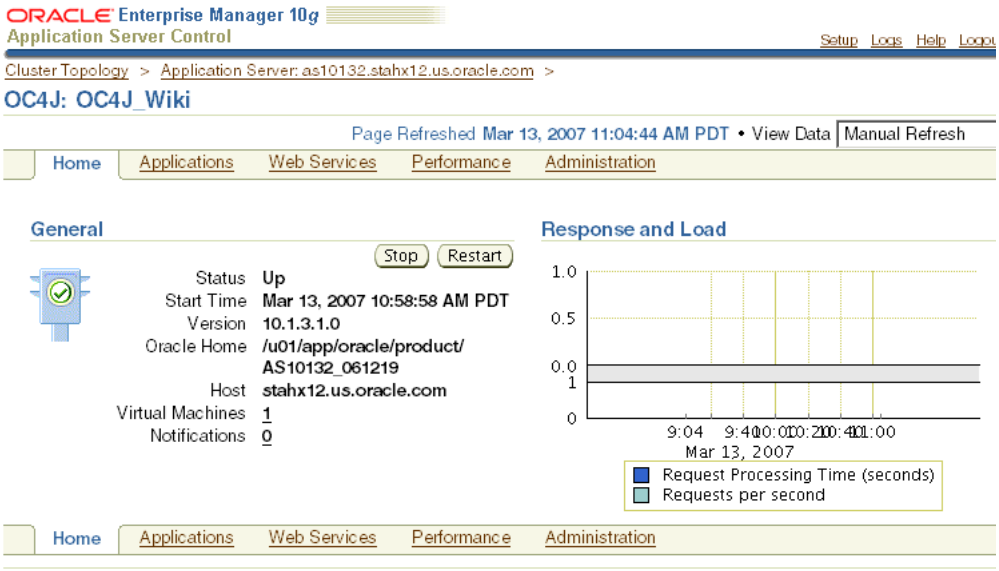


This is a text description of jpsdg_wiki_em1.gif. This figure shows the Application Server Control with several OC4J instances listed.

****

4. Fill in the values for your new OC4J instance. Be sure to check **Start this OC4J instance after creation**, then click **Create**.
5. Follow the instructions in Section 18.3, "Configuring Your Application Server or Standalone OC4J to Run Portlets" to configure your OC4J instance to run portlets.
6. Click the name of OC4J instance to which you plan to deploy wiki. The Home tab displays (Figure 6–2).

Figure 6–2 Home Tab for OC4J Instance

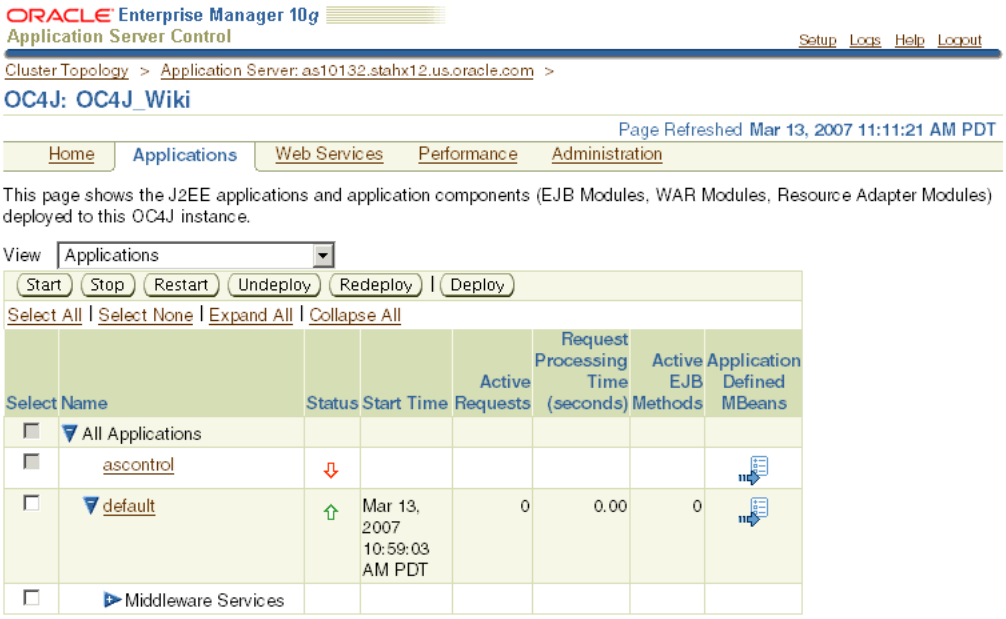


This is a text description of jpsdg_wiki_em2.gif. This figure shows the Home tab of the OC4J instance to which you are planning to deploy owc_wiki.ear.

****

7. Click the **Applications** tab (Figure 6–3).

Figure 6–3 Applications Tab for OC4J Instance



This is a text description of jpsdg_wiki_em3.gif. This figure shows the Applications tab of the OC4J instance to which you are planning to deploy owc_wiki.ear.

****

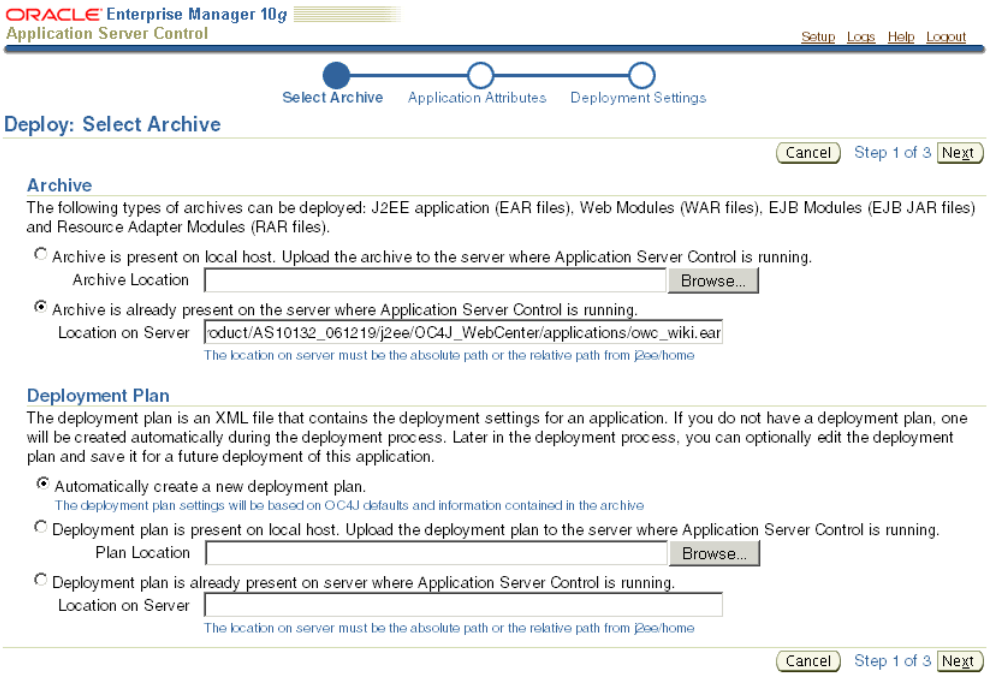
8. Click **Deploy**.

9. Choose the location of the EAR file (local or host) and specify the path (Figure 6–4).

   Tip: The Oracle WebCenter Wiki EAR file is `owc_wiki.ear`.

10. Choose **Automatically create a new deployment plan** (Figure 6–4).

Figure 6–4 Deploy: Select Archive Page



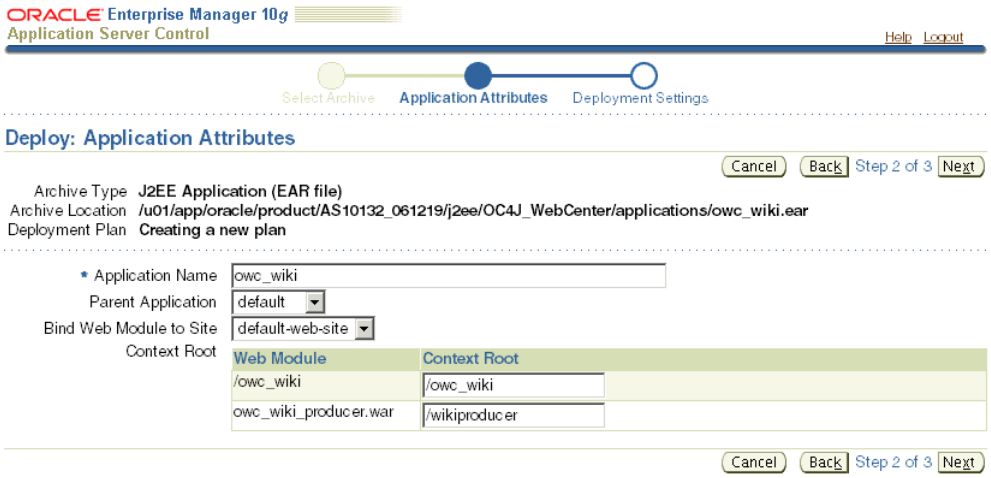
This is a text description of jpsdg_wiki_em4.gif. This figure shows the Select Archive page of the Deployment Wizard filled out for owc_wiki.ear.

****

11. Click **Next**. Note that, if the EAR file is local, it may take several minutes to upload to the server.

12. Enter **Application Name**, for example, `owc_wiki` and ensure that the Context Root is also set to something similar to the application name. You will use the context root to access the application (Figure 6–5).

Figure 6–5 Deploy: Application Attributes Page

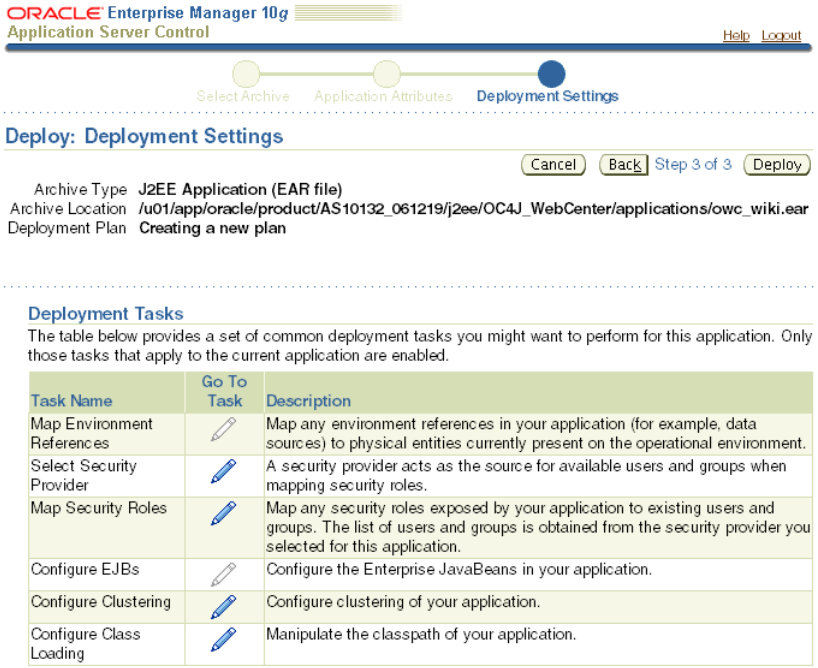


This is a text description of jpsdg_wiki_em5.gif. This figure shows the Application Attributes page of the Deployment Wizard with the Application Name entered.

****

13. Click **Next**.
14. Verify the deployment settings (Figure 6–6).

Figure 6–6 Deploy: Deployment Settings Page



This is a text description of jpsdg_wiki_em6.gif. This figure shows the Deployment Settings page of the Deployment Wizard.

****

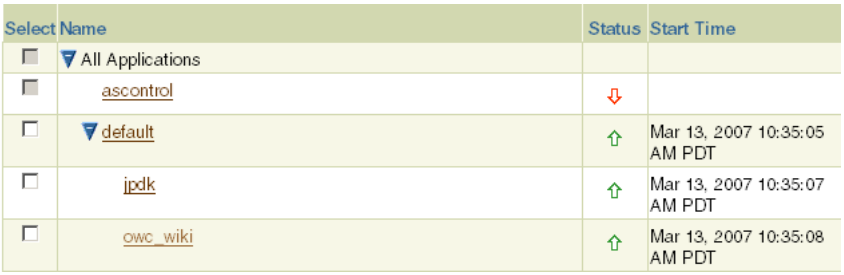
15. Click **Deploy**. Once the deployment has completed successfully, you see a confirmation message. If the deployment is not successful, review the log and confirm that you had the correct deployment settings.

Note: Oracle WebCenter Wiki includes an HSQL database which runs on a specific port (1475). If that port is not available, you can modify the port after the installation in the `beans.xml` file, which is located in `OC4J_HOME/applications/application_root/owc_wiki/WEB-INF/classes`.

---

16. Click **Return**. You should now see the application running in your OC4J instance (Figure 6–7).

Figure 6–7 Applications Tab with Wiki Application Deployed



This is a text description of jpsdg_wiki_warn.gif. This figure shows the warning message that you receive when you try to start the Java SSO application and it has some configuration problems. Typically, this message indicates that you have a clustered environment, which requires special configuration.

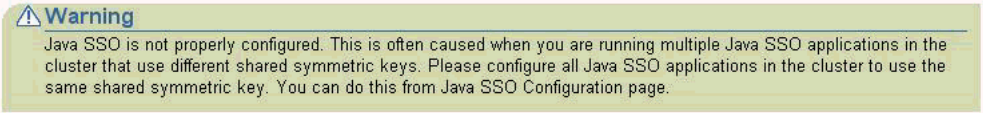
****

6.2.2 Configuring Security

Because Oracle WebCenter Wiki uses Oracle WebCenter Suite Java SSO, you must enable Java SSO on the Application Server to which you deploy Oracle WebCenter Wiki. To verify that Java SSO is running, go back to Enterprise Manager and start the OC4J instance. Java SSO should appear as an application under the OC4J instance. Once the instance is started, confirm that Java SSO is running. If Java SSO is down, then start it.

When you start Java SSO, you may get a warning message (Figure 6–8) that Java SSO is not properly configured. This warning indicates that you are running multiple Java SSO applications in a cluster. To ensure that you properly configure Java SSO for a clustered environment, refer to Section 6.2.2.2, "Clustered Configurations for Java SSO".

Figure 6–8 Warning for Java SSO Configuration



Note: For more information on Java SSO, refer to the *Oracle Containers for J2EE Security Guide* at http://download-west.oracle.com/docs/cd/B32110_01/web.1013/b28957/javasso.htm#BABGJCFD. For additional information on SSO setup, refer to Chapter 14 "OC4J Java Single Sign-On" of the *Oracle Containers for J2EE Security Guide*, located at http://download-west.oracle.com/docs/cd/B32110_01/web.1013/b28957/javasso.htm#JISEC1120

This section contains information on:

- User Groups
- Clustered Configurations for Java SSO
- Generating the Passphrase

6.2.2.1 User Groups

Only WebCenter application users who possess the security role `authenticated-users` can log in and view Oracle WebCenter Wiki pages. By default, there are two Oracle WebCenter Suite security groups (`oc4j-administrators` and `users`) that possess this security role. Only members that belong to one of these two groups can access the wiki pages. You can assign (or remove) the role `authenticated-user` to any Oracle WebCenter Suite groups or users to control who can log in and view the wiki pages. To do so, see Chapter "General Tasks for OC4J Security" in the *Oracle Containers for J2EE Security Guide*, located here: http://download-west.oracle.com/docs/cd/B32110_01/web.1013/b28957/deploysimple.htm#CHDIGIFJ.

After deployment, there is a default wiki user `oc4jadmin`, which is assigned to the wiki role `ADMIN`. When a user logs into a WebCenter application and accesses the wiki for the first time, Oracle WebCenter Wiki checks if a wiki user with the same user name already exists in its user repository. If one already exists, the user is authenticated and accesses the wiki. If the name does not already exist, Oracle WebCenter Wiki creates a new wiki user with the same name as the WebCenter application user and assigns the user to the wiki role `USER`.

As the wiki administrator (a user assigned to the wiki role ADMIN), you can create new users. Once you have created a user either manually or automatically, you can manually change the wiki user's role to ADMIN or USER.

Note: For more information on wiki administration, refer to Appendix E, "Administering Oracle WebCenter Wiki".

6.2.2.2 Clustered Configurations for Java SSO

You can set `custom.sso.key.alias` from Enterprise Manager. Click **Java SSO Configuration** at the bottom of the **Cluster Topology** page for your application server instance. Refer to the OC4J Java Single Sign-On chapter in the *Oracle Containers for J2EE Security Guide*.

When you configure Java SSO from Enterprise Manager, users may encounter an error where the Oracle Application Server home page displays when they try to logout. To correct this problem, delete the following properties from `jazn.xml`:

```
property name="custom.sso.cookie.domain" value=""
```

```
property name="custom.sso.url.param" value=""
```

6.2.2.3 Generating the Passphrase

After you deploy Oracle WebCenter Wiki, you must generate a passphrase that the wiki developer will use when calling methods in the Oracle WebCenter Wiki Web Service or running the sample portlets, as described later in Section 6.4.1.2, "Web Service Security".

> **Note:** You may skip this section if you do not plan to use the Oracle WebCenter Wiki Web Service.

To generate the parameter key, perform the following steps:

1. Log in to Enterprise Manager.
2. Under the OC4J instance where you installed Oracle WebCenter Wiki, locate the deployed Oracle WebCenter Wiki application. The application root should be similar to `owc_wiki`.
3. Click the application.
4. Click the **owc_wiki** module.
5. Click the **Administration** tab.
6. On the Administration page, click **Environment Entry Mappings**.
7. Locate the task: **Environment Entry Mappings** and update the value for `/oracle/webCenter/owcWiki/webServiceSecurityPassphrase`.
8. Update the value with a new passphrase., which is an arbitrary value that you create, then click **OK**. When you update the deployed value, deploying the application again will not overwrite the value you have set.

6.2.3 Locations

Once you have deployed the EAR file and set up your wiki configuration file, you can integrate wiki into your WebCenter applications, as described in Section 6.4, "Integrating Oracle WebCenter Wiki into a WebCenter Application". Table 6–1 describes access points for the wiki.

Table 6–1 Web Access Points and Locations

Web Access Point	URL
wiki	`http://host:port/owc_wiki`
Web Service end point	`http://host:port/owc_wiki/services/WikiRemoteService`
Web Service WSDL file	`http://host:port/owc_wiki/services/WikiRemoteService?WSDL`
portlet producer URL	`http://host:port/wikiproducer/portlets/wsrp2?WSDL`

> **Note:** Use the host and port for the location to which you deployed Oracle WebCenter Wiki.

Oracle WebCenter Wiki stores wiki pages in the file system under *OC4J_HOME*/applications/*application_name*/owc_wiki/pages. The folders at this location are the domains and the files in each folder are the pages.

6.3 Using Oracle WebCenter Wiki

The Oracle WebCenter Wiki enables users to create, edit and modify pages without having to perform any of the administrative tasks, which are described in Appendix E, "Administering Oracle WebCenter Wiki". Oracle WebCenter Wiki provides a number of other standard wiki features, such as adding attachments to pages, bookmarks, and search.

This section contains the following:

- Structure of Wiki Content
- Editing and Creating Pages
- Wiki Markup

6.3.1 Structure of Wiki Content

Oracle WebCenter Wiki categorizes its content into domains and pages. Wiki users can create and edit pages. Administrators create domains that contain these pages. The administrator can also create a menu for each domain, which enables users to quickly access domain pages and other built-in functions, such as popular pages. The wiki tracks every version of the page, including the author, the date and time of the modification.

6.3.2 Editing and Creating Pages

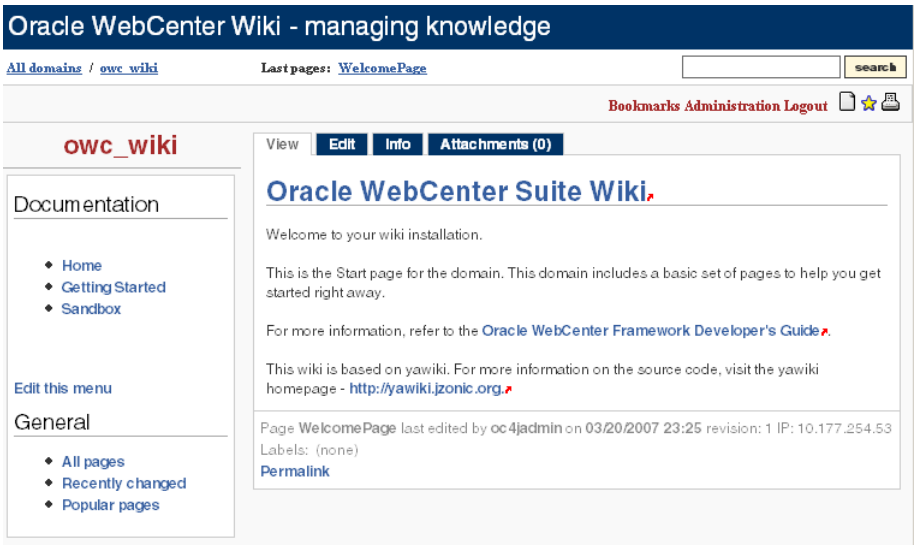
This section shows you how to edit existing pages and create new pages with Oracle WebCenter Wiki.

To access the wiki, go to the following URL, then log in.

http://*host*:*port*/owc_wiki

Note: The host and port are the location where you installed Oracle WebCenter Wiki. If you are not an authenticated earlier, a login prompt displays, where you can enter your user name and password.

Figure 6–9 Wiki Home Page



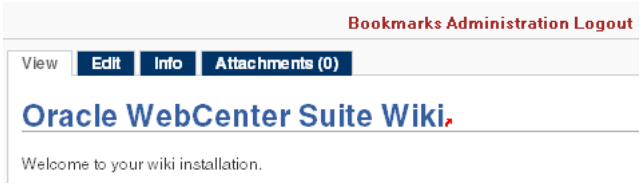
In the upper right corner, several links enable you to navigate to other areas of the wiki: bookmarks and administration. You can view the pages you have bookmarked and administer the wiki (when you are logged in as the administrator).

Note: For more information about administering the wiki and the various options on the Administration page, refer to.Section E.1, "Accessing the Administration Mode".

6.3.2.1 Editing Pages

Users can perform several tasks on a page, such as viewing it, editing it, viewing information about the page, and adding attachments. To perform each of these tasks, simply click the appropriate tab, as shown in Figure 6–10.

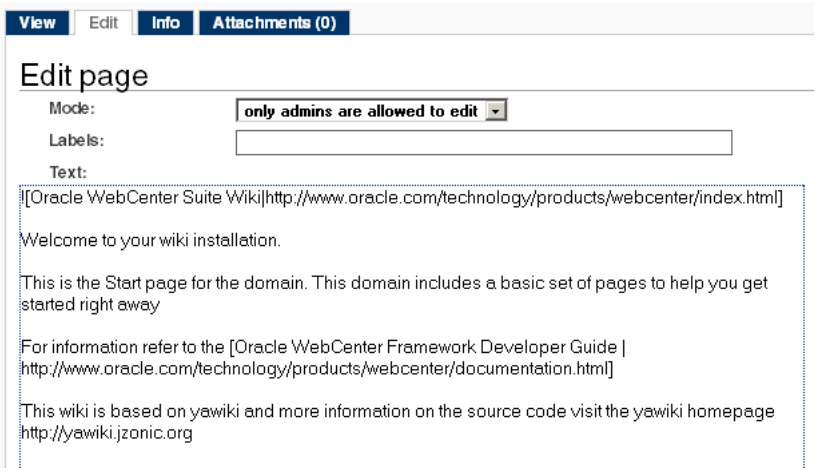
Figure 6–10 Wiki Tabs



To edit the current page, click the **Edit** tab, which then displays the Edit page text box where you can modify the page source, as shown in Figure 6–11. Here, you can control who can edit the page, add and remove page labels, and modify the page. The page source uses the wiki syntax, which is described in Section 6.3.3, "Wiki Markup". When you finish editing the page, click **Submit Query** at the bottom of the page. If you do not want to save the changes, click **Cancel**. If you do not click either Submit Query or Cancel, then navigate to another page (or click the View tab), Oracle WebCenter Wiki locks the currently edited page typically for 10 minutes.

Note: For information on the administrative task of locking and unlocking pages, refer to Section E.3, "Locking and Unlocking Pages"

Figure 6–11 Editing a Wiki Page



6.3.2.2 Creating Pages

As a user, you can create new pages within an existing domain that the administrator has created.

To create new pages within a domain, perform the following steps:

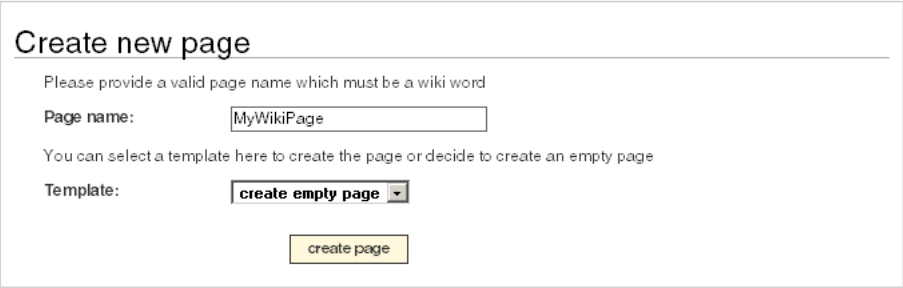
1. In the wiki, in the upper left corner, click **All Domains**, then click the domain in which you wish to create a page.
2. In the upper right corner, click the **Add Page** icon next to the Logout link, as shown in Figure 6–12.

Figure 6–12 Add Page Icon

3. On the Create new page screen as shown in Figure 6–13, in the **Page name** field, enter a page name, such as `MyWikiPage`. Ensure that you follow the wiki naming convention for pages. You can also choose a template for your page.

Note: For more information on the wiki page naming convention, see Section 6.3.3, "Wiki Markup".

Figure 6–13 Creating a New Page



4. Click **Create Page**. The Edit page screen displays where you can modify the contents of your page.

6.3.3 Wiki Markup

Oracle WebCenter Wiki uses a special markup language to format the content of the pages. This section describes some commonly used rules and examples of usage. For general information about creating and editing pages, refer to Section 6.3.2, "Editing and Creating Pages".

Table 6–2 Commonly Used Wiki Formatting Rules

Formatting Rule	Description	Syntax Examples

Headers	Define headers using exclamation points (!). The number of exclamation points defines the header depth.	`!Header1`
		`!!Header2`
		`!!!!Header4`

Emphasis	Wrap the text you wish to emphasize with the following characters:
	Bold: *
	Italics: #
	Underlined: _
	Note that you cannot use this markup at the beginning of a line.
		`The following is*bold text*.`
		`The following is #italicized text#.`
		`The following is _ underlined text_.`

Links	Display external links either by simply entering the URL to display the URL (http://www.oracle.com) or by using the following to display the URL name: [name of the link | URL] Display internal links to other wiki pages by using the following: [name of the link | wiki page you want to link] Note that if the internal page does not exist, wiki will create a new one and display a question mark (?) next to the page name in the View mode, which users can click to edit the page.	`[ oracle | http://www.oracle.com ]` `[ Seattle | SeattleSupportPage ]`

Table 6–2 (Cont.) Commonly Used Wiki Formatting Rules

Formatting Rule	Description	Syntax Examples
Wiki Page Names	Use the camel notation to name your wiki pages. This notation uses an initial upper case letter followed by lower case letters, then another upper case letter and another series of lower case letters, for example, MyWikiPage. To use an alternate name for your page, use the following convention: [alternate name | Wiki page name]	`[ My Page | MyPage ]`

Lists	At the beginning of a new line, use an asterisk (*) to denote a bulleted list or the number sign (#) to denote a numbered list.	`* bulleted item 1`
		`* bulleted item 2`
		
		`# numbered item 1`
		`# numbered item 2`

Tables	Use HTML to create a table. Open and close the table with the <table></table> tag and define columns by using the "|" symbol.	`<table width="75%" border="0">`
		`*col1* | *col2*`
		`Hello | world`
		`Here | I am`
		`</table>`

Images	Use standard HTML to define an image source.	`<img src="http://www.anyserver.com/image.gif"/>`

6.4 Integrating Oracle WebCenter Wiki into a WebCenter Application

You can integrate Oracle WebCenter Wiki into your WebCenter application components, which access the Oracle WebCenter Wiki service and the pages stored in it. This section explains how you can use the Oracle WebCenter Wiki Web Service to enable users to access the wiki This section also explains how to create JSR 168 portlets that access information in the wiki.

This section contains the following:

- Oracle WebCenter Wiki Web Service Interface
- Sample Portlets
- Writing a Portlet

6.4.1 Oracle WebCenter Wiki Web Service Interface

Oracle WebCenter Wiki provides a Web Service that enables interaction between your WebCenter application and the wiki. Once you have installed Oracle WebCenter Wiki, you can access the Web Service end point by using the following URL:

`http://localhost:port/owc_wiki/services/WikiRemoteService`

> **Note:** In this URL, the host and port information refers to the computer where you installed Oracle WebCenter Wiki.

The Oracle WebCenter Wiki Web Service provides access to obtain information and content from the wiki pages and domains. It also enables the creation, modification,

and removal of wiki pages and domains. You can use Oracle JDeveloper to create a proxy for the Web Service from the WSDL definition, located here:

`http://host:port/owc_wiki/services/WikiRemoteService?WSDL`

The sample portlets code bundle also contains a proxy. You can find the this proxy in the `web_service_proxy.jar` file, located in `portlet_producer_sample.zip` in the Oracle WebCenter Wiki files. You can find more information about the proxy by viewing the Javadoc located in the `portlet_producer_sample.zip` file.

This section describes the Web Service interface. For information about the Web Service security, refer to Section 6.4.1.2, "Web Service Security".

6.4.1.1 Definition of the Interface

Some of the methods return information in JavaBeans. Table 6–3 shows the attributes of the `DomainInfo` and `PageInfo` beans. You can also use the getter methods of the described attributes, for example, `long getCreated()`.

Table 6–3 Web Service Data Structures

`DomainInfo` Bean	**`PageInfo` Bean**
`String domain;`	`String domain;`
`String description;`	`String name;`
`String author;`	`int revision;`
`long created;`	`int views;`
`String startPage;`	`String author;`
	`long created;`
	`String editor;`
	`long modified;`
	`String viewURL;`
	`String editURL;`

The Web Service methods include methods for accessing and performing actions on the domains and pages in the wiki, as described in Table 6–4 and Table 6–5.

Table 6–4 Domain-Related Methods

Attribute	**Method**
`DomainInfo`	`getDomainInfo(String domainName, String key)`
`DomainInfo[]`	`getAllDomainInfo(String key)`
`void`	`createDomain (String domainName, String description, String startPage, String key)`
`void`	`deleteDomain(String domainName, String key)`

Table 6–5 Page-Related Methods

Return Type	**Method**	**Description**
`PageInfo`	`getPageInfo (String domainName, String pageName, String key)`	

Table 6–5 (Cont.) Page-Related Methods

Return Type	Method	Description
`PageInfo[]`	`getAllPageInfo (String domainName, String key)`	
`void`	`createPage (String domainName, String pageName, String key)`	
`void`	`deletePage (String domainName, String pageName, String key)`	
`String`	`getPlainPage (String domainName, String pageName, String key)`	Returns the page content in the wiki markup format.
`String`	`getRenderedPage (String domainName, String pageName, String key)`	Returns the content rendered to HTML.
`void`	`savePage (String domainName, String pageName, String content, String key)`	String content takes the page's new content in wiki markup format

6.4.1.2 Web Service Security

All Oracle WebCenter Wiki Web Service methods are protected to prevent unauthorized access. Every method contains a String key parameter to ensure authorized access. This key is generated as a function of a user's name and a preconfigured passphrase, using the `KeyEncryptor.genkey(<username>, <passphrase>)` method. The passphrase is an arbitrary string which the administrator sets up in the Oracle WebCenter Wiki application after installation (see Section 6.2.2.3, "Generating the Passphrase"). As the Oracle WebCenter Wiki developer, you need to know this passphrase in order to use the Web Service interface to access the wiki.

> **Note:** The class `KeyEncryptor` is located in the Java library `ora_wiki_sec.jar` in the `portlet_producer_sample.zip` file. You can also find Javadoc for this library in the ZIP file.

To create the key, use the following method:

```
String key = KeyEncryptor.genkey(username, passphrase);
```

> **Note:** In this method, the username refers to the name of the user on whose behalf the Web Service is making the call (for example, the user logged into your WebCenter application who is accessing the wiki by means of a portlet) and the passphrase refers to the password you wish that user to use.

6.4.1.3 Example Java program

The following code is an example of a Java program that accesses the Web Service interface. This program will list all the page names in the Training domain.

```
package oracle.webcenter.wiki.ws.test;
import  oracle.webcenter.wiki.ws.*;
import  oracle.webcenter.wiki.security.*;
```

```
public class ListPages
{
```

In this example, because we have deployed the wiki to the Oracle WebCenter Preconfigured OC4J, we will hard code the Web Service end point. You can, however, parameterize the end point.

```
  private static final String endpoint =
    "http://localhost:6688/owc_wiki/services/WikiRemoteService";
```

Each Web Service method must authenticate the caller to the Web Service. Authentication consists of a user name and a preconfigured passphrase In this example, we will hard code these values.

```
  private static final String username = "jsmith";
  private static final String passphrase = "passphrase";
```

We will also hard code the domain name in this example.

```
  private static final String domain = "Training";

  public static void main(String[] args) throws Exception
    {
      try
      {
```

Next, we will create a client-side proxy to access the Web Service.

```
      WikiRemoteServiceClient client =
          new WikiRemoteServiceClient();
```

We then set the end point of the proxy to the actual location where we deployed the Web Service.

```
      client.setEndpoint(endpoint);
```

Each Web Service method must pass a security key to authenticate the user and call the method. We can calculate the key based on the user's name and the Web Service-configured passphrase.

```
       String key = KeyEncryptor.genkey(username, passphrase);
```

Using the Web Service proxy, we fetch into an array the information about all the pages in the selected domain: If there is no such domain, the program will throw an exception. If the domain does not contain any pages, the program returns an empty array.

```
      PageInfo[] pages = client.getAllPageInfo(domain, key);
      System.out.println("Pages in " + domain + " domain:");
```

Throughout the pages array, print the name of each page using the getter method `getName()`.

```
      for (int i = 0; i < pages.length; i++)
      {
        System.out.println("  " + pages[i].getName());
      }
    }
  }
```

If there is an exception, the program captures the error and prints it.

```
    catch (Exception e)
    {
      System.out.println("Exception: " + e);
    }
  }
}
```

6.4.1.4 Creating and Using a Data Control

To integrate wiki into your WebCenter application, you can also use Oracle JDeveloper's Web Service Data Control wizard to create a data control based on the Oracle WebCenter Wiki Web Service. Once you have created the data control, you can drag and drop wiki operations onto your WebCenter application page.

6.4.2 Sample Portlets

Oracle WebCenter Wiki contains four sample JSR 168 portlets, which you can add to a page in your WebCenter application to enable your users to access certain operations in the wiki. When you install Oracle WebCenter Wiki, you automatically install the sample portlet producer. You can access the portlet producer test page at:

`http://`*`host`*`:`*`port`*`/wikiproducer/info`

> **Note:** Use the host and port for the location to which you deployed Oracle WebCenter Wiki. The source of the portlets is contained in the file `portlet_producer_sample.zip` in the Oracle WebCenter Wiki files.

The test page shown in Figure 6–14 displays. Because the portlets use Oracle's extension to pass parameters, use the WSRP 2.0 URL when you register the producer to a WebCenter application:

`http://host:port/wikiproducer/portlets/wsrp2?WSDL`

Figure 6–14 Sample Portlet Producer Test Page

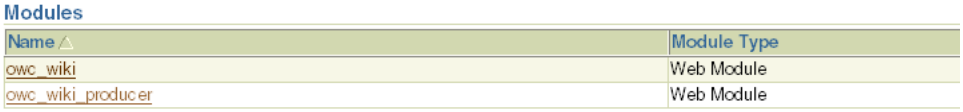
6.4.2.1 Setting Up the Portlet Environment

The sample portlets use an environment entry for the Web Service endpoint.

To set up the environment entry, perform the following steps:

1. In Oracle WebCenter Wiki, update the Environment Entry in Enterprise Manager. First, log in to Enterprise Manager.
2. Locate the deployed Oracle WebCenter Wiki application. The application root should be `owc_wiki`.
3. Click the application, for example **owc_wiki**.
4. Click the **owc_wiki_producer** module, as shown in Figure 6–15.

Figure 6–15 OWC_Wiki_Producer Module



5. On the Administration page, click **Environment Entry Mappings**.
6. Locate the task `/oracle/webCenter/owcWiki/portlet/webServiceEndPoint`.
7. Set this value to the Web Service Endpoint:

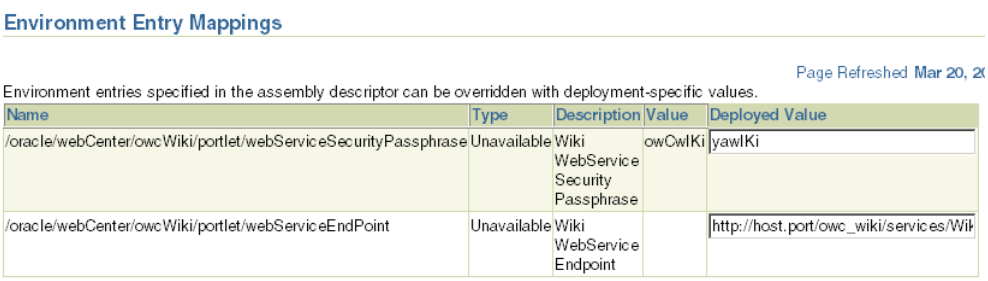
   `http://host:port/owc_wiki/services/WikiRemoteService`

8. Then, update the passphrase for the portlet producer to match the one you created in Section 6.2.2.3, "Generating the Passphrase".

To update the passphrase for the portlet producer, on the Environment Entry mappings page, locate the task: Environment Entry Mappings and update the value for `/oracle/webCenter/owcWiki/webServiceSecurityPassphrase`.

9. Update the value with the passphrase you entered in Section 6.2.2.3, "Generating the Passphrase". When you update the deployed value, deploying the application again will not overwrite the value you have set.

Figure 6–16 Environment Entry Mappings



Now that you have set up the environment for the portlets, you can view the sample portlets in a WebCenter application, as described in Section 6.4.2.2, "Viewing the Sample Portlets".

6.4.2.2 Viewing the Sample Portlets

Table 6–6 contains descriptions of the four sample portlets included with Oracle WebCenter Wiki.

Table 6–6 Sample Portlets and Their Descriptions

Portlet	Description
Page Information	This portlet shows the information about a selected wiki page. You can also customize this portlet to enable the user to view or edit the selected page.
Create Domain	This portlet enables the user to create a new domain in the wiki.
Create Page	This portlet enables the user to select an existing domain in the wiki and create a new page within the domain.
Select Page	This portlet enables the user to select an existing domain in the wiki, then choose an existing page from the wiki to display in a new browser window.

To view the sample portlets, perform the following steps:

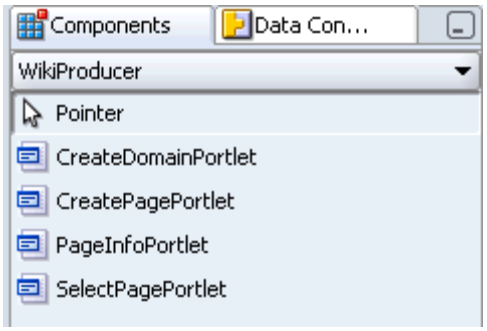
1. In Oracle JDeveloper, create a new WebCenter application.
2. Register the WSRP producer with your new application, using the following URL endpoint for the WSDL:

   `http://host:port/wikiproducer/portlets/wsrp2?WSDL`

Note: The host and port refer to the computer hosting the OC4J instance containing the sample portlets. For steps to register a WSRP producer, refer to Section 4.3.1.1, "Registering WSRP Portlet Producers".

3. After you register the WSRP producer, return to the Applications Navigator.
4. Within the ViewController project, create a JSF JSP page, and select all Portlet Technologies for the page.
5. You can now drag and drop the sample portlets from the Component Palette, as shown in Figure 6–17, onto your JSF JSP page.

Figure 6–17 Component Palette



6. After you add the four portlets to your page, run the page to your browser. You should see page shown in Figure 6–18 and Figure 6–19.

Figure 6–18 Sample Portlets: CreateDomainPortlet and CreatePagePortlet

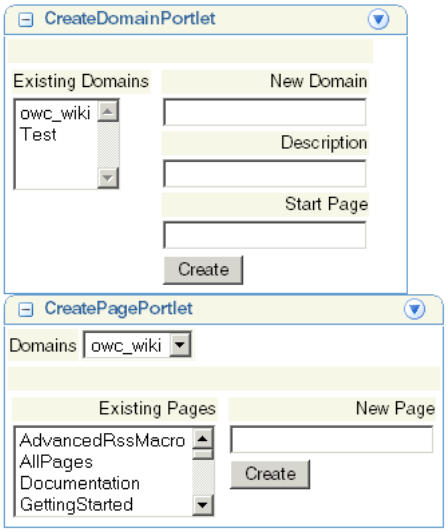
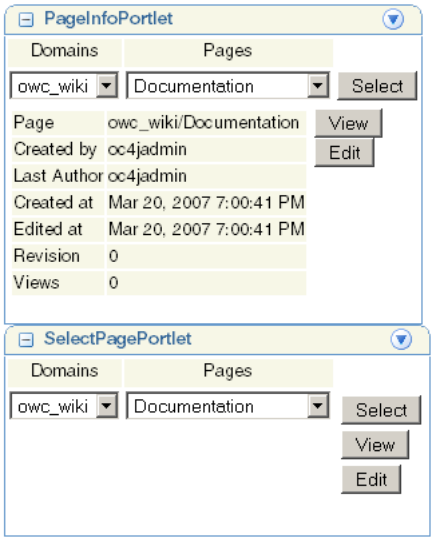


Figure 6–19 Sample Portlets: PageInfoPortlet and SelectPagePortlet



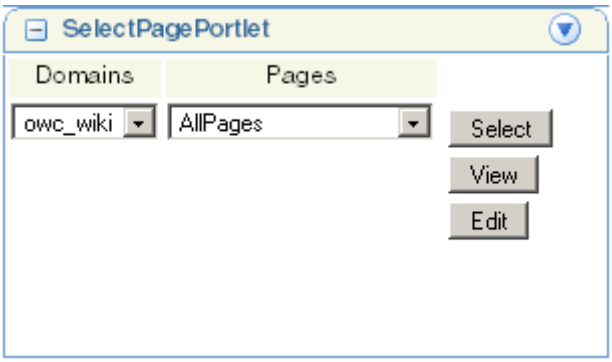
Note: If you are unable to view the sample portlets, you may need to implement security on your page. To do so, while your page is open in Oracle JDeveloper, choose **Tools > ADF Security Wizard**, then accept the default settings in the wizard.

When you run your application to your browser, you will be prompted with a login prompt. Enter the default administrator user name and password to view the portlets (for example, `oc4jadmin` and `welcome`).

6.4.2.3 SelectPagePortlet

As shown in Figure 6–20, this portlet enables users to select a domain, then a page in the selected domain. Oracle WebCenter Wiki then passes the name of the selected domain and page to the WebCenter page. This portlet also enables the user to easily view or edit the selected page.

Figure 6–20 SelectPagePortlet



The existing domains display in the first drop-down list. Pages within the currently selected domain display in the second drop-down list. When the user selects a

different domain, the second drop-down list refreshes and shows the pages of the newly chosen domain.

The two buttons open the selected page in a new browser window. The View button opens the page in the wiki's View mode, and the Edit button displays the page in the wiki's Edit mode.

In addition to the general customizable portlet attributes, this portlet has the customizable attributes shown in Figure 6–21:

Figure 6–21 Customize Page of the Select Page Portlet

Table 6–7 shows the customizable options and their descriptions.

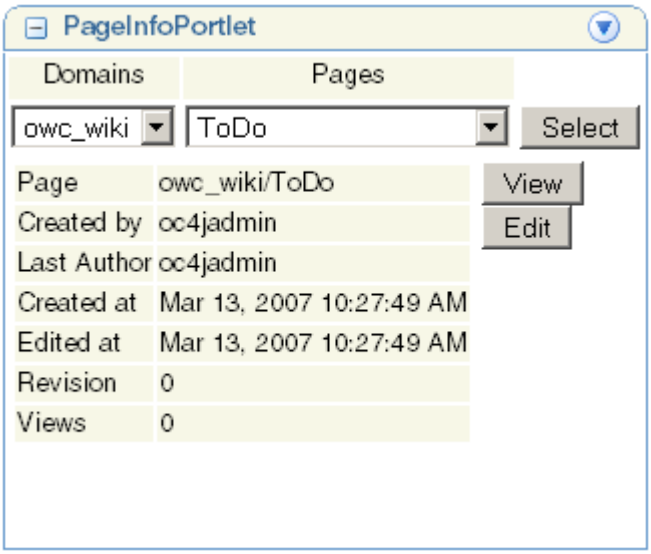
Table 6–7 SelectPagePortlet Customizable Options

Customizable Option	Description
Has Page Select Section	Deselecting this check box hides the Page drop-down list. You can also use this portlet to enable users to only select the domain.
Has Select Button	Toggles the display of the Select button.
Has View Button	Toggles the display of the View button.
Has Edit Button	Toggles the display of the Edit button.

6.4.2.4 PageInfoPortlet

This portlet enables users to view information about a selected wiki page as shown in Figure 6–22.

Figure 6–22 PageInformation Portlet



The information displayed about the page includes the name of the domain and page, the user who created and who edited the last time the page, and the dates for the page creation. Revisions show how many times the users modified the page since creation. Views show how many times users have displayed the page since creation.

Clicking the View or Edit button opens the page in wiki, within the same browser window.

In addition to the general customizable portlet attributes, this portlet has the customizable attributes shown in Figure 6–23:

Figure 6–23 Customize Page of the Page Information Portlet

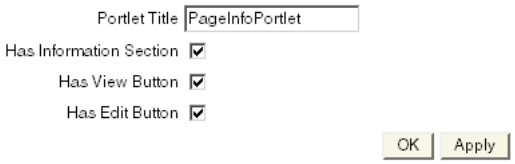


Table 6–8 shows the customizable attributes and their descriptions.

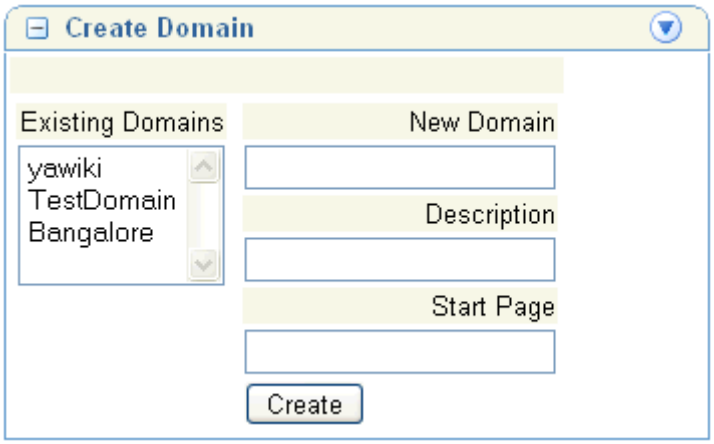
Table 6–8 Page Information Portlet Customizable Options

Customizable Attribute	Description
Has Information Section	Deselecting this check box hides the information displayed about the page. Only the domain and page name display.
Has View Button	Toggles the display of the View button.
Has Edit Button	Toggles the display of the Edit button.

6.4.2.5 CreateDomainPortlet

Your users can create a new domain using this portlet as shown in Figure 6–24.

Figure 6–24 CreateDomainPortlet



The Existing Domains drop-down list displays the domains that already exist within the wiki. The user can enter a new domain name, a description, and a start page for the new domain in this portlet. Clicking on the Create button creates the new domain. A confirmation or error message displays within the portlet to alert the user whether Oracle WebCenter Wiki created the page.

This portlet has no additional customizable attributes.

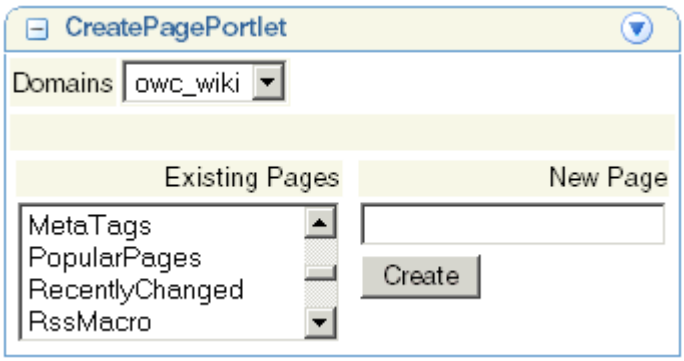
Note: If you are unable to view this sample portlet, you may need to implement security on your page. To do so, while your page is open in Oracle JDeveloper, choose **Tools > ADF Security Wizard**, then accept the default settings in the wizard.

When you run your application to your browser, you will be prompted with a login prompt. Enter the default administrator user name and password to view the portlets (for example, `oc4jadmin` and `welcome`).

6.4.2.6 CreatePagePortlet

Your users can select a domain and create a new page within that domain using the Create Page Portlet as shown in Figure 6–25.

Figure 6–25 CreatePagePortlet



The user can choose a domain and view the existing pages within that domain. Then, the user can enter a new page name in the New Page field and click the Create button to create the page. A confirmation or error message displays within the portlet to alert the user whether Oracle WebCenter Wiki created the page.

This portlet has no additional customizable attributes.

Note: If you are unable to view this sample portlet, you may need to implement security on your page. To do so, while your page is open in Oracle JDeveloper, choose **Tools > ADF Security Wizard**, then accept the default settings in the wizard.

When you run your application to your browser, you will be prompted with a login prompt. Enter the default administrator user name and password to view the portlets (for example, `oc4jadmin` and `welcome`).

6.4.3 Writing a Portlet

This section shows you how to create a portlet to access the wiki functionality using the Web Service interface described in Section 6.4.1, "Oracle WebCenter Wiki Web Service Interface". You can then add this portlet to a WebCenter application page.

The sample portlets include a Page Information portlet that describes a selected page in the wiki, including such information as: the name of the domain, the name of the page, the name of the author and the date of the latest update. The following code illustrates how to write a portlet using the Oracle WebCenter Wiki Web Service interface. This portlet is a simplified version of the Page Information portlet available in the sample files.

This portlet accepts two parameters, `domain` and `page`, and displays information about the page based on these values. The portlet also has two buttons through which you can view or edit the selected page using the wiki user interface.

The JSP in this example implements the show mode for this JSR 168 portlet. In the following highlighted code, replace the localhost and port values with the location where your wiki is deployed.

Note: In this example, we have hard coded the Web Service endpoint in the endpoint string variable. In a complex portlet, you can use a different value, such as one from a customizable attribute.

```
<%@ page contentType="text/html"
    pageEncoding="windows-1252"
    import="javax.portlet.*,
            java.util.*,
            java.text.*, oracle.webcenter.wiki.security.KeyEncryptor,
    oracle.webcenter.wiki.ws.*"
%>
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet"%>
<portlet:defineObjects/>

<%
  String endpoint = http://localhost:6688/owc_wiki/services/WikiRemoteService
```

Here, you get the values of the `domain` and `page` parameters:

```
String domainName = renderRequest.getParameter("domain");
if ((domainName != null) && (domainName.length() == 0))
  domainName = null;
String pageName = renderRequest.getParameter("page");
if ((pageName != null) && (pageName.length() == 0))
  pageName = null;
```

Next, provide an error message if the user does not provide a value for either of the parameters:

```
  if ((domainName == null) || (pageName == null))
  {
%>
<i>No page selected for display!</i>
```

Use the Date format to display the Date data type in medium detail and using the current locale:

```
<%
  }
  else
  {
    DateFormat df =
      DateFormat.getDateTimeInstance(DateFormat.MEDIUM,
                                     DateFormat.MEDIUM,
                                     renderRequest.getLocale());
```

Create a proxy object and set the endpoint:

```
    WikiServiceSoapHttpPortClient client =
      new WikiServiceSoapHttpPortClient();
    client.setEndpoint(endpoint);
```

Use the passphrase generated by your administrator (see Section 6.2.2.3, "Generating the Passphrase") to access the secured Web Service method, then use the client proxy's `getPageInfo()` method to obtain the `PageInfo` JavaBean that contains the selected page's attributes:

```
    String username = "jsmith";
    String passphrase = "passphrase";

    KeyEncryptor.genkey(username, passphrase);

    PageInfo info = client.getPageInfo(domainName, pageName, key);
```

If the `getPageInfo` method returns a null value, the selected page does not exist and displays an error message:

```
    if (info == null)
    {
%>
<b>No such page:</b> <%= domainName %> /  <%= pageName %>
<%
    }
```

If the `getPageInfo` method returns a value, the domain name then the page name display:

```
    else
    {
```

```
 %>
   <table>
     <tr><td valign="top">
       <table>
         <tr class="portlet-table-text">
           <td align="left">Page</td>
           <td align="left"><%= domainName %>/<%= pageName %></td>
         </tr>
```

The info object is a JavaBean holding the page's attributes. You can use the getter methods to access the individual attributes. Here, the `info.getAuthor()` object returns the page's author:

```
         <tr class="portlet-table-text">
           <td align="left">Created by</td>
           <td align="left"><%= info.getAuthor() %></td>
         </tr>
```

Similarly, you can use other getter methods to obtain other attributes.

```
         <tr class="portlet-table-text">
           <td align="left">Last Author</td>
           <td align="left"><%= info.getEditor() %></td>
         </tr>
```

Format the date information, such as the creation date, before displaying it:

```
         <tr class="portlet-table-text">
           <td align="left">Created at</td>
           <td align="left"><%= df.format(new Date(info.getCreated().getTime()))
%></td>
         </tr>
         <tr class="portlet-table-text">
           <td align="left">Edited at</td>
           <td align="left"><%= df.format(new Date(info.getModified().getTime()))
%></td>
         </tr>
         <tr class="portlet-table-text">
           <td align="left">Revision</td>
           <td align="left"><%= info.getRevision() %></td>
         </tr>
         <tr class="portlet-table-text">
           <td align="left">Views</td>
           <td align="left"><%= info.getViews() %></td>
         </tr>
       </table>
```

Next, create two buttons to enable users to view or edit the page in wiki. Both buttons have their own separate `<form>` tags and receive the action URL of the form element from the info bean:

```
     </td>
     <td align="left" valign="top">
   <form action="<%= info.getViewURL() %>" method="GET">
     <input type="submit" class="portlet-form-button" value="View"
   </form>
   <form action="<%= info.getEditURL() %>" method="GET">
     <input type="submit" class="portlet-form-button" value="Edit"
   </form>
     </td></tr>
   </table>
```

```
<%
      }
   }
%>
```

7

Integrating Oracle WebCenter Discussions

This chapter explains how you can embed discussion forums in your WebCenter applications. The discussion forum is a powerful service that enables users to share information and discuss topics embedded within your WebCenter application.

7.1 Introduction to Oracle WebCenter Discussions

One of the services that you can integrate into your WebCenter application is discussion forums. Users can browse threads in a convenient portlet to locate pertinent messages, and then go to the discussion forum to read more and add their own posts or replies. In particular, Oracle WebCenter Discussions provides you a J2EE application with an open architecture and extensive features. You can easily integrate Oracle WebCenter Discussions into your WebCenter application.

7.2 Integrating Oracle WebCenter Discussions

This section describes the following:

7.2.1 How to Install and Configure Oracle WebCenter Discussions

To install the Oracle WebCenter Discussions J2EE application, do the following:

1. Unzip `jive_forums_silver_5_1_0.zip` from the companion CD. Follow the installation instructions located in:

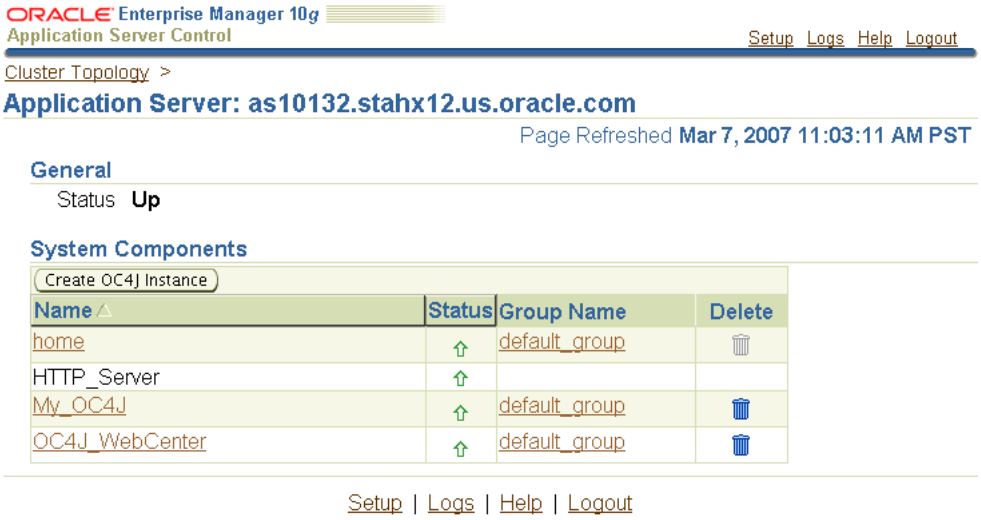
   Note: *`owc_discussions`* is the name of the directory in which you chose to unzip `jive_forums_silver_5_1_0.zip`.

   `owc_discussions``\jive_forums_silver_5_1_0\documentation\install-guide.html`

   When you reach the point in the Oracle WebCenter Discussions installation instructions where you must install the `jiveforums.war` file in an application server, you need to deploy the WAR file through Oracle Enterprise Manager 10*g* as described in the steps that follow.

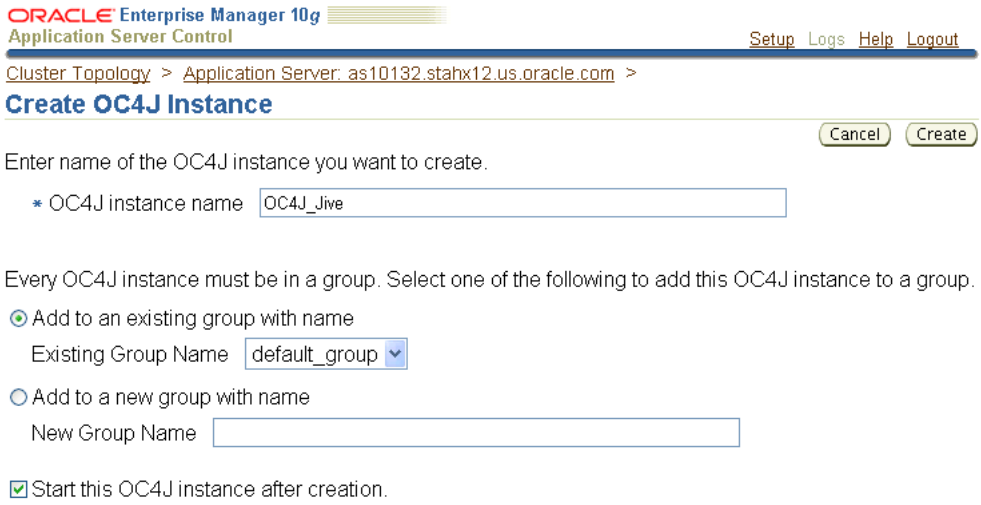
2. Log in to the Application Server Control as an administrator.
3. Click the Oracle Application Server instance to go to its application server page.
4. We recommend you create a new OC4J instance to contain Oracle WebCenter Discussions. Although not recommended, you can use an existing OC4J instance. To create a new OC4J instance, click **Create OC4J Instance** (Figure 7–1).

Figure 7–1 Oracle Application Server Instance



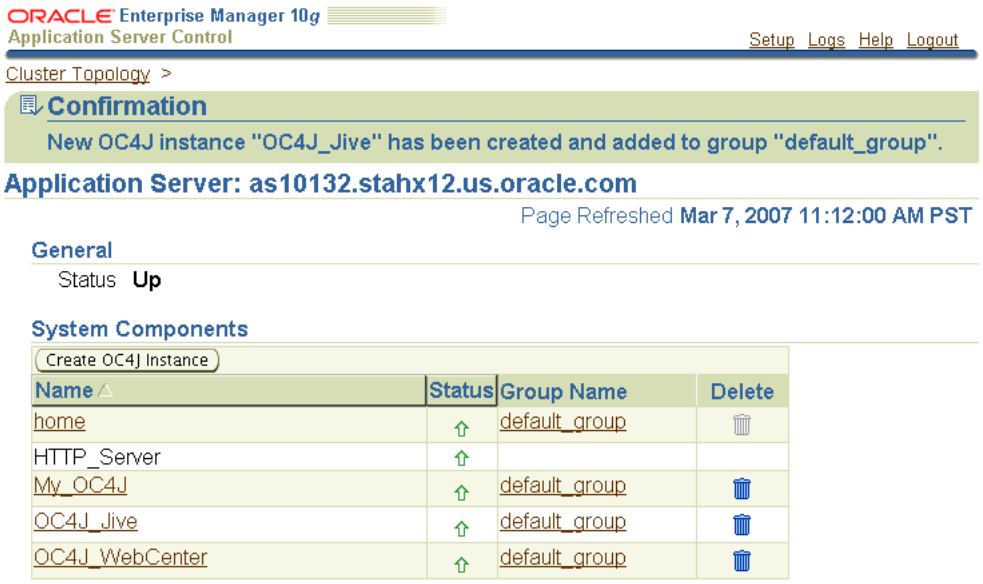
5. Fill in the values for the new OC4J instance (Figure 7–2). Be sure to check **Start this OC4J instance after creation**.

Figure 7–2 Create OC4J Instance



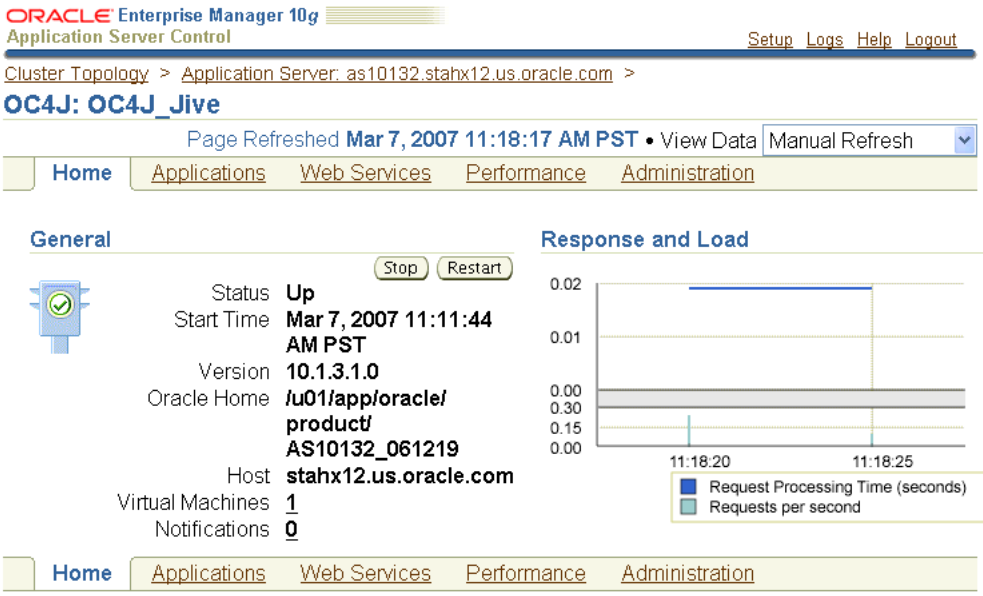
6. Click **Create**. Your instance should now appear in the list of System Components (Figure 7–3).

Figure 7–3 Application Server Page of Application Server Control



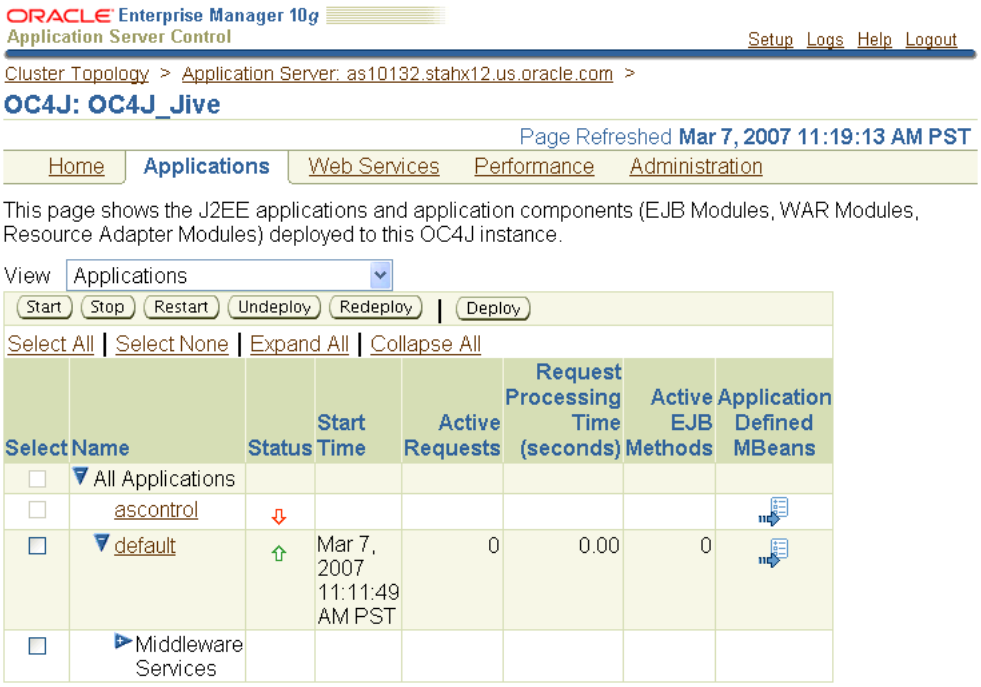
7. Follow the instructions in Section 18.3, "Configuring Your Application Server or Standalone OC4J to Run Portlets" to configure your OC4J instance to run portlets.
8. Click the name of the OC4J instance. The Home tab of the instance appears (Figure 7–4).

Figure 7–4 Home Tab for OC4J Instance



9. Click the **Applications** tab (Figure 7–5).

Figure 7–5 Applications Tab for OC4J Instance

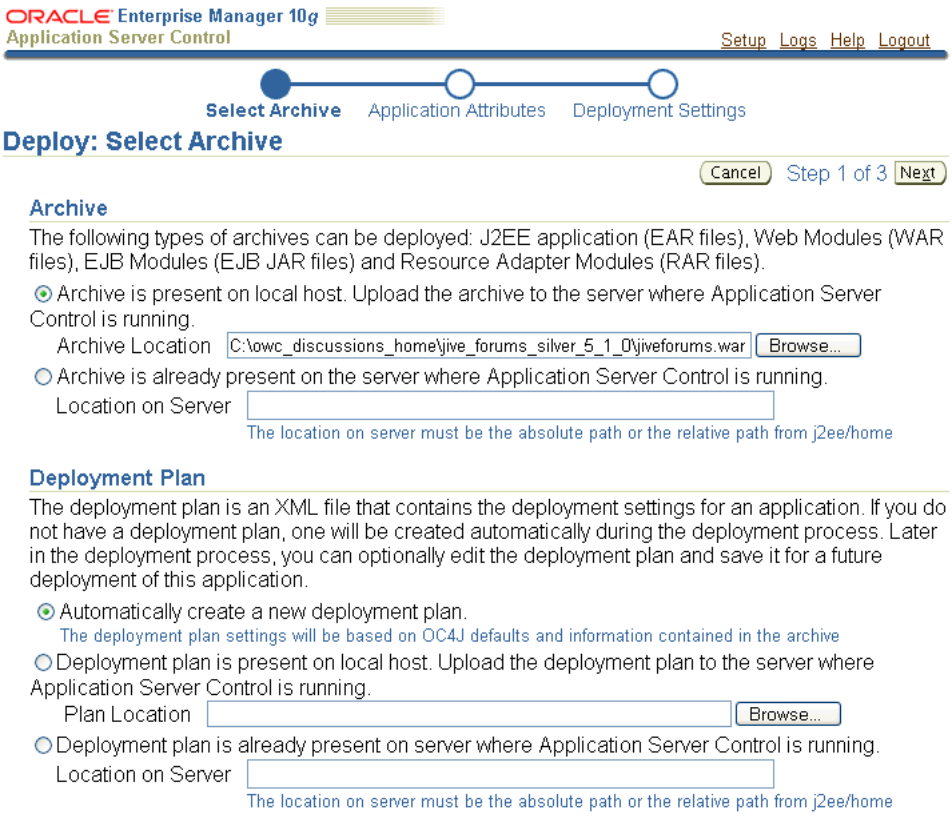


10. Click **Deploy**.
11. Choose the location of the WAR file (local or host) and specify the path (Figure 7–6). From the companion CD, the path is:

    owc_discussions\jive_forums_silver_5_1_0\jiveforums.war

12. Choose **Automatically create a new deployment plan** (Figure 7–6).

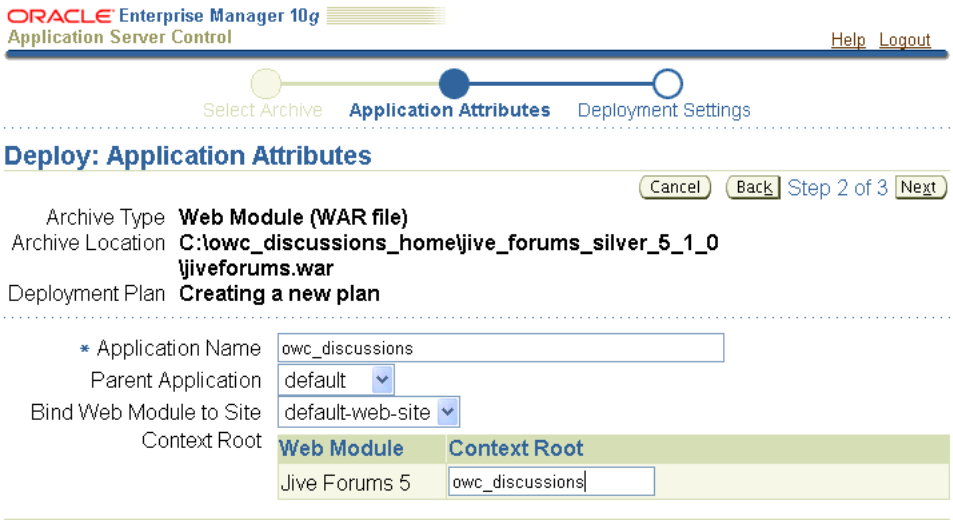
Figure 7–6 Deploy: Select Archive Page



13. Click **Next**. Note that, if the WAR file is local, it may take several minutes to upload to the server.

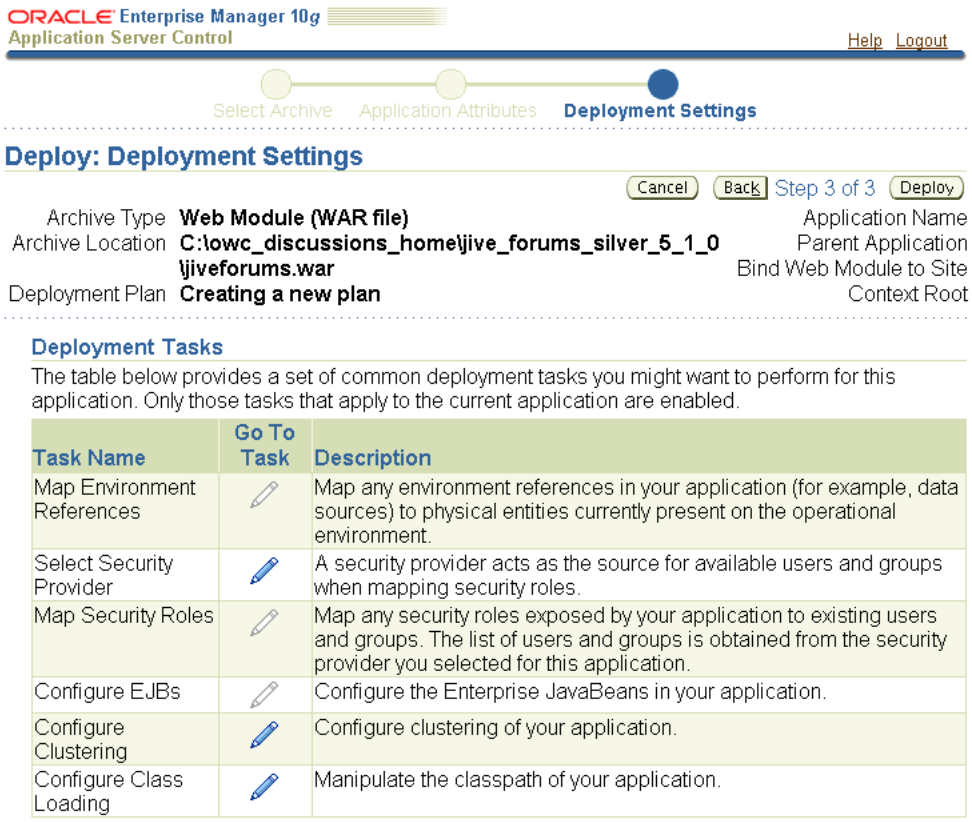
14. Enter **Application Name** (for example, `owc_discussions`). Ensure also that the **Context Root** is something meaningful (for example, `owc_discussions`) because you will use this context root to call your Wiki application from this point forward (Figure 7–7).

Figure 7–7 Deploy: Application Attributes Page



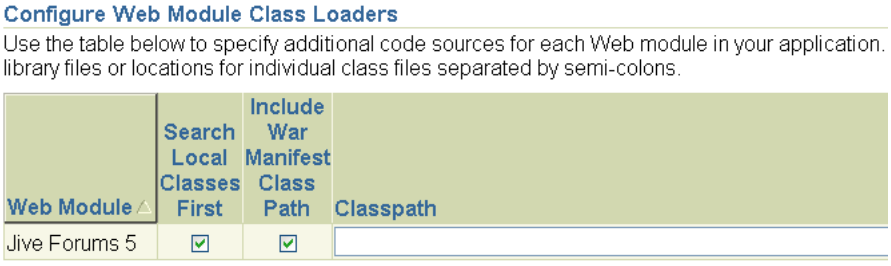
15. Click **Next**.
16. The Deployment Settings page appears (Figure 7–8). Click the **Go to Task** icon for **Configure Class Loading**.

Figure 7–8 Deploy: Deployment Settings Page



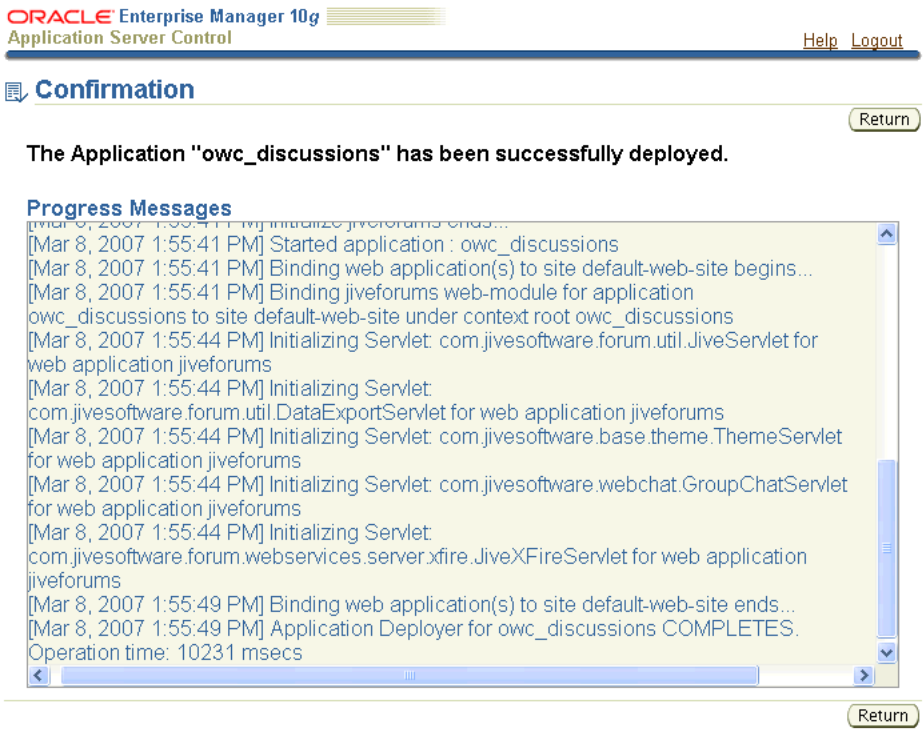
17. From the **Configure Class Loading** page, go to the **Configure Web Module Class Loaders** section (Figure 7–9) and check **Search Local Classes First** for the **Jive Forums 5** Web module.

Figure 7–9 Configure Web Module Class Loaders Section



18. Click **OK**. Note the informational message at the top of the page that indicates that the deployment plan was successfully updated.
19. Click **Deploy**.
20. Once the deployment has completed successfully, you get a confirmation message (Figure 7–10). If the deployment is not successful, review the log and confirm that you had the correct deployment settings.

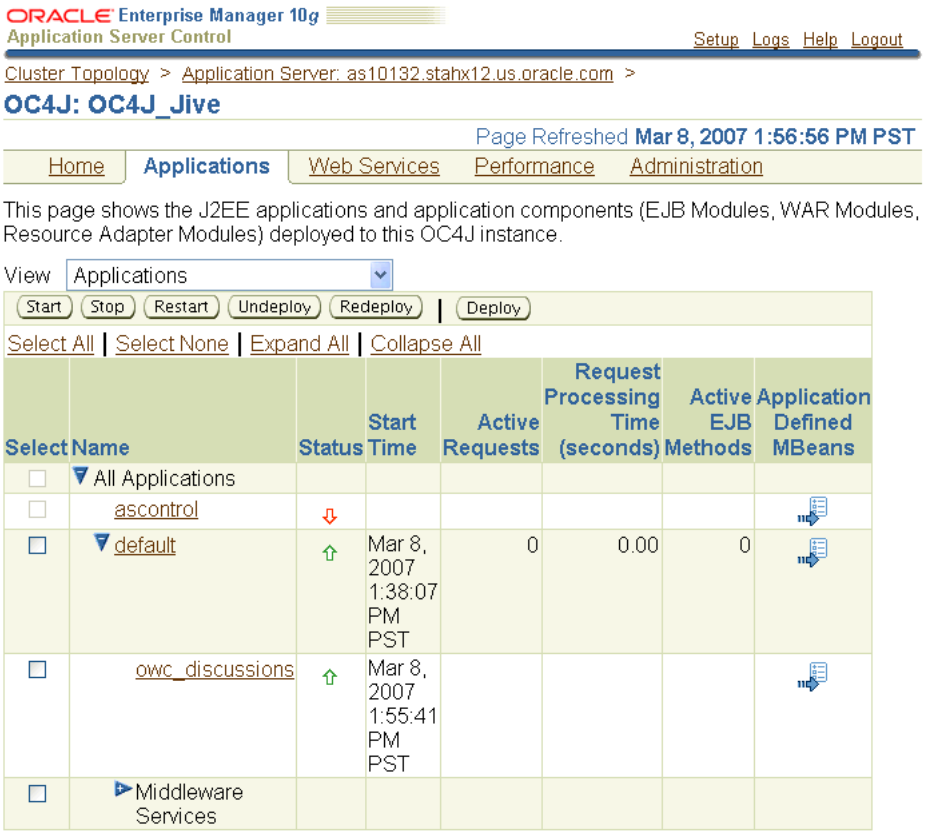
Figure 7–10 Confirmation of Deployment



21. Click **Return**. You should now see the application running in your OC4J instance (Figure 7–11).

> **Note:** If you do not intend to use the discussion forum portlets, restart the OC4J instance now.

Figure 7–11 Applications Tab with Oracle WebCenter Discussions Application Deployed



22. Return to the Oracle WebCenter Discussions installation instructions in `owc_discussions\jive_forums_silver_5_1_0\documentation\install-guide.html` and follow the remainder of the steps. Under the User, Group, and Authentication Systems configuration, the LDAP credential store is not supported. The recommended option is the default user management. Therefore, when you go through the setup tool, choose the default option when prompted to **Choose a user, group and authentication system**. This default setup is overridden when you perform the steps in Section 7.2.2, "How to Configure Java SSO with Your Oracle WebCenter Discussions Application and Portlet".

23. To confirm that you have successfully set up Oracle WebCenter Discussions, go to the Admin Console and attempt to login as the administrator you created during the installation process:

    `http://host:port/owc_discussions/admin/`

24. If you are able to successfully login to the Admin Console, you can proceed to Section 7.2.2, "How to Configure Java SSO with Your Oracle WebCenter Discussions Application and Portlet" and perform the steps to integrate Oracle WebCenter Discussions and the discussion forum portlet with Java SSO.

7.2.2 How to Configure Java SSO with Your Oracle WebCenter Discussions Application and Portlet

To integrate the discussion forum portlet and Oracle WebCenter Discussions Web application in your WebCenter application, you must perform the steps in this section

after you have successfully deployed and configured the Oracle WebCenter Discussions Web application.

These steps describe how to properly configure security for Oracle WebCenter Discussions. The Java SSO integration requires custom classes to override the standard `AuthFactory` and `UserManager` classes. In `OracleSSOAuthFactory` and `OracleSSOUserManager`, Oracle ADF APIs check user authentication and user manager. In addition, you must override a few action classes for user login/logout, and filter classes for presence and administration. Finally, to integrate the ADF authentication servlet for Java SSO login, you need to change the `web.xml` and `orion-application.xml`.

To integrate your discussion forum portlet and the Oracle WebCenter Discussions application, perform the following steps:

1. To enable single sign-on between the Oracle WebCenter Discussions application and the discussion forum portlet, you use Java SSO to authenticate users from a central user data store. You must synchronize an Oracle WebCenter Discussions user and the Oracle Application Server administrator user. When using the JAZN authentication model, you create an `oc4jadmin` user in the Oracle WebCenter Discussions administration interface and assign it administrative privileges as follows:

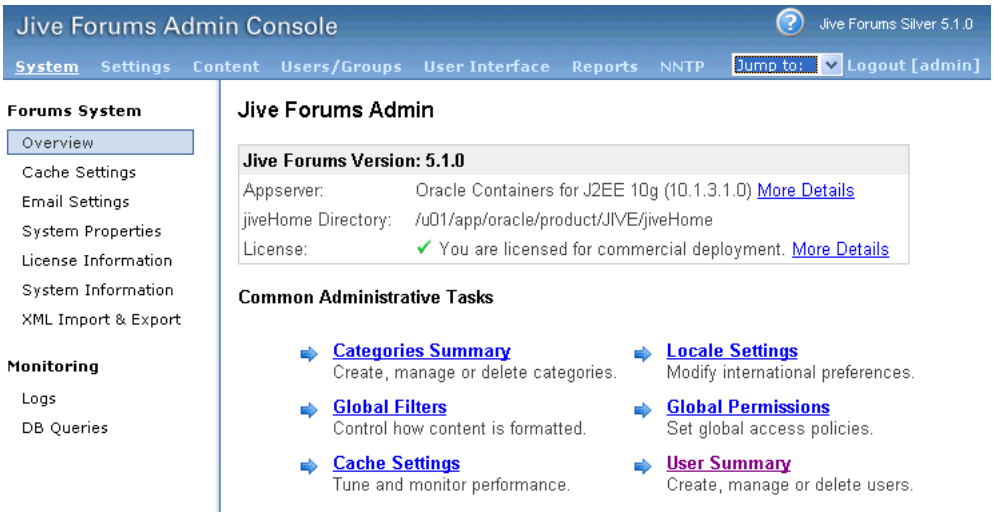
   Note: It is mandatory that you create the `oc4jadmin` user at this point. Otherwise, you will not be able to log in to the Oracle WebCenter Discussions application later.

   a. Open the Oracle WebCenter Discussions administration interface by going to the following URL:

      `http://host:port/owc_discussions/admin/`

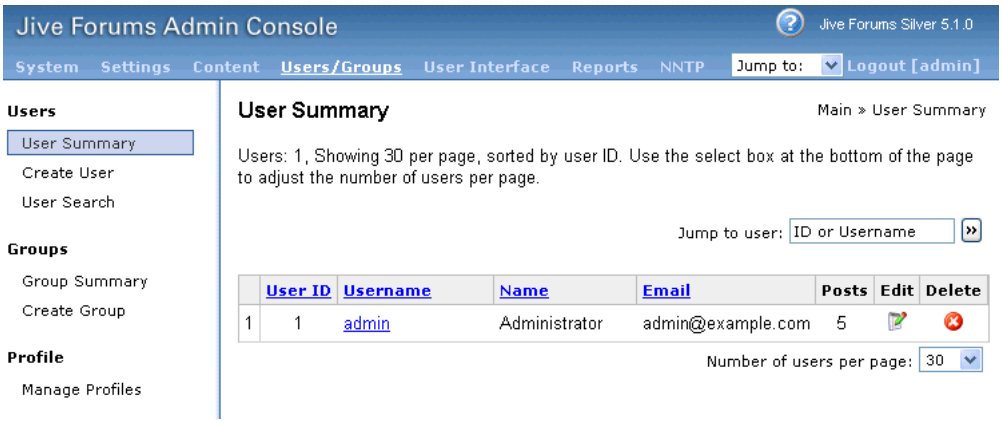
   b. Login using the administrator user name and password you created during the installation of Oracle WebCenter Discussions. The Oracle WebCenter Discussions Admin Console appears as shown in Figure 7–12.

Figure 7–12 Oracle WebCenter Discussions Admin Console



   c. Click the **Users/Groups** tab (Figure 7–13).

Figure 7–13 User/Groups Tab of Admin Console



d. Click **Create User** from the left pane.

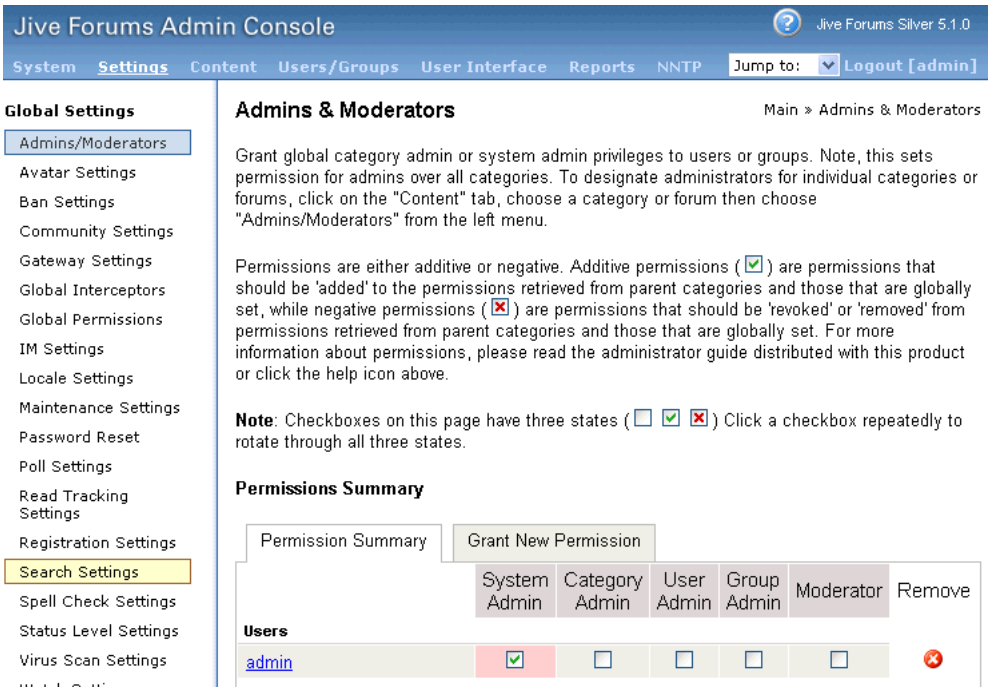
e. Specify `oc4jadmin` for **Username** and fill in the other required properties (Figure 7–14).

Figure 7–14 Create User Page

f. Click **Create User**.

g. Once the `oc4jadmin` user is created, you need to grant it administrator privileges. Click the **Settings** tab (Figure 7–15).

Figure 7–15 Settings Tab of the Admin Console

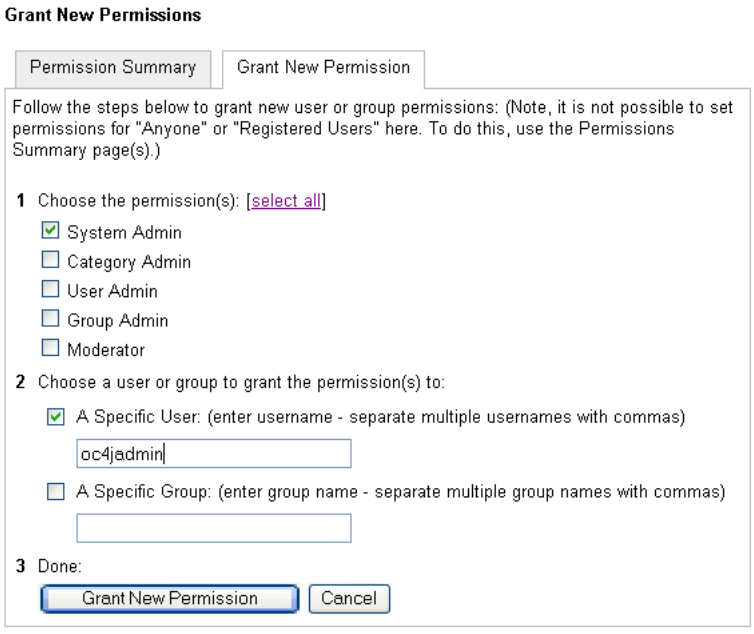


h. Click **Grant New Permissions**.

i. For **Choose the permission(s)**, select **System Admin** (Figure 7–16).

j. For **Choose a user or group to grant the permission(s) to**, select **A Specific User** and enter `oc4jadmin` in the adjoining field (Figure 7–16).

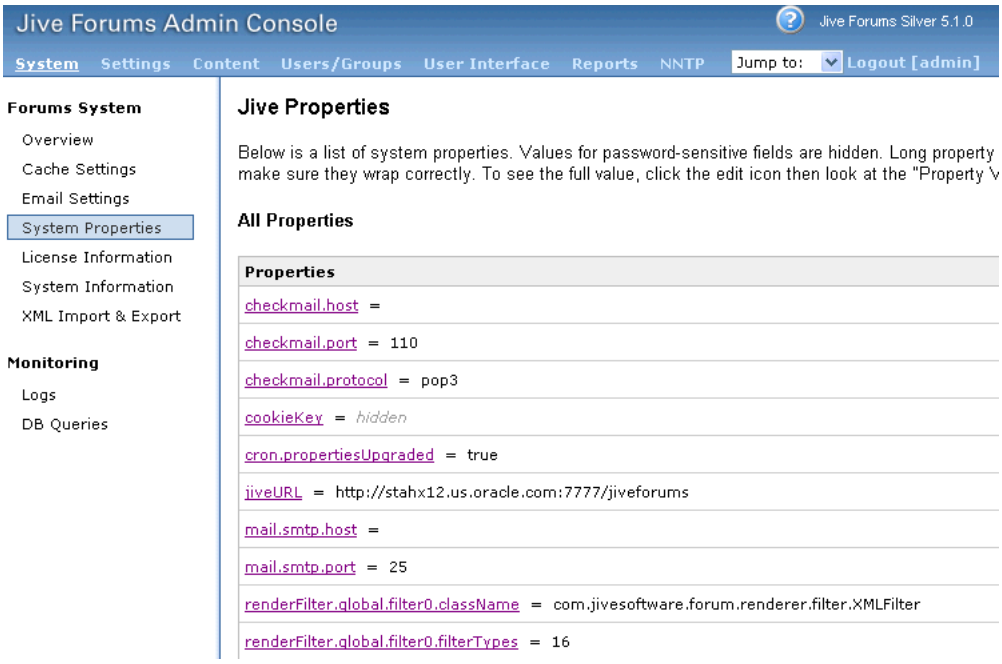
Figure 7–16 Grant New Permissions



k. Click **Grant New Permission**.

2. Click the **System** tab. You need to change the default classes for `AuthFactory` and `UserManager` by adding two new properties.
3. Click **System Properties** (Figure 7–17) and scroll to the bottom of the page, where you should see the **Add new property** section.

Figure 7–17 System Properties of System Tab

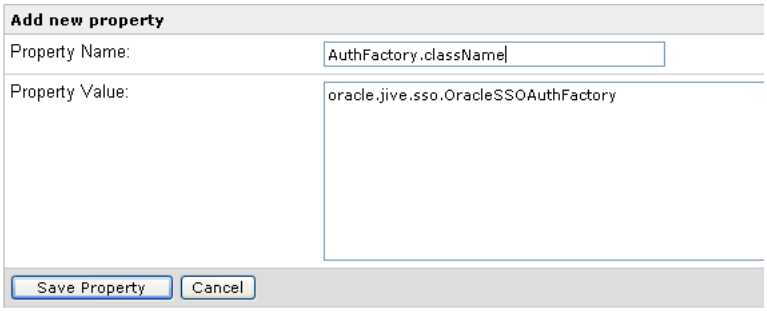


4. For **Property Name**, enter `AuthFactory.className`.
5. For **Property Value**, enter:

   `oracle.jive.sso.OracleSSOAuthFactory`

6. Click **Save Property** (Figure 7–18).

Figure 7–18 Add new property Section



7. Repeat the previous three steps using `UserManager.className` as the **Property Name** and the following as the **Property Value**:

   `oracle.jive.sso.OracleSSOUserManager`

   This class extends the Oracle WebCenter Discussions `DbUserManager`.

8. To complete the portlet setup process, you now must modify some configuration files. The recommended method of making these modifications is to run `deploy-jive-portlet.jar`, which can be found on the companion CD. `deploy-jive-portlet.jar` configures `web.xml`, `xwork-community.xml` (in `jiveforums-5.1.0.jar`), `orion-application.xml`, and `jive_startup.xml`, and it unzips `oracle-jive-portlet.zip`. We recommend the use of `deploy-jive-portlet.jar` because it reduces the risk of manual errors, such as typos, in the configuration files. It also automatically backs up the previous versions of your configurations files, which can be useful if you later undeploy the portlet and return to your previous configuration

   Alternatively, if for any reason you want to perform this process manually, you can follow the steps in Section 7.2.2.2, "Manual Configuration Steps for Portlet Deployment".

   To run `deploy-jive-portlet.jar`, perform the following steps:

   a. Copy `deploy-jive-portlet.jar` and `oracle-jive-portlet.zip` from the companion CD to:

      `j2ee/`*`OC4J_instance`*`/applications/owc_discussions/jiveforums`

   b. Ensure that you have JDK 1.5 in your PATH variable. If not, you can set PATH to point to the JDK 1.5 found in *`ORACLE_HOME`*`/jdk/bin`, where *`ORACLE_HOME`* is the Oracle WebCenter home.

   c. From Enterprise Manager, stop the OC4J instance that runs your Oracle WebCenter Discussions application.

   d. Run the following command line:

      ```
      java -client -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80
       -jar deploy-jive-portlet.jar
      ```

      Note: If you are running through a proxy, you need to specify values for `http.proxyHost` and `http.proxyPort` on the command line as well.

      If you need to undeploy for some reason, you can run the following command line:

      ```
      java -client -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80
       -jar deploy-jive-portlet.jar undeploy
      ```

   e. When you run the command line, the tool prompts you for the full path to your `jiveHome` directory. Enter the path that you used when you set up the `jiveHome` directory according to the instructions in:

      `owc_discussions``\jive_forums_silver_5_1_0\documentation\install-guide.html`

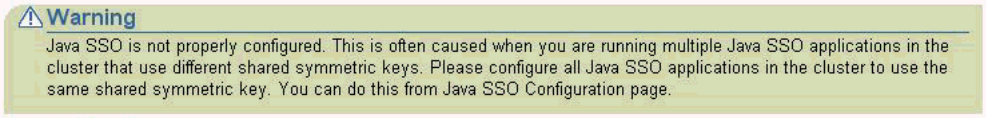
9. Delete the following directory:

   `j2ee/`*`OC4J_instance`*`/application-deployment/owc_discussions/`

10. Go back to Enterprise Manager and start the OC4J instance. Java SSO should appear as an application under the OC4J instance. Once the instance is started, confirm that Java SSO is running. If Java SSO is down, then start it.

When you start Java SSO, you might get a warning message (Figure 7–19) that Java SSO is not properly configured. This warning indicates that you are running multiple Java SSO applications in a cluster. To ensure that you properly configure Java SSO for a clustered environment, refer to Section 7.2.2.1.1, "Clustered Configurations for Java SSO"

Figure 7–19 Warning for Java SSO Configuration



11. Upon successful completion of these steps, your Oracle WebCenter Discussions software and the sample portlet are installed and ready for use.
12. To create new users and start working with discussion forums, you need to create users in your underlying security model. In this case, we used the Java SSO model. For information on Java SSO security and how to create users, refer to *Oracle Containers for J2EE Security Guide*. If you are using file-based security, refer to Section 7.2.2.1.3, "File-Based Security" for configuration tips.

    When you log in with a new user in Java SSO, it creates the same user in the Oracle WebCenter Discussions user management with normal user privileges. To grant this user administrator privileges, perform the following steps:

    Note: The steps that follow are similar to the ones you performed on the `oc4jadmin` user earlier in this procedure.

    ---

    a. Open the Oracle WebCenter Discussions administration interface by going to the following URL:

       `http://host:port/owc_discussions/admin/`

    b. Login using the administrator user name and password you created during the installation of Oracle WebCenter Discussions. The Oracle WebCenter Discussions Admin Console appears.

    c. From the Admin Console, click the **Settings** tab.

    d. Click **Grant New Permissions**.

    e. For **Choose the permission(s)**, select **System Admin**.

    f. For **Choose a user or group to grant the permission(s) to**, select **A Specific User** and enter the name of the user in the adjoining field.

    g. Click **Grant New Permission**.
13. You can also use the Admin Console to create and manage categories, forums, users and groups. Refer to the *Jive Forums Administrator's Guide* (`forums-admin-guide.pdf`) on the companion CD for more information.
14. For more information on consuming the discussion forum portlet in your WebCenter application, refer to Section 7.2.3, "How to Consume the Discussion Forum Portlet in Your WebCenter Application".

7.2.2.1 Configuration Tips

This section provides configuration tips for your Oracle WebCenter Discussions application and portlet:

- Clustered Configurations for Java SSO
- Database Dependency
- File-Based Security

7.2.2.1.1 Clustered Configurations for Java SSO You can set `custom.sso.key.alias` from Enterprise Manager. Click **Java SSO Configuration** at the bottom of the **Cluster Topology** page for your application server instance. Refer to the OC4J Java Single Sign-On chapter in the *Oracle Containers for J2EE Security Guide*.

When you configure Java SSO from Enterprise Manager, users may encounter an error where the Oracle Application Server home page displays when they try to logout. To correct this problem, delete the following properties from `jazn.xml`:

```
property name="custom.sso.cookie.domain" value=""
property name="custom.sso.url.param" value=""
```

7.2.2.1.2 Database Dependency You cannot change the underlying database of Oracle WebCenter Discussions. If you try to redeploy the Oracle WebCenter Discussions application to a new database, it will not function correctly because it relies on some customized features of the original Oracle WebCenter Discussions database.

7.2.2.1.3 File-Based Security In general, we recommend that you use LDAP for your user data store. In cases where you must use file-based security, you should be aware of the following issue:

- When configuring the file-based security provider for a clustered OC4J, JAZN does not pick up the configured `system-jazn-data.xml` file. Therefore, when you create a new user from Enterprise Manager, it is only added in `j2ee/home/config/system-jazn-data.xml` and is not available in the other, clustered OC4J instances.

  To resolve this issue, you must manually copy and paste the user and role from `j2ee/home/config/system-jazn-data.xml` to the clustered *`OC4J_HOME`*`/config/system-jazn-data.xml`.

7.2.2.2 Manual Configuration Steps for Portlet Deployment

As an alternative to running `deploy-jive-portlet.jar`, you can manually perform the following steps:

1. Extract `xwork-community.xml` from *`ORACLE_HOME`*`/j2ee/`*`OC4J_instance`*`/applications/owc_discussions/`*`owcd_root_context`*`/WEB-INF/lib/jiveforums-5.1.0.jar`. To extract the file, run the following command:

   ```
   jar xvf jiveforums-5.1.0.jar xwork-community.xml
   ```

> **Note:** For the `jar` commands in these steps, we recommend using the `jar` executable located in *`ORACLE_HOME`*`/jdk/bin`.

2. Open `xwork-community.xml` in a text editor and modify all of the login/logout actions between the `<!-- Base actions -->` and `<!--Default skin -->` tags to use the classes available in the `jsso-action-classes.xml` file located in the following directory:

   ORACLE_HOME/j2ee/*OC4J_instance*/applications/owc_discussions/*owcd_root_context*/WEB-INF

3. After making the changes, save the file and run the following command to copy it back into the JAR file:

   ```
   jar uvf jiveforums-5.1.0.jar xwork-community.xml
   ```

4. Log in to the Application Server Control as an administrator.
5. Click the Oracle Application Server instance to go to its application server page.
6. Click the OC4J instance where you are running the Oracle WebCenter Discussions application.
7. Click **Stop**.
8. Make a backup of the `web.xml` and `/WEB-INF/lib/jiveforums-5.1.0.jar` files from the following directory:

   ORACLE_HOME/j2ee/*OC4J_instance*/applications/owc_discussions/*owcd_root_context*

9. The portlet is available in the `oracle-jive-portlet.zip` file on the companion CD. Unzip the files into:

   ORACLE_HOME/j2ee/*OC4J_instance*/applications/owc_discussions/*owcd_root_context*

10. To configure the discussion forum portlet with the Oracle WebCenter Discussions application, you need to modify the `web.xml` file in *ORACLE_HOME*/j2ee/*OC4J_instance*/applications/owc_discussions/*owcd_root_context*/WEB-INF as follows:

    a. Change the XML header to look like the following:

       ```
       <?xml version = '1.0' encoding = 'UTF-8' standalone = 'yes'?>
       ```

    b. Open the file called `jive-portlet-for-web.xml` located in *ORACLE_HOME*/j2ee/*OC4J_instance*/applications/owc_discussions/*owcd_root_context*/WEB-INF.

    c. Select all of the content of the file and copy and paste it into `web.xml`, right after the `<web-app>` tag. This changes alters the security model and makes some role changes. Note that the security role and authentication constraints can be changed to be synchronized with roles and users added from the Enterprise Manager console.

    d. Modify the `AdminActionFilter` and `PresenceFilter` in the `web.xml` file as shown in Example 7–1 to override the Java SSO integration. If these filters are not already present, you need to create them.

Example 7–1 web.xml Filter Modifications

```
<filter>
 <filter-name>AdminActionFilter</filter-name>
  <filter-class>
   oracle.jive.sso.actions.SSOAdminActionFilter
  </filter-class>
</filter>
```

```
<filter>
 <filter-name>PresenceFilter</filter-name>
  <filter-class>
   oracle.jive.sso.actions.SSOPresenceFilter
  </filter-class>
</filter>
```

11. You must now modify the `orion-application.xml` file to point to the correct JAZN provider. If you have not previously modified the `orion-application.xml` located in *ORACLE_HOME*`/j2ee/`*OC4J_instance*`/applications/owc_discussions/`*owcd_root_context*`/META-INF`, you can simply replace it with the `orion-application.xml.default` file located in the same directory.

    If you have previously made changes to `orion-application.xml` that you need to preserve, then you can edit the file to add or replace the JAZN provider to look similar to the following:

    ```
    <library path="./adf"></library>
    <jazn provider="XML" default-realm="jazn.com">
      <jazn-web-app auth-method="CUSTOM_AUTH"/>
    </jazn>
    ```

12. To clear any cached copies, delete the following directory:

    ORACLE_HOME`/j2ee/`*OC4J_instance*`/applications/owc_discussions/jiveforums/j2ee/home/applications-deployments/`*jive_directory*

13. Set `admin.tryAlternativeLogin` in `jiveHome/jive_startup.xml` by adding the following lines somewhere before the `</jive>` tag:

    ```
    <admin>
      <tryAlternativeLogin>true</tryAlternativeLogin>
    </admin>
    ```

14. Return to step 10 in Section 7.2.2, "How to Configure Java SSO with Your Oracle WebCenter Discussions Application and Portlet" and complete the remainder of the procedure

7.2.3 How to Consume the Discussion Forum Portlet in Your WebCenter Application

To consume the discussion forum portlet in your WebCenter application, do the following:

1. Start Oracle JDeveloper and create a new WebCenter application or open an existing one.
2. Register your producer according to the instructions in Section 4.3.1.1, "Registering WSRP Portlet Producers". Your URL Endpoint should be of the following form:

    `http://`*host*`:`*port*`/owc_discussions/portlets/wsrp2?WSDL`

    Your producer should then be available for consumption from the Component Palette.
3. Open a JSP or create a new one according to the information in Section 4.2, "Building WebCenter Application-Enabled Pages in Oracle JDeveloper with Oracle ADF". Note that for the discussion forum portlet to work, you must choose **Automatically Expose UI Components in a New Managed Bean** on the Component Binding page of the Create JSF JSP wizard.

For information about the security configuration of the application consuming the discussion forum portlet, refer to Section 7.2.3.1, "Configuring Security for the Portlet and Oracle WebCenter Discussions Application".

4. Drag and drop the `JiveSamplePortlet` onto your page. Refer to Section 4.3.2, "Adding Portlets to a Page" for more information on this procedure.
5. Once your portlet is on the page, you need to contextually link the portlet to the page using the data in Table 7–1 and Table 7–2. Refer to Section 4.5.1, "Linking Portlets to Pages" for the complete linking procedure.

Table 7–1 Page Parameter Names and Values

Parameter ID	Parameter Value
`page_view`	The name of any available view in the portlet, for example, `category_view`.
`containerID`	The corresponding containerID for the specified `page_view` value. For example, you would use the parent category id (1) for category and forum, parent forum id for thread, and parent thread id for messages. Refer to the Jive Forums documentation for more information.

Table 7–2 Page Variable Default Values

Page Variable	Default Value
`JivePortlet1_1_page_view`	`${(bindings.page_view == null || bindings.page_view == '') ? 'category_view' : bindings.page_view}`
`JivePortlet1_1_containerId`	`${(bindings.containerId== null || bindings.containerId== '') ? '1' : bindings.containerId}`

6. Run the page to confirm that the portlet appears as expected.

7.2.3.1 Configuring Security for the Portlet and Oracle WebCenter Discussions Application

You can optionally configure the application consuming the discussion forum portlet to use Oracle ADF security to authenticate users with Java SSO. By configuring security this way, the same authentication token is used to navigate from portlet links to Oracle WebCenter Discussions application and users only need to login once.

To configure Oracle ADF security for your consumer application, perform the steps described in Section 10.2.2, "Configuring Security for Your Application".

8

Integrating Oracle Secure Enterprise Search

In WebCenter applications, it may often be useful to provide users with the capacity to search some data or portion of the data presented to them on a page. To achieve this type of functionality, you can use a Web Service data control to call the searching function of Oracle Secure Enterprise Search.

This chapter explains how to integrate search into your WebCenter applications.

For more information, see the Oracle Secure Enterprise Search documentation that comes with the product.

To build a data control that uses Oracle Secure Enterprise Search, perform the following steps:

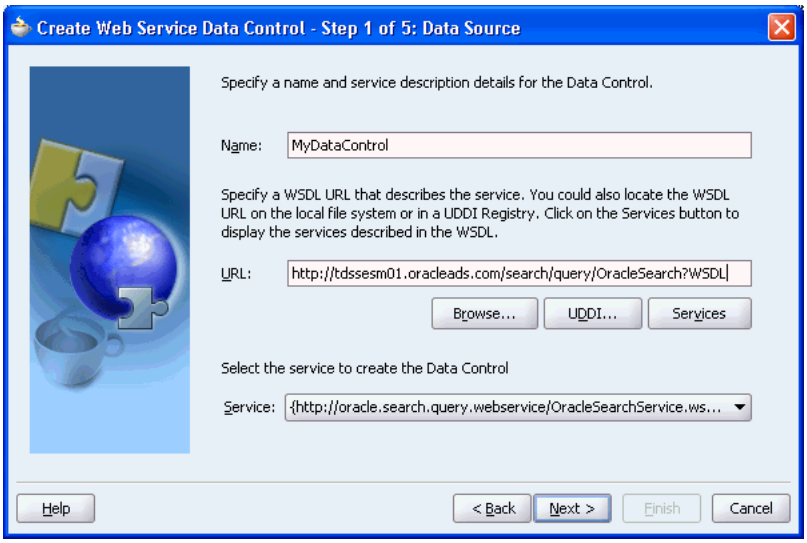
1. If you have not already done so, you must install Oracle Secure Enterprise Search by referring to the installation guide for your platform.
2. If you have not already done so, create an application and a project.
3. Right-click the project where want to incorporate searching and choose **New** from the context menu.
4. Expand **Business Tier** and select **Web Services**. If you do not see **Web Services**, then choose **All Technologies** from the **Filter By** list.
5. In the **Items** list, choose **Web Service Data Control** and click **OK**.
6. If the Welcome page of the Web Service Data Control Wizard appears, then click **Next**.
7. Enter a name for the data control.
8. Enter a Web Services Description Language (WSDL) URL. The WSDL URL for Oracle Secure Enterprise Search is of the form:

   `http://host:port/search/query/OracleSearch?WSDL`

   Note: When Oracle Secure Enterprise Search is installed, the endpoint listed in the WSDL points to a placeholder location. Once the data control is created, you must correct the endpoint URL using the Structure pane. This step will be described later in this procedure.

9. Click **Services**. The wizard page should now look similar to Figure 8–1.

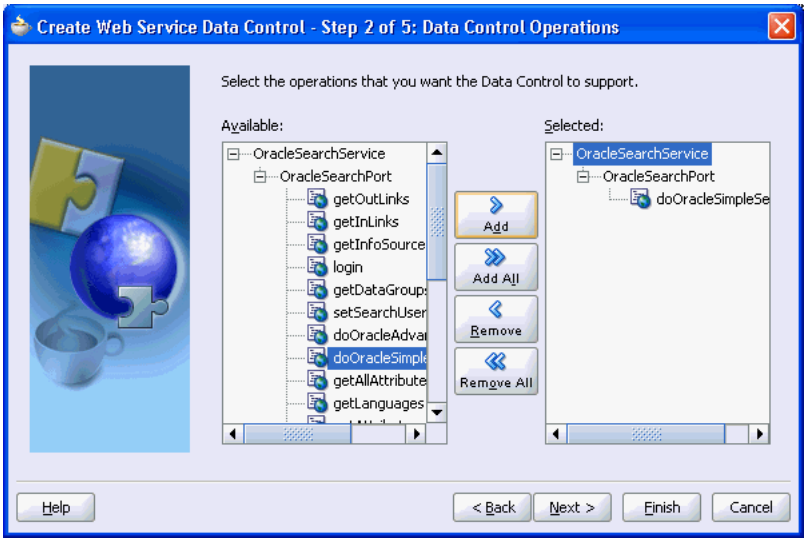
Figure 8–1 Data Source Page of Web Service Data Control Wizard



10. Click **Next**.

11. From the Data Control Operations page, you choose which methods you want to expose by moving them from the **Available** list to the **Selected** list. For the purposes of this example, move **doOracleSimpleSearch** to the **Selected** list. When you are done, the page should look similar to Figure 8–2.

Figure 8–2 Data Control Operations Page of Web Service Data Control Wizard



12. For the purposes of this example, you can now click **Finish**. If you want to review the other pages of the wizard first, then click **Next** until you reach the end of the wizard and then click **Finish**.

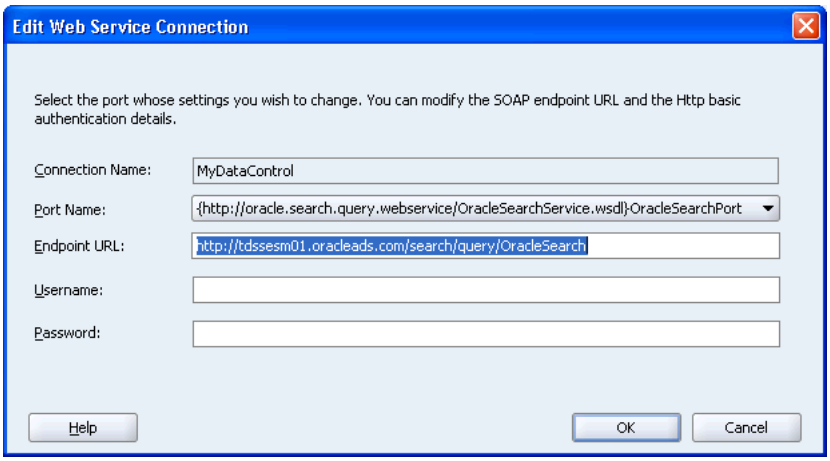
13. In the Application Navigator, expand **Application Sources** and then *project_name*.

14. Click **`DataControls.dcx`** and its structure will appear in the Structure pane below the Application Navigator.

15. Right-click your data control in the Structure pane and choose **Edt Web Service Connection** from the context menu.

16. The Edit Web Service Connection dialog appears as shown in Figure 8–3. As noted earlier, the endpoint listed in the WSDL points to a placeholder location. You must now correct the endpoint URL.

Figure 8–3 Edit Web Service Connection Dialog



17. Click **OK**.

18. Now you can create a page on which to drop your new data control. Right-click your project and choose **New** from the context menu.

19. Under **Web Tier**, choose **JSF**. Under **Items**, choose **JSF JSP**.

20. Click **OK**.

21. Proceed through the JSF JSP wizard accepting the default settings for everything except **File Name**. Choose a meaningful file name.

22. Click **Finish** when ready.

23. Once your JSF JSP appears in the editor, click the **Data Control** tab to bring up the **Data Control Palette**. You should see your data control displayed in the Data Control Palette. If not, then the previous steps must not have completed successfully or perhaps you are in the wrong application.

24. Expand the top level node of your data control and then expand **Return** and **Return**. You can now see all of the data that is returned by the data control.

25. For the purposes of this example, drag and drop **resultElements** onto your page.

26. Select **Tables**, **ADF Read-only Table** from the context menu that appears. The Action Binding Editor appears.

27. For the sake of simplicity, you can hard code the values for each parameter as shown in Table 8–1.

Table 8–1 Parameter Values for Action Binding Editor

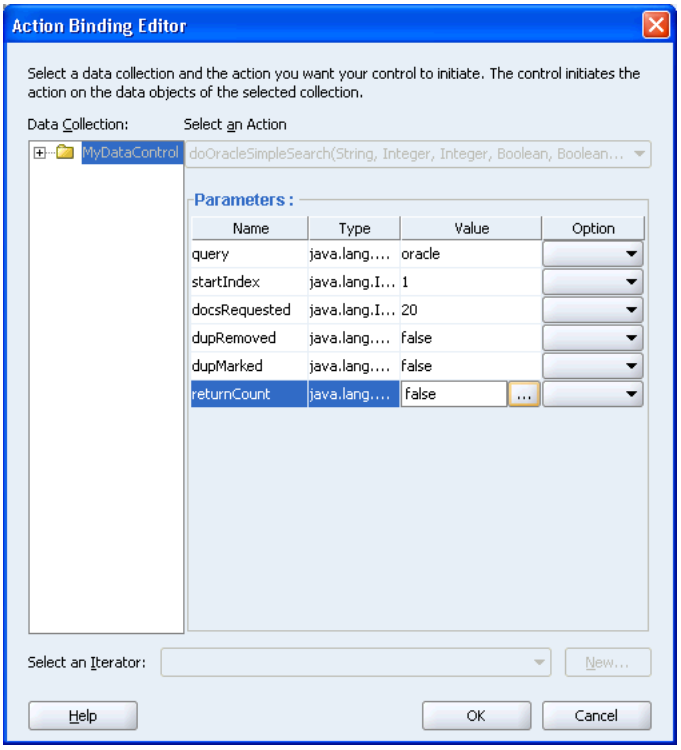
Parameter	Value
query	The string you want to search for, for example, oracle

Table 8–1 (Cont.) Parameter Values for Action Binding Editor

Parameter	Value
startIndex	1
docsRequested	20
dupRemoved	false
dupMarked	false
returnCount	false

28. When you are done, the dialog should look like Figure 8–4. Click **OK**.

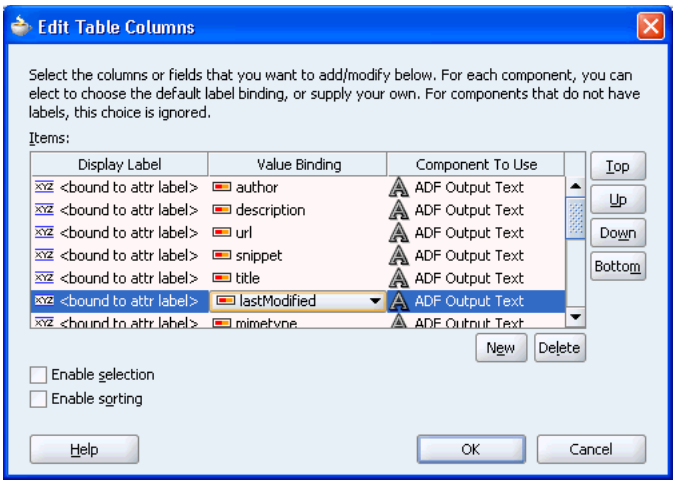
Figure 8–4 Action Binding Editor



29. Click **OK**.

30. The Edit Table Columns dialog appears. Select **lastModified** and click **Delete** (Figure 8–5).

Figure 8–5 Edit Table Columns Dialog



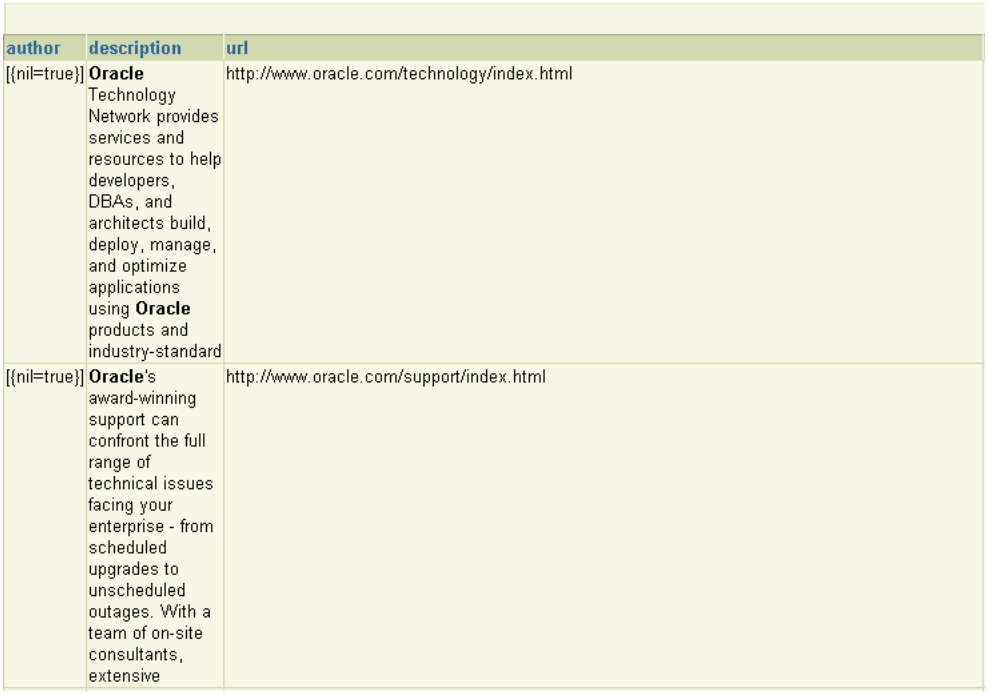
31. Click **OK**.
32. You must now remove the `lastModified` column from the page definition as well. Right-click your page in the Structure pane and choose **Go To Page Definition** from the context menu.
33. Find `<Item Value="lastModified"/>` and delete it.
34. Now you can run your page. Right-click the page in the Application Navigator and choose **Run** from the context menu.
35. Browse through your output and notice how occurrences of the word oracle are surrounded by `<b></b>` tags. Figure 8–6 shows a small sample fragment of this output. Currently, the search hits are not formatted because you have not yet converted the view component that renders this column to `OutputFormatted`.

Figure 8–6 Sample of Searched Output, Hits Unformatted

author	description	url
[{nil=true}]	Oracle Technology Network provides services and resources to help developers, DBAs, and architects build, deploy, manage, and optimize applications using Oracle products and industry-standard	http://www.oracle.com/technology/index.html
[{nil=true}]	Oracle's award-winning support can confront the full range of technical issues facing your enterprise - from scheduled upgrades to unscheduled outages. With a team of on-site consultants, extensive	http://www.oracle.com/support/index.html

36. Back in Oracle JDeveloper, return to your JSP by clicking its tab in the editor.
37. In the Structure pane, find and expand one of the columns that contains oracle, for example, the description column.
38. Right-click **`af:outputText - #{row.snippet}`** for that column and choose **Convert** from the context menu. The Convert dialog appears.
39. Select **OutputFormatted** and click **OK**.
40. If the Confirm Convert dialog appears, then click **OK** again.
41. Run the page again to see the changes. You should now see output similar to that in Figure 8–7. Notice how the `<b></b>` around the word oracle is now interpreted so that search hits appear in bold.

Figure 8–7 Sample of Searched Output, Hits Formatted



author	description	url
[{nil=true}]	**Oracle** Technology Network provides services and resources to help developers, DBAs, and architects build, deploy, manage, and optimize applications using **Oracle** products and industry-standard	http://www.oracle.com/technology/index.html
[{nil=true}]	**Oracle**'s award-winning support can confront the full range of technical issues facing your enterprise - from scheduled upgrades to unscheduled outages. With a team of on-site consultants, extensive	http://www.oracle.com/support/index.html

42. Using the Structure pane, delete the extraneous columns of your table. When you are done, you should only have `row.author`, `row.title`, `row.snippet`, `row.score`, and `row.language` left.
43. Repeat steps 35 through 40 for each of the columns that remains on your page.
44. Run the page again. Note how you see automatic pagination in the upper right corner of the table. The reason for this behavior is that the default number of rows for an Oracle ADF table is 10, but you chose to return 20 results. To enhance the page further, you will provide an estimated count.

    From the Data Control Palette, drag and drop **estimatedHitCount** onto the page. Select **Texts**, **ADF Output Text w/ Label** from the context menu.
45. Another useful feature for this table would be a link to open the document. You can achieve this by dragging a **GoLink** from under **ADF Faces Core** in the **Component Palette** and dropping it right in front of **outputFormatted** of the **title** column. Note that you can perform this step a little more accurately in the Structure pane.
46. In the Property Inspector, change the **Text** attribute to **Open**.
47. Right-click the **af:goLink - Open** in the Structure pane and choose **Properties** from the context menu.
48. Click the **Bind** button next to **Destination**.
49. In the Bind to Data dialog, expand the **row** node under **JSP Objects** and find the **url** row and add it to the **Expression**. Click **OK**.
50. Click **OK**.
51. Click the **Source** tab. You should see something similar to the following in the source for your page:

```
<af:goLink text="Open" destination="#{row.url}"/>
```

52. Run the page again. Confirm that the Open link and the estimatedHitCount are present and working properly.
53. The last feature to add is an input box and a submit button to enable users to perform their own custom searches. Click the **Design** tab.
54. Return to the Data Control Palette and expand the first **Parameters** node under your data control.
55. Drag **query** and drop it just above the left corner of your table. Choose **Texts**, **ADF Input Text w/ Label** from the context menu.
56. Drag **doOracleSimpleSearch** from the Data Control Palette and drop it just below the parameter input box you just created. Choose **Methods**, **ADF Command Button** from the context menu.
57. To avoid getting an error when someone first runs the page, you must set a default value for the query parameter. Go to the page definition, `PageDef.xml`.
58. In the Structure pane, expand **variables** under **executables** and select **doOracleSimpleSearch_query**.
59. In the Property Inspector, select the **DefaultValue** property and enter a default search, for example, **oracle**.
60. Run the page. Enter a different search term in the box and click the button you just created. The results should change.

Note: Because you previously limited the number of results returned, the hit count you added at the bottom of the page is now artificially limited. To fix this problem and get an accurate hit count, find the following line in your `PageDef.xml` file:

```
<NamedData NDName="returnCount" NDValue="false"
   NDType="java.lang.Boolean"/>
```

and change it to:

```
<NamedData NDName="returnCount" NDValue="true"
   NDType="java.lang.Boolean"/>
```

9

Defining and Applying Styles to Core Customizable Components

This chapter provides information about using core customizable component style selectors with Oracle ADF Faces skins and discusses how to register a skin and configure an application to use the skin. Additionally, it lists and describes core customizable component style and icon selectors as well as style-related properties. Finally, it describes how to build a skin selector, from which users can choose a skin at run time, and how to make selections persist across user sessions.

This chapter contains the following sections:

9.1 Introduction to Skins, Style Selectors, and Style-Related Properties

Oracle WebCenter Framework provides two opportunities for applying style information:

- Build a skin using *style selectors* and apply the skin to a WebCenter application. Add defined style selectors to an Oracle ADF Faces skin to generate a standard cascading style sheet (CSS). The act of applying a skin to your application is called *skinning*.
- Use Oracle JDeveloper *style properties* to specify style information through the Property Inspector. Using Oracle JDeveloper style properties overrides the style information from the skin CSS.

Before you begin, it is useful to have some understanding of the technologies at work. This section provides brief overviews of Oracle ADF Faces skins, style selectors, and style properties. It contains the following subsections:

9.1.1 About Oracle ADF Faces Skins

A skin is a style sheet based on the CSS 3.0 syntax that is specified in one place for an entire application. Instead of styling each component, or inserting a style sheet on each page, you can create one skin for the entire application. Every component automatically uses the styles as described by the skin. No design-time code changes are required.

Oracle ADF Faces provides three skins for use in your applications:

- **Oracle**—The default skin. The Oracle skin conforms to Oracle's user interface standards for applications (known as *Oracle Browser Look and Feel*, or *Oracle BLAF*).
- **Minimal**—The Minimal skin provides a modest amount of formatting.
- **Simple**—The Simple skin contains almost no formatting.

All of these skins are included in the ADF Faces 10.1.3 component library, but the source is not exposed to users in this release. You obtain access to these skins by including this and a few other libraries in the application. For information about including relevant libraries, see Section 9.2.1, "How to Make Default Skins Available in an Application".

In addition to the default skins, you can create your own custom skin with your company's preferred look and feel. Custom skins can extend or override the style definitions provided through the Simple skin. Currently, the Simple skin is the only extendable skin. This means that when you apply your own CSS, (that is, your custom skin) all the things you *do not* include in your CSS are inherited from the Simple skin.

Once you create a custom skin, you must register it as a valid skin in the application and then configure the application to use the skin. For more information, see Section 9.2.3, "How to Register a Skin" and Section 9.2.4, "How to Tell an Application to Use a Particular Skin".

When you use a custom skin, applying the Simple skin is implicit. There is no need to explicitly specify its use.

You can create three different color schemes for portlets and core customizable components (`PanelCustomizable` and `ShowDetailFrame`): *Light*, *Medium*, and *Dark*. For information about using these, see Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components".

9.1.2 About Style Selectors

Style sheet rules encompass a *style selector*, which identifies an element, and a set of *style definitions*, which describe the element's appearance. Example 9–1 illustrates a style selector and definition that apply to the `ShowDetailFrame` core customizable component.

Example 9–1 Core Customizable Component Style Selector and Style Definition

```
af|showDetailFrame::main-menu-container
{
   background:#FFFFFF;
   border-left:1px #969664 solid;
   border-right:1px #515151 solid;
   border-top:1px #969664 solid;
   border-bottom:1px #515151 solid;
   width:110px
}
```

Example 9–1 defines styles for the main menu container of a `ShowDetailFrame` component. The style definition specifies menu background color, menu width, and the thickness and color of the menu's surrounding borders.

Oracle ADF Faces skins use three types of style selectors:

- **Standard selectors**—Directly represent an element that can have styles applied to it. For example `af|body` represents the `af:body` component. You can set CSS styles, properties, and icons for this element.
- **Selectors with pseudo elements**—Denote a specific area of a component that can have styles applied. Pseudo elements are easily recognizable by a double colon followed by the portion of the component that the selector represents. For example, `af|showDetailFrame::header-top-border` is the style selector for the top border of the header of a `ShowDetailFrame` component.
- **Selectors that use the alias pseudo class**—Used for a selector that sets styles for more than one component or more than one portion of a component. For example, the `.AFMenuBarItem:alias` selector defines skin properties that are shared by all `af:menuBar` items, such as `af|menuBar::enabled` and `af|menuBar::selected`.

Any selectors that you do not override with your custom skin use the style selector style definition provided in the Simple skin.

Typically, you will not customize the look and feel of every component available in the Oracle ADF Faces component library. By reviewing your application using, for example, the Simple skin, you can determine the components to customize.

For lists, descriptions, and code samples of core customizable component style selectors, see Section 9.3, "Specifying Style Definitions for Portlet and Core Customizable Component Style and Icon Selectors". For information about selectors other than those for core customizable components, see "Selectors for Skinning ADF Faces Components," available on the Oracle Technology Network at:

`http://www.oracle.com/technology/products/jdev/htdocs/partners/addins/exchange/jsf/doc/skin-selectors.html`

9.1.3 About Component Style Properties

You can adjust the look and feel of `ShowDetailFrame` and `PanelCustomizable` components at design time by changing the style-related properties `InlineStyle`, `ContentInlineStyle`, and `StyleClass`. Any style-related property you specify at design time overrides the comparable style specified in the application skin or CSS for that particular instance of the component.

For example, imagine that you have placed two `ShowDetailFrame` components on an application page. You change the style-related properties of the first component. You do not change the style-related properties of the second. The change to style-related properties affects only the first instance.

On a given instance of a component, style specifications entered through the Property Inspector at design time take precedence over comparable style definitions supplied through a skin or CSS at run time.

For more information, see Section 9.4, "Defining Styles Through the Property Inspector".

9.2 Applying Custom Skins to Applications

To use a custom skin in an application, perform the following steps:

- ADF Faces Components and ADF Faces HTML must be included in the application resources.
- The skin or CSS must be added to the relevant project.
- Custom skins must be registered with the application.
- The application must be told which skin to use.

This section describes how to meet these requirements. It includes the following subsections:

- Section 9.2.1, "How to Make Default Skins Available in an Application"
- Section 9.2.2, "How to Add a Custom Skin to an Application"
- Section 9.2.3, "How to Register a Skin"
- Section 9.2.4, "How to Tell an Application to Use a Particular Skin"

9.2.1 How to Make Default Skins Available in an Application

The default Oracle ADF Faces skins, *Oracle*, *Minimal*, and *Simple* are part of the ADF Faces 10.1.3 component library. Any application to which you will apply a skin must include the following libraries:

- ADF Faces Components
- JSF Core

The following libraries, although not required, are useful additions to a WebCenter application:

- ADF Faces HTML
- JSF HTML

This section provides information about how to include a library in an application and how to configure the application to use the library in the desired manner.

To include a library in an application, perform the following steps:

1. In the Applications Navigator, double-click the project that will use a skin.

   This opens the **Project Properties** dialog box.
2. In the left pane of the Project Properties dialog box, select **JSP Tag Libraries**.
3. Click the **Add** button below the list of libraries, and select the following libraries:
   - ADF Faces Components
   - ADF Faces HTML
   - JSF Core
   - JSF HTML
4. Click **OK** to add the selected libraries.
5. In turn, select each library listed under **Distributed libraries**, and select the check box **Execute Tags in JSP Visual Editor**.
6. Click **OK**.

9.2.2 How to Add a Custom Skin to an Application

You have two options for adding a custom skin to an application:

- Create a cascading style sheet (CSS) within Oracle JDeveloper, which will place the CSS properly in a project's source files.
- Add an externally created CSS to the project root.

This section briefly describes both approaches. It includes the following subsections:

- Section 9.2.2.1, "How to Create a CSS in Oracle JDeveloper"
- Section 9.2.2.2, "How to Add a CSS to a Project Root"

9.2.2.1 How to Create a CSS in Oracle JDeveloper

To create a CSS in Oracle JDeveloper, perform the following steps:

1. In the Applications Navigator, right-click a project (or a file within a project) that belongs to the application that will use a skin, and select **New** from the context menu.
2. In the New Gallery under **Categories**, expand the **Web Tier** and select **HTML**.
3. In the New Gallery under **Items**, select **CSS file**.
4. Click **OK**.

   This starts the **Create Cascading Style Sheet** wizard.
5. In the **File Name** field, provide a name for the CSS.
6. In the **Directory Name** field, provide the path where the CSS will be stored.

   Enter a path, accept the default, or click **Browse** and navigate to the location where the CSS should be stored. However you do this, always keep the CSS beneath the project root.
7. Click **OK** to create the CSS.

You can now open the CSS in the Editor pane and define styles for your application, and specifically for core customizable components. For information about core customizable component style selectors, see Section 9.3, "Specifying Style Definitions for Portlet and Core Customizable Component Style and Icon Selectors".

9.2.2.2 How to Add a CSS to a Project Root

When you create a CSS outside the context of Oracle JDeveloper, you must move or copy the CSS to an Oracle JDeveloper project root. This ensures that the CSS is packaged along with other project resource files when the application is deployed

To add an externally created CSS to a project root, perform the following steps:

1. In your computer file system, copy the CSS.

   Be sure to include referenced images files and other dependent resources in the copy or move to maintain link integrity between the CSS and any resources it references.
2. In the file system, navigate to the `Web Content` folder in the relevant project.

   For example:

```
<JDev_Home>
   jdev
      mywork
```

```
<Application Name>
   <Project Name>
      Web Content
```

3. Paste or move the CSS and other resources here, or create an additional folder for storing the skin and its resources at or below this level in the file-system hierarchy.

9.2.3 How to Register a Skin

Registering a skin involves creating a file named `adf-faces-skins.xml` and populating it with a short list of tags that identify the skin's ID, family, location, and the like. This section describes how to create and populate the `adf-faces-skins.xml` file.

To create the `adf-faces-skins.xml` file, perform the following steps:

1. In the Applications Navigator, right-click the `WEB-INF` folder in a project belonging to the application to which you will apply a skin and select **New** from the context menu.
2. Under the **General** node in the New Gallery, select **XML**.

   Note: If the XML option does not display in the New Gallery under the General node, then expand the **Filter By** scope to **All Technologies**.

3. In the right pane, select **XML Document**.
4. Click **OK**.

   This displays the **Create XML File** dialog box.
5. In the **File Name** field, enter the file name `adf-faces-skins.xml`.
6. In the **Directory Name** field, enter the path to the location where the file should be stored, or accept the default.

   The file must be stored in the `WEB-INF` folder.

   Optionally, click the **Browse** button and navigate to the storage location.
7. Click **OK** to create the file.

   The new file opens in the Editor pane.
8. In the **Editor** pane, enter the tags required to register a skin.

   Example 9–2 shows a populated `adf-faces-skins.xml` file.

Example 9–2 A Populated adf-faces-skins.xml File

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<skins xmlns="http://xmlns.oracle.com/adf/view/faces/skin">
   <skin>
      <id>mycompany.desktop</id>
      <family>mycompany</family>
      <render-kit-id>oracle.adf.desktop</render-kit-id>
      <style-sheet-name>skins/mycompany/myCompanySkin.css</style-sheet-name>
   </skin>
</skins>
```

Table 9–1 lists and describes the tags to use in this file.

9. Save your work.

Table 9–1 Tags Used in the adf-faces-skins.xml File

Tag	Description

`skins`	This tag opens and closes the specifications for all skins. Enter the `skins` tag as follows:
	`<skins xmlns="http://xmlns.oracle.com/adf/view/faces/skin">`
	`  <skin>[skin specification]</skin>`
	`</skins>`

`skin`	This tag opens and closes the specifications for a particular skin. Enter one skin tag for each skin you want to register. Register multiple skins when you plan to use different skins under different conditions, for example one for desktop deployment and one for PDA deployment.
`id`	Used for referencing a skin in an Expression Language (EL) expression. For example, to have different skins for different locales, you can create an EL expression that will select the correct skin based on its ID.
`family`	Identifies the family to which a skin belongs. This value is used in the `adf-faces-config.xml` file to identify the skin to the application.
`render-kit-id`	Determines which render-kit to use for the skin. Enter one of the following: ▪ `oracle.adf.desktop`—the skin is used automatically when the application is rendered on a desktop. ▪ `oracle.adf.pda`—the skin is used automatically when the application is rendered on a PDA.
`style-sheet-name`	Defines the path to the custom skin or CSS file, relative to the project root. If your project is not picking up the skin, then it might be because this URL is improperly specified.
`bundle-name`	(Optional) The fully qualified name of the resource bundle used to display text on the components. This is not discussed within the scope of this chapter. For additional information, see Oracle JDeveloper online help.

9.2.4 How to Tell an Application to Use a Particular Skin

To tell an application to use a particular skin, provide a value for the `skin-family` element in the `adf-faces-config.xml` file.

To provide a value for the `skin-family` element, perform the following steps:

1. Open the `adf-faces-config.xml` file, located in the project's `WEB-INF` folder.
2. Replace the default value for `<skin-family>` with the family name of the skin to be used or an Expression Language expression that references a skin bean.

   You find this value between the `<family></family>` tag in the `adf-faces-skins.xml` file, also located in the `WEB-INF` folder. For example, suppose that the `family` tag in `adf-faces-skins.xml` appears as follows:

   ```
   <family>mycompany</family>
   ```

   Then, in the `adf-faces-config.xml` file, you enter the following:

   ```
   <skin-family>mycompany</skin-family>
   ```

Or

```
<skin-family>#{skinBean.currentSkin}</skin-family>
```

Note: This is also where and how you specify any of the default skins provided through the ADF Faces Component library. That is, *Oracle*, *Minimal*, and *Simple*.

Only one `<skin-family>` can be entered in this file. When a value that references a custom skin is used, the Simple skin is implicitly included to style all otherwise unstyled component elements.

The `<skin-family>` value is not case-sensitive.

---

3. Save your work.

9.3 Specifying Style Definitions for Portlet and Core Customizable Component Style and Icon Selectors

The appearance of core customizable components—`PanelCustomizable` and `ShowDetailFrame`—included with Oracle WebCenter Suite can be controlled with their own style selectors. This section includes a series of tables that list, describe, and provide examples for the style selectors associated with core customizable components. Additionally, it describes how to use the `background` property to choose one of three skin-defined looks for customizable components.

Note: In addition to defining styles for `ShowDetailFrame` components, `ShowDetailFrame` style and icon selectors define styles for portlets.

To use style and icon selectors, add them to your CSS/skin, provide the desired style definitions, then apply the skin to your application (Section 9.2, "Applying Custom Skins to Applications").

This section contains the following subsections:

- Section 9.3.1, "Core Customizable Component Property Keys"
- Section 9.3.2, "Global Style Selectors"
- Section 9.3.3, "ShowDetailFrame Style Selectors"
- Section 9.3.4, "PanelCustomizable Style Selectors"
- Section 9.3.5, "Icon Selectors for Core Customizable Components"
- Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components"
- Section 9.3.7, "What You May Need to Know About Oracle ADF Faces Skin Resources"

9.3.1 Core Customizable Component Property Keys

Use property keys to control the display of custom menu items and component action icons. Though you include property keys in your custom skin, they are not represented in the generated CSS that results from the skin.

To explain, skins go through a process that results in a generated CSS. In turn, the generated CSS is consumed by the application. Most customizable component style selectors are represented in the generated CSS. Property keys are the exception. Although they affect the application as much as any other component style selector, they are not represented in the final generated CSS.

Table 9–2 lists and describes property keys relevant to core customizable components.

Table 9–2 Property Keys of Core Customizable Components

Property Key	Description
`showDetailFrame {-ora-additional-actions-position-last:true}`	This property key positions additional actions relative to seeded actions on the component's Actions menu. Set to `false` to position additional actions before seeded actions. By default, additional actions are positioned after seeded actions. The default value is `true`.
`showDetailFrame {-ora-menu-icon-display:false}`	This property key controls the display of icons next to their related commands on a `showDetailFrame` Actions menu. Set to `true` to display icons to the left of each action on the Actions menu. By default, no icons are displayed to the left of individual actions. The default value is `false`. For information about specifying the icons to use when this property key is set to `true`, see Section 9.3.5, "Icon Selectors for Core Customizable Components".
`panelCustomizable {-ora-menu-icon-display:false}`	This property key controls the display of icons next to their related commands on a `PanelCustomizable` Actions menu. Set to `true` to display icons to the left of each action on the Actions menu. By default, no icons are displayed to the left of individual actions. The default value is `false`. For information about specifying the icons to use when this property key is set to `true`, see Section 9.3.5, "Icon Selectors for Core Customizable Components".

9.3.2 Global Style Selectors

Use global style selectors to define styles for multiple components within the application. Table 9–3 lists and describes the global style selectors relevant to Oracle WebCenter Suite components.

The elements styled by global style selectors are illustrated in Figure 9–1:

Figure 9–1 Elements Styled by Global Style Selectors

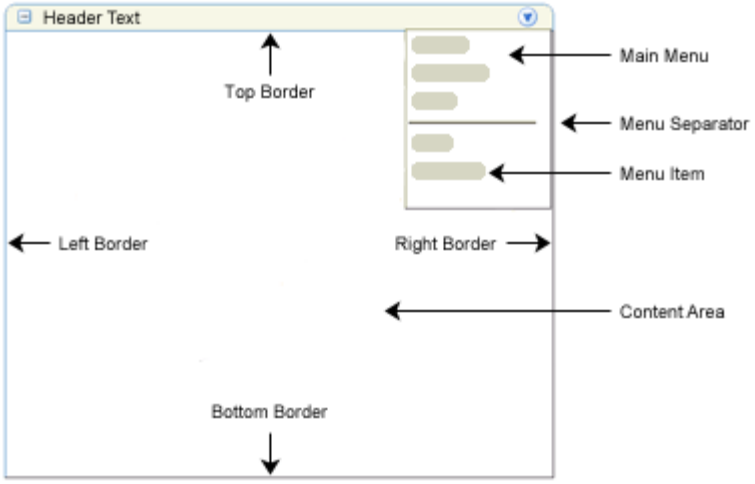


Table 9–3 Global Style Selectors

Style Selector	Description

`.AFCUSTFrameComponentBorderBaseLight:alias`
`.AFCUSTFrameComponentBorderBaseMedium:alias`
`.AFCUSTFrameComponentBorderBaseDark:alias`
 	
Example
`.AFCUSTFrameComponentBorderBaseDark:alias`
`{`
`  border-style:solid;`
`  border-width:1px;`
`  border-color:#979991;`
`}`
	Styles the left, right, and bottom border of the component. It does not include the top border.
	
	Note: See Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components" for an explanation of *light*, *medium*, *dark*.

`.AFCUSTFrameComponentTopBorderBaseLight:alias`
`.AFCUSTFrameComponentTopBorderBaseMedium:alias`
`.AFCUSTFrameComponentTopBorderBaseDark:alias`
 	
Example
`.AFCUSTFrameComponentTopBorderBaseMedium:alias`
`{`
`  border-style:solid;`
`  border-width:1px;`
`  border-color:#979991;`
`}`
	Styles the top border of the component.
	
	Note: See Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components" for an explanation of *light*, *medium*, *dark*.

`.AFCUSTFrameComponentHeaderBase:alias`
 	
Example
`.AFCUSTFrameComponentHeaderBase:alias`
`{`
`  color:#333333;`
`  vertical-align:top;`
`  font-size:11px;`
`  font-family:Tahoma;`
`  font-weight:bold;`
`  height:20px;`
`  border-style:none;`
`}`
	Styles the component header text.
	
	Note: To style the background color of a component header, use the component's `header-top-border` style selector

`.AFCUSTFrameComponentContentBase:alias`
 	
Example
`.AFCUSTFrameComponentContentBase:alias`
`{`
`  color:#333333;`
`  vertical-align:top;`
`  font-size:11px;`
`  font-family:Tahoma;`
`  font-weight:bold;`
`  height:20px;`
`  border-style:none;`
`}`
	Styles the content in the component. The border-style included here is for an internal content border apart from the border surrounding the component.

Table 9–3 (Cont.) Global Style Selectors

Style Selector	Description
`.AFCUSTFrameComponentMenuGroupBase:alias`	Styles the component main menu container.

Example
`.AFCUSTFrameComponentMenuGroupBase:alias`
`{`
`  font-family:Tahoma;`
`  color:#003366;`
`  font-size:11px;`
`}`	

`.AFCUSTFrameComponentMenuItemBase:alias`	Styles menu items.

Example
`.AFCUSTFrameComponentMenuItemBase:alias`
`{`
`  font-family:Tahoma;`
`  color:#003366;`
`  font-size:11px;`
`}`	

`.AFCUSTFrameComponentMenuSeparatorBase:alias`	Styles the menu item separator.

Example
`.AFCUSTFrameComponentMenuSeparatorBase:alias`
`{`
`  padding:0px 2px 0px 2px;`
`}`	

9.3.3 ShowDetailFrame Style Selectors

Use the style selectors listed in Table 9–4 to skin the `ShowDetailFrame` and portlet components.

> **Note:** In WebCenter applications, each portlet is rendered with portlet *chrome* (see Section 14.2, "Portlet Anatomy"). Portlet chrome shares the same chrome rendering mechanism as a `ShowDetailFrame` component. Thus, the style and icon selectors that apply to `ShowDetailFrame` also apply to portlet chrome. In other words, in addition to defining styles for `ShowDetailFrame` components, use `ShowDetailFrame` style and icon selectors to define styles for portlets.

In Table 9–4, some `ShowDetailFrame` style selectors have *light*, *medium*, and *dark* color scheme options. For an explanation of these options, see Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components".

Figure 9–2 illustrates most of the elements styled by `ShowDetailFrame` style selectors. Not depicted are the submenu container, submenu items, hovered-over menu and submenu items, and the actions icon separator, which sets the space around header icons. Note that with `ShowDetailFrame`, the `content` style selector

(`af|showDetailFrame::content-[light,medium,dark]`) styles the bottom, left, and right component borders. Compare this to global style selectors, which provide a selector specifically for the bottom and side borders and a style selector for the top border.

Figure 9–2 Elements Styled by ShowDetailFrame Style Selectors

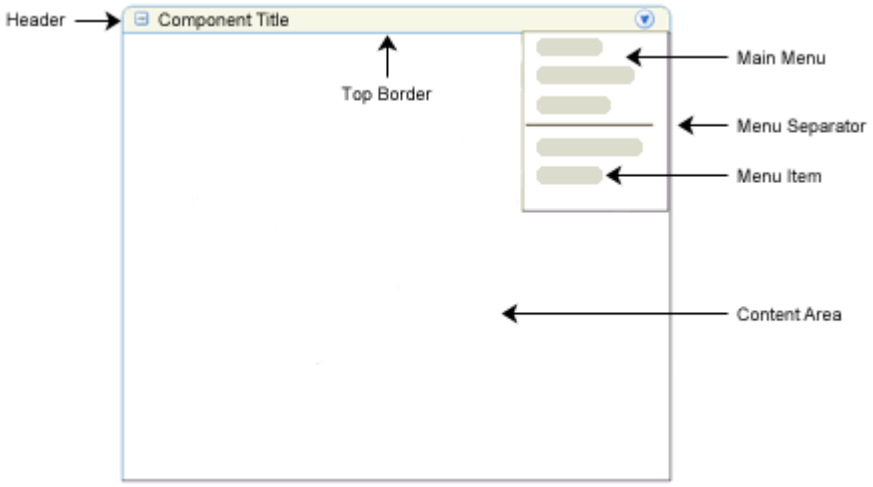


Table 9–4 ShowDetailFrame Style Selectors

Style Selector	Description

`af|showDetailFrame::header-top-border-light`
`af|showDetailFrame::header-top-border-medium`
`af|showDetailFrame::header-top-border-dark`
 	
Example
`af|showDetailFrame::header-top-border-medium`
`{`
`  border-top:1px #6699CC solid;`
`  border-bottom:1px #6699CC solid;`
`  background-color:#F7F7E7;`
`}`
	Specifies the style for the top and bottom border of a component header as well as the header background color.

`af|showDetailFrame::header-light`
`af|showDetailFrame::header-medium`
`af|showDetailFrame::header-dark`
 	
Example
`af|showDetailFrame::header-medium`
`{`
`  color:#336699;`
`  font-family:Arial,Geneva,sans-serif;`
`  font-size:small;`
`  font-weight:bold;`
`  white-space:nowrap;`
`  vertical-align:middle;`
`  width:100%;`
`  position:relative`
`}`
	Specifies the style for text in the component's header. The header is usually a banner of color that contains a title and links to menus and other types of actions.
	
	Define the header background color with the `af|showDetailFrame::header-top-border-[light,medium,dark]` style element. Use icon selectors to specify the icons to use in the component header as well as the shape of the header's left and right sides.

`af|showDetailFrame::content-light`
`af|showDetailFrame::content-medium`
`af|showDetailFrame::content-dark`
 	
Example
`af|showDetailFrame::content-dark`
`{`
`  border-left:1px #6699CC solid;`
`  border-right:1px #515151 solid;`
`  border-bottom:1px #515151 solid;`
`  position:relative;`
`  width:100%`
`}`
	Specifies the style for the component's left, right, and bottom borders.

Table 9–4 (Cont.) ShowDetailFrame Style Selectors

Style Selector	Description
`af|showDetailFrame::main-menu-container`	Specifies the style for the component's main menu container.

Example
`af|showDetailFrame::main-menu-container`
`{`
`  background:#FFFFFF;`
`  border-left:1px #969664 solid;`
`  border-right:1px #515151 solid;`
`  border-top:1px #969664 solid;`
`  border-bottom:1px #515151 solid;`
`  width:110px`
`}`	

`af|showDetailFrame::sub-menu-container`	Specifies the style for the component's submenu container.

Example
`af|showDetailFrame::sub-menu-container`
`{`
`  background:#FFFFFF;`
`  border-left:1px #969664 solid;`
`  border-right:1px #515151 solid;`
`  border-top:1px #969664 solid;`
`  border-bottom:1px #515151 solid;`
`}`	

`A.af|showDetailFrame::menu-item`	Specifies the style for an individual item on the component's main menu.

Example
`A.af|showDetailFrame::menu-item`
`{`
`  font-family:Arial,Geneva,sans-serif;`
`  font-weight:normal;`
`  font-size:small;`
`  color:#000000;`
`  display:block;`
`  cursor:pointer;`
`  text-decoration:none;`
`  white-space:nowrap;`
`  background:#FFFFFF;`
`  padding-top:4px;`
`  padding-bottom:3px;`
`  padding-left:5px;`
`  padding-right:5px;`
`  width:100%`
`}`	

Table 9–4 (Cont.) ShowDetailFrame Style Selectors

Style Selector	Description

`A:hover.af|showDetailFrame::menu-item`
 	
Example
`A:hover.af|showDetailFrame::menu-item`
`{`
`  background-color:#CCCC99`
`}`
	Specifies the style to render when a user pauses the mouse pointer over a component main menu item.

`A.af|showDetailFrame::sub-menu-item`
 	
Example
`A.af|showDetailFrame::sub-menu-item`
`{`
`  font-family:Arial,Geneva,sans-serif;`
`  font-weight:normal;`
`  font-size:small;`
`  color:#000000;`
`  display:block;`
`  cursor:pointer;`
`  text-decoration:none;`
`  white-space:nowrap;`
`  background:#FFFFFF;`
`  padding-top:4px;`
`  padding-bottom:3px;`
`  padding-left:5px;`
`  padding-right:5px;`
`  width:100%`
`}`
	Specifies the style for an individual item on the component's submenu.

`A:hover.af|showDetailFrame::sub-menu-item`
 	
Example
`A:hover.af|showDetailFrame::sub-menu-item`
`{`
`  background-color:#CCCC99`
`}`
	Specifies the style to render when a user pauses the mouse pointer over a component submenu item.

`af|showDetailFrame::actions-image-separator`
 	
Example
`af|showDetailFrame::actions-image-separator`
`{`
`  padding-right:5px;`
`  padding-top:1px;`
`  padding-bottom:1px`
`}`
	Specifies the amount of padding to provide around the component's Actions, Minimize, and Restore icons.
	
	See also Section 9.3.5, "Icon Selectors for Core Customizable Components".

Table 9–4 (Cont.) ShowDetailFrame Style Selectors

Style Selector	Description

`af|showDetailFrame::menu-item-separator`
 	
Example
`af|showDetailFrame::menu-item-separator`
`{`
`  border-top:1px solid #969664;`
`  border-bottom:1px solid #515151;`
`  margin:4px 2px`
`}`
	Specifies the style for the line that separates a command or groups of commands on the component's Actions menu.
	
	In the default case, a separator appears to be a single thick line. This is achieved using `border-top` and `border-bottom` elements to style the separator. A user who creates a custom skin can style the separator differently. For example, a user can create a separator that displays as a rectangular bar with a colorful background.

`A.af|showDetailFrame::title-clickable`
 	
Example
`A.af|showDetailFrame::title-clickable`
`{`
`  font-family:Arial,Geneva,sans-serif;`
`  font-size:small;`
`  font-weight:bold;`
`  text-decoration:none;`
`  display:block;color:#336699;`
`  margin-left:5px`
`}`
	Specifies the style to render for the component's title when the title is a link.

`af|showDetailFrame::no-header-content-light`
`af|showDetailFrame::no-header-content-medium`
`af|showDetailFrame::no-header-content-dark`
 	
Example
`af|showDetailFrame::no-header-content-medium`
`{`
`  border-left:1px #6699CC solid;`
`  border-right:1px #515151 solid;`
`  border-bottom:1px #515151 solid;`
`  border-top:1px #6699CC solid;`
`  position:relative;`
`  width:100%`
`}`
	Specifies the style to render for all four component borders when the component header is turned off.

9.3.4 PanelCustomizable Style Selectors

Use the style selectors listed in Table 9–5 to skin `PanelCustomizable` components. See Section 9.3.5, "Icon Selectors for Core Customizable Components" for icon selectors relevant to the `PanelCustomizable` component.

In Table 9–5, you may note that some of the style selectors have three color-scheme selections: *light*, *medium*, and *dark*. For an explanation of these selections, see Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components".

Figure 9–3 illustrates most of the elements styled by `PanelCustomizable` style selectors. Not depicted are the submenu container, submenu items, and hovered-over menu and submenu items. Note that with `PanelCustomizable`, the `content` style selector (`af|panelCustomizable::content-[light,medium,dark]`) styles the

bottom, left, and right component borders. Compare this to global style selectors, which provide a selector specifically for the bottom and side borders and a style selector for the top border.

Figure 9–3 Elements Styled by PanelCustomizable Style Selectors

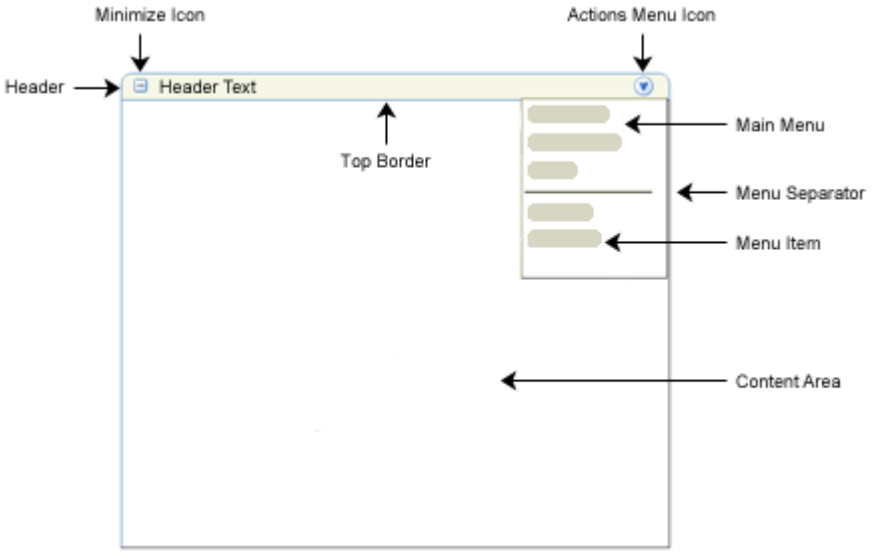


Table 9–5 PanelCustomizable Style Selectors

Style Selector	Description

`af|panelCustomizable::header-top-border-light`
`af|panelCustomizable::header-top-border-medium`
`af|panelCustomizable::header-top-border-dark`
 	
Example
`af|panelCustomizable::header-top-border-medium`
`{`
`  border-top:1px #6699CC solid;`
`  border-bottom:1px #6699CC solid;`
`  background-color:#F7F7E7;`
`  background-image:url(/<path>/setJoin.gif);`
`}`
	Specifies the style for the top and bottom border of a component header as well as the header background color.

`af|panelCustomizable::header-light`
`af|panelCustomizable::header-medium`
`af|panelCustomizable::header-dark`
 	
Example
`af|panelCustomizable::header-medium`
`{`
`  color:#336699;`
`  font-family:Arial,Geneva,sans-serif;`
`  font-size:small;`
`  font-weight:bold;`
`  white-space:nowrap;`
`  vertical-align:middle;`
`  width:100%;`
`  position:relative`
`}`
	Specifies the style for text in the component's header. The header is usually a banner of color that contains a title and links to menus and other types of actions.
	
	Define the header background color with the `af|panelCustomizable::header-top-border-[light,medium,dark]` style element. Use icon selectors to specify the icons to use in the component header as well as the shape of the header's left and right sides.

`af|panelCustomizable::content-light`
`af|panelCustomizable::content-medium`
`af|panelCustomizable::content-dark`
 	
Example
`af|panelCustomizable::content-light`
`{`
`  border-left:1px #6699CC solid;`
`  border-right:1px #515151 solid;`
`  border-bottom:1px #515151 solid;`
`  position:relative;`
`  width:100%`
`}`
	Specifies the style for the component's left, right, and bottom borders.

Table 9–5 (Cont.) PanelCustomizable Style Selectors

Style Selector	Description

`af|panelCustomizable::main-menu-container`
 	
Example
`af|panelCustomizable::main-menu-container`
`{`
`  background:#FFFFFF;`
`  border-left:1px #969664 solid;`
`  border-right:1px #515151 solid;`
`  border-top:1px #969664 solid;`
`  border-bottom:1px #515151 solid;`
`  width:110px`
`}`
	Specifies the style for the component's main menu container.

`af|panelCustomizable::sub-menu-container`
 	
Example
`af|panelCustomizable::sub-menu-container`
`{`
`  background:#FFFFFF;`
`  border-left:1px #969664 solid;`
`  border-right:1px #515151 solid;`
`  border-top:1px #969664 solid;`
`  border-bottom:1px #515151 solid;`
`}`
	Specifies the style for the component's submenu container.

`A.af|panelCustomizable::menu-item`
 	
Example
`A.af|panelCustomizable::menu-item`
`{`
`  font-family:Arial,Geneva,sans-serif;`
`  font-weight:normal;`
`  font-size:small;`
`  color:#000000;`
`  display:block;`
`  cursor:pointer;`
`  text-decoration:none;`
`  white-space:nowrap;`
`  background:#FFFFFF;`
`  padding-top:4px;`
`  padding-bottom:3px;`
`  padding-left:5px;`
`  padding-right:5px;`
`  width:100%`
`}`
	Specifies the style for an individual item on the component's main menu.

Table 9–5 (Cont.) PanelCustomizable Style Selectors

Style Selector	Description

`A:hover.af|panelCustomizable::menu-item`
 	
Example
`A:hover.af|panelCustomizable::menu-item`
`{`
`  background-color:#CCCC99`
`}`
	Specifies the style to render when a user pauses the mouse pointer over a main menu item.

`A.af|panelCustomizable::sub-menu-item`
 	
Example
`A.af|panelCustomizable::sub-menu-item`
`{`
`  font-family:Arial,Geneva,sans-serif;`
`  font-weight:normal;`
`  font-size:small;`
`  color:#000000;`
`  display:block;`
`  cursor:pointer;`
`  text-decoration:none;`
`  white-space:nowrap;`
`  background:#FFFFFF;`
`  padding-top:4px;`
`  padding-bottom:3px;`
`  padding-left:5px;`
`  padding-right:5px;`
`  width:100%`
`}`
	Specifies the style for an individual item on a submenu.

`A:hover.af|panelCustomizable::sub-menu-item`
 	
Example
`A:hover.af|panelCustomizable::sub-menu-item`
`{`
`  background-color:#CCCC99`
`}`
	Specifies the style to render when a user pauses the mouse pointer over a submenu item.

Table 9–5 (Cont.) PanelCustomizable Style Selectors

Style Selector	Description

`af|panelCustomizable::actions-image-separator`
 	
Example
`af|panelCustomizable::actions-image-separator`
`{`
`  padding-right:5px;`
`  padding-top:1px;`
`  padding-bottom:1px`
`}`
	Specifies the amount of padding to provide around the component's Actions, Minimize, and Restore icons.
	
	See also Section 9.3.5, "Icon Selectors for Core Customizable Components".

`af|panelCustomizable::menu-item-separator`
 	
Example
`af|panelCustomizable::menu-item-separator`
`{`
`  border-top:1px solid #969664;`
`  border-bottom:1px solid #515151;`
`  margin:4px 2px`
`}`
	Specifies the style for the line that separates a command or groups of commands on the component's Actions menu.

`af|panelCustomizable::no-header-content`
 	
Example
`af|panelCustomizable::no-header-content`
`{`
`  border-left:1px #6699CC solid;`
`  border-right:1px #515151 solid;`
`  border-bottom:1px #515151 solid;`
`  border-top:1px #6699CC solid;`
`  position:relative;`
`  width:100%`
`}`
	Specifies the style to render for all four component borders when the component header is turned off.

9.3.5 Icon Selectors for Core Customizable Components

The selectors described in Table 9–6 apply to the icons used with core customizable components. Icons are displayed or are not displayed depending on whether the component's `ora-menu-icon-display` property key is set to `true` or `false`. Property keys are described in Section 9.3.1, "Core Customizable Component Property Keys".

Each icon selector has a *light*, *medium*, and *dark* scheme. For an explanation of these color schemes, see Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components".

For easy, error-free portability, store all application icons under the WebCenter application's root folder.

Note: In WebCenter applications, each portlet is rendered with portlet *chrome* (see Section 14.2, "Portlet Anatomy"). Portlet chrome shares the same portlet chrome rendering mechanism as a `ShowDetailFrame` component. This being the case, the style and icon selectors that apply to `ShowDetailFrame` also apply to portlet chrome. In other words, in addition to defining styles for `ShowDetailFrame` components, use `ShowDetailFrame` style and icon selectors to define styles for portlets.

Table 9–6 Icon Selectors for Core Customizable Components

Selector	Description

`showDetailFrame`
`showDetailFrame::light-ActionsIcon:alias`
`showDetailFrame::medium-ActionsIcon:alias`
`showDetailFrame::dark-ActionsIcon:alias`
 	
Example
`showDetailFrame::light-ActionsIcon:alias`
`{`
`  content:url(/css/images/action.gif)`
`}`
	This icon represents the Actions menu. The Actions menu lists the actions a user can perform on the component.
	
	In a WebCenter application, the Actions icon is rendered on the right corner of the component header.

`panelCustomizable`
`panelCustomizable::light-ActionsIcon:alias`
`panelCustomizable::medium-ActionsIcon:alias`
`panelCustomizable::dark-ActionsIcon:alias`
 	
Example
`panelCustomizable::dark-ActionsIcon:alias`
`{`
`  content:url(/css/images/action.gif)`
`}`	

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-MinimizeIcon:alias
showDetailFrame::medium-MinimizeIcon:alias
showDetailFrame::dark-MinimizeIcon:alias
Example
showDetailFrame::light-MinimizeIcon:alias
{
content:url(/css/images/minimize.gif)
}
panelCustomizable
panelCustomizable::light-MinimizeIcon:alias
panelCustomizable::medium-MinimizeIcon:alias
panelCustomizable::dark-MinimizeIcon:alias
Example
panelCustomizable::medium-MinimizeIcon:alias
{
content:url(/css/images/minimize.gif)
}
	This icon represents the Minimize option. Minimize collapses the view of the component like a window shade.
	In a WebCenter application, the Minimize icon is rendered on the left side of the component header.
	See also, showDetailFrame::light-ExpandIcon:alias.

showDetailFrame::light-MaximizeIcon:alias
showDetailFrame::medium-MaximizeIcon:alias
showDetailFrame::dark-MaximizeIcon:alias
Example
showDetailFrame::medium-MaximizeIcon:alias
{
content:url(/css/images/maximize.gif)
}
	This icon represents the Maximize option, which expands the ShowDetailFrame component to the dimensions of the PanelCustomizable component that contains it. Where multiple ShowDetailFrame components display in the same container, these are displaced while the maximized ShowDetailFrame remains maximized.
	In a WebCenter application, the Maximize icon is displayed to the left of the Maximize command on the component's Actions menu.

panelCustomizable::light-MaximizeIcon:alias
panelCustomizable::medium-MaximizeIcon:alias
panelCustomizable::dark-MaximizeIcon:alias
Example
panelCustomizable::dark-MaximizeIcon:alias
{
content:url(/css/images/maximize.gif)
}
	This icon represents the Maximize option, which expands component display to the dimensions of the container. Where multiple components display in the same container, these are displaced by the maximized component.
	In a WebCenter application, the Maximize icon is displayed to the left of the Maximize command on the component's Actions menu.

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-RestoreIcon:alias
showDetailFrame::medium-RestoreIcon:alias
showDetailFrame::dark-RestoreIcon:alias
Example
showDetailFrame::medium-RestoreIcon:alias
{
content:url(/css/images/restore.gif)
}
panelCustomizable
panelCustomizable::light-RestoreIcon:alias
panelCustomizable::medium-RestoreIcon:alias
panelCustomizable::dark-RestoreIcon:alias
Example
panelCustomizable::dark-RestoreIcon:alias
{
content:url(/css/images/restore.gif)
}
	This icon represents the Restore option, which restores maximized views to their default display modes.
	In a WebCenter application, the Restore icon is rendered to the left of the Restore command on the component's Actions menu.

showDetailFrame
showDetailFrame::light-ExpandIcon:alias
showDetailFrame::medium-ExpandIcon:alias
showDetailFrame::dark-ExpandIcon:alias
Example
showDetailFrame::medium-ExpandIcon:alias
{
content:url(/css/images/expand.gif)
}
panelCustomizable
panelCustomizable::light-ExpandIcon:alias
panelCustomizable::medium-ExpandIcon:alias
panelCustomizable::dark-ExpandIcon:alias
Example
panelCustomizable::dark-ExpandIcon:alias
{
content:url(/css/images/expand.gif)
}
	The Expand icon represents the action that expands a component that has been minimized. The Expand icon toggles with the Minimize icon. That is, when the component is minimized, the Expand icon is displayed; when the component is expanded, the Minimize icon is displayed.
	In a WebCenter application, the Expand icon is displayed on the left side of the component header.

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-MoveIcon:alias
showDetailFrame::medium-MoveIcon:alias
showDetailFrame::dark-MoveIcon:alias
Example
showDetailFrame::light-MoveIcon:alias
{
content:url(/css/images/move.gif)
}
panelCustomizable
panelCustomizable::light-MoveIcon:alias
panelCustomizable::medium-MoveIcon:alias
panelCustomizable::dark-MoveIcon:alias
Example
af|panelCustomizable::dark-MoveIcon:alias
{
content:url(/css/images/move.gif)
}
	This icon represents the Move option, which enables rearrangement of a component's location in relation to the other components on the page.
	In a WebCenter application, the Move icon is displayed to the left of the Move command on the component's Actions menu.

showDetailFrame
showDetailFrame::light-MoveLeftIcon:alias
showDetailFrame::medium-MoveLeftIcon:alias
showDetailFrame::dark-MoveLeftIcon:alias
Example
showDetailFrame::dark-MoveLeftIcon:alias
{
content:url(/css/images/left.gif)
}
panelCustomizable
panelCustomizable::light-MoveLeftIcon:alias
panelCustomizable::medium-MoveLeftIcon:alias
panelCustomizable::dark-MoveLeftIcon:alias
Example
panelCustomizable::light-MoveLeftIcon:alias
{
content:url(/css/images/left.gif)
}
	This icon represents the Move Left option on the component submenu. Move Left rearranges the component horizontally, one position closer to the left boundary of the page. For example, imagine three horizontally arranged components. You select Move Left on the rightmost component. It becomes the middle component.
	In a WebCenter application, the Move Left icon is displayed to the left of the Move Left submenu item on the component's Actions menu.

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-MoveRightIcon:alias
showDetailFrame::medium-MoveRightIcon:alias
showDetailFrame::dark-MoveRightIcon:alias
Example
showDetailFrame::medium-MoveRightIcon:alias
{
content:url(/css/images/right.gif)
}
panelCustomizable
panelCustomizable::light-MoveRightIcon:alias
panelCustomizable::medium-MoveRightIcon:alias
panelCustomizable::dark-MoveRightIcon:alias
Example
panelCustomizable::medium-MoveRightIcon:alias
{
content:url(/css/images/right.gif)
}
	This icon represents the Move Right option on the component submenu. Move Right rearranges the component horizontally, one position closer to the right boundary of the page. For example, imagine three horizontally arranged components. You select Move Right on the leftmost component. It becomes the middle component.
	In a WebCenter application, the Move Right icon is displayed to the left of the Move Right submenu item on the component's Actions menu.

showDetailFrame
showDetailFrame::light-MoveUpIcon:alias
showDetailFrame::medium-MoveUpIcon:alias
showDetailFrame::dark-MoveUpIcon:alias
Example
showDetailFrame::light-MoveUpIcon:alias
{
content:url(/css/images/up.gif)
}
panelCustomizable
panelCustomizable::light-MoveUpIcon:alias
panelCustomizable::medium-MoveUpIcon:alias
panelCustomizable::dark-MoveUpIcon:alias
Example
panelCustomizable::medium-MoveUpIcon:alias
{
content:url(/css/images/up.gif)
}
	This icon represents the Move Up option on the component submenu. Move Up rearranges the component vertically in relation to the other components on the page. For example, imagine three vertically arranged components. You select Move Up on the middle component. It becomes the topmost component.
	In a WebCenter application, the Move Up icon is displayed to the left of the Move Up submenu item on the component's Actions menu.

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-MoveDownIcon:alias
showDetailFrame::medium-MoveDownIcon:alias
showDetailFrame::dark-MoveDownIcon:alias
 	
Example
showDetailFrame::dark-MoveDownIcon:alias
{
content:url(/css/images/down.gif)
}
 	
panelCustomizable
panelCustomizable::light-MoveDownIcon:alias
panelCustomizable::medium-MoveDownIcon:alias
panelCustomizable::dark-MoveDownIcon:alias
 	
Example
panelCustomizable::medium-MoveDownIcon:alias
{
content:url(/css/images/down.gif)
}
	This icon represents the Move Down option on the component submenu. Move Down rearranges the component vertically in relation to the other components on the page. For example, imagine three vertically arranged components. You select Move Down on the middle component. It becomes the bottommost component.
	
	In a WebCenter application, the Move Down icon is displayed to the left of the Move Down submenu item on the component's Actions menu.

showDetailFrame
showDetailFrame::light-HeaderLeftIcon:alias
showDetailFrame::medium-HeaderLeftIcon:alias
showDetailFrame::dark-HeaderLeftIcon:alias
 	
Example
showDetailFrame::light-HeaderLeftIcon:alias
{
content:url(/css/images/headerleft.gif)
}
 	
panelCustomizable
panelCustomizable::light-HeaderLeftIcon:alias
panelCustomizable::medium-HeaderLeftIcon:alias
panelCustomizable::dark-HeaderLeftIcon:alias
 	
Example
panelCustomizable::medium-HeaderLeftIcon:alias
{
content:url(/css/images/headerleft.gif)
}
	This icon provides an image for the top-left corner of the component header.

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-HeaderRightIcon:alias
showDetailFrame::medium-HeaderRightIcon:alias
showDetailFrame::dark-HeaderRightIcon:alias
 	
Example
showDetailFrame::light-HeaderRightIcon:alias
{
content:url(/css/images/headerright.gif)
}
	This icon provides the image for the top-right corner of the component header.

panelCustomizable
panelCustomizable::light-HeaderRightIcon:alias
panelCustomizable::medium-HeaderRightIcon:alias
panelCustomizable::dark-HeaderRightIcon:alias
 	
Example
panelCustomizable::medium-HeaderRightIcon:alias
{
content:url(/css/images/headerright.gif)
}	

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-ToolbarLeftIcon:alias
showDetailFrame::medium-ToolbarLeftIcon:alias
showDetailFrame::dark-ToolbarLeftIcon:alias
 	
Example
showDetailFrame::dark-ToolbarLeftIcon:alias
{
content:url(/css/images/toolbarleft.gif)
}
	This icon provides the left portion of the component's FadeIn-FadeOut toolbar.
	The FadeIn-FadeOut toolbar comes into play when the `adfp:portlet` tag attribute `isSeededInteractionAvailable` is set to `true` and `displayHeader` is set to `false`.
	The toolbar contains the Actions menu that would otherwise be displayed on the header. To invoke the toolbar, users move their mouse over the component content area.
	If the page design is very simple, then the FadeIn-FadeOut toolbar may not display, even when `displayHeader` is set to `false` and `isSeededInteractionAvailable` is set to `true`.

panelCustomizable
panelCustomizable::light-ToolbarLeftIcon:alias
panelCustomizable::medium-ToolbarLeftIcon:alias
panelCustomizable::dark-ToolbarLeftIcon:alias
 	
Example
panelCustomizable::light-ToolbarLeftIcon:alias
{
content:url(/css/images/toolbarleft.gif)
}	

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

`showDetailFrame`
`showDetailFrame::light-ToolbarRightIcon:alias`
`showDetailFrame::medium-ToolbarRightIcon:alias`
`showDetailFrame::dark-ToolbarRightIcon:alias`
 	
Example
`showDetailFrame::medium-ToolbarRightIcon:alias`
`{`
`  content:url(/css/images/toolbarright.gif)`
`}`
	This icon provides the right portion of the component's FadeIn-FadeOut toolbar.
	
	See the description for `showDetailFrame::light-ToolbarLeftIcon:alias` for an explanation of the FadeIn-FadeOut toolbar.

`panelCustomizable`
`panelCustomizable::light-ToolbarRightIcon:alias`
`panelCustomizable::medium-ToolbarRightIcon:alias`
`panelCustomizable::dark-ToolbarRightIcon:alias`
 	
Example
`panelCustomizable::dark-ToolbarRightIcon:alias`
`{`
`  content:url(/css/images/toolbarright.gif)`
`}`	

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-ToolbarCenterIcon:alias
showDetailFrame::medium-ToolbarCenterIcon:alias
showDetailFrame::dark-ToolbarCenterIcon:alias
 	
Example
showDetailFrame::medium-ToolbarCenterIcon:alias
{
content:url(/css/images/toolbarcenter.gif
)
}
	This icon provides the center portion of the component's FadeIn-FadeOut toolbar.
	
	See the description for showDetailFrame::light-ToolbarLeftIcon:alias for an explanation of the FadeIn-FadeOut toolbar.

panelCustomizable
panelCustomizable::light-ToolbarCenterIcon:alias
panelCustomizable::medium-ToolbarCenterIcon:alias
panelCustomizable::dark-ToolbarCenterIcon:alias
 	
Example
panelCustomizable::dark-ToolbarCenterIcon:alias
{
content:url(/css/images/toolbarcenter.gif
)
}	

Table 9–6 (Cont.) Icon Selectors for Core Customizable Components

Selector	Description

showDetailFrame
showDetailFrame::light-EditIcon:alias
showDetailFrame::medium-EditIcon:alias
showDetailFrame::dark-EditIcon:alias
 	
Example
showDetailFrame::dark-EditIcon:alias
{
content:url(/css/images/edit.gif)
}
 	
panelCustomizable
panelCustomizable::light-EditIcon:alias
panelCustomizable::medium-EditIcon:alias
panelCustomizable::dark-EditIcon:alias
 	
Example
panelCustomizable::light-EditIcon:alias
{
content:url(/css/images/edit.gif)
}
	This icon represents the Edit option on the component menu. In a WebCenter application, the Edit icon is displayed to the left of the Edit menu item on the component's Actions menu.

showDetailFrame
showDetailFrame::light-HelpIcon:alias
showDetailFrame::medium-HelpIcon:alias
showDetailFrame::dark-HelpIcon:alias
 	
Example
showDetailFrame::dark-HelpIcon:alias
{
content:url(/css/images/help.gif)
}
 	
panelCustomizable
panelCustomizable::light-HelpIcon:alias
panelCustomizable::medium-HelpIcon:alias
panelCustomizable::dark-HelpIcon:alias
 	
Example
panelCustomizable::light-HelpIcon:alias
{
content:url(/css/images/help.gif)
}
	This icon represents the Help option on the component menu. In a WebCenter application, the Help icon is displayed to the left of the Help menu item on the component's Actions menu.

9.3.6 Applying Color Schemes to Portlets and Core Customizable Components

You can define three distinct looks in your CSS/skin and specify which one to use through the `background` property in the Oracle JDeveloper Property Inspector.

> **Note:** In addition to defining styles for `ShowDetailFrame` components, use `ShowDetailFrame` style and icon selectors to define styles for portlets.

Portlets, `PanelCustomizable`, and `ShowDetailFrame` components include a `background` property for which you can select a value of *light*, *medium*, or *dark*. The default skins, *Oracle*, *Minimal*, and *Simple*, include three versions of style selectors: a light version, a medium version, and a dark version. Depending on which value is specified for a component or portlet instance's `background` property, the skin will apply the relevant style or icon selector version.

The same principal applies to custom skins. For example, imagine that you have created a custom skin and defined a *light*, *medium*, and *dark* version of the `af|showDetailFrame::header-top-border` style selector:

