Oracle® Fusion Middleware

Developer's Guide for Oracle Portal

11*g* Release 1 (11.1.1)

E10238-03

January 2011

Oracle Fusion Middleware Developer's Guide for Oracle Portal, 11*g* Release 1 (11.1.1)

E10238-03

Primary Author: Swati Thacker

Contributing Authors: Showvik Roy Chowdhuri, Frank Rovitto, Lalithashree Rajesh, Promila Chitkara, Susan Highmoor, Vanessa Wang, Kumar Dhanagopal

Contributors: Gareth Bryan, Joan Carter, Candace Fender, Tugdual Grall, Helen Grembowicz, Christian Hauser, Peter Henty, Karthik Lakshmanan, Bill Lankenau, Pankaj Mittal, Peter Moskovits, Lei Oh, Jitinder Sethi, Ingrid Snedecor, Julie Tower, Sue Vickers, Alistair Wilson

Contents

Preface

This manual describes how to build portlets for Oracle Portal (Oracle Portal) using a variety of tools and technologies. This manual includes information that helps you understand the various technology choices open to you, choose the technology that best meets your requirements, and use the appropriate tools to build and deploy your portlets.

Intended Audience

This manual is intended primarily for portal developers, but page designers may also find it useful. This manual guides you through the process of first understanding and choosing a portlet technology, and then building your portlets with that technology.

What Is a Portal Developer? A portal developer is a user who writes code to help make a portal meet the specific requirements of an organization. For example, a portal developer may build portlets and make them available to page designers and other users for inclusion on their pages. This type of portal developer is also referred to as a portlet developer. A portal developer may also use the public APIs provided with OracleAS Portal to perform certain portal tasks programmatically, rather than through the product's user interface. A portal developer will generally, although not always, be someone with at least some programming knowledge. The privileges assigned to a portal developer depend on the type of tasks that developer performs.

What Is a Portlet Developer? A portlet developer is a user with the following global privileges: Create All Portal DB Providers and Manage All Shared Components. Since OracleAS Portal offers such a wide spectrum of tools and technologies for building portlets, a portlet developer may or may not have substantial programming background.

> **Note:** You can find information on using Portlet Builder in Appendix A "Creating Portlets with the Portlet Builder" of the *Oracle Fusion Middleware Developer's Guide for Oracle Portal 10g Release 2 (10.1.4)* in the Oracle Application Server 10*g* Release 2 (10.1.2.0.2) library located on the Oracle Technology Network (OTN) (`http://www.oracle.com/technology/documentation/appserver.html`).

What Is a Page Designer? A page designer, also known as a page manager, is a user with the Manage privilege on a page. A user with this privilege can perform any action on the page and can create sub-pages under the page. The page designer is often responsible for designing the layout (or region configuration) of the page and assigning privileges on the page to other users (for example, to determine who can add content to the page).

The scope of a page designer's control over a page may be limited if the page is based on a template.

More On OTN

For information about the different privileges in Oracle Portal and how these affect the tasks you can perform, see the *Oracle Portal User's Guide*.

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.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/support/contact.html` or visit `http://www.oracle.com/accessibility/support.html` if you are hearing impaired.

Related Documents

For more information, see the following manuals in the Oracle Portal documentation set:

- *Oracle Fusion Middleware Release Notes*
- *Oracle Fusion Middleware User's Guide for Oracle Portal*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*

Note: You can find all documentation related to Oracle Portal on the *Oracle Fusion Middleware Documentation Library*.

You may also find the following manuals in the Oracle Application Server documentation set useful:

- *Oracle Fusion Middleware Concepts*
- *Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server*
- *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*
- *Oracle Fusion Middleware PL/SQL Web Toolkit Reference*

Note: You can find documentation related to Oracle Application Server on OTN (`http://www.oracle.com/technology/documentation/index.html`).

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.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
...	Ellipsis points in an example mean that information not directly related to the example has been omitted.
MW_HOME	Represents the full path of the Middleware home.
ORACLE_HOME	Represents the full path of the Oracle home.
ORACLE_INSTANCE	Represents the full path of the instance home associated with ORACLE_HOME.
INFRA_ORACLE_HOME	Represents the full path of the Oracle Application Server Infrastructure Oracle home, and is used where it is necessary to distinguish between the middle tier, Oracle Application Server Infrastructure, or Oracle Metadata Repository.
METADATA_REP_ORACLE_HOME	Represents the full path of the OracleAS Infrastructure home containing the Oracle Metadata Repository, and is used where it is necessary to distinguish between the middle tier, Oracle Application Server Infrastructure, or Oracle Metadata Repository.

Part I

Portlet Overview

Part I contains the following chapters:

1

Understanding Portlets

This chapter provides an overview of portlets and describes, with help of examples, the use of portlets. It explains portlet anatomy, and the resources to create portlets. This chapter contains the following sections:

1.1 Introduction to Portal Development

Oracle Portal enables you to present information from multiple, unrelated data sources in one, organized view. This view, a portal page, can contain one or more components—called *portlets*—that can each get their content from different data sources.