```
af|showDetailFrame::header-top-border-light
af|showDetailFrame::header-top-border-medium
af|showDetailFrame::header-top-border-dark
```

You have three portlets on your application page. You select *light* as the value for the first portlet's `background` property; *medium* for the second portlet's `background` property; *dark* for the third portlet's `background` property. At run time, the skin will apply style definitions in the following manner:

- `af|showDetailFrame::header-top-border-light` to the first portlet
- `af|showDetailFrame::header-top-border-medium` to the second portlet
- `af|showDetailFrame::header-top-border-dark` to the third portlet

The terms *light*, *medium*, and *dark* are not necessarily meaningful in terms of lightness or darkness. You can define each version (light, medium, and dark) using unique colors. For example, for a given style selector, its light version can be defined using greens, its medium version using reds, and its dark version using yellows. The terms *light*, *medium*, and *dark* merely differentiate.

Set up a CSS/skin with three style options for the selectors that support them, then set the component's `background` property to specify which option to apply on a given instance.

9.3.7 What You May Need to Know About Oracle ADF Faces Skin Resources

The Oracle Technology Network offers a useful source of information about the selectors you can use for skinning Oracle ADF Faces components. The following article lists and describes many of the style selectors provided through Oracle ADF Faces:

`http://www.oracle.com/technology/products/jdev/htdocs/partners/addins/exchange/jsf/doc/skin-selectors.html`

> **Note:** Core Customizable Components `PanelCustomizable` and `ShowDetailFrame` are not included in this article because they are new to this release and specific to the WebCenter Suite extension to Oracle JDeveloper.

The article, "Developing and Using Oracle ADF Faces Skins," includes a link to a WAR file that contains the `Oracle`, `Minimal`, and `Simple` skins and steps you through the process of applying them to a sample page. It includes information about creating an application from the WAR file to give you the full experience of its examples. You will find this article at the following URL:

`http://www.oracle.com/technology/products/jdev/101/howtos/adfskins`

Note: To download the WAR file using Internet Explorer, go to:

`http://www.oracle.com/technology/products/jdev/101/howtos/adfskins/index.html`

Right-click the `adffaces-skin.war` link, and select **Save Target As** from the context menu. Change the file extension from `xml` to `war`. Internet Explorer otherwise has difficulty downloading the WAR file.

9.4 Defining Styles Through the Property Inspector

You can override the look and feel of portlets and core customizable components by changing the style-related properties `InlineStyle`, `ContentInlineStyle`, and `StyleClass`. Any style you specify through the Oracle JDeveloper Property Inspector overrides the comparable style specified in an application skin/style sheet (CSS). Additionally, properties set on a portlet or a component instance affect only that instance of the portlet or component. Other portlet or component instances in the application are not affected.

Note: The `background` property is also useful in adjusting the look and feel of portlets and core customizable components. Unlike `InlineStyle`, `ContentInlineStyle`, and `StyleClass` properties, the `background` property works in conjunction with skins or CSSs. For more information about the `background` property, see Section 9.3.6, "Applying Color Schemes to Portlets and Core Customizable Components".

This section describes style-related properties and their uses. It contains the following subsections:

- Section 9.4.1, "Understanding Style-Related Properties"
- Section 9.4.2, "Changing Style-Related Properties"
- Section 9.4.3, "Attributes of the ContentInlineStyle and InlineStyle Properties"

9.4.1 Understanding Style-Related Properties

Before you use style-related properties, you should have a good understanding of them. This section compares style-related properties and provides examples of their use. It includes the following subsections:

- Section 9.4.1.1, "Understanding ContentInlineStyle and InlineStyle Properties"
- Section 9.4.1.2, "What You Should Know About EL and the InlineStyle Property"
- Section 9.4.1.3, "Understanding the StyleClass Property"

9.4.1.1 Understanding ContentInlineStyle and InlineStyle Properties

The style properties `InlineStyle` and `ContentInlineStyle` are alike in the types of attributes they support. They differ in their range of influence. While `InlineStyle` provides style information for the *entire component*, `ContentInlineStyle` provides style information only for *component content*.

For example, when you set the `background-color` attribute using `InlineStyle` property, the component and its content are set in the color specified. However, if you set a different background color for the `ContentInlineStyle` property, then the *component header* is styled by the `InlineStyle` color, while *component content* is styled by the `ContentInlineStyle` color.

This also means that, on component content, the value specified for `ContentInlineStyle` takes precedence over the value specified for `InlineStyle`. Additionally, `ContentInlineStyle` on a component instance takes precedence over both `InlineStyle` and the `ContentInlineStyle` values of a parent component (such as a portlet nested in a `PanelCustomizable` component).

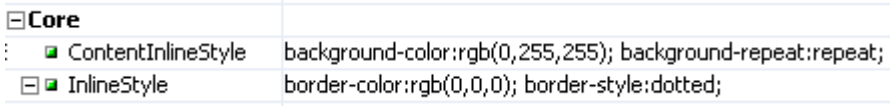
Unlike `InlineStyle`, you can use `ContentInlineStyle` to turn portlet borders off, as shown in Example 9–3.

Example 9–3 Turning Portlet Borders Off

```
border-style:none; /*in the Property Inspector*/
contentInlineStyle="border-style:none;" /*in the adfp:portlet tag*/
```

In the Property Inspector, `InlineStyle` provides a list of style attributes for which you can provide values. `ContentInlineStyle` does not. Both `ContentInlineStyle` and `InlineStyle` support manual entry of style attributes. This includes attributes outside the scope of those listed in the Property Inspector, provided those attributes are in compliance with, at least, CSS 2.0. Figure 9–4 illustrates the inclusion of style-related properties in the Property Inspector.

Figure 9–4 Defining Styles for ContentInlineStyle and InlineStyle in the Property Inspector



Example 9–4 shows the format such attributes take in the `adfp:portlet` tag.

Example 9–4 Specifying Styles in the adfp:portlet Tag

```
<adfp:portlet value="#{bindings.RichTextPortlet1_1}"
  portletType="/oracle/adf/portlet/RTPWSRP2_1157051886703/ap/E0default_65adffff_
       010d_1000_800c_8d9049ad9bf7"
    inlineStyle="border-color:rgb(0,0,0); border-style:dotted;"
    contentInlineStyle="background-color:rgb(0,255,255);
       background-repeat:repeat;"/>
```

9.4.1.2 What You Should Know About EL and the InlineStyle Property

You can also use Expression Language (EL) expressions for `InlineStyle` to conditionally set inline style attributes. Example 9–5 illustrates one approach.

Example 9–5 Using EL to Conditionally Set an Inline Style Attribute

```
<af:outputText value="#{row.assignedDate eq
```

```
null?res['srsearch.highlightUnassigned']:row.assignedDate}"
      binding="#{backing_SRSearch.outputText6}"
      id="outputText6"
      inlineStyle="#{row.assignedDate eq null?'color:red;':''}">
```

9.4.1.3 Understanding the StyleClass Property

The `StyleClass` property provides a means of overriding styles specified in a skin or CSS and a way to apply "unrelated" style classes to a component. For example, using the `StyleClass` property you can apply a `PanelCustomizable` style class to a `ShowDetailFrame` component.

Using the `StyleClass` property to override skin styles is the one way to use style properties to affect look and feel while still complying with accessibility guidelines. `InlineStyle` and `ContentInlineStyle` are not viewed as compliant.

Style classes are expressed differently than *style selectors*. Pipes (|) and double-colons (::) in a style selector are replaced by underscores when the selector is expressed as a style class in the Property Inspector or in a source code tag.

In a skin, a style selector can be expressed as shown in Example 9–6.

Example 9–6 Style Selector Expressed in a Skin

```
af|panelCustomizable::no-header-content
```

The same style selector is expressed as a style class in a component tag as shown in Example 9–7.

Example 9–7 Style Selector Expressed as a Style Class in Source Code

```
<cust:showDetailFrame id="showDetailFrame1"
       text="showDetailFrame 1"
       binding="#{backing_untitled1.showDetailFrame1}"
       styleClass="af_panelCustomizable_no-headercontent">
```

In the Property Inspector, it is expressed as shown in Example 9–8.

Example 9–8 Style Selector Expressed as a Style Class in the Property Inspector

```
af_panelCustomizable_no-headercontent
```

> **Note:** Icon selectors are not used as style classes. For a list of core customizable component style selectors, see Section 9.3, "Specifying Style Definitions for Portlet and Core Customizable Component Style and Icon Selectors".

The `StyleClass` property also enables for the entry of multiple style classes. Separate each style class by a space. You can enter style classes manually—that is, in the source code—or through the Property Inspector.

The Property Inspector provides a `StyleClass` dialog box for entering and arranging a list of style classes to be applied to the component. Click the Edit icon next to the `StyleClass` property in the Property Inspector to access the dialog box.

9.4.2 Changing Style-Related Properties

Change property values through the Oracle JDeveloper Property Inspector.

To specify style properties for a portlet or a core customizable component, perform the following steps:

1. In Oracle JDeveloper, open a page that contains a portlet or one or more core customizable components (`PanelCustomizable` or `ShowDetailFrame`).
2. In the Oracle JDeveloper editor, select the component to be styled.

   The Property Inspector populates with component-related properties. Double-click the Property Inspector tab to maximize its display size for easy editing. Double-click again to restore the Property Inspector to its usual display size.
3. In the Property Inspector, provide values as desired for the `ContentInlineStyle`, `InlineStyle`, and `StyleClass` properties.

   Note: All valid CSS attributes are not listed in the Property Inspector. For example, if you want to set the width of a `ShowDetailFrame`, you will not find the `Width` attribute in the Property Inspector under `InlineStyle`. You can go to the Source tab and type a style attribute into the component tag, as long as the attribute is compliant with at least CSS 2.0.
4. Click in an attribute's value column, and change values as desired as the following:
   - The `InlineStyle` property automatically populates with these values in the appropriate syntax.
   - Set style values manually for the `ContentInlineStyle` property.
   - For the `StyleClass` property:

     Click the column next to `StyleClass`, then click the Edit icon (**...**) to display the `StyleClass` dialog box.

     In the `StyleClass` dialog box, click **New** to add a new skeleton item to the list of style classes.

     Click the skeleton item to enter a style class.

     Style classes already applied to page components are listed (and selectable) under the **List** heading in the `StyleClass` dialog box.
5. Save your work.

For information about the attributes associated with style-related properties, see Section 9.4.3, "Attributes of the ContentInlineStyle and InlineStyle Properties".

9.4.3 Attributes of the ContentInlineStyle and InlineStyle Properties

To override run-time CSS style definitions on a specific component instance, at design time use the style-related properties available in the Property Inspector. For information about how to do this, see Section 9.4.2, "Changing Style-Related Properties".

In the Property Inspector, the `InLineStyle` property provides a sublist of attributes for styling the component. When you provide a value for one of these sublist attributes, the appropriate syntax (including the attribute name) populates the `InlineStyle` property. For example, when you assign the value `Arial,Helvetica,Geneva,sans-serif` to the `font-family` attribute, the `InlineStyle` property automatically populates with the following:

```
font-family:Arial Helvetica Geneva sans-serif;
```

In addition to the attributes provided under `InlineStyle`, you can manually enter attributes that do not appear on the Property Inspector's attribute list for both `InlineStyle` and `ContentInlineStyle`, as long as those attributes are in compliance with at least CSS 2.0. Such attributes have the following format in the `adfp:portlet` tag:

```
inlineStyle="border-style:none;font-family:Arial Helvetica Geneva sans-serif;"
```

The attributes of the `InlineStyle` property come from the World Wide Web Consortium's CSS standard. For additional information, go to:

`http://www.w3.org/Style/CSS/`

Table 9–7 lists and describes some of the style attributes available for customizable components through the Oracle JDeveloper Property Inspector. Note that portlets also use these style attributes.

Table 9–7 Style Attributes of Portlets and Core Customizable Components

Attribute	Description
`background-color`	The background color of the component. Choose from a list of colors, or click the Edit icon to select from a color palette.
`background-image`	The image that is displayed in the component background. Select to inherit from a parent component or to display no image, or click the Edit icon to select an image.
`background-repeat`	Specify whether and how the background image should repeat. Choose from: ■ `<none>`: Forgo repeating the image. ■ `no-repeat`: Forgo repeating the image. ■ `repeat`: Repeat the image to fill the container. ■ `repeat-x`: Repeat the image horizontally but not vertically. ■ `repeat-y`: Repeat the image vertically but not horizontally. ■ `inherit`: Inherit this setting from a parent component.
`border-color`	The color of the border that surrounds the component. Choose from a list of colors, or click the Edit icon to select from a color palette.

Table 9–7 (Cont.) Style Attributes of Portlets and Core Customizable Components

Attribute	Description
`border-style`	The style of border to draw around the component. Choose from: ■ `<none>`: Apply no style to the border. ■ `none`: Do not display a border (use border-style:none on the `ContentInlineStyle` property to turn off portlet. borders. Using this attribute with `InlineStyle` does not turn off portlet borders.) ■ `hidden`: Hide the border. ■ `dotted`: Display the border as a line of dots. ■ `double`: Display the border as a double line. ■ `groove`: Display the border as a grooved line. ■ `ridge`: Display the border as a ridged line. ■ `inset`: Display the border as a concave line. ■ `outset`: Display the border as a convex line. ■ `dashed`: Display the border as a line of dashes. ■ `solid`: Display the border as a solid line. ■ `inherit`: Inherit the border style from a parent component.
`border-width`	The thickness of the component border. Select `thick`, `medium`, `thin`, or `inherit`, to inherit border width from a parent component. Alternatively, click the Edit icon and define a thickness in your preferred unit of measurement.
`color`	The color of component text. Select from a list, or click the Edit icon to select from a color palette.
`font-family`	The font family (such as Arial, Helvetica, sans-serif) for component text. Enter this value manually.
`font-size`	The size of component text. Choose from: ■ `Choose Length`: Specify an absolute value for font-size in your preferred unit of measurement. ■ `Choose Percentage`: Specify font size as a percentage of normal text size defined for the browser. ■ `inherit`: Inherit font size from a parent component. ■ `large`: Select the next size larger than the font associated with the parent component. ■ `larger`: Increase the font size to larger than the font. associated with the parent component (`larger` is larger than `large`). ■ `medium`: Set the font size to the default font size of the parent component. ■ `small`: Set the font size to smaller than the default font size of the parent component. ■ `smaller`: Set the font size to smaller than the default font size of the parent component (`smaller` is smaller than `small`).

Table 9–7 (Cont.) Style Attributes of Portlets and Core Customizable Components

Attribute	Description
`font-style`	Specifies the font style of the component text. Choose from: ■ `<none>`: Apply no style to the font. ■ `normal`: The font is not italic. ■ `italic`: The font is italic. ■ `oblique`: The font is slanted to look italic, though the font family may not have a formal italic member. ■ `inherit`: Inherit font style from a parent component.
`font-weight`	Set the thickness of component text. Choose from: ■ `<none>`: Specify no weight for component font. ■ `normal`: Apply same weight at the parent component's default font weight. ■ `bold`: Set component text in bold. ■ `bolder`: Set component text darker than bold. ■ `light`: Set component text lighter than normal. ■ `lighter`: Set component text lighter than light. ■ `100–900`: Specify a value for text darkness—`100` is lightest and `900` is darkest—`400` is normal weight; `700` is bold. ■ `inherit`: Inherit font weight from a parent component.
`height`	The spacing to apply between lines of continuous text (also known as *leading*). Choose from: ■ `Choose Length`: Specify an absolute value for line height. ■ `Choose Percentage`: Set line height as a percentage of the line height of the parent component. ■ `inherit`: Inherit line height from the parent component. ■ `auto`: Set line height automatically, usually about 2 points larger than font size.
`list-style-image`	Specify an image to use as an indicator of a list item. Specify an image instead of defining a `list-style-type`. Choose from: ■ `inherit`: Inherit a list style image from a parent component. ■ `none`: Use no image as an indicator of a list item. Or click the Edit icon and select an image.
`list-style-type`	The type of indicator to use for items on a list. Use this instead of specifying a image through `list-style-image`. Select from among many styles offered on the list, including `<none>`, which means the browser's default style is used, and `inherit`, which means the `list-style-type` is inherited from a parent component.
`margin`	The border of space surrounding component content. Choose from: ■ `Choose Length`: Specify an absolute value for all margins. ■ `Choose Percentage`: Set a value as a percentage of the margin of a parent component. ■ `inherit`: Inherit margin value from a parent component. ■ `auto`: Set the value automatically according to browser defaults.

Table 9–7 (Cont.) Style Attributes of Portlets and Core Customizable Components

Attribute	Description
`outline-color`	The color of an outline surrounding the component. This differs from `border` in that a border falls inside an outline. Select from a list, or click the Edit icon to select from a color palette.
`outline-style`	The style of outline to draw around the component. See `border-style` for selection options.
`outline-width`	The thickness of the component outline. Select `thick`, `medium`, `thin`, or `inherit`, to inherit border width from a parent component. Alternatively, click the Edit icon and define a thickness in your preferred unit of measurement.
`padding`	The amount of space between the component and its margin or, if there is a border, between the component and its border. Choose from: ■ `Choose Length`: Specify an absolute value in your preferred unit of measurement. ■ `Choose Percentage`: Specify the value as a percentage of the padding set for a parent component. ■ `inherit`: Inherit the value from a parent component.
`text-align`	The horizontal alignment of component text. Choose from: ■ `<none>`: Use browser default. ■ `left` ■ `right` ■ `center` ■ `justify`: Align text so that all lines start and end at the same point left and right. ■ `auto`: Apply a value automatically, either left or right. ■ `inherit`: Inherit alignment from a parent component.
`text-decoration`	Decorative value to apply to component text. Choose from: ■ `<none>`: Use the browser default for hyperlinks; otherwise, no text decoration. ■ `none`: No text decoration, including no underline for hyperlinks. ■ `underline`: Underline all component text. ■ `overline`: Draw a line over all component text. ■ `line-through`: Strike out all component text. ■ `blink`: Make component text blink. ■ `inherit`: Inherit text decoration from a parent component.

Table 9–7 (Cont.) Style Attributes of Portlets and Core Customizable Components

Attribute	Description
`vertical-align`	Vertical alignment of component text. Choose from: ■ `Choose Length`: Specify an absolute value in your preferred unit of measurement. ■ `Choose Percentage`: Specify a value as a percentage of the value set for a parent component. ■ `inherit`: Inherit the value from a parent component. ■ `baseline`: Set the text on the baseline of text row. ■ `bottom`: Set the text on the bottom of text row, lower than baseline. ■ `middle`: Set the text in the middle of text row. ■ `sub`: Set the text as subscript. ■ `super`: Set the text as superscript.

9.5 Building a Run-Time Skin Selector

You may care to make the ability to switch skins at run time available to your portal administrators or to your users. This section describes two scenarios for providing a run-time skin switcher and persisting the switcher's changes across user sessions:

- Section 9.5.1, "Changing Skins for One User"
- Section 9.5.2, "Changing Skins for All Users"

9.5.1 Changing Skins for One User

This section describes how to store skin changes on a browser-by-browser basis. When a user changes the application skin, the application displays the change, which is stored in a browser cookie on the user's local computer. Subsequent browser sessions on the local computer display the application using the selected skin. If you select a different browser, disable browser cookies, or display the application on a different computer, then the skin change is not reflected.

Making skin changes persistent on a browser-by-browser basis entails four steps:

- Section 9.5.1.1, "Creating a Java Class to Serve As a Managed Bean (ClientSkinBean.java)"
- Section 9.5.1.2, "Registering the ClientSkinBean Class As a JSF Managed Bean"
- Section 9.5.1.3, "Using an Expression Language Expression to Reference the Bean"
- Section 9.5.1.4, "Adding User Interface Components to the Page for Switching the Skin"

9.5.1.1 Creating a Java Class to Serve As a Managed Bean (ClientSkinBean.java)

To create a Java class to serve as a managed bean, perform the following steps:

1. In the Applications Navigator, right-click the ViewController project and select **New** from the context menu.
2. In the New Gallery, expand the **General** node and select **Simple Files**.
3. In the New Gallery, under **Items** select **Java Class**.
4. Click **OK**.

5. In the **Name** field of the Create Java Class dialog box, enter a name for the Java class.

   For example, `ClientSkinBean`.

6. Ensure that `view` is specified in the **Package** field, and `java.lang.Object` is specified in the **Extends** field; leave **Optional Attributes** at their default values.

7. Click **OK**.

   The Java class appears in the Applications Navigator under the following hierarchy:

   ```
   <project_name> */ For example, ViewController /*
      Application Sources
         view
            <Java_class_name> */ For example, ClientSkinBean.java /*
   ```

8. In the Applications Navigator, right-click the Java class and select **Open** from the context menu.

   This opens the Java class in the Editor pane.

9. In the Editor pane, switch to Source view and add code for setting a persistent skin through a browser cookie.

   Example 9–9 illustrates one implementation of this code.

Example 9–9 Java Class for Persisting a Skin Change Through a Browser Cookie

```
package view;

import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
*  ClientSkinBean is a class for persisting ADF JSF skin changes as
*  a cookie in the user's browser.
*  It is intended to be used as a JSF backing bean.
*  It stores the skin setting under the cookie name "ADFSkin".
*/
public class ClientSkinBean {
   private String _skin = null;
   // You can modify the _cookieName for your own application.
   private static String _cookieName = "ADFSkin";

   public ClientSkinBean() {
   }

/**
     *  Get the current skin setting. The default is oracle.
     *  @return the name of the skin to use
     */
   public String getSkin() {
      if (_skin == null)
      {
         FacesContext context = FacesContext.getCurrentInstance();
         ExternalContext extContext = context.getExternalContext();
         HttpServletRequest servletRequest =
```

```
            (HttpServletRequest) extContext.getRequest();
         Cookie[] cookies = servletRequest.getCookies();

         for (int i = 0; i < cookies.length; i++) {
            Cookie cookie = cookies[i];
            if (_cookieName.equals(cookie.getName()))
            {
               _skin = cookie.getValue();
               return _skin;
            }
      }
         _skin = "oracle";
         return _skin;
   }
   else
   return _skin;
  }

/**
  *  Make the skin setting persistent. The default is oracle.
  *  @param skin the name of the skin.
  */
public void setSkin(String skin) {
   FacesContext context = FacesContext.getCurrentInstance();
   ExternalContext extContext = context.getExternalContext();
   HttpServletResponse servletResponse =
      (HttpServletResponse) extContext.getResponse();

   Cookie userCookie = new Cookie(_cookieName, skin);
   // Set the cookie's age to a year. You can set this to your own value.
   userCookie.setMaxAge(60*60*24*365);
   servletResponse.addCookie(userCookie);
   _skin = skin;
  }
}
```

10. Save your changes.

9.5.1.2 Registering the ClientSkinBean Class As a JSF Managed Bean

Use the `faces-config.xml` file to register the Java class as a managed bean. The `faces-config.xml` file is stored in the project's WEB-INF folder.

To register the Java class, perform the following steps:

1. In the Applications Navigator, right-click the `faces-config.xml` file and select **Open** from the context menu.
2. Register the Java class that you created in Section 9.5.1.1.

   Example 9–10 illustrates how to register the class created in Section 9.5.1.1. The portion related to skin is bold.

Example 9–10 Registering a Java Class as a Managed Bean

```
<faces-config xmlns="http://java.sun.com/JSF/Configuration">
   <managed-bean>
      <managed-bean-name>clientSkinBean</managed-bean-name>
      <managed-bean-class>view.ClientSkinBean</managed-bean-class>
      <managed-bean-scope>session</managed-bean-scope>
   </managed-bean>
   <application>
```

```
      <default-render-kit-id>oracle.adf.core</default-render-kit-id>
   </application>
</faces-config>
```

3. Save your changes.

9.5.1.3 Using an Expression Language Expression to Reference the Bean

Use an Expression Language (EL) expression in the `adf-faces-config.xml` file to reference the managed bean. Referencing the bean enables for the expression of a variable value, rather than a fixed value, for application skin selection.

For more information about specifying an application skin, see Section 9.2, "Applying Custom Skins to Applications".

To enter an EL expression in an application's `adf-faces-config.xml` file, perform the following steps:

1. In the Applications Navigator, right-click the `adf-faces-config.xml` file and select **Open** from the context menu.

   The `adf-faces-config.xml` file is located in the project WEB-INF folder.

2. Between the `<skin-family></skin-family>` tags, enter an EL expression that references the managed bean.

   For example:

   ```
   <adf-faces-config xmlns="http://xmlns.oracle.com/adf/view/faces/config">
      <skin-family>#{clientSkinBean.skin}</skin-family>
   </adf-faces-config>
   ```

3. Save your changes.

9.5.1.4 Adding User Interface Components to the Page for Switching the Skin

Once you put the logic in place to switch skins at run time, it is time to provide the User Interface (UI) components that render the skin switcher.

To create a skin-switcher UI, perform the following steps:

1. Create an application `jspx` file or open an existing file.

   To open an existing file, right-click it in the Applications Navigator and select **Open** from the context menu. Application `jspx` files are stored in the project's `Web Content` folder. For information about creating an application `jspx` file, see "Requirements for Pages".

2. In the Editor pane, switch to Source view, and enter the logic for a skin switcher.

   Example 9–11 illustrates one implementation of skin-switcher code.

Example 9–11 One Implementation of a Skin Switcher

```
<jsp:root version="2.0">
   <jsp:output omit-xml-declaration="true" doctype-root-element="HTML"
      doctype-system="http://www.w3.org/TR/html4/loose.dtd"
      doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN"/>
   <jsp:directive.page contentType="text/html;charset=windows-1252"/>
   <f:view>
      <afh:html>
         <afh:head title="ClientChangeSkin">
            <meta http-equiv="Content-Type" content="text/html;
               charset=windows-1252"/>
```

```
          </afh:head>
          <afh:body>
             <h:form>
                <af:selectOneChoice label="Select Look and Feel"
                   id="showOneChoice1" value="#{clientSkinBean.skin}">
<!--  Specify skins to display on selection list: default, custom, or both  -->
                   <af:selectItem label="oracle" value="oracle"/>
                   <af:selectItem label="[menu item label]" value="[skin name]"/>
                   <af:selectItem label="[menu item label]" value="[skin name]"/>
                   <af:selectItem label="minimal" value="minimal"/>
                   <af:selectItem label="simple" value="simple"/>
                </af:selectOneChoice>
                <af:commandButton text="Apply Selected Look and Feel"
                   id="commandButton1"/>
             </h:form>
          </afh:body>
       </afh:html>
   </f:view>
</jsp:root>
```

3. Save your changes.

Run the page and try switching skins. Select a different skin from the list and click the command button to implement the change take effect. If you use the code provided in Example 9–11, then observe the changes to the command button, particularly between the `oracle` and `simple` skins. In creating your own code, you may want to reference your own skins instead of these default skins. When you close and then open the browser, the skin change persists.

9.5.2 Changing Skins for All Users

This approach differs from the approach described in Section 9.5.1 in terms of who is affected by the skin change. In this approach, a skin change affects everyone who uses the application, and not just the browser where the change was made. If the application will run in an environment where browsers could have cookies disabled, then this approach should be used. Because the change affects all users, this approach is typically used in conjunction with some security mechanism to ensure that only authorized users can make the change.

Making skin persistent across all users entails four steps:

- Section 9.5.2.1, "Creating a Java Class to Serve As a Managed Bean (ServerSkinBean.java)"
- Section 9.5.2.2, "Registering the ServerSkinBean Class As a JSF Managed Bean"
- Section 9.5.2.3, "Using an EL Expression to Reference the Bean"
- Section 9.5.2.4, "Adding UI Components to the Page for Switching the Skin"

9.5.2.1 Creating a Java Class to Serve As a Managed Bean (ServerSkinBean.java)

To create a Java class to serve as a managed bean, perform the following steps:

1. In the Applications Navigator, right-click the ViewController project and select **New** from the context menu
2. In the New Gallery, expand the **General** node and select **Simple Files**.
3. In the New Gallery, under **Items** select **Java Class**.
4. Click **OK**.

5. In the **Name** field of the Create Java Class dialog box, enter a name for the Java class.

   For example, `ServerSkinBean`.

6. Ensure that `view` is specified in the **Package** field, and `java.lang.Object` is specified in the **Extends** field; leave **Optional Attributes** at their default values.

7. Click **OK**.

   The Java class appears in the Applications Navigator under the following hierarchy:

   ```
   <project_name> */ For example, ViewController /*
      Application Sources
         view
            <Java_class_name> */ For example, ServerSkinBean.java /*
   ```

8. In the Applications Navigator, right-click the Java class and select **Open** from the context menu.

   This opens the Java class in the Editor pane.

9. In the Editor pane, switch to Source view and add code for making the skin persistent through a browser cookie.

   Example 9–12 illustrates one implementation of this code.

Example 9–12 Java Class for Persisting a Skin Change that Affects All Users

```
package view;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;

import java.io.FileWriter;
import java.io.IOException;

import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletResponse;

/**
  *  ServerSkinBean is a class for making ADF JSF skin persistent as
  *  application customization. It is intended to be used as a JSF backing bean.
  *  It stores the skin setting under the application's context root in a
  *  simple text file called appSkinConfig.
  */
public class ServerSkinBean {
   private String _skin = null;
   private static String _defaultSkin = "oracle";

   public ServerSkinBean() {
   }

   /**
     *  Get the current skin setting. The default is oracle.
     *  @return the name of the skin to use
     */
   public String getSkin() {
```

```
        FacesContext context = FacesContext.getCurrentInstance();
        ExternalContext extContext = context.getExternalContext();
        ServletContext svtCtx = (ServletContext)extContext.getContext();
        String appSkinConfig = svtCtx.getRealPath("/" + "appSkinConfig");

        try {
              // Attempt to open the skin config file and read the setting
              FileReader fileReader =
                    new FileReader(appSkinConfig);
              BufferedReader bufferedReader = new BufferedReader(fileReader);
              _skin = bufferedReader.readLine();
              if ((_skin == null) || (_skin.length() == 0) || (_
                    skin.trim().length() == 0)) {
              setSkin(_defaultSkin);
             }
              return _skin;
        } catch (FileNotFoundException e) {
              setSkin(_defaultSkin);
              return _skin;
        } // no saved state
        catch (IOException e) {
              System.out.println(e.getMessage());
              _skin = _defaultSkin;
              return _skin;
        }
    }

/**
    *  Make the skin setting persistent. The default is oracle.
    *  @param skin the name of the skin.
    */
public void setSkin(String skin) {
        FacesContext context = FacesContext.getCurrentInstance();
        ExternalContext extContext = context.getExternalContext();
        ServletContext svtCtx = (ServletContext)extContext.getContext();
        String appSkinConfig = svtCtx.getRealPath("/" + "appSkinConfig");

        try
        {
            _skin = skin;
            FileWriter writer = new FileWriter(appSkinConfig, false);
            writer.write(_skin + "\n");
            writer.close();
        }
        catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

10. Save your changes.

9.5.2.2 Registering the ServerSkinBean Class As a JSF Managed Bean

Use the `faces-config.xml` file to register the Java class as a managed bean. The `faces-config.xml` file is stored in the project's WEB-INF folder.

To register the Java class, perform the following steps:

1. In the Applications Navigator, right-click the `faces-config.xml` file and select **Open** from the context menu.

2. Register the Java class you created in Section 9.5.2.1.

   Example 9–13 illustrates how to register the class created in Section 9.5.2.1. The portion relevant to skin is bold.

Example 9–13 Registering a Java Class as a Managed Bean

```
<faces-config xmlns="http://java.sun.com/JSF/Configuration">
   <managed-bean>
      <managed-bean-name>serverSkinBean</managed-bean-name>
      <managed-bean-class>view.ServerSkinBean</managed-bean-class>
      <managed-bean-scope>session</managed-bean-scope>
   </managed-bean>
   <application>
      <default-render-kit-id>oracle.adf.core</default-render-kit-id>
   </application>
</faces-config>
```

3. Save your changes.

9.5.2.3 Using an EL Expression to Reference the Bean

Use an EL expression in the `adf-faces-config.xml` file to reference the managed bean. Referencing the bean enables for the expression of a variable value, rather than a fixed value, for application skin selection.

For more information about specifying an application skin, see Section 9.2, "Applying Custom Skins to Applications".

To enter an EL expression in an application's `adf-faces-config.xml` file, perform the following steps:

1. In the Applications Navigator, right-click the `adf-faces-config.xml` file and select **Open** from the context menu.

   The `adf-faces-config.xml` file is located in the project WEB-INF folder.

2. Between the `<skin-family></skin-family>` tags, enter an EL expression that references the managed bean.

   For example:

   ```
   <adf-faces-configxmlns="http://xmlns.oracle.com/adf/view/faces/config">
      <skin-family>#{serverSkinBean.skin}</skin-family>
   </adf-faces-config>
   ```

3. Save your changes.

9.5.2.4 Adding UI Components to the Page for Switching the Skin

Once you put the logic in place to switch skins at run time, it is time to provide the UI components that render the skin switcher.

To create a skin switcher UI, perform the following steps:

1. Create an application `jspx` file or open an existing file.

   To open an existing file, right-click it in the Applications Navigator and select **Open** from the context menu. Application `jspx` files are stored in the project's `Web Content` folder. For information about creating an application `jspx` file, see "Requirements for Pages".

2. In the Editor pane, switch to Source view, and enter the logic for a skin switcher.

Example 9–14 illustrates one implementation of skin-switcher code.

Example 9–14 One Implementation of a Skin Switcher

```
<jsp:root version="2.0">
   <jsp:output omit-xml-declaration="true" doctype-root-element="HTML"
      doctype-system="http://www.w3.org/TR/html4/loose.dtd"
      doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN"/>
   <jsp:directive.page contentType="text/html;charset=windows-1252"/>
   <f:view>
      <afh:html>
         <afh:head title="ServerChangeSkin">
            <meta http-equiv="Content-Type" content="text/html;
               charset=windows-1252"/>
         </afh:head>
         <afh:body>
            <h:form>
               <af:selectOneChoice label="Select Look and Feel"
                  id="showOneChoice1" value="#{serverSkinBean.skin}">
<!--  Specify skins to display on selection list: default, custom, or both  -->
                  <af:selectItem label="[menu item label]" value="[skin name]"/>
                  <af:selectItem label="[menu item label]" value="[skin name]"/>
                  <af:selectItem label="oracle" value="oracle"/>
                  <af:selectItem label="minimal" value="minimal"/>
                  <af:selectItem label="simple" value="simple"/>
               </af:selectOneChoice>
               <af:commandButton text="Apply Selected Look and Feel"
                  id="commandButton1"/>
            </h:form>
         </afh:body>
      </afh:html>
   </f:view>
</jsp:root>
```

3. Save your changes.

Run the page and try switching skins. Select a different skin from the list and click the command button to make the change take effect. If you use the code provided in Example 9–14, then observe the changes to the command button, particularly between the `oracle` and `simple` skins. When you close and then open the browser, the skin change persists.

In creating your own code, you may want to reference your own skins instead of these default skins. Example 9–14 provides two entries where custom skins can be referenced. You can reference as many custom skins as you want. You can also omit referencing default skins.

10

Securing Your WebCenter Application

This chapter describes how to use Oracle ADF Security in your WebCenter application to handle authentication and authorization.

This chapter includes the following sections:

10.1 Introduction to WebCenter Application Security

WebCenter applications are dynamic, run time driven, and often involve input from users in the form of customization and personalization. Therefore, the use of traditional J2EE security is limiting. To go beyond the limitations of J2EE security, you can use Oracle Application Development Framework (Oracle ADF) security, which is based on the Java Authentication and Authorization Service (JAAS). JAAS is a standard security Application Programming Interface (API) that is added to the Java language through the Java Community Process and enables applications to authenticate users and enforce authorization.

J2EE security secures a path based on roles. For example, in the SRDemo application, three core roles, which are J2EE security roles, determine who is authorized to perform certain functions or to have access to pages. Each user must be classified with one of three roles: user, technician, or manager. During the development of the application, security constraints are defined that map specific URL patterns to a specified J2EE security role. For example, the URL pattern `/app/management/*` can be mapped to

the manager role so that only managers can access those pages. Constraints and users or security roles are defined in the `web.xml` file, while the mapping of the static roles to those within the identity management solution is defined in the deployment descriptor for the application server (in OC4J, this is the `orion-web.xml` file). Because both of these files are deployed with the application, you cannot change the security constraints at run time. To create a new role, you must redeploy the application.

The use of the JAAS-based Oracle ADF Security model addresses these problems. Oracle ADF implements a JAAS security model through integration with JAZN, Oracle's implementation of the JAAS service. While to date, JAAS has mostly been used for authentication, it was also designed to enable for the definition of an authorization model. (After all, that is what the second A in JAAS is all about!). Traditionally this required the use of custom code and was not easy to implement, but Oracle ADF Security simplifies the implementation of a JAAS authorization model by exposing it in a declarative way.

With Oracle ADF Security, the required permissions (activities) to access an application are not a static role definition, but instead are deployed with the application. Because these permissions are mapped at run time, changes in the security profile information, such as the addition of a new role after deployment, will automatically be applied without a need to update the application. Furthermore, because Oracle ADF Security is not constrained by the URL to a secured resource, you can define more granular permissions to resources. In your applications, you can also use Expression Language (EL) to show or hide items on a page based on a user's permissions, which are defined in the run time policy store. This policy store can be defined in one of the two JAZN resource providers: JAZN-XML, which specifies the permission grants in the file `system-jazn-data.xml`, or JAZN-LDAP, which stored these grants in an LDAPv3-compliant directory, such as Oracle Internet Directory.

Using Oracle ADF Security, you can simplify the organization of your application. Because you can specify security by a declarative permission rather than through the URL mapping of a security constraint, the location of the resource to be secured is no longer mandated by those security constraints and you can organize pages in whatever structure is logical for your application. For example, you can store pages in a single flat directory structure based on their file type (`jsp`, `jspx`, and so on), or in a more traditional hierarchical directory tree.

While Oracle ADF Security adds significant new functionality to an application, the JAAS specification itself is part of the J2EE declarative security model. That is, it is an extension of the overall J2EE container security platform. Thus, the more granular Oracle ADF Security implementation can be viewed as an extension to the standard J2EE container security and is executed after the standard security constraints have been processed.

Within the Oracle ADF framework, JAAS-based security is enforced by the use of specific servlet filters and the data binding layer of the application. The filters and the binding layer work together to trap incoming requests, determine the current user's permissions, and block or enable the request accordingly.

The use of Oracle ADF security enables WebCenter applications to easily adjust to real-world business security requirements, because rather than securing paths, you secure actions with JAAS. JAAS-based Oracle ADF Security provides:

- More granular declarative inter-portlet security. Because J2EE security is URL-based or page-based, it is not possible to have a finer level of access control. With Oracle ADF security, different roles can perform different levels of activity at the same URL.

- The ability to create new roles and their associated access privileges at run time, because the policies are stored external to the application.
- Simplified permission assignment by using hierarchical roles, which enable for the inheritance of permissions. While J2EE security roles that are used by J2EE security constraints are flat, JAAS permissions reference enterprise roles, which can be nested.

The following topics are covered in this section:

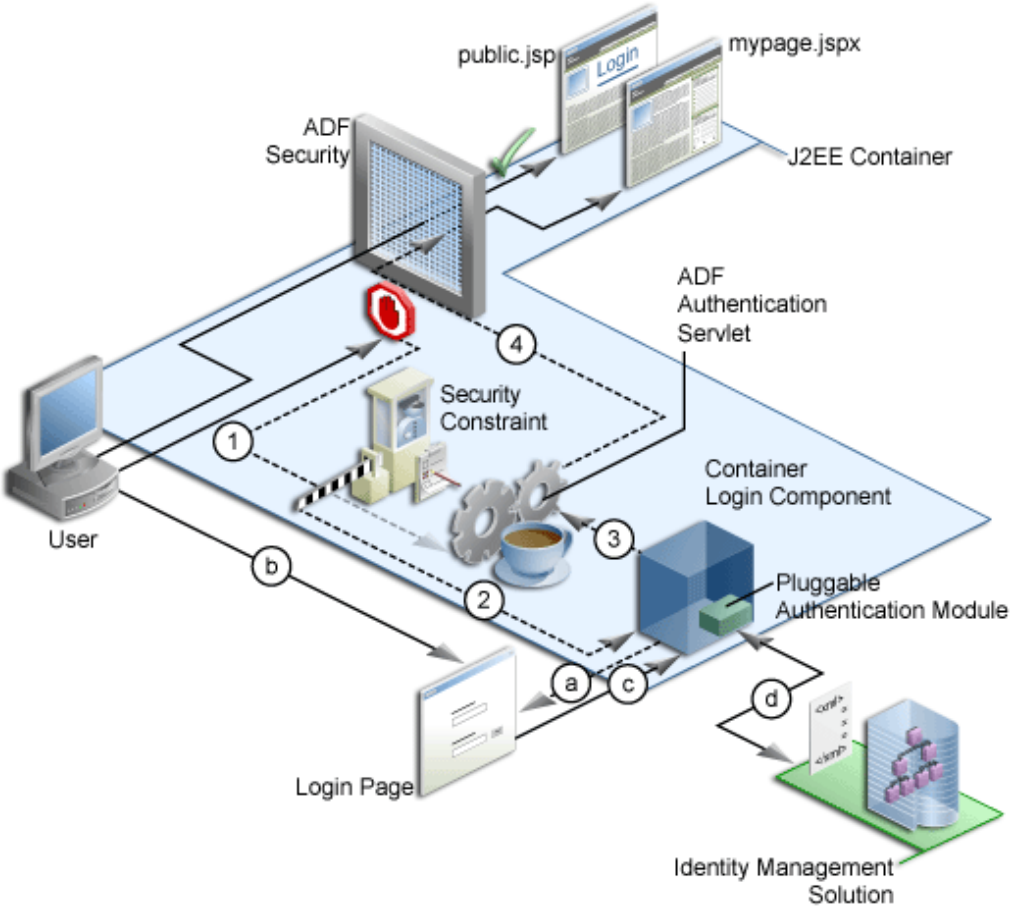
10.1.1 Authentication

Oracle ADF Security enables for implicit and explicit authentication. In an implicit authentication scenario, as shown in Figure 10–1, when an unauthenticated user tries to access a page, the adfBindings servlet filter intercepts the request and checks to see if the page is defined as viewable by *anyone*.

On the first access to a page, if there is no subject defined, then one is created containing the *anonymous* user principal and the anyone role principal. With this role principal, the user can access any page on which the *view* privilege has been granted to the anyone role. For example, `public.jsp`. See Section 10.1.2, "Authorization" for a discussion on authorization.

However, if the requested page is secured (that is, not defined as viewable to anyone, such as `mypage.jspx`) then the `adfBindings` servlet filter redirects the request to the Oracle ADF authentication servlet (Step 1), passing in the URL to the requested page as the success URL.

Figure 10–1 Oracle ADF Security Implicit Authentication



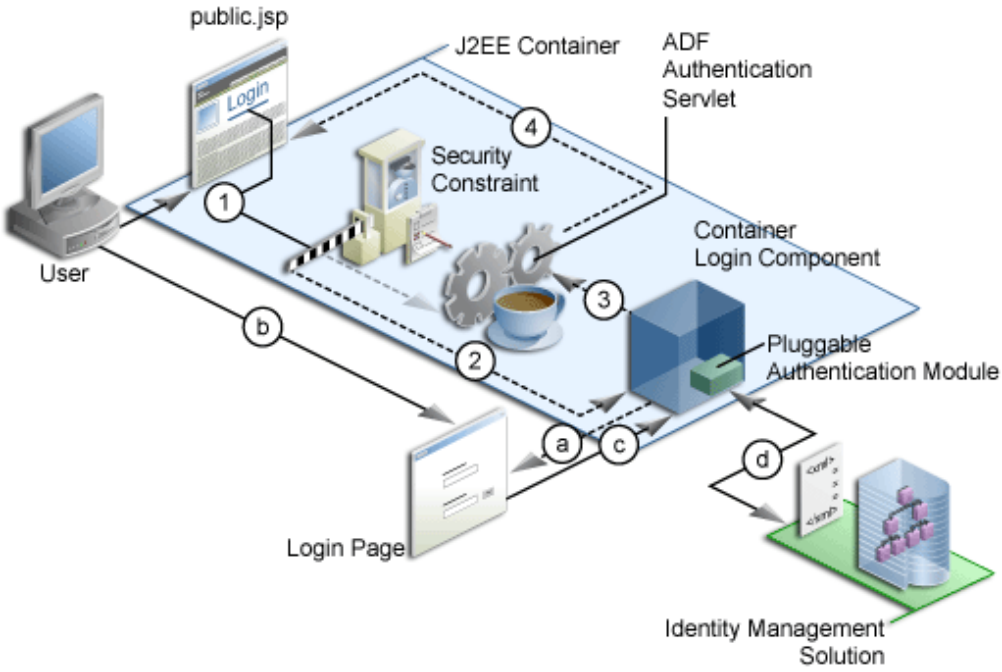
The `adfAuthentication` servlet has a J2EE security constraint on it, that results in the J2EE container invoking the configured login mechanism (Step 2). Based on the container's login configuration, the user is prompted to authenticate. In the case of form-based authentication, the appropriate login form is displayed (Step a) and the user enters his credentials (Step b), after which the form is posted back to the container's `j_security_check()` method (Step c).

The J2EE container authenticates the user, using the configured pluggable authentication module (Step d) and on successful authentication, the container redirects the user back to the servlet that initiated the authentication challenge. In this case, that is the `adfAuthentication` servlet (Step 3). On returning to the `adfAuthentication` servlet, the success URL value is used to subsequently redirect to the originally requested URL (Step 4).

In an explicit authentication scenario, as shown in Figure 10–2, an unauthenticated user (with only the anonymous user principal and anyone role) clicks the **Login** link on a public page (Step 1). The Login link is a direct request to the `adfAuthentication` servlet, which is secured through a J2EE security constraint.

In this scenario, the current page is passed as a parameter to `adfAuthentication` servlet. As with the implicit case, the security constraint redirects the user to the container's login component (Step 2). After the container authenticates the user, as described in steps a through d in the implicit authentication case, the request is returned to the `adfAuthentication` servlet (Step 3), which subsequently returns the user to the public page, but now with his new user and role principal.

Figure 10–2 Oracle ADF Security Explicit Authentication

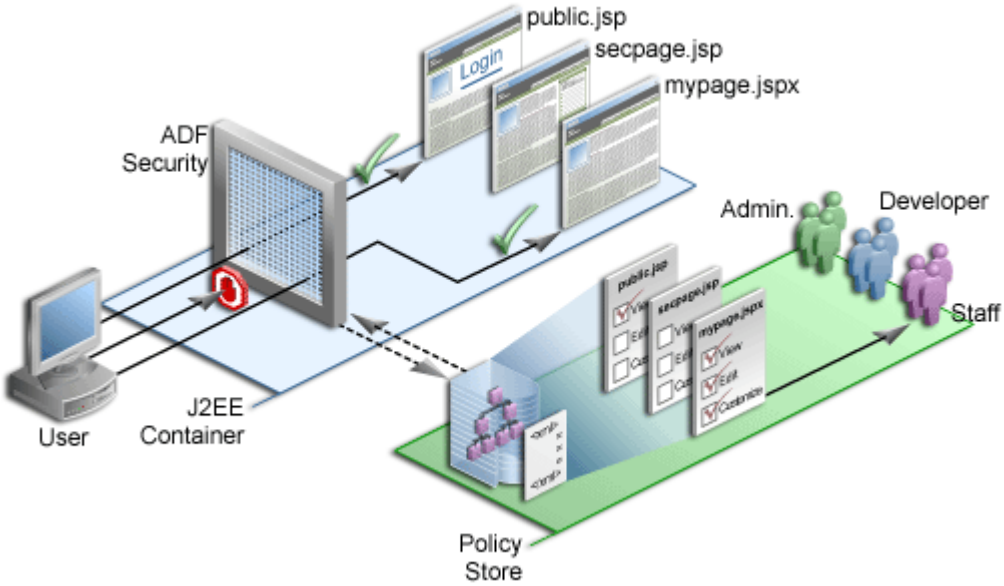


10.1.2 Authorization

This section describes how the authorization functionality has moved to a policy store definition that is accessed at run time and contains permissions against actions on objects. These permissions are established when setting the authorization on the objects in Oracle JDeveloper.

Figure 10–3 illustrates the authorization process.

Figure 10–3 Oracle ADF Security Authorization



The user is a member of the enterprise role *Staff* in the identity management solution. When the user tries to access `mypage.jspx`, the Oracle ADF Security enforcement

logic intercepts the request and checks the page definition of that page to see if permission is required.

Because the user has not yet logged in, the security context does not have a subject (a container that represents the user) yet, and it creates a subject with the anonymous user principal (a unique definition of the user) and the anyone role principal.

Any permission granted to the anyone role effectively makes the secured resource a public resource, accessible by all users (both authenticated and unauthenticated). Hence, with the anyone role principal, the user can access any page on which the view privilege has been granted to the anyone role. For example, the `public.jsp` page.

Because `mypage.jspx` requires the view privilege to a role other than anyone, the user is challenged to authenticate. After successful authentication, the user will have a specific subject. The security enforcement logic checks the policy store to determine which role is required to view `mypage.jspx` and whether the user is a member of that role. In this case, the view privilege has been granted to the role Staff and because the user is a member of this role, he is enabled to navigate to `mypage.jspx`.

Similarly, when the user tries to access `secpage.jsp`, a page to which the user does not have the `view` privilege, access is denied.

10.1.2.1 Oracle ADF Permissions

Oracle ADF features a number of security-aware components, such as pages and data controls. These components map to a specific permission class that has a set of predefined scoped actions. By granting a permission to an action, a developer can authorize a user or role to perform the specified action on the associated secured component.

> **Caution:** In this release, policy information in the JAAS policy store is not scoped by application. As a result, if two applications refer to the same permission target, then they will both refer to the same grants against that target. This may produce unintentional results. To avoid this, applications should name their resources accordingly to provide for application scoping. For example, instead of simply calling a page `Page1.jspx`, a more specific name such as `App1_Page1.jspx` will help isolate the policies.

Oracle ADF Security defines the following four authorization points:

- Page
- Method
- Iterator
- Attribute

Using the Authorization Editor shown in Figure 10–10, you can define the desired authorization to perform an action on an object to a specific role.

The Authorization Editor exposes enterprise roles that are defined in the policy store (the `system-jazn-data.xml` file located in the embedded OC4J) and displays the actions that are defined against a specific component type (a page, a method, an iterator, or an attribute). Examples of these actions are View and Edit. To implement the authorization policy, check the action against one or more of the displayed roles.

The Authorization Editor writes the permission information to the policy store. The policy defines a permission type, the resource that is secured, the actions that can be

performed against that resource, and to whom that policy is being assigned or granted. The policy is defined by a Grant, which contains both a Grantee and one or more Permissions.

A Grantee defines to whom the policy is applied. Example 10–1 shows how grants are defined in the `system-jazn-data.xml` file.

Example 10–1 Grants in the system-jazn-data.xml file

```
<grant>
  <grantee>
    <principals>
      <principal>
        <class>oracle.adf.share.security.authentication.ADFRolePrincipal</class>
        <name>anyone</name>
      </principal>
      . . .
    </principals>
  </grantee>
  <permissions>
    <permission>
       <class>oracle.adf.share.security.authorization.RegionPermission</class>
       <name>oracle.srdemo.view.pageDefs.app_SRWelcomePageDef</name>
       <actions>view</actions>
    </permission>
    . . .
  </permissions>
</grant>
```

In this Grant, the role principal *anyone* has been assigned View permission on the SRWelcome page. This permission has been specified against the associated page definition file of the SRWelcome page.

> **Note:** Page security policies are implemented through the `RegionPermission` class. In this sample, the *anyone* role has also been given permission to execute any method in the application (defined by the `MethodPermission` class) through the use of the wildcard `(*)` as the name of the method.

Page Permission

Page authorization policies are defined against the *Page Definition* file. To edit page permissions, perform the steps in Section 10.2.3.2, "Securing Pages in Your Application".

The available actions for a page are shown in Table 10–1.

Table 10–1 Page Permissions

Action	Description
View	View the page.
Personalize	Personalize portlets on the page. This page permission is required if you want to personalize portlets that are exposed on a page. Personalization enables you to make changes that are visible only to yourself.
Customize	Customize a page. Changes made will be visible to all users.

Table 10–1 (Cont.) Page Permissions

Action	Description
Edit	Edit content displayed on the page. The Edit action is not applicable for this release.
Grant	Grant privileges to other users. This action is not used in the design time.

Caution: Granting View permissions to a page does not automatically grant access to any methods, iterators, and attributes that are added to the page. These must be secured separately.

Method Permission

Method permissions secure the ability to execute specific named method classes within the application.

The available actions for methods are shown in Table 10–2.

Table 10–2 Method Permissions

Action	Description
Invoke	Execute a named method in the application

Method permissions are often used to enforce application security within a page rather than access to a page itself. For example, you can hide a Delete button based on the user's ability or inability to execute the delete method.

Iterator permission

Iterator permissions relate to the ability to scroll through a data set that is exposed through a data control and perform Data Manipulation Language (DML)-style actions on that data.

The available actions for iterators are shown in Table 10–3.

Table 10–3 Iterator Permissions

Action	Description
Create	Create a new record
Read	Read the current data set pointed to by the iterator
Update	Update the currently selected data set
Delete	Delete the currently selected data set

Attribute permission

Attribute permissions relate to the ability to view and update the specific attributes of an object returned within an iterator.

The available actions for attributes are shown in Table 10–4.

Table 10–4 Attribute Permissions

Action	Description
Read	Read the value of the specified attribute

Table 10–4 (Cont.) Attribute Permissions

Action	Description
Update	Update the value of the specified attribute

10.1.2.2 Anonymous Access

It is a common requirement that some portal pages should be seen by all users regardless of their specific access privileges. An Internet site's Welcome page, for example, should be seen by all visitors to the site, while a corporate site should be public only to those who have identified themselves through authentication.

In both cases, the page may be considered public, because the ability to view the page is not defined by the users' specific permissions. Rather, the difference is whether the user is anonymous or a known identity.

The use of public is different from traditional J2EE security, which does not let you to distinguish between completely unsecured (security has not been turned on or implemented) and public content.

In the Oracle ADF Security model, you explicitly differentiate between the absence of security and public access to content, by granting access privileges to the *anyone* role principal. *anyone* is a role that encompasses both known and anonymous users (thus, permission granted to *anyone* enables access to a resource by unauthenticated users, for example, guest users). To implement public access to authenticated users only, the policy must be defined for the *users* role principal.

10.1.3 External Application Credentials and Portlets

External Applications and Credential Provisioning in the Oracle WebCenter Suite provide a means of accessing content from applications that require user authentication. When an Oracle PDK portlet producer's implementation depends on an application that handles its own authentication, you can associate that producer with an external application definition. At design time, this is a simple matter of registering the external application, then selecting the external application from a list when you register or edit an Oracle PDK portlet producer. At run time, the producer uses the information associated with the external application to authenticate the user to the application, and consequently consume its portlets. Note that the producer code is responsible for actually performing the authentication interaction with the external application. The external application support provided with Oracle WebCenter Suite simply provides the information needed for authentication to the portlet producer through the Oracle PDK.

The user provides login credentials when prompted and these credentials are preserved in a credential store. The credential store subsequently supplies that information during authentication. The user supplies the credentials only once. This information is stored at the WebCenter application end, mapped to the user's WebCenter application user name. The mapping information is read from the credential store for further requests. The portlet run time framework obtains the user's mapped credentials from the credential store.

The Credential Provisioning page is a JSF page (`.jspx`) that is built based on the information provided through the external application definition. At run time, the Credential Provisioning page displays login data fields composed of the data fields specified through external application registration. Users fill in the data fields with their login information. Login information is passed to the producer, which in turn passes the login values to the application. The application provides the producer with

the requested portlets. Entering the credentials in the Credential Provisioning page also results in the credentials being persisted in the credential store.

For example, a producer provides a weather portlet from a portlet-producing application that has its own authentication mechanism. The developer:

- Registers an external application through the External Application Registration Wizard that captures information about the application's authentication mechanism.
- Adds a Credential Provisioning page that is a prebuilt page. This page can be modified for look and feel. At run time, this page displays the login data entry fields specified through external application registration.
- Associates the external application to an Oracle PDK Portlet Producer, during producer registration or edit.

At run time, when a user accesses the weather portlet, a login page (that is, the Credential Provisioning page) is displayed, and the user enters login information. This information is passed through the producer to the portlet-providing application, which then passes the weather portlet back to the producer (after authentication). The producer, in turn, provides the weather portlet to the portlet-consuming application, which displays the portlet to the user.

The login information the user entered is preserved in a credential store, which handles logins for future sessions. The user does not have to enter login information again (unless the user's credentials change).

For information about how to register an external application and create a Credential Provisioning page, see Section 10.7, "Accessing External Applications Requiring Credentials".

10.2 Setting Up Security for Your Application

This section describes how you can secure your application by using Oracle ADF Security, which is based on JAAS. Defining security for your application involves the following high-level steps:

- Section 10.2.1, "Defining Roles for Developing Secured WebCenter Applications"
- Section 10.2.2, "Configuring Security for Your Application"
- Section 10.2.3, "Defining Access Policies"
- Section 10.2.4, "Enforcing Security Policies in Your Application"
- Section 10.2.5, "Configuring Deployment Descriptor Files with Security Information"

10.2.1 Defining Roles for Developing Secured WebCenter Applications

As WebCenter security is based on a role-based access control mechanism with permissions granted to enterprise roles, you must create the appropriate roles in the identity management system that you use in the development environment. In the case of development environment, these roles should exist in the `system-jazn-data.xml` file, which is the default identity management solution.

There are two options for defining these roles:

- The developer maintains a list of the actual roles that will be used in the production environment and uses these roles for policy definition. This enables for simple migration of policies at deployment time.

- The developer defines a list of temporary role names that represent the ultimate production roles and uses these roles for policy definition. At deployment time, these policies must be updated to reflect the actual production roles. See Section 12.2.4.2, "Updating Policy Information (Optional)" for information about how to update the policy information in the `app-jazn-data.xml` file while deploying your application.

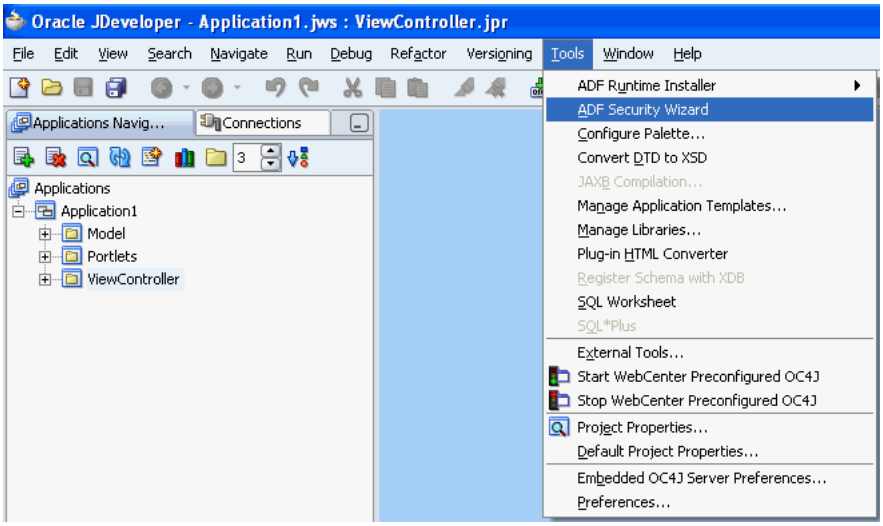
To create a set of test roles and users within the development environment, see the appendix titled "How to Set Up the Tutorial Identity Store" in the *Oracle WebCenter Framework Tutorial*.

10.2.2 Configuring Security for Your Application

Secure your application by running the ADF Security Wizard as follows:

1. In the Applications Navigator, select **ViewController**.
2. From the Tools menu, choose **ADF Security Wizard** as shown in Figure 10–4. The ADF Security Wizard will guide you through the configuration process.

Figure 10–4 ADF Security Wizard Option in the Tools Menu

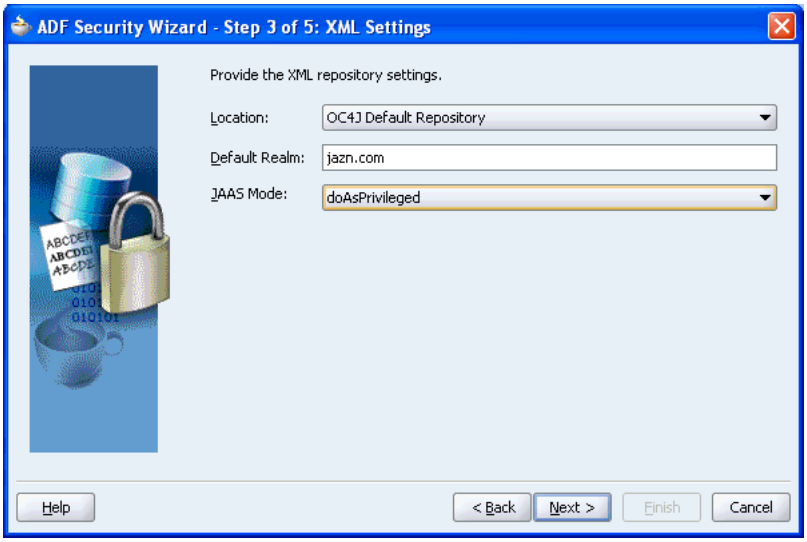


3. If needed, click **Next** to skip the Welcome page.
4. Select **Enforce Authorization** as shown in Figure 10–5, if it is not selected. This option configures the `adfAuthentication` servlet and configures authorization rules (appropriate filters to enable for checking of the current user's permissions on the page).

Figure 10–5 Authentication Page of the ADF Security Wizard

5. Click **Next** to move to the next page of the wizard.
6. Choose the appropriate JAAS provider to use with the application. Oracle ADF Security authenticates users against a resource provider. By default, this is the lightweight JAZN XML Provider which stores its realm and policy information in `system-jazn-data.xml`.
7. Click **Next** to display the next page of the wizard.
8. On this page, set the Location to **OC4J Default Repository**, Default Realm to **jazn.com**, and set JAAS Mode to **doAsPrivileged** as shown in Figure 10–6 and click **Next**.

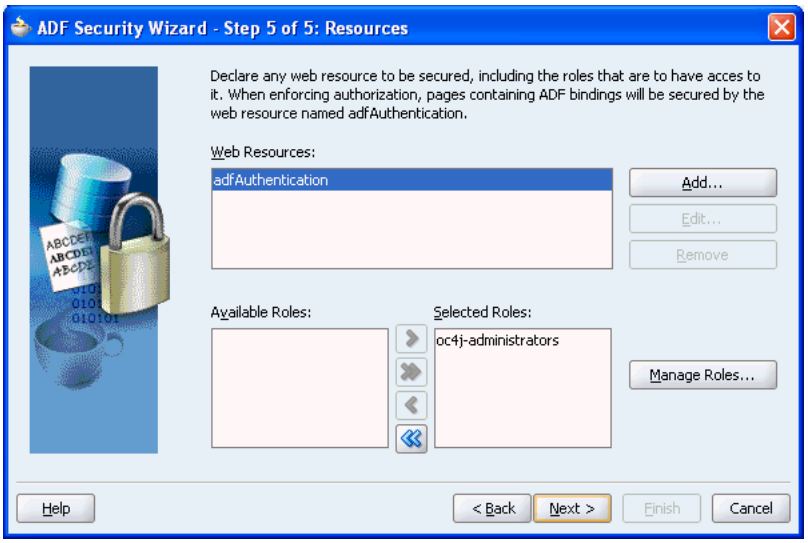
Figure 10–6 XML Settings Page of the ADF Security Wizard



9. On the Login page, choose the form of authentication that will be used by the application. This process of authentication is delegated to the container in which the application is running and will use the security manager that is defined within the application.

10. Click **Next** to display the final page of the wizard, shown in Figure 10–7.

Figure 10–7 Resources Page of the ADF Security Wizard



This page defines resources within your application that are to be secured, using standard J2EE security constraints. With the JAAS-based security model used by WebCenter applications the only web resource that must be secured using a security constraint is the `adfAuthentication` servlet.

As authentication is delegated to the container, the use of a security constraint against the adfAuthentication servlet enables for the definition of a single standard URL that can be used as a login or logout link throughout the application.

Note: J2EE container managed security defines a standard method for logon. There is no standard to log out of an application, so while the login process is delegated to the container, the logout process is handled by the adfAuthentication servlet itself.

While you may use this page to add further security constraints on web resources used by the application, you must not delete the adfAuthentication resource. Deleting it would prevent the ability to log on to the application. As every user of the application is required to be able to log on, the security constraint defined against the adfAuthentication servlet should enable all users to access this web resource. As such, the security role associated with the constraint should encompass all users.

As the only seeded role in the container is oc4j-administrators, you must create an appropriate new role for all users. For example, `ValidUsers`.

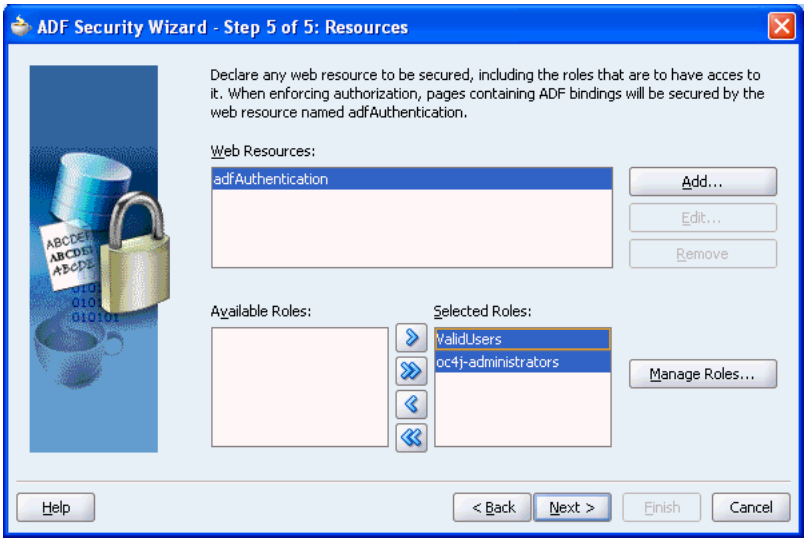
11. Click **Manage Roles**.

12. Click **Add**, and enter the name `ValidUsers`.

    Later on, you'll map this J2EE role to one of the identity store roles, `users`. This role maintains a list of every valid user. From a security perspective, allocating permissions to this role effectively defines an authenticated public resource. That is, it would be available to all users without a need for the definition of specific permissions

13. Click **OK**. The `ValidUsers` role should appear in the list.
14. Click **Close**.
15. Click the double arrow (**Add All**) to move everything in the Available Roles list to the Selected Roles list as shown in Figure 10–8. You can remove oc4j-administrators from the Selected Roles list.

Figure 10–8 Final Page of ADF Security Wizard



16. Click **Next** and then click **Finish**.

What Happens When You Use the ADF Security Wizard

Once ADF security has been enabled, all your application resources are secured by default. That means that all pages, iterators, attributes, and methods in your application are now secure and require an explicit security policy. Therefore, once the security wizard is run, you must explicitly grant the appropriate privilege (for example, view on a page), or a security violation will be raised.

By executing the ADF Security Wizard, the files that configure your application to be secured are updated. Table 10–5 shows which files are updated and the changes that are made to the files.

Table 10–5 Files updated by the ADF Security Wizard

File	Configuration Performed by the ADF Security Wizard
`web.xml`	▪ Oracle ADF authentication servlet definition ▪ Servlet mapping for enforcing security ▪ Security constraint on the authentication web resource ▪ Login configuration ▪ Required security roles
`orion-application.xml`	▪ Security provider type ▪ Default realm ▪ JAAS execution mode

Table 10–5 (Cont.) Files updated by the ADF Security Wizard

File	Configuration Performed by the ADF Security Wizard
`adf-config.xml`	■ JAAS security context ■ Security enabled for the application (The `authorizationEnforce` parameter in the `<JaasSecurityContext>` element is set to `true`)

See Appendix C, "Files for WebCenter Applications" for more information.

10.2.3 Defining Access Policies

This section covers the following topics:

- Section 10.2.3.1, "Getting Information from the Oracle ADF Security Context"
- Section 10.2.3.2, "Securing Pages in Your Application"
- Section 10.2.3.3, "Securing Iterators, Attributes, and Methods in Your Application"
- Section 10.2.3.4, "Applying Security on JCR Data Controls"
- Section 10.2.3.5, "Using Regular Expressions to Define Policies on Groups of Resources"

10.2.3.1 Getting Information from the Oracle ADF Security Context

The implementation of security in a WebCenter application is by definition an implementation of the security infrastructure of the Oracle ADF framework. As such, the security context of the framework enables access to information that will be required as you define the policies and the overall security for your application.

This section contains the following topics:

- Determining if Security is Enabled
- Determining if the User is Authenticated
- Determining the Current User Name
- Determining Membership of a J2EE Security Role

Determining if Security is Enabled

As the enforcement of Oracle ADF Security can be turned on and off at the container level independent from the application, you should determine if Oracle ADF Security is enabled prior to making permission checks. This can be achieved by evaluating the `isAuthorizationEnabled()` method of the Oracle ADF Security context as shown in Example 10–2.

Example 10–2 Using the isAuthorizationEnabled() Method of the Oracle ADF Security Context

```
if (ADFContext.getCurrent().getSecurityContext().isAuthorizationEnabled()){
  //Permission checks are performed here.
}
```

Determining if the User is Authenticated

As the user principal in a WebCenter application is never null (that is, it is either `anonymous` for unauthenticated users or the actual user name for authenticated users), it is not possible to simply check if the user principal is null, to determine if the user has logged on or not. As such, you must use a method to take into account that a

user principal of `anonymous` indicates that the user has not authenticated. This can be achieved by evaluating the `isAuthenticated()` method of the Oracle ADF Security context as shown in Example 10–3.

Example 10–3 Using the isAuthenticated() Method of the Oracle ADF Security Context

```
// ============ User's Authenticated Status =============
public boolean isAuthenticated() {
_authenticated = ADFContext.getCurrent().getSecurityContext().isAuthenticated();
    return _authenticated;
}
```

Determining the Current User Name

WebCenter applications support the concept of public pages that, while secured, are available to all users. Furthermore, components on the WebCenter pages, such as portlets, require knowledge of the current user identity. As such, the user name in a WebCenter application will never be null. If an unauthenticated user accesses the page, the user name "anonymous" will be passed to page components.

You can determine the current user's name by evaluating the `getUserName()` method of the Oracle ADF Security context as shown in Example 10–4. This method returns the string "anonymous" for all unauthenticated users and the actual authenticated user's name for authenticated users.

Example 10–4 Using the getUserName() Method of the Oracle ADF Security Context

```
// ============ Current User's Name/PrincipalName =============
     public String getCurrentUser() {
      _currentUser = ADFContext.getCurrent().getSecurityContext().getUserName();
         return _currentUser;
     }
```

Because the traditional method for determining a user name in a Faces-based application (`FacesContext.getCurrentInstance().getExternalContext().getRemoteUser()`) returns null for unauthenticated users, you need to use additional logic to handle the public user case if you use that method.

Determining Membership of a J2EE Security Role

Although WebCenter application security is centered around JAAS policies, you will likely still need to use J2EE security roles to secure components within an application page based on role membership. As WebCenter applications are JavaServer Faces-based applications, you can use the `isUserInRole(roleName)` method of the Faces external context as shown in Example 10–5 to determine if a user is in a specified role.

In this example, a convenience method (`checkIsUserInRole`) is defined. The use of this method within a managed bean enables you to expose membership of a named role as an attribute, which can then be used in EL.

Example 10–5 Using the isUserInRole(roleName)) Method of the Faces Context

```
public boolean checkIsUserInRole(String roleName){
        return
(FacesContext.getCurrentInstance().getExternalContext().isUserInRole(roleName));
}

public boolean isTechnician() {
```

```
        return (checkIsUserInRole("technicians"));
}
```

10.2.3.2 Securing Pages in Your Application

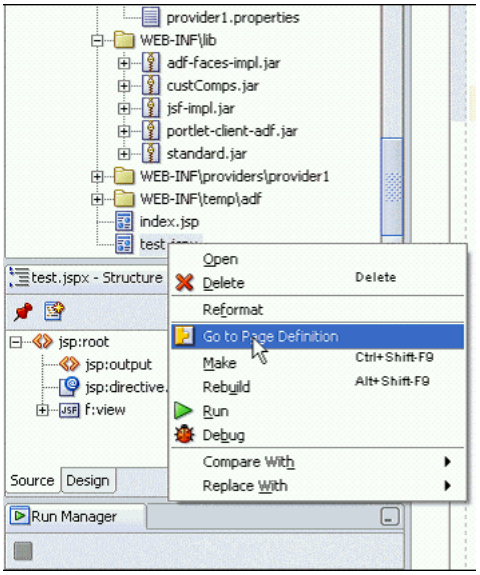
This section describes the steps involved in securing the pages in your application. You can restrict page access to the role members that you defined in the identity store and dictate the actions that role members can perform on the page. This configuration is stored in the page definition file (`<page_name>PageDef.xml`). You can access this file as shown in Figure 10–9.

> **Note:** When users run a portlet that has an Edit mode, the personalize option in the portlet menu appears only to authenticated users of the application. Anonymous or public users will not see the option to personalize the portlet through Edit mode. Hence, you must have implemented some form of security for your application in order for users to personalize their portlets. If you are a developer creating portlets and pages, then you may want to quickly test the Edit mode of your portlet without creating a complete security model for your application. See Section 10.6, "Configuring Basic Authentication for Testing Portlet Personalization" for an explanation of how you can quickly add the necessary security for testing portlet personalization.

To enable security for a page in your application, perform the following steps:

1. Right-click your `*.jspx` page in the Applications Navigator.
2. Choose **Go to Page Definition** as shown in Figure 10–9.

Figure 10–9 Go To Page Definition Option in Applications Navigator



If the Page Definition does not exist yet, then click **Yes** to create one for the selected page.

3. In the Structure pane, right-click **<*page_name*>PageDef** and choose **Edit Authorization**.

The Authorization Editor should list the identity store roles for your application and the page actions available to each role:

- **Grant** - Users may administer (grant or revoke) page permissions. This action is not used in the design time.
- **Edit** - Users may edit content displayed on the page. The Edit action is not applicable for this release.
- **Customize** - Users may modify the page. Users who are not granted this permission, will not be able to modify the page.
- **Personalize** - Users may personalize portlets on the page. For users who are not granted this permission, links or buttons that put page portlets into personalization mode are not displayed.
- **View** - Users may view the page. Users who are not granted this permission will see an authorization error when they try to access the page.

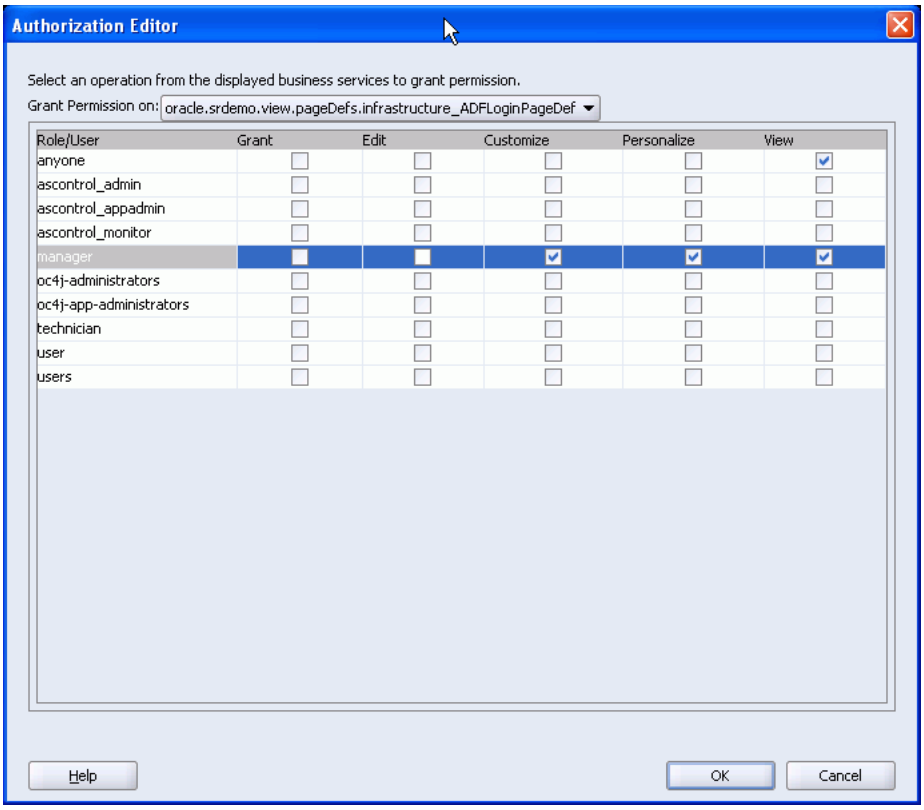
Note: To define a public page, grant View permission to the pseudo role `anyone`, which does not physically exist in the identity management solution. When a request for a WebCenter application is received from an unauthenticated user, a subject is created with the role principal `anyone` and the user principal `anonymous`.

---

4. Use the Authorization Editor to grant various permissions on the selected page.

   For example, Figure 10–10 shows that users with the `manager` role can view, personalize, and customize the selected page, while users with the `anyone` role can view the page.

Figure 10–10 Authorization Editor Used for Defining Security



5. Click **OK**. A security policy has now been defined for the page. You can repeat the same steps for other pages in your application.

Note: While there is a one-to-one relationship between the page definition file and the page you are securing, it is also possible to secure areas within a page (for example, a `ShowOneTab`) by using a headless (dummy) page definition file that represents a specific section of the page. This page definition is not actually tied to a physical page, but can still have a policy defined for it.

As such, by defining view permission on this headless page definition, you can show and hide a section of a page by referencing the headless page definition rather than the actual page definition of a target page.

If the section of the page can be secured by role membership, then the `isUserInRole` method described in "Determining Membership of a J2EE Security Role" can also be used.

10.2.3.3 Securing Iterators, Attributes, and Methods in Your Application

Certain Oracle ADF objects are "security-aware," meaning that there are predefined component-specific permissions that a developer can grant for a given resource.

Once Oracle ADF Security has been configured, all objects that access the binding layer are now secured and you must explicitly define appropriate access rights to those objects. This includes all the iterators, attributes, and methods that may be added to the page, in components such as content integration datacontrols. As such, when you define access policies on a page, you must not only define the policy for the

page itself, but also for the iterators, attributes, and methods contained within that page.

For more information about securing iterators, attributes, and methods, see the section titled "Implementing Authorization Using Oracle ADF Security" in the *Oracle Application Development Framework Developer's Guide*.

Note: To simplify the addition of a policy to all the iterators, attributes, and methods contained within your page, you can use regular expressions to define named sets of objects, enabling a single policy to cover multiple secured objects. See Section 10.2.3.5, "Using Regular Expressions to Define Policies on Groups of Resources" for more information about using regular expressions to define policies on groups of resources.

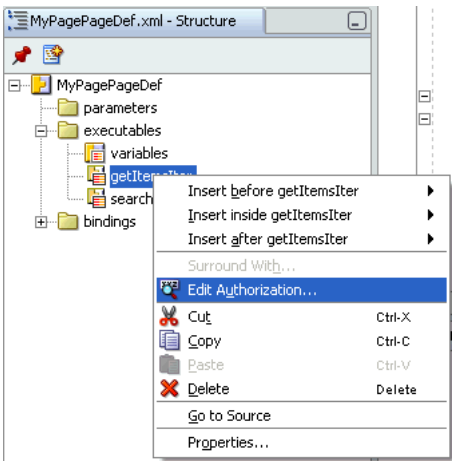
10.2.3.4 Applying Security on JCR Data Controls

This section describes the procedure for defining security on Java Content Repository (JCR) data controls that are used for adding content to your page at design time. For information about data controls, see Section 5.3, "Using JCR Data Controls: Examples". To enable security for new method iterators, method action bindings, and attribute bindings of a data control that you created in the page definition, perform the following steps:

> **See Also:** Section 10.1.2.1, "Oracle ADF Permissions" for information about permissions.

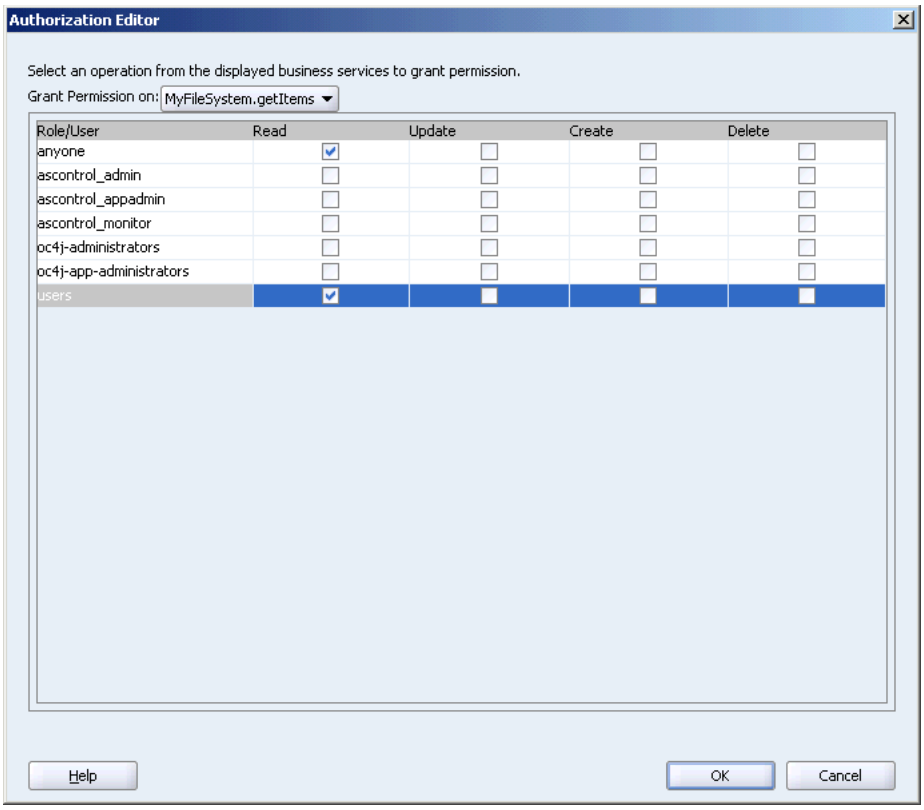
1. In the Application Navigator, right-click your JSPX page and select **Go to Page Definition**. The page definition is displayed.
2. To grant permission on method iterators, expand the executables node in the Structure pane, right-click a method iterator, and select **Edit Authorization**, as shown in Figure 10–11.

Figure 10–11 Authorization Option for a Method Iterator



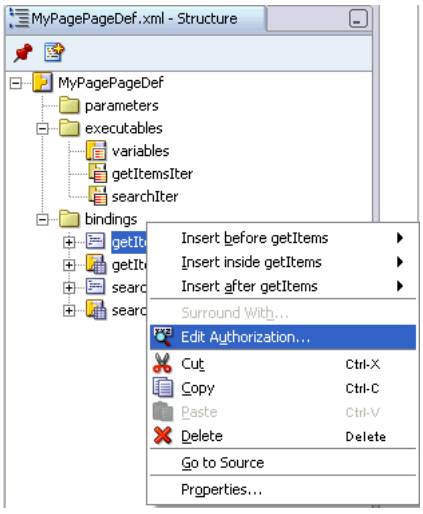
3. In the Authorization Editor dialog box, set only the `Read` permission, as shown in Figure 10–12.

Figure 10–12 Authorization Editor for a Method Iterator



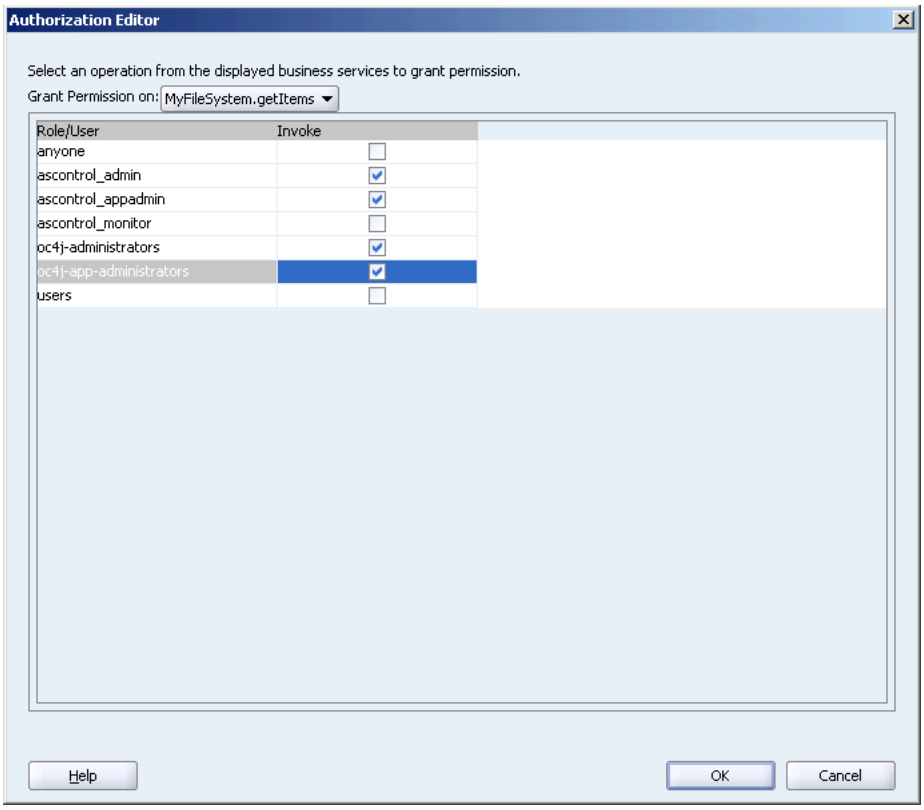
4. To grant permission on method action bindings, expand the bindings node in the Structure pane, right-click a method action binding, and select **Edit Authorization**, as shown in Figure 10–13.

Figure 10–13 Edit Authorization Option for a Method Action Binding



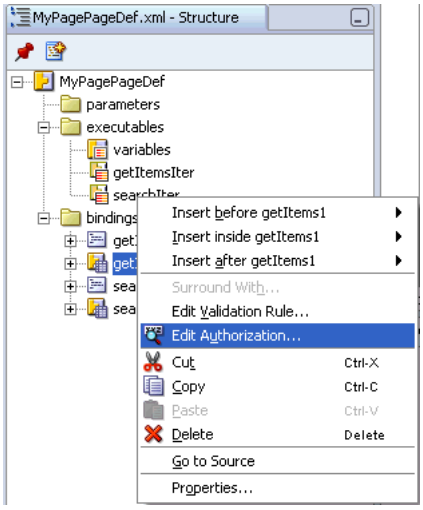
5. In the Authorization Editor dialog box, set only the `Invoke` permission, as shown in Figure 10–14.

Figure 10–14 Authorization Editor for a Method Action Binding



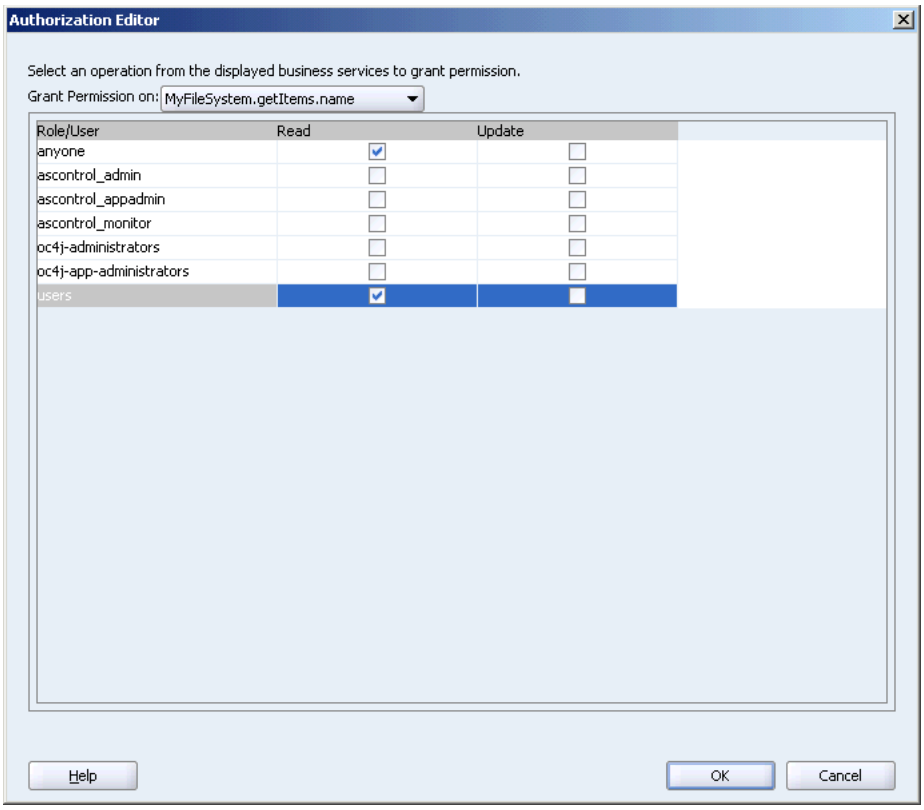
6. Click **OK**.
7. To grant permission on attribute bindings, expand the bindings node in the Structure pane, right-click an attribute binding, and select **Edit Authorization**, as shown in Figure 10–15.

Figure 10–15 Authorization Option for an Attribute Binding



8. In the Authorization Editor dialog box, set only `Read` permissions, as shown in Figure 10–16.

Figure 10–16 Authorization Editor for an Attribute Binding



9. To set permissions for other available operations, select the operations from the **Grant Permission on** list.

10. Click **OK**.

10.2.3.5 Using Regular Expressions to Define Policies on Groups of Resources

Once Oracle ADF security has been enabled, all your application resources are secured by default. That means that all pages, iterators, attributes, and methods in your application are now secure and require an explicit security policy. Therefore, once the ADF Security Wizard is run, you must explicitly grant the appropriate privilege (for example, View on a page), or a security violation will be raised.

Consider Example 10–6, in which each method name starts with `method_`.

Example 10–6 Method Permission Defined in the system-jazn-data.xml File

```
<principal>
 <class> oracle.adf.share.security.authentication.ADFRolePrincipal
</class>
 <name>anyone</name>
</principal>

<permission>
   <class oracle.adf.share.security.authorization.MethodPermission </class>
   <name>method_1</name>
   <actions>invoke</actions>
</permission>

<permission>
   <class> oracle.adf.share.security.authorization.MethodPermission </class>
```

```
   <name>method_2</name>
   <actions>invoke</actions>
</permission>

<permission>
   <class> oracle.adf.share.security.authorization.MethodPermission </class>
   <name>method_3</name>
   <actions>invoke</actions>
</permission>

<permission>
   <class> oracle.adf.share.security.authorization.MethodPermission </class>
   <name>method_4</name>
   <actions>invoke</actions>
</permission>
...
```

As there are potentially many more individual methods for which the *anyone* role must be granted the invoke privilege, you can greatly simplify the policy definition by replacing the name with a regular expression that represents the set of methods to which `anyone` is granted the invoke permission.

For example, the previous listing of permissions could be replaced with a single permission, as shown in Example 10–7.

Example 10–7 Using Regular Expressions to Define Permission for Methods

```
<principal>
 <class> oracle.adf.share.security.authentication.ADFRolePrincipal
</class>
 <name>anyone</name>
</principal>

<permission>
   <class oracle.adf.share.security.authorization.MethodPermission </class>
   <name>method_*</name>
   <actions>invoke</actions>
</permission>
```

As the Authorization Editor does not support the use of regular expressions in the user interface, you must edit the policy directly in the policy store.

Caution: If the policy is updated manually, consider the following important points:

- You must propagate the manual changes made in the policy store to the `app-jazn-data.xml` file before you deploy the application. See Section 12.2.4.2, "Updating Policy Information (Optional)" for the steps to be performed.
- You must restart Oracle Containers for J2EE (OC4J) each time your policy has been modified.

The use of more complex regular expressions enables you to define business rules in the policy, thus creating a very targeted set of permissions. For example, you can grant the invoke permission on all methods and deny specific methods at the same time by defining a subtraction set in your regular expression. Example 10–8 shows how the

invoke permission is granted to the `anyone` role for all methods except those where the method name starts with delete.

Example 10–8 Granting the Invoke Permission to the anyone Role for Specific Methods

```
<principal>
   <class>oracle.adf.share.security.authentication.ADFRolePrincipal</class>
   <name>anyone</name>
</principal>
…
<permission>
   <class>oracle.adf.share.security.authorization.MethodPermission</class>
   <name>[^(delete)].*</name>
   <actions>invoke</actions>
</permission>
```

Table 10–6 shows some of the basic regular expression metacharacters that you can use in your policy definitions.

Table 10–6 Description of Metacharacters

Metacharacter	Description
[abc]	a, b, or c (included in list)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a to z or A to Z, inclusive (range)
[a-d[m-p]]	a to d, or m to p ~= [a-dm-p](union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, without b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p

10.2.4 Enforcing Security Policies in Your Application

While the existence of a policy will prevent unauthorized users from accessing a secured resource, trying to access the resource would result in a security exception. A good security practice dictates that a user should not be aware of resources and capabilities to which they do not have access.

For example, if a user does not have permission to view an administrative page, then all navigation components that point to that page should be dynamically removed for that user.

While the application must first evaluate the policy to determine whether the user has the appropriate permission, ultimately the ability to attempt access to a secured resource or function (such as a delete button) is controlled by the User Interface (UI) component's `Rendered` property.

By default the Rendered property is set to `true`. By dynamically changing this value based on the permission, the UI component can be shown or hidden. For example, if the user has the appropriate permission, the `Rendered` property should be set to `true` so that the UI component is shown. If they do not have permission, the property should be set to `false` and the UI component is hidden from view.

Note: The ability to evaluate a policy is limited to the current request. Therefore, it is important to understand where the policy evaluation occurrs, because evaluating the policy at anything other than the request scope can lead to unexpected results.

The following sections discuss evaluating policies using EL and Java. EL enables you to evaluate the policy directly in the UI, while the use of Java enables you to evaluate the policy from within a managed bean.

This section covers the following topics:

- Section 10.2.4.1, "Evaluating Policies Using Expression Language (EL)"
- Section 10.2.4.2, "Evaluating Policies Using Java"

10.2.4.1 Evaluating Policies Using Expression Language (EL)

The use of EL within a UI element enables for properties to be defined dynamically, resulting in modification of the UI component at run time. In the case of securing resources, the UI property of interest is the rendered property, which enables the developer to show and hide components based on available permissions.

To evaluate a policy using EL, you must use the `permissioninfo` method in the binding layer (`#{bindings...}`) that relates to the associated permission type (page, method, attribute, and iterator). The `permissioninfo` method returns true or false, based on whether the user has permission to perform the specified action.

In the case of pages, the target page definition is passed as an argument to the `permissioninfo` method. For all other permission types, the secured object is explicitly named under the bindings element.

The following tables show the EL that is required to determine if a user has the associated permission. If the user has the appropriate permission, the EL expression evaluates to true, otherwise it returns `false`.

Table 10–7 EL to Determine Action Permission on Pages

Privileged Action	Required EL
View	`#{bindings.permissionInfo['MyPagePageDef'].allowsView}`
Personalize	`#{bindings.permissionInfo['MyPagePageDef'].allowsPersonalize}`
Customize	`#{bindings.permissionInfo['MyPagePageDef'].allowsCustomize}`
Edit	`#{bindings.permissionInfo['MyPagePageDef'].allowsEdit}` (not currently supported)
Grant	`#{bindings.permissionInfo['MyPagePageDef'].allowsGrant}` (not currently supported)

Note: In the case of page permission, the value of the page definition can be specified dynamically by using late-binding EL within a managed bean.

Table 10–8 EL to Determine Action Permission on Methods

Privileged Action	Required EL
Invoke	`#{bindings.myMethod.permissionInfo.allowsInvoke}`

Table 10–9 EL to Determine Action Permission on Iterators

Privileged Action	Required EL
Read	`#{bindings.MyIterator.permissionInfo.allowsRead}`
Create	`#{bindings.MyIterator.permissionInfo.allowsCreate}`
Update	`#{bindings.MyIterator.permissionInfo.allowsUpdate}`
Delete	`#{bindings.MyIterator.permissionInfo.allowsDelete}`

Table 10–10 EL to Determine Action Permission on Attributes

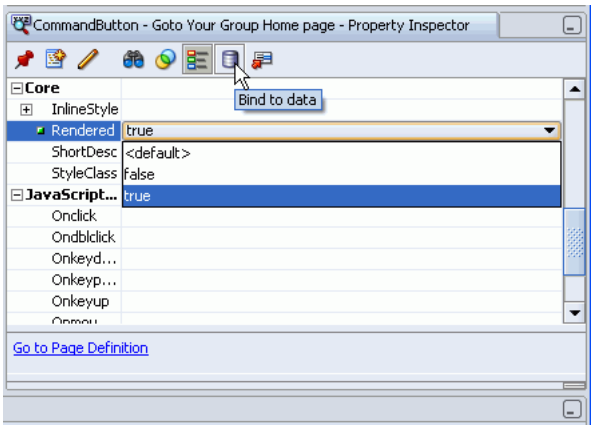
Privileged Action	Required EL
Read	`#{bindings.myAttribute.permissionInfo.allowsRead}`
Update	`#{bindings.myAttribute.permissionInfo.allowsCreate}`

Note: The `permissionInfo` attribute of the ADF Bindings layer is not exposed in the Variables Navigator of the Bind to Data dialog box (the bindings editor). As such, the EL expressions that are defined in Table 10–7 to Table 10–10 must be entered manually into the property field or in the Expression field of the Bind to Data dialog box.

To associate the rendering of a navigation component to a user's granted permissions on a target page, perform the following steps:

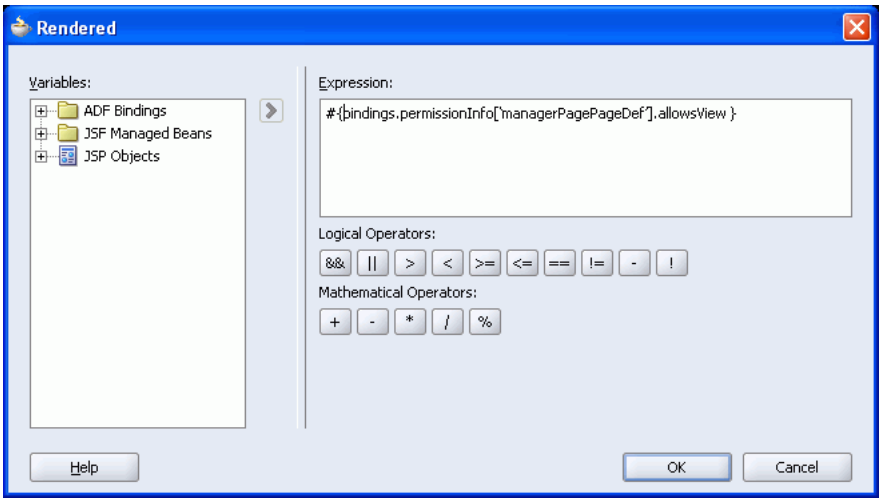
1. Open the page that contains the navigation component in Design view.
2. Select the component that is used to navigate to the secured page.
3. In the Property Inspector, select the **Rendered** property.
4. Click the **Bind to Data** icon as shown in Figure 10–17.

Figure 10–17 Binding the Rendered Property to Data



5. Enter the EL expression, `#{bindings.permissionInfo['managerPagePageDef'].allowsView}` as shown in Figure 10–18. In this example, `managerPage` is the secured target page.

Figure 10–18 Defining EL in the Bind to Data Dialog Box



6. Click **OK**.

7. Run the application. The component will be rendered or hidden based on the user's ability to view the target page.

Delayed Evaluation of EL for Session-Scoped Managed Beans

The ability to evaluate a security permission is scoped to the request. If you want to evaluate permissions to access a target page from a managed bean that is scoped to a higher level than Request (for example, a global menu that is backed by a session-scoped managed bean), you must implement delayed EL evaluation (late-binding). By passing in the target page as a managed property of the bean you ensure that the EL is evaluated only after the required binding information is available to the session-scoped bean. As EL is evaluated immediately when the page is executed, placing the EL expression directly in the properties of a UI component, backed by a session-scoped bean, would result in an out-of-scope error.

Example 10–9 shows a property (`Authorized`) of a session-scoped bean that returns true or false based on a user's ability to view a named target page. In this case, the _

`targetPageDef` variable is a managed property containing the name of the target page. Within the UI, the EL would reference the `authorized` property rather than `bindings.permissionInfo`.

Example 10–9 Delayed EL Evaluation in a Session-Scoped Managed Bean

```
public boolean isAuthorized()
{
 if (_targetPageDef != null) {
  FacesContext ctx = FacesContext.getCurrentInstance();
  ValueBinding   vb = ctx.getApplication().createValueBinding (
  "#{bindings.permissionInfo['" + _targetPageDef + "'].allowsView}" );
  if (vb != null) {
    Object authResult = vb.getValue(ctx);
    return (Boolean) authResult;
    }
  else  {
    ctx.addMessage(null, new FacesMessage (
    FacesMessage.SEVERITY_WARN, "Access Permission not defined! " , null));
    return(true);
    }
}
```

10.2.4.2 Evaluating Policies Using Java

To evaluate the security policies from within Java, you can use the `hasPermission` method of the Oracle ADF Security context. This method takes a permission object (defined by the resource and action combination) and returns true if the user has the corresponding permission.

In Example 10–10, a convenience function is defined to enable you to pass in the name of the page and the desired action, returning true or false based on the user's permissions. As this convenience function is checking page permissions, the RegionPermission class is used to define the permission object that is passed to the `hasPermission` method.

Example 10–10 Using the hasPermission Method to Evaluate Access Policies

```
private boolean TestPermission (String PageName, String Action)  {
  Permission p = new RegionPermission("view.pageDefs." + PageName + "PageDef",
Action);
  if (p != null) {
     return ADFContext.getCurrent().getSecurityContext().hasPermission(p);
 }
 else {
     return (true);
 }
```

As it is possible to determine the user's permission for a target page from within a backing bean, you can now use this convenience method to dynamically alter the result of a Faces navigation action. In Example 10–11, you can see that a single command button can point to different target pages depending on the user's permission. By checking the View permission from the most secured page (the manager page) to the least secured page (the public welcome page), the command button will direct the user to the page that corresponds to their permission level by applying the appropriate action. The backing bean that returns the appropriate action is using the convenience method defined in Example 10–10.

Example 10–11

```
//CommandButton Definition
<af:commandButton text="Goto Your Group Home page"
  binding="#{backing_content.commandButton1}"
  id="commandButton1"

  action="#{backing_content.getSecureNavigationAction}"/>

//Backing Bean Code
    public String getSecureNavigationAction() {
      String ActionName;
      if (TestPermission("ManagerPage", "view"))
        ActionName = "goToManagerPage";
      else if (TestPermission("EmployeePage", "view"))
        ActionName = "goToEmployeePage";
      else
        ActionName = "goToWelcomePage";
      return (ActionName);
    }
```

10.2.5 Configuring Deployment Descriptor Files with Security Information

As WebCenter applications are pure J2EE applications, even though the ADF Security Wizard has configured to implement JAAS-based security, it is still necessary to define the role mapping for any J2EE security roles that are used by the application to the corresponding roles in the identity management solution. For example, you must map the J2EE security role `ValidUsers` (the role that was defined in the previous section for the ADFAuthentication servlet) to the corresponding role that encompasses all users. You define role mapping in the `orion-web.xml` file. When you run an application in the embedded OC4J in Oracle JDeveloper, the application is run in place and it is not actually deployed to the container. As such, there are some additional deployment descriptor requirements for running the secured application directly inside Oracle JDeveloper's embedded OC4J.

This section contains the following topics:

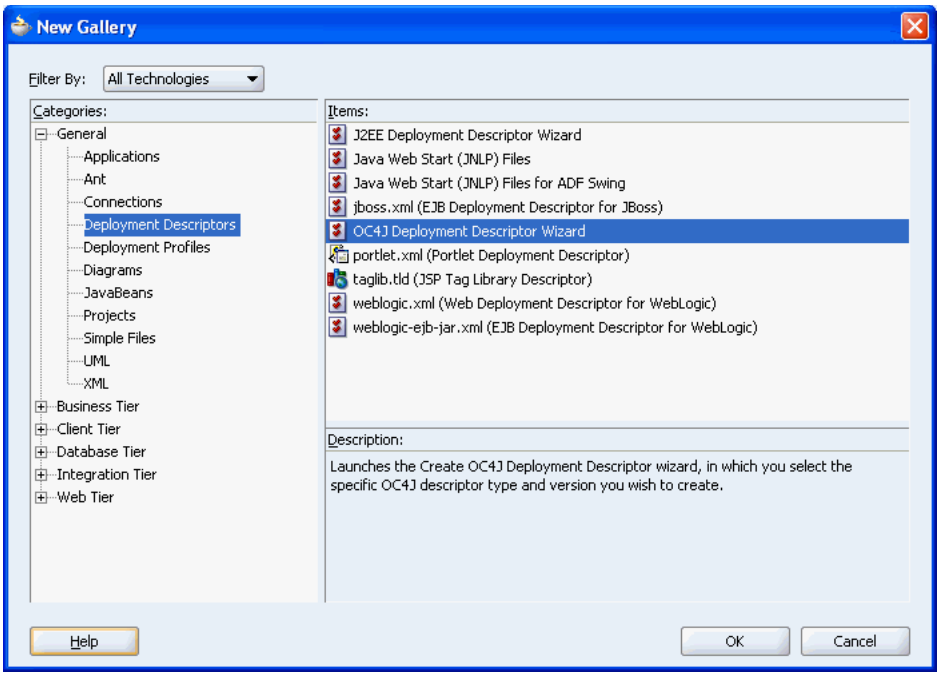
- Section 10.2.5.1, "Creating the Deployment Descriptor File"
- Section 10.2.5.2, "Configuring Security Role Mappings"
- Section 10.2.5.3, "Additional Requirement for Running the Application in Oracle JDeveloper's Embedded OC4J"

10.2.5.1 Creating the Deployment Descriptor File

To create the `orion-web.xml` deployment descriptior file, perform the following steps:

1. In the Applications Navigator, right-click **ViewController** and select **New**.
2. Select **All Technologies** from the list at the top.
3. In the panel on the left, expand **General** and then select **Deployment Descriptors**. The panel on the right lists the different types of deployment descriptors that are available, as shown in Figure 10–19.

Figure 10–19 Deployment Descriptor Selection



4. Select **OC4J Deployment Descriptor Wizard** and then click **OK**.

   This starts the OC4J Deployment Descriptor Wizard.

5. Click **Next** to skip the Welcome page.

6. Select **orion-web.xml** and then click **Next**.

   The file name will be grayed out if the file already exists.

7. Select **10.0** and then click **Finish**.

   You will now see `orion-web.xml` in the Applications Navigator, under View Controller, Web Content, WEB-INF.

10.2.5.2 Configuring Security Role Mappings

To map the J2EE security roles that you defined within your application to the identity store roles, perform the steps in this section.

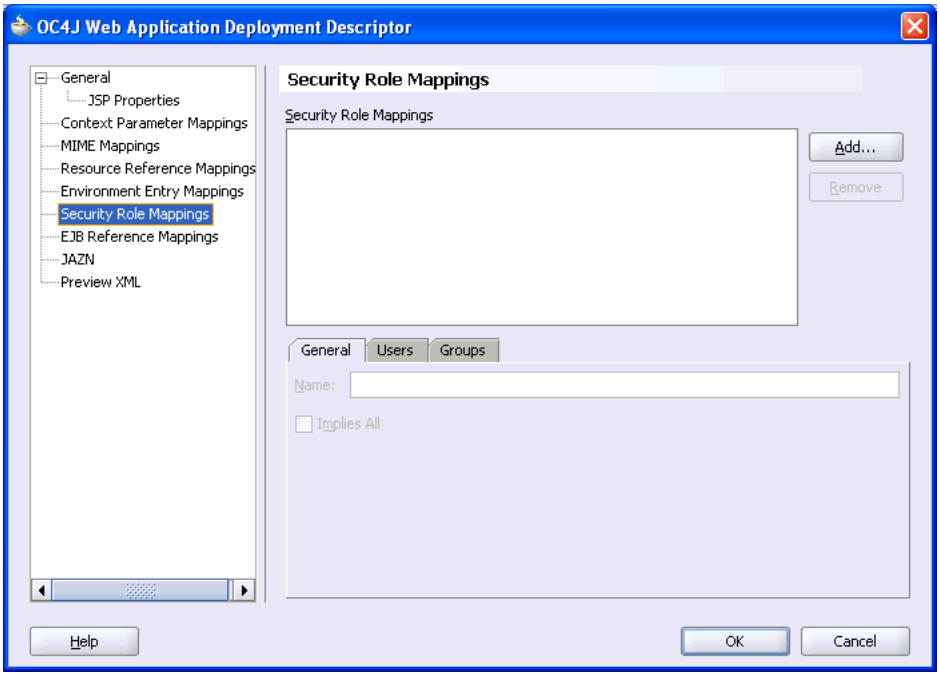
> **Note:** At a minimum, you must define a mapping for the J2EE security role that is associated with the security constraint protecting the ADFAuthentication servlet.

This section uses an example where the J2EE Security role `ValidUsers` is mapped to the identity store role `users`. Configure security role mappings in `orion-web.xml` as follows:

1. Right-click **orion-web.xml** and select **Properties** to set some additional deployment options.

2. Select **Security Role Mappings** in the panel on the left. This displays a panel on the right where you can add these security role mapping as shown in Figure 10–20.

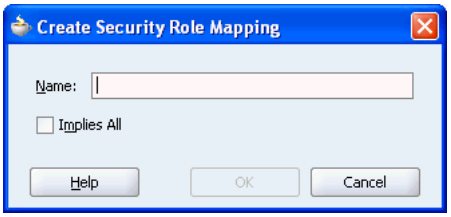
Figure 10–20 OC4J Web Application Deployment Descriptor Dialog Box



3. Create the security role mappings as follows:

   a. Click **Add**. This displays a window, as shown in Figure 10–21.

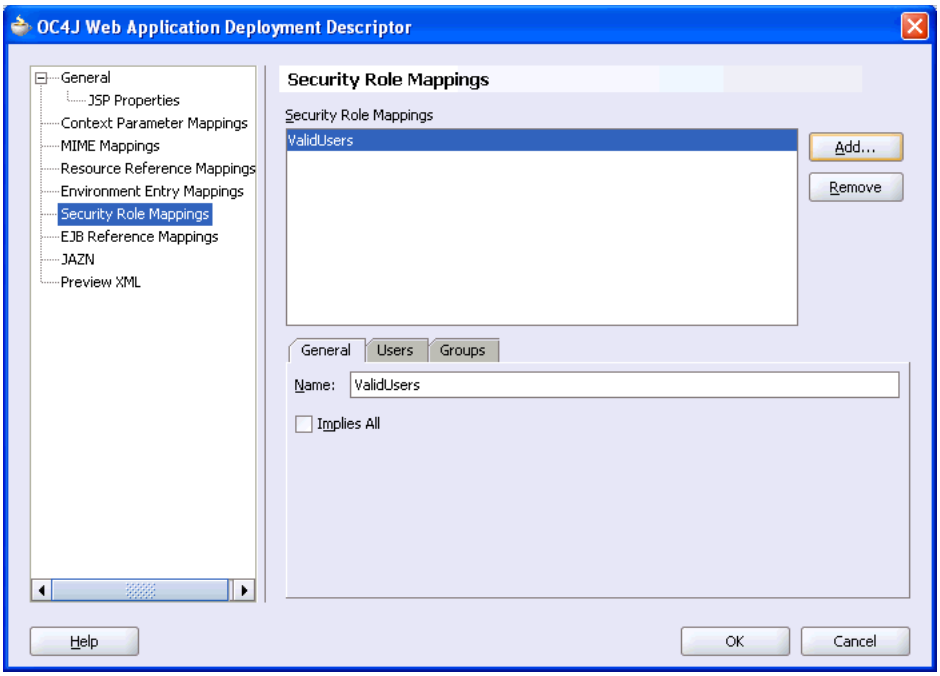
Figure 10–21 Create Security Role Mapping Dialog Box



   b. For Name, enter the J2EE security role name, `ValidUsers`.

   c. Click **OK**. The role name that you just entered is displayed in the mappings panel and also on the General tab, as shown in Figure 10–22. To edit the role name use the Name property on the General tab.

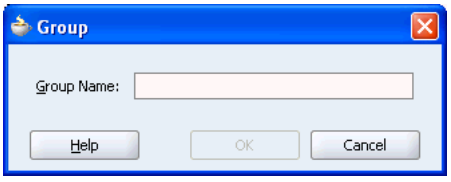
Figure 10–22 ValidUsers Role in the Security Role Mappings Section



d. Click the **Groups** tab. Note that the J2EE security role `ValidUsers` is highlighted in the mappings panel. This means that you are about to map a group of users to this J2EE security role.

e. Click the Add button that is to the right of the Group Names panel. The Group dialog box is displayed as shown in Figure 10–23.

Figure 10–23 Group Dialog Box

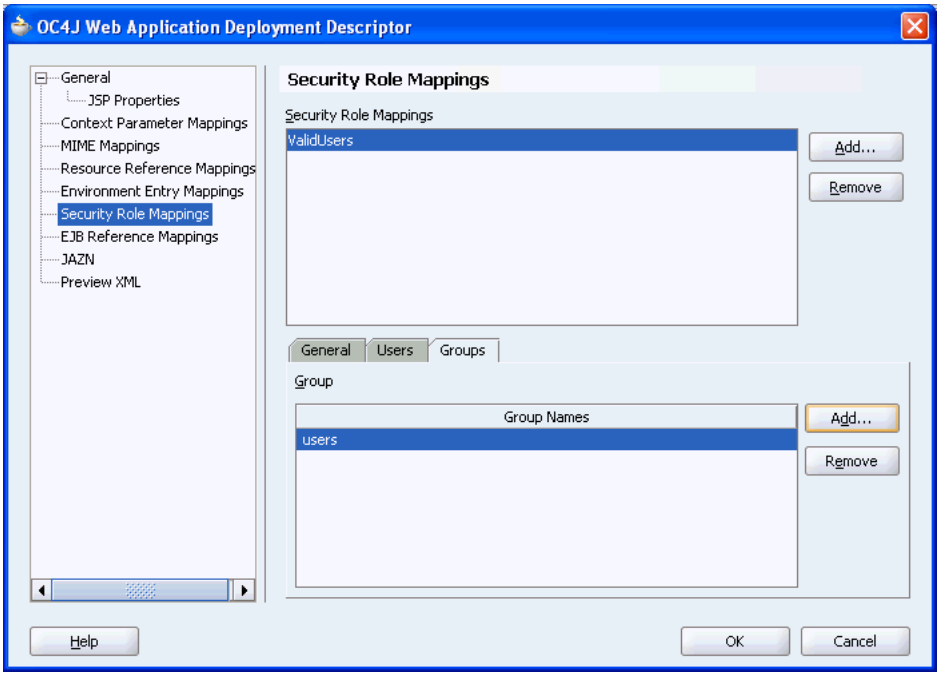


f. For Group Name, enter `users`.

g. Click **OK**.

In this step you mapped the J2EE security role `ValidUsers` to the identity store role `users`. This mapping is shown in Figure 10–24.

Figure 10–24 ValidUsers Role Mapped to users Group



4. Click **OK** to save changes to the OC4J deployment descriptor.

   If you examine the source code for the configuration file `orion-web.xml`, then you should see the security-role-mapping entry as follows:

```
<security-role-mapping name="ValidUsers" impliesAll="false">
   <group name="users"></group>
</security-role-mapping>
```

This completes the OC4J Web application deployment descriptor configuration requirements for deployment to a remote application server. To run your application in the embedded OC4J, the extra configuration described in the next section must be performed.

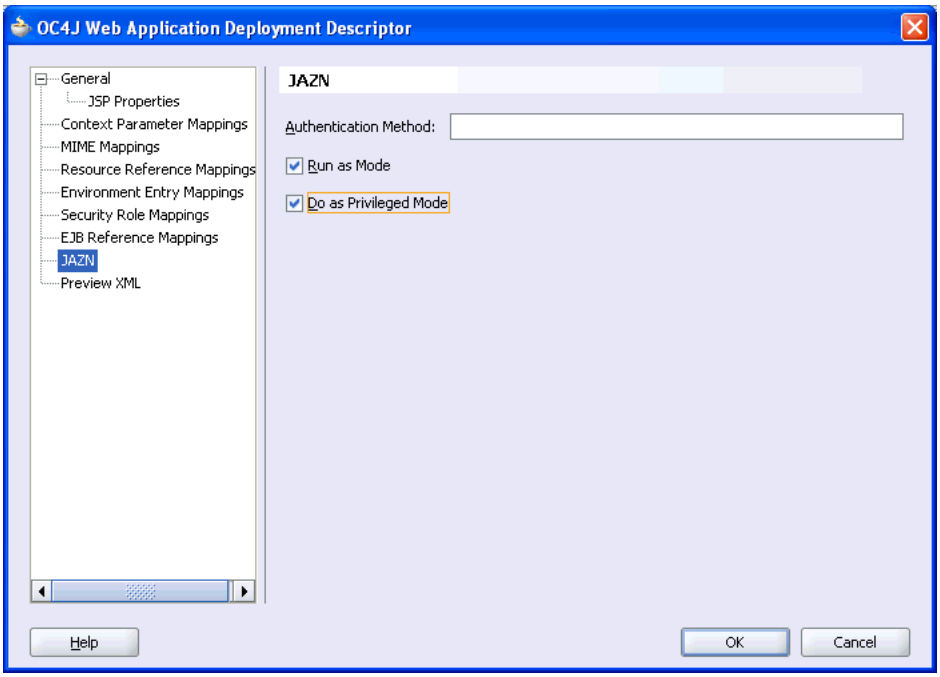
10.2.5.3 Additional Requirement for Running the Application in Oracle JDeveloper's Embedded OC4J

While deploying a WebCenter application to a remote application server requires you to run the Predeployment tool as well as the JAZN Migration tool, Oracle JDeveloper's embedded OC4J enables you to run the application directly. When the application is deployed to a remote application server (or standalone OC4J), the required JAAS mode is determined from the `orion-application.xml` deployment descriptor file, which is configured by the ADF Security Wizard. However, when running the application in Oracle JDeveloper's embedded OC4J, you must also specify the JAAS information in the `orion-web.xml` deployment descriptor file. If you do not add this information to the `orion-web.xml` file, then the security will be enforced on a deployed server, but it will not be reflected when you run it in the embedded OC4J. To to enforce security when running in the embedded OC4J, perform the following steps:

1. Right-click `orion-web.xml` and select **Properties** to set some additional deployment options.
2. Select JAZN in the panel on the left. This displays several JAAS authentication options.

3. Select **Run as Mode** and **Do as Privileged Mode**, as shown in Figure 10–25.

Figure 10–25 Authentication Options Selection



The Run as Mode option indicates that your servlet has special privileges and the Do as Privileged Mode option specifies the privileges that are enabled.

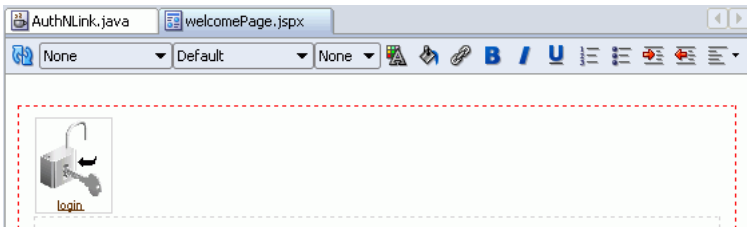
Setting both options adds the following line to `orion-web.xml`:

```
<jazn-web-app runas-mode="true" doasprivileged-mode="true"/>
```

10.3 Creating a Login Component for Your Application

In this section you will create a standard login component that can be added to any page in your application to enable users to authenticate or subsequently log off. This component keeps track of the authenticated state of the user and returns the appropriate login or logout URLs and icons. Furthermore, it keeps track of the name of the current user (anonymous or the name of the logged in user). Hence, using this login component enables the developer a single, consistent object. Figure 10–26 shows a login icon added to the global menu facet of an Oracle ADF application page.

Figure 10–26 Login Icon on the Page



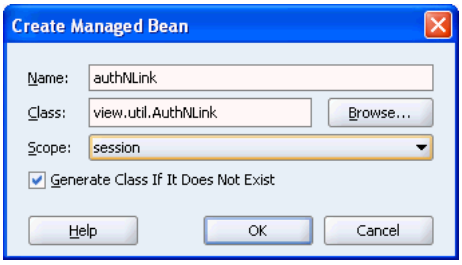
The login component will redirect the users back to the current page once they are authenticated.

> **Note:** You may want to alter the component's code to redirect to a welcome page if the current page is not publicly accessible.

To create a login component, perform the following steps:

1. Open your application's ViewController project.
2. In the Applications Navigator, expand the **WEB-INF** node and open the `faces-config.xml` file.
3. In the Structure pane, select the **Overview** tab.
4. Right-click **Managed Beans** and select **Insert managed-bean**. The Create Managed Bean dialog box is displayed.
5. Specify `authNLink` for the name, `view.util.AuthNLink` for the class, and set the scope to **session**, as shown in Figure 10–27. Select **Generate Class If It Does Not Exist**, if it is not already selected.

Figure 10–27 Create Managed Bean Dialog Box

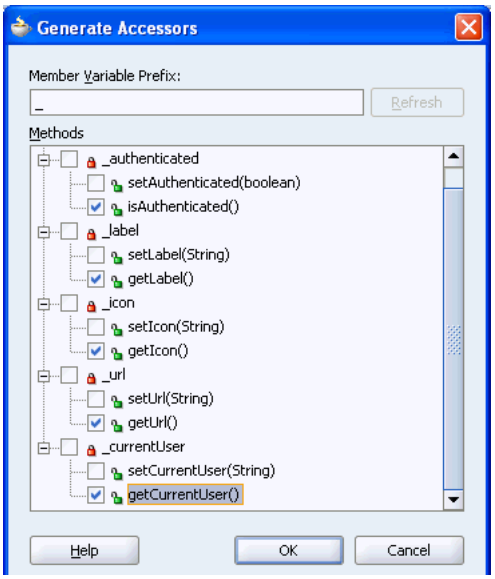


6. Click **OK** and open the `view.util.AuthNLink.java` file under the application sources.
7. Add the following private variables to the managed bean definition as shown in bold in the following example:

```
public class AuthNLink {
    private boolean _authenticated = false;
    private String _label = null;
    private String _icon = null;
    private String _url = null;
    private String _currentUser = null;
```

8. From the Source menu, select **Generate Accessors**.
9. Expand each node and check the getter methods (the ones that start with `is` and `get`) as shown in Figure 10–28.

Figure 10–28 Generate Accessors Dialog Box



10. Define the methods, as shown in Example 10–12.

> **Note:** Import the `ADFContext` and `FacesContext` when prompted, by pressing ALT-Enter with the cursor over the appropriate line.

This example has fixed English labels. To internationalize the code, you can define these strings in a resource bundle. See the section titled "Internationalizing Your Application" in *Oracle Application Development Framework Developer's Guide* for more information about internationalization.

Example 10–12 Login Component Code

```
// ============ User's Authenticated Status =============
    public boolean isAuthenticated() {
     _authenticated =
ADFContext.getCurrent().getSecurityContext().isAuthenticated();
    return _authenticated;
    }
    // ================== Link Label ========================
     public String getLabel()
     {
      // toggle link text based on authenticated state of the user.
      if (isAuthenticated())
         { _label = "Click here to log in"; }
       else
         { _label = " Click here to log out";  }
        return _label;
     }
    // ================== Link Icon ========================
    public String getIcon() {
        // toggle icon based on authenticated state of the user.
        if (isAuthenticated())
            _icon = "logout.gif";
        else
            _icon = "login.gif";
```

```
    return (_icon);
}
// ================== Link URL ======================
 public String getUrl()
 {
   String currentPage = null;
   String urlBaseRef  = null;
   String urlBaseRef2 = null;
   FacesContext fctx = FacesContext.getCurrentInstance();
        currentPage = "/faces" + fctx.getViewRoot().getViewId();
                if (isAuthenticated())
    _url = "/adfAuthentication?logout=true&end_url=" + currentPage;
   else
    _url = "/adfAuthentication?success_url=" + currentPage;
   return (_url);
 }
// ============ Current User's Name/PrincipalName =============
 public String getCurrentUser() {
  _currentUser = ADFContext.getCurrent().getSecurityContext().getUserName();
     return _currentUser;
 }
```

In this code, the component uses the `isAuthenticated` method of the Oracle ADF Security Context to determine if the user is currently authenticated and modifies the link text, the link text URL, and the associated icon accordingly.

11. Copy your login and logout image files (GIF, JPG, or PNG files) to the `public_html` directory of your project.

> **Note:** The images used should reference the appropriate skin image if your application uses skins. See Chapter 9, "Defining and Applying Styles to Core Customizable Components" for more information about skins.

Adding the Login Component to a Page

To add a login component to a page, perform the following steps:

1. Create an Oracle ADF Faces page. See Section 4.2, "Building WebCenter Application-Enabled Pages in Oracle JDeveloper with Oracle ADF" for more information.
2. Open the page in Design view.
3. From the Component Palette, select **ADF Faces Core**.
4. Select a `menuButtons` component and drag it onto the page.

> **Note:** If you are using an Oracle ADF `PanelPage` component to lay out the page, you should place the global navigation items such as the login link in the `menuGlobal` facet. This will place the link in a consistent location on the page (by default, this is the top-right corner of the page).

5. Select a `goMenuItem` and drag it onto the `MenuButtons` component.
6. In the Property Inspector, set the `Text`, `Destination`, and `Icon` properties of the `goMenuItem` to the values provided in the following table:

Property	Value
Text	#{authNLink.label}
Destination	#{authNLink.url}
Icon	#{authNLink.icon}

To set these properties, navigate to the `authNLink` node under JSF Managed Beans in the Binding to Data dialog box and set the values provided in the table. You can access the Bind to Data dialog box in either of the following ways:

- Right-click **goMenuItem** in the Structure pane and click **Properties**. In the Properties dialog box, click the **Bind to Data** icon next to the property.
- In the Property Inspector, click the **Bind to Data** icon in the filed next to the property.

Figure 10–29 and Figure 10–30 show the settings for the `Text` and `Destination` properties in the Bind to Data dialog box.

Figure 10–29 Bind to Data Dialog Box for the Text Property

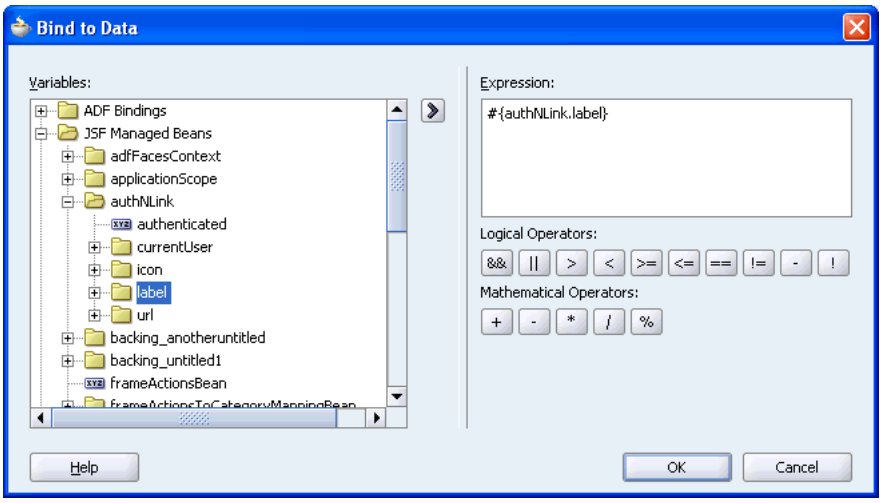
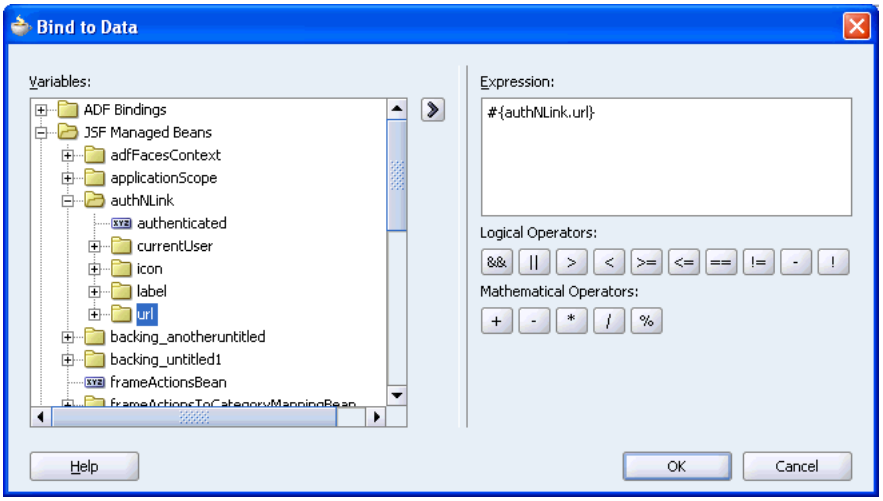
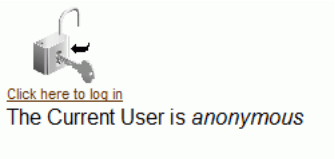


Figure 10–30 Bind to Data Dialog Box for the Destination Property



7. As the login component keeps track of the current user, you can also display the user's name on your page. To do this, from the Component Palette, select **ADF Faces Core** and drag an **OutputFormatted** component onto the page.
8. Set the value of the `OutputFormatted` to **The Current User is #{authNLink.currentUser}**.
9. Save the page and run it. It will look similar to Figure 10–31.

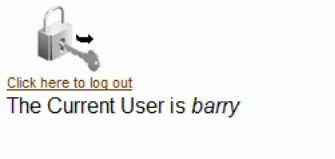
Figure 10–31 Page with a Log In Link



> **Note:** To enforce security in your application, you must first perform the steps described in Section 10.2, "Setting Up Security for Your Application". You must, at a minimum, grant view privileges to the *anyone* role.

10. Click **Log in** and log in as a user with the appropriate credentials. Once logged in, the page will look similar to Figure 10–32.

Figure 10–32 Page with a Log Out Link



10.4 Creating a Login Page for Your Application

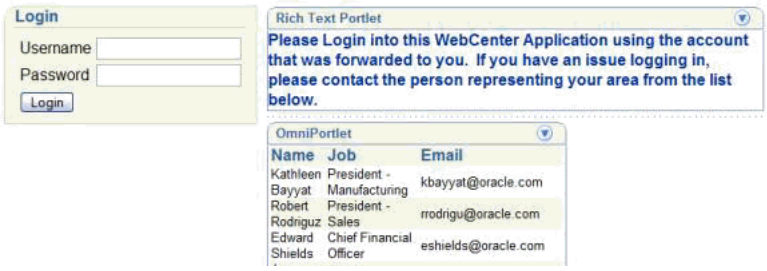
In this section you will create the new login page that users are redirected to for authentication. WebCenter applications typically have a notion of public pages and enable for explicit as well as implicit authentication. This means that users can log in to the application by clicking the login link before they navigate to secured content (explicit), or they can navigate to a secured page, which will redirect them to the login page for the application (implicit). See Section 10.1, "Introduction to WebCenter Application Security" for more information about implicit and explicit authentication. Figure 10–33 shows a sample login page you are going to build in this chapter. The addition of portlets to the login page enables the login page itself to be indistinguishable from the other pages in your WebCenter application.

Note: This section discusses creating an Oracle ADF Faces-based login page that enables you to include customizable components and portlets. However, if adding these components is not a requirement, then a simple JSP or HTML login page can be also used.

Container-based authentication relies on the `j_SecurityCheck` method within the container's security model. Both the Oracle ADF Faces-based login page and the simplified login pages use this method to enforce authentication.

See the section titled "Step 1: Creating a Login Page" in the *Oracle WebCenter Framework Tutorial* for a detailed discussion on how to build a simple login page.

Figure 10–33 Login Page



Creating a Login page for your application involves the following tasks:

- Section 10.4.1, "Creating an Oracle ADF Faces-Based Login Page"
- Section 10.4.2, "Adding Login Code to the Backing Bean"
- Section 10.4.3, "Adding Portlets to the Login Page"
- Section 10.4.4, "Configuring the web.xml File for an Oracle ADF Faces-Based Login Page"
- Section 10.4.5, "Editing Authorization for the Login Page"

10.4.1 Creating an Oracle ADF Faces-Based Login Page

To create the Oracle ADF Faces-based login page, perform the following steps:

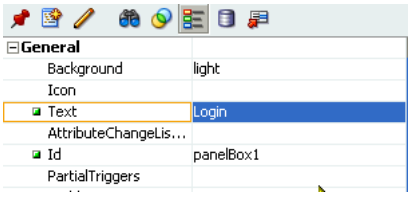
1. In the Applications Navigator, under the ViewController project, right-click your application and select **New**.
2. In the New Gallery dialog box, expand the Web Tier node.
3. Select **JSF**.
4. In the Items list, select **JSF JSP**.
5. Click **OK** to display the Create JSF JSP dialog box.
6. If you are on the Welcome page of the wizard, then click **Next** to display the JSP File page.
7. In the File Name field, specify a name for your login page. For example, `ADFLogin.jspx`.
8. Select **JSP Document (*.jspx)**.
9. Click **Next** to display the Component Binding page.

10. Select **Automatically Expose UI Components in a New Managed Bean**.
11. Click **Next** to display the Tag Libraries page and select **ALL Libraries**.
12. Make sure that the following libraries are listed in Selected Libraries:
    - JSF Core
    - JSF HTML
    - ADF Faces Components
    - ADF Faces HTML
    - ADF Portlet Components
    - Customizable Components Core
13. Click **Finish** to create the page.
14. Save the page.
15. From the Component Palette, select **Customizable Components Core**.
16. Select the **PanelCustomizable** and drag it onto the Structure pane above the `h:form` node. Because you will be creating a custom HTML form in the `PanelBox`, click **No** in the Confirm Add Form Element dialog box.
17. In the Property Inspector, set the layout of the Panel Customizable to **Horizontal**.
18. In the Structure pane, drag the `h:form` node onto the `cust:panelCustomizable` node as shown in Figure 10–34.

Figure 10–34 h:form Node

19. Open the Component Palette, select **ADF Faces Core** and drag a **PanelBox** above the `h:form` tag in the Structure pane.
20. Set the `Text` property of the `PanelBox` to **Login** as shown in Figure 10–35.

Figure 10–35 Text Property of the panelBox



21. Select an **OutputText** component and drag it onto the `PanelBox` component.
22. Save the page.

In the next section, you will see how the backing bean injects the appropriate login form into this `PanelBox` area.

10.4.2 Adding Login Code to the Backing Bean

Now that you have created the login page as an Oracle ADF Faces page, you cannot just add the login form using form elements from the component palette. This would cause the form elements to be serialized and remapped at run time by the Oracle ADF Faces life cycle. Instead, you can inject the HTML for the login form at run time, by including it in the backing bean and dynamically showing this at run time. To include the HTML code in a backing bean and to reference the HTML login form in the login page, perform the following steps:

1. In the Applications Navigator, expand the Application Resources node and open the `<Page_Name>.java` backing bean, for example, **ADFLogin.java**.
2. To define the bean `_loginFormBlock`, create a new attribute by adding the following in the declaration section of the `ADFLogin.java` file:

   ```
   private String _loginFormBlock;
   ```

3. Add the `get` method for this attribute right before the closing brace (`}`) in this Java class as shown in Example 10–13.

Example 10–13 LoginFormBlock code that injects the Login Form into the Login Page

```
public String getLoginFormBlock()
{
    String htmlBlock      = null;
    String userNameLabel  = "Username";
    String passwordLabel  = "Password";
    String buttonLabel    = "Login";


    htmlBlock = "\n\n" +
     "<!-- === Login Form Block Generated in Backing Bean ==== -->\n" +
     "<form name=\"LoginForm\" id=\"LoginForm\" \n" +
     "      action=\"j_security_check\" method=\"POST\" >\n" +
     " <table cellspacing=\"5\" cellpadding=\"0\" border=\"0\" width=\"50%\">\n" +
     " <tr>\n" +
     "  <td nowrap>" + userNameLabel + "</td>\n" +
     "  <td nowrap><input type=\"text\" name=\"j_username\"/></td>\n" +
     " </tr>\n" +
     " <tr>\n" +
     "  <td nowrap>" + passwordLabel + "</td>\n" +
     "  <td nowrap><input type=\"password\" name=\"j_password\"/></td> \n" +
     " </tr>\n" +
     " <tr>\n" +
     "  <td><input type=\"submit\" value=\"" + buttonLabel + "\"/></td> \n" +
     " </tr>\n" +
     " </table>\n" +
     "</form>\n" +
     "<!-- ================================================ -->\n\n" ;
    _loginFormBlock = htmlBlock;
    return (_loginFormBlock);
    }
```

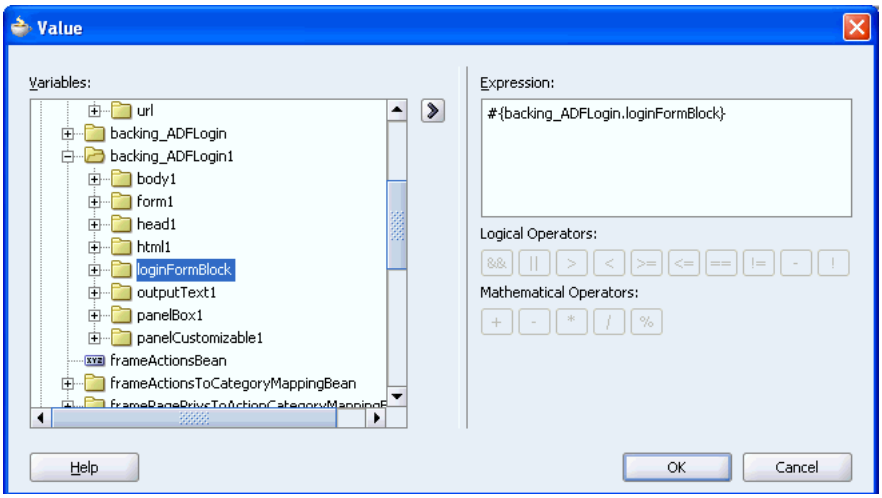
   This code returns the entire login block as a simple output string (simplifying the need for `<verbatim>` tags).

4. Save the Java file.

5. In the Property Inspector, set the value of the previously created `outputText` object to the following value of the `loginFormBlock` attribute, as shown in Figure 10–36:

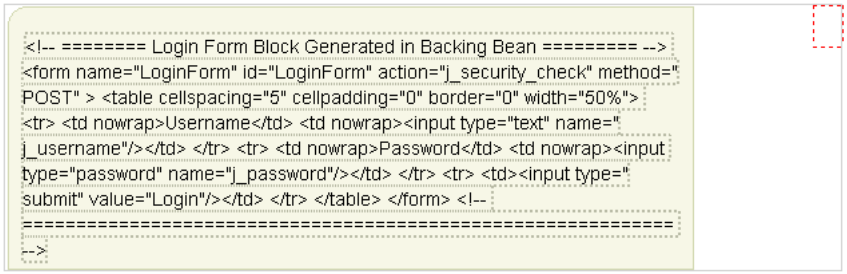
   ```
   #{backing_ADFLogin.loginFormBlock}
   ```

Figure 10–36 loginFormBlock Attribute



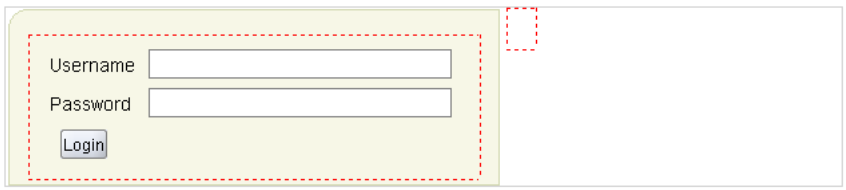
This will result in the full sting appearing as "Text" within the page as shown in Figure 10–37.

Figure 10–37 Login Form Block Generated in the Backing Bean



6. In the Property Inspector, set the `Escape` property of the `outputText` object to `False` to render the string as static HTML as shown in Figure 10–38.

Figure 10–38 HTML Login Form



7. Save the file.

10.4.3 Adding Portlets to the Login Page

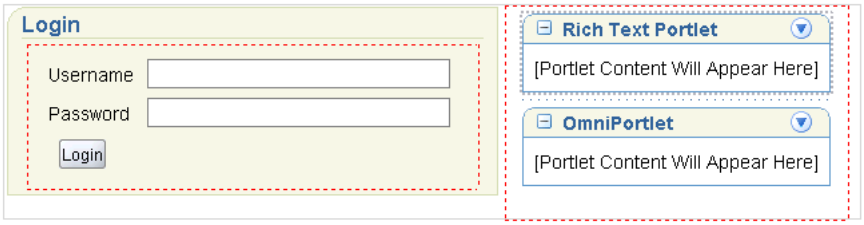
While this step is optional, the benefit of using an Oracle ADF Faces-based login page for your WebCenter application is that you can add portlets to the page to add customizable information to the page or make it part of the application itself. In this example, you will add a Rich Text portlet and an OmniPortlet to the login page. See Section 14.3.1, "Rich Text Portlet" and Section 14.3.4, "OmniPortlet" for more information.

Make sure your portlet producers have been registered before proceeding. See Section 4.3.1.1, "Registering WSRP Portlet Producers" and Section 4.3.1.2, "Registering PDK-Java Portlet Producers" for details.

To add portlets to the login page, perform the following steps:

1. Drag a **PanelCustomizable** onto the `h:form` tag in the Structure pane.
2. From the Component Palette, select **RichTextPortlet Producer**, then select the Rich Text portlet from the list and drag it onto the `PanelCustomizable` component.
3. From the Component Palette, select **ADF Faces Core** and drag an **ObjectSeparator** below the Rich Text portlet on the `PanelCustomizable` component.
4. From the Component Palette, select **OmniPortlet Producer**, then select the OmniPortlet from the list and drag it onto the `PanelCustomizable` component.
5. Save the page. It will look like Figure 10–39.

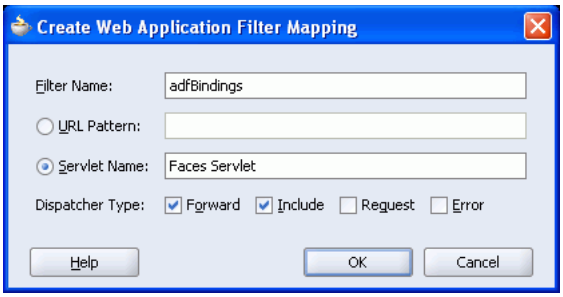
Figure 10–39 Login Page with Portlets



Because the login page is called from the container as part of the login process, the request must be forwarded to the ADF binding filter to establish the appropriate portlet and security context. To do this, you must configure a mapping for the ADF Binding filter in the `web.xml` file. To do this, perform the following steps:

1. In the Applications Navigator, expand the WEB-INF node, right-click **web.xml** and select **properties** to open the property palette.
2. Select **Filter Mappings** in the left panel and click **add** to define a new mapping for the `adfBindings` Filter. This displays the Create Web Application Filter Mapping dialog box.
3. Specify **adfBindings** for the filter name and click the Servlet Name option and specify **Faces Servlet** as the servlet name. Ensure that the Forward and Include dispatcher types are selected as shown in Figure 10–40.

Figure 10–40 Create Web Application Filter Mapping Dialog Box



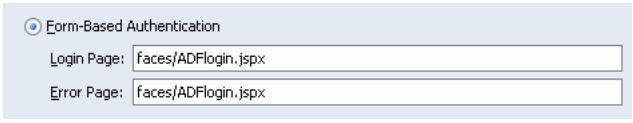
4. Click **OK**.

10.4.4 Configuring the web.xml File for an Oracle ADF Faces-Based Login Page

As the login page is called directly from the container, it is not part of the Oracle ADF Faces navigation process. As such, you must force a call to the Oracle ADF Faces servlet when calling the login page. To do this, perform the following steps:

1. Expand the WEB-INF node in the Applications Navigator, right-click **web.xml** and select **properties** to open the property palette.
2. Click **Login Configuration** in the properties Panel.
3. Set the value of the Login Page to include a reference to the Oracle ADF Faces servlet such that the login page can be part of the Oracle ADF Faces life cycle `faces/ADFlogin.jspx` page as shown in Figure 10–41.

Figure 10–41 Adding a Reference to the Faces Servlet in the Login Configuration



4. Click **OK** to save the changes to `web.xml`.

10.4.5 Editing Authorization for the Login Page

As the application is secured by Oracle ADF security, all pages in the application are secured and therefore need explicit policies defined against them. As all users are required to be able to log on, the login page must be publicly accessible. If the page is not defined as public then the container will continually redirect to the defined authentication point before enabling access to the page (which in this case is the authentication page).

You must define the appropriate privileges for the login page so that users with specific roles can perform specific actions. For example, all users (the role `anyone`) must be able to view the page, while specific users that are required to edit the portlets must have customize permission on the login page.

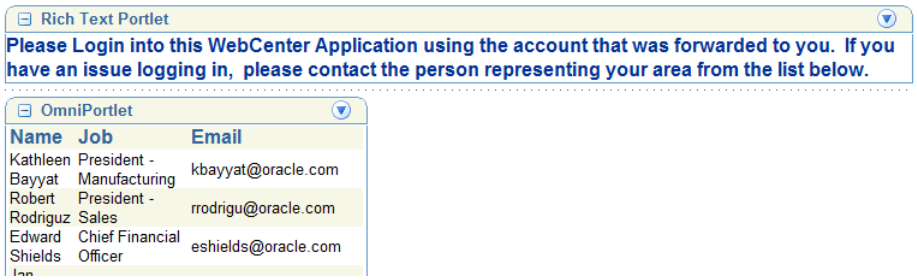
> **Note:** To enforce security in your application, you must first configure the security infrastructure for your application as described in Section 10.2, "Setting Up Security for Your Application".

To define access policies on the login page, perform the following steps:

1. In the Applications Navigator, right-click **ADFLogin.jspx**.
2. Select **Go to Page Definition**.
3. Click **Yes**, if you are prompted to create a new page definition. The page definition file opens in the Structure pane.
4. Right-click the Page Definition file and select **Edit Authorization**. This displays the Authorization Editor.
5. Grant **View** privilege to `anyone`.
6. Click **OK**.
7. As the Login Page contains customizable portlets, the appropriate role must be granted Customize privilege on the page.

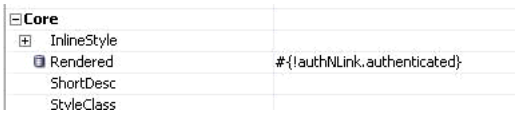
   Given that the login page contains customizable portlets, which will be edited by a group of users that has the appropriate permission, the login page must also be accessible from within the application itself (for example, from an administration page). In that case the user is directed to the page through the standard Oracle ADF Faces navigation rather than directly from the container. As such, the login component does not make sense and hence should not be rendered, thus preventing authenticated users from accidentally submitting the login form again. Figure 10–42 shows the login page as seen by an authenticated user.

Figure 10–42 Login Page Seen by an Authenticated User



   Click the `PanelBox` in the previously created `ADFLogin.jspx` page and set the `rendered` property to be `false` if the user is currently authenticated. To do this, use the `authenticated` property of the `authNLink` managed bean as shown in Figure 10–43.

Figure 10–43 Setting the Rendered Property Based on the Authenticated State of the User



8. Ensure that you have a navigation case defined in the `faces-config.xml` file to enable an authenticated user to access the login page. You can then use the appropriate Oracle ADF Faces navigation component in your application to enable you to navigate to the login page to customize it.
9. Save the pages and run the application.

10.5 Creating a Public Welcome Page for Your Application

As WebCenter applications are generally secured, there is always a need for a starting point or home page for unauthenticated users. To create this public welcome page, you create an Oracle ADF Faces page to act as the entry point for the application, which contains links to other pages within the application. However, only links to public pages should be rendered to unauthenticated users and conversely, links to secured pages should be rendered only after the user has logged in and has the appropriate privileges to view the target page.

To create a public page for your application, perform the following tasks:

- Section 10.5.1, "Making the Welcome Page Public"
- Section 10.5.2, "Adding Login and Logout Links"
- Section 10.5.3, "Hiding Links to Secured Pages"

10.5.1 Making the Welcome Page Public

After you have created a regular Oracle ADF Faces page as described in Section 4.2, "Building WebCenter Application-Enabled Pages in Oracle JDeveloper with Oracle ADF", you must make the page accessible to all users (authenticated as well as unauthenticated). To do this, you must perform the following steps:

> **Note:** To enforce security in your application, you must first perform the steps described in Section 10.2, "Setting Up Security for Your Application".

1. Right-click the welcome page, for example, `welcome.jspx`, and go to the page definition. Create a new one if prompted.
2. In the Structure pane, right-click the page definition and select **Edit Authorization**. The Authorization Editor is displayed.
3. Grant the View privilege to the role `anyone`.
4. Grant Customize and Personalize privileges as required.
5. Click **OK** and save the page.

This will add the view privilege to the anyone role in the `system-jazn-data.xml` file. In an application protected by Oracle ADF Security each user is automatically made a member of this pseudo role (the anyone role principal is automatically added to the user's subject). Therefore, a public page is a special type of secured page that is available to everyone. This is different from J2EE security, in which a public page is defined by the simple lack of security constraints against that page. The Oracle ADF Security model, therefore differentiates between a page secured for public access and the absence of a secured implementation.

10.5.2 Adding Login and Logout Links

You can add login and logout links to your public welcome page so that users can explicitly log in and out while they are in the application. While J2EE Container managed security supports the concept of authentication when accessing a secured resource, there is no standard way to log out and stay within a secured application. However, this is a common practice in WebCenter applications. Either staying on the same page if that page is public or returning to the welcome page if that page is secured. While adding the login and logout links to each page would let the user to

end their login session anywhere within the application (and return to the welcome page), having these links on the welcome page enables them to explicitly authenticate on entering the application.

To add the login and logout links, you must add a login component to your application and then add the login and logout links to your page as described in Section 10.3, "Creating a Login Component for Your Application".

10.5.3 Hiding Links to Secured Pages

As an anonymous user should not have access to any secured pages, any navigation component on the welcome page that points to a secured page should be hidden from view based on the following two criteria:

- Is the user authenticated with a known user identity?
- Does the specified user identity have permission to view the target?

If either of these criteria has not been met, the `rendered` attribute of any navigation component on a public page that point to a secured resource, must have its `rendered` property set to `false`, thus hiding it from the anonymous user. To enforce these rules within your welcome page, see section Section 10.2.4, "Enforcing Security Policies in Your Application".

10.6 Configuring Basic Authentication for Testing Portlet Personalization

Portlet personalizations are tied to particular, authenticated users. Hence, when running a portlet that has an Edit mode, the Personalize option in the portlet's dropdown menu only appears to authenticated users of the application. Anonymous or public users will not have the option to personalize the portlet. If you are a developer creating portlets and pages, then you may want to quickly test the Edit mode of your portlet without creating a complete security model for your application. To perform this sort of testing, you can easily configure some very basic authentication for your application and then remove it when you are done testing:

Note: This procedure is useful for any portlet that has an Edit mode (Omniportlet, Web Clipping, JPS, and PDK-Java).

---

1. Create a user `sking` and the manager role as described in Section 10.2.1, "Defining Roles for Developing Secured WebCenter Applications".
2. Secure your application using the ADF Security Wizard as described in Section 10.2.2, "Configuring Security for Your Application". On the Login page of the wizard, select **HTTP Basic Authentication (RFC 2617)**. This specifies that the application will use basic authentication.
3. Secure the page that contains your portlets by performing the steps in Section 10.2.3.2, "Securing Pages in Your Application".
4. Create the `orion-web.xml` file to run your application in the embedded OC4J as described in Section 10.2.5.1, "Creating the Deployment Descriptor File".
5. Run the page in the embedded OC4J and log in as a valid user and test your portlet's edit mode.

When you are done testing your portlet's Edit mode, you can quickly remove this test security by do the following:

1. In the Applications Navigator, click the project that contains a page with the portlet you want to test.
2. From the **Tools** menu, choose **ADF Security Wizard**.
3. If the Welcome page appears, then click **Next**.
4. Choose **Remove All ADF Security Settings**.
5. Click **Next** until you come to the Finish page of the wizard. Click **Finish**. The security is removed. If you want to ensure that the security has been removed, then exit your browser and rerun the application. When you access the page, you should not be prompted to login and the personalize option should be gone from the portlet's dropdown menu.

10.7 Accessing External Applications Requiring Credentials

When an Oracle PDK portlet producer's implementation depends on an application that handles its own authentication, you must associate that application with the producer. At design time, this is a simple matter of registering the external application, then selecting the external application from a list when you register or edit an Oracle PDK portlet producer.

The Oracle WebCenter Suite provides a means of registering external applications with a WebCenter application and adding a Credential Provisioning page for use in logging into the external application. For more information about the Credential Provisioning page, see Section 10.1.3, "External Application Credentials and Portlets".

Caution: In this release, WebCenter application external application support is for use in portlets only. That is, portlets can return a user's stored credential set for an application and use it to authenticate the user to a remote application. However, there is currently no support for directly linking to an external application from the portlet. For example, starting an external application from a link on a WebCenter application page. It is important to ensure that portlets that are developed to use external applications avoid the use of direct (deep) links to an external application, because this would result in an authentication request but, the login URL is not sent to portlets. This happens because there is no automatic proxy single sign-on. If possible, Oracle recommends that portlets use an inline rendering model, where URLs to the external application are accessed through the portlet's framework.

This section provides information about registering external applications and adding a Credential Provisioning page. Additionally, it describes the process of editing and deleting registration details. It contains the following subsections:

- Section 10.7.1, "Working with External Applications"
- Section 10.7.2, "Working with Credential Provisioning Pages"

10.7.1 Working with External Applications

This section provides information about registering external applications and editing and deleting registration details. It contains the following subsections:

- Section 10.7.1.1, "Registering an External Application"

- Section 10.7.1.2, "Editing External Application Registration Details"
- Section 10.7.1.3, "Deleting External Application Registration Details"

See Also: Section 4.3, "Consuming Portlets"

10.7.1.1 Registering an External Application

Use the Register External Application Wizard to identify and store information about the type of data required to authenticate to an external application, such as the names of login.

To register an external application:

1. In the Applications Navigator, right-click a WebCenter application or project and select **New** from the context menu.
2. In the New Gallery, select **External Applications** under the **General** node.
3. In the right pane, select External Application, and click **OK**.

   This opens the Register External Application Wizard.
4. On the Welcome page, click **Next** to move to the Name page.

   Optionally, before leaving the Welcome page, select the **Skip this Page Next Time** check box to forgo viewing the Welcome page on subsequent uses of this wizard.
5. On the Name page, in the **Name** field enter a unique name to identify the application.

   This name must be unique within the WebCenter application.
6. Click **Next**.
7. On the General page, in the **Login URL** field enter the URL to which the HTML login page is submitted.

   View the HTML source of the application's login form to retrieve this URL.

   Note: The Predeployment tool does not enable for modification of external application data at the predeployment stage. Therefore, if your external application is hosted on a different computer or instance, then you must update the external application login URL just prior to creating an EAR file from the WebCenter deployment profile. Change the first part of the login URL from `http://m1.abc.com:7777/` to, for example, `http://lbr.abc.com/`.
8. In the **User Name/ID FieldName** field, enter the label that the application uses for the user name field, for example, `User Name`.
9. In the **Password FieldName** field, enter the label that the application uses for the password field, for example `Password`.
10. From the **Authentication Method** list, select the application's login method.

    Choose from the following:

    - **GET**

      Presents a page request to a server. Submits the login credentials as part of the login URL.

- **POST**

  Submits login credentials within the body of a form.

- **BASIC**

  Submits login credentials as part of the login URL. Note that the Basic authentication method poses a security risk because the user name and password are exposed in certain instances.

11. Click **Next**.
12. On the Additional Fields page, enter the names and values of any additional fields that are submitted with the external application's login form:

    Click the **Add Field** button to create a new input field:

    - **Field Name**

      Enter a unique name for any additional field that requires user input on the external application HTML login form.

    - **Field Value**

      Enter a default value for the corresponding field name.

    - **Display to User**

      Select to display the field on the external application login screen. If the field is not displayed (unchecked), then a default value must be specified, which will be used to login into the external application for all users. If the value is user-specific, then the field must be displayed to the user who can then provide values for it in the external application Credential Provisioning page.

    Note: The Delete Field option can be used to delete selected rows.

13. Click **OK** to register the external application.

Once you have registered the external application, you must associate a producer with it. You can do this when you register an Oracle PDK-Java Portlet Producer or when you edit one. Relevant settings include **Associate producer with external application**, on the producer's **Connection** tab, and E**nable Producer Sessions**, on the producer's **Registration Details** tab (for external applications, this property should be enabled automatically). For more information, see Section 4.3.1.2, "Registering PDK-Java Portlet Producers".

Note: If you register a producer that requires external application authentication, but do not associate this producer with the external application, then when you add a portlet from this producer to a JSF JSP page and run the page, you may find the Update login credentials link that points to a Credential Provisioning page (that was created for another producer requiring external application authentication) may be displayed with the following error:

```
1. External application ID was found to be null while rendering the
Credential Provisioning page!
2. Cause: The credential page was probably run standalone.
3. Action: The credential page needs to be invoked from the link in
portlets, from producers associated with the external application.
Running the credential page standalone is not supported.
```

After external application registration, create a Credential Provisioning page. This page provides a means for dynamically displaying external application authentication data fields (that is, a login page). Additionally, any application that uses external application portlets should have their pages protected, including the Credential Provisioning page. This can be accomplished through the **ADF Security Wizard**, on the Oracle JDeveloper **Tools** menu. For more information about the Credential Provisioning page, see Section 10.7.2, "Working with Credential Provisioning Pages".

10.7.1.2 Editing External Application Registration Details

Use the Edit External Application Registration Information Wizard to revise the registration details provided for an external application.

To edit external application registration details:

1. In the Applications Navigator, right-click an external application and select **Edit** from the context menu.
2. In the Edit External Application Registration Information Wizard, click a tab to bring it forward and revise its values.

   Choose from the following:

   - **Name**
   - **General**
   - **Additional Fields**

   For more information, see Section 10.7.1.1, "Registering an External Application".
3. Click **OK** to save your changes and exit the wizard, or click **Cancel** to exit the wizard without saving.

10.7.1.3 Deleting External Application Registration Details

To delete external application registration information, perform the following steps:

1. In the Applications Navigator, right-click an external application and select **Delete** from the context menu.

   Alternatively, you can select an external application in the Applications Navigator and from the Edit menu, select **Delete**.
2. In the External Application Delete dialog box, select **Yes**.

If you are deleting the lone external application registration in the WebCenter application, then remember to also remove its associated Credential Provisioning page from each of the application's projects. In addition, Oracle recommends that you deregister the Oracle PDK portlet producer with which the external application is associated. If you do not deregister the associated producer after deleting the external application, then the portlets of these providers are likely to stop functioning and throw run time errors.

10.7.2 Working with Credential Provisioning Pages

Credential Provisioning pages consume application registration details to provide an application login page where users can enter their credentials and authenticate to the external application. User credentials are preserved in a credential store, which handles logins for future sessions. The user does not have to enter login information again (unless the user's credentials change).

This section provides information about adding a Credential Provisioning page at design time and adding credentials at run time. It contains the following subsections:

- Section 10.7.2.1, "Adding a Credential Provisioning Page"
- Section 10.7.2.2, "Adding Credentials at Run time"

10.7.2.1 Adding a Credential Provisioning Page

The Credential Provisioning page is a JSF page (`.jspx`) that is built based on the information provided through external application registration.

To add a Credential Provisioning page:

1. Register an external application.

   For more information, see Section 10.7.1.1, "Registering an External Application".

2. In the Applications Navigator, right-click a project scoped for creating portlets and select **New** from the context menu.

   For information about creating projects scoped for creating portlets, see Section 3.1, "Creating a WebCenter Application".

3. In the New Gallery, select **External Applications** under the General node.

4. In the right pane, select **Credential Provisioning Page**.

5. Click **OK**.

   The file `CredentialProvisioner.jspx` is added to the root of the project by default under the Web Content node in the Applications Navigator. If you should manually move or rename this page, then you must update the navigation rule, more precisely, the `<to-view-id>` entry in the `_adfp_external_apps_credential_page <from-outcome>` navigation rule, to reflect this change.

The dynamic rendering of login information is made possible through data binding with `extAppCredentialProvBean`. Although you are free to adjust the look and feel of this page however you want, you must not in any way alter any `extAppCredentialProvBean` entries in the `faces-config.xml` file.

Should you mistakenly alter `extAppCredentialProvBean` entries, then you can overwrite this file by adding the Credential Provisioning page again from the New gallery.

10.7.2.2 Adding Credentials at Run time

At run time, the Credential Provisioning page displays login data fields. In addition to the user name and password fields, data fields specified through external application registration are also displayed. Users fill in the fields with their login information. This information is passed to the producer, which in turn passes the login values to the application. The application provides the producer with the requested portlet.

The user provides login credentials when prompted and these credentials are preserved in a credential store. The credential store subsequently supplies that information during authentication. The user supplies credentials only once (unless those credentials change).

To add credentials to a Credential Provisioning page at run time:

1. On the portlet page, enter the user name to access the external application.
2. Enter the password to access the external application.
3. Fill in any other fields provided.

   The inclusion of additional information-gathering fields will vary according to the external application definition being accessed.
4. Click **OK** to store credentials and return to the portlet page, from which you can now access the portlet.

For future portlet sessions, this login will be transparent, as your user credentials are supplied from the credential store. The login page will display again should your credentials change.

If required, then you must access the Credential Provisioning page at run time only by clicking the Update login information link in the external application portlet. Do not try to access the Credential Provisioning page by invoking it directly or in any other manner.

Note: The Credential Provisioning page must be added within each project containing pages with portlets from external application-enabled producers. If this is not done, then at run time, you will not be able to access the Update login information link on the portlet.

10.8 Registering Custom Certificates with the Keystore

Secure Sockets Layer (SSL) Communication requires the use of trusted certificates issued by a certificate authority, which vouches for the authenticity of the certificates that it issues or signs. Widely accepted certificate authorities are listed in the keystore, the `cacerts` file, available in the `<JDEV_HOME>\jdk\jre\lib\security` directory. If a portlet producer uses a security certificate issued by a non-widely accepted certificate authority and you try to access portlets from this producer, a security alert is displayed informing you that the security certificate was issued from a certificate authority you do not trust. This means the certificate is not available in the keystore. To avoid being prompted each time you access such portlets, you must register this certificate with the keystore.

To register a certificate with the keystore, perform the following steps:

1. Navigate to `<JDEV_HOME>\jdk\jre\lib\security`.
2. Back up the `cacerts` file.

3. Access the producer URL in Internet Explorer to get the certificate.

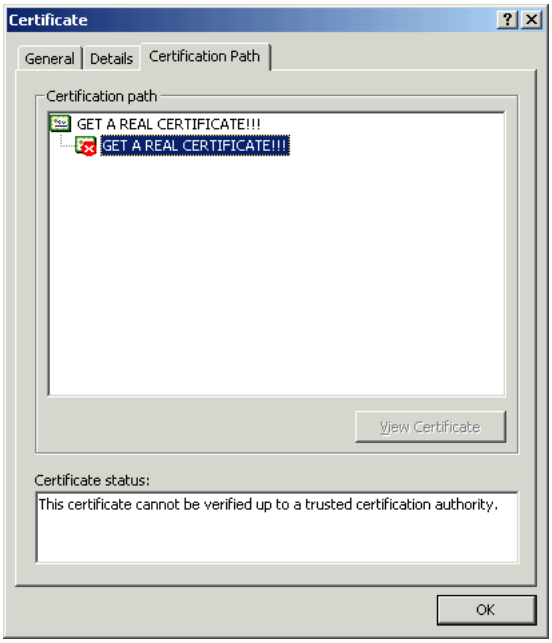
> **Note:** Recent versions of FireFox do not provide a means to export certificates.

4. In the Security Alert dialog box, shown in Figure 10–44, click **View Certificate**.

Figure 10–44 Security Alert Dialog Box

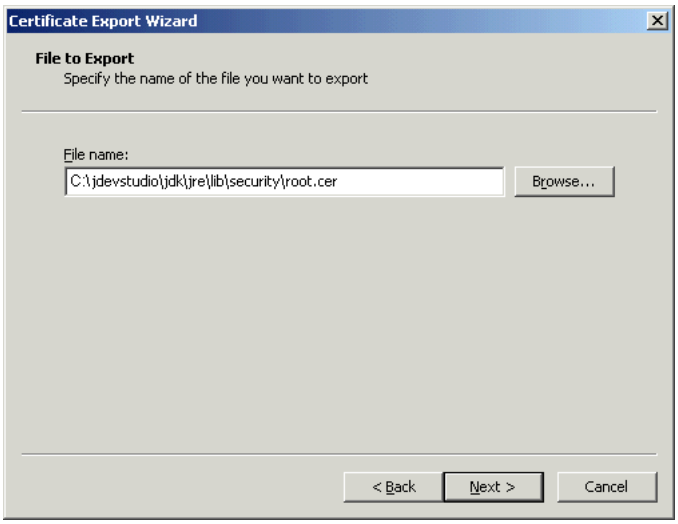
5. In the Certificate dialog box, click the **Certification Path** tab.
6. The dummy child certificate is selected by default as shown in Figure 10–45. Select the root certificate and click **View Certificate**.

Figure 10–45 Certificate Dialog Box



7. Click the **Details** tab, and click **Copy to File**.
8. In the Certificate Export Wizard, accept the default settings and click **Next** until you reach the File to Export screen, shown in Figure 10–46.

Figure 10–46 File to Export Screen of the Certificate Export Wizard



9. In the File Name field, enter <*JDEV_HOME*>\jdk\jre\lib\security\root.cer and click **Next**.
10. Click **Finish**.
11. In the command prompt, set your default directory to <*JDEV_HOME*>\jdk\jre\lib\security and run the following command:

    ```
    keytool -import -file root.cer -keystore cacerts -storepass changeit
    ```

    By running this command, the root.cer certificate is imported into the keystore.
12. Enter y at the prompt to confirm that you trust this certificate.
13. Verify that the cacerts file is updated with the certificate.

10.9 Overriding Inherited Security on Portlets and Customizable Components

Individual actions on portlets and customizable components are not secured by default. Rather, the ability to customize a portlet or customizable component as a whole is inherited from the page permissions. If you want to grant more granular activities within a portlet or customizable component, then you can override the page-level security inheritance and define security directly on the required actions.

The ability of a user to perform actions on portlets and customizable components is inherited from the page security based on the value of the application-wide switch, enableSecurity, in the adf-config.xml file. If you selected the WebCenter application template while creating your application, then the adf-config.xml file is located in the <*APPLICATION_NAME*>/.adf/META-INF directory. The enableSecurity element is not available by default in adf-config.xml. To override or extend the page-level security inheritance for portlets and customizable components, you must add the enableSecurity element under the portlets security and customizable components security sections in the adf-config.xml file, as shown in Example 10–14 and Example 10–15.

Example 10–14 enableSecurity Element in the Portlet Security Section in adf-config.xml

```
<!--
```

```
================================================================================
PORTLETS ACTIONS SECURITY
================================================================================
-->
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>
    <adfp:actionsCategory>
       ....................................
</adfp:adf-config-child>
```

Example 10–15 enableSecurity Element in the Customizable Components Security Section in adf-config.xml

```
<!--
================================================================================
CUSTOMIZABLE COMPONENTS ACTIONS SECURITY
================================================================================
-->
<cust:customizableComponentsSecurity
xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <cust:enableSecurity value="true"/>
    <cust:actionsCategory>
       ....................................
</cust:customizableComponentsSecurity>
```

Security for actions on portlets and customizable components can be implemented at the following levels:

- Page level: You can define security for portlets and customizable components such that page-level privileges are inherited by these components. This is the default behavior.

  By default, portlets and customizable components inherit permissible actions from the defined page-level permissions such as personalize or customize. That is, a user who has *customize* privileges on the page has permission on the customize action for the components on that page. The `enableSecurity` element enables you to override the security inheritance behavior and can take either of the following values:

  – `true`: If set to `true` (the default), then the ability for a user to modify a component will first be determined from the page permissions and then adjusted according to the current set of actions defined for that type of permission. If a user has customize permission, then the actions that constitute the customize category (move, customize, and so on) are available to the user, but they will be overridden by the actions that are defined in the `adf-config.xml` file. For example, a page designer wants to enable the end user to be able to customize portlets, but not customize the page layout. By setting `enableSecurity` to `true`, the page designer enforces that users must first have customize permission on the page. Setting `customizeActionsCategory` to `false` for customizable components will prevent the customization of the page layout, yet still enabling portlet customization. (As the default for `customizeActionsCategory` is `true`, it does not need to be set explicitly for portlets.).

  – `false`: If set to `false`, then the behavior is to ignore the user's page permissions and base the available actions on the manually specified list in the `adf-config.xml` file. In this case, the actions are global and available to all users. That is, the default page privileges (View, Personalize, and Customize

for portlets and View and Customize for customizable components) are available for all portlets and customizable components.

- Actions category level: You can define security on all actions for portlets or customizable components that belong to a named category.

  You can add an `actionsCategory` element in the `adf-config.xml` file to define security on multiple actions simultaneously. Depending on the `actionCategory` attributes that you enable, appropriate privileges are provided on the portlets or customizable components.

- Actions level: You can define security on individual actions for portlets or customizable components.

  You can use the `actions` element in the `adf-config.xml` file to enable or disable individual actions. Depending on the `actions` attributes that you enable, appropriate privileges are provided on the portlets or customizable components.

Notes:

- Privileges can be inherited from the parent only. Inheritance from a component in any other position in the hierarchy is not supported
- Although the security override implementation for portlets and customizable components is similar, they are independent from each other. Therefore, if you place a portlet inside a customizable component (for example, in a `PanelCustomizable` component), the portlet will not inherit override settings from the customizable component. Instead it will use the security override settings that are defined for portlets.
- Settings made at the actions category level or actions level are applicable for all component instances in the application. These settings cannot be made for a single instance of a portlet or customizable component.

The following sections describe how you can implement security on portlets and customizable components actions, at the actions category level and actions level:

- Section 10.9.1, "Portlets Security"
- Section 10.9.2, "Customizable Components Security"

10.9.1 Portlets Security

You can define portlet security if actions on portlets are inherited from the page at the application level by setting `enableSecurity` to `true` in the portlets security section of the `adf-config.xml` file. A value of `true` implies that the user's permissions are determined from the page permission and then augmented according to the `actionsCategory` and `actions` elements specified. By defining actions categories and individual actions, you can control the exposure of the individual actions available within the given page permissions.

To implement security for actions on portlets at various levels as described earlier, you must define security settings at the following sections:

- Defining Security at the Actions Category Level
- Defining Security at the Actions Level

10.9.1.1 Defining Security at the Actions Category Level

You can add an `actionsCategory` element in the portlets security section in the `adf-config.xml` file to define the group of actions that are exposed on the portlets within the application. Depending on the `actionsCategory` attributes that you enable, appropriate privileges are provided on the portlets. Table 10–11 describes the different `actionsCategory` attributes and the portlet actions they support by default.

Table 10–11 actionsCategory Attributes and Portlets Actions Mapping

Attribute Value	Actions Supported
viewActionsCategory	Render isHelpModeAvailable isNormalModeAvailable isAboutModeAvailable isPreviewModeAvailable isDetailModeAvailable isLinkModeAvailable isPrintModeAvailable
customizeActionsCategory	isMovable isCustomizeModeAvailable isMinimizable isMaximizable isConfigModeAvailable
personalizeActionsCategory	isPersonalizeModeAvailable

Example 10–16 shows the `actionsCategory` entry that you can add to the portlets security section in the `adf-config.xml` file. In this example, `customizeActionsCategory` is set to `false` to prevent customization. You can use Expression Language (EL) for the values of these elements.

Example 10–16 actionsCategory Element in the Portlets Security Section

```
<!--
================================================================================
PORTLETS ACTIONS SECURITY
================================================================================
-->
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>
    <adfp:actionsCategory>
      <adfp:actionCategory name="viewActionsCategory" value="true"/>
      <adfp:actionCategory name="customizeActionsCategory" value="false"/>
      <adfp:actionCategory name="personalizeActionsCategory" value="true"/>
    </adfp:actionsCategory>

    <adfp:actions>
    ......................................
    </adfp:actions>

</adfp:adf-config-child>
```

10.9.1.2 Defining Security at the Actions Level

You can use the `actions` element in the portlets security section of the `adf-config.xml` file to enable or disable individual portlet actions. Depending on the `action` attributes that you enable, appropriate privileges are provided on the portlets.

Example 10–17 shows an example of an `actions` entry that you can add to the portlets security section in the `adf-config.xml` file. You can use EL for the values of these elements. In this case you prevent customization by setting `isCustomizeModeAvailable` to `false`.

Example 10–17 actions Element in the Portlets Security Section

```
<!--
================================================================================
PORTLETS ACTIONS SECURITY
================================================================================
-->
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>
    <adfp:actionsCategory>
    ....................................
    </adfp:actionsCategory>

    <adfp:actions>
      <adfp:action name="Render" value="true"/>
      <adfp:action name="isMovable" value="true"/>
      <adfp:action name="isCustomizeModeAvailable" value="false"/>
      <adfp:action name="isPersonalizeModeAvailable" value="true"/>
    </adfp:actions>

</adfp:adf-config-child>
```

Using EL to Prevent Customization of Portlets Outside of Business Hours

An example to show when you may need to override inherited portlet security is an application that is configured to disable portlet customization outside of standard business hours. For this, you must first create a managed bean (for example, a managed bean called `appBusinessRules`), containing the method shown in Example 10–18.

Example 10–18 InsideBizHours Method Defined in appBusinessRules Managed Bean

```
public String isInsideBizHours()
  {
    Calendar rightNow = Calendar.getInstance();
    int      currentHr = rightNow.get(rightNow.HOUR_OF_DAY);

    // Do not enable customize operation outside of standard business hours

    if ((currentHr > 9) && (currentHr < 17))
       return "true";
    else
       return "false";
  }
```

You can then reference this managed bean from the `actionsCategory` element in the portlet security section of the `adf-config.xml` file, as shown in Example 10–19.

Example 10–19 InsideBizHours Method Referenced in the adf-config.xml File

```
<adfp:adf-config-child xmlns:adfp="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>

    <adfp:actionsCategory>
      <adfp:actionCategory name="customizeActionsCategory"
      value="#{appBusinessRules.InsideBizHours}"/>
    </adfp:actionsCategory>

</adfp:adf-config-child>
```

In this example, the `customizeActionsCategory` will be set to `true` only if the application is run within business hours. Outside of these hours, the portlet cannot be customized even if the user had that permission granted on the page. All other categories that are not explicitly defined, will be inherited from the page.

Using EL to Prevent Personalization and Customization of Portlets Outside the Corporate Network

In this example the managed bean checks the IP address of the request to determine whether the user has accessed the application through the corporate proxy server or from within the corporate network. In this simple example, assume that if the request has the proxy server's IP address, then it is coming from outside the corporate network. In general it is not advised to base security strictly on IP addresses, because these can be compromised. For this, you must add the method shown in Example 10–20 to the managed bean:

Example 10–20 InsideCorpNetwork Method Defined in appBusinessRules Managed Bean

```
public boolean isInsideCorpNetwork()
  {
    // Do not enable personalize and customize operations
    // for requests that go through the corporate proxy

    FacesContext       ctx = FacesContext.getCurrentInstance();
    ExternalContext    ectx = ctx.getExternalContext();
    HttpServletRequest myrequest = (HttpServletRequest) ectx.getRequest();
    String             currentIP = myrequest.getRemoteAddr();

    if (currentIP.equals(getProxyServerIP()))
        return false;
    else
        return true;
  }
```

You can then reference this managed bean from the `actionsCategory` element in the portlet security section of the `adf-config.xml` file, as shown in Example 10–21.

Example 10–21 InsideCorpNetwork Method Referenced in the adf-config.xml File

```
<adfp:adf-config-child xmlns="http://xmlns.oracle.com/adfp/portlet">
  <adfp:enableSecurity value="true"/>

    <adfp:actionsCategory>
      <adfp:actionCategory name="customizeActionsCategory"
                           value="#{appBusinessRules.InsideCorpNetwork}"/>
      <adfp:actionCategory name="personalizeActionsCategory"
                           value="#{appBusinessRules.InsideCorpNetwork}"/>
```

```
    </adfp:actionsCategory>
</adfp:adf-config-child>
```

In this example, the `customizeActionsCategory` and the `personalizeActionsCategory` will be set to `true` only if the IP address of the request for the application does not match that of the corporate proxy. The assumption is that the internal requests would have a valid client IP address. All other categories that are not explicitly defined, will be inherited from the page.

10.9.2 Customizable Components Security

You can define security for actions on customizable components at the application level if `enableSecurity` is set to `true` in the customizable components security section of the `adf-config.xml` file. By default, this element is set to `true` when you implement security for your application. A value of `true` implies that permission checks are made in addition to the `actionsCategory` and `actions` values specified in the `adf-config.xml`.

To implement security for actions on customizable components at various levels as described earlier, you must perform the tasks outlined in the following sections:

- Defining Security at the Actions Category Level
- Defining Security at the Actions Level

10.9.2.1 Defining Security at the Actions Category Level

You can add an `actionsCategory` element in the customizable components security section in the `adf-config.xml` file to define security on multiple customizable components actions simultaneously. Depending on the `actionsCategory` attributes that you enable, appropriate privileges are provided on the customizable components.

Table 10–12 describes the different `actionsCategory` attributes and the customizable components actions they support by default.

Table 10–12 actionsCategory Attributes and Customizable Components Actions Mapping

Attribute Value	Actions Supported
viewActionsCategory	Render isHelpAvailable
customizeActionsCategory	isMovable isCustomizeModeAvailable isMinimizable isMaximizable isShowContentEnabled
editActionsCategory	isEditable

Example 10–22 shows the `actionsCategory` entry that you can add to the customizable components security section in the `adf-config.xml` file.

Example 10–22 actionsCategory Element in the Customizable Components Security Section

```
<!--
================================================================================
```

```
CUSTOMIZABLE COMPONENTS ACTIONS SECURITY
================================================================================
-->
<cust:customizableComponentsSecurity
xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <cust:enableSecurity value="true"/>

  <cust:actionsCategory>
    <cust:actionCategory name="viewActionsCategory" value="true"/>
    <cust:actionCategory name="customizeActionsCategory" value="false"/>
    <cust:actionCategory name="editActionsCategory" value="true"/>
  </cust:actionsCategory>

  <cust:actions>
  ......................................
  </cust:actions>

</cust:customizableComponentsSecurity>
```

You can use EL for the values of these elements, as shown in Example 10–23.

Example 10–23 EL Used in Customizable Components an actionCategory Entry

```
<cust:actionsCategory>
  <cust:actionCategory name="customizeActionsCategory"
value="#{appBusinessRules.InsideCorpNetwork}"/>
</cust:actionsCategory>
```

The managed bean, `appBusinessRules`, is defined as shown in Example 10–20.

10.9.2.2 Defining Security at the Actions Level

You can use the `actions` element in the customizable components security section of the `adf-config.xml` file to enable or disable individual customizable components actions. Depending on the `actions` attributes that you enable, appropriate privileges are provided on the customizable components.

Example 10–24 shows the `actions` entry that you can add to the customizable components section of the `adf-config.xml` file. You can use EL for the values of these elements.

Example 10–24 action Elements in the Customizable Components Security Section

```
<!--
================================================================================
CUSTOMIZABLE COMPONENTS ACTIONS SECURITY
================================================================================
-->
<cust:customizableComponentsSecurity
xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <cust:enableSecurity value="true"/>

  <cust:actionsCategory>
    ......................................
  </cust:actionsCategory>

  <cust:actions>
    <cust:action name="isHelpAvailable" value="true"/>
    <cust:action name="isEditable" value="true"/>
    <cust:action name="isCustomizeModeAvailable" value="false"/>
```

```
  </cust:actions>

</cust:customizableComponentsSecurity>
```

You have seen how to enable or disable security for actions on customizable components by editing the `adf-config.xml` file. The `actions` and `actionsCategory` elements in the `adf-config.xml` file have certain default mappings as shown in Table 10–12. You can change these default mappings by editing the `faces-config.xml` file. These settings are optional and the steps involved are described in the next section.

10.10 Securing Identity Propagation Through WSRP Producers With WS-Security

The Web Services for Remote Portlets (WSRP) specification indicates that Web Services Security (WS-Security) can be leveraged for providing secure identity propagation between the consumer and the portlet producer. However, WSRP in and of itself does not provide secure identity propagation of the end user's identity to the portlet producer. The WSRP specification explicitly defers to other security standards for secure identity propagation and does not go into the specific WS-Security profiles or options that should be employed. In the absence of a secure mechanism, WSRP defines the concept of *user categories*, which can be mapped to security roles like the ones used by the JSR168 portlets. By using a combination of WSRP and WS-Security, you can ensure end-to-end security.

Identity propagation without WS-Security

When using WSRP without WS-Security, the `userContext` structure within the SOAP message contains user profile information and user category information. This information is not considered secure and should only be used for personalization and customization functionality. It should not be used for authorization of sensitive resources. This information is also exposed in the JSR168 APIs, `isUserInRole(`*`role`*`)` and `getUserPrincipal`. The code in Example 10–25 shows how a sample portlet's markup rendering code may use the `isUserInRole` API to decide what content to display.

Example 10–25 isUserInRole(role) API

```
private void doViewHtml(RenderRequest request, RenderResponse response)
          throws PortletException,
                 IOException {
        // To do: markup the required content.
        PrintWriter out = response.getWriter();
        out.print("<p>Welcome");
        out.println("</p>");
        if (request.isUserInRole("moderator")){
            out.println("<p>MODERATOR</p>" );
        } else {
            out.println("<p>not moderator</p>" );
        }
        if (request.isUserInRole("participant")){
            out.println("<p>PARTICIPANT</p>" );
        } else {
            out.println("<p>not participant</p>" );
        }
        if (request.isUserInRole("viewer")){
            out.println("<p>VIEWER</p>" );
        } else {
```

```
            out.println("<p>not viewer</p>" );
        }
    }
```

Identity Propagation with WS-Security

When WS-Security is leveraged with WSRP, the user's identity is propagated outside of the SOAP message body, in the WS-Security header. This is a user assertion, using the Username Token format, and is digitally signed to authenticate the consumer and to ensure the integrity of the assertion.

When this mechanism is used, the JSR 168 APIs `isUserInRole` and `getUserPrincipal` are established based on the security context resulting from the WS-Security authentication, rather than the information in the SOAP message's `userContext`.

The use of WS-Security adds some complexity to the configuration and management of the WebCenter application and the set of producers it consumes. However, when the situation warrants its use, it becomes an important ingredient of the SOA architecture that ensures the security of the information being published by the WebCenter application.

Oracle WebCenter Framework supports the following token profiles (to digitally sign the security token and message body to ensure authenticity and integrity):

- Username token without password
- SAML token (uses the *sender vouches* method that the producer uses to confirm the subject assertion)

Digitally signing the security token and the SOAP message body accomplishes the following objectives:

- Consumer Authentication
- Assertion and Message Integrity

Consumer Authentication

When a portlet producer is generating sensitive information, for example paystub information, it is imperative that it only responds to requests to show the information from a legitimate consumer.

By using WS-Security, and having the producer digitally sign the security token and the message body, the producer can verify the signature using the public key of the legitimate consumer. If the signature cannot be verified, then it means that the request may have come from a fraudulent consumer. By requiring the verification of the digital signature, the sensitive information will only be sent to the legitimate consumer.

Assertion and Message Integrity

In addition to verifying the identity of the consumer making the Web Service requests, digitally signing the security token and the message body also ensures that the token and the message have not been tampered with. This prevents such problems as man-in-the-middle attacks where a legitimate request might be intercepted and the user name in the security token replaced with another user name to see the paystub information coming back for the other user. By digitally signing the token, it cannot be tampered with. Any modification to the token would result in the inability to verify the signature on the producer end, and would result in a SOAP fault to be returned instead of the requested paystub information.

Supported Producers

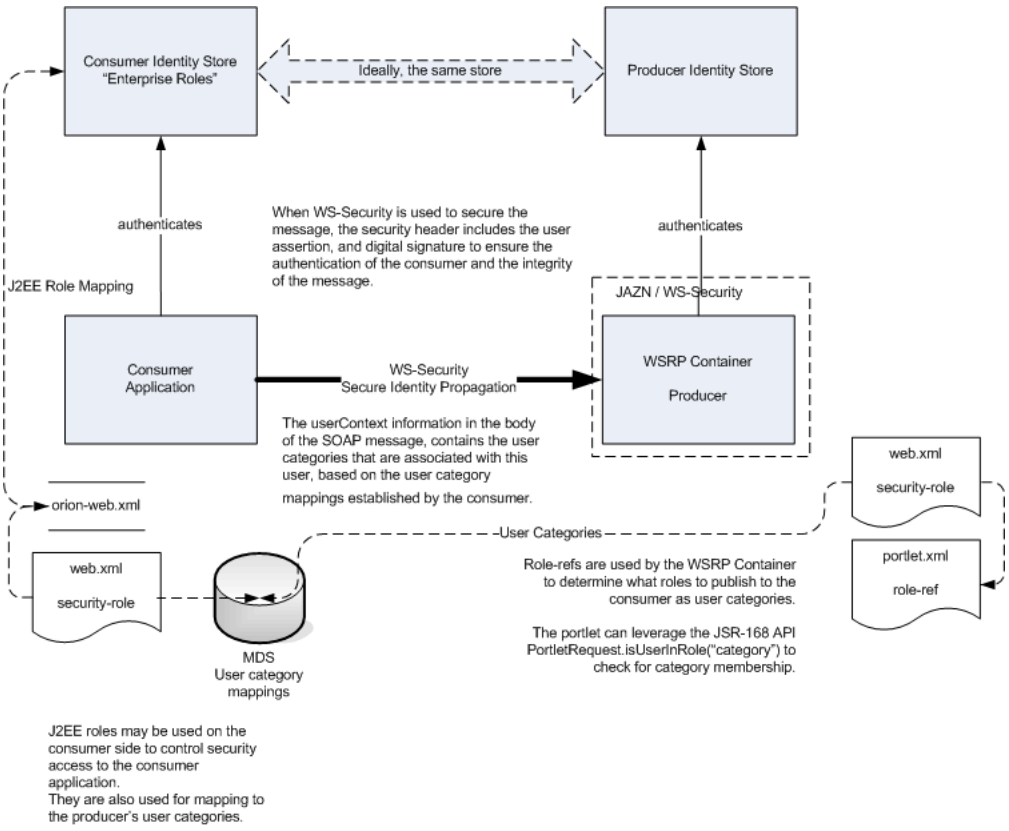
WS-Security implementation is supported by the Oracle WebCenter Suite 10.1.3.2 WSRP container. Other WSRP vendors may also be able to support the WS-Security configuration of Username Token without password, with XML digital signature on the Username Token and the SOAP Message body.

Security Domain Implication

When using secure identity propagation as described in this section, the user name of the user authenticated to the consumer (WebCenter application) is propagated to the producer without any remapping or providing any credentials. There is an inherent assumption that the producer understands this user name and can locate this user in its associated security domain. Consequently, it is highly desirable to ensure that the consumer and producer share the same security provider (identity store) to simplify the management of this configuration.

Figure 10–47 summarizes the overall WSRP portlet security architecture.

Figure 10–47 WSRP Portlet Security Architecture



The following sections discuss how to secure access to JSR-168 standards-based WSRP portlets from WebCenter applications:

- Section 10.10.1, "Setting Up the Keystores"
- Section 10.10.2, "Configuring the Producer"
- Section 10.10.3, "Configuring the Consumer"

10.10.1 Setting Up the Keystores

The security credentials of the WSRP producer and WebCenter application can be obtained and managed either by using a Java Key Store (JKS), or an Oracle wallet. A keystore or wallet is a file that provides information about available public and private keys. Keys are used for a variety of purposes, including authentication and data integrity. User certificates and the trust points needed to validate the certificates of peers are also stored securely in the wallet or keystore.

See the *Oracle Application Server Web Services Security Guide* for information about the Oracle wallet and JKS.

This section covers the following topics:

- Section 10.10.1.1, "Setting Up Keystores Using an Oracle Wallet"
- Section 10.10.1.2, "Setting Up the Keystores Using Java Keystore"

10.10.1.1 Setting Up Keystores Using an Oracle Wallet

You can use the `orapki` utility to generate Oracle wallets. This section covers the following topics:

- Creating a Wallet for the Certificate Authority
- Creating a Wallet for the Consumer
- Creating a Wallet for the Producer

Creating a Wallet for the Certificate Authority

Certificates are signed data structures that bind a network identity with a corresponding public key. Certificates can be of two types; user certificates and trusted certificates. User certificates are used by end entities, including server applications, to validate an end entity's identity in a public and private key exchange. In comparison, trusted certificates are any certificates that you trust, such as those provided by CAs to validate the user certificates that they issue.

In this section, you generate your own Certificate Authority (CA) for testing purposes. If you want to use a real Certificate Authority, then this step can be bypassed. A Certificate Authority is the entity that vouches for the authenticity of the certificates that it issues or signs.

To create a Certificate Authority, perform the following steps:

1. Create a directory to hold the Certificate Authority related files, and navigate to that directory as shown in the following example:

   ```
   mkdir ca
   cd ca
   ```

2. Create the wallet to represent the Certificate Authority by using the `orapki` utility located in the *`ORACLE_HOME`*`/bin` directory as shown in the following example:

   ```
   orapki wallet create -wallet ca_wallet -pwd welcome1
   ```

3. Add a self-signed root certificate to the wallet as shown in the following example:

   ```
   orapki wallet add -wallet ca_wallet -pwd welcome1 -dn "cn=root_test,c=us"
   -keysize 2048 -self_signed -validity 3650
   ```

4. Export the public key of the Certificate Authority, in the form of a certificate for use by other wallets you will be creating, as shown in the following example:

   ```
   orapki wallet export -wallet ca_wallet -pwd welcome1 -dn "cn=root_test,c=us"
   ```

```
-cert root_test.cer
cd ..
```

This command exports the self-signed certificate to the `root_test.cer` file to the `ca` directory.

Creating a Wallet for the Consumer

In this section, you create a wallet for the consumer. The consumer here is the WebCenter application, which consumes portlets generated by the remote portlet producer over WSRP.

To create a wallet for the consumer, perform the following steps:

1. Create a directory, say `consumer`, in the `ca` directory you created earlier, to hold all related files, and navigate to the `consumer` directory as shown in the following example:

   ```
   mkdir consumer
   cd consumer
   ```

2. Create an empty wallet and specify a password for wallet access as shown in the following example:

   ```
   orapki wallet create -wallet consumer_wallet -pwd welcome1
   ```

3. Add an entry into the wallet from which a certificate request can be generated and generate the request as shown in the following example:

   ```
   orapki wallet add -wallet consumer_wallet -pwd welcome1 -dn
   "cn=mywebcenter,c=us" -keysize 2048
   ```

4. Export the certificate request from the wallet, as shown in the following example:

   ```
   orapki wallet export -wallet consumer_wallet -pwd welcome1 -dn
   "cn=mywebcenter,c=us" -request creq.txt
   ```

   This command exports the certificate request to the specified file, `creq.txt`.

5. Generate a signed certificate using the Certificate Authority wallet you created earlier and the certificate request you just created. To do this, use the command shown in the following example:

   ```
   orapki cert create -wallet ../ca_wallet -pwd welcome1 -request creq.txt -cert
   mywebcenter.cer -validity 3650
   ```

   This command creates a certificate, `cert.txt`, with a validity of 3650 days. This is analogous to the Certificate Authority issuing a certificate.

6. Inspect the certificate by using the command shown in the following example and observe that it indicates the issuer as the Certificate Authority, `test_root`, from the Certificate Authority wallet:

   ```
   orapki cert display -cert mywebcenter.cer -complete
   ```

   Following is the output from this command

   ```
   { fingerprint = d72ad668f2080eec50b65d3254e7b4d5, notBefore = Mon Dec 04
   14:38:5
   4 PST 2006, notAfter = Thu Dec 01 14:38:54 PST 2016, holder =
   CN=mywebcenter,C=u
   s, issuer = CN=root_test,C=us, serialNo = 0, sigAlgOID = 1.2.840.113549.1.1.4,
   key = { modulus =
   31488417603689540745711138033912400115391681945788693813007316 4
   ```

```
664917020541127889664630565552725441026050573917027861624168256606809892049389
1
360831237075887305873063338571507380207379608033245190275466249764786154827350
2
493119692881169209078102001234563295996925684413076084734426776382350590303727
3
965761202310608213778699966810908961360386767479647453599582711610045294570955
0
402846802960912760738389189463081695774153591234566636578940332934721703667228
6
450600446522568944798929764752670159998681374777857283106791642181937421250234
8
721050281761374247372486433677314919818203891541370526492342149172084184 97,
exponent = 65537 } }
```

7. Add the Certificate Authority's root certificate to the consumer wallet as shown in the following example:

```
orapki wallet add -wallet consumer_wallet -pwd welcome1 -trusted_cert -cert
../root_test.cer
```

Note: If you are using a real Certificate Authority, such as Verisign or any Certificate Authority from the following list, then you do not need to perform this step:

- CN=GTE CyberTrust Root,O=GTE Corporation,C=US
- OU=Class 3 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
- OU=Class 2 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
- OU=Class 1 Public Primary Certification Authority,O=VeriSign\, Inc.,C=US
- OU=Secure Server Certification Authority,O=RSA Data Security\, Inc.,C=US
- CN=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\, Inc.,O=GTE Corporation,C=US
- CN=Entrust.net Secure Server Certification Authority,OU=(c) 2000 Entrust.net Limited,OU=www.entrust.net/SSL_CPS incorp. by ref. (limits liab.), O=Entrust.net
- CN=Entrust.net Certification Authority (2048),OU=(c) 1999 Entrust.net Limited,OU=www.entrust.net/CPS_2048 incorp. by ref. (limits liab.), O=Entrust.net
- CN=Entrust.net Secure Server Certification Authority,OU=(c) 1999 Entrust.net Limited,OU=www.entrust.net/CPS incorp. by ref. (limits liab.), O=Entrust.net, C=US

These well-known Certificate Authority root certificates are already present in the Oracle wallet. If you are creating your own Certificate Authority for testing, or using a less well-known Certificate Authority, then you should do the following step to add the root certificate used for issuing the certificate into the wallet.

---

8. Add the newly issued user certificate, `mywebcenter.cer`, that represents the WebCenter application's identity into the wallet, as shown in the following example:

```
orapki wallet add -wallet consumer_wallet -pwd welcome1 -user_cert -cert
mywebcenter.cer
```

9. Display the contents of the wallet by using the command shown in the following example:

```
orapki wallet display -wallet consumer_wallet -pwd welcome1
```

Following is the output from this command:

```
Requested Certificates:

User Certificates:
Subject:        CN=mywebcenter,C=us

Trusted Certificates:
Subject:        CN=root_test,C=us
Subject:        CN=GTE CyberTrust Root,O=GTE Corporation,C=US
Subject:        OU=Class 3 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
Subject:        OU=Class 2 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
Subject:        OU=Class 1 Public Primary Certification Authority,O=VeriSign\,
Inc.,C=US
Subject:        OU=Secure Server Certification Authority,O=RSA Data Security\,
Inc.,C=US
Subject:        CN=GTE CyberTrust Global Root,OU=GTE CyberTrust Solutions\,
Inc.,O=GTE Corporation,C=US
Subject:        CN=Entrust.net Secure Server Certification Authority,OU=(c)
2000
Entrust.net Limited,OU=www.entrust.net/SSL_CPS incorp. by ref. (limits liab.),
O=Entrust.net
Subject:        CN=Entrust.net Certification Authority (2048),OU=(c) 1999
Entrust.net Limited,OU=www.entrust.net/CPS_2048 incorp. by ref. (limits
liab.),O=Entrust.net
Subject:        CN=Entrust.net Secure Server Certification Authority,OU=(c)
1999
Entrust.net Limited,OU=www.entrust.net/CPS incorp. by ref. (limits liab.),O=Ent
rust.net,C=US
```

Creating a Wallet for the Producer

The final step of the keystore setup process is generating the wallet for the producer to use. The producer will need the public key of the consumer to verify the authenticity of the security tokens received from the consumer in the WS-Security headers of the requests it receives over its `getMarkup` interface.

To do so, it needs a wallet that contains the certificate of the consumer and the root certificate used to sign it. These are added to the wallet as trusted certificates.

To create a wallet for the producer, perform the following steps:

1. Create the directory for the producer in the `ca` directory you created earlier, and navigate to that producer as shown in the following example:

```
mkdir producer
cd producer
```

2. Create an empty wallet using the command shown in the following example:

```
orapki wallet create -wallet producer_wallet -pwd welcome1
```

3. If required, add the root certificate of the CA to the wallet as shown in the following example:

```
orapki wallet add -wallet producer_wallet -pwd welcome1 -trusted_cert -cert
../root_test.cer
```

4. Add the consumer's certificate as a trusted certificate to the producer's wallet as shown in the following example:

```
orapki wallet add -wallet producer_wallet -pwd welcome1 -trusted_cert -cert
../consumer/mywebcenter.cer
```

5. Display the contents of the wallet by using the command shown in the following example:

```
orapki wallet display -wallet producer_wallet -pwd welcome1
```

10.10.1.2 Setting Up the Keystores Using Java Keystore

The Java Keystore (JKS) is the proprietary keystore format defined by Sun Microsystems. To create and manage the keys and certificates in the JKS, use the `keytool` utility that is distributed with the Java JDK.

This section discusses how to configure keystores and keys using a JKS. To enable Web Services Security (WS-Security) trusted authentication for the WSRP producer and WebCenter application, you must first configure keystores at both consumer and producer sides. Use the `keytool` utility provided with Oracle JDeveloper under *<JDEV_HOME>*`/jdk/bin` to set up keystores.

This section covers the following topics:

- Creating a Java Keystore for the Consumer
- Creating a Java Keystore for the Producer

Creating a Java Keystore for the Consumer

The consumer here is the WebCenter application, which consumes portlets generated by the remote portlet producer over WSRP. To create a Java keystore for the consumer, perform the following steps:

1. Go to *JDEV_HOME*`/jdk/bin` and open a command prompt.
2. Create a new key pair (a public key and an associated private key) using the `genKey` command as follows:

```
keytool -genkey -alias consumer -keyalg "RSA" -sigalg "SHA1withRSA" -dname
"CN=test, OU=Unknown, O=Unknown, L=Unknown, ST=California, C=US" -keypass
welcome1 -keystore consumer.jks -storepass welcome1
```

3. Display the keystore. The command shown in the following example displays the contents of the keystore:

```
keytool -list -v -keystore consumer.jks -storepass welcome1
```

4. Create a certificate request file for the signature key pair using the following command:

```
keytool -certreq -file consumer.csr -alias consumer -keystore consumer.jks
-keypass welcome1 -storepass welcome1
```

5. Request a trusted certificate from a Certificate Authority (CA).

   There are many public Certification Authorities, such as VeriSign, Thawte, Entrust, and so on. You can also run your own Certification Authority using products such as the Netscape or Microsoft Certificate Servers or the Entrust CA product for your organization.

6. Create a `root.cer` file, copy and paste the contents of the root certificate provided by the CA into this file.

7. Import the root certificate using the following command:

   ```
   keytool -import -file root.cer -keystore consumer.jks -storepass welcome1
   ```

   Repeat this step for any intermediate certificates from the CA.

8. Create a `trusted.cer` file, copy and paste the contents of the trusted certificate provided by the CA into this file.

9. Import the trusted certificate using the following command:

   ```
   keytool -import -file trusted.cer -alias consumer -keypass welcome1 -keystore
   consumer.jks -storepass welcome1
   ```

Creating a Java Keystore for the Producer

The next step of the keystore setup process is generating the Java keystore for the producer. The producer uses the public key of the consumer to verify the authenticity of the security tokens received from the consumer in the WS-Security headers of the requests it receives over its `getMarkup` interface. To do this, the producer needs a Java keystore that contains the certificate of the consumer and the root certificate used to sign it. These certificates are added to the Java keystore as trusted certificates.

To create a Java keystore for the producer, perform the following steps:

1. Go to *`JDEV_HOME`*`/jdk/bin` and open a command prompt.

2. Export the certificate from the consumer Java keystore using the following command:

   ```
   keytool -export -file producer.cer -alias consumer -keystore consumer.jks
   -storepass welcome1
   ```

3. Import the certificate you exported in the previous step into the producer Java keystore using the following command:

   ```
   keytool -import -file producer.cer -alias consumer -keystore producer.jks
   -storepass welcome1
   ```

10.10.2 Configuring the Producer

The required configuration on the producer end will vary depending on the WSRP container that you are consuming through your application. You can consume portlets from the Oracle WSRP Container running on Oracle WebCenter Suite 10.1.3.2.0.

> **Notes:** For integration with Oracle WebCenter Framework, the token profile that must be configured is Username Token with no password or SAML token.

The example in this section discusses the configuration for the Oracle WSRP Container on an Oracle WebCenter Suite 10.1.3.2.0 installation. This is the Oracle WSRP container

deployed onto Oracle Application Server release 10.1.3.2.0. This leverages WS-Security integration with OC4J.

Deploying and Configuring the Producer

Before you configure the producer for WS-Security, you must first deploy your standards-compliant portlet producer to the Oracle WebCenter Suite 10.1.3.2.0 WSRP Container by performing the steps in Section 18.9, "Deploying Your Portlet to an Application Server".

After you have deployed the producer, configure the producer for WS-Security by performing the following steps:

1. Navigate to the following URL to access the Application Server Control Console:

   `http://<host_name>.<domain>:<port>/em`

   For example:

   `http://server.domain.com:7781/em`

   To find the URL for your console, look at the `readme.txt` file. After installation, this text file is saved to the following location:

   On UNIX: *ORACLE_HOME*`/install/readme.txt`

   On Windows: *ORACLE_HOME*`\install\readme.txt`

2. Log in.

   The user name and password are those you specified during installation.

3. On the Application Server Control Console home page, navigate to the OC4J instance. For example, OC4J_WebCenter.

4. Click the **Web Services** tab.

   If you do not see any Web services listed, then it is because the Web service has not been requested yet and as a result, it is not visible in the Application Server Control Console.

5. To enable the display of the Web service interfaces in Application Server Control Console, perform the following steps:

   a. Click the **Applications** tab, then click the application. For example, `richtextportlet`.

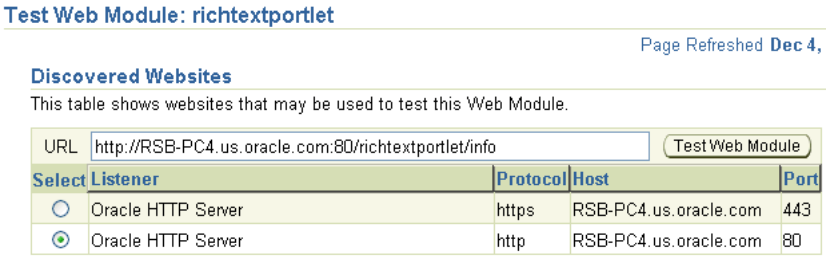
   b. On the Application page, click the web module of the producer application.

   c. On the Web Module page, click the **Test Web Module** link.

      On the Test Web Module page, you see the base URL of the producer as a standard web module.

   d. Add `/info` to the base URL as shown in Figure 10–48.

Figure 10–48 Editing the Base URL of the Producer



e. Click **Test Web Module**.

f. On the WSRP Producer Test Page, under Web Services Description Language (WSDL) URLs, click one of the links.

The WSDL of the portlet is displayed in a browser.

If you selected WSRP v1 WSDL, then in this WSDL, you should find a line that shows the location of the WSRP Base Service as shown in Example 10–26 and Figure 10–49:

Example 10–26 Base Service URL in WSRP v1 WSDL

```
http://hostname.company.com:7781/secondSampleV1/portlets/WSRPBaseService
```

Figure 10–49 Base Service URL in WSRP v1 WSDL

```
  targetNamespace="urn:oasis:names:tc:wsrp:v1:wsdl">
  <import namespace="urn:oasis:names:tc:wsrp:v1:bind" location="wsrp_v1_bindings.wsdl" />
- <wsdl:service name="WSRP_v1_Service">
  - <wsdl:port binding="bind:WSRP_v1_Markup_Binding_SOAP" name="WSRPBaseService">
      <soap:address location="http://RSB-
        PC4.us.oracle.com/richtextportlet/portlets/WSRPBaseService" />
    </wsdl:port>
  - <wsdl:port binding="bind:WSRP_v1_ServiceDescription_Binding_SOAP"
      name="WSRPServiceDescriptionService">
      <soap:address location="http://RSB-
        PC4.us.oracle.com/richtextportlet/portlets/WSRPServiceDescriptionService" />
    </wsdl:port>
  - <wsdl:port binding="bind:WSRP_v1_Registration_Binding_SOAP"
```

If you selected WSRP v2 WSDL, then in this WSDL, you should find a line that shows the location of the WSRP Markup Service as shown in Example 10–27 and Figure 10–50:

Example 10–27 Markup Service URL in WSRP v2 WSDL

```
http://hostname.company.com:7782/secportlet/portlets/WSRP_v2_Markup_Service
```

Figure 10–50 Markup Service URL in WSRP v2 WSDL

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:bind="urn:oasis:names:tc:wsrp:v2:bind" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="urn:oasis:names:tc:wsrp:v2:wsdl">
    <import namespace="urn:oasis:names:tc:wsrp:v2:bind" location="wsrp_v2_bindings.wsdl" />
  - <wsdl:service name="WSRP_v2_Service">
    - <wsdl:port binding="bind:WSRP_v2_ServiceDescription_Binding_SOAP"
        name="WSRP_v2_ServiceDescription_Service">
        <soap:address location="http://RSB-
          PC4.us.oracle.com/richtextportlet/portlets/WSRP_v2_ServiceDescription_Service" />
      </wsdl:port>
    - <wsdl:port binding="bind:WSRP_v2_Markup_Binding_SOAP"
        name="WSRP_v2_Markup_Service">
        <soap:address location="http://RSB-
          PC4.us.oracle.com/richtextportlet/portlets/WSRP_v2_Markup_Service" />
      </wsdl:port>
    - <wsdl:port binding="bind:WSRP_v2_Registration_Binding_SOAP"
        name="WSRP_v2_Registration_Service">
        <soap:address location="http://RSB-
          PC4.us.oracle.com/richtextportlet/portlets/WSRP_v2_Registration_Service" />
      </wsdl:port>
    - <wsdl:port binding="bind:WSRP_v2_PortletManagement_Binding_SOAP"
```

g. Copy one of these URLs into a browser window to access the WSRP Base or Markup Service.

You can ignore the resulting screen. The reason for requesting this page is to get `WSRPBaseService` (or Markup) to show up in the list of Web Services as shown in Figure 10–51. This service, which includes the `getMarkup` interface is the only one that must be configured for WS-Security.

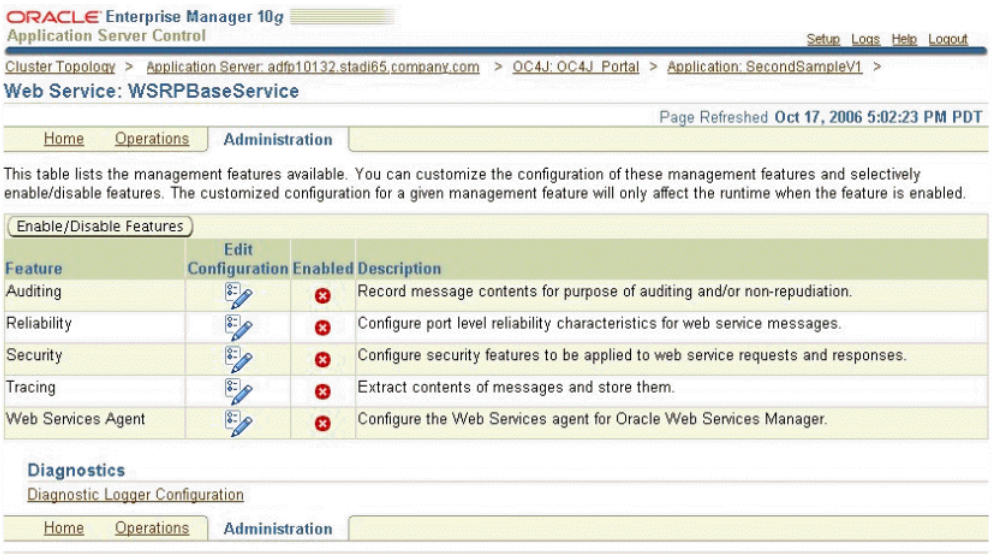
h. On the Application page, click **Web Services**.

i. Click **WSRP_v2_Markup_Service**.

j. Click the **Administration** tab.

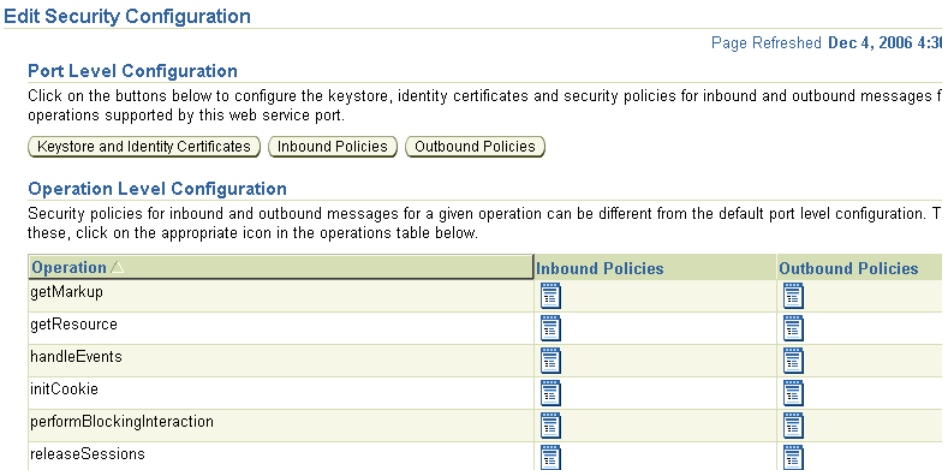
k. On the Administration tab, shown in Figure 10–51, click the **Edit Configuration** icon adjacent to the Security feature.

Figure 10–51 WSRPBaseService in Application Server Control Console



l. On the Edit Security Configuration Page, shown in Figure 10–52, you can perform port level or operation level configurations.

Figure 10–52 Edit Security Configuration Page



Configuring the Keystore

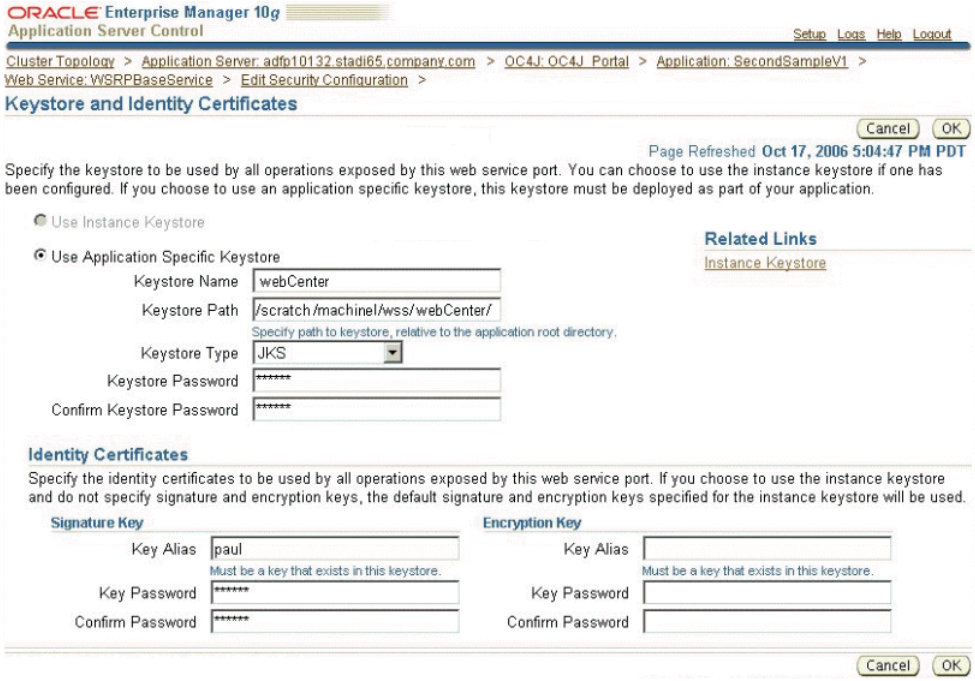
To configure the keystore, perform the following tasks:

1. In the Application Server Control Console, click the **Applications** tab, the relevant application link, the **Web Service** tab, the **WSRPBaseService** link, the **Administration** tab, and then **Enable /Disable Feature**.
2. Select **Security** from the list and click **Move** to move Security to the Enabled Features list.
3. Click **OK**.
4. Click **Edit Configuration** next to Security.
5. Click **Keystore and Identity Certificates** and select **Use Application Specific Keystore**. Enter the following values for the keystore:
6. On the Keystore and Identity Certificates page, shown in Figure 10–53, specify the following values for the keystore:

```
Keystore Name = <Any name>
Keystore Path =  <Keystore path in the file system>
Keystore Type = JKS
Keystore Password = <Keystore password>
Confirm Keystore Password = <same as above><>
```

> **Note:** Leave the Identity Certificates section empty if you are using an Oracle wallet. WS-Security code looks for the appropriate certificate in the wallet to verify the signature.

Figure 10–53 Keystore and Identity Certificates Page



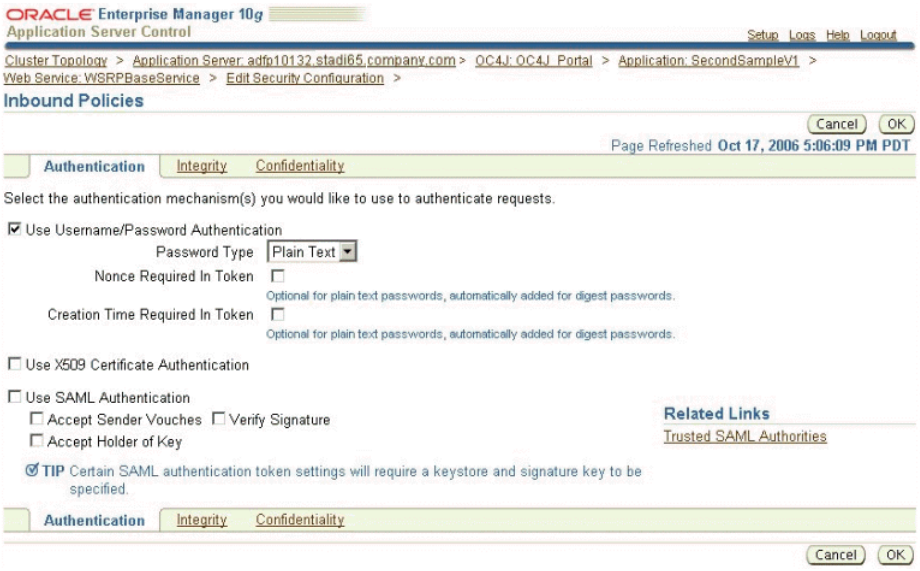
7. Click **OK**.

Configuring the Inbound Policy

To configure the inbound policy, perform the following steps:

1. In the Application Server Control Console, click the **Applications** tab, the relevant application link, the **Web Service** tab, the **WSRPBaseService** link, the **Administration** tab, and then **Enable /Disable Feature**. Select **Security** from the list and click **OK**.
2. Click the **Edit Configuration** icon displayed against the Security feature.
3. Click **Inbound Policies**.
4. On the Inbound Policies Authentication page in the Application Server Control Console, if you select **User Username /Password Authentication** as shown in Figure 10–54, then specify the password type.

Figure 10–54 Inbound Policies Page



If you are using SAML authentication, then you must select **Accept Sender Vouches** and **Verify Signature**.

5. Optionally, click the **Integrity** tab and select the **Require Message Body to be Signed** option, as shown in Figure 10–55.

Figure 10–55 Integrity Tab of Inbound Policies Page

The message body is signed at the consumer end. By selecting this option, the signature on the message body is verified at the producer end, thereby adding extra integrity protection for the request.

6. Click **OK**.

Note: When you redeploy your WebCenter application, you must configure inbound policies again.

Configuring the Web Service for Username Token without Password

Perform this configuration when the user is already authenticated and trusted, and only its existence in the identity store must be verified. You can configure the Web service for Username Token without a password by manually editing the `wsmgmt.xml` file.

> **Note:** Performing this configuration using the Application Server Control Console is not supported in this release.

The `wsmgmt.xml` file is an instance-level configuration file, which holds the entire security configuration for the Web services deployed in an OC4J instance. If an inbound element is configured, then the server interceptor uses the `wsmgmt.xml` file at run time to enforce the security policy.

To perform this configuration manually, do the following:

1. Locate the `wsmgmt.xml` file under the *`OC4J_HOME`*`/config/` directory, where *`OC4J_HOME`* is the location of OC4J under the Oracle home directory. Example 10–28 shows the code from this file.

Example 10–28 Sample wsmgmt.xml File

```
<port app="SecondSampleV1" web="SecondSampleV1" service="WSRP_v1_Service"
port="WSRPBaseService">
  <runtime enabled="security">
    <security>
      <key-store name="portal" path="/scratch/pencarna/wss/portal/portal.jks"
 type="JKS" store-pass="->SecondSampleV1.WSRPBaseService.keystore.portal.jks"/>
      <signature-key alias="paul"
key-pass="->SecondSampleV1.WSRPBaseService.key.paul"/>
      <inbound>
        <verify-username-token password-type="PLAINTEXT" require-nonce="false"
require-created="false"/>
      </inbound>
    </security>
  </runtime>
</port>
```

2. In this file, change the following line:

   ```
   <verify-username-token password-type="PLAINTEXT" require-nonce="false"
   require-created="false"/>
   ```

   To:

   ```
   <verify-username-token>
     <property name="username.token.allow.nopassword" value ="true"/>
   </verify-username-token>
   ```

   The value of the `username.token.allow.nopassword` Boolean property determines whether the Web service will authenticate a username token without requiring a password.

3. Save the `wsmgmt.xml` file.

10.10.3 Configuring the Consumer

The consumer of the WSRP portlets is the WebCenter application. This section describes the configuration steps needed on the WebCenter application to include WS-Security headers in the portlet requests.

This section covers the following topics:

- Transferring Keystore Information to the Consumer System
- Updating the Keystore Path
- Configuring the Consumer for a Username Token Profile
- Configuring the Consumer for a SAML Token Profile

Transferring Keystore Information to the Consumer System

To enable your WebCenter Application to use the keystore, the consumer keystore must be moved to the system that is running the application.

To move the consumer keystore, perform the following steps:

1. Navigate to the `C:\ca\consumer\consumer_wallet` directory.
2. FTP or copy the `ewallet.p12` file to the system that is running the WebCenter Application.

Updating the Keystore Path

After you deploy the WebCenter application to an Oracle Application Server instance, you must update the keystore path on this instance to use an absolute path. For example, `/scratch/machine1/product/10.1.3.2.0/OracleAS_1/j2ee/OC4J_WebCenter/applications/myWSRPApp/adf/META-INF/webCenter.jks`.

You can use the `setCredential` operation in the Credentials MBean in Application Server Control Console to update the keystore path. The Credentials MBean is an application MBean used to update secure properties related to a connection. Perform the `setCredential` operation by following the steps in Section 12.5, "Updating Credentials in a Deployed Application".

Restart the OC4J instance after updating the keystore path.

Configuring the Consumer for a Username Token Profile

You can define security settings for a user name token profile at the consumer end while registering the WSRP producer with your WebCenter application using Oracle JDeveloper.

To register the WSRP producer with your WebCenter application and to define security, perform the following steps:

1. In the Applications Navigator in Oracle JDeveloper, right-click the project under which to create the producer and select **New** from the context menu.
2. In the New Gallery dialog box, under Categories, expand the Web Tier node and select **Portlets**.
3. Under **Items**, select **WSRP Producer Registration**.
4. Click **OK**.
5. Step through the Register WSRP Portlet Producer Wizard by specifying the following:

a. On Step 1, specify a unique name for the connection.

b. On Step 2, specify the producer's URL Endpoint.

c. On Step 3, accept the Default Execution Time value of 60.

d. On Step 4, specify the following:

   – Token Profile: `Username Token`

   – Default User: <*user_ name*>

   – XML Signature: `Binary Security Token`

e. On Step 5, specify the following:

   – Store Path: Click **Browse** to navigate and select the keystore you created in Section 10.10.1.2, "Setting Up the Keystores Using Java Keystore".

   – Store Password: <*Keystore password*>

   – Store Type: <*Populated automatically*>

   – Signature Key Alias: Select from the populated aliases.

   – Signature Key Password: <*Password for the certificate alias*>

   See Section 4.3.1.1, "Registering WSRP Portlet Producers" for a description of these fields.

f. Click **Finish** to complete the registration.

6. Create a new JSF JSP page, `UNTPage.jspx`, under the ViewController project.

7. Select **Automatically Expose UI Components in a New Managed Bean** for the page.

8. Use the following libraries:

   - ADF Faces Components
   - ADF Faces HTML
   - ADF Portlet Components
   - Customizable Components
   - JSF Core
   - JSF HTML

9. In the Component Palette, select the producer you registered earlier and drop a portlet inside the form (use the Structure pane to verify it is inside the form).

10. Optionally, secure the page by performing the steps in Section 10.2.3.2, "Securing Pages in Your Application".

11. Run the page and log in using the appropriate credentials. You should be able to see the portlet.

Configuring the Consumer for a SAML Token Profile

You can define security settings for a SAML token profile at the consumer end while registering the WSRP producer with your WebCenter application using Oracle JDeveloper.

To register the WSRP producer with your WebCenter application and to define security, perform the following steps:

1. In the Applications Navigator in Oracle JDeveloper, right-click the project under which to create the producer and select **New** from the context menu.
2. In the New Gallery dialog box, under Categories, expand the Web Tier node and select **Portlets**.
3. Under **Items**, select **WSRP Producer Registration**.
4. Click **OK**.
5. Step through the Register WSRP Portlet Producer Wizard by specifying the following:
   a. On Step 1, specify a unique name for the connection.
   b. On Step 2, specify the producer's URL Endpoint.
   c. On Step 3, accept the Default Execution Time value of 60.
   d. On Step 4, specify the following:
      - Token Profile: `SAML Token`
      - Default User: <*user_ name*>
      - Issuer Name: `www.oracle.com`
      - XML Signature: `Binary Security Token`
   e. On Step 5, specify the following:
      - Store Path: Click **Browse** to navigate and select the keystore you created in Section 10.10.1.2, "Setting Up the Keystores Using Java Keystore".
      - Store Password: <*Keystore password*>
      - Store Type: <*Populated automatically*>
      - Signature Key Alias: Select from the populated aliases.
      - Signature Key Password: <*Password for the certificate alias*>

      See Section 4.3.1.1, "Registering WSRP Portlet Producers" for a description of these fields.
   f. Click **Finish** to complete the registration.
6. Create a new JSF JSP page, `UNTPage.jspx`, under the ViewController project.
7. Select **Automatically Expose UI Components in a New Managed Bean** for the page.
8. Use the following libraries:
   - ADF Faces Components
   - ADF Faces HTML
   - ADF Portlet Components
   - Customizable Components
   - JSF Core
   - JSF HTML
9. In the Component Palette, select the producer you registered earlier and drop a portlet inside the form (use the Structure pane to verify it is inside the form).
10. Optionally, secure the page by performing the steps in Section 10.2.3.2, "Securing Pages in Your Application".

11. Run the page and log in using the appropriate credentials. You should be able to see the portlet.

10.11 Configuring a WebCenter Application to Use LDAP and Single Sign-On

In the production environment, applications are deployed to OC4J in Oracle Application Server. Typically, the OC4J instance would use an LDAP-based Oracle Internet Directory as the repository for Oracle Identity Management and Oracle Single Sign-On for authentication. Therefore, it is necessary that you configure your WebCenter application to use Oracle Internet Directory or an external LDAP provider as the identity store and to use Single Sign-On for authentication.

This configuration is done while deploying the application to an OC4J in Oracle Application Server. For information about the steps involved in deploying your WebCenter application, see Figure 12–4.

This section covers the following topics:

- Section 10.11.1, "Configuring a WebCenter Application to Use Oracle Internet Directory"
- Section 10.11.2, "Configuring a WebCenter Application to Use Oracle Single Sign-On"
- Section 10.11.3, "Configuring a WebCenter Application to Use Java Single Sign-On"
- Section 10.11.4, "Configuring a WebCenter Application to Use an External LDAP Provider"
- Section 10.11.5, "Configuring a WebCenter Application to Use Oracle Access Manager"

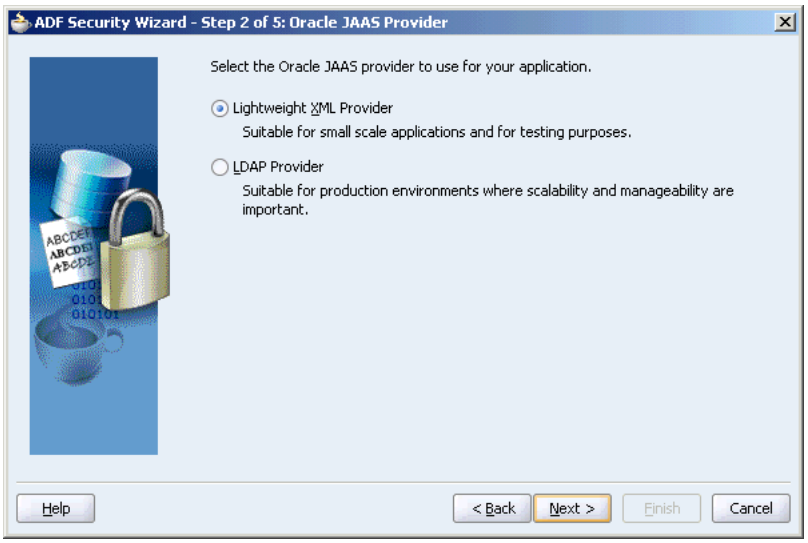
10.11.1 Configuring a WebCenter Application to Use Oracle Internet Directory

You can configure your WebCenter application to use Oracle Internet Directory in either of the following ways:

- Run the ADF Security Wizard and select the **default Lightweight XML Provider** on the Oracle JAAS Provider screen, shown in Figure 10–56, and then configure the application at run time using Application Server Control Console to use an LDAP-based Provider, like Oracle Internet Directory.
- Run the ADF Security Wizard and select **LDAP Provider** on the Oracle JAAS Provider screen, shown in Figure 10–56, to start with an LDAP-based provider directly. In this case, it is expected that you use the same LDAP provider at run time that you used in the development environment.

> **Note:** This option will only work for authentication. If you are using Oracle ADF authorization and plan to use the authorization screen to establish permission grants for the security aware components, then this setting does not result in the policies being stored automatically in the LDAP server. Therefore, if you are using Oracle ADF authorization, then you must use the XML provider during development and then configure for the LDAP provider during or after deployment.

Figure 10–56 Oracle JAAS Provider Screen in the ADF Security Wizard



The steps to configure your WebCenter application to use Oracle Internet Directory are as follows:

- Developing a WebCenter Application and Setting Up Oracle ADF Security
- Packaging Your WebCenter application
- Predeploying Your WebCenter application
- Associating the OC4J instance with Oracle Internet Directory
- Deploying Your WebCenter Application
- Changing the Security Provider for Your WebCenter Application to Use Oracle Internet Directory
- Configuring Role Mapping
- Restarting OC4J
- Migrating Security and Application Roles

Developing a WebCenter Application and Setting Up Oracle ADF Security

Develop a WebCenter application and set up Oracle ADF Security for this application using the ADF Security Wizard.

Packaging Your WebCenter application

In this step, all the required files are packaged in a standard J2EE format and directory structure, as an EAR file. Follow the steps in Section 12.2.1, "Packaging Your WebCenter Application" for performing this task.

Predeploying Your WebCenter application

In this step, the development references contained in the file are modified to be the target references. The targeted EAR file is created using the Predeployment tool packaged with Oracle JDeveloper. Follow the steps in Section 12.2.2.2, "Predeploying WebCenter Applications and JCR Adapter-based Applications" for performing this task.

Associating the OC4J instance with Oracle Internet Directory

Prior to deploying your WebCenter application to an OC4J instance in Oracle Application Server, you must associate Oracle Internet Directory with the OC4J instance.

Perform this task only if the OC4J instance is not already associated with Oracle Internet Directory. Use the Application Server Control Console to associate your OC4J instance with an instance of the LDAP-based Oracle Internet Directory, the repository for Oracle Identity Management. To do this, perform the following steps:

1. In the Application Server Control Console, navigate to the OC4J Home page for your instance and click the **Administration** tab.
2. In the resulting Administration page, choose the Identity Management task (one of the Security tasks).
3. In the resulting Identity Management page, choose **Configure**.

   This assumes no Oracle Internet Directory instance was previously associated with the OC4J instance, so that the Oracle Internet Directory host name and port are listed as `not configured`. If a different Oracle Internet Directory instance was previously associated with this OC4J instance, then see section titled "Changing the Oracle Internet Directory Association" in the *Oracle Containers for J2EE Security Guide*.
4. In the resulting Configure Identity Management: Connection Information page, do the following:
   - Specify the fully qualified host name for the Oracle Internet Directory instance (`myoid.oracle.com`, for example).
   - Specify the distinguished name for the Oracle Internet Directory user, such as cn=orcladmin. Each user in a directory must have a unique distinguished name. The user specified here must belong to the `iASAdmins` role in the Oracle Internet Directory instance.
   - Specify the password for the Oracle Internet Directory user. This will also be set as the default password for the `oc4jadmin` user created in Oracle Internet Directory (unless the `oc4jadmin` account had previously been created, due to associating a different OC4J instance with the `Oracle Internet Directory` instance).
   - Specify whether to use SSL connections or non-SSL connections to the Oracle Internet Directory instance, and the appropriate port to use. The port for SSL is typically `636`; for non-SSL it is typically `389`. (To change the SSL or port setting later, you would have to redo the OC4J-Oracle Internet Directory association, as described in the section in the OC4J Security Guide referenced in the previous step.)
   - When you are done, go to the next page.
5. In the Configure Identity Management: Application Server Control page, you can optionally specify Oracle Identity Management as the security provider for Application Server Control. (If you do this, then only users and roles defined in the Oracle Internet Directory instance will be able to access Application Server Control.)

   When you are done, go to the next page.
6. In the Configure Identity Management: Deployed Applications page, you can optionally specify Oracle Internet Directory (actually, Oracle Identity

Management), with or without SSO, as the security provider for each deployed application in the OC4J instance.

When you are done, choose **Configure**, and restart the instance when prompted. This completes the OC4J-Oracle Internet Directory association process and takes you back to the Identity Management page.

Deploying Your WebCenter Application

You are now ready to deploy your WebCenter application to an OC4J instance using Application Server Control Console. Follow the steps in Section 12.2.3, "Deploying Your WebCenter Application Using Application Server Control Console" to perform this task.

> **Note:** You can change the security provider to Oracle Internet Directory during deployment as one of the points in the deployment plan. In case this is not done, the following section describes how to change the security provider to Oracle Internet Directory after deployment.

At this point your WebCenter application will be running, but as the policy information has not been migrated yet, all the security features will not be working.

Changing the Security Provider for Your WebCenter Application to Use Oracle Internet Directory

The next step is to add Oracle Internet Directory as the security provider for your WebCenter application. To do this, perform the following steps:

1. In the Application Server Control Console, navigate to the OC4J instance that contains your WebCenter application, select your application, and click the **Administration** tab.
2. In the resulting Administration page, choose the **Security Provider** task.
3. Click **Change Security Provider**.
4. From the **Security Provider Type** list, select **Oracle Identity Management Security Provider**.
5. Click **OK**.
6. Restart your WebCenter application.

Migrating Security and Application Roles

The tasks involved in migrating security and application roles to Oracle Internet Directory are as follows:

- Use the JAZN Migration tool to migrate policy information from the `app-jazn-data.xml` file packaged as part of the application EAR file to an LDIF file for the LDAP-based Oracle Internet Directory.

  This tool facilitates the transition from the development environment to a deployed production environment.

  Example 10–29 describes the syntax to facilitate migration from XML to LDAP.

Example 10–29 Syntax for XML to LDAP Migration

```
java oracle.security.jazn.tools.JAZNMigrationTool
-D binddn -w passwd -h ldaphost -p ldapport -st xml -dt ldap -sf
```

```
<source file> -df <dest file> -m policy|realm|all(default)
```

See Section 12.2.4.3, "Using the JAZN Migration Tool" for more information about using the JAZN Migration tool.

- Modify the LDIF file to remove any entries that already exist in Oracle Internet Directory. User passwords have to comply with the Oracle Internet Directory password policies.
- Use the `ldapmodify` command to add the policy information in the LDIF file to Oracle Internet Directory. Follow the steps in Section 12.2.4.4, "Using the ldapmodify Command-Line Tool" for performing this task.

> **See Also:** *Oracle Internet Directory Administrator's Guide* for more information about the `ldapmodify` command.

Configuring Role Mapping

If the development environment was configured with the corresponding enterprise role references and the subsequent migration of JAZN information was performed using the JAZN Migration tool in `all` mode, then the deployed application will have the correct role mapping available to it.

However, if the development environment used temporary roles to map policies and security constraints, then you must convert these role references to the appropriate enterprise role prior to deploying these policies. For example, if the developer used the enterprise role *mymanagers* for a role that equates to the *hr-directors* role on the deployment server, then the `app-jazn-data.xml` file and the `orion-application.xml` file must be updated accordingly. In `app-jazn-data.xml`, you change the policy information before you run the JAZN Migration tool as shown in Example 10–30.

Example 10–30 Grants in an app-jazn-data.xml File

```
<grant>
  <grantee>
    <principals>
      <principal>
        <realm-name>jazn.com</realm-name>
        <type>role</type>
        <class>oracle.security.jazn.spi.xml.XMLRealmRole</class>
        <name>hr-directors</name>
      </principal>
    </principals>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.adf.share.security.authorization.RegionPermission</class>
      <name>view.pageDefs.welcomePageDef</name>
      <actions>customize,personalize,view</actions>
    </permission>
  </permissions>
</grant>
```

Likewise, the J2EE security to enterprise role mapping defined in the `orion-application.xml` file (located in the *ORACLE_HOME*`/j2ee/home/application-deployments/<`*Application_Name*`>` directory), must be updated to reflect the deployment server's enterprise roles. For example, you can change the mapping of the J2EE security role *hr_managers* from the

role used in development to the production role *hr-directors* as shown in Example 10–31.

Example 10–31 Security Role Mapping in the orion-application.xml File

```
<security-role-mapping name="hr_managers">
  <group name="hr-directors" />
</security-role-mapping>
```

Restarting OC4J

Restart your OC4J instance.

Your WebCenter application is now ready for use.

10.11.2 Configuring a WebCenter Application to Use Oracle Single Sign-On

Oracle Single Sign-On authenticates users against Oracle Internet Directory. Therefore, to configure your WebCenter application to use Oracle Single Sign-On, you must first configure the OC4J instance to use Oracle Internet Directory by performing the steps described in the section, "Associating the OC4J instance with Oracle Internet Directory".

To use Oracle Single Sign-On with your application, perform the following steps:

1. In the Application Server Control Console, navigate to the OC4J Home page for your instance and click the **Administration** tab.
2. In the resulting Administration page, choose the **Security provider** task.
3. Select the **Enable SSO Authentication** option.
4. Perform all the steps described in the section titled "Configure SSO (Optional)" in the *Oracle Containers for J2EE Security Guide*.

   While performing the steps in the OC4J Security Guide, under the first step, "Run the SSO Registration Tool", use the following example to run the SSO Registration Tool:

   ```
   % $ORACLE_HOME/sso/bin/ssoreg.sh -oracle_home_path $ORACLE_HOME \
   -site_name myhost.mydomain.com -config_mod_osso TRUE \
   -mod_osso_url http://myhost.mydomain.com:7777 -remote_midtier \
   -config_file $ORACLE_HOME/Apache/Apache/conf/osso/midtier_osso.conf
   ```

   A new file will be created with the name you specify, containing the SSO partner application configuration for the middle tier you are associating with SSO. Copy this file to the middle tier you are configuring.

   > **Note:** Using `osso.conf` as the configuration file name will result in an error because you should not overwrite the existing `osso.conf` file.
   >
   > It is recommend that the file name indicate the host and port of the middle tier. For example, `midtier1_7779_osso.conf`.

5. Click **OK**.

10.11.3 Configuring a WebCenter Application to Use Java Single Sign-On

The Java Single Sign-On (SSO) solution does not rely on additional required infrastructureS like other Oracle Single Sign-On products. Java SSO is based on the OracleAS JAAS Provider Identity Management framework.

To configure your WebCenter application to use Java SSO, you must start the Java SSO application in one of the OC4J instances, and then configure Java SSO and partner applications through Application Server Control Console.

For the actual steps to be performed and information about how to set this up in a distributed environment, see the chapter titled "OC4J Java Single Sign-On" in the *Oracle Containers for J2EE Security Guide*.

You can configure Java SSO to work with Oracle Internet Directory or any external LDAP provider. You can do this using Application Server Control Console, or by setting the `<jazn>` element in `orion-application.xml` file to `LDAP`. See the chapter titled "OC4J Java Single Sign-On" in the *Oracle Containers for J2EE Security Guide* for the actual steps to be performed.

10.11.4 Configuring a WebCenter Application to Use an External LDAP Provider

You can configure your WebCenter application to use an external LDAP provider as the identity store. To do this, perform the following steps:

1. Configure your WebCenter application to use an LDAP provider in either of the following ways:
   - Run the ADF Security Wizard and select the **default Lightweight XML Provider** on the Oracle JAAS Provider screen, shown in Figure 10–56, and then configure the application at run time using Application Server Control Console to use an LDAP-based Provider, like Oracle Internet Directory.
   - Run the ADF Security Wizard and select **LDAP Provider** on the Oracle JAAS Provider screen, shown in Figure 10–56, to start with an LDAP-based provider directly. In this case, it is expected that you use the same LDAP provider at run time that you used in the development environment.
2. Develop a WebCenter application and set up Oracle ADF Security for this application using the ADF Security Wizard.
3. Package your WebCenter application by performing the steps in Section 12.2.1, "Packaging Your WebCenter Application".
4. Predeploy your WebCenter application by performing the steps in Section 12.2.2, "Predeploying Your WebCenter Application".
5. Follow the steps in Section 12.2.3, "Deploying Your WebCenter Application Using Application Server Control Console" to deploy your WebCenter application.
6. Add the external LDAP provider as the security provider by performing the steps in "Changing the Security Provider for Your WebCenter Application to Use Oracle Internet Directory" and selecting **Oracle Security Provider for 3rd Party LDAP Server** from the Security Provider Type list.
7. If you are using an LDAP provider for the first time, then configure role mapping by performing the steps in "Configuring Role Mapping".
8. Restart OC4J.
9. Use the JAZN Migration tool to migrate policy information from the `app-jazn-data.xml` file packaged as part of the application EAR file to an XML file for the external LDAP provider.

Example 10–32 describes the syntax to facilitate migration of the `app-jazn-data.xml` file to a `system-jazn-data.xml` file for the external LDAP provider.

Example 10–32 Sample XML to XML Migration

```
java oracle.security.jazn.tools.JAZNMigrationTool -st xml -dt xml
-sf ORACLE_HOME/j2ee/OC4J_
Apps/applications/webCenterArchive1/adf/META-INF/app-jazn-data.xml
-df ORACLE_HOME/j2ee/OC4J_Apps/config/system-jazn-data.xml -m all
```

See Section 12.2.4.3, "Using the JAZN Migration Tool" for more information.

This is to migrate policy information only (to the target `system-jazn-data.xml` file)—you cannot use JAZN Migration tool to directly migrate user and role information into the external LDAP provider, because external LDAP providers store only user and group information. Policies and other information is stored in XML files. The JAZN Migration tool cannot create an LDIF file appropriate for external LDAP providers. It is capable of creating LDIF files (with the appropriate with the directory information tree and schema) only for Oracle Internet Directory.

10. Manually edit the migrated policy information to use `LDAPPrincipal`.

    After you migrate the policy information as discussed in the previous step, the resulting entries in the target `system-jazn-data.xml` file will have grantees referencing the `XMLRealmRole` principal class. For use with an external LDAP provider, these entries must be updated to reference the `LDAPPrincipal` class instead. For example, assume `system-jazn-data.xml` includes the following principal configuration in a grantee element:

    ```
    <principal>
    <realm-name>jazn.com</realm-name>
    <type>role</type>
    <class>oracle.security.jazn.spi.xml.XMLRealmRole</class>
    <name>jdoe</name>
    </principal>
    ```

    This must be updated manually to remove the `<realm-name>` and `<type>` elements and to specify `LDAPPrincipal` instead of `XMLRealmRole`, as follows:

    ```
    <principal>
    <class>oracle.security.jazn.realm.LDAPPrincipal</class>
    <name>jdoe</name>
    </principal>
    ```

    If the name has a realm prefix such as `jazn.com/jdoe`, then the realm prefix must be removed so that the name is simply `jdoe`.

11. Manually create any necessary user and role accounts in the external LDAP provider. Ensure that the user and role names you create conform to the principals referenced in the policy configuration you migrated to the target server in Step 9.

    See *Oracle Internet Directory Administrator's Guide* for information about performing this task.

    See Also: See the chapter titled "Authorization in OC4J" in the *Oracle Containers for J2EE Security Guide* to know more about the JAAS mode and how to configure it.

10.11.5 Configuring a WebCenter Application to Use Oracle Access Manager

To configure your WebCenter application to use Oracle Access Manager, formerly known as Oracle COREid Access and Identity, see the chapter titled "Oracle Access Manager" in the *Oracle Containers for J2EE Security Guide*.

When switching from the file-based security provider to Oracle Access Manager, the policies in `system-jazn-data.xml` must be updated as specified in the "Oracle Access Manager" chapter of the *Oracle Containers for J2EE Security Guide*.

Additionally, consider the following:

- The `grantee` element should not have `realm-name` and `type` subelements. If the name is prefixed with a realm, then it must be removed. For example, `jazn.com/users` must be changed to `users`.
- References to the class `oracle.security.jazn.spi.xml.XMLRealmRole` must be changed to `oracle.security.jazn.realm.CoreIDPrincipal`.

The grant shown in Example 10–33 must be updated to the one shown in Example 10–34.

Example 10–33 system-jazn-data.xml File Content with realm-name and type Subelements

```
<grant>
   <grantee>
      <principals>
         <principal>
            <realm-name>jazn.com</realm-name>
            <type>role</type>
            <name>page-viewer</name>
         </principal>
      </principals>
   </grantee>
   <permissions>
      <permission>
         <class>oracle.adf.share.security.authorization.MethodPermission</class>
         <name>testjcr.getItems</name>
         <actions>invoke</actions>
      </permission>
      <permission>
         <class>oracle.adf.share.security.authorization.MethodPermission</class>
         <name>testjcr.advancedSearch</name>
         <actions>invoke</actions>
      </permission>
      <permission>
         <class>oracle.adf.share.security.authorization.RegionPermission</class>
         <name>view.pageDefs.PortalExtAppPageDef</name>
         <actions>view</actions>
      </permission>
      . . .
   </permissions>
</grant>
```

Example 10–34 system-jazn-data.xml File for Use with Oracle Access Manager

```
<grant>
   <grantee>
      <principals>
         <principal>
```

```
            <class>oracle.security.jazn.realm.CoreIDPrincipal</class>
            <name>page-viewer</name>
         </principal>
      </principals>
   </grantee>
   <permissions>
      <permission>
         <class>oracle.adf.share.security.authorization.MethodPermission</class>
         <name>testjcr.getItems</name>
         <actions>invoke</actions>
      </permission>
      <permission>
         <class>oracle.adf.share.security.authorization.MethodPermission</class>
         <name>testjcr.advancedSearch</name>
         <actions>invoke</actions>
      </permission>
      <permission>
         <class>oracle.adf.share.security.authorization.RegionPermission</class>
         <name>view.pageDefs.PortalExtAppPageDef</name>
         <actions>view</actions>
      </permission>
      . . .
   </permissions>
</grant>
```

In addition to performing the configuration tasks described in the *Oracle Containers for J2EE Security Guide*, you must perform the following tasks:

1. If you are using an LDAP provider for the first time, then configure role mapping by performing the steps in "Configuring Role Mapping".
2. Restart OC4J.
3. Use the JAZN Migration tool to migrate policy information from the `app-jazn-data.xml` file packaged as part of the application EAR file to an XML file for the external LDAP provider.

   Example 10–35 describes the syntax to facilitate migration of the `app-jazn-data.xml` file to a `system-jazn-data.xml` file for the external LDAP provider.

Example 10–35 Sample XML to XML Migration

```
java oracle.security.jazn.tools.JAZNMigrationTool -st xml -dt xml
-sf ORACLE_HOME/j2ee/OC4J_
Apps/applications/webCenterArchive1/adf/META-INF/app-jazn-data.xml
-df ORACLE_HOME/j2ee/OC4J_Apps/config/system-jazn-data.xml -m all
```

See Section 12.2.4.3, "Using the JAZN Migration Tool" for more information.

11

Working Productively in Teams

This chapter contains advice for working on Oracle WebCenter Suite projects using Oracle JDeveloper and CVS, the integrated source control system. Before reading this chapter, you should read the chapter in *Oracle Application Development Framework Developer's Guide* regarding team development with CVS.

11.1 General Advice for Using CVS on WebCenter Applications

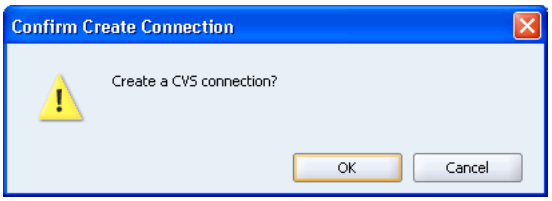
Working with CVS on a WebCenter application is similar to any other development environment with a source control system. However, it is quite different from the in-place editing paradigm used in Oracle Application Server Portal (OracleAS Portal). You should carefully review the Oracle JDeveloper online help system and *Oracle Application Development Framework Developer's Guide* for more information about CVS.

Getting Started

If you do not already have your application files source controlled in CVS, then you must import them into CVS before you can get started. To import the application files into CVS quickly, do the following:

1. Open Oracle JDeveloper.
2. Select **View**, **CVS Navigator** to bring up the CVS Navigator.
3. Select your application in the Applications Navigator and under **Versioning**, select **Import Module**. If you already have a CVS connection, then you will be taken directly to the Import to CVS wizard. If you do not already have a CVS connection, then you will be prompted to create one as follows:
   a. If the Confirm Create Connection dialog box (Figure 11–1) appears, then click **OK**.

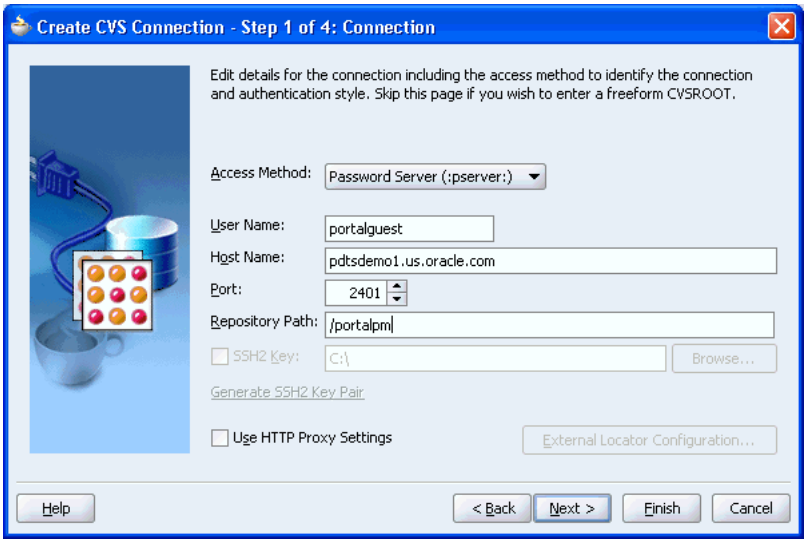
Figure 11–1 Confirm Create Connection Dialog Box



b. If you are on the Welcome page of the wizard, then click **Next** to display the Connection page.

c. On the Connection page (Figure 11–2), enter your connection information for CVS.

Figure 11–2 Connection Page of Create CVS Connection Wizard



d. Click **Next**.

e. On the Root page (Figure 11–3), enter the **Value of CVSROOT**.

Figure 11–3 Root Page of Create CVS Connection Wizard

f. Click **Next**.

g. On the Test page (Figure 11–4), click **Test Connection**. The Log In to CVS dialog box is displayed, as shown in Figure 11–5.

Figure 11–4 Test Page of Create CVS Connection Wizard

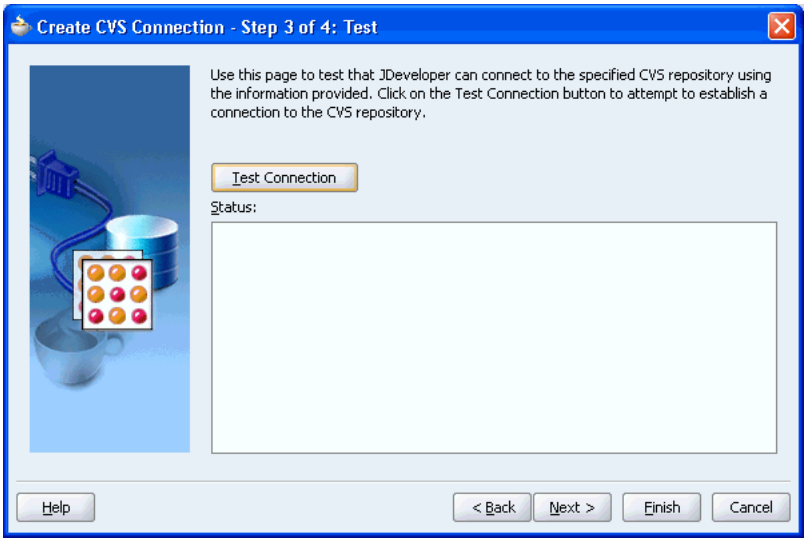
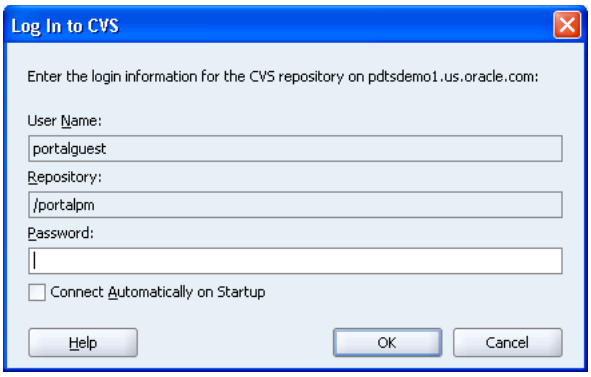


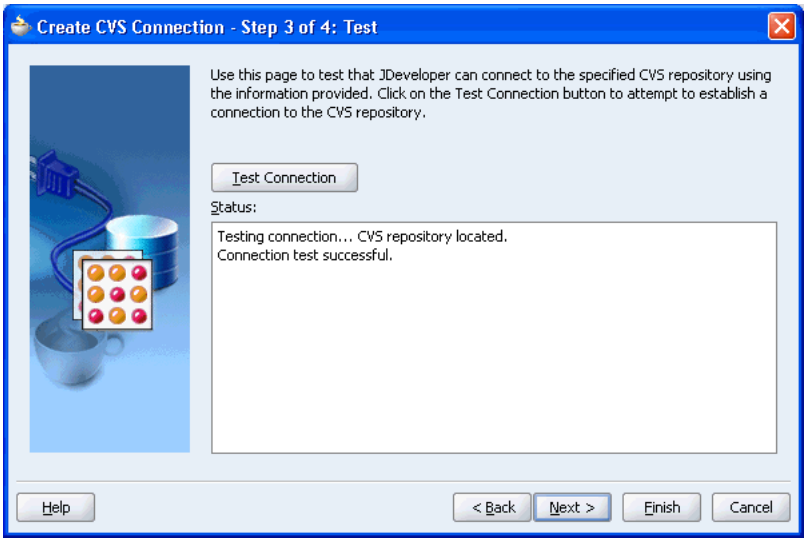
Figure 11–5 Log In to CVS Dialog Box



h. Enter your password and click **OK**.

i. If the test is successful, then you should see something similar to Figure 11–6.

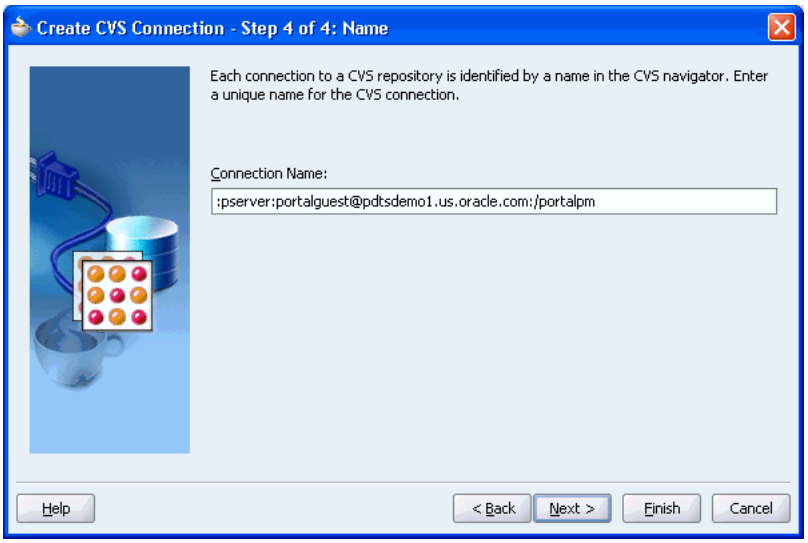
Figure 11–6 Successful Test Page of Create CVS Connection Wizard



j. Click **Next**.

k. On the Name page, enter a value for **Connection Name**. The default name is the CVSROOT value, but you can change it to something more descriptive, as shown in Figure 11–7.

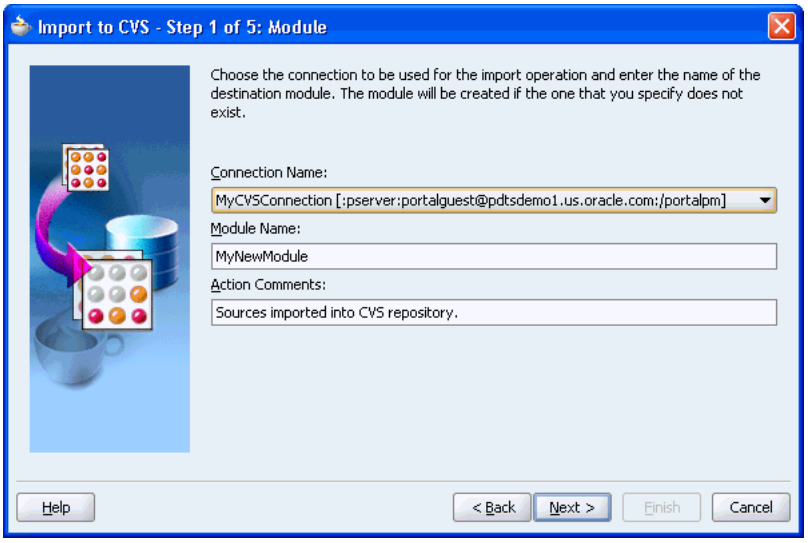
Figure 11–7 Name Page of the Create CVS Connection Wizard



I. Click **Finish**. The Import to CVS wizard is displayed.

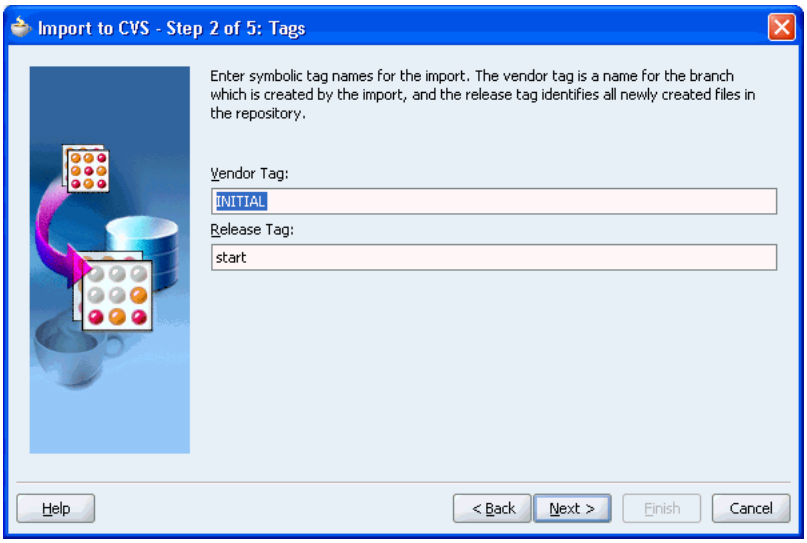
4. Click **Next** on the Welcome page of the wizard to display the Module page.
5. On the Module page, choose **Connection Name** from the list.
6. Enter a **Module Name** for the new module, as shown in Figure 11–8.

Figure 11–8 Module Page of Import to CVS Wizard



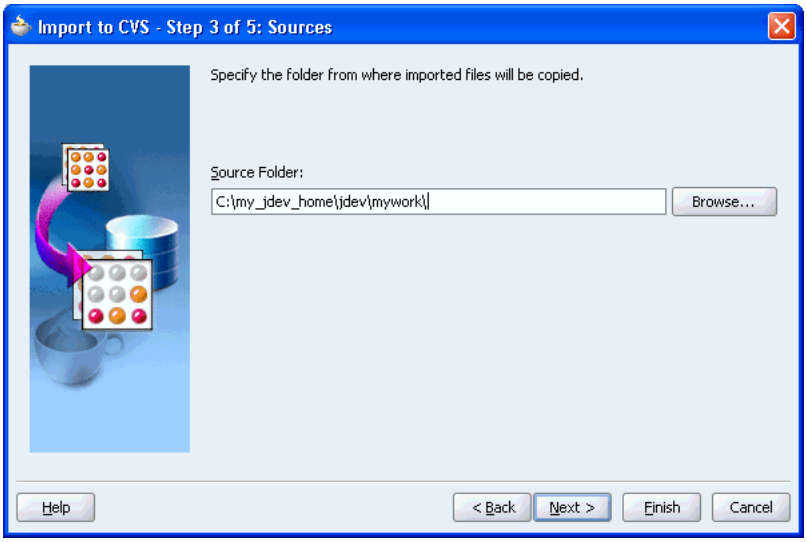
7. Click **Next**.
8. The Tags page specifies the tag names for the creation of the project. Leave the default values unchanged for the purposes of this example, as shown in Figure 11–9.

Figure 11–9 Tags Page of Import to CVS Wizard



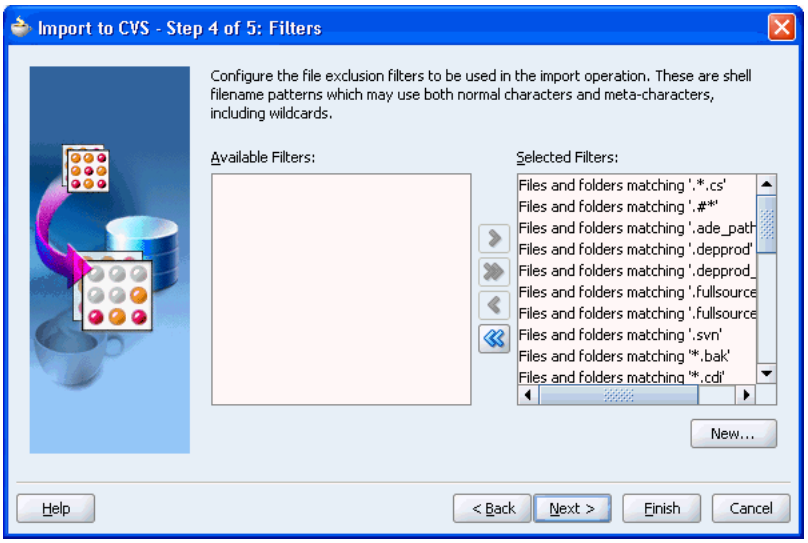
9. Click **Next**.
10. The Sources page specifies the location of the application. Leave the default value unchanged for the purposes of this example, as shown in Figure 11–10.

Figure 11–10 Sources Page of the Import to CVS Wizard



11. Click **Next**.
12. The Filters page specifies filters to exclude files from the import operation. Leave the default filters unchanged for the purposes of this example, as shown in Figure 11–11.

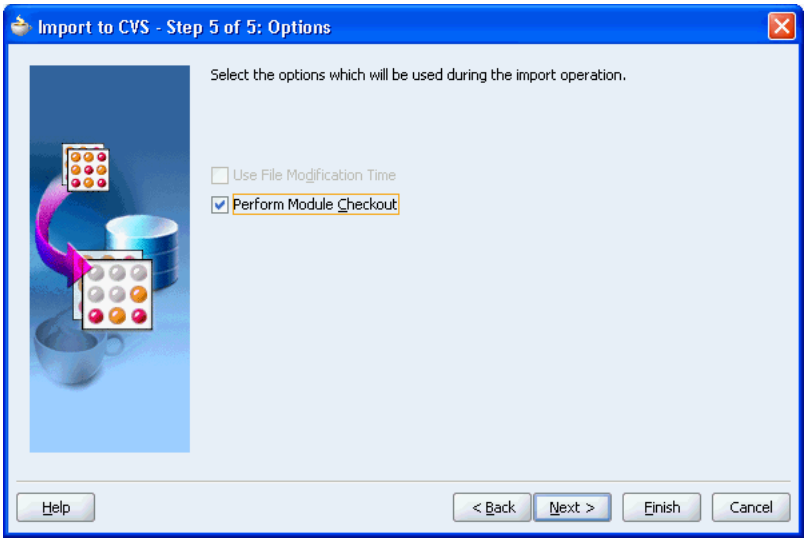
Figure 11–11 Filters Page of Import to CVS Wizard



13. Click **Next**.

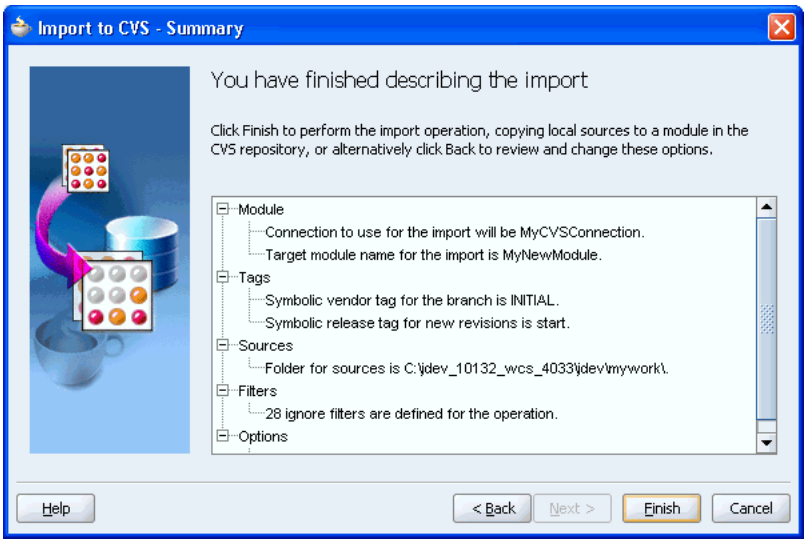
14. The Options page lets you specify whether you want to immediately check out the newly created module and begin versioning your project. Leave the default choice unchanged for the purposes of this example, as shown in Figure 11–12.

Figure 11–12 Options Page of Import to CVS Wizard



15. Click **Next**. A summary of your chosen settings for this import operation is displayed for your review, as shown in Figure 11–13.

Figure 11–13 Summary Page of Import to CVS Wizard



16. Click **Finish** to execute the import operation. Other members of the team will now be able to check out the newly created module and commit changes.

11.2 Advice for WebCenter Application Files in CVS

One of the most important aspects of working with any source control system is understanding which files particular actions will touch. Without this knowledge, you may inadvertently corrupt the source by checking in or out either too few or too many files given the actions you are performing on the project. The tips in this section aim to help you understand what files are needed for the main actions you will perform in building a WebCenter application.

11.2.1 Files Associated With Common Objects

The main objects with which you work in a WebCenter application are as follows:

- Pages
- Portlets
- Producers
- Data controls

Each of these are fairly complex objects, made up of several different metadata files. For information about page metadata files, see *Oracle Application Development Framework Developer's Guide*. For information about portlet and producer metadata files, see Appendix C, "Files for WebCenter Applications".

11.2.2 Developer Actions Affecting Metadata Files

Table 11–1 lists the major developer actions and the files that are affected by these actions. You must take this information into account while managing your files in CVS.

Table 11–1 Files Affected by Developer Actions

Action	Files Affected
Registering a producer	`connections.xml` is updated for a connection to the producer. `adf-config.xml` is updated to configure the metadata repository and the external application configuration. *`projectname`*`.jws` is updated to include the libraries needed for any WebCenter application. *`applicationroot`*`/mds/` adds a number of files pertaining to the registered producer. `WEB-INF/lib` is the place to which portlet run time libraries are copied.
De-registering a producer	The producer is gone from the application, but you may need to manually remove the metadata files from CVS. Everything else should be left alone in case they have other producers which rely on the `web.xml`, `adf-config.xml`, `connections.xml`.
Placing a portlet on a page	*`pagename`*`.jspx` is updated to add the `<adfp:portlet>` tag. The `.jspx` is also transformed. JSF Faces tags are replaced with Oracle ADF faces tags. *`pagename`*`PageDef.xml` is updated with `<portlet>` tag. `WEB-INF/web.xml` is updated to include the portlet servlets and filters. *`applicationroot`*`/mds/` adds a number of files pertaining to the newly cloned portlet instance. `faces-config.xml` is added/updated whenever you add an Oracle ADF component (such as a portlet) to a page. `adf-faces-config.xml` is added/updated whenever you add an Oracle ADF component to a page.
Removing a portlet from a page	Same files as placing a portlet on a page.

11.3 Implementing Common Requirements Once

It is good practice for the administrator of the project to implement any common developer requirements once and then checkin that version for all to use. By planning ahead and having the administrator take care of these common requirements up front, you can reduce redundancy and error.

For example, suppose two developers need to add Omniportlet on different pages of the application. If the project administrator has already registered the Omniportlet producer, then it is readily available for both of them to use. If not, then each developer will likely register the Omniportlet producer separately, leading to unnecessary duplication and confusion.

Another example would be the case of many developers needing content from the same Oracle Content Database repository. One person should setup and checkin the needed connection first. Other developers can then simply reuse the same data control

11.4 Producer Considerations

When working in a team environment, bear in mind the following considerations pertaining to producers:

- Producer Connections

11.4.1 Producer Connections

When building an EAR file for your project, connections are loaded for the entire application rather than individual projects. For example, suppose you have an application with two projects, P1 and P2. P1 has 100 registered producers and P2 has no producers. When you build an EAR file for either project, all 100 of P1's producer connections are loaded into `connections.xml`. Note, though, that you can also edit `connections.xml` manually.

When you run the Predeployment tool to create a targeted EAR file, all of the connections in `connections.xml` must be accessible. Hence, in our example, the generation of a targeted EAR file for either project would fail if any of the 100 producer connections for P1 are unavailable for some reason. Given this behavior, you must carefully consider how you plan to subdivide your overall development effort into applications and projects.

11.4.2 Producer Name Clashes

While Oracle WebCenter Framework enables you to register two producers under the same name, it is generally better to avoid this situation. For example, if two developers working on the same application inadvertently register a producer with the same name, then it is usually best to change one of the producer's names to be unique. If you have two producers registered under the same name, then it becomes very difficult to distinguish between them when debugging errors or performing administrative tasks on the producers.

11.4.3 Combining Portlets from Different Producers

In some cases, you might have multiple developers building portlets and ultimately you want those portlets to be combined under a single producer. To achieve this goal, your developers must be conscious of some potential issues.

- Portlet names and JSP paths could clash. Portlet developers should use prearranged class package names and JSP paths to avoid naming clashes when portlets are combined within one producer in one application.
- When you create a JPS portlet in the Portlet Wizard, the directory for the portlet modes defaults to `portlet`*n*`\html\`*mode_name*, where *n* is a number that increments for each portlet you create. To avoid directory and file name clashes, portlet developers should change the directory name on the Content Type and Portlet Modes page of the Portlet Wizard. Select the portlet mode and then change the directory name in the corresponding field to something unique.
- When you combine the portlets into one producer, you must manually merge the portlet descriptor files, avoiding identifier clashes as you do so as follows:
  - JPS portlet identifiers in the `portlet.xml` and `oracle-portlet.xml` files are automatically generated starting from `portlet1`. When you manually merge multiple portlet descriptor files into one, you must change any portlet identifiers that clash with one another.
  - Similarly, the PDK-Java portlet identifiers in `provider.xml` are automatically generated starting from `1`. When you manually merge multiple `provider.xml` files, you must change any portlet identifiers that clash with one another.

- You must also manually merge any Web descriptor (`web.xml`) changes, for example, security role information.
- For PDK-Java portlets, you might also need to manually merge `.properties` files.

11.5 Security Considerations

Each time a developer updates the policy for a page or component (for example, an iterator or data control) using the Authorization Editor, the `system-jazn-data.xml` file in Oracle JDeveloper's embedded Oracle Containers for J2EE (OC4J) `config` directory (*JDEV_HOME*`\system\oracle.j2ee.10.1.3.xx.xx\embedded-oc4j\config`) is updated. At the same time, these changes are also propagated to the `app-jazn-data.xml` file, to enable the policies to be deployed with the application. As a result of this model, you should adhere to the following guidelines:

- Policies are global, based on the secured object's name. Hence, development teams must use a global naming convention to ensure unique naming for objects in their components. If the object defined in a grant already exists, then the policies for that object are merged, which can lead to unexpected results.
- If you made any manual changes to the `system-jazn-data.xml` file (such as using regular expressions or wildcards), then you must manually replicate these updates in the `app-jazn-data.xml` file prior to checking it into CVS. For more information about `app-jazn-data.xml`, see Section C.6.1, "app-jazn-data.xml".
- To ensure that the `system-jazn-data.xml` file is always accurate and up-to-date, Oracle recommends that each developer runs the JAZN Migration tool with `app-jazn-data.xml` from CVS as the source and the `system-jazn-data.xml` file in *JDEV_HOME*`\system\oracle.j2ee.10.1.3.xx.xx\embedded-oc4j\config` as the destination.
- To ensure that the `system-jazn-data.xml` files are always accurate and up-to-date, Oracle recommends that each developer perform the following steps whenever they take the latest application files from CVS.

> **Note:** Because the `app-jazn-data.xml` does not appear in the Applications Navigator, it is often difficult to determine when it has changed. Hence, the safest guideline is to merge it into `system-jazn-data.xml` every time you bring down the latest files from CVS.

As developers need to perform this procedure fairly frequently, it makes sense to develop a simple batch script that takes care of it. Example 11–1 shows a sample of such a batch script. Note that you must modify *xx.xx* in the `EMBED_SYS_JAZN` path for your environment.

Example 11–1 Sample MS Windows Batch Script for Updating system-jazn-data.xml File

```
@ECHO OFF
CLS
ECHO ================== Merging Policies ====================

SET JDEV_HOME=C:\jdev
SET APP_JAZN=%JDEV_HOME%\jdev\mywork\Application1\.adf\META-INF\app-jazn-data.xml
```

```
SET EMBED_SYS_JAZN=%JDEV_HOME%\jdev\system\oracle.j2ee.10.1.3.xx.xx\embedded-oc4j\
  config\system-jazn-data.xml
SET CLASSPATH=%JDEV_HOME%\j2ee\home\jazn.jar;%JDEV_HOME%\BC4J\lib\adfshare.jar

ECHO Java Home   : %JDEV_HOME%
ECHO Source File : %APP_JAZN%
ECHO Dest File   : %EMBED_SYS_JAZN%

ECHO .
ECHO Updating Oracle JDeveloper's embedded OC4J system-jazn-data.xml
java oracle.security.jazn.tools.JAZNMigrationTool -sr jazn.com -dr jazn.com
  -st xml -dt xml -sf %APP_JAZN% -df %EMBED_SYS_JAZN% -m all

ECHO ============================================================
```

Example 11–2 shows the output that you get when you run the batch script from Example 11–1.

Example 11–2 Sample Output from Batch Script

```
================== Merging Policies ====================
Java Home   : C:\JDev
Source File : C:\JDev\jdev\mywork\Application1\.adf\META-INF\app-jazn-data.xml
Dest File   : C:\JDev\jdev\system\oracle.j2ee.10.1.3.40.40\embedded-oc4j\config\
system-jazn-data.xml
.
Updating Oracle JDeveloper's embedded OC4J system-jazn-data.xml
============================================================
```

Part III

Deploying and Monitoring Your WebCenter Application

Part III contains the following chapters:

12

Deploying Your WebCenter Application

Deployment is the process through which application files are packaged as an archive file and transferred to the target Oracle Application Server. This chapter describes concepts related to WebCenter application deployment and explains the procedures you must perform to package, predeploy, deploy, and undeploy WebCenter applications.

Read the following sections to understand the WebCenter application deployment and learn step-by-step procedures:

12.1 Introduction to WebCenter Application Deployment

This section introduces you to the WebCenter application life cycle and WebCenter application deployment in the production and development environments. It includes the following sections:

12.1.1 Understanding the WebCenter Application Deployment Life Cycle

WebCenter applications differ from traditional J2EE applications in that they support run-time customization of the application's pages and of the portlets contained within these pages. Customizations are stored as follows:

- WebCenter application customizations are stored in Oracle Metadata Services (MDS) on the file system.
- Portlet producer customizations (or preferences) are stored in a Preference Store, which can be file-based or located in a database.

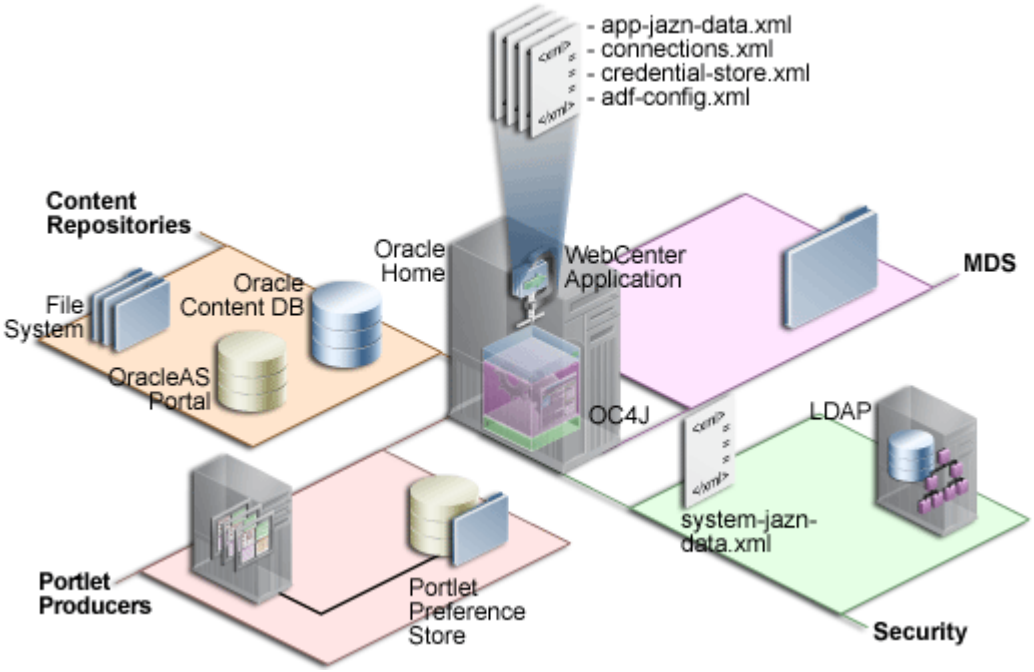
WebCenter applications typically consume portlets from various portlet producers and can connect to content repositories, such as Oracle Content Database (Oracle Content DB) and Oracle Application Server Portal (OracleAS Portal). The connection details are stored inside of the WebCenter application `.ear` file (`connections.xml`).

Therefore, it is common that when a WebCenter application is deployed on a stage or production server that some, or all, of the details for the external dependencies may have changed. For example, the following items may have changed:

- Portlet producer end-points
- Connection details for content repositories
- Credentials for the connections
- MDS location
- Portlet producer preference store locations
- Policy and identity information

Figure 12–1 describes external dependencies for WebCenter applications.

Figure 12–1 WebCenter Applications and External Dependencies



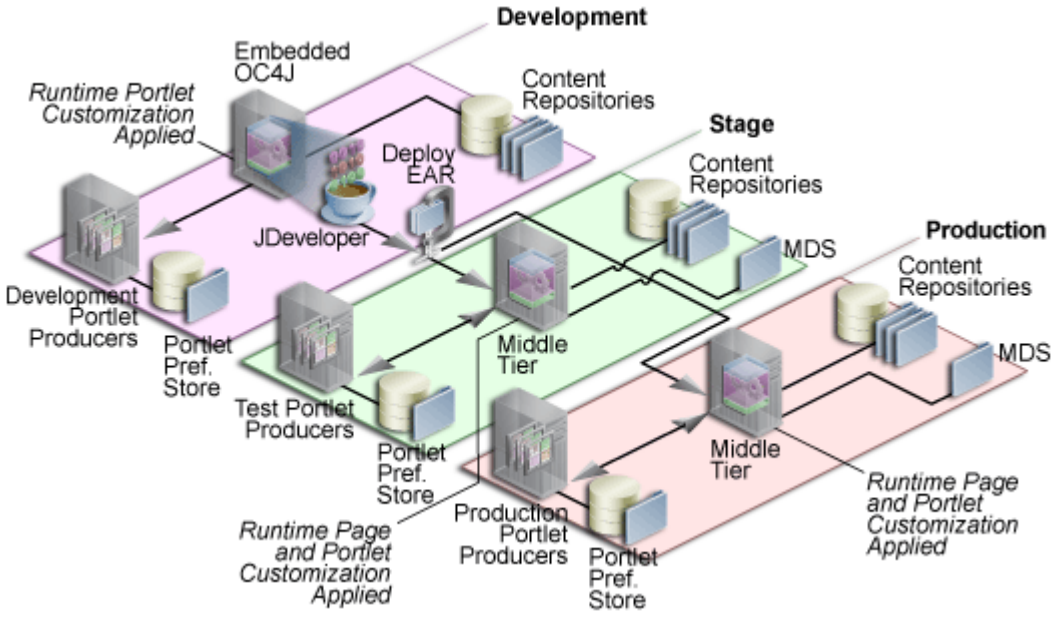
To address the additional requirement, the Oracle WebCenter Framework provides the *Predeployment tool*. This tool can migrate customizations that are associated to a WebCenter application from one location to another and it enables you to reconfigure WebCenter application as well, simplifying the deployment process.

The complexity of the deployment process is largely determined by the following two factors:

- Number of deployment stages
- Number of stages during which customizations are enabled

Once portlets have been added to a page using Oracle JDeveloper, you have to decide in which stages of the life cycle customizations are enabled (development, stage, and production). While portlet customizations made within the development environment (for example, in Omniportlet and Webclipping portlets) are persisted, page customizations are not written to MDS. This enables each subsequent execution of the application to start with the initial page design so that developers can iteratively modify their page without a resultant clash occurring from previous customizations. It should be noted that page and portlet customizations made postdevelopment (for example, in stage and production), are persisted in MDS, as shown in Figure 12–2. These postdevelopment customizations are processed separately from the base application.

Figure 12–2 Development, Stage, and Production Environments



WebCenter Application Scenarios

The following WebCenter application deployment phases can be distinguished:

- Initial deployment: The application is deployed in the production system for the first time.
- Subsequent deployment: A second version of the application is rolled out over a previously deployed version. In this case, it is likely that run-time modifications have occurred since the initial deployment.
- Parallel deployment: The application is deployed across multiple production servers for use in highly available configuration. This is conceptually the same for each, but treated as a separate case. See *Oracle Application Server High Availability Guide* for more information about high availability.

The following sections discuss how initial and subsequent deployments are performed in different scenarios:

- Section 12.1.1.1, "Scenario 1: Portlet Customization in the Development Environment"

- Section 12.1.1.2, "Scenario 2: Customization Performed Only In the Production Environment"
- Section 12.1.1.3, "Scenario 3: Customization of Deployed Applications in Both the Stage and Production Environments"

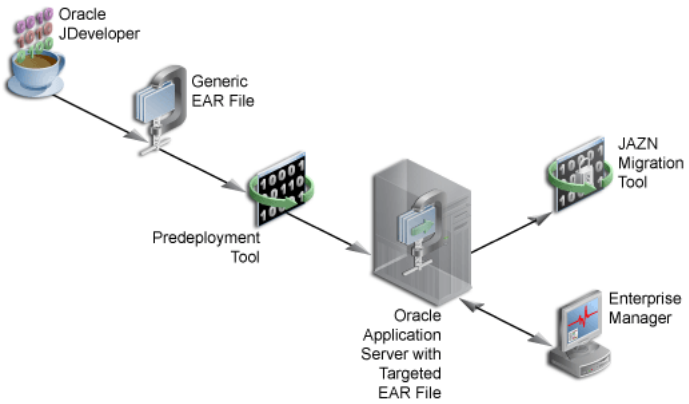
12.1.1.1 Scenario 1: Portlet Customization in the Development Environment

In this scenario, developers perform page customizations and portlet customizations when running the application using the embedded Oracle Containers for J2EE (OC4J) in Oracle JDeveloper. As page customizations are not persisted in the development environment, only portlet customizations are packaged as part of the generic EAR file. The following sections describe initial and subsequent deployment processes:

Initial Deployment

Figure 12–3 describes the cascade deployment of WebCenter applications.

Figure 12–3 Cascade Deployment of WebCenter Applications



The following steps provide an overview of the initial deployment process:

1. Create an application using WebCenter Application template, as described in Section 3.1.1, "Creating a WebCenter Application Using a Template".
2. Run the application in the embedded OC4J and customize the portlets that are contained in the application, as required.
3. Deploy the application to an EAR file, as described in Section 12.2.1.4, "Creating the Generic EAR File".
4. Transfer the generic EAR file to the stage platform.
5. Run the Predeployment tool on the stage platform to unpack the customizations to that server's local file system. Perform the following procedures, if they are applicable to your environment:
   - Run the Predeployment tool to reconfigure any portlets to reference producers accessed by the stage server, using the Predeployment tool. See Section 12.2.2, "Predeploying Your WebCenter Application" for more information.
   - If your WebCenter application uses content stored in Oracle Content DB using the Oracle Content DB data control, then use the Predeployment tool to reconfigure Oracle Content DB for producer connections, as described in Reconfiguring Parameters of Oracle Content DB-based Content Integration Applications.

- If your WebCenter application uses content stored in OracleAS Portal using the OracleAS Portal-based content data control and the JNDI name used in the OracleAS Portal data source is different to that used in the development environment, then update OracleAS Portal connection information in the `connection.xml` file, as described in Reconfiguring OracleAS Portal and File System Adapters.
- If your WebCenter application uses content stored on the file system using the file system data control, then update file system connection information in the `connection.xml` file, as described in Reconfiguring OracleAS Portal and File System Adapters. However, the file system adapter is used only for development purposes.
- If your MDS location has changed, then reconfigure the application to use the new location using the Predeployment tool, as described in Section 12.2.2, "Predeploying Your WebCenter Application".

6. Deploy the targeted EAR file created by the Predeployment tool to the OC4J instance (standalone or on Oracle Application Server) on the stage platform, as described in Section 12.2.6.2, "Deploying to Standalone OC4J".
7. If your application uses encrypted attributes, such as, passwords in the credential store, then use the Credential Mbean to change them, as described in Section 12.5, "Updating Credentials in a Deployed Application".
8. If the embedded OC4J and the deployment servers are not using the same identity store, then run the JAZN Migration tool against the `app-jazn-data.xml` file found within the `META-INF` directory of the deployed application. Merge policies into the local `system-jazn-data.xml` policy store or generate an equivalent LDIF file for incorporation into a central LDAP based policy store, as described in Section 12.2.4, "Migrating Security and Application Roles".
9. After testing, repeat the steps on the production server.

Subsequent Deployment

In this case, all customizations are performed in the development environment with no subsequent customizations at run time. The subsequent deployment can be seen as overwriting of the initial deployment. Repeat the steps each time you update application.

12.1.1.2 Scenario 2: Customization Performed Only In the Production Environment

If customizations of the application only occur in the production environment, then the subsequent redeployment of the application requires only a single point of truth for customizations and therefore it can be performed by using a single export and import procedure during the upgrade. The steps described in this section assume that the testing phase will occur independently of this deployment.

> **Note:** It is recommended that the application be run in a non-customizable mode during the period of the upgrade to a subsequent version of the application. For information, see Chapter 10, "Securing Your WebCenter Application". If this is not possible, then a last minute export of customizations should occur prior to the predeployment of a subsequent version of the application.

Initial Deployment

Perform steps described in Initial Deployment of Section 12.1.1.1, "Scenario 1: Portlet Customization in the Development Environment".

Subsequent Deployment

The following steps provide an overview of the subsequent deployment process:

1. Extract the page and portlet customizations made in the production environment into a separate customization-only archive file by running the Predeployment tool in the export mode (`-export`), as described in Section 12.4, "Transporting Customizations Between Environments".
2. Transfer the generic EAR file created in the development environment to the production server once the stage process is complete.
3. Predeploy the new generic EAR file from the development environment to the production platform and subsequently deploy the new version of the application to the production application server.
4. Import the *exported* run-time customizations (made during the upgrade process), by running the Predeployment tool in the import mode, as described in Section 12.4, "Transporting Customizations Between Environments".

12.1.1.3 Scenario 3: Customization of Deployed Applications in Both the Stage and Production Environments

In this scenario, deployed applications are customized in the stage environment, before the applications are deployed to the production environment. The following sections describe initial and subsequent deployment processes.

Initial Deployment

The following steps provide an overview of the initial deployment process.

1. Perform steps described in Initial Deployment of Section 12.1.1.1, "Scenario 1: Portlet Customization in the Development Environment".
2. Perform run-time customizations after deployment to the stage environment.
3. Extract the page and portlet customizations made in the stage environment into a separate customization-only archive file by running the Predeployment tool in the export mode (`-export`), as described in Section 12.4, "Transporting Customizations Between Environments".
4. Once testing is complete in the stage environment, predeploy the original generic EAR file (the one that was produced in the development environment) to the production server and deploy the application accordingly.
5. Import the run-time customizations that were made during the stage deployment into the application in the production environment, by running the Predeployment tool in the import mode (`-import`), as described in Section 12.4, "Transporting Customizations Between Environments".

Subsequent Deployment

As customizations are performed postdeployment, they are stored separately from the application. In this case, the customizations occur during the stage deployment. If further customization occurred on the production platform after deployment from the stage environment, then these customizations should also be exported prior to a subsequent deployment and used as the most recent customizations for that deployment.

The following steps provide an overview of the subsequent deployment process:

1. Extract the production platform's page and portlet customizations into a separate customization-only archive file by running the Predeployment tool in the export mode (`-export`), as described in Section 12.4, "Transporting Customizations Between Environments".

   Note: It is recommended that further customizations of the production application be prevented during the application upgrade process, until the subsequent version of the application is deployed to the production server. For information, see Chapter 10, "Securing Your WebCenter Application". If this is not possible, then a subsequent export of the customizations will be required.

2. Predeploy and deploy the new version of the application on the stage platform, as described in steps 5 to 8 in Initial Deployment of Section 12.1.1.1, "Scenario 1: Portlet Customization in the Development Environment".

3. Once deployed to the stage platform, update the application to include the latest set of end user customizations from the production server using the appropriate customization archive as the input file. To do so, run the Predeployment tool in the import mode (`-import`), as described in Section 12.4, "Transporting Customizations Between Environments". Consider the following while updating your application:

   - If the production application was run in a non-customizable mode during the upgrade process, then use the customization archive created in step 1.
   - If customization of the production application was enabled during the upgrade process (for example, the upgrade was staged over a number of days or weeks to enable for load testing, and so on), then a new export of the customization from the production system should be taken prior to applying the changes to the stage system. Use this new export EAR file as the input file for the Predeployment tool.

   This will synchronize the stage environment with the customizations that are used by the application end users.

   Note: Any customizations performed by end users during production execution take precedence over *seeded* values that are deployed with the application. This is because customizations are defined by named attributes or value pairs, and if the new version of the application has defined a different value for the named attributes of customized objects, then the imported value will override the new deployed version. For all other attributes the new deployed version will be used.

4. Once the application is ready to be deployed to the production server, any subsequent customizations performed on the stage server should be exported using the Predeployment tool, as described in Section 12.4, "Transporting Customizations Between Environments". Likewise, if the production system was not run in a non-customizable mode, then an export of the most up-to-date customization should be taken from the production system by running the Predeployment tool in the export mode.

5. Predeploy the new generic EAR file from the development environment to the production platform and subsequently deploy the new version of the application to the production application server.
6. Import the run-time customizations that were made during the stage deployment, to the application in the production environment, by running the Predeployment tool in the import mode, as described in Section 12.4, "Transporting Customizations Between Environments".
7. Apply any subsequent customizations that may have been added to the production system during the application upgrade process, by running the Predeployment tool in the import mode, as described in Section 12.4, "Transporting Customizations Between Environments".

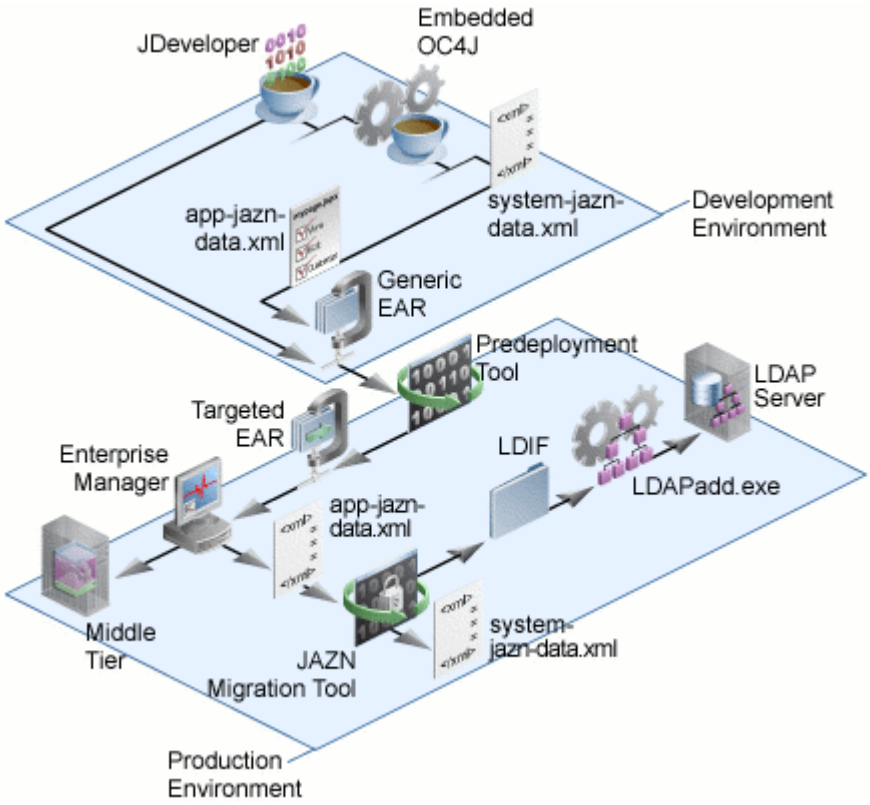
12.1.2 About WebCenter Application Deployment in the Production Environment

In the production environment, applications are deployed to OC4J in Oracle Application Server using Application Server Control Console.

> **Note:** An alternate way to deploy your WebCenter application is to use the command-line interface. However, Oracle recommends that you use only Application Server Control Console to deploy your applications, because it offers ease-of-use and enables quick deployment.

Figure 12–4 depicts the process flow to deploy a WebCenter application to an OC4J instance in Oracle Application Server in the production environment. This figure shows that a generic EAR file is created using Oracle JDeveloper in the development environment. Then, the targeted EAR file is created from the generic EAR file using the Predeployment tool. The configuration files in the targeted EAR file are configured to work with the target system. Finally, the targeted EAR file is deployed to OC4J in Oracle Application Server in the production environment using Application Server Control Console. This figure also shows that the JAZN Migration tool migrates security role and policy information from the development environment to the production environment for the XML or the LDAP providers.

Figure 12–4 Deployment of a WebCenter Application to an OC4J in Oracle Application Server



The procedure of creating the generic EAR file is called *packaging*. The procedure of creating the targeted EAR file is called *predeployment*, and the procedure of deploying the targeted EAR file to OC4J is called *deployment*. In the production environment, you can also use WebCenter Ant Tasks to deploy your WebCenter applications.

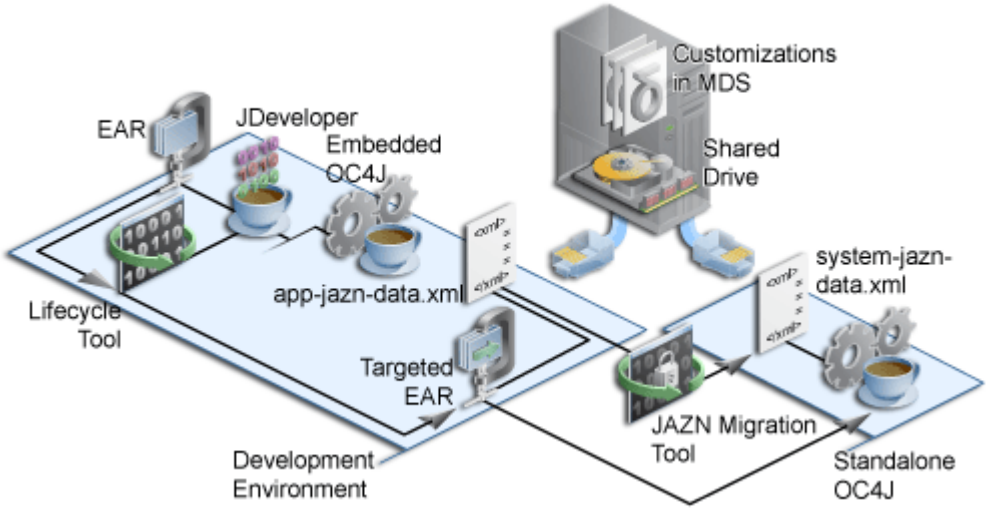
12.1.3 About WebCenter Application Deployment in the Development Environment

In the development environment, application deployment should be quick, because developers must test their applications frequently at run time. Oracle JDeveloper enables you to deploy your WebCenter application to the following:

- A standalone OC4J instance that is either running on the same system as Oracle JDeveloper, or using a shared network drive for the MDS location, as shown in Figure 12–5.
- An OC4J instance on Oracle Application Server running on the same computer as Oracle JDeveloper.

In this case, Oracle JDeveloper runs the Predeployment tool automatically as you deploy to OC4J. However, in this simplified deployment, you cannot change portlet producer end points and MDS must be accessible from the development environment. Additionally, you must run the JAZN Migration tool against the `app-jazn-data.xml` file that is part of the generic EAR file to update the `system-jazn-data.xml` file for the OC4J instance you are deploying your application on. For more information, see Section 12.2.6.2, "Deploying to Standalone OC4J".

Figure 12–5 Deployment of a WebCenter Application to Embedded and Standalone OC4J Instances



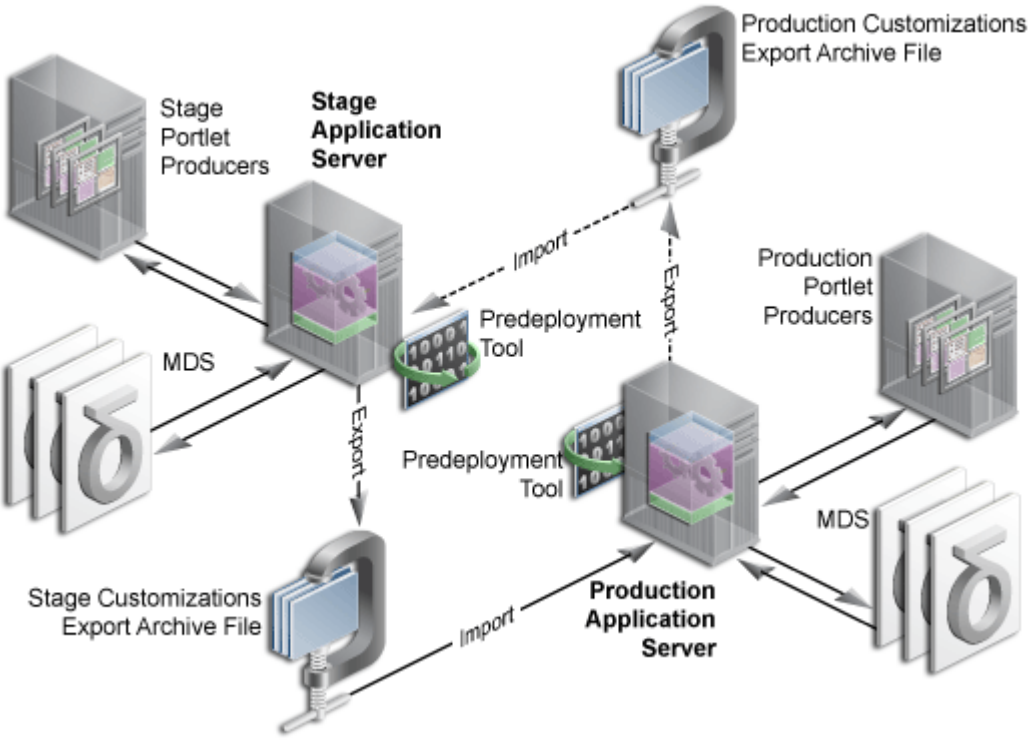
You can also use the embedded OC4J in Oracle JDeveloper to quickly test your application in a browser. For more information, see Section 12.2.6.1, "Deploying to Embedded OC4J".

12.1.4 About Transporting Customizations between Environments

The Predeployment tool available with Oracle WebCenter Suite lets you transport portlet customizations from a stage environment to a production environment. Common scenarios of portlet customizations is the change of a portlet title, defining an OmniPortlet, and so on.

The Predeployment tool enables you to export customizations made in a stage environment and import them into the production environment, as shown in Figure 12–6. Similarly, you can export customizations from the production environment and import those to the stage environment. To do this, you run the Predeployment tool in export and import mode, as described in Section 12.4, "Transporting Customizations Between Environments".

Figure 12–6 Customizations Export and Import



12.2 Deploying Your WebCenter Application

You can deploy your WebCenter applications in a production environment using Application Server Control Console. In the process, you package your WebCenter application in a generic EAR or WAR file. After that, you run the Predeployment tool against this file to remap the WebCenter application's external dependencies, for example, portlet producer end points and the MDS location. The Predeployment tool generates a targeted EAR file that is ready to deploy on the remote system. Additionally, some tasks related to security, content integration, and external applications may need to be performed as part of the deployment process.

Read the following sections to understand how deployment-related tasks are performed:

- Section 12.2.1, "Packaging Your WebCenter Application"
- Section 12.2.2, "Predeploying Your WebCenter Application"
- Section 12.2.3, "Deploying Your WebCenter Application Using Application Server Control Console"
- Section 12.2.4, "Migrating Security and Application Roles"
- Section 12.2.5, "Deploying Your WebCenter Application Using the Command Line"
- Section 12.2.6, "Deploying Your WebCenter Application Using Oracle JDeveloper"
- Section 12.2.7, "Deploying an External Application"
- Section 12.2.8, "Deploying a Content Integration Application"

12.2.1 Packaging Your WebCenter Application

To deploy a WebCenter application, all required files must be packaged in a standard J2EE format and directory structure, in an EAR file. All the packaging and deployment instructions for the WebCenter application are configured through a deployment profile. The deployment profile manages all the components necessary for the deployment of an application. It is basically a configuration file, which includes names of the pages, portlets, customizations, and metadata comprising the application, the type and name of the archive file to be created, dependency information, platform-specific instructions, and more.

This section includes the following topics:

- Section 12.2.1.1, "What You Should Know About Packaging a WebCenter Application"
- Section 12.2.1.2, "Creating the WebCenter Application WAR Deployment Profile"
- Section 12.2.1.3, "Manually Creating and Editing the orion-application.xml File"
- Section 12.2.1.4, "Creating the Generic EAR File"

12.2.1.1 What You Should Know About Packaging a WebCenter Application

This section describes in the following sections important factors that you must know before you package your WebCenter application:

- What You Should Know About Packaging Security Information
- What You Should Know About Packaging an External Application

What You Should Know About Packaging Security Information

The policy information for OC4J, which includes permissions, users, roles, and role memberships for your application, is typically stored in the `system-jazn-data.xml` file. WebCenter application-specific policy information is stored in the `app-jazn-data.xml` file and this file is updated when policy information changes. In the production environment, the `app-jazn-data.xml` file is used as an input file for the JAZN Migration tool that updates the Lightweight Directory Interchange Format (LDIF) file or the production `system-jazn-data.xml` file. The LDIF file is imported into Oracle Internet Directory for the administrator's use. For further information, see Section 12.2.4, "Migrating Security and Application Roles".

What You Should Know About Packaging an External Application

Consider the following points when you package an external application:

- Before packaging an external application, ensure that values predefined in the external application, such as login URLs point to those required in the production environment. You must do this because these values cannot be modified at the predeployment stage. This is in contrast to producer URLs of WebCenter applications, which can be changed while generating the targeted EAR files.
- If your external application is hosted on a different computer or instance, then you must update the external application login URL prior to creating an EAR file from the deployment profile. Change the first part of the login URL to reflect the new host name. For example, from `http://m1.abc.com:7777/` to `http://lbr.abc.com/`.

12.2.1.2 Creating the WebCenter Application WAR Deployment Profile

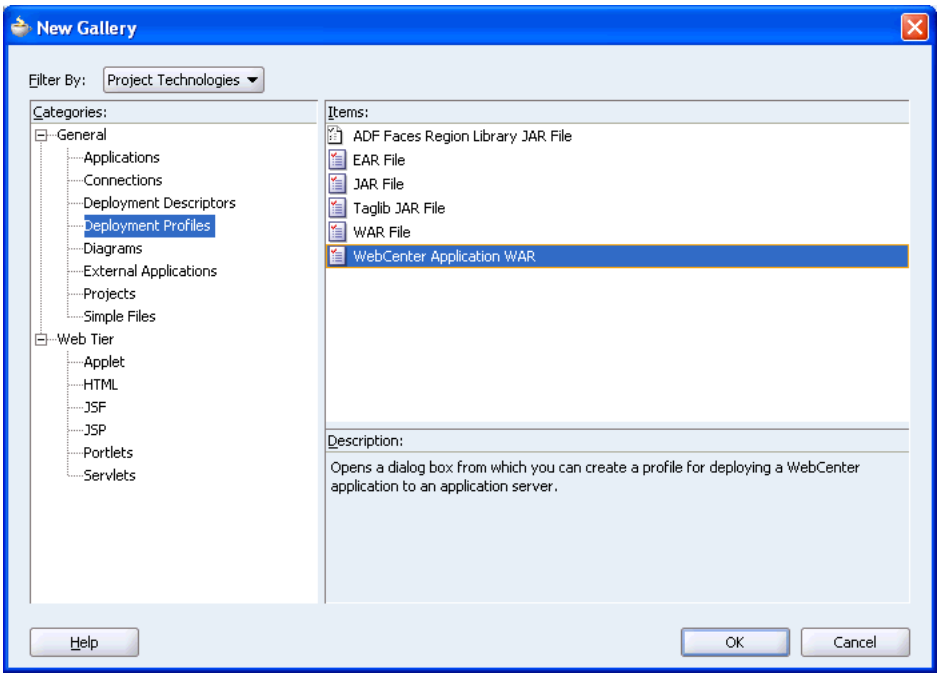
To deploy your WebCenter application, you must use a *WebCenter application WAR deployment profile*, which is then included in a generic EAR file, as described in Section 12.2.1.4, "Creating the Generic EAR File".

> **Note:** If your WebCenter application contains portlets and producers, then register the first producer before creating the deployment profile. See Section 4.3.1, "Registering Portlet Producers" for information about registering producers. During the first producer registration, the MDS instance definition is created in the `adf-config.xml` file. To define the MDS file contributors, the deployment profile creation process uses this MDS instance definition.

To create the deployment profile, perform the following steps:

1. From the Application Navigator, select the **ViewController** folder. Then, choose **New** from the **File** menu. The New Gallery window is displayed.
2. From Filter By, select **Project Technologies** and under General, select **Deployment Profiles**. Then, under Items select **WebCenter Application WAR** (as shown in Figure 12–7) and click **OK**.

Figure 12–7 WebCenter Application WAR File Item in the New Gallery Window



3. The Create WebCenter Application Deployment Profile window is displayed. Enter a name for the deployment profile and click **OK**. Your deployment profile (`.deploy`) is created under the Resources folder.
4. Go to the Application Navigator and expand the **Resources** folder. Then, right-click the deployment profile and select **Properties**. The WAR Deployment Properties dialog box is displayed.
5. Under General, select **Specify J2EE Web Context Root** and enter the J2EE context root in the corresponding field.

6. Select **Platform** and then select your connection from Target Connection. If you do not have a connection, then see Section 12.2.6.2.1, "Defining Standalone OC4J Connection Details". Use can use this option to specify a connection to an OC4J instance on Oracle Application Server.

> **Note:** If you select a connection from Target Connection, then the `orion-application.xml` will be updated automatically when you create the EAR file. If you cannot create a connection, you must manually add the `orion-application.xml` file, as described in Section 12.2.1.3, "Manually Creating and Editing the orion-application.xml File".

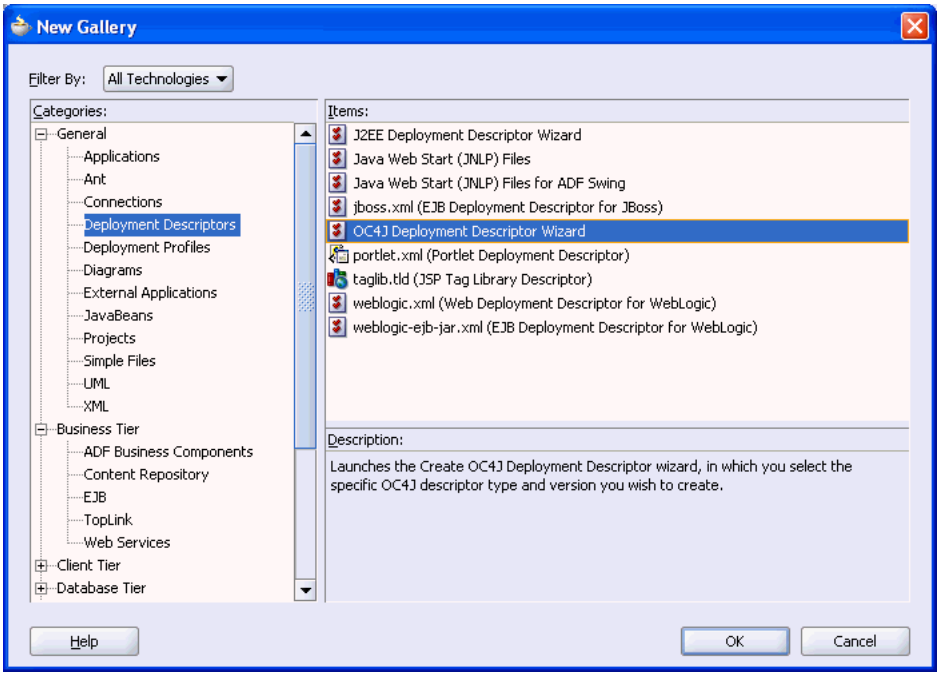
12.2.1.3 Manually Creating and Editing the orion-application.xml File

To manually create the `orion-application.xml` file, you must create a deployment descriptor and specify the `./adf` library path, which is required to point to the `adf-config.xml` file that the Predeployment tool uses while predeploying an application, as described in Section 12.2.2, "Predeploying Your WebCenter Application".

To manually create and edit the `orion-application.xml` file, perform the following steps:

1. In Oracle JDeveloper, select your project, then go to File and select **New**. The New Gallery window is displayed.
2. Under General, select **Deployment Descriptors** and under Items select **OC4J Deployment Descriptor Wizard**, as shown in Figure 12–8.

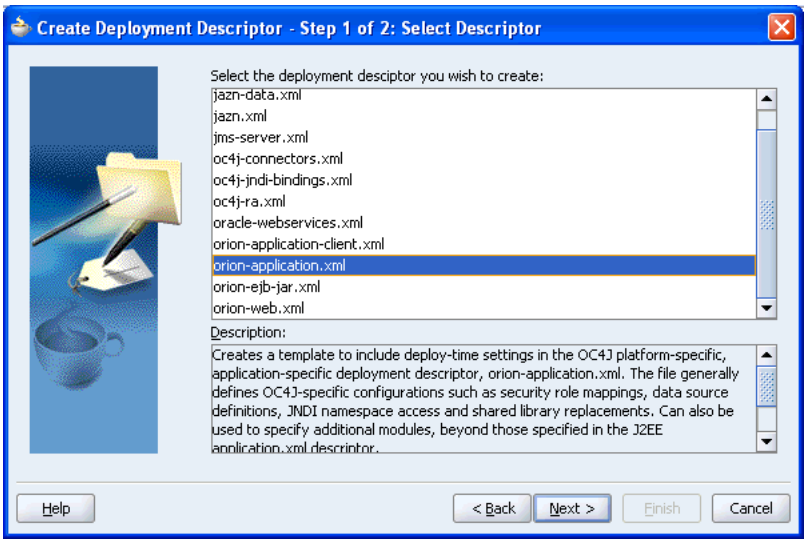
Figure 12–8 New Gallery - OC4J Deployment Descriptor Wizard



3. Click **OK**. The Create Deployment Descriptor - Welcome page is displayed.
4. Click **Next** to display the Select Descriptor page.

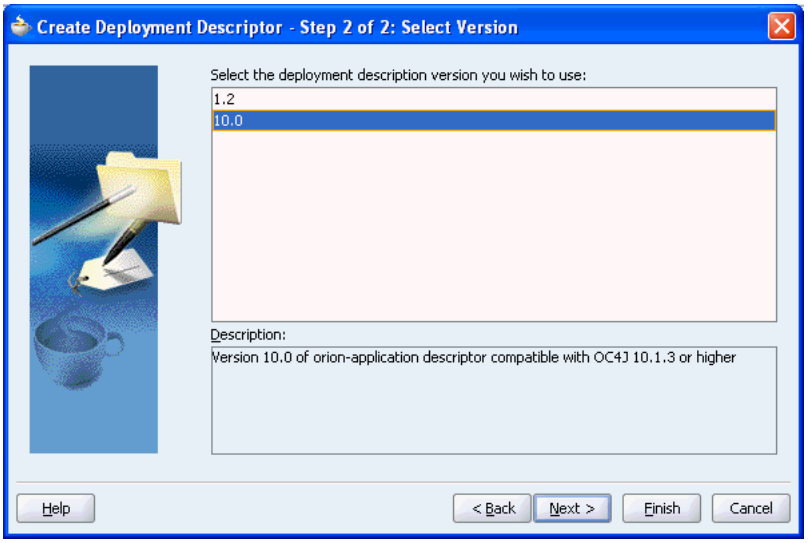
5. Select **orion-application.xml**, as shown in Figure 12–9, and click **Next** to display the Select Version page.

Figure 12–9 Select Descriptor - orion-application.xml



6. Select **10.0**, as shown in Figure 12–10, and then click **Next**.

Figure 12–10 Select Version



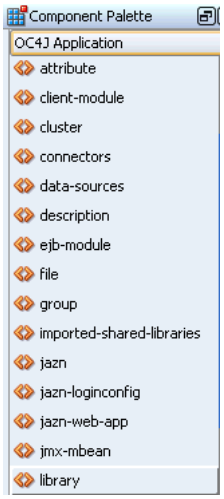
7. Click **Finish** to complete the creation of `orion-application.xml`.

8. To view the `orion-application.xml` file, go to your project in the Application Navigator and expand the **Application Sources** and **META-INF** folders, as shown in Figure 12–11.

Figure 12–11 Application Navigator - orion-application.xml

9. Double-click the `orion-application.xml` file to open it.
10. Open the Component Palette, if it is not already opened. Select the **OC4J Application** option and then select **library**, as shown in Figure 12–12.

Figure 12–12 Library Option



11. Drop the library node on to the `orion-application.xml` page. The `<library> </library>` element is added.
12. In the Property Inspector, enter `./adf` for the path.
13. From OC4J Application, select **jazn** and drop it on to the page.
14. In the Property Inspector, enter `./jazn-data.xml` for location and select **XML** for Provider.
15. Now, select **data-sources** and drop it on to the page.
16. In the Property Inspector, enter `./data-sources.xml` for the path. Example 12–1 shows the sample `orion-application.xml` file.

Example 12–1 Sample Orion-Application.xml File

```
<?xml version = '1.0'?>
<orion-application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/orion-appli
cation-10_0.xsd">
<library path="./adf" />
<jazn location="./jazn-data.xml" provider="XML"/>
<data-sources path="./data-sources.xml"/>
</orion-application>
```

12.2.1.4 Creating the Generic EAR File

To create the generic EAR file, right-click the *name*.deploy file and select **Deploy to EAR file**. The deployment finished message appears in the Deployment - Log window.

On Windows, the default location of the EAR file is *ORACLE_HOME*\jdev*mywork**MyApplication*\ViewController\deploy

On Linux, the default location of the EAR file is *ORACLE_HOME*/jdev/*mywork*/*MyApplication*/ViewController/deploy

You can change this location that Oracle JDeveloper uses and create your application in any preferred location. To do so, right-click your deployment profile (.deploy) in the Application Navigator and select Properties. In the General tab of the WAR Deployment Properties dialog box change the path to EAR and WAR files. You can also change your application name.

The generic EAR file is used by the Predeployment tool to create the targeted EAR file, which is deployed to OC4J in Oracle Application Server in the production environment, using Application Server Control Console.

12.2.2 Predeploying Your WebCenter Application

Before you deploy the WebCenter Application EAR file to OC4J in Oracle Application Server, the development references contained in the file must be modified to be specific to the target instance. You use the Predeployment tool to create a targeted EAR file. The targeted EAR file is typically created on the computer that has the Oracle Application Server installation, to which the application is going to be being deployed. This ensures that the MDS directory is created in the appropriate location on the target system, and the targeted EAR file contains the correct MDS path. The targeted EAR file is deployed to OC4J in Oracle Application Server using Application Server Control Console.

The Predeployment tool performs the following when the EAR file is run through the tool:

- Creates the MDS store on the target system.
- Updates the adf-config.xml file: The existing MDS directory in this file points to the development environment. MDS is the dedicated store in which metadata specific to WebCenter applications, such as page customizations, is stored. The Predeployment tool changes the MDS directory to the production environment, based on the location that you provide when prompted.
- Updates the connections.xml file: The URL and proxy information of the producers contained in this file is updated based on the input that you provide when prompted by the Predeployment tool.
- Imports the portlet customizations from the MDS Export data set to the production producers, that is, creates the MDS store on the target system. However, the credential store is not migrated during predeployment or export and import.

This section includes the following topics:

- Section 12.2.2.1, "What You Should Know About Predeployment"
- Section 12.2.2.2, "Predeploying WebCenter Applications and JCR Adapter-based Applications"

12.2.2.1 What You Should Know About Predeployment

Consider the following points when you predeploy your WebCenter application:

- You can run the Predeployment tool at any time to reconfigure any of the connection details and other exposed settings. After predeploying your application, you must redeploy the application again.
- The generic EAR file contains relative paths set up by Oracle JDeveloper and you can specify multiple paths specified by a comma separated list, for example, `../../mds/;../../`. However, when deploying using the Predeployment tool to the targeted EAR file, only one MDS path can be specified in the `profile.xml` file and this must be an absolute path, for example, `C:\MDS\deploy`.
- The predeployment tool accepts an MDS path without a terminating slash (`/`).
- A shared MDS path, that is, a network mounted drive, is treated as a local drive. The stored MDS path should be the path of the mounted drive from the computer on which you plan to deploy the targeted EAR file. If you use a shared MDS, then you must only execute the Predeployment tool once to produce the targeted EAR file, and then you deploy the targeted EAR file on other OC4J instances. For information about how to select the MDS path using Oracle JDeveloper, see Section 12.2.6.2.2, "Deploying Your WebCenter Application to Standalone OC4J".
- If multiple instances of OC4J in Oracle Application Server share an MDS location, then the `adf-config.xml` file should be updated manually on each location to specify the path of the shared MDS from that location. This must be done because the drive on which the MDS is mounted could be different in each instance. For example, if there are two OC4J instances, then in one instance the MDS may be mounted on drive F and in another instance, the MDS may be mounted on drive E, so the `adf-config.xml` file should be updated on both OC4J instances. However, to avoid updating the `adf-config.xml` file due to different mount points, it is recommended to keep the mount point same on all OC4J instances.
- Before predeploying an external application, ensure that values predefined in the external application such as login URLs, point to those required in the production environment. You must do this because these values cannot be modified during predeployment or postdeployment. This is in contrast to producer URLs of WebCenter applications that can be changed using the Predeployment tool while generating the targeted EAR files.
- If you are accessing an Secure Sockets layer (SSL) enabled producer, and if the producer's security certificate is not listed in the keystore, then before predeploying your application, you must register the security certificate with the keystore. To do this, follow the steps described in Section 10.8, "Registering Custom Certificates with the Keystore".
- The Predeployment tool does not support signed EAR files.
- The Predeployment tool does not support reconfiguration of secure connections in WebCenter applications that contain portlets.

12.2.2.2 Predeploying WebCenter Applications and JCR Adapter-based Applications

This section discusses predeployment procedures for WebCenter applications and content integration applications, as follows:

- Predeploying Your WebCenter Application Using the Predeployment Tool
- Reconfiguring WSRP Portlet Producers

- Reconfiguring Parameters of Oracle Content DB-based Content Integration Applications
- Reconfiguring OracleAS Portal and File System Adapters

Predeploying Your WebCenter Application Using the Predeployment Tool

The Predeployment tool is used to create targeted EAR files from generic EAR files.

The Predeployment tool uses the `-jar` java command line, which calls `portlet-client-deploy.jar` that ensures the correct classpath and main class for the tool. By default, the Predeployment tool JARs and its dependencies are located in *ORACLE_HOME*`/adfp/lib`.

To predeploy your application, run the following command from `ORACLE_HOME` of the target system:

> **Tip:** You can run this command from any other directory if absolute paths are specified in the arguments to the Predeployment tool.