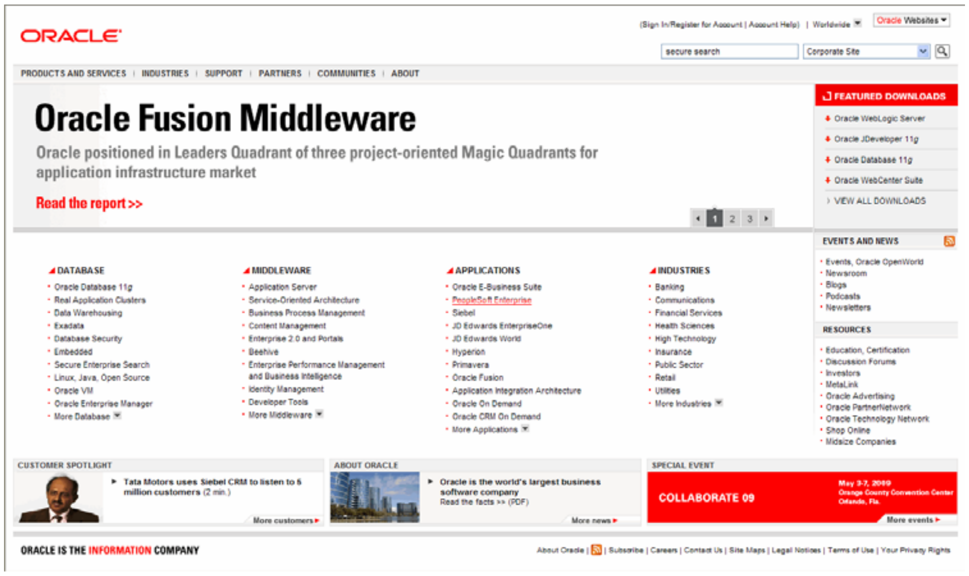
Oracle Portal has all the tools you need for developing portlets and adding them to your portal pages. Oracle Portal's tools support a wide range of development skills: from the novice business developer to the experienced IT programmer. You can develop portlets either declaratively, through the Oracle Portal user interface, or programmatically, through Oracle Portal's collection of application programming interfaces (APIs), known as the Oracle Portal Developer Kit (PDK). Additionally, you can develop portlets through other development tools, external to Oracle Portal, and integrate them through the PDK and an Oracle Portal entity called a *provider*. To learn more about providers, see Chapter 2, "Portlet Technologies Matrix."

This chapter defines portlets, lists and describes some sources for pre-built portlets and resources for building portlets, and suggests the best resource for the job.

1.2 Understanding Portlets

A portlet is a reusable Web component that can draw content from many different sources. A typical portlet is one that displays summaries of Web content, as shown in Figure 1–1.

Figure 1–1 Portlets on the My Oracle Home Page


ORACLE
(Sign In/Register for Account | Account Help) | Worldwide
Oracle Websites
secure search
Corporate Site
PRODUCTS AND SERVICES | INDUSTRIES | SUPPORT | PARTNERS | COMMUNITIES | ABOUT
1 2 3

Oracle Fusion Middleware

Oracle positioned in Leaders Quadrant of three project-oriented Magic Quadrants for application infrastructure market

Read the report >>

FEATURED DOWNLOADS
- Oracle WebLogic Server
- Oracle JDeveloper 11g
- Oracle Database 11g
- Oracle WebCenter Suite
- VIEW ALL DOWNLOADS

DATABASE
- Oracle Database 11g
- Real Application Clusters
- Data Warehousing
- Exadata
- Database Security
- Embedded
- Secure Enterprise Search
- Linux, Java, Open Source
- Oracle VM
- Oracle Enterprise Manager
- More Database

MIDDLEWARE
- Application Server
- Service-Oriented Architecture
- Business Process Management
- Content Management
- Enterprise 2.0 and Portals
- Beehive
- Enterprise Performance Management and Business Intelligence
- Identity Management
- Developer Tools
- More Middleware

APPLICATIONS
- Oracle E-Business Suite
- PeopleSoft Enterprise
- Siebel
- JD Edwards EnterpriseOne
- JD Edwards World
- Hyperion
- Primavera
- Oracle Fusion
- Application Integration Architecture
- Oracle On Demand
- Oracle CRM On Demand
- More Applications

INDUSTRIES
- Banking
- Communications
- Financial Services
- Health Sciences
- High Technology
- Insurance
- Public Sector
- Retail
- Utilities
- More Industries

EVENTS AND NEWS
- Events, Oracle OpenWorld
- Newsroom
- Blogs
- Podcasts
- Newsletters

RESOURCES
- Education, Certification
- Discussion Forums
- Investors
- MetaLink
- Oracle Advertising
- Oracle PartnerNetwork
- Oracle Technology Network
- Shop Online
- Midsize Companies

CUSTOMER SPOTLIGHT
Tata Motors uses Siebel CRM to listen to 5 million customers (2 min.)
More customers

ABOUT ORACLE
Oracle is the world's largest business software company
Read the facts >> (PDF)
More news

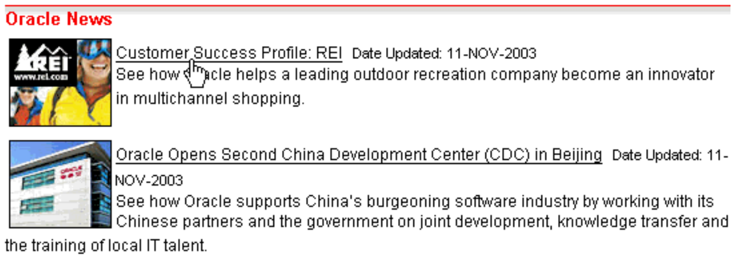
SPECIAL EVENT
COLLABORATE 09
May 3-7, 2009
Orange County Convention Center
Orlando, Fla.
More events

ORACLE IS THE INFORMATION COMPANY

About Oracle | | Subscribe | Careers | Contact Us | Site Maps | Legal Notices | Terms of Use | Your Privacy Rights

For example, in your portal you may have a news feed portlet that supplies linked news article headlines that are each accompanied by a sentence describing the content of the article (Figure 1–2).

Figure 1–2 The Oracle News Portlet on the My Oracle Home Page



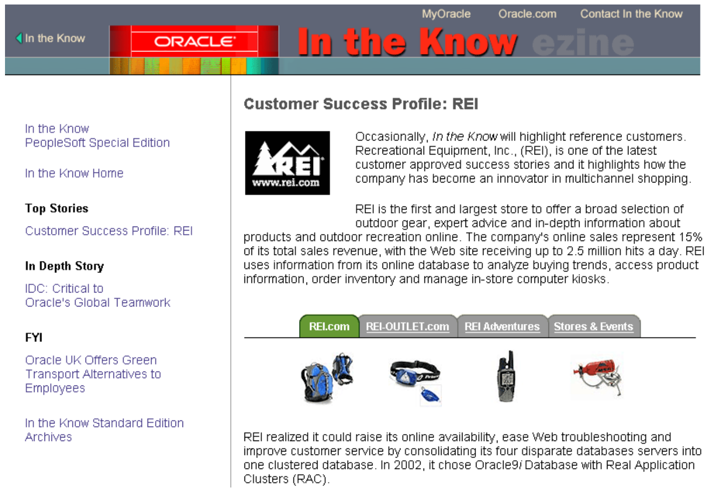
Oracle News

Customer Success Profile: REI Date Updated: 11-NOV-2003
See how Oracle helps a leading outdoor recreation company become an innovator in multichannel shopping.

Oracle Opens Second China Development Center (CDC) in Beijing Date Updated: 11-NOV-2003
See how Oracle supports China's burgeoning software industry by working with its Chinese partners and the government on joint development, knowledge transfer and the training of local IT talent.

Users click the linked headlines to get to the full text of the article, which is hosted on an external news service (Figure 1–3). The portlet has a somewhat dynamic nature in that headlines change automatically as news stories are added and removed at the source.

Figure 1–3 Content Target from a Portlet Link



In the Know
PeopleSoft Special Edition

In the Know Home

Top Stories

Customer Success Profile: REI

In Depth Story

IDC: Critical to
Oracle's Global Teamwork

FYI

Oracle UK Offers Green
Transport Alternatives to
Employees

In the Know Standard Edition
Archives

Customer Success Profile: REI

Occasionally, *In the Know* will highlight reference customers. Recreational Equipment, Inc., (REI), is one of the latest customer approved success stories and it highlights how the company has become an innovator in multichannel shopping.

REI is the first and largest store to offer a broad selection of outdoor gear, expert advice and in-depth information about products and outdoor recreation online. The company's online sales represent 15% of its total sales revenue, with the Web site receiving up to 2.5 million hits a day. REI uses information from its online database to analyze buying trends, access product information, order inventory and manage in-store computer kiosks.

REI realized it could raise its online availability, ease Web troubleshooting and improve customer service by consolidating its four disparate databases servers into one clustered database. In 2002, it chose Oracle9*i* Database with Real Application Clusters (RAC).

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.

1.3 Portlet Anatomy

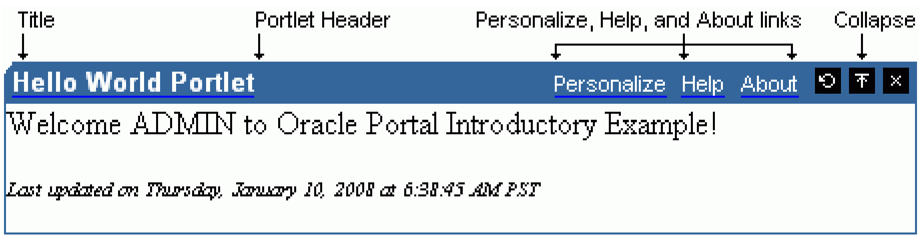
In Oracle Portal a portlet on a page is rendered in an HTML table cell. A portlet can display various types of content, such as HTML, formatted text, images, or elements of an HTML form.

Figure 1–4 illustrates a typical portlet anatomy. This includes a header that contains the portlet title. You can create a hyperlink in the portlet title, so that when a user clicks the title, the portlet displays in a full browser page. A portlet can also include a border, to distinguish the layout from other portlets on the page.

You, the portlet developer, can expose links such as Personalize, Help, and About, to the page designer, who can then turn them on or off. Clicking the **Personalize** link displays a number of options where the end user can personalize various attributes of the portlet. Clicking the **Help** link displays a window containing help text that you can create to assist the end user with the portlet. Clicking the **About** link displays a window that you can create to describe the contents of the portlet.

Each portlet also contains the refresh icon and a standard collapse icon, which the end user can click to collapse or expand the portlet on the page.

Figure 1–4 Portlet Anatomy



Welcome ADMIN to Oracle Portal Introductory Example!

Last updated on Thursday, January 10, 2008 at 6:38:45 AM PST

1.4 Portlet Resources

Portlet resources include the many prebuilt portlets available out of the box from many sources, including Oracle Portal, Oracle E-Business Suite, and third-party sources. Portlet resources also include portlet-building tools available through the Oracle Portal user interface as well as from the PDK and other Oracle 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 1.4.1, "Out-of-the-Box Portlets"
- Section 1.4.2, "Other Sources of Prebuilt Portlets"
- Section 1.4.3, "Web Clipping"
- Section 1.4.4, "OmniPortlet"
- Section 1.4.5, "Portlet Builder"
- Section 1.4.7, "Programmatic Portlets"

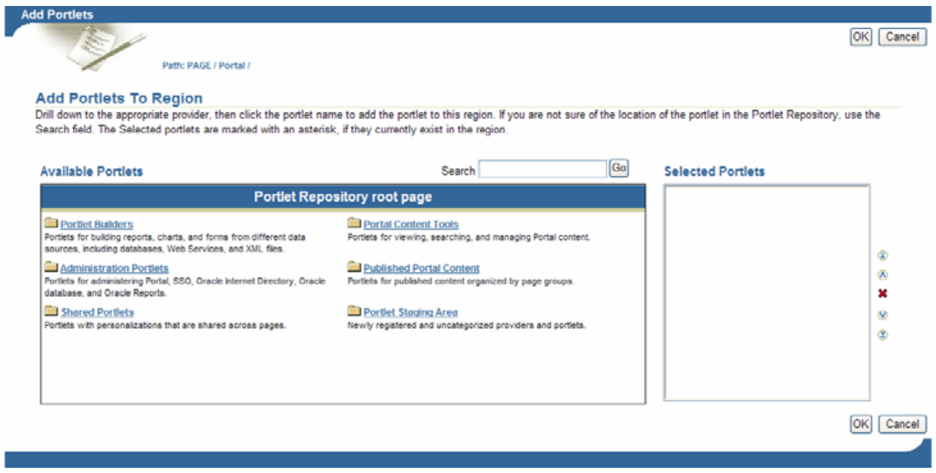
This section introduces you to the various portlet resources. For specific information on each tool and its benefits, see Chapter 2, "Portlet Technologies Matrix."