```
ORACLE_HOME/jdk/bin/java -jar ORACLE_HOME/adfp/lib/portlet-client-deploy.jar
-predeploy -source <genericEAR> -target <targetedEAR>
[ -configuration <config file> [-profile <profile name>] ] -backup <directory
path>
```

where:

`portlet-client-deploy.jar` is the predeployment JAR file from which the Predeployment tool is run.

`genericEAR` is the generic EAR file created using Oracle JDeveloper.

`targetedEAR` is the targeted EAR file, which the Predeployment tool creates during predeployment.

`config file` is the XML mapping file, which contains information about one or more predeployment profiles, as shown in Example 12–2. A config file may contain multiple predeployment profiles. If it does, then you must specify which one you want to use when you run the Predeployment tool. If there is only one profile in the configuration file, you do not need to specify the profile name. Example 12–3 shows a sample config file with multiple profiles.

`profile` is the name used inside the XML mapping file to identify a particular predeployment profile. If you do not specify a profile name in the mapping file, then the tool checks that there is an entry. If there is only one entry, then the tool uses it. But if there are several entries, then it generates an error because it cannot discern which entry to use.

Note: In WSRP portlets, when the `XSD` file for the portlets is located on an external site such as OASIS, then the Predeployment tool obtains validation information from the external site. To enable the Predeployment tool to obtain validation from Internet, specify the standard Java proxy configuration properties before the `-jar` flag, as shown in the following example:

```
ORACLE_HOME/jdk/bin/java -Dhttp.proxyHost=www-proxy.my.company.com
-Dhttp.proxyPort=80 -jar adfp/lib/portlet-client-deploy.jar
```

`-backup` is optional. It accepts as its value the path to targeted EAR file in the application home of your Oracle Application Server, that is, *ORACLE_*

HOME/j2ee/home/applications/YourApplication. The backup and recovery configuration of the specified ORACLE_HOME is updated so that the MDS path that the user enters during predeployment is included as part of a backup. The following example discusses the use of -backup option:

```
ORACLE_HOME\jdk\bin\java -jar portlet-client-deploy.jar -source c:\EAR\sample.ear
-target c:\EAR\sampleTarget2.ear -predeploy -backup
ORACLE_HOME\j2ee\home\applications\sampleTarget2
```

In this example, the MDS path provided while running the Predeployment tool is updated in <*ORACLE_HOME*>/backup_restore/config/config_misc_files.inp. The config_misc_files.inp file already exists and is appended with the MDS path. For more information, see section titled Backup Strategy and Procedures in *Oracle Application Server Administrator's Guide*.

Example 12–2 shows a sample config file with a single profile.

Example 12–2 Sample Config File with a Single Profile

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<DeployConfig xmlns="http://xmlns.oracle.com/webcenter/lifecycle/deployconfig">
  <profile name="Production">
   <mds_metadata_path>
          /home/mydirectory/mdstest
   </mds_ metadata_path>
    <producers>
   <producer name="MyWebProducerConnection" proxyHost="" proxyPort="0"
    serviceURL="http://host_name:port/adapter/portal"/>
    </producers>
  </profile>
</DeployConfig>
```

Example 12–3 shows a sample config file with multiple profiles.

Example 12–3 Sample Config File with Multiple Profiles

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<DeployConfig xmlns="http://xmlns.oracle.com/webcenter/lifecycle/deployconfig">
<profile name="Production">
   <mds_metadata_path>
   /WorkEnvLifecycle/Lifecycle/jpswork/components/Lifecycle/data/MDS
   </mds_metadata_path>
   <producers>
      <producer name="WSRPProducerConnection1"
proxyHost="www-proxy.my.company.com" proxyPort="80"
serviceURL="http://host:port/adapter/portal/portletapp/portlets/WSRPBaseService?WS
DL"/>
   </producers>
</profile>
<profile name="portletArchiveWSRP203760">
    <mds_metadata_path>
     /WorkEnvLifecycle/Lifecycle/jpswork/components/Lifecycle/data/MDS
      </mds_metadata_path>
    <producers>
        <producer name="WSRPProducerConnection1"
proxyHost="www-proxy.my.company.com" proxyPort="80"
     serviceURL="http://host:port/adapter/portal/nonexist/portlets?ABCD"/>
     </producers>
</profile>
<profile name="portletArchiveWSRP203750">
     <mds_metadata_path>
```

```
    /WorkEnvLifecycle/Lifecycle/jpswork/components/Lifecycle/data/NoPermission
     </mds_metadata_path>
     <producers>
        <producer name="WSRPProducerConnection1"
proxyHost="www-proxy.my.company.com" proxyPort="80"
serviceURL="http://host:port/adapter/portal/portletapp/portlets/WSRPBaseService?WS
DL"/>
     </producers>
</profile>
<profile name="portletArchiveWSRP197300">
    <mds_metadata_path>
      /WorkEnvLifecycle/Lifecycle/jpswork/components/Lifecycle/data/MDS/197300
      </mds_metadata_path>
     <producers>
       <producer name="WSRPProducerConnection1"
proxyHost="www-proxy.my.company.com" proxyPort="80"
serviceURL="http://host:port/adapter/portal/portletapp/portlets/WSRPBaseService?WS
DL"/>
    </producers>
</profile>
```

Example 12–4 shows a sample of the output that the Predeployment tool generates while predeploying a WebCenter application. If you do not specify the config file, then the Predeployment tool prompts you for the information, which is in **bold** in the sample.

Example 12–4 Output of the Predeployment Tool for a WebCenter Application

```
http://host:port/jpdk/providers/sample
$ java -jar portlet-client-deploy.jar -source
/Automation/PDKValid1.ear -target /Automation/PDKValid1Target1.ear -predeploy
Processing Arguments
Run Mode : 1
Source : /Automation/PDKValid1.ear
Target : /Automation/PDKValid1Target1.ear
...
Cleaning up /tmp/predeploy/
Processing source EAR file
Source EAR file processed
Processing adf-config.xml
...
Producer : PdkPortletProducer1_1160553606037366 2ab97-010e-1000-8001-984463fc1d8d
Current Service URL :
http://host:port/jpdk/providers/sample
Current Proxy URL   :
Current Proxy Port   : 0
Do you want to modify this connection? (Y/N [default=N]) :
Y
Enter new Service URL (or Enter to leave it as it is):
http://newhost:newport/jpdk/providers/sample
Do you want to save this new Deployment Profile (Y/N [default=N]) :
Y
Enter a name for this Deployment Profile (e.g. 'Production') :
PDKValid1
Enter the path for the Deployment Profile XML file (e.g. 'C:\profile.xml'):
/JDEV/DepFiles/PDKValid1.cml
...
MDS extracted from source EAR file
Creating target EAR file
Processing source EAR file
```

```
...
source MDS Path (temp) : /tmp/predeploy/mds
Production MDS Path : /MDS/PDKValid1
Connections.xml Path : /tmp/predeploy/connections.xml.new
Export ID : /export
<Date> <Time> oracle.mds
...
Backing up existing target EAR to /Automation/PDKValid1Target1.ear.old
Moving /tmp/predeploy/PDKValid1Target1.ear to /Automation/PDKValid1Target1.ear
Target EAR /Automation/PDKValid1Target1.ear created
Cleaning up /tmp/predeploy/
```

Reconfiguring WSRP Portlet Producers

The Predeployment tool does not support reconfiguration of WSRP portlet producers. Therefore, if you want to deploy your WebCenter application to a production environment where WS-Security is implemented, then register the WSRP producer URLs in the development environment using Oracle JDeveloper. To learn how to register producer URLs, see Section 4.3.1.1, "Registering WSRP Portlet Producers".

Reconfiguring Parameters of Oracle Content DB-based Content Integration Applications

During the predeployment phase, the Predeployment tool lets you reconfigure parameters of Oracle Content DB based content integration applications. You can use the Predeployment tool to reconfigure parameters of both versions 10.1.3.2 and 10.2 of Oracle Content DB. For information about Oracle Content DB parameters, see Section 5.2.4, "Configuring a Content Data Control Based on the Oracle Content DB Adapter" and Section 5.2.5, "Configuring a Content Data Control Based on Oracle Content DB Version 10.2".

> **Note:** While reconfiguring the parameters of a content integration application that is based on Oracle Content DB, you cannot remap a nonsecure connection with a secure connection or a secure connection with a nonsecure connection. Therefore, the type of connections that you map must be of the same type.

When you run the Predeployment tool to reconfigure Oracle Content DB parameters, it prompts you as discussed in Table 12–1.

Table 12–1 Parameters Prompted by the Predeployment Tool

Parameters	Values, if WS-Security is Implemented	Values, if S2S is Implemented
Do you want to modify this connection? (Y/N [default=N]):	`Y` or `N`, as required.	`Y` or `N`, as required.
What is the value of 'Server URL' (or Enter to leave it as is):	Enter the server URL in format: http://*host*:*port*/content/ws	Enter the server URL in format: http://*host*:*port*/content/ws
What is the value of 'Server Version' (or Enter to leave it as is):	`10.3.1.2`	`10.2`
What is the value of 'Trusted Authentication Method' (or Enter to leave it as is):	`WS-Security`	`S2S`

Table 12–1 (Cont.) Parameters Prompted by the Predeployment Tool

Parameters	Values, if WS-Security is Implemented	Values, if S2S is Implemented
What is the value of 'S2S Application Name' (or Enter to leave it as is):	Leave it blank.	Enter name of the application. This is configured in Oracle Internet Directory and must be set here together with the S2S application password.
What is the value of 'KeyStore File Location' (or Enter to leave it as is):	Enter the location where the keystore file should be located in the production environment.	Leave it blank.
What is the value of 'KeyStore Type' (or Enter to leave it as is):	Keystore type is usually `JKS` or `PKCS12`.	Leave it blank.
What is the value of 'Server Public Key Alias' (or Enter to leave it as is):	Enter the alias value that you specified while configuring the keystores.	Leave it blank.
What is the value of 'Private Key Alias' (or Enter to leave it as is):	Enter the private key alias that you specified while configuring the keystores.	Leave it blank.

Note: If you do not have a security setup, then specify only the URL and press Enter to skip other parameters.

Example 12–5 shows remapping of Oracle Content DB version 10.1.3.2 parameters.

Example 12–5 Sample Output of the Predeployment Tool for an Oracle Content DB Version 10.1.3.2-based Application

```
Development MDS Repository Path : C:\adfsrdemojsf\mds;C:\adfsrdemojsf
...
What is the value of 'Server URL' (or Enter to leave it as is):
http://yourhost:port/content/ws
What is the value of 'Server Version' (or Enter to leave it as is):
10.3.1.2
What is the value of 'Trusted Authentication Method' (or Enter to leave it as is):
WS-Security
What is the value of 'S2S Application Name' (or Enter to leave it as is):
<blank>
What is the value of 'KeyStore File Location' (or Enter to leave it as
is):/private1/keystore/client-keystore.jks
What is the value of 'KeyStore Type' (or Enter to leave it as is):
JKS
What is the value of 'Server Public Key Alias' (or Enter to leave it as is):
server
What is the value of 'Private Key Alias' (or Enter to leave it as is):
keyalias
```

Reconfiguring OracleAS Portal and File System Adapters

If you want to reconfigure your OracleAS Portal and File System repositories to point to different instances, then you must manually edit the `connections.xml` file. After deployment, the `connections.xml` file can be found at

ORACLE_HOME`/j2ee/home/applications/<`*Application name*`>/adf/META-INF/`

Example 12–6 shows the segment of the `connections.xml` file, which must be modified to reconfigure the File System adapter.

Example 12–6 Connections.xml for the File System Adapter

```
<StringRefAddr addrType="jcr_basePath">
            <Contents>/private/myfiles</Contents>
     </StringRefAddr>
```

Example 12–7 shows the segment of the `connections.xml` file, which must be modified to reconfigure the OracleAS Portal adapter.

Example 12–7 Connections.xml for the OracleAS Portal

```
<StringRefAddr addrType="jcr_dataSourceName">
            <Contents>jdbc/DBConnection1DS</Contents>
</StringRefAddr>
```

> **Note:** While reconfiguring the parameters of a content integration application that is based on OracleAS Portal, you cannot remap a nonsecure connection with a secure connection or a secure connection with a nonsecure connection. Therefore, the type of connections that you map must be of the same type.

12.2.3 Deploying Your WebCenter Application Using Application Server Control Console

This section describes the procedures to deploy and test your WebCenter application using Application Server Control Console, as follows:

- Section 12.2.3.1, "Deploying Your WebCenter Application"
- Section 12.2.3.2, "Testing Your WebCenter Application"

12.2.3.1 Deploying Your WebCenter Application

To deploy your application using Application Server Control Console, you need the targeted EAR file of your application. The targeted EAR file must be created on the system to which the application is being deployed. If you have not created the generic and targeted EAR files already, then see Section 12.2.1, "Packaging Your WebCenter Application" and Section 12.2.2, "Predeploying Your WebCenter Application" respectively.

> **Note:** For detailed information about deploying applications to OC4J, see *Oracle Containers for J2EE Deployment Guide* on the OracleAS Portal Documentation page on Oracle Technology Network (OTN).

To deploy your application using Application Server Control Console, perform the following steps:

1. To access Application Server Control Console, navigate to the following URL: `http://<host_name>.<domain>:<port>/em`.

   For example, `http://test.acme.com:8888/em`.

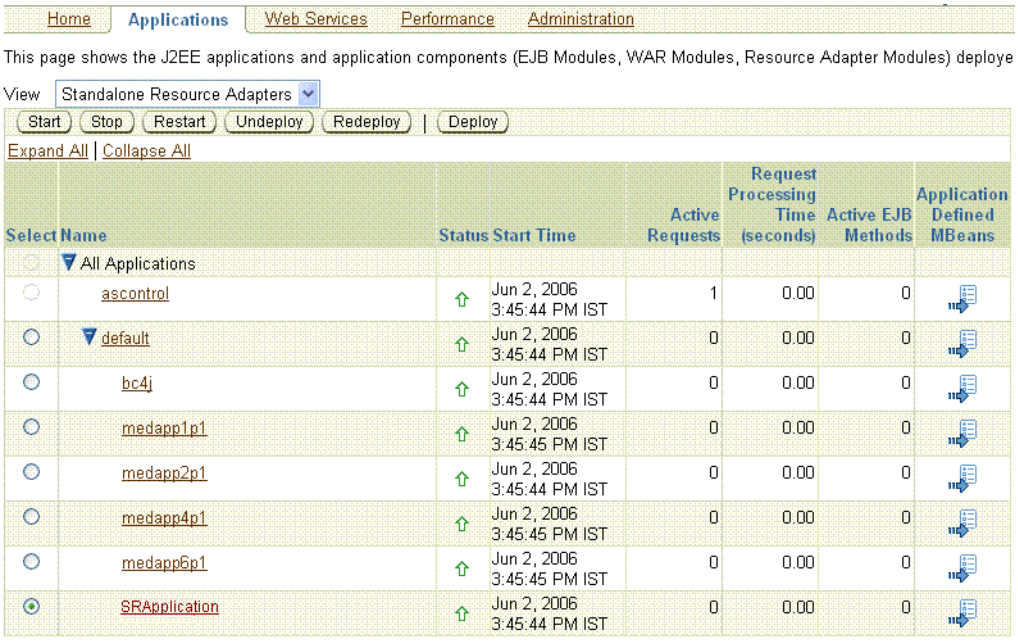
To find the exact URL for your Application Server Control Console, look at `readme.txt`. After installation, this text file is saved to the following Oracle Application Server location:

On UNIX: *ORACLE_HOME*`/install/readme.txt`

On Windows: *ORACLE_HOME*`\install\readme.txt`

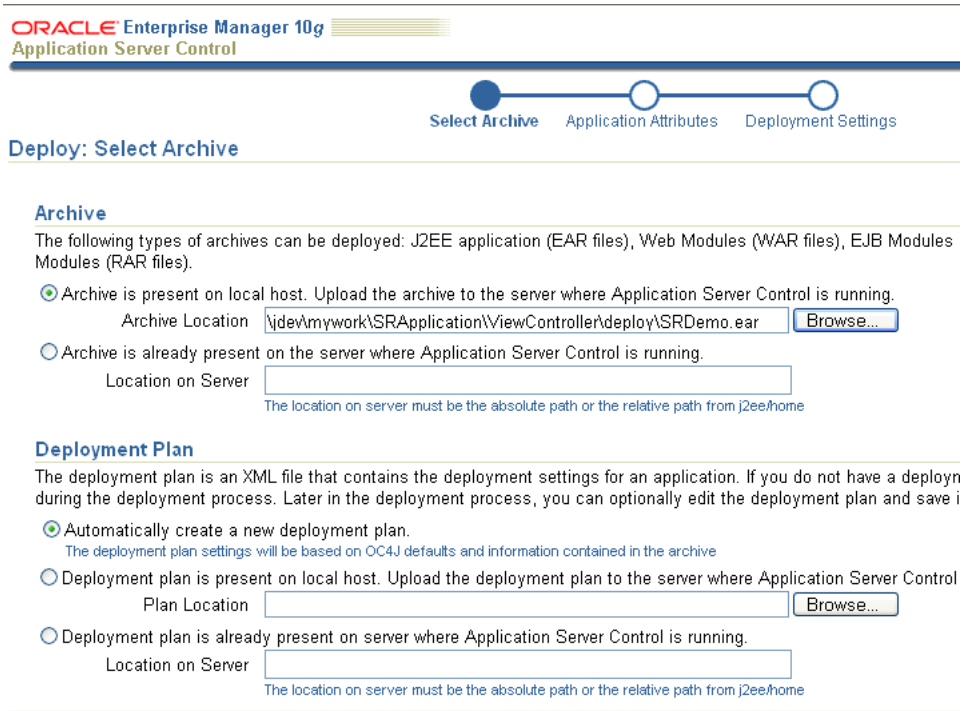
2. Log in to the Application Server Control Console. The Cluster Topology page is displayed.
3. On the Cluster Topology page, click the link to your application server.
4. Under System Components, click the **Create OC4J Instance** button. The Create OC4J Instance page is displayed.
5. Specify a name for the OC4J instance, for example, `myoc4j`.
6. Make sure the Start this OC4J instance after creation check box is not selected.
7. Click **Create**.
8. On Oracle Application Server, select the OC4J instance that you just created.
9. Select the **Applications** tab (Figure 12–13) and click **Deploy**. The Deploy: Select Archive page is displayed.

Figure 12–13 Applications Tab of Application Server Control Console



10. If the EAR file is located on the local computer, then select the **Archive is present on local host** option and browse to the location of the EAR file, as shown in Figure 12–14, or select the **Archive is already present on the server where Application Server Control is running** option. Then, click **Next**.

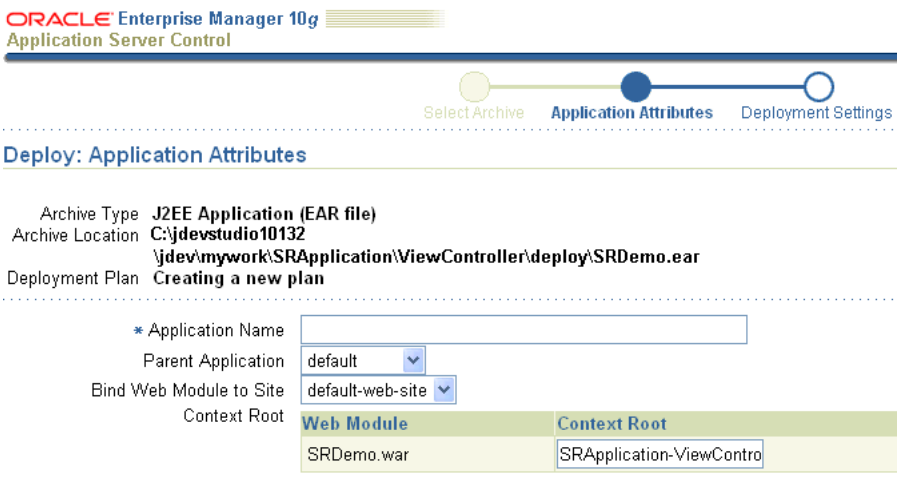
Figure 12–14 Deploy: Select Archive in Application Server Control Console



Note: You can use the Redeploy button to redeploy the application. You can also stop and restart the application from here.

11. In the Deploy: Application Attributes page (Figure 12–15), enter the context root of your Web application as the application name or what you configured while creating the deployment profile described in Section 12.2.1.2, "Creating the WebCenter Application WAR Deployment Profile".

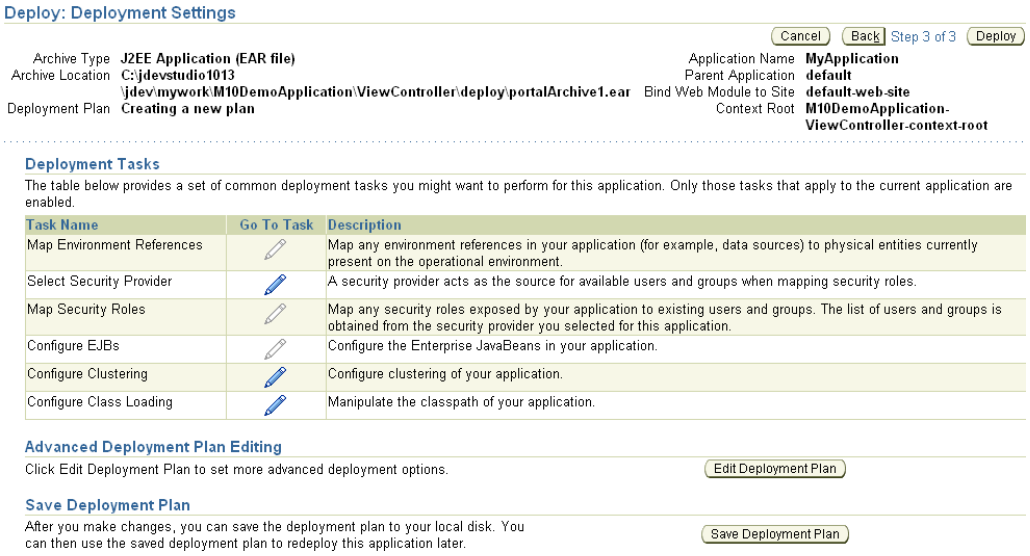
Figure 12–15 Application Attributes in Application Server Control Console



12. Deployment Settings page is displayed, as shown in Figure 12–16. If required, then make the changes in this page and click **Deploy**.

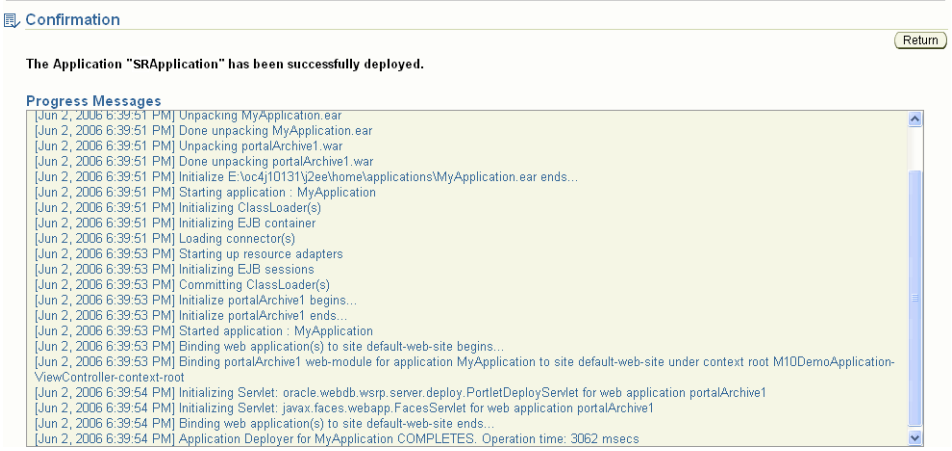
Note: If you have not yet migrated the security policies by running the JAZN Migration tool, then you can defer the selection of the Security Provider.

Figure 12–16 Deployment Settings in Application Server Control Console



13. The deployment confirmation page is displayed, as shown in Figure 12–17.

Figure 12–17 Confirmation Message in Application Server Control Console



14. Migrate the security information, as described in Section 12.2.4, "Migrating Security and Application Roles".

12.2.3.2 Testing Your WebCenter Application

Open a browser window and navigate to the welcome page of your application. You must enter a URL that uses the context root defined in the deployment profile, followed by `/faces/`, and the page name itself. The URL format you require is `http://<host>:<port>/<context-root>/faces/<page-name>`

For example:

```
http://localhost:8888/SRApplication/faces/Welcome.jspx
```

where:

`SRApplication` is the context root specified in the deployment profile, as described in Section 12.2.1.2, "Creating the WebCenter Application WAR Deployment Profile".

12.2.4 Migrating Security and Application Roles

After you have deployed your WebCenter application, you must use the JAZN Migration tool to migrate your application's realm and policy information. To facilitate the deployment of this security information, Oracle JDeveloper packages the specific policies for the application in the `app-jazn-data.xml` file. In your development environment, the `app-jazn-data.xml` is located in your application's `.adf\META-INF` directory. Postdeployment, the file is unpacked to the directory *ORACLE_HOME*`/j2ee/<oc4j_instance>/applications/<app-name>/adf/META-INF`.

The `app-jazn-data.xml` file is created when a policy is defined in the application using the authorization editor of Oracle JDeveloper. Each time a developer updates the policy for a page or component (for example, an iterator or data control), Oracle JDeveloper's embedded OC4J's `system-jazn-data.xml`, located in *JDEV_HOME*`\system\oracle.j2ee.10.1.3.xx.xx\embedded-oc4j\config`, is updated. At the same time, these changes are also propagated to file `app-jazn-data.xml`. The elements and attributes contained in the `app-jazn-data.xml` file are a subset of OC4J's `system-jazn-data.xml` file.

> **See Also:** The section titled "OracleAS JAAS Provider Configuration Files" in *Oracle Containers for J2EE Security Guide*.

When you migrate security information with the JAZN Migration tool, the destination for this information will either be a file-based provider (for example, `system-jazn-data.xml`) or an LDAP provider. In the latter case, the output from the migration tool is an LDIF file that must be subsequently loaded into the LDAP directory (for example, Oracle Internet Directory).

This section discusses the following:

- Section 12.2.4.1, "About Modes of Migration"
- Section 12.2.4.2, "Updating Policy Information (Optional)"
- Section 12.2.4.3, "Using the JAZN Migration Tool"
- Section 12.2.4.4, "Using the ldapmodify Command-Line Tool"

12.2.4.1 About Modes of Migration

The JAZN Migration tool supports the following modes of operation:

- **Realm mode**: This is used to migrate only users and roles. All users in the source realm are migrated when the migration mode is `realm` or `all`.

  When the output LDIF file is generated, passwords from `jazn-data.xml` are exposed in clear text. The administrator maintains and protects the generated LDIF file so that no one else can access the LDIF file.

- **Policy mode**: This is used to migrate grantees and the permissions granted to these grantees.

- **All mode**: This is used to migrate users, roles, grantees, and the permissions granted to these grantees. The grantees can be realm grantees or non-realm grantees.

12.2.4.2 Updating Policy Information (Optional)

If you have defined a list of temporary role names in the development environment that represent the ultimate production roles and have used these roles for policy definition, then at deployment time, these policies must be updated to reflect the actual production roles. Skip this section if you have used actual production role names in the development environment. See Section 10.2.1, "Defining Roles for Developing Secured WebCenter Applications" for more information.

To update policy information with actual production roles, replace the temporary development roles in the `app-jazn-data.xml` file with those you are going to use in the production environment, prior to running the JAZN Migration tool.

Example 12–8 shows the `app-jazn-data.xml` file with a *managers* role defined in the development environment.

Example 12–8 app-jazn-data.xml file with a managers Role

```
<roles>
     <role>
       <name>managers</name>
         <guid>58B213307F7811DBBF8F39184ABB7640</guid>
           <members>
           </members>
     </role>
</roles>
.
.
.
<grant>
  <grantee>
    <principals>
      <principal>
        <realm-name>jazn.com</realm-name>
          <type>role</type>
          <class>oracle.security.jazn.spi.xml.XMLRealmRole</class>
          <name>managers</name>
      </principal>
    </principals>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.adf.share.security.authorization.RegionPermission</class>
      <name>view.pageDefs.testPageDef</name>
      <actions>customize,personalize,view</actions>
    </permission>
  </permissions>
</grant>
```

To map this role to the *doc_managers* role in the production environment, update the name element specified for the role and principal, as shown in Example 12–9.

Example 12–9 Updated app-jazn-data.xml File with the doc_managers Role

```
<roles>
     <role>
```

```
        <name>doc_managers</name>
          <guid>58B213307F7811DBBF8F39184ABB7640</guid>
            <members>
            </members>
      </role>
</roles>
.
.
.
<grant>
  <grantee>
    <principals>
      <principal>
        <realm-name>jazn.com</realm-name>
          <type>role</type>
          <class>oracle.security.jazn.spi.xml.XMLRealmRole</class>
          <name>doc_managers</name>
      </principal>
    </principals>
  </grantee>
  <permissions>
    <permission>
      <class>oracle.adf.share.security.authorization.RegionPermission</class>
      <name>view.pageDefs.testPageDef</name>
      <actions>customize,personalize,view</actions>
    </permission>
  </permissions>
</grant>
```

If the named role does not exist in the identity management solution at the time you run the JAZN Migration tool, it will be added. If it does exist, then the role that is defined in the `app-jazn-data.xml` file will not overwrite the existing role definition, but apply the policy to the existing role in the identity management solution.

12.2.4.3 Using the JAZN Migration Tool

The JAZN Migration tool is packaged as an executable utility, which runs from the command-line. Example 12–10 describes the syntax for the JAZN Migration tool and Table 12–2 describes the elements of the syntax. Before you run the JAZN Migration tool, you must set the `CLASSPATH` to *ORACLE_HOME*`/j2ee/home/jazn.jar;`*ORACLE_HOME*`/BC4J/lib/adfshare.jar`.

> **Note:** You must shutdown OC4J before running the JAZN Migration tool to migrate security policies from the `app-jazn-data.xml` file to the `system-jazn-data.xml` file. Once the policies are migrated, restart OC4J.

Example 12–10 Syntax for the JAZN Migration Tool

```
java oracle.security.jazn.tools.JAZNMigrationTool -help
java JAZNMigrationTool [-D binddn] [-w passwd] [-h ldaphost] [-p ldapport]
                                [-sf filename] [-df LDIF_filename]
                                [-sr source_realm] [-dr dest_realm]
                                [-st source_type] [-dt dest_type]
                                [-m policy|realm|all]
                                [-help]
```

Table 12–2 Description of the JAZN Migration Tool Syntax

Element	Description
`-D`	The Oracle Internet Directory user name.
`-w`	The Oracle Internet Directory user password.
`-h`	The Oracle Internet Directory host. Default: location value in *`ORACLE_HOME`*`/j2ee/home/config/jazn.xml`
`-p`	The Oracle Internet Directory port. Default: location value in *`ORACLE_HOME`*`/j2ee/home/config/jazn.xml`
`-sf`	Path to XML provider file. Default: *`ORACLE_HOME`*`/j2ee/home/config/system-jazn-data.xml`
`-df`	Path to destination LDIF file. Default: *`ORACLE_HOME`*`/j2ee/home/config/system-jazn-data.xml`, if `-dt` is XML.
`-sr`	Name of the realm to be migrated. Default: The realm name when there is only one realm
`-dr`	Name of the destination subscriber realm in the Oracle Internet Directory. Default: The realm name when there is only one realm, and if `-dt` is `xml`. The default subscriber realm if `-dt` is `ldap`.
`-st`	Type of provider at the source. Default: `xml`
`-dt`	Type of provider at the destination. Available options are `ldap` and `xml`. Default: `ldap`
`-m`	The mode for running the tool. Available options are `policy`, `realm`, and `all`. Default: `all` Use the `policy` mode if only the JAAS policies have to be migrated. Use the `realm` mode, if only realm users and roles have to be migrated. Use the `all` mode, if user, roles, and JAAS policies have to be migrated.
`-help`	Display usage.

Example 12–11 describes the syntax to facilitate migration from XML to XML. Table 12–2 describes the elements of the syntax.

Example 12–11 Syntax for XML to XML Migration

```
java oracle.security.jazn.tools.JAZNMigrationTool
-sr sourcerealm -dr destrealm -st xml -dt xml -sf -<sourcefile> -df <destfile> -m
policy|realm|all
```

Example 12–12 describes the syntax to facilitate migration from XML to LDAP. Table 12–2 describes the elements of the syntax.

Example 12–12 Syntax for XML to LDAP Migration

```
java oracle.security.jazn.tools.JAZNMigrationTool
-D binddn -w passwd -h ldaphost -p ldapport -sr sourcerealm -st xml -dt ldap -sf
<source file> -df <dest file> -m policy|realm|all(default)
```

12.2.4.4 Using the ldapmodify Command-Line Tool

The `ldapmodify` command-line tool enables you to add, delete, or replace attributes for entries by supplying an LDIF file as input. Example 12–13 describes the syntax for ldapmodify and Table 12–3 discusses the arguments for `ldapmodify`.

> **Note:** The `ldapmodify` command is available in the OracleAS Infrastructure installation that you have associated with your application. Therefore, run the `ldapmodify` command from the OracleAS Infrastructure host.

Example 12–13 Syntax for ldapmodify

```
ldapmodify -h <oid_host_name> -p <oidport> -D <binddn> -w <password>  -f
<ldiffile> -v -c -o <errors_ldiffile>
```

Table 12–3 Arguments for ldapmodify

Elements	Description
`-h`	The host name or IP address of the Oracle Internet Directory server, for example, `infrahost.company.com`. Host name is mandatory.
`-p`	The port number used to connect to the Oracle Internet Directory server, for example, `389`. Port number is optional.
`-D`	The DN of the Oracle Internet Directory user, for example, `cn=orcladmin`. It is needed to bind to the directory and is mandatory.
`-w`	The user password needed to bind to the directory, for example, `welcome1`. Password is mandatory.
`-f`	The full path and file name of the input file that contains the data you want to import, for example, `entry.ldif`. This is mandatory.
`-v -c -o`	The tool continues on errors and writes all the LDIF lines causing the errors into the specified error LDIF file so that they can be corrected and resubmitted.

12.2.5 Deploying Your WebCenter Application Using the Command Line

This section describes the procedure to deploy a WebCenter application using the command-line interface. The command-line tool deploys the targeted EAR file that the Predeployment tool generates from the generic EAR file. If you have not created the generic and targeted EAR files already, then see Section 12.2.1, "Packaging Your WebCenter Application" and Section 12.2.2, "Predeploying Your WebCenter Application" respectively.

In this section, you will use command-line syntax for the following:

- Section 12.2.5.1, "Deploying the EAR File to OC4J in Oracle Application Server"
- Section 12.2.5.2, "Testing the Deployment"

- Section 12.2.5.3, "Binding web_module to a Web Site"

12.2.5.1 Deploying the EAR File to OC4J in Oracle Application Server

To deploy the targeted EAR file to OC4J in Oracle Application Server, perform the following steps:

1. Place the targeted EAR file in a directory in your Oracle Application Server. For example, *ORACLE_HOME*/j2ee/home/applications/
2. Go to your J2EE home. The path should be like *ORACLE_HOME*/j2ee/home/
3. To deploy the application, run the following command:

```
java -jar admin.jar ormi://localhost/<ormi port> <admin_username> <admin_pwd>
deploy -file <full_path_of_ear_file> -deploymentName <deployment_name>
```

where:

admin.jar is available at *ORACLE_HOME*/j2ee/home/

ormi://localhost/<ormi port> is the Remote Method Invocation (RMI) port with default value 23791. Its value gets stored in the config file. This is stored in the rmi.xml file available at <*ORACLE_HOME*>/j2ee/home/config.

<admin_username> is the user name of the administrator of your OC4J in Oracle Application Server.

<admin_pwd> is the password of the administrator of your OC4J in Oracle Application Server.

<deployment_name> is the unique name provided for the deployment in progress.

> **Note:** You can deploy an application multiple times while providing a unique name for each deployment. If the deployment with a name already exists, then it is treated as a redeployment.

12.2.5.2 Testing the Deployment

In OC4J, verify the following:

- In the applications directory, the EAR file appears expanded. The path of the applications directory should be like *ORACLE_HOME*\j2ee\home\applications.
- The application-deployments directory contains files related to the deployed application.
- The <deployment_name> directory is created in both applications and application-deployments directories.

12.2.5.3 Binding web_module to a Web Site

To bind the application to the port to which OC4J listens, perform the following steps:

1. Run the following command:

```
java -jar admin.jar ormi://host/<ormi_port>/ <admin_username> <admin_pwd>
-bindWebApp <deployment_name> <web_module_name> <webSiteName> <context_root>
```

where:

ormi_port is the RMI port. Its value is stored in the config file.

`deployment_name` is the name of the deployed application.

`web_module_name` is the name of the module to bind. This is specified for the application.

`webSiteName` is the host name of the site to which the application is deployed.

`context_root` is the Web Application's Context Root stored in the deployment profile.

2. Access your application with the appropriate URL, which should be in the following format:

```
http://host:port/web_site_name/context_root/faces/page_name.jspx
```

> **Note:** If you have *<webSiteName>* as the default `http-web-site`, then you do not need to include it in the URL, so the format of your URL is `http://`*`host:port/context_root`*`.`

Your WebCenter application is now deployed and ready to use.

12.2.6 Deploying Your WebCenter Application Using Oracle JDeveloper

You can use Oracle JDeveloper to deploy your WebCenter application to a standalone OC4J by performing the following steps:

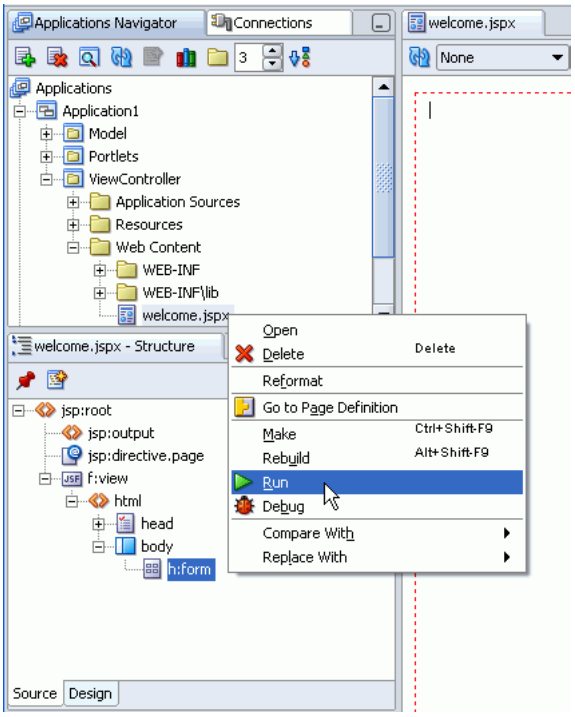
- Section 12.2.6.1, "Deploying to Embedded OC4J"
- Section 12.2.6.2, "Deploying to Standalone OC4J"

> **Note:** If your Oracle Application Server is on the same computer as Oracle JDeveloper, then you can use Oracle JDeveloper to deploy your application to an OC4J instance on the Oracle Application Server.

12.2.6.1 Deploying to Embedded OC4J

To deploy your content pages to the embedded OC4J, and test them in a browser, right-click your project and select **Run**, as shown in Figure 12–18.

Figure 12–18 Testing with Embedded OC4J



The page that you selected will display in the browser.

12.2.6.2 Deploying to Standalone OC4J

This section describes the following procedures:

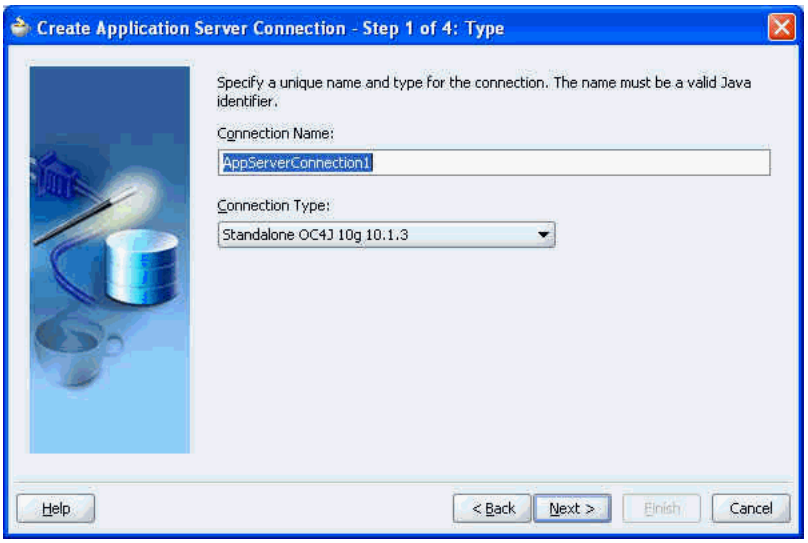
- Section 12.2.6.2.1, "Defining Standalone OC4J Connection Details"
- Section 12.2.6.2.2, "Deploying Your WebCenter Application to Standalone OC4J"

12.2.6.2.1 Defining Standalone OC4J Connection Details

To create a direct connection to the standalone OC4J on which you will deploy your application, perform the following steps:

1. In the Navigator, click the **Connections** tab.
2. Right-click **Application Server** and choose **New Application Server Connection**. The Welcome page is displayed.
3. Click **Next** to exit the Welcome page. The Step 1 of 4: Type window is displayed, as shown in Figure 12–19.

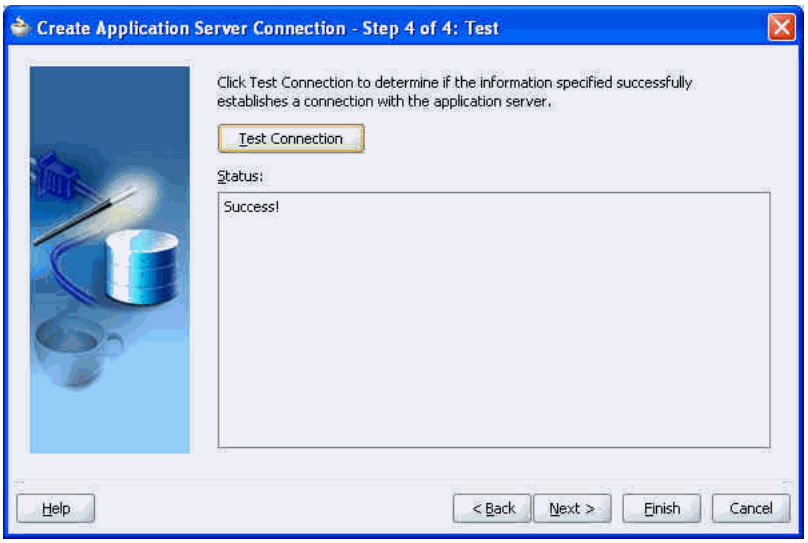
Figure 12–19 Create Application Server Connection - Step 1 of 4 Window



4. On step 1, enter a name for your standalone OC4J connection in the Connection Name field.
5. Then, choose the appropriate connection type from the list. For example, Standalone OC4J 10g 10.1.3 and click **Next**.
6. On step 2, enter a valid user name and password, such as `oc4jadmin` and `oracle1`, and then click **Next**.
7. On step 3, enter the host name of the standalone OC4J, for example, *`myhost.mycompany`*`.com`, its RMI Port, for example, `23795`, and then click **Next**.

   The RMI information is stored in the `rmi.xml` file available under *`ORACLE_HOME`*`/j2ee/home/config`.
8. Click **Test Connection**. The Success! message appears, as shown in Figure 12–20. Click **Finish**. If the test fails, you may need to revise your connection information.

Figure 12–20 Server Connection Successful



Now include this connection in the deployment profile required to deploy your portlet, as described in Section 12.2.1.2, "Creating the WebCenter Application WAR Deployment Profile".

12.2.6.2.2 Deploying Your WebCenter Application to Standalone OC4J

In this section, you will use the connection details you defined in Section 12.2.6.2.1, "Defining Standalone OC4J Connection Details"to deploy your application to your standalone OC4J. To use this connection, you must configure your deployment profile, as discussed in Section 12.2.1, "Packaging Your WebCenter Application".

> **Note:** When deploying a WebCenter application to a standalone OC4J instance, the Predeployment tool runs under the cover. For information about the Predeployment tool, see Section 12.2.2, "Predeploying Your WebCenter Application".

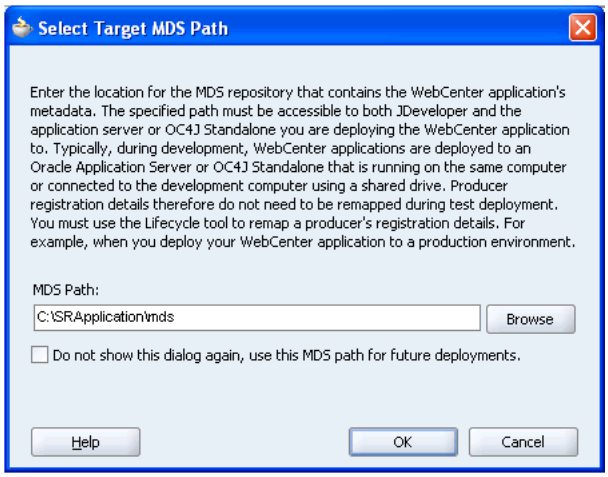
To deploy your application to the standalone OC4J instance, the OC4J must be running on the same computer as Oracle JDeveloper as both Oracle JDeveloper and OC4J require access to the MDS directory that is specified at deployment time. If you choose to deploy your target EAR file from a shared MDS path, that is, a network mounted drive, then it is treated as a local drive to the file system. So, the stored MDS path includes the path of the mounted drive from the system at which you plan to deploy. Hence, to use it in other locations, the target EAR file should be deployed there. Similarly, if you change the location of the MDS, all instances must be redeployed. While deploying an application, the MDS path can be modified. However, producer URLs cannot be changed in this type of deployment.

When you deploy an external application to a standalone OC4J instance using Oracle JDeveloper, user credentials entered while testing the application are not deployed.

To deploy your application, perform the following steps:

1. From the Applications Navigator, right-click *name*`.deploy` under the Resources folder and select the targeted connection. The Select Target MDS Path window is displayed, as shown in Figure 12–21.

Figure 12–21 Select Target MDS Path Window



2. Click **Browse** to select the MDS directory and click **OK**.

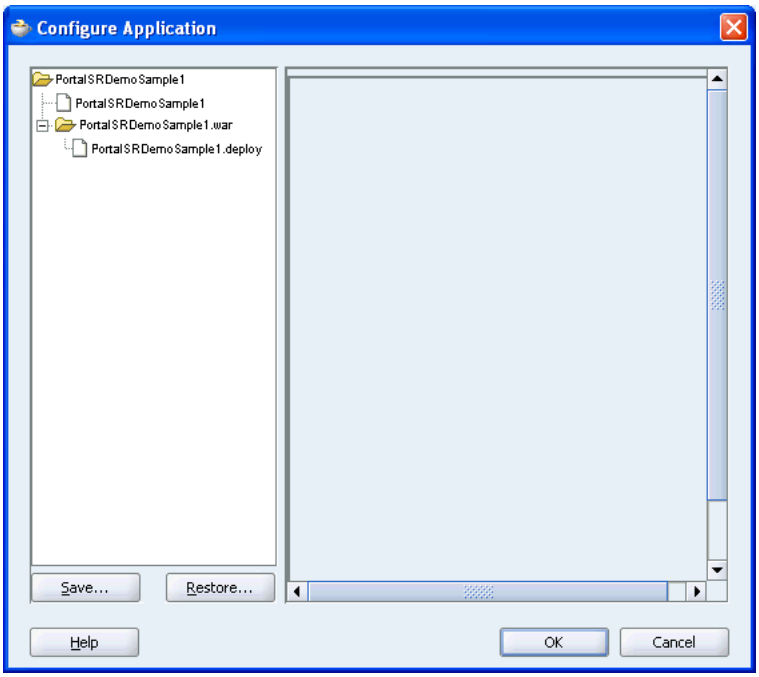
   If the directory you specify does not exist, Oracle JDeveloper creates it for you.

Caution: To avoid browser access to the MDS content, always specify the MDS path to be in a directory that is outside the Web content root. Additionally, always create the MDS repository for your deployment outside the working directory for your application. Keeping separate MDS directories for development and deployment will not only avoid confusion but will prevent content in your application's source MDS repository from being overwritten.

EAR files are generated and uploaded to the standalone OC4J. EAR files contain a number of configuration (`.xml` files) and all the files and libraries used by the application.

3. The Configure Application window is displayed, as shown in Figure 12–22. Click **OK**.

Figure 12–22 Configure Application Window



When the deployment is successful, the page displays in the browser.

Migrating Security Information

To migrate security information to a standalone OC4J, see the procedure in Section 12.2.4.3, "Using the JAZN Migration Tool". This section contains the syntax to migrate security-related data from XML to XML, as shown in Example 12–11.

12.2.7 Deploying an External Application

To deploy an external application, see the procedures in the following sections:

- Section 12.2.3, "Deploying Your WebCenter Application Using Application Server Control Console"
- Section 12.2.5, "Deploying Your WebCenter Application Using the Command Line"

- Section 12.2.6, "Deploying Your WebCenter Application Using Oracle JDeveloper"

12.2.8 Deploying a Content Integration Application

To deploy a content integration application, see the procedures in the following sections:

- Section 12.2.1, "Packaging Your WebCenter Application"
- Section 12.2.2, "Predeploying Your WebCenter Application"
- Section 12.2.3, "Deploying Your WebCenter Application Using Application Server Control Console"
- Section 12.2.4, "Migrating Security and Application Roles"
- Section 12.2.5, "Deploying Your WebCenter Application Using the Command Line"
- Section 12.2.6, "Deploying Your WebCenter Application Using Oracle JDeveloper"

12.3 Deploying Your WebCenter Application with WebCenter Ant Tasks

This section provides an overview of WebCenter Ant Tasks in Oracle WebCenter Framework and describes how to use Ant Tasks to automate deployment of WebCenter applications.

This section covers the following:

- Section 12.3.1, "Overview of WebCenter Ant Tasks"
- Section 12.3.2, "Preparing to Use Ant Tasks"
- Section 12.3.3, "Deploying Your WebCenter Application with Ant Tasks"

12.3.1 Overview of WebCenter Ant Tasks

This section describes the following Another neat tool (Ant) tasks to automate creation of a config file, migration of JAZN data, predeployment of an EAR file, and export and import of customizations:

- webcenter:generateConfigTemplate
- webcenter:preDeploy
- webcenter:exportMdsData
- webcenter:importMdsData
- webcenter:generateMdsExportSet

The WebCenter Ant Tasks discussed here must be used with Apache Ant version 1.6.5, which is distributed for free. For information and most recent Apache Ant product documentation, see `http://ant.apache.org/manual/`

webcenter:generateConfigTemplate

This task generates a configuration file template and populates it with IDs of the producer connections that are used in your WebCenter application. The generated template can then include MDS path and provider remapping details. Example 12–14 describes syntax of the `generateConfigTemplate` task. You must create a configuration template before using the tasks described later in this section.

Example 12–14 webcenter:generateConfigTemplate

```
<webcenter:generateConfigTemplate ear="Source ear file" file="Name of the config
file." />
```

> **Note:** This task generates a configuration file that contains the default MDS path. You must modify this path to specify preferred MDS location.

webcenter:preDeploy

This task is used to predeploy your WebCenter application. It creates a targeted EAR file, which can be deployed to a local application server or a standalone OC4J. The configuration file must contain MDS path and remapping of providers, if any. In addition, a profile name must be specified even if there is only one profile, because this Ant task does not create a default profile like the Predeployment tool does.Example 12–15 describes syntax of the `webcenter:preDeploy` task.

Example 12–15 webcenter:preDeploy

```
<webcenter:predeploy sourceEAR="Source ear file."
                  targetEAR="Targeted ear file."
                  config="Configuration file with MDS path and provider remapping
                  details."
                  profile="Profile name to use from the config file."
</webcenter:preDeploy>
```

> **See Also:** For more information, see *Oracle Containers for J2EE Deployment Guide* and Section 12.2.2.2, "Predeploying WebCenter Applications and JCR Adapter-based Applications"

webcenter:exportMdsData

This task enables export of MDS data from a deployed application into a customizations export archive file. Example 12–16 describes syntax of the `webcenter:exportMdsData` task.

Example 12–16 webcenter:exportMdsData

```
<webcenter:exportMdsData ear="Target ear file."
                  deployedApp="Deployed application from which to export data."
                  mdsPath="MDS path of the jdev application."
                  connectionPath="Path to connections.xml."/>
```

> **See Also:** Section 12.4, "Transporting Customizations Between Environments"

webcenter:importMdsData

This task enables import of customizations from a deployment application into your customizations archive file that was previously exported. Example 12–17 describes syntax of the `webcenter:importMdsData` task.

Example 12–17 webcenter:importMdsData

```
<webcenter:importMdsData ear="Source ear file"
                deployedApp="Import from deployed application into your EAR."/>
```

See Also: Section 12.4, "Transporting Customizations Between Environments"

webcenter:generateMdsExportSet

This task enables the export of design time customizations made to portlets in a WebCenter application to a specified file system path. Example 12–18 describes syntax of the `webcenter:generateMdsExportSet` task.

Example 12–18 webcenter:generateMdsExportSet

```
<webcenter:generateMdsExportSet applicationPath="Path to the WebCenter application
root,
for example, C:\JDeveloper\mywork\MyApplication." targetPath="File system location
to store generated export set."/>
```

12.3.2 Preparing to Use Ant Tasks

The following sections describe prerequisites for Ant tasks and procedures to meet those prerequisites:

- Section 12.3.2.1, "Incorporating Ant Tasks in Your Oracle WebCenter Framework"
- Section 12.3.2.2, "Installing Ant Tasks"

12.3.2.1 Incorporating Ant Tasks in Your Oracle WebCenter Framework

The WebCenter installation includes Ant 1.6.5 and the files for the WebCenter Ant tasks. Before you can use the Ant tasks, you must incorporate them into your environment. To do so, perform the following steps:

1. Set *ORACLE_HOME* environment variable by running the following syntax:

   ```
   setenv ORACLE_HOME ORACLE_HOME/oc4j
   ```

2. Update `CLASSPATH` environment variable to include `ant/lib` directory by running to following syntax:

   ```
   setenv CLASSPATH ${CLASSPATH}:ORACLE_HOME/oc4j/ant/lib
   ```

3. Set `ANT_HOME` by running the following syntax:

   On UNIX:

   ```
   setenv ANT_HOME ORACLE_HOME/ant
   ```

   On Windows

   ```
   set ANT_HOME=ORACLE_HOME\ant
   ```

Note: You must use the `ant` command located in *ORACLE_HOME*`/ant/bin` to ensure that the libraries in *ORACLE_HOME*`/ant/lib` are loaded. Keep this in mind when you define variables, such as ANT_HOME. Invoking the `ant` command from any other directory may not load the appropriate libraries successfully.

12.3.2.2 Installing Ant Tasks

Ant tasks in WebCenter Framework are available out-of-the-box in the OC4J instances that are installed as part of Oracle Application Server. To use these Ant tasks on a standalone OC4J instance, Oracle Application Server 10.1.3.1.0 or earlier, or a third-party application server, you must first run the Oracle ADF Runtime installer to add the following files:

- `<OC4J_HOME>/ant/lib/ant-oracle-adfp.jar`
- `<OC4J_HOME>/adfp/utilities/ant-adfp-classes.jar`
- `<OC4J_HOME>/adfp/utilities/ant-oracle-adfp.xml`

> **Caution:** Do not run the ADF Runtime installer on an Oracle Application Server 10.1.3.2.0 instance as you may get errors and will not be able to restart the application server instance.

A visual check is recommended to ensure that the Oracle ADF Runtime installer has run properly and Ant jars exist in appropriate directories.

If you deploy from an `OC4J_HOME` instance using the same build file that you used in Oracle JDeveloper, then make sure that the following code excerpt is still valid:

```
<import file="ORACLE_HOME/adfp/utilities/ant-oracle-adfp.xml"/>
```

You can make it valid in the following ways:

- If you are using environment variables, then set the `ORACLE_HOME` environment variable to point to `OC4J_HOME`.
- If you are using hard coded path, then update it appropriately.

12.3.3 Deploying Your WebCenter Application with Ant Tasks

This section covers the following procedures:

- Section 12.3.3.1, "Namespacing Class Definitions"
- Section 12.3.3.2, "Creating the build.xml File"
- Section 12.3.3.3, "Deploying with the build.xml File"

12.3.3.1 Namespacing Class Definitions

To use Ant tasks, namespace class definitions in your `build.xml` file, as shown in the following example:

```
<project name="Project1" default="all" xmlns:webcenter="antlib:oracle.adfp">
```

`xmlns` attribute of project enables you to import WebCenter Ant Tasks.

The `webcenter` prefix is customizable and users can change it as required. If this prefix is changed, all the deployment tasks must start with new prefix, for example `<prefix>:deploy`, `<prefix>:import`, and so on.

12.3.3.2 Creating the build.xml File

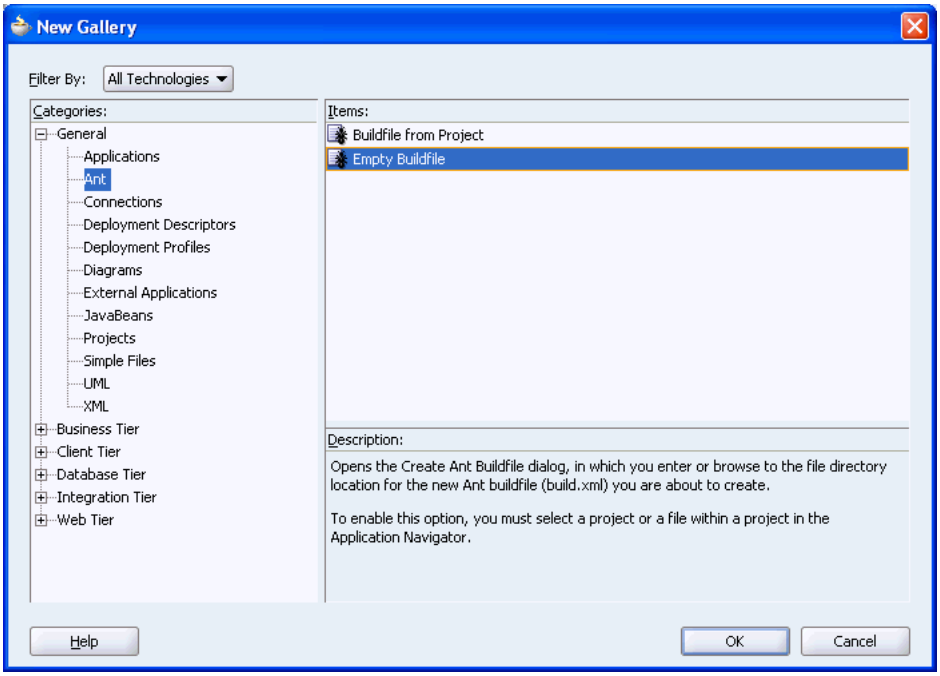
To deploy your WebCenter application with Ant tasks, you must first create a `build.xml` file containing Ant tasks. This `build.xml` file contains a project and targets. Targets contain task elements and each task element of the build file can have an `id` attribute, which can later be referred to by the unique value supplied to this. For

information about WebCenter Ant tasks, see Section 12.3.1, "Overview of WebCenter Ant Tasks".

To create the `build.xml` file, perform the following steps:

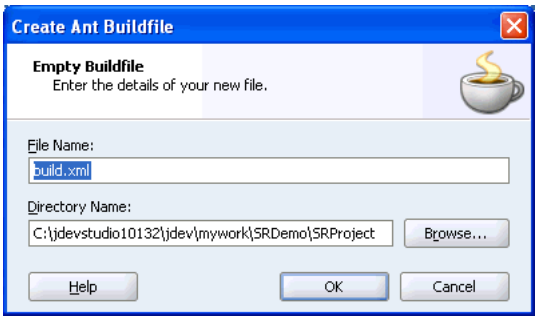
1. In Oracle JDeveloper, go to the Application Navigator and select your project. Then, from the File menu select **New**. The New Gallery dialog box is displayed.
2. Under General select **Ant**, and under Items select **Empty Buildfile**, as shown in Figure 12–23.

Figure 12–23 New Gallery - Empty Buildfile



3. Click **OK**. The **Create Ant Buildfile** dialog box is displayed, as shown in Figure 12–24.

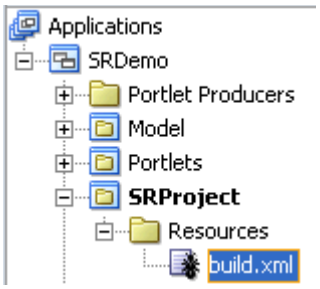
Figure 12–24 Create Ant Buildfile



4. Click **OK**. The `build.xml` file is created under the Resources folder, as shown in Figure 12–25.

> **Note:** `build.xml` is a typical name given to any WebCenter Ant build file and it must remain unchanged.

Figure 12–25 Application Navigator - build.xml



5. Under Resources, double-click the `build.xml` file to open it.
6. In the Source mode, enter Ant tasks, as described in Example 12–19, with appropriate values.

Example 12–19 Sample Build File

```
<?xml version="1.0" encoding="US-ASCII" ?>
<project name="Project1" default="deploy2server"
xmlns:webcenter="antlib:oracle.adfp"
                                        xmlns:oracle="antlib:oracle">
  <property name="ORACLE_HOME" value="c:/jdev"/>
  <import file="${ORACLE_HOME}/adfp/utilities/ant-oracle-adfp.xml"/>
  <property name="proj" value="c:/jdev/jdev/mywork/Application1/Project1/deploy"/>

<!-Config Task. This task generates a configuration file template. -->

  <target name="config">
    <webcenter:generateConfigTemplate ear="${proj}/webcenterArchive1.ear"
                    file="${proj}/config.xml"/>
  </target>

  <!--Predeploy Task. This tasks creates a targeted EAR from a generic EAR. -->

<target name="predeploy">
    <webcenter:predeploy sourceEAR="${proj}/webcenterArchive1.ear"
                    targetEAR="${proj}/target.ear"
                    config="${proj}/config.xml"
                    profile="Template"/>
  </target>

<!--Deploy to Server Task. This task deploys the targeted EAR to the specified
server. -->
  <target name="deploy2server" depends="predeploy">
    <oracle:deploy deployerUri="deployer:oc4j:localhost:23791"
                   userid="oc4jadmin"
                   password="welcome1"
                   file="${proj}/target.ear"
                   deploymentName="webcenterArchive1"
                   bindAllWebApps="default-web-site"
                   logFile="${proj}/deploy.log"/>

<!-- Export and Import tasks Note: Do not run export and import tasks from Oracle
JDeveloper.
Run these tasks in stage/production environment instead-->

  </target>
  <target name="export">
```

```
    <webcenter:exportMdsData ear="${proj}/export.ear"

deployedApp="C:/jdev/j2ee/home/applications/webcenterArchive1"/>
  </target>
  <target name="import">
    <webcenter:importMdsData ear="${proj}/export.ear"

deployedApp="C:/jdev/j2ee/home/applications/webcenterArchive1"/>

<!--FTP Task.-->

  </target>
  <target name="ftp_mds">
<ftp server="localhost" remotedir="C:/tmp"
         userid="ftpuser" password="welcome1"
         binary="no" verbose="yes">
        <fileset dir="C:/tmp/MDS"/>
    </ftp>
  </target>

<!-- Note: You can use an FTP Ant task together with the oracle:deploy task to
copy the MDS directory to a remote OC4J and deploy the application.-->

<target name="ftp_and_deploy" depends="ftp_mds">
      <webcenter:deploy deployerUri="deployer:oc4j:localhost:23791"
             userid="oc4jadmin"
             password="welcome1"
             file="${proj}/target.ear"
             deploymentName="webcenterArchive1"
             bindAllWebApps="default-web-site"
             logFile="${proj}/deploy.log"/>
  </target>
</project>
```

7. Save the `build.xml` file.

12.3.3.3 Deploying with the build.xml File

To deploy your WebCenter application using the `build.xml` file, run the `ant` command from the directory that contains the `build.xml` file. For instance, if you run the `ant predeploy` command based on Example 12–19, it will call `webcenter:predeploy`, which will create the targeted EAR file from the generic EAR file. Then, `webcenter:deploy` will deploy the targeted EAR file to the OC4J instance specified in the `build.xml` file. `oracle:deploy` is the J2EE Ant task and it is available out-of-the-box.

See Also: *Oracle Containers for J2EE Deployment Guide*

12.4 Transporting Customizations Between Environments

You can export and import customizations made to pages and portlets (PDK-Java and WSRP version 2 producers) of an already deployed application by running the Predeployment tool in the Export and Import modes. The following sections describe the procedures to export customizations from the stage environment and import them into the production environment:

- Section 12.4.1, "Exporting Customizations"
- Section 12.4.2, "Importing Customizations"

12.4.1 Exporting Customizations

The Predeployment tool exports customizations to a customizations export archive file. This customizations export archive file is later imported into the MDS directory in the production environment.

To export customizations from the stage environment to the production environment, run the following syntax:

```
ORACLE_HOME/jdk/bin/java -jar adfp/lib/portlet-client-deploy.jar -export
-deployedapp <deployed_app_path> -target <targetedEAR>
```

where:

`targetedEAR` is the name of the customizations export archive file created by the export process. It contains all the customizations that can be imported into the same application deployed in another instance.

`deployed_app_path` is the location of the already deployed application from which you are exporting customizations, for example, *`ORACLE_HOME`*`/j2ee/home/applications/`*`YourApplication`*.

The Predeployment tool uses the customizations archive file that was exported during the export, to import customizations into the production environment.

12.4.2 Importing Customizations

You can import customizations that you exported to the customizations archive file, into your production environment. To import these customizations to your production environment, perform the following steps:

1. In the command prompt, run the following syntax:

   ```
   ORACLE_HOME/jdk/bin/java -jar adfp/lib/portlet-client-deploy.jar -import
   -source <genericEAR> -deployedapp <deployed_app_path>
   ```

   where:

   `genericEAR` is the customizations import archive file that was previously exported, which was created in the Export mode. It contains the customizations to be imported into the deployed application.

   `deployed_app_path` is the location of the already deployed application to which you want to import the customizations, for example, *`ORACLE_HOME`*`/j2ee/home/applications/`*`YourApplication`*.

2. If your WebCenter application also includes WSRP version 2 producers, then you must restart OC4J to see the customizations.

Customizations are now imported into your application in the production environment.

12.5 Updating Credentials in a Deployed Application

WebCenter applications persist secure properties associated with connections (stored in `connections.xml`) in an encrypted format in the `credential-store.xml` file. In design time, you can update these secure properties in Oracle JDeveloper, but to update secure properties postdeployment on the target system, you must use the Credentials Java Management Extension (JMX) MBean. This MBean can be accessed by running Application Server Control Console. Changes made to the connections' secure properties are updated in the `credential-store.xml` file.

Including the Credentials MBean in Your Application

To use the Credentials MBean, you must first ensure that it is available in your application and that your application is running. The MBean is included in the *ORACLE_HOME*/BC4J/lib/oracle.extapp.runtime.jar file that is part of your Oracle JDeveloper installation. In an Oracle Application Server installation, this JAR file is available in the *ORACLE_HOME*/adfp/lib directory.

The Credentials MBean is only available to applications that include the Oracle ADF authentication servlet. The Oracle ADF authentication servlet is already added to your application if you implemented security by using the Oracle ADF Security Wizard in Oracle JDeveloper.

For applications that do not use Oracle ADF Security for user authentication, but include connections with secure properties, you can add the AuthenticationServlet to your application manually by including the following XML code excerpt in your web.xml file:

```
<servlet>
   <servlet-name>AuthenticationServlet</servlet-name>
   <servlet-class>
      oracle.adf.share.security.authentication.AuthenticationServlet
   </servlet-class>
   <load-on-startup>0</load-on-startup>
</servlet>
```

Restart your Oracle Application Server instance after updating the web.xml file.

To ensure that the AuthenticationServlet is loaded when your application is started, it is necessary to include the <load-on-startup> element even though the application may never use it.

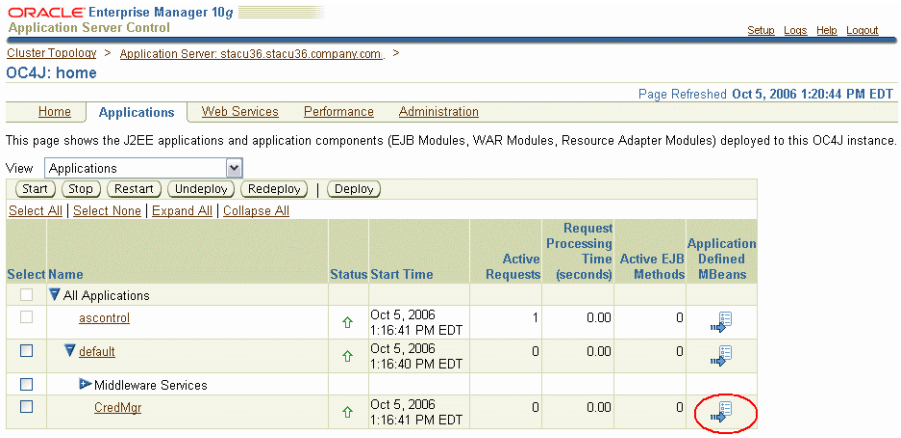
> **Note:** The MBean is available only when your application is running.

Updating Secure Connections Properties

To update secure connections properties using the Credentials MBean, perform the following steps:

1. Log in to the Application Server Control Console. The Home page is displayed.
2. Select the OC4J instance to which your application is deployed.
3. Click the Applications tab and navigate to the Application MBeans page as shown in Figure 12–26.

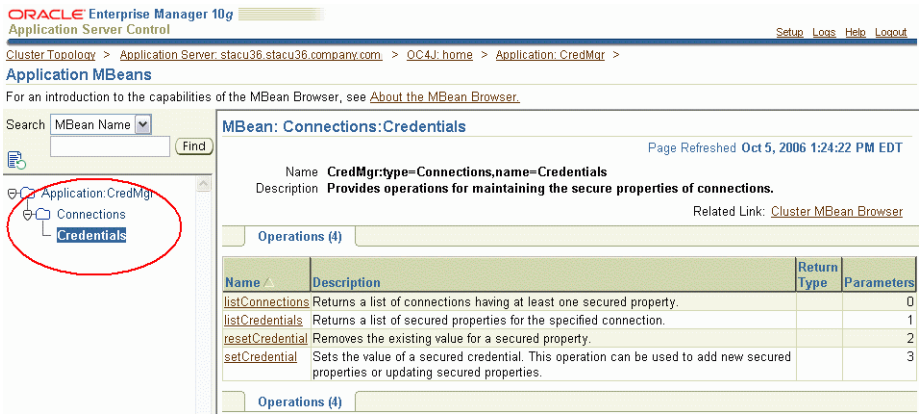
Figure 12–26 Applications Tab



4. From the list of MBean names in the left-hand frame, shown in Figure 12–27, select `Credentials`.

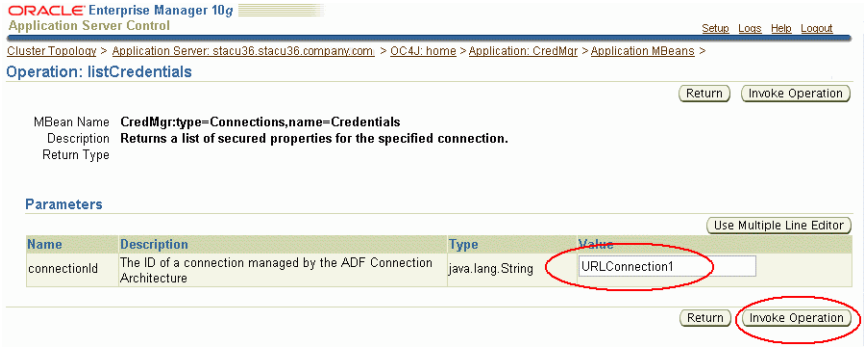
   The right-hand frame displays a description of this MBean and the Operations tab with the four operations that you can perform using this MBean.

Figure 12–27 Credentials MBean Page



5. Select the **listConnections** operation to list all connections that have at least one secured property.
6. On the Operation: listConnections page, click **Invoke Operation** to obtain a list of such connections. Write down the connection names, because you will need them to perform operations later.
7. Click **Return**.
8. Select the **listCredentials** operation on the MBean: Connections: Credentials page to list the credentials for any of the connections you wrote down in the previous step. The Operation: listCredentials page is displayed, as shown in Figure 12–28.

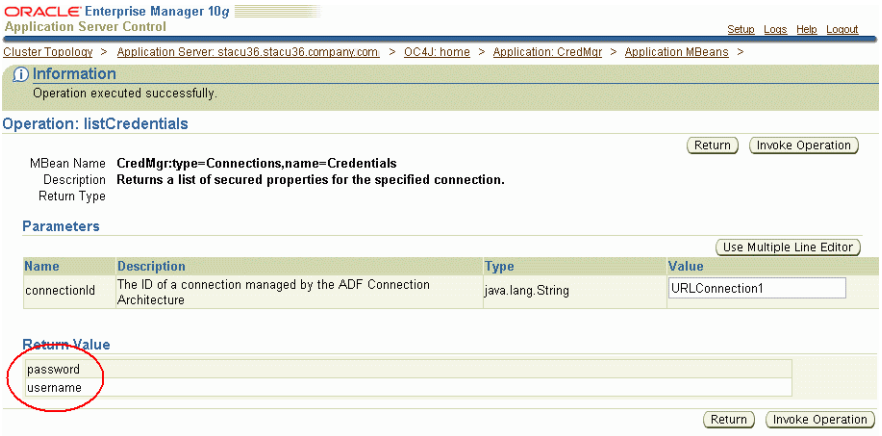
Figure 12–28 listCredentials Operation Page



9. Specify the connection name and click **Invoke Operation** to view the secured properties for that connection. All secure properties for the selected connection are listed as shown in Figure 12–29. Write down the credential names, because you will need them to perform the operations later.

> **Note:** To ensure a high level of security, the MBean does not expose the existing credential values.

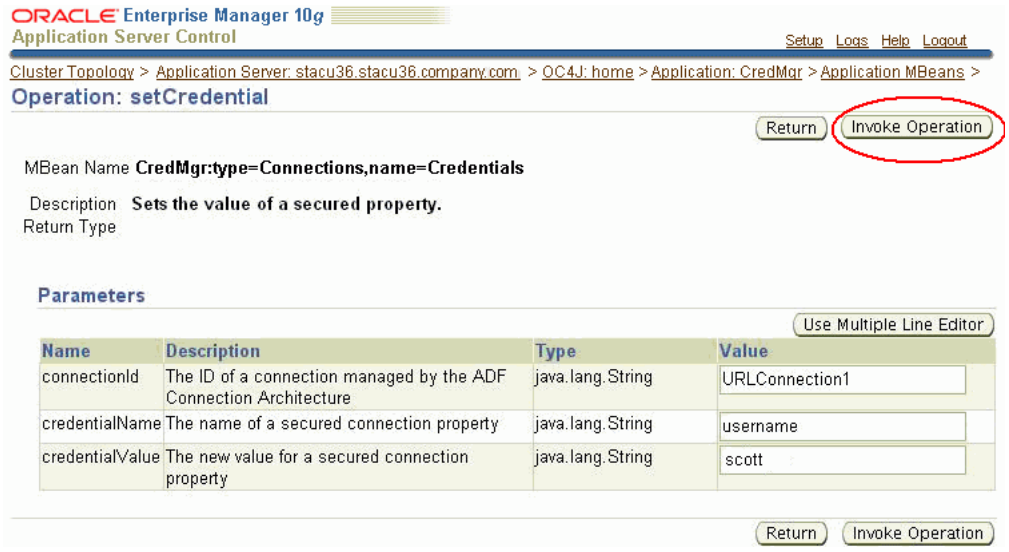
Figure 12–29 listCredentials Operation Page with Secure Properties Listed



10. Click **Return**.

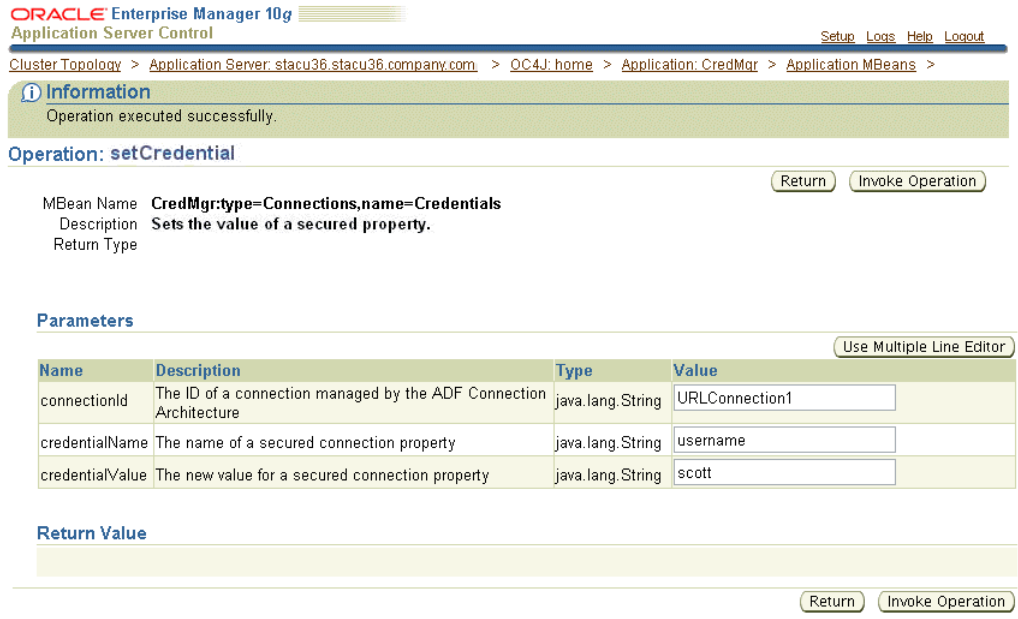
11. Select the **setCredential** operation on the MBean: Connections: Credentials page to update the secure properties of a connection. The Operation: setCredentials page is displayed, as shown in Figure 12–30.

Figure 12–30 setCredential Operation Page Used for Setting a Secured Property Value



12. Specify the connection name, secure property name, and new property value, and click **Invoke Operation** to change the value of a secured property. The resulting page shows that the secured property was updated successfully, as shown in Figure 12–31.

Figure 12–31 setCredentials Operation Results Page



Note: Ensure that you specify the connection and property name correctly. The property names and values are case-sensitive.

13. Click **Return**.

14. Select the **resetCredential** operation on the MBean: Connections: Credentials page to reset any of the secure properties of a connection.

15. Specify the connection name, secure property name and click **Invoke Operation** to reset the value of a secure property.
16. Repeat steps 4 to 15 as required. Changes you make to secure properties are committed immediately.

> **Note:** You can perform the four operations in any order. There is no sequence to be followed.

Copying the credential-store.xml File back to the Development Environment

When you update secure credentials in the deployed application, internally the `credential-store.xml` file is updated with this information. However, these credentials are lost when the application is redeployed from the development environment. This is because the `credential-store.xml` file in the deployed application is overwritten when the application is redeployed.

To ensure that the credentials you updated earlier are available in the redeployed application, you must copy the `credential-store.xml` file back to the development environment so that it can be repackaged with the application to be redeployed. However, credentials entered or modified after copying the file to the development environment and before redeploying the application will be lost.

In the development environment, you may have to update some credentials using appropriate wizards, as those credentials can be specific to stage and production environments. In other words, you must update credentials that you updated using the MBean in the stage and production environments, to original values so that they work in the development environment. Additionally, you must update the same credentials again after deployment, using the MBean.

12.6 Cloning WebCenter Applications

Cloning is the process of copying an existing installation to a different location while preserving its configuration. This section describes the following scenarios for cloning WebCenter applications:

- Section 12.6.1, "Expanding an Oracle Application Server Cluster"
- Section 12.6.2, "Using Cloning to Move from Stage to Production"

12.6.1 Expanding an Oracle Application Server Cluster

In this scenario, you create a new or another node in a cluster or a cluster-ready setup. All settings such as MDS, producer preferences, connections, and load balancing router (LBR) configuration remain unchanged.

The following are the perquisites for expanding an Oracle Application Server cluster:

- The Oracle home must be set up for clustering.
- The MDS location must be shared.
- The portlet producer preference stores (file or database) must be shared.

To use cloning to expand an Oracle Application Server cluster, follow the steps in section 9.7 Example: Using Cloning to Expand an Oracle Application Server Cluster of chapter titled Cloning Application Server Middle-Tier Instances" in *Oracle Application Server Administrator's Guide*. As you are going to deploy the cloned target instance in exactly the same environment as the cloned source instance, it is assumed that none of

the external dependencies shown in Figure 12–1, such as portlet producer end-points, have changed. Following the cloning steps outlined in the chapter titled Cloning Application Server Middle-Tier Instances" in Oracle Application Server Administrator's Guide will therefore be enough.

12.6.2 Using Cloning to Move from Stage to Production

As you can see in Figure 12–1, WebCenter applications typically have a number of external dependencies, such as, portlet producers and content repositories. When you move from stage to production, the configuration details for these external dependencies may change. For example, the database connection details for Oracle Content Database (Oracle Content DB) may be different in the production environment.

To use cloning for moving from stage to production, perform the following procedures:

1. Clone the Oracle home as described in chapter titled Cloning Application Server Middle-Tier Instances" in *Oracle Application Server Administrator's Guide*.
2. Perform the following procedures, if they are applicable to your environment:
   - If your WebCenter application uses content stored in Oracle Content DB using the Oracle Content DB data control, then use the Predeployment tool to reconfigure Oracle Content DB for producer connections, as described in Reconfiguring Parameters of Oracle Content DB-based Content Integration Applications.
   - If your WebCenter application uses content stored in OracleAS Portal using the OracleAS Portal-based content data control and the JNDI name used in the OracleAS Portal data source is different to that used in the development environment, then update OracleAS Portal connection information in the `connection.xml` file, as described in Reconfiguring OracleAS Portal and File System Adapters.
   - If your application uses encrypted attributes, such as, passwords in the credential store, then use the Credential Mbean to change them, as described in Section 12.5, "Updating Credentials in a Deployed Application".
   - If your WebCenter application uses content stored on the file system using the file system data control, then update file system connection information in the `connection.xml` file, as described in Reconfiguring OracleAS Portal and File System Adapters. However, the file system adapter is used only for development purposes.
   - If your MDS location has changed, then reconfigure the application to use the new location using the Predeployment tool, as described in Section 12.2.2, "Predeploying Your WebCenter Application".
   - If the stage and production servers are not using the same identity store, then migrate the security policy information from stage to production, as described in Section 12.2.4, "Migrating Security and Application Roles". If during this process you must migrate from an XML provider (`sytem-jazn-data.xml`) to an LDAP provider, then you must run the JAZN Migration tool and the `ldapmodify` command, as discussed in Example 12–12 and Example 12–13.
3. Export portlet and page customizations from stage to production, as described in Section 12.4, "Transporting Customizations Between Environments". These customizations are first exported to a customizations export archive file in the

stage environment. Then, this customizations archive file is imported into the MDS directory in the production environment.

4. Test your cloned application to see if it is working as expected.

12.7 Configuring Your WebCenter Application to Run in a Distributed Environment

For information about configuring your WebCenter application to run in a distributed environment, see *Oracle Application Server Enterprise Deployment Guide* and chapter titled "Active-Active Topologies" in *Oracle Application Server High Availability Guide*.

12.8 Undeploying Your WebCenter Application

You can undeploy your application either using Application Server Control Console or the command-line interface. It is recommended that you use Application Server Control Console; however, for your better understanding, this sections also describes the syntax to undeploy your WebCenter application using the command-line interface.

Read the following sections to learn the procedures to undeploy your WebCenter application:

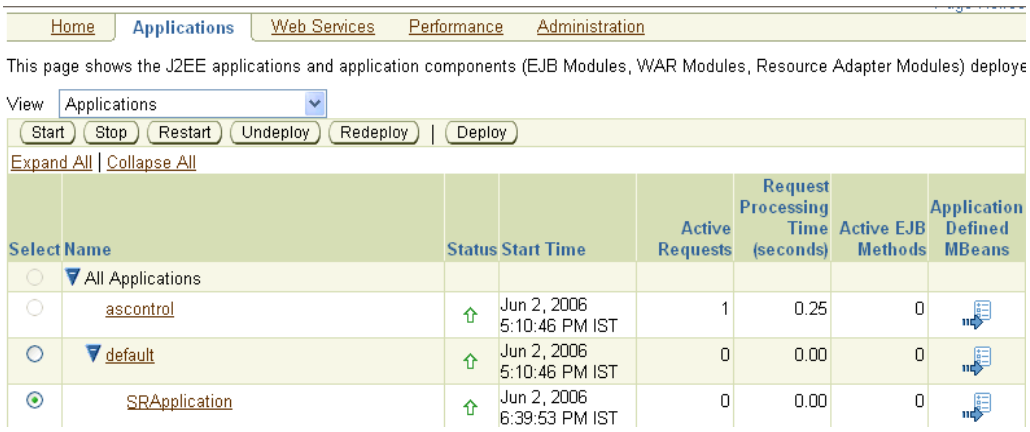
- Section 12.8.1, "Undeploying Your WebCenter Application Using Application Server Control Console"
- Section 12.8.2, "Undeploying Your WebCenter Application Using the Command Line"

12.8.1 Undeploying Your WebCenter Application Using Application Server Control Console

To undeploy your application, perform the following steps:

1. Log in to the Application Server Control Console. The Home page is displayed.
2. Click the **Applications** tab. The Applications page is displayed, as shown in Figure 12–32.

Figure 12–32 Applications Page in Application Server Control Console

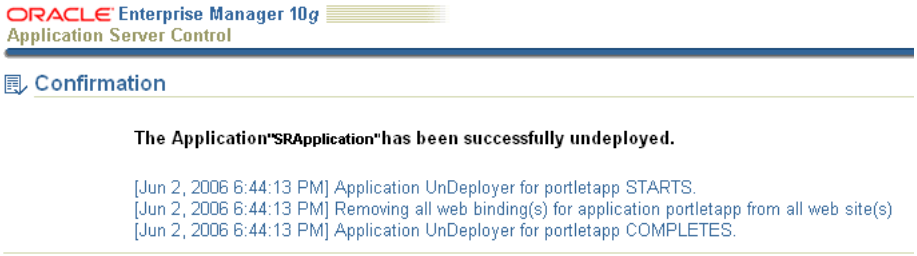


3. Click **Undeploy**. The Undeploy Application warning page is displayed, as shown in Figure 12–33.

Figure 12–33 Undeploy Page in Oracle Enterprise Manager

4. Click **Yes** to undeploy your application. The undeployment Confirmation page is displayed, as shown in Figure 12–34.

Figure 12–34 Undeploy Confirmation Page in Application Server Control Console



Your application is undeployed now. It is no longer displayed under the Applications tab. However, the MDS directory of your application is still available.

12.8.2 Undeploying Your WebCenter Application Using the Command Line

To undeploy your application using the command line, run the following:

```
ORACLE_HOME/jdk/bin/java -jar admin.jar ormi://localhost/<ormi port> <admin_
username> <admin_pwd> -undeploy
-deploymentName <deployment_name> <true|false>
```

13

Monitoring Your WebCenter Application

Oracle Enterprise Manager 10*g* Application Server Control Console is the administration console for the Oracle Application Server and it provides a Web-based user interface for deploying, configuring, and monitoring WebCenter applications. The Application Server Control Console also monitors the real-time performance of any producers and portlets that are used in WebCenter application.

This section covers the following topics:

- "Displaying the Application Server Control Console"
- "Navigating to WebCenter Application Pages"
- "Interpreting the Information in Oracle Enterprise Manager 10g"

13.1 Displaying the Application Server Control Console

To access the Application Server Control Console, perform the following steps:

1. Navigate to the following URL:

   `http://`*`host_name.domain`*`:`*`port`*`/em`

   For example:

   `http://myhost.acme.com:8988/em.`

2. Log in.

To find the URL for your console, look at `readme.txt`. After installation, this text file is saved to the following location:

On UNIX: `ORACLE_HOME/install/readme.txt`

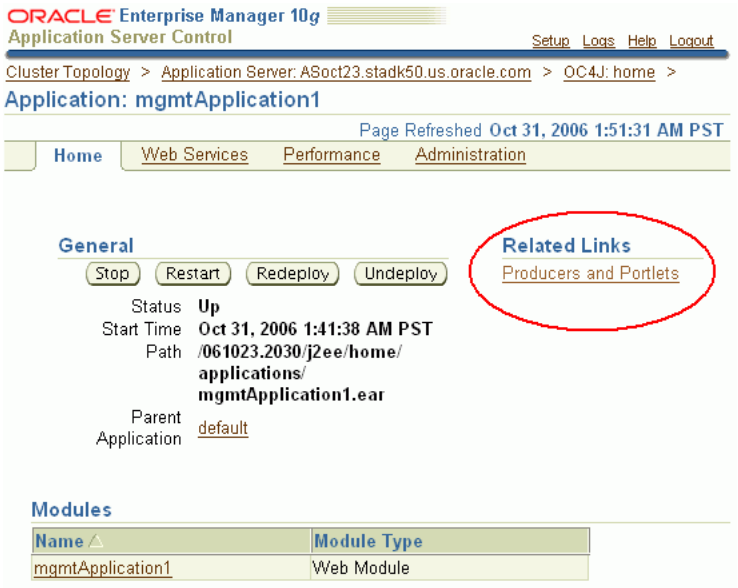
On Windows: `ORACLE_HOME\install\readme.txt`

13.2 Navigating to WebCenter Application Pages

To navigate to pages specific to WebCenter applications, perform the following steps:

1. Display the Application Server Control Console.
2. Navigate to the home page of your WebCenter application (Figure 13–1).

Figure 13–1 WebCenter Application Home Page



3. From the Related Links section, click **Producers and Portlets**.

Note: If you do not see a *Producers and Portlets* link, then either:

- No producers or portlets exist for the selected application.
- Statistics are not yet available for any producer or portlet used by the application. Statistics become available when a portlet producer is accessed for the first time. When the Oracle Containers for J2EE (OC4J) instance is restarted, you do not see this link until a page containing portlets is accessed.

The Portlet Producers page is displayed. This page lists every portlet producer accessed by the WebCenter application and summarizes producer performance. For more detail, see Portlet Producers - Summary Page.

4. From the Portlet Producers page you can do the following tasks:
   - Monitor the current status and performance of portlet producers being accessed by a WebCenter application.
   - Drill down to more detailed information for a particular producer and any of its portlets. First, click a producer name (Portlet Producer - Detail Page), and then click a portlet name (Portlet - Detail Page).
   - Click **Reset Statistics** if you want to start collecting a new set of statistics for all the producers shown on this page.

13.3 Interpreting the Information in Oracle Enterprise Manager 10*g*

Information specific to WebCenter applications is available on the following pages:

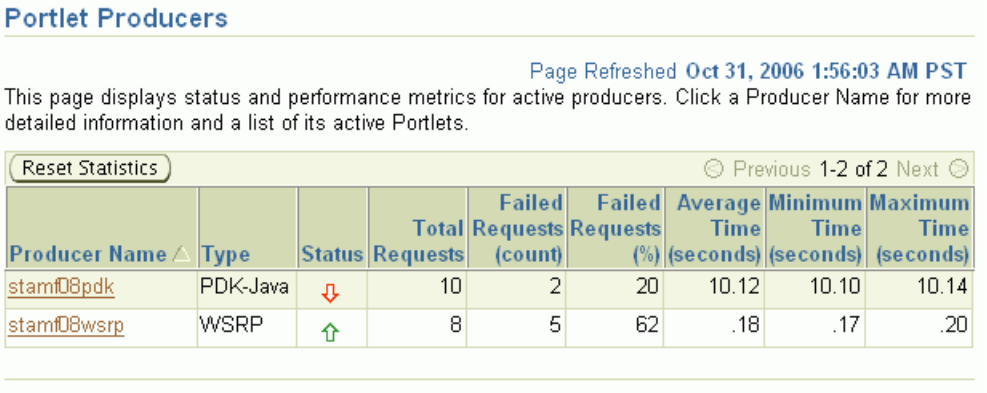
- Portlet Producers - Summary Page
- Portlet Producer - Detail Page
- Portlet - Detail Page

Metrics displayed on these pages are based on data collected from running WebCenter applications. Typically, data collection begins at application start-up. If you want to start collecting a new set of metrics, then click the **Reset Statistics** button on the appropriate page—you can reset the metrics for all portlet producers, a single producer, or a single portlet.

13.3.1 Portlet Producers - Summary Page

This page shows status and performance information for all portlet producers accessed by a WebCenter application (Figure 13–2).

Figure 13–2 Portlet Producers Page



The information available on this page is described in the following table:

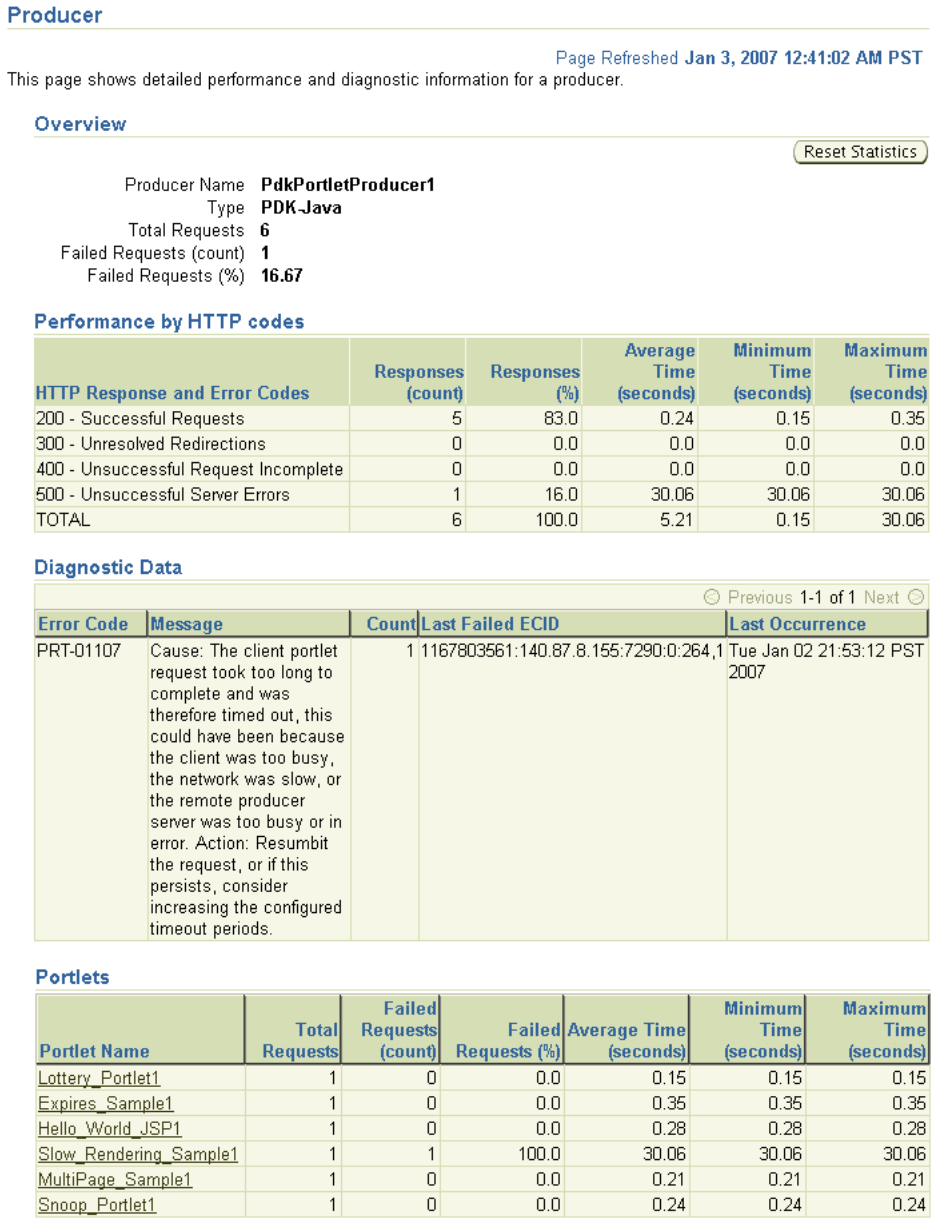
Field	Description
Reset Statistics	Click to reset statistics for all portlet producers accessed by this WebCenter application and collect a new set of statistics from scratch. Use this option whilst debugging performance issues and to establish performance metrics for each request.
Producer Name	Name of the portlet producer. Click the name of a portlet producer to go to the "Portlet Producer - Detail Page".
Type	Either **PDK-Java** or **WSRP**. ▪ PDK-Java portlet producer - deployed to a J2EE application server, which is often remote and communicates through Simple Object Access Protocol (SOAP) over HTTP. ▪ WSRP portlet producer - Web Services for Remote Portlets (WSRP) is a Web services standard that enables interoperability between a standards enabled container and any WSRP application.
Status	Indicates whether the portlet producer is contactable. Either **Up** or **Down**. ▪ Up - The portlet producer is contactable. ▪ Down - The portlet producer cannot be contacted. The producer instance might be down, or there could be some network connectivity issues.

Field	Description
Total Requests	Total number of requests made to each portlet producer. This metric measures each WebCenter application-related portlet request and therefore, due to cache hits, errors, or timeouts on the application, this total may be higher than the number of actual HTTP requests made to the producer server. By sorting the table on this column, you can find the most frequently accessed portlet producer in your WebCenter application.
Failed Requests (count)	Number of portlet producer requests that failed. Use this statistic in conjunction with *Failed Request (%)* to identify producers with the most failures, or the highest failure rate. Any request that fails is included in this count. This includes WebCenter application-related failures such as timeouts and internal errors, as well as remote/server failures such as requests returned with response codes HTTP4xx or HTTP5xx, responses with a bad content type, and SOAP faults, where applicable.
Failed Requests (%)	Percentage of portlet producer requests that failed.
Average Time (seconds)	Average response time for requests to a portlet producer, regardless of the result. (Average response time data for each result type is available on the Portlet Producer - Detail Page.) Use this metric to detect non-performant portlet producers. If you use this metric in conjunction with the metric *Total Requests*, then you can prioritize which producer to focus on. The Average Time metric can also help you determine the best-case performance scenario for a producer. Ideally, the average and maximum response times (*Maximum Time*) should be close to the minimum response time (*Minimum Time*).
Minimum Time (seconds)	Minimum response time for a portlet producer request, regardless of the result.
Maximum Time (seconds)	Maximum response time for a portlet producer request, regardless of the result. Use this metric to find the worst performing portlet producer.

13.3.2 Portlet Producer - Detail Page

This page shows performance and diagnostic information for a portlet producer accessed by a WebCenter application, as shown in Figure 13–3.

Figure 13–3 Producer Detail Page



Producer

Page Refreshed **Jan 3, 2007 12:41:02 AM PST**

This page shows detailed performance and diagnostic information for a producer.

Overview

Reset Statistics

Producer Name **PdkPortletProducer1**
Type **PDK-Java**
Total Requests **6**
Failed Requests (count) **1**
Failed Requests (%) **16.67**

Performance by HTTP codes

HTTP Response and Error Codes	Responses (count)	Responses (%)	Average Time (seconds)	Minimum Time (seconds)	Maximum Time (seconds)
200 - Successful Requests	5	83.0	0.24	0.15	0.35
300 - Unresolved Redirections	0	0.0	0.0	0.0	0.0
400 - Unsuccessful Request Incomplete	0	0.0	0.0	0.0	0.0
500 - Unsuccessful Server Errors	1	16.0	30.06	30.06	30.06
TOTAL	6	100.0	5.21	0.15	30.06

Diagnostic Data

Previous 1-1 of 1 Next

Error Code	Message	Count	Last Failed ECID	Last Occurrence
PRT-01107	Cause: The client portlet request took too long to complete and was therefore timed out, this could have been because the client was too busy, the network was slow, or the remote producer server was too busy or in error. Action: Resumbit the request, or if this persists, consider increasing the configured timeout periods.	1	1167803561:140.87.8.155:7290:0:264,1	Tue Jan 02 21:53:12 PST 2007

Portlets

Portlet Name	Total Requests	Failed Requests (count)	Failed Requests (%)	Average Time (seconds)	Minimum Time (seconds)	Maximum Time (seconds)
Lottery_Portlet1	1	0	0.0	0.15	0.15	0.15
Expires_Sample1	1	0	0.0	0.35	0.35	0.35
Hello_World_JSP1	1	0	0.0	0.28	0.28	0.28
Slow_Rendering_Sample1	1	1	100.0	30.06	30.06	30.06
MultiPage_Sample1	1	0	0.0	0.21	0.21	0.21
Snoop_Portlet1	1	0	0.0	0.24	0.24	0.24

The information available on this page is described in the following tables:

- Overview
- Performance by HTTP Codes
- Diagnostic Data
- Portlets

Overview

The Overview table displays overall performance metrics for a portlet producer.

Field	Description
Reset Statistics	Click to reset statistics for a specific portlet producer and start collecting a new set of statistics from scratch. Use this option whilst debugging performance issues and to establish performance metrics for each request.
Producer Name	Name of the portlet producer.
Type	Either **PDK-Java** or **WSRP**. ■ PDK-Java portlet producer - deployed to a J2EE application server, which is often remote and communicates through Simple Object Access Protocol (SOAP) over HTTP. ■ WSRP portlet producer - WSRP is a Web services standard that enables interoperability between a standards enabled container and any WSRP application.
Total Requests	Total number of requests handled by a portlet producer.
Failed Requests (count)	Number of portlet producer requests that failed. Any request that fails is included in this count. This includes WebCenter application-related failures such as timeouts and internal errors, as well as remote/server failures such as requests returned with response codes HTTP4xx or HTTP5xx, responses with a bad content type, and SOAP faults, where applicable.
Failed Requests (%)	Percentage of portlet producer requests that failed.

Performance by HTTP Codes

The Performance by HTTP Codes table provides a detailed analysis of portlet producer requests, by HTTP response code. For each response code the following information is provided:

- Responses (count) - Number of portlet producer requests
- Responses (%) - Percentage of overall portlet producer requests
- Average Time - Average response time (in seconds)
- Minimum Time - Minimum response time (in seconds)
- Maximum Time - Maximum response time (in seconds)

The breakdown of performance statistics by HTTP response code can help you identify which factors are driving up the total average response time. For example, failures due to portlet producer timeouts would adversely affect the total average response time.

> **Note:** These metrics measure each WebCenter application-related portlet request and therefore, due to cache hits, errors, or timeouts on the application, the counts and times displayed in this table are slightly different to actual HTTP requests made to the producer server.

HTTP Response and Error Codes	Description
200 -Successful Requests	Portlet producer requests that succeeded, including requests returning HTTP2xx response codes and requests serviced from the cache without the need for an HTTP request.

HTTP Response and Error Codes	Description
300 - Unresolved Redirections	Portlet producer requests that return any HTTP3xx response code.
400 - Unsuccessful Request Incomplete	Portlet producer requests that return any HTTP4xx response code.
500 - Unsuccessful Server Errors	Portlet producer requests that return HTTP5xx response codes, or which failed due to a WebCenter application-related error, timeout, bad content type response, or SOAP fault.
Total	Total number of requests.

Diagnostic Data

The Diagnostic Data table provides detailed diagnostic information relating to failed portlet producer requests.

When a portlet producer request fails the error is mapped to an application error code (PRT-xxxx) and the details are summarized in this table.

The table lists diagnostic information by error code and shows the number of times the error occurred. In addition, the table provides cause and action information relating to the error code.

Use this table to find the most frequent errors and also the last error for a producer. If the error is not due to the portlet producer, then use the ECID (Execution Context Identifier) to find error messages that relate to other Oracle Application Server components.

Field	Description
Error Code	Error code of the failed request.
Message	Cause of the error and appropriate corrective action.
Count	Number of requests raising this error code.

Field	Description
Last Failed ECID	Execution Context Identifier (ECID) of the last request that raised this error code. Each request has a unique number (ECID) to enable administrators to track requests and this is very useful whilst troubleshooting a request as it passes through Oracle Application Server components. To find occurrences of this ECID in OC4J log files, enter the ECID displayed here in the Oracle Enterprise Manager 10*g Search Logs* page. This will enable you to correlate messages from several components and diagnose application errors or performance problems. To access the *Search Log*s page (Figure 13–5): 1. Click the **Logs** link. 2. Select the log files you want to search. 3. Click the **Search** button. 4. Expand **Advanced Search Options**. 5. Choose **Execution Context ID** from the Log Message Field list. 6. Enter the ECID in the **Value** field. 7. Click **Search**.
Last Occurrence	Date and time the error last occurred.

Portlets

This table displays overall performance metrics for portlets serviced by this portlet producer.

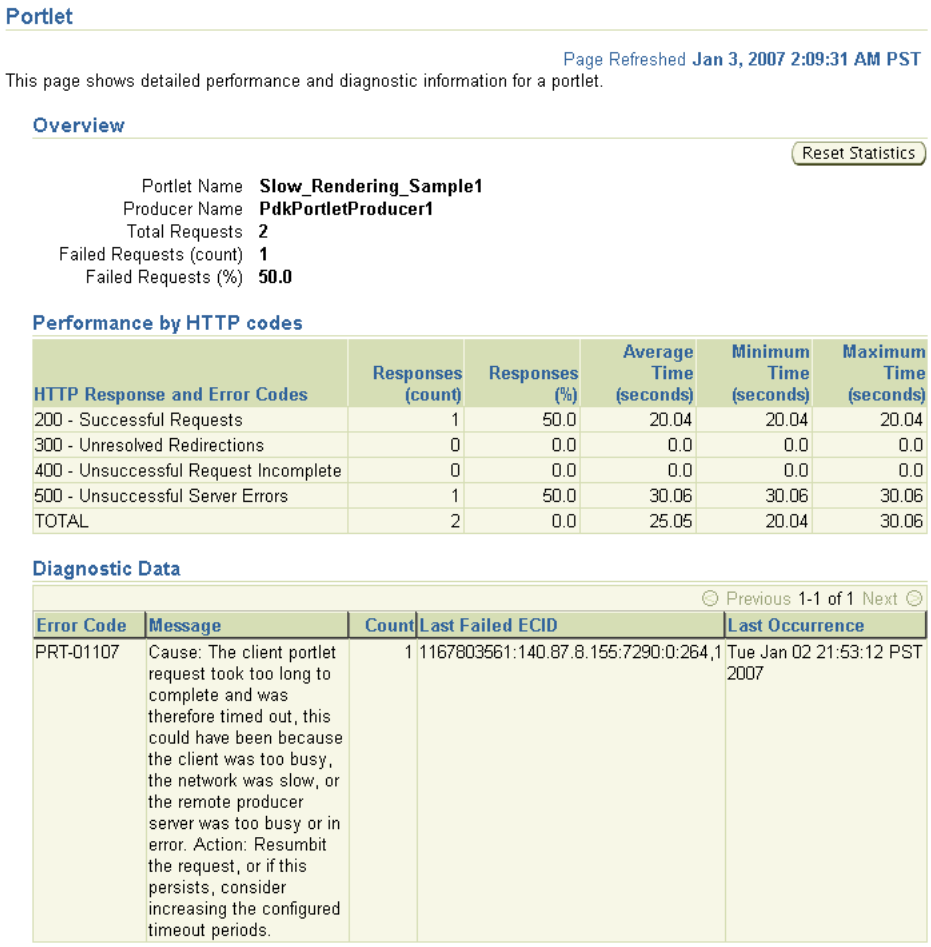
Field	Description
Portlet Name	Name of the portlet. Click the name to go to the "Portlet - Detail Page".
Total Requests	Total number of requests made to each portlet. This metric measures each WebCenter application-related portlet request and therefore, due to cache hits, errors, or timeouts on the application, this total may be higher than the number of actual HTTP requests made to the producer server. By sorting the table on this column, you can find the most frequently accessed portlet in your WebCenter application.
Failed Requests (count)	Number of portlet requests that failed. Use this statistic in conjunction with *Failed Request (%)* to identify portlets with the most failures, or the highest failure rate. Any request that fails is included in this count. This includes WebCenter application-related failures such as timeouts and internal errors, as well as remote/server failures such as requests returned with response codes HTTP4xx or HTTP5xx, responses with a bad content type, and SOAP faults, where applicable.

Field	Description
Failed Requests (%)	Percentage of portlet requests that failed.
Average Time (seconds)	Average response time for requests to the portlet, regardless of the result. Use this metric to detect non-performant portlets. If you use this metric in conjunction with the metric *Total Requests*, then you can prioritize which portlet to focus on. The Average Time metric can also help you determine the best-case performance scenario for a portlet. Ideally, the average and maximum response times (*Maximum Time*) should be close to the minimum response time (*Minimum Time*).
Minimum Time (seconds)	Minimum response time for a portlet request, regardless of the result.
Maximum Time (seconds)	Maximum response time for a portlet request, regardless of the result. Use this metric to find the worst performing portlet.

13.3.3 Portlet - Detail Page

This page shows detailed performance and diagnostic information for a portlet accessed by a WebCenter application (Figure 13–4).

Figure 13–4 Portlet Detail Page



The information available on this page is described in the following tables:

- Overview
- Performance by HTTP Codes
- Diagnostic Data

Overview

The Overview table displays overall performance metrics for a portlet.

Field	Description
Portlet Name	Name of the portlet.
Producer Name	Name of the producer providing the portlet.
Total Requests	Total number of requests handled by the portlet.

Field	Description
Failed Requests (count)	Statistics relating to unsuccessful portlet requests: ■ Count - Number of unsuccessful portlet requests. ■ % - Percentage of portlet requests that were unsuccessful. ■ Avg. Time - Average response time of all unsuccessful requests to the portlet. ■ Min. Time - Minimum response time, in seconds, of a unsuccessful portlet request. ■ Max. Time - Maximum response time, in seconds, of a unsuccessful portlet request.
Failed Requests (%)	Statistics relating to unsuccessful portlet requests: ■ Count - Number of unsuccessful portlet requests. ■ % - Percentage of portlet requests that were unsuccessful. ■ Avg. Time - Average response time of all unsuccessful requests to this portlet. ■ Min. Time - Minimum response time, in seconds, of a unsuccessful portlet request. ■ Max. Time - Maximum response time, in seconds, of a unsuccessful portlet request.

Performance by HTTP Codes

The Performance by HTTP Codes table provides a detailed analysis of portlet requests, by HTTP response code. For each response code the following information is provided:

- Responses (count) - Total number of portlet requests
- Responses (%) - Percentage of portlet requests
- Average Time - Average response time (in seconds)
- Minimum Time - Minimum response time (in seconds)
- Maximum Time - Maximum response time (in seconds)

The breakdown of performance statistics by HTTP response code can help you identify which factors are driving up the total average response time. For example, failures due to portlet producer timeouts would adversely affect the total average response time.

> **Note:** Successful requests that do not require an HTTP request to the remote producer, such as a cache hit, are included in the HTTP2xx category. Requests that fail for non-HTTP status code reasons, such as a WebCenter application-related timeout or error, bad content type response or SOAP fault, are included in the HTTP5xx category.

HTTP Response and Error Codes	Description
200 -Successful Requests	Portlet requests that return any HTTP2xx response code, or which were successful without requiring an HTTP request to the remote producer, for example, a cache hit.
300 - Unresolved Redirections	Portlet requests that return any HTTP3xx response code.

HTTP Response and Error Codes	Description
400 - Unsuccessful Request Incomplete	Portlet requests that return any HTTP4xx response code.
500 - Unsuccessful Server Errors	Portlet requests that failed for any reason, including requests that return HTTP5xx response codes, or which failed due to a WebCenter application-related error, timeout, bad content type response, or SOAP fault.
Total	Total number of requests.

Diagnostic Data

The Diagnostic Data table provides detailed diagnostic information relating to failed portlet requests.

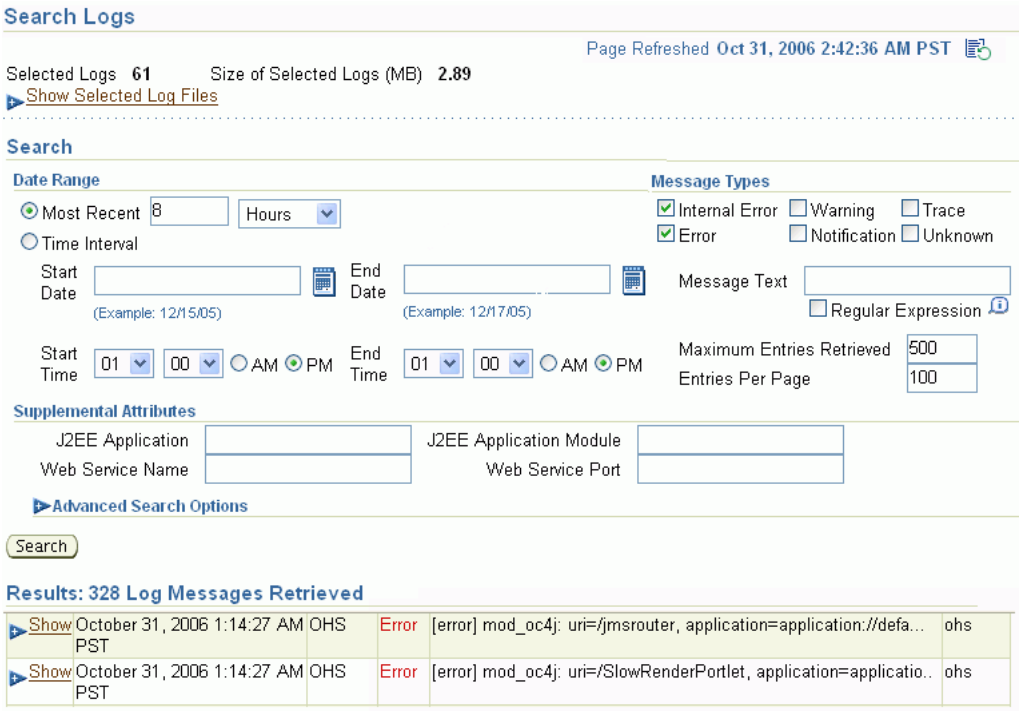
When a portlet request fails the error is mapped to an application error code (PRT-xxxx) and the details are summarized in this table.

The table lists diagnostic information by error code and shows the number of times the error occurred. In addition, the table provides cause and action information relating to the error code.

Use this table to find the most frequent errors and also the last error for this portlet. If the error is not due to the portlet, then use the ECID to find error messages that relate to other components.

Field	Description
Error Code	Error code of the failed request.
Message	Cause of the error and appropriate corrective action.
Count	Number of requests raising this error code.
Last Failed ECID	Execution Context Identifier (ECID) of the last request that raised this error code. Each request has a unique number (ECID) to enable administrators to track requests and this is very useful whilst troubleshooting a request as it passes through Oracle Application Server components. To find occurrences of this ECID in OC4J log files, enter the ECID displayed here in the Oracle Enterprise Manager 10*g Search Logs* page. This will enable you to correlate messages from several components and diagnose application errors or performance problems. To access the *Search Log*s page (Figure 13–5): 1. Click the **Logs** link. 2. Select the log files you want to search. 3. Click the **Search** button. 4. Expand **Advanced Search Options**. 5. Choose **Execution Context ID** from the Log Message Field list. 6. Enter the ECID in the **Value** field. 7. Click **Search**.
Last Occurrence	Date and time the error last occurred.

Figure 13–5 Search Logs Page



Part IV

Building Portlets

Part IV contains the following chapters:

14

Understanding Portlets

Oracle WebCenter Framework has all the tools a Java developer needs for developing portlets and adding them to WebCenter application pages. Portlet creation wizards enable rapid development of portlet frameworks.

This chapter provides an overview of portlets. It contains the following sections:

- Section 14.1, "Understanding Portlets"
- Section 14.2, "Portlet Anatomy"
- Section 14.3, "Portlet Resources"

14.1 Understanding Portlets

A portlet is a reusable Web component that can draw content from many different sources. Figure 14–1 illustrates the Most Active Customers portlet, a portlet constructed at run time using OmniPortlet. For more information about OmniPortlet, see Section 14.3.4, "OmniPortlet".

Figure 14–1 The Most Frequent Customers Portlet

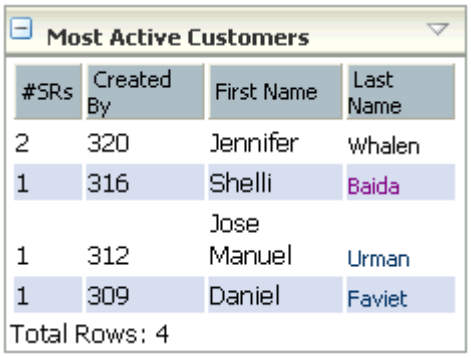


Figure 14–2 illustrates the Customer Details portlet, which works in conjunction with the Most Active Customers portlet. In Most Active Customers, users click a customer name to open the Customer Details portlet and view such information as the customer's e-mail and street addresses.

Figure 14–2 Customer Details Portlet

Portlets provide a means of presenting data from multiple sources in a meaningful and related way. Portlets can display excerpts of other Web sites, generate summaries of key information, perform searches, and access assembled collections of information from a variety of data sources. Because different portlets can be placed on a common page, the user receives a single-source experience. In reality, the content may be derived from multiple sources.

In a WebCenter application, a portlet may or may not be rendered in an inline frame, or `<iframe>`. Inline frames enable the placement of a document within a rectangular region that includes scrollbars and borders.

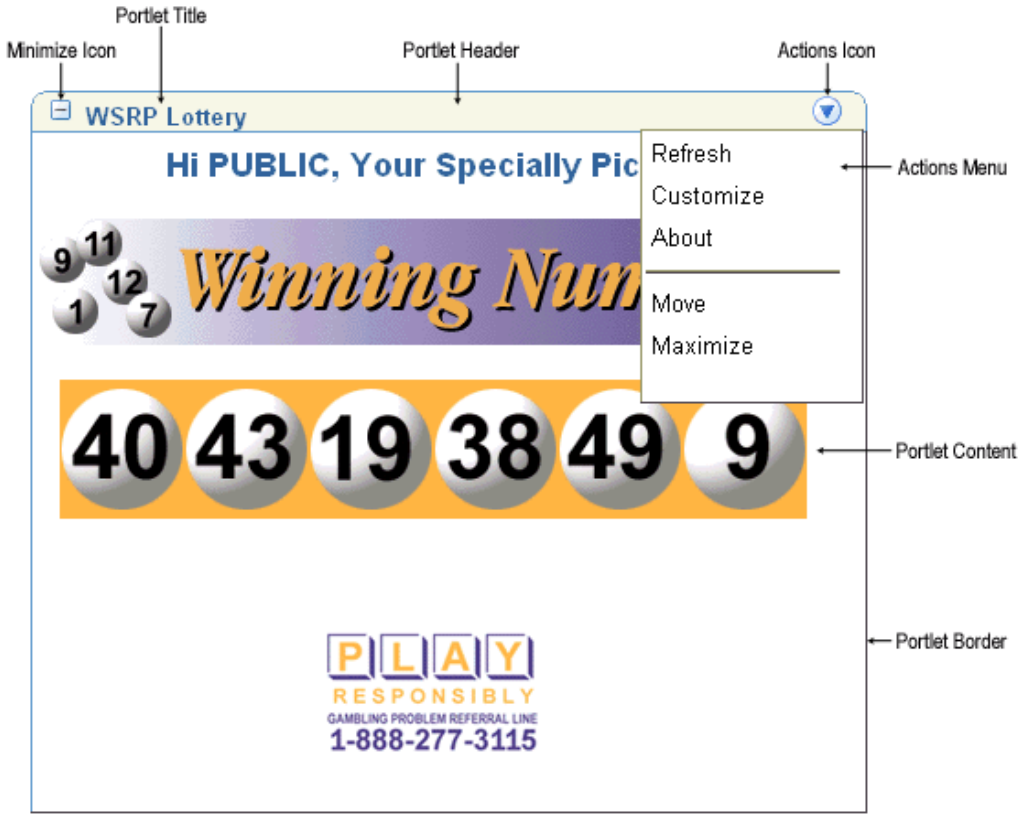
Note: For more information about portlets and inline frames, see Section 4.3.3.6, "iframes and form Tags".

Within this inline frame, portlets can display many types of content, including HTML, formatted text, images, or elements of an HTML form.

14.2 Portlet Anatomy

Portlet anatomy is the visual representation of the portlet on a page. Figure 14–3 illustrates a typical portlet anatomy.

Figure 14–3 Portlet Anatomy



What is rendered on the page is controlled not only by the portlet's own logic, but by the attributes of the portlet tag that binds the portlet to the page. Values for these attributes are specified at the design-time of the application that consumes the portlet, rather than through the portlet's own logic.

The steps to including a portlet on an application page include the following:

1. Create the portlet.
2. Deploy the portlet to a portlet container (a producer).
3. Register the producer with the application that will consume the portlet.
4. Add the portlet to an application page.
5. Specify values for attributes of the portlet tag that binds the portlet to the application page.

> **Note:** For information about attributes of the `adfp:portlet` tag, see Section 4.3.3, "Setting Attribute Values for the adfp:portlet Tag".

For example, at application design time you can specify through portlet tag attributes that the run-time portlet should display a header and that its border should be of a specified thickness and color. In the header, you can include a portlet title as well as an Actions menu icon. The Actions menu icon is displayed on a portlet only when the portlet header is displayed. If you choose not to display a header, then the Actions menu is displayed on a FadeIn-FadeOut toolbar that renders on a mouse rollover.

These elements are sometimes referred to as *portlet chrome*. The appearance of portlet chrome can be controlled through a style sheet as well as through style-related attributes of the `adfp:portlet` tag. The values of style-related attributes take precedence over styles specified through a style sheet, or *skin*.

> **Note:** For information about skins and style-related attributes, see Chapter 9, "Defining and Applying Styles to Core Customizable Components".

Through portlet tag attributes, you can include or omit display commands on the Actions menu. Actions menu items controlled through portlet tag attributes include *Maximize* and *Restore*. Maximize causes the maximized portlet to displace all other displayed portlets. These are displayed again once a user selects *Restore*.

> **Note:** The Maximize attribute is meaningful for a portlet only when the portlet is placed inside a `PanelCustomizable` core customizable component.

Other Actions menu items controlled through portlet tag attributes include the display or omission of the mode settings that were specified when the portlet was developed. If the portlet was built without including additional modes, then these commands do not appear on the Actions menu even when you indicate that they should through portlet tag attributes. In other words, portlet tag attributes are sometimes merely on/off switches that enable or disable the portlet's own built-in functionality.

Mode settings that appear on the Actions menu include such modes as *About*, *Help*, *Personalize*, and *Customize*.

Users select Personalize to alter their personal view of the portlet. The Personalize command is displayed on the Actions menu only to authenticated users (that is, users who are logged in). It does not display to Public or unauthenticated users. You must implement some form of application security for users to be able to personalize their portlet views.

> **Note:** If you are a developer creating portlets, and you want to test the Personalize mode without creating a complete security model for your application, then see Section 10.6, "Configuring Basic Authentication for Testing Portlet Personalization".

Customization enables application administrators to edit a portlet's default settings at run time. All users see the results of a customization.

> **Note:** A typical customization setting is Portlet Title. At run time, the portlet administrator can determine what title should appear in the portlet header. The Portlet Title can also be set at design-time through portlet properties, using the `text` attribute of the `adfp:portlet` tag. Consider however that supplying a value to the `text` attribute at design-time prevents customization of the Portlet Title at run time.
>
> For additional information about portlet tag attributes, see Section 4.3.3, "Setting Attribute Values for the adfp:portlet Tag".

14.3 Portlet Resources

Portlet resources include the many prebuilt portlets available out of the box from many sources, including Oracle Application Server Portal (OracleAS Portal), Oracle E-Business Suite, and third-party sources. Portlet resources also include programmatic portlets built through the WebCenter application's JSR 168 (standards-based) and Oracle PDK-Java wizards, and through other portlet-building tools. Each of these tools offers different product features that are targeted toward different developer roles.

This section describes different portlet resources, suggests the level of expertise required to use them, and provides examples of when they might best be used. It includes the following subsections:

- Section 14.3.1, "Rich Text Portlet"
- Section 14.3.2, "Prebuilt Portlets"
- Section 14.3.3, "Web Clipping"
- Section 14.3.4, "OmniPortlet"
- Section 14.3.5, "Programmatic Portlets"

This section introduces you to the various portlet resources. For specific information about each tool and its benefits, see Chapter 15, "Portlet Technologies Matrix".

14.3.1 Rich Text Portlet

What Is It?

The Rich Text portlet, based on the WSRP 2.0 standard, offers browser-based rich text editing at run time. Select **Customize** from the portlet's **Actions** menu to invoke a toolbar with all the rich-text editing tools you will need to insert, update, and format display text. Click the editor's **Submit** button to save your changes and hide the toolbar. (Selecting **Refresh** from the portlet's Actions menu also hides the toolbar).

With the Rich Text portlet, when an authorized user selects the **Customize** menu item the portlet enters edit mode while the page that contains the portlet remains in view mode.

Portlet configuration settings—available at design-time—include controls for the portlet's name, description, and display settings.

Who Is the Intended User?

Application developers can add the Rich Text portlet to a page and provide initial content. Unless the application developer prohibits it, end users with the appropriate privileges can update the content of the Rich Text Portlet at run time.

When Should It Be Used?

The Rich Text portlet is a useful tool for run time posting of enterprise announcements and news items. It provides straightforward, easy-to-use controls for entering and formatting display text.

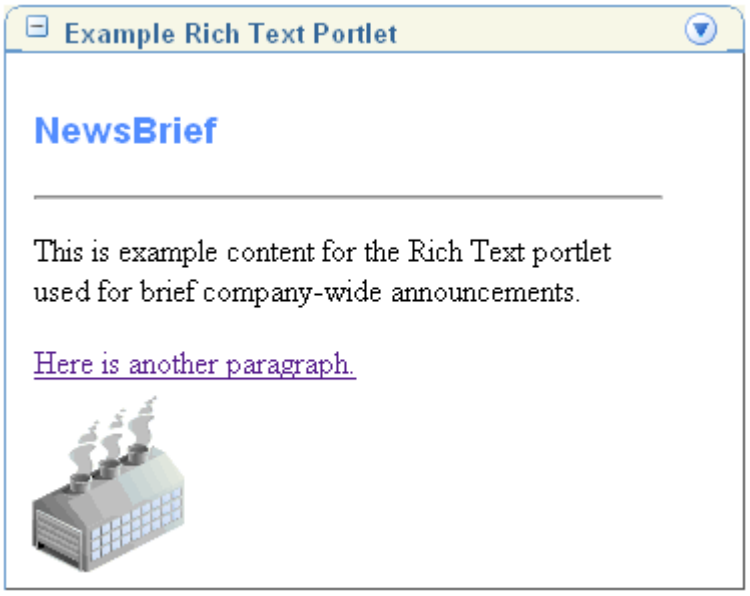
Using the Rich Text Portlet

This section steps you through the process of accessing Rich Text portlet controls and tours you through them. To illustrate the process of using the portlet, a simple example of display text was created using a cascading style sheet, text, color, a link, and an image. Remember that you use Rich Text portlet controls at run time, that is, once the portlet has been placed on an application page, and the application page is run.

The following exercise steps you through run time creation of a simple display text example, similar to Figure 14–4.

> **Note:** At design-time (when you place the Rich Text portlet on an application page), the portlet's `allModesSharedScreen` attribute should be set to `true`. This enables all portlet modes to display without leaving the context of a given page. You can do this by adding the attribute to the `adfp:portlet` tag or by changing the value through the Oracle JDeveloper Property Inspector.
>
> For information about obtaining the Rich Text portlet, see Section 3.2, "Using the Preconfigured OC4J". For information about adding the Rich Text portlet to a page, see Section 4.3, "Consuming Portlets".

Figure 14–4 Example Display Text in the Rich Text Portlet



To add content through the Rich Text portlet, perform the following steps:

1. Click the Actions icon in the right corner of the portlet header.
2. From the resulting menu, select **Customize**.

   The Rich Text editor loads into the portlet.
3. Toggle the editor to HTML edit mode by clicking the Change Edit Mode icon (see Table 14–1).
4. Enter a reference to your preferred CSS.

   For example:

   ```
   <link rel="stylesheet" type="text/css"
   href="http://mycompany.com:7632/stylesheets/newsbriefs.css">
   ```

   Note, however, that a stylesheet is not required.
5. Toggle the editor to WYSIWYG edit mode, and enter the following text:

   ```
   NewsBrief
   This is example content for the Rich Text portlet used for brief company-wide
   ```

```
announcements.

Here is another paragraph.
```

6. Select the text `NewsBrief`, and click the Text Foreground Color icon (Figure 14–5).

Figure 14–5 Text Foreground Color Icon

7. In the color palette, select a color, and then click **OK**.
8. With the heading still highlighted, select `4` from the **Font Size** list.
9. Toggle the editor to HTML edit mode, wrap the heading and the color and size specifications in the tag for the top heading level in your CSS.

   For example:

```
<h1>
   <font style="color: rgb(82, 140, 255);" size="4">
   <span style="font-family: arial,helvetica,sans-serif;">
   NewsBrief</span></font>
</h1>
```

10. Add a horizontal rule after `</h1>`.

    For example:

```
</h1><hr>
```

11. Toggle the editor to WYSIWYG edit mode and highlight the text `Here is another paragraph`.
12. Click the Insert Hyperlink icon (Figure 14–6), and enter a hyperlink to a favorite site.

Figure 14–6 Insert Hyperlink Icon

For example:

```
http://www.oracle.com
```

13. Click **OK** to close the Insert Hyperlink dialog box.
14. Press **Enter** to create a new line, and click the Insert Image icon (Figure 14–7).

Figure 14–7 Insert Image Icon

15. In the Insert Image dialog box, enter the location of an image.

    To obtain any image's location, right-click the image and select **Properties** from the context menu. The location is available for copying from the image's property sheet.

16. Click **OK** to close the Insert Image dialog box.

17. Click the **Submit** button below the Rich Text editor to save your changes and exit the editor.

Table 14–1 lists and describes the controls available on the Rich Text portlet edit mode toolbar.

Table 14–1 Rich Text Portlet Controls

Button/Keystrokes	Mode: WYSIWYG/ HTML/Both	Function
	WYSIWYG	Undo last action. Click again to undo preceding actions.
Ctrl + **Z**		
	WYSIWYG	Redo last action. Click again to perform additional redo's.
Ctrl + **Y**		
	WYSIWYG	Select the font in which to set the selected or subsequent text. Choose from the fonts available on your file system.
	WYSIWYG	Select a font size in which to set the selected or subsequent text. Values are set against the browser or the current size default. ■ 1 to 7 sets the font size as a value relative to the browser default font size. A value of 3 is equivalent to the browser default. Values lesser than 3 set the font smaller. Values greater than 3 set the font larger. ■ +1 to +7 increases the font size relative to the default font size (`font size=3`). ■ -1 to -7 decreases the font size relative to the default font size (`font size=3`).
	Both	**Bold** selected or subsequent text.
	Both	*Italicize* selected or subsequent text.
	Both	Underline selected or subsequent text.
	WYSIWYG	Set the text color.
	WYSIWYG	Set the text background color. This is also called *highlighting*. The selected color appears behind the text.

Table 14–1 (Cont.) Rich Text Portlet Controls

Button/Keystrokes	Mode: WYSIWYG/ HTML/Both	Function
< >	Both	Switch between a graphical edit mode (WYSIWYG) and an HTML edit mode. WYSIWYG is the default. Use the WYSIWYG edit mode to display text as it will appear on your page. Use HTML edit mode to modify the HTML source code.
	WYSIWYG	Add/change a link. You can create any kind of standard Web link. For example: ■ Link to another Web page. Enter the full URL, for example: `http://www.oracle.com` ■ Link to an e-mail address by entering "mailto:", for example: `mailto:info@oracle.com`. When users click this type of link, a new, blank e-mail message window opens, addressed to the addressee you enter here, in the current browser's default e-mail application. For a `mailto` link to work, the user's browser preferences must be properly configured. In Netscape browsers, you can find e-mail settings under Preferences in the Mail & Newsgroups section. In Internet Explorer browsers, you can find e-mail settings under Internet Options on the Programs tab.
	WYSIWYG	Insert an image. Ultimately, insert the URL to an image. You can obtain an image URL by right-clicking the image and selecting **Properties** from the context menu.
	WYSIWYG	Align the current paragraph left, center, or right.
	WYSIWYG	Add/remove a bulleted or numbered list. Once you start a list, every time you press **Enter** a new bulleted or numbered item is added. Press **Enter** twice to end the list.

Table 14–1 (Cont.) Rich Text Portlet Controls

Button/Keystrokes	Mode: WYSIWYG/ HTML/Both	Function
 	WYSIWYG	Decrease or increase the current paragraph's indentation. **Note**: You must use these keys to increase or decrease indents in WYSIWYG mode. Using the **Tab** key does not indent a paragraph.
Ctrl + **P**	Both	Print the content in the editor window. **Note**: Your cursor must be in the editor window for this key combination to limit printing to the editor's content.

14.3.2 Prebuilt Portlets

What Are They?

Prebuilt portlets include partner portlets and integration solutions.

Partner portlets are available through Oracle's partnerships with leading system integrators, software vendors, and content providers. You can access these portlets by using the keywords `portal` or `portlet` when searching the Oracle PartnerNetwork Solutions Catalog, available at `http://solutions.oracle.com`. Examples of these include portlets for the following purposes:

- Generating point-to-point driving directions
- Accessing Information Technology (IT) information from a wide variety of sources
- Viewing summary information about news, stocks, and weather

Portal Integration (POINT) Solutions provide solutions for customers who require basic functionality for popular applications, such as Microsoft Exchange, Lotus Notes, SAP, IMAP, SMTP, and the like. These portlets are available on the Oracle Technology Network (OTN) Portal Integration Solutions page:

`http://www.oracle.com/technology/products/ias/portal/point.html`

Who Is the Intended User?

Fully developed, downloadable portlets are best suited for use by application developers who understand how to download, install, and register producers in Oracle WebCenter Framework. They are available for use by all levels of experience.

When Should They Be Used?

Use prebuilt portlets when your needs are satisfied by the functions the portlets offer and the level of personalization readily available is sufficient to complete the desired task.

Consider alternatives when you need to extend or personalize the portlet, for example, when you need a different user interface or when the functionality you require is not available out of the box.

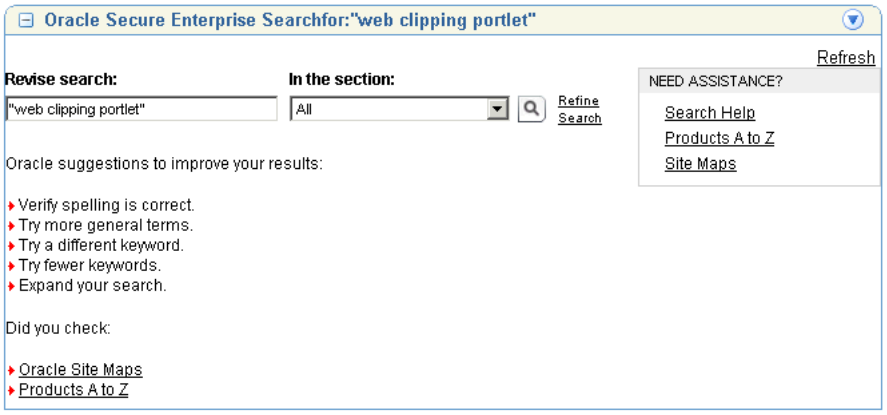
14.3.3 Web Clipping

What Is It?

Web Clipping is a browser-based declarative tool that enables the integration of any Web application with a WebCenter application. Web Clipping is designed to provide quick integration by leveraging the Web application's existing user interface. Web Clipping has been implemented as a PDK-Java producer.

To create a Web Clipping portlet, the WebCenter application developer uses a Web browser to navigate to a Web page that contains desired content. Through the Web Clipping Studio, the application developer can drill down through a visual rendering of the target page to choose the desired the content (Figure 14–8).

Figure 14–8 Selected Web Clipping Displayed in a Web Clipping Portlet



Web Clipping supports the following:

- **Navigation through various styles of login mechanisms,** including form- and JavaScript-based submission and HTTP Basic and Digest Authentication with cookie-based session management.
- **Fuzzy matching of clippings,** If a Web clipping gets reordered within the source page or if its character font, size, or style changes, then it will still be identified correctly by the Web Clipping engine and delivered as the portlet content.
- **Reuse of a wide range of Web content,** including basic support of pages written with HTML 4.0.1, JavaScript, applets, and plug-in enabled content, retrieved through HTTP GET and POST (form submission).
- **Personalization,** enabling an application developer to expose input parameters that users can modify when they personalize the portlet. These parameters can be exposed as public parameters that an application developer can map as page parameters. This feature enables users to obtain personalized clippings.
- **Integrated authenticated Web content through Single Sign-On,** including integration with external applications, which enables you to leverage Oracle Single Sign-On and to clip content from authenticated external Web sites.
- **Inline rendering,** enabling you to set up Web Clipping portlets to display links within the context of the portlet. As a result, when a user clicks a link in the Web Clipping portlet, the results display within the same portlet. You can use this feature with internal and external Web sites.
- **Proxy Authentication,** including support for global proxy authentication as well as authentication for each user. You can specify the realm of the proxy server and

whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.

- **Resource Tunneling** of images.
- **Open Transport API** for customizing authentication mechanisms to clipped sites.
- **Security Enhancement** that enables administrators to control access to content that can be clipped by the Web Clipping portlet.

Who Is the Intended User?

Web Clipping is best suited for use by application developers and component developers who want to leverage an existing Web page for rapid portlet development. This portlet can be added to a page by any user with the appropriate privileges.

When Should It Be Used?

Use Web Clipping when you want to repurpose live content and functionality from an existing Web page and expose it in your WebCenter application as a portlet. Consider alternatives if you want to change the way information is presented in the clipped portlet. That is, you do not need to control the User Interface (UI) or application flow, and you are accessing Web-based applications. For a greater level of control, use OmniPortlet's Web page data source instead of Web Clipping. (See Section 14.3.4, "OmniPortlet".)

The following are some examples of when you can consider using the Web Clipping portlet:

- **Stock chart portlet.** You want to create a portlet that displays the stock market's daily performance chart from your financial advisor's Web site. You could clip this information from an external Web site, even if your company is using a proxy.
- **Web mail portlet.** Your users want to access their confidential Web mail accounts through a portlet and to display their in-boxes in the portlet.

For more information about using Web Clipping, see Chapter 17, "Creating Content-Based Portlets with Web Clipping".

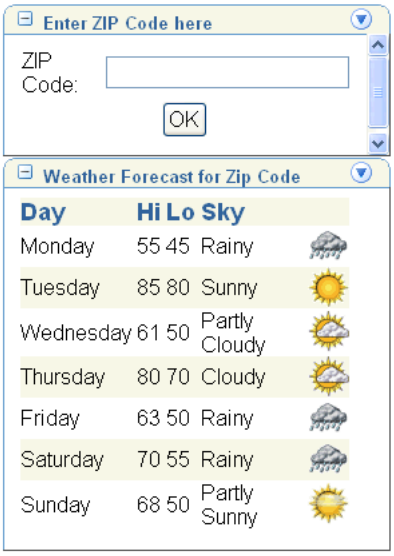
Note: To use the Web Clipping portlet, OmniPortlet, or the Simple Parameter Form with Windows 2000, you must use Netscape 7.0 or later, or Microsoft Internet Explorer 5.5 or later.

14.3.4 OmniPortlet

What Is It?

OmniPortlet is a declarative portlet-building tool that enables you to build portlets against a variety of data sources, including XML files, character-separated value files (CSV, for example, spreadsheets), Web Services, databases, Web pages, and SAP data sources. OmniPortlet users can also choose a prebuilt layout for the data. Prebuilt layouts include tabular, news, bullet, form, chart, or HTML. HTML layout enables OmniPortlet users to write their own HTML and inject the data into the HTML. Figure 14–9 shows an OmniPortlet with a tabular format.

Figure 14–9 A Simple Parameter Form (top) and an OmniPortlet Using Tabular Format



Like Web Clipping, OmniPortlet supports proxy authentication, including support for global proxy authentication as well as authentication for each user. You can specify whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password.

You'll find information about OmniPortlet on Portal Center. Navigate to the following URL, then click the **Portlet Development** link:

`http://www.oracle.com/technology/products/ias/portal/index.html`

Who Is the Intended User?

Business users with a minimum knowledge of the URLs to their targeted data will find OmniPortlet a valuable tool.

When Should It Be Used?

Use OmniPortlet when you want to build portlets rapidly against a variety of data sources with a variety of layouts. Consider alternatives when you want complete control of the design and functionality of the portlet.

The following are some examples of when you can consider using OmniPortlet:

- **RSS news feed portlet:** You want to create a portlet that displays live, scrolling news information to your users. The data comes from a Really Simple Syndication (RSS) news feed, such as Oracle Technology Network Headlines. You also want the portlet to contain hyperlinks to the news source.
- **Sales chart portlet:** You want to present up-to-date information about your company's sales results. You also want to display data in the form of a pie chart, and your company stores its sales information in a remote relational database.
- **SAP portlet:** You want to display information from a company's SAP system. To minimize the load on the company's SAP Business Suite, the information retrieved from the system must be cached for each user for the entire day.

Note: The SAP data source is not included with Oracle WebCenter Framework. To learn more about using the SAP data source, visit the Oracle Portal Integration Solutions page on OTN:

`http://www.oracle.com/technology/products/ias/portal/point.html`

For more information about OmniPortlet, see Chapter 16, "Creating Portlets with OmniPortlet".

14.3.5 Programmatic Portlets

What Are They?

Programmatic portlets are portlets that you write yourself, in Java, using either the standard Java Portlet Specification (JPS) or PDK-Java. Oracle WebCenter Framework provides two declarative wizards for simplifying the creation of standards-based JSR 168 portlets and Oracle PDK-Java portlets. These wizards assist in the construction of the framework within which you create the portlet. Each wizard may include easy steps for the following:

- Configuring general portlet properties
- Specifying names and search terms
- Setting permissible content types and mapping display modes
- Specifying user-customizable preferences
- Adding security roles
- Enabling default caching
- Adding initialization parameters
- Adding navigation parameters

Who Is the Intended User?

Use of the wizards is easy, but the creation of portlet logic is best performed by experienced and knowledgeable Java developers who are comfortable with the Java Portlet Specification or PDK-Java and who understand the configuration of producers.

When Should They Be Used?

Use programmatic portlets when you have very specialized business rules or logic or when you require personalized authentication, granular processing of dynamic results, and complete user interface control. Additionally, use programmatic portlets when you need to satisfy any of the following conditions:

Consider using the programmatic approach when the out-of-the-box portlets do not address your needs.

The following list provides a couple of examples of when you can consider using programmatic portlets:

- **Photo Album portlet:** You want to create a a portlet that facilitates uploading, storing, and viewing user photos.

- **Shopping Cart portlet:** You want to create a portlet that facilitates the viewing and purchasing of, for example, company-branded items, such as mouse pads, pens, flash drives, tee shirts, and the like.

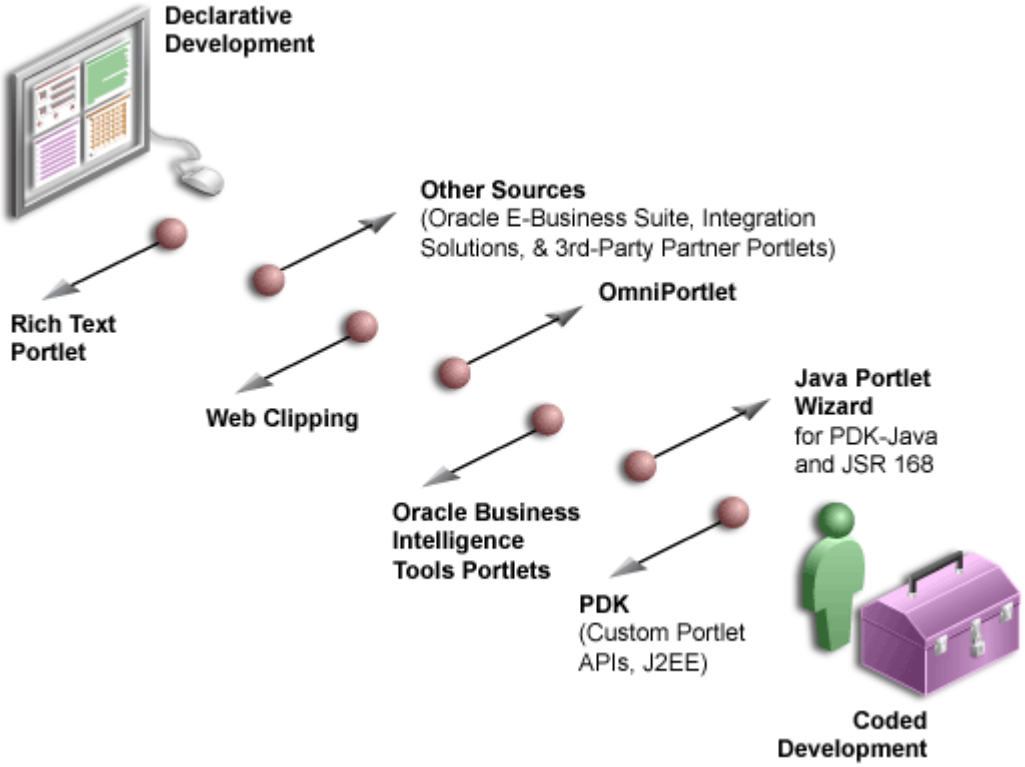
For more information about using programmatic portlets, see Chapter 18, "Creating Java Portlets" and Chapter 19, "Enhancing Java Portlets".

14.3.6 Deciding Which Tool to Use

Figure 14–10 illustrates the spectrum of portlet resources described in the previous section. Notice how one end of the spectrum is geared toward a more declarative development environment (that is, develop-through-wizard) while the other end focuses more on hand-coding. You can choose your tool depending on which type of environment is most comfortable and suitable to your skill-base.

For more information about deciding which tool to use, see Chapter 15, "Portlet Technologies Matrix".

Figure 14–10 Portlet Resources from Declarative to Coded Development



15

Portlet Technologies Matrix

This chapter describes portlet features, characteristics, technologies, and tools to help you decide which portlet building technology best suits your needs. It includes the following sections:

15.1 The Portlet Technologies Matrix

Table 15–1, " Portlet Building Technologies Comparison Matrix" summarizes the technologies and tools you can use with Oracle WebCenter Framework. The matrix describes the tools and technologies that are covered in more detail in this guide: OmniPortlet, Web Clipping portlet, and programmatic portlets, including standards-based (JSR 168) portlets and PDK-Java portlets. Additionally, it includes Oracle Portlet Factory.

Note: While these are the primary tools for building portlets, additional tools and technologies exist, such as other Oracle products, including Oracle Reports, OracleBI Discoverer, and the JSF Portlet Bridge. These other tools are not covered in this guide.

The other sections in this chapter provide further detail on the characteristics listed in Table 15–1. Use the table to quickly scan all the features and characteristics, then see the subsequent sections for more in-depth information.

Table 15–1 Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	Oracle Portlet Factory	Programmatic Portlets Standards-Based/PDK-Java
General Suitability			
A simple wizard-based tool, accessible from a browser, that assists in retrieving and presenting Web content that originates from other Web sites in a WebCenter application.	Wizard-based tool, accessible from a browser, that assists in retrieving and presenting data from a wide variety of data sources.	A portlet creation environment for the development, deployment, and maintenance of custom and composite portlets that interact with enterprise applications, such as SAP.	PDK-Java offers Oracle-specific application programming interfaces (APIs) for building portlets for use in WebCenter applications and OracleAS Portal. Standards-based portlets additionally work with portals of other vendors. Oracle Oracle WebCenter Framework supports both WSRP and JSR-168 standards.
Expertise Required			
No expertise required.	Basic understanding of one or more supported data sources and the concepts of portlet and page parameters.	Basic understanding of J2EE Portlet life cycle and basic coding knowledge without the need to know the Java language. The target audience for this tool is the business developer.	Java, Servlet, JSP knowledge.
Supported Data Sources (for details, see Section 15.3, "Expertise Required")			
Any Web site accessible on the network over HTTP or HTTPS.	CSV, XML, Web Service, SAP, SQL, Web site, JCA.	Web Services, SQL, PeopleSoft, JDE, and SAP.	No limitations.
Deployment Type			
PDK-Java producers	PDK-Java producers	WSRP and PDK-Java producers	PDK-Java uses PDK-Java producers. Standards-based portlets use WSRP producers.
Caching Style			
Expiry-based caching, validation-based caching (auto invalidate when personalized).	Expiry-based caching, validation-based caching (auto invalidate when personalized).	Cache Control builder enables caching of output related to an action within a model for a specified amount of time.	Expiry-based, validation, and invalidation caching, Edge Side Includes. **Note:** JSR 168 does not support validation based caching. WSRP 1.0 does. If you use a pure WSRP portlet, then validation-based caching is also supported. If you host a JSR portlet on WSRP (as is done in Oracle Oracle WebCenter Framework) then validation-based caching is not supported.

Table 15–1 (Cont.) Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	Oracle Portlet Factory	Programmatic Portlets Standards-Based/PDK-Java
Development Tool			
Browser - wizard.	Browser - wizard.	Portlet Factory tool (*Builders* and *Models*).	Oracle JDeveloper Java Portlet wizard (or any other Java development environment).
Portlet Creation Style			
Design-time at run time.	Design-time at run time.	Develop first, add later.	Develop first, add later.
User Interface Flexibility			
N/A	Very flexible, by using the HTML layout.	Flexible. Portlet code is generated based on the choices made in a Builder wizard. Developers can view generated code, but are never required to interact with it.	Very flexible.
Ability to Capture Content from Web Sites			
Yes, by its nature.	Yes, by using the Web Page Data Source.	No. This is a tool for aggregating data, content, and processes from existing enterprise applications, such as SAP, and deploying them as custom portlets.	Yes. For PDK-Java, use the `oracle.portal.provider.v2.*` package. For standards-based portlets, use the `java.net` package.
Ability to Render Content Inline			
Yes	No. However, inline rendering can be achieved through public portlet parameters.	Yes, for multipage portlets.	Yes. For PDK-Java, use private portlet parameters. Standards-based portlets include servlets and JSPs, using the method `PortletContext.getRequestDispatcher()`.
Charting Capability			
No	Yes, 2D-3D charts.	Out-of-the-box demonstration version of Greenpoint WebCharts builder for creation of charts and graphs.	Yes, using BI Beans.
Public Portlet Parameter Support			
Yes	Yes	Access public parameters through the `RequestInputs` API.	Yes **Note:** In standards-based portlets, support is available for public parameters using WSRP 2.0's navigational parameter feature along with Oracle WebCenter Framework extensions to JSR 168.
Private Portlet Parameter Support			
No	No	No, however, elements called *variables* function much the same as parameters within a family of Portlet Factory portlets.	Yes

Table 15–1 (Cont.) Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	Oracle Portlet Factory	Programmatic Portlets Standards-Based/PDK-Java
Ability to Hide and Show Portlets Based on User Privileges			
No, though it is possible to apply security managers that are not exposed through the user interface (UI).	No, though it is possible to apply security managers that are not exposed through the UI.	No	Yes. For PDK-Java, by using the Security managers. For standards-based portlets, the Servlet security model is supported by using methods such as `PortletRequest.isUserInRole()` and `PortletRequest.getUserPrincipal()`.
Multilingual Support			
N/A	Yes	Yes	Yes
Pagination Support			
N/A	No	Yes	Yes, programmatically
Authenticating to External Applications			
External application integration supported.	Basic authentication support if the data source requires it.	External application integration supported.	For PDK-Java portlets, external application integration is supported. LDAP integration is supported when the portlet is running behind the same firewall as the LDAP server. For standards-based portlets, though not specifically supported, external application support is feasible through custom user attributes.) LDAP integration is supported.

15.2 General Suitability

This section describes each portlet-building technology in terms of its usage characteristics (for example, wizard-based or programmatic).

15.2.1 Web Clipping

Web Clipping is a simple wizard-based tool that assists with retrieving and presenting Web content that originates from other Web sites in a WebCenter application. Web Clipping does not require a technical background.

Examples of portlets you can build using Web Clipping

The examples of portlets that you can build by using Web Clipping are as follows:

- Stock chart portlet
- Web mail portlet
- News portlet containing dynamic content from an existing Web site

15.2.2 OmniPortlet

OmniPortlet is an easy-to-use, wizard-based tool for presenting information from a wide variety of data sources in a variety of formats. OmniPortlet runs completely in the browser. Drop OmniPortlet on a page, click the Define link, and choose a data source and a presentation format. Select from a wide variety of data sources as follows:

- Spreadsheet
- SQL
- XML
- Web Service
- Web page

OmniPortlet does not require the use of an additional development tool or a strong technical background. Even so, it can be used for building reusable, high-performing portlets.

Examples of portlets you can build using OmniPortlet

The examples of portlets that you can build by using OmniPortlet are as follows:

- RSS news feed portlet
- Sales chart portlet
- SAP BAPI-based portlet

15.2.3 Oracle Portlet Factory

Oracle Portlet Factory assists the business developer in creating portlets that aggregate data, content, and processes from existing enterprise applications, such as SAP applications.

Developers rapidly build portlets by pulling together a sequence of highly-adaptive, reusable software components, called *Builders*. These builders perform specific application functions, such as querying a database, executing a business process within an application, or rendering an output UI. Developers assemble Builders into Models, similar to the way a spreadsheet model is created by snapping formulas together. Models are executed at run time to dynamically generate application code, including JSPs, Java classes, and XML documents, as well as all of the low-level artifacts that constitute the portlet application. Business developers can view the generated application code, but they are never required to work with it.

Both Oracle PDK-Java and JSR 168 portlet types are supported.

Examples of portlets you can build using Oracle Portlet Factory

The examples of portlets that you can build by using Oracle Portlet Factory are as follows:

- SAP form portlet
- SAP multipage portlet
- SAP transaction portlet

15.2.4 Programmatic Portlets

If the wizard-based portlet building tools do not satisfy your needs, then you can build your portlets programmatically using Java. The Java Community Process standardized

the Java portlet APIs in 2003. Portlets built against the Java Specification Request (JSR) 168 standard are interoperable across different portal platforms. The Java Portlet Wizard, a tool available through the WebCenter Suite, assists with building Java portlets.

> **Note:** When building portlets in Java, you have full control over your portlet's functionality. For example, you can control what it looks like and how it behaves.

Examples of portlets you can build using Java

The examples of portlets that you can build by using Java are as follows:

- Discussion forum portlet
- E-mail portlet

15.3 Expertise Required

While some of the portlet building tools do not require portlet development skills, others assume a strong technical background. This section describes each tool in terms of the level of knowledge required to use it effectively.

15.3.1 Web Clipping

Web Clipping does not require a technical background. However, if you want to parameterize the Web page content that you clipped, then you must have an understanding of public portlet parameters and page parameters.

15.3.2 OmniPortlet

OmniPortlet requires a basic knowledge of the data source you want to leverage in your portlet. Table 15–2 lists the types of data sources that can be used with OmniPortlet and describes the type of information required to work with each type.

Table 15–2 OmniPortlet Data Sources

Data Source	Required Information
Spreadsheet	The URL that points to the spreadsheet containing the data to display in the portlet.
SQL	The connection information to the data source and the SQL query that retrieves the data from the database.
XML	The location of the XML source and optionally the address of the XSL filter and the XML schema.
Web service	The Web Services Description Language (WSDL) URL, the method of the Web service, and optionally the XSL filter URL and the XML schema URL.
Web page	The Web page data source uses the same environment as Web Clipping. No technical background is required.
J2EE Connector Architecture	Although not displayed on the OmniPortlet Wizard's Type page, a J2EE Connector Architecture (JCA) 1.0 adapter is also available. JCA provides a mechanism to store and retrieve enterprise data such as that held in ERP systems (Oracle Financials, SAP, PeopleSoft, and so on).

15.3.3 Oracle Portlet Factory

Oracle Portlet Factory is aimed at the business developer looking for a tool that facilitates rapid development and does not require knowledge of Java or J2EE. Development consists of Builders (such as buttons, loops, actions, and so on) constructed inside of Models. Code is generated automatically. Developers can view generated code, but are not required to work with it in any way.

15.3.4 Programmatic Portlets

To build Java portlets, you must know at least a subset of J2EE. Knowing HTML, Java servlets, and XML is a must, and JSP experience is recommended. Additional Java knowledge is optional, depending on the task you want to perform. Using Java portlets you can access any data source supported by the Java language.

Note: Additional tools, such as Oracle Portlet Factory, are available for making Java portlet development easier.

15.4 Deployment Type

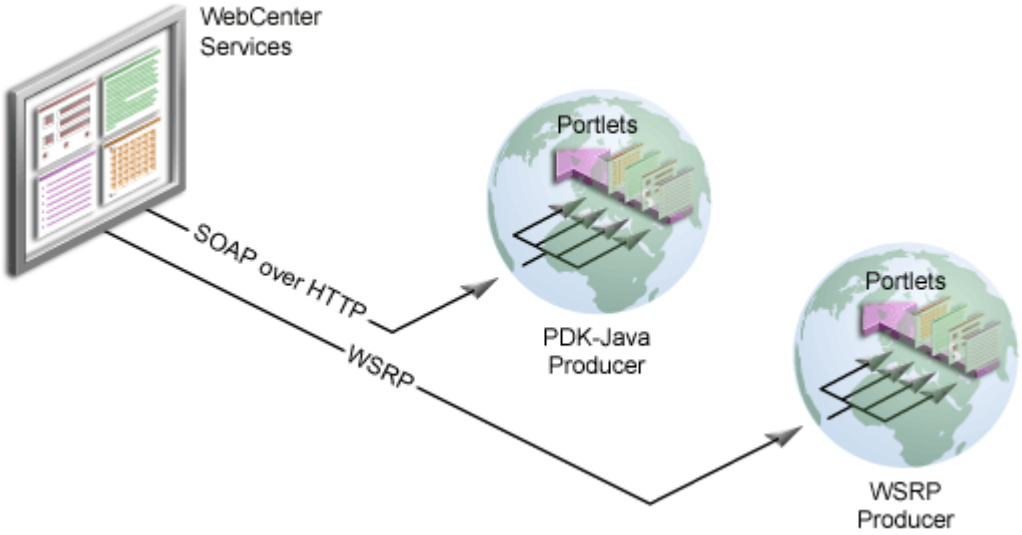
Before a portlet can be consumed by an application, you must first deploy it, then register the producer you've deployed the portlet to. As shown in Figure 15–1, portlets can be deployed through the following two producer types:

- PDK-Java producers
- WSRP producers

Within WSRP, both version 1.0 and 2.0 are supported.

PDK-Java portlets are deployed to a J2EE application server, which is often remote and communicates with the consumer through Simple Object Access Protocol (SOAP) over HTTP. JSR 168 portlets are deployed to a WSRP producer, which is also remote and communicates with the consumer through WSRP (Web Services for Remote Portlets).

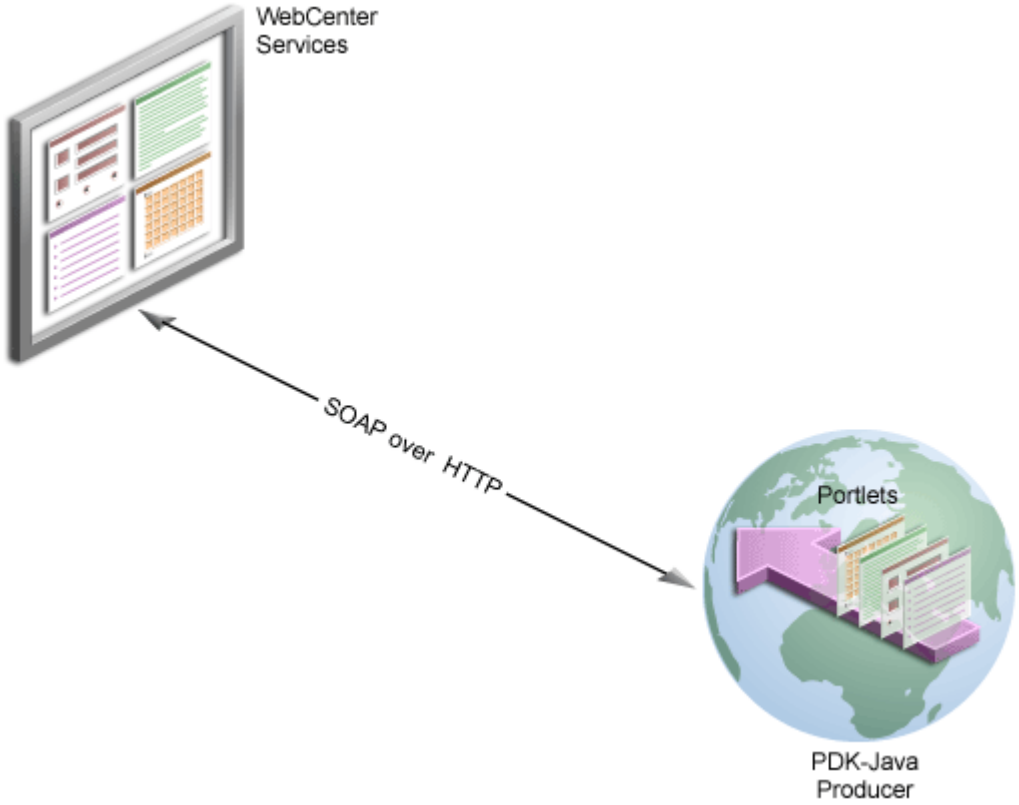
Figure 15–1 Portlet Producer Overview



15.4.1 PDK-Java Producers

PDK-Java producers use open standards, such as XML, SOAP, HTTP, or J2EE for deployment, definition, and communication with applications. Figure 15–2 shows how OracleAS Portal incorporates portlets from a PDK-Java producer and the PDK-Java producer communicates with WebCenter application using SOAP over HTTP.

Figure 15–2 PDK-Java Producers



There are several benefits to developing portlets and exposing them through PDK-Java producers as follows:

- Deploy portlets remotely.
- Leverage existing Web application code to create portlets.
- Specify producers declaratively.
- Use standard Java technologies (for example, servlets and JSPs) to develop portlets.

To expose your portlets using a PDK-Java producer, you must first create a producer that manages your portlets and communicates with Oracle WebCenter Framework using SOAP. To learn how to expose your portlets using a PDK-Java producer, see Section 18.5, "Building JPS-Compliant Portlets with Oracle JDeveloper".

15.4.2 WSRP Producers

Oracle WebCenter Suite supports Web Services for Remote Portlets (WSRP) versions 1.0 and 2.0. WSRP 2.0 support is for a preliminary (that is, preproduction) version of WSRP 2.0. This emerging standard provides support for inter-portlet communication and export or import of portlet customizations.

Use of WSRP 2.0 requires use of Oracle-specific extensions. For example, portlets produced by WSRP 2.0 producers must be deployed to Oracle's container (OC4J) to take advantage of the benefits this newer version provides. This is because standard portlet APIs (such as JSR 286) have not yet evolved to the level of the WSRP 2.0 communication protocol.

The Producer Registration wizard is the entry point for registering both WSRP 1.0 and 2.0 producers. The wizard automatically recognizes whether WSRP 1.0 or 2.0 is in play.

Architecturally, WSRP producers are very similar to PDK-Java producers. WSRP is a communication protocol between WebCenter application servers and portlet containers. WSRP is a standard that enables the plug-and-play of visual, user-facing Web services with intermediary Web applications.

Being a standard, WSRP enables interoperability between a standards-enabled container based on a particular language (such as JSR 168, .NET, Perl) and any WSRP portal. So, a portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any application that supports this standard.

> **Note:** For more information about the WSRP architecture, see "The Relationship Between WSRP and JPS" in Chapter 18, "Creating Java Portlets".

To make standard portlets (such as JSR 168, .NET, Perl) available to a WebCenter application, you must package them in a portlet application and deploy them to a WSRP container. To learn more about WSRP, see the WSRP and JSR 168 Standards page on the Oracle Technology Network:

`http://www.oracle.com/technology/products/ias/portal/standards.html`

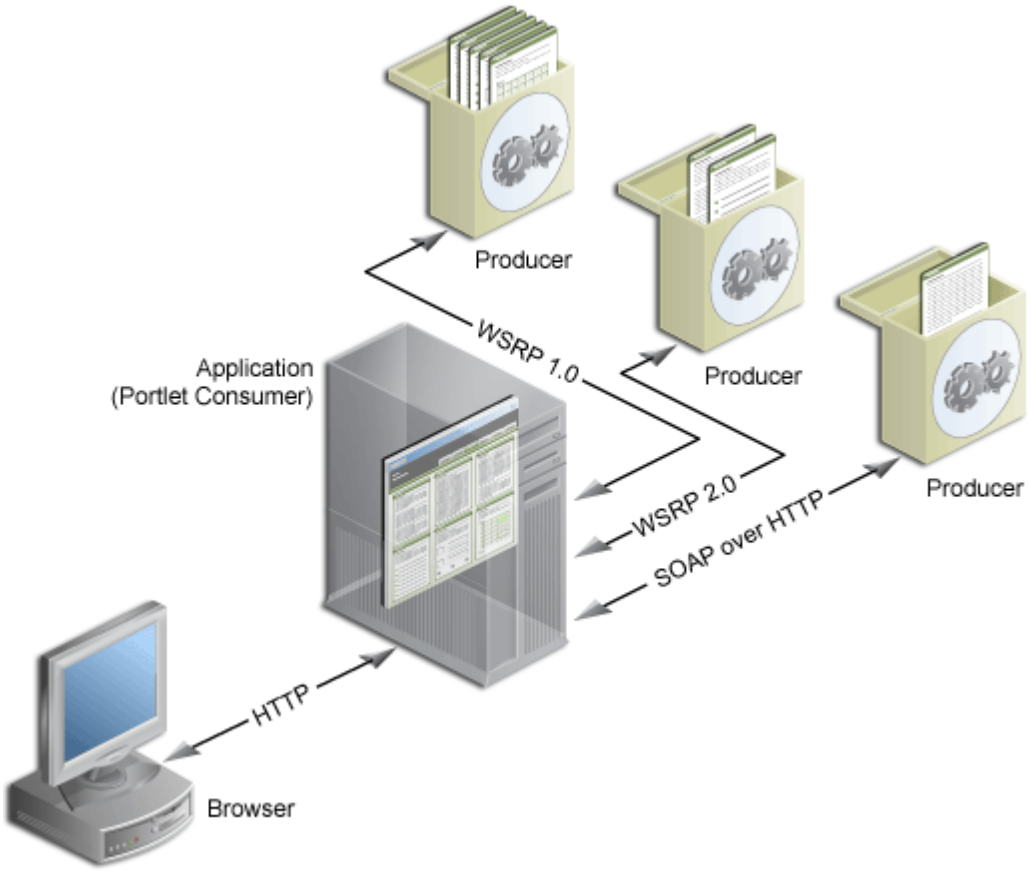
To learn how to package your portlets in a WSRP container, see Section 18.9, "Deploying Your Portlet to an Application Server". You can also test your WSRP producers online using the OracleAS Portal Verification Service:

`http://portalstandards.oracle.com/portal/page/portal/OracleHostedWSRPPortal/Welcome`

15.4.3 Producer Architecture

Figure 15–3 illustrates the basic architecture of portlet producers.

Figure 15–3 Producer Architecture



When users display a page in their Web browsers, the flow of the request works as follows:

1. The user requests a page from the Web browser by entering a URL in the browser's address field.
2. The browser transmits the request to the application over HTTP.
3. The application contacts the portlet producers that provide the portlets that display on the requested page.
4. The producers make the necessary calls to their portlets so that the portlets generate the portlet content in the form of HTML or XML code.
5. The producers return the portlet content back to the application using their relevant protocols:
   - JSR 168 portlets are initialized by WSRP producers, which communicate using the WSRP 1.0 or 2.0 protocol.
   - PDK-Java portlets are initialized by PDK-Java producers, which communicate using SOAP over HTTP.

More On OTN

Note: For more information about the portlet and producer architecture, visit the Portlet Development page on Portal Center (`http://www.oracle.com/technology/products/ias/portal/index.html`).

Web Clipping, OmniPortlet, Oracle Portlet Factory portlets, and Java portlets communicate with Oracle WebCenter Framework through either WSRP or PDK-Java producers. You must register these producers with Oracle WebCenter Framework before you can use the portlets they produce in your WebCenter application.

The latest versions of Web Clipping and OmniPortlet are available through the preconfigured Oracle Containers for J2EE (OC4J). For more information, see Section 3.2, "Using the Preconfigured OC4J".

15.5 Caching Style

Portlet caching is key to rapid response to user requests. Portlets implement validation-based and expires-based caching using Java Object Cache. Invalidation-based caching continues to be implemented by a Web Cache that fronts the PDK-Java producer running the Web Clipping portlet and OmniPortlet.

Caching rules can be specified at a portlet's container level, encoded in the portlet's own logic, or, for JSR 168 portlets, established through the portlet wizard. Provided it is specified, container-level caching takes over when caching is not part of the portlet code.

At the application level, Oracle WebCenter Framework supports use of a Java cache for the establishment of application-level caching rules.

When not using caching, you may find accessing various data sources with Web Clipping and OmniPortlet to be time consuming. When you enable caching at the application level, you instruct the Java cache to maintain a copy of the portlet content. When data that was previously cached is requested, no time is lost in contacting the data source and regenerating its content. Instead, the previously cached portlet content is returned.

A portlet's content weighs heavily in determining the type of caching the portlet should use. For example:

- **Expiry-based caching**: Consider using expiry-based caching when the portlet content is static or when it is not critical that the most up-to-date content be displayed. When using expiry-based caching, you must specify the caching period.
- **Validation-based caching:** Consider using validation-based caching for portlets with dynamic content that changes frequently or unpredictably. The portlet associates its content with a caching key and returns the key value along with the content. When the portlet content is requested, the portlet decides, based on the caching key, if the current content is valid. If the portlet content is valid, then it returns a response indicating that the cached content can be used (that is, the content is valid) or generates the new portlet content and returns it along with a new caching key for that content.
- **Invalidation-based caching:** Invalidation-based caching is the most complex, but also the most flexible, form of caching. Consider using invalidation-based caching when you require the efficiency of expiry-based caching with the ability to invalidate cache content.

15.5.1 Web Clipping and OmniPortlet

In addition to invalidation-based caching, expiry-based caching can be specified for the Web Clipping portlet and OmniPortlet. Additionally, these portlets are refreshed automatically when they are personalized.

15.5.2 Oracle Portlet Factory

Oracle Portlet Factory has caching capabilities through the Cache Control builder. This builder enables the caching of the output of a specific action within a model for a specified amount of time.

15.5.3 Programmatic Portlets

Java portlets support expiry- and validation-based caching. These portlets can be cached in full, or Edge Side Includes (ESI) can be used to cache fragments of portlets.

15.6 Development Tool

This section describes the development tools you can use to build different types of portlets.

15.6.1 Web Clipping and OmniPortlet

OmniPortlet and Web Clipping use a browser-based wizard as the development tool.

15.6.2 Oracle Portlet Factory

Oracle Portlet Factory runs inside the Eclipse Integrated Development Environment (IDE).

15.6.3 Programmatic Portlets

Although you can use any Java development environment to build Java portlets, it is highly recommended that you use Oracle JDeveloper, a professional IDE. While you can consider other IDEs, Oracle JDeveloper includes the Java Portlet Wizard, to minimize your Java portlet development efforts.

The Java Portlet Wizard generates a starting skeleton and file structure for both JSR 168 and PDK-Java portlets. You need to add only your own business logic to the skeleton. Oracle JDeveloper can also package and deploy your applications to your J2EE container, such as OC4J. Also, Oracle JDeveloper helps you test your portlet producer. Oracle recommends that you use the preconfigured OC4J, provided through Oracle WebCenter Framework, as your development Java portlet run time environment. For more information, see Chapter 3, "Preparing Your Development Environment".

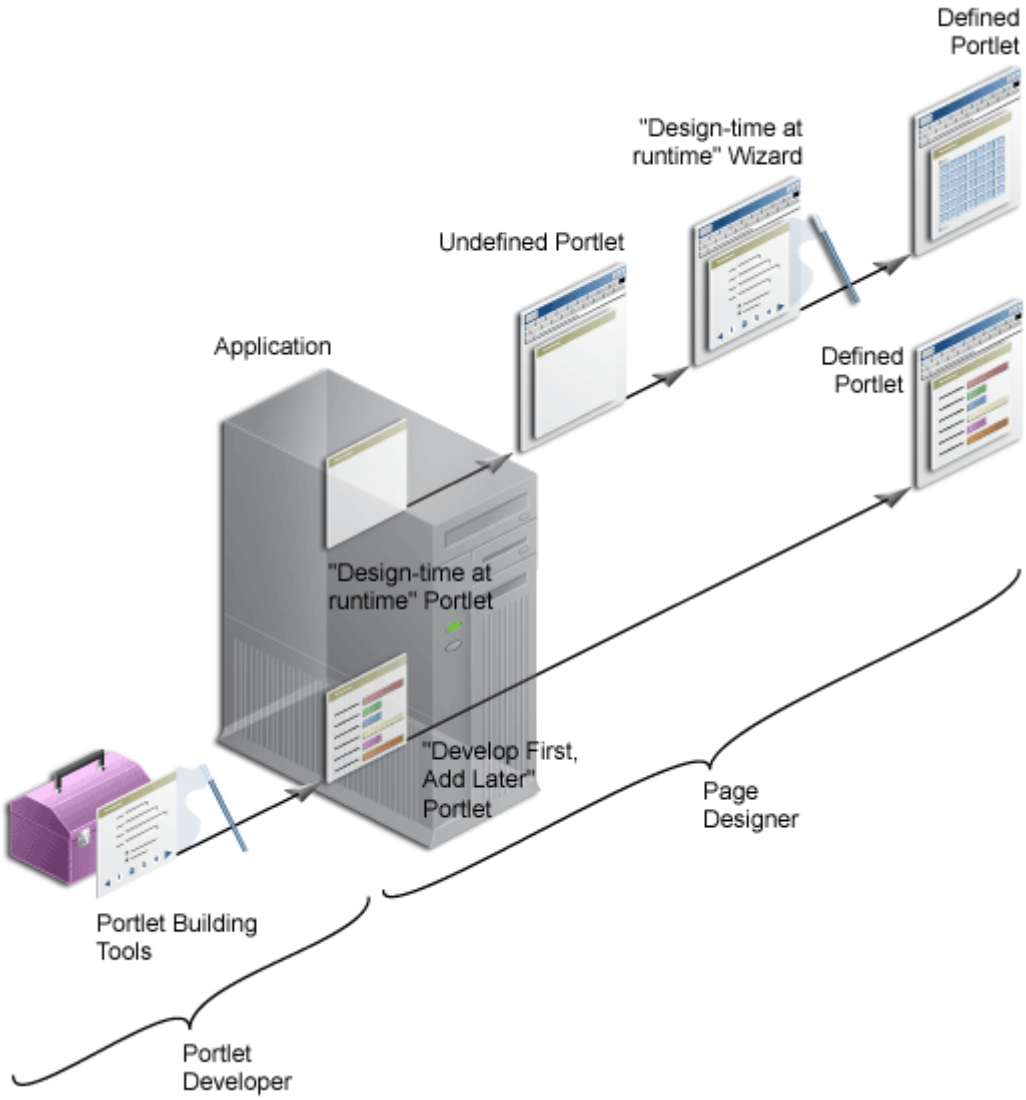
15.7 Portlet Creation Style

Oracle WebCenter Framework supports the following types of portlet creation (Figure 15–4):

- Develop first, add later
- Design-time at run time

Develop first, add later portlet creation is usually the task of the *portlet* developer; design-time at run time portlet creation is the *application* developer's responsibility.

Figure 15–4 Portlet Creation Style



15.7.1 OmniPortlet and Web Clipping

OmniPortlet and Web Clipping both offer a "design-time at run time" portlet creation style. Register the portlet producers with the application that will consume the portlets, add the portlets to an application page, run the application, and then define the portlets in-place on the page.

15.7.2 Oracle Portlet Factory

Oracle Portlet Factory offers a "develop first, add later" portlet creation style. Developers use the Portlet Factory's UI to place Builders (such as buttons, loops, actions, and so on) inside Models. Code is generated automatically. Developers can view generated code, but are not required to work with it in any way. The development sequence for Oracle Portlet Factory is build the Model, deploy it to a producer, register the producer with the application that will consume the portlet, and then add the portlet to an application page.

15.7.3 Programmatic Portlets

Typically programmatic portlets offer a "develop first, add later" portlet creation style. Two wizards are available through Oracle WebCenter Framework to assist with the creation of Oracle PDK-Java and JSR 168 portlets. The wizards generate the basic files required for portlet creation. The developer hand-codes the portlet logic. The development sequence for programmatic portlets is to create the portlet, deploy it to a producer, register the producer with the application that will consume the portlet, and then add the portlet to an application page.

Note: With extensive coding, you can create "design-time at run time" Java portlets. For example, Web Clipping and OmniPortlet are both Java portlets.

15.8 User Interface Flexibility

This section describes the portlet building tools in terms of the control you have over the user interface.

15.8.1 Web Clipping

Because of its nature, Web Clipping always displays the remote Web site content, therefore UI flexibility is not a requirement for this portlet.

15.8.2 OmniPortlet

OmniPortlet enables you to use a number of different prebuilt layouts, such as scrolling news, tabular, and chart. You can also use the built-in HTML layout to personalize the look and feel of your portlet using HTML and JavaScript.

Note: When using JavaScript in portlets, developers must ensure that the JavaScript identifiers are qualified. That is, identifiers must be unique for each portlet instance and must not clash with the JavaScript on the page.

15.8.3 Oracle Portlet Factory

With Oracle Portlet Factory, you interact with Models and Builders. Each Builder has a wizard that steps you through the construction of Model content. Portlet code is generated based on the choices you make in the wizard. Portlet Factory simplifies the task of creating tables. Many out-of-the-box UI templates and styles ease the task of giving portlets a distinct look and feel.

15.8.4 Programmatic Portlets

In Java portlets, you have full control over your portlet's user interface. Your portlet is free to generate any HTML content that conforms to the rendering rules for pages.

15.9 Ability to Capture Content from Web Sites

This section describes the portlet building tools in terms of their ability to include content from other sources.

15.9.1 Web Clipping

For portlets that display content from a remote Web site as it is presented at the source location, the best tool to use is Web Clipping. Web Clipping can tolerate the changes of the source HTML page to some extent. If a clipped table moves from one place to another in the source page, then the Web Clipping engine can find the table again using the internal "fuzzy match" algorithm. Portlets built with Web Clipping can also maintain sessions to the remote Web sites. Web Clipping also supports user personalization of HTML form values.

15.9.2 OmniPortlet

For portlets using the data but not the layout from a remote Web site, the best choice is OmniPortlet. Use OmniPortlet to retrieve the data, process the data (format, filter, and so on), and present it in a portlet in a tabular, chart, or news format. OmniPortlet is a powerful tool that extracts data from Web pages by using its Web Page data source.

15.9.3 Oracle Portlet Factory

Oracle Portlet Factory is a tool for aggregating data, content, and processes from existing enterprise applications, such as SAP, and deploying them as custom portlets. As such, it has no facility for grabbing and re-using Web content.

15.9.4 Programmatic Portlets

Java portlets can take advantage of low-level Java networking APIs to retrieve and process content from remote Web sites. To avoid unnecessary development efforts, before choosing Java always make sure that Web Clipping or OmniPortlet are not viable options.

15.10 Ability to Render Content Inline

Active elements in portlets, such as links or form buttons, enable users to navigate to remote URLs. In a News portlet, for example, a user can click a hyperlink to navigate to a news site with detailed information about news of interest. For example, a user clicks a news-summary link in a News portlet, leaves the application page, and lands on the news site.

You may have a requirement to keep your users within the context of the application page by rendering the requested content within the same portlet container. For example, a user clicks a news-summary link in a News portlet, and the portlet refreshes with the detailed news article.

This maintenance of context is what rendering content inline means .

15.10.1 Web Clipping

The Web Clipping portlet supports URL rewriting for achieving inline content rendering. It can process the links originating from the source Web site and rewrite them to achieve the desired functionality.

The following options are available:

- Select not to rewrite the URLs within the portlet, in which case clicking the links takes users out of the WebCenter application to the Web site that provides the clipping. Whenever the link brings the user to a place that requires authentication, the user must enter login information before the link target is displayed.

- If the Web Clipping provider is registered with an external application and the clipping requires authentication, then you can instruct Web Clipping to rewrite all URLs within the portlet to point to the Login Server. In this case, navigation will cause the user to leave the WebCenter application, while also using the Login Server to log the browser into the External Application.
- Select to rewrite all URLs within the portlet (inline rendering) to point back to the page so that all browsing within the Web Clipping portlet remains within the WebCenter application. If the Web Clipping provider is registered with an External Application, then this will cause the Web Clipping provider to log itself in to the External Application. In this case, the navigation within the WebCenter application through the Web Clipping provider is authenticated in the External Application.

15.10.2 OmniPortlet

Rendering content inline is not supported, but you can achieve inline rendering using public portlet parameters.

15.10.3 Oracle Portlet Factory

Oracle Portlet Factory offers support for rendering a multipage portlet in the same portlet container, in other words, in line. Links to additional portlet pages are logical, rather than physical. That is, they map to a logical location rather than a physical address. This provides superior link integrity in instances where portlets are moved.

15.10.4 Programmatic Portlets

As you have full control over the links and buttons in Java portlets, you can easily implement the inline rendering functionality. To achieve inline rendering, you must append the private portlet parameters to the page URL.

If you use Struts in your portlet, then the PDK-Struts integration framework renders your content always in the same portlet container. Oracle recommends, however, that you use ADF Faces navigation for your new WebCenter application portlets.

If your portlet consists of multiple JSPs (for example, several steps in a survey or wizard), then your portlet can make use of a special parameter to specify at run time the JSP to use to render the content.

15.11 Charting Capability

This section describes the portlet building tools in terms of their charting functionality.

15.11.1 Web Clipping

Web Clipping clips pre-existing content. So, while it does not create charts, it can retrieve and present HTML content that contains charts.

15.11.2 OmniPortlet

OmniPortlet supports bar, line, and pie chart types. Charts are dynamically generated images, which can include hyperlinks.

15.11.3 Oracle Portlet Factory

Oracle Portlet Factory includes a demonstration version of the Greenpoint Web Chart builder. This builder enables the creation of sophisticated charts and graphs. A full license can be obtained from Greenpoint:

http://www.webcharts3d.com/

15.11.4 Programmatic Portlets

You can create sophisticated charts programmatically in Java portlets using Oracle's Business Intelligence (BI) Beans.

Note: Oracle Reports and Oracle Discoverer portlets use BI Beans to create professional graphs.

15.12 Public Portlet Parameter Support

Typically, a portlet's state is opaque (private); however, in Oracle WebCenter Framework portlets can describe public inputs (parameters) so values can be coordinated by the consuming application with other constituents of that application.

Inputs can include, public portlet parameters and private portlet parameters. These can be described as follows:

- **Public portlet parameters:** Use public portlet parameters to pass values to a portlet. Public parameters can assist with rendering portlet content that is specific to a particular page or user. Portlet parameters are created by the component developer and then exposed to the application developer through the user interface. After adding a portlet to a page, application developers can assign values to public portlet parameters to make the information displayed in the portlet specific to the page.

  Assigned values can be specific (such as a constant), a system variable (for example, the user name), or a page parameter. At run time, the portlet receives the values from the sources specified.

- **Private portlet parameters:** Use private portlet parameters to implement internal navigation in a portlet. Parameters are passed to the portlet every time the page is requested. Private portlet parameters can be passed exclusively from the portlet instance to the same portlet instance. Private portlet parameters do not require a full page refresh.

  Note: A Refresh action is available for inclusion on the portlet's Actions menu (`isNormalModeAvailable`). It refreshes the portlet without triggering a full page refresh. See Section 4.3.3, "Setting Attribute Values for the adfp:portlet Tag" for details about setting this action programmatically.

Portlets supporting public portlet parameters enable application developers to tailor data input for each portlet instance. The component developer can focus on the portlet logic, while the application developer can address the interaction between the application page and its portlets.

All portlet building technologies discussed in this chapter (OmniPortlet, Web Clipping, Oracle Portlet Factory, and programmatic portlets) support public portlet

parameters. OmniPortlet and Web Clipping provide complete support through their wizard interface. Oracle Portlet Factory supports public portlet parameters through the `RequestInputs` API. You can get the `RequestInputs` object from the `WebAppAccess` object. You can add public portlet parameter support to your programmatic portlets programmatically or with the Java Portlet wizard.

15.13 Private Portlet Parameter Support

This section describes the portlet building tools in terms of their support for private parameters.

15.13.1 OmniPortlet and Web Clipping

With the OmniPortlet and Web Clipping portlets, component developers do not have access to private portlet parameters.

15.13.2 Oracle Portlet Factory

Oracle Portlet Factory uses elements called *variables* that function much the same way as parameters within a family of Portlet Factory portlets.

15.13.3 Programmatic Portlets

In your Java portlets, you can implement internal navigation using private portlet parameters.

15.14 Ability to Hide and Show Portlets Based on User Privileges

This section describes the portlet building tools in terms of their support for authorization functionality.

15.14.1 Web Clipping and OmniPortlet

Dynamically hide and show portlets built with Web Clipping and OmniPortlet by using security managers. Although Web Clipping and OmniPortlet do not expose security managers through the user interface, they make them available for editing through their XML provider definition files.

15.14.2 Programmatic Portlets

PDK-Java provides a number of security managers for Java portlets. For example:

- **Group security manager:** The group security manager controls access to portlets based on group membership. For example, it shows the portlet to users who are members of a specified group, and hides the portlet from non-members.
- **Authentication level security manager:** The authentication level security manager controls access to the portlets based on authentication level. For example, it shows the portlet to authenticated users, and hides it from public users.

JSR 168 portlets support the standard servlet mechanisms.

15.15 Multilingual Support

This section describes the portlet building tools in terms of their support for other languages.

Web Clipping, OmniPortlet, and Java portlets display textual information in the language selected by the end user. Oracle Portlet Factory supports localized content through the following two Builders:

- Localized Resource
- Imported Page

15.16 Pagination Support

Support for pagination is useful when a portlet must display a relatively large set of records.

15.16.1 Web Clipping

Web Clipping does not support pagination.

15.16.2 OmniPortlet

OmniPortlet does not support pagination.

15.16.3 Oracle Portlet Factory

Oracle Portlet Factory supports pagination out-of-the-box. Select a check box, and specify the number of rows to display.

15.16.4 Programmatic Portlets

With Java, portlet pagination can be implemented programmatically.

15.17 Authenticating to External Applications

This section describes the portlet building tools in terms of authentication for external applications.

15.17.1 Web Clipping

Web Clipping's integration with the external application framework provides a fully automated mechanism to store passwords to external Web sites. All you must do is provide an External Application ID when registering the Web Clipping producer.

15.17.2 OmniPortlet

OmniPortlet enables you to store connection information when the data source is password protected. The credentials to access the data source can either be shared across all users or saved individually for each user. OmniPortlet is capable of storing database credentials as well as HTTP basic authentication user name-password pairs. Credentials are stored in a secured metadata services repository.

15.17.3 Oracle Portlet Factory

Access to external applications is supported through a Java class Builder that interacts with the PDK-Java API. In turn, the API communicates with the external application. After the initial login to the external application, the user's login credentials are saved in a credential store, from which they are subsequently retrieved.

15.17.4 Programmatic Portlets

Java portlets support programmatic integration with the external application framework as well as any LDAP server, such as Oracle Internet Directory.

16

Creating Portlets with OmniPortlet

This chapter provides an overview of OmniPortlet and explains the user interface elements associated with OmniPortlet. This chapter contains the following sections:

- Introduction to OmniPortlet
- OmniPortlet Wizard
- Parameters

For troubleshooting information regarding OmniPortlet, see Appendix G, "Troubleshooting WebCenter Applications". For information about registering and configuring OmniPortlet, see Section 4.3.1, "Registering Portlet Producers".

16.1 Introduction to OmniPortlet

OmniPortlet is a subcomponent of Oracle WebCenter Framework that enables developers to easily publish data from various data sources using a variety of layouts without writing any code. You can base an OmniPortlet on almost any kind of data source, such as a Web service, a SQL database, spreadsheet (character-separated values), XML, and even application data from an existing Web page.

> **Note:** You can find more information about developing different types of portlets in Chapter 14, "Understanding Portlets", and information about producers and other portlet technologies in Chapter 15, "Portlet Technologies Matrix".

OmniPortlet enables the WebCenter application developer and component developer to do the following:

- Display data from multiple sources (CSV, XML, Web service, SQL, and so on)
- Sort the data to display
- Format data using a variety of layouts (bulleted list, chart, HTML, and so on)
- Use portlet parameters
- Expose personalizable settings to page viewers

To display personalized data, you can refine the retrieved data by filtering the results returned from a data source, and parameterize the credential information used to access secure data. Out of the box, OmniPortlet provides the most common layout for portlets: tabular, chart, HTML, news, bulleted list, and form.

As described in Chapter 4, "Populating Pages", you can add an OmniPortlet to a JSP document created through Oracle JDeveloper. OmniPortlet is included in the

WebCenter Preconfigured Oracle Containers for J2EE (OC4J) that is installed with Oracle JDeveloper. After you start the WebCenter Preconfigured OC4J, you can register the OmniPortlet producer by using the Oracle JDeveloper PDK-Java Producer Registration wizard. When this producer is registered, the portlets become available on the Oracle JDeveloper Component Palette. From here, you can drag-and-drop the portlets onto your `*.jspx` page.

Note: For more information about:

- Registering the producer, see Section 4.3.1, "Registering Portlet Producers".
- Installing, initializing, and stopping the WebCenter Preconfigured OC4J and pointing Oracle JDeveloper to it, see Section 3.2, "Using the Preconfigured OC4J".
- Adding an instance of OmniPortlet to your page, see Section 4.3.2, "Adding Portlets to a Page".

16.2 OmniPortlet Wizard

Once you add an instance of OmniPortlet to your page, you must run your page to your browser, then click the **Define** link to start the OmniPortlet Wizard. For information about adding an instance of OmniPortlet to your page, see Section 4.3.2, "Adding Portlets to a Page".

Note: When you add an instance of OmniPortlet onto your page in Oracle JDeveloper, open the Property Inspector for the portlet and ensure that the AllModesSharedScreen and RenderPortletInFrame properties are set as follows:

- **AllModesSharedScreen** is set to `False` to display the Customize and Personalize in full page size.
- **RenderPortletInFrame** is set to `True` to display the OmniPortlet in its own iFrame in the View mode.

The OmniPortlet Wizard initially contains five steps. When you first define your OmniPortlet, you set the data source type, data source options, filter options, view options, and layout. When you've completed these steps of the wizard, you can reenter the wizard by clicking the **Customize** link for the portlet. When you reenter the wizard, you can change the definitions on the Source, Filter, View, and Layout tabs.

This section provides a high-level overview of the five tabs, which are described in Table 16–1. You can also find information in the online Help (accessible by clicking the **Help** link in the product), which describes the options on each tab.

Table 16–1 OmniPortlet Wizard and Customize Mode

Step/Tab	Description
Type	Provides your data source options. Displays only in the initial definition of the portlet, and is not available when customizing the defaults of the portlet.
Source	Provides the options for the selected data source, such as the URL of the Web Service you want to use. You can change these options later when editing the defaults of the portlet.

Table 16–1 (Cont.) OmniPortlet Wizard and Customize Mode

Step/Tab	Description
Filter	Provides sorting options at the WebCenter application level to enable you to refine your results. You can change these options later when editing the defaults of the portlet.
View	Provides options for displaying portlet header and footer text, the layout style, and caching. You can change these options later when editing the defaults of the portlet.
Layout	Provides detailed options for customizing the layout. You can change these options later when editing the defaults of the portlet.

16.2.1 Type

When you first start OmniPortlet, the Type step displays, which enables you to choose your data source (Figure 16–1). Out of the box, OmniPortlet supports the data sources shown in Table 16–2.

Figure 16–1 Type Tab of the OmniPortlet Wizard

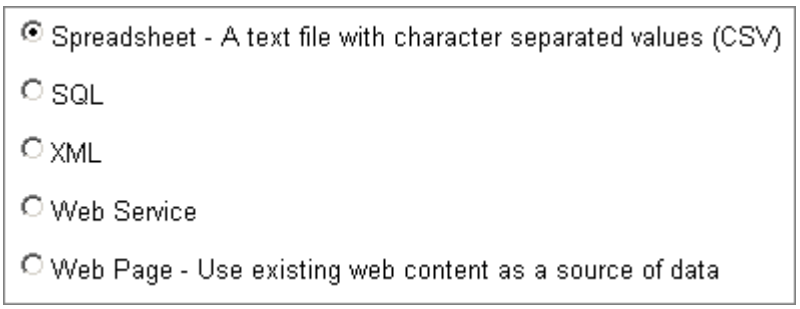


Table 16–2 Supported Data Source Types

Data Source Type	Description
Spreadsheet	Displays data from a text file containing character-separated values (CSV).
SQL	Displays data from a database using SQL.
XML	Displays data from an XML file.
Web Service	Displays data from a discrete business service that can be accessed over the Internet using standard protocols.
Web Page	Displays data based on existing Web content.

After you complete the OmniPortlet Wizard and edit the defaults of the portlet, you cannot change the data source type.

16.2.2 Source

After you've chosen your data source type, the Source step of the OmniPortlet Wizard displays. This step adapts to the data source you've chosen, enabling you to specify the options offered by that data source. The Source tab contains a Proxy Authentication section if the OmniPortlet producer has been configured to use a proxy server requiring authentication, and a Connection section where you can provide the necessary information for connecting to the data source.

This section contains information about the following two common areas on the Source tab:

- Proxy Authentication
- Connection Information

Later, this section also describes the portion of the Source tab specific to each data source. The data sources available are as follows:

- Spreadsheet
- SQL
- XML
- Web Service
- Web Page

> **Note:** For more information about the Source tab options, click **Help** in the upper right corner of the page.

16.2.2.1 Proxy Authentication

OmniPortlet supports proxy authentication, including support for global proxy authentication and authentication for each user. You can specify whether all users will automatically log in using a user name and password you provide, each user will log in using an individual user name and password, or all users will log in using a specified user name and password. If the OmniPortlet producer has been set up to use proxy authentication that requires your login, then a Proxy Authentication section displays on the Source tab where you can enter this information.

The Proxy Authentication section only displays for the following data sources, which may require you to use a proxy server to access them:

- CSV (character-separated values)
- XML
- Web Service
- Web Page

For more information about configuring the OmniPortlet producer to use proxy authentication, see the online Help topic that displays when you click **Help** on the Edit Producers: OmniPortlet Producer page. If the OmniPortlet producer is configured to "Require login for all users," then each user must set his or her own login information as follows:

- For page designers, set this in Customize: Source tab.

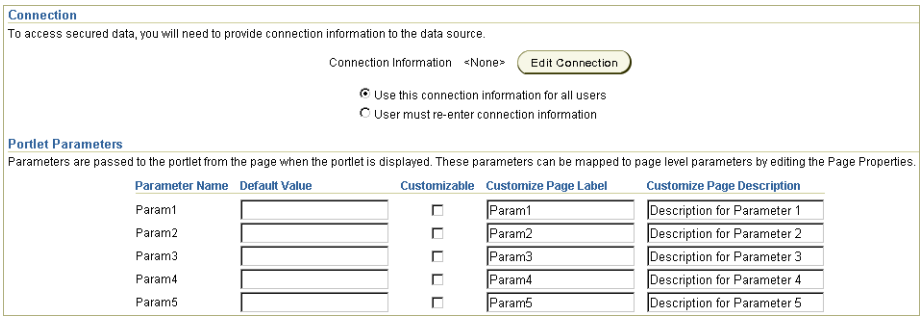
- For page viewers, set this on the Personalize screen.

> **Notes:** To access the Customize: Source tab, click the **Customize** link for the portlet you want to modify.
>
> If you are using the Web Page data source, then the Proxy Authentication section displays in the Web Clipping Studio, after you have clicked the Select Web Page button on the Source tab.

16.2.2.2 Connection Information

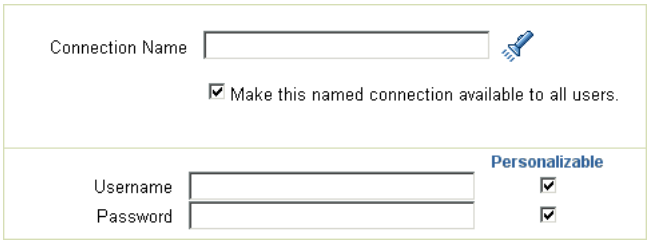
For each data source except the Web Page data source, the Source step contains a Connection section, where you can define the connection information to access secured data. The Source step for all data sources include a Portlet Parameters section, where you can define the parameters for the portlet (Figure 16–2). You can then map the portlet parameters to the page-level parameters.

Figure 16–2 Source Tab: Connection and Portlet Parameters Section



To edit the connection information, click the **Edit Connection** button and fill out the information about the page shown in Figure 16–3. On this page, you can enter a name for the connection information, as well as the user name and password. For the SQL data source, you can enter more information to specify the driver you want to use to connect to the data source. For more information, see Section 16.2.2.4, "SQL".

Figure 16–3 Edit Connection Page



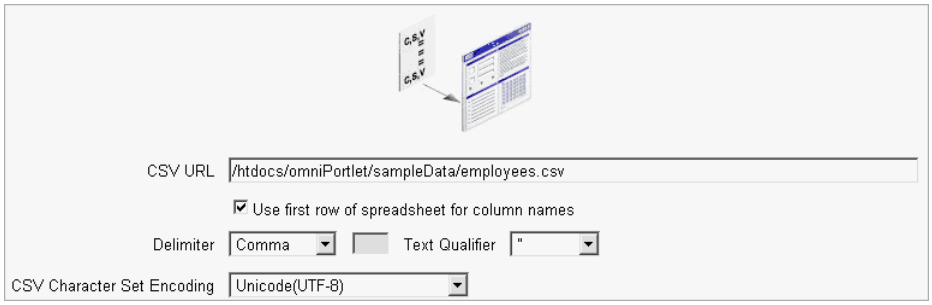
Note: For more information about the Connection section and the Edit Connection button, click **Help** on the Source tab of the OmniPortlet wizard.

16.2.2.3 Spreadsheet

Spreadsheets are a common method of storing small data sets. OmniPortlet enables you to share spreadsheets by supporting character-separated values (CSV) as a data source. On the Source tab, you specify the location of the CSV file (Figure 16–4). If the file is located on a secure server, then you can specify the connection information in the Connection section described in Figure 16–2. You can also select the character set to use when WebCenter Suite reads the file, as well as the delimiter and text qualifier.

Note: As the OmniPortlet producer exists and executes in a tier different from the WebCenter application and does not have access to the session information, you must expose CSV files as PUBLIC in order for OmniPortlet to access them.

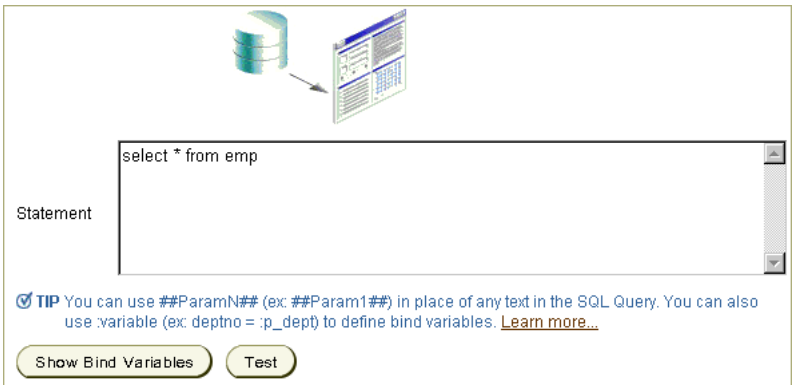
Figure 16–4 Source Tab: Spreadsheet



16.2.2.4 SQL

The relational database is the most common place to store data. OmniPortlet enables you to use standard JDBC drivers and provides out-of-the-box access to Oracle and any JDBC database. You can specify the driver type when you configure the connection information. Figure 16–5 shows the Source tab for a SQL data source.

Figure 16–5 Source Tab: SQL



16.2.2.4.1 SQL Connection Information You can use the DataDirect JDBC drivers to access other relational databases. To configure OmniPortlet to use these drivers, see Section B.2.2, "Configuring the OmniPortlet Producer to Access Other Relational Databases"

After the driver is installed, you'll notice it listed in the Driver Name list on the Connection dialog box on the Source tab, as shown in Figure 16–6.

Figure 16–6 Connection information about the SQL Source Tab

Note: For more information about DataDirect drivers, see the *Certification Matrix for Oracle Application Server and DataDirect JDBC* (http://www.oracle.com/technology/tech/java/oc4j/htdocs/datadirect-jdbc-certification.html) and the OC4J page on Oracle Technology Network (OTN) (http://www.oracle.com/technology/software/products/ias/htdocs/utilsoft.html).

When you want to use one of the DataDirect drivers, you must use a unique connection string format: `hostname:port`, where *hostname* is the name of the server where the database is running, and *port* is the listening port of the database.

16.2.2.4.2 Using Stored Procedures You can also make a call to Stored Procedures instead of SQL statement to add business logic to your data. You can create your package and stored procedure in your database and refer the stored procedure in OmniPortlet.

For example, you could do the following using the SCOTT sample schema:

1. Create a package and declare a ref cursor:

```
create or replace package emp_pack is
type empcurr is ref cursor;
end;
```

2. Define a stored procedure, for example following procedure accepts `JOB` as parameter and returns a ref cursor, where JOB Column in the `scott.Emp` table, its value can be `CLERK`, `MANAGER`, and so on.

```
create or replace procedure emp_proc(eset OUT emp_pack.empcurr,
jname IN VARCHAR2)
is
sql_statement varchar2(200);
begin
sql_statement := 'select empno,ename,hiredate
from emp
where job = '''||jname||'''
order by EMPNO,hiredate';
open eset for sql_statement;
end;
```

3. Add the PL/SQL statements from steps 1 and 2 to a SQL file (for example, `proc.sql`) and save it to a directory.
4. Connect to the database using the following command:

```
sqlplus userid/password@Connection_String
```

   Replace `userid`, `password`, and `Connection_String` the connection information to your database. You can find the connection string in the `tnsnames.ora` file within your `ORACLE_HOME/network/admin` directory.

5. Run the procedure:

```
@proc
```

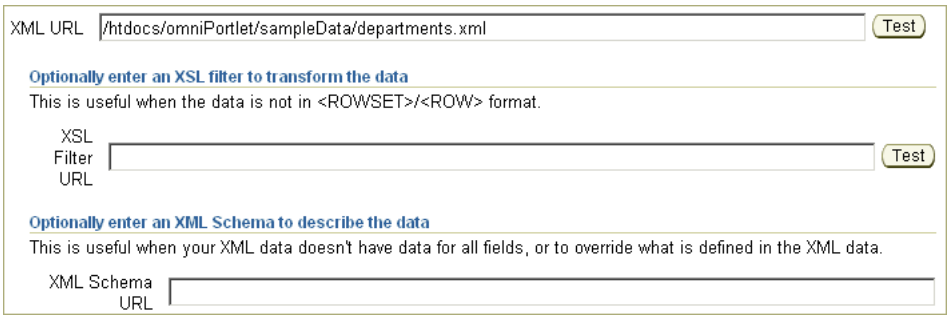
6. Finally, create an OmniPortlet based on the SQL data source, enter the appropriate database connection information. In the SQL Statement box, enter the following code:

```
call emp_proc('CLERK')
```

16.2.2.5 XML

You can access XML data sources across the intranet or Internet. On the Source tab, you can specify the URL of the XML file that contains your data as shown in Figure 16–7.

Figure 16–7 Source Tab: XML



Next to the XML URL and the XSL Filter URL fields are Test buttons which you can use to validate your XML data source and the XSL filter.

The specified XML file can either be in tabular (ROWSET/ROW) structure, or you can provide an XML Style Sheet (XSL) to transform the data into the ROWSET/ROW structure. The following image shows an example of the ROWSET/ROW structure of an XML data source.

```
<TEAM>
  <EMPLOYEE>
    <DEPTNO>10</DEPTNO>
    <ENAME>KING</ENAME>
    <JOB>PRESIDENT</JOB>
    <SAL>5000</SAL>
  </EMPLOYEE>
    <DEPTNO>20</DEPTNO>
    <ENAME>SCOTT</ENAME>
    <JOB>ANALYST</JOB>
    <SAL>3000</SAL>
  <EMPLOYEE>
</TEAM>
```

In this example, the <TEAM> tags delineate the rowset, and the <EMPLOYEE> tags delineate the rows.

Regardless of the format of the XML file, OmniPortlet automatically inspects the XML to determine the column names, which will then be used to define the layout. If you want to specify this information yourself, then you can supply a URL to an XML schema that describes the data.

Similar to the other data sources, you can also specify the connection information for this data source, if the XML file is located on a secured server protected by HTTP Basic Authentication.

Note: As the OmniPortlet producer exists and executes in a tier different from the WebCenter application and does not have access to the session information, you must expose XML files as PUBLIC in order for OmniPortlet to access them.

16.2.2.6 Web Service

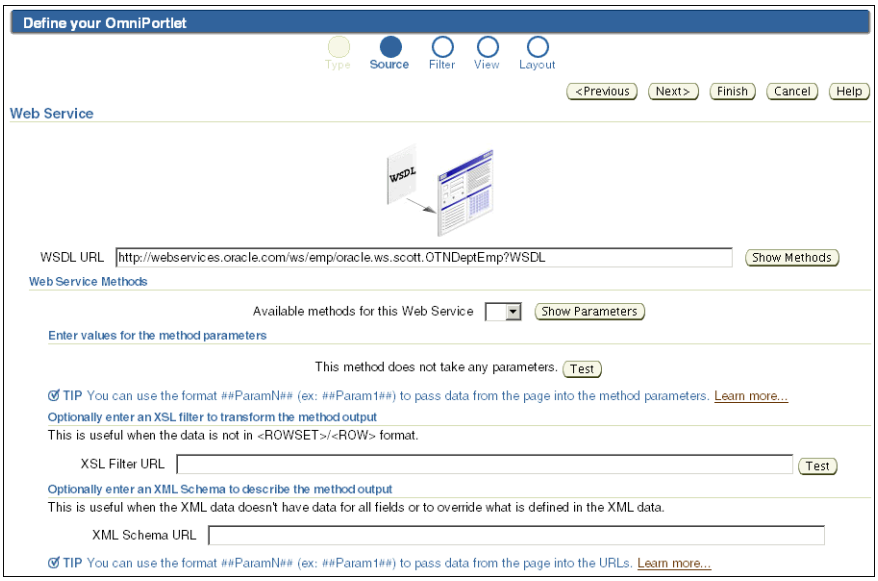
A Web Service is a discrete business service that can be programmatically accessed over the Internet using standard protocols, such as SOAP and HTTP. Web Services are not specific to platform and not specific to language specific, and are typically registered with a Web Service broker. When you find a Web Service you want to use, you must obtain the URL to the Web Service Description Language (WSDL) file that describes the Web Service and specifies the methods that can be called, the expected parameters, and a description of the returned data.

OmniPortlet supports both types of Web Services, Document and RPC (Remote Procedure Calls). After a WSDL document/file is supplied, it is parsed, and the available methods that can be called display on the Source tab.

Similar to the XML data source, OmniPortlet expects the Web Service data in ROWSET/ROW format, though you can also use an XSL file to transform the data. OmniPortlet inspects the WSDL document/file to determine the column names, though you may also specify an XML schema to describe the returned data set.

Figure 16–8 shows the Source tab for a sample Web service.

Figure 16–8 Source Tab: Web Service



16.2.2.7 Web Page

OmniPortlet enables you to use existing Web content as a source of data to publish information to your WebCenter application. It provides and renders clipped Web content as a data source.

The Web Page data source extends the scope offered by the Web Clipping Portlet to include scraping functionality. It also supports the following features:

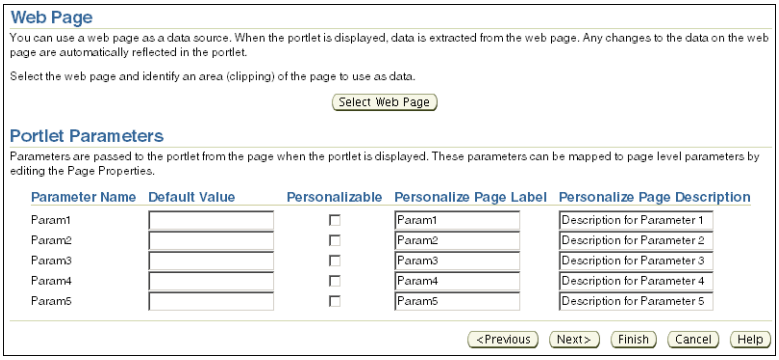
- **Navigation through various login mechanisms**, including form- and JavaScript-based submission, and HTTP Basic and Digest Authentication with cookie-based session management.
- **Fuzzy matching of clippings.** If a Web clipping gets reordered within the source page or if its character font, size, or style changes, then it will still be identified correctly by the Web page data source and delivered as the portlet content.
- **Reuse of a wide range of Web content**, including basic support of pages written with HTML 4.0.1 and JavaScript, retrieved through HTTP GET and POST (form submission).

All Web clipping definitions are stored persistently in the Oracle Metadata Services (MDS) by default. However, you can alternatively use an Oracle database. Using MDS does not require any changes in the configuration files. If you use an Oracle database as the Web Clipping repository, then you must update the `provider.xml` file.

Any secure information, such as passwords, is stored in encrypted form, according to the DES (Data Encryption Standard), using Oracle Database encryption technology.

The Source tab of the OmniPortlet Wizard (Figure 16–9) enables you to start the Web Clipping Studio by clicking the Select Web Page button. Once you start the Web Clipping Studio, you can see the Oracle Application Server Web Clipping online Help.

Figure 16–9 Source Tab: Web Page

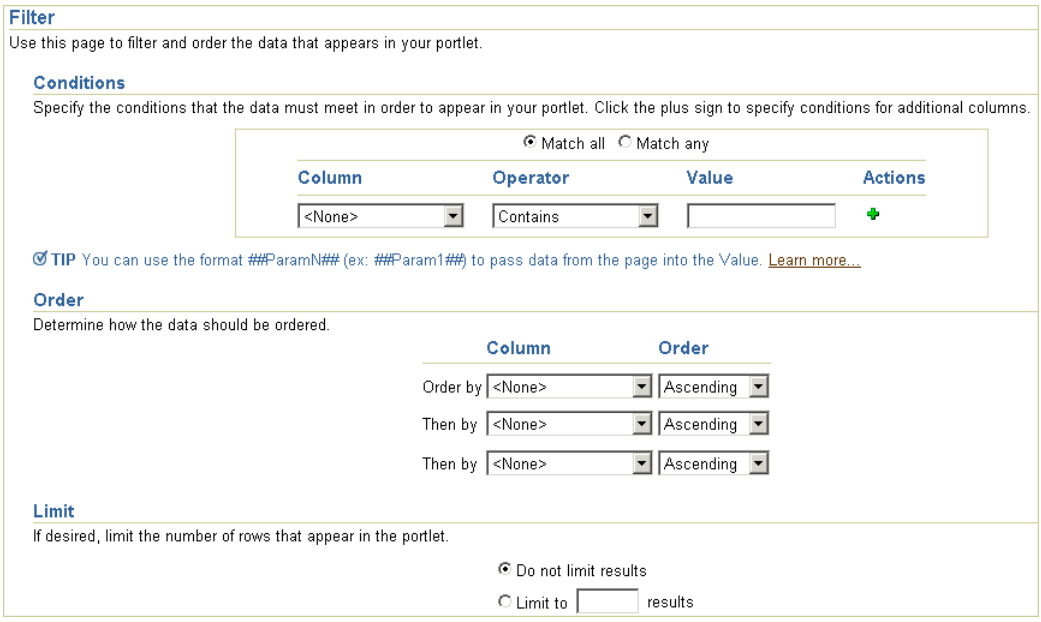


Note: For more information about using the Web Clipping Studio or creating a Web Clipping Portlet, see Chapter 17, "Creating Content-Based Portlets with Web Clipping".

16.2.3 Filter

After you've selected the data source and specified the data source options, you can further refine your data by using OmniPortlet's filtering options. To use filtering efficiently, it is better to refine the data as much as possible at the data source level on the Source tab, then use the options on the Filter tab to streamline the data. For example, if you are using a SQL data source, then you could use a WHERE clause to return only specific data from the specified columns. In this case, you could skip the Filter tab and continue to the View page of the wizard. However, if there are no filtering options at the data source level, then you can use the options on the Filter tab to sort your data (Figure 16–10).

Figure 16–10 Filter Tab

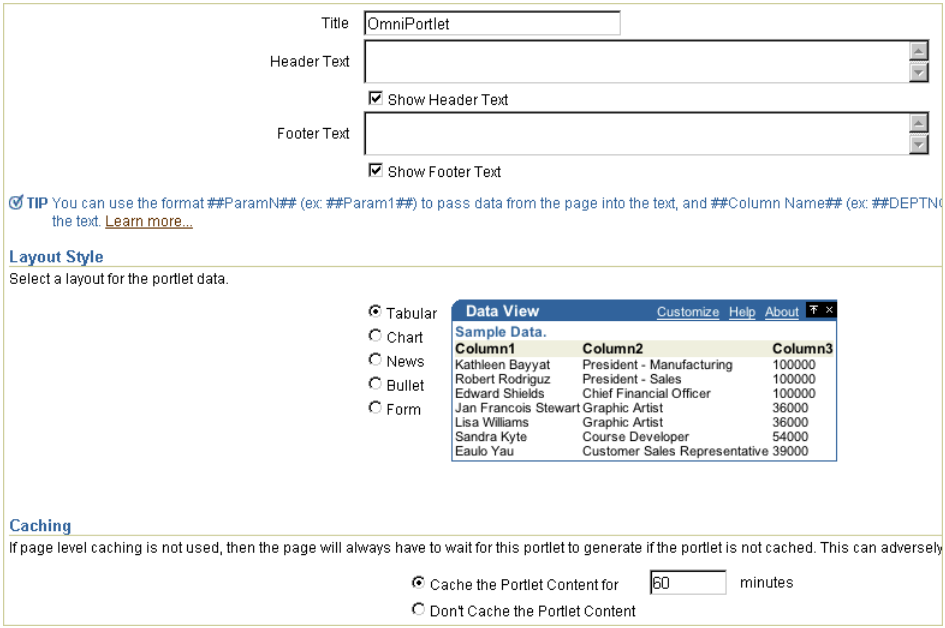


16.2.4 View

Once you've specified the data and sorted it, you can choose the view options and layout for your OmniPortlet. The View tab (Figure 16–11) enables you to add Header and Footer text, choose a Layout style that you can later refine on the Layout tab, and enable caching. You can choose from the following layouts:

- Tabular
- Chart
- News
- Bullet
- Form
- HTML

Figure 16–11 View Tab



Note: For more information about the different layout styles you can use with OmniPortlet, see the next section or click **Help** in the upper right corner of the page in the OmniPortlet Wizard.

16.2.5 Layout

The Layout tab changes depending on the Layout Style you chose on the View tab, and enables you to further customize the appearance of your portlet. For example, OmniPortlet supports drill-down hyperlinks in the chart layout. That is, you can set up the chart so that when a user clicks on a specific part of the chart, an action occurs (for example, jump to another URL).

For the other layout styles, you can define each column to display in a specific format, such as plain text, HTML, an image, button, or field. For example, suppose you selected a data source that includes a URL to an image. To see this image, you can select Image for the display of this column. Each column can also be mapped to an action, similar to the behavior of chart hyperlinks.

The following layout styles are available with OmniPortlet:

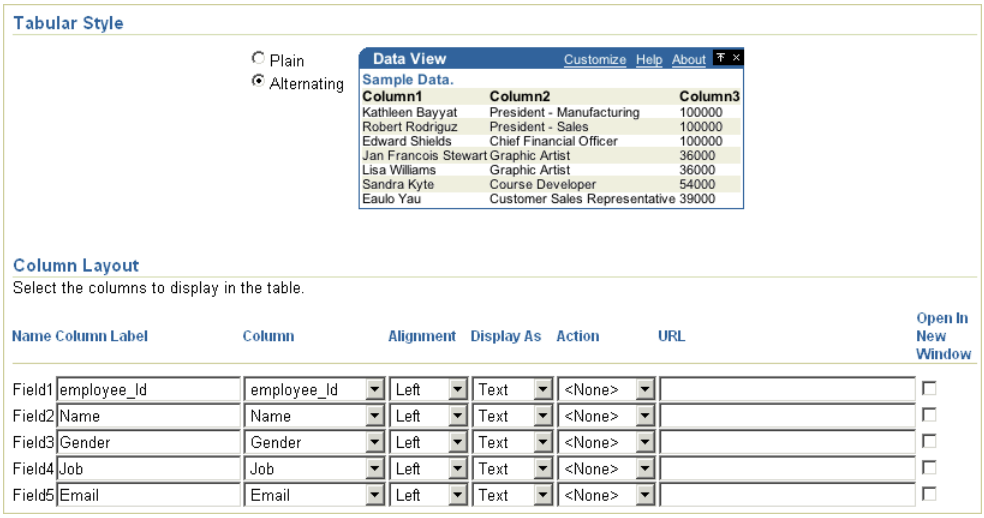
- Tabular Layout
- Chart Layout
- News Layout
- Bullet Layout
- Form Layout
- HTML Layout

Note: As events are not currently supported, selecting an action when designing your layout may produce unexpected results.

16.2.5.1 Tabular Layout

Once you've chosen the tabular style on the View tab, you can refine the layout on the Layout tab (Figure 16–12). Typically, you use the tabular layout if you have one or more columns of data that you want to display in a table. You can choose Plain to display all rows in the table without any background color, or Alternating to display a background color for every other row in the table.

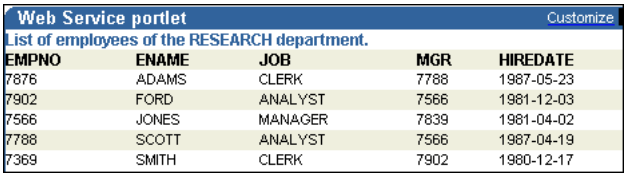
Figure 16–12 Layout Tab: Tabular Style



Note: You can control the background color of a portlet by using styles, as described in Chapter 9, "Defining and Applying Styles to Core Customizable Components".

In the Column Layout section, you can choose which data columns to display in the portlet, then select a display format for the data. Here, you can set a column to display a hyperlink, so that a secondary Web page displays when the user clicks that column in the table. You can also specify whether the secondary Web page displays in a new window. Figure 16–13 shows an example of an OmniPortlet using a tabular format.

Figure 16–13 Example of an OmniPortlet Using a Tabular Layout



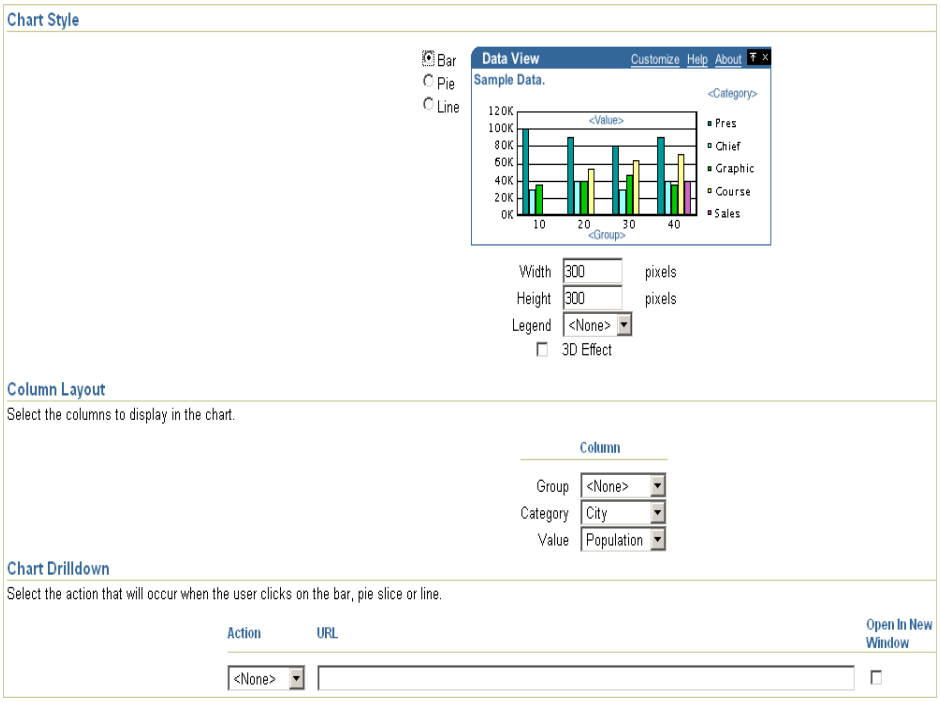
Note: For more information about using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

16.2.5.2 Chart Layout

You can use the chart layout to display your data graphically, as a bar, pie, or line chart. On the Layout tab (Figure 16–14), you select the chart style and the column layout. When you choose the column layout, you can choose the groups, or columns

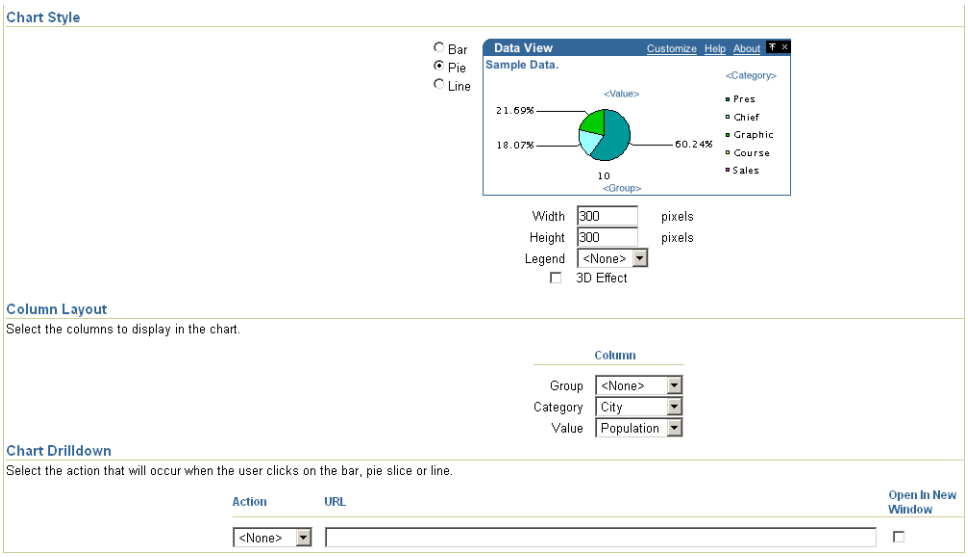
on which the labels will be based. The category defines the values that will be used to create the chart legend, and the value determines the relative size of the bars, lines, or slices in the chart. You can also select whether the sections of the chart should point to a hyperlink, and whether the targeted information should display in a new window. Figure 16–15 shows an example of the Layout tab for a pie chart layout.

Figure 16–14 Layout Tab: Chart



Note: To group the information in the chart, you must group the information at the data level (for example, in your SQL query statement). Also, if numeric values in a data source contain formatted strings, commas, or currency (for example, $32,789.00), then they are considered to be text and ignored when the chart is generated. You should remove these formatting characters if you want them to be correctly read as numerical values.

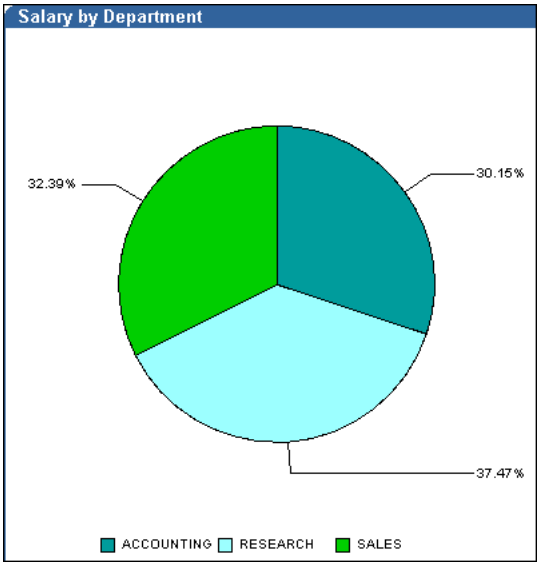
Figure 16–15 Example of the Layout Tab for a Pie Chart Layout



You can also define chart hyperlinks so that each bar, pie section, or line links to another Web page. For example, you can display a chart portlet and a report portlet on your page, then set up the chart hyperlink to display a row in the report that displays more detailed information about the selected data.

In Figure 16–16, you can see the results of the options selected on the Layout tab in the previous image. Below the chart, you can see that the category, which was Department on the Layout tab, is used for the legend.

Figure 16–16 Example of an OmniPortlet Using a Pie Chart Layout



16.2.5.3 News Layout

You can use the news layout to display links to articles with brief descriptions for each. You can use this layout to publish information in standard XML formats, such as RDF (Resource Description Framework) or RSS (RDF Site Summary) to your page. In the Column Layout section (Figure 16–17), you can add a heading that displays at the top of the portlet. You can also add a logo, or use the scrolling layout so that the user can

view all the information in the portlet as it moves vertically. Here, also, you can enter a URL so that another Web page displays when the user clicks on specific data in the portlet. You can also specify whether the secondary Web page displays in a new window.

Figure 16–17 Layout Tab: News

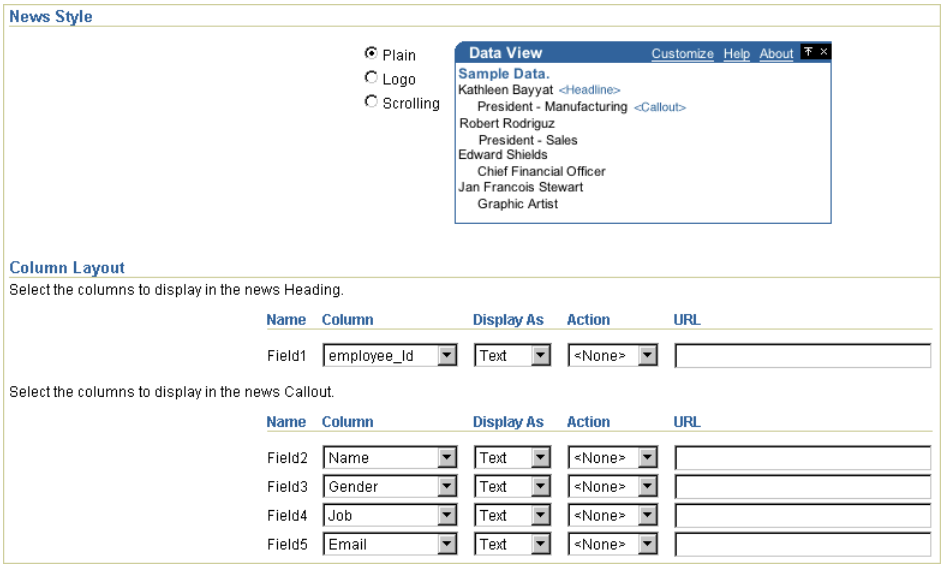
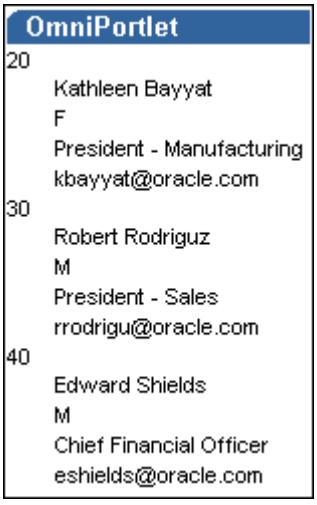


Figure 16–18 shows an example OmniPortlet using a news layout.

Note: The News Layout Scroll type in OmniPortlet is supported on Microsoft Internet Explorer and Netscape 7.0.

Figure 16–18 Example of an OmniPortlet Using a News Layout



Note: For more information about using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

16.2.5.4 Bullet Layout

You can use the bullet layout to display your data in a bulleted list. The Layout tab (Figure 16–19) provides a variety of different bullet and numbered bullet styles. In the Column Layout section, you can choose how the columns will display in the portlet, as well as whether a second Web page will display when the user clicks that column. You can also specify whether the second Web page displays in a new window.

Figure 16–19 Layout Tab: Bullet

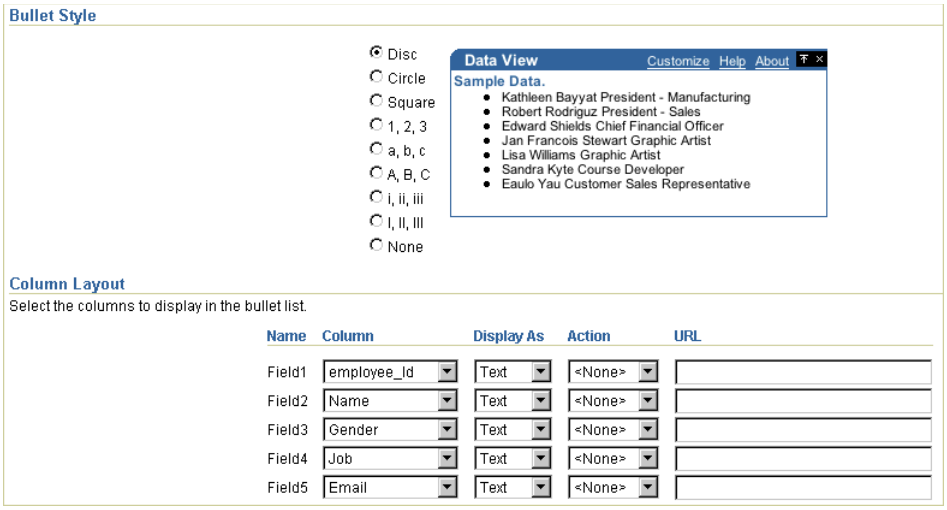


Figure 16–20 shows an example of an OmniPortlet using a bullet layout.

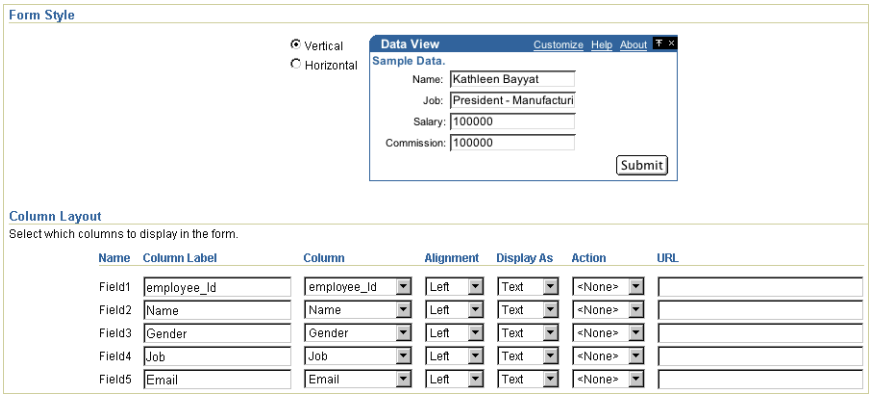
Figure 16–20 Example of an OmniPortlet Using a Bullet Layout

Note: For more information about using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

16.2.5.5 Form Layout

You can use the form layout (Figure 16–21) if you have data you want to display as labels or default values in a form, such as Name: <name>. You can then use portlet parameters to pass data to the selected row.

Figure 16–21 Layout Tab: Form



You can also specify whether to display the target of a URL in a new window (Figure 16–22). Figure 16–23 shows an example of an OmniPortlet using a Form layout.

Figure 16–22 Open In New Window Check Box

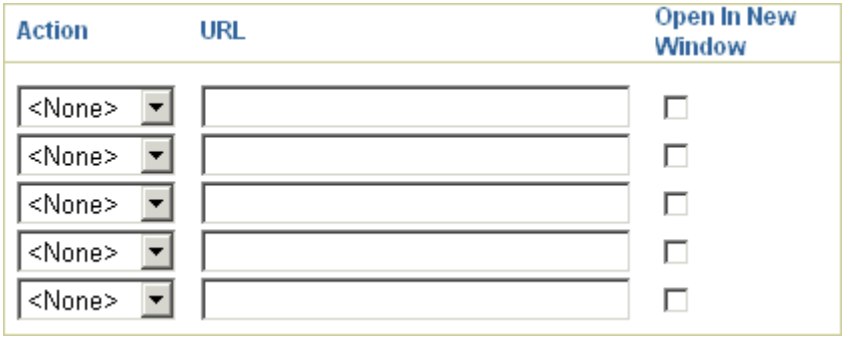
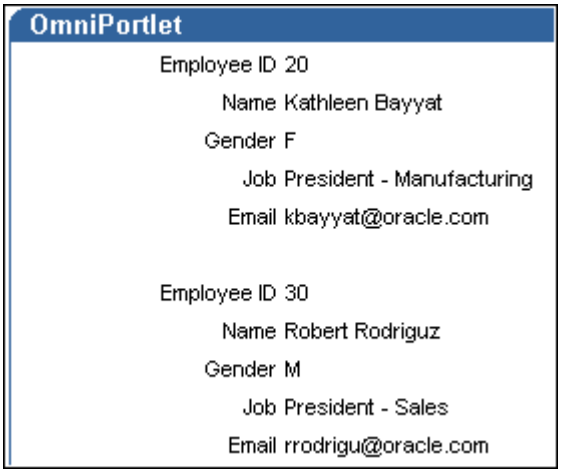


Figure 16–23 Example of an OmniPortlet Using a Form Layout



Note: For more information about using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

16.2.5.6 HTML Layout

You can use the HTML layout to create a customized look and feel for your portlet by choosing from either a built-in HTML layout and modifying the code, or by creating a new layout from scratch. You can hand-code your own HTML or JavaScript based on data columns that OmniPortlet has retrieved based on the selected data source (Figure 16–24). By coding your own HTML and JavaScript, you have full control over the appearance and develop a rich interface for your portlet.

For more information about using the fields on this tab, click the **Help** button in the wizard. For an example of using JavaScript in the HTML layout, choose the Sortable Table layout from the Quick Start list on this tab.

> **Note:** The maximum number of characters you can enter in each of the sections (Heading, Repeating, and Footer) is 30,000 (30k).

Figure 16–24 Layout Tab: HTML

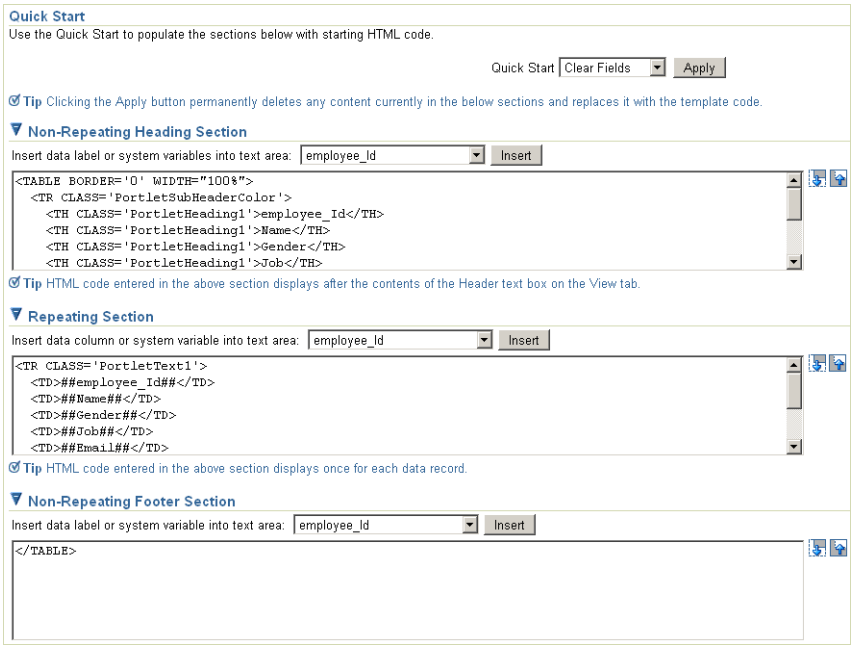


Figure 16–25 shows an example of an OmniPortlet using the HTML layout.

Figure 16–25 Example of an OmniPortlet Using the HTML Layout

16.2.6 Customize mode

After you have created your OmniPortlet and returned to your application, you can click the **Customize** link for your portlet to change the portlet options if required. You will notice that, in the Customize mode, there are tabs that correspond to the different steps in the OmniPortlet Wizard (except for the Type step) to directly access the different options.

When you edit an OmniPortlet using the Customize mode, keep in mind the following notes:

- Any modifications you make to your portlet using the Customize mode apply to all users, regardless of the current session language and the locale of the user's browser.
- You can personalize the portlet at run time by clicking the **Personalize** link on the portlet. Personalizing the portlet creates a copy of the personalization object. As all properties are duplicated, subsequently modifying the portlet through the Customize mode does not affect the personalized version of the portlet. To ensure the latest Customize changes are made to the portlet, you must click **Personalize** again (after you have made the modifications in the Customize mode), then select the **Reset to Defaults** option.
- By default, the OmniPortlet producer uses the file-based Preference Store to store the personalization object, which stores the object in a file system in the middle-tier. If you decide to deploy OmniPortlet in a multiple middle-tier environment, then you must configure the File Preference Store to a shared file system, or use the Database Preference Store (DBPreferenceStore). To do so, follow the steps in Section B.2.3, "Configure Portal Tools and Web Producers (Optional)".

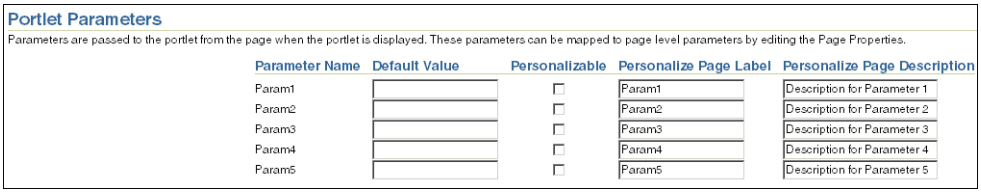
16.3 Parameters

You can define up to five portlet parameters for an OmniPortlet. You can define parameters in the following screens:

- On the Source tab of the wizard when you define the OmniPortlet
- On the Source tab when you select **Customize** for the OmniPortlet

Figure 16–26 shows the Portlet Parameters section on the Source tab.

Figure 16–26 Source Tab: Portlet Parameters Section



Note: You can learn more about portlet parameters in the online Help, which you can access by clicking the **Help** link on the Source tab in the OmniPortlet Wizard. The online Help describes portlet parameters in detail, and how to set them up for your OmniPortlet.

Once you have set up portlet parameters in your OmniPortlet, you can contextually map the portlet to other portlets or components on a page. For more information about doing so, see Section 4.5, "Contextually Linking Components".

16.4 Summary

In this chapter, you learned about OmniPortlet and its capabilities. You also learned about using parameters. You can find more information about using the various tools in OmniPortlet by clicking the **Help** link on each of the pages in the wizard.

Note: For more information about configuring OmniPortlet, see Appendix B.2, "OmniPortlet Configuration Tips".

17

Creating Content-Based Portlets with Web Clipping

This chapter provides a brief description of Web Clipping portlets and producers and explains how you can register a Web Clipping producer and use this producer to add Web Clipping portlets to a JSP document created through Oracle JDeveloper.

This chapter contains the following sections:

See Appendix B, "Additional Portlet Configuration" for additional Web Clipping portlet configurations like configuring the repository, configuring proxy settings, and securing the Web Clipping producer.

17.1 Introduction to Web Clipping

Web Clipping is a browser-based declarative tool that enables you to integrate any Web application with your WebCenter application. It is designed to give you quick integration by leveraging the Web application's existing user interface. You can drag-and-drop Web Clipping portlets on to a `*.jspx` page. The Web Clipping producer and portlets are available when you install the WebCenter Preconfigured Oracle Containers for J2EE (OC4J), which is available by default as part of an Oracle WebCenter Framework installation. When this producer is registered, the portlets become available on the Oracle JDeveloper Component Palette. From here, you can drag-and-drop the portlets onto your `*.jspx` page.

Notes:

- For more information about installing, initializing, and stopping the WebCenter Preconfigured OC4J and pointing Oracle JDeveloper to it, see Section 3.2, "Using the Preconfigured OC4J".
- To learn more about registering a Web producer, see Section 4.3.1, "Registering Portlet Producers".

With Web Clipping, you can collect Web content into portlets in a single centralized Web page. You can use Web Clipping to consolidate content from Web sites scattered throughout a large organization.

Web Clipping enables clipping of an entire Web page or a portion of it and reusing it as a portlet. Basic and HTML-form-based sites may be clipped. Use Web Clipping when you want to copy content from an existing Web page and expose it in your WebCenter application as a portlet. Web Clipping portlets supports the following:

- **Navigation through various styles of login mechanisms**, including form- and JavaScript-based submission and HTTP Basic and Digest Authentication with cookie-based session management.
- **Fuzzy matching of clippings**, meaning that if a Web clipping gets reordered within the source page or if its character font, size, or style changes, then it will still be identified correctly by the Web Clipping engine and delivered as the portlet content.
- **Reuse of a wide range of Web content**, including basic support of pages written with HTML 4.0.1, JavaScript, applets, and plug-in enabled content, retrieved through HTTP GET and POST (form submission).
- **Personalization**, enabling page designers to expose input parameters that page viewers can modify when they personalize the portlet. These parameters can be exposed as public parameters that a page designer can map as page parameters. This feature enables end users to obtain personalized clippings.
- **Integrated authenticated Web content through Single Sign-On**, including integration with external applications, which enables you to leverage Oracle Single Sign-On and to clip content from authenticated external Web sites.
- **Inline rendering**, enabling you to set up Web Clipping portlets to display links within the context of the portlet. As a result, when a user clicks a link in the Web Clipping portlet, the results display within the same portlet. You can use this feature with internal and external Web sites.
- **Proxy authentication**, including support for global proxy authentication and authentication for each user. You can specify proxy server authentication details including type (Basic or Digest) and realm in the `provider.xml` file. In addition, you can specify one of the following schemes for entering user credentials:
  - All users automatically log in using a user name and password you provide.
  - All users will need to log in using a user name and password they provide.
  - All public users (not authenticated into the WebCenter application) automatically log in using a user name and password you provide, while valid users (authenticated into the WebCenter application) will need to log in using a user name and password they provide.

  See the Section B.3.2, "Configuring HTTP or HTTPS Proxy Settings" for more information.
- **Navigation and clipping of HTTPS-based external Web sites**, if appropriate server certificates are acquired.
- **Clipping of page content from HTML 4.0.1 pages**, including the following:
  - Clipping of <applet>, <body>, , <embed>, <img>, <object>, <ol>, , <table>, and <ul> tagged content
  - Preservation of <head> styles and fonts, and Cascading Style Sheets (CSS)
  - UTF-8 compliant character sets

  - Navigation through hyperlinks (HTTP GET), form submissions (HTTP POST), frames, and URL redirection

- **Globalization Support** in URLs and URL parameters. See Section 17.5, "Current Limitations for Web Clipping" for information about how Web Clipping determines the character set of clipped content.

You can select one of the following as the Web Clipping repository:

- Oracle Metadata Services
- Oracle Application Server infrastructure database
- Other Oracle Database of version 9i or later

Oracle Metadata Services is the default option, which saves the Web Clipping definition in the file system. If you select Oracle Metadata Services as your Web clipping repository, then you can use Web Clipping even without a database. Web Clipping definitions can be stored persistently in the repository.

Any secure information, such as passwords, is stored in encrypted form, according to the Data Encryption Standard (DES), using Oracle encryption technology.

> **See Also:** Section B.3.1, "Configuring the Web Clipping Repository" for details about configuring a repository.

17.2 Adding Web Page Content to a Page

To add Web page content to a page, follow the steps described in the following sections:

- Section 17.2.1, "Registering a Web Clipping Producer"
- Section 17.2.2, "Adding a Web Clipping Portlet to a Page"
- Section 17.2.3, "Selecting a Section of a Web Page to Display in the Web Clipping Portlet"
- Section 17.2.4, "Setting Web Clipping Portlet Properties"

17.2.1 Registering a Web Clipping Producer

To add Web Clipping portlets to your page, you must first register the Web Clipping producer. Register your Web Clipping producer by referring to Section 4.3.1, "Registering Portlet Producers" for details.

17.2.2 Adding a Web Clipping Portlet to a Page

To add a Web Clipping portlet to a `*.jspx` page, perform the following steps:

1. Start Oracle JDeveloper and the WebCenter Preconfigured OC4J (see Section 3.2.2, "Starting and Stopping the Preconfigured OC4J").
2. In the Applications Navigator, right-click the `*.jspx` file, and select **Open** from the context menu.

   The `*.jspx` file is available in the following hierarchy:

```
Applications
 <ApplicationName>
  <ProjectName>
   Web Content
```

3. In the Component Palette, select the Web Clipping producer that you registered earlier.
4. Select a Web Clipping portlet, and drag it on top of the `h:form` component in the Structure pane in Oracle JDeveloper.

   If you are using a `PanelCustomizable` or `ShowDetailFrame` component, then drag the portlet on top of that component instead of `h:form`. In the Structure pane, the Web Clipping portlet should display under the `PanelCustomizable` or `ShowDetailFrame` component. In the Page Editor, the portlet should display inside the `PanelCustomizable` or `ShowDetailFrame` component.

   Note: Ensure that the new Web Clipping portlet is selected in the Structure pane, and in the Property Inspector set `AllModesSharedScreen` to `false`.

   If you do not set this property to `false`, when you personalize the Web Clipping portlet at run time, the text displayed on the page may be distorted.

5. Right-click the `*.jspx` file and select **Run** from the context menu.

   This will start the embedded OC4J server, start your default browser, and display the Web Clipping Portlet. On the resulting page, you can select a Web page that you want to expose in your WebCenter application. You can then use Web Clipping Studio to select a section of the Web page for inclusion.

17.2.3 Selecting a Section of a Web Page to Display in the Web Clipping Portlet

To select a section of a Web page to display in the Web Clipping portlet, you use the Web Clipping Studio. Using the Web Clipping Studio, you can do the following:

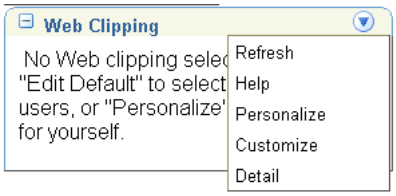
- Browse for Web content
- Section the chosen target page
- Choose the exact portion of the Web content to clip
- Preview the clipped content as a portlet
- Save the clipped content as a portlet
- Set portlet properties and save the updated portlet information

To select a section of a Web page to display in the Web Clipping portlet, perform the following steps:

1. Click the **Actions** icon on the header of the Web Clipping portlet, and select either of the following:
   - **Customize**, to select a Web page that can be used by all users.
   - **Personalize**, to enable end users to personalize their own view of the content in a Web Clipping portlet.

Note: When running a portlet that has an Edit mode, the Personalize option in the portlet's menu only appears to authenticated users of the application. Anonymous or public users will not see the option to personalize the portlet through Edit mode. Therefore, you must have implemented some form of security for your application in order for users to personalize their portlets. If you are a developer creating portlets and pages, then you may want to quickly test the Edit mode of your portlet without creating a complete security model for your application. See Section 10.6, "Configuring Basic Authentication for Testing Portlet Personalization" for an explanation of how you can quickly add the necessary security for testing portlet personalization.

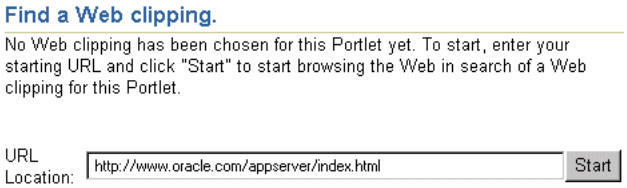
Figure 17–1 Editing Default Settings



The Find a Web clipping page is displayed.

2. In the **URL Location** field, enter the location of the starting Web page that links to the content you want to clip, as shown in Figure 17–2.

Figure 17–2 Specifying a URL



3. Click **Start**.

The Web Clipping Studio displays the page you specified, as shown in Figure 17–3.

Figure 17–3 Browsing to a Page Containing Content for a Web Clipping

4. Browse to the page that contains the content you want to clip.

   As you click hyperlinks in the Web page, your navigation links are recorded.

   Note: Any browsing operations that do not contribute to the eventual Web clipping will be discarded. Only the significant browsing operations are recorded for later playback during the show mode; any discarded links are not visited.

   For any Web sites that require HTTP Basic or Digest Authentication, a form is displayed that requests user name and password information. This encoded authentication information is recorded as part of the browsing information.

   ---

5. Once you display the page that contains the content you want to clip, in the Web Clipping Studio banner, click **Section**, as shown in Figure 17–4.

Figure 17–4 Sectioning the Target Web Page

Sectioning divides the target Web page into its clippable sections, as shown in Figure 17–5. After you click **Section**, you are no longer able to browse links in the displayed page. If you want to continue navigation, then click **Unsection** in the Web Clipping Studio banner.

Figure 17–5 Sectioned Target Web Page

6. At the top-left corner of the section of the Web content you want to clip, click **Choose.**

   You can choose only one section as a clipping at a time.

   > **Note:** To increase the number of sections available from which to choose, click **Section Smaller** in the Web Clipping Studio banner. For example, you would click **Section Smaller** to drill down one level of nested tables. To decrease the number of sections available from which to choose, click **Section Larger.**

7. Web Clipping Studio displays a preview of your chosen section. If it is the section you want, then click **Select** in the Web Clipping Studio banner. The Web Clipping Studio displays the Find a Web clipping page, with the properties of the clipping.

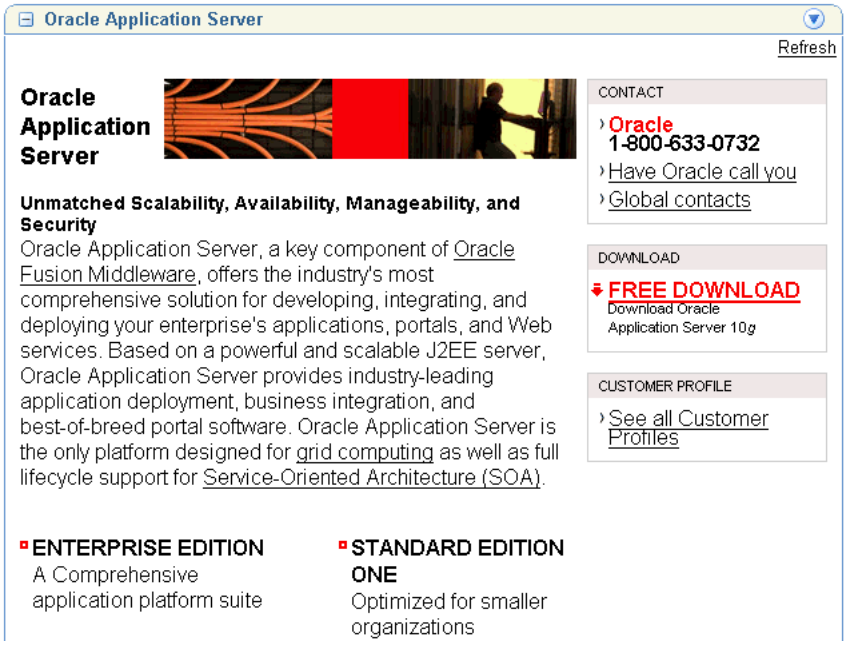
   If you do not want to use the section you clipped in your portlet, then click **Unselect** to return to the page containing the section. You can choose another section on the page, or click **Unsection** to navigate to another page.

   Some sections may contain no data, only whitespace. For example, a Web page may contain an HTML <DIV> tag that contains no text or images. If you click **Choose** on a section that contains no data, then Web Clipping displays a preview, but the preview correctly shows only whitespace. In this case, click **Unselect** in the preview page to return to the sectioned page. Then, select a section containing data.

8. In the Find a Web clipping page, click **OK** to display the selected Web Clipping in the Web Clipping portlet on your page. (You can edit default properties in the page. See Section 17.2.4, "Setting Web Clipping Portlet Properties" for more information.)

   Figure 17–6 shows the content added to the Web Clipping portlet.

Figure 17–6 Clipped Content Added to the Web Clipping Portlet on a WebCenter Application Page



Note: The **Refresh** link in the Web Clipping Portlet retrieves fresh data from the originating Web site. This link reloads the portlet, but it may retrieve the data from the cache, depending upon the settings for expiration.

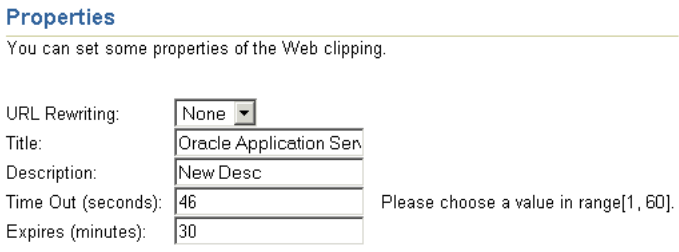
17.2.4 Setting Web Clipping Portlet Properties

You can edit various portlet settings to change the appearance of the Web Clipping portlet and to specify how end users can interact with the portlet.

To set Web Clipping portlet properties, perform the following steps:

1. Click the **Actions** icon on the header of the Web Clipping portlet, and select **Customize**. Web Clipping Studio displays the Find a Web clipping page with a Properties section, as shown in Figure 17–7.

Figure 17–7 Properties Section of Find a Web Clipping Page



2. From the **URL Rewriting** list in the **Properties** section, choose **Inline** if you want link targets to be displayed inside the portlet, or choose **None** if you want link targets to replace the current page in the browser.

> **Note:** If you have integrated with an external application or are logged into the clipped site, and if you choose **Inline** for URL Rewriting, then the session is maintained to the clipped site while browsing.

3. In the **Title** field, enter a title to display in the portlet banner.
4. In the **Description** field, enter a description of the portlet.
5. In the **Time Out (seconds)** field, enter the amount of time (in seconds) for the Web Clipping producer to attempt to contact the Web page from which the content was clipped.
6. In the **Expires (minutes)** field, enter the amount of time (in minutes) that cached content is valid. Any requests for portlet content that occur within the time period you specify will be satisfied from the cache.

   Once the cache period is exceeded, requests for portlet content will be satisfied by retrieving content from the portlet's Web Clipping data source. The cache will also be refreshed with this content.
7. If you entered any information in a form while clipping content for the Web Clipping portlet, then the **Parameterize Inputs** section is available. Select the **Click to start parameterizing** check box to customize parameters associated with the Web Clipping portlet content. Then perform the following steps:
   a. From the **Parameters** list, choose the parameters that you want to customize.
   b. From the **Personalizable** list, select a parameter if you want to enable end users to provide their own values for the parameters when they personalize the portlet. Select **None** if you do not want to enable this.
   c. In the **Display Name** field, enter a name to be displayed for the parameter.
   d. In the **Default Value** field, enter a value to use by default for the parameter.

   Section 17.4.2, "Personalizing a Web Clipping Portlet" provides an example of personalizing parameters.
8. Click **OK**.

17.3 Integrating Authenticated Web Content Using Single Sign-On

This section walks you through an example that demonstrates how you can leverage Oracle Single Sign-On to integrate content from external Web sites that require authentication into a Web Clipping portlet.

The example incorporates a secured page from Oracle Metalink (an external application) into a Web Clipping portlet.

To integrate an external application, perform the following steps:

1. Register the external application in Oracle JDeveloper, specifying the authentication information by performing the following steps.
   a. Start Oracle JDeveloper.
   b. In the Applications Navigator, right-click your project and select **New**.
   c. In the New Gallery dialog box, under the General category, select **External Application**.

d. From the list of items displayed, select **External Application**, and click **OK**.

e. In the Register External Application wizard, click **Next** on the welcome screen.

f. On step 1 of the wizard, enter a name for the application, for example, `Metalink`.

g. On step 2 of the wizard, enter the following details:

   – For Login URL, enter the URL to log in to the application, for example, `http://metalink.oracle.com/metalink/plsql/sit_main.showSitemap?p_showTitle=0`. To determine the URL, navigate to the desired application in a browser and note the URL.

     For Form-based Authorization, view the source of the login page for the external application and note the URL to be accessed during the login action.

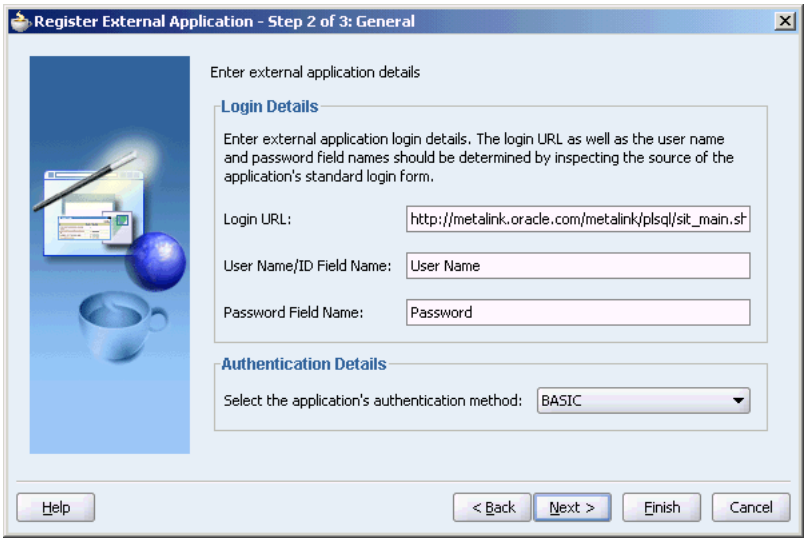
   – For User Name/ID FieldName, enter the field name that the external application uses for the user name. Determine the field name by viewing the source for the desired page. If the Authentication method uses Basic Authentication, then you do not need to enter a field name. For Metalink, you do not need to enter anything in this field

   – For Password FieldName, enter the field name that the external application uses for the password. Determine the field name by viewing the source for the desired page. If the Authentication method uses Basic Authentication, then you do not need to enter a field name. For Metalink, you do not need to enter anything in this field.

   – Select **BASIC** as the authentication method.

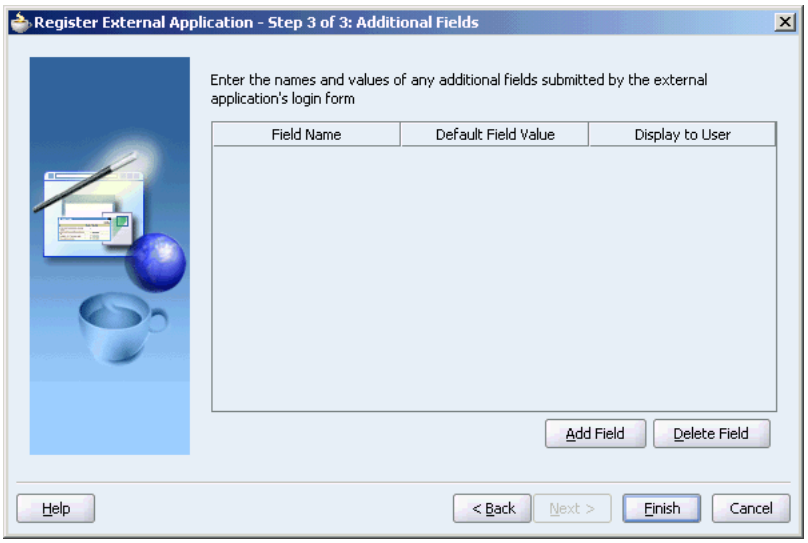
     Figure 17–8 shows step 2 of the Register External Application wizard.

Figure 17–8 Registering an External Application



h. On step 3 of the wizard, you can enter names and values of any additional fields that are submitted with the login form of the external application. To specify a field name that is used to indicate a redirection URL, enter `redirectFieldName` for Field Name. For this example, you do not need to enter additional fields. Figure 17–9 shows step 3 of the Register External Application wizard.

Figure 17–9 Specifying Redirection



i. Click **Finish**.

2. Create a credential provisioning page that will store the credentials for the external application. To do this, perform the following steps:

   a. In the Applications Navigator, right-click your project and select **New**.

   b. In the New Gallery dialog box, under the General category, select **External Application**.

   c. From the list of items displayed, select **Credential Provisioning Page**, and click **OK**.

   d. This creates the `CredentialProvisioner.jspx` page, which stores the external application login information. This information is used while implementing external application-related portlets.

3. For the Web Clipping portlet, create a new Web Clipping producer by performing the following steps:

   a. In the Applications Navigator, right-click your project and select **New**.

   b. In the New Gallery dialog box, under the Web Tier category, select **Portlets**.

   c. From the list of items displayed, select **Oracle PDK-Java Producer Registration**.

   d. On step 1 of the Register Oracle PDK Portlet Producer wizard, enter `webClippingMetalink` for the Name.

   e. Click **Next.**

   f. On step 2 of the wizard, do the following:

      – Specify the URL for the Web Clipping producer in the following format:

        `http://`*`host:port`*`/portalTools/webClipping/providers/webClipping`

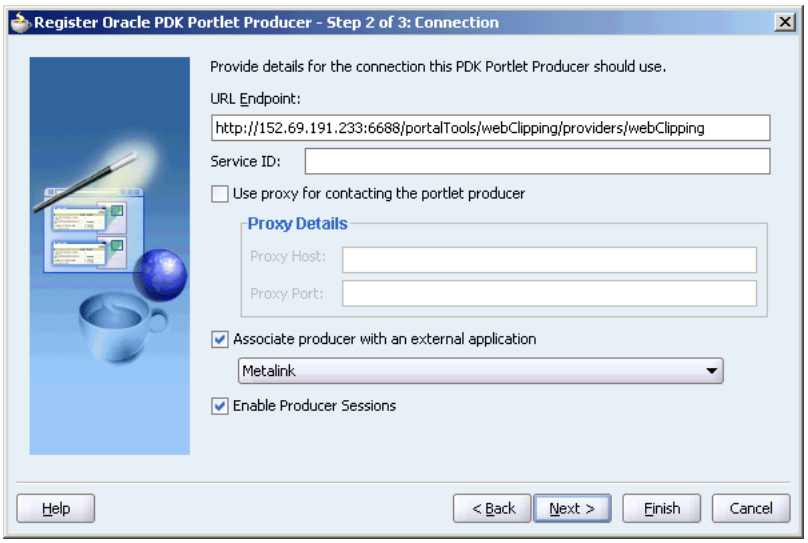
        Note that *`host:port`* refers to the host and port where the producers are located.

      – If you use a proxy server to contact the Web producers from your application, then enter the proxy details.

– Click to select the **Associate producer with an external application** option, and from the list of values, select **Metalink** that you created earlier. The Enable Producer Sessions option also gets selected in this step.

Figure 17–10 shows step 2 of the Register Oracle PDK Portlet Producer wizard.

Figure 17–10 Specifying an External Application for a Web Clipping Producer



g. On step 3 of the wizard, specify the execution timeout, subscriber ID, and shared key values, if required.

h. Click **Finish**.

i. In the registration confirmation dialog box, click **OK**.

4. Add a portlet to a `*.jspx` page, using the `webClippingMetalink` producer that you just created. Section 17.2.2, "Adding a Web Clipping Portlet to a Page" describes in detail how to add a portlet.

5. Run the `*.jspx` page.

6. If you have not entered your credentials for the External Application representing Metalink, then the portlet will contain an **Update login information** link. Click the link and enter your credentials. Then, click **OK.**

7. Select a section of a page to display in the Web Clipping portlet, by performing the following steps:

a. Click the **Actions** icon on the header of the Web Clipping portlet, and select **Customize**.

The Find a Web Clipping page is displayed.

b. In the **URL Location** field, the default URL for the External Application is displayed.

c. Click **Start.** The Web Clipping Studio displays the page from the integrated external application.

d. Browse to the page that contains the content you want to clip. After you display the page that contains the content you want to clip, click **Section** in the

Web Clipping Studio banner. Figure 17–11 shows the external application displayed in Web Clipping Studio.

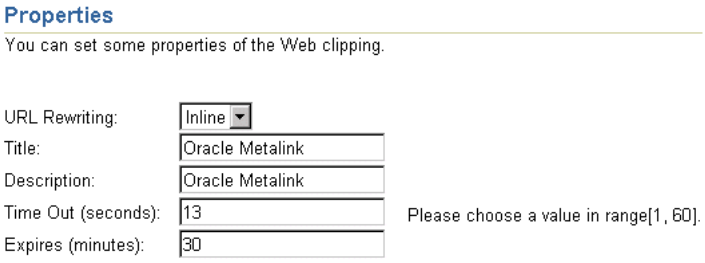
Figure 17–11 External Application in Web Clipping Studio

e. At the top-left corner of the section of the Web content you want to clip, click **Choose.**

f. Web Clipping Studio displays a preview of your chosen section. If it is the section you want, then click **Select** in the Web Clipping Studio banner.

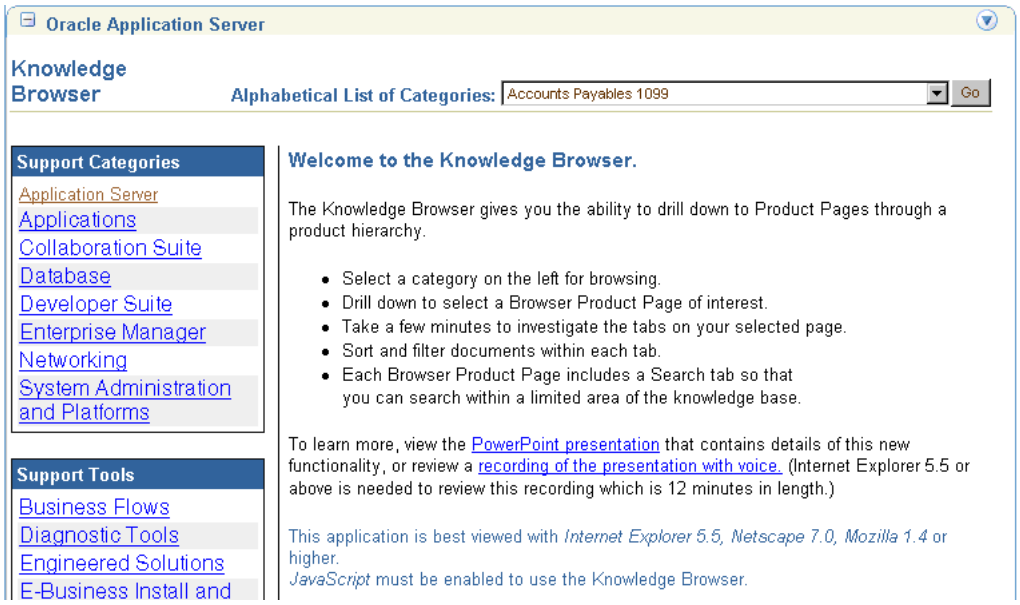
The Web Clipping Studio displays the Find a Web Clipping page, with the properties of the clipping as shown in Figure 17–12.

Figure 17–12 Properties of the External Application



g. In the Find a Web Clipping page, from the **URL Rewriting** list, choose **Inline** to specify that you want link targets displayed inside the portlet, rather than in a new browser window. Click **OK** to display the selected Web clipping in the Web Clipping portlet on your page, as shown in Figure 17–13.

Figure 17–13 External Application Displayed in Portlet



Now, the Web clipping, even though it is from a page requiring authentication, is available in your portlet.

Note that you can associate only one external application with a producer. For each external application, you must register a new producer. Each WebCenter application user accesses the authenticated content using their user name and password for that system, not the page designer's credentials.

Accessing External Application Images without Authentication

If you have integrated content from an external application that requires authentication into a Web Clipping portlet, and this content contains URLs that are links to images in the external application, then to ensure that the images are rendered without having to authenticate the browser with the external application, you must rewrite the URLs to use resource proxy. To do this, set the `rewriteImageLink` element in the Web Clipping producer's `web.xml` file to `true` as shown in the following example:

```
<!-- Rewrite the Image links to use Resource Proxy -->
<env-entry>
   <env-entry-name>oracle/webclipping/rewriteImageLink</env-entry-name>
   <env-entry-type>java.lang.Boolean</env-entry-type>
   <env-entry-value>true</env-entry-value>
</env-entry>
```

17.4 Adding a Web Clipping That Users Can Personalize

This section walks you through an example that demonstrates how you can enable end users to personalize their own view of the content in a Web Clipping portlet.

In the example, you perform the following tasks:

- Selecting a Clipping in OTN
- Personalizing a Web Clipping Portlet

17.4.1 Selecting a Clipping in OTN

In this task, you navigate to the Oracle Technology Network (OTN) and search for specific information, then select the results as the clipping for your portlet. To do this, perform the following steps:

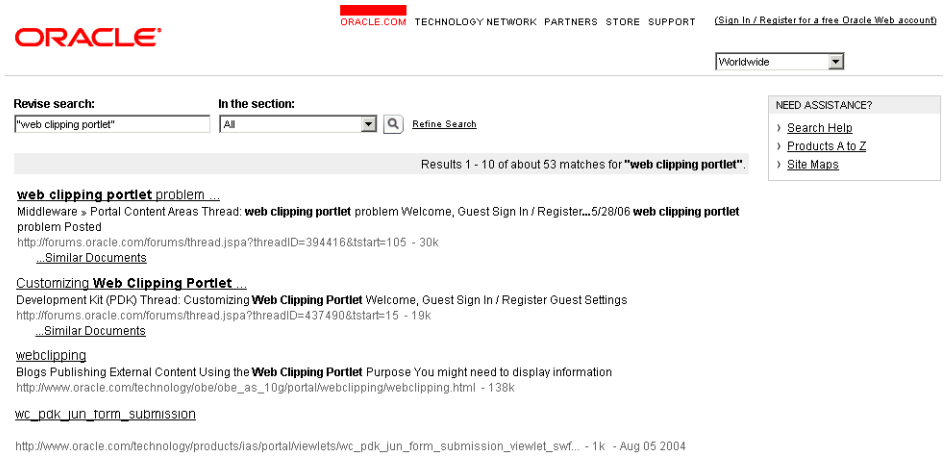
1. Click the **Actions** icon on the header of the Web Clipping portlet, and select **Customize**.
2. In the Web Clipping Studio's Find a Web clipping page, in the **URL Location** field, enter:

   `http://www.oracle.com/technology/products/ias/portal/index.html`

   Click **Start**. OTN displays the Portal Center page.
3. Enter a search string in the **Search** field at the top of the page. For this example, enter `"web clipping portlet"` (including the quotation marks), then click the **Search** icon.

   The Search result is displayed in the Web Clipping Studio, as shown in Figure 17–14.

Figure 17–14 Searching for information about OTN



4. Click **Section**. Web Clipping Studio divides the target Web page into its clippable sections, as shown in Figure 17–15.

Figure 17–15 Sectioning the Target Web Page

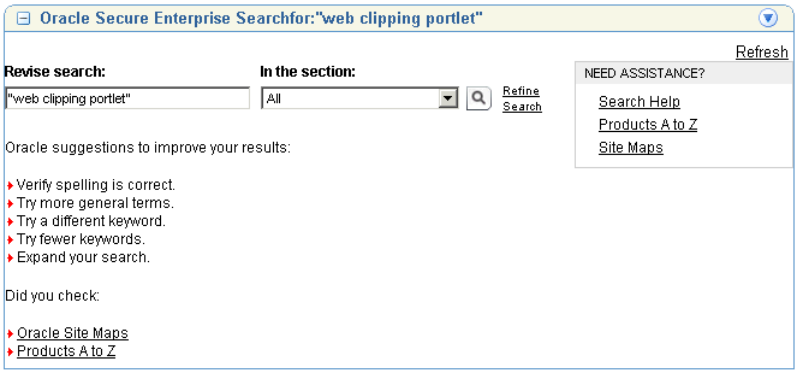
5. At the top-left corner of the search result, click **Choose**.

A preview of the search result section displays.

Some sections may contain no data, only whitespace. For example, a Web page may contain an HTML <DIV> tag that contains no text or images. If you click **Choose** on a section that contains no data, then Web Clipping displays a preview, but the preview correctly shows only whitespace. In this case, click **Unselect** in the preview page to return to the sectioned page. Then, select a section containing data.

6. Click **Select** to confirm that the search result section is the one you want to clip.
7. In the Find a Web Clipping page, click **OK** to display the selected Web Clipping in the Web Clipping portlet on your page. Figure 17–16 shows the Web Clipping displayed in the page.

Figure 17–16 Selected Web Clipping Displayed in Web Clipping Portlet



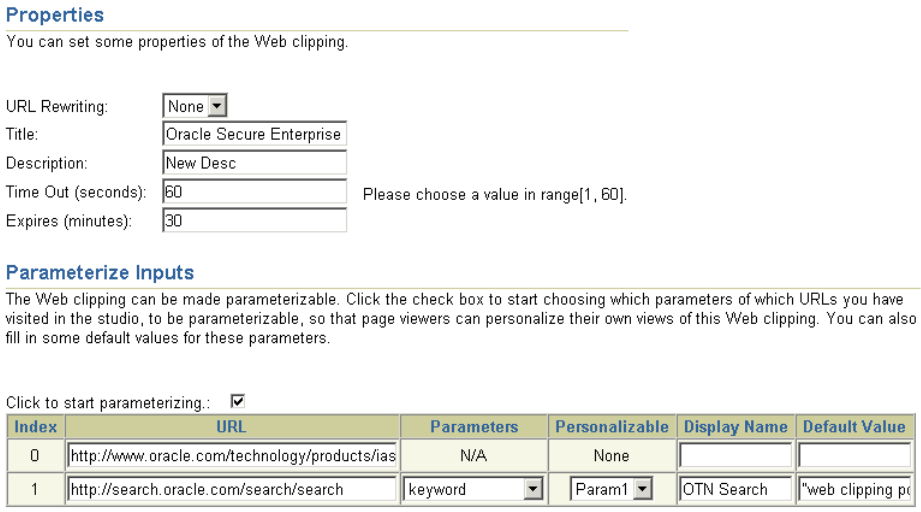
17.4.2 Personalizing a Web Clipping Portlet

In this task, you edit the properties of the Web Clipping portlet to enable end users to display different search results in the portlet. To do this, perform the following tasks:

1. On the Web Clipping portlet you just added, click the **Actions** icon on the header of the Web Clipping portlet, and select **Customize**.
2. In the Find a Web Clipping page, modify the following items in the **Properties** section:
   - From the **URL Rewriting** list, choose **Inline** to specify that you want link targets displayed inside the portlet, rather than in a new browser window.
   - In the **Title** field, enter `OTN Search`. This title displays in the header of your Web Clipping portlet, as well as the pages where users can personalize parameters for the Web clipping.

   Figure 17–17 shows the **Properties** and **Parameterize Inputs** sections of the Find a Web Clipping page.

Figure 17–17 Setting Properties for a Web Clipping

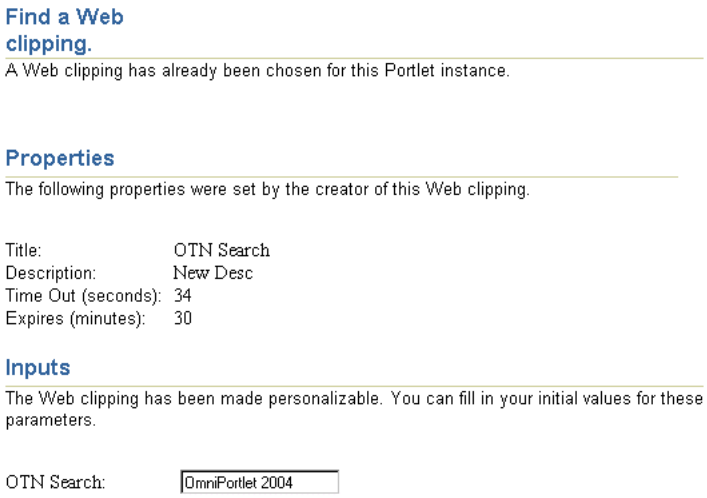


3. Because the content displayed in the portlet was reached by entering information in the **Search** field on OTN, you can customize the parameters used by the search to enable end users to specify their own search string.
4. Under the Parameterize Inputs section, select **Click to start parameterizing**, and make the following changes in the parameters table:
   - In the **Parameters** column, choose **keyword** from the list.
   - In the **Personalizable** column, choose **Param1** from the list.
   - In the **Display Name** column, enter `OTN Search`.
   - Make sure that **Default Value** displays **"web clipping portlet"** to be sure you have selected the right parameter.
5. Click **OK** to display the default search results in the Web Clipping portlet on your page.
6. Click the **Actions** icon on the header of the Web Clipping portlet, and select **Personalize**.

> **Note:** When running a portlet that has an Edit mode, the Personalize option in the portlet's menu only appears to authenticated users of the application. Anonymous or public users will not see the option to personalize the portlet through Edit mode. Therefore, you must have implemented some form of security for your application in order for users to personalize their portlets. If you are a developer creating portlets and pages, then you may want to quickly test the Edit mode of your portlet without creating a complete security model for your application. See Section 10.6, "Configuring Basic Authentication for Testing Portlet Personalization" for an explanation of how you can quickly add the necessary security for testing portlet personalization.

7. In the page that displays, scroll down to the **Inputs** section. Notice that the parameter field for the search string is labeled **OTN Search,** as you specified for the Display Name for this parameter. In the **OTN Search** field, enter a different search string. For example, enter `OmniPortlet 2004`, as shown in Figure 17–18.

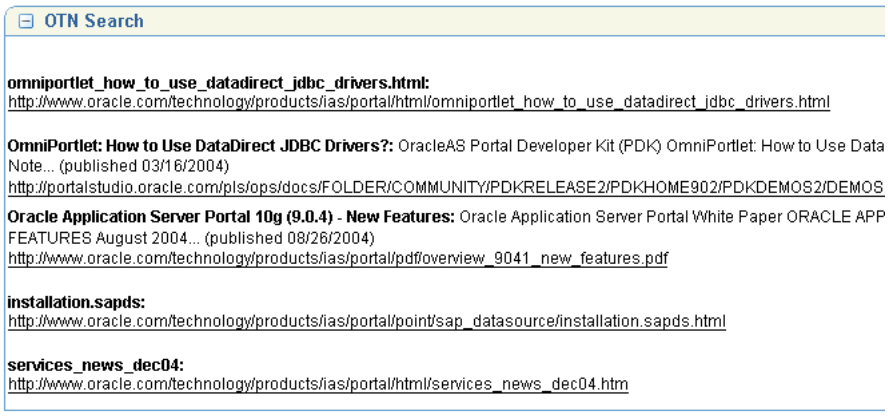
Figure 17–18 Specifying Input for Parameters



8. Click **OK.**

   The Web Clipping portlet now displays the results of performing a search on OTN for OmniPortlet 2004 information, as shown in Figure 17–19.

Figure 17–19 New Web Clipping Result Based on Customer Input Parameter



17.5 Current Limitations for Web Clipping

This section describes current limitations for Web Clipping. For information about the latest limitations in this release, be sure to read the "Oracle Oracle WebCenter Framework" chapter in the *Oracle Application Server Release Notes*. Following are the limitations:

- If the site to which you are connecting uses a large amount of JavaScript to manipulate cookies or uses the JavaScript method `document.write` to modify the HTML document being written, then you may not be able to clip content from the site.
- When you integrate with partner applications (through the use of mod_osso), you cannot clip directly through those partner applications in an authenticated manner. However, you can use the partner applications through the external application framework.

- You cannot use the Web Clipping portlet to clip Oracle Application Server Portal (OracleAS Portal) pages. As a workaround, reregister the same producer in the destination portal and edit the portal manually.
- Note the following about Web Clipping and the use of cascading style sheets (CSS):
  - If a Web page contains more than one portlet that uses a CSS, then they should not conflict if the CSS uses distinct style names, such as OraRef, to specify a style within an HTML tag, rather than using an HTML tag name, such as <A>, as the name of the style.
  - If one portlet uses a CSS, and that CSS overwrites the behavior of HTML tags by using the name of the tag, such as <A>, as the name of the style, and a second portlet on the same page does not use a CSS, the second portlet will be affected by the style instructions of the CSS of the first portlet.
  - If two portlets on the same page use a different CSS and each CSS overwrites the behavior of HTML tags by using the name of an HTML tag, such as <A>, as the name of the style, then the style that will be displayed depends on the browser.
- Web Clipping checks for Globalization Support settings in the following way:
  1. Web Clipping checks the `Content-Type` in the HTTP header for the charset attribute. If this is present, then it assumes that this is the character encoding of the HTML page.
  2. If the charset attribute is not present, then it checks the HTML `META` tag on the page to determine the character encoding.
  3. If the HTML `META` tag is not found, then Web Clipping uses the charset in the previous browsed page. If this is the first page, then it defaults to the ISO-8859-1 character encoding.
  4. If the value of the charset for `Content-Type` or `META` tag is not supported (for example, if the charset was specified as NONE), then Web clipping uses the default character set, ISO-8859-1, not the charset in the previously browsed page.
- To use the Web Clipping portlet, you must use Netscape 7.0 or later, or Microsoft Internet Explorer 5.5 or later for Windows 2000, or Microsoft Internet Explorer 6.0 or later for Windows XP.

  If you use browser versions older than these, then you may encounter JavaScript errors.

For troubleshooting information, see Appendix G, "Troubleshooting WebCenter Applications".

17.6 Summary

In this chapter, you learned how to use Web Clipping to add Web content to a page, including adding authenticated content and personalized content to a page. For more information about using Web Clipping, click the **Help** link on any of the Web Clipping pages.

18

Creating Java Portlets

This chapter explains how to create Java portlets based on the Java Portlet Specification (JSR 168) or the Oracle Application Server Portal Developer Kit-Java (PDK-Java) using the JSR 168 Java Portlet Wizard and Java Portlet Wizard in Oracle JDeveloper. This chapter includes the following sections:

The source code for many of the examples referenced in this chapter is available as part of the Portlet Developer's Kit (PDK). You can download the PDK from it's page on Oracle Technology Network (OTN):

`http://www.oracle.com/technology/products/ias/portal/pdk.html`

When you unzip PDK-Java, you will find the examples in a zip file:

`../pdk/jpdk/v2/src.zip`

To access the JavaDoc reference for PDK-Java, extract `jpdk.war` from inside of:

`../pdk/jpdk/v2/jpdk.ear`

Then unzip `jpdk.war`. The JavaDoc is located in a folder called `apidoc`.

18.1 Guidelines for Creating Java Portlets

When you write your portlets in Java for either the Java Portlet Specification (JPS) or PDK-Java, you should follow the best practices described in this section, which are as follows:

- Section 18.1.2, "Guidelines for Navigation within a Portlet"
- Section 18.1.3, "Guidelines for JavaScript"

18.1.1 Guidelines for Portlet Modes

Portlet mode exhibits the run time portlet functionality seen by users. JPS offers some modes not offered by PDK-Java and vice versa. If you are coding portlets to JPS, then you can declare custom portlet modes in `portlet.xml` that map to the extra modes offered by PDK-Java, or to accommodate any other functionality you may want to provide. For example, the JSR 168 Java Portlet Wizard for JPS portlets includes a custom mode called print, which you can use to provide a printer friendly version of the portlet. Defining custom modes is especially useful if the portlet must interoperate with portal implementations from other vendors.

A portlet may have the following portlet modes, each with its own visualization and behavior:

- Shared Screen Mode (View Mode for JPS)
- Edit Mode (JPS and PDK-Java)
- Edit Defaults Mode (JPS and PDK-Java)
- Preview Mode (JPS and PDK-Java)
- Full Screen Mode (PDK-Java)
- Help Mode (JPS and PDK-Java)
- About Mode (JPS and PDK-Java)

18.1.1.1 Shared Screen Mode (View Mode for JPS)

A portlet uses Shared Screen mode (known as View mode in JPS) to appear on a page with other portlets. This is the mode most people think about when they envision a portlet. When developing portlets, you must consider all of the factors that may influence the portlet's appearance on the page, such as the portlet's containing object and the other portlets with which your portlet will share the page. For example, suppose you choose to place your portlet inside of an HTML table cell. This would mean the portlet could display only content that can be rendered within a table cell. Furthermore, the actual size of the table cell will vary depending on user settings, the browser width, and the amount and style of content in the portlet.

18.1.1.1.1 HTML Guidelines for Rendering Portlets Plain HTML is the most basic way to render portlets and provides a great deal of flexibility to portlet developers. You can use almost any standard HTML paradigm, such as links, forms, images, tables, as long as it can display within an HTML table cell. Improperly written HTML may appear inconsistently across different browsers and, in the worst case, could cause parts of your page not to appear at all. Ensure that you adhere to the following rules:

- **Use standard HTML.** The official HTML specification is available from the W3C (more information available at: `http://www.w3.org/MarkUp/`).
- **Avoid unterminated and extraneous tags.** The behavior of pages with improperly terminated tags is unpredictable because it depends on what the browser chooses to do. Tools like weblint (`http://www.weblint.org/`) and HTML Tidy (`http://www.w3.org/People/Raggett/tidy/`) can help detect and fix hanging and unnecessary tags.
- **Consider restrictions imposed by container objects.** If your portlet is contained inside of an HTML element, such as a table cell, then you must ensure that your

portlet can be rendered within that container. For example, if you place a portlet in a table cell, then you could not use frames in the portlet because they do not appear when inserted in a table.

- **Keep portlet content concise.** Do not try to take full screen content and expose it through a small portlet. You will only end up with portlet content too small or cramped for smaller monitors. Full screen content is best viewed in Full Screen mode of PDK-Java.
- **Do not create fixed-width HTML tables in portlets.** You have no way to tell how wide a column your portlet will have on a user's page. If your portlet requires more room than given, then it might overlap with another portlet in certain browsers.
- **Avoid long, unbroken lines of text.** The result is similar to what happens with wide fixed-width tables. Your portlet might overlap other portlets in certain browsers.
- **Check behavior when resizing the page.** Test your portlet's behavior when the browser window is resized to ensure that it works in different browser window sizes.
- **Check behavior when the default browser font changes.** People may choose whatever font size they want and they can change it at any time. Your portlet should handle these situations gracefully.

The HTML you use also affects the perceived performance of your site. Users judge performance based on how long it takes for them to see the page they requested, and browsers require time to interpret and display HTML. Given that, you should consider the following:

- **Avoid lengthy, complex HTML.** Portlets share a page with other portlets. Thus, portlet generation times can significantly effect the overall performance of the page. If portlets must render complex HTML or wait for external resources, such as third-party applications, then it can greatly slow the rendering of the page.

18.1.1.1.2 Cascading Style Sheet Guidelines for Rendering Portlets

The fonts and colors of every portlet on a page should match the style settings chosen by the user. To accomplish this goal, these style selections are embedded automatically using a Cascading Style Sheet (CSS) on each page. The portlets access these settings for their fonts and colors, either directly or using the Application Programming Interface (API).

While different browsers have implemented varying levels of the full CSS specification, Oracle WebCenter Framework uses a very basic subset of this specification to enable for consistent fonts and colors. CSS implementation levels should not affect the consistency of your pages across browsers. Follow these guidelines for using CSS:

- **Use CSS instead of hard coding.** Hard coding fonts and colors is extremely dangerous. If you hard code fonts and colors, then your portlet may look out of place when the user changes the page style settings. Since you have no way of knowing the user's font and color preference choices, you might also choose to hard code a font color that turns out to be the same as the user's chosen background color, in which case your portlet appears to be invisible to that user.
- **Use the CSS APIs to format your text.** The stylesheet definition is available at the top of pages, but you should not call it directly. Instead, use the APIs provided to format your text appropriately. This method ensures that your portlets work even if the stylesheet changes in the future.

- **Avoid using CSS for absolute positioning.** Since users can personalize their pages, you cannot guarantee that your portlet can appear in a particular spot.
- **Follow Accessibility Standards.** You should ensure that you code your style sheets according to existing accessibility standards (more information available at http://www.w3.org/TR/WCAG10-CSS-TECHS/).

18.1.1.2 Edit Mode (JPS and PDK-Java)

A portlet uses Edit mode to enable users to personalize the behavior of the portlet. Edit mode provides a list of settings that the user can change. These settings may include the title, type of content, formatting, amount of information, defaults for form elements, and anything that affects the appearance or content of the portlet.

Users typically access a portlet's Edit mode by choosing **Personalize** from the portlet's dropdown list of options. When users choose **Personalize**, a new page appears in the same browser window. The portlet typically creates a Web page representing a dialog box to choose the portlet's settings. After applying the settings, users automatically return to the original page.

18.1.1.2.1 Guidelines for Edit Mode Operations The following guidelines should govern what you expose to users in Edit mode:

- **Enable users to personalize the title of the portlet.** The same portlet may be added to the same page several times. Enabling the user to personalize the title helps alleviate confusion.
- **If using caching, invalidate the content.** If personalizations cause a change in portlet display or content, then you must ensure that the portlet content is regenerated and not returned from the cache. Otherwise, the user may see incorrect content.
- **Do not use Edit mode as an administrative tool.** Edit mode is meant to give users a way of changing the behavior of their portlets. If you need to change producer settings or do other administrative tasks, then you should create secured portlets specifically for those tasks.

18.1.1.2.2 Guidelines for Buttons in Edit Mode For consistency and user convenience, Edit mode should implement the following buttons in the following order:

- **OK** saves the user personalizations and returns the portlet to view mode.
- **Apply** saves the user personalizations and reloads the current page.
- **Cancel** returns the portlet to view mode without saving changes.

18.1.1.2.3 Guidelines for Rendering Personalization Values When you show the forms used to change personalization settings, you should default the values such that the user does not have to constantly re-enter settings. When rendering the personalization values, use the following sequence to provide consistent behavior:

1. **User preference:** Query and display this user's personalizations, if available.
2. **Instance defaults:** If no user personalizations are found, then query and display system defaults for the portlet instance. These are set in Edit Defaults mode and apply only to this portlet instance.
3. **Portlet defaults:** If no system default personalizations are found, then display general portlet defaults, which may be blank. General portlet defaults are sometimes hard coded into the portlet but should be overridden if either of the two previous conditions apply.

This logic enables the personalizations to be presented in a predictable way, consistent with the other portlets in the WebCenter application. PDK-Java makes this type of logic easy to implement.

18.1.1.3 Edit Defaults Mode (JPS and PDK-Java)

A portlet uses the Edit Defaults mode to enable administrators to customize the default behavior of a particular portlet instance. Edit Defaults mode provides a list of settings that the application developer can change. These settings may include the title, type of content, formatting, amount of information, defaults for form elements, and anything that affects the appearance or content of the portlet.

These default personalization settings can change the appearance and content of that individual portlet for all users. Because Edit Defaults mode defines the system-level defaults for what a portlet displays and how it displays it, this mode should not be used as an administrative tool or for managing other portlets.

Administrators access Edit Defaults mode, when editing a page, by choosing **Customize** from the portlet's dropdown list.

When users click the Customize icon, the portlet displays in the same browser window. The portlet typically creates a Web page representing a dialog box to personalize the portlet instance settings. After applying the settings, users are automatically returned to the original page.

18.1.1.3.1 Guideline for Edit Defaults Mode Options The following guideline should govern what you expose to page designers in Edit Defaults mode:

- **Do not use Edit Defaults mode as an administrative tool.** Edit Defaults mode gives users a way of changing the behavior of their portlets. If you need to change producer settings or do other administrative tasks, then you should create secured portlets specifically for those tasks.

18.1.1.3.2 Guidelines for Buttons in Edit Defaults Mode For consistency and user convenience, Edit Defaults mode should implement the following buttons in the following order:

- **OK** saves the user personalizations and returns the portlet to view mode.
- **Apply** saves the user personalizations and reloads the current page.
- **Cancel** returns the portlet to view mode without saving changes.

18.1.1.3.3 Guidelines for Rendering Personalization Values When you show the forms used to change personalization settings, you should default the values so that the application developer does not have to constantly re-enter settings. When rendering personalization values, use the following sequence to provide consistent behavior:

1. **Instance preferences:** Query and display system defaults for the portlet instance.
2. **Portlet defaults**: If no system default personalizations are found, then display general portlet defaults, which may be blank. General portlet defaults are sometimes hard coded into the portlet but should be overridden by system defaults.

This logic enables the personalizations to be presented in a predictable way, consistent with the other portlets in the WebCenter application.

18.1.1.4 Preview Mode (JPS and PDK-Java)

A portlet uses Preview mode to show the user how the portlet looks before adding it to a page. Preview mode visually represents what the portlet can do. Not all portlet consumers will call this mode. For example, Oracle Application Server Portal (OracleAS Portal) makes use of this mode but Oracle WebCenter Framework does not.

OracleAS Portal calls it when the user clicks the Preview icon from the Add Portlet page. A window then displays the preview of the chosen portlet. The user has the option to add that portlet to the page.

Note: This mode has no particular application in WebCenter applications, but used in OracleAS Portal's Portlet Repository, where it renders as a magnifying glass icon, which users click to preview a portlet.

Guidelines for Preview Mode

The following guidelines should govern what you expose to users in Preview mode:

- **Provide an idea of what the portlet does.** Preview mode should generate enough content for the user to get an idea of the actual content and functionality of the portlet.
- **Keep your portlet previews small.** The amount of data produced in this mode should not exceed a few lines of HTML or a screen shot. Preview mode appears in a small area, and exceeding the window's size looks unprofessional and forces users to scroll.
- **Do not use live hyperlinks.** Links may not work as expected when rendered in Preview mode. Hyperlinks can be simulated using the underline font.
- **Do not use active form buttons.** Forms may not work as you expect them to when rendered in Preview mode. If you decide to render form elements, then do not link them to anything.

18.1.1.5 Full Screen Mode (PDK-Java)

Portlets use Full Screen mode to provide a larger version of the portlet for displaying additional details. Full Screen mode lets a portlet have the entire window to itself. Not all portlet consumers will call this mode. For example, OracleAS Portal makes use of this mode but Oracle WebCenter Framework does not. In OracleAS Portal, users access a portlet's Full Screen mode by clicking the title of the portlet.

For example, if a portlet displays expense information, then it could show a summary of the top ten spenders in Shared Screen mode and the spending totals for everyone in Full Screen mode. Portlets can also provide a shortcut to Web applications. If a portlet provided an interface to submitting receipts for expenses in Shared Screen mode, then it could link to the entire expense application from Full Screen mode.

Technically, JPS portlets do not have Full Screen mode. However, you can implement the equivalent of Full Screen mode for a JPS portlet with View mode (Shared Screen mode) and a maximized state for the window.

18.1.1.6 Help Mode (JPS and PDK-Java)

A portlet uses Help mode to display information about the functionality of the portlet and how to use it. The user should be able to find useful information about the portlet, its content, and its capabilities with this mode.

Users access a portlet's Help mode by choosing the Help action in the portlet.

Guideline for Help Mode

The following guideline should govern what you expose to users in Help mode:

- **Describe how to use the portlet.** Users may not know all the features your portlet provides just from its interface. Describe the features and how to get the most out of them.

18.1.1.7 About Mode (JPS and PDK-Java)

Users should be able to see what version of the portlet is currently running, its publication and copyright information, and how to contact the author. Portlets that require registration may link to Web-based applications or contact information from this mode, as well.

Users access a portlet's About mode by choosing **About** from the dropdown list in the portlet's chrome. A new page appears in the same browser window. The portlet can either generate the content for this new page or take the user to an existing page or application.

Guideline for About Mode

The following guideline should govern what you expose to users in About mode:

- **Display relevant copyright, version, and author information.** Users want to know what portlet they are using and where they can get more information. The about page may become important when supporting your portlets.

18.1.2 Guidelines for Navigation within a Portlet

In some ways, navigation between different sections or pages of a single portlet is identical to navigation between standard Web pages. Users can submit forms and click links. In typical, simple Web pages, both of these actions involve sending a message directly to the server responsible for rendering the new content, which is then returned to the client. In portlets, which comprise only part of a page, the form submission or link rendered within the portlet does not directly target the portlet. It passes information to the portlet through the WebCenter application. If a link or form within a portlet does not refer back to the application, then following that link takes the user away from the application, which is not typically the desired behavior.

The component developer does not need to know the detailed mechanics of how the parameters of a form or link get passed around between the user, application, and portlet. However, they must understand that they cannot write links in a portlet the same way they do for typical, simple Web pages.

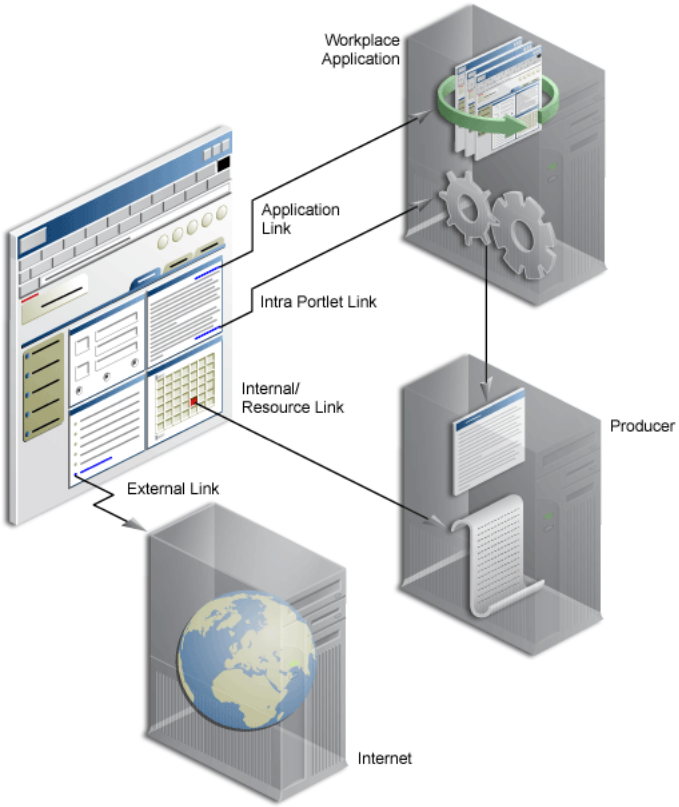
Types of Links for Portlets

A portlet may render links of four classes, as follows:

- **Intraportlet links** require the portlet to be aware of the address of the WebCenter application because they actually refer to it in some way.
- **Application links**, like intraportlet links, must be aware of the address of the WebCenter application for the same reason.
- **External links** make no reference to the WebCenter application and work in portlets as they would do in a normal Web page.
- **Internal/Resource links**, like external links, also make no reference to the WebCenter application.

Figure 18–1 contains a summary of these link types. The arrows indicate how the links reference the resources to which they logically refer.

Figure 18–1 WebCenter Application Link Types



18.1.2.1 Intraportlet Links

Intraportlet links go to different sections or pages within a given portlet. Strictly speaking, they refer to the page containing the portlet, but they contain parameters that cause the portlet to render a different section or page within that page when it is requested by the user.

As a direct consequence, a portlet cannot expect to render links to different sections or pages of itself using relative links or absolute links based on its own server context. Intraportlet link are useful for intraportlet navigation, either as links or form submission targets.

18.1.2.2 Application Links

Application links refer to significant pages within the WebCenter application, such as the user's home page.

18.1.2.3 External Links

External links refer neither to the portlet (through a page) nor to any part of the WebCenter application. If selected, these links take the user away from the application, for example, www.oracle.com.

18.1.2.4 Internal/Resource Links

Internal/Resource links refer to internal (to the portlet) resources. Sometimes they are exclusively used internally during portlet rendering, for example as a server side include. On other occasions, they may be used externally to reference portlet resources like images. In this latter case, you can use the PDK-Java `constructResourceURL` method in the `UrlUtils` class to retrieve images from behind a firewall using resource proxy. Note that in order for resource proxying to work, you must first set the JNDI variable, `oracle/portal/provider/sample/resourceUrlKey`, for the producer. For more information about setting JNDI variables, see Section 19.2.3.2, "Setting JNDI Variable Values".

For example, `lottery.jsp` of the lottery sample, which is available with PDK-Java, contains resource proxy requests for images.

```
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ page session="false" import="oracle.portal.provider.v2.render.*" %>
<%@ page import="oracle.portal.provider.v2.render.http.HttpPortletRendererUtil" %>
<%@ page import="oracle.portal.provider.v2.url.UrlUtils" %>
<%@ page import="oracle.portal.sample.v2.devguide.lottery.*" %>
<%
  LottoPicker picker = new LottoPicker();
  picker.setIdentity(request.getRemoteAddr() ); %>
<% PortletRenderRequest portletRequest = (PortletRenderRequest)
request.getAttribute("oracle.portal.PortletRenderRequest"); %>
<% String name = portletRequest.getUser().getName(); %>
<P class="PortletHeading1" ALIGN="CENTER">Hi <%= name %>, Your Specially
  Picked</P>
<P ALIGN="CENTER"><IMG SRC="<%= UrlUtils.constructResourceURL(portletRequest,
  HttpPortletRendererUtil.absoluteLink(request, "images/winningnumbers.gif")) %>"
  WIDTH="450" HEIGHT="69" ALIGN="BOTTOM" BORDER="0"></P>
<P>
<P ALIGN="CENTER">
<TABLE ALIGN="CENTER" BORDER="0" CELLPADDING="0" CELLSPACING="0">
<TR>
<%
     int [] picks = picker.getPicks();
     for (int i = 0; i < picks.length; i++) {
%>
             <TD>
    <IMG SRC="<%= UrlUtils.constructResourceURL(portletRequest,
      HttpPortletRendererUtil.absoluteLink(request, "images/ball" + picks[i])) %>
      .gif" WIDTH="68" HEIGHT="76" ALIGN="BOTTOM" BORDER="0">
     </TD>
<%
     }
```

For session-based producers, any cookies returned from the original `initSession` call to the producer are sent with the request back to the producer to maintain the right session context.

18.1.3 Guidelines for JavaScript

JavaScript can often be useful within a portlet, but bear in mind the following guidelines within your portlets:

- Portlets with JavaScript should be rendered in Inline Frames (IFRAMEs). You can do this by setting `RenderPortletInFrame` to true. For more information about the `RenderPortletInFrame` attribute and how to set it, see Section 4.3.3, "Setting Attribute Values for the adfp:portlet Tag".

- You should never use JavaScript to redirect the page in which the portlet is rendered. If you need to direct users elsewhere, then you should do so in your portlet action handling code or open a new window in the browser.
- Ensure that identifiers in your JavaScript are qualified.By qualifying your identifiers, you ensure that they are unique and do not clash with any JavaScript on the page.

18.2 Introduction to Java Portlet Specification (JPS) and WSRP

Organizations engaged in WebCenter application projects have found application integration to be a major issue. Until now, users developed portlets using proprietary APIs for a single portal platform and often faced a shortage of available portlets from a particular portal vendor. All this changes with the introduction of the following standards:

- Web Services for Remote Portlets (WSRP)
- Java Portlet Specification (JPS)[1] based on JSR 168

These two standards enable the development of portlets that interoperate with different portal products, and therefore widen the availability of portlets within an organization. This wider availability can, in turn, dramatically increase an organization's productivity when building WebCenter applications.

WSRP is a Web services standard that enables the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container and any WSRP portal. WSRP defines the following:

- Web Services Definition Language (WSDL) interface for the invocation of WSRP services
- Markup fragment rules for markup emitted by WSRP services
- The method to publish, find, and bind WSRP services and metadata

JPS is a specification that defines a set of APIs to enable interoperability between portlets and portals, addressing the areas of aggregation, personalization, presentation, and security. JPS defines container services which provide the following:

- A portlet API for coding portlet functionality
- The URL-rewriting mechanism for creating user interaction within a portlet container
- The security and personalization of portlets

Oracle actively participates in the WSRP committee and is also a member of the expert group for JPS.

[1] The Java Portlet Specification 1.0 arose from Java Specification Request 168 and the JSR168 Expert Group.

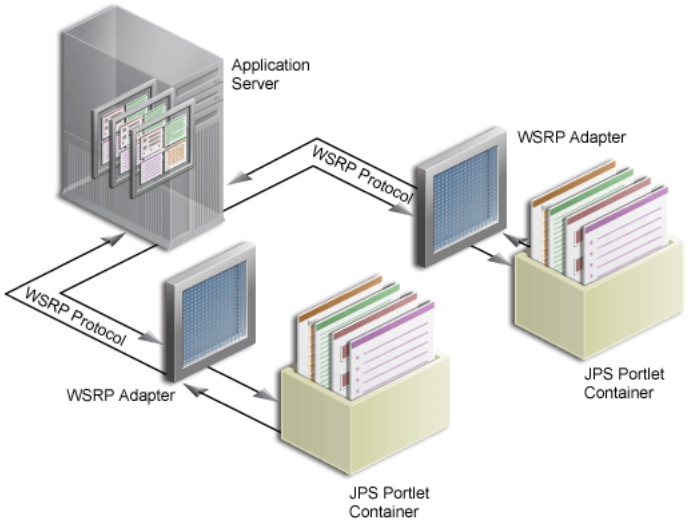
Note: .HTML forms can be submitted using either the `get` or `post` method, but the WSRP standard only requires the consumer (WebCenter application) to use the `post` method. Support of the `get` method is optional according to the standard. Since application consumers are not required to support the `get` method, Oracle recommends that you use the `post` method when developing your portlets.

The Relationship Between WSRP and JPS

WSRP is a communication protocol between WebCenter application servers and portlet containers, while JPS describes the Java Portlet API for building portlets. Combining these standards enables developers to integrate their applications from any internal or external source as portlets with WSRP portals. Building pages becomes as simple as selecting portlets from the Oracle JDeveloper Component Palette. Figure 18–2 shows the architecture of the WSRP specification.

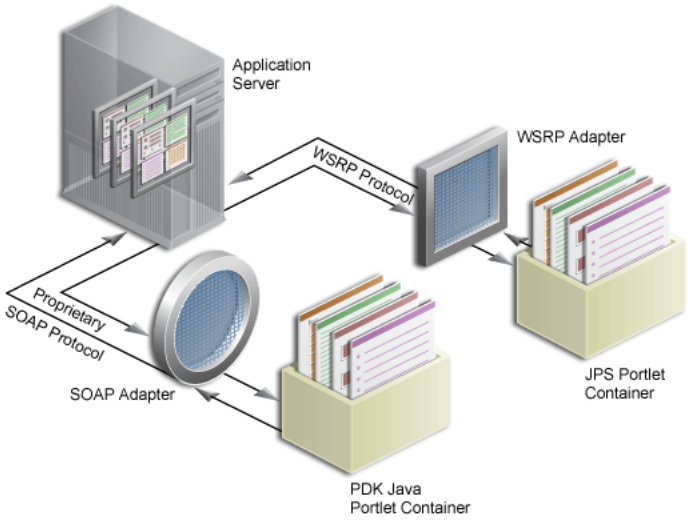
Note: Figure 18–2 illustrates the use of JPS portlets with WSRP, but it should be noted that WSRP can also work with non-JPS portlets.

Figure 18–2 WSRP Specification Architecture



Oracle WebCenter Framework is able to support communication between the WebCenter application and both the new Java Portlet APIs as well as our existing APIs (PDK-Java). Figure 18–3 shows the architecture of the WSRP support. Notice that the JPS-compliant portlet container uses the WSRP protocol for communication and the PDK-Java portlet container uses Oracle's proprietary SOAP protocol for communication.

Figure 18–3 Oracle WebCenter Portlet Architecture



See Section 10.10, "Securing Identity Propagation Through WSRP Producers With WS-Security" for a description of how JSR 168 security concepts are exposed through WSRP.

18.3 Configuring Your Application Server or Standalone OC4J to Run Portlets

You can run portlets from any number of configurations, each with its own set of requirements. This section describes the configuration requirements for two of the more common portlet scenarios: configuring Oracle Containers for J2EE (OC4J) in Oracle Application Server and configuring a standalone OC4J. If you have Oracle Application Server 10.1.3.2.0, then you already have an OC4J that is preconfigured to run portlet producers (OC4J_WebCenter). If you prefer not to use OC4J_WebCenter, or if you plan to use a standalone OC4J to run your portlet producers, then follow the steps provided in this section.

Note: Oracle JDeveloper provides a preconfigured OC4J that supports PDK-Java and WSRP portlets out of the box. For information about the preconfigured OC4J, see Chapter 3, "Preparing Your Development Environment".

To configure an application server OC4J instance or a standalone OC4J to run portlets, perform the following steps:

1. For an OC4J that is part of Oracle Application Server, create a new OC4J instance into which you can deploy your portlets:
   a. Log in to the Application Server Control Console as `oc4jadmin` at:

      `http://host_name:port_number/em`

      For example:

      `http://localhost:8888/em`

   b. On the Cluster Topology page, click the link to your application server.

c. Under System Components, click the **Create OC4J Instance** button.

d. On the Create OC4J Instance page, specify a name for the OC4J instance.

For example, to create a separate instance for WSRP producers, you might name the instance `wsrp`. You can use any name you prefer.

e. Make sure the **Start this OC4J instance after creation** check box is not selected.

f. Click **Create**.

For a standalone OC4J, stop the standalone OC4J in the Application Server Control Console:

a. Log in to Application Server Control as `oc4jadmin` at:

`http://`*`host_name`*`:`*`port_number`*`/em`

For example:

`http://localhost:8888/em`

b. On the OC4J home page, click **Stop**.

2. If required, set up a preference store.

Most portlet producers use a *File* preference store by default (the Web Clipping portlet is the exception; for more information, see Section 18.4.3, "What You Should Know About the Web Clipping Portlet and a Preference Store"). The File preference store is configured and ready to use out of the box. For high availability, you can configure the preference store to use a database instead. Using a database preference store requires additional configuration. For more information, see Section 18.4, "Setting Up a Preference Store".

3. Download and install the portlet run time and the sample portlet producers.

You will find the relevant download at the following URL:

`http://www.oracle.com/technology/products/webcenter/pdk.html`

The detailed steps for installation are provided in the readme files included with the download.

4. Start OC4J by navigating to *`ORACLE_HOME`*`\bin` (where *`ORACLE_HOME`* is the source location of your OC4J) and running the following command:

```
oc4j -start
```

5. Test the configuration to confirm it is working correctly.

You can use the sample WSRP producer, `wsrp-samples.ear`, for this purpose. You will find `wsrp-samples.ear` in your portlet container directory.

For the standalone OC4J, this file is in the file you downloaded in step 3. For the Application Server OC4J you can find this file in *`ORACLE_HOME`*`\adfp\lib`.

Before proceeding, you must deploy the sample portlets EAR file.

To deploy the sample portlets EAR file, perform the following steps:

a. Return to the Application Server Control Console.

Log in to Application Server Control as `oc4jadmin` at:

`http://`*`host_name`*`:`*`port_number`*`/em`

For example:

```
http://localhost:8888/em
```

b. On Oracle Application Server, click the Oracle Application Server middle-tier instance to which you plan to deploy portlets.

c. On Oracle Application Server, click the OC4J instance that you created earlier, `wsrp`.

d. Click the **Applications** tab.

e. Click **Deploy**.

The Deploy: Select Archive page is displayed.

f. Select **Archive is present on local host. Upload the archive to the server where Application Server Control is running.**

g. In **Archive Location**, enter the path to and the filename `wsrp-samples.ear`.

h. Select **Automatically create a new deployment plan.**

i. Click **Next**.

Wait while the system uploads the EAR file. When the file is uploaded, the Deploy: Application Attributes page is displayed.

j. For **Application Name**, enter `sampleportlets`.

k. For **Context Root**:

For a standalone OC4J, enter `portletapp`.

For the Application Server instance, enter some other name that is not already in use.

l. Click **Next**.

The Deploy: Deployment Settings page is displayed.

m. Click **Deploy**.

You can alter any deployment settings on this page, but, in this case, just click Deploy.

The Deployment Confirmation page is displayed.

n. Carefully check the settings to ensure that they are correct.

o. Click **Return** to dismiss the Confirmation page.

See *Oracle Containers for J2EE Deployment Guide* for complete information about how to deploy an EAR file.

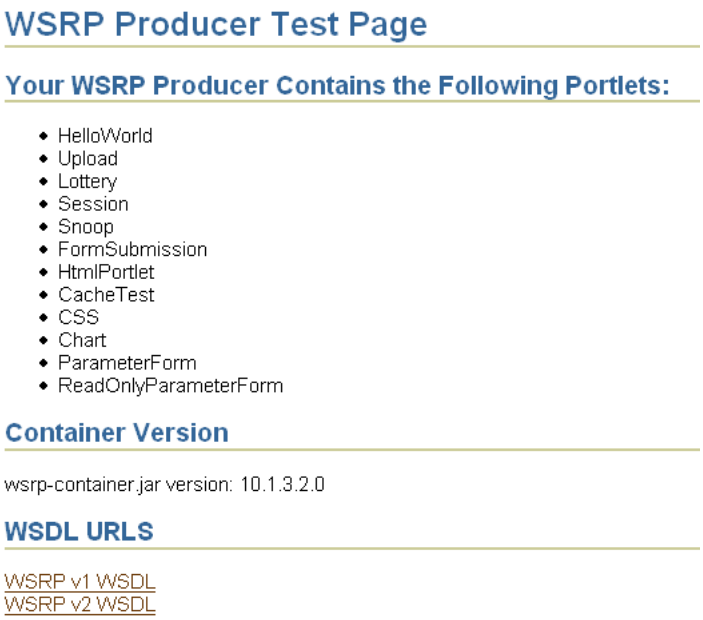
6. Once the sample EAR file is deployed, test its WSDL URL by entering it into a browser.

For the WSDL URL syntax, use the following structure:

```
http://<host>:<port>/portletapp/info
```

The Web page displays content like that depicted in Figure 18–4.

Figure 18–4 Example WSRP Producer (WSDL URL) Test Page



7. Copy the link location of one of the portlet producers listed under WSDL URLS, and register this portlet producer by following the instructions in Section 18.10, "Registering and Viewing Your Portlet".

   Further test the configuration by adding some of the sample portlets from the newly registered portlet producer to a page and displaying the page. If the registration fails for any reason or you cannot add portlets to a page, then see Appendix G, "Troubleshooting WebCenter Applications".

18.4 Setting Up a Preference Store

The portlet preference store is used for persisting consumer registration handles and portlet preference data. Portlet producers can use one of two types of preference store: *File* and *Database*. For most portlet producers, the file preference store is specified by default and is ready to use out of the box. The database preference store requires additional configuration.

In a clustered environment, Oracle recommends the use of a database preference store. If you prefer to use a file-based preference store, then you must also use a shared file system.

This section describes how to set up a database preference store and a file-based preference store. It includes the following subsections:

- Section 18.4.1, "Setting Up a Database Preference Store"
- Section 18.4.2, "Setting Up a File-Based Preference Store"
- Section 18.4.3, "What You Should Know About the Web Clipping Portlet and a Preference Store"

18.4.1 Setting Up a Database Preference Store

A database preference store is desirable in environments where high availability is key. Setting up a database preference store for a portlet producer requires a little extra configuration over what is required for a default file preference store. For example:

- A schema for storing preferences must be created in the database.
- Connection details must be mapped to the data source.
- The preference store type must be specified using the `persistentStore` JNDI environment variable (WSRP) or the `preferenceStore` tag (PDK-Java).

This section describes how to set up a database preference store for WSRP and PDK-Java producers. It contains the following subsections:

- Section 18.4.1.1, "Creating the Schema for a Database Preference Store"
- Section 18.4.1.2, "Mapping Connection Details to a JDBC Data Source"
- Section 18.4.1.3, "Setting Portlet Preference Store Variables in Producer Configuration Files"
- Section 18.4.1.4, "Database-Related Attributes and Parameters of the preferenceStore Tag"

18.4.1.1 Creating the Schema for a Database Preference Store

To use a database preference store, both WSRP and PDK-Java producers require the creation of a database schema. You can create the schema in the Oracle Application Server infrastructure database or any other Oracle database for this purpose.

The script `dbprefstore.sql` is available for creating database schemas for portlet producer database preference stores. In one operation, the script creates database schemas for WSRP and PDK-Java producers.

Note: For additional information about creating a schema for a Web Clipping producer, see Section 18.4.3, "What You Should Know About the Web Clipping Portlet and a Preference Store".

The script is named `dbprefstore.sql`, and it is located at:

ORACLE_HOME`/j2ee/home/database/wsrp/`

Note: Although the directory path for this script includes `wsrp`, it creates database objects for both WSRP and PDK-Java preference stores.

To create database schemas for WSRP and PDK-Java producer preference stores, perform the following steps:

1. Connect to SQL*Plus using the `SYS` account and `SYSDBA` database administrator role.

   At the prompt:

   a. `Enter user-name: SYS AS SYSDBA`

   b. `Enter password:` *`SYS_password`*

2. At the SQL prompt, enter the following:

```
SQL> @ORACLE_HOME\j2ee\home\database\wsrp\dbprefstore.sql
```

3. When prompted, enter a user name and password for the WSRP and PDK-Java preference store database schema.

   For example, your user name might be `portlet_prefs`.

Once the command executes, a database preference store is created and the schema is populated with the required database objects.

> **Note:** For information about migrating portlet preferences from one database preference store to another, or from a database preference store to a file preference store, see Appendix B, "Additional Portlet Configuration".

18.4.1.2 Mapping Connection Details to a JDBC Data Source

Once the portlet preference schema is created, you must tell the producers where to find the portlet preference store. You must do this twice: once for WSRP and once for PDK-Java.

Tell producers where to find the portlet preference store by mapping connection details to the JDBC data source, using the Application Server Control Console. These details are stored in the `data-sources.xml` file.

This section describes how to do this mapping for both WSRP and PDK-Java portlet producers, noting where there are slight variations.

To map connection details to a JDBC data source for WSRP and PDK-Java portlet producers, perform the following steps:

1. Log in to the Application Server Control Console.

   Typically, the URL for the Application Server Control Console is in the following format:

   `http://hostname:port/em/`

2. Navigate to the home page for the OC4J you want to configure.
3. Click the **Administration** tab.

   The Administration Tasks list is displayed.

4. Under the **Services** task, click the **Go to Task** icon for JDBC Resources.

   The JDBC Resources page is displayed.

5. In the **Connection Pools** section, click **Create**.

   The Create Connection Pool - Application page is displayed. On this page, you can specify the application to which the new connection pool is to be added. Additionally, you can specify whether you want to create a connection pool or use an existing one.

6. From the **Application** list, select **default**.
7. Click **Continue**.

   The Create Connection Pool page is displayed. Use this page to specify the details required to create a connection pool.

8. Enter values as described in Table 18–1.

Table 18–1 Create Connection Pool Settings (Database Portlet Preference Store)

Setting	Value

Name	Enter a name for this connection pool.
	For example, for a WSRP producer you might enter:
	`OracleWSRPPool`
	
	For a PDK-Java producer, you might enter:
	`OraclePDKPool`

Connection Factory Class	Accept the default value: `oracle.jdbc.pool.OracleDataSource`
JDBC URL	Enter the JDBC URL for the Oracle database that contains the schema that you created for the preference store using the following syntax: `jdbc:oracle:thin:@//`*`dbhost`*`:`*`dbport`*`/`*`service_name`* For example: `jdbc:oracle:thin:@//shobeen:1521/sales_us`
Username	Enter the user name for the database where you created the schema for the preference store.
Use Indirect Password/*Indirect Password*	For the user specified in the **Username** field, enter the indirect password for the database that contains the preference store schema. For example: *`role`*`/`*`username`* For example: `users/Scott`

9. Click **Finish**.

   The JDBC Resources page is displayed.

10. Click the **Test Connection** icon for the newly created connection.

    If the test fails, then review the settings for your pool to ensure that they are correct.

11. In the **Data Sources** section, click **Create**.

    The Create Data Source - Application & Type page is displayed. Use this page to specify the application to which the new data source is to be added and the data source type.

12. From the **Application** list, select **default**.

13. Click **Continue**.

    The Create Data Source - Managed Data Source page is displayed. Use this page to specify details for creating a managed data source.

14. Enter values as described in Table 18–2.

Table 18–2 Create Data Source Settings (WSRP Producer Preference Store)

Setting	Value

Name	Enter a name for your data source.
	For example, for a WSRP producer preference store you might enter:
	`WSRP_PREF_DS`
	For a PDK-Java preference store, you might enter:
	`PDK_PREF_DS`

JNDI Location	Enter the JNDI path as follows: `jdbc/portletPrefs`
Transaction Level	Accept the default value: `Global & Local Transactions`

Connection Pool	Choose the connection pool that you created earlier.
	For example, for a WSRP producer preference store you might enter:
	`OracleWSRPPool`
	For a PDK-Java producer preference store
	`OraclePDKPool`

Login Timeout (seconds)	Accept the default value, `0`.

15. Click **Finish**.

18.4.1.3 Setting Portlet Preference Store Variables in Producer Configuration Files

After you run the script to set up the database objects for a database preference store and map connection details to the JDBC data source, both portlet producer types require a little additional adjustment in their configuration files—`web.xml` for WSRP producers and `provider.xml` for PDK-Java producers. In these files, you must specify the preference store type and, in some cases, the preference store location.

This section describes how to accomplish such tasks. It contains the following subsections:

- Section 18.4.1.3.1, "Setting Preference Store-Related Variables for a WSRP Portlet Producer"
- Section 18.4.1.3.2, "Setting Preference Store-Related Variables for a PDK-Java Portlet Producer"

18.4.1.3.1 Setting Preference Store-Related Variables for a WSRP Portlet Producer

Once your preference store database objects are created and the portlet producer knows where to find them, you must tell the portlet producer that a database preference store will be used instead of the default file preference store. You do this by setting the JNDI variable, `persistentStore` to the value `Database`.

Set JNDI preference store variables for all WSRP producers in their associated `web.xml` files.

Example 18–1 lists sample file locations for various WSRP producers' `web.xml` files:

Example 18–1 Sample File Locations for web.xml Files

```
Rich Text Portlet Producer:
ORACLE_HOME\j2ee\OC4J_
WebCenter\applications\richtextportlet\richtextportlet\WEB-INF

WSRP Samples Producer:
ORACLE_HOME\j2ee\OC4J_WebCenter\applications\portletapp\wsrp-samples\WEB-INF
```

Table 18–3 lists and describes the JNDI variable that is relevant to a database preference store.

Table 18–3 WSRP Producer Database Preference Store-Related JNDI Variable

Variable Name	Variable Value	Description
`oracle/portal/wsrp/server/persistentStore`	`Database`	Determines which data store (File or Database) is used for persisting a portlet application's consumer registration handles and portlet preferences. Permissible values: {`File`, `Database`}

Example 18–2 illustrates database preference store set-up in a producer's `web.xml` file. If you are moving to a database preference store from a file-based preference store, then be sure to remove the `fileStoreRoot` tag from the preference store set up (see Example 18–5 for an example of the `fileStoreRoot` tag).

> **Note:** Preference-store-related variables may or may not be present in a WSRP producer's `web.xml` file. Before you enter the variables, search for them. If the variables are present, then modify them as desired. If the variables are not present, then enter them as illustrated in Example 18–2.

Example 18–2 Configuring web.xml to Use a Database Preference Store

```
<env-entry>
   <env-entry-name>oracle/portal/wsrp/server/persistentStore</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value>Database</env-entry-value>
</env-entry>
```

Once you have altered all relevant `web.xml` files, restart your OC4J instance.

18.4.1.3.2 Setting Preference Store-Related Variables for a PDK-Java Portlet Producer Once your preference store database objects are created and the PDK-Java producer knows where to find them, you must tell the PDK-Java producer that a database preference store will be used instead of the default file preference store. You do this by setting the value for the `preferenceStore` tag in each PDK-Java producer's `provider.xml` file.

Example 18–3 lists sample file locations for various PDK-Java producers' `provider.xml` files:

Example 18–3 Sample File Locations for provider.xml Files

```
Omniportlet:
ORACLE_HOME\j2ee\OC4J_
WebCenter\applications\portalTools\omniPortlet\WEB-INF\providers\omniPortlet

JPDK samples:
ORACLE_HOME\j2ee\OC4J_
WebCenter\applications\jpdk\jpdk\WEB-INF\providers\providername
```

Note: For information about configuring a repository for the Web Clipping portlet producer, see Section 18.4.3, "What You Should Know About the Web Clipping Portlet and a Preference Store".

Modify the `preferenceStore` tag to use a database preference store (Example 18–4).

Example 18–4 Configuring provider.xml to Use a Database Preference Store

```
<preferenceStore class="oracle.portal.provider.v2.preference.DBPreferenceStore">
   <name>producer_name</name>
   <connection>jdbc/PooledConnection</connection>
</preferenceStore>
```

In Example 18–4, you would substitute *producer_name* with the actual producer name (the name you enter here is your choice). Otherwise, use the example exactly as written.

Note: For a list and description of database-related attributes and parameters of the `preferenceStore` tag, see Section 18.4.1.4, "Database-Related Attributes and Parameters of the preferenceStore Tag".

Once you have altered all relevant `provider.xml` files, restart your OC4J instance.

18.4.1.4 Database-Related Attributes and Parameters of the preferenceStore Tag

Table 18–4 lists and describes the attributes and parameters used with the `preferenceStore` tag when a database preference store is specified.

Table 18–4 Database-Related Attributes and Parameters of the preferenceStore Tag

Attribute/Parameter	Description
`class`	This required attribute specifies the Java class that defines the location and other details of portlet preferences. For example, a database preference store might use: `<preferenceStore` **`class="oracle.portal.provider.v2.preference.DBPreferenceStore"`**`>`

`name`	This required parameter provides a name for the preference store. Use any value you choose. For example:
	`<preferenceStore`
	`class="oracle.portal.provider.v2.preference.DBPreferenceStore">`
	`<name>MyPDKProducerPreferenceStore</name>`
	`</preferenceStore>`

`connection`	This required parameter (for a database preference store) points to a connection defined in J2EE's `data-sources.xml` file. For example:
	`<preferenceStore`
	`class="oracle.portal.provider.v2.preference.DBPreferenceStore">`
	`<name>MyPDKProducerPreferenceStore</name>`
	`<connection>jdbc/PooledConnection</connection>`
	`</preferenceStore>`

18.4.2 Setting Up a File-Based Preference Store

A file-based preference store is the default preference store type for portlet producers and is configured out of the box.

> **Note:** If you choose to use a file-based preference store in a clustered environment, then each node in the cluster must use the same location.

In the event you are moving from a database preference store to a file-based preference store or you want to change the location of the file-based preference store, this section describes how to set up a file-based preference store for both WSRP and PDK-Java portlet producers. It contains the following subsections:

- Section 18.4.2.1, "Configuring a WSRP Producer to Use a File-Based Preference Store"
- Section 18.4.2.2, "Configuring a PDK-Java Producer to Use a File-Based Preference Store"

> **Note:** For information about migrating portlet preferences from a database preference store to a file preference store, see Appendix B, "Additional Portlet Configuration".

18.4.2.1 Configuring a WSRP Producer to Use a File-Based Preference Store

For a WSRP producer, all of the configuration required for a file-based preference store occurred in the producer's `web.xml` file. Multiple producers mean multiple `web.xml` files. In such an environment, be sure to configure all relevant `web.xml` files.

Note: See Example 18–1 for some sample file locations of various WSRP producers' `web.xml` files.

Two JNDI variables control preference store functionality for WSRP producers: `persistentStore` and `fileStoreRoot`. Both database and file preference stores require the `persistentStore` variable. The `fileStoreRoot` variable is used only in the presence of a file preference store.

Note: For additional information about setting JNDI variables, see Section 19.2.3, "Using JNDI Variables".

Table 18–5 lists and describes the WSRP producer preference store-related JNDI variables that are relevant to a file-based preference store.

Table 18–5 WSRP Producer File Preference Store-Related JNDI Variables

Variable Name	Default Variable Value	Description
`oracle/portal/wsrp/server/persistentStore`	`File`	Determines which data store (File or Database) is used for persisting a portlet application's consumer registration handles and portlet preferences. Permissible values: {`File`, `Database`}
`oracle/portal/wsrp/server/fileStoreRoot`	`portletdata`	Defines the path to the root directory to be used by the file preference store. Absolute paths are interpreted relative to the file system root. Relative paths are interpreted relative to the *`ORACLE_HOME\portal`* directory. Note that all producers running within the same OC4J instance must use the same path for this variable. Otherwise, you will get a `Portlet unavailable` error for some portlets.

Example 18–5 depicts default producer preference store settings in a WSRP producer's `web.xml` file. Relevant settings are marked in bold:

Note: Preference store-related variables may or may not be present in a WSRP producer's `web.xml` file. Before you enter the variables, search for them. If the variables are present, then modify them as desired. If the variables are not present, then enter them as illustrated in Example 18–5.

Example 18–5 Configuring web.xml to Use a File-Based Preference Store

```
<env-entry>
   <env-entry-name>oracle/portal/wsrp/server/persistentStore</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value>File</env-entry-value>
```

```
</env-entry>
<env-entry>
   <env-entry-name>oracle/portal/wsrp/server/fileStoreRoot</env-entry-name>
   <env-entry-type>java.lang.String</env-entry-type>
   <env-entry-value>portletdata</env-entry-value>
</env-entry>
```

Once you change all relevant `web.xml` files, restart your OC4J instance.

18.4.2.2 Configuring a PDK-Java Producer to Use a File-Based Preference Store

PDK-Java producers use the `preferenceStore` tag in their `provider.xml` configuration files to indicate the type of producer preference store to use. Multiple producers mean multiple `provider.xml` files. In such an environment, be sure to configure all relevant `provider.xml` files.

Note: See Example 18–3 for some sample file locations of various PDK-Java producers' `provider.xml` files.

Modify the `preferenceStore` tag to use a file-based preference store. Add the tag if necessary. In Example 18–6, you would substitute *producer_name* with the actual producer name (the name you enter here is your choice). Otherwise, use the example exactly as written for all PDK-Java portlets except OmniPortlet (see Example 18–7).

Example 18–6 Configuring provider.xml to Use a File-Based Preference Store