1.4.1 Out-of-the-Box Portlets

What Are They?

Out-of-the-box portlets are prebuilt, fully developed, registered portlets that are immediately available from the Portlet Repository when you install Oracle Portal (Figure 1–5). They include such portlets as Search, Saved Searches, Favorites, and My Notifications.

Figure 1–5 The Portlet Repository



Add Portlets

Path: PAGE / Portal /

Add Portlets To Region

Drill down to the appropriate provider, then click the portlet name to add the portlet to this region. If you are not sure of the location of the portlet in the Portlet Repository, use the Search field. The Selected portlets are marked with an asterisk, if they currently exist in the region.

Available Portlets

Search

Selected Portlets

Portlet Repository root page

Portlet Builders
Portlets for building reports, charts, and forms from different data sources, including databases, Web Services, and XML files.

Portal Content Tools
Portlets for viewing, searching, and managing Portal content.

Administration Portlets
Portlets for administering Portal, SSO, Oracle Internet Directory, Oracle database, and Oracle Reports.

Published Portal Content
Portlets for published content organized by page groups.

Shared Portlets
Portlets with personalizations that are shared across pages.

Portlet Staging Area
Newly registered and uncategorized providers and portlets.

You'll find information on the prebuilt portlets in Oracle Portal in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

Who Is the Intended User?

Out-of-the-box portlets are best suited for use by end users and page designers, though they are available to users at all levels of expertise.

When Should They Be Used?

Use out-of-the-box 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, such as when you need a different user interface, or when the functionality you require is not available out of the box.

For more information on when you should use each technology, see Chapter 2, "Portlet Technologies Matrix."

1.4.2 Other Sources of Prebuilt Portlets

What Are They?

Other sources of prebuilt portlets include partner portlets and integration solutions.

More On OTN

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:

`http://solutions.oracle.com`

Examples of these prebuilt portlets include portlets for the following purposes:

- Generating point-to-point driving directions
- Accessing IT information from a wide variety of sources
- Viewing summary information on news, stocks, and weather

More On OTN

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 Portal Integration Solutions page on OTN:

`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 end users and page designers who understand how to download, install, and register Web and database providers in Oracle Portal. They are available for use by all levels of experience.

When Should They Be Used?

As with out-of-the-box portlets, use prebuilt portlets from other sources 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.

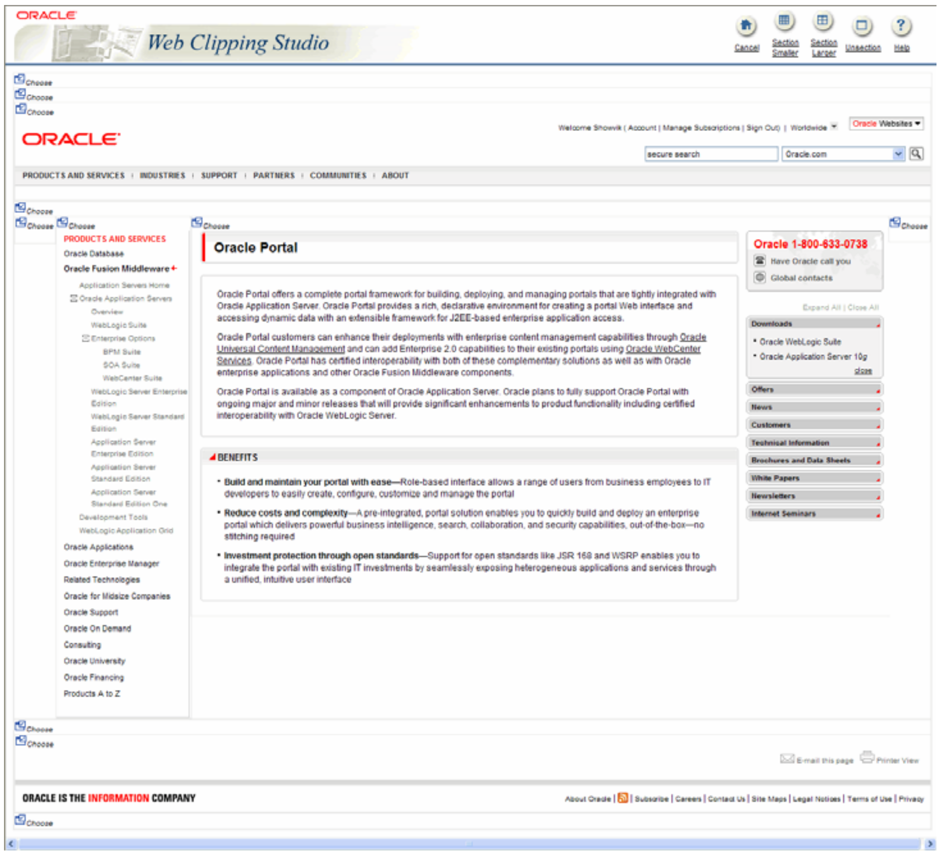
1.4.3 Web Clipping

What Is It?

Web Clipping is a browser-based declarative tool that enables you to integrate any Web application with Oracle Portal. It is designed to give you quick integration by leveraging the Web application's existing user interface. Web Clipping has been implemented as a Web provider using the PDK-Java, which is a component of Oracle Portal.

To create a Web Clipping portlet, the portal page designer uses a Web browser to navigate to the Web page that contains the desired content. Through the Web Clipping Studio, the page designer can drill down through a visual rendering of the target page to choose the desired the content (Figure 1–6 and Figure 1–7).