```
<preferenceStore class="oracle.portal.provider.v2.preference.FilePreferenceStore">
   <name>producer_name</name>
   <useHashing>true</useHashing>
</preferenceStore>
```

OmniPortlet uses a different `preferenceStore` class from the standard class used by all other PDK-Java portlet producers. The OmniPortlet `preferenceStore` class extends the default class with features specific to OmniPortlet. Additionally, it makes use of the `rootDirectory` parameter, which specifies where portlet preferences are stored.

Note: The `rootDirectory` parameter is not exclusive to OmniPortlet. Other portlets can use it as well. For example:

```
<rootDirectory>
   ORACLE_HOME/portal/portletdata/tools/providerBuilder
</rootDirectory>
```

When the `rootDirectory` parameter is not specified, a default location is used for storing portlet preferences. This is the same location where the producer's provider definition file is stored.

In `rootDirectory`, the expression `${oracle.home}` is unique to OmniPortlet, and must not be used with other portlet preference store configurations.

For a list and description of file-related attributes and parameters of the `preferenceStore` tag, see Section 18.4.2.3, "File-Related Attributes and Parameters of the preferenceStore Tag".

Example 18–7 illustrates the OmniPortlet `preferenceStore` class specification for a file-based preference store. When you specify a file-based preference store for OmniPortlet, use the parameters and parameter values provided in Example 18–7.

Example 18–7 Configuring OmniPortlet's provider.xml File to Use a File-Based Preference Store

```
<preferenceStore class="oracle.webdb.reformlet.ReformletFilePreferenceStore">
   <name>omniPortletprefStore</name>
   <useHashing>true</useHashing>
   <rootDirectory>
      ${oracle.home}/portal/portletdata/tools/omniPortlet
   </rootDirectory>
</preferenceStore>
```

Once you change all relevant `provider.xml` files, restart your OC4J instance.

18.4.2.3 File-Related Attributes and Parameters of the preferenceStore Tag

Table 18–6 lists and describes the attributes and parameters used with the `preferenceStore` tag when a file-based preference store is specified.

Table 18–6 File-Related Attributes and Parameters of the preferenceStore Tag

`class`	This required attribute specifies the Java class that defines the location and other details of portlet preferences. For example, most PDK-Java portlet producers use the following class for a file-based preference store: `<preferenceStore` **`class="oracle.portal.provider.v2.preference.FilePreferenceStore">`** OmniPortlet uses its own class. For example: `<preferenceStore` **`class="oracle.webdb.reformlet.ReformletFilePreferenceStore">`** For more information, see Section 18.4.2.2, "Configuring a PDK-Java Producer to Use a File-Based Preference Store".

`name`	This required parameter provides a name for the preference store. Use any value you choose. For example:
	`<preferenceStore`
	`class="oracle.portal.provider.v2.preference.DBPreferenceStore">`
	`<name>MyPDKProducerPreferenceStore</name>`
	`</preferenceStore>`

`rootDirectory`	This optional parameter specifies the location where the file-based preference store preferences are stored.
	`<preferenceStore`
	`class="oracle.webdb.reformlet.ReformletFilePreferenceStore">`
	`<name>PDKProducerPreferenceStore</name>`
	`<useHashing>true</useHashing>`
	`<rootDirectory>`
	`ORACLE_HOME/portal/portletdata/tools/providerBuilder`
	`</rootDirectory>`
	`</preferenceStore>`
	
	OmniPortlet has its own value for this parameter. For example:
	`<rootDirectory>`
	`${oracle.home}/portal/portletdata/tools/omniPortlet`
	`</rootDirectory>`
	
	For more information, see Section 18.4.2.2, "Configuring a PDK-Java Producer to Use a File-Based Preference Store".
	When this parameter is not included with the `preferenceStore` tag, the default file-based preference store location is used. This is the same folder where the producer's provider definition file is stored.

`useHashing`	This optional parameter takes the value `true` or `false`. When it is `true`, each preference data file is stored in an extra subdirectory with a name determined by hashing the data file name. Using this parameter can improve file system performance by limiting the number of preference data files stored in a single directory.
	For example:
	`<preferenceStore`
	`class="oracle.portal.provider.v2.preference.FilePreferenceStore">`
	`<name>PDKProducerPreferenceStore</name>`
	`<useHashing>true</useHashing>`
	`</preferenceStore>`

18.4.3 What You Should Know About the Web Clipping Portlet and a Preference Store

The Web Clipping portlet provides a unique case in terms of its use of a preference store. It does not use one. Rather it places Web Clipping definitions and clippings into a repository. This can be either an Oracle MetaData Services (MDS) repository (the default, out-of-the-box configuration located on a file system) or a database repository (best for high availability).

This section describes how to configure a Web Clipping portlet producer to use a database repository or an MDS repository for Web Clipping definitions and clippings. It contains the following subsections:

- Section 18.4.3.1, "Configuring a Web Clipping Portlet Producer to Use a Database Repository"
- Section 18.4.3.2, "Configuring a Web Clipping Portlet Producer to Use an Oracle Metadata Services Repository"
- Section 18.4.3.3, "Attributes and Child Tags of the repositoryInfo Tag"

18.4.3.1 Configuring a Web Clipping Portlet Producer to Use a Database Repository

To use a database repository for storing Web Clipping definitions and clippings, you must complete two tasks:

- Section 18.4.3.1.1, "Creating a Database Schema for Web Clipping Portlet Definitions and Clippings"
- Section 18.4.3.1.2, "Configuring the Web Clipping Portlet's provider.xml File"

18.4.3.1.1 Creating a Database Schema for Web Clipping Portlet Definitions and Clippings

To create a database schema for Web Clipping definitions and clippings, run the Java command described in Example 18–8.

Example 18–8 Java Command for Creating a Schema for Web Clipping Portlet Definitions and Clippings

```
ORACLE_HOME/jdk/bin/java -classpath ORACLE_HOME/lib/xmlparserv2.jar:
ORACLE_HOME/jdbc/lib/ojdbc14.jar:ORACLE_HOME/portal/jlib/wce.jar
oracle.portal.wcs.Installer -installSchema -username dbuser -password
dbpassword -dburl jdbc:oracle:thin:@//host:port/dbid
```

> **Note:** The classpath in Example 18–8 uses different separators for UNIX and Windows. On UNIX systems, the `classpath` uses a colon (:) separator. On Windows systems, the `classpath` uses a semicolon (;) separator.

Where:

- *ORACLE_HOME* is the path to your Oracle home directory
- *dbuser* is the database user for the schema

  Consider using the same database user you used to create the WSRP and PDK-Java preference store database schema. See Section 18.4.1.1, "Creating the Schema for a Database Preference Store". If you do not use the same user, then you must create a new user and grant it connect and resource privileges.
- *dbpassword* is the specified user's password

- *dburl* is the URL for the database

  This is the database that contains the schema that you created for Web Clipping portlet definitions and clippings using the following syntax:

  ```
  jdbc:oracle:thin:@//dbhost:dbport/service_name
  ```

  For example:

  ```
  jdbc:oracle:thin:@//shobeen:1521/sales_us
  ```

18.4.3.1.2 Configuring the Web Clipping Portlet's provider.xml File

Set up the Web Clipping producer database repository in the producer's `provider.xml` file located at:

```
ORACLE_HOME\j2ee\OC4J_
WebCenter\applications\portalTools\webClipping\WEB-INF\providers\webClipping
```

To use a database repository, update the `repositoryInfo` tag to use the `DatabaseInformation` class as illustrated in Example 18–9. Tag parameters are listed and described in Section 18.4.3.3, "Attributes and Child Tags of the repositoryInfo Tag"

Example 18–9 Configuring the Web Clipping provider.xml File to Use a Database Repository for Web Clipping Definitions and Clippings

```
<repositoryInfo class="oracle.portal.wcs.provider.info.DatabaseInformation">
   <useRAA>false</useRAA>
   <databaseHost>infradbhost</databaseHost>
   <databasePort>1521</databasePort>
   <databaseSid>iasdb</databaseSid>
   <databaseUsername>webclip_user</databaseUsername>
   <databasePassword>!AX3tR</databasePassword>
   <useASO>false</useASO>
</repositoryInfo>
```

Restart your OC4J instance after making changes to the `provider.xml` file.

18.4.3.2 Configuring a Web Clipping Portlet Producer to Use an Oracle Metadata Services Repository

Set up the Web Clipping producer MDS repository in the producer's `provider.xml` file located at:

```
ORACLE_HOME\j2ee\OC4J_
WebCenter\applications\portalTools\webClipping\WEB-INF\providers\webClipping
```

To use an MDS repository, update the `repositoryInfo` tag to use the `MdsInformation` class as illustrated in Example 18–10. Tag parameters are listed and described in Section 18.4.3.3, "Attributes and Child Tags of the repositoryInfo Tag"

Example 18–10 Configuring the Web Clipping provider.xml File to Use an Oracle Metadata Services Repository for Web Clipping Definitions and Clippings

```
<repositoryInfo class="oracle.portal.wcs.provider.info.MdsInformation">
   <mdsConfigLocation>mds-config.xml</mdsConfigLocation>
</repositoryInfo>
```

This configuration points to the `mds-config.xml` file that specifies the actual MDS location. By default, the `mds-config.xml` file is located at:

```
ORACLE_HOME\j2ee\OC4J_WebCenter\applications\portalTools\webClipping\WEB-INF
```

The `mds-config.xml` file specifies the location of the repository in a property tag (Example 18–11).

Example 18–11 Specifying the Location of the MDS Repository in the mds-config.xml File

```
<property name="metadata-path" value="portletdata/tools/webClipping"/>
```

> **Note:** If you manually reconfigure the value for `metadata-path` in `mds-config.xml`, then that directory must exist at the time the producer is restarted.

The location specified for `value` is relative to *ORACLE_HOME*`\portal`. Any relative path specified will be interpreted to be relative to *ORACLE_HOME*`\portal`. If you want to use another location, for example, a location outside the Oracle home, then specify an absolute path, such as `c:\mds`.

Restart your OC4J instance after making the changes outlined in this section.

18.4.3.3 Attributes and Child Tags of the repositoryInfo Tag

Table 18–7 lists and describes the attributes and child tags of the `repositoryInfo` tag.

> **Note:** The attributes and child tags of the `repositoryInfo` tag are also described in the comments in the Web Clipping `provider.xml` file.
>
> In the comments in the `provider.xml` file, the example provided for *Oracle9i Database or later using AS Infrastructure Database* is specific to OracleAS Portal and its infrastructure database and application programming interfaces. That example should not be used for WebCenter application implementations.

Table 18–7 Attributes and Child Tags of the repositoryInfo Tag

Attributes/Parameter	MDS/database	Description

`class`	both	The `class` attribute specifies the type of repository that is used to store Web Clipping definitions. The two possible values for this attribute are:
		`oracle.portal.wcs.provider.info.MdsInformation`
		This value signifies that MDS is used to store the Web Clipping definitions and that MDS configuration is pushed to the `mds-config.xml` file.
		`oracle.portal.wcs.provider.info.DatabaseInformation`
		This value signifies that an Oracle9*i* Database or later is used to store the Web Clipping definitions and that the database connection details will be included as children in the `repositoryInfo` tag.

`mdsConfigLocation`	MDS	Use the `mdsConfigLocation` tag when the value for the `class` attribute indicates an MDS repository. It points to the MDS configuration file, `mds-config.xml`. This configuration points to the `mds-config.xml` file that specifies the actual MDS location. The `mds-config.xml` file is located at: *ORACLE_HOME*`\j2ee\OC4J_WebCenter\applications\portalTools\webClipping\WEB-INF` For more information, see Example 18–11.
`useASO`	database	Set to `true` or `false`: ■ Specify `true` if you want to use Advanced Security Option to encrypt the communication channel between the Web Clipping and the database. This is provided for introducing added security in case-sensitive data is contained in the clipped content. ■ Specify `false` to omit this option.
`useRAA`	database	Set to `true` or `false`: ■ Specify `true` if the Repository Access APIs will be used to access the database connection parameters. Specifying `true` is equivalent to making the Web Clipping producer use the default OracleAS Infrastructure Database as the repository. Specifying `true` removes the need for other `repositoryInfo` child tags. ■ Specify `false` to omit this option.
`databaseHost`	database	Specify the host name of the Oracle database. Use only version 9*i* or later. For example: `mycompany.dbhost.com`
`databasePort`	database	Specify the port number of the Oracle database listener. This is usually `1521`.

Table 18–7 (Cont.) Attributes and Child Tags of the repositoryInfo Tag

Attributes/Parameter	MDS/database	Description
`databaseSid`	database	Specify the Oracle SID of the database that will host the Web Clipping repository.
`databaseUsername`	database	Enter the user name used to log in to the database.
`databasePassword`	database	Enter a plain text password for the specified database username. Prefix the password with an exclamation point (!) to enable the Web Clipping producer to encrypt the password once the producer starts. For example: `!AX3tR`

18.5 Building JPS-Compliant Portlets with Oracle JDeveloper

Using the JSR 168 Java Portlet Wizard in Oracle JDeveloper you can expose your portlet over WSRP 2.0 quickly and easily. Note that the wizard supports both WSRP 1.0 and WSRP 2.0. If you choose to create a WSRP 2.0 portlet, then an additional page appears in the wizard for the WSRP 2.0 enhancements.

This section assumes the following:

- You are familiar with portlet terminology such as portlet modes. See Chapter 14, "Understanding Portlets" and Section 18.1, "Guidelines for Creating Java Portlets".
- You have installed the preconfigured OC4J provided on OTN. See Section 3.2, "Using the Preconfigured OC4J".
- You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it.

This section walks you through the JSR 168 Java Portlet Wizard. You can choose which portlet modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML). The wizard then creates a simple sample implementation for each of the selected modes.

> **Note:** The figures in this section were taken with a Look and Feel setting of Windows in Oracle JDeveloper. If your Look and Feel is set to Oracle, then what you see on your screen will vary slightly, but the content and functionality remains the same. To change the Look and Feel setting, select **Preferences** from the Tools menu, and then select Environment.

The steps to create a portlet using the JSR 168 Java Portlet Wizard are as follows:

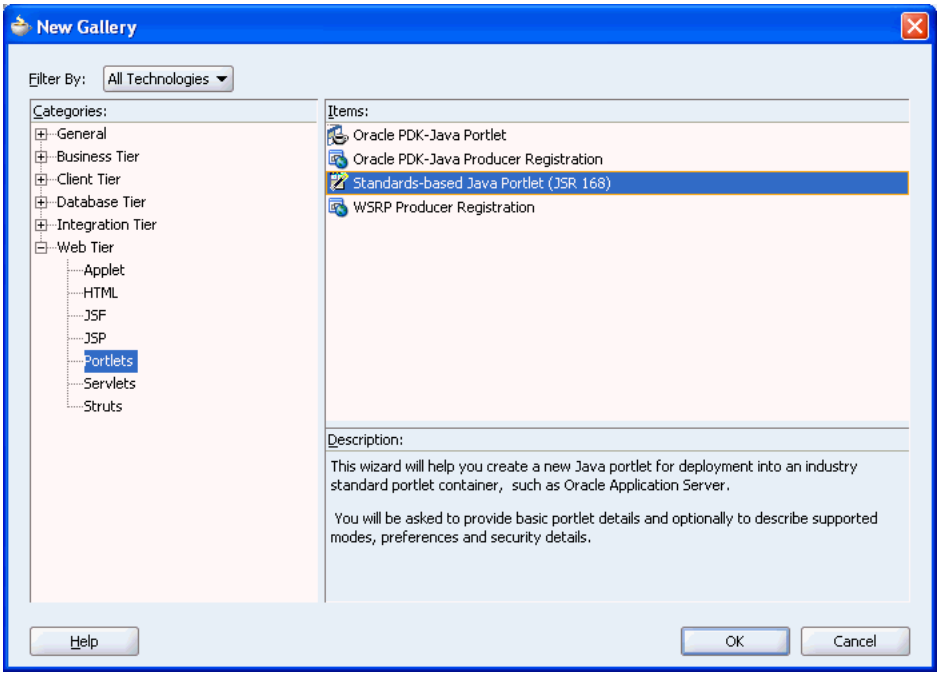
1. Start Oracle JDeveloper.
2. In the Applications Navigator, expand the application under which you want to create your portlet.
3. Right-click the project under which you want to create your portlet, and select **New.**

> **Note:** If you chose the WebCenter Application Template when you created the application, then create the portlet in the Portlets project. If you do not have such an application or project yet, then see Section 3.1, "Creating a WebCenter Application".

4. In the New Gallery, expand the Web Tier category and select **Portlets**.
5. In the Items list, select **Standards-based Java Portlets (JSR 168)**, as shown in Figure 18–5.

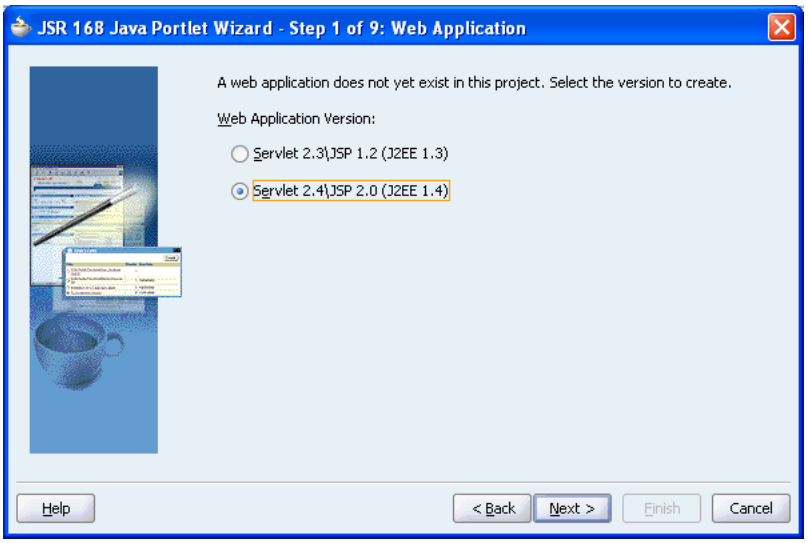
> **Note:** Selecting Java Portlet opens the Portlet Wizard for creating JPS-compliant portlets. Clicking Oracle PDK Java Portlet opens the Portlet Wizard for creating PDK-Java portlets.

Figure 18–5 New Dialog Box



6. Click **OK**. The JSR 168 Java Portlet Wizard opens.
7. If you are on the Welcome page of the wizard, then click **Next** to display the Web Application page (Figure 18–6).

Figure 18–6 Web Application Page

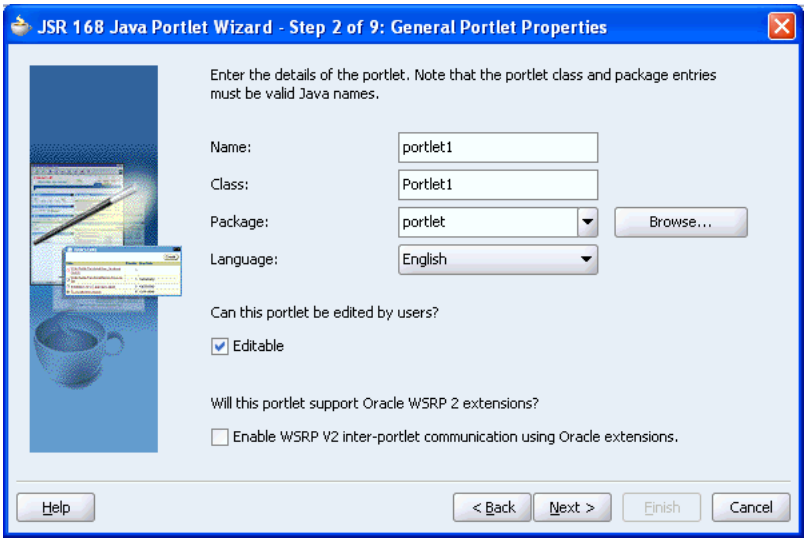


8. Select the **Web Application Version** option for the project. Note that Servlet 2.4 is supported only in Oracle Application Server Release 10.1.3 and later.

> **Note:** If your project already contains any JSPs or JSF JSPs, then the Web Application Version will have already been set. If this is the case, then you will not see this page, and will instead go straight to the General Portlet Properties page.

9. Click **Next** to display the General Portlet Properties page (Figure 18–7).

Figure 18–7 General Portlet Properties Page



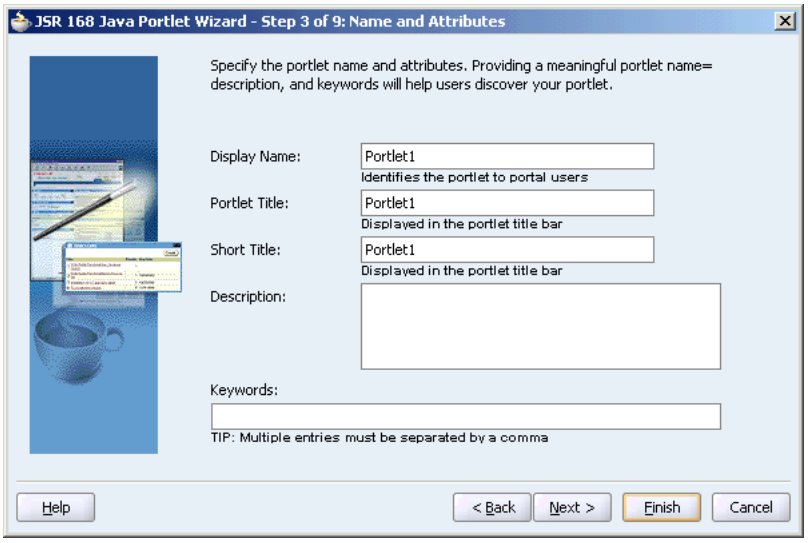
10. In the Class field, enter a name for the class that the wizard will create for the portlet. You can accept the default name provided or supply your own.

11. In the Package list, select the package in which the class will reside. Click the **Browse** button to find packages within the project. If you do not select a specific package, then the wizard uses the default package of the project.
12. In the Default Language list, select the default language that your portlet will support. The wizard uses English by default.
13. Select **Editable** to add an edit mode. In the wizard, this option is selected by default.
14. Select **Enable WSRP V2 inter-portlet communication using Oracle extensions** to indicate that this portlet will support Oracle WSRP 2 extensions. Selecting this option creates the `oracle-portlet.xml` file, which is used for WSRP 2.0 features, such as navigation parameters. The WSRP 2.0 standard extends WSRP 1.0 by including support for inter-portlet communication (through navigation parameters) and export/import of portlet customizations.

> **Note:** JSR 168 portlets built with the Oracle extensions can be consumed by any consumer that supports WSRP 2.0. To leverage WSRP 2.0, the portlets must be deployed to the Oracle Application Server release 10.1.3.2.

15. Click **Next** to display the Name and Attributes page (Figure 18–8).

Figure 18–8 Name and Attribution Page



16. In the Display Name field, enter a name for your portlet. This portlet attribute is not used by WebCenter application, but may be useful with portlets consumed by other applications. For example, OracleAS Portal uses this value on its portlet pick list (that is, in the Portlet Repository).
17. In the Portlet Title field, enter a title for your portlet. This title will be displayed on the portlet header when the portlet appears on a page.
18. In the Short Title field, enter a shorter title for your portlet. This portlet attribute is not implemented in WebCenter applications, but may be useful with portlets consumed by other applications. For example, if you use this portlet in OracleAS Portal, and plan to provide a mobile option to your users, then enter a Short Title

for the portlet, to be displayed in the portlet's header in place of the longer Portlet Title.

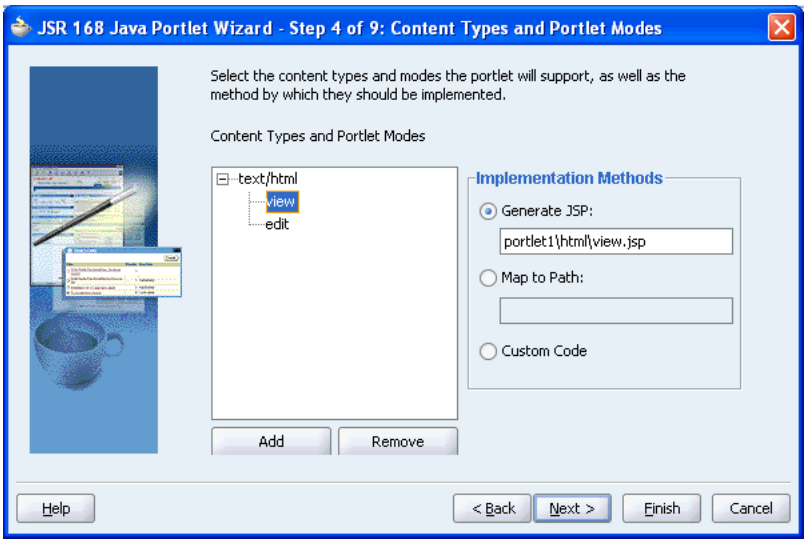
19. In the Description field, enter a description of your portlet. This portlet attribute is not used by WebCenter applications, but may be useful with portlets consumed by other applications. For example, if you use this portlet in OracleAS Portal, then the description displays beneath the portlet in the Portlet Repository.

20. In the Keywords field, enter any additional keywords to help users find your portlet in a search. This portlet attribute is not implemented in WebCenter applications, but may be useful with portlets consumed by other applications.

21. Click **Next** to display the Content Types and Portlet Modes page shown in Figure 18–9.

    Alternatively, you can click **Finish** to create the portlet immediately, using the default values for all remaining settings.

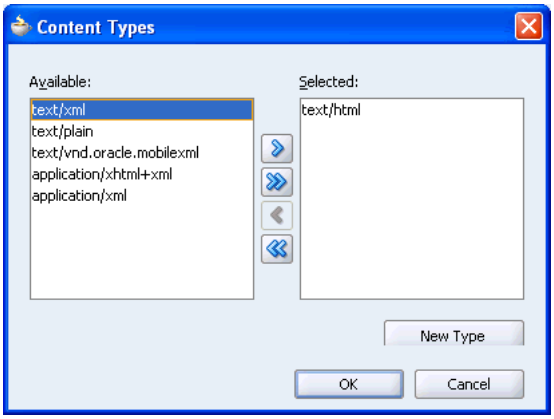
Figure 18–9 Content Types and Portlet Modes Page



22. By default, your portlet will display text/html as the content type. If you want to add other content types, then select **text/html**, then click **Add.**

    The list of available content types displays (Figure 18–10). Select the desired content types in the Available list and use the arrow buttons to move them to the Selected list. When you are finished, click **OK.**

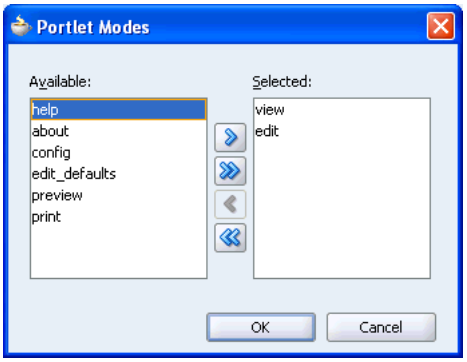
Figure 18–10 Content Types Dialog Box



23. By default, your portlet includes View mode. If you selected Customizable on the General Portlet Properties page, then your portlet also includes Edit mode. If you want to include additional portlet modes, then select an existing portlet mode (for example, view), then click **Add.**

    The list of available portlet modes displays (Figure 18–11). You can add portlet modes by moving the desired modes from the Available list to the Selected list. When you are finished, click **OK.** For more information about portlet modes, see Section 18.1, "Guidelines for Creating Java Portlets".

Figure 18–11 Portlet Modes Dialog Box



24. Once you have added all of the desired portlet modes, choose the function to be performed for each mode. For each portlet mode, click the portlet mode and select a option on the right, as follows:

    - Select Generate JSP if you want Oracle JDeveloper to generate a JSP for the portlet mode. Enter a name for the JSP in the corresponding field, or accept the default.

      When you complete the wizard, the generated JSP displays in the Applications Navigator where it can be selected for further development. This is the default selection for all portlet display modes. This selection enters code in the generated portlet java class that routes requests for the given mode to the generated JSP.

    - Select Map to Path if you want to map the portlet mode to a an existing Web resource, such as a page. Enter the path in the corresponding field. With this selection, you must write the targeted resource or file yourself. The target could be, for example, a JSP, a servlet, or an HTML file. This selection enters

code in the generated portlet java class that routes requests for the given mode to the specified target.

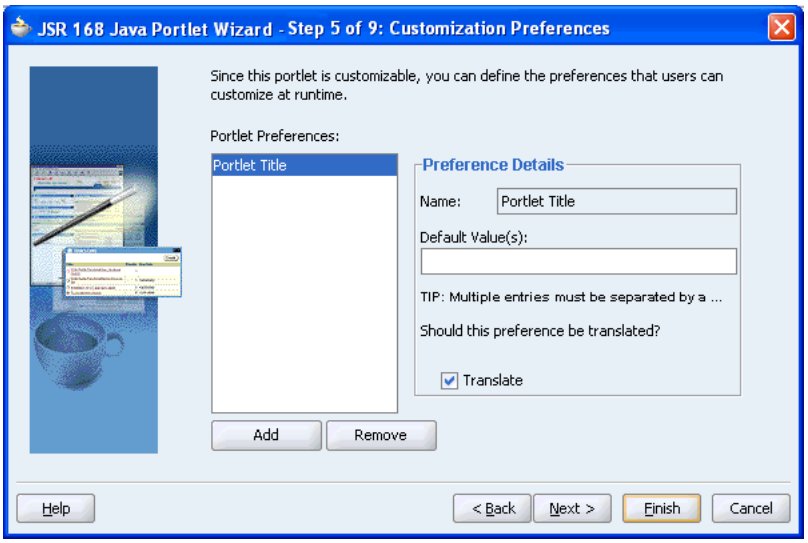
- Select Custom Code if you want to implement the portlet mode though a custom coded object. You will create this object later. This selection generates a skeleton method to render content (`private void do<MODE_NAME><CONTENT_TYPE>`) in the generated portlet Java class. You must update this code to render useful content.

25. Click **Next.**

    If you selected Editable on the General Properties page earlier in the wizard, then the Customization Preferences page displays (Figure 18–12). Go to step 26.

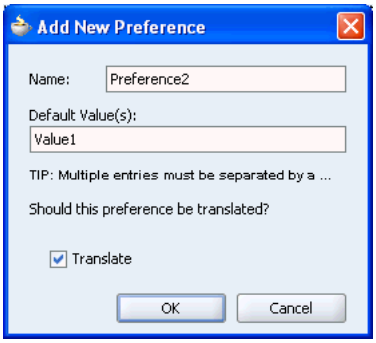
    If you did not select this option, then the Security Roles page displays (Figure 18–12). Go to step 34.

Figure 18–12 Customization Preferences Page



26. If you want to include additional customization preference, then click **Add.** The Add New Preferences dialog box displays (Figure 18–13).

Figure 18–13 Add New Preference Dialog Box



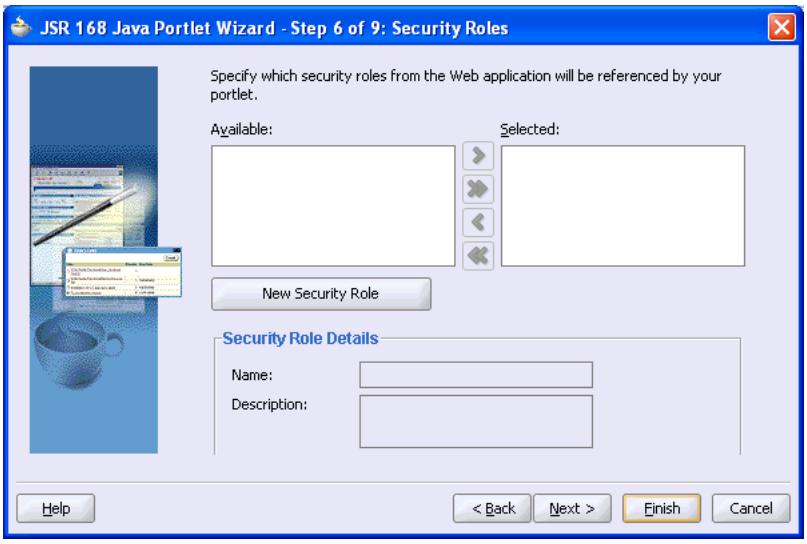
27. In the Name field, enter a name for the new customization preference. The name must be unique in the portlet. Use only letters, numbers, and the underscore character.

28. In the Default Value(s) field, enter one or more default value for the new customization preference. Separate multiple values with commas.
29. If you want the customization preference value to be translated, then select the **Translate** check box. If you select this option, then Oracle JDeveloper generates a resource bundle class with strings for which you can obtain translations. At run time, the portlet references the resource bundle entries.

> **Note:** The Name is always translated, but there is not always a need to translate the Default Value. For example, if the value is an integer, then no translation is needed.

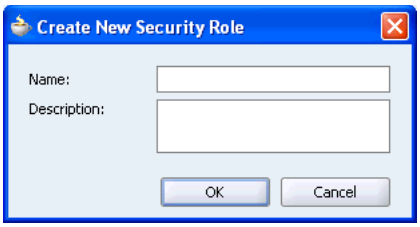
30. Click **OK.** Repeat the preceding steps if you want to add more customization preferences.
31. To edit details for existing customization preferences, select the preference in the Portlet Preferences list, and edit the fields in the Preference Details section.
32. To delete an existing customization preference, select the preference in the Portlet Preferences list, and click **Remove.**
33. Click **Next** to display the Security Roles page (Figure 18–14).

Figure 18–14 Security Roles Page



34. JSR 168 portlets may use J2EE security roles that are defined in `web.xml` and referenced in `portlet.xml`. The Available list displays the security roles defined in the portlet application's web deployment file (`web.xml`). Moving a security role from the Available list to the Selected list creates a reference of the security role in the application's portlet deployment file (`portlet.xml`) that refers to the security role in `web.xml`.
35. If you want to define a new security role, then click **New Security Role.** The Create New Security Role dialog box displays (Figure 18–15).

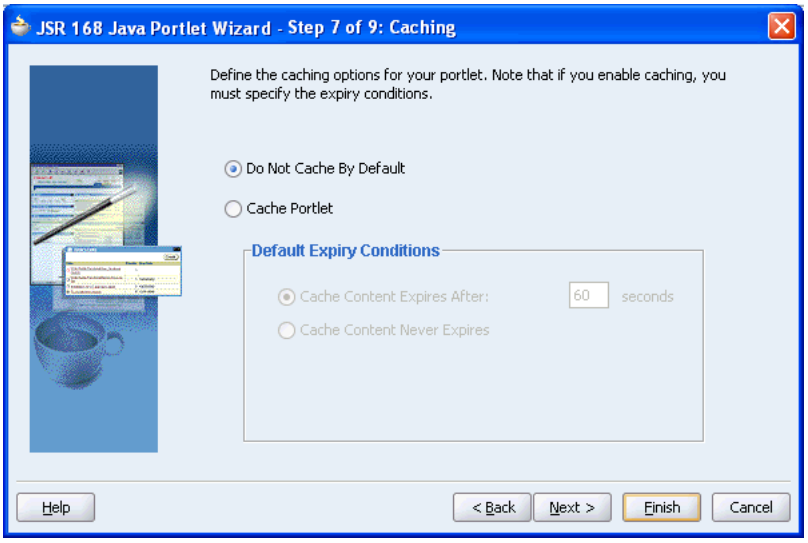
Figure 18–15 Create New Security Role Dialog Box



36. In the Name field, enter a unique name for the security role.
37. In the Description field, enter a description for the security role, explaining the access privileges and restrictions this role will have on the portlet.
38. Click **OK.**

    The new security role is added to the Available list. You can also manually create security roles, as described in Section 10.2, "Setting Up Security for Your Application".
39. Click **Next** to display the Caching page (Figure 18–16).

Figure 18–16 Caching Page



40. If you want to enable caching for your portlet, then select **Cache Portlet**.

    Selecting this option indicates that portlet caching is managed by the portlet container. The portlet itself may choose to cache content for any given response. The settings on this page apply only when the portlet itself does not specify a caching condition for a response.
41. In the Default Expiry Conditions section, select:
    - Cache Content Expires After [] seconds if you want the cached portlet content to expire after a certain amount of time. Specify the time limit in the adjacent field.
    - Cache Content Never Expires if you do not want the cached portlet content to expire. You may want to select this option if the portlet contains static content that is unlikely to change.

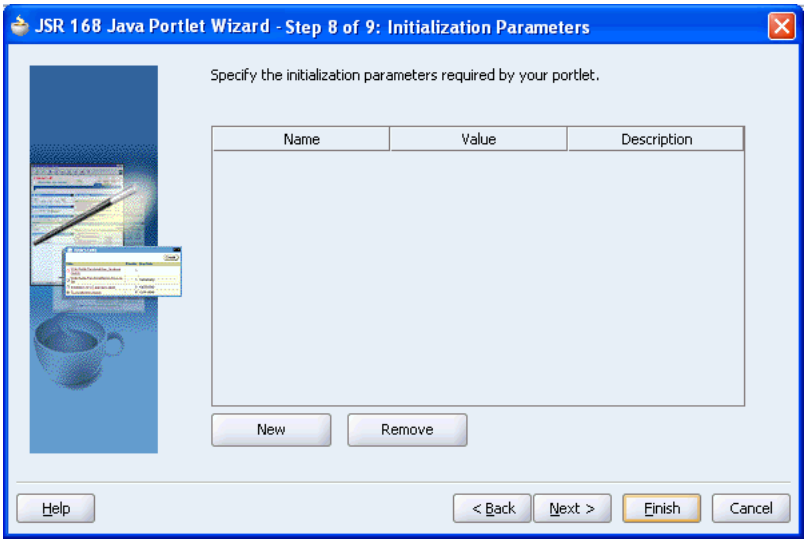
Note: If you do not want any default caching for this portlet, then choose Do Not Cache By Default. In this case, the wizard actually sets a cache duration of 0 seconds. As stated earlier, this cache setting only comes into play when the portlet itself does not specify a caching condition for a response.

If you choose no caching here and you later decide that you want default caching for the portlet, then you can easily go back and change the cache duration value in the `portlet.xml` file, which is generated by the wizard, to a number greater than zero.

---

42. Click **Next** to display the Initialization Parameters page, shown in Figure 18–17.

Figure 18–17 Initialization Parameters Page



43. Initialization parameters provide the Web application developer, who decides what goes into the `.war` file, an alternative to JNDI variables for configuring the behavior of all of the different components of the Web application (for example, servlets and portlets) in a compatible way. These initialization parameters are added to the `portlet.xml` file.

    If you want to add an initialization parameter, then click **New.** This adds a new row to the table of parameters. You can then double click the row to edit the details.

44. In the Name field, enter a unique name for the initialization parameter. Use only letters, numbers, and the underscore character.

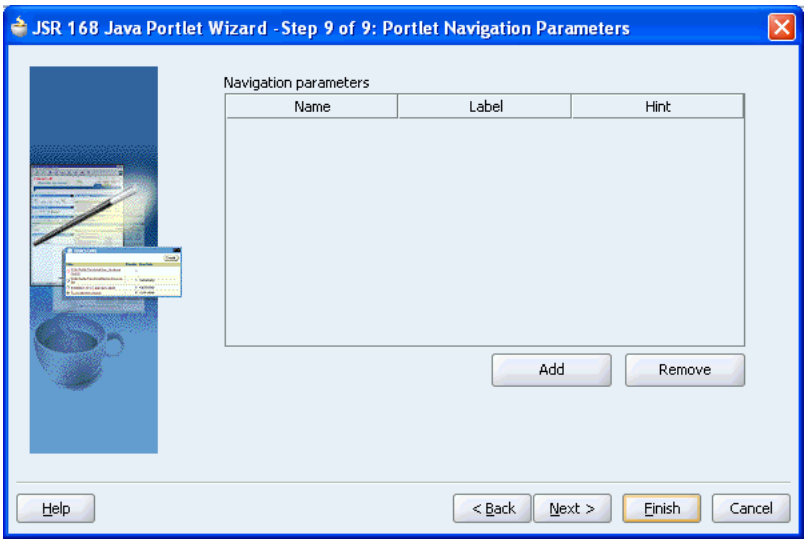
45. In the Value field, enter a default value for the parameter

46. In the Description field, enter a description for the parameter.

47. To delete an initialization parameter, select it in the table of parameters and click **Remove.**

48. Click **Next** to display the Portlet Navigation Parameters page (Figure 18–18).

Figure 18–18 Portlet Navigation Parameters Page



For information about Portlet Navigation Parameters, see Section 19.1.2, "Implementing Navigational Parameters (WSRP 2.0)".

49. Click **Next** to display the Finish page.

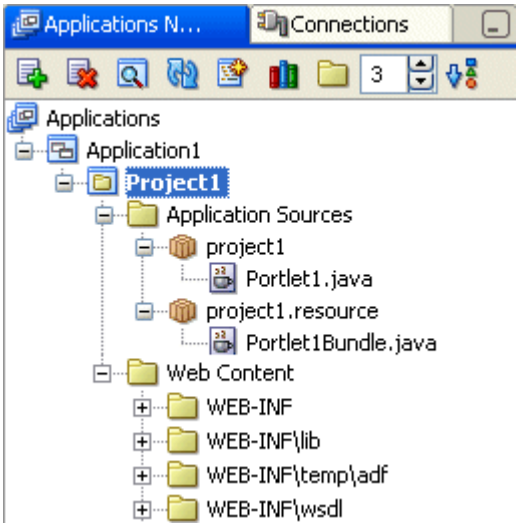
50. Click **Finish** to generate the files for your portlet. The following files should be generated for your project node in the Application Navigator (see Figure 18–19):

    - If you selected Generate JSP for the portlet modes, generated code for each mode

      If you selected Custom Code instead, then that code will reside in the portlet's Java class

    - Two Java classes:
      - `<packagename>.<portletname>.java` is invoked by the portlet container and contains all the methods required by the portlet standards
      - `<packagename>.<portletname>Bundle.java` contains all the translation strings for the portlet
    - `portlet.xml`
    - `oracle-portlet.xml`
    - `web.xml`

Figure 18–19 Example of Files Generated for a JPS-Compliant Portlet



The next step is to add your own portlet logic to the portlet to implement your desired functionality. See Section 18.8, "Adding Portlet Logic".

18.6 Introduction to PDK-Java

PDK-Java gives you a framework to simplify the development of Java portlets by providing commonly required utilities and enabling you to leverage existing development skills and application components such as JSPs, servlets, and static HTML pages. PDK-Java also enables you to create portlets without having to deal directly with the complexity of communications between Oracle WebCenter Framework and producers.

The PDK-Java framework is divided into the following areas:

- The **Producer Adapter** insulates the developer from the HTTP syntax defined by Oracle WebCenter Framework for communication with Web producers. It translates the information passed between Oracle WebCenter Framework and your Java Web producer. Without an adapter, your producer would not only manage portlets, but it would also have to communicate this information directly to Oracle WebCenter Framework in the expected language. The adapter eliminates the need for your Web producer to understand the portal language and vice-versa.
- The **Producer Interface** defines the APIs (functions) required by your Java implementation to integrate with the Producer Adapter. The Producer Adapter receives messages from the WebCenter application, translates them into calls to the Producer Interface, and translates the producer's response into a format that the application can understand. The Producer Interface contains a set of Java classes that define the methods your producer needs to implement and, in many cases, provides a standard implementation. Some of the primary classes are as follows:
  - `ProviderDefinition` (`oracle.portal.provider.v2.ProviderDefinition`)
  - `ProviderInstance` (`oracle.portal.provider.v2.ProviderInstance`)
  - `PortletDefinition` (`oracle.portal.provider.v2.PortletDefinition`)

  - `PortletInstance` (`oracle.portal.provider.v2.PortletInstance`)
  - `ParameterDefinition` (`oracle.portal.provider.v2.ParameterDefinition`)
  - `EventDefinition` (`oracle.portal.provider.v2.EventDefinition`)
- The **Producer Runtime** provides a base implementation that follows the specification of the Producer Interface. The Producer Runtime includes a set of default classes that implement each one of the Producer Interfaces and enables you to leverage the rendering, personalization, and security frameworks provided with PDK-Java. These classes and the associated frameworks simplify the development of a producer by implementing common functions for Oracle WebCenter Framework requests and providing a declarative mechanism for configuring the producer. Using the Producer Runtime, you can focus your development efforts on the portlets themselves rather than the infrastructure needed to communicate with the WebCenter application. If the standard behavior of the Producer Runtime does not meet your requirements, then you can easily extend or override specific behaviors. Some of the primary classes are as follows:
  - `DefaultProviderDefinition` (`oracle.portal.provider.v2.DefaultProviderDefinition`)
  - `DefaultProviderInstance` (`oracle.portal.provider.v2.DefaultProviderInstance`)
  - `DefaultPortletDefinition` (`oracle.portal.provider.v2.DefaultPortletDefinition`)
  - `DefaultPortletInstance` (`oracle.portal.provider.v2.DefaultPortletInstance`)
  - `PortletRenderer` (`oracle.portal.provider.v2.render.PortletRenderer`)
  - `PortletPersonalizationManager` (`oracle.portal.provider.v2.personalize.PortletPersonalizationManager`)
  - `PortletSecurityManager` (`oracle.portal.provider.v1.http.DefaultSecurityManager`)
- The **Producer Utilities** provide methods for simplifying the rendering of portlets. The utilities include methods for constructing valid links (`hrefs`), rendering the portlet's container (including the header), rendering HTML forms that work within a page, and supporting portlet caching.

Guidelines for PDK-Java Portlets

In Oracle WebCenter Framework, PDK-Java portlets work somewhat differently than they did in OracleAS Portal. As a result, you need to be aware of the following new design considerations when you build PDK-Java portlets in Oracle WebCenter Framework:

- Your portlet should not contain any code that relies upon the URL format or parameters in the request that were not explicitly added by your portlet
- You should never assume that your portlet is the only one on a page, regardless of the portlet mode. For example, even if your are in Edit mode, you should not assume that yours is the only portlet on the page.

- Never write a portlet mode that simply redirects. A redirect can only be issued while processing a post to your portlet or following a link generated by your portlet.

18.7 Building PDK-Java Portlets with Oracle JDeveloper

Using the Java Portlet Wizard in Oracle JDeveloper you can begin your portlet development quickly and easily.

This section assumes the following:

- You are familiar with portlet terminology such as portlet modes. See Chapter 14, "Understanding Portlets" and Section 18.1, "Guidelines for Creating Java Portlets".
- You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it. You can download Oracle JDeveloper from OTN. Visit the Oracle JDeveloper page on OTN:

  `http://www.oracle.com/technology/products/jdev/index.html`

This section walks you through the Java Portlet Wizard. You can choose which portlet modes you want to implement and the implementation method (JSP, HTTP servlet, Java class, or HTML). The wizard then creates a simple sample implementation for each of the selected modes.

> **Note:** The figures in this section were taken with a Look and Feel setting of Windows in Oracle JDeveloper. If your Look and Feel is set to Oracle, then what you see on your screen will vary slightly, but the content and functionality remains the same. To change the Look and Feel setting, select Preferences from the Tools menu, and then select Environment.

The steps to create a portlet and producer are as follows:

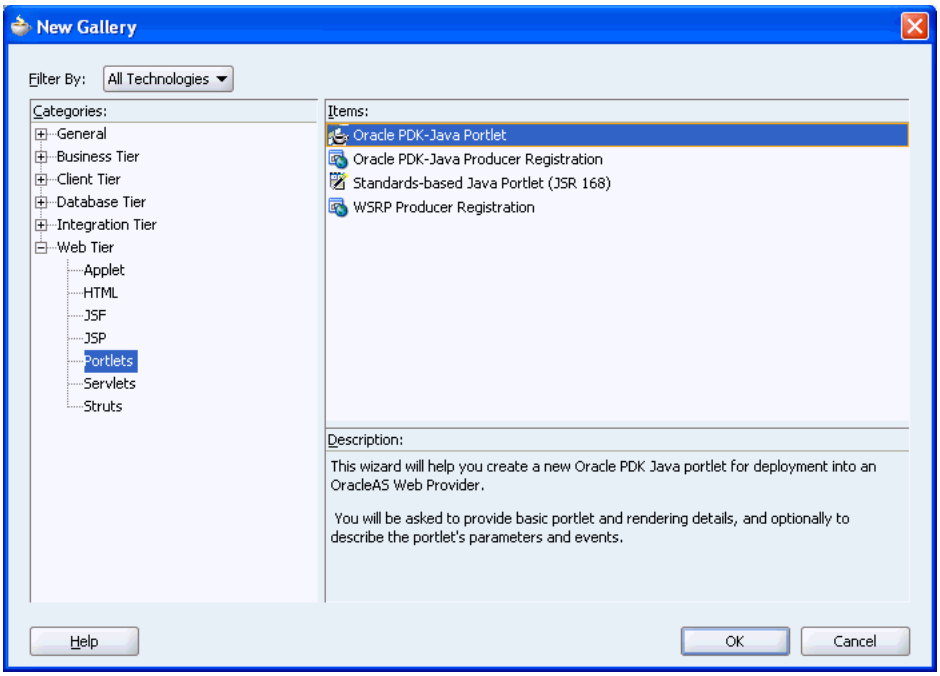
1. Start Oracle JDeveloper.
2. In the Applications Navigator, expand the application under which you want to create your portlet.
3. Right-click the project under which you want to create your portlet, and select **New.**

   > **Note:** If you do not have such an application or project yet, then see Section 3.1, "Creating a WebCenter Application".

4. In the New Gallery, expand the Web Tier category and select **Portlets**.
5. In the Items list, select **Oracle PDK Java Portlet**, as shown in Figure 18–20.

   > **Note:** Selecting Java Portlet opens the Portlet Wizard for creating JPS-compliant portlets. Selecting Oracle PDK Java Portlet opens the Portlet Wizard for creating PDK-Java portlets.

Figure 18–20 New Gallery Dialog Box for Oracle PDK Java Portlet

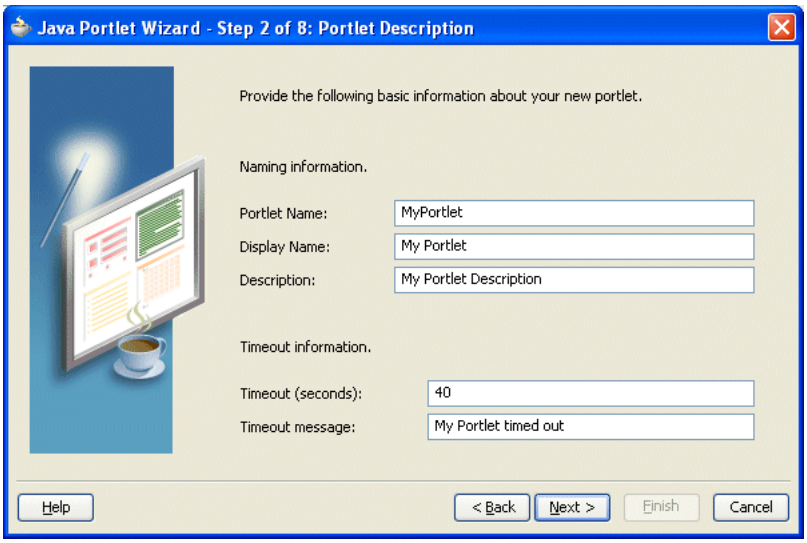


6. Click **OK**. The Java Portlet Wizard displays.
7. If you are on the Welcome page of the wizard, then click **Next** to display the Web Application page (Figure 18–6).
8. Select the **Web Application Version** option for the project.

> **Note:** If your project already contains any JSPs or JSF JSPs, then the Web Application Version will have already been set. If this is the case, then you will not see this page, and will instead go straight to the General Portlet Properties page.

9. Click **Next** to display the Portlet Description page (Figure 18–21).

Figure 18–21 Portlet Description Page

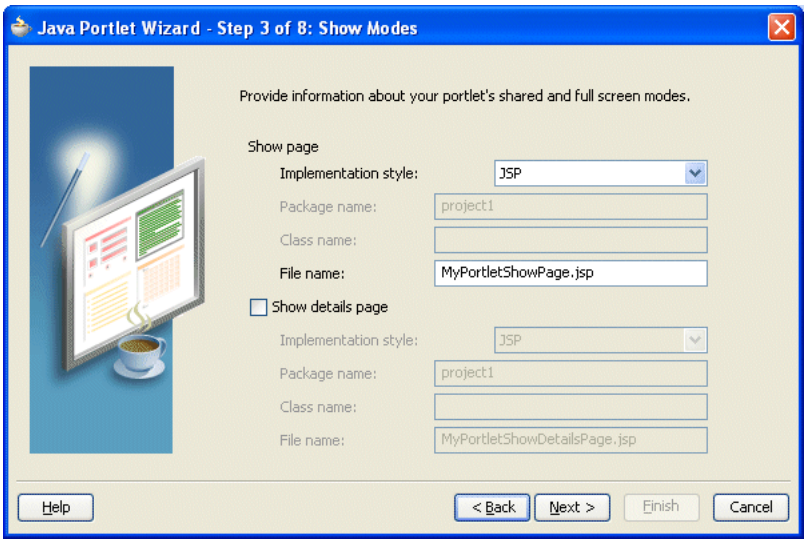


10. In the Portlet Name field, enter a meaningful name for your portlet. This name is used internally and is not exposed to users.
11. In the Display Name field, enter a display name for your portlet. This name will be displayed in portlet selection lists, such as the Component Palette, where users choose which portlets to add to a page.
12. In the Description field, enter a description of your portlet.

> **Note:** This portlet attribute is not implemented in WebCenter applications, but may be useful with portlets consumed by other applications. For example, if you use this portlet in OracleAS Portal, then the description displays beneath the portlet in the Portlet Repository.

13. In the Timeout field, enter the number of seconds to enable for rendering the portlet.
14. In the Timeout Message field, enter a message to display if the rendering of the portlet exceeds the timeout value specified.
15. Click **Next** to display the portlet modes page (Figure 18–22).

Figure 18–22 Portlet Modes Page



16. Under Show page, select the implementation style for Shared Screen mode from the Implementation style list:

    - Select JSP to implement the portlet's Shared Screen mode as a JavaServer Page. In the File name field, enter the name of the file to be generated by the wizard.
    - Select HTTP Servlet to implement the portlet's Shared Screen mode as an HTTP servlet. In the Package name field, enter the name of the package that contains the HTTP servlet. In the Class name field, enter the Java class to be referenced in conjunction with the portlet's Shared Screen mode.
    - Select HTML File to implement the portlet's Shared Screen mode as an HTML file. In the File name field, enter the name of the file to be generated by the wizard. Note that, when you choose HTML File, it results in the following being added inside the `<renderer>` element of your `provider.xml` file:

      ```
      <showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
          <resourcePath>/hub_inside/index.html</resourcePath>
          <contentType>text/html</contentType>
          <charSet>UTF-8</charSet>
       </showPage>
      ```

      `<charSet>` tells the producer what character set to use to encode the HTML page. The default character set specified by the wizard is UTF-8. If you require character set encoding other than UTF-8, then you must update this element of `provider.xml` accordingly.
    - Select Java class to implement the portlet's Shared Screen mode as a Java class. In the Package name field, enter the name of the package that contains the Java class. In the Class name field, enter the name of the Java class.

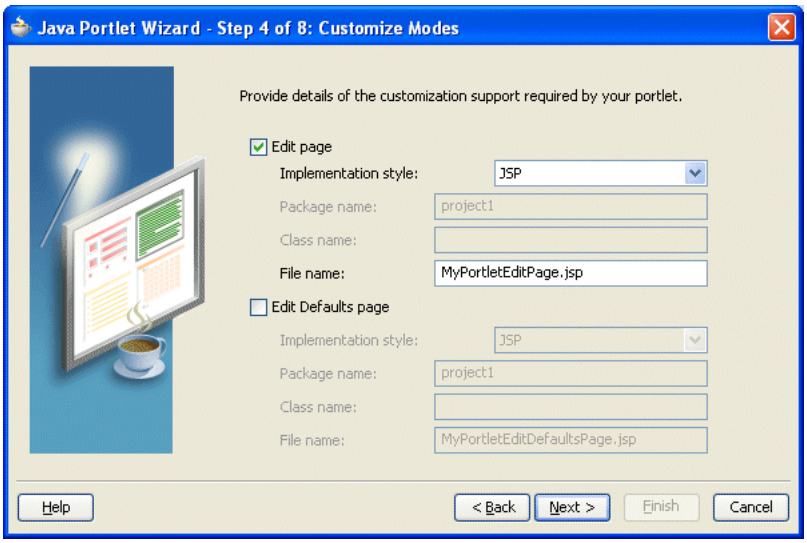
    For more information about Shared Screen mode, see Section 18.1.1.1, "Shared Screen Mode (View Mode for JPS)".

17. If you want to implement Full Screen mode for your portlet, then select **Show Details** page, then select an Implementation style as described earlier for Show page.

    For more information about Full Screen mode, see Section 18.1.1.5, "Full Screen Mode (PDK-Java)".

18. Click **Next** to display the Customize Modes page (Figure 18–23).

Figure 18–23 Customize Modes Page



19. Edit page is selected by default. If you want to implement Edit mode for your portlet, then select an implementation style as described earlier for Show page. If you do not want to implement Edit mode, then clear the Edit page check box.

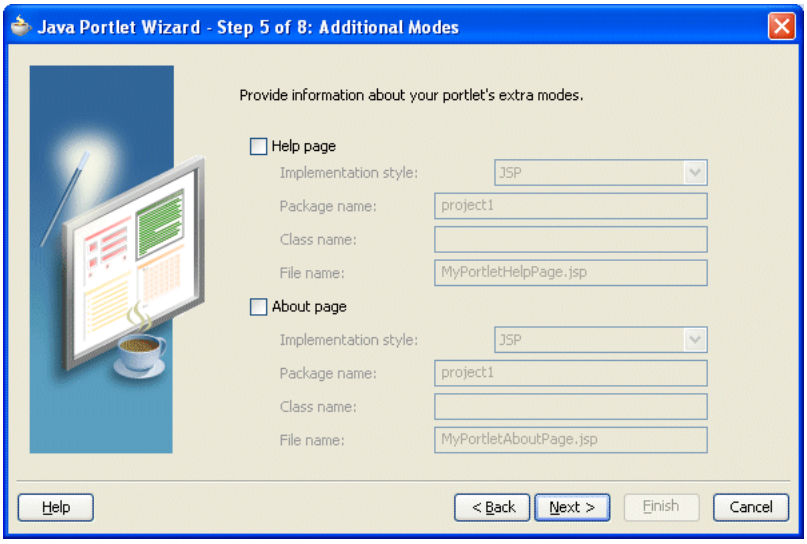
    For more information about Edit mode, see Section 18.1.1.2, "Edit Mode (JPS and PDK-Java)".

20. If you want to implement Edit Defaults mode for your portlet, select **Edit Defaults** page, then select an Implementation style as described earlier for Show page.

    For more information about Edit Defaults mode, see Section 18.1.1.3, "Edit Defaults Mode (JPS and PDK-Java)".

21. Click **Next** to display the Additional Modes page (Figure 18–24).

Figure 18–24 Additional Modes Page



22. If you want to implement Help mode for your portlet, select **Help page**, then select an Implementation style as described earlier for Show page.

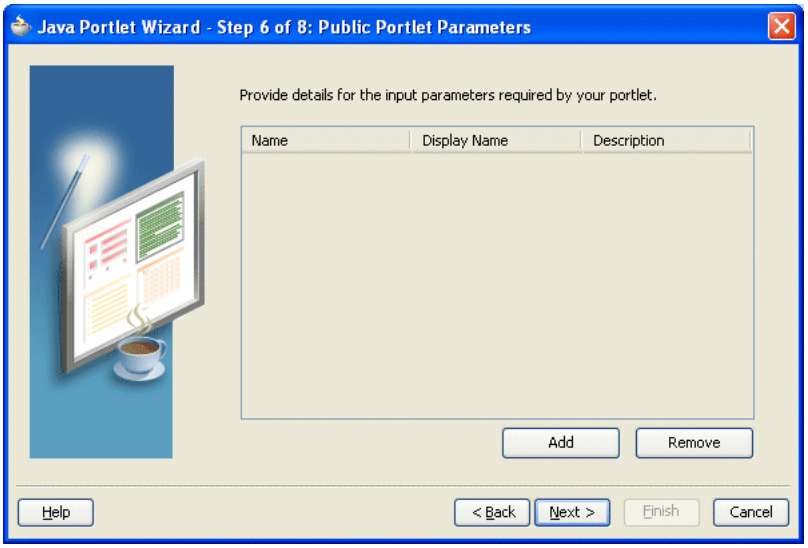
    For more information about Help mode, see Section 18.1.1.6, "Help Mode (JPS and PDK-Java)".

23. If you want to implement About mode for your portlet, select **About page**, then select an Implementation style as described earlier for Show page.

    For more information about About mode, see Section 18.1.1.7, "About Mode (JPS and PDK-Java)".

24. Click **Next** to display the Public Portlet Parameters page (Figure 18–25).

Figure 18–25 Public Portlet Parameters Page



25. If you want to add public parameters to your portlet, then click **Add** to create a blank row for entering parameter details. For more information about using parameters, see Section 19.2.2, "Passing Parameters and Submitting Events".

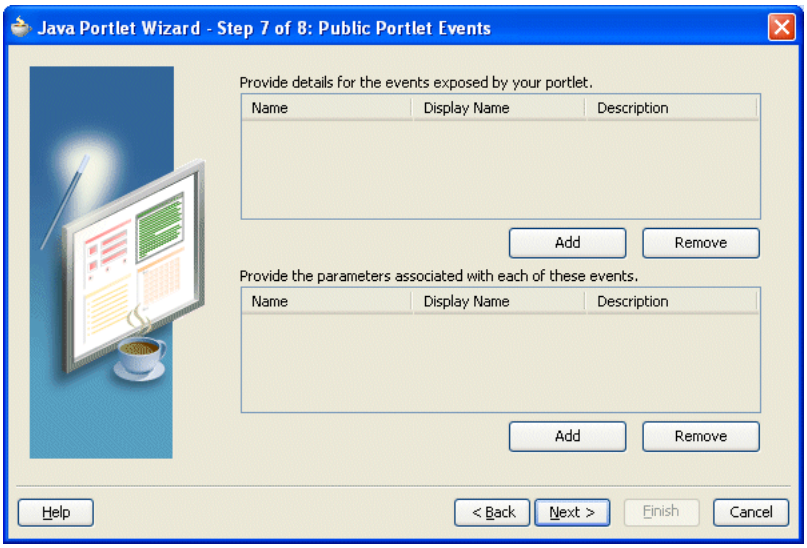
26. In the Name field, enter an internal name for the parameter, for example, `MyParam`.

27. In the Display Name field, enter a name to display to users, for example, `My Portlet Parameter`.

28. In the Description field, enter descriptive information about the parameter

29. Click **Next** to display the Public Portlet Events page (Figure 18–26).

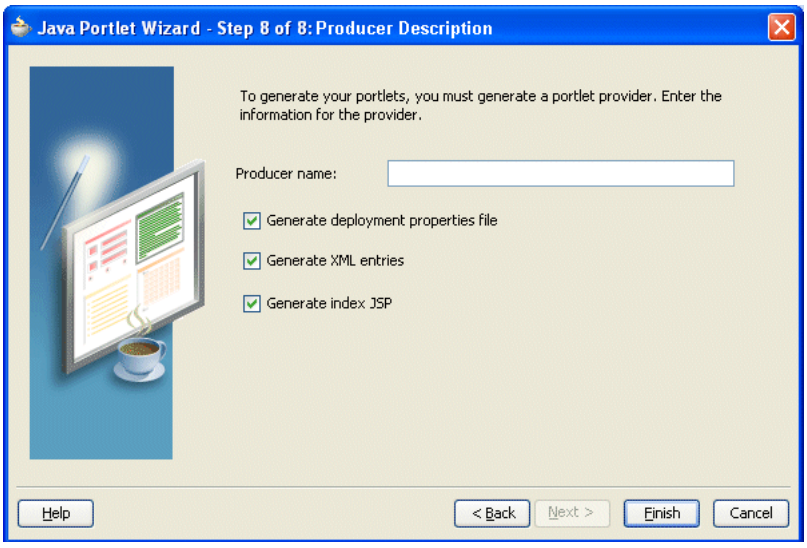
Figure 18–26 Public Portlet Events Page



30. For more information about using events, see Section 19.2.2, "Passing Parameters and Submitting Events".

31. Click **Next** to display the Producer Description page (Figure 18–27).

Figure 18–27 Provider Description Page



32. In the Producer name field, enter a name for the producer.

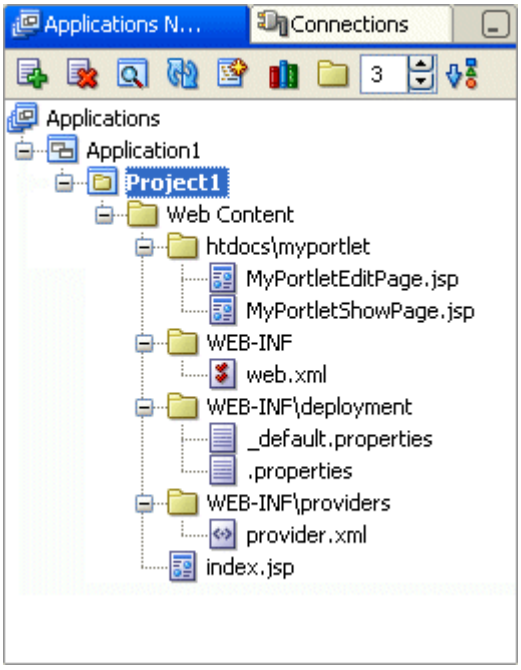
33. Select Generate deployment properties files to generate two `.properties` files:

    - *<serviceID>*`.properties` defines properties for a producer with that service ID. The service ID has the same value as the producer name.
    - `_default.properties` is a default properties file. A producer Web application may have more than one producer, each with its own service ID. On registration, if no service ID is defined, then the default properties file is used.

34. Select Generate XML entries to automatically generate a `provider.xml` file for the producer. This is the producer definition file that contains details of the portlets, including those generated by the wizard, belonging to the producer.
35. Select Generate index JSP to automatically generate an index.jsp file. This file lists all the producers that reside in your Web application with hyperlinks that enable easy access to producer test pages.
36. Click **Finish** to generate the files for your portlet. The following files should be generated for your project in the Application Navigator (see Figure 18–28):
    - Files for each portlet mode you selected
    - `provider.xml`
    - `web.xml`
    - `index.jsp`
    - `_default.properties`
    - `<serviceID>.properties`

    All these files are required to deploy and run the portlet successfully, except for `index.jsp`, which is used by Oracle JDeveloper for testing purposes.

Figure 18–28 Application Navigator



18.8 Adding Portlet Logic

After you create the default implementation of your portlet, you can extend the sample code with your own business logic to implement the desired functionality and features. See the JPS or JavaDoc for more information about adding functionality and features. For a specific example, see Step 7: Adding Some Simple Logic to the Portlet in the Building and Testing Your First Portlet chapter of *Oracle Oracle WebCenter Framework Tutorial*.

18.9 Deploying Your Portlet to an Application Server

After you finish the wizard and successfully generate your portlet, you are ready to deploy it to an application server.

If you chose to create a JPS-compliant portlet, then you can deploy it using the wizard for any vendor's JPS-compliant container. The steps in Section 18.9.1, "Deploying Your JPS-compliant WebCenter Application Portlet" show how to deploy a JPS-compliant portlet to Oracle's WSRP container running on OC4J. If you created a PDK-Java portlet, then you can deploy it to OC4J.

> **Note:** The WSRP portlets must be deployed to the OC4J_Webcenter instance in Oracle Application Server and not to the home instance.

This section includes the following:

- Section 18.9.1, "Deploying Your JPS-compliant WebCenter Application Portlet"
- Section 18.9.2, "Deploying Your PDK-Java Portlet"
- Section 18.9.3, "Deploying a Third-Party JPS-compliant Portlet"
- Section 18.9.4, "Validating Your JPS-Compliant Portlet and Producer"
- Section 18.9.5, "Validating Your PDK-Java Portlet and Producer"

18.9.1 Deploying Your JPS-compliant WebCenter Application Portlet

This section describes the procedure to deploy a JPS-compliant WebCenter application portlet that you create using Oracle JDeveloper. You need to create a WAR deployment profile and deploy it to OC4J.

To create and deploy a WAR file, perform the following steps:

1. Make sure that a connection exists to your preconfigured OC4J instance. To start preconfigured OC4J, go to Tools and select **Start WebCenter Preconfigured OC4J**.

   > **Note:** If you want to use a standalone OC4J instance, then see Section 12.2.6.2.1, "Defining Standalone OC4J Connection Details" to learn how to create a connection to it.

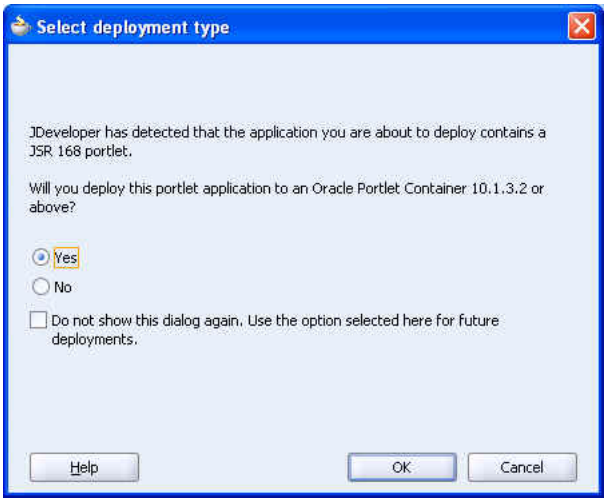
   To stop the preconfigured OC4J instance, go to Tools and select **Stop WebCenter Preconfigured OC4J**.
2. Connect to the preconfigured OC4J as described in the `readme` file. To open this file, go to Help and select **Open WebCenter Preconfigured OC4J readme**.
3. In the Applications Navigator, right-click the project that contains your portlet, and select **New**.
4. In the New Gallery, expand the General category and select **Deployment Profiles**.
5. In the Items list, select **WAR File**.
6. Click **OK** to display the Create Deployment Profile dialog box.
7. In the Deployment Profile Name field, enter a meaningful name for the deployment profile.

8. In the Directory Name field, enter a location for the deployment profile. You can accept the default location or specify your own.
9. Click **OK**. This creates your deployment profile.
10. In the Application Navigator, expand the **Resources** node and double-click your `.deploy` file to display the WAR Deployment Profile Properties dialog box.
11. Under Web Application's Context Root, select **Specify J2EE Web Context Root** and enter the J2EE context root in the corresponding field.
12. Click **OK**.
13. Right-click the newly created deployment profile, select **Deploy to**, then select the application server connection to which you want to deploy the portlet.

    The Select deployment type dialog box is displayed (Figure 18–29).

> **Note:** The Select deployment type dialog box displays only if you selected the Deploy to EAR option earlier. Select **Yes** if you are deploying to an Oracle Application Server 10.1.3.2.0 instance and select **No** if you are deploying to an Oracle Application Server 10.1.2.0.2 instance.

Figure 18–29 Select Deployment Type



The Oracle Portlet Container exposes JSR 168 portlets through the WSRP protocol. To make portlets accessible as a Web service, the portlet application must contain a Web service descriptor document, that is, a WSDL document. This dialog box is used to indicate that a Web Service Descriptor document (WSDL document) should be injected into the portlet application when the application is packaged as an EAR file.

Select Yes or No depending on your portlet container type and version, as follows:

- Oracle Portlet Container (Oracle Application Server or OC4J) version 10.1.3.1, 10.1.3.2: Select Yes to inject the WSDL document into your portlet application.
- Oracle Portlet Container version 10.1.2, 10.1.2.0.2: Select No to forgo injecting the WSDL document into your portlet application. The portlet container provided with Oracle Application Server version 10.1.3 includes a deployment

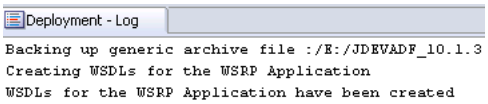
servlet that automatically injects the WSDL document into the deployed application.

- Third-party portlet containers: Select No to forgo injecting the WSDL document into your portlet application.

Select **Do not show this dialog again. Use the option selected here for future deployments** option to prevent future displays of the Select deployment type dialog box. The previous options (Yes or No) are used on future deployments of JSR 168 portlets from this application. To change this setting, delete the deployment profile and re-create. When the Select Deployment Type dialog box opens, select your preferred setting.

14. To insert the WSDL document into your portlet application, select the **Yes** option, if it is not selected already, and click **OK**. The Deployment Log window displays the application packaging and deployment status (Figure 18–30).

Figure 18–30 Deployment Log



15. In the Configure Application dialog box, you can enter settings that are specific to the container configuration files (for example, `orion-web.xml`). In most cases, you can just accept the default values by clicking **OK**.

16. When the Deployment finished message displays in the Deployment Log at the bottom of Oracle JDeveloper, verify that no errors occurred. You can, however, safely ignore the following error message:

```
006-11-30 08:14:25.627 WARNING J2EE DEP-00002 There are multiple mappings for
servlet name: WSRPServiceDescriptionService, but only the first one
(/portlets/WSRPServiceDescriptionService) will be considered for webservice
endpoint.
```

When you redeploy your portlets to the portlet container, all existing sessions between the producer and all of its consumers are lost. If a consumer tries to reuse an existing producer session, then it may receive an error message the first time it tries to contact the producer after redeployment.

```
Error: Could not get markup. The cookie or session is invalid or there is a
runtime exception.
```

To reestablish the producer's session, refresh the page. You won't see this error message if you are re-accessing the portlet from a new browser session because it automatically establishes a new producer session.

18.9.2 Deploying Your PDK-Java Portlet

This section describes the procedure to deploy a PDK-Java portlet that you create using Oracle JDeveloper. You need to create a WAR deployment profile and deploy it to OC4J.

To create and deploy a WAR file, perform the following steps:

1. Make sure that a connection exists to your preconfigured OC4J instance. To start preconfigured OC4J, go to Tools and select **Start WebCenter Preconfigured OC4J**.

> **Note:** If you want to use a standalone OC4J instance, then see Section 12.2.6.2.1, "Defining Standalone OC4J Connection Details" to learn how to create a connection to it.

To stop the preconfigured OC4J instance, go to Tools and select **Stop WebCenter Preconfigured OC4J**.

2. Connect to the preconfigured OC4J as described in the `readme` file. To open this file, go to Help and select **Open WebCenter Preconfigured OC4J readme**.
3. In the Applications Navigator, right-click the project that contains your portlet, and select **New**.
4. In the Categories list, expand the General category and click **Deployment Profiles**.
5. In the Items list, click **WAR File**.
6. Click **OK**.
7. In the Create Deployment Profile dialog box, change the name to something meaningful (for example, `myj2eeportlet1.deploy`).
8. Click **OK**.
9. In the WAR Deployment Profile Properties dialog box, perform the following steps:
   a. Click **Specify J2EE Web Context Root** and enter `myj2eeportlet1`.
   b. In the pane on the left, select **Contributors**.
   c. Select **Portlet Development**.
   d. Click **OK**.
10. Select **File**, and **Save All**.
11. Expand the Resources node, right-click the deployment profile (for example, `myj2eeportlet1.deploy`), select **Deploy to**, and then select the application server connection (for example, `PDKJavaOC4J`).
12. In the Configure Application dialog, you can enter settings that are specific to the container configuration files (for example, `orion-web.xml`). In most cases, you can just accept the default values by clicking **OK**.
13. When the Deployment Finished message displays in the Deployment Log at the bottom of Oracle JDeveloper, verify that no errors have occurred.
14. Construct the URL you need to test and register your portlet as follows:

    `http://`*`host:port/context-root`*`/providers`

    where *`host`* is the server to which your provider has been deployed.

    `port` is the OracleAS Web Cache HTTP Listener port from the Ports tab of the Application Server Control Console main page.

    `context-root` is the Web Application's Context Root, which is found in the WAR Deployment Profile Properties under General.
15. Enter the URL you constructed in the preceding step in your browser to ensure the successful deployment of the portlet.

Note: If you are deploying your portlets to standalone OC4J or a third party container, then for the shared library, you must copy the JARs by using any of the following methods:

- Copy the JARs from the 10.1.3.2 WebCenter installation.
- Download the preconfigured OC4J from the Portal PDK page on OTN, that will have these JARs in the shared library.
- Get these JARs from the 10.1.3.2 JDeveloper Studio Edition.

If you are using the preconfigured OC4J in the WebCenter instance of an Application Server 10.1.3.2 installation, then the shared library is already available.

It is referred to as `oracle.portal`.

18.9.3 Deploying a Third-Party JPS-compliant Portlet

This section describes the procedure to deploy a third-party JPS-compliant portlet to OC4J.

To deploy a third-party portlet, perform the following steps:

1. Obtain the third-party JSR 168 portlet producer EAR file.
2. Convert a JSR 168 portlet producer EAR file into a WSRP EAR file: Conversion steps are covered in Converting a JSR 168 Portlet Producer EAR File into a WSRP EAR File.
3. Deploy the WSRP EAR file: The procedure to deploy a WSRP EAR file is similar to the procedure covered in Section 12.2.3, "Deploying Your WebCenter Application Using Application Server Control Console".

Converting a JSR 168 Portlet Producer EAR File into a WSRP EAR File

If deploying JSR-168 portlets to the WSRP Oracle Portlet Container the portlet application EAR files must be converted into a WSRP application containing the necessary WSDL documents. To convert the JSR 168 portlet producer EAR file into a WSRP EAR file, run the WSRP producer predeployment tool located in the *JDEV_HOME*`/adfp/lib` directory, as follows:

```
java -jar wsrp-predeploy.jar <source EAR> <targeted EAR>
```

The `wsrp-predeploy.jar` is also available at *ORACLE_HOME*`/j2ee/OC4J_WebCenter/shared-lib/oracle.wsrp/1.0`.

For JPS-compliant portlets developed with servlet version 2.3, you must specify Web proxies using the following command:

```
java -Dhttp.proxyHost=<proxy host> -Dhttp.proxyPort=<proxy port> -jar
wsrp-predeploy.jar <Source EAR file> <Targeted EAR file>
```

where:

`proxy host` is the server to which your producer has been deployed.

`proxy port` is the HTTP Listener port.

`wsrp-predeploy.jar` is located in the *JDEV_HOME*`/adfp/lib` or *ORACLE_HOME*`/j2ee/OC4J_Webcenter/shared-lib/oracle.wsrp/1.0` directory.

The `source EAR` file is the name of the JSR 168 EAR file.

The `targeted EAR` file is the name of the new EAR file to be created. If the file name for the targeted EAR file is not specified, then a new EAR file called `WSRP-<source EAR>` is produced.

In the following example Web proxy is specified:

```
java -Dhttp.proxyHost=myhttpproxy.com -Dhttp.proxyPort=80 -jar wsrp-predeploy.jar
wsrp-samples.ear
```

This example produces `WSRP-wsrp-samples.ear`.

18.9.4 Validating Your JPS-Compliant Portlet and Producer

You can test your JPS-Compliant portlet and producer to check the configuration and to ensure that the portlet and its producer operate correctly. To test your WSRP portlet and producer, run the producer URL in your browser in the following syntax:

`http://<host>:<port>/<context-root>/info`

where:

- `host` is the server to which your producer has been deployed.
- `port` is the HTTP Listener port (set in the `default-web-site.xml` Web site configuration file created for the instance.).
- `context-root` is the Web application's context root.

A test page similar to Figure 18–31 is displayed.

Figure 18–31 WSRP Producer Test Page

Note: This procedure is for testing purposes only. After this procedure, you still need to register your producer as described in Section 18.10, "Registering and Viewing Your Portlet".

To disable your test page, see the procedure in "Disabling a WSRP Test Page".

18.9.5 Validating Your PDK-Java Portlet and Producer

After you have deployed your portlet, you need to check the configuration to ensure that the portlet and its producer operate correctly.

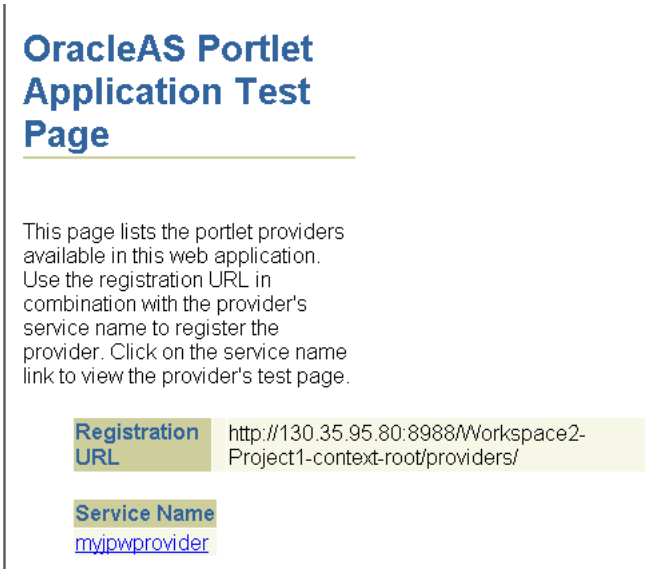
> **Note:** This procedure is for testing purposes only. After this procedure, you still need to register your producer as described in Section 18.10, "Registering and Viewing Your Portlet".

To validate your portlet and producer, perform the following steps:

1. In the Applications Navigator, under the project that contains your portlet, right-click `index.jsp`, and select **Run**.

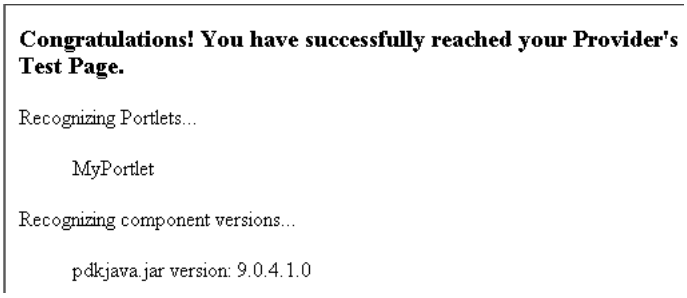
   Your browser should open with a page similar to the one shown in Figure 18–32.

Figure 18–32 Portlet Application Test Page



2. Click the link underneath Service Name. Your browser should open with a page similar to the one shown in Figure 18–33. Note that you need the URL from this page to register your producer, which is the next task.

Figure 18–33 Producer Test Page



Alternatively, enter the URL for your portlet to check that it is working. The URL is constructed as follows:

```
http://host:port/context-root/providers/producer_name
```

where:

- *host* is the server to which your producer has been deployed.
- *port* is the HTTP Listener port (set in the `default-web-site.xml` Web site configuration file created for the instance.).
- *context-root* is the Web Application's Context Root, which you specified earlier and can be found in the WAR Deployment Profile Properties under General.
- *producer_name* is the name of the portlet's producer. A WAR file may contain more than one producer, hence you should always include the name of the producer for clarity. Otherwise, you will get the default producer, which is simply the last producer created.

You should see a page similar to the one in Figure 18–33.

18.10 Registering and Viewing Your Portlet

After you've created and deployed the producer and its portlets, you should register the producer with an application and add it to a page to check that it is working correctly. Registering a producer gives applications the information they need to locate and communicate with that producer. After you register a producer, the producer and its portlets become available in the Component Palette.

To register producers for your standards-based portlets, follow the instructions provided in Section 4.3.1.1, "Registering WSRP Portlet Producers".

To register producers for your PDK-Java portlets, follow the instructions provided in Section 4.3.1.2, "Registering PDK-Java Portlet Producers".

To add your portlets to a page, follow the instructions provided in Section 4.3.2, "Adding Portlets to a Page".

19

Enhancing Java Portlets

This chapter explains how to enhance Java portlets you created with the Oracle JDeveloper Portlet Wizard, and how to make portlets out of your struts and JSF applications. This chapter contains the following sections:

- Section 19.1, "Enhancing Java Portlet Specification (JPS) Portlets"
- Section 19.2, "Enhancing PDK-Java Portlets"
- Section 19.3, "Testing Portlet Personalization"
- Section 19.4, "Building Struts Portlets"
- Section 19.5, "Building Portlets from Oracle ADF Faces Applications (JSF Portlet Bridge)"

The source code for many of the examples referenced in this chapter is available as part of the Portlet Developer's Kit (PDK). You can download the PDK from it's page on Oracle Technology Network (OTN):

`http://www.oracle.com/technology/products/ias/portal/pdk.html`

When you unzip PDK-Java, you will find the examples in:

`../pdk/jpdk/v2/src/oracle/portal/sample/v2/devguide`

You can find the JavaDoc reference for PDK-Java in:

`../pdk/jpdk/v2/apidoc`

19.1 Enhancing Java Portlet Specification (JPS) Portlets

Once you have built your initial portlet in the Portlet Wizard as described in Section 18.5, "Building JPS-Compliant Portlets with Oracle JDeveloper", you will want to enhance it. Because JPS portlets adhere to the Java standards, you can find substantial information about enhancing them from many different sources, such as third-party books and Web pages. Some of the more important enhancements that you might want to perform are as follows:

- Section 19.1.1, "Adding Personalization"
- Section 19.1.2, "Implementing Navigational Parameters (WSRP 2.0)"
- Section 19.1.3, "Implementing Export/Import of Customizations (WSRP 2.0)"
- Section 19.1.4, "Implementing Rewritten URLs for Resource Proxy"
- Section 19.1.5, "Implementing Security for JPS Portlets"

19.1.1 Adding Personalization

In this section, you enhance the portlet you created in Section 18.5, "Building JPS-Compliant Portlets with Oracle JDeveloper" with some code that enables a user in Edit or Edit Defaults mode to paste HTML into a field for the portlet to render. You will also see how easily you can redeploy a portlet.

19.1.1.1 Assumptions

The following assumptions are made to perform the tasks in this section:

- You built a portlet using the wizard, successfully registered the producer, and added the portlet to the page.
- In the wizard, on the Customization Preferences page, you added a preference called `portletcontent`.
- You enabled Oracle ADF security for your application by performing the steps in Section 10.2, "Setting Up Security for Your Application". The user must be logged in to the application as an authenticated user to access the personalization feature.

19.1.1.2 Implementing Personalization

In this section, you add some code to My Java Portlet, redeploy the portlet, and then test it. To do this, perform the following steps:

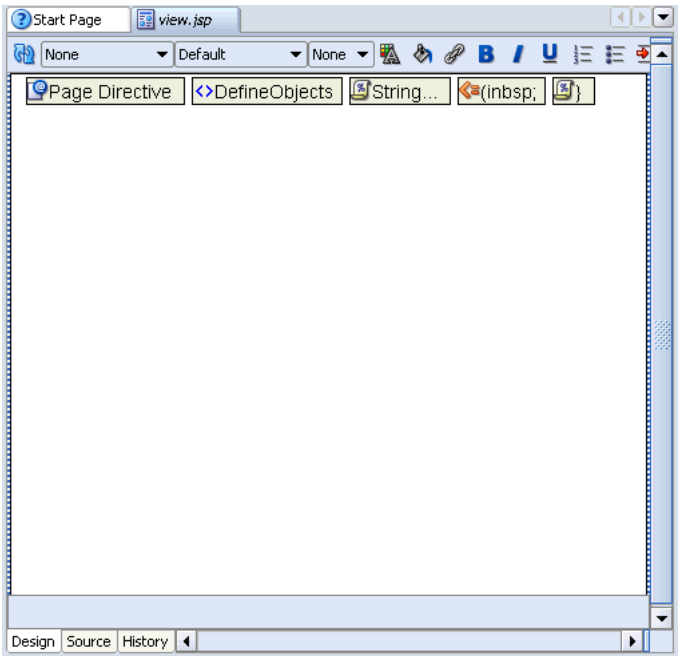
1. In Oracle JDeveloper, double-click the `view.jsp` file for your JPS-Standard portlet in the Application Navigator.
2. Add the code in Example 19–1 to the Source view of your `view.jsp`:

Example 19–1 view.jsp Sample Code

```
<%@ page contentType="text/html"
    import="javax.portlet.*,java.util.*,Portlets.Portlet1,
    Portlets.resource.Portlet1Bundle"%>
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet"%>
<portlet:defineObjects/>
<%
String[] str = {"Portlet Content"};
PortletPreferences prefs = renderRequest.getPreferences();
str = prefs.getValues("portletContent",str);
for (int i=0; i<str.length; i++)
{
%><%=(i<str.length-1)?str[i]+", ":str[i]%><%}%>
```

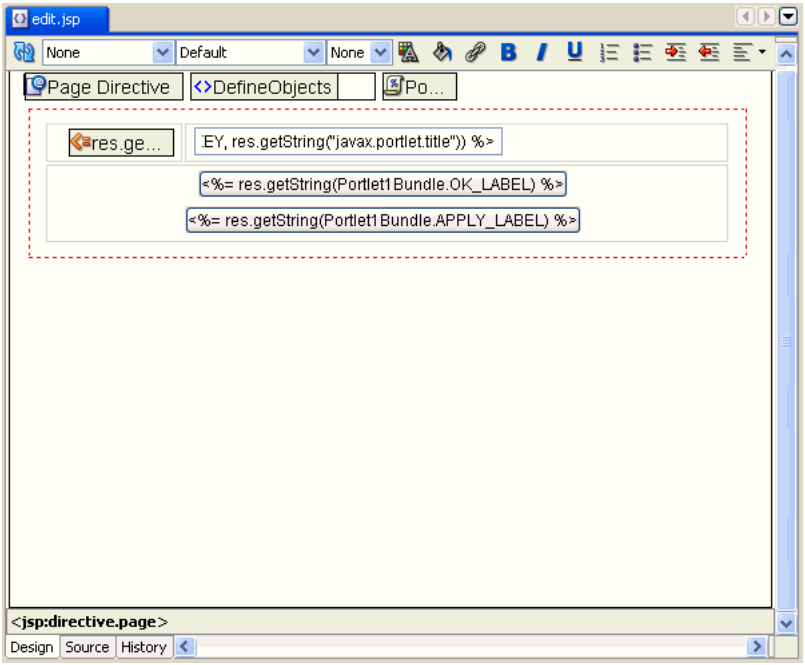
3. Now click the **Design** tab. Your page should look something like Figure 19–1.

Figure 19–1 view.jsp in the Design View



4. Open `edit.jsp` in the visual designer and click the **Design** tab. Notice that the JSP consists of a form field, a form input field, and two form button fields, as shown in Figure 19–2.

Figure 19–2 edit.jsp in the Design View



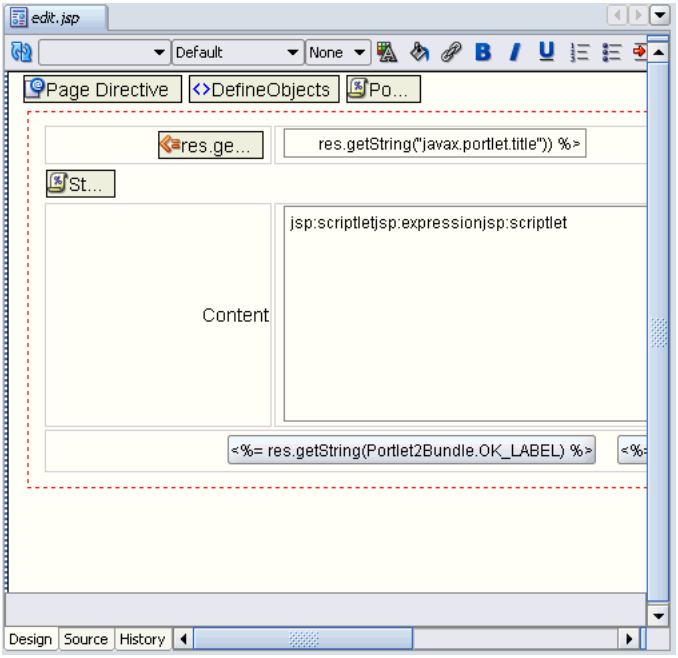
5. Add the code that is indicated in bold in the following code excerpt to implement a form field called **Content**:

Example 19–2 edit.jsp Sample Code

```
<%@ page contentType="text/html"
    pageEncoding="windows-1252"
    import="javax.portlet.*,java.util.*,
    portlet.Portlet2,portlet.resource.Portlet2Bundle"%>
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet"%>
<portlet:defineObjects/>
<%
    PortletPreferences prefs = renderRequest.getPreferences();
    ResourceBundle res =
        portletConfig.getResourceBundle(renderRequest.getLocale());
%>
<FORM ACTION="<portlet:actionURL/>" METHOD="POST">
<TABLE BORDER="0">
<TR><TD WIDTH="20%">
<P CLASS="portlet-form-field" ALIGN="right">
<%= res.getString(Portlet2Bundle.PORTLETTITLE) %>
</P></TD><TD WIDTH="80%">
<INPUT CLASS="portlet-form-input-field" TYPE="TEXT"
  NAME="<%= Portlet2.PORTLETTITLE_KEY %>"
  VALUE="<%= prefs.getValue(Portlet2.PORTLETTITLE_KEY,
             res.getString("javax.portlet.title")) %>"
  SIZE="20">
</TD></TR>
<%
    String[] str = {"Portlet Content"};
    str = prefs.getValues("portletContent",str);
%>
<tr><td width="20%">
<p class="portlet-form-field" align="right">
Content
</p>
</td><td width="80%">
<textarea rows="10" cols="60" class="portlet-form-input-field"
  name="portletContent"><%
for (int i=0; i<str.length; i++)
{%><%= (i<str.length-1) ? str[i]+", " : str[i] %><%}%>
</textarea>
</td></tr>
<TR><TD COLSPAN="2" ALIGN="CENTER">
<INPUT CLASS="portlet-form-button" TYPE="SUBMIT" NAME="<%= Portlet2.OK_ACTION
%>"
  VALUE="<%= res.getString(Portlet2Bundle.OK_LABEL) %>">
<INPUT CLASS="portlet-form-button" TYPE="SUBMIT" NAME="<%= Portlet2.APPLY
ACTION %>"
  VALUE="<%= res.getString(Portlet2Bundle.APPLY_LABEL) %>">
</TD></TR>
</TABLE>
</FORM>
```

6. Click the **Design** tab. Your page should now look similar to Figure 19–3.

Figure 19–3 Modified edit.jsp in the Design View



7. Open `WelcomePortlet.java` in the visual editor and insert the following two lines of code (indicated in bold) in the `processAction` method:

```
// Save the preferences.
PortletPreferences prefs = request.getPreferences();
String param = request.getParameter(PORTLETTITLE_KEY);
prefs.setValues(PORTLETTITLE_KEY, buildValueArray(param));
String contentParam = request.getParameter("portletContent");
prefs.setValues("portletContent", buildValueArray(contentParam));
prefs.store();
```

8. Redeploy the portlet. Notice that Oracle JDeveloper automatically saves and compiles the code before deploying the portlet. See Section 18.9, "Deploying Your Portlet to an Application Server" for a reminder of how to perform this step.
9. Reload the page that contains the portlet. The portlet displays the text Portlet Content, which was one of the changes you made in Oracle JDeveloper.
10. Click the **Personalize** link. You can see the new form field that you added in Oracle JDeveloper.
11. Enter the following HTML in the Content field, replacing the words Portlet Content.

```
<p>Read <em>The Path to Portlet Interoperability</em> by John Edwards in
<strong>Oracle Magazine</strong>, Nov-Dec 2003. </p>
<p>It discusses JSR 168 and WSRP open portals. </p>
```

12. Click **Apply** and then click **Close**. The HTML is rendered in the portlet.

19.1.2 Implementing Navigational Parameters (WSRP 2.0)

While JSR 168 and WSRP 1.0 do not address public portlet parameters, WSRP 2.0 introduces navigational parameters to enable inter-portlet communication. With the release of the new portlet Application Programming Interface (API) standard, JSR 286,

the need for vendor-specific API extensions will not be necessary any more. Until then, you are required to use the Oracle-specific portlet container and API extensions.

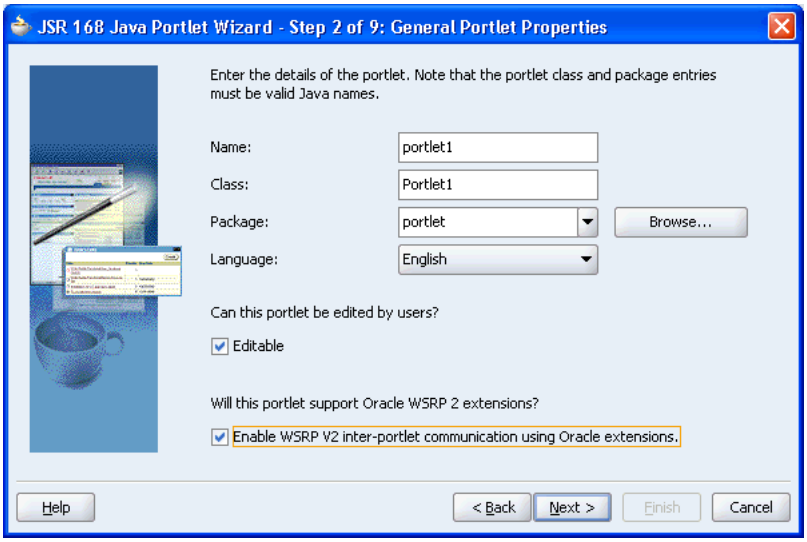
Using the Portlet Wizard, you can easily create a portlet with navigational parameters. When you register the producer and drop the portlet on a page, the portlet's parameters are automatically linked to page variables.

Adding Public Parameters to JPS Portlets Using WSRP 2.0 Navigational Parameters

To add parameters to your portlet, perform the following steps:

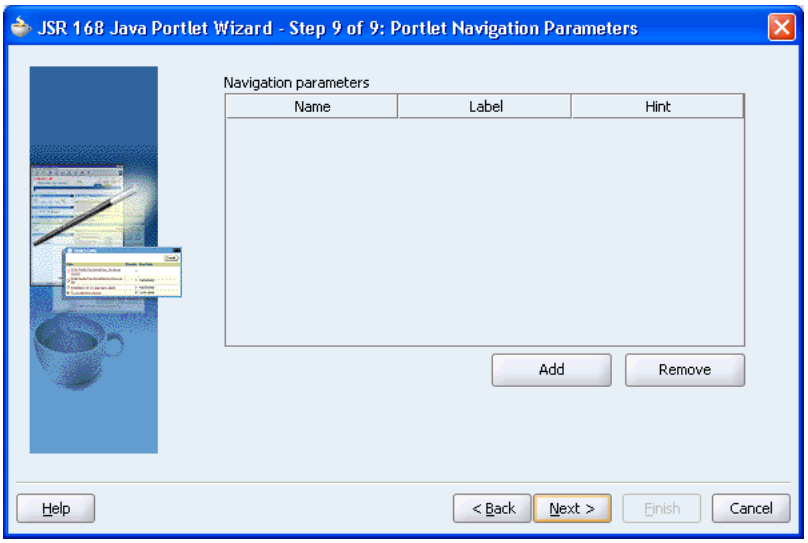
1. Right-click your project and choose **New** from the context menu.
2. From New Galleries, choose the **Portlets** category and then the **Standards-based Java Portlet (JSR 168)** item.
3. When you reach the General Properties page of the wizard, be sure to select the **Enable WSRP V2 inter-portlet communication using Oracle extensions** check box as shown in Figure 19–4.

Figure 19–4 General Portlet Properties Page in Portlet Wizard



4. Proceed through the wizard as you normally would until you reach the Portlet Navigation Parameters page. See Section 18.5, "Building JPS-Compliant Portlets with Oracle JDeveloper" for basic information about going through the wizard.
5. When you reach the Portlet Navigation Parameters page (Figure 19–5), click the **Add** button. A new row is added to the **Navigation parameters** table.

Figure 19–5 Portlet Navigation Parameters Page of Portlet Wizard



6. Click the default name that appears in the **Name** column. Use the Backspace key to delete the default name.
7. Enter a new parameter name, for example, `Parameter_01`.
8. Repeat steps 6 and 7 for the **Label** and **Hint** columns.
9. Repeat steps 5 through 8 for each public parameter that you want to add to the portlet.
10. Click **Next**.
11. Click **Finish**.
12. Once you finish the Portlet Wizard, your portlet is created.
13. In the Applications Navigator, expand the Web Content and WEB-INF nodes. You should see your `portlet.xml` and `web.xml` and `oracle-portlet.xml` files there.
14. Right-click `oracle-portlet.xml` and choose **Open** from the context menu. You should see entries for the parameters that you added on the Portlet Navigation Parameters page in the wizard, as shown in Example 19–3.

Example 19–3 oracle-portlet.xml Sample, Navigational Parameters

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<portlet-app-extension
 xsi:schemaLocation="http://xmlns.oracle.com/portlet/oracle-portlet-app"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://xmlns.oracle.com/portlet/oracle-portlet-app">
    <portlet-extension>
        <portlet-name>portlet1</portlet-name>
        <navigation-parameters>
            <name>Parameter_01</name>
            <type>xsi:string</type>
            <label xml:lang="en">Parameter 1</label>
            <hint xml:lang="en">First parameter.</hint>
        </navigation-parameters>
        <navigation-parameters>
            <name>Parameter_02</name>
            <type>xsi:string</type>
```

```
            <label xml:lang="en">Parameter 2</label>
            <hint xml:lang="en">Second parameter.</hint>
        </navigation-parameters>
        <portlet-id>1164232649525</portlet-id>
    </portlet-extension>
</portlet-app-extension>
```

15. The sample portlet in Example 19–4 shows you a portlet that contains a form and submits navigation parameters.

Example 19–4 Form Portlet Submitting Parameters

```
/* Copyright (c) 2006, Oracle. All rights reserved.  */
package oracle.portlet.server.wsrpsample;
import java.io.IOException;
import java.io.PrintWriter;
import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;
import javax.portlet.PortletException;
import javax.portlet.PortletMode;
import javax.portlet.PortletPreferences;
import javax.portlet.PortletSession;
import javax.portlet.PortletURL;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import javax.portlet.WindowState;

public class ParameterFormPortlet extends CustomizablePortlet
{
  // Form field names
  public static final String PARAMETER1  = "ora_wsrp_navigparam_Parameter1";
  public static final String PARAMETER2  = "ora_wsrp_navigparam_Parameter2";
  public static final String PARAMETER3  = "ora_wsrp_navigparam_Parameter3";

  public static final String FORM_PARAMETER1  = "form_Parameter1";
  public static final String FORM_PARAMETER2  = "form_Parameter2";
  public static final String FORM_PARAMETER3  = "form_Parameter3";
  public static final String FORM_SUBMIT      = "dosub";
  public static final String P1PROMPT_KEY      = "p1Prompt";
  public static final String P2PROMPT_KEY      = "p2Prompt";
  public static final String P3PROMPT_KEY      = "p3Prompt";

  public static final String   MODE_NAME_PARAM  = "mode";
  public static final String   MODE_VIEW         = "view";
  public static final String   MODE_EDITDEFAULTS = "editDefaults";
  /**
   * Helper method to serve up the VIEW mode.
   *
   * @param     request the portlet request
   * @param     response the portlet response
   *
   * @exception PortletException if the portlet has trouble fulfilling the
   *            request
   * @exception IOException if the streaming causes an I/O problem
   */
  public void doView (RenderRequest request,
                      RenderResponse response)
      throws PortletException, IOException
  {
```

```
String p1Prompt = request.getPreferences().getValue(P1PROMPT_KEY,
     "Parameter 1:");
String p2Prompt = request.getPreferences().getValue(P2PROMPT_KEY,
     "Parameter 2:");
String p3Prompt = request.getPreferences().getValue(P3PROMPT_KEY,
     "Parameter 3:");

response.setContentType("text/html; charset=UTF-8");
PrintWriter out = response.getWriter();
PortletURL formURL = response.createActionURL();

//Retrieve any values for the nav parameters from the request
String param1 = request.getParameter(PARAMETER1);
String param2 = request.getParameter(PARAMETER2);
String param3 = request.getParameter(PARAMETER3);
param1 = (param1 != null)?param1:"";
param2 = (param2 != null)?param2:"";
param3 = (param3 != null)?param3:"";

out.print("<form method=\"POST\" action=\"");
out.print(formURL.toString());
out.println("\">");

out.print("<center><table>");
if (!"".equals(p1Prompt))
{
  out.print("<tr><td class=\"portlet-form-field\">");
  out.print(p1Prompt);
  out.print("</td><td>");
  out.print("<input  class=\"portlet-form-input-field\"
        type=\"text\" size=\"20\" name=\"");
  out.print(FORM_PARAMETER1);
  out.print("\" value=\"");
  out.print(escapeForHTMLAttr(param1));
  out.print("\">");
  out.print("</td></tr>");
}
else
{
  out.print("<input type=\"hidden\" name=\"");
  out.print(FORM_PARAMETER1);
  out.print("\" value=\"\">");
}
if (!"".equals(p2Prompt))
{
  out.print("<tr><td class=\"portlet-form-field\">");
  out.print(p2Prompt);
  out.print("</td><td>");
  out.print("<input class=\"portlet-form-input-field\"
        type=\"text\" size=\"20\" name=\"");
  out.print(FORM_PARAMETER2);
  out.print("\" value=\"");
  out.print(escapeForHTMLAttr(param2));
  out.print("\">");
  out.print("</td></tr>");
}
else
{
  out.print("<input type=\"hidden\" name=\"");
  out.print(FORM_PARAMETER2);
```

```
      out.print("\" value=\"\">");
    }
    if (!"".equals(p3Prompt))
    {
      out.print("<tr><td class=\"portlet-form-field\">");
      out.print(p3Prompt);
      out.print("</td><td>");
      out.print("<input class=\"portlet-form-input-field\"
            type=\"text\" size=\"20\" name=\"");
      out.print(FORM_PARAMETER3);
      out.print("\" value=\"");
      out.print(escapeForHTMLAttr(param3));
      out.print("\">");
      out.print("</td></tr>");
    }
    else
    {
      out.print("<input type=\"hidden\" name=\"");
      out.print(FORM_PARAMETER3);
      out.print("\" value=\"\">");
    }
    out.print("<tr><td colspan=\"2\" align=\"center\">");

    out.print("<input type=\"submit\" class=\"portlet-form-button\" name=\"");
    out.print(FORM_SUBMIT);
    out.print("\" value=\"OK\">");
    out.print("</td></tr></table>");

    out.print("<input type=\"hidden\" name=\"");
    out.print(MODE_NAME_PARAM);
    out.print("\" value=\"");
    out.print(MODE_VIEW);
    out.print("\">");

    out.print("</form>");
    out.println("</center>");
}

/**
 * Handles actions.
 *
 * @param     request the portlet request
 * @param     actionResponse the portlet response
 *
 * @exception IOException if streaming causes an I/O problem
 * @exception PortletException if something else goes wrong
 */
public void processAction (ActionRequest request,
                           ActionResponse actionResponse)
    throws PortletException, IOException
{
  //
  // Determine what kind of action we have by examining the mode parameter
  //
  boolean editDefaults = MODE_EDITDEFAULTS.equals(request.getParameter
      (MODE_NAME_PARAM));
  boolean viewMode = MODE_VIEW.equals(request.getParameter(MODE_NAME_PARAM));
  String param1 = request.getParameter(FORM_PARAMETER1);
  String param2 = request.getParameter(FORM_PARAMETER2);
  String param3 = request.getParameter(FORM_PARAMETER3);
```

```
  String title = request.getParameter(TITLE_KEY);
  String p1Prompt = request.getParameter(P1PROMPT_KEY);
  String p2Prompt = request.getParameter(P2PROMPT_KEY);
  String p3Prompt = request.getParameter(P3PROMPT_KEY);

  if (viewMode)
  {
    //
    // Set the new parameter values. These will be intepreted by the
    // container as navigational parameters as the names match the names of
    // the declared parameters.
    //
    actionResponse.setRenderParameter(PARAMETER1, param1);
    actionResponse.setRenderParameter(PARAMETER2, param2);
    actionResponse.setRenderParameter(PARAMETER3, param3);
  }
  else if (editDefaults)
  {
    //
    // In edit defaults we just set the new title and parameter prompts in
    // the preference store.
    //
    PortletPreferences prefs = request.getPreferences();
    prefs.setValue(TITLE_KEY, title);
    prefs.setValue(P1PROMPT_KEY, p1Prompt);
    prefs.setValue(P2PROMPT_KEY, p2Prompt);
    prefs.setValue(P3PROMPT_KEY, p3Prompt);
    prefs.store();

    actionResponse.setPortletMode(PortletMode.VIEW);
    actionResponse.setWindowState(WindowState.NORMAL);
  }
  else
  {
    super.processAction(request, actionResponse);
  }
}

/**
 * Helper method to serve up the EDIT mode.
 *
 * @param     request the portlet request
 * @param     response the portlet response
 *
 * @exception PortletException if the portlet cannot fulfill the request
 * @exception UnavailableException if the portlet is unavailable to perform edit
 * @exception PortletSecurityException if the portlet cannot fullfill this
 *            request because of security reasons
 * @exception IOException if the streaming causes an I/O problem
 */
public void doEdit (RenderRequest request,
                    RenderResponse response)
    throws PortletException, IOException
{
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();

    String p1Prompt = request.getPreferences().getValue(P1PROMPT_KEY,
           "Parameter 1:");
```

```
String p2Prompt = request.getPreferences().getValue(P2PROMPT_KEY,
       "Parameter 2:");
String p3Prompt = request.getPreferences().getValue(P3PROMPT_KEY,
       "Parameter 3:");
// create an action link back to this portlet

out.print("<FORM ACTION=\"");
out.print(getEditFormSubmitURL(request, response));
out.println("\" METHOD=\"POST\">");
out.println("<TABLE BORDER=\"0\">");
out.println("<TR>");
out.println("<TD WIDTH=\"20%\">");
out.println("<P CLASS=\"portlet-form-field\" ALIGN=\"RIGHT\">Title:");
out.println("</TD>");
out.println("<TD WIDTH=\"80%\">");
out.print("<INPUT CLASS=\"portlet-form-input-field\"
       TYPE=\"TEXT\" NAME=\"");
out.print(TITLE_KEY);
out.print("\" VALUE=\"");
out.print(getTitle(request));
out.println("\" SIZE=\"20\">");
out.println("</TD>");
out.println("</TR>");
out.println("<TR>");
out.println("<TD WIDTH=\"20%\">");
out.println("<P CLASS=\"portlet-form-field\"
       ALIGN=\"RIGHT\">Parameter 1 Prompt:");
out.println("</TD>");
out.println("<TD WIDTH=\"80%\">");
out.print("<INPUT CLASS=\"portlet-form-input-field\"
       TYPE=\"TEXT\" NAME=\"");
out.print(P1PROMPT_KEY);
out.print("\" VALUE=\"");
out.print(p1Prompt);
out.println("\" SIZE=\"20\">");
out.println("</TD>");
out.println("</TR>");
out.println("<TR>");
out.println("<TD WIDTH=\"20%\">");
out.println("<P CLASS=\"portlet-form-field\"
       ALIGN=\"RIGHT\">Parameter 2 Prompt:");
out.println("</TD>");
out.println("<TD WIDTH=\"80%\">");
out.print("<INPUT CLASS=\"portlet-form-input-field\"
       TYPE=\"TEXT\" NAME=\"");
out.print(P2PROMPT_KEY);
out.print("\" VALUE=\"");
out.print(p2Prompt);
out.println("\" SIZE=\"20\">");
out.println("</TD>");
out.println("</TR>");
out.println("<TR>");
out.println("<TD WIDTH=\"20%\">");
out.println("<P CLASS=\"portlet-form-field\"
       ALIGN=\"RIGHT\">Parameter 3 Prompt:");
out.println("</TD>");
out.println("<TD WIDTH=\"80%\">");
out.print("<INPUT CLASS=\"portlet-form-input-field\"
       TYPE=\"TEXT\" NAME=\"");
out.print(P3PROMPT_KEY);
```

```
      out.print("\" VALUE=\"");
      out.print(p3Prompt);
      out.println("\" SIZE=\"20\">");
      out.println("</TD>");
      out.println("</TR>");
      out.println("<TR>");
      out.println("<TD COLSPAN=\"2\" ALIGN=\"CENTER\">");
      out.print("<INPUT CLASS=\"portlet-form-button\" TYPE=\"SUBMIT\" NAME=\"");
      out.print(OK_ACTION);
      out.print("\" VALUE=\"OK\">");

      out.print("<input type=\"hidden\" name=\"");
      out.print(MODE_NAME_PARAM);
      out.print("\" value=\"");
      out.print(MODE_EDITDEFAULTS);
      out.print("\">");

      out.print("<INPUT CLASS=\"portlet-form-button\" TYPE=\"SUBMIT\" NAME=\"");
      out.print(APPLY_ACTION);
      out.print("\" VALUE=\"Apply\">");
      out.println("</TD>");
      out.println("</TR>");
      out.println("</TABLE>");
      out.println("</FORM>");
  }

  protected String escapeForHTMLAttr(String original)
  {
    String escaped = original.replaceAll("&", "&amp;");
    escaped = escaped.replaceAll("<", "&lt;");
    escaped = escaped.replaceAll(">", "&gt;");
    escaped = escaped.replaceAll("\"", "&quot;");
    escaped = escaped.replaceAll("'", "&quot;");

    return escaped;
  }
}
```

16. You can use the following APIs to read parameters passed to the portlet.

```
String param1 = request.getParameter("Parameter_01");
String param2 = request.getParameter("Parameter_02");
```

The sample portlet in Example 19–5 reads navigational parameters.

Example 19–5 Portlet Reading Parameters

```
/* Copyright (c) 2006, Oracle. All rights reserved.  */
package oracle.portlet.server.wsrpsample;
import java.io.IOException;
import java.io.PrintWriter;

import javax.portlet.PortletException;
import javax.portlet.PortletURL;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;

public class ReadOnlyParameterFormPortlet extends ParameterFormPortlet
{
  /**
   * Helper method to serve up the VIEW mode.
```

```
 *
 * @param     request the portlet request
 * @param     response the portlet response
 *
 * @exception PortletException if the portlet has trouble fulfilling the
 *            request
 * @exception IOException if the streaming causes an I/O problem
 */
public void doView (RenderRequest request,
                    RenderResponse response)
    throws PortletException, IOException
{

    String p1Prompt = request.getPreferences().getValue(P1PROMPT_KEY,
           "Parameter 1:");
    String p2Prompt = request.getPreferences().getValue(P2PROMPT_KEY,
           "Parameter 2:");
    String p3Prompt = request.getPreferences().getValue(P3PROMPT_KEY,
           "Parameter 3:");

    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();

    //Retrieve any values for the nav parameters from the request
    String param1 = request.getParameter(PARAMETER1);
    String param2 = request.getParameter(PARAMETER2);
    String param3 = request.getParameter(PARAMETER3);
    param1 = (param1 != null)?param1:"";
    param2 = (param2 != null)?param2:"";
    param3 = (param3 != null)?param3:"";

    out.print("<center><table>");

    if (!"".equals(p1Prompt))
    {
      out.print("<tr><td class=\"portlet-form-field\">");
      out.print(p1Prompt);
      out.print("</td><td>");
      out.print(escapeForHTMLAttr(param1));
      out.print("</td></tr>");
    }

    if (!"".equals(p2Prompt))
    {
      out.print("<tr><td class=\"portlet-form-field\">");
      out.print(p2Prompt);
      out.print("</td><td>");
      out.print(escapeForHTMLAttr(param2));
      out.print("</td></tr>");
    }

    if (!"".equals(p3Prompt))
    {
      out.print("<tr><td class=\"portlet-form-field\">");
      out.print(p3Prompt);
      out.print("</td><td>");
      out.print(escapeForHTMLAttr(param3));
      out.print("</td></tr>");
    }
```

```
      out.println("</table></center>");
  }
}
```

17. Register your producer so that these two portlets appear in the Component Palette. Follow the instructions provided in Section 4.3.1.1, "Registering WSRP Portlet Producers".

18. On a new or existing page, drag and drop the two portlets that you just created from the Component Palette. See Section 4.3.2, "Adding Portlets to a Page" for more information.

19. Right-click an element of the page in the Structure pane and choose **Go to Page Definition**. A page definition similar to Example 19–6 should appear. Notice the variables at the page level and the parameters at the portlet level.

Example 19–6 Page Definition File Sample

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="10.1.3.38.82" id="untitled2PageDef"
                Package="view.pageDefs">
  <parameters/>
  <executables>
    <variableIterator id="variables">
      <variable Name="Parameter_01"    Type="java.lang.Object"/>
      <variable Name="Parameter_02"    Type="java.lang.Object"/>
      <variable Name="Parameter_03"    Type="java.lang.Object"/>
    </variableIterator>
    <portlet id="portlet1"
        portletInstance="/oracle/adf/portlet/Samples_1153150955927/
          applicationPortlets/E10default_7d3d5f76_010c_1000_8004_8a031f56d48d"
        class="oracle.adf.model.portlet.binding.PortletBinding"
        xmlns="http://xmlns.oracle.com/portlet/bindings">
      <parameters>
        <parameter name="ora_wsrp_navigparam_Parameter1"
                   pageVariable="Parameter_01"/>
        <parameter name="ora_wsrp_navigparam_Parameter2"
                   pageVariable="Parameter_02"/>
        <parameter name="ora_wsrp_navigparam_Parameter3"
                   pageVariable="Parameter_03"/>
      </parameters>
    </portlet>
    <portlet id="portlet2"
        portletInstance="/oracle/adf/portlet/Samples_1153150955927/
          applicationPortlets/E11default_7d3d7ef3_010c_1000_8005_8a031f56d48d"
        class="oracle.adf.model.portlet.binding.PortletBinding"
        xmlns="http://xmlns.oracle.com/portlet/bindings">
      <parameters>
        <parameter name="ora_wsrp_navigparam_Parameter1"
                   pageVariable="Parameter_01"/>
        <parameter name="ora_wsrp_navigparam_Parameter2"
                   pageVariable="Parameter_02"/>
        <parameter name="ora_wsrp_navigparam_Parameter3"
                   pageVariable="Parameter_03"/>
      </parameters>
    </portlet>
  </executables>
</pageDefinition>
```

19.1.3 Implementing Export/Import of Customizations (WSRP 2.0)

Another new feature that arrives with WSRP 2.0 is the ability to keep customizations with portlets when moving them from one deployment to another. For example, suppose that you create a portlet and then customize its title within your development environment. If you have enabled export and import for that portlet and its producer, then the customized title will be transported along with the portlet when you deploy it in production environment. If you do not enable export and import, then all customizations are lost when you transport the portlet from one deployment environment to another.

To implement export or import for a portlet and its producer, perform the following steps:

1. Open the `oracle-portlet.xml` file for your producer. Add the `allow-export` and `allow-import` tags for each portlet and the producer as shown in Example 19–7.

Example 19–7 oracle-portlet.xml Sample, Export/Import

```
<portlet-app-extension xsi:schemaLocation="./oracle-portlet-app.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="./oracle-portlet-app.xsd">
    <portlet-extension>
        <portlet-name>portlet1</portlet-name>
        <navigation-parameters>
            <name>Parameter_01</name>
            <type>xsi:string</type>
            <label xml:lang="en">First Parameter</label>
            <hint xml:lang="en">hint0</hint>
        </navigation-parameters>
        <navigation-parameters>
            <name>Parameter_02</name>
            <type>xsi:string</type>
            <label xml:lang="en">Second Parameter</label>
            <hint xml:lang="en">hint1</hint>
        </navigation-parameters>
        <navigation-parameters>
            <name>Parameter_03</name>
            <type>xsi:string</type>
            <label xml:lang="en">Third Parameter</label>
            <hint xml:lang="en">hint2</hint>
        </navigation-parameters>
        <portlet-id>1</portlet-id>
        <allow-export>true</allow-export>
        <allow-import>true</allow-import>
        <portlet-name>portlet2</portlet-name>
        <navigation-parameters>
            <name>Parameter_01</name>
            <type>xsi:string</type>
            <label xml:lang="en">First Parameter</label>
            <hint xml:lang="en">hint0</hint>
        </navigation-parameters>
        <navigation-parameters>
            <name>Parameter_02</name>
            <type>xsi:string</type>
            <label xml:lang="en">Second Parameter</label>
            <hint xml:lang="en">hint1</hint>
        </navigation-parameters>
        <navigation-parameters>
```

```
            <name>Parameter_03</name>
            <type>xsi:string</type>
            <label xml:lang="en">Third Parameter</label>
            <hint xml:lang="en">hint2</hint>
        </navigation-parameters>
        <portlet-id>2</portlet-id>
        <allow-export>true</allow-export>
        <allow-import>true</allow-import>
    </portlet-extension>
    <allow-export>true</allow-export>
    <allow-import>true</allow-import>
</portlet-app-extension>
```

19.1.4 Implementing Rewritten URLs for Resource Proxy

Resource proxying is the standard way to retrieve resources with WSRP. To avoid problems with URLs within your portlet, you can set a flag to rewrite all of the URLs within a specified resource. For example, if have an HTML fragment that contains URLs, then you could set this flag to rewrite its URLs taking into account the WSRP resource proxying.

You indicate that you want URLs rewritten by setting the `PortletRequest` attribute, `oracle.portlet.server.resourceRequiresRewriting`, to true. For example, you might include a code excerpt similar to the one in Example 19–8 to use resource proxying for a URL that you are encoding. Note that typically you would want to encapsulate this code within a method to avoid repeating it for every URL individually.

Example 19–8 Resource Proxy for WSRP

```
request.setAttribute("oracle.portlet.server.resourceRequiresRewriting",
     Boolean.TRUE);
String url = response.encodeURL(pathToResourceForRewriting);
request.removeAttribute("oracle.portlet.server.resourceRequiresRewriting");
```

If you do not specifically set `oracle.portlet.server.resourceRequiresRewriting`, then it defaults to false, meaning that URLs are not rewritten. You must explicitly activate the feature by setting this attribute to true.

19.1.5 Implementing Security for JPS Portlets

You can secure JPS portlets that are deployed to a WSRP producer by configuring security at the WSRP producer end and the client end. See Section 10.10, "Securing Identity Propagation Through WSRP Producers With WS-Security" for information about securing a JPS portlet through its WSRP producer.

19.2 Enhancing PDK-Java Portlets

Once you have built your initial portlet in the Portlet Wizard as described in Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper", you may perform the following tasks to enhance it:

- Section 19.2.1, "Adding Portlet Modes"
- Section 19.2.2, "Passing Parameters and Submitting Events"
- Section 19.2.3, "Using JNDI Variables"

- Section 19.2.4, "Accessing Session Information"
- Section 19.2.5, "Implementing Portlet Security"
- Section 19.2.6, "Enhancing Portlet Performance with Caching"

This section assumes the following:

- You are familiar with portlet terminology such as portlet modes. See Chapter 14, "Understanding Portlets" and Section 18.1, "Guidelines for Creating Java Portlets".
- You have already downloaded and installed the Java Portlet Container and have an Oracle Containers for J2EE (OC4J) container to which you may deploy your portlets.
- You are already familiar with Oracle JDeveloper and know how to build and deploy Java components using it. You can download Oracle JDeveloper from OTN. Visit the Oracle JDeveloper page on OTN:

  `http://www.oracle.com/technology/products/jdev/index.html`

19.2.1 Adding Portlet Modes

In the Portlet Wizard, you add portlet modes by checking boxes on the wizard pages. See Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper" for more information about using the wizard. For each portlet mode that you select in the wizard, a basic HelloWorld skeleton is created. If you need to add a portlet mode after creating the portlet, then you can do that manually by updating `provider.xml` and HTML or JSPs in Oracle JDeveloper. The following sections explain how to add portlet modes to a PDK-Java portlet:

- Section 19.2.1.1, "Assumptions"
- Section 19.2.1.2, "Implementing Extra Portlet Modes"
- Section 19.2.1.3, "Updating the XML Producer Definition"
- Section 19.2.1.4, "Viewing the Portlet"

Once you have completed this section, you will be able to implement any portlet mode using `RenderManager` because the principles are the same for all modes. For example, even though this section does not describe how to implement the About mode in detail, you will understand how to do it, as the process is the same as for Help mode, which is described here.

For more detailed information about the PDK run time classes used in this section, see the JavaDoc on OTN by clicking **Java Doc API** on the Portlet Development page available at

`http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html`

For more information about the syntax of `provider.xml`, see the producer JavaDoc:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

19.2.1.1 Assumptions

The following assumptions are made to perform the tasks in this section:

- You built a portlet using the wizard, successfully registered the producer, and added the portlet to the page.

19.2.1.2 Implementing Extra Portlet Modes

Your first task when creating portlet modes manually is to create an HTML file or JSP for each mode. For example, if you want to implement Help mode, then you need to create an HTML file to provide Help content.

To create an HTML file to preview content, perform the following steps:

1. In Oracle JDeveloper, open the project that contains your portlets.
2. Under Web Content, `htdocs\myportlet`, create an HTML page called `HelpPage.html`. The content of the file could be something similar to the following:

```
<p>This is the <i>Help</i> mode of your portlet!</p>
```

Once you have created the HTML file for Help content, you are ready to update the XML producer definition.

19.2.1.3 Updating the XML Producer Definition

When you want to expose additional portlet modes you must update your XML producer definition as follows:

- Set a Boolean flag that indicates to the PDK Framework that a link or icon to that mode should be rendered.
- Point to the HTML file or JSP that you created for that mode.

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

For example, if you want to render Help mode, then perform the following steps:

1. Edit the producer definition file, `provider.xml` and add the tag to activate Preview mode within the `<portlet></portlet>` tags:

```
<hasHelp>true</hasHelp>
```

2. Specify the Help page to be the HTML page that you created in Section 19.2.1.2, "Implementing Extra Portlet Modes":

```
<helpPage>/htdocs/myportlet/MyPortletHelpPage.html</helpPage>
```

3. Save the updates to `provider.xml`.
4. Redeploy your portlet. See step 8 in Section 18.9, "Deploying Your Portlet to an Application Server".

   When you redeploy, Oracle JDeveloper automatically saves and compiles the code before deploying the portlet.

19.2.1.4 Viewing the Portlet

To view the new portlet modes, you must ensure that your updated XML producer definition is reparsed. To do this, perform the following steps:

1. Copy the HTML file you created in Section 19.2.1.2, "Implementing Extra Portlet Modes" and `provider.xml` to the OC4J instance where you plan to deploy the portlet.
2. Refresh the producer.

3. Refresh the page containing your portlet.
4. Click the **Help** link.

19.2.2 Passing Parameters and Submitting Events

PDK-Java and Oracle WebCenter Framework provides public and private portlet parameters, and private events to enable portlet developers to easily write reusable, complex portlets. The Portlet Wizard in Oracle JDeveloper creates portlets that are already set up to use parameters and events. This feature enables you to focus solely on adding business logic to your portlets and does not require any changes to `provider.xml`.

For an overview of parameters and events, see the following:

- Section 15.12, "Public Portlet Parameter Support"
- Section 15.13, "Private Portlet Parameter Support"

This section covers the following:

- Section 19.2.2.1, "Assumptions"
- Section 19.2.2.2, "Adding Public Parameters"
- Section 19.2.2.3, "Passing Private Portlet Parameters"
- Section 19.2.2.4, "Creating Private Events"

19.2.2.1 Assumptions

The following assumptions are made to perform the tasks in this section:

1. You have followed through and understood Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

> **Note:** Each portlet is limited to 4K of data. The lengths of parameter and event names, display names, and descriptions all contribute toward this 4K limit. Hence, you should not use an excessive number of parameters and events for each portlet, or give them lengthy names and descriptions.

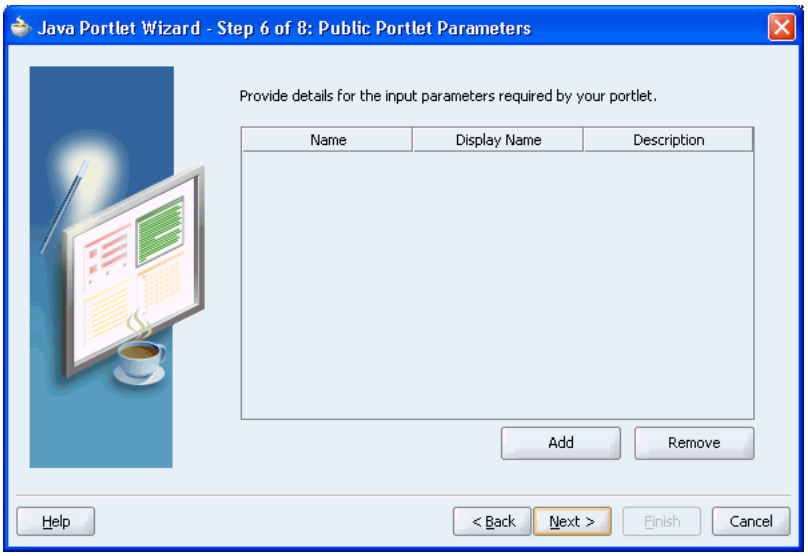
19.2.2.2 Adding Public Parameters

Using the Portlet Wizard, you can easily create a portlet with public parameters. When you register the producer and drop the portlet on a page, the portlet's parameters are automatically linked to page variables.

To add parameters to your portlet, perform the following tasks:

1. Right-click your project and choose **New** from the context menu.
2. From New Galleries, choose the **Portlets** category and then the **Oracle PDK Java Portlet** item.
3. Proceed through the wizard as you normally would until you reach the Public Portlet Parameters page. See Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper" for basic information about going through the wizard.
4. When you reach the Public Portlet Parameters page, as shown in Figure 19–6, click the **Add** button. A new row is added to the table.

Figure 19–6 Public Portlet Parameters Page of Portlet Wizard



5. Click the default name that appears in the **Name** column. Use the Backspace key to delete the default name.
6. Enter a new parameter name, for example, `Parameter_01`.
7. Repeat steps 6 and 7 for the **Display Name** and **Description** columns.
8. Repeat steps 4 through 7 for each public parameter that you want to add to the portlet.
9. Click **Finish**.
10. Once you finish the Portlet Wizard, your portlet is created.
11. In the Applications Navigator, expand the Web Content and `WEB-INF\providers\`*`providername`* nodes. You should see your `provider.xml` file there.
12. Right-click `provider.xml` and choose **Open** from the context menu. You should see entries for the parameters that you added on the Public Portlet Parameters page in the wizard, as shown in Example 19–9.

Example 19–9 provider.xml Sample, Public Parameters

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<?providerDefinition version="3.1"?>
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
  <session>false</session>
  <passAllUrlParams>false</passAllUrlParams>
  <preferenceStore class=
     "oracle.portal.provider.v2.preference.FilePreferenceStore">
   <name>prefStore1</name>
   <useHashing>true</useHashing>
  </preferenceStore>
  <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
      <id>1</id>
      <name>MyPortlet</name>
      <title>My Portlet</title>
      <description>My Portlet Description</description>
      <timeout>40</timeout>
      <showEditToPublic>false</showEditToPublic>
```

```
      <hasAbout>false</hasAbout>
      <showEdit>true</showEdit>
      <hasHelp>false</hasHelp>
      <showEditDefault>false</showEditDefault>
      <showDetails>false</showDetails>
      <inputParameter class=
        "oracle.portal.provider.v2.DefaultParameterDefinition">
       <name>Parameter_01</name>
       <displayName>Parameter_01</displayName>
       <description>My first parameter</description>
      </inputParameter>
      <inputParameter class=
        "oracle.portal.provider.v2.DefaultParameterDefinition">
       <name>Parameter_02</name>
       <displayName>Parameter_02</displayName>
       <description>My second parameter</description>
      </inputParameter>
      <inputParameter class=
        "oracle.portal.provider.v2.DefaultParameterDefinition">
       <name>Parameter_03</name>
       <displayName>Parameter_03</displayName>
      </inputParameter>
      <renderer class="oracle.portal.provider.v2.render.RenderManager">
         <renderContainer>true</renderContainer>
         <renderCustomize>true</renderCustomize>
         <autoRedirect>true</autoRedirect>
         <contentType>text/html</contentType>
         <showPage>/htdocs/myportlet/MyPortletShowPage.jsp</showPage>
         <editPage>/htdocs/myportlet/MyPortletEditPage.jsp</editPage>
      </renderer>
      <personalizationManager class=
        "oracle.portal.provider.v2.personalize.PrefStorePersonalizationManager">
       <dataClass>
        oracle.portal.provider.v2.personalize.NameValuePersonalizationObject
       </dataClass>
      </personalizationManager>
   </portlet>
</provider>
```

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

13. The wizard also generates code in the *portletname*`ShowPage.jsp` for your portlet. Expand the `htdocs\`*portletname* node in the Applications Navigator.
14. Right-click *portletname*`ShowPage.jsp` and choose **Open** from the context menu. The code should look something like Example 19–10. Notice the code that retrieves the parameters.

Example 19–10 ShowPage.jsp Sample

```
<%@page contentType="text/html; charset=windows-1252"
        import="oracle.portal.provider.v2.render.PortletRenderRequest"
        import="oracle.portal.provider.v2.http.HttpCommonConstants"
        import="oracle.portal.provider.v2.ParameterDefinition"
%>
<%
   PortletRenderRequest pReq = (PortletRenderRequest)
```

```
      request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
%>
<P>Hello <%= pReq.getUser().getName() %>.</P>
<P>This is the <b><i>Show</i></b> render mode!</P>
<%
   ParameterDefinition[] params =
       pReq.getPortletDefinition().getInputParameters();
%>
<p>This portlet's input parameters are...</p>
<table align="left" width="50%" ><tr><td><span
class="PortletHeading1">Name</span></td><td><span
class="PortletHeading1">Value</span></td></tr>
<%
   String name = null;
   String value = null;
   String[] values = null;
   for (int i = 0; i < params.length; i++)
   {
       name = params[i].getName();
       values = pReq.getParameterValues(name);
       if (values != null)
       {
           StringBuffer temp = new StringBuffer();
           for (int j = 0; j < values.length; j++)
           {
               temp.append(values[j]);
               if (j + 1 != values.length)
               {
                   temp.append(", ");
               }
           }
           value = temp.toString();
       }
       else
       {
           value = "No values have been submitted yet.";
       }
%>
<tr>
  <td><span class="PortletText2"><%= name %></span></td>
  <td><span class="PortletText2"><%= value %></span></td>
</tr>
<%
   }
%>
</table>
```

15. Add logic to your portlet that enables it to submit parameter values entered by users.
16. Repeat steps 1 through 15 and add logic to this second portlet that enables it to simply display parameter values that it retrieves.
17. Register your producer so that these two portlets appear in the Component Palette. Follow the instructions provided in Section 4.3.1.2, "Registering PDK-Java Portlet Producers".
18. On a new or existing page, drag and drop the two portlets that you just created from the Component Palette. See Section 4.3.2, "Adding Portlets to a Page" for more information.

19. Right-click an element of the page in the Structure pane and choose **Go to Page Definition**. A page definition similar to Example 19–11 should appear. Notice the variables at the page level and the parameters at the portlet level.

Example 19–11 Page Definition File Sample

```
<?xml version="1.0" encoding="UTF-8"?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="10.1.3.38.90" id="untitled1PageDef"
                Package="view.pageDefs">
  <executables>
    <variableIterator id="variables">
      <variable Name="portlet1_Parameter_01" Type="java.lang.Object"/>
      <variable Name="portlet1_Parameter_02" Type="java.lang.Object"/>
      <variable Name="portlet1_Parameter_03" Type="java.lang.Object"/>
    </variableIterator>
    <portlet id="portlet1"
       portletInstance="/oracle/adf/portlet/PdkPortletProducer1_1153936627784
          /applicationPortlets/Portlet1_abfc5a10_010c_1000_8003_82235f50d831"
        class="oracle.adf.model.portlet.binding.PortletBinding"
        xmlns="http://xmlns.oracle.com/portlet/bindings">
      <parameters>
        <parameter name="Parameter_01" pageVariable="portlet1_Parameter_01"/>
        <parameter name="Parameter_02" pageVariable="portlet1_Parameter_02"/>
        <parameter name="Parameter_03" pageVariable="portlet1_Parameter_03"/>
      </parameters>
    </portlet>
  </executables>
</pageDefinition>
```

20. You should now be able to run your application, enter values in the first portlet you created, and see them displayed in the second portlet

19.2.2.3 Passing Private Portlet Parameters

In some cases, you might need a parameter that is known only to the portlet instance. These parameters are known as private parameters because they have no connection to the page and are known only to the portlet. Private parameters often come in handy when you are building navigation for your portlet. For example, if you have a multipage portlet, then you can use these private parameters to jump to another resource of the portlet.

This section covers the following:

- Section 19.2.2.3.1, "Private Parameters"
- Section 19.2.2.3.2, "Portlet URL Types"
- Section 19.2.2.3.3, "Building Links with the Portlet URL Types"
- Section 19.2.2.3.4, "Building Forms with the Portlet URL Types"
- Section 19.2.2.3.5, "Implementing Navigation within a Portlet"
- Section 19.2.2.3.6, "Restricting Navigation to Resources"

19.2.2.3.1 Private Parameters Private parameters are used in classic Web applications to pass information from links or forms in the browser back to the server. The server in turn takes actions and returns the appropriate content. For example, if the user of a dictionary Web site asks for information about hedgehogs, then the URL submitted to the server might append a private parameter as follows:

```
http://dictionary.reference.com/search?q=Hedgehog
```

If the server is responsible for rendering the whole page and the client communicates directly with the server, then this form of URL works well. In a WebCenter application, the client does not communicate directly with portlets. Instead, Oracle WebCenter Framework mediates between the client and the portlet. Moreover, because most pages have multiple portlets, Oracle WebCenter Framework communicates with multiple portlets.

For example, suppose a page contains two portlets, a thesaurus portlet and a dictionary portlet. Both portlets use `q` as a parameter to record the search queries made by the user. If the user queries the thesaurus portlet, then the URL used to rerequest the page with the updated thesaurus portlet must contain the thesaurus portlet's parameter, `q`. The thesaurus parameter must also be distinguished from dictionary portlet parameter 1, which performs the same function for that portlet. An example URL with the properly qualified thesaurus parameter might look something like the following:

```
http://host/portal/page?_pageid=33,1&_dad=portal&_schema=PORTAL
   &_piref33_38279_33_1_1.q=Hedgehog
```

Notice the fully qualified parameter name, `_piref33_38279_33_1_1.q`. It identifies the parameter and distinguishes it from other parameters on the page. Further, notice that the URL contains some parameters unrelated to any portlet. These parameters are untouched by the portlet because it does not own them.

You must ensure that the portlet meets the following criteria:

- It properly qualifies its own parameters when they are built into links and forms.
- It leaves unchanged any parameters that do not belong to it.

The following API call transforms an unqualified parameter name into a qualified parameter name:

```
HttpPortletRendererUtil.portletParameter(HttpServletRequest request, String
param);
```

`HttpPortletRendererUtil` is in the package `oracle.portal.provider.v2.render.http`.

For example:

```
qualParamQ = HttpPortletRendererUtil.portletParameter(r, "q");
```

To fetch the value of a portlet parameter from the incoming request, you can use the following API:

> **Note:** The API converts the parameter name into the qualified parameter name before fetching the value from the incoming request. Hence, you need not perform this step.

```
PortletRenderRequest.getQualifiedParameter(String name)
```

`PortletRenderRequest` is in the package `oracle.portal.provider.v2.render`.

For example:

```
valueQ = r.getQualifiedParameter("q");
```

The other aspect of a portlet's responsibilities with respect to private parameters is to not disturb the parameters on the URL that it does not own. The utilities you may use to ensure adherence to this rule are discussed in Section 19.2.2.3.3, "Building Links with the Portlet URL Types" and Section 19.2.2.3.4, "Building Forms with the Portlet URL Types".

19.2.2.3.2 Portlet URL Types When a portlet renders itself, Oracle WebCenter Framework passes it various URLs, which the portlet can then use to render links. You can fetch and manipulate these URLs to simplify the task of creating links. The following is a list of the URLs provided to portlets:

- **PAGE_LINK** is a URL to the page upon which the portlet instance resides. You use this URL as the basis for all intraportlet links. If the portlet renders a link that navigates the user to another section of the same portlet, then this navigation must be encoded as a set of parameters using the PAGE_LINK.
- **DESIGN_LINK** is a URL to the portlet's personalization (Edit mode) page. A portlet's Edit and Edit Defaults modes are not rendered on the same page as the portlet. The Edit and Edit Defaults modes take over the entire browser window. The portlet's Edit and Edit Defaults modes are not necessarily accessible to every user. It represents a minimal, static framework in which the portlet is free to render its personalization options. This URL is only of use when rendering Personalize links.
- **BACK_LINK** is a URL to a useful return point from the current page where the portlet renders itself. For example, when the portlet is rendering it's personalization page (Edit mode), this link refers to the page on which the portlet resides and from which the user navigated to the personalization page. Consequently, it is the link you would encode in the buttons that accept or cancel the pending action. This URL is only useful for the desktop rendering of portlets (usually in Edit or Edit Defaults mode).

19.2.2.3.3 Building Links with the Portlet URL Types To build links with Portlet URL types, you need to access them and use them when writing portlet rendering code. To fetch the URL for a link, you call the following APIs in PDK-Java:

```
portletRenderRequest.getRenderContext().getPageURL()
portletRenderRequest.getRenderContext().getEventURL()
portletRenderRequest.getRenderContext().getDesignURL()
portletRenderRequest.getRenderContext().getLoginServerURL()
portletRenderRequest.getRenderContext().getBackURL()
```

In portlet navigation, you need to add (or update) your portlet's parameters in the page URL. To perform this task, you can use the following API to build a suitable URL:

```
UrlUtils.constructLink(
    PortletRenderRequest pr,
    int linkType, -- UrlUtils.PAGE_LINK in this case
    NameValue[] params,
    boolean encodeParams,
    boolean replaceParams)
```

`UrlUtils` resides in the package called `oracle.portal.provider.v2.url`. Notice that you do not actually fetch the page URL yourself. Rather you use one of the supplied portlet URL types, `UrlUtils.PAGE_LINK`.

The parameter names in the `params` argument should be fully qualified. Moreover, assuming that you properly qualify the parameters, `UrlUtils.constructLink`

with the appropriate `linkType` does not disturb other URL parameters that are not owned by the portlet.

An alternative version of `UrlUtils.contructLink` accepts a URL as the basis for the returned URL. If you require an HTML link, then you can use `UrlUtils.constructHTMLLink` to produce a complete anchor element.

The following example portlet, `ThesaurusLink.jsp`, uses the parameter `q` to identify the word for which to search the thesaurus. It then creates links on the found, related words that the user may follow to get the thesaurus to operate on that new word. See the example in Section 19.2.2.3.4, "Building Forms with the Portlet URL Types" to see the initial submission form that sets the value of `q`.

```
<%
    String paramNameQ = "q";
    String qualParamNameQ =
    HttpPortletRendererUtil.portletParameter(paramNameQ);
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramValueQ = pRequest.getQualifiedParameter(paramNameQ);
%>
<!-- Output the HTML content -->
<center>
    Words similar to <%= paramValueQ %>
    <br>
    Click the link to search for words related to that word.
    <br>
    <ul>
<%
        String[] relatedWords = Thesaurus.getRelatedWords(paramValueQ);
        NameValue[] linkParams = new NameValue[1];
        for (int i=0; i<=relatedWords.length; i++)
        {
            linkParams[0] = new NameValue(
                qualParamNameQ, relatedWords[i]);
%>
            <li>
            <b> <%= relatedWords[i] %> </b>
            <%= UrlUtils.constructHTMLLink(
                pRequest,
                UrlUtils.PAGE_LINK,
                "(words related to " + relatedWords[i] + ")",
                "",
                linkParams,
                true,
                true)%>
           </li>
<%
        }
%>
   </ul>
</center>
```

19.2.2.3.4 Building Forms with the Portlet URL Types Use of portlet parameters in forms is little different from links. The following two fundamental rules continue to apply:

- Qualify the portlet's parameter names.
- Do not manipulate or remove the other parameters on the incoming URL.

In terms of markup and behavior, forms and links differ quite considerably. However, just as with links, PDK-Java contains utilities for complying with these two basic rules.

The code for properly qualifying the portlet's parameter name is the same as described in Section 19.2.2.3.3, "Building Links with the Portlet URL Types". After all, a parameter name is just a string, whether it be a link on a page or the name of a form element.

Forms differ from links in the way you ensure that the other parameters in the URL remain untouched. Once you open the form in the markup, you can make use of one of the following APIs:

```
UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName);
UrlUtils.htmlFormHiddenFields(someURL);
```

where `formName = UrlUtils.htmlFormName(pRequest,null).`

> **Note:** Just as parameters in URLs and element names in forms require qualification to avoid clashing with other portlets on the page, form names must be fully qualified because any given page might have several forms on it.

The `htmlFormHiddenFields` utility writes HTML hidden form elements into the form, one form element for each parameter on the specified URL that is not owned by the portlet.

```
<INPUT TYPE="hidden" name="paramName" value="paramValue">
```

Thus, you need only to add their portlet's parameters to the form.

The other item of which you need to be aware is how to derive the submission target of your form. In most cases, the submission target is the current page:

```
formTarget = UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK)
```

The value of `formTarget` can be the action attribute in an HTML form or the target attribute in a `SimpleForm`. Even though the method name includes HTML, it actually just returns a URL and thus you can use it in mobile portlets, too.

The following example form renders the thesaurus portlet's submission form. See the example in Section 19.2.2.3.3, "Building Links with the Portlet URL Types" for the portlet that results from the submission of this form.

```
<%
    String paramNameSubmit = "submit";
    String paramNameQ = "q";
    String qualParamNameQ =
        HttpPortletRendererUtil.portletParameter(paramNameQ);
    String qualParamNameSubmit =
    HttpPortletRendererUtil.portletParameter(paramNameSubmit);
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String formName = UrlUtils.htmlFormName(pRequest,"query_form");
%>
<!-- Output the HTML content -->
<center>
    <b>Thesaurus</b>
    Enter the word you wish to search for
    <form name="<%= formName %>" method="POST"
        action="<%= UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK) %>">
```

```
        <%= UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName)%>
        <table><tr><td>
            Word of interest:
        </td><td>
            <input
                type="text"
                size="20"
                name="<%= qualParamNameQ %>"
                value="">
        </td></tr></table>
        <input type=submit name="<%= qualParamNameSubmit %>" Value="Search">
    </form>
</center>
```

19.2.2.3.5 Implementing Navigation within a Portlet You can implement navigation within a portlet in one of three ways, as follows:

- Pass navigation information in rendered URLs using private portlet parameters. Branching logic within the portlet code then determines which section of the portlet to render based on the URL. This option represents a small extension to the thesaurus example presented in Section 19.2.2.3.3, "Building Links with the Portlet URL Types" and Section 19.2.2.3.4, "Building Forms with the Portlet URL Types". Basically, instead of performing thesaurus search operations using the value of parameter `q`, the portlet branches based on the parameter value and renders different content accordingly.
- Pass navigation information as described in the previous item but use PDK-Java to interpret the parameter and thus branch on its value. This option requires some further changes to the thesaurus example and is more fully explained later in this section.
- Use session storage to record the portlet state and private parameters to represent actions rather than explicit navigation. This method provides the only way that you can restore the portlet to it's previous state when the user navigates off the page containing the portlet. Once the user leaves the page, all private portlet parameters are lost and you can only restore the state from session storage, assuming you previously stored it there. This option requires that you understand and implement session storage. See Section 19.2.4.2, "Implementing Session Storage" for more information about implementing session storage.

The following portlet code comes from the multipage example in the sample producer of PDK-Java:

```
<portlet>
    <id>11</id>
    <name>Multipage</name>
    <title>MultiPage Sample</title>
    <shortTitle>MultiPage</shortTitle>
    <description>
        This portlet depicts switching between two screens all
        in the context of a Portal page.
    </description>
    <timeout>40</timeout>
    <timeoutMessage>MultiPage Sample timed out</timeoutMessage>
    <renderer class="oracle.portal.provider.v2.render.RenderManager">
        <contentType>text/html</contentType>
        <showPage>/htdocs/multipage/first.jsp</showPage>
        <pageParameterName>next_page</pageParameterName>
    </renderer>
```

```
</portlet>
```

Notice that the value of `pageParameterName` is the name of a portlet parameter, `next_page`, that the PDK-Java framework intercepts and interprets as an override to the value of the `showPage` parameter. If the PDK-Java framework encounters the qualified version of the parameter when the multipage portlet is requested, then it will render the resource identified by `next_page` rather than `first.jsp`. Note that PDK-Java does not render the parameter within the portlet, that responsibility falls to the portlet.

You can modify the thesaurus example to operate with the use of this parameter. Specifically, you can use the form submission portlet to be the input for the thesaurus (the first page of the portlet), then navigate the user to the results page, which contains links to drill further into the thesaurus. The following examples illustrate these changes.

> **Note:** The example that follows is most useful for relatively simple cases, such as this thesaurus example. If your requirements are more complex (for example, you want to build a wizard experience), then you should consider using an MVC framework such as Struts. For information about how to build portlets from struts applications, see Section 19.4, "Building Struts Portlets".

`ThesaurusForm.jsp`:

```
<%
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramNameSubmit = "submit";
    String paramNameQ = "q";
    String qualParamNameQ =
        HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);
    String qualParamNameSubmit =
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameSubmit);
    String formName = UrlUtils.htmlFormName(pRequest,"query_form");
%>
<!-- Output the HTML content -->
<center>
    <b>Thesaurus</b>
    Enter the word you wish to search for
    <form name="<%= formName %>" method="POST"
        action="<%= UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK) %>">
        <%= UrlUtils.htmlFormHiddenFields(pRequest,UrlUtils.PAGE_LINK, formName)
%>
        <%= UrlUtils.emitHiddenField(
                HttpPortletRendererUtil.portletParameter(request, "next_page"),
                "htdocs/path/ThesaurusLink.jsp" ) %>
        <table><tr><td>
            Word of interest:
        </td><td>
            <input
                type="text"
                size="20"
                name="<%= qualParamNameQ %>"
                value="">
        </td></tr></table>
        <input type=submit name="<%= qualParamNameSubmit %>" Value="Search">
    </form>
```

```
</center>
```

Notice how `next_page` must be explicitly set to point to `ThesaurusLink.jsp`. If you do not explicitly set `next_page` in this way, then it defaults to the resource registered in `provider.xml`, which is `ThesaurusForm.jsp`.

`ThesaurusLink.jsp`:

```
<%
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramNameQ = "q";
    String paramNameNextPage = "next_page";
    String qualParamNameQ =
    HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);
    String qualParamNameNextPage =
        HttpPortletRendererUtil.portletParameter(pRequest, paramNameNextPage);
    String paramValueQ = pRequest.getQualifiedParameter(paramNameQ);
%>
<!-- Output the HTML content -->
<center>
    Words similar to <%= paramValueQ %>
    <br>
    Click the link to search for words related to that word.
    <br>
    <ul>
<%
        Thesaurus t = new Thesaurus();
        String[] relatedWords = t.getRelatedWords(paramValueQ);
        NameValue[] linkParams = new NameValue[2];
        linkParams[0] = new NameValue(
            qualParamNameNextPage, "htdocs/path/ThesaurusLink.jsp");
        for (int i=0; i<relatedWords.length; i++)
        {
            linkParams[1] = new NameValue(
                qualParamNameQ, relatedWords[i]);
%>
            <li>
            <b> <%= relatedWords[i] %> </b>
            <%= UrlUtils.constructHTMLLink(
                pRequest,
                UrlUtils.PAGE_LINK,
                "(words related to " + relatedWords[i] + ")",
                "",
                linkParams,
                true,
                true)%>
           </li>
<%
        }
%>
   </ul>
   <a href="<%=XMLUtil.escapeXMLAttribute
               (pRequest.getRenderContext().getPageURL())%>">
      Reset Portlet
   </a>
</center>
```

19.2.2.3.6 Restricting Navigation to Resources One of the dangers of implementing navigation with private parameters is that users could potentially navigate to portlet

resources that you prefer to restrict. To control the resources to which the user may navigate, you can create a whitelist of acceptable resources to which a user may navigate. If you do not construct a whitelist to restrict navigation, then your portlet's resources will be accessible according to the following default rules:

- Any path immediately beneath the servlet root context is navigable. For example, `/index.jsp` is accessible but `/WEB-INF/web.xml` is not.
- Any path under the `htdocs` directory is navigable. For example, both `/htdocs/multipage/first.jsp` and `/htdocs/lottery/lotto.jsp` are accessible.

To change this default behavior, you can add permissible path values to the provider definition file, `provider.xml`. For example, suppose you have a portlet where a JSP is used as a controller to forward requests to other pages depending on the `pageParameterName` private parameter. The XML excerpt in Example 19–12 would only enable resources under `/htdocs/multiportlet` to be shown. All other resources would be restricted.

Example 19–12 Whitelist Excerpt from the provider.xml File

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
   <id>1</id>
   <name>Multipage</name>
   <title>A MultiPage Portlet</title>
   ...
   <renderer class="oracle.portal.provider.v2.render.RenderManager">
          <contentType>text/html</contentType>
          <showPage>/htdocs/multiportlet/controller.jsp</showPage>
          <pageParameterName>show_page</pageParameterName>
          <allowedPath>/htdocs/multiportlet/*</allowedPath>
   </renderer>
</portlet>
```

The pattern matching rules for this feature are similar to URL pattern matching in `web.xml` files. The rules are as follows:

- To match the defined patterns, the resource path must exactly match unless wildcards are used.
- The first wildcard is for path matching and consists of a string beginning with `/` and ending with `/*`. Any resource whose path starts with this string will be matched. For an `<allowedPath>` value of `/htdocs/sub1/*`, valid values of the private parameter would include `/htdocs/sub1/file.jsp` and `/htdocs/sub1/sub2/file2.jsp`.
- The second wildcard is for file type matching and consists of a string starting with `*.` and ending with a file extension. Valid values for the page parameter will end with that file extension. For an `<allowedPathvalue>` of `*.jsp`, valid values of the private parameter would include `/htdocs/sub1/file.jsp` and `/htdocs/sub1/file2.jsp`.

19.2.2.4 Creating Private Events

In some cases, it is useful for a portlet to complete a transaction before rendering a page on which it resides rather than having the transaction and the rendering executed simultaneously. For example, suppose a portlet has a link that initiates an update of some data value that might effect other portlets on the page. If the transaction occurred simultaneously with a refresh of the page, then other portlets that rely on that data value may or may not be refreshed with the latest value. Furthermore, when

transactions and rendering are tied together in this way, an action such as the user hitting **Back** in their browser could cause the transaction to be repeated, perhaps creating a duplicate record.

In JPS portlets, this situation is solved using the `processAction` method, which enables an individual portlet to complete a transaction, such as updating a value, before enabling the page rendering to occur. PDK-Java does not have `processAction`, but you can achieve the same results by submitting data to your servlet through a different mechanism. If you are using Struts for the page flow and control of a portlet, then you could use the `form` tag and transaction tokens to avoid submitting the same parameter twice. See Section 19.4, "Building Struts Portlets" for more information about Struts portlets.

Another possibility is to submit data through Edit mode rather than Shared Screen mode. Requests based on full page portlet modes, such as Edit mode, are sent only to the portlet that generated the link. Other portlets on the same portal page never even see a render request. Hence, these full page portlet modes provide you with the capability to execute a transaction for a portlet separately from the page and its other portlets.

Once you have processed the portlet's submission, you redirect from the full page portlet mode back to the page using the back URL. This action has two desirable effects, as follows:

- It returns the user to the page from which they came.
- It clears all traces of the form submission from the browser.

As a result, any refreshing of the page is guaranteed to occur after the processing of the data submission. Because the page refresh comes after the submission you can be sure that all portlets on the page will access the updated data and not cause a duplicate submission.

This technique is illustrated by a sample portlet in PDK-Java called the private event submission portlet. It demonstrates submitting a simple form to the portlet and logging the contents of the form. In the private event submission portlet, you overload Edit mode to handle both the data submission and the portlet's rendering for personalizations. Note that any of the other full page portlet modes (Help, About, and Edit Defaults) would be equally effective for this purpose.

Edit mode for this portlet includes additional code that first looks for a specific parameter. If this parameter is present, then it means that the request represents a private event. The same mode can handle many different private events by using different values for the distinguishing parameter. If the distinguishing parameter does not exist, then Edit mode falls through to the standard portlet personalization logic.

After handling the private event, this mode redirects to the page using exactly the same logic that Edit mode uses when a user clicks **OK**. See `EditServlet.java` in the sample files for the complete source code illustrating this technique.

Note: When you use this technique, you must take care that the details of your event are persisted somewhere. As portlet rendering does not happen in the same request cycle as the event processing, any data from the event required to render the portlet must be available from storage. Otherwise, the portlet or the page may lack the data it requires the next time it is rendered.

If you need to persist your data, then be sure to store it in a qualified manner (by the user and portlet reference). Otherwise, you may accidentally associate an event from one user/portlet with the rendering of the same portlet for another user on a different page, or even associate the same user/same portlet with a different portlet reference on the same page.

19.2.3 Using JNDI Variables

When writing Java portlets, you may set deployment specific properties through the JNDI service such that their values may be retrieved from your producer code. In this way, you can specify any property in a producer deployment and then easily access it anywhere in your producer code. PDK-Java provides utilities to enable the retrieval of both producer and non-producer JNDI variables within a J2EE container. To use JNDI variables, you need to perform the following tasks:

- Section 19.2.3.1, "Declaring JNDI Variables"
- Section 19.2.3.2, "Setting JNDI Variable Values"
- Section 19.2.3.3, "Retrieving JNDI Variables"

19.2.3.1 Declaring JNDI Variables

You declare JNDI variables in the `web.xml` file for your producer. The format for declaring a JNDI variable is as follows:

```
<env-entry>
    <env-entry-name>variableName</env-entry-name>
    <env-entry-type>variableType</env-entry-type>
    <env-entry-value>variableValue</env-entry-value>
</env-entry>
```

The `env-entry-name` element contains the name by which you want identify the variable. `env-entry-type` contains the fully qualified Java type of the variable. `env-entry-value` contains the variable's default value.

19.2.3.1.1 Variable Types In the `env-entry-type` element, you should supply the fully qualified Java type of the variable, which will be expected by your Java code. The Java types you may use in your JNDI variables are as follows:

- `java.lang.Boolean`
- `java.lang.String`
- `java.lang.Integer`
- `java.lang.Double`
- `java.lang.Float`

The J2EE container uses these type declarations to automatically construct an object of the specified type and gives it the specified value when you retrieve that variable in your code.

19.2.3.1.2 Variable Naming Conventions The PDK-Java defines a number of environment variables that can be set at the individual producer service level or at the Web application level. To avoid naming conflicts between different producer services or different application components packaged in the same Web application, Oracle recommends you devise some naming convention.

> **Note:** If you use the `EnvLookup` method, then you must use `oracle/portal/provider/service/property`. You cannot substitute your own company name or component in this case.

For example:

- Producer service specific names should be of the form:

  `{company}/{component name}/{producer name}/{variable name}`

- Shared names should be of the form:

  `{company}/{component name}/{producer name}/global`

where:

- `{company}` is the name of the company owning the application.
- `{component name}` is the name of the application or component with which the producer is associated.
- `{producer name}` is the service name of the producer.
- `{variable name}` is the name of the variable itself.

As you can see, these naming conventions are similar to those used for Java packages. This approach minimizes the chance of name collisions between applications or application components. PDK-Java provides utilities that enable you to retrieve variables in this form without hard coding the service name of the producer into your servlets or JSPs. The service name need only be defined in the producer's WAR file. See Section 19.2.3.3, "Retrieving JNDI Variables" for more information about retrieving JNDI variables.

19.2.3.1.3 Examples The following examples illustrate producer variable names:

```
oracle/portal/myProvider/myDeploymentProperty
oracle/portal/myprovider/myProperties/myProperty
```

The following example illustrates non-producer variable names:

```
oracle/portal/myOtherProperty
```

19.2.3.2 Setting JNDI Variable Values

In your producer deployment, you may want to set a new value for some or all of your JNDI variables. You can perform this task in one of two ways as follows:

- If you are using Oracle Enterprise Manager 10*g*, then you can set the variables from there.
- If you are using a standalone instance of OC4J, then you need to manually change the variable values in the PDK-Java's `orion-web.xml` file.

19.2.3.2.1 Setting Values in Oracle Enterprise Manager 10*g* To set variable values in Oracle Enterprise Manager 10*g*, do the following:

1. In Oracle Enterprise Manager 10*g*, click the **Application Server** instance where you have deployed PDK-Java.
2. Click the Name representing this deployment.
3. Click the **Applications** tab.
4. From the **View** pulldown list, choose **Modules**.
5. Click the Name of the module.
6. Click the **Administration** tab.
7. Click the **Go to Task** icon for **Environment Entries Mapping**.
8. On the Environment Entries Mapping page, all of the environment variables are listed. The default value, if it exists, is listed under the Value heading. To update a value for a particular variable, enter the new value in the text box in that variable's row.
9. When you have updated all of the variable values that you want, click **OK**.
10. Restart the OC4J instance for the new settings to take effect.

19.2.3.2.2 Setting Values Manually

To set variable values manually, do the following:

1. Open the `orion-web.xml` file in a text editor. If the file does not exist, then you must create it. For a full Oracle Application Server installation, you can locate this file in:

   ORACLE_HOME\j2ee*OC4J_instance*\application-deployments*deployment_name*

   For a standalone instance of OC4J, you can locate it in:

   ORACLE_HOME\j2ee\home\application-deployments\jpdk\jpdk

2. For each deployment property you want to set, add the following entry:

   ```
   <env-entry-mapping name="jndi_var_name">value</env-entry-mapping>
   ```

3. Save and close the file. When complete, your file should look something like the following:

   ```
   <orion-web-app
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation=
         "http://xmlns.oracle.com/oracleas/schema/orion-web-10_0.xsd"
     deployment-version="10.1.3.0.0"
     deployment-time="1134715871939"
     default-charset="iso-8859-1"
     jsp-cache-directory="./persistence"
     jsp-cache-tlds="standard"
     autojoin-session="true"
     temporary-directory="./temp"
     servlet-webdir="/servlet/"
     context-root="newapp1p"
     schema-major-version="10" schema-minor-version="0" >
       <!-- Uncomment this element to control web application class loader
            behavior.
       <web-app-class-loader search-local-classes-first="true"
            include-war-manifest-class-path="true" />
   <web-app>
       -->
   <env-entry-mapping name="oracle/portal/sample/rootDirectory">
             D:\prefs</env-entry-mapping>
   ```

```
    <env-entry-mapping name="oracle/portal/sample/definition">
              D:\definitions\def.xml</env-entry-mapping>
    </web-app>
    </orion-web-app>
```

4. Restart the OC4J instance for the new settings to take effect.

19.2.3.3 Retrieving JNDI Variables

JNDI is a standard J2EE technology. As such, you can access JNDI variables through J2EE APIs. For example:

```
String myVarName = "oracle/portal/myProvider/myVar"
String myVar = null;
try
{
   InitialContext ic = new InitialContext();
   myVar = (String)ic.lookup("java:env/" + myVarName);
}
catch(NamingException ne)
{
   exception handling logic
}
```

In addition to the basic J2EE APIs, PDK-Java includes a simple utility class for retrieving the values of variables defined and used by the PDK itself. These variables all conform to the naming convention described in Section 19.2.3.1.2, "Variable Naming Conventions" and are of the form:

```
oracle/portal/provider_service_name/variable_name
oracle/portal/variable_name
```

To use these APIs, you need only provide the *provider_service_name* and the *variable_name*. The utilities construct the full JNDI variable name, based on the information you provide, and look up the variable using code similar to that shown earlier and return the value of the variable.

The `EnvLookup` class (`oracle.portal.utils.EnvLookup`) provides two `lookup()` methods. One retrieves producer variables and the other retrieves non-producer variables. Both methods return a `java.lang.Object`, which can be cast to the Java type you are expecting.

The following code example illustrates the retrieval of a producer variable:

```
EnvLookup el = new EnvLookup();
String s = (String)el.lookup(myProviderName, myVariableName);
```

`myProviderName` represents the service name for your producer, which makes up part of the variable name. `myVariableName` represents the portion of the variable name that would come after the producer's service name. The example assumes the variable being retrieved is of type `java.lang.String`.

To retrieve a non-producer variable, you use the same code, you pass only one parameter, the variable name, to the `lookup()`, again excluding the `oracle/portal` prefix.

```
EnvLookup el = new EnvLookup();
Object o = el.lookup(myVariableName);
```

Table 19–1 shows the JNDI variables provided by default with PDK-Java. If you do not declare these variables, then PDK-Java looks for their values in their original locations (`web.xml` and the deployment properties file).

Table 19–1 PDK-Java JNDI Variables

Variable	Description
`oracle/portal/provider/`*`provider_name`*`/autoReload`	Boolean auto reload flag. Defaults to true.
`oracle/portal/provider/`*`provider_name`*`/definition`	Location of producer's definition file.
`oracle/portal/provider/global/log/logLevel`	Log setting (0 through 8). 0 being no logging and 8 the most possible logging.
`oracle/portal/provider/`*`provider_name`*`/maxTimeDifference`	Producer's HMAC time difference.
`oracle/portal/provider/<service_name>/resourceUrlKey`	Authentication key for resource proxying through the Parallel Page Engine. See *Oracle Application Server Portal Configuration Guide* for more information.
`oracle/portal/provider/`*`provider_name`*`/rootDirectory`	Location for producer personalizations. No default value.
`oracle/portal/provider/`*`provider_name`*`/sharedKey`	HMAC shared key. No default value.
`oracle/portal/provider/`*`provider_name`*`/showTestPage`	(non-producer) A Boolean flag that determines if a producer's test page is accessible. Defaults to true.
`oracle/portal/provider/global/transportEnabled`	A Boolean flag that determines whether Edit Defaults personalizations may be exported and imported.

19.2.4 Accessing Session Information

When a user accesses a page, it initiates a public, unauthenticated session and tracks information about the session across requests. If the user logs in, then this session becomes an authenticated session of the logged-in user. This session terminates when any of the following occur:

- The browser session terminates (that is, the user closes all the browser windows).
- The user explicitly logs out.
- The session times out because the user's idle time exceeds the configured limit.

As part of the metadata generation, all of the producers that contribute portlets to the page are contacted, if they specified during registration that they be called for some special processing. This call enables producers to do processing based on the user session, log the user in the producer's application if needed, and establish producer sessions. For producers, this call is referred to as `initSession`. As most Web-enabled applications track sessions using cookies, this API call enables the producer of the application to return cookies.

You can use the session store to save and retrieve information that persists during the portal session. This information is only available, and useful, to you during the life of the session. You should store only temporary information in the session store.

Application developers may use the session store to save information related to the current user session. Data in the session store can be shared across portlets.

If the information you want to store must persist across sessions, then you may want to store it in the preference store instead. Some common applications of the session store are as follows:

- To cache data that is expensive to load or calculate (for example, search results).
- To cache the current state of a portlet (for example, the current range, or page, of search results displayed in the portlet, or sequence of events performed by user).

> **Note:** If you need to replicate session state across middle tiers, then you must mark the Web application as distributable and create a cluster island for your OC4J servers. For more information, see *Oracle Containers for J2EE Servlet Developer's Guide*.

Before you implement session storage, you should carefully consider the performance costs. Because portlets and producers are remote, it can be a relatively expensive operation to create and maintain even a small amount of information in the session store. For this reason, you may want to avoid altogether any session storage for public pages that are accessed frequently by many users.

Furthermore, while using the session store with producers, you create a stateful application that needs to track state information in memory. Similarly, you create a stateful application if you use the file-system implementation of preference store.

If scalability is an important concern for you, then a stateful application may cause you problems. Stateful applications can affect the load-balancing and failover mechanism for your configuration. Even though you may deploy multiple middle-tiers, you must implement sticky routing (where the same node handles subsequent requests in the same session) to track state. Sticky routing may result in lopsided load-balancing or loss of session data in case a node crashes, affecting failover. This issue is one reason why many developers prefer to build stateless applications. However, if scalability is not a concern, then a stateful application should present no problems for you.

In the example in this section, session storage is used to count the number of times your portlet has rendered in Shared Screen mode.

19.2.4.1 Assumptions

The following assumptions are made to perform the tasks in this section:

1. You have followed through and understood Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

19.2.4.2 Implementing Session Storage

The PDK Framework represents the session with a `ProviderSession` object, which is established during the call to the Provider Instance's `initSession` method. This object is associated with the `ProviderUser`. To make data persistent between requests, you need to write data into the session object using the `setAttribute` method on the `ProviderSession` object. This method maps a `java.lang.Object` to a `java.lang.String` and stores that mapping inside the session object. The `String` can then be used to retrieve the `Object` during a subsequent request, provided the session is still valid.

A producer session may become invalid for the following reasons:

- The session times out.
- The `invalidate` method on `ProviderSession` is called.
- The JVM process running the servlet container is terminated.

All portlets contained by the same `ProviderInstance` share the same session for a particular `ProviderUser`. Therefore, data unique to a particular portlet instance must be mapped to a unique `String` in the session. This is accomplished using the `portletParameter` method in the `PortletRendererUtil` class. This method makes a supplied `String` parameter or attribute name unique to a `PortletInstance`, by prefixing it with a generated identifier for that instance. You can use the returned instance-specific name to write portlet instance data into the session.

For more detailed information about the PDK Framework classes, see the JavaDoc on OTN by clicking **Java Doc API** on the Portlet Development page available at

`http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html`

To implement session storage, you need to perform the following tasks:

- Import `ProviderSession`, `PortletRendererUtil`, and `HttpPortletRendererUtil`.
- Retrieve the producer session.
- Read and write the session by accessing it from within your Java portlet.
- Set the session to true in `provider.xml`.
- Register the producer for session storage and set the Login Frequency.

The steps that follow describe how to add a session count to your portlet that displays how many times the portlet has been rendered for the current session.

1. After using the wizard to create a portlet, you can edit the JSP for the Show page in Oracle JDeveloper. You need to import the following classes:

```
<%@page contentType="text/html; charset=windows-1252"
import="oracle.portal.provider.v2.render.PortletRenderRequest"
import="oracle.portal.provider.v2.http.HttpCommonConstants"
import="oracle.portal.provider.v2.ProviderSession"
import="oracle.portal.provider.v2.render.PortletRendererUtil"
import="oracle.portal.provider.v2.render.http.HttpPortletRendererUtil"
%>
```

2. Insert code that checks for a valid session first and then increments the count and displays it. If the session is valid and a previously stored value exists, then you display the value, increment the count, and store the new value. If the session is valid but no previously stored value exists, then you initialize a new count starting with 1, and display and store the value. You also want to obtain the unique string key for this portlet and then use an it in an array to count the session. If no session information was received, then you want to provide information to the user indicating they may need to log back in.

```
<%
PortletRenderRequest pReq = (PortletRenderRequest)
request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
ProviderSession pSession = pReq.getSession();
  if (pSession != null)
```

```
  {
    String key = PortletRendererUtil.portletParameter(pReq, "count");
    Integer i = (Integer)pSession.getAttribute(key);
    if (i == null)
    {
      i = new Integer(0);
    }
    i = new Integer(i.intValue()+1);
    pSession.setAttribute(key, i);
%>

<p>Render count in this session: <%=i%> </p>

<%
  }
  else
  {
%>

<p>The session has become invalid</p>
<br>
Please log out and log in again.
<%
  }
%>
```

3. By default, the wizard does not set session to true in `provider.xml`. You need to update this flag in order for the producer to receive session information from the portal. You should only set this tag to true if you are using session information in your producer or portlets. By setting this flag to true, extra load is added to the producer calls.

   ```
   <provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
   <session>true</session>
   ```

   For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

   http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

4. Register your producer with session support. For a reminder on how to register your portlet, see Section 18.10, "Registering and Viewing Your Portlet".

19.2.4.3 Viewing the Portlet

If you have not already added your Java portlet to a page, then do so now. Ensure that you perform the following tasks:

- Refresh the producer to accept the new changes.
- Re-login in case your session is no longer valid.

19.2.5 Implementing Portlet Security

This section describes the available security services for your Java portlet.

For more detailed information about the PDK classes referred to in this section, see the JavaDoc on OTN by clicking **Java Doc API** on the Portlet Development page available at

http://www.oracle.com/technology/products/ias/portal/portlet_development_10g1014.html

19.2.5.1 Assumptions

The following assumptions are made to perform the tasks in this section:

1. You have followed through and understood Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

19.2.5.2 Introduction to Portlet Security Features

This section introduces the major features that are available to secure your portlet producers.

19.2.5.2.1 Authentication When a user first logs in, they must enter their password to verify their identity and obtain access. See Chapter 10, "Securing Your WebCenter Application" for more information.

Once the user is authenticated, the producer code has access to the authenticated user's identity from the `PortletRenderRequest` that is available from the `HttpServletRequest` object as follows:

```
PortletRenderRequest pr = (PortletRenderRequest)request.getAttribute
   (HttpCommonConstants.PORTLET_RENDER_REQUEST);
   String userName = pr.getUser().getName();
```

When using this user identity for sensitive operations, it is important to ensure that you have configured this producer to use basic message authentication to secure the integrity of the identity assertion.

You also have the option of configuring the OC4J containing the JPDK producer to use JAZN-LDAP and then use J2EE security, in which case you can obtain the user's identity with the following method, again through the `HttpServletRequest` object:

```
String userName = request.getRemoteUser();
```

And if you need the user `Principal`, you can also use:

```
Principal userPrincipal = request.getUserPrincipal();
```

When configuring the producer's OC4J container in this manner, make sure to configure this producer to use enhanced message authentication to secure the integrity of this form of identity assertion.

19.2.5.2.2 Authorization Authorization determines if a particular user may view or interact with a portlet. For more information about authorization checking, see Chapter 10, "Securing Your WebCenter Application".

19.2.5.2.3 Communication Security To this point, user authentication and authorization are covered, which do not check the authenticity of messages received by a producer. To completely secure your producers, secure the communication with a producer. If the communication is not secured, then it is possible for someone to imitate an application instance and fool the producer into returning sensitive information. There are three types of communication security:

- **Server Authentication** restricts access to a producer to a small number of recognized computers. This method compares the IP address or the host name of an incoming HTTP message with a list of trusted hosts. If the IP address or host

name is in the list, then the message is passed to the producer. If not, it is rejected before reaching the producer. See Section 19.2.5.5, "Server Security" for more information.

- **Message Authentication** appends a checksum based on a shared key to producer messages. When a message is received by the producer, the authenticity of the message is confirmed by calculating the expected value of the checksum and comparing it with the actual value received. If the values are the same, then the message is accepted. If they are different, then the message is rejected without further processing. The checksum includes a time stamp to reduce the chance of a message being illegally recorded in transit and resent later. See Section 19.2.5.6, "Message Authentication" for more information.
- **Message Encryption** relies on the use of the HTTPS protocol for communication between applications and producers. Messages are strongly encrypted to protect the data therein. Encryption provides a high level of security, but it incurs a performance penalty due to the additional processing required for each message.
- **User Input Escape** causes the application to escape any user input strings and treat them as text only to protect against XSS attacks, where an attacker attempts to pass in malicious scripts through user input forms. For example, if a portlet title is customizable, then an attacker might attempt to pass scripts or commands to the portlet through the title parameter entry. For this reason, the default behavior is to escape user input and thus disable any incoming scripts. See Section 19.2.5.7, "User Input Escape" for more information.

19.2.5.3 Single Sign-On

Portlets act as windows into an application. They display summary information and provide a way to access the full functionality of the application. Portlets display some portions of the application in the WebCenter application and typically enable the user to perform some application tasks.

> **Note:** Deep links from the portlet into the application are not supported in Oracle WebCenter Framework. Users can only access what is presented within the application portlet itself.

An application may need to authenticate the user accessing the application through the portlet. Following are the possible application authentication methods:

- External Application. In this case, the Oracle Application Server Portal (OracleAS Portal) user is different from the application user, but the application user name and password are managed by the OracleAS Portal user.
- No Application Authentication. In this case, the communication between producer and OracleAS Portal is not protected at all.

19.2.5.3.1 External Application An external application uses a different authentication server than the WebCenter application. This means that when a user is already logged into the WebCenter application, you want to also log them into the external application without having to type in their user name or password.

> **Note:** Deep links from the portlet into the application are not supported in Oracle WebCenter Framework. Users can only access what is presented within the application portlet itself.

Applications that manage the authentication of users can be loosely integrated with Oracle Single Sign-On if the administrator registers them as external applications. When a user who was previously authenticated by Oracle Single Sign-On accesses an external application for the first time, Oracle Single Sign-On attempts to authenticate the user with the external application. The authentication process submits an HTTP request that combines the registration information and the user's user name and password for the application. If the user has not yet registered their user name and password for the external application, then Oracle Single Sign-On prompts the user for the required information before making the authentication request. When a user supplies a user name and password for an external application, Oracle Single Sign-On maps the new user name and password to the WebCenter application user name and stores them. They will be used the next time the user needs authentication with the external application.

The advantages of an external application implementation are as follows:

- Enables integration with many portals. If, however, one of the portals is preferred over the others, then the application could be integrated as a partner application of that preferred portal and an external application of the others.
- Provides a single sign-on experience for users. However, users still must maintain different user names and passwords. In addition, the external application user name mapping must be maintained.
- Enables integration with multiple portals independent of their user repositories and Oracle Single Sign-On.
- Avoids the requirement of having access to the application source code.

The disadvantages of an external application implementation are as follows:

- Does not share the same user repository as the portal, which requires additional maintenance of user information by the end user.
- Transmits the user name and password to the producer in plain text, unless you implement Secure Sockets Layer (SSL).

19.2.5.3.2 No Application Authentication The producer trusts the WebCenter application instance sending the request completely. The producer can determine if the user is logged in and the portal user name, but the application has not authenticated the user.

The advantages of no application authentication are as follows:

- Provides the easiest form of integration and the fastest to implement.

The disadvantages of no application authentication are as follows:

- Provides the least security.
- Provides the weakest integration with the WebCenter application.

19.2.5.4 Portlet Security Managers

Portlet security managers are implemented within a producer to verify that a given user may view an instance of the portlet. When a user views a page with a portlet instance on it, security managers determine whether the user has the appropriate privileges to see the portlet. Implementing access control methods in the producer restricts the retrieval of content from a portlet (that is, hides the portlet) from users without the appropriate privileges. Only if the specified characteristics, such as user details and preferences, pass the authorization logic will the content be retrieved for the user. If no portlet security methods are implemented in the producer, then any user name may be passed in, even fictitious, unauthenticated ones.

A producer can implement two portlet security methods as follows:

- Get a list of portlets.
- Verify the accessibility of the portlet.

Portlets have access to the WebCenter application user privileges and groups of which the user is a member. The following information can be used by the security methods:

- The default group of the user
- The privileges of a user or group
- The highest available privilege of a user across all groups
- The objects the user can access (only in database producers)

`AuthLevelSecurityManager` has access to the following information about authorization level:

- Strongly authenticated.

  The user has been authenticated by Oracle Single Sign-On in the current WebCenter application session, that is, the user logged in with a valid user name and password, and requested the portlet in the context of that session.

- Weakly authenticated.

  A user who was previously strongly authenticated returns to view a page without an active session. A persistent cookie (maintained by the user's browser) indicates that in some previous session the user logged on with a valid user name and password.

- Public or not authenticated.

  The user has not logged in within the context of the current session, and does not have a persistent cookie to indicate that such a state previously existed.

To incorporate these security services into your Java portlet, you simply need to update `provider.xml` and set the security level to strong, weak, or public. Place the following XML right before the `</portlet>` tag in `provider.xml`:

```
<securityManager class="oracle.portal.provider.v2.security.AuthLevelSecurityManager">
   <securityLevel>strong</securityLevel>
</securityManager>
```

After you make this change to `provider.xml`, refresh the producer.

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

The advantage of security methods is as follows:

- You can enable a portlet to produce different output depending on the level of authorization.

The disadvantage of security methods is as follows:

- Most security manager implementations will use the authorization level or some other user specific element in an incoming message. A check of this type could be bypassed by an entity imitating a WebCenter application instance.

19.2.5.4.1 Viewing the Portlet After you add a security manager to a portlet, you can validate it by following these steps:

1. Ensure you are logged in to a WebCenter application instance with privileges to create pages and add portlets to a page.
2. Create a new page, ensuring it is visible to PUBLIC.
3. Add your Java portlet to the page.
4. Make a note of the direct URL to your new page.
5. Now log out of the WebCenter application instance by clicking the **Logout** link.
6. Directly access the page by entering the URL noted in Step 4 into your browser's address bar.

You will see the page created in Step 2 but not the portlet added in Step 3. When you added the portlet to the page, you were logged in and hence strongly authenticated. The PDK run time detected this and enabled you to add the portlet. When you logged out and viewed the page, you were no longer strongly authenticated and hence the PDK Framework did not enable rendering of the portlet's contents.

If you log in again and view the page, then you will see that the portlet is still there.

19.2.5.4.2 Implementing Your Own Security Manager If your portlet requires special security arrangements which are not provided by the implementations shipped with the PDK, then you will need to supply your own custom `PortletSecurityManager` controller class. To do this, extend the `oracle.portal.provider.v2.security.PortletSecurityManager` class and supply implementations for the two methods specified by the interface. Then replace the class attribute of the `securityManager` controller element in the XML producer definition with you new class name and configure child elements appropriately.

19.2.5.5 Server Security

One way to prevent unauthorized access to producers is to restrict access to the producer to known client computers at the server level. Because only the identified clients may access the producer, this method defends against denial of service attacks.

The way in which HTTP servers handle this kind of security tends to vary. You need to investigate your particular HTTP server to understand how to implement this level of security.

The advantages of server security are as follows:

- It limits access to the producer to trusted hosts only.
- It simplifies configuration.

The disadvantages of server security are as follows:

- Oracle Web Cache does not have IP address checking capability. If Oracle Web Cache sits in front of a producer, then you have no protection from a client on any host sending show requests to Oracle Web Cache.
- Restricting access to certain IP addresses and host names may be circumvented by sending messages to a producer containing fake IP addresses and host names. This trick is difficult to perform effectively as return messages go to the computer whose IP address was copied, but it can still cause problems.

19.2.5.6 Message Authentication

PDK-Java supports message authentication so that access may be limited to a specified producer instance or group of producer instances. A producer is registered with a

secret shared key known only to the WebCenter application and producer administrators.

The WebCenter application sends a digital signature, calculated using a Hashed Message Authentication Code (HMAC) algorithm, with each message to a producer. A producer may authenticate the message by checking the signature using its own copy of the shared key. This technique may be used in Secure Sockets Layer (SSL) communication with a producer instead of client certificates.

The WebCenter application calculates a signature based on user information, a shared key and a time stamp. The signature and time stamp are then sent as part of the SOAP message. The time stamp is based on UTC (coordinated universal time, the scientific name for Greenwich Mean Time) so that time stamps can be used in messages between computers in different time zones.

When the producer receives this message it generates its own copy of the signature. If the signatures agree, it will then compare the message time stamp with the current time. If the difference between the two is within an acceptable value, then the message is considered authentic and is processed accordingly.

A single producer instance cannot support more than one shared key because it could cause security and administration problems. For instance, if one copy of the shared key is compromised in some way, then the producer administrator has to create a new key and distribute it to all of the application clients, who then must update their producer definitions. The way around this problem is to deploy different producer services, specifying a unique shared key for each service. Each producer service has its own deployment properties file so that each service is configured independently of the others. The overhead of deploying multiple producer services within the same producer adapter is relatively small.

In a producer without Oracle Web Cache in front of it, this use of the same signature cookie over the lifetime of a producer session implies a trade-off between performance and the security provided by authenticating the requests. The signature cookie value is only calculated once after the initial SOAP request establishes the session with the producer. The shorter the producer session timeout, the more often a signature will be calculated providing greater security against a show request being resent illegally. However, the SOAP request required to establish a session incurs a time penalty.

In a producer using Oracle Web Cache to cache show request responses, you have a similar trade-off. Cached content is secured in the sense that incoming requests must include the signature cookie to retrieve it, but caching content for an extended period of time leaves the producer open to illegal show requests.

While the signature element provides protection against interception and resending of messages, it does nothing to prevent interception and reading of message contents. Messages are still transmitted in plain text. If you are concerned about the content of messages being read by unauthorized people, then you should use message authentication in conjunction with SSL.

The advantage of message authentication is as follows:

- Ensures that the message received by a producer comes from a legitimate WebCenter application instance.

The disadvantages of message authentication are as follows:

- Causes administration problems if a producer serves more than one portal.
- Entails performance implications if made very secure by having a short session timeout.

19.2.5.7 User Input Escape

By accepting user input without escaping it to text, you run the risk of an XSS attack, where an attacker attempts to pass in malicious scripts through user input forms. For example, if a portlet title is customizable, then an attacker might attempt to pass scripts or commands to the portlet through the title string. PDK-Java provides the following features to ensure that you can protect your portlets from such attacks:

- Default Container Encoding
- Escape Methods

19.2.5.7.1 Default Container Encoding To prevent any script inside a portlet title from being executed, the framework default container renderer class encodes any script characters. This default behavior is controlled through a JNDI variable, `escapeStrings`. When set to `true`, the markup tags in portlet titles are rendered as visible tag characters. For example, a title customization of `<i>title</i>` will be rendered as `<i>title</i>` not *title*. This mode is secure, but, if it is not the desired behavior, then you can set `escapeStrings` to `false` for that producer.

`escapeStrings` applies to all logical producers within a producer. You can set the value of `escapeStrings` from the Application Server Control Console as you would any other JNDI variable. See Section 19.2.3.2, "Setting JNDI Variable Values" for more information.

19.2.5.7.2 Escape Methods If you have code that renders customized values, then you need to ensure that you escape those input values appropriately to avoid XSS attacks. This requirement applies to code for rendering pages in any mode. PDK-Java supplies two new static methods for this purpose. They are in the Java class `oracle.portal.provider.v2.url.UrlUtils`, and can be described as follows:

- `public static escapeString(`*`string_text`*`)` escapes any script characters in a given string. For example, less than < becomes `&lt`. This method is unaffected by the `escapeStrings` JNDI variable and is the secure, recommended method to use.
- `public static escapeStringByFlag(`*`string_text`*`)` escapes any script characters in a given string. This method is controlled by the `escapeStrings` JNDI variable and is therefore less secure and not the recommended method to use.

For example:

```
title = UrlUtils.escapeString(data.getPortletTitle());
```

19.2.6 Enhancing Portlet Performance with Caching

In the previous sections of this chapter, you learned how to write fully functional Java portlets using the PDK Framework. Once you complete the basic functionality of your portlet, you may want to turn your attention to portlet performance.

Caching is a common technique for enhancing the performance of Web sites that include a great deal of dynamic content. The overhead involved in retrieving data and generating the output for dynamic content can be significantly reduced by proxying requests through a local agent backed by a large, low-latency data store known as a cache. The cache agent responds to a request in one of two ways, as follows:

- If a valid version of the requested content exists in the cache, then the agent simply returns the existing cached copy, thus skipping the costly process of content retrieval and generation. This condition is called a cache hit.

- If a valid version of the requested content does not exist in the cache, then the agent forwards the request to its destination and awaits the return of the content. The agent returns the content to the requestor and stores a local copy in its cache for reuse if a subsequent request for the same content arises. This condition is called a cache miss.

Producers generate dynamic content (that is, portlets) and they reside remotely from the WebCenter application instance on which they are deployed. As such, caching might improve their performance. The architecture lends itself well to caching. You can cache the portlets rendered by your producer and reuse the cached copies to handle subsequent requests, minimizing the overhead your producer imposes on page assembly.

The producer can use any one of three different caching methods, depending upon which one is best suited to the application. The methods differ chiefly in how they determine whether content is still valid. Following are the three caching methods:

1. **Expiry-based Caching**: When a producer receives a render request, it stamps its response with an expiry time. The rendered response remains in the cache and fills all subsequent requests for the same content until its expiry time passes. This caching scheme is perhaps the simplest and most performant because the test for cache validity requires very little overhead and does not involve any network round trips. Expiry-based caching suits applications where the content has a well-defined life span. For content with a less certain life span, however, expiry-based caching is less effective. See Section 19.2.6.2, "Activating Caching" and Section 19.2.6.3, "Adding Expiry-Based Caching" for more information.

2. **Invalidation-based Caching:** Invalidation-based caching works essentially the same way as expiry-based caching, except that the contents of the cache can expire or become invalid at any time. Invalidation of cache content is usually triggered by an event.

   For example, if you have a calendar portlet that shows your appointments for the day, then the content for the portlet could be generated once, the first time you show the calendar for that day. The content remains cached until something happens to change your schedule for that day, such as the addition of an appointment, the deletion of an existing appointment, or a change of time for an appointment. Each of these change events can trigger an action in the calendar application. When such an event occurred, your calendar application can generate an invalidation request for any cached portlet content affected by the change. The next time you view a page containing your calendar portlet, the cache will not contain an entry. Your producer will be contacted to regenerate the new content with the modified schedule.

   This method is a very efficient way to cache content because the originator of the content, that is, your producer, is contacted only when new content needs to be generated, but you are not bound to a fixed regeneration schedule. See Section 19.2.6.2, "Activating Caching" and Section 19.2.6.4, "Adding Invalidation Based Caching" for more information.

3. **Validation-based Caching**: When a producer receives a render request, it stamps its response with a version identifier (or E Tag). The response goes into the cache, but, before the PPE can reuse the cached response, it must determine whether the cached version is still valid. It sends the producer a render request that includes the version identifier of the cached content. The producer determines whether the version identifier remains valid. If the version identifier is still valid, then the producer immediately sends a lightweight response to the PPE without any content, which indicates the cached version can be used. Otherwise, the producer generates new content with a new version identifier, which replaces the

previously cached version. In this form of caching, the PPE must always confirm with the producer whether the content is up to date. The validity of the cached copy is determined by some logic in the producer. The advantage of this approach is that the producer controls the use of the cached content rather than relying on a fixed period of time. See Section 19.2.6.2, "Activating Caching" and Section 19.2.6.5, "Adding Validation-Based Caching" for more information.

19.2.6.1 Assumptions

The following assumptions are made to perform the tasks in these sections:

1. You have followed through and understood Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

19.2.6.2 Activating Caching

To use the caching features in your producers, you must first activate the middle tier cache. This cache is known as the PL/SQL Cache because it is the same cache used by mod_plsql, the Oracle HTTP Server plug-in that calls database procedures, and hence database producers, over HTTP.

Usually, your OracleAS Portal administrator is responsible for the configuration details of caching.

For invalidation-based caching, you need to configure Oracle Web Cache in front of the producer. Bear in mind that remote producers often do not have Oracle Web Cache installed. For more information about Oracle Web Cache, see the *Oracle Application Server Web Cache Administrator's Guide*.

Once you have installed and configured Oracle Web Cache, ensure the following in the Oracle Web Cache Manager:

- It points to the host name and port of the producer.
- Caching rules do not cause the caching of producer content. Content should be cached according to the surrogate control headers generated by the producer in its response.

19.2.6.3 Adding Expiry-Based Caching

Expiry-based caching is one of the simplest caching schemes to implement, and can be activated declaratively in your XML producer definition. You can set an expiry time for the output of any `ManagedRenderer` you use by setting its `pageExpires` property to the number of minutes you want the output to be cached for. For example, suppose you want portlet output to be cached for one minute. To add expiry-based caching, perform the following steps:

1. After you have used the Portlet Wizard to build a portlet as described in Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper", edit the `provider.xml` file and set the `pageExpires` property tag of showPage to 1. This will set an expires entry of 1 minute for the portlet.

   By default the wizard generates a standard and compressed tag for `showPage`. You need to expand the tag to include a subtag of `pageExpires`:

```
<showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
   <resourcePath>/htdocs/mycacheportlet/MyCachePortletShowPage.jsp
      </resourcePath>
   <pageExpires>1</pageExpires>
</showPage>
```

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

2. Test that the portlet is cached for 1 minute by adding some JSP code to your show page. You can simply add the current time to your JSP.

```
<%@page contentType="text/html; charset=windows-1252"
   import="oracle.portal.provider.v2.render.PortletRenderRequest"
   import="oracle.portal.provider.v2.http.HttpCommonConstants"
   import="java.util.Date"
   import="java.text.DateFormat"
%>

<%
 PortletRenderRequest pReq = (PortletRenderRequest)
   request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
 DateFormat df = DateFormat.getDateTimeInstance(DateFormat.LONG,
   DateFormat.LONG,pReq.getLocale());
 String time = df.format(new Date());
%>

<P>Hello <%=pReq.getUser().getName() %>.</P>
<P>This is the <b><i>Edit</i></b> render mode!</P>
<P>This information is correct as of <%=time%>.</P>
```

When viewing the portlet, you see that the time (including seconds) is constant for 1 minute. After the time has expired, the portlet displays the most current time and a new cache is set.

19.2.6.4 Adding Invalidation Based Caching

When using Oracle Web Cache, requests for content are sent using HTTP and content is either returned from the cache or the HTTP request is forwarded to the originator of the content. When content is returned to Oracle Web Cache, it is added to the cache before being returned to the requestor. Cache content is invalidated by sending invalidation requests directly to Oracle Web Cache. PDK-Java uses the Java API for Web Cache (JAWC) to generate invalidation requests. This section describes how to configure Oracle Web Cache and use the invalidation-based caching sample that comes with PDK-Java.

Not all requests sent to Oracle Web Cache are cached. In order for the content to be cached, the content must include directives that tell Oracle Web Cache to cache the content. Usually Oracle Web Cache uses the URL associated with the request as the cache key, but you can specify additional keys by setting special HTTP headers, known as surrogate control headers, on the response.

To configure a Java portlet to use invalidation-based caching, you do the following:

- Configuring Oracle Web Cache. See *Oracle Application Server Web Cache Administrator's Guide* for more information.
- Section 19.2.6.4.1, "Configuring the Producer Servlet"
- Section 19.2.6.4.2, "Defining the Oracle Web Cache Invalidation Port"
- Section 19.2.6.4.3, "Configuring the XML Producer Definition"
- Section 19.2.6.4.4, "Manually Invalidating the Cache"

19.2.6.4.1 Configuring the Producer Servlet To enable invalidation-based caching, you must switch it on at the producer servlet level. The flag is set in an initialization parameter inside the PDK-Java Web application deployment descriptor, `web.xml`. For example:

```
<servlet>
...
    <servlet-class>oracle.webdb.provider.v2.adapter.SOAPServlet</servlet-class>
    <init-param>
      <param-name>invalidation_caching</param-name>
      <param-value>true</param-value>
    </init-param>
</servlet>
```

If the flag is not defined here, then invalidation-based caching is switched off. The status of this flag can be checked by displaying the producer's test page. If the `testPageURI` property is not set in the `sample.properties` file, then the producer code displays the test page in the old default style. The old style test page correctly picks up the invalidation caching flag from the `web.xml` file. The format of the test page URL is as follows:

```
http://provider_hostname:port/jpdk/providers/sample
```

19.2.6.4.2 Defining the Oracle Web Cache Invalidation Port If you are using an Oracle Application Server installation type where PDK-Java was automatically preinstalled, then you should find that Oracle Web Cache invalidation settings have already been preconfigured in `MID_TIER_ORACLE_HOME/portal/conf/cache.xml`. In this case, you can safely ignore the instructions in this section and proceed to Section 19.2.6.4.3, "Configuring the XML Producer Definition". Otherwise, you will need to configure the invalidation portlet for Oracle Web Cache.

First, you need the user name and password for the invalidation ports of the Oracle Web Cache instances associated with your application server. After you obtain those, use the provided `oracle.webdb.provider.v2.cache.Obfuscate` Java utility to convert the user name and password into an obfuscated string of the format required by the JAWC API. In a default Oracle Application Server installation, the user name for the invalidation port is usually `invalidator` and the password is usually the same as the application server administrator's password. For example, suppose you installed Oracle Application Server in `D:\as904`, with an invalidation port user name of `invalidator` and a password of `welcome`. You would run the following command:

> **Note:** The command that follows assumes that `pdkjava.jar` and `jawc.jar` are present in *`ORACLE_HOME`*`/j2ee/home/shared-lib/oracle.jpdk/1.0`, as required by the PDK-Java installation guide.
>
> If you are using a standalone OC4J installation, then you need to substitute in the path to the java executable an external Java 2 SDK installation.

```
D:\as904\jdk\bin\java -classpath
D:\as904\j2ee\home\shared-lib\oracle.jpdk\1.0\pdkjava.jar
  oracle.webdb.provider.v2.cache.Obfuscate invalidator:welcome
```

If successful, the command should print a hexadecimal string like the following:

```
0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11
```

Now, use this information to create a JAWC configuration file, `cache.xml`, which specifies one or more Oracle Web Cache instances and their invalidation ports. For example:

```
<?xml version="1.0"?>
<webcache>
<invalidation
 host="cache.mycompany.com"
 port="4001"
authorization="0510198d5df8efd5779406342be2528aa0cccb179ea6b77baf49f019f5075a3a11"/>
</webcache>
```

JAWC finds this configuration file using the system property `oracle.http.configfile`. To set this system property for an OC4J instance within an Oracle Application Server installation, simply add an appropriate line to the `oc4j.properties` file for the particular instance in which PDK-Java is installed (for example, `MID_TIER_ORACLE_HOME/j2ee/OC4Jinstance/config/oc4j.properties`) and then restart that instance. For example:

```
oracle.http.configfile=fully_qualified_filename
```

If you are running OC4J standalone, then you can specify the option in the command line you use to start it.

```
java -Doracle.http.configfile=<fully_qualified_filename> -jar oc4j.jar
```

> **Note:** As the location of the cache configuration file is defined as a system property, only one cache may be defined for each server instance. It is not possible to have two different producer instances behind separate Oracle Web Cache configurations.

19.2.6.4.3 Configuring the XML Producer Definition Using a combination of tags in `provider.xml`, you can activate automatic invalidation-based caching for an entire portlet or some of its pages. To turn on automatic invalidation-based caching, you need to declare it as follows either at the level of individual pages or the renderer, to indicate that invalidation-based caching should be activated for all pages:

```
<useInvalidationCaching>true</useInvalidationCaching>
```

If your portlet supports personalization (through the Edit or Edit Defaults modes), then you may also want to declare `<autoInvalidate>true</autoInvalidate>` for your renderer. This declaration causes invalidation of all the cached content for the portlet when you click OK or Apply in Edit mode, thus causing new markup to be generated based on the personalized settings.

The maximum time for holding the page in the cache is the minimum of the following:

- Maximum expiry time defined in the caching system.
- producer default (24 hours) if no maximum expiry time is specified.
- The time in minutes of the `<pageExpires>` tag, if present.

The following excerpt from `provider.xml` specifies that this portlet shall be cached for up to 5 minutes and shall be automatically invalidated upon personalization:

```
<renderer class="oracle.portal.provider.v2.render.RenderManager">
 <contentType>text/html</contentType>
 <renderContainer>true</renderContainer>
```

```
 <autoRedirect>true</autoRedirect>
 <autoInvalidate>true</autoInvalidate>
 <showPage class="oracle.portal.provider.v2.render.http.ResourceRenderer">
  <resourcePath>/htdocs/invalidation/invalidation1.jsp</resourcePath>
  <useInvalidationCaching>true</useInvalidationCaching>
  <pageExpires>5</pageExpires>
 </showPage>
 <editPage class="oracle.portal.sample.v2.devguide.invalidation.
   InvalidationEditRenderer"/>
</renderer>
```

> **Note:** The `pageExpires` tag is also used for normal expiry based caching. These two forms of caching are mutually exclusive. Invalidation-based caching occurred in an Oracle Web Cache instance located in the same place as the producer. Pages stored using expiry based caching are cached in the middle tier.

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

19.2.6.4.4 Manually Invalidating the Cache

You may want the cached version of the portlet invalidated when a request is processed or information somewhere has been updated. In these cases, you may want to manually invalidate the cache.

The invalidation-based caching portlet sample included with PDK-Java contains a single portlet that displays the time the content was cached and a link to trigger an invalidation request. The first time a page request is made to the producer through the cache, the response is cached. Subsequent requests for the portlet content are fulfilled by returning content from Oracle Web Cache. When you click the link at the bottom of the portlet an invalidation request is generated by the producer that removes the portlet from the cache. The next request for the portlet is forwarded to the producer and the producer generates a new portlet with the current time.

To perform invalidation calls to Oracle Web Cache, first you need to have a handle to a `ServletInvalidationContext` object. You can get this handle by calling the `static getServletInvalidationContext()` method of the `ServletInvalidationContextFactory` class.

Once you have the handle, you need to specify the cache key. In the cache key, you need to specify whether you want to invalidate a particular portlet instance, all the instances of a portlet, or all the portlets managed by a producer. To perform this task, you use the methods of the `ServletInvalidationContext` class or the methods of its super class, `InvalidationContext`. This is where you can specify whether the portlet should be cached for this particular user (USER level). If there is no user specified in the cache key, then the cached content is returned to all users, which means you are using SYSTEM level caching.

The following example invalidates a portlet instance and calls the `setFullProviderPath()` and `setPortletReference()` methods. When the caching key is set, you call the `invalidate()` method on the `InvalidationContext` object that sends the invalidation message to Oracle Web Cache.

```
ServletInvalidationContext inv =
  ServletInvalidationContextFactory.getServletInvalidationContext();
inv.setFullProviderPath
```

```
  ((ServletPortletRenderRequest)pReq);
inv.setPortletReference
  (pReq.getPortletReference());
int num = inv.invalidate();
```

Review the Web cache sample producer for more information.

19.2.6.5 Adding Validation-Based Caching

Adding validation-based caching requires slightly more effort, but gives you explicit control over exactly which requests to your producer are cache hits. As an example, you may want to update the cache only when data within the portlet has changed. To implement this algorithm, you need to override the `prepareResponse` method. The signature of the `BaseManagedRenderer.prepareResponse` method is:

```
public boolean prepareResponse(PortletRenderRequest pr)
                        throws PortletException,
                               PortletNotFoundException
```

In your version of `prepareResponse()`, you need to do the following:

- Retrieve the cached version identifier set by the PPE in the render request by calling the `HttpPortletRendererUtil.getCachedVersion()` method:

  ```
  public static java.lang.String getCachedVersion
      (PortletRenderRequest request)
  ```

- If the portlet finds the previously cached version valid, then the appropriate header has to be set by calling the `HttpPortletRendererUtil.useCachedVersion()` method. It also instructs the `RenderManager` that it won't be necessary to call `renderBody()` to render the portlet body.

  ```
  public static void useCachedVersion(PortletRenderRequest request)
  ```

  Otherwise, use `HttpPortletRendererUtil.setCachedVersion()` to generate a new version of the portlet, which will be cached. It also indicates to the PPE that the `renderBody()` method has to be called to regenerate the portlet content.

  ```
  public static void setCachedVersion(PortletRenderRequest request,
                                      java.lang.String version,
                                      int level)
                               throws java.lang.IllegalArgumentException
  ```

For validation-based caching, you need not update `provider.xml`. You can view the portlet by refreshing the page or adding the portlet to a page and updating the content. If the content has changed, then the portlet shows the new content. If the content has not changed, then a cached version of the portlet is displayed.

19.3 Testing Portlet Personalization

If you have implemented personalization for your portlet, then the Personalize link will only appear on the portlet for authenticated users. Hence, to test the personalization of a portlet (the Personalize link), you must have some form of security implemented for the application consuming the portlet. For testing purposes, you may prefer to just configure the most basic authentication possible. See Section 10.6, "Configuring Basic Authentication for Testing Portlet Personalization" for more information.

19.4 Building Struts Portlets

This section describes the framework for building Struts portlets with Oracle JDeveloper for use in a WebCenter application. You will learn how to build a Struts portlet from an existing application by adding the Struts Tag Library from the Oracle Application Server Portal Developer Kit (version 9.0.4.0.2 or later) to Oracle JDeveloper, then use the Oracle PDK Java Portlet wizard to create the Java portlet itself. This sections covers the following tasks:

- Section 19.4.1, "The Apache Struts Framework"
- Section 19.4.2, "Creating a Struts Portlet"

19.4.1 The Apache Struts Framework

This section discusses the use of the Apache Struts. Struts is an implementation of the Model-View-Controller (MVC) design pattern. The following topics are discussed in this section:

- Section 19.4.1.1, "Model View Controller Overview"
- Section 19.4.1.2, "Apache Struts Overview"
- Section 19.4.1.3, "OracleAS PDK Integration with Struts"
- Section 19.4.1.4, "Summary"

19.4.1.1 Model View Controller Overview

Enterprise applications undertake several distinct tasks, as follows:

- Data access
- Business logic implementation
- User interface display
- User interaction
- Application (page) Flow

The Model View Controller (MVC) architecture provides a way of compartmentalizing these tasks, based on the premise that activities, such as data presentation, should be separate from data access. This architecture enables you to easily plug a data source into the application without having to rewrite the user interface. MVC enables the logical separation of an application into three distinct layers, the Model, the View, and the Controller.

The Model

The Model is the repository for the application data and business logic. One facet of the Model's purpose is to retrieve data from and persist data to the database. It is also responsible for exposing the data in such a way that the View can access it, and for implementing a business logic layer to validate and consume the data entered through the View. At the application level, the Model acts as the validation and abstraction layer between the user interface and the business data that is displayed. The database server itself is simply a persistence layer for the Model.

The View

The View is responsible for rendering the Model data using JSPs. The View code does not include a hardcoded application or navigation logic, but may contain some logic to carry out tasks like displaying conditional data based on a user role. When an end user

executes an action within the HTML page that the View renders, an event is submitted to the Controller. The Controller then determines the next step.

The Controller

The Controller is the linchpin of the MVC pattern. Every user action carried out in the View is submitted through the Controller. The Controller then performs the next action, based on the content of the request from the browser.

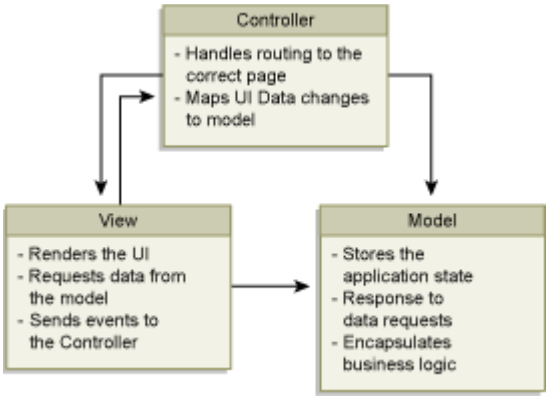
The Controller can be driven in several different ways. For example, you can use URL arguments to route the requests to the correct code. The MVC pattern itself determines the function of the Controller, not how it should work.

Benefits

The MVC architecture provides a clear and modular view of the application and its design. By separating the different components and roles of the application logic, it enables developers to design applications as a series of simple and different components: the Model, the View, and the Controller. This pattern should help to create applications that are easier to maintain and evolve. For example, once you create one view, you can easily create another view using the same business logic. Because of the ease and reusability, the MVC pattern is the most widely used pattern in the context of Web-based application development.

Figure 19–7 shows how the MVC pattern applies to a conventional thin-client Web application:

Figure 19–7 MVC Pattern



19.4.1.2 Apache Struts Overview

The Apache Struts framework (`http://struts.apache.org`) is one of the most popular frameworks for building Web applications, and provides an architecture based on the JSP Model 2 approach of the MVC design paradigm. In the Model 2 approach, end user requests are managed by a servlet that controls the flow, and uses components such as JavaBeans or EJBs to access and manipulate the data. It then uses JSPs to render the application content in a Web browser. This model differs from JSP Model 1, where the JSPs managed the browser request and data access.

The Struts framework provides its own HTTP Servlet as a controller component. The Struts framework is driven by an XML configuration file that contains the page flow of the application. Struts does not provide the Model, but enables developers to integrate it to any data access mechanism, for example EJBs, TopLink, or JDBC. The most common technology for writing View components is JSP and Struts provides various

tag libraries to help in writing these, although some of these tags have now been superseded by the Java Standard Tag Library (JSTL), which may also be used.

Note: For more information about JSTL and JSF, see the FAQ on the Apache Software Foundation Web site (`http://struts.apache.org/kickstart.html`).

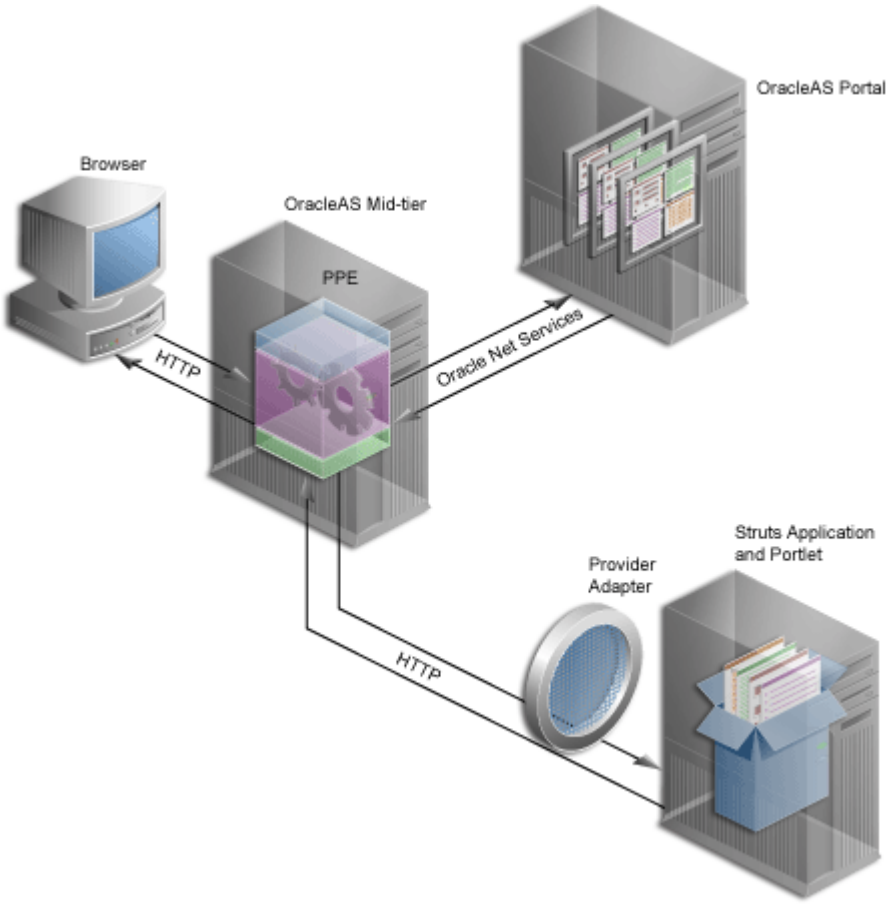
19.4.1.3 OracleAS PDK Integration with Struts

The OracleAS PDK contains numerous examples and documents regarding the usage of the OracleAS Portal APIs, such as personalization and caching. The integration of the application flow and business logic is not part of the portlet APIs. By using the Struts framework, however, you can leverage the MVC architecture to create and publish applications within your enterprise portal.

Oracle Struts Portlet

To create a portlet using the Struts framework, or to generate a portlet from an existing Struts application, you must deploy all the components in the J2EE container. In the context of OracleAS Portal, the Struts application is called by the PPE, and not by the browser as compared to a standalone Struts application. When a portlet show call is made, the page engine sends a request to the Struts portlet renderer, which then forwards the request to the Apache Struts Controller servlet, as shown in Figure 19–8.

Figure 19–8 Integrating Struts Applications with OracleAS Portal



The following code shows a portion of the producer definition file (`provider.xml`):

```
...
<renderContainer>true</renderContainer>
    <renderCustomize>true</renderCustomize>
    <autoRedirect>true</autoRedirect>
    <contentType>text/html</contentType>
    <showPage class="oracle.portal.provider.v2.render.http.StrutsRenderer">
        <defaultAction>showCustomer.do</defaultAction>
    </showPage>
</renderer>
...
```

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

The `showPage` tag defines the business logic that will be executed in the portlet mode of the portlet. The `showPage` of the Struts portlet contains two important components, which are as follows:

1. The renderer class (`oracle.portal.provider.v2.render.http.StrutsRenderer`), which receives the portlet request from the PPE and acts as a proxy to forward the request to the Struts Action Servlet.
2. The `defaultAction` tag, which defines the Struts action that will be used by default when the portlet is called for the first time.

The PDK-Java enables you to easily develop a view (Portal View) of your Struts application. This view enforces a consistent look and feel of your Struts portlet using portal styles, and enables the end user to use the application within the portal.

To create a Struts portlet, you must use the OracleAS Portal JSP tags, which are extensions of the default Struts JSP tags. This development process is similar to that of creating a standalone Struts application. To learn how to build a Struts portlet, see Section 19.4.2.1, "Creating a Struts Portlet". Also, as the portlet and struts application must also be in the same Servlet Context, you must create a single Web application that contains both elements. To learn how to easily create this Web application in Oracle JDeveloper, see the next section, Section 19.4.2.1, "Creating a Struts Portlet".

19.4.1.4 Summary

Apache Struts has become the default standard for developing MVC-based J2EE applications, because it offers a clean and simple implementation of the MVC design paradigm. This framework enables you, as the portlet developer, to separate the different components of an application, and to leverage the Struts controller to easily publish an existing Struts application to OracleAS Portal without completely changing the existing business logic.

Note: For more information about the Oracle Application Server Portal Developer Kit, see Portal Center (http://www.oracle.com/technology/products/ias/portal/pdk.html)

19.4.2 Creating a Struts Portlet

OracleAS PDK contains new extensions to integrate Apache Struts applications. This section explains how to build a portlet from an existing struts application. You can also follow these steps to create a portlet that uses the Model View Controller paradigm. To learn more about the Apache Struts framework, see Section 19.4.1, "The Apache Struts Framework". The PDK-Java extensions described in this section rely on Apache Struts 1.1.

This section contains the following steps:

- Section 19.4.2.1, "Creating a Struts Portlet"
- Section 19.4.2.2, "Registering the Producer"
- Section 19.4.2.3, "Summary"

19.4.2.1 Creating a Struts Portlet

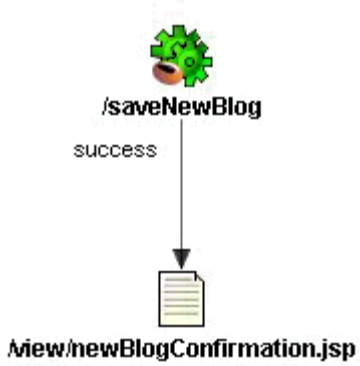
To publish a part of an existing Struts application as portlet, Oracle recommends that you first create a new view to serve as the Portal View of your application. This view uses existing objects (`Actions`, `ActionForm`, and so on) with a new mapping and new JSPs.

Note: Although Oracle recommends that you create a Portal View of your application, you could alternatively replace your application's struts tags with PDK-Java struts tags. This approach enables your application to run both as a standalone struts application and a portlet.

In this example, you will create a portlet that enables you to add a new entry to a Web Logger (Blog). Figure 19–9 and Figure 19–10 show how you submit a blog and save a blog entry.

Figure 19–9 Submitting a Blog

Figure 19–10 Saving a Blog Entry



`prepareNewBlog` is a simple empty action that redirects the request to the `enterNewBlog.jsp` page. This page shows a form for submitting a new blog.

The corresponding entry in the `struts-config.xml` is:

```
<action path="/prepareNewBlog" scope="request"
type="view.PrepareNewBlogAction" >
    <forward name="success" path="/view/enterNewBlog.jsp"/>
</action>
<action path="/saveNewBlog" name="blogForm" scope="request"
    type="view.SaveNewBlogAction" input"/view/enterNewBlog.jsp"  >
    <forward name="success" path="/view/newBlogConfirmation.jsp"/>
</action>
```

19.4.2.1.1 Create a New Flow and View to Host the Portlet Actions To create a new view, first create a new set of `ActionMappings` (page flow) that will redirect the various actions and requests to Portal-specific JSPs.

```
<action path="/portal/prepareNewBlog" scope="request"
    type="view.PrepareNewBlogAction" >
    <forward name="success" path="/view/portal/enterNewBlog.jsp"/>
</action>
<action path="/portal/saveNewBlog" name="blogForm" scope="request"
type="view.SaveNewBlogAction" input="/view/enterNewBlog.jsp"  >
    <forward name="success" path="/view/portal/newBlogConfirmation.jsp"/>
</action>
```

As you can see, only the path attributes are modified. The `FormBean` Action responsible for the application business logic remains unchanged.

19.4.2.1.2 Creating the New JSPs As specified in the previous step, the actions forward the request to new JSPs, which are responsible for rendering the portlet content. Your new portlet view JSPs can share the HTML with the standalone view, but be sure that the portlet meets the following criteria:

- Uses Portal styles that enforce a consistent look and feel with the rest of the portal page.
- Contains HTML code that is enabled in HTML table cells (that is, no `<html>`, `<body>`, and `<frame>` tags).
- Renders portal-aware links and forms. This is necessary to ensure that your Struts portlet renders its content inline, thus keeping your users within the context of the portal page by rendering the requested content within the same portlet container.

To achieve inline rendering in your Struts portlet, you must use OracleAS PDK tags:

```
<pdk-struts-html:form action="/portal/saveNewBlog.do">
...
...
</pdk-struts-html:form>
```

During the rendering of the portlet, one of the JSP tags (for example, the `pdk-struts-html:form` tag), submits the form to the Parallel Page Engine (PPE), which then sends the parameters to the Struts portlet. The Struts controller executes the logic behind these actions and returns the next JSP to the portlet within the portal page.

The PDK contains all the Struts tags, and extends all the tags that are related to URLs. The following is a list of the PDK extended tags:

- `form`: creates an HTML form and embeds the portal page context in the form to ensure inline rendering
- `text`: renders fields on the form.
- `link` and `rewrite`: create a link to the portal page, and are required for inline rendering
- `img`: creates an absolute link that points to the producer. If you want to use this tag in the context of Internet Web sites that have firewalls, then you must make sure the producer is directly accessible from the Internet. If it is not possible, then you can deploy the images to the OracleAS Portal middle tier and use the Apache Struts image link to generate a relative link (relative to the portal, not to the application).

> **Note:** You can register the OracleAS PDK with Oracle JDeveloper so that you can drop the tags from the Oracle JDeveloper Components Palette. For more information, see the *Registering a Custom Tag Library in JDeveloper* section in the Oracle JDeveloper online Help.

19.4.2.1.3 Creating a Portlet You can create your Struts portlet either manually or by using the Java Portlet Wizard. Although the wizard does not explicitly offer Struts support, you can use the wizard to build your Struts portlet.

To create a **portlet**, perform the following steps:

1. In Oracle JDeveloper, open the Java Portlet Wizard to create an Oracle PDK Java Portlet.

   > **Note:** The Java Portlet and Oracle PDK Java Portlet options are used to create JPS-compliant portlets and PDK-Java portlets respectively. Clicking **Java Portlet** or **Oracle PDK Java Portlet** opens the Java Portlet Wizard. For more information about opening the wizard, see Section 18.7, "Building PDK-Java Portlets with Oracle JDeveloper".

2. For the **Implementation Style** of the show page, select **Java Class**.
3. For the **Package Name**, enter `oracle.portal.provider.v2.render.http`
4. For the **Class Name**, enter `StrutsRenderer`. This generates the skeleton of the portlet renderer class, `StrutsRenderer`.

5. As the `StrutsRenderer` is part of the PDK, you do not need this generated file. So, when you finish the wizard, you must delete the file generated by the wizard. To do so, click the file in the System Navigator window, then from the **File** menu, select **Erase from Disk** in Oracle JDeveloper.
6. Edit the `provider.xml` and change the following properties:

   At the producer level, perform the following:

   - If you want users to always return to the same portlet state as when they left the portal page, then you can configure the struts renderer to save the struts action in the struts context:

     ```
     <actionInSession>true</actionInSession>
     ```

     If you prefer that users always start from the beginning of the portlet when they return from outside the portal page, then you should not save the struts action:

     ```
     <actionInSession>false</actionInSession>
     ```

     Note that this setting is the default behavior.

   - If the Struts application uses sessions (for example, the form sysnchronizer token mechanism is used or `<actionInSession>` is set to true), then enable session handling:

     ```
     <session>true</session>
     ```

   At the portlet level, perform the following:

   - Specify the first action to raise when the portlet is called. Use the following code:

     ```
     <showPage class="oracle.portal.provider.v2.render.http.StrutsRenderer">
     <defaultAction>/portal/prepareNewBlog.do</defaultAction>
     </showPage>
     ```

For more information about the syntax of `provider.xml`, see the producer JavaDoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

19.4.2.1.4 Extending the Portlet to Add Business Logic In your application, you should add code specific to your environment, such as the user's information, personalization, and localization. To do so, you can create a new `Action` class that is only called in context, and handles all business logic.

19.4.2.2 Registering the Producer

Now that your portlet is ready to be used by consumers, you must make it accessible by registering it. For information about how to register your PDK-Java portlet, see Section 18.10, "Registering and Viewing Your Portlet".

19.4.2.3 Summary

Oracle Application Server enables you to easily create Struts portlets using Oracle JDeveloper and publish existing Struts application portlets. For more information about using the Oracle JDeveloper Java Portlet wizards, see Chapter 18, "Creating Java Portlets".

19.4.3 Creating an Oracle Application Development Framework Portlet

Similarly to Struts, Oracle ADF relies on the MVC design pattern. Oracle ADF applications leveraging the Struts controller can be turned into portlets and deployed the same way as Struts applications. See Section 19.4.2, "Creating a Struts Portlet".

> **Note:** After creating the Oracle ADF portlet, you may find that the JSP page does not display correctly. This is because the Parallel Page Engine request is sent to a producer through a SOAP request (`oracle.webdb.provider.v2.adapter.SOAPServlet`), which implies that the portal does not serve the page as a standard `.JSP` page. To resolve this, create the `ADFBindingFilter` filter.

To create the `ADFBindingFilter` filter and filter mappings, include the following in your `web.xml` file:

```
<filter>
   <filter-name>ADFBindingFilter</filter-name>
   <filter-class>oracle.adf.model.servlet.ADFBindingFilter</filter-class>
   <init-param>
     <param-name>encoding</param-name>
     <param-value>windows-1252</param-value>
   </init-param>
 </filter>
 <filter-mapping>
   <filter-name>ADFBindingFilter</filter-name>
   <url-pattern>*.jsp</url-pattern>
 </filter-mapping>
 <filter-mapping>
   <filter-name>ADFBindingFilter</filter-name>
   <url-pattern>*.jspx</url-pattern>
 </filter-mapping>
 <filter-mapping>
   <filter-name>ADFBindingFilter</filter-name>
   <url-pattern>/*</url-pattern>
 </filter-mapping>
 <filter-mapping>
   <filter-name>ADFBindingFilter</filter-name>
   <url-pattern>*</url-pattern>
 </filter-mapping>
 <filter-mapping>
   <filter-name>ADFBindingFilter</filter-name>
   <servlet-name>action</servlet-name>
 </filter-mapping>
 <filter-mapping>
   <filter-name>ADFBindingFilter</filter-name>
   <servlet-name>jsp</servlet-name>
 </filter-mapping>
```

19.5 Building Portlets from Oracle ADF Faces Applications (JSF Portlet Bridge)

In some cases, you may want to take an Oracle ADF Faces application that you have built and expose it as a portlet. Such a portlet, sometimes called a JSF portlet bridge, executes Oracle ADF Faces pages as JPS (JSR 168) portlets.

Note: Unless otherwise noted, the material in this section applies equally to the JSF reference implementation.

This section provides the following information about exposing Oracle ADF Faces applications as JPS portlets:

- Creating a JSF Portlet
- Guidelines for Oracle ADF Faces Applications

19.5.1 Creating a JSF Portlet

To create a JSF portlet from an existing application, perform the following steps:

1. Open your JSF application workspace. The first thing you must do to portletize your application is add the necessary JSF portlet bridge library to your project.
2. Right-click the project containing the application of which you plan to make a portlet. Choose **Project Properties** from the context menu.
3. From the left pane, click **Libraries**.
4. Click **Add Library**.
5. Select **Portlet Faces Bridge**.
6. Click **OK**.
7. Click **OK**. Now you need to create a Portlet Deployment Descriptor that enables you to deploy your application as a portlet producer.
8. Right-click your project and choose **New** from the context menu.
9. Choose **All Technologies** from the **Filter By** list.
10. Expand the **General** category from the pane on the left.
11. Click **Deployment Descriptors**.
12. Choose **portlet.xml (Portlet Deployment Descriptor)** from the **Items** list on the right.
13. Click **OK**.
14. You should now see `portlet.xml` under **Web Content**, **WEB-INF** in the Applications Navigator. Open `portlet.xml` and go to the Source tab.
15. Add or modify the necessary `init-params` in `portlet.xml` for your particular application. For example, the `portlet.xml` for an Oracle ADF application that uses ADF binding might look something like the one in Example 19–13. For non-ADF binding, it might look something like the one in Example 19–14.

    In particular, note the `init-params` in bold. The parameters are as follows:

    - The `DefaultPage.view` parameter specifies the default page of the application to use as the View mode for the portlet. Its location is relative to the `web-app-context-root` and always start with a `/`.
    - The `BridgeLifecycleListeners` parameter specifies the filter classes to use. You must ensure that your `portlet.xml` file is configured to match the filters specified in your `web.xml` file. If you do not have the proper classes defined in `portlet.xml`, then you will receive a missing class error. You specify the classes in the `BridgeLifecycleListeners init-param` in your `portlet.xml` file:

```
<init-param>
  <name>BridgeLifecycleListeners</name>
  <value>listener_class_n[,listener_class_n+1,...]</value>
</init-param>
```

For example, suppose that your application includes `AdfFacesFilter` filter in its `web.xml`:

```
<filter>
  <filter-name>adfFaces</filter-name>
  <filter-class>oracle.adf.view.faces.webapp.AdfFacesFilter</filter-class>
</filter>
```

You would then need to include the following class in your `portlet.xml`:

```
<init-param>
  <name>BridgeLifecycleListeners</name>

<value>oracle.portlet.server.bridges.jsf.adf.ADFFacesBridgeLifecycleListene
r
    </value>
</init-param>
```

Similarly, suppose that `ADFBindingFilter` filter is defined in `web.xml` as follows:

```
<filter>
  <filter-name>adfBindings</filter-name>
  <filter-class>oracle.adf.model.servlet.ADFBindingFilter</filter-class>
</filter>
```

You would then need to include the following in your `portlet.xml`:

```
<init-param>
  <name>BridgeLifecycleListeners</name>
<value>oracle.portlet.server.bridges.jsf.adf.ADFFacesBridgeLifecycleListene
r,
oracle.portlet.server.bridges.jsf.adf.BindingFacesBridgeLifecycleListener
</value>
</init-param>
```

Example 19–13 Sample portlet.xml for JSF Portlet (ADF Binding)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
   version="1.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
       http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
   <portlet>
     <description>ADF Faces Demo Portlet</description>
     <portlet-name>ADFFacesDemo</portlet-name>
     <display-name>ADF Faces Demo portlet</display-name>
     <portlet-class>oracle.portlet.server.bridges.jsf.FacesPortlet
     </portlet-class>
     <init-param>
      <name>DefaultPage.view</name>
      <value>/index.jspx</value>
     </init-param>
     <init-param>
      <name>BridgeLifecycleListeners</name>
      <value>
       oracle.portlet.server.bridges.jsf.adf.ADFFacesBridgeLifecycleListener,
```

```
      oracle.portlet.server.bridges.jsf.adf.BindingFacesBridgeLifecycleListener
     </value>
    </init-param>
   <supports>
    <mime-type>text/html</mime-type>
    <portlet-mode>VIEW</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
     <title>ADF Faces Demo Portlet</title>
     <short-title>ADFFacesDemo</short-title>
    </portlet-info>
  </portlet>
</portlet-app>
```

Example 19–14 Sample portlet.xml for JSF Portlet (Non-ADF Binding)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
   version="1.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
       http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
   <portlet>
     <description>ADF Faces Demo Portlet</description>
     <portlet-name>ADFFacesDemo</portlet-name>
     <display-name>ADF Faces Demo portlet</display-name>
     <portlet-class>oracle.portlet.server.bridges.jsf.FacesPortlet
      </portlet-class>
    <init-param>
     <name>DefaultPage.view</name>
     <value>/index.jspx</value>
    </init-param>
    <init-param>
     <name>BridgeLifecycleListeners</name>
      <value>
       oracle.portlet.server.bridges.jsf.adf.ADFFacesBridgeLifecycleListener
      </value>
    </init-param>
    <supports>
     <mime-type>text/html</mime-type>
     <portlet-mode>VIEW</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
     <title>ADF Faces Demo Portlet</title>
     <short-title>ADFFacesDemo</short-title>
    </portlet-info>
  </portlet>
</portlet-app>
```

- Modify the `web.xml` for your particular application to use client-side state saving method by defining the following in your `web.xml`:

  ```
  <context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>server</param-value>
  </context-param>
  ```

16. You can now deploy your application as you would any Web application. See Section 18.9, "Deploying Your Portlet to an Application Server".

17. Once you have successfully deployed your application, you can register it as a portlet producer with any other application. See Section 4.3.1.1, "Registering WSRP Portlet Producers". Note that your application now runs in dual mode. You can continue to access it as regular Web application or consume it as a portlet producer.
18. Now that your producer is deployed and registered, you can consume your JSF application's portlet as you would any other portlet. See Section 4.3.2, "Adding Portlets to a Page".

19.5.1.1 Passing Parameters

You can enable communication between a JSF Portlet Bridge and the underlying JSF application by configuring the portlet bridge to pass on WSRP 2.0 navigational parameters as request parameters to the underlying application. For this, you must add such parameters as navigational parameters in oracle-portlet.xml as shown in the following example:

```
<navigation-parameters>
    <name>Parameter_01</name>
    <type>xsi:string</type>
    <label xml:lang="en">Parameter 1</label>
    <hint xml:lang="en">First parameter.</hint>
</navigation-parameters>
```

Example 19–15, Example 19–16, and Example 19–17 show the sample code used to enable parameter passing between a JSF Portlet Bridge and the consuming application. Items of interest are shown in bold.

Example 19–15 Sample Code of JSF Page (JSFParameterForm.zip:JSFParamDisplay.jspx)

```
  <f:view>
    <afh:html>
      <afh:head title="ParamForm">
        <meta http-equiv="Content-Type"
              content="text/html; charset=windows-1252"/>
      </afh:head>
      <afh:body>
       <h:form>
        <af:outputLabel value="Parameter 1:"/><af:outputText value="#{param.ora
                                              _wsrp_navigparam_Parameter1}"/>
       </h:form>
      </afh:body>
    </afh:html>
  </f:view>
```

Example 19–16 Sample Code of oracle-portlet.xml (JSFParameterForm.zip:oracle-portlet.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <portlet-app-extension
       xmlns="http://xmlns.oracle.com/portlet/oracle-portlet-app" version="1.0"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
  <portlet-extension>
    <portlet-name>JSFParameterDisplay</portlet-name>
      <navigation-parameters>
        <name>ora_wsrp_navigparam_Parameter1</name>
```

```
        <type>xsi:string</type>
        <label xml:lang="en">First parameter</label>
        <hint xml:lang="en">First parameter set by portlet</hint>
      </navigation-parameters>
      <portlet-id>1</portlet-id>
  </portlet-extension>
</portlet-app-extension>
```

Example 19–17 Sample Code of Page Definition File of The Consumer Page (InterPortletComm.zip:PortletCommPageDef.xml)

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="10.1.3.40.66" id="PortletCommPageDef"
                                    Package="view.pageDefs">
 <parameters/>
   <executables>
     <variableIterator id="variables">
        <variable Name="ParameterFormPortlet1_1_ora_wsrp_navigparam_Parameter1"
                                                        Type="java.lang.Object"/>
     </variableIterator>
     <portlet id="JSFParameterDisplayPortlet1_1"
        portletInstance="/oracle/adf/portlet/jsfParam_1178341548071/ap/E0default
                                      _5aa185c9_0112_1000_8005_8d90409dd9cb"
        class="oracle.adf.model.portlet.binding.PortletBinding"
        retainPortletHeader="false"
        xmlns="http://xmlns.oracle.com/portlet/bindings">
   <parameters>
        <parameter name="ora_wsrp_navigparam_Parameter1"
                 pageVariable="ParameterFormPortlet1_1_ora_wsrp_navigparam
                                                            _Parameter1"/>
   </parameters>
  </portlet>
 </executables>
</pageDefinition>
```

For more information about passing parameters, see Section 4.5.1, "Linking Portlets to Pages".

19.5.2 Guidelines for Oracle ADF Faces Applications

To publish an Oracle ADF Faces page as a portlet, you must first code your Oracle ADF Faces pages such that they emit markup that conforms with JSR 168 portlet markup fragment rules. Fortunately, most of the markup in an Oracle ADF Faces page comes from the Oracle ADF Faces components, most of which naturally render markup in a style that is compatible with portlets.

For those components that might cause problems in a portlet environment, the application developer must take special care. Some components generate markup that conflicts with the portlet environment and hence restricts their use. Other components may enable program control (inputs) that enable developers to introduce values that conflict with the portlet environment. In this latter case, you as the developer must be aware of the potential to publish the page as a portlet and therefore properly encode a value.

The guidelines that follow lay out the issues of which you should be aware as you code Oracle ADF pages that you may later choose to publish as portlets.

19.5.2.1 General Guidelines

The guidelines that follow lay out the issues of which you should be aware as you code Oracle ADF pages that you may later choose to publish as portlets.

- When building a JSF portlet bridge in Oracle JDeveloper, you should build your application with one of the Web Application templates that employs JSF.
- You cannot portletize applications that already contain portlets.
- Using the client to manage and save the Faces view state is not supported. Therefore, you must configure your application to have this state managed by the server. Because the Oracle ADF Faces default is client-managed, you must explicitly configure the application to use server-side state management by defining the following in your `web.xml`:

```
<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>server</param-value>
</context-param>
```

- Deep links aren't directly supported. A by product of the JSR 168 portlet container implementation on WSRP is that session cookie management is proxied by the consuming application rather than the client. Portlets that deep link to their full service application usually rely on shared session state to enable the transition from the current portlet context. As most applications rely on maintaining session context with a cookie, the current architecture prevents such state sharing. A deep link from the client directly to the producer server to invoke the application does not establish a session cookie between the consumer and the producer, hence a second session will be established. Applications wanting to share such state need to implement their own schemes for transferring the data between the two contexts. A common implementation is to write this state to a reachable location and pass a reference to this state in the deep link.
- Java EE login is not supported. Java EE applications can be configured with a number of authentication techniques, such as Basic and Digest. As portlets are fragments incorporated into a consumer's page, it is generally expected that direct authentication occurs between the client and the consumer, not the client and the portlet producer. As a result, these authentication techniques are not supported in JSR 168. In the JSR 168 case, Java EE authentication occurs through WS-Security mechanisms that enable the Web service consumer to be authenticated and verified by the producer, and propagate the user identity for user authentication and authorization. Published Oracle ADF artifacts should not contain login links that trigger Java EE authentication.
- If the consumer of the JSF portlet is OracleAS Portal, then the following may occur:
  - A bug (5526946) causes the portlet not to render on the first request after you bounce OC4J or if the portlet has been idle for a while. The message says:

    ```
    Error: The portlet could not be contacted.
    ```

    You can resolve this message by refreshing the page.
  - Navigation may fail when you have multiple JSF portlet bridges on a single page. This failure usually occurs because the identifiers of HTML elements and the names of JavaScript functions are conflicting. To work around this problem, make sure your applications use unique identifiers and JavaScript variables.

19.5.2.2 Portlet Guidelines

The portlet guidelines are as follows

- For resources and links, you must specify the location relative to the `web-app-context-root`. Otherwise, your images and other resources will not be found by the portlet. Do not use relative (../) path notation. Portlets in Oracle WebCenter Framework run remotely and are accessed using a SOAP protocol (WSRP). The latter means that the regular Web application concept of request path is meaningless in a JSR 168 container. The JSR 168 specification reflects this by mandating that all resource URLs either be absolute or context path relative.
- Do not redirect or forward a request within your JSP. JSR 168 only supports `requestDispatcher.include()`. The use of `httpServletResponse.sendRedirect()` or `requestDispatcher.forward()` results in exceptions and errors. To work properly in a portlet environment, you must implement navigation with navigation rules in `faces-config.xml`.
- Minimize the memory size and use of request scoped data. The JSF portlet bridge internally promotes request scoped data to session scope because the portlet action/render phases are permitted to span two requests. Furthermore, unlike a JSF Web application, a portlet may be rendered many times after an action has been submitted. In each of these cases, the duration of the request data must be longer then a single http request. To minimize overall memory consumption in the application, you should only store the minimal amount of data in the request scope to be able to render successfully.

19.5.2.3 Oracle ADF Faces Guidelines

The Oracle ADF faces guidelines are as follows:

- Oracle ADF Faces client side implementation is not supported. One weakness of JSR 168 and WSRP 1.0 is that they do not account for portlets using client-side programming techniques, such as Ajax or hidden frames, for partial page rendering. As a result, all Oracle ADF Faces facilities that rely on client-side programming techniques do not work. These facilities include the following:
  - Oracle ADF Faces partial page refresh (PPR) facility
  - The client-rendered aspect of Oracle ADF Faces rich client components
  - Oracle ADF popups, `<af:popup>`, which fail to work because of their reliance on the PPR facility
  - The Oracle ADF Faces `SelectText`, `SelectDate`, and `SelectColor` components, which use visual assist popups

    To avoid breaking existing applications, the popup is disabled when used in Oracle ADF Faces pages exposed as a portlet. Thus, functionally speaking, these components are equivalent to `InputText`.
- File upload is not supported. To support file upload, Oracle ADF Faces requires facilities that do not exist in JSR 168. Hence, a file upload request does not work.
- The `objectMedia` component is not supported in portlets.

19.5.2.4 Oracle ADF Guidelines

The Oracle ADF guidelines are as follows:

- Portlet customization using Oracle Metadata Services is not supported. The JSR 168 preference model is incompatible with an environment that manages its own

customizations in its own repository, such as Oracle ADF using Oracle Metadata Services.

- When a WebCenter application consumes a JSF portlet bridge, the portlet can only use JSR 168 portlet styles (instead of the standard Oracle ADF styles). This may result in lost of fidelity.
- Oracle ADF components/code that handle `prepareModel` must be idempotent. Any code executing during the `prepareModel` phase must be rerunnable without efect to the underlying data/business logic. When executing in the portlet environment, `prepareModel` is called twice during the complete JSF life cycle as opposed to being called once when executing within a regular Web application.

  The reason for this difference is that the JSF portlet bridge executes JSF in two requests not one. It is implemented as if every JSF request redirected before the render phase. The consequence of this approach is that the JSF `restoreView` phase is called both when the request is first submitted and then again when request to render is received.
- Do not access or reference request parameters from model code except in page parameters. Besides being a cleaner MVC2 implementation, following this guideline avoids problems when such artifacts are published as portlets. Where regular JSF artifacts run their entire lifecyle in a single request, the JSF portlet bridge executes these artifacts in two requests, as if every JSF request redirected before the render phase.

  This two phase model enables the clearing of submitted parameters before rendering in a manner that enables such clearing to be communicated all the way back to the client, which makes this request something you could bookmark. As a result, request parameters do not exist during the render phase. As described previously, prepareModel is invoked again in this rendering phase. Any references to request parameters in this phase handler will therefore fail. You should avoid code like either of the following fragments:

```
<invokeAction id="doExecuteWithParams"
              Binds="ExecuteWithParams"
              Refresh="prepareModel"
              RefreshCondition="${param.id != null}"
/>

<invokeAction id="doExecuteWithParams"
              Binds="ExecuteWithParams"
              Refresh="renderModel"
              RefreshCondition="${param.id != null}"
/>
```

- Do not reference page parameters in the `prepareModel` phase. This issue relates to the same problem just described for request parameters. Generally, page parameters depend on request parameters and are evaluated during the `restoreView` phase. As this phase is called a second time during portlet rendering and request parameters are not available, such use will fail. Instead, move any dependency on a page parameter value into the model before JSF transitions from its execute phases to its render phase.

Part V

Appendixes

Part V contains the following appendixes:

- Appendix A, "Reuse of OracleAS Portal Components"
- Appendix B, "Additional Portlet Configuration"
- Appendix C, "Files for WebCenter Applications"
- Appendix D, "Manually Packaging and Deploying PDK Portlet Producers"
- Appendix E, "Administering Oracle WebCenter Wiki"
- Appendix F, "Node Type Definitions for Oracle WebCenter Adapters"
- Appendix G, "Troubleshooting WebCenter Applications"

A

Reuse of OracleAS Portal Components

If you have used Oracle Application Server Portal (OracleAS Portal) in the past, then this appendix can help you to understand how you might make use of the components of your portal in a WebCenter application. While no direct migration is available from OracleAS Portal to Oracle WebCenter Suite, many of the components of your portal can be brought forward and reused in a WebCenter application built with Oracle WebCenter Framework.

A.1 Reusing Your OracleAS Portal Components in WebCenter Suite

The two most basic building blocks of OracleAS Portal are as follows:

- Items are individual pieces of content (text, hyperlink, image, and so on) that reside on a page in an item region. Users with an appropriate privilege level can add items to a page. Item content and metadata are stored in the OracleAS Portal schema of the Oracle Application Server Metadata Repository. Items are rendered on the page according to the layout, style, and attribute display defined for the item region.

  Because items are stored in a content repository, you can retrieve them with Java Content Repository (JCR) data controls. A default adapter for the OracleAS Portal content repository is provided with Oracle WebCenter Framework. See Section A.1.2, "Reusing Items" for more information.

- Portlets are the fundamental building blocks of a portal page. OracleAS Portal provides several ways to build portlets programmatically and to integrate any kind of Web content. Portlets may be implemented using various technologies, such as Java, JSPs, Java servlets, PL/SQL, Perl, ASP, and so on. The Portal Developer Kit (PDK) covers the standards-based (JSR 168) portlet development options that OracleAS Portal provides. You can also use the Portlet Builder to create your own portlets.

  In Oracle WebCenter Framework, you can make portlets available by registering their producers (also known as providers) with the application. See Section A.1.1, "Reusing Portlets" for more information.

You can reuse each of these building blocks in your WebCenter application.

> **Note:** In Oracle WebCenter Framework, there is currently no page template feature. You can, however, reuse a JSP by copying and pasting its content in a newly created page.

A.1.1 Reusing Portlets

To reuse portlets in your WebCenter application, all you need to do is register the producers (sometimes referred to as providers in OracleAS Portal) as you would any other producer. See Section 4.3, "Consuming Portlets" for information about consuming portlets in a WebCenter application. The sections that follow describe some areas where behavior of portlets differs from OracleAS Portal.

- iframes
- Events
- Mobile Portlets
- Portlet Chrome
- Personalizations and Customizations
- OracleAS Portal System Resources
- Partner and External Applications
- Federated Portal Adapter
- PDK-Java Producers from Earlier Oracle Application Server Versions

A.1.1.1 iframes

By default, a portlet may be placed inside of an iframe. If your portlet is placed in an iframe, then it effects the ability of the portlet to locate itself on the page. Inside of an iframe, your `window.location` setting may not give the expected results. For more information about when your portlet may be placed inside of an iframe, see Section 4.3.3.6, "iframes and form Tags"

A.1.1.2 Events

Events are not supported in Oracle WebCenter Framework. If you have PDK-Java portlets that use events and you deploy them in a Oracle WebCenter Framework environment, then the events are ignored.

A.1.1.3 Mobile Portlets

Mobile portlets are not supported in Oracle WebCenter Framework. If you have PDK-Java mobile portlets, then they will not work in a Oracle WebCenter Framework environment.

A.1.1.4 Portlet Chrome

In Oracle WebCenter Framework, the portlet does not provide the chrome. The consuming application provides the chrome. This behavior conforms to the WSRP convention, but it is a change from OracleAS Portal, where PDK-Java provided the chrome for portlets. Hence, the PDK-Java portlet header is filtered out in the Oracle WebCenter Framework environment.

As part of this filtering process, Oracle WebCenter Framework parses the title area of PDK-Java portlets to preserve the title and add the necessary portlet mode links. The filtering algorithm is as follows:

- If the portlet has multiple tables, then the title area is considered to be the first table. If the portlet has only one table, then the title area is considered to be the first row in that table.
- The first text in the title area is considered to be the title. If the portlet has no tables or a discernible title, then the title is taken from the `adfp:portlet` component.

- Any link with that contains the fragment `_mode= parameter` is considered to be a portlet mode link. These links are moved into the portlet menu by Oracle WebCenter Framework.

If you wish to bypass the rendition of the header section of your portlet chrome and reuse its original, PDK-Java header, then you can set the `displayHeader` attribute of the `adfp:portlet` tag to `false` and `retainPortletHeader` as `true` in the associated portlet bindings of the page definition Extensible Markup Language (XML). Setting `retainPortletHeader=true` in the page definition portlet binding retains the portlet header in the portlet response. Setting `displayHeader=false` in the `adf:portlet` tag suppresses the application's header for the portlet chrome, and hides the icons and links normally displayed in the header.

A.1.1.5 Personalizations and Customizations

Portlet customizations and personalizations that you or your users applied to your portlets in OracleAS Portal will not be carried over to your WebCenter application. In a WebCenter application, your OracleAS Portal portlets are treated as new instances. Hence, previous personalizations and customizations are not available.

A.1.1.6 OracleAS Portal System Resources

Portlets built with a target consumer of OracleAS Portal in mind tend to make invalid assumptions about the run time environment. For example, the portlet might assume the presence of OracleAS Portal resources, such as images or Javascript functions. In Oracle WebCenter Framework, these resources are not available. Hence, you need to bundle the resources the portlet needs with the producer to ensure that the portlet runs properly in the Oracle WebCenter Framework environment.

Another common issue with portlets designed for OracleAS Portal consumption is the portlet assuming the OracleAS Portal URL format and the presence of OracleAS Portal parameters. The page URL of an Oracle ADF application is different from that of OracleAS Portal. Therefore, portlet developers cannot make assumptions about the page URL and must prepare their PDK-Java portlets for execution in the Oracle WebCenter Framework environment by using the proper portlet Application Programming Interface (API)s.

A.1.1.7 Partner and External Applications

Partner applications are not supported. External applications are supported, but they send a null login URL. Any links wrapped by the login URL (links through Oracle Single Sign-On) will not work.

A.1.1.8 Federated Portal Adapter

The Federated Portal Adapter (FPA) enables you to make PL/SQL portlets, Portlet Builder portlets, and OracleAS Portal pages visible in your WebCenter applications. Note that for pages to be available through the FPA, they must have been exposed as portlets in OracleAS Portal. These FPA portlets are read only in a WebCenter application. Users can see their contents but not interact with them.

Implementing FPA for your portal exposes the OracleAS Portal portlets through the PDK-Java Web producer protocol making it possible to register them as PDK-Java producers. Registering the producer with a WebCenter application causes the portlets to appear in the Component Palette, and you can then drag and drop them onto pages as you would any other portlet. Without FPA, you cannot access these producers from a WebCenter application.

To register your FPA producer with Oracle WebCenter Framework, perform the following steps:

1. Implement FPA for your OracleAS Portal instance based on the information in *Oracle Application Server Portal Configuration Guide*.
2. For any pages in the OracleAS Portal instance that you want to expose through FPA, publish them as portlets.
3. For secured portlets, configure HMAC in the OracleAS Portal instance. This procedure is described in *Oracle Application Server Portal Configuration Guide*. Note that for WebCenter applications the sending portal is always:

   ```
   http://www.oracle.com/adapter/portal
   ```

4. In Oracle JDeveloper, register the producer as a PDK-Java producer using the FPA URL and Service ID.
5. The producer should now appear in the Component Palette and you can drag and drop its portlets onto a page.
6. Since portlets exposed through FPA may not be customized or personalized from within a WebCenter application, you should suppress the appearance of Customize and Personalize in the Actions menu. You can remove these choices from the Actions menu by setting the `isCustomizeModeAvailable` and `isPersonalizeModeAvailable` attributes of the `adfp:portlet` tag to `false`. For more information, see Section 4.3.3.2, "Actions Attributes of the adfp:portlet Tag".
7. Confirm that the users of the WebCenter application have appropriate privileges on the portlets and pages exposed through FPA, and that they map to the users of the OracleAS Portal instance. The user name for a person on the WebCenter application should map to the user name of the same person on OracleAS Portal. For example, JSMITH should represent the same person in both the WebCenter application and OracleAS Portal.
8. Now run the page.

A.1.1.9 PDK-Java Producers from Earlier Oracle Application Server Versions

If you have a producer that you designed and deployed on Oracle Application Server Release 2 (10.1.2) or earlier, then you can reuse it in either one of two ways:

- You can consume a portlet from a producer running on an Oracle Application Server Release 2 (10.1.2.0.2) middle tier. In this case, the producer's code is running on Release 2 (10.1.2.0.2) while the application consuming the producer is running on Release 3 (10.1.3.2).
- You can redeploy the producer application as an EAR file on WebCenter Suite and consume its portlets in other WebCenter applications. In this case, both the producer's code and the consuming application are running on Release 3 (10.1.3.2).

In either of these cases, you must take some additional steps to get the producer's portlets to operate correctly. Portlets built for OracleAS Portal expect certain boilerplate images to be present on the page assembly servlet and they will break if those images are not available. These images were included in the middle-tier servlet by default in Oracle Application Server Release 2 (10.1.2.0.2), but for Oracle WebCenter Framework you must manually ensure that the images can be found by portlets.

- Consuming a Portlet from OracleAS Portal

- Redeploying PDK-Java Producers from OracleAS Portal

A.1.1.9.1 Consuming a Portlet from OracleAS Portal In WebCenter Suite, to consume a portlet running on a page in an OracleAS Portal instance, you can do either of the following:

- Obtain the OracleAS Portal images zip file from the Oracle Technology Network and unzip it inside the ADFP servlet.
- Properly configure the resource servlet as follows:

1. Add an initialization parameter to the Web servlet as shown is Example A–1.

Example A–1 Web Servlet Initialization Parameters

```
<servlet>
 <servlet-name>SOAPServlet</servlet-name>
 <display-name>SOAPServlet</display-name>
 <description>Extended Portal SOAP Server</description>
 <servlet-class>oracle.webdb.provider.v2.adapter.SOAPServlet</servlet-class>
 ...
 <init-param>
   <param-name>resourceServletMapping</param-name>
   <param-value>/pdkresource</param-value>
 </init-param>
</servlet>
```

2. Add a servlet definition for the resource servlet as shown in Example A–2.

Example A–2 Servlet Definition for Resource Servlet

```
<servlet>
 <servlet-name>ResourceServlet</servlet-name>
  <display-name>ResourceServlet</display-name>
  <description>Image resource servlet</description>
  <servlet-class>oracle.webdb.provider.v2.adapter.ResourceServlet</servlet-class>
</servlet>
```

3. Add a servlet mapping for the resource servlet as shown in Example A–3.

Example A–3 Servlet Mapping for Resource Servlet

```
<servlet-mapping>
 <servlet-name>ResourceServlet</servlet-name>
 <url-pattern>/pdkresource/*</url-pattern>
</servlet-mapping>
```

4. Now you can register this PDK-Java producer as you would any other PDK-Java producer. See Section 4.3.1.2, "Registering PDK-Java Portlet Producers" for more information about registering PDK-Java producers.

A.1.1.9.2 Redeploying PDK-Java Producers from OracleAS Portal If you choose to take a PDK-Java producer from OracleAS Portal and redeploy it on WebCenter Suite, then you need to enable the resource servlet as follows:

1. Add an initialization parameter to the Web servlet as shown in Example A–4.

Example A–4 Web Servlet Initialization Parameter

```
<servlet>
 <servlet-name>SOAPServlet</servlet-name>
```

```
 <display-name>SOAPServlet</display-name>
 <description>Extended Portal SOAP Server</description>
 <servlet-class>oracle.webdb.provider.v2.adapter.SOAPServlet</servlet-class>
 ...
 <init-param>
   <param-name>resourceServletMapping</param-name>
   <param-value>/pdkresource</param-value>
 </init-param>
</servlet>
```

2. Add a servlet definition for the resource servlet as shown in Example A–5.

Example A–5 Servlet Definition for Resource Servlet

```
<servlet>
 <servlet-name>ResourceServlet</servlet-name>
  <display-name>ResourceServlet</display-name>
  <description>Image resource servlet</description>
  <servlet-class>oracle.webdb.provider.v2.adapter.ResourceServlet</servlet-class>
</servlet>
```

3. Add a servlet mapping for the resource servlet as Example A–6.

Example A–6 Servlet Mapping for Resource Servlet

```
<servlet-mapping>
 <servlet-name>ResourceServlet</servlet-name>
 <url-pattern>/pdkresource/*</url-pattern>
</servlet-mapping>
```

4. Now you can register this PDK-Java producer as you would any other PDK-Java producer. See Section 4.3.1.2, "Registering PDK-Java Portlet Producers" for more information about registering PDK-Java producers.

A.1.2 Reusing Items

Items from OracleAS Portal have no equivalent in Oracle WebCenter Framework. As such, creating, maintaining, and publishing items is not supported in Oracle WebCenter Framework. To replicate some of the behavior of items with Oracle WebCenter Framework, you can do either one of the following:

- Use JCR data controls to include content from OracleAS Portal. OracleAS Portal is one of the content repositories that Oracle WebCenter Framework supports out of the box in the Data Control Wizard in Oracle JDeveloper. For more information about integrating content from OracleAS Portal, see Chapter 5, "Integrating Content".
- Store item-like content in Oracle Content Database and publish it with standard Oracle ADF components such as `adf:tree`. For more information, see Chapter 5, "Integrating Content".

B

Additional Portlet Configuration

This appendix discusses configuration information for some of the portlet technologies available with Oracle WebCenter Suite. This chapter includes the following sections:

- Section B.1, "Java Portlet Configuration Tips"
- Section B.2, "OmniPortlet Configuration Tips"
- Section B.3, "Web Clipping Portlet Configuration Tips"
- Section B.4, "Portlet Preference Store Migration Utilities"

B.1 Java Portlet Configuration Tips

This section contains configuration information for Java portlets.

Disabling a WSRP Test Page

To disable your WSRP test page, perform the following steps:

1. In Oracle JDeveloper, go to the Application Navigator and expand the **Web Content** and **WEB-INF** folders.
2. Double-click the `web.xml` file to open it.
3. In the Source mode, look for the following element and delete it:

```
<servlet-mapping>
      <servlet-name>WSRPTestPage</servlet-name>
      <url-pattern>/info</url-pattern>
   </servlet-mapping>
```

> **Note:** If you do not want to delete the element as you may need it later, then put the element in comment tag, as shown in the following example:
>
> ```
> <servlet-mapping>
>       <!--Added by WSRP install tool-->
>       <servlet-name>WSRPTestPage</servlet-name>
>       <url-pattern>/info</url-pattern>
>    </servlet-mapping>
> ```

4. Save the `web.xml` file.
5. Run your test page in a browser, an error occurs.

B.2 OmniPortlet Configuration Tips

This section contains configuration information for OmniPortlet. To learn more about the OmniPortlet wizard, see Chapter 16, "Creating Portlets with OmniPortlet". This section contains configuration information for the following areas:

B.2.1 Configuring the OmniPortlet Producer to Access Data Outside a Firewall

If the OmniPortlet producer is inside your firewall, then you must configure OmniPortlet to access URLs of data (such as CSV, XML, or Web Services) located outside the firewall. You configure the proxy information in the `provider.xml` file. Table B–1 provides a list of parameters and their descriptions.

Table B–1 Provider.xml Tags

Parameter	Description
`httpProxyHost`	Enter the host name of a proxy server if one is required to make a URL connection from the OmniPortlet producer to its data sources.
`httpProxyPort`	Enter the port number for the HTTP Proxy Host.
`dontProxyFor`	Enter the name of any domain or hosts to which you can directly connect, bypassing a proxy server. Domain names are the part of a URL that contain the names of a business, or organization, or government agency, for example: `*.company.com, *.us.company.com` Hosts can be fully qualified host names or can be IP addresses.
`proxyUseAuth`	Acceptable values: `true` | `false` Enter true if your proxy server requires authentication. The authentication parameters will be specified by the following tags: `proxyType`, `proxyRealm`, `proxyUseGlobal`, `proxyUserName`, and `proxyPassword`.
proxyType	Acceptable values: `Basic` | `Digest` Choose the type of proxy server this provider. Note: For more information about basic or digest authentication, see `http://www.faqs.org/rfcs/rfc2617.html`.
`proxyRealm`	Enter the name of the realm of the proxy server that is accessed by the user according to the login information described later in the table. If you do not know the name of the realm, then contact the administrator of the proxy server.
`proxyUseGlobal`	Acceptable values: true | `false` If true, then the `<proxyUser>` and `<proxyPassword>` values are used for all users. Users do not see the Proxy Authentication section on the Source tab and Personalize screen. If false, then page designer must log in using the Proxy Authentication section on the Source tab when they define the portlet. The end user must log in using the Proxy Authentication section on the Personalize screen. If `<proxyUsername>` and `<proxyPassword>` are given, then they are only used for public users.
`proxyUserName`	Enter the username to log in to the proxy server.
`ProxyPassword`	Enter the password for the specified username. You must prefix ! before your plain password text. It will then be encrypted in the `provider.xml` file for protection once the producer starts.

The following is a basic example of using a proxy to access data outside a firewall:

```
<proxyInfo class="mycompany.portal.provider.v2.ProxyInformation">
<httpProxyHost>www-proxy.mycompany.com</httpProxyHost>
<httpProxyPort>80</httpProxyPort>
<proxyUseAuth>false</proxyUseAuth>
</proxyInfo>
```

The following example requires a login and basic authentication for all users for the proxy server:

```
<proxyInfo class="mycompany.portal.provider.v2.ProxyInformation">
<httpProxyHost>stport823.mycompany.com</httpProxyHost>
<httpProxyPort>8080</httpProxyPort>
<proxyUseAuth>true</proxyUseAuth>
<proxyType>Basic</proxyType>
<proxyRealm>stport823</proxyRealm>
<proxyUseGlobal>false</proxyUseGlobal>
</proxyInfo>
```

B.2.2 Configuring the OmniPortlet Producer to Access Other Relational Databases

Perform this step if you want to access other relational databases with OmniPortlet. The OmniPortlet SQL data source is preconfigured to access Oracle Databases using the Oracle JDBC driver, and ODBC data sources using the JDBC-ODBC driver from Sun Microsystems.

> **See Also:** For a list of supported databases, Certification Matrix for Oracle Application Server and DataDirect JDBC available at
>
> http://www.oracle.com/technology/tech/java/oc4j/htdocs/datadirect-jdbc-certification.html

You can configure the OmniPortlet SQL data source to access other relational databases by using DataDirect JDBC drivers. To do this, perform the following steps:

- Installing DataDirect JDBC Drivers
- Registering DataDirect Drivers in OmniPortlet

B.2.2.1 Installing DataDirect JDBC Drivers

DataDirect JDBC drivers are packaged in a single ZIP file containing the different drivers used to access supported databases. Download the ZIP file from the following location:

http://www.oracle.com/technology/software/products/ias/htdocs/utilsoft.html

To install DataDirect JDBC drivers, perform the following steps:

1. Unzip the contents of the ZIP file into a temporary directory, for example `/temp/datadirect`.
2. Create the *ORACLE_HOME*`/j2ee/<`*OC4J_INSTANCE_HOME*`>/applib` directory if it does not already exist.
3. From the `/temp/datadirect/lib` directory, copy the DataDirect JDBC drivers to the *ORACLE_HOME*`/j2ee/<`*OC4J_INSTANCE_HOME*`>/applib` directory.
4. Check the configuration of the OC4J_Portal instance to ensure that the DataDirect libraries are loaded. To do this, perform the following steps:

a. Open the *ORACLE_HOME*/j2ee/<*OC4J_INSTANCE_HOME*>/config/server.xml file.

b. Add the Extensible Markup Language (XML) entry, <code-source path="../applib"/>, to the oracle.portal shared-library tag:

```
<shared-library name="oracle.portal" version="10.1.3.2.0" library
compatible="true">
  <code-source path="../applib"/>
```

c. Save and close the file.

B.2.2.2 Registering DataDirect Drivers in OmniPortlet

OmniPortlet is implemented as a Web producer and all the configuration properties are stored in the provider.xml file. To use DataDirect JDBC drivers with OmniPortlet, you must register these drivers in the provider.xml file.

To register the new DataDirect JDBC drivers, perform the following steps:

1. Back up the file, *ORACLE_HOME*/j2ee/<*OC4J_INSTANCE_HOME*>/applications/portalTools/omniPortlet/WEB-INF/providers/omniPortlet/provider.xml, and then open the file.
2. Add the drivers that you want to use for the SQL data source configuration entry. To do this, perform the following:
   a. Search for the XML tag, driverInfo.
   b. Add a new entry after the last driverInfo tag.

   Following is an example showing a Microsoft SQL Server entry:

   - For OmniPortlet version 9.0.4.1 or later:

```
<!-- registration of DataDirect Connect for JDBC SQL Server driver -->
<driverInfo class="oracle.webdb.reformlet.data.jdbc.JDBCDriverInfo">
  <name>Microsoft SQL Server</name>
  <sourceDataBase>other</sourceDataBase>
  <subProtocol>sqlserver</subProtocol>
  <connectString>mainProtocol:subProtocol://databaseName</connectString>
  <driverClassName>com.oracle.ias.jdbc.sqlserver.SQLServerDriver
  </driverClassName>
  <dataSourceClassName>com.oracle.ias.jdbcx.sqlserver.SQLServerDataSource
  </dataSourceClassName>
  <connHandlerClass>oracle.webdb.reformlet.data.jdbc.JDBCConnectionHandler
  </connHandlerClass>
  <connPoolSize>5</connPoolSize>
  <loginTimeOut>30</loginTimeOut>
</driverInfo>
```

   - For OmniPortlet versions before 9.0.4.1:

```
<!-- registration of DataDirect Connect for JDBC SQL Server driver -->
<driverInfo class="oracle.webdb.reformlet.data.jdbc.JDBCDriverInfo">
  <name>Microsoft SQL Server</name>
  <sourceDataBase>other</sourceDataBase>
  <subProtocol>sqlserver</subProtocol>
  <connectString>mainProtocol:subProtocol://databaseName</connectString>
  <driverClassName>com.oracle.ias.jdbc.sqlserver.SQLServerDriver
  </driverClassName>
  <connHandlerClass>
  oracle.webdb.reformlet.data.jdbc.JDBCODBCConnectionHandler
  </connHandlerClass>
  <connPoolSize>5</connPoolSize>
```

```
    <loginTimeOut>30</loginTimeOut>
</driverInfo>
```

Table B–2 describes the parameters in the `driverInfo` property.

Table B–2 Parameters in the driverInfo Property

Parameter	Description
`name`	Name of the database you want to use. This name will be used on the Source tab of the OmniPortlet wizard.
`sourceDataBase`	Internal value. Set the value to `other`.
`subProtocol`	JDBC subprotocol name used by OmniPortlet to create the connection string, for example `sqlserver`, `sybase`, or `db2`. To get the list of subprotocol names, see the DataDirect JDBC driver documentation using the links provided at the end of this section.
`connectString`	Description of the connect string format. For DataDirect drivers, the format is: `mainProtocol:subProtocol://databaseName`
`driverClassName`	Name of the driver class. To get the different values, see the DataDirect JDBC driver documentation using the links provided at the end of this section.
`dataSourceClassName`	Name of the data source class that implements connection pooling. This parameter is only available in OmniPortlet version 9.0.4.1 or later. See Table B–3 for the right data source class name for your driver.

`connHandlerClass`	Class used by OmniPortlet to manage the driver and connection pooling. The value is either of the following:
	■ For OmniPortlet version 9.0.4.1 or later:
	`oracle.webdb.reformlet.data.jdbc.JDBCConnectionHandler`
	■ For OmniPortlet versions before 9.0.4.1:
	`oracle.webdb.reformlet.data.jdbc.JDBCODBCConnectionHandler`

`connPoolSize`	Minimum number of connections that are opened by the connection pool.
`loginTimeOut`	Maximum time, in seconds, that this data source will wait while attempting to connect to a database.

Table B–3 lists the values for the `driverClassName` and `dataSourceClassName` properties for specific DataDirect JDBC drivers.

Table B–3 Parameters and Values for driverClassName and dataSourceClassName

DataDirect Drivers Supported	Properties

Microsoft SQL Server	■ Parameter: `driverClassName`
	Value:
	`com.oracle.ias.jdbc.sqlserver.SQLServerDriver`
	■ Parameter: `dataSourceClassName`
	Value:
	`com.oracle.ias.jdbcx.sqlserver.SQLServerDataSource`

Table B–3 (Cont.) Parameters and Values for driverClassName and

DataDirect Drivers Supported	Properties
Sybase	■ Parameter: `driverClassName` Value: `com.oracle.ias.jdbc.sybase.SybaseDriver` ■ Parameter: `dataSourceClassName` Value: `com.oracle.ias.jdbcx.sybase.SybaseDataSource`
DB2	■ Parameter: `driverClassName` Value: `com.oracle.ias.jdbc.db2.DB2Driver` ■ Parameter: `dataSourceClassName` Value: `com.oracle.ias.jdbcx.db2.DB2DataSource`
Informix	■ Parameter: `driverClassName` Value: `com.oracle.ias.jdbc.informix.InformixDriver` ■ Parameter: `dataSourceClassName` Value: `com.oracle.ias.jdbcx.informix.InformixDataSource`

3. Save the `provider.xml` file.
4. Stop and start the Oracle Application Server instance. To do so, navigate to your ORACLE_HOME, then to the subdirectory `opmn/bin`.
5. Type the following command:

```
opmnctl restartproc process-type=<OC4J_Instance_Name>
```

Note: If you are using OmniPortlet in a multiple nodes configuration, for example, in a clustering or load-balancing environment, then you must manually copy the `provider.xml` file on each node.

See Also: For more information about DataDirect JDBC drivers, see the following documentation:

- Certification Matrix for Oracle Application Server and DataDirect JDBC available at

  `http://www.oracle.com/technology/tech/java/oc4j/htdocs/datadirect-jdbc-certification.html`

- The OC4J page on Oracle Technology Network (OTN) is available at

  `http://www.oracle.com/technology/software/products/ias/htdocs/utilsoft.html`

- How to use DataDirect JDBC drivers in OmniPortlet in Chapter 16, "Creating Portlets with OmniPortlet".

Troubleshooting Information

If you encounter errors or problems when configuring or using the OmniPortlet producer, then see Appendix G, "Troubleshooting WebCenter Applications" for troubleshooting information.

B.2.3 Configure Portal Tools and Web Producers (Optional)

To ensure that the Portal Tools (OmniPortlet and OracleAS Web Clipping) producers, locally built, and custom built Web producers work properly, in the middle-tier environment, some additional configuration is required. If OmniPortlet or any other Web producers already have customization in the file system, then PDK-Java provides a Preference Store Migration/Upgrade Utility that can be used to migrate the existing customizations to the database and upgrade customizations from earlier releases. See Section B.4, "Portlet Preference Store Migration Utilities" for more information about the PDK Preference Store Migration Utility.

Configuring Portal Tools Producers in the Multiple Middle-Tier Environment

By default, the OmniPortlet producer uses the file-based Preference Store. In a multiple middle-tier environment, you must configure the File Preference Store to a shared file system, or use the database Preference Store (`DBPreferenceStore`).

To configure OmniPortlet's File Preference Store to use a shared file system, set the `<rootDirectory>` tag in OmniPortlet's `provider.xml` file to the shared file system path:

```
<preferenceStore class="oracle.webdb.reformlet.ReformletFilePreferenceStore">
    <name>omniPortletprefStore</name>
    <useHashing>true</useHashing>
    <rootDirectory>shared-file-system-path</rootDirectory>
 </preferenceStore>
```

To configure OmniPortlet producer to use DBPreferenceStore, perform the following steps:

1. Navigate to the directory *ORACLE_HOME*`/j2ee/OC4J_Portal/applications/jpdk/jpdk/doc/dbPreferenceStore`. For example:

   ```
   cd ORACLE_HOME/j2ee/OC4J_
   Portal/applications/jpdk/jpdk/doc/dbPreferenceStore
   ```

2. Create a user on the database containing the PORTAL schema, and grant create resource and connect privileges, using the `create user` and `grant connect` commands in SQL*Plus. Substitute the actual password in the following command. Do not use the default password of `welcome`, as this poses a security risk. For example:

   ```
   create user prefstore identified by password;
   grant connect, resource to prefstore;
   ```

3. Connect as user `prefstore` and run the `jpdk_preference_store2.sql` script as follows in SQL*Plus:

   ```
   @jpdk_preference_store2
   ```

4. Add the following entry to the file `data-sources.xml`, located in the directory *ORACLE_HOME*`/j2ee/OC4J_Portal/config`:

   ```
   <connection-pool name="ConPool_1">
   <connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"
   ```

```
username="prefstore"
password="password"
url="jdbc:oracle:thin:@infra.host.com:1521:orcl">
</connection-factory>
</connection-pool>

<managed-data-source name="PooledConnection"
connection-pool-name="ConPool_1"
jndi-name="jdbc/PooledConnection"/>
```

Note: Embedding passwords in deployment and configuration files poses a security risk. If you do not want to use a clear text password in the `data-sources.xml` file, then you can create an indirect password by performing the following steps:

1. Edit the *ORACLE_HOME*`/j2ee/OC4J_Portal/config/jazn-data.xml` file to add the `prefstore` user in the `jazn.com` realm as shown in the following example (You can create a new realm for this instead of using the `jazn.com` realm):

   ```
   <realm>
       <name>jazn.com</name>
       <users>
         <user>
            <name>prefstore</name>
            <display-name>OmniPortletprefstore</display-name>
            <description>OmniPortlet prefstore</description>
            <credentials>!welcome</credentials>
         </user>
         <user>
            ...
   ```

   Note that the password is included in the `<credentials>` element and is prefixed with an exclamation (!) mark. The next time Oracle Containers for J2EE (OC4J) reads the `jazn-data.xml`, it will rewrite the file with this password obfuscated.

2. Edit the *ORACLE_HOME*`/j2ee/OC4J_Portal/config/data-sources.xml` file again to use the indirect password that you created in the previous step by replacing the password attribute as follows:

   ```
   password="->jazn.com/prefstore"
   ```

   For more information about creating an indirect password, see *Oracle Containers for J2EE Security Guide*.

---

5. Edit the file `provider.xml` located in the directory *ORACLE_HOME*`/j2ee/OC4J_Portal/applications/portalTools/omniPortlet/WEB-INF/providers/omniPortlet`. Edit the `preferenceStore` tag as shown in bold:

   ```
   <provider class="oracle.webdb.reformlet.ReformletProvider">
      <vaultId>0</vaultId>
      <session>true</session>
      <preferenceStore
   class="oracle.portal.provider.v2.preference.DBPreferenceStore">
         <name>omniPortletprefStore</name>
         <connection>jdbc/PooledConnection</connection>
      </preferenceStore>
   ```

6. Restart OC4J_Portal.

You can find more information about configuring the database Preference Store in the PDK article titled "Installing the DBPreferenceStore Sample (V2)", (with the filename `installing.db.preference.store.v2.html`) located in the `pdksoftware10132.zip` file located on the Oracle Technology Network (http://www.oracle.com/technology/products/webcenter/index.html).

If you have already created an OmniPortlet instance with customizations in the file system, then you must migrate these customizations to the database using the Preference Store Migration Utility. To run the migration utility, perform the following steps:

1. Navigate to the middle-tier Oracle home directory using the following command:

```
cd ORACLE_HOME
```

2. Run the following command to migrate OmniPortlet data from a file-based Preference Store (`FilePreferenceStore`) to the database Preference Store (`DBPreferenceStore`):

```
java -classpath
lib/dms.jar:jdbc/lib/ojdbc14dms.jar:portal/jlib/pdkjava.jar:portal/jlib/
ptlshare.jar oracle.portal.provider.v2.preference.MigrationTool -mode
filetodb -pref1UseHashing true -pref1RootDirectory
portal/portletdata/tools/omniPortlet
-pref2User <User_Name> -pref2Password <User_Password> -pref2URL
jdbc:oracle:thin:@infra.host.com:1521:orcl
```

See Section B.4, "Portlet Preference Store Migration Utilities" for more information about the PDK Preference Store Migration Utility.

6. Typically, you perform the HTTP Proxy configuration for OmniPortlet and Web Clipping before you configure the LBR. To do it after the LBR is configured, perform the following steps:

   a. The Portal Tools configuration information is stored in the `provider.xml` file on the middle-tier server. You must update the configuration directly on one middle tier (for example, **M1**) and then propagate it across all middle tiers front-ended by the LBR. Before you do this, you must shut down all middle tiers except **M1**.

   b. You can change the HTTP Proxy settings in the `provider.xml` file. For more information, see Section B.2.1, "Configuring the OmniPortlet Producer to Access Data Outside a Firewall".

   c. Propagate the changes made to the `provider.xml` file to middle tier **M2**:

      – Copy *ORACLE_HOME*`/j2ee/OC4J_Portal/applications/portalTools/omniPortlet/WEB-INF/providers/omniPortlet/provider.xml` from **M1** to **M2**.

      – Copy *ORACLE_HOME*`/j2ee/OC4J_Portal/applications/portalTools/webClipping/WEB-INF/providers/webClipping/provider.xml` from **M1** to **M2**.

7. Copy the *ORACLE_HOME*`/j2ee/OC4J_Portal/config/data-sources.xml` file from **M1** to **M2**.

8. Copy the *ORACLE_HOME*`/j2ee/OC4J_Portal/config/jazn-data.xml` file file from **M1** to **M2**.

9. Restart middle tier **M2**.
10. Update portlet producer registration in your WebCenter application. Change the first part of the producer registration URL from `http://m1.abc.com:7777/` to `http://lbr.abc.com/`.
11. Verify that OmniPortlet and the Web Clipping producers work properly through the LBR, by going to the test pages at the following URLs.
    - OmniPortlet Producer: `http://lbr.abc.com/portalTools/omniPortlet/producers/omniPortlet`

      If you see the "No Portlets Available" message under the Portlet Information section in the OmniPortlet Producer test page, then you may not have configured OmniPortlet correctly in Step 1. If OmniPortlet is configured correctly, then the OmniPortlet and Simple Parameter Form portlets are available on the test page.
    - Web Clipping Producer: `http://lbr.abc.com/portalTools/webClipping/producers/webClipping`

      Note: If you want to use the OracleAS Web Clipping portlet, or the Web Page Data Source for OmniPortlet, then you must also enable session binding in Oracle Web Cache.

B.3 Web Clipping Portlet Configuration Tips

Before you use Web Clipping, you must perform some administrative tasks, including the following:

- Configuring the Web Clipping Repository
- Configuring HTTP or HTTPS Proxy Settings
- Securing the Web Clipping Producer

B.3.1 Configuring the Web Clipping Repository

Web clippings have definitions that must be stored persistently in the Oracle Metadata Services store. Alternatively, the Web Clipping repository can also be hosted by an Oracle Database.

You can view the Web Clipping repository configuration by accessing the Web Clipping Producer Test Page at:

`http://<host>:<port>/portalTools/webClipping/providers/webClipping`

The Web Clipping Test page automatically detects whether or not the Web Clipping producer is configured with a valid repository. If it is not, then the **Status** column for the Web Clipping Repository displays **Not Configured**. Figure B–1 shows the Web Clipping Test page.

Figure B–1 Web Clipping - Producer Test Page

You cannot change the Web Clipping configuration information using the Producer Test Page. As WebCenter Framework administrator, you can configure the Web Clipping repository by setting appropriate values in the `provider.xml` file located in the `<ORACLE_HOME>\j2ee\home\applications\portalTools\webClipping\WEB-INF\providers\webClipping` directory.

By default, the Web Clipping producer is configured to use Oracle Metadata Services as the Web Clipping Repository. You can select either of the following as the Web Clipping repository:

- Oracle Metadata Services
- Other Oracle Databases of versions 9*i* or later

To change the repository settings, perform the following tasks:

1. Open the file, `<ORACLE_HOME>\j2ee\home\applications\portalTools\webClipping\WEB-INF\providers\webClipping\provider.xml` in a text editor.
2. Specify one of the following to be your Web Clipping repository:
   - **Oracle Metadata Services (default)**: If you select Oracle Metadata Services as your Web Clipping repository, then the Web Clipping definition is saved in the file system. You can use Web Clipping even without a database.

     This is the default Web Clipping repository option. To use Oracle Metadata Services as the repository, specify the path to the `mds-config.xml` file in the `mdsConfigLocation` entry in the `provider.xml` file (by following the guidelines under this entry) as shown in Example B–1.

Example B–1 Setting Oracle Metadata Services as Web Clipping Repository

```
- <repositoryInfo class="oracle.portal.wcs.provider.info.MdsInformation">
- <!--
    Specify the location of the MDS configuration file here. It can be
    absolute or relative. The definition of absolute pathname is system
```

```
   dependent. On UNIX systems, a pathname is absolute if it begins with a
   single forward slash "/". On Microsoft Windows systems, a pathname is
   absolute if it begins with a drive specifier followed by single
   backslash "\", or if begins with a double backslash "\\".

   When a relative path is specified, it is assumed to be relative to the
   base directory identified by
   OH/j2ee/home/applications/portalTools/webClipping/WEB-INF or
   OH/j2ee/OC4J_Portal/applications/portalTools/webClipping/WEB-INF
   depending on where the portalTools EAR file is deployed.
 -->
 <mdsConfigLocation>mds-config.xml</mdsConfigLocation>
 </repositoryInfo>
```

Note: The `mds-config.xml` file contains the metadata store configuration information, including the path to the metadata store. You can change the path to the metadata store by editing the `metadata-path` property in *<ORACLE_HOME>*`\j2ee\home\applications\portalTools\webClipping\WEB-INF\mds-config.xml` as shown in the following example:

```
<property name="metadata-path"
value="portletdata/tools/webClipping"/>
```

For a multiple middle tier deployment, change the metadata path to a shared file system.

- **Other Oracle9i or later Database**: If you select this option, then the Web Clipping repository is stored in a user defined schema in Oracle Database. In this case, you must specify the connection parameters of Oracle Database. You can also specify this option if you want to use the OracleAS Infrastructure database as the Web Clipping repository.

  To use Oracle Database as the Web Clipping repository, comment out the default repository setting (shown in Example B–1) in the `provider.xml` file and uncomment the entry shown in Example B–2.

Example B–2 Setting Oracle Database 9i or Later as Web Clipping Repository

```
- <!--
 Uncomment the following and set the connection information to use database as the
repository
    <repositoryInfo class="oracle.portal.wcs.provider.info.DatabaseInformation">
       <useRAA>false</useRAA>
       <databaseHost>mycompany.dbhost.com</databaseHost>
       <databasePort>1521</databasePort>
       <databaseSid>iasdb</databaseSid>
       <databaseUsername>scott</databaseUsername>
       <databasePassword>!tiger</databasePassword>
       <useASO>false</useASO>
    </repositoryInfo>
 -->
```

In this entry, you must set the database connection parameters, which can be described as follows:

  - `databaseHost`: Host name of the computer running the database.

- `databasePort`: Port number of the database listener.
- `databaseSid`: The SID of the database. Use this format when the connect string is in the format, `host:port:sid`. For example: `myhost.company.com:1521:mydb`.
- `databaseUsername`: The database account user name.
- `databasePassword`: The database account password. To ensure security, you must enter plain text passwords prefixed with the `!` character as shown in Example B–2, to allow the Web Clipping Producer to encrypt the password in the producer.

If you require a secure database connection, then enable the Advanced Security Option (ASO) by setting the `useASO` entry to `true`.

See Section B.3.3, "Securing the Web Clipping Producer" for more information about configuring the Advanced Security Option.

3. Save the `provider.xml` file.
4. Restart the OC4J instance.

B.3.2 Configuring HTTP or HTTPS Proxy Settings

Your HTTP or HTTPS proxy settings must be set to enable the Web Clipping Studio to connect to Web sites outside of your firewall. You can specify the settings by manually editing the `provider.xml` file.

As the WebCenter Application administrator, you can set proxy settings manually according to your HTTP or HTTPS configuration. Edit the appropriate entries in the `provider.xml` file located in the `<ORACLE_HOME>\j2ee\home\applications\portalTools\webClipping\WEB-INF\providers\webClipping` directory.

Example shows the relevant portion of `provider.xml`.

```
- <!--
 proxy information: Fill the following up if you have a proxy
 server between the provider and external sites.
   <proxyInfo class="oracle.portal.provider.v2.ProxyInformation">
      <httpProxyHost>yourproxy.yourdomain.com</httpProxyHost>
      <httpProxyPort>80</httpProxyPort>
      <dontProxyFor>.yourdomain.com</dontProxyFor>
   </proxyInfo>

  -->
```

Table B–1, available earlier in this appendix, describes the proxy settings you must make in the `provider.xml` file. The descriptions in the table are applicable for Web Clipping producers also.

Note: For environments that use a proxy server to reach external Web sites, you can use the `dontProxyFor` entry to specify the proxy exception list. Web Clipping uses the proxy exception list to restrict users from clipping content from unauthorized external Web sites.

Users attempting to reach a Web site in one of the listed domains, from the Web Clipping Studio, will then receive an HTTP timeout error.

B.3.3 Securing the Web Clipping Producer

The preceding sections described the administrative tasks that must be performed before you are able to use the Web Clipping producer. The following sections describe some security configuration options that you should implement to enable the Web Clipping producer to access trusted sites and encrypt the channel between itself and the database:

- Adding Certificates for Trusted Sites
- Configuring Oracle Advanced Security for the Web Clipping Producer

B.3.3.1 Adding Certificates for Trusted Sites

When a user navigates to a secure site, the Web site typically returns a certificate, identifying itself to the user when asking for secure information. If the user accepts the certificate, then the certificate is placed into the list of trusted certificates of the browser so that a secure channel can be opened between the browser and that server. Like a Web browser, the Web Clipping producer acts as an HTTP client to external Web sites. In order for the Web Clipping producer to keep track of trusted sites, it makes use of a file that stores the certificates of those sites, namely the `ca-bundle.crt` file, located in the *ORACLE_HOME*`/portal/conf` directory.

The shipped `ca-bundle.crt` is an exported version of the trusted server certificate file from Oracle Wallet Manager. The default trusted server certificate in Oracle Wallet Manager does not cover all possible server certificates that exist on the Web. For this reason, when a user navigates to a secure server using HTTPS, the user can get an Secure Sockets Layer (SSL) Hand-shake failed exception in the Web Clipping Studio. To solve this problem, the `ca-bundle.crt` file must be augmented with new trusted sites that are visited. As a WebCenter Application administrator, you must do the following to extend the shipped `ca-bundle.crt` file:

1. Use a browser (preferably Internet Explorer) to download the root server certificate from each external HTTPS Web site in BASE64 format that is visited, and is missing from the trusted certificate file.
2. Use Oracle Wallet Manager to import each certificate.
3. Export the trusted server certificates into a file and copy its content into the `ca-bundle.crt` file.

For more information about Oracle Wallet Manager, see the *Oracle Database Advanced Security Administrator's Guide* in the Oracle Database documentation on OTN, `http://www.oracle.com/technology/`.

B.3.3.2 Configuring Oracle Advanced Security for the Web Clipping Producer

The Web Clipping producer can use Oracle Advanced Security Option (ASO) to secure and encrypt the channel between itself and the database that hosts the Web Clipping repository. This feature is available only if you have selected any Oracle Database as the Web Clipping repository. This feature is disabled by default as Oracle Metadata Services is the default Web Clipping repository. To enable it, perform the following steps:

1. Open the file, <*ORACLE_HOME*>`\j2ee\home\applications\portalTools\webClipping\WEB-INF\providers\webClipping\provider.xml` in a text editor.
2. Under the repository settings section in the file (shown in Example B–2), set the `useASO` entry to `true`.
3. Save the `provider.xml` file.

In addition, you must set the following ASO configuration parameters in the `sqlnet.ora` file to ensure that the database connections created between the Web Clipping producer and the database hosting the Web Clipping Repository are using ASO. See the *Oracle Advanced Security Administrator's Guide* for the list of values to use as all possible combinations of parameters are described in detail.

- `SQLNET.AUTHENTICATION_SERVICES` -- This parameter is used to select a supported authentication method in making database connections with ASO. See the *Oracle Advanced Security Administrator's Guide* for more information about setting this parameter.
- `SQLNET.CRYPTO_SEED` -- This parameter denotes the cryptographic seed value (FIPS 140-1 setting), used in making database connections with ASO.

  See the *Oracle Advanced Security Administrator's Guide* for more information about setting this parameter.

> **Note:** When setting these parameters after the initial configuration (where the database parameters are already set up), the database connections are assumed to be open already. Because enabling ASO affects all connections made to the database, it is advisable to restart the OC4J instance containing the Web Clipping producer to reset all the current connections to now use ASO. You would also must do this when disabling ASO.

B.4 Portlet Preference Store Migration Utilities

A preference store is a mechanism for storing information like user preference data, portlet and producer settings, or even portlet data. Preferences can be stored in a database, which is recommended for high availability configurations, or a file system. You can migrate the following stores for your WebCenter applications:

- JPS Portlet Preference Store
- PDK-Java Portlet Preference Store
- Web Clipping Repository

B.4.1 JPS Portlet Preference Store

A WSRP container preference store is a mechanism used for persisting consumer registration and portlet preference data. Currently, there are two preference store implementations, *database preference store* and *file preference store*. A database preference store persists data using a relational database. A file preference store persists data using the file system. For WSRP Release 2, the default is to use the file preference store at *ORACLE_HOME*`/portal/portletdata`. For the 10.1.2 version of the portlet container, the default (and only) preference store is the database preference store.

With the introduction of the file preference store, you can remove the dependency on the database. The file preference store is used as the default preference store. Therefore, you may want to migrate from an existing database preference store to a file preference store. Note, however, that the database preference store is recommended for high availability configurations.

B.4.1.1 PersistenceMigrationTool

The WSRP container preference store migration utility, `PersistenceMigrationTool`, enables you to migrate existing data between

different preference stores (for example, from a database preference store to a file preference store). This utility also enables upgrading users to ensure that their existing locale-specific portlet preference data uses a naming format compatible with the latest JPS release. You can also use this utility to migrate between source and destination stores of the same type. This enables data to be moved from one database store to another.

The syntax of the `PersistenceMigrationTool` is:

Note: You can also obtain this syntax from the command line by entering the following command:

```
java -classpath
ORACLE_HOME/j2ee/wsrp/shared-lib/oracle.wsrp/1.0/
  wsrp-container.jar:
ORACLE_HOME/javacache/lib/cache.jar:
ORACLE_HOME/webservices/lib/saaj-api.jar:
ORACLE_HOME/webservices/lib/orasaaj.jar:
ORACLE_HOME/jdbc/lib/ojdbc14.jar
oracle.portlet.server.containerimpl.PersistenceMigrationTool
```

```
java oracle.portlet.server.containerimpl.PersistenceMigrationTool
-sourceType file | db
-destType file | db
{-sourcePath dir |
 -sourceUsername username -sourcePassword password -sourceDatabase db}
{-destPath dir | destUsername username -destPassword password -destDatabase db}
[-debug]
```

where

`sourceType` indicates whether the source store is in a file or database. You can have source and destination stores of the same type. Hence, you can migrate from one database to another or one file system to another.

`destType` indicates whether the destination store is in a file or database. You can have source and destination stores of the same type. Hence, you can migrate from one database to another or one file system to another.

`sourcePath` is the location of a file-based preference store. This argument is required when `sourceType` is `file`.

`sourceUsername` is the database user name for a preference store database. This argument is required when `sourceType` is `db`.

`sourcePassword` is the database password for a preference store database. This argument is required when `sourceType` is `db`.

`sourceDatabase` is the name of a preference store database. This argument is required when `sourceType` is `db`.

`destPath` is the location of a file-based preference store. This argument is required when `destType` is `file`.

`destUsername` is the database user name for a preference store database. This argument is required when `destType` is `db`.

`destPassword` is the database password for a preference store database. This argument is required when `destType` is `db`.

destDatabase is the name of a preference store database. This argument is required when destType is db.

debug turns on full logging through standard output to enable users to diagnose issues that arise when the tool runs.

Example B–3 demonstrates running the PersistenceMigrationTool utility. In this example, preferences from a database store are copied to a file store.

Example B–3 Running the PersistenceMigrationTool Utility

```
java oracle.portlet.server.containerimpl.PersistenceMigrationTool -sourceType
db -sourceUsername scott -sourcePassword tiger -sourceDatabase
abc.mycompany.com:1521:e10gdev3 -destType file -destRoot /data/prefs
```

B.4.1.2 How to Determine and Set Your Preference Store

To determine which data store is used to persist data, you must add the persistentStore JNDI environment variable to the web.xml file (*ORACLE_HOME*\j2ee\OC4J_Portal\applications\portletapp\wsrp-samples\WEB-INF\web.xml). The default value for this variable is File. If you keep the default value, then you must also add the fileStoreRoot variable to specify the file system location for the file store. The default value for this variable is portletdata, and this value is relative to the *ORACLE_HOME*/portal directory, which means the default location for the file store is *ORACLE_HOME*/portal/portletdata.

If you deployed the sample WSRP producer or if your application is based on this sample producer, then these JNDI environment variables may already be present in the web.xml file. Add the code excerpt shown in Example B–4 to the web.xml file (the environment variables and their default values are shown in bold):

Example B–4 persistentStore and fileStoreRoot Variables in the web.xml File

```
<env-entry>
    <env-entry-name>oracle/portal/wsrp/server/persistentStore</env-entry-name>
    <env-entry-type>java.lang.String</env-entry-type>
    <env-entry-value>File</env-entry-value>
</env-entry>
<env-entry>
    <env-entry-name>oracle/portal/wsrp/server/fileStoreRoot</env-entry-name>
    <env-entry-type>java.lang.String</env-entry/type>
    <env-entry-value>portletdata</env-entry-value>
</env-entry>
```

> **Note:** You can leave the default values in the file if you do not want to explicitly select the type of preference store or location for the store.

B.4.2 PDK-Java Portlet Preference Store

PDK-Java has two PreferenceStore implementations, DBPreferenceStore and FilePreferenceStore. DBPreferenceStore persists data using a JDBC compatible relational database and FilePreferenceStore persists data using the file system.

MigrationTool

If you have already installed OracleAS PDK, then you can manage the information stored in the preference store by using the Preference Store Migration and Upgrade

Utility, which is included in the `pdkjava.jar` file. Note that you must run this tool from *`ORACLE_HOME`*. The syntax of the migration utility is:

> **Note:** You can also obtain this syntax from the command line by entering the following command:
>
> ```
> java -classpath lib/dms.jar:jdbc/lib/ojdbc14dms.jar:
>   portal/jlib/pdkjava.jar:portal/jlib/ptlshare.jar
>   oracle.portal.provider.v2.preference.MigrationTool
> ```

```
java -classpath lib/dms.jar:jdbc/lib/ojdbc14dms.jar:portal/jlib/pdkjava.jar:
                portal/jlib/ptlshare.jar
 oracle.portal.provider.v2.preference.MigrationTool
  -mode [file | db | filetodb | filetofile | dbtofile | dbtodb]
  [-remap language | locale]
  [-countries iso_country_code]
  [-pref1UseHashing true | false]
  {-pref1RootDirectory directory |
   -pref1User username -pref1Password password -pref1URL url}
  [-pref1UseHashing true | false]
  {-pref2RootDirectory directory |
   -pref2User username -pref2Password password -pref2URL url}
  [-upfixwpi filename]
```

where:

`-mode` is the mode in which you want to run the Preference Store Migration and Upgrade Utility.

- `file` or `db` indicates that you want to run in upgrade mode. See Upgrade Mode for more information about this mode.
- `filetodb`, `filetofile`, `dbtofile`, or `dbtodb` indicates that you want to run in migration mode. See Migration Mode for more information about this mode.

`-remap` is the `localePersonalizationLevel` (`language` or `locale`). Note that you only must use this option if you want to change `localePersonalizationLevel` as part of your upgrade/migration.

`-countries` specifies a prioritized list of ISO country codes, indicating your order of preference in a collision between remapped preferences for different countries. `-countries` is only meaningful if you also specified the `-remap` option.

`-pref1UseHashing` is whether you want to employ hashing on the source for this operation.

`-pref1RootDirectory` is the path of a source file system, for example, `j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample`.

`-pref1User` is the user name for a source database.

`-pref1Password` is the password for a source database.

`-pref1URL` is the URL to a source database, for example, `jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid`.

`-pref2UseHashing` is whether you want to employ hashing on the destination for this operation.

`-pref2RootDirectory` is the path of a destination file system, for example, `j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample`.

-pref2User is the user name for a destination database.

-pref2Password is the password for a destination database.

-pref2URL is the URL to a destination database, for example, jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid.

- upfixwpi indicates a log file for the operation.

> **Note:** After running the utility, it is recommended that you restart the OC4J instance with the portlet container and Oracle HTTP Server to ensure that the latest preference store information is used.

Upgrade Mode

Use an upgrade mode to upgrade data in place, and to modify existing locale-specific preferences in the preference store so that the naming format used is compatible with the current version of OracleAS Portal and a given localePersonalizationLevel setting.

Table B–4 describes the upgrade modes in which you can run the utility.

Table B–4 Upgrade Modes in Which to Run the Utility

Mode	Description
file	Use when data in a FilePreferenceStore must be upgraded.
db	Use when data in a DBPreferenceStore must be upgraded.

An upgrade mode can be used in the following scenarios:

- You have upgraded from OracleAS PDK 9.0.4.0.0 or earlier and want to use existing portlets with the default localePersonalizationLevel setting of language (In earlier releases, the default setting was locale).
- You have upgraded from OracleAS Portal 9.0.2.0.0 or earlier and want to use existing portlets with a localePersonalizationLevel setting of locale (OracleAS Portal now uses different names for some locales and therefore some existing data must be remapped).
- You want to change the localePersonalizationLevel for an existing portlet from locale to language or vice-versa.

When using an upgrade mode, you must use the -remap option to specify the localePersonalizationLevel (language or locale) that you are upgrading to. You can also use the -countries option to specify a prioritized list of ISO country codes, indicating your order of preference in a collision between remapped preferences for different countries. For example, if you specify -remap language -countries GB,US in the command, then it means that if the utility comes across both US English and British English preferences (en_US and en_GB) in a given preference store, it will remap the British English preference to become the English-wide preference (en).

> **Note:** While running the utility in db mode, for the pref1User and pref1password properties, use the values specified in the JDBC connection definition in the *<j2ee-home>*/config/data-sources.xml file.

Migration Mode

Use a migration mode to copy data from a source preference store to a target preference store. When the utility is run in this mode, the preference stores for all the portlet definitions are updated.

Table B–5 describes the migration modes in which you can run the utility.

Table B–5 Migration Modes in Which to Run the Utility

Mode	Description
`filetodb`	Use when data must be copied from a `FilePreferenceStore` to a `DBPreferenceStore`.
`filetofile`	Use when data must be copied from one `FilePreferenceStore` to another `FilePreferenceStore` that is in a different location.
`dbtofile`	Use when data must be copied from a `DBPreferenceStore` to a `FilePreferenceStore`.
`dbtodb`	Use when data must be copied from one `DBPreferenceStore` to another `DBPreferenceStore` that is based on a different database table.

If the destination for the operation is a database, then you must ensure that the destination database has the appropriate tables before running the migration utility. You create these tables by running the `jpdk_preference_store2.sql` which is found in *`ORACLE_HOME`*`/j2ee/OC4J_WebCenter/applications/jpdk/jpdk/doc/dbPreferenceStore`.

When using a migration mode, you can also use the `-remap` and `-countries` options to specify that the data should be upgraded in the course of being migrated, that is, locale-specific preferences should be remapped appropriately.

The other options accepted by the utility are used to specify the properties of the preference stores involved in the upgrade or migration process. These options must correspond to the tags you specify in `provider.xml` to describe the preference stores. For more information about the properties you can set on a preference store, see the *PDK-Java XML Provider Definition Tag Reference* document on Portal Studio:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/apidoc/oracle/portal/provider/v2/ProviderDefinition.html`

Properties beginning with the prefix `-pref1` correspond to properties of the source preference store (in an upgrade mode this is the only preference store). For example, specifying `-pref1UseHashing true -pref1RootDirectory j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample` would set the `useHashing` and `rootDirectory` properties of a source FilePreferenceStore.

Note: If you installed the OracleAS PDK on a standalone OC4J instance, or if you downloaded the preconfigured standalone OC4J with OracleAS PDK, then the source preference store will be available in the following location:

```
ORACLE_HOME/j2ee/home/applications/jpdk/jpdk/WEB-INF/providers
/sample
```

If you installed OracleAS Portal as part of the Oracle Application Server release, then the source preference store will be available in the following location:

```
ORACLE_HOME/j2ee/OC4J_Portal/applications/jpdk/jpdk/WEB-INF
/providers/sample
```

When one of the migration basic modes is selected, properties beginning with the prefix `-pref2` correspond to properties of the target preference store. For example, specifying `-pref2User portlet_prefs -pref2Password portlet_prefs -pref2URL jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid` would set the database connection details on a target DBPreferenceStore.

Example B–5 and Example B–6 illustrate the usage of the utility.

Example B–5 PDK-Java Migration Utility Command Line, Upgrade

```
java -classpath lib/dms.jar:jdbc/lib/ojdbc14dms.jar:
      portal/jlib/pdkjava.jar:portal/jlib/ptlshare.jarjava
oracle.portal.provider.v2.preference.MigrationTool
 -mode file -remap language
 -countries GB,US -pref1UseHashing true
 -pref1RootDirectory j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample
```

Example B–6 PDK-Java Migration Utility Command Line, Migration

```
java -classpath lib/dms.jar:jdbc/lib/ojdbc14dms.jar:
      portal/jlib/pdkjava.jar:portal/jlib/ptlshare.jarjava
oracle.portal.provider.v2.preference.MigrationTool -mode filetodb -remap locale
 -countries AR,MX -pref1UseHashing true
 -pref1RootDirectory j2ee/home/applications/jpdk/jpdk/WEB-INF/providers/sample
 -pref2User portlet_prefs
 -pref2Password portlet_prefs
 -pref2URL jdbc:oracle:thin:@myserver.mydomain.com:1521:mysid
```

B.4.3 Web Clipping Repository

Web Clipping does not have a preference store as such, but it stores Web Clipping definitions and associated metadata. By default, it uses Oracle Metadata Services (MDS), which is file-based, for this purpose, but you can also configure Web Clipping to use a database. To migrate this repository for WebCenter applications, you can use the Predeployment tool in export and import mode to go from MDS to a database, or vice versa. This procedure must be done for each application as follows:

1. Run the Predeployment tool in export mode on all WebCenter applications that use the Web Clipping producer. For more information, see Section 12.4.1, "Exporting Customizations".

2. Update the producer to use a different repository. For example, a database, as described in Section 18.4.3.1, "Configuring a Web Clipping Portlet Producer to Use a Database Repository".
3. Run the Predeployment tool in import mode on all WebCenter applications that use the Web Clipping producer. For more information, see Section 12.4.2, "Importing Customizations".

Note: If Oracle Application Server Portal (OracleAS Portal) is consuming the Web Clipping producer, then you must use OracleAS Portal's export and import utilities to migrate the repository.

C

Files for WebCenter Applications

This appendix provides reference information about the files that are created and modified as you build up your WebCenter application.

See Also:

- For a complete reference for the Oracle Application Development Framework (Oracle ADF) metadata files that you create in your data model and user interface projects, see the *Oracle Application Development Framework Developer's Guide*.
- To learn more how files that are affected by major actions, see Section 11.2.2, "Developer Actions Affecting Metadata Files".

C.1 About Files

When you use Oracle WebCenter Framework to build applications and components, you will notice a number of files are created as you perform such actions as building and consuming portlets. As you work with your application, you will find it useful to know a little bit about each of these files and how they relate to your application. You can group the files affected by the Oracle WebCenter Framework into two broad categories as follows:

- Files that are common to any Oracle ADF application, such as `web.xml`. For these files, you must know what specific additions and changes are made for WebCenter applications. Those modifications are described in this appendix, but for more complete descriptions of these common files, you can see *Oracle Application Development Framework Developer's Guide*.
- Files that are unique to WebCenter applications, such as `portlet.xml`. For these files, you must know what the file is for and what it contains. These files are described completely in this appendix.

C.2 Files Overview

The files that get created and modified are closely associated with the objects that you create as part of your WebCenter application. Hence, the easiest way to discuss these files is by object as follows:

- Files Related to JPS Portlets
- Files Related to PDK-Java Portlets
- Files Related to Pages

C.3 Files Related to JPS Portlets

When you build a JPS portlet, the following files are created for you:

- Created at design time:
  - oracle-portlet.xml
  - oracle-portlet-tags.jar
  - portlet.xml
  - portlet_mode.jsp
  - portlet_name.java
  - portlet_nameBundle.jar
  - web.xml
- Created at portlet deployment time:
  - profile_name.deploy

C.3.1 oracle-portlet.xml

`oracle-portlet.xml` is an Oracle extension of `portlet.xml` to support WSRP 2.0 features. For example, a portlet's navigational parameters (a WSRP 2.0 feature) are defined in `oracle-portlet.xml`. In the future, when JSR 286 becomes available, this extension file will no longer be necessary.

C.3.1.1 oracle-portlet.xml Syntax

The top-level element of oracle-portlet.xml is `<portlet-app-extension>`:

```
<portlet-app-extension
 xsi:schemaLocation="http://xmlns.oracle.com/portlet/oracle-portlet-app"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://xmlns.oracle.com/portlet/oracle-portlet-app">
```

where all attributes should have the values shown in Example C–1.

Example C–1 oracle-portlet.xml Element Hierarchy

```
<portlet-app-extension>
 <allow-export> ... </allow-export>
 <allow-impoty> ... </allow-import>
 <portlet-extension>
  <portlet-name> ... </portlet-name>
    <navigation-parameters>
      <name> ... </name>
      <type> ... </type>
      <label> ... </label>
      <hint> ... </hint>
      <usage> ... </usage>
      <aliases> ... </aliases>
    </navigation-parameters>
    <portlet-id> ... </portlet-id>
    <allow-export> ... </allow-export>
    <allow-impoty> ... </allow-import>
 </portlet-extension>
</portlet-app-extension>
```

The child elements have the following usages:

- <portlet-app-extension> indicates the start and end of the portlet application definition.
- <portlet-extension> indicates the start and end of a portlet definition.
- <portlet-name> identifies the portlet to which the extensions that follow it apply.
- <navigation parameters> define the parameters for the previously identified portlet.
- <name> is the name of the navigation parameter. This name must be unique within the portlet.
- <type> is the type of the navigation parameter. Currently the only supported type is string.
- <label> is the label for the navigation parameter that end users see on the customization and personalization pages of the portlet.
- <hint> is currently not used.
- <usage> is currently not used.
- <aliases> is currently not used.
- <portlet-id> is the unique identifier of the portlet.
- <allow-export> is a flag that indicates whether the portlet supports the export of its customization data. This value can be `true` or `false`.
- <allow-import> is a flag that indicates whether the portlet supports the import of its customization data. This value can be `true` or `false`.

C.3.1.2 oracle-portlet.xml Sample With Navigation Parameters

Example C–2 provides a sample of `oracle-portlet.xml` with two portlets with three navigational parameters each.

Example C–2 oracle-portlet.xml Sample With Navigational Parameters

```
<portlet-extension>
  <portlet-name>ParameterForm</portlet-name>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter1</name>
    <type>xsi:string</type>
    <label xml:lang="en">First parameter</label>
    <hint xml:lang="en">First parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter2</name>
    <type>xsi:string</type>
    <label xml:lang="en">Second parameter</label>
    <hint xml:lang="en">Second parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter3</name>
    <type>xsi:string</type>
    <label xml:lang="en">Third parameter</label>
    <hint xml:lang="en">Third parameter set by portlet</hint>
```

```
    <usage/>
    <aliases/>
   </navigation-parameters>
    <portlet-id>4</portlet-id>
    <allow-export>true</allow-export>
    <allow-import>true</allow-import>
</portlet-extension>
<portlet-extension>
  <portlet-name>ReadOnlyParameterForm</portlet-name>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter1</name>
    <type>xsi:string</type>
    <label xml:lang="en">First parameter</label>
    <hint xml:lang="en">First parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter2</name>
    <type>xsi:string</type>
    <label xml:lang="en">Second parameter</label>
    <hint xml:lang="en">Second parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <navigation-parameters>
    <name>ora_wsrp_navigparam_Parameter3</name>
    <type>xsi:string</type>
    <label xml:lang="en">Third parameter</label>
    <hint xml:lang="en">Third parameter set by portlet</hint>
    <usage/>
    <aliases/>
   </navigation-parameters>
   <portlet-id>5</portlet-id>
   <allow-export>true</allow-export>
   <allow-import>true</allow-import>
</portlet-extension>
```

C.3.2 oracle-portlet-tags.jar

`oracle-portlet-tags.jar` is the Oracle implementation of the JSP tag library defined by the Java Portlet Specification.

C.3.3 portlet.xml

`portlet.xml` defines the characteristics of your JPS portlet. For complete details on `portlet.xml`, you should see the Java Portlet Specification available on the following URL:

`http://jcp.org/en/jsr/detail?id=168`

Example C–3 provides a sample fragment from a `portlet.xml` file. Note that this example does not include all of the available elements of `portlet.xml`.

Example C–3 portlet.xml Sample

```
<portlet>
   <description xml:lang="en">JSR 168 map portlet </description>
   <portlet-name>portlet1</portlet-name>
```

```
   <display-name xml:lang="en">Map Portlet</display-name>
   <portlet-class>jsrportlet.MapPortlet</portlet-class>
   <expiration-cache>0</expiration-cache>
   <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>edit</portlet-mode>
   </supports>
   <supported-locale>en</supported-locale>
   <resource-bundle>jsr.resource.MapPortletBundle</resource-bundle>
   <portlet-preferences>
      <preference>
         <name>portletTitle</name>
      </preference>
   </portlet-preferences>
   <security-role-ref>
      <role-name>viewer</role-name>
   </security-role-ref>
</portlet>
```

For JSR 168 portlets, the `portlet.xml` file contains all information related to portlets and their settings. Note that not all of these settings are used in the previous sample.

`<description>` provides a description of the portlet, which can be used to provide details to the end user.

`<portlet-name>` uniquely identifies the portlet within the portlet application.

`<display-name>` is used when presenting a list of available portlets to the user.

`<portlet-class>` contains the fully qualified class name of the class implementing the `javax.portlet.Portlet` interface or extending the `GenericPortlet` abstract class that becomes the entry point for the portlet logic. The portlet container uses this class when it invokes the portlet life cycle methods.

`<supports>` provides information about the portlet modes supported for each content type.

`<title>` is the static title of the portlet, usually displayed in the portlet decoration on the portlet window.

`<short-title>` is the title that is used on devices (such as mobile phones) that have limited display capabilities.

`<keywords>` are used by applications that offer search capabilities for their users.

`<security-role-ref>` maps a role name to a security role in `web.xml`. The list of roles in `web.xml` that the `<security-role-ref>` maps to is published to the consumer as the producer's user categories. In `web.xml`, `<security-role>` appears similar to the following:

```
<security-role>
   <description>Viewer role</description>
   <role-name>viewer</role-name>
</security-role>
```

C.3.4 portlet_mode.jsp

For each portlet mode that you choose to create for your portlet, a corresponding JSP file is created in your *`portlet_name`*`\html` directory to define that mode. For example, if you choose to have View and Edit modes for your portlet, then you will need `view.jsp` and `edit.jsp` in your *`portlet_name`*`\html` directory. For JPS portlets, you can have the following JSP files for your portlet modes:

- about.jsp
- config.jsp
- edit_defaults.jsp
- edit.jsp
- help.jsp
- preview.jsp
- print.jsp
- view.jsp

For further explanation of portlet modes, see Section 18.1.1, "Guidelines for Portlet Modes".

C.3.5 portlet_name.java

portlet_name.java is the class that acts as the entry point for the portlet logic. This class must implement the javax.portlet.Portlet interface or extend the GenericPortlet abstract class. The portlet container uses this class when it invokes the portlet life cycle methods.

C.3.6 portlet_nameBundle.jar

*portlet_name*Bundle.jar is a resource bundle class, containing translation of the strings used by the portlet.

C.3.7 web.xml

web.xml is a file common to J2EE development. For more information about web.xml, see *Oracle Application Development Framework Developer's Guide* and *Oracle Containers for J2EE Security Guide*.

C.3.8 profile_name.deploy

profile_name.deploy stores a set of deployment characteristics for a deployment profile. *profile_name*.deploy is created when you right-click web.xml and choose **Create WAR Deployment Profile**. You can use the deployment profile to deploy your portlet application to a Web or enterprise application archive, or to a portlet container.

C.4 Files Related to PDK-Java Portlets

When you build a PDK-Java portlet, the following files are created for you:

- Created at design time:
  - _default.properties
  - index.jsp
  - portlet_name_modePage.jsp
  - producer_name.properties
  - provider.xml
  - web.xml
- Created at portlet deployment time:

- profile_name.deploy

C.4.1 producer_name.properties

producer_name.properties specifies deployment details about the producer, such as the location of the provider.xml file. For example, this file is used if the registration URL to the PDK-Java samples is of the form:

http://*host*:*port*/jpdk/provider/samples

or:

http://*host*:*port*/jpdk/provider

where the service ID field contains samples. See also Section C.4.2, "_default.properties".

C.4.2 _default.properties

_default.properties specifies deployment details about the producer, such as the location of the provider.xml file. For example, this file is used if the registration URL to the PDK-Java samples is of the form:

http://*host*:*port*/jpdk/provider

Note that the producer name is not supplied here so it has to default. See also Section C.4.1, "producer_name.properties".

C.4.3 index.jsp

index.jsp serves as a convenient starting point when testing PDK-Java producers from Oracle JDeveloper. This file lists all of the producers available in the application.

C.4.4 portlet_name_modePage.jsp

For each portlet mode that you choose to create for your portlet, a corresponding JSP file is created in your \htdocs*portlet_name* directory to define that mode. For example, if you choose to have View and Edit modes for a portlet named portletOne, then you need portletOneShowPage.jsp and PortletOneEditPage.jsp in your \htdocs\portletOne directory. For PDK-Java portlets, you can have the following JSP files for your portlet modes:

- *portlet_name*AboutPage.jsp
- *portlet_name*EditDefaultsPage.jsp
- *portlet_name*EditPage.jsp
- *portlet_name*HelpPage.jsp
- *portlet_name*ShowDetailsPage.jsp
- *portlet_name*ShowPage.jsp

For further explanation of portlet modes, see Section 18.1.1, "Guidelines for Portlet Modes".

C.4.5 provider.xml

provider.xml is the definition file for your PDK-Java producer. Note that PDK-Java producers were formerly referred to as providers.

C.4.5.1 provider.xml Syntax

For more information about the elements and syntax of `provider.xml`, see the Provider Definition Extensible Markup Language (XML) Tag Reference v2 on the Oracle Technology Network.

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

C.4.5.2 provider.xml Sample

Example C–4 provides a sample `provider.xml` file.

Example C–4 provider.xml Sample

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?providerDefinition version="3.1"?>
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
 <localePersonalizationLevel>none</localePersonalizationLevel>
 <session>true</session>
 <defaultLocale>en</defaultLocale>
 <preferenceStore
  class="oracle.portal.provider.v2.preference.FilePreferenceStore">
  <name>prefStore1</name>
  <useHashing>true</useHashing>
 </preferenceStore>
 <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
  <id>1</id>
  <name>SampleRenderer</name>
  <title>SampleRenderer example</title>
  <shortTitle>SampleRenderer</shortTitle>
  <description>Example portlet rendered using the SampleRenderer</description>
  <timeout>40</timeout>
  <timeoutMessage>SampleRenderer example timed out</timeoutMessage>
  <acceptContentType>text/html</acceptContentType>
  <showEdit>true</showEdit>
  <showEditToPublic>false</showEditToPublic>
  <showEditDefault>true</showEditDefault>
  <showPreview>true</showPreview>
  <showDetails>true</showDetails>
  <hasHelp>true</hasHelp>
  <hasAbout>true</hasAbout>
  <renderer
   class="oracle.portal.sample.v2.devguide.samplerenderer.SampleRenderer"/>
   <personalizationManager
   class="oracle.portal.provider.v2.personalize.PrefStorePersonalizationManager">
   <dataClass>oracle.portal.provider.v2.personalize.
     NameValuePersonalizationObject
   </dataClass>
   </personalizationManager>
 </portlet>
</provider>
```

C.4.6 web.xml

`web.xml` is a file common to J2EE development. For more information about `web.xml`, see *Oracle Application Development Framework Developer's Guide*.

C.4.7 profile_name.deploy

profile_name.deploy stores a set of deployment characteristics for a deployment profile. *profile_name*.deploy is created when you right-click web.xml and choose **Create WAR Deployment Profile**. You can use the deployment profile to deploy your portlet application to a Web or enterprise application archive, or to a portlet container.

C.5 Files Related to Pages

When you create or modify pages, the following files are created or modified:

- Created at page design time:
  - adf-config.xml
  - adf-faces-config.xml
  - DataBindings.cpx
  - faces-config.xml
  - page_name.jspx
  - PageDef.xml
  - web.xml
- Created at application deployment time:
  - profile_name.deploy
  - mds Subdirectory
- Created upon consumption of JPS portlets
  - wsdl Subdirectory

C.5.1 adf-config.xml

adf-config.xml is used by the Oracle WebCenter Framework to configure its portlet client. The <adf-portlet-config> element in adf-config.xml configures the portlet client. Table C–1 describes the child elements of <adf-portlet-config>. The init-param names in the first column correspond to the names of the servlet init-params used when the portlet client is accessed through the Web adapter.

Table C–1 Child Elements of adf-portlet-config

Element (init-param)	Description	Default Value
parallelPoolSize (parallel.pool.size)	The number of threads to use for parallel execution of tasks.	10
parallelQueueSize (parallel.queue.size)	The size of the queue of tasks waiting for parallel execution. Tasks are rejected once the queue size is exceeded.	20
defaultTimeout (default.timeout)	The default timeout period in seconds for requests made to producers. This value is used when a timeout is not defined at the portlet or producer level.	10

Table C–1 (Cont.) Child Elements of adf-portlet-config

Element (init-param)	Description	Default Value
`minimumTimeout` (`minimum.timeout`)	The minimum timeout period in seconds for requests made to producers. This value is used to impose a lower limit on timeout periods specified by portlets or producers.	0.1
`maximumTimeout` (`maximum.timeout`)	The maximum timeout period in seconds for requests made to producers. This value is used to impose an upper limit on timeout periods specified by portlets or producers.	60
`resourceProxyPath` (`resource.proxy.path`)	The base path of the resource proxy servlet, relative to the context root of the application. Used to construct links to the resource servlet within portlet markup.	/resourceproxy
`supportedLocales` (`supported.locales`)	The set of supported locales defined using strings of the form: `language[_country[_variant]]`	Commented out by default. You should uncomment it if you must have multiple locales. See Example C–5.
`portletTechnologies` (`portlet.technologies`)	The set of portlet technologies supported by the client defined by the fully qualified names of classes that implement the PortletTechnologyConfig interface.	{o.p.c.ci.web.WebPortletTechnologyConfig, o.p.c.ci.wsrp.WSRPPortletTechnologyConfig}
`cacheSettings` (`cache.*`)	Cache configuration information. Used to enable or disable the cache, define its maximum size and impose limits on the amount of space available for different users and subscribers.	The cache is enabled and no size restrictions are imposed.

<adf-portlet-config> element Sample

Example C–5 illustrates the usage of the `<adf-portlet-config>` element.

Example C–5 <adf-portlet-config> element Sample

```
<adf-portlet-config xmlns="http://xmlns.oracle.com/adf/portlet/config">
  <supportedLocales>
    <value>en</value>
    <value>fr</value>
    <value>de</value>
    <value>es</value>
  </supportedLocales>
  <portletTechnologies>
   <value>oracle.portlet.client.containerimpl.web.
     WebPortletTechnologyConfig</value>
   <value>oracle.portlet.client.containerimpl.wsrp.
     WSRPPortletTechnologyConfig</value>
  </portletTechnologies>
  <defaultTimeout>20</defaultTimeout>
  <minimumTimeout>1</minimumTimeout>
  <maximumTimeout>60</maximumTimeout>
```

```
  <resourceProxyPath>/portletresource</resourceProxyPath>
  <cacheSettings>
    <maxSize>10000000</maxSize>
    <subscriber default="true">
      <systemLevel>
        <maxSize>5000000</maxSize>
      </systemLevel>
      <userLevel>
        <maxSize>8000000</maxSize>
      </userLevel>
    </subscriber>
  </cacheSettings>
</adf-portlet-config>
```

C.5.2 adf-faces-config.xml

`adf-faces-config.xml` is a file common to Oracle ADF applications using JSF as the view technology. For more information about it, see *Oracle Application Development Framework Developer's Guide*.

adf-faces-config.xml Sample

Example C–6 provides a sample `adf-faces-config.xml` file.

Example C–6 adf-faces-config.xml Sample

```
<?xml version="1.0" encoding="windows-1252"?>
<adf-faces-config xmlns="http://xmlns.oracle.com/adf/view/faces/config">
  <skin-family>#{skinBean.currentSkin}</skin-family>
</adf-faces-config>
```

C.5.3 DataBindings.cpx

`DataBindings.cpx` is a file common to Web applications. For more information about it, see *Oracle Application Development Framework Developer's Guide*.

C.5.4 faces-config.xml

`faces-config.xml` is a file common to JSF applications. It describes the page flow of your application. For more information about it, see *Oracle Application Development Framework Developer's Guide*.

C.5.5 page_name.jspx

`page_name``.jspx` is the JSP file for your page. Whenever you add or remove components, such as portlets or data controls from the page, this file is updated.

C.5.6 PageDef.xml

`PageDef.xml` is a file common to Oracle ADF applications. This file holds information about portlet bindings. Also, portlet parameters can be tied to page variables in this file. To learn more about how parameters are wired through `PageDef.xml`, see Section 4.5, "Contextually Linking Components".

For more information about `PageDef.xml`, see *Oracle Application Development Framework Developer's Guide*.

PageDef.xml Sample

Example C–7 provides a sample `PageDef.xml` file.

Example C–7 PageDef.xml Sample

```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
   version="10.1.3.37.97" id="app_SRFeedbackPageDef"
   Package="oracle.srdemo.view.pageDefs">
  <parameters/>
  <executables>
    <methodIterator id="findAllServiceRequestIter"
                    Binds="findAllServiceRequest.result"
                    DataControl="SRPublicFacade" RangeSize="4"
                    BeanClass="oracle.srdemo.model.entities.ServiceRequest"/>
    <variableIterator id="variables">
      <variable Name="portlet1_Param1" Type="java.lang.Object"
      DefaultValue="
      ${bindings.findAllServiceRequestIter.currentRow.dataProvider['svrId']}"/>
      <variable Name="portlet1_Param2" Type="java.lang.Object"/>
      <variable Name="portlet1_Param3" Type="java.lang.Object"/>
      <variable Name="portlet1_Param4" Type="java.lang.Object"/>
      <variable Name="portlet1_Param5" Type="java.lang.Object"/>
    </variableIterator>
    <methodIterator id="findServiceRequestByIdIter"
                    Binds="findServiceRequestById.result"
                    DataControl="SRPublicFacade" RangeSize="10"
                    BeanClass="oracle.srdemo.model.entities.ServiceRequest"/>
    <portlet id="portlet1"
      portletInstance="/oracle/adf/portlet/OmniProducer_1150310748178/
      applicationPortlets/Portlet100_eebc7f18_010b_1000_8001_82235f640cea"
      class="oracle.adf.model.portlet.binding.PortletBinding"
        xmlns="http://xmlns.oracle.com/portlet/bindings">
      <parameters>
        <parameter name="Param1" pageVariable="portlet1_Param1"/>
        <parameter name="Param2" pageVariable="portlet1_Param2"/>
        <parameter name="Param3" pageVariable="portlet1_Param3"/>
        <parameter name="Param4" pageVariable="portlet1_Param4"/>
        <parameter name="Param5" pageVariable="portlet1_Param5"/>
      </parameters>
    </portlet>
  </executables>
  <bindings>
    <methodAction id="findAllServiceRequest"
                  InstanceName="SRPublicFacade.dataProvider"
                  DataControl="SRPublicFacade"
                  MethodName="findAllServiceRequest" RequiresUpdateModel="true"
                  Action="999" IsViewObjectMethod="false"
                  ReturnName="SRPublicFacade.methodResults.
                     SRPublicFacade_dataProvider_findAllServiceRequest_result"/>
    <table id="findAllServiceRequest1" IterBinding="findAllServiceRequestIter">
      <AttrNames>
        <Item Value="assignedDate"/>
        <Item Value="problemDescription"/>
        <Item Value="requestDate"/>
        <Item Value="status"/>
        <Item Value="svrId"/>
        <Item Value="custComment"/>
        <Item Value="custCommentDate"/>
        <Item Value="custCommentContactBy"/>
```

```
        <Item Value="mgrNotes"/>
        <Item Value="mgrNotesDate"/>
      </AttrNames>
    </table>
    <methodAction id="findServiceRequestById"
                  InstanceName="SRPublicFacade.dataProvider"
                  DataControl="SRPublicFacade"
                  MethodName="findServiceRequestById" RequiresUpdateModel="true"
                  Action="999" IsViewObjectMethod="false"
                  ReturnName="SRPublicFacade.methodResults.
                     SRPublicFacade_dataProvider_findServiceRequestById_result">
      <NamedData NDName="svrIdParam"
         NDValue=
          "${bindings.findAllServiceRequestIter.currentRow.dataProvider['svrId']}"
         NDType="java.lang.Integer"/>
    </methodAction>
    <attributeValues id="svrId" IterBinding="findServiceRequestByIdIter">
      <AttrNames>
        <Item Value="svrId"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="custComment" IterBinding="findServiceRequestByIdIter">
      <AttrNames>
        <Item Value="custComment"/>
      </AttrNames>
    </attributeValues>
    <attributeValues id="mgrNotes" IterBinding="findServiceRequestByIdIter">
      <AttrNames>
        <Item Value="mgrNotes"/>
      </AttrNames>
    </attributeValues>
    <list id="ServiceRequestcustCommentContactBy"
          IterBinding="findServiceRequestByIdIter" ListOperMode="0"
          StaticList="true" NullValueFlag="1">
      <AttrNames>
        <Item Value="custCommentContactBy"/>
      </AttrNames>
      <ValueList>
        <Item Value=" "/>
        <Item Value="Phone"/>
        <Item Value="Email"/>
        <Item Value="SMS"/>
      </ValueList>
    </list>
    <methodAction id="mergeEntity" InstanceName="SRPublicFacade.dataProvider"
                  DataControl="SRPublicFacade" MethodName="mergeEntity"
                  RequiresUpdateModel="true" Action="999"
                  IsViewObjectMethod="false"
                  ReturnName="SRPublicFacade.methodResults.
                              SRPublicFacade_dataProvider_mergeEntity_result">
      <NamedData NDName="entity"
         NDValue=
         "${bindings.findServiceRequestByIdIter.currentRow.dataProvider}"
         NDType="java.lang.Object"/>
    </methodAction>
  </bindings>
</pageDefinition>
```

C.5.7 web.xml

`web.xml` is a file common to J2EE development. For more information about `web.xml`, see *Oracle Application Development Framework Developer's Guide*.

C.5.8 profile_name.deploy

`profile_name``.deploy` stores a set of deployment characteristics of a deployment profile. *`profile_name`*`.deploy` is created when you right-click `web.xml` and choose **Create WAR Deployment Profile**. You can use the deployment profile to deploy your application to a Web or enterprise application archive.

C.5.9 mds Subdirectory

When you deploy a WebCenter application, a `mds` subdirectory is created in your project directory. This subdirectory contains other subdirectories and metadata files, such as portlet customizations and personalizations.

C.5.10 wsdl Subdirectory

When you consume JPS portlets in an application, a `wsdl` subdirectory is created in the `WEB-INF` directory. The files in this subdirectory are internal files created for portlets from a WSRP producer.

C.6 Files Related to Security

When you implement or modify security for your application, the following file is created or modified:

- app-jazn-data.xml

C.6.1 app-jazn-data.xml

The `app-jazn-data.xml` file is used to facilitate the deployment of realm and policy information for your application. In your development environment (Oracle JDeveloper), `app-jazn-data.xml` is located in your application's `.adf/META-INF` directory. Postdeployment, the file is unpacked to the directory *`ORACLE_HOME`*`/j2ee/`*`oc4j_instance`*`/applications/app-name/adf/META-INF`.

The `app-jazn-data.xml` file is created when a policy is defined in the application using the authorization editor of Oracle JDeveloper. Each time a developer updates the policy for a page or component (for example, an iterator or data control), Oracle JDeveloper's embedded Oracle Containers for J2EE's (OC4J) `system-jazn-data.xml`, located in *`JDEV_HOME`*`\system\oracle.j2ee.10.1.3.xx.xx\embedded-oc4j\config`, is updated. At the same time, these changes are also propagated to file `app-jazn-data.xml`.

> **Note:** `app-jazn-data.xml` does not appear in the Applications Navigator in Oracle JDeveloper.

When you migrate security information with the JAZN Migration tool, the `app-jazn-data.xml` file can be used as the source file for the migration. See Section 12.2.4, "Migrating Security and Application Roles" for more information about migrating roles.

The elements and attributes contained in the `app-jazn-data.xml` file are a subset of OC4J's `system-jazn-data.xml` file. See *Oracle Containers for J2EE Security Guide* for more information.

app-jazn-data.xml Sample

Example C–8 provides a sample `app-jazn-data.xml` file. In this example, the view privilege on the `test.jspx` page is granted to the anonymous role, while the customize, personalize, and view privileges are granted to the user role.

> **Note:** The realm information in the `app-jazn-data.xml` file includes only roles and not the individual users that are part of that role.

Example C–8 app-jazn-data.xml Sample

```
<?xml version="1.0" encoding="UTF-8" standalone='yes'?>
<jazn-data
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/
    jazn-data-10_0.xsd"
  schema-major-version="10"
  schema-minor-version="0"
>
<!-- JAZN Realm Data -->
<jazn-realm>
    <realm>
        <name>jazn.com</name>
        <users>
        </users>
        <roles>
            <role>
                <name>users</name>
                <guid>58B213307F7811DBBF8F39184ABB7640</guid>
                <members>
                </members>
            </role>
        </roles>
    </realm>
</jazn-realm>
<!-- JACC Repository Data -->
<jacc-repository>
</jacc-repository>
<jazn-policy>
    <grant>
        <grantee>
            <principals>
                <principal>
          <class>oracle.adf.share.security.authentication.ADFRolePrincipal</class>
          <name>anyone</name>
                </principal>
            </principals>
        </grantee>
        <permissions>
            <permission>
          <class>oracle.adf.share.security.authorization.RegionPermission</class>
          <name>view.pageDefs.testPageDef</name>
                <actions>view</actions>
            </permission>
```

```
            </permissions>
        </grant>
        <grant>
            <grantee>
                <principals>
                    <principal>
                        <realm-name>jazn.com</realm-name>
                        <type>role</type>
                        <class>oracle.security.jazn.spi.xml.XMLRealmRole</class>
                        <name>users</name>
                    </principal>
                </principals>
            </grantee>
            <permissions>
                <permission>
               <class>oracle.adf.share.security.authorization.RegionPermission</class>
               <name>view.pageDefs.testPageDef</name>
               <actions>customize,personalize,view</actions>
                </permission>
            </permissions>
        </grant>
    </jazn-policy>
    <jazn-permission-classes>
    </jazn-permission-classes>
    </jazn-data>
```

D

Manually Packaging and Deploying PDK Portlet Producers

This appendix explains how to manually package your portlet producer implementation into a portable format suitable for deployment on the Oracle Application Server or another J2EE application server. It then explains how to deploy the resulting EAR file in an Oracle Application Server environment and subsequently register it with one or more OracleAS Portal instances. The sections in this appendix are as follows:

> **Note:** In general, Oracle recommends that you package and deploy your producers using the tools available in Oracle JDeveloper. However, if you find that you must package and deploy your producers manually for some reason, then refer to the tasks in this appendix.

> **Note:** Throughout this chapter, you will see references to ORACLE_HOME. ORACLE_HOME represents the full path of the Oracle home, and is used in cases where it is easy to determine which Oracle home is referenced. The following conventions are used in procedures where it is necessary to distinguish between the middle tier, OracleAS Infrastructure, or Oracle Application Server Metadata Repository Oracle home:
>
> - MID_TIER_ORACLE_HOME, represents the full path of the middle-tier Oracle home.
> - INFRA_ORACLE_HOME, represents the full path of the Oracle Application Server Infrastructure Oracle home.
> - METADATA_REP_ORACLE_HOME, represents the full path of the OracleAS Infrastructure home containing the Oracle Application Server Metadata Repository.

D.1 Introduction

Before preceding with packaging and deploying your producer, you must understand the following basic concepts:

D.1.1 WAR and EAR files

WAR and EAR files are used to deploy applications on a J2EE application server, such as Oracle Application Server. The WAR and EAR files encapsulate all of the components necessary to run an application in a single file. These files make the deployment of an application very easy and consistent, reducing the possibility of errors when moving an application from development to test, and test to production.

- **WAR files** represent a Web application and include all the components of that Web application, including Java libraries or classes, servlet definitions and parameter settings, JSP files, static HTML files, and any other required resources.
- **EAR files** represent an enterprise application.

D.1.2 Service Identifiers

PDK-Java enables you to deploy multiple producers under a single adapter servlet. The producers are identified by a service identifier. When you deploy a new producer, you must assign a service identifier to the producer and use that service identifier when creating your producer WAR file. The service name is used to look up a file called *service_id*`.properties`, which defines the characteristics of the producer, such as whether to display its test page.

For example, you can register the PDK-Java samples producer using the following URL and a service identifier of `urn:sample`:

```
http://mycompany.com/jpdk/providers
```

Alternatively, you can use a URL of the form:

```
http://mycompany.com/jpdk/providers/sample
```

where the producer name (sample) is appended to the URL of the PDK-Java samples producer. In this case, you should leave the Service Id field blank when registering the producer.

You can specify the service identifier separately in cases where multiple portals are sharing the same producer. By registering each portal with a different service identifier, you can specify the producer properties for each consumer independently.

Once your producer has been deployed, you must use the correct service identifier to register your producer with OracleAS Portal, which ensures that requests are routed to the correct producer. If the adapter servlet receives a request without a service identifier, then the request goes to the default producer.

Note: If you do not know the service identifier, then check the producer test page or contact the administrator of the producer. If you are using the Federated Portal Adapter, then the URL points to the adapter, not the producer, thus you must enter a value for this field. In this case, the service identifier would be `urn:` followed by the name of the database provider.

D.2 Packaging and Deploying Your Producers

The following sections show the steps you must perform to package and deploy a producer manually:

- Packaging Your Producer
- Deploying Your EAR File
- Testing Deployment
- Setting Deployment Properties
- Securing Your Producer
- Registering Your Producer

D.2.1 Packaging Your Producer

The steps in this section explain how to manually package a WAR file. If you are familiar with one of the various utilities for assembling WAR files, then you are free to assemble your WAR file that way.