Figure 1–6 Selecting Web Content through the Web Clipping Studio


ORACLE
Web Clipping Studio
Cancel
Section Smaller
Section Larger
Unsection
Help
Choose
Welcome Showvik (Account | Manage Subscriptions | Sign Out) | Worldwide
Oracle Websites
secure search
Oracle.com
PRODUCTS AND SERVICES | INDUSTRIES | SUPPORT | PARTNERS | COMMUNITIES | ABOUT
Oracle 1-800-633-0738
Have Oracle call you
Global contacts
Expand All | Close All
E-mail this page
Printer View
ORACLE IS THE INFORMATION COMPANY
About Oracle | Subscribe | Careers | Contact Us | Site Maps | Legal Notices | Terms of Use | Privacy

PRODUCTS AND SERVICES

- Oracle Database
- Oracle Fusion Middleware +
  - Application Servers Home
  - Oracle Application Servers
    - Overview
    - WebLogic Suite
    - Enterprise Options
      - BPM Suite
      - SOA Suite
      - WebCenter Suite
    - WebLogic Server Enterprise Edition
    - WebLogic Server Standard Edition
    - Application Server Enterprise Edition
    - Application Server Standard Edition
    - Application Server Standard Edition One
  - Development Tools
  - WebLogic Application Grid
- Oracle Applications
- Oracle Enterprise Manager
- Related Technologies
- Oracle for Midsize Companies
- Oracle Support
- Oracle On Demand
- Consulting
- Oracle University
- Oracle Financing
- Products A to Z

Oracle Portal

Oracle Portal offers a complete portal framework for building, deploying, and managing portals that are tightly integrated with Oracle Application Server. Oracle Portal provides a rich, declarative environment for creating a portal Web interface and accessing dynamic data with an extensible framework for J2EE-based enterprise application access.

Oracle Portal customers can enhance their deployments with enterprise content management capabilities through Oracle Universal Content Management and can add Enterprise 2.0 capabilities to their existing portals using Oracle WebCenter Services. Oracle Portal has certified interoperability with both of these complementary solutions as well as with Oracle enterprise applications and other Oracle Fusion Middleware components.

Oracle Portal is available as a component of Oracle Application Server. Oracle plans to fully support Oracle Portal with ongoing major and minor releases that will provide significant enhancements to product functionality including certified interoperability with Oracle WebLogic Server.

BENEFITS

- **Build and maintain your portal with ease**—Role-based interface allows a range of users from business employees to IT developers to easily create, configure, customize and manage the portal
- **Reduce costs and complexity**—A pre-integrated, portal solution enables you to quickly build and deploy an enterprise portal which delivers powerful business intelligence, search, collaboration, and security capabilities, out-of-the-box—no stitching required
- **Investment protection through open standards**—Support for open standards like JSR 168 and WSRP enables you to integrate the portal with existing IT investments by seamlessly exposing heterogeneous applications and services through a unified, intuitive user interface

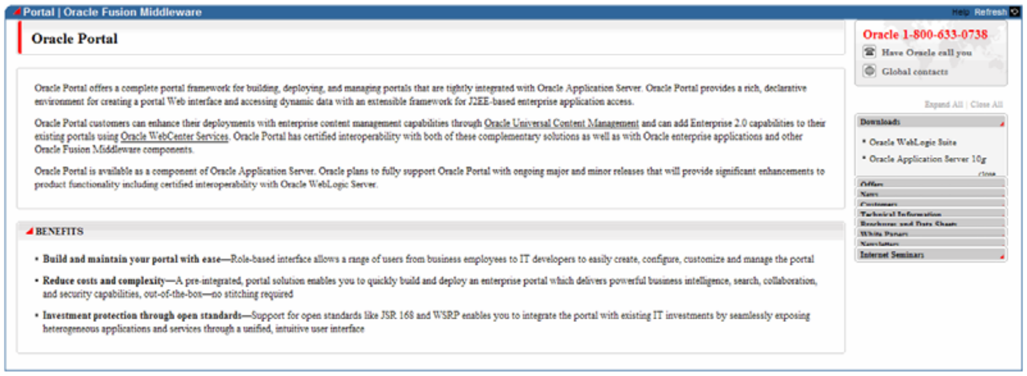
Downloads

- Oracle WebLogic Suite
- Oracle Application Server 10g

close

Offers
News
Customers
Technical Information
Brochures and Data Sheets
White Papers
Newsletters
Internet Seminars

Figure 1–7 Clipped Content Rendered as a Portlet in Portal



Oracle Portal

Oracle Portal offers a complete portal framework for building, deploying, and managing portals that are tightly integrated with Oracle Application Server. Oracle Portal provides a rich, declarative environment for creating a portal Web interface and accessing dynamic data with an extensible framework for J2EE-based enterprise application access.

Oracle Portal customers can enhance their deployments with enterprise content management capabilities through Oracle Universal Content Management and can add Enterprise 2.0 capabilities to their existing portals using Oracle WebCenter Services. Oracle Portal has certified interoperability with both of these complementary solutions as well as with Oracle enterprise applications and other Oracle Fusion Middleware components.

Oracle Portal is available as a component of Oracle Application Server. Oracle plans to fully support Oracle Portal with ongoing major and minor releases that will provide significant enhancements to product functionality including certified interoperability with Oracle WebLogic Server.

BENEFITS

- **Build and maintain your portal with ease**—Role-based interface allows a range of users from business employees to IT developers to easily create, configure, customize and manage the portal
- **Reduce costs and complexity**—A pre-integrated, portal solution enables you to quickly build and deploy an enterprise portal which delivers powerful business intelligence, search, collaboration, and security capabilities, out-of-the-box—no stitching required
- **Investment protection through open standards**—Support for open standards like JSR 168 and WSRP enables you to integrate the portal with existing IT investments by seamlessly exposing heterogeneous applications and services through a unified, intuitive user interface

Oracle 1-800-633-0738

Have Oracle call you

Global contacts

- Oracle WebLogic Suite
- Oracle Application Server 10g

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, 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,** allowing a page designer 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 Oracle Portal 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 Application Server 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 per-user authentication. 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 Tunnelling** 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.
- **Migration from URL-based portlets,** enabling you to migrate your URL-based portlets to Web Clipping.

Who Is the Intended User?

Web Clipping is best suited for use by page designers and portlet developers who want to leverage an existing Web page for rapid portlet development. The Web Clipping portlet is accessible out of the box, and is available in the Portlet Repository of Oracle Portal. 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 portal as a portlet. Consider alternatives if you want to change the way information is presented in the clipped portlet. That is, you don't need to control the user interface 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.

The following are some examples of when you might 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.
- **Personalized weather portlet.** You want to create a portlet that displays weather information from a major Internet weather site, and you want your users to be able to personalize the portlet by providing the desired zip code.
- **Web mail portlet.** Your users want to access their confidential Web mail accounts through a portlet and display their in-boxes in the portlet.

For more information on using Web Clipping, see Chapter 5, "Creating Content-Based Portlets with Web Clipping."

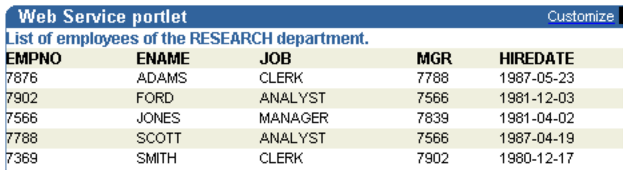
Note: To use the Web Clipping portlet, OmniPortlet, or the Simple Parameter Form with Windows (2000 with SP4, 2003 (32 bit) with SP2/R2 or later, XP with SP2/R2 or later, Vista, and Server 2008), you must use Firefox 3.0 or later, or Microsoft Internet Explorer 7 or later.

1.4.4 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 that HTML. Figure 1–8 shows an OmniPortlet with a tabular format.

Figure 1–8 An OmniPortlet Using Tabular Format



List of employees of the RESEARCH department.

EMPNO	ENAME	JOB	MGR	HIREDATE
7876	ADAMS	CLERK	7788	1987-05-23
7902	FORD	ANALYST	7566	1981-12-03
7566	JONES	MANAGER	7839	1981-04-02
7788	SCOTT	ANALYST	7566	1987-04-19
7369	SMITH	CLERK	7902	1980-12-17

Note: The SAP data source is not included with Oracle Portal. 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`

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.

More On OTN

You'll find information about OmniPortlet on Portal Center. Navigate to the following URL, then click **Portlet Development:**

`http://portalcenter.oracle.com`

Who Is the Intended User?

OmniPortlet is best suited for use by business users with a minimum knowlegde of the URLs to their targeted data.

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 might 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 the 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 on 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.

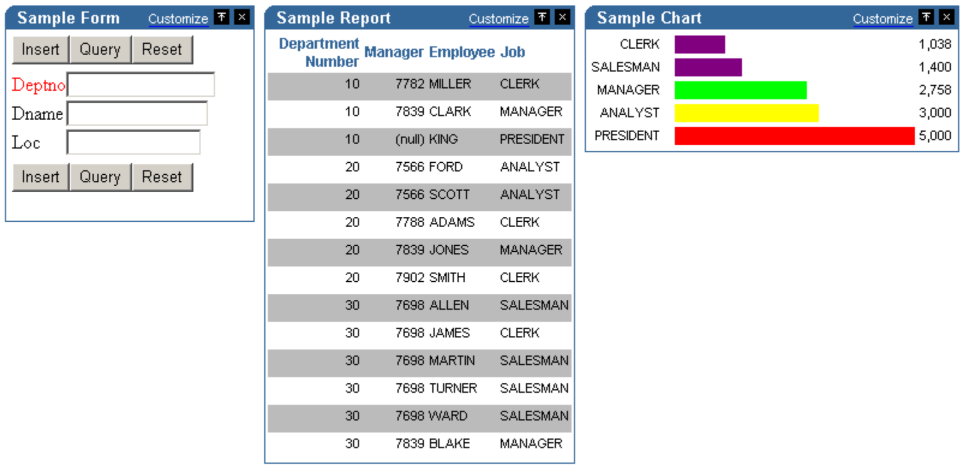
For more information about OmniPortlet, see Chapter 3, "Creating Portlets with OmniPortlet" and Chapter 4, "Building Example Portlets with OmniPortlet."

1.4.5 Portlet Builder

What Is It?

Oracle Portal includes a number of portlet-building wizards that are accessible through the Provider tab in the Portal Navigator. These wizards can be used to build charts, reports, forms, calendars, and lists of values as shown in Figure 1–9.

Figure 1–9 Sample Form, Report, and Chart from the Portlet Builder



Department Number	Manager	Employee	Job
10	7782	MILLER	CLERK
10	7839	CLARK	MANAGER
10	(null)	KING	PRESIDENT
20	7566	FORD	ANALYST
20	7566	SCOTT	ANALYST
20	7788	ADAMS	CLERK
20	7839	JONES	MANAGER
20	7902	SMITH	CLERK
30	7698	ALLEN	SALESMAN
30	7698	JAMES	CLERK
30	7698	MARTIN	SALESMAN
30	7698	TURNER	SALESMAN
30	7698	WARD	SALESMAN
30	7839	BLAKE	MANAGER

CLERK	1,038
SALESMAN	1,400
MANAGER	2,758
ANALYST	3,000
PRESIDENT	5,000

When Should It Be Used?

It is recommended that you use OmniPortlet as an alternative to Portlet Builder whenever possible. OmniPortlet provides more flexibility and a separation of data and layout which enables you to change from a report to chart without re-creating the entire portlet (as is required with Portlet Builder). OmniPortlet also provides more options for deployment to many different portals simultaneously. Oracle Portal will continue to support Portlet Builder as a portlet building option. However, new features and enhancements will be directed toward the OmniPortlet tool.