- Preparing Your Directories
- Specifying Your Default Service
- Creating Your WAR File
- Creating Your EAR File

D.2.1.1 Preparing Your Directories

In preparation for creating your WAR file, you must perform the following steps:

1. Create a working directory where you can collect the necessary files.
2. Extract the `template.war` file from `/pdk/jpdk/v2/template.war` into your working directory. Make sure that you extract the file paths, too.
3. If your producer needs any additional JAR files, then add them to the `WEB-INF/lib` directory.
4. If your producer needs any additional Java classes not contained in a JAR file, then add them to the `WEB-INF/classes` directory. Make sure that you save the class file in a directory structure that corresponds to their Java package names.
5. Add any static HTML files, JSPs and images to your working directory. Create subdirectories as needed to organize the files. Note that the subdirectories will become part of the path necessary to access the HTML or JSP files.
6. Create a subdirectory for your producer under the producers directory.
7. Copy the `_default.properties` file to *`service_name`*`.properties` and edit it to reflect your producer's configuration.
8. Set the `definition` value in the *`provider_name`*`.properties` file that is available in the `WEB-INF/deployment` folder, as follows:

   ```
   definition=providers/provider_dir_you_created/provider.xml
   ```

9. Place your producer definition file in the subdirectory you just created.
10. Edit `_default.properties` to reflect the configuration settings of your default producer. The default producer is accessed if a service identifier is not specified in

a request. See Section D.2.1.2, "Specifying Your Default Service" for more information about this step.

11. If you use servlets to render content, then edit `WEB-INF/web.xml` to add your servlets to the list of predefined servlets. Be careful not to remove the entries for servlets required by PDK-Java.

D.2.1.2 Specifying Your Default Service

The default service is the producer that receives any request without a service name. You specify a default producer by editing the `_default.properties` file in the deployment directory of your WAR file.

Edit the definition entry to point to the producer definition file that represents your default producer. Paths should be relative to the WEB-INF directory within your WAR file, not the physical location of the file in the file system.

The `_default.properties` file looks similar to the following:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/sample/provider.xml
autoReload=true
```

D.2.1.3 Creating Your WAR File

Once you have specified the contents of your WAR file, you are ready to create the WAR file itself. To create the WAR file, perform the following steps:

1. Zip the contents of the working directory you created in Section D.2.1.1, "Preparing Your Directories", including the subdirectory paths but not the working directory path itself.
2. Rename the resulting file to give it a meaningful name and change the extension to `.war`.

D.2.1.4 Creating Your EAR File

To create the EAR file manually, perform the following steps:

1. Create another working directory for the creation of your EAR file.
2. Extract the `template.ear` file from `/pdk/jpdk/v2/template.ear` into your working directory. Make sure that you extract the file paths, too.
3. Open the `META-INF/application.xml` file that was contained in the template EAR file. It should look something like the following:

```
<?xml version "1.0">
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.
          //DTD J2EE Application 1.3//EN"
          "http://java.sun.com/j2ee/dtds/application_1_3.dtd">
<application>
    <display-name>Display Name of the Application</display-name>
    <description>Description of the application</description>
    <module>
        <web>
            <web-uri>yourwarfile.war</web-uri>
            <context-root>/</context-root>
        </web>
    </module>
</application>
```

Table D–1 describes the elements of `application.xml`.

Table D–1 Elements of application.xml

Element	Description
`<display-name>`	Is the name of the application.
`<description>`	Is a description of the application and its functions.
`<web-uri>`	Is the name of your WAR file.
`<context-root>`	Is the prefix you would like to map to your application by default (for example, `/myapp`).

4. Save `application.xml` back to the same location without changing the name of the file.
5. Copy the WAR file you created earlier into your working directory. Put it in the working directory itself, not a subdirectory.
6. Zip the contents of the working directory, including the subdirectory paths but not the working directory path itself.
7. Rename the resulting file to give it a meaningful name and change the extension to `.ear`.

D.2.2 Deploying Your EAR File

You can deploy your EAR file in any of the following ways depending upon your requirements:

- Deploying with the Grid Control Console
- Deploying Manually with dcmctl
- Deploying Manually to Standalone OC4J

D.2.2.1 Deploying with the Grid Control Console

To deploy your EAR file using the Grid Control Console, you must have PDK-Java installed in your target OC4J. To deploy your EAR file using this method, perform the following steps:

1. Start the Grid Control Console and navigate to the home page of the OC4J instance in which you configured PDK-Java (for example, `OC4J_Portal`).
2. On the Applications tab, click **Deploy EAR file**.
3. Enter the information in Table D–2.

Table D–2 Application Tab Settings

Setting	Value
J2EE Application	Browse to the location of your EAR file in your local file system
Application Name	Enter the unique name you want to associate with your producer application.
Parent Application	Use default.

4. Click **Continue**. The URL mapping for Web Modules appears. The mappings will default to the context roots specified in `application.xml` (for example,

/myapp), but you can change them to avoid clashing with the context roots of other deployed applications.

5. Click **Finish**. A summary appears with all of the information you entered.
6. Click **Deploy**.
7. Click **OK**.

   Your producer application is now deployed to your Oracle Application Server instance. You should see the newly deployed application in the list of applications for the selected OC4J instance. Once you have successfully deployed your EAR file, you must test the deployment. See Section D.2.3, "Testing Deployment".

D.2.2.2 Deploying Manually with dcmctl

To deploy your EAR file using `dcmctl`, you must have PDK-Java installed in your target OC4J.

> **Note:** Before using `dcmctl` to manage an Oracle Application Server instance, make sure you have no Grid Control Console processes managing that same instance. If multiple processes manage the same instance, then you run the risk of inconsistencies or corruption in the data used to manage the instance.

You deploy your EAR file using the command-line deployment utility `dcmctl`: (Arguments in brackets are optional.)

```
cd MID_TIER_ORACLE_HOME/dcm/bin/
./dcmctl deployApplication -f file -a app_name
   [-co comp_name] [-enableIIOP] [-rc rootcontext] [-pa parent_name]
```

where:

file is the name of the EAR or WAR file you want to deploy.

app_name is the name of the application specified by the user in the original deployment.

comp_name is the name of the OC4J instance to which the application will be deployed. The default is the home instance. (Optional)

`enable IIOP` enables the Internet Inter-Orb Protocol. (Optional)

rootcontext is the base path used in the URL to access the Web module (for example, `http://hostname:port/context root`). This option applies only to the deployment of WAR files. (Optional)

parent_name is the parent application name. The parent application contains common classes used by child applications. (Optional)

Your producer application is now deployed to your Oracle Application Server instance. Once you have successfully deployed your EAR file, you must test the deployment. See Section D.2.3, "Testing Deployment".

D.2.2.3 Deploying Manually to Standalone OC4J

To deploy your EAR file to a standalone instance of OC4J, it must be a compatible version and have PDK-Java installed. To deploy your EAR file in this method, perform the following steps:

1. If OC4J is not already running, then start it as a background process with the following commands:

   On Microsoft Windows:

   ```
   cd OC4j_HOME\j2ee\home
   start java -server -Xmx256m -jar oc4j.jar
   ```

   On UNIX/Linux (Bourne shell):

   ```
   cd OC4J_HOME/j2ee/home
   java -server -Xmx256m -jar oc4j.jar &
   ```

   where:

   `OC4J_HOME` is your OC4J installation root directory (for example, `D:\oc4j904`).

   > **Note:** The `-Xmx256m` option specifies a maximum heap size of 256 Mb for the OC4J process, which is the recommended setting. You can raise or lower this setting to suit your application. If you encounter `java.lang.OutOfMemoryError` exceptions, then you should raise this setting.

2. Deploy your EAR file using the following command:

   ```
   java -jar admin.jar ormi://localhost admin admin_password -deploy
    -deploymentName application_name -file ear_file_path
   ```

   where:

   `admin_password` is the OC4J administration password you set on installation.

   `application_name` is the unique name given to the application for administrative purposes.

   `ear_file_path` is the full path to your EAR file on the file system.

3. For each of the WAR files in your EAR file (as listed in `OC4J_HOME/j2ee/home/applications/application_name/application.xml`), bind the corresponding Web applications to a URI path on your Web site with the following command:

   ```
   java -jar admin.jar ormi://localhost admin admin_password
   -bindWebApp application_name web_app_name
   file:OC4J_HOME/j2ee/home/config/default-web-site.xml context_root
   ```

   where:

   `web_app_name` is WAR file name without the `.war` extension (for example, `jpdk`).

   `context_root` is the URI path prefix you would like to be mapped to that Web application (for example, `/myapp`).

   Your producer application is now deployed to your Oracle Application Server instance. Once you have successfully deployed your EAR file, you must test the deployment. See Section D.2.3, "Testing Deployment".

D.2.3 Testing Deployment

To test your producer deployment, you access the producer test page with a URL of the following form:

```
http://host:port/context_root/providers
```

where:

host and *port* are the host name and port number of the HTTP listener for your target OC4J instance. In an Oracle Application Server installation with Oracle Web Cache installed, `port` should be the Oracle Web Cache listener port (for example, 7777). In a standalone OC4J installation, the default HTTP port number is 8888.

context_root is the URI path prefix you mapped to the producer Web application on deployment (for example, `/myapp`) or the default one specified in `application.xml` in a manual deployment with `dcmctl`.

For example:

```
http://my.host.com:7777/newProvider/providers
```

If your `.properties` file specifies `showTestPage=true`, then you should see the familiar test page for your default producer. To view the test page for a specific producer service, you can append the service name to the URL. For example:

```
http://my.host.com:7777/newProvider/providers/myService
```

D.2.4 Setting Deployment Properties

In PDK-Java, you can specify a number of deployment properties through JNDI variables. Table D–3 provides a list of these variables with descriptions.

Table D–3 JNDI Variables for Producer Deployment

Variable	Description
`oracle/portal/provider/global/log/logLevel`	Is the logging level (0-8) used by PDK-Java and applies to all producers.
`oracle/portal/`*service_name*`/showTestPage`	Is a Boolean flag that specifies whether a producer's test page is accessible. The default value is true.
`oracle/portal/`*service_name*`/maxTimeDifference`	Is the producer's HMAC time difference.
`oracle/portal/`*service_name*`/definition`	Is the location of the producer's definition file, `provider.xml`.
`oracle/portal/`*service_name*`/autoReload`	Is a Boolean auto reload flag. The default value is true.
`oracle/portal/`*service_name*`/sharedKey`	Is the HMAC shared key. It has no default value.
`oracle/portal/`*service_name*`/rootDirectory`	Is the location for producer customizations. It has no default value.

Setting the Variables

You can set the values of the variables in Table D–3 as you would any other JNDI variables. See Section 19.2.3.2, "Setting JNDI Variable Values" for information about how to set JNDI variables.

D.2.5 Securing Your Producer

When using the PDK-Java framework in a production environment, you should secure your producers. See Chapter 10, "Securing Your WebCenter Application" for more information about securing producers.

D.2.6 Registering Your Producer

Once you have successfully deployed and verified your producer, you can register it as you would any other producer. See Section 18.10, "Registering and Viewing Your Portlet" for more information about registering your producer.

E

Administering Oracle WebCenter Wiki

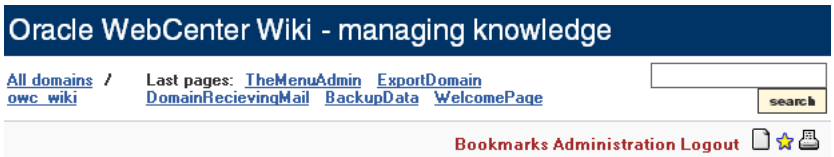
This appendix discusses basic Oracle WebCenter Wiki administration tasks. This appendix contains the following sections:

Note: For more information on administrating wiki and other related topics, refer to `http://yawiki.jzonic.org/page/show.jz?page=yawiki:TheIntroduction`.

E.1 Accessing the Administration Mode

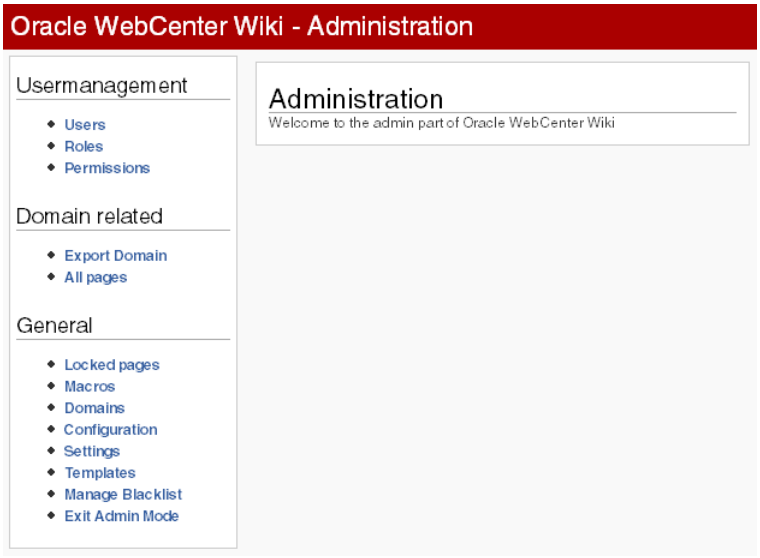
To access the Oracle WebCenter Wiki Administration mode, first log in using the administrator user name and password you created during the installation of Oracle WebCenter Wiki. In the upper right menu of each wiki page, you can click the **Administration** link to access the Administration mode. Figure E–1 shows this link.

Figure E–1 Administration Link



Once you click this link, you see the Administration mode, as shown in Figure E–2.

Figure E–2 Administration Mode



To exit the Administration mode, click **Exit Admin Mode** under General.

E.2 Domains and Menus

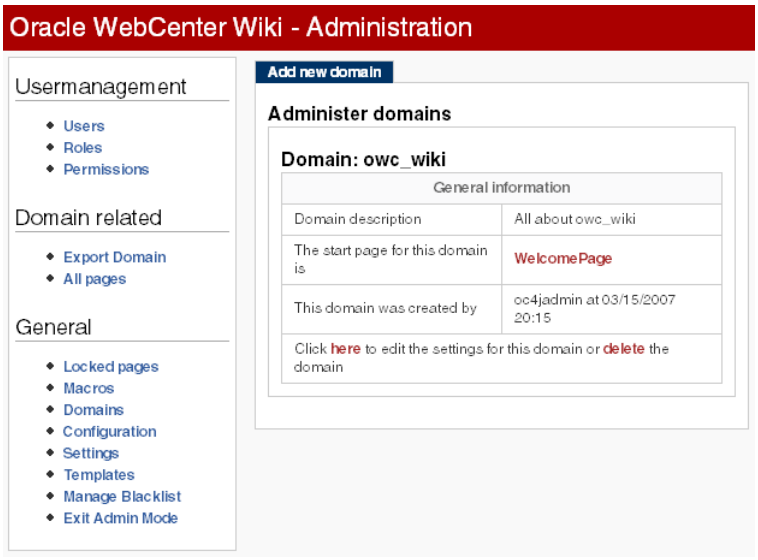
The wiki administrator organizes the wiki pages into domains, then can create menus to enable easy access to pages within each domain. This section discusses basic domain and menu administration tasks.

E.2.1 Domains

As a wiki administrator, you create, edit, or delete the domains that will contain the pages the wiki users then create, edit, or delete. In Oracle WebCenter Wiki, a domain is the highest level of categorization and all wiki pages must belong to a domain.

To create a new domain, in the Administration mode, under General, click **Domains**.

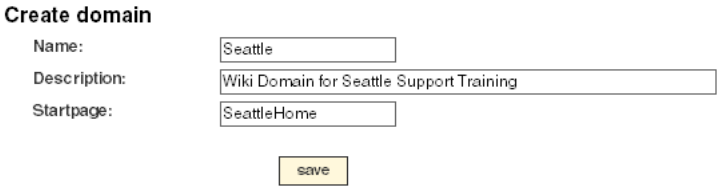
Figure E–3 Administration Page of the Wiki



To create a new domain as shown in Figure E–3, click **Add a new domain**. Enter a domain name, a description, and a name for your new start page, then click **Save**, as shown in Figure E–4.

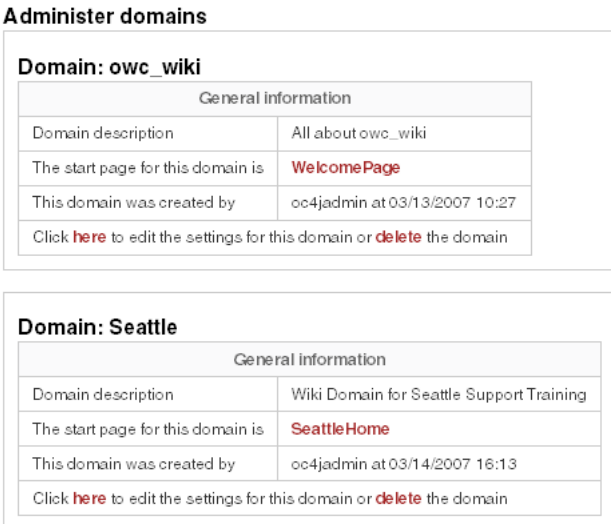
> **Tip:** When naming your page, ensure you adhere to the wiki markup standards, as explained in Table 6–2 in Chapter 6, "Integrating Oracle WebCenter Wiki".

Figure E–4 Creating a New Domain



The new domain displays on the Administer domains page as shown in Figure E–5.

Figure E–5 Administer Domains Page



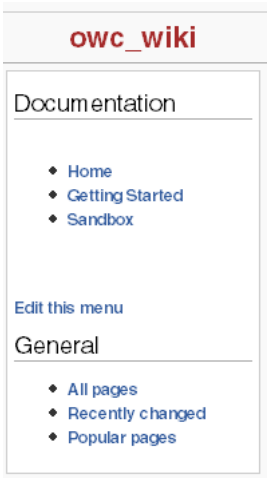
Return to the wiki home page by clicking **Exit Admin Mode** in the navigation bar. If your new start page does not automatically display, click the **All Domains** link in the upper left corner to view the existing domains. You can then click the new page in your new domain, for example My Wiki. Once you have created a new page, you can edit it using the Edit tab.

E.2.2 Menus

As an administrator, you can create a single menu for each domain, then modify them. The following is an example of how you can modify the existing menu in the domain.

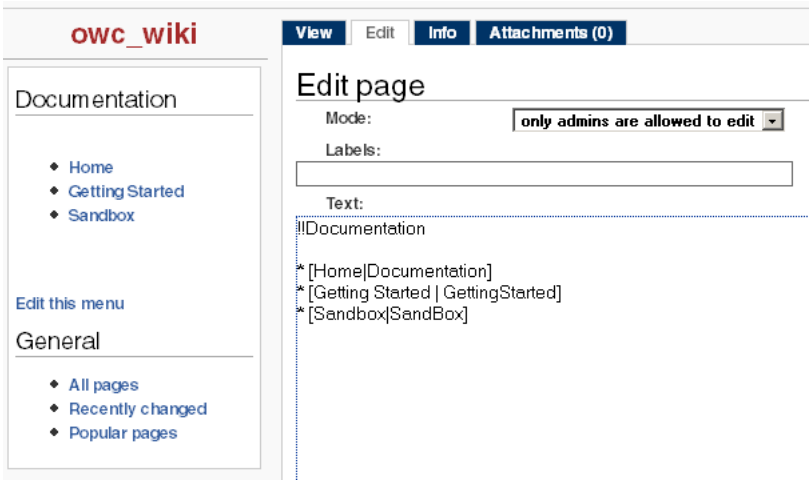
First, in the menu, click the **Edit this menu** link, as shown in Figure E–6.

Figure E–6 Edit Link on the Menu



When you click the edit link, you can edit the menu the same way you edit a page, using the Edit page text box as shown in Figure E–7.

Figure E–7 Editing the Menu



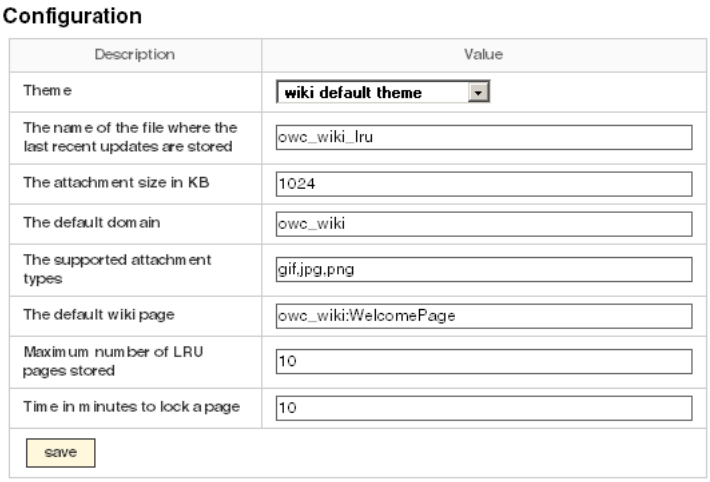
The menu consists of menu topics. The menu topics display as headers of every block of menu items. For example, in the owc_wiki domain in Oracle WebCenter Wiki, General is menu topic and All Pages is a menu item. The topics display in the order you create them. Within each topic, you can define menu items, which can be links to wiki pages or links to targets external to the wiki. When you create a menu item, you must specify a name and either a wiki page name or a URL. The name displays in the menu.

Note: After you create a menu, it is good practice to change the mode to only administrators are allowed to edit (from the **Mode** drop-down list, select this option). Although wiki automatically removes the **Edit this menu** link from the menu if the registered user is not administrator, users may accidentally edit the menu page.

E.3 Locking and Unlocking Pages

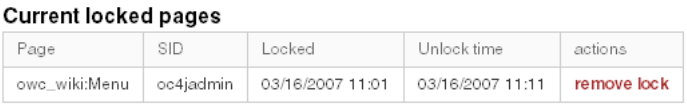
As the administrator, you can configure the locking and unlocking of wiki pages. Every time a user edits a wiki page, the page is locked for a specified time period for that particular user before other users can modify that page. As the administrator, you can configure the time period during which only the current user can modify the page. To specify the time period (in minutes), click the **Configuration** link under General in the administration menu, then enter a time in the **Time in minutes to lock a page** field. Figure E–8 shows this page.

Figure E–8 Configuration Page



You can also unlock these pages by clicking the **Locked Pages** link under the General category. Here, you can view a list of all the locked pages and unlock a page by clicking the **remove lock** link next to the desired page. Figure E–9 shows this page.

Figure E–9 Unlocking a Page



E.4 User Interface Templates

Templates enable you to set up a framework for users when they create pages. You can create new user interface templates as well as edit or delete existing ones. To access the templates, in the Administration mode, under General, click **Templates**. Use the links next to an existing template name to modify it. Or, click **Add new template** to create a new template (Figure E–10).When you name, edit, and create a template, use the wiki markup language described in Section 6.3.3, "Wiki Markup". Template names should follow the same convention as page names.

Figure E–10 Managing Templates

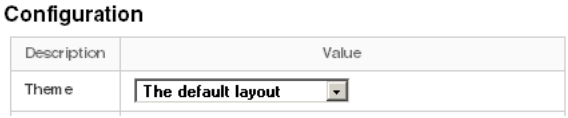
After you create a new template, users can choose to use this new template when creating a page (Figure E–11).

Figure E–11 Creating a New Page with a Template

E.5 Changing Themes of the Wiki Page

You can apply themes to change the look and feel of your wiki. To change the theme, in the Administration mode, under General, click **Configuration**. Choose a different theme from the Theme drop-down list.

Figure E–12 Choosing a Theme



Exit the Administration mode to see your changes take effect.

E.6 Monitoring Oracle WebCenter Wiki

The monitoring log file is located in *OC4J_HOME*/bin/owc_wiki.log. To change the log level, modify the jlo_logging.xml file located here:

ORACLE_HOME/j2ee/home/owc_wiki/

You can change the targets of the loggers in this file. The following targets are currently supported: trace, info, debug, warn, error, and fatal. You can also use two special targets: off (to switch of all the targets) or all (to switch on all the targets). For more information on the jLo logger, see http://jlo.jzonic.org/GettingStarted.html.

Note: You can also change the location of the log file by using the jLo handlers. For more information, see http://jlo.jzonic.org/AllHandlers.html.

For additional troubleshooting information, refer to Appendix G, "Troubleshooting WebCenter Applications".

E.7 Backing Up and Restoring Wiki Content

You can back up wiki content by using the file system. To do so, make a copy of the following folder:

OC4J_HOME/applications/*application_name*/owc_wiki/pages

You can restore the content by overwriting this folder with the back-up copy of the folder.

E.8 Exporting a Domain

Oracle WebCenter Wiki enables you to export a wiki domain as static HTML files. You can use the wiki as a simple content management system (CMS) on a local computer, then export the content and upload the HTML files to a Web server.

When you export the contents of your domain, Oracle WebCenter Wiki replaces the wiki links with the corresponding HTML filename. The wiki renames all start pages in your domain `index.html`.

> **Note:** While you can export plain HTML content, you cannot currently export attachments or images.

While your domain displays, access the Administration mode. In the navigation bar, under **Domain related**, click **Export Domain**. If you do not see this option and you are logged in as an administrator, you may need to add this specific permission. To do so, under **User Management**, click **Roles**. Then, edit the ADMIN role and add the **ExportDomain** permission. For more information, refer to Section E.10, "Permissions".

On the Export domain page, you can update the options described in Table E–1. After you have chosen the desired options, click **export domain**.

Table E–1 Export Domain Options

Option	Action
export file	Choose a template from this drop-down list. This list contains the velocity templates that are located in the owc_wiki/export directory. Oracle WebCenter Wiki uses this template when rendering each static HTML file for this domain.
css file	Choose a CSS file from this drop-down list that Oracle WebCenter Wiki will export into the same directory as the static HTML files. These CSS files are located in the same directory as the templates.
The directory	Type the directory path where you want the HTML and CSS files to be exported.

E.9 Blocking an IP Address

You can block Edit access to the wiki from a certain IP address. To do so, under General, click **Manage Blacklist** (Figure E–13). Enter the IP address you wish to block, then click **Save**. The IP address you have blocked displays in the Blacklist.

Figure E–13 Manage Blacklist

E.10 Permissions

You can set permissions for the two roles `user` and `admin` for the wiki operations from the Administration mode. In the navigation bar, under User Management, click **Roles**. Under the role you want to modify, click **edit**. For example, under the USER role, click edit to view the page in Figure E–14.

Figure E–14 User Role Permissions

E.11 Enabling Anonymous Access to Oracle WebCenter Wiki

By default, only registered users can access pages in the Oracle WebCenter Wiki. However, you can also enable anonymous access so that users can view pages without logging in. To do so, in the Administration mode, under General, click **Settings**. Next to **Is this owc_wiki installation only available for registered users**, choose **NO** from the drop-down list (Figure E–15), then click the **save** button.

Figure E–15 Anonymous Access Setting

E.12 Other Configuration Parameters

In the Administration mode, you can configure the following options:

- The name of the file where the last recent updates are stored
- The default attachment size in kilobytes (KB)
- The default domain
- The supported attachment types
- The default wiki page
- The maximum number of LRU pages stored

You can find these options and others by choosing **Configuration** under General, as shown in Figure E–16.

Figure E–16 Configuration Page

F

Node Type Definitions for Oracle WebCenter Adapters

The chapter describes node type definitions for Oracle WebCenter adapters for Microsoft SharePoint, EMC Documentum, and IBM Lotus Domino. It contains the following sections:

- Section F.1, "Node Type Definitions for the Oracle WebCenter Adapter for IBM Lotus Domino"
- Section F.2, "Node Type Definitions for the Oracle WebCenter Adapter for Microsoft SharePoint"
- Section F.3, "Node Type Definitions for the Oracle WebCenter Adapter for EMC Documentum"

F.1 Node Type Definitions for the Oracle WebCenter Adapter for IBM Lotus Domino

This section covers the following:

- Section F.1.1, "Reading"
- Section F.1.2, "Node Type Mapping"
- Section F.1.3, "Searching"
- Section F.1.4, "Authorization"

F.1.1 Reading

The Oracle WebCenter adapter for IBM Lotus Domino maps the repository workspace to a single Lotus Domino database. The adapter can read Domino documents, views, and view entries. All documents are available under the root folder, `/DOCUMENTS`. Views with view entries are visible under the folder, `/VIEWS`. However, soft deleted documents are not visible through this adapter.

A Domino database allows all documents to be accessed without a hierarchy. The Oracle WebCenter adapter for IBM Lotus Domino allows access to all documents as `mix:referenceable` nodes under a `/DOCUMENTS` root folder. To handle multiple nodes at the same level, the adapter builds an artificial hierarchy for documents in the database. The hierarchy has a fixed depth and is built based on document identifiers. A document identifier consists of eight hexadecimal characters. For example a document with note Id `NT000008F6` is visible in workspace as a node at the following path:

```
/DOCUMENTS/0/0/0/0/0/8/8F6
```

Folder node names are created from leading characters of a document Id and the final document node name is the IBM Lotus Notes identifier.

Database views are visible under the folder, `/VIEWS`. A database view may be hierarchical, and there may be views providing different hierarchies for the same set of documents. Views correspond to `ld:view` child nodes under `/VIEWS`. The name of each `ld:view` node is the name of the view. Within each view there are `ld:category` and `ld:viewEntry` child nodes. The `ld:category` nodes preserve the view's hierarchical structure, and are named as defined by the view. The leaf nodes in the hierarchy are `ld:viewEntry` nodes. An `ld:viewEntry` node is an `nt:linkedFile` node. Its `jcr:content` value is a reference property pointing to an `ld:document` under the `/DOCUMENTS` node.

F.1.2 Node Type Mapping

The Oracle WebCenter adapter for IBM Lotus Domino maps a Domino database (`.nsf`) content to a JCR workspace using subtypes of the JCR `nt:hierarchyNode` node type. The content of the workspace is organized around folders, files, and links.

F.1.2.1 IBM Lotus Notes/Domino Namespace

The Oracle WebCenter adapter for IBM Lotus Domino uses its own namespace for the repository node types. This namespace is denoted by prefix `ld`.

F.1.2.2 Documents

The artificial hierarchy of folders created by the adapter uses built-in `nt:folder` node type.

A document in the JCR workspace of the adapter represents the IBM Lotus Domino document object.

The main node types in the document mapping are as shown in Table F–1.

Table F–1 Main Node Types for Oracle WebCenter Adapter for IBM Lotus Domino

JCR Type	JCR Super Types	Description
ld:document	mix:referenceable, nt:file	Represents a Domino document in the `/DOCUMENTS` hierarchy. Properties of this node represent system properties of the IBM Lotus Domino document.
ld:content	nt:unstructured	The jcr:content child node of an `ld:document`, representing the content of the Domino document.
ld:attachment	nt:hierarchyNode	Represents file attachments of an IBM Lotus Domino document. If the document has attachments, the `ld:content` has a child node `$FILE` of type `nt:folder`, which in turn contains all the attachments represented as `ld:attachment` nodes.
ld:view	nt:folder	Represents a view in the `/VIEWS` hierarchy. A view includes links (`ld:viewEntry`) to the documents contained in the given view.
ld:category	nt:folder	Child nodes of an `ld:view` or `ld:category` node, corresponding to the hierarchy defined by the view.
ld:viewEntry	nt:linkedFile	Represents a link to the document contained in the view.

The following sample node with one attachment (`readme.txt`) illustrates the document structure in the IBM Lotus Domino mapping. This sample does not show all node properties.

```
+ 8F6 (ld:document)
 + jcr:content (ld:content)
  - Body (string)
  + $FILE (nt:folder)
   + readme.txt (ld:attachment)
    - jcr:data (binary)
```

In this simplified notation, a + sign denotes a node, while a – sign denotes a property.

Document System Properties

Documents are mapped to `ld:document` node type, which extends the `nt:file` node type as shown in Id:document Node Type.

The properties of the `ld:document` node correspond to only the system properties of an IBM Lotus Domino document. The name of the JCR property is created by prepending the Domino property name by `ld: prefix`, for example, `ld:authors` corresponds to a document's authors property. For more information on the properties defined for Domino objects, see the IBM Lotus Domino documentation.

Id:document Node Type

NodeTypeName
ld:document
Supertypes
mix:referenceable
nt:file
IsMixin
false
HasOrderableChildNodes
false
PrimaryItemName
jcr:content
PropertyDefinition
Name: ld:authors
RequiredType: String
ValueConstraints: []
DefaultValues: []
AutoCreated: false
Mandatory: false
OnParentVersion: COPY
Protected: true
Multiple: true
PropertyDefinition
Name: ld:deleted
RequiredType: Boolean
ValueConstraints: []
DefaultValues: []
AutoCreated: false
Mandatory: false
OnParentVersion: COPY
Protected: true
Multiple: false
PropertyDefinition
Name: ld:encrypted
RequiredType: Boolean
ValueConstraints: []

```
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:encryptionKeys
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: true
PropertyDefinition
   Name: ld:encryptOnSend
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:folderReferences
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:hasEmbedded
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:httpURL
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:key
   RequiredType: String
   ValueConstraints: []
```

```
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:lastAccessed
   RequiredType: Date
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:lastModified
   RequiredType: Date
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:lockHolders
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:nameOfProfile
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:newNote
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:noteID
   RequiredType: String
   ValueConstraints: []
```

```
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:notesURL
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:parentDocument
   RequiredType: Reference
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:parentDocumentUNID
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:profile
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
 PropertyDefinition
   Name: ld:response
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:responses
   RequiredType: Reference
   ValueConstraints: []
```

```
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:saveMessageOnSend
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:sentByAgent
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:signed
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:signer
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:signOnSend
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:size
   RequiredType: Long
   ValueConstraints: []
```

```
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:universalID
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:valid
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:verifier
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
```

Document Properties Managed by Applications

The content managed by a Domino application, for example, the IBM Lotus Notes email and collaboration client, appears as properties and child nodes of the `ld:document`'s `jcr:content` node. An `ld:document`'s `jcr:content` node is always of type `ld:content`. The `ld:content` node type includes residual property and node type definitions. These residual definitions allow an `ld:content` node to store the application specific data for any Domino database. For example, an email application such as IBM Lotus Notes uses attributes such as:

- Subject - for email subject
- From - for sender of the email
- Body - for body of the email

These are mapped to similarly named properties of the `jcr:content` node, that is, `jcr:content/Subject`, `jcr:content/From`, and `jcr:content/Body`.

If you want to build applications using the Domino application content, you must understand the content definition and semantics of the native Domino application using the IBM Lotus Domino database.

Note: Some properties are reserved by the application and have a dollar sign ($) prepended to the name, for example, `$MessageID` in case of Lotus Notes email databases. The adapter maps these properties with these reserved names.

The `jcr:data` property contains an XML representation of the entire document structure, and not just the binary content of any document attachments.

The `ld:content` node type definition is:

NodeTypeName
ld:content
Supertypes
nt:unstructured
IsMixin
false
HasOrderableChildNodes
false
PrimaryItemName
jcr:data
PropertyDefinition
Name: jcr:data
RequiredType: Binary
ValueConstraints: []
DefaultValues: []
AutoCreated: false
Mandatory: false
OnParentVersion: COPY
Protected: true
Multiple: false
PropertyDefinition
Name: jcr:mimeType
RequiredType: String
ValueConstraints: []
DefaultValues: []
AutoCreated: false
Mandatory: false
OnParentVersion: COPY
Protected: true
Multiple: false

Mapping of Attachments

A Domino document may have one or more file attachments which contain binary data. If a document has attachments, the document's `jcr:content` node will have a child node of type `nt:folder` called `$FILE`. This `$FILE` node will contain the attachments as child nodes of type `ld:attachment`.

Each `ld:attachment` child node corresponds to a single attachment, and is named to match the file name of the attachment, for example:

```
+ 8F6 (ld:document)
 + jcr:content (ld:content)
  - Body (string)
  + $FILE (nt:folder)
   + readme.txt (ld:attachment)
    - jcr:data (binary)
```

where + represents a node and – a property.

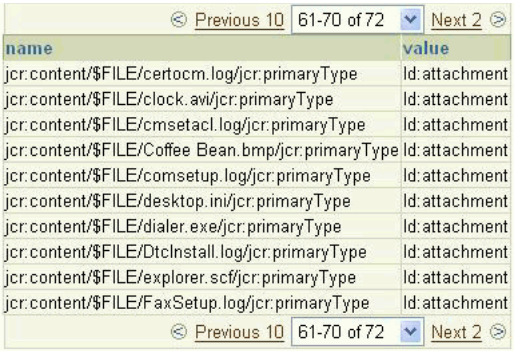
The following is the node type definition of `ld:attachment`:

```
NodeTypeName
   ld:attachment
Supertypes
   nt:hierarchyNode
IsMixin
   false
HasOrderableChildNodes
   false
PrimaryItemName
   jcr:data
PropertyDefinition
   Name: jcr:data
   RequiredType: Binary
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: jcr:mimeType
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
```

Figure F–1 and Figure F–2 show portions of sample attributes of a single email document. Figure F–2 displays the message attachments, that is, the document's `$FILE` items in the IBM Lotus Domino database.

Figure F–1 Sample Attributes of an Email Document

Figure F–2 Sample Attributes of an Email Document



Previous 10 | 61-70 of 72 | Next 2

name	value
jcr:content/$FILE/certocm.log/jcr:primaryType	ld:attachment
jcr:content/$FILE/clock.avi/jcr:primaryType	ld:attachment
jcr:content/$FILE/cmsetacl.log/jcr:primaryType	ld:attachment
jcr:content/$FILE/Coffee Bean.bmp/jcr:primaryType	ld:attachment
jcr:content/$FILE/comsetup.log/jcr:primaryType	ld:attachment
jcr:content/$FILE/desktop.ini/jcr:primaryType	ld:attachment
jcr:content/$FILE/dialer.exe/jcr:primaryType	ld:attachment
jcr:content/$FILE/DtcInstall.log/jcr:primaryType	ld:attachment
jcr:content/$FILE/explorer.scf/jcr:primaryType	ld:attachment
jcr:content/$FILE/FaxSetup.log/jcr:primaryType	ld:attachment

Previous 10 | 61-70 of 72 | Next 2

F.1.2.3 Views

Each database view corresponds to an `ld:view` node under `/VIEWS` in the JCR workspace. The `ld:view` node's properties represent the system properties of the Domino view, as described in Document System Properties.

An `ld:view` node can have `ld:category` and `ld:viewEntry` child nodes.

The `ld:category` nodes correspond to the categories in a hierarchical view, and the `ld:viewEntry` nodes are the leaves of the hierarchy, pointing to `ld:document` nodes under `/DOCUMENTS`.

The `ld:view` node type definition is:

```
NodeTypeName
   ld:view
Supertypes
   nt:folder
IsMixin
   false
HasOrderableChildNodes
   true
PrimaryItemName
   null
PropertyDefinition
   Name: ld:aliases
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:autoUpdate
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:backgroundColor
   RequiredType: Long
```

```
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:calendar
      RequiredType: Boolean
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:categorized
      RequiredType: Boolean
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:columnCount
      RequiredType: Long
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:columnNames
      RequiredType: String
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: true
   PropertyDefinition
      Name: ld:columnValues
      RequiredType: String
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: true
   PropertyDefinition
      Name: ld:conflict
      RequiredType: Boolean
```

```
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:defaultView
      RequiredType: Boolean
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:entryCount
      RequiredType: Long
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:folder
      RequiredType: Boolean
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:headerLines
      RequiredType: Long
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:hierarchical
      RequiredType: Boolean
      ValueConstraints: []
      DefaultValues: []
      AutoCreated: false
      Mandatory: false
      OnParentVersion: COPY
      Protected: true
      Multiple: false
   PropertyDefinition
      Name: ld:httpURL
      RequiredType: String
```

```
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:lastModified
   RequiredType: Date
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:lockHolders
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:modified
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:name
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:notesURL
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:private
   RequiredType: Boolean
```

```
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:prohibitDesignRefresh
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:protectReaders
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:readers
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:rowLines
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:selectionFormula
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:spacing
   RequiredType: Long
```

```
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:topLevelEntryCount
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:universalID
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:viewInheritedName
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
ChildNodeDefinition
   Name: *
   RequiredPrimaryTypes: [ld:category]
   DefaultPrimaryType none
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   SameNameSiblings: true
ChildNodeDefinition
   Name: *
   RequiredPrimaryTypes: [ld:viewEntry]
   DefaultPrimaryType none
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   SameNameSiblings: true
```

The node type definition for `ld:category` is:

```
NodeTypeName
   ld:category
Supertypes
```

```
   nt:folder
IsMixin
   false
HasOrderableChildNodes
   true
PrimaryItemName
   null
PropertyDefinition
   Name: ld:category
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:childCount
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:columnIndentLevel
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:columnValues
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:conflict
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:descendantCount
   RequiredType: Long
```

```
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:document
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:indentLevel
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:siblingCount
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:total
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:valid
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
ChildNodeDefinition
   Name: *
   RequiredPrimaryTypes: [ld:viewEntry]
```

```
   DefaultPrimaryType none
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   SameNameSiblings: true
```

The node type definition for `ld:viewEntry` is:

```
NodeTypeName
   ld:viewEntry
Supertypes
   nt:linkedFile
IsMixin
   false
HasOrderableChildNodes
   false
PrimaryItemName
   null
PropertyDefinition
   Name: ld:category
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:childCount
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:columnIndentLevel
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:columnValues
   RequiredType: String
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: true
PropertyDefinition
   Name: ld:conflict
```

```
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:descendantCount
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:document
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: false
   Multiple: false
PropertyDefinition
   Name: ld:indentLevel
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:siblingCount
   RequiredType: Long
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:total
   RequiredType: Boolean
   ValueConstraints: []
   DefaultValues: []
   AutoCreated: false
   Mandatory: false
   OnParentVersion: COPY
   Protected: true
   Multiple: false
PropertyDefinition
   Name: ld:valid
```

```
RequiredType: Boolean
ValueConstraints: []
DefaultValues: []
AutoCreated: false
Mandatory: false
OnParentVersion: COPY
Protected: true
Multiple: false
```

F.1.3 Searching

The Oracle WebCenter adapter for IBM Lotus Domino supports queries for documents stored in an IBM Lotus Notes database. It converts JCR queries into corresponding Domino queries and translates Domino search results into JCR query results. For example, `//element(*, ld:document)`, a simple XPath query, returns all documents stored in Lotus Notes database.

The Oracle WebCenter adapter for IBM Lotus Domino supports a partial mapping of the JCR query model to the Domino search using the Formula Language. Only the part of JCR query syntax, which is directly supported by the underlying repository, is implemented. The currently supported features include:

- Type constraint: `element(*, …)`
- Property constraints `(=, <, >, <=, >=, !=, jcr:like())`
- Boolean predicates: and, or, not
- Ordering specifier

Additionally, the Oracle WebCenter adapter for IBM Lotus Domino supports full-text search using the `jcr:contains()` function and Lotus Domino search syntax. When searching in a view for documents, a XPath query must use the `jcr:deref` function to create a query that applies to the documents selected by the view.

> **Note:** The search function in Oracle WebCenter adapter for IBM Lotus Domino does not support `nt:folder` nodes. This is because the IBM Lotus Domino database is a flat object store, and the object hierarchy under `/DOCUMENTS` in the adapter's JCR mapping is artificial. See Section F.1.1, "Reading" and Section F.1.2, "Node Type Mapping" for information.

Sample XPath queries

The following are sample XPath queries:

By node type:

```
/jcr:root//element(*, ld:document)
```

By document metadata:

```
/jcr:root//element(*, ld:document)[@ld:noteID = '8F6']
/jcr:root//element(*, ld:document)[@jcr:created >
xs:dateTime('2002-10-10T17:00:00.000Z')]
```

By document content:

```
/jcr:root//element(*, ld:document)[jcr:content/@Form = 'Message']
/jcr:root//element(*, ld:document)[jcr:like(jcr:content/@Subject, 'Test%')]
```

Full text search:

```
/jcr:root//element(*, ld:document)[jcr:contains(jcr:content/@jcr:data,'keyword')]
```

Search for all documents:

```
/jcr:root/DOCUMENTS//element(*, ld:document)
```

Search in a view (`$Inbox`) for a document with application property Subject containing `test`:

```
/jcr:root//VIEWS/_x0028__x0024_Inbox_x0029_
//jcr:deref(@jcr:content)[jcr:contains(@Subject,'test')]
```

F.1.4 Authorization

The Oracle WebCenter adapter for IBM Lotus Domino reads Access Control Lists (ACLs) from IBM Lotus Domino databases and converts them into JCR permissions accessible through JCR API.

F.1.4.1 Workspaces Access

Users must have at least `READER` level access to corresponding database to be able to login into the workspace.

F.1.4.2 Document read access

Users have access to documents to which they have sufficient rights. It means that either documents have no special Readers/Authors fields or users are listed in those fields directly.

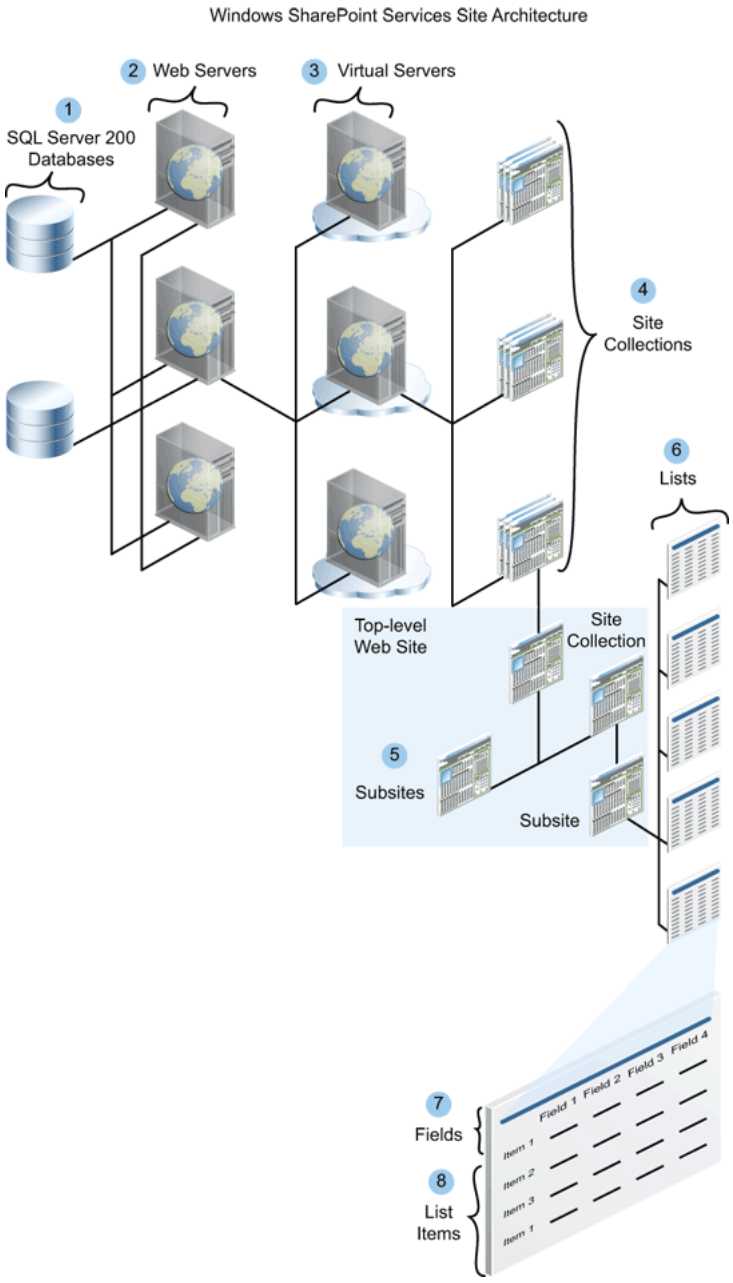
A private document is visible through the adapter if the administrator account used by the adapter connection is specified directly in the document's Readers or Authors fields. A private document will only be invisible to the adapter if this administrator account is removed from the document's Readers and Authors field.

F.2 Node Type Definitions for the Oracle WebCenter Adapter for Microsoft SharePoint

The following description of Microsoft SharePoint Services is provided to support the explanation of the node types of Oracle WebCenter adapter for Microsoft SharePoint. Refer to your Microsoft SharePoint Services documentation for a full description of SharePoint Services.

Figure F–3 illustrates the structure of SharePoint Services.

Figure F–3 SharePoint Architecture



The SQL Server 2000 databases (1) are a collection of content databases that are used to store SharePoint repository data.

The Web Servers (2), represent the collection of front-end Web servers that in turn contain one or more Virtual Servers (3).

Each Virtual Server has one or more Site Collections (4).

Each Site Collection includes any number of Site objects.

Within a Site collection there is a Top-level Web Site (5).

The Top-level Web Site corresponds to the root of the JCR hierarchy of the Oracle WebCenter adapter for Microsoft SharePoint.

A Site object can contain a collection of Subsites and a collection of Lists.

Each List object (6) contains Fields (7) and List Items (8).

The Fields provide information about the List Items. Each List Item represents a single object contained within the List.

SharePoint provides several predefined list types, for example, Document Library, Links List, Contacts List, and it also allows for creation of custom list types.

The following sections describe how these SharePoint structures are represented by the Oracle WebCenter adapter for Microsoft SharePoint:

- Section F.2.1, "SharePoint Namespace"
- Section F.2.2, "Object"
- Section F.2.3, "Collection"
- Section F.2.4, "Site"
- Section F.2.5, "Template"
- Section F.2.6, "Item"
- Section F.2.7, "Web"
- Section F.2.8, "Web Site"
- Section F.2.9, "List"
- Section F.2.10, "Field"
- Section F.2.11, "Form"
- Section F.2.12, "View"
- Section F.2.13, "List Item"
- Section F.2.14, "Folder and Files"

F.2.1 SharePoint Namespace

The Oracle WebCenter adapter for Microsoft SharePoint defines a single namespace for all node types used by SharePoint. In this documentation this namespace is denoted by the prefix `sp`.

F.2.2 Object

The `sp:Object` type is for all SharePoint objects. All node types of the Oracle WebCenter adapter for Microsoft SharePoint are extended from it. This object has the signature shown in Table F–2.

Table F–2 Object Signature

Type	Value
Namespace	sp
Local name	Object
Super type	nt:base

The `sp:Object` type does not define any child nodes or properties.

F.2.3 Collection

The `sp:Collection` type represents a collection, and is the base type of all collections. It has the signature shown in Table F–3.

Table F–3 Collection Signature

Type	Value
Namespace	sp
Local name	Collection
Super types	sp:Object

The `sp:Collection` type does not define child nodes or any properties.

F.2.3.1 Web Collection

The `sp:WebCollection` type represents collection of SharePoint web sites. The `sp:WebCollection` type has the signature shown in Table F–4.

Table F–4 Web Collection Signature

Type	Value
Namespace	sp
Local name	WebCollection
Super types	sp:Collection

The `sp:WebCollection` type defines the child node types shown in Table F–5.

Table F–5 Child Node Types of Web Collection

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:Web Collection	No	No	No	No	sp:Web	sp:Web

The `sp:WebCollection` type does not define any properties.

F.2.3.2 List Collection

The `sp:ListCollection` type represents a collection of SharePoint lists. The `sp:ListCollection` type has the signature shown in Table F–6.

Table F–6 List Collection Signature

Type	Value
Namespace	sp
Local name	ListCollection
Super types	sp:Collection

The `sp:ListCollection` type defines the child node types shown in Table F–7.

Table F–7 Child Node Types of List Collection

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:List Collection	No	No	No	No	sp:List	sp:List

The `sp:ListCollection` type does not define any properties.

F.2.3.3 Field Collection

The `sp:FieldCollection` type represents field collection of SharePoint list. The `sp:FieldCollection` type has the signature shown in Table F–8.

Table F–8 Field Collection Signature

Type	Value
Namespace	sp
Local name	FieldCollection
Super types	sp:Collection

The `sp:FieldCollection` type defines child node types shown in Table F–9.

Table F–9 Child Node Types of Field Collection

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:Field Collection	No	No	No	No	sp:Field	sp:Field
*	sp:FieldCollection	No	No	No	No	sp:Calculated Field	sp:Calculated Field
*	sp:FieldCollection	No	No	No	No	sp:Choice Field	sp:Choice Field
*	sp:FieldCollection	No	No	No	No	sp:Currency Field	sp:Currency Field
*	sp:FieldCollection	No	No	No	No	sp:DateTime Field	sp:DateTime Field
*	sp:FieldCollection	No	No	No	No	sp:MultiChoice Field	sp:MultiChoice Field
*	sp:FieldCollection	No	No	No	No	sp:MultiLine TextField	sp:MultiLine TextField
*	sp:FieldCollection	No	No	No	No	sp:Number Field	sp:Number Field
*	sp:FieldCollection	No	No	No	No	sp:Lookup Field	sp:LookupField

Table F–9 (Cont.) Child Node Types of Field Collection

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:FieldCollection	No	No	No	No	sp:RatingScale Field	sp:Rating Scale Field
*	sp:FieldCollection	No	No	No	No	sp:TextField	sp:TextField
*	sp:FieldCollection	No	No	No	No	sp:URLField	sp:URLField
*	sp:FieldCollection	No	No	No	No	sp:UserField	sp:UserField
*	sp:FieldCollection	No	No	No	No	sp:Computed Field	sp:Computed Field

The `sp:FieldCollection` type does not define any properties.

F.2.3.4 Site Template Collection

The `sp:SiteTemplateCollection` type represents collection of site's templates. The `sp:SiteTemplateCollection` type has the signature shown in Table F–10.

Table F–10 Site Template Collection Signature

Type	Value
Namespace	sp
Local name	SiteTemplateCollection
Super types	sp:Collection

The `sp:SiteTemplateCollection` type defines child node types shown in Table F–11.

Table F–11 Child Node Types of Site Template Collection

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:Template Collection	No	No	No	No	sp: SiteTemplate	sp: SiteTemplate

The `sp:SiteTemplateCollection` type does not define any properties.

F.2.3.5 Item Collection

The `sp:ItemCollection` node type represents collection of files and folders in the SharePoint document library. The `sp:ItemCollection` type has the signature shown in Table F–12.

Table F–12 Item Collection Signature

Type	Value
Namespace	sp
Local name	ItemCollection
Super types	sp:Collection

The `sp:ItemCollection` type defines child node types shown in Table F–13.

Table F–13 Child Node Types of Item Collection

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:ItemCollection	No	No	No	No	sp:ListItem	sp:ListItem

The `sp:ItemCollection` type does not define any properties.

F.2.3.6 File Collection

The `sp:FileCollection` type represents collection of files and folders in the SharePoint document library. The `sp:FileCollection` type has the signature shown in Table F–14.

Table F–14 File Collection Signature

Type	Value
Namespace	sp
Local name	FileCollection
Super types	sp:Collection

The `sp:FileCollection` type defines child node types shown in Table F–15.

Table F–15 Child Node Types of File Collection

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:FileCollection	No	No	No	No	sp:ListItem	sp:ListItem
*	sp:FileCollection	No	No	No	No	sp:Folder	sp:Folder

The `sp:FileCollection` type does not define any properties.

F.2.4 Site

The `sp:Site` type represents site and is the root node of JCR. The `sp:Site` type has the signature shown in Table F–16.

Table F–16 Site Signature

Type	Value
Namespace	sp
Local name	Site
Super types	sp:Object

The `sp:Site` type defines child node types shown in Table F–17.

Table F–17 Child Nodes Types of Site

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
sp:RootWeb	sp:Site	Yes	Yes	No	Yes	sp:WebSite	sp:WebSite
sp:Templates	sp:Site	Yes	Yes	No	Yes	sp:TemplateCollection	sp:TemplateCollection

The `sp:Site` type defines the properties shown in Table F–18.

Table F–18 Site Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
allowUnsafeUpdates	Boolean	Yes	False	No	No	Yes	Contains a flag that specifies whether to allow updates to the database as a result of a GET request or without requiring a security validation.
allWebs	Reference	No	-	No	Yes	Yes	Contains references on all Web sites available within the site collection, including the top-level site and nested subsites.
catchAccessDeniedException	Boolean	Yes	False	No	No	Yes	Contains a flag that specifies whether to handle access denied exceptions and require user authentication.
certificationDate	Date	No	-	No	No	Yes	Contains date on which usage was checked for the site collection.

Table F–18 (Cont.) Site Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
deadWebNotificationCount	Long	No	-	No	No	Yes	Contains the number of Web site notifications within the site collection that are out of use.
globalPermMask	String	No	-	No	No	Yes	Contains the rights for the permissions mask that is used globally on the virtual server.
hostName	String	No	-	No	No	Yes	Contains the name of the server that hosts the site collection.
iisAllowsAnonymous	Boolean	Yes	False	No	No	Yes	Contains a flag that indicates whether anonymous access is enabled in Internet Information Services (IIS).
lastContentModifiedDate	Date	No	-	No	No	Yes	Contains the date and time in Coordinated Universal Time (UTC) when the content of the site was last changed.
lastSecurityModifiedDate	Date	No	-	No	No	Yes	Contains the date and time in Coordinated Universal Time (UTC) when security on the site was last changed.
lockIssue	String	No	-	No	No	Yes	Contains the explanation for locking a site collection.
owner	String	No	-	No	No	Yes	Contains the owner of the site collection.
port	Long	No	-	No	No	Yes	Contains the port number used for input and output on the virtual server containing the site collection.
portalName	String	No	-	No	No	Yes	Contains the name of a portal.

Table F–18 (Cont.) Site Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
portalUrl	String	No	-	No	No	Yes	Contains the URL to a portal.
protocol	String	No	-	No	No	Yes	Contains the protocol, HTTP or HTTPS that is used by the server.
quota	String	No	-	No	No	Yes	Contains a quota for the site collection.
readLocked	Boolean	Yes	False	No	No	Yes	Contains a flag that indicates whether the site collection is locked and unavailable for Read access.
secondaryContact	String	No	-	No	No	Yes	Contains the secondary contact used for the site collection.
serverReletiveUrl	String	No	-	No	No	Yes	Contains the server-relative URL of the top-level Web site in the site collection.
url	String	No	-	Yes	No	Yes	Contains the URL of the top-level Web site in the site collection, including host name, port number, and path.
usageInfo	String	No	-	No	No	Yes	Contains information about site usage, including bandwidth, storage, and number of visits to the site collection.
warningNotificationSent	Boolean	Yes	False	No	No	Yes	Contains a flag that indicates whether a warning notification was sent.

Table F–18 (Cont.) Site Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
writeLocked	Boolean	Yes	False	No	No	Yes	Contains a flag that indicates whether the site collection is locked and unavailabe for Write access.

F.2.5 Template

This section describes the following node types:

- List Template
- Site Template

F.2.5.1 List Template

The `sp:ListTemplate` type represents templates of lists and contains the signature shown in Table F–19.

Table F–19 List Template Signature

Type	Value
Namespace	sp
Local name	ListTemplate
Super types	sp:Object

The `sp:ListTemplate` type does not define any child node types. The `sp:ListTemplate` type defines the properties shown in Table F–20.

Table F–20 List Template Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
baseType	String	No	-	Yes	No	Yes	Contains the base type for the list definition or list template.
catalog	Boolean	Yes	-	Yes	No	Yes	Specifies whether the list definition is for a site gallery, a list gallery, or a Web Part gallery.
description	String	No	-	Yes	No	Yes	Contains the description of the list definition or list template that is displayed in the user interface.
displayName	String	No	-	Yes	No	Yes	Contains the display name for the list definition or list template.

Table F–20 (Cont.) List Template Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
dontSaveInTemplate	Boolean	Yes	-	Yes	No	Yes	Contains a flag that specifies whether to exclude the content of the list when the list is saved as a custom list template or when the site to which the list belongs is saved as a custom site template through the user interface.
documentTemplate	String	Yes	-	Yes	No	Yes	Corresponds the type of the default creating document.
hidden	Boolean	Yes	False	Yes	No	Yes	Contains a flag that specifies whether the list definition or list template is hidden from Web site users and does not appear as an option on the Documents and Lists page.
hiddenList	Boolean	Yes	False	Yes	No	Yes	Contains a flag that specifies whether a list created from the list definition is hidden.
image	String	No	-	Yes	No	Yes	Contains the relative path, based on the server, for the image used to represent the list definition or list template.
isCustomTemplate	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the `SPListTemplate` object represents a list template.
name	String	No	-	Yes	No	Yes	Contains the display name for the list definition or list template.

Table F–20 (Cont.) List Template Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
onQuickLaunch	Boolean	Yes	False	Yes	No	Yes	Contains a flag that specifies whether any list created with the list definition or list template is displayed on the Quick Launch bar.
rootWebOnly	Boolean	Yes	-	Yes	No	Yes	Specifies whether the list created from the definition exists only in the root Web site of a site collection.
securityBits	String	Yes	-	Yes	No	Yes	Defines read, write, and schema design security. Each digit in the string corresponds to the three security settings contained in the List of Lists database table. This attribute does not apply to document libraries.
type	String	No	-	Yes	No	Yes	Contains the type of the list definition or list template.
unique	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the list definition or list template is used to create only one list in the site collection.

F.2.5.2 Site Template

The `sp:SiteTemplate` type represents templates of the site and contains the signature shown in Table F–21.

Table F–21 Site Template Signature

Type	Value
Namespace	sp
Local name	SiteTemplate
Super types	sp:Object

The `sp:SiteTemplate` type does not define any child node types. The `sp:SiteTemplate` type defines the properties shown in Table F–22.

Table F–22 Site Template Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
id	String	Yes	-	Yes	No	Yes	Contains the unique template identifier.
description	String	No	-	Yes	No	Yes	Contains the description for the site definition or site template.
imageUrl	String	No	-	Yes	No	Yes	Contains the URL for the image that is used to represent the site definition or site template in the user interface.
isCustomTemplate	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the `SPSiteTemplate` object is a site template.
isHidden	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the site definition or site template is displayed as an option for creating a site collection on the Template Selection page.
isUnique	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the site created from the site definition or site template inherits from its parent Web site.
name	String	No	-	Yes	No	Yes	Contains the name of the site definition or site template.
title	String	No	-	Yes	No	Yes	Contains the display name for the site definition or site template.

F.2.6 Item

The `sp:Item` node is a base JCR type of a List and Web SharePoint object. It extends List and Web node types of SharePoint. The `sp:Item` type has the signature shown in Table F–23.

Table F–23 Item Signature

Type	Value
Namespace	sp
Local name	Item
Super types	sp:Object

The `sp:Item` type does not define child node types. It contains the properties shown in Table F–24.

Table F–24 Item Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
title	String	No	-	Yes	No	Yes	Contains title of item.
description	String	No	-	No	No	Yes	Contains description of item.
author	String	Yes	Current user	Yes	No	Yes	Contains name of user, who created item.
lastModified	Date	Yes	Current date	Yes	No	Yes	Contains the date when the item was last modified.
lastModifiedForceRecrawl	Date	Yes	Current date	Yes	No	Yes	Contains date of last modification, which was forced by recrawl.
validSecurityInfo	Boolean	No	True	Yes	No	Yes	Contains a flag that shows that security information is valid.
inheritedSecurity	Boolean	No	False	Yes	No	Yes	Contains a flag that shows that security is equals to parent item.
allowAnonymousAccess	Boolean	No	False	Yes	No	Yes	Contains a flag that shows that anonymous user has access to item.
anonymousViewListItems	Boolean	No	False	Yes	No	Yes	Contains a flag that shows that anonymous user can view list items.

Table F–24 (Cont.) Item Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
permission	String	No	-	No	No	Yes	Contains permissions as string.

F.2.7 Web

The `sp:Web` is a JCR type. It provides information about the Web SharePoint object. The `sp:Web` type has the signature shown in Table F–25.

Table F–25 Web Signature

Type	Value
Namespace	sp
Local name	Web
Super types	sp:Item

The `sp:Web` type defines the child node types shown in Table F–26.

Table F–26 Child Node Types of Web

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
sp:List	sp:Web	Yes	Yes	No	Yes	sp:List Collection	sp:List Collection

The `sp:Web` type defines the properties shown in Table F–27.

Table F–27 Web Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
url	String	No	-	Yes	No	Yes	Contains URL of site.
webId	String	Yes	Generated	Yes	No	Yes	Contains SharePoint internal Id of site.
language	Long	No	-	Yes	No	Yes	Contains language of site.
noIndex	String	No	-	No	No	Yes	Contains no index text.
externalSecurity	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that the site supports external security.
categoryId	String	No	-	No	No	Yes	Contains Id of site's category.
categoryName	String	No	-	No	No	Yes	Contains name of site's category.

Table F–27 (Cont.) Web Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
categoryIdPath	String	No	-	No	No	Yes	Contains path of id category of site.
isBucketWeb	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that site is bucket web.
usedInAutocat	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that site is used in autocat.
categoryBucketId	Boolean	No	-	No	No	Yes	Contains Id of category's bucket.

The `sp:Web` type provides the default values for properties of extended node types, as shown in Table F–28.

Table F–28 Default Values for Properties of Extended Node Types

Name	Super Node Type	Default Value
inheritedSecurity	sp:Item	False
allowAnonymousAccess	sp:Item	False
anonymousViewListItems	sp:Item	True

F.2.8 Web Site

The `sp:WebSite` type provides information about a Web SharePoint object. The `sp:WebSite` type has the signature shown in Table F–29.

Table F–29 Web Site Signature

Type	Value
Namespace	sp
Local name	Web Site
Super types	sp:Web

The `sp:WebSite` type defines the child node types shown in Table F–30.

Table F–30 Child Nodes Types of Web Site

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
sp:Webs	sp:Web	yes	yes	no	yes	sp:Web Collection	sp:Web Collection

The `sp:WebSite` type does not define any properties.

F.2.9 List

The `sp:List` type provides information about a SharePoint list object. The `sp:List` type has the signature shown in Table F–31.

Table F–31 List Signature

Type	Value
Namespace	sp
Local name	List
Super types	sp:Item nt:hierarchyNode

The `sp:List` type defines the child node types shown in Table F–32.

Table F–32 Child Node Types of List

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
sp:Items	sp:List	Yes	Yes	No	Yes	sp:Item Collection	sp:Item Collection
sp:RegionalSettings	sp:List	Yes	Yes	No	Yes	sp:Regional Settings	sp:Regional Settings
sp:Fields	sp:List	Yes	Yes	No	Yes	sp:Field Collection	sp:Field Collection
sp:Forms	sp:List	Yes	Yes	No	Yes	sp:Form Collection	sp:Form Collection
sp:Views	sp:List	Yes	Yes	No	Yes	sp:View Collection	sp:View Collection

The `sp:List` type defines the properties shown in Table F–33.

Table F–33 List Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
allowDeletion	Boolean	Yes	True	Yes	No	Yes	Contains a flag that shows *do allow deletion* items from list.
allowMultiResponses	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows *do allow multi responses*.

Table F–33 (Cont.) List Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
anonymousPermMask	Long	No	-	Yes	No	Yes	Contains type of anonymous perm mask.
author	Long	No	-	Yes	No	Yes	Contains Id of the user who created the list.
baseTemplate	String	No	-	Yes	No	Yes	Contains base template of list.
baseType	String	No	-	Yes	No	Yes	Contains a list of the base type.
created	Date	No	-	Yes	No	Yes	Contains the date when the list was created.
defaultView	Reference	No	-	Yes	No	Yes	Contains the reference on default view for the list.
defaultViewUrl	String	No	-	No	No	Yes	Contains default URL to view the list.
description	String	No	-	No	No	Yes	Contains description of the list.
direction	String	No	-	Yes	No	Yes	Contains direction of the list.
enableAttachments	Boolean	Yes	True	Yes	No	Yes	Contains a flag that shows that attachment is enabled.
enableModeration	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that moderation is enabled.
enableVersioning	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that versioning is enabled.
eventSinkAssembly	String	No	-	No	No	Yes	Contains *strong* name of file in the Global Assembly Cache.
eventSinkClass	String	No	-	No	No	Yes	Contains class name of the event handler.
eventSinkData	String	No	-	No	No	Yes	Contains arbitrary string that the event handler uses.

Table F–33 (Cont.) List Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
excludeFromTemplate	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the content of the list is included when the site to which the list belongs is saved as a custom site template.
hidden	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that the list is hidden.
imageUrl	String	No	-	Yes	No	Yes	Contains URL of the image.
itemCount	Long	No	-	Yes	No	Yes	Contains count of list items.
lastItemDeletedDate	Date	No	-	Yes	No	Yes	Contains date of the last deleted item.
lastItemModifiedDate	Date	No	-	Yes	No	Yes	Contains date and time that an item, field, or property of the list was last modified.
multipleDataList	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that list supports multiple data.
onQuickLaunch	Boolean	Yes	False	Yes	No	Yes	Contains a flag that specifies whether the list appears on the Quick Launch area of the home page.
ordered	Boolean	Yes	False	Yes	No	Yes	Contains a flag that shows that the list is *ordered*.
permissions	String	No	-	No	No	Yes	Contains the collection of permission objects, which represents all the permissions for the list.
propertiesXml	String	No	-	No	No	Yes	Contains a fragment in Collaborative Application Markup Language (CAML) that specifies property values for the list.

Table F–33 (Cont.) List Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
readSecurity	Long	No	-	No	No	Yes	Contains type of read security.
rootFolder	String	Yes	Parent/title	Yes	No	Yes	Contains folder of list.
schemaXml	String	No	-	No	No	Yes	
showUser	Boolean	Yes	True	Yes	No	Yes	Contains a flag that indicates that users should be shown.
title	String	No	-	Yes	No	Yes	Contains the displayed title for the list.
version	Long	Yes	0	Yes	No	Yes	Contains the version number of the list.
writeSecurity	Long	Yes	1	Yes	No	Yes	Contains write security type of list.

The `sp:List` type provides the default values for properties of extended node types, as shown in Table F–34.

Table F–34 Default Values for Properties of Extended Node Types

Name	Super Node Type	Default Value
inheritedSecurity	sp:Item	True
allowAnonymousAccess	sp:Item	False
anonymousViewListItems	sp:Item	False

F.2.10 Field

The `sp:Field` type provides information about field of SharePoint list object. This is the base type of all fields. The `sp:Field` type has the signature shown in Table F–35.

Table F–35 Field Signature

Type	Value
Namespace	sp
Local name	Field
Super types	sp:Object

The `sp:Field` type does not define child node types. It contains the properties shown in Table F–36.

Table F–36 Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
colName	String	Yes	-	Yes	No	Yes	Contains name of the column.
readOnly	Boolean	Yes	True	Yes	No	Yes	Contains a read-only flag.
hidden	Boolean	Yes	-	Yes	No	Yes	Contains a hidden flag.
type	String	Yes	-	Yes	No	Yes	Contains type of the field.
name	String	Yes	-	Yes	No	Yes	Contains name of the field.
displayName	String	Yes	-	Yes	No	Yes	Contains display name of the field.
fromBaseType	Boolean	Yes	-	Yes	No	Yes	Contains a flag that shows origin of the field.
required	Boolean	Yes	True	Yes	No	Yes	Contains a flag that determines whether the field requires values.
canToggleHidden	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the column can be hidden through the user interface.
defaultFormula	String	No	-	No	No	Yes	Contains default formula for calculation fields.
defaulValue	String	No	-	No	No	Yes	Contains default value of the field.
authoringInfo	String	No	-	No	No	Yes	Contains authoring information of the field.
description	String	No	-	No	No	Yes	Contains description of the field.
direction	String	No	-	No	No	Yes	Contains direction of the field.
displaySize	String	No	-	No	No	Yes	Contains display size of the field.
filterable	Boolean	No	False	Yes	No	Yes	Contains a filterable flag.
filterableBySchema	Boolean	No	False	Yes	No	Yes	Contains a filterable by schema flag.
filterableNoRecurence	Boolean	No	False	Yes	No	Yes	Contains a filterable no recurrence flag.

Table F–36 (Cont.) Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
filterableNoRecurrence BySchem	Boolean	No	False	Yes	No	Yes	Contains a filterable no recurrence by schema flag.
imeMode	String	No	-	No	No	Yes	Contains Input Method Editor (IME) mode.
reorderable	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether values in the field can be reordered.
sealed	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether other fields can be derived from the field.
sortable	Boolean	Yes	False	Yes	No	Yes	Contains a flag that determines whether the field can be sorted.
sortableBy Schema	Boolean	No	False	Yes	No	Yes	Contains a sortable by schema flag.

F.2.10.1 Field Type

The SharePoint repository supports the types of fields shown in Table F–37.

Table F–37 Field Types

Type	Description	JSR Type
Attachments	Contains attachments	
Boolean	Contains Boolean values that are stored in the database as 1 or 0.	
Calculated	Contains calculated values.	sp:CalculatedField
Choice	Specifies a predetermined set of values that can be used to enter data into the field.	sp:ChoiceField
Computed	Specifies an abstract field type that depends on other fields for its content and definition.	sp:ComputedField
Counter	Contains an integer used for internal Id fields.	sp:NubmerField
CrossProjectLink	Specifies a link between projects in a Meetings Workspace site.	sp:UrlField
Currency	Contains currency values formatted based on a specific locale.	sp:CurrencyField
DateTime	Contains date and time values.	sp:DateTimeField
Error	Contains errors.	
File	Contains files.	

Table F–37 (Cont.) Field Types

Type	Description	JSR Type
GridChoice	Specifies a Choice field for a data sheet.	sp:ChoiceField
Guid	Contains GUIDs.	
Integer	Contains positive or negative integer values.	sp:NumberField
Invalid	Not used.	
Lookup	Contains references to values in other lists.	sp:LookupField
MaxItems	Contains the maximum number of items.	
ModStat	Specifies Content Approval status.	sp:MultiChoiceField
MultiChoice	Contains multiple values per list item.	sp:MultiChoiceField
Note	Specifies a field that can contain multiple lines of text.	sp:MultiLineTextField
Number	Contains floating point numbers.	sp:NumberField
Recurrence	Specifies a field used in calendars for recurring events and, like computed fields, an abstract field type that depends on other fields for its content and definition.	
Text	Contains a single line of text.	sp:TextField
Threading	Specifies a field that is used in the creation and display of threaded Web discussions.	
URL	Contains hyperlinks.	sp:UrlField
User	Specifies users of a SharePoint site.	sp:UserField

F.2.10.2 Computed Field

The `sp:ComputedField` type represents a field that depends on other fields in a SharePoint list. The `sp:ComputedField` type contains signature shown in Table F–38.

Table F–38 Computed Field Signature

Type	Value
Namespace	Sp
Local name	ComputedField
Super types	sp:Field

The `sp:ComputedField` type does not define child node types. It contains the properties shown in Table F–39.

Table F–39 Computed Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
FieldRefs	String	No	-	Yes	Yes	Yes	Contains a collection of field references used to define computed fields.

Table F–39 (Cont.) Computed Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
DisplayPattern	String	No	-	Yes	No	Yes	Is used to define how the field is rendered.

F.2.10.3 Calculated Field

The `sp:CalculatedField` type represents a calculated field in a SharePoint list. The `sp:CalculatedField` type contains the signature shown in Table F–40.

Table F–40 Calculated Field Signature

Type	Value
Namespace	Sp
Local name	CalculatedField
Super types	sp:Field

The `sp:CalculatedField` type does not define child node types. It defines properties shown in Table F–41.

Table F–41 Calculated Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
currency	String	No	-	No	No	Yes	Contains the currency format that is used in the field.
dateFormat	String	No	-	No	No	Yes	Contains the date and time formatting that is used in the field.
displayFormat	String	No	-	No	No	Yes	Contains the number format that is displayed in the field.
formula	String	No	-	Yes	No	Yes	Contains formula used for calculation in the field.
outputType	String	No	-	Yes	No	Yes	Contains type of field.
showAsPersantage	Boolean	No	False	Yes	No	Yes	Contains a flag that determines whether values in the field are displayed as percentages.

F.2.10.4 Choice Field

The `sp:ChoiceField` type represents a choice field in a SharePoint list. It contains the signature shown in Table F–42.

Table F–42 Choice Field Signature

Type	Value
Namespace	Sp
Local name	ChoiseField
Super types	sp:Field

The `sp:ChoiceField` type does not define child node types. It contains the properties shown in Table F–43.

Table F–43 Choice Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
editFormat	String	No	-	Yes	No	Yes	Contains format whether to display the choice field as radio buttons or as a dropdown list.
soratable	Boolean	Yes	False	Yes	No	Yes	Contains a flag that indicates whether the field can be sorted.

F.2.10.5 Number Field

The `sp:NumberField` type represents a number field in a SharePoint list. The `sp:NumberField` type contains the signature shown in Table F–44.

Table F–44 Number Field Signature

Type	Value
Namespace	sp
Local name	NumberField
Super types	sp:Field

The `sp:NumberField` type does not define any child node types. It defines the properties shown in Table F–45.

Table F–45 Number Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
displayFormat	String	No	-	Yes	No	Yes	Contains the number of decimal places to use when displaying a field.
maximumValue	Double	No	-	Yes	No	Yes	Contains maximum value for the field.
minimumValue	Double	No	-	Yes	No	Yes	Contains minimum value of field.

Table F–45 (Cont.) Number Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
showAsPercentage	Boolean	No	False	Yes	No	Yes	Contains a flag that specifies whether the field is rendered as a percentage.

F.2.10.6 Currency Field

The `sp:CurrencyField` type represents a currency field in a SharePoint list. The `sp:CurrencyField` type contains the signature shown in Table F–46.

Table F–46 Currency Field Signature

Type	Value
Namespace	Sp
Local name	CurrencyField
Super types	sp:NumberField

The `sp:CurrencyField` type does not define any child node types. It defines the properties shown in Table F–47.

Table F–47 Currency Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
currency	String	No	-	Yes	No	Yes	Contains currency used in the field.

F.2.10.7 Date Time Field

The `sp:DateTimeField` type represents a date and time field in a SharePoint list. It contains the signature shown in Table F–48.

Table F–48 Date Time Field Signature

Type	Value
Namespace	Sp
Local name	DateTimeField
Super types	sp:Field

The `sp:DateTimeField` type does not define child node types. It contains the properties shown in Table F–49.

Table F–49 Date Time Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
calendarType	String	No	-	Yes	No	Yes	Contains the type of calendar that is used to display the field.

Table F–49 (Cont.) Date Time Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
displayFormat	String	No	-	Yes	No	Yes	Contains the type of date and time format that is used in the field.

F.2.10.8 Lookup Field

The `sp:LookupField` type represents a lookup field in a SharePoint list. The `sp:LookupField` type contains the signature shown in Table F–50.

Table F–50 Lookup Field Signature

Type	Value
Namespace	Sp
Local name	LookupField
Super types	sp:Field

The `sp:LookupField` type does not define child node types. It defines the properties shown in Table F–51.

Table F–51 Lookup Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
lookupField	String	No	-	Yes	No	Yes	Contains the field from a non-local list to which this field is a lookup.
lookupList	String	No	-	Yes	No	Yes	Contains the GUID of the list to use for the lookup.

F.2.10.9 Multi Choice Field

The `sp:MultiChoiseField` type represents a field that can contain multiple values. The `sp:MultiChoiseField` type contains the signature shown in Table F–52.

Table F–52 Multi Choice Field Signature

Type	Value
Namespace	Sp
Local name	MultiChoiseField
Super types	sp:Field

The `sp:MultiChoiseField` type does not define child node types. It defines properties shown in Table F–53.

Table F–53 Multi Choice Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
choices	String	No	-	Yes	Yes	Yes	Contains the choices used in the multichoice field.
fillInChoise	Boolean	No	False	Yes	No	Yes	Contains a flag that determines whether a text box for typing an alternative value is provided for the multichoice field.
sortable	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether the multichoice field can be sorted.

F.2.10.10 Multi Line Text Field

The `sp:MultiListTextField` type represents a text field that can contain multiple lines. The `sp:MultiListTextField` type contains the signature shown in Table F–54.

Table F–54 Multi Line Text Field

Type	Value
Namespace	Sp
Local name	MultiLineTextField
Super types	sp:Field

The `sp:MultiListTextField` type does not define child node types. It defines the properties shown in Table F–55.

Table F–55 Multi Line Text Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
allowHyperlink	Boolean	No	False	Yes	No	Yes	Contains a flag that specifies whether hyperlinks can be used in the field.
filterable	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether the multiline text field can be filtered.
numberOfLines	Long	No	False	Yes	No	Yes	Contains the number of lines to display in the field.

Table F–55 (Cont.) Multi Line Text Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
richText	Boolean	No	False	Yes	No	Yes	Contains a flag that specifies whether rich text formatting can be used in the field.
sortable	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether the multiline text field can be sorted.

F.2.10.11 Rating Scale Field

The `sp:RatingScaleField` type represents a rating scale field used in surveys. The `sp:RatingScaleField` type contains the signature shown in Table F–56.

Table F–56 Rating Scale Field Signature

Type	Value
Namespace	Sp
Local name	RatingScaleField
Super types	sp:MultiChoiceField

The `sp:RatingScaleField` type does not define child node types. It defines the properties shown in Table F–57.

Table F–57 Rating Scale Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
filterable	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether the rating-scale field can be filtered.
gridEndNumber	Long	No	-	Yes	No	Yes	Contains the end number for the rating scale.
gridNAOptionText	String	No	-	Yes	No	Yes	Contains the text to display for the not applicable (N/A) option.
gridStartNumber	Long	No	-	Yes	No	Yes	Contains the start number for the rating scale.
gridTextRangeAverage	String	No	-	Yes	No	Yes	Contains the text to display for average ratings.
gridRangeHigh	String	No	-	Yes	No	Yes	Contains the text to display for high ratings.

Table F–57 (Cont.) Rating Scale Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
gridRangeLow	String	No	-	Yes	No	Yes	Contains the text to display for low ratings.
sortable	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether the rating-scale field can be sorted.

F.2.10.12 Text Field

The `sp:TextField` type represents a rating scale field used in surveys. The `sp:TextField` type contains the signature shown in Table F–58.

Table F–58 Text Field Signature

Type	Value
Namespace	Sp
Local name	TextField
Super types	sp:Field

The `sp:TextField` type does not define child node types. It defines properties shown in Table F–59.

Table F–59 Text Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
maxLength	Long	No	255	Yes	No	Yes	Contains the maximum number of characters that can be typed in the field.

F.2.10.13 URL Field

The `sp:UrlField` type represents a rating scale field used in surveys. The `sp:UrlField` type contains the signature shown in Table F–60.

Table F–60 URL Field Signature

Type	Value
Namespace	Sp
Local name	UrlField
Super types	sp:Field

The `sp:UrlField` type does not define child node types. It defines the properties shown in Table F–61.

Table F–61 URL Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
displayFormat	String	No	-	Yes	No	Yes	Contains value that determines whether the field is displayed as an image or as a hyperlink.
filterable	Boolean	No	False	Yes	No	Yes	Contains a flag that indicates whether the URL field can be filtered.

F.2.10.14 User Field

The `sp:UserField` type represents a user field. The `sp:UserField` type contains the signature shown in Table F–62.

Table F–62 User Field Signature

Type	Value
Namespace	Sp
Local name	UserField
Super types	sp:LookupField

The `sp:UserField` type does not define child node types. It defines the properties shown in Table F–63.

Table F–63 User Field Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
allowDisplay	Boolean	No	True	Yes	No	Yes	Contains a flag that specifies whether to display the name of the user who created or last modified the field.
presence	Boolean	No	False	Yes	No	Yes	Contains a flag that specifies whether presence is enabled on the field to display user names and e-mail addresses.

F.2.11 Form

The `sp:Form` type represents a list item to create, display, or edit a form, in a list. It contains the signature shown in Table F–64.

Table F–64 Form Signature

Type	Value
Namespace	sp
Local name	Form
Super types	sp:Object

The `sp:Form` type does not define child node types. It defines the properties shown in Table F–65.

Table F–65 Form Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
schemaXml	String	No	-	Yes	No	Yes	Contains the schema in CAML that defines the form.
type	String	No	-	Yes	No	Yes	Contains the type of the form.
url	String	No	-	Yes	No	Yes	Gets the site-relative URL of the form.

F.2.12 View

The `sp:View` type represents a view of the data contained in a list on a SharePoint site. The `sp:View` type contains the signature shown in Table F–66.

Table F–66 View Signature

Type	Value
Namespace	sp
Local name	View
Super types	sp:Object

The `sp:View` type defines the child node types shown in Table F–67.

Table F–67 Child Node Types of View

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
sp:Field	sp:View	Yes	Yes	No	Yes	sp:Field Collection	sp:Field Collection

The `sp:View` type defines the properties shown in Table F–68.

Table F–68 View Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
aggregations	String	No	-	No	No	Yes	Contains field references for one or more aggregate or total columns used in a view.
aggregationStatus	String	No	-	No	No	Yes	Contains a string that specifies whether aggregate or total columns are used in the view.
baseViewID	String	No	-	Yes	No	Yes	Contains the Id of the base view for the view.
defaultView	Boolean	Yes	False	Yes	No	Yes	Contains the value that specifies whether the view is the default view.
editorModified	Boolean	Yes	False	Yes	No	Yes	Contains the value that indicates whether the view was modified by using an HTML editor.
formats	String	No	-	Yes	Yes	Yes	Contains definitions for column and row formatting used in a datasheet view.
groupByFooter	String	No	-	Yes	No	Yes	Contains the definition of the Group By footer that is used in the view.
groupByHeader	String	No	-	Yes	No	Yes	Contains the definition of the Group By header used in the view.
hidden	Boolean	Yes	False	Yes	No	Yes	Contains the hidden flag.
htmlSchemaXml	String	No	-	Yes	No	Yes	Contains the entire schema for the view.
moderationType	String	No	-	Yes	No	Yes	Contains the moderation type for the view.
name	String	No	-	Yes	No	Yes	Contains the GUID that identifies the view.

Table F–68 (Cont.) View Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
orderedView	Boolean	Yes	False	Yes	No	Yes	Contains the value that indicates whether users can reorder items through the user interface.
paged	Boolean	Yes	False	Yes	No	Yes	Contains the value that specifies whether the list supports displaying more items by adding more pages.
personalView	Boolean	Yes	False	Yes	No	Yes	Contains the value that indicates whether the view is personalized.
query	String	No	-	Yes	No	Yes	Contains an XML string representing the query for the view.
readOnlyView	Boolean	Yes	False	Yes	No	Yes	Contains the value that indicates whether the view is read-only.
recurrenceRowset	Boolean	Yes	False	Yes	No	Yes	Contains the value that specifies whether the view supports recurrence rowsets.
rowLimit	Long	No	-	Yes	No	Yes	Contains the maximum number of items to return in a view.
rowLimitExceeded	String	No	-	Yes	No	Yes	Contains the alternate rendering when the rowLimit is exceeded.
schemaXml	String	No	-	Yes	No	Yes	Contains the schema that defines the view.
scope	String	No	-	No	No	Yes	Contains the recursive scope for the view of a document library.
styleId	String	No	-	Yes	No	Yes	Contains the Id of the view style for the view.

Table F–68 (Cont.) View Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
threaded	Boolean	Yes	False	Yes	No	Yes	Contains the value that indicates whether the view is threaded.
title	String	No	-	Yes	No	Yes	Contains the display name for the view.
toolbar	String	No	-	Yes	No	Yes	Contains the toolbar for the view.
toolbarType	String	No	-	Yes	No	Yes	Contains the type of the toolbar for the view.
type	String	No	-	Yes	No	Yes	Contains the type of the view.
url	String	No	-	Yes	No	Yes	Contains the URL of the main page that includes the view.
viewBody	String	No	-	Yes	No	Yes	Contains the body of the view.
viewEmpty	String	No	-	Yes	No	Yes	Is displayed when the query returns no list data for the view.
viewFooter	String	No	-	Yes	No	Yes	Contains the footer area for the view.
viewHeader	String	No	-	Yes	No	Yes	Contains the header area for the view.

F.2.13 List Item

The `sp:ListItem` is a JSP type of SharePoint's `SPListItem` class. It represents an item, or row in a list. It contains the signature shown in Table F–69.

Table F–69 List Item Signature

Type	Value
Namespace	sp
Local name	ListItem
Super types	sp:Object nt:hierarchyNode

The `sp:ListItem` type defines child node types shown in Table F–70.

Table F–70 Child Node Types of List Item

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
sp:Attachment	sp:ListItem	No	No	No	Yes	sp:Attachment	sp:Attachment
sp:ListItem	sp:ListItem	No	No	No	Yes	sp:ListItem	sp:ListItem

The `sp:ListItem` type does not define common properties. All properties are built automatically for each type of list item.

F.2.14 Folder and Files

This section describes the following types of JCR nodes:

- Folder
- File

F.2.14.1 Folder

The `sp:Folder` type represents the folder in the SharePoint document library. The `sp:Folder` type contains the signature shown in Table F–71.

Table F–71 Folder Signature

Type	Value
Namespace	sp
Local name	Folder
Super types	sp:ListItem

The `sp:Folder` type defines child node types shown in Table F–72.

Table F–72 Child Node Types of Folder

Name	Declared Node Type	Auto-created	Mandatory	Allows Same Name Siblings	Protected	Default Primary Node Type	Required Primary Types
*	sp:Folder	No	No	No	No	sp:ListItem	sp:ListItem
*	sp:Folder	No	No	No	No	sp:Folder	sp:Folder
*	sp:Folder	No	No	No	No	sp:FileItem	sp:FileItem
*	sp:Folder	No	No	No	No	sp:Attachment	sp:Attachment

The `sp:Folder` type defines the properties shown in Table F–73.

Table F–73 Folder Properties

Name	Required Type	Auto-Created	Default Value	Mandatory	Multiple	Protected	Description
itemCount	Long	Yes	-	Yes	No	Yes	Contains the count of files and folders in the folder.

F.2.14.2 File

The `sp:File` type represents the file in the SharePoint document library. The `sp:File` type contains the signature shown in Table F–74.

Table F–74 File Signature

Type	Value
Namespace	sp
Local name	File
Super types	sp:ListItem nt:file

The `sp:File` type does not define any child node types and properties.

F.2.14.3 Attachment

The `sp:Attachment` type represents the attachment of list item. The `sp:Attachment` type contains the signature shown in Table F–75.

Table F–75 Attachment Signature

Type	Value
Namespace	sp
Local name	Attachment
Super types	sp:Object nt:file

The `sp:Attachment` type does not define child node types and properties.

F.3 Node Type Definitions for the Oracle WebCenter Adapter for EMC Documentum

This section describes node types for the Oracle WebCenter adapter for EMC Documentum. It contains the following subsections:

- Section F.3.1, "Documentum Namespace"
- Section F.3.2, "Node Type Mapping"
- Section F.3.3, "Hierarchy Mapping"
- Section F.3.4, "Virtual Documents Mapping"
- Section F.3.5, "Accessing Permissions Mapping"

- Section F.3.6, "Caching for Oracle WebCenter Adapter for EMC Documentum"

F.3.1 Documentum Namespace

The Oracle WebCenter adapter for EMC Documentum uses its own namespace for the repository node types. This namespace is denoted by prefix `dctm` in the following descriptions.

F.3.2 Node Type Mapping

All of the registered Documentum types are available through the JCR adapter by means of a simple type mapping mechanism. The Oracle WebCenter adapter for EMC Documentum uses the prefix `dctm` as a default. If the default is used already, the rule `dctm(n)` namespace will be used, where `n = integer`.

Table F–76 lists the main node types that show how the document hierarchy is mapped to types, that is, the content mapping classes to map Documentum objects to JCR node types.

Table F–76 Content Mapping Classes to Map Documentum Objects to JCR Node Types

JCR Type	Description
dctm:dmr_content	Represents a binary data of document content.
dctm:dm_document	Base class for all Documentum documents.
dctm:dm_folder	Represent a list of folders and documents. (Super class for cabinet)
dctm:dm_root	Represents an root folder or set of cabinets, depends of configuration.

All JCR node types have one or several super types. Table F–77 lists Documentum node types and its super types.

Table F–77 Documentum Node Types and Super Types

JCR Types	JCR Super Types
dctm:dm_acl	dctm:dm_base, mix:referenceable
dctm:dm_activity	dctm:dm_sysobject, dctm:dm_base, mix:referenceable
dctm:dm_aggr_domain	dctm:dm_domain, dctm:dm_base, mix:referenceable
dctm:dm_alias_set	dctm:dm_base, mix:referenceable
dctm:dm_assembly	dctm:dm_base, mix:referenceable
dctm:dm_audittrail	dctm:dm_base, mix:referenceable
dctm:dm_audittrail_acl	dctm:dm_audittrail, mix:referenceable
dctm:dm_audittrail_group	dctm:dm_audittrail, mix:referenceable
dctm:dm_base	mix:referenceable, mix:referenceable
dctm:dm_blobstore	dctm:dm_store, mix:referenceable
dctm:dm_builtin_expr	dctm:dm_expression, mix:referenceable
dctm:dm_business_pro	dctm:dm_sysobject, mix:referenceable
dctm:dm_cabinet	dctm:dm_folder, mix:referenceable
dctm:dm_cond_expr	dctm:dm_func_expr, mix:referenceable

Table F–77 (Cont.) Documentum Node Types and Super Types

JCR Types	JCR Super Types
dctm:dm_cond_id_expr	dctm:dm_cond_expr, mix:referenceable
dctm:dm_dd_info	dctm:dm_base, mix:referenceable
dctm:dm_display_config	dctm:dm_base, mix:referenceable
dctm:dm_docbaseid_map	dctm:dm_base, mix:referenceable
dctm:dm_decision	dctm:dm_sysobject, mix:referenceable
dctm:dm_distributedstore	dctm:dm_store, mix:referenceable
dctm:dm_document	nt:file, dctm:dm_sysobject, mix:referenceable, rep:AccessControllable
dctm:dm_domain	dctm:dm_base, mix:referenceable
dctm:dm_aggr_domain	dctm:dm_domain, mix:referenceable
dctm:dm_dump_record	dctm:dm_base, mix:referenceable
dctm:dm_expression	dctm:dm_base, mix:referenceable
dctm:dm_builtin_expr	dctm:dm_expression, mix:referenceable
dctm:dm_func_expr	dctm:dm_expression, mix:referenceable
dctm:dm_literal_expr	dctm:dm_expression, mix:referenceable
dctm:dm_format	dctm:dm_base, mix:referenceable
dctm:dm_fulltext_index	dctm:dm_base, mix:referenceable
dctm:dm_group	dctm:dm_base, rep:Group mix:referenceable
dctm:dm_id_range	dctm:dm_base, mix:referenceable
dctm:dm_extern_file	dctm:dm_extern_store, mix:referenceable
dctm:dm_extern_free	dctm:dm_extern_store, mix:referenceable
dctm:dm_extern_store	dctm:dm_store, mix:referenceable
dctm:dm_extern_url	dctm:dm_extern_store, mix:referenceable
dctm:dm_federation	dctm:dm_sysobject, mix:referenceable
dctm:dm_filestore	dctm:dm_store, mix:referenceable
dctm:dm_folder	nt:folder, dctm:dm_sysobject, mix:referenceable, rep:AccessControllable
dctm:dm_foreign_key	dctm:dm_relation_type, mix:referenceable
dctm:dm_format_ preferences	dctm:dm_document, mix:referenceable
Dctm:dm_sysobject	dctm:dm_base, mix:referenceable

F.3.3 Hierarchy Mapping

The Oracle WebCenter adapter for EMC Documentum presents the content hierarchically, in a similar way to the Documentum Web Console. The Documentum workspace root, `dctm:dm_root` contains list of administrative folders and cabinets, `dctm:dm_cabinet`. Each cabinet contains a hierarchy of system objects.

F.3.4 Virtual Documents Mapping

The JCR adapter supports Documentum virtual documents. A virtual document appears in the JCR hierarchy as a document node with child document nodes corresponding to the associated documents or virtual documents. In addition, the child document nodes also appear as document nodes in the parent folder that contains the virtual document. The `dctm:r_is_virtual_doc` property indicates if a document is a virtual document.

F.3.5 Accessing Permissions Mapping

The Documentum access control permissions are mapped to JCR permissions as shown in Table F–78.

Table F–78 Permission Mapping

Documentum Permission	Description	JCR Permission
Browse	Read and browse folders	Read
Read	Read and browse folders and content	Read
Relate	Make references for documents	Read
Version	View document history	Read
Write	Modify a document	Read, Write
Delete	Read, modify, or delete	Write, Remove

F.3.6 Caching for Oracle WebCenter Adapter for EMC Documentum

The Oracle WebCenter adapter for EMC Documentum caches information retrieved from the Documentum repository. It periodically checks for content updates, such as modified, created, or deleted Documentum objects as well as user and permission updates. The frequency with which this cache is updated can be set by configuring the Refresh Interval adapter parameter. See Table 5–15 for more information.

G

Troubleshooting WebCenter Applications

This appendix is intended to help users diagnose problems they encounter in building and deploying their WebCenter applications and their portlets. This appendix covers troubleshooting WebCenter applications and portlets. It contains the following topics:

- Section G.1, "Problems and Solutions"
- Section G.2, "Diagnosing WebCenter Applications (Logging)"
- Section G.3, "Need More Help?"

G.1 Problems and Solutions

This section describes common problems and solutions that may arise when working with Oracle WebCenter Framework. It is divided into the falling areas:

- Section G.1.1, "Troubleshooting Your WebCenter Application"
- Section G.1.2, "Troubleshooting Generic Portlet Problems"
- Section G.1.3, "Troubleshooting JPS Portlets"
- Section G.1.4, "Troubleshooting PDK-Java Portlets"
- Section G.1.5, "Troubleshooting Portlets Built from Oracle ADF Faces Applications"
- Section G.1.6, "Troubleshooting OmniPortlet Problems"
- Section G.1.7, "Troubleshooting Application Life Cycle Issues"

G.1.1 Troubleshooting Your WebCenter Application

This section describes problems and solutions for WebCenter applications.

G.1.1.1 Credentials MBean Not Showing Up in Oracle Enterprise Manager

The Credentials MBean, used for updating secure credentials in a deployed application, does not show up when you try to access it using Application Server Control Console.

Problem 1

The application is not running.

Solution 1

Start the application.

Problem 2

The application's `web.xml` file does not include the Oracle Application Development Framework (Oracle ADF) Authentication Servlet.

Solution 2

Ensure that the application's `web.xml` includes the Oracle ADF Authentication Servlet. See Section 12.5, "Updating Credentials in a Deployed Application" for information about how to do this.

Problem 3

The Oracle ADF Authentication Servlet was not loaded on startup.

Solution 3

Ensure that the Authentication Servlet is loaded on startup. See Section 12.5, "Updating Credentials in a Deployed Application" for information about how to do this

Problem 4

The `oracle.extapp.runtime.jar` file may not be available in your application classpath.

Solution 4

Ensure that the oracle.extapp.runtime.jar file is available in your application classpath.

G.1.1.2 Large WebCenter Application Fails With Various Errors

You are running a large WebCenter application (multiple pages with multiple components on each page) and the application is failing with various errors.

Problem

The default memory size setting for some project templates is too small for large WebCenter applications. This issue can cause you application to fail with various errors.

Solution

Increase the `MaxPermSize` of your project to accommodate the larger WebCenter application as follows:

1. Go to **Project Properties** for the project.
2. Choose Run/Debug in the panel on the left.
3. Click the **Edit** button for **Run Configurations**.
4. In **Java Options**, enter the following:

   ```
   -XX:MaxPermSize=256m
   ```

5. Click **OK**.
6. Click **OK**.

G.1.1.3 Error While Deploying a WebCenter Application to a Preseeded Standalone OC4J

When you deploy your WebCenter application to a preseeded standalone OC4J, an error occurs indicating insufficient memory.

Problem

When you deploy your WebCenter application to a preseeded standalone OC4J, the following error occurs:

```
2007-03-05 17:17:47.140 INTERNAL_ERROR oracle.adf.share.config.ADFConfigImpl null
date time java.lang.OutOfMemoryError: PermGen space
date time java.lang.OutOfMemoryError: PermGen space
date time java.lang.OutOfMemoryError: PermGen space
date time java.lang.OutOfMemoryError: PermGen space
date time java.lang.OutOfMemoryError: PermGen space
date time java.lang.OutOfMemoryError: PermGen space
```

Before running the Predeployment tool, the permanent memory size used by the preseeded standalone OC4J needs to be increased, as a number of classes are loaded during the predeployment stage.

Solution

Ensure that you include the JVM argument to increase the permanent memory size. The argument string should include `-XX:MaxPermSize=256m` or higher to increase the permanent memory used for classloading, as shown in the following examples:

On Microsoft Windows:

```
cd OC4j_HOME\j2ee\home
start java -server -XX:MAXPermSize=512M -Xmx256m -jar oc4j.jar
```

On UNIX/Linux (Bourne shell):

```
cd OC4J_HOME/j2ee/home
java -server -XX:MAXPermSize=512M -Xmx256m -jar oc4j.jar &
```

G.1.2 Troubleshooting Generic Portlet Problems

This section describes common problems and solutions for portlets.

G.1.2.1 Portlet Appears Twice in Component Palette

At design time in Oracle JDeveloper, you see the same portlets listed twice under the same producer in the Component Palette.

Problem

Two or more producers were given the same name when they were registered. Producers given the same name display only once on the Component Palette. So, if you register two producers and give them each the name *MyProducer*, then the name *MyProducer* displays only once on the Component Palette, and both producers' portlets are listed under the one instance. In addition to the duplication problem, this makes it difficult to determine which portlets come from which producer.

Solution

Edit the producer, and give it a unique name.

G.1.2.2 Customizations Missing for Duplicate Application

When you deploy the same application twice in the same OC4J instance under different names with different URLs for the same producer, the portlet customizations appear for only one of the applications and not the other.

Problem

Because the same application is deployed twice, both applications use producer and URL connections with the same names. Consequently, if the producer URLs are different between the two applications, then only one producer's customizations are available because `connections.xml` is loaded globally, not for each application. Therefore, whichever application is loaded first is the one whose connection entries are used. After that, the connections are pooled globally in memory and the second application to be accessed will use the first application's connection mappings. Thus, the second application's customizations are inaccessible because its producer URL is different from the first.

Solution

If you plan to deploy the same application multiple times to the same OC4J instance, then you must ensure that each application deployment uses the same connection mappings.

G.1.2.3 Error Accessing the Update login information Link on a Portlet

When you click the Update login information link on an external application portlet, you get the following error:

```
"javax.faces.el.MethodNotFoundException: processExtAppsCredentailEvent"
```

Problem

The Update login information link can be accessed only if you have created a Credential Provisioning page to store user credentials. Currently, there is no Credential Provisioning page within the project containing the external application portlet.

Solution

Create a Credential Provisioning page within the project containing the page with the portlet from an external application enabled producer. See Section 10.7.2.1, "Adding a Credential Provisioning Page" for the steps to be performed.

G.1.2.4 Portlet Producers Not Accessible

Your Oracle Application Server instance is up and running, but producers are not accessible.

Problem

If you chose the WebCenter Only option when installing, then your portlet producers are not running on the same HTTP port as the Oracle Application Server hosting your applications.

Solution

You can easily find the port on which your portlets are running by accessing the Welcome page of your Oracle Application Server:

```
http://myhost:port
```

Click the portlet links in the top right area of the page.

G.1.2.5 Error in Credential Provisioning Page Displayed Using the Update Login Information Link

When you click the Update login information link on an external application portlet, the Credential Provisioning page is displayed with the following error:

```
1. External application ID was found to be null while rendering the Credential
Provisioning page!
2. Cause: The credential page was probably run standalone or invoked from a
portlet added from a producer that must be but has not been associated with an
external application.
3. Action: The credential page must be invoked from the link in portlets added
from producers associated with the external application. Running the credential
page standalone is not supported.
```

Problem

You registered a producer that required external application authentication, but did not define and associate an external application with this producer. You then added a portlet from this producer on the page, which at run time, displayed a link to provision credentials. The error with the null external application ID implies the producer has not been associated with the external application to provision credentials.

Solution

For any producer requiring external application authentication, ensure that you associate the producer with the external application.

G.1.2.6 Error in Displaying Portlet at Run Time

At run time, one of the portlets on the page displays the following error:

```
Unable to get portlet response (Internal Error) for portlet binding portlet_name_1
```

And on checking the log file, this error has the following cause listed:

```
oracle.adf.extapp.exception.ExtAppNotFoundException: The external application with
ID /oracle/adf/externalApps/extApp243128437261430784.xml cannot be found.
```

Problem

An external application was accidentally or intentionally deleted but the producer it was associated with was not deregistered.

Solution

If the external application was deleted by accident, then create the relevant external application again and associate the producer with it.

If you intentionally deleted an external application, then make sure that you deregister the producer with which the application was associated. See Section 4.3.1.6, "Deregistering a Portlet Producer" for the steps to be performed.

G.1.2.7 Portlet Error When Page URLs Vary

When you access a portlet page from different URLs within the same session, OC4J loses the session and consequently you get errors accessing the portlet thereafter.

Problem

After accessing a page from different URLs, you receive a portlet unavailable error. For example, suppose that you first access a portlet page through a fully qualified URL, such as `http://hostname.domain:port`. Later, during the same session, you use a shortcut URL that omits the domain or port. Upon this second access, you receive portlet unavailable errors because the session has been lost.

Solution

Ensure that you specify URLs for portlet pages consistently. You should either fully qualify them all of the time or shortcut them (subtract domain and port) all of the time. You cannot mix and match the URL formats.

G.1.3 Troubleshooting JPS Portlets

This section describes common problems and solutions for JPS (JSR 168) portlets.

G.1.3.1 WS-Security SAML Verification by Producer Fails

When a consuming WebCenter application tries to assert security with WS-Security to the producer using a SAML token, the verification of the assertion fails because the system clocks of the two computers are not synchronized.

Problem

The system clock on the computer running the producer is ahead of the system clock on the computer running the WebCenter application trying to consume the producer. This situation results in the following:

- A `Portlet unavailable` message appearing in the portlet markup.
- The following error message appearing in the J2EE log (`log.xml`):

```
<MSG_TEXT>Assertion NOT_BEFORE_CONDITION invalid : Mon Nov 06 13:15:04 PST
2006arrival time : Mon Nov 06 13:14:59 PST 2006</MSG_TEXT>
```

The verification fails because the `SAMLToken` indicates the time when the message was generated and the assertion is not valid before that time.

Solution

Ensure that the clocks are synchronized or that the clock of the producer is behind the consuming WebCenter application's clock. Refer to your operating system documentation to determine the best way to check and set the system clock. Note that, if your producer and consumer are both deployed to the same Oracle Application Server instance, then they are running off the same clock and this problem will not occur.

G.1.3.2 Portlet Unavailable for Producer with WS Security

You successfully registered your producer with WS Security, but the producer's portlets appear as unavailable on pages at run time.

Problem 1

You can have inadvertently entered incorrect values (for example, for **Issuer**) in the WSRP Producer Registration Wizard when you registered the portlet. The wizard cannot actually validate the security information, hence the registration can complete successfully, but the portlets are inaccessible at run time due to some incorrect security values.

Solution 1

Deregister the portlet and then reregister it ensuring that you have the correct values specified, particularly for things like **Issuer**.

Problem 2

Your keystore path was not configured correctly to use an absolute path, or your Oracle Containers for J2EE (OC4J) instance was not restarted after the configuration. Your log file (typically located in *ORACLE_HOME*/j2ee/OC4J_WebCenter/log/OC4J_WebCenter_default_group_1/oc4j) contains WARNING level log messages like the following:

```
<MSG_TEXT>
Couldn't resolve kestore path :
/ASINSTALL10132/j2ee/OC4J_WebCenter/META-INF/portal.jks
</MSG_TEXT>

<MSG_TEXT>
Invalid keystore path META-INF/portal.jks
</MSG_TEXT>
```

Solution 2

Perform the steps in Updating the Keystore Path under Section 10.10.3, "Configuring the Consumer".

G.1.3.3 Error When Converting the JPS Producer EAR File to WSRP Producer EAR File

Your JPS portlet producer EAR file fails to convert to a WSRP producer EAR file.

Problem

When attempting to convert a JPS portlet producer EAR file into a WSRP producer EAR file, the following exception is thrown:

```
Exception in thread "main"
oracle.portlet.server.containerimpl.deploy.PortletDeployException:
A network error occurred during XML transformation. Check your network, firewall
and proxy settings.
```

Solution

For JPS-compliant portlets developed with servlet version 2.3, you must specify Web proxies using a command of the form:

```
java -Dhttp.proxyHost=proxy_host -Dhttp.proxyPort=proxy_port -jar
wsrp-predeploy.jar source_ear_file targeted_ear_file
```

G.1.3.4 Portlets Unavailable for Producer with Different Preference Store Path

Portlets are throwing a `Portlet unavailable` error because their producer has a different preference store path than the rest of the producers in the OC4J instance.

Problem

JPS portlets for a particular producer are unavailable even though portlets for other producers within the same OC4J instance are available.

Solution

The preference store path, `oracle/portal/wsrp/server/fileStoreRoot`, in `web.xml` must be the same for all producers deployed to the same OC4J instance. If you simply let the preference store location default rather than explicitly specifying it, then all producers will use `portletdata` as the default location. If you explicitly specify a path for any one of the producers in the OC4J instance, then you must ensure

that all of the other producers in that same instance use that same path. Otherwise, some portlets will be unavailable.

G.1.3.5 JPS portlet Does Not Work on Pluto

The JPS portlet that is created by using JDeveloper does not work on Pluto.

Problem

When you try to deploy the JPS portlet on Pluto, the portlet throws an error.

Solution

Perform the following steps in the wizard to get the JPS portlets working on Pluto:

1. The portlet mode JSPs must be changed to be created under the `WEB-INF` folder.
2. In the newly created web.xml, remove all attributes except for `version="2.4"` in the `<web-app>` root. This applies only if you have selected the Web Application version as `Servlet 2.4\JSP 2.0(J2EE 1.4)` while creating the JSR 168 portlet.

G.1.4 Troubleshooting PDK-Java Portlets

This section describes common problems and solutions for PDK-Java portlets.

G.1.4.1 Redirect Error in PDK-Java Portlet

Your portlet displayed the following message when accessed:

`##REDIRECT##<`*`URL`*`>`

Problem

Your portlet probably issued a redirect call to some URL immediately upon switching to a particular portlet mode. Portlets cannot issue redirects except as the result of a user action within the portlet, such as a form post.

Solution

Rewrite your portlet to avoid this behavior.

G.1.4.2 Images Not Appearing in Full Page Portlet Modes

When you switch to a full page portlet mode (for example, Edit mode), images are broken or missing.

Problem 1

You are running the producer on a WebCenter Suite OC4J instance and the portlet is running on an Oracle ADF page. If the page markup for images contains a relative URL, like `/images/`*`image_name`*, then the most likely cause of the problem is that the images are not present on the ADFP servlet.

If the markup contains a full URL to the image, then the most likely cause of the problem is that the producer is not configured to use resource proxying. In this case, the URL refers directly to a producer, which cannot be directly accessible to the user, rather than through the ADFP servlet. If resource proxying is used, the image URL points to the resource proxy servlet.

Solution 1

Perform the following actions:

- Make sure the resource servlet configuration details are included in the `web.xml` for the producer application and bounce the producer if you must add them. You should now see full URLs for each image tag `src` attribute. See Section A.1.1.9.2, "Redeploying PDK-Java Producers from OracleAS Portal".
- Get the images zip file from the Oracle Technology Network and unzip it inside the ADFP servlet.

Problem 2

You are running the producer on a WebCenter Suite OC4J instance and the portlet is running on an Oracle Application Server Portal (OracleAS Portal) page. In this case, the most likely problem is that you do not have the resource proxy properly configured.

Solution 2

Make sure the resource servlet configuration details are included in the `web.xml` for the producer application and bounce the producer if you must add them. You should now see full URLs for each image tag `src` attribute. See Section A.1.1.9.1, "Consuming a Portlet from OracleAS Portal".

G.1.4.3 PDK-Java Portlet With Non-ASCII Characters Fails

Problem

A PDK-Java portlet fails when the resource for any portlet mode is a JSP whose file name contains non-ASCII characters.

Solution

Rename the JSP portlet mode resources to use ASCII characters only.

G.1.4.4 PDK-Java Producer with Multibyte Characters in Service ID Fails

PDK-Java producers cannot have multibyte characters in their Service IDs.

Problem

Registration fails and the producer test page returns an error:

```
oracle.webdb.provider.v2.utils.soap.SOAPException: Can't read deployment
properties for service: ??????
```

Solution

Rebuild the producer and provide a Service ID without multibyte characters.

G.1.4.5 Images Not Found Running PDK-Java Portlets of a JSF Application

Problem

Images accessed by a relative URL cannot be found in an Oracle WebCenter Framework environment as they were in an OracleAS Portal environment.

Solution

You can resolve this issue in one of two ways:

- Add the images to the Oracle WebCenter Framework servlet so that the relative URLs in image markup will point to them.

- Use `UrlUtils.constructResourceURL` to generate and render an absolute link that will refer either to a proxy for the resource (if resource proxying has been set up for the producer) or directly to the image resource on the Web producer. See Section 18.1.2.4, "Internal/Resource Links" for more information about Web producer resource proxying.

G.1.5 Troubleshooting Portlets Built from Oracle ADF Faces Applications

This section describes common problems and solutions that you may encounter while you code Oracle ADF pages that you can later choose to publish as portlets, or while running portlets built from Oracle ADF Faces applications.

G.1.5.1 Error Creating the Portlet

When you try to portletize your Oracle ADF page, you get an error and are unable to proceed.

Problem

Your application was not built using a Web Application template that employs JSF.

Solution

Ensure that your application is built with one of the Web Application templates that employs JSF.

G.1.5.2 Error Finding Images and Resources in Your Portlet

When you run your portlet, you get errors because the images and resources could not be located.

Problem

The paths to these files are not specified to be relative to the Web application context root.

Solution

Specify the image and resource locations relative to the `web-app-context-root`. Do not use relative (../) path notation.

G.1.5.3 Private Portlet Parameters Lost on Navigating to Another Page

When running a portlet, if you navigate away from the page and return to it again, then you may find that the portlet state is lost. That is, the private portlet parameter settings you made earlier are lost.

Problem

For Oracle ADF Faces applications portlets, the state must be saved on the server side. By default, Oracle ADF Faces is client-managed.

Solution

You must explicitly configure the application to use server-side state management by defining the following in your `web.xml`:

```
<context-param>
  <param-name>javax.faces.state_saving_method</param-name>
  <param-value>server</param-value>
</context-param>
```

G.1.5.4 Error When Testing a Producer's WSDL URL

When testing the producer's Web Services Description Language (WSDL) URL in development environment, you get an error.

Problem

The OC4J instance was not bounced.

Solution

Restart OC4J before testing the WSDL URL in a browser.

G.1.5.5 Error When Accessing WSRP Portlets

When WSRP portlets are deployed to an OC4J instance in Oracle Application Server and tested in a browser, 404 error appears.

Problem

WSRP portlets were deployed to the home instance of OC4J in Oracle Application Server, which did not have a WSRP container.

Solution

Deploy WSRP portlets to the OC4J_Portal instance of Oracle Application Server instead of the home instance.

G.1.5.6 Unable to Navigate to Another Page from the Portlet

Unable to navigate to another page from your portlet.

Problem

You may have used `response.sendRedirect("")` to navigate to another page.

Solution

To work properly in a portlet environment, you must implement navigation with navigation rules in `faces-config.xml`.

G.1.5.7 Portlet Not Rendered on Page

You may experience problems when trying to render your JSF portlet on a page.

Problem

The page definition, `pageDef.xml`, must have the `doExecuteWithParams` executable's `Refresh` attribute set to `renderModel` rather than `prepareModel`.

Solution

Set the `Refresh` attribute to `renderModel` as shown in the following example:

```
<invokeAction id="doExecuteWithParams" Binds="ExecuteWithParams"
   Refresh="renderModel" RefreshCondition="${param.id != null}"/>
```

G.1.5.8 Missing Class Error When Deploying a Portlet

You get a missing class error when you update the portlet deployment descriptor, `portlet.xml`, and then try to deploy the application.

Problem

The proper classes are not defined in `portlet.xml`. Your `portlet.xml` file may not be configured to match the filters specified in your `web.xml` file

Solution

Specify the classes using the `BridgeLifecycleListeners` attribute in your `portlet.xml` file as follows:

```
<init-param>
  <name>BridgeLifecycleListeners</name>
  <value>listener_class_n[,listener_class_n+1,...]</value>
</init-param>
```

For example, suppose that your application includes the following code excerpt in its `web.xml`:

```
<filter>
  <filter-name>adfFaces</filter-name>
  <filter-class>oracle.adf.view.faces.webapp.AdfFacesFilter</filter-class>
</filter>
```

You would then must include the following class in your `portlet.xml`:

```
<init-param>
  <name>BridgeLifecycleListeners</name>
  <value>oracle.portlet.server.bridges.jsf.adf.ADFFacesBridgeLifecycleListener
    </value>
</init-param>
```

Similarly, suppose that `ADFBindingFilter` is defined in `web.xml` as follows:

```
<filter>
  <filter-name>adfBindings</filter-name>
  <filter-class>oracle.adf.model.servlet.ADFBindingFilter</filter-class>
</filter>
```

You would then must include the following class in your `portlet.xml`:

```
<init-param>
  <name>BridgeLifecycleListeners</name>
<value>oracle.portlet.server.bridges.jsf.adf.
    BindingFacesBridgeLifecycleListener</value>
</init-param>
```

G.1.5.9 Portlet Unavailable Error

When turning a JSF application into a portlet using the JSF portlet bridge, packaging, deploying, and consuming the portlet, the portlet displays the following error message:

```
Portlet Unavailable
```

Problem

Your application is not using Oracle ADF binding, but the `portlet.xml` contains `BindingFacesBridgeLifecycleListener`. You probably have something similar to the following in your `portlet.xml`:

```
<init-param>
 <name>BridgeLifecycleListeners</name>
 <value>
  oracle.portlet.server.bridges.jsf.adf.ADFFacesBridgeLifecycleListener,
```

```
    oracle.portlet.server.bridges.jsf.adf.BindingFacesBridgeLifecycleListener
   </value>
  </init-param>
```

Solution

Ensure that your `portlet.xml` does not make reference to `BindingFacesBridgeLifecycleListener`. It should look similar to the following:

```
<init-param>
 <name>BridgeLifecycleListeners</name>
  <value>
   oracle.portlet.server.bridges.jsf.adf.ADFFacesBridgeLifecycleListener
  </value>
</init-param>
```

Notice the omission of `BindingFacesBridgeLifecycleListener` in this case.

G.1.6 Troubleshooting OmniPortlet Problems

This section provides information to help you troubleshoot problems you may encounter while using OmniPortlet.

G.1.6.1 Cannot Define OmniPortlet Using the Define Link

You are not able to define the OmniPortlet at run time by using the **Define** link.

Problem

OmniPortlet only supports a `RenderPortletInIFrame` value of `true`. This means that OmniPortlet must be rendered within an IFrame and therefore, the OmniPortlet property, `RenderPortletInIFrame`, must be set to `true`. In the Property Inspector, the `RenderPortletInIFrame` property is available under Display Options.

Currently, the `RenderPortletInIFrame` property has a value of `false` and, as a result, when you click the **Define** link at run time, the Type tab may not display and you cannot proceed with defining the OmniPortlet.

Solution

You can choose Customize from the Action list to define OmniPortlet, or select the OmniPortlet in the Structure pane in Oracle JDeveloper, and in the Property Inspector, set `RenderPortletInIFrame` to `true`.

G.1.7 Troubleshooting Application Life Cycle Issues

This section describes common problems and solutions you may encounter when deploying or predeploying your WebCenter application or its components.

G.1.7.1 Predeployment Tool Unable to Create Temporary Area on MS Windows

The Predeployment tool cannot unlock the temporary directory to create temporary files and therefore it fails.

Problem

When you abnormally terminate the Predeployment tool (for example, by pressing Ctl-C), it may have some temporary files locked and, on MS Windows, the locks may be transferred to the invoking `cmd` process. Hence, the locks on the temporary files are not released and, when you try to run the Predeployment tool again, the locks prevent

it from creating the necessary temporary files for the operation. The following error is thrown:

```
Unable to create temporary work area
```

Solution

Invoke the Windows Task Manager to find the invoking `cmd` process and terminate it.

G.1.7.2 Cannot Create Generic EAR File

When deploying the application to an EAR file in Oracle JDeveloper, you receive an error similar to the following in the deployment console.

```
Error during export
```

Problem

One of the applications portlet producers could be unavailable.

Solution

If the producer is down, then you must restart it. If the producer is up and you still receive this error, then you should contact Oracle Support with the exception stack from the Oracle JDeveloper log window.

G.1.7.3 Unable to Find MDS for Portlet Producer

After specifying all the settings requested, the Predeployment tool failed with the error:

```
Unable to find MDS Repository for Portlet Producers
```

Problem

You may not have created the generic EAR file with the WebCenter Application WAR deployment profile. You may have inadvertently used the WAR File deployment profile.

Solution

Recreate the deployment profile being sure to choose WebCenter Application WAR. Regenerate the generic EAR file using the corrected deployment profile.

G.1.7.4 Predeployment Tool Fails with Unexpected PortletException

After specifying all the settings requested, the Predeployment tool failed with one of the following errors:

```
Unexpected PortletException
```

```
Error during import
```

Problem

The target producer may be down.

Solution

Check the target producer to make sure it is up and running. If it is down, then restart it.

G.1.7.5 Portlets Not Appearing on Deployed Application Pages

The application runs fine in Oracle JDeveloper's embedded OC4J, but, after deploying to an OC4J instance outside of Oracle JDeveloper, all of the portlets on the page show the error:

```
No object with ID /oracle/adf/portlet/...
```

Problem

The Oracle Metadata Services (MDS) directory containing the portlets' metadata cannot be found.

Solution

When you create the generic deployment descriptor for the application in Oracle JDeveloper, make sure you associate the descriptor with a target platform (for example, OC4J).

G.1.7.6 No MDS Data Found, MDSRuntimeException

When attempting to run the page, the browser shows either one of the following errors:

```
Servlet error: An exception occurred.  The current application deployment
descriptors do not allow ...

MDSRuntimeException: No metadata found for metadata object .../*.jspx
```

Problem

These errors could be caused by any number of problems:

- This is an Oracle ADF Faces page and your URL do not invoke the faces servlet.
- `adf-config.xml` contains an invalid metadata path specification.
- `web.xml` contains an invalid `oracle.mds.web-app-root` specification.
- The view documents do not exist at the specified location.
- The application does not contain `orion-application.xml`.

Solution

Check all of the following:

- Make sure your URL references the faces servlet. Typically, it should look something like the following:

  ```
  <protocol>://<host>:<port>/<context root>/faces/<jspx document path>
  ```

- In the MDS directory, the location of the view document should be:

  metadata-path_in_adf-config.xml/oracle.mds.web-app-root_in_web.xml/ view_document_path_after_/faces_in_the_URL

  Check to make sure the settings in `adf-config.xml` and `web.xml` point to a valid path and the documents do exist in the specified path.

  Note: When running from Oracle JDeveloper, the `metadata-path` entry in `adf-config` could be a relative path. If so, it's relative to the location of `adf-config.xml`.

- For OC4J, the application requires `orion-application.xml`. Otherwise, `adf-config.xml` cannot be located. If you cannot see an `orion-application.xml` in the deployed application's META-INF directory, then make sure your application deployment descriptor in Oracle JDeveloper is associated with an OC4J/Oracle Application Server connection for the platform setting.

G.1.7.7 Garbled MDS Path When Predeploying on Europeans MS Windows

If your MDS path contains Western European characters, such as an umlaut over a vowel, then the path generated by the Predeployment tool becomes garbled.

Problem

After predeployment, the MDS path in `adf-config.xml` is garbled and unusable. This corrupted path will cause errors in any attempted deployment of your application.

Solution

Check the value of your code page by executing `chcp` at a command prompt. If the value returned is a DOS code page for the language (for example, 850), then you must change it to a Windows code page (for example, 1252). To change the code page, you execute `chcp` *`code_page_number`* at the command prompt. For example:

```
chcp 1252
```

G.2 Diagnosing WebCenter Applications (Logging)

Oracle WebCenter Framework can record information related to the requests they receive in log files. This section provides instructions to diagnose run time and design time problems and covers the following topics:

- Understanding Logging
- Configuring Logging
- Viewing the Log

G.2.1 Understanding Logging

Oracle WebCenter Framework logging is based upon the JDK logging framework, which includes the JDK logging Application Programming Interface (API). JDK logging has a standard API, `java.util.logging.Logger`, that has been available in the Java Platform since JDK 1.4. To enhance the basic JDK logging, the Oracle Diagnostic Logging (ODL) framework provides support for extended, Oracle-specific message formatting and configuration through Extensible markup Language (XML) files. You can choose to use JDK logging by itself, with no ODL enhancements, or you can use the JDK API to log diagnostic messages and ODL to provide enhanced output formatting and configuration through XML files. Both methods are described in the sections that follow.

G.2.2 Configuring Logging

Before you configure logging, you must know about the following:

- Logger Names and Scope
- Logging Levels

The same logging framework is used for both design time and run time, hence the only difference in configuration is that, in design time (Oracle JDeveloper), you must ensure that `jdev.conf` points to the correct logging configuration file. This step is called out where appropriate in the procedures that follow.

You can configure logging in either of the following ways. The first method provides you with enhanced ODL output. The second method provides only the standard JDK logging output. In all cases, though, the basic message bodies provide the same information.

- Configuring Logging Through the ODL Configuration File
- Configuring Logging Through the Default JDK Logging Properties File

G.2.2.1 Logger Names and Scope

Loggers provide diagnostic messages for particular components or areas of your WebCenter application. Table G–1 lists the available loggers and what they track. Note that the loggers adhere to the standard, hierarchical naming scheme. Hence, you can specify a parent logger, such as `oracle.portlet`, to turn on logging for all of its children, `oracle.portlet.binding` and `oracle.portlet.client`.

Table G–1 Available Loggers for Oracle WebCenter Framework

Logger Name	Description
`oracle.portlet`	Provides logging for both portlet bindings and portlet run time.
`oracle.portlet.binding`	Provides logging for portlet bindings.
`oracle.portlet.client`	Provides logging for portlet run time.
`oracle.portlet.management.mbean`	Provides logging for the portlet MBean.
`oracle.portal`	Provides logging for portal server components, including WSRP, PDK-Java, Omniportlet, and Web Clipping.
`oracle.vcr`	Provides logging for all Java Content Repository (JCR)-related functions.
`oracle.vcr.datacontrol`	Provides logging for JCR data controls.
`oracle.vcr.jam`	Provides logging for the JCR Adapter Manager.
`oracle.vcr.share`	Provides logging for the JCR run time shared package.
`oracle.vcr.dav`	Provides logging for the JCR Get Handler.
`oracle.vcr.adapter`	Provides logging for JCR adapters.

Table G–1 (Cont.) Available Loggers for Oracle WebCenter Framework

Logger Name	Description
`oracle.adfdtinternal.config`	Provides logging for configuring an application workspace in Oracle JDeveloper for a WebCenter application.
`oracle.adfdtinternal.model.portlet`	Provides logging for the portlet design time extension.
`oracle.adfinternal.model.portlet`	Provides logging for the portlet design time model.
`oracle.adfinternal.view.faces.renderkit.html.customizable`	Provides logging for customizable components.
`oracle.adfinternal.view.faces.ui.action`	Provides logging for customizable components.
`oracle.adf.share.security logger`	Provides logging for Oracle ADF security.
`oracle.adf.extapp`	Provides logging for external application and credential mapping service.
`oracle.adf.richtextportlet`	Provides logging for the Rich Text Portlet.

G.2.2.2 Logging Levels

By specifying the level of logging, you can choose the amount of log information to record. The default Java log levels and their ODL counterparts are listed in Table G–2. You can use the JDK logging levels in both JDK and ODL logging configuration files. The ODL levels can only be used in ODL logging configuration files.

Table G–2 Logging Scope

JDK Logging Level	ODL Logging Level	Description
SEVERE+100[1] (only specified in code)	INTERNAL_ERROR	Log unexpected errors or exceptions. These errors usually imply bugs in the code and customers must call support to report the problem.
SEVERE	ERROR:1	Log system errors requiring attention from the system administrator.
WARNING	WARNING:1	Log actions or conditions discovered that should be reviewed and may require an action before an error occurs.
INFO	NOTIFICATION:1	**Default logging level.** Log normal actions or events. This could be a user operation, such as *login completed*, or an automatic operation, such as a l*og file rotation*.

Table G–2 (Cont.) Logging Scope

JDK Logging Level	ODL Logging Level	Description
CONFIG	NOTIFICATION:16	Log configuration-related messages or problems.
FINE	TRACE:1	Log trace or debug messages used for debugging or performance monitoring. Typically includes detailed event data.
FINER	TRACE:16	Log fairly detailed trace or debug messages.
FINEST	TRACE:32	Log highly detailed trace or debug messages.

[1] SEVERE+100 can only be used within code. You cannot use it in the configuration files.

For more information about JDK logging levels, see:

`http://java.sun.com/j2se/1.4.2/docs/api/java/util/logging/Level.html`

G.2.2.3 Configuring Logging Through the ODL Configuration File

To configure logging that takes advantage of ODL features, you use the `j2ee-logging.xml` file that is installed in the `ORACLE_HOME/j2ee/`*`oc4j_instance`*`/config` directory.

The ODL framework provides plug-in components that complement the standard Java framework to automatically integrate log data with Oracle log analysis tools. In the ODL framework, log files are formatted in plain text or XML, enabling them to be more easily parsed and reused by other Oracle Application Server and custom developed components.

The `oracle.core.ojdl.logging` package includes a Handler class, `ODLHandler`, which generates the Logger output in either XML-based ODL format or plain text. To enable Java Loggers to output log messages in the ODL format, each logger must be mapped to `ODLHandler`. This mapping is managed through `j2ee-logging.xml`. See *Oracle Containers for J2EE Developer's Guide* for detailed information about configuring Java loggers to use the ODL framework. For general information about `j2ee-logging.xml`, see *Oracle Application Development Framework Developer's Guide*.

To configure logging information by using the `j2ee-logging.xml` file and to output log messages in the ODL format, perform the following steps:

1. Navigate to the *`ORACLE_HOME`*`/j2ee/home/config` directory and open the `j2ee-logging.xml` file in an editor. Note that if the component is deployed to an instance other than the home instance, then you will must substitute for `home` the path accordingly. For example, the path for an OC4J instance called `OC4J_WebCenter` would be *`ORACLE_HOME`*`/j2ee/OC4J_WebCenter/config`.
2. Choose whether you want log output in XML or plain text format. The default format is XML.

Note: In `j2ee-logging.xml`, you can also change other log properties, such as the location of the log file. For example:

```
<property name='path' value='C:/TEMP/log/'/>
```

- Example G–1 provides a sample `j2ee-logging.xml` fragment that configures logging for XML output.

Example G–1 j2ee-logging.xml for XML Logging Output

```
<logging_configuration>
       <log_handlers>
         ...
         <log_handler name='ptlLoggingHandler'
           class='oracle.core.ojdl.logging.ODLHandlerFactory'>
           <property name='path' value='C:/TEMP/log/'/>
           <property name='maxFileSize' value='104857600'/>
           <property name='maxLogSize' value='1048576000'/>
         </log_handler>
       </log_handlers>
       <loggers>
         ...
         <logger name='oracle.adfinternal.model.portlet'
                 level='FINER'
                 useParentHandlers='false'>
           <handler name='ptlLoggingHandler'/>
         </logger>
         <logger name='oracle.adfdtinternal.model.portlet'
                 level='FINER'
                 useParentHandlers='false'>
           <handler name='ptlLoggingHandler'/>
         </logger>
         <logger name='oracle.portlet.client'
                 level='FINE'
                 useParentHandlers='false'>
           <handler name='ptlLoggingHandler'/>
         </logger>
       </loggers>
    </logging_configuration>
```

- Example G–2 provides a sample `j2ee-logging.xml` fragment that configures logging for plain text output. Note the lines in bold. The first two bold lines configure the log for plain text output. The second two bold lines reduce the clutter, which is typical when you are planning to view the output in plain text format.

Example G–2 j2ee-logging.xml for Plain Text Output

```
<logging_configuration>
       <log_handlers>
         ...
         <log_handler name='ptlLoggingHandler'
           class='oracle.core.ojdl.logging.ODLHandlerFactory'>
           <property name='path' value='C:/TEMP/log/diag.log'/>
           <property name='maxFileSize' value='104857600'/>
           <property name='maxLogSize' value='1048576000'/>
           <property name='format' value='ODL-Text'/>
           <property name='useSourceClassAndMethod' value='true'/>
           <property name='useDefaultAttributes' value='false'/>
         </log_handler>
       </log_handlers>
       <loggers>
         ...
         <logger name='oracle.adfinternal.model.portlet'
                 level='FINER'
```

```
            useParentHandlers='false'>
        <handler name='ptlLoggingHandler'/>
      </logger>
      <logger name='oracle.adfdtinternal.model.portlet'
            level='FINER'
            useParentHandlers='false'>
        <handler name='ptlLoggingHandler'/>
      </logger>
      <logger name='oracle.portlet.client'
            level='FINE'
            useParentHandlers='false'>
        <handler name='ptlLoggingHandler'/>
      </logger>
    </loggers>
  </logging_configuration>
```

3. Save the `j2ee-logging.xml` file.
4. In the design time environment, you must point to `j2ee-logging.xml` for logger settings by updating `JDEV_HOME/jdev/bin/jdev.conf` as follows:

```
# Add OJDL jars in the classpath
AddJavaLibFile ../../BC4J/lib/adfshare.jar
AddJavaLibFile ../../diagnostics/lib/ojdl.jar
# Add logger configuration for diagnostics logging
AddVMOption
-Djava.util.logging.config.class=oracle.core.ojdl.logging.LoggingConfiguration
AddVMOption -Doracle.core.ojdl.logging.config.file=
JDEV_HOME/j2ee/home/config/j2ee-logging.xml
```

5. Restart the Oracle Application Server.

   The log file will be generated in the location you specified in `j2ee-logging.xml` file, that is, in `C:/TEMP/log/diag.log`. You can also view the log file in Oracle Enterprise Manager 10*g*.

> **Note:** In the design time environment, instead of using the default `j2ee-logging.xml` file, you can create a custom properties file and use that file to specify logging configuration. However, if you use a custom properties file, then you must edit the *<JDEV_HOME>*`/jdev/bin/jdev.conf` file to point to the new properties file.

G.2.2.4 Configuring Logging Through the Default JDK Logging Properties File

To configure standard JDK logging, you can use the default logging properties file, `jdk/jre/lib/logging.properties`, and perform the following steps:

1. Navigate to the `jdk/jre/lib` directory and open `logging.properties` in an editor.
2. Under the `Global properties` section, specify the log handler class to be used. This determines whether log information must be output to the console or recorded in a log file. By default, console output is configured and the entry appears as follows:

   ```
   handlers= java.util.logging.ConsoleHandler
   ```

   To enable file output also, uncomment the `FileHandler` entry, which appears as follows:

```
#handlers= java.util.logging.FileHandler, java.util.logging.ConsoleHandler
```

3. Specify the global logging level by setting `level` under the `Global properties` section. This controls the type of events that must be logged. The default global logging level is `INFO`. Supported log levels are described in Section G.2.2.2, "Logging Levels".
4. Under the `Handler specific properties` section, specify values for the following:
   - `java.util.logging.FileHandler.pattern`: Specify a name and location for the log file

     Note: At design time, if you specify the `FileHandler` pattern as `portlet.log`, then the log file will be generated under the `<JDev_install>/jdev/bin` directory.

     See the following location for information about `FileHandler` patterns to be used for Unix and Windows Platforms:

     http://java.sun.com/j2se/1.4.2/docs/api/java/util/logging/FileHandler.html

   - `java.util.logging.ConsoleHandler.level`: Specify the logging level, which limits the message that is printed to the console
5. Under the `Facility specific properties` section, you can specify any extra controls for each logger as shown in the following examples:

```
oracle.adfinternal.model.portlet.level = FINER
oracle.adfdtinternal.model.portlet.level = FINER
oracle.portlet.client.level = FINE
oracle.vcr.datacontrol.level = FINE
oracle.vcr.adapter.fs.level = FINE
```

6. Save the `logging.properties` file.

A sample configuration in `logging.properties` file is shown in Example G–3, with new settings shown in **bold** text.

Example G–3 Sample Configuration in the logging.properties File

```
# "handlers" specifies a comma separated list of log Handler
# classes.  These handlers will be installed during VM startup.
# Note that these classes must be on the system classpath.
# By default we only configure a ConsoleHandler, which will only
# show messages at the INFO and above levels.
handlers= java.util.logging.ConsoleHandler

# To also add the FileHandler, use the following line instead.
handlers= java.util.logging.FileHandler, java.util.logging.ConsoleHandler

  # Default global logging level.
  .level= INFO
  ############################################################
  # Handler specific properties.
  # Describes specific configuration info for Handlers.
  ############################################################

  # default file output is in user's home directory.
```

```
java.util.logging.FileHandler.pattern = %h/portlet%u.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.formatter = java.util.logging.SimpleFormatter

# Limit the message that are printed on the console to INFO and above.
java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter

############################################################
# Facility specific properties.
# Provides extra control for each logger.
############################################################

oracle.adfinternal.model.portlet.level = FINER
oracle.adfdtinternal.model.portlet.level = FINER
oracle.portlet.client.level = FINE
```

G.2.2.4.1 Configuring Logging Through Custom JDK Logger Properties File

At design time, instead of using the default `logging.properties` file, you can create a custom properties file and use that file to specify standard JDK logging configuration. However, if you use a custom properties file, you must edit the `JDEV_HOME/jdev/bin/jdev.conf` file and update the following entry to point to the new properties file:

```
AddVMOption -Djava.util.logging.config.file=<path and name of the properties file>
```

For example, you can create a custom file called `portlet-logging.properties` under the `/tmp` directory. Copy the content from the default `logging.properties` file to the new file and update configuration information as described in Section G.2.2.4, "Configuring Logging Through the Default JDK Logging Properties File". After this, to point to `portlet-logging.properties` for logger settings, you must update the `JDEV_HOME/jdev/bin/jdev.conf` file as follows:

```
AddVMOption -Djava.util.logging.config.file=/tmp/portlet-logging.properties
```

Logger settings are now picked up from the `portlet-logging.properties` file that you created.

For information about how to configure custom JDK logger properties files, see

G.2.3 Viewing the Log

You can open the log file from whatever location you specified in the configuration file using an appropriate editor or viewer. If you are using Oracle JDeveloper, then you can view the diagnostic output in the Oracle JDeveloper log window. The default location of the Oracle JDeveloper log file is `JDEV_HOME/jdev/bin/portlet.log`.

G.3 Need More Help?

You can find more solutions on Oracle *MetaLink* at

`http://metalink.oracle.com`

If you do not find a solution for your problem, then log a service request.

Contact Oracle Support Services.

See Also: *Oracle Application Server Release Notes for Microsoft Windows*, available on Oracle Technology Network (OTN) http://www.oracle.com/technology/documentation/index.html

Glossary

About mode

A **portlet mode** that typically displays information such as copyright, version, and author of the portlet.

ADF

Application Development Framework. A range of technologies aimed at making **Java EE** application development faster and simpler for developers while at the same time taking advantage of proven software patterns to ensure that the developed application is scalable, performant, and the like.

API

Application Programming Interface. A set of exposed data structures and functions that an application can use to invoke services on an application object, such as a **portlet**.

Application Development Framework

See **ADF**.

Application Programming Interface

See **API**.

authenticated user

A user who is logged into a **WebCenter application**. By default, an authenticated user can access public and secured information, such as pages and **portlet**s.

Contrast with **public user**s, who can access public content only.

authorization

The policies that define the access rights of an individual or group to a secured resource. This resource may be a page or component within a page.

authorized user

An individual who has access to a secured resource. For non-public resources, this individual is also an **authenticated user**.

caching

The act of storing frequently accessed information, typically Web pages, in a location where it can be accessed quickly to avoid frequent content generation.

See also **expiry-based caching**, **invalidation-based caching**, and **validation-based caching**.

check out/check in

A mechanism that enables a user to lock information, by checking it out, so that other users cannot modify that same piece of information. This prevents users from overwriting each other's changes. After making modifications, the user releases it by checking it back in, making it available again for other users to modify.

component developer

The developer who builds components (such as portlets, **JavaServer Faces** components, and Web services).

container

An application program or subsystem in which the program building block, known as a component, is run.

content integration services

Services provided by **Oracle WebCenter Suite** to enable developers to display content from a **content repository**, such as by creating **data control**s.

content repository

A specialized storage and management mechanism, such as author-based versioning, full textual searching, content categorization and attribution, and is optimized for storing unstructured information, which differentiates it from a data repository.

content repository data control

A **data control** sourced though a content repository. In a **WebCenter application**, you can create content repository data controls for the following content repositories: **OracleAS Portal**, **Oracle Content Database**, and third-party repositories supporting the Java Content Repository (JCR) standard, or your local file system.

credential provisioning page

A **JSF** (`*.jspx`) page used for authenticating to an **external application**. At run time, the Credential Provisioning page displays login data fields consisting of the data fields specified through external application registration. Login information is passed to the producer, which in turn passes the login values to the external application. The application provides the producer with the requested portlets.

After authentication, the user's login credentials are preserved in a **credential store**, which subsequently supplies that information at future sessions. Unless his information changes, the user supplies his credentials only once.

credential store

A storage area that preserves the login credentials a user provides for authentication to an **external application**.

CSS

Cascading Style Sheet. A simple mechanism for ensuring a consistent look and feel or adding style, such as fonts, colors, and spacing, to Web documents.

Customize mode

A **portlet mode** that enables administrators to set the default values for portlet preferences for all users.

dashboard page

An easy-to-read user interface that organizes and presents metrics and key performance indicators related to business activity and business intelligence.

data control

A mechanism that provides an abstraction of the business service's data model. The ADF data controls provide a consistent mechanism for clients and Web application controllers to access data objects, collections, methods, and operations.

deployment profile

A file used in application deployment that specifies the following types of information:

- The source files, deployment descriptors, and other auxiliary files that will be packages
- The type and name of the archive file to be created
- Dependency information
- Platform-specific instructions
- Other information

Oracle WebCenter Services provides a special deployment profile, the **WebCenter application** WAR deployment profile, that includes an option to export project metadata.

EAR

Enterprise Archive file. A **Java EE** archive file that is used in deploying applications on a Java EE application server. **WebCenter application**s are deployed using both a generic EAR file containing the application and the respective run-time customization and a targeted EAR file containing only the application for deployment to the application server. EAR files simplify application deployment by reducing the possibility of errors when moving an application from development to test, and test to production.

See also **JAR** and **WAR**.

ECMA-262 specification

A standardization of scripting programming languages, such as **ECMAScript** and **JavaScript**.

ECMAScript

A scripting programming language, standardized by Ecma International according to the **ECMA-262 specification**. Frequently referred to as **JavaScript** or JScript, which are both extensions of the ECMA-262 specification.

Edit mode

A **portlet mode** that enables personalization of the portlet for each user, for each instance.

See also **Edit Defaults mode**.

Edit Defaults mode

(**JSR 168** portlets only.) A **portlet mode** that enables personalization of a JSR 168 portlet. Edit_defaults mode is a display mode for the JSR 168 portlet's properties. In a

WebCenter application, the edit_defaults mode displays on the portlet's Actions menu as the Customize command.

See also **Edit mode**.

EL

Expression Language (EL) provides a short-hand way of working with Web application data by providing operators for retrieving and manipulating application data residing in a **Java EE** Web container. In a **WebCenter application**, EL expressions are encapsulated in the characters "#{" and "}" and typically come in the form #{object.data} where *object* represents any Java object or **ADF** component placed in the Java EE Web container's page, request, session, or application's scope.

Enterprise Archive file

See **EAR**.

Enterprise Manager

See **OEM**.

expiry-based caching

A **caching** method that uses a retention period to specify how long the item is valid in the cache before a refresh is required. When there is a request for the item beyond the retention period, it is refreshed in the cache.

See also **invalidation-based caching** and **validation-based caching**.

Expression Language

See **EL**.

Extensible Markup Language

See **XML**.

external application

Applications in the **Oracle WebCenter Suite** that provide a means of accommodating applications external to the Oracle WebCenter Suite that require user authentication.

Federated Portal Adapter

See **FPA**.

FPA

Federated Portal Adapter. A module in the portal instance (written in both Java and PL/SQL) that receives SOAP messages for a **PDK-Java producer**, parses the SOAP, and then dispatches the messages to a database producer as PL/SQL procedure calls. In effect, the FPA makes a database producer behave exactly the same way as a PDK-Java producer, enabling users to distribute their database producers across database servers. All remote producers can be treated as PDK-Java producers, hiding their implementation (database or Web) from the user. The most common use is to share database producer s (including page groups) owned by one portal instance among other portal instances and **WebCenter application**s.

Full Screen mode

(**PDK-Java** portlets only.) A **portlet mode** that provides more content than can be shown in the portlet when it is sharing a page with other portlets.

HA

High Availability. A collection of solutions to ensure that your applications meet the required availability to achieve your business goals, eliminating single points of failure with no or minimal outage in service.

Help mode

A **portlet mode** that displays usage information about the functionality of the portlet.

High Availability

See **HA**.

HTML

Hypertext Markup Language. A format for encoding hypertext documents that may contain text, graphics, and references to programs and other hypertext documents.

HTTP

Hypertext Transfer Protocol. The underlying format, or protocol, used across the Web to format and transmit messages and determine what actions **Web server**s and browsers should take in response to various commands. HTTP is the protocol typically used between **Oracle Application Server** and its clients.

Hypertext Markup Language

See **HTML**.

Hypertext Transfer Protocol

See **HTTP**.

IDE

Integrated Development Environment. A visual application development tool containing editors, debuggers, screen painters, object browsers, and the like.

infrastructure administrator

The administrator responsible for the Oracle Application Server infrastructure used by the **WebCenter application**. The infrastructure administrator's tasks would include such things as configuring Oracle Identity Management and Oracle Application Server High Availability Solutions, as well as configuration of production content repositories.

initialization parameters

The parameters initialized upon the start-up of a standard JSR 168 portlet. Initialization parameters provide an alternative to JNDI (Java Naming and Directory Interface) variables. Use initialization parameters instead of JNDI to configure the behavior of all of the different components of the portlet—for example, servlets and other portlets—in a compatible way. In **Oracle WebCenter Suite**, initialization parameters are entered into the `portlet.xml` file.

Integrated Development Environment

See **IDE**.

invalidation-based caching

A **caching** method where an item remains in the cache until it is explicitly invalidated. For example, a user may update an item, requiring the item in the cache to be

invalidated. The next time there is a request for the invalidated item, it is refreshed in the cache.

See also **expiry-based caching** and **validation-based caching**.

J2EE

See **Java EE**.

J2SE

Java 2 Platform, Standard Edition. A platform that enables application developers to develop, deploy, and manage Java applets and applications on a desktop client platform such as a personal computer or workstation. J2SE not only defines API standards, but also specifies the deployment of enterprise applications, thus enabling application server administrators to perform the deployment regardless of the vendor of the J2SE server.

See also **OC4J**.

JAAS

Java Authentication and Authorization Service (JAAS) is a Java package that enables applications to authenticate and enforce access controls upon users. JAAS is designed to complement Java 2 security and implements a Java version of the standard Pluggable Authentication Module (PAM) framework. This enables an application to remain independent from the authentication service, and supports the use of custom authentication modules.

JAAS extends the access control architecture of the Java 2 Security Model to support subject-based authorization. It also supports declarative security settings, in deployment descriptors, instead of being limited to code-based security settings.

JAR

A Java archive file. JAR files contain the class, image, and sound files for a Java application or applet. JAR files may also be compressed.

See also **EAR** and **WAR**.

Java Authentication and Authorization Service

See **JAAS**.

Java EE

Also known as Java EE 5. Java Enterprise Edition 5 Platform. A platform that enables application developers to develop, deploy, and manage multitier, server-centric, enterprise-level applications. The Java EE platform offers a multitiered distributed application model, integrated XML-based data interchange, a unified security model, and flexible transaction control. You can build your own Java EE portlets and expose them through Web producers.

See also **OC4J**.

Java Enterprise Edition 5 Platform

See **Java EE**.

Java 2 Platform, Standard Edition

See **J2SE**.

JavaScript

A scripting language developed by Netscape that enables generation of **portlet**s that introduce dynamic behavior in otherwise static HTML. This language is compliant with the European Computer Manufacturing Association's **ECMA-262 specification** (ECMA-262 standard). An alternative name for this EMCA-262 language is **ECMAScript**.

Java Specification Request

See **JSR 168**.

JavaServer Faces

See **JSF**.

JavaServer Page

See **JSP**.

JCR 1.0

Java Content Repository 1.0. Also known as **JSR 170**. It proposes a standard access and interaction API for content repositories, much like JDBC does for databases.

JSF

JavaServer Faces (JSF) is a new standard Java framework for building Web applications. It simplifies development by providing a component-centric approach to developing Java Web user interfaces. JSF offers rich and robust **API**s that provide programming flexibility and ensures that applications are well designed with greater maintainability by integrating the Model-View-Controller (**MVC**) design pattern into its architecture. As JSF is a Java standard developed through Java Community Process, development tools like **Oracle JDeveloper** are fully empowered to provide easy to use, visual, and productive development environments for JSF.

JSF JSP

JavaServer Faces JavaServer Page. JSF JSPs differ from plain JSPs through their support of **Oracle ADF Faces** components for the user interface and JSF technology for page navigation. JSF JSP pages leverage the advantages of the Oracle **Application Development Framework** (Oracle ADF) by using the ADF Model binding capabilities for the components in the pages.

JPS

Java Portlet Specification. Standardizes how components for portal servers are to be developed. This specification defines a common portlet **API** and infrastructure that provides facilities for personalization, presentation, and security. Portlets using this API and adhering to the specification will be product-agnostic, and can be deployed to any portal product that conforms to the specification. See also **JSR 168**.

JSP

JavaServer Pages. An extension to servlet functionality that provides a simple programmatic interface to Web pages. JSPs are HTML pages with special tags and embedded Java code that is executed on the Web or application server. JSPs provide dynamic functionality to HTML pages. They are actually compiled into servlets when first requested and run in the servlet container.

See also **JSP tags**.

JSP tags

Tags that can be embedded in **JSP**s to enclose Java code. These tags use the `<jsp:` syntax and enclose action elements in the JSP with `begin` and `end` tags similar to **XML** elements.

JSR 168

Java Specification Request (JSR) 168. Defines a set of APIs for building standards-based portlets using Java. Portlets built to this specification can be rendered to a portal locally or deployed to a WSRP container for rendering portlets remotely. For more information, see `http://jcp.org/en/jsr/detail?id=168`.

JSR 170

See **JCR 1.0**

key store

A storage area that contains the private key that is used to sign a **WSRP** portlet **producer**'s security assertions and to select the signature key alias that corresponds to the private key to be used for signing.

MDS

Oracle Metadata Services. A core technology of the **Application Development Framework**. MDS provides a unified architecture for defining and using metadata in an extensible and customizable manner.

middle tier

Part of the Oracle Application Server architecture that handles **HTTP** user requests by forwarding them to the appropriate portal database or **producer** and manages **caching** of content.

Model-View-Controller

See **MVC**.

MVC

Model-View-Controller. A classic design pattern often used by applications that need the ability to maintain multiple views of the same data. The MVC pattern hinges on a clean separation of objects into one of three categories: models for maintaining data, views for displaying all or a portion of the data, and controllers for handling events that affect the model or views. Because of this separation, multiple views and controllers can interface with the same model. Even new types of views and controllers that never existed before, such as portlets, can interface with a model without forcing a change in the model design.

navigation parameter

Parameters in a **WSRP** container that map to the render parameters with the same name in **JSR 168** portlet code. Navigation parameters are exposed by the portlet to the consumer. The consumer stores and manages parameter values and sends them on every invocation to the portlet. Navigation parameters are a WSRP version 2 feature.

OC4J

Oracle Containers for J2EE. The **Java EE** server component of Oracle Application Server written entirely in Java that executes on the standard Java Development Kit (JDK) Virtual Machine (Java VM). It includes a **JSP** Translator, a Java servlet container, and an Enterprise JavaBeans (JB) container.

OEM

See **Oracle Enterprise Manager**

OID

See **Oracle Internet Directory**

OmniPortlet

A component of **Oracle WebCenter Suite** that enables you to inject portal-like capabilities, such as portlets, content integration, and customization, into your **Oracle ADF Faces** applications.

Oracle ADF Faces

Oracle **ADF** Faces is a rich set of user interface components based on the new JavaServer Faces JSR (JSR 127). Oracle ADF Faces provide various user interface components with built-in functionality, such as data tables, hierarchical tables, and color and date pickers, that can be customized and reused in an application.

Oracle Application Server

Oracle's integrated application server:

- Is standards compliant (**Java EE**, Web services, and **XML**)
- Delivers a comprehensive set of capabilities, including portal, **caching**, wireless, integration, and personalization
- Provides a single, unified platform for Java and **Java EE**, Web services, XML, SQL, and PL/SQL

Oracle Application Server Portal

A component of **Oracle Application Server** used for the development, deployment, administration, and configuration of enterprise class **portal**s. OracleAS Portal incorporates a portal building framework with self-service publishing features to enable you to create and manage information accessed within your portal.

Oracle Single Sign-On

A component of **Oracle Application Server** that enables users to log in to all features of the Oracle Application Server product suite, as well as to other Web applications, using a single user name and password. OracleAS Portal is integrated with Oracle Single Sign-On as a partner application and delegates authentication to it.

OracleAS Portal

See **Oracle Application Server Portal**.

Oracle Containers for J2EE

See **OC4J**.

Oracle Content Database

A consolidated, database-centric content management application that provides a comprehensive, integrated solution for file and document life cycle management. Oracle Content Database (Oracle Content DB) runs on **Oracle Application Server**, a JCR adapter for accessing content, and Oracle Database. It provides a scalable content management repository. Oracle Content DB also offers a comprehensive set of Web services that developers can use to build and enhance content management applications.

Oracle Drive

A native Windows application that lets users use Windows Explorer, Microsoft Office, and other Windows applications to access content in **Oracle Content Database**, and enables access to **Oracle Application Server Portal**. Oracle Drive displays files and folders in Oracle Content DB as a mapped drive in Windows Explorer. Oracle Drive provides an effective offline solution that lets users edit files on their computers when offline, and then synchronize with the server when they reconnect.

Oracle Enterprise Manager

Oracle Enterprise Manager 10*g* is a component of the **Oracle Application Server** that enables administrators to manage Oracle Application Server services through a single environment.

Oracle HTTP Server

The **Web server** component of **Oracle Application Server**, built on Apache Web server technology and used to service **HTTP** requests. It is the part of the **middle tier** that handles requests between the Web and **Oracle Application Server Portal**. Extensions to the Oracle HTTP Server support Java **servlet**s, **JSP**s, Perl, PL/SQL, and CGI applications.

Oracle Internet Directory

Oracle Internet Directory is Oracle's LDAP V3 compliant LDAP server. It is used by **Oracle Application Server** as the default repository provisioning users and groups.The repository for storing **Oracle Application Server Portal** user credentials and group memberships. By default, the **Oracle Single Sign-On** authenticates user credentials against Oracle Internet Directory information about dispersed users and network resources. Oracle Internet Directory combines LDAP version 3 with the high performance, scalability, robustness, and availability of the Oracle database.

Oracle JDeveloper

Oracle JDeveloper is an integrated development environment (**IDE**) for building applications and Web services using the latest industry standards for Java, XML, and SQL. Developers can use Oracle JDeveloper to create Java portlets.

Oracle Metadata Services

See **MDS**.

Oracle Technology Network

See **OTN**.

Oracle WebCenter Framework

A set of features provided by **Oracle WebCenter Suite** that augments the Java Server Faces (JSF) environment by providing additional integration and run-time customization options It is the basis of the Oracle WebCenter Suite, and supports the creation and execution of context-rich applications, which can come in the form of human interaction, files and documents, or a clear representation of where the user is within a complex work process. It includes such features as:

- Portlet support
- **content integration services**
- **JSF** portlet bridge
- Search Framework

- customizable components

Oracle WebCenter Services

A suite of services included in the **Oracle WebCenter Suite** that enables you to enhance your **Oracle ADF Faces** applications with WebCenter application capabilities, such as portlets, content integration, and customization. Includes design time extensions to **Oracle JDeveloper** to make it easier to build **WebCenter application**s. The services include:

- **Oracle Content Database**
- **Secure Enterprise Search**
- communication services

Oracle WebCenter Suite

A suite of services that enables you to build **WebCenter application**s. Oracle WebCenter Suite reduces the front-end labor historically required to bring necessary business components to the user by capitalizing on the notion of Service Oriented Architecture (SOA). The suite includes a wide range of plug-and-play products, tools, and services that make it easy to build the applications your users need. Oracle WebCenter Suite includes:

- **Oracle WebCenter Services**
- **Oracle WebCenter Framework**
- **content integration services**
- **ADF**
- **Secure Enterprise Search**
- Mobile Services
- Portlet Pack

OTN

Oracle Technology Network. The online Oracle technical community that provides a variety of technical resources for building Oracle-based applications. You can access OTN at `http://www.oracle.com/technology/`.

page parameter

A parameter that enables your page to take values through its URL. Page parameters are defined using the `<parameter>` tag at the top of your `PageDef.xml`. You can bind page parameters to your **page variable**s.

page variable

A variable that binds your public portlet parameter to the page. Page variables are defined within the `<variableIterator>` of your `PageDef.xml`. One page variable can be bound to multiple public portlet parameters.

personalization

Users' adjustments to their own personal views of a portlet instance.

PDK

See **PDK-Java**.

PDK-Java

Java Portlet Developer Kit. The development framework used to build and integrate Web content and applications with **Oracle WebCenter Suite**. It includes toolkits, samples, and technical articles that help make portal development simple. You can take existing Java **servlet**s, **JSP**s, **URL**-accessible content and Web services and turn them into **portlet**s. It is typically used by external developers and vendors to create portlets and services.

portal

A common interface (that is, a Web page) that provides a personalized, single point of interaction with Web-based applications and information relevant to individual users or class of users.

Portal Developer Kit

See **PDK-Java**.

Portal Tools extension

An extension available through the Oracle JDeveloper Update wizard that installs a standalone Oracle Containers for J2EE application server (**OC4J**), preconfigured producers, and several prebuilt portlets, including **OmniPortlet**, the **Web Clipping portlet**, and the **Rich Text portlet**.

portlet

A reusable, pluggable Web component that typically displays portions of Web content. Portlets are the fundamental building blocks of a portal page. Using the Portlet Builder, you can easily create your own portlets. OracleAS Portal also provides several ways to build portlets programmatically and to integrate any kind of Web content. Portlets may be implemented using various technologies, such as Java, **JSP**s, Java **servlet**s, PL/SQL, Perl, ASP, and so on. The **PDK-Java** covers the standard-based portlet development options that **Oracle WebCenter Suite** provides.

portlet mode

The ways by which a **portlet** can be called to display information. These methods include:

- **Shared Screen mode** or **View mode**
- **Edit mode** or **Edit Defaults mode**
- **Customize mode**
- **Help mode**
- **About mode**
- **Full Screen mode** or **Show Details Page mode**

Predeployment Tool

A utility for **WebCenter application**s that helps you configure your target system with the new producer registrations you have added to your application in Oracle JDeveloper. You must run this utility before deploying your application. You can also use this utility after deployment to migrate metadata from stage to production, such as for exporting and importing your customizations. This tool also enables you to define the **MDS** repository location to enable run-time customizations to be migrated.

private parameter

A portlet parameter that is known only to the portlet itself and has no connection to the page on which the portlet resides.

Contrast with **public parameter**.

producer

A producer communication link between portlet consumers (such as a **WebCenter application** or a **portal**). When the portal renders a portal page, it calls the producer of each portlet on the page, which in turn executes their portlets and returns the results in the form of portlet content. A producer can contain one or more portlets. A portlet can be contained by only one producer.

Oracle WebCenter Suite supports two types of producers:

- Oracle **PDK-Java** producers: Deployed to a **Java EE** application server, which is often remote and communicates through Simple Object Access Protocol (SOAP) over HTTP.
- **Web Services for Remote Portlets** (WSRP): A Web services standard that enables the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container based on a particular language (such as **JSR 168**, .NET, Perl) and any WSRP portal. A portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard. **Oracle WebCenter Suite** supports a preliminary version of WSRP 2.0.

programmatic portlets

Portlets constructed in a non-declarative manner using **API**s. Also referred to as *hand-* or *manually coded* portlets.

proxy server

A proxy server typically sits on a network firewall and enables clients behind the firewall to access Web resources. All requests from clients go to the proxy server rather than directly to the destination server. The proxy server forwards the request to the destination server and passes the received information back to the client. The proxy server channels all Web traffic at a site through a single, secure port; this enables an organization to create a secure firewall by preventing Internet access to internal computers, while enabling Web access.

public parameter

A portlet parameter that is known to the page and bound to it by way of a page variable.

Contrast with **private parameter**.

public user

A user who can access, but is not logged into, a **WebCenter application**. A public user can view any page that has been marked as public, but cannot personalize or edit any content, or view pages that have any form of access control.

Contrast with **authenticated user**.

Reverse Proxy Server

A server process that hides the physical location of internal servers by exposing the servers as a single public site. Requests to the public site are routed to the appropriate internal server.

Rich Text portlet

A portlet, based on the **WSRP** standard, offering browser-based rich text editing at run time on a deployed Oracle ADF JavaServer Faces JSP. The Rich Text portlet is provided through the **Portal Tools extension**.

Secure Enterprise Search

See **SES**.

service ID

A PDK-Java producer's unique identifier. PDK-Java enables you to deploy multiple producers under a single adapter servlet. Different producers are identified by their unique service IDs. A service ID is required only when a service ID/producer name is not appended to the URL endpoint.

servlet

A Java program that usually runs on a **Web server**, extending the Web server's functionality. **HTTP** servlets take client HTTP requests, generate dynamic content (such as through querying a database), and provide an HTTP response.

SES

Secure Enterprise Search (SES) provides an easy-to-use, Internet-search-like user experience for public and secure sources. Based on crawling agents, the search can include structured and unstructured, public and secure content. Secure Enterprise Search is part of the **Oracle WebCenter Suite**.

Shared Screen mode

A **portlet mode** that renders the body of the portlet and enables you to display a portlet on a page that can contain other portlets. Every portlet must have at least a Shared Screen mode.

See also **View mode**.

Show Details Page mode

A **portlet mode** that provides full-browser display of the portlet. For example, a portlet in **Show Page mode** could be limited to displaying only the ten most recently submitted expense reports, while the same portlet in Show Details Page mode could show all submissions.

Contrast with **Show Page mode**.

Show modes

Types of **portlet mode**s encompassing **Show Page mode** and **Show Details Page mode**.

Show Page mode

A **portlet mode** that provides a smaller portlet display to enable space for additional portlets and other objects in the browser window. For example, a portlet in Show Page mode could be limited to displaying only the ten most recently submitted expense

reports, while the same portlet in Show Details Page mode could show all submissions.

Contrast with **Show Details Page mode**.

struts

A development framework for Java servlet applications based upon the **MVC** design paradigm.

URL

Uniform Resource Locator. A compact string representation of the location for a resource that is available through the Internet. It is also the format Web clients use to encode requests to **Oracle Application Server**.

URL parameter

See **private parameter**.

validation-based caching

A **caching** method that uses a validation check to determine if the cached item is still valid.

Contrast with **expiry-based caching** and **invalidation-based caching**.

View mode

(**JSR 168** portlets only.) A **portlet mode** that enables you to display a JSR 168 portlet on a page that can contain other portlets. It is the only required mode for JSR 168 portlets.

See also **Shared Screen mode**.

WAR

Web application archive file. This file is used in deploying applications on a **Java EE** application server. WAR files encapsulate in a single module all of the components necessary to run an application. WAR files typically contain an application's **servlet**, **JSP**, and **JSF JSP** components.

See also **EAR** and **JAR**.

Web Application Archive file

See **WAR**.

Web clipping

A feature that enables page designers to collect Web content into a single centralized portal. It can be used to consolidate content from hundreds of different Web sites scattered throughout a large organization.

Web Clipping portlet

A browser-based declarative tool that enables you to integrate any Web application with your **WebCenter application**. It is designed to give you quick integration by leveraging the Web application's existing user interface. You can drag and drop Web Clipping portlets on to a `*.jspx` page.

Web server

A program that delivers Web pages.

Web Services for Remote Portlets

See **WSRP**.

WebCenter application

An ADF application that combines Web content, portlets, and collaborative services for the end user. Administrators can customize the **WebCenter application** based on their roles and skill levels in the organization.

WebCenter application administrator

The administrator responsible for maintaining the **WebCenter application**. This administrator performs tasks such as implementing the branding for the WebCenter application, making new content available, modifying pages, and granting and revoking privileges.

WebCenter application developer

The developer who plans, builds, and maintains a **WebCenter application** using the Oracle Application Development Framework, **Oracle JDeveloper**, and the **Oracle WebCenter Suite**.

WebCenter application end user

The WebCenter application end user is the run time user of the **WebCenter application**, who accesses pages, portlets, and content, and personalizes portlets (assuming the appropriate privileges).

WebCenter Extension for Oracle JDeveloper

An extension available through the Oracle JDeveloper Update Wizard that installs the necessary libraries, templates, wizards, and dialogs needed to build and deploy **WebCenter application**s in **Oracle JDeveloper**.

WebCenter Framework

See **Oracle WebCenter Framework**.

WebCenter Services

See **Oracle WebCenter Services**.

WebCenter Suite

See **Oracle WebCenter Suite**.

WSRP

Web Services for Remote Portlets (WSRP) is a Web services standard that enables the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container based on a particular language (such as JSR 168, .NET, Perl) and any WSRP portal. A portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard. **Oracle WebCenter Suite** supports a preliminary version of WSRP 2.0.

XML

Extensible Markup Language (XML) is an open standard for describing data using a subset of the SGML syntax.

XSL

Extensible Stylesheet Language (XSL) is the language used within stylesheets to transform or render **XML** documents.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X