For more information about OmniPortlet, see Section 1.4.4, "OmniPortlet." For more information about Portlet Builder, see Appendix A, "Creating Portlets with the Portlet Builder."

1.4.6 JSF Portlets

What Are They?

JSF portlets are created using the JSF Portlet Bridge. The JSF Portlet Bridge enables application developers to expose their existing JSF applications and task flows as JSR 168 portlets. The portlet bridge simplifies the integration of JSF applications with WSRP portlet consumers, such as Oracle Portal.

JSF portlets do not require separate source code from that of the JSF application. Since these portlets are created using the portlet bridge, you need to only maintain one source for both your application and your portlets. Similarly, when you deploy your JSF application, JSF portlets are also deployed with it. Therefore, using the bridge eliminates the need to store, maintain, and deploy your portlets separately from your application.

Note: JSF Portlet Bridge integrate Oracle WebCenter Services into Oracle Portal. It is only available if you are using a licensed Oracle WebCentre.

Who Is the Intended User?
Application developers with knowledge of Faces and WSRP.

When Should They Be Used?
JSF portlets are best suited when application developers intend to display contents from a JSF application as a portlet without hosting the entire application, or without separately building a portlet for the same. Once portletized, the consumption of the portlet is the same as registering any WSRP provider using their provider URLs.

For more information, see *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Suite*.

1.4.7 Programmatic Portlets

What Are They?
Programmatic portlets are portlets that you write yourself, in Java or PL/SQL. The Oracle PDK contains a set of portlet-building APIs that you can use to create programmatic portlets.

More On OTN

You'll find more information about these APIs on Portal Center:

`http://portalcenter.oracle.com`

Note: The PDK-PL/SQL is not described in detail in this manual. For specific information on the PDK-PL/SQL, refer to the Developer Services area on Portal Center:

`http://portalcenter.oracle.com`

Who Is the Intended User?
These tools are best used by experienced and knowledgeable IT developers.

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:

- You're building a portlet from the start and need complete control over all of its functionality.
- You know Java or PL/SQL.
- You are comfortable with the PDK and the configuration of Oracle Portal providers.

Consider using this approach when the out-of-the-box declarative tools do not address your needs.

The following are some examples of when you might consider using Java portlets created with the Oracle Portal Developer Kit:

- **Discussion forum portlet.** You want to create a portlet that integrates your company's JSP-based discussion forum application with Oracle Portal. The discussion forum posts are stored in a relational database. The portlet must also follow the strict look and feel of your company's Internet Web site.
- **E-mail portlet.** You want to create a portlet that enables users to send e-mail from the company's intranet portal. You must integrate the e-mail portlet with the company's LDAP server so that the users can use the address book on the LDAP server.

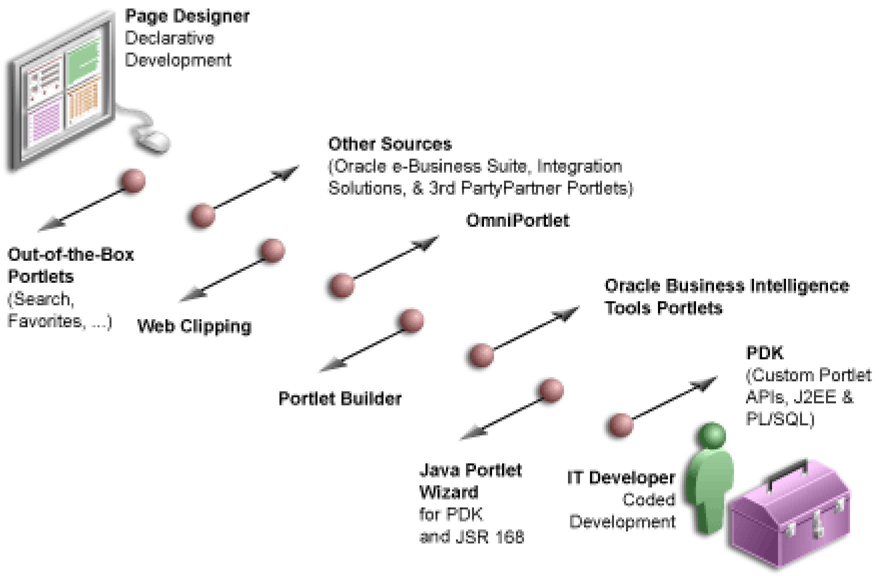
For more information about using the PDK-Java, see Chapter 6, "Creating Java Portlets" and Chapter 7, "Enhancing Java Portlets." For more information about using the PDK-PL/SQL, see Chapter 8, "Creating PL/SQL Portlets."

1.4.8 Deciding Which Tool to Use

Figure 1–10 illustrates the spectrum of portlet resources described in the previous section. Notice how one end of the spectrum is geared toward a more declarative environment as required by page designers while the other end focuses more on hand-coding and portlet developers. You can choose your tool depending on which type of environment is most comfortable and suitable for your skill-base.

For more information on deciding which tool to use, refer to Chapter 2, "Portlet Technologies Matrix".

Figure 1–10 Portlet Resources from Page Designers to Experienced Developers



2 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:

2.1 The Portlet Technologies Matrix

Table 2–1, " Portlet Building Technologies Comparison Matrix" summarizes the technologies and tools you can use with Oracle Portal on one axis, and the features and characteristics on the other. The matrix describes the tools and technologies that are covered in more detail in this guide: OmniPortlet, Web Clipping, the Java Portlets (PDK-Java) including Standards, Portlet Builder as an appendix, and PL/SQL Portlets (PDK-PL/SQL) (in the matrix only).

Note: While these are the primary tools for building portlets, additional tools and technologies exist, such as other Oracle products, including Oracle Reports and Oracle Business Intelligence Discoverer. These other tools are not covered in this guide.

The other sections in this chapter provide further detail on the characteristics listed in Table 2–1. Use the table to quickly scan all the features and characteristics, then see the subsequent sections for more in-depth information.

Table 2–1 Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	PDK-Java	Standards	Portlet Builder	PDK-PL/SQL
General Suitability					
A simple wizard-based tool that helps you retrieve and present Web content, originating from other Web sites, in your portal.	Wizard-based tool, accessible from the browser. Capable of retrieving and presenting data from a wide variety of data sources.	APIs for portlets built specifically for Oracle Portal.	Portlets that should work with portals of other vendors. Oracle supports both WSRP and JSR-168.	Wizard-based tool, accessible from the browser. Best suited for simple, DB-centric applications or portlets.	APIs for portlets built specifically for Oracle Portal.
Expertise Required					
No expertise required.	Basic understanding of one or more supported data sources and the concepts of portlet and page parameters and events.	Java, Servlet, JSP knowledge.	Java, Servlet, JSP knowledge.	Basic understanding of relational DB concepts. Optionally SQL, PL/SQL.	SQL, PL/SQL, PL/SQL Web Toolkit.
Supported Data Sources (for details, see Section 2.3, "Expertise Required")					
Any Web site accessible on the network over HTTP or HTTPS.	CSV, XML, Web Service, SAP, SQL, Web site, JCA.	No limitations.	No limitations.	SQL (local DB or remote DB through DB link)	SQL (local DB or remote DB through DB link)
Deployment Type					
Web provider	Web provider	Web provider	WSRP	Database provider	Database provider
Caching Style					
Expiry-based caching, invalidation-based caching (auto invalidate when personalized).	Expiry-based caching, invalidation-based caching (auto invalidate when personalized).	Expiry-based, validation, and invalidation caching, ESI.	Validation and expiry-based caching.	Expiry-based caching.	Expiry-based, validation, and invalidation caching.
Development Tool					
Browser - wizard.	Browser - wizard.	Oracle JDeveloper - Java Portlet Wizard (or any other Java development environment - without the Wizard).	Oracle JDeveloper - Java Portlet Wizard (or any other Java development environment - without the Wizard).	Browser - optionally PL/SQL development environment.	PL/SQL development environment.
Portlet Creation Style					

Table 2–1 (Cont.) Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	PDK-Java	Standards	Portlet Builder	PDK-PL/SQL
Develop in place.	Develop in place.	No in-place portlet building experience. Add portlet to page, edit defaults, and personalize.	No in-place portlet building experience. Add portlet to page, edit defaults, and personalize.	Develop first, then add and develop in place.	No in-place portlet building experience. Add portlet to page, edit defaults, and personalize.
User Interface Flexibility					
N/A	Very flexible, by using the HTML layout.	Very flexible.	Very flexible.	Limited.	Very flexible.
Ability to Capture Content from Web Sites					
Yes, by its nature.	Yes, by using the Web Data Source.	Yes, by using the java.net package.	Yes, by using the java.net package.	No	Yes, by using the UTIL_HTTP package.
Ability to Render Content Inline					
Yes. (Supported by Web Clipping 9.0.4.0.2 or later.)	No URL rewriting support, but can be achieved by using public portlet parameters and events.	By using private portlet parameters.	Include servlets and JSPs (using the PortletContext.getRequestDispatcher() method).	Pagination in reports and charts is rendered inline.	By using private portlet parameters.
Charting Capability					
N/A	Yes, 2D-3D charts.	By using BI Beans.	By using BI Beans.	HTML charts.	Programmatically, HTML charts.
Public Portlet Parameters Support					
Yes. (Supported by Web Clipping 9.0.4.0.2 or later.)	Yes	Yes	No	Yes	Yes
Private Portlet Parameter Support					
N/A	N/A	Yes	Yes	No	Yes
Event Support					
Yes	Yes	Yes	Portlet private events (actions).	No	No
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 UI.	No, though it is possible to apply security managers that are not exposed through the UI.	Yes, by using the Security managers.	Yes, the Servlet security model is supported by using methods such as PortletRequest.isUserInRole() and PortletRequest.getUserPrincipal().	Yes	Yes, by using the Security APIs.
Multilingual Support					
N/A	Yes	Yes	Yes	No	Yes

Table 2–1 (Cont.) Portlet Building Technologies Comparison Matrix

Web Clipping	OmniPortlet	PDK-Java	Standards	Portlet Builder	PDK-PL/SQL
Pagination Support					
N/A	No	Programmatically	Programmatically	Yes	Programmatically
Single Sign-On and External Application Integration					
Web Clipping 9.0.4.0.2 and higher supports external application integration.	Basic authentication support if the data source requires it.	External application integration supported. LDAP integration is supported when the portlet is running behind the same firewall as the LDAP server.	No. (Feasible through custom user attributes.) LDAP integration is supported.	No. (It runs in the Oracle Portal repository; it does not require SSO integration.)	SSO is enabled by using mod_oso.

2.2 General Suitability

This section describes each portlet-building technology in terms of its usage characteristics (for example, wizard-based or programmatic).

2.2.1 Web Clipping

Web Clipping is a simple wizard-based tool that helps you retrieve and present Web content, originating from other Web sites, in your portal. Web Clipping does not require you to have any technical background.

Examples of portlets you can build using Web Clipping

You can build the following portlets using Web Clipping:

- Stock chart portlet
- Personalized weather portlet
- Web mail portlet

2.2.2 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 portal page, click the Define link, and choose a data source and presentation format. Select from a wide variety of data sources including the following:

- Spreadsheet
- SQL
- XML
- Web Service
- Web page

OmniPortlet does not require you to use an additional development tool or have a strong technical background. Even so, you can use it to build reusable and high-performing portlets.

Examples of portlets you can create with OmniPortlet

You can create the following portlets using OmniPortlet:

- RSS news feed portlet
- Sales chart portlet
- SAP Business Suite portlet

2.2.3 Java Portlets

If the wizard-based portlet building tools do not satisfy your needs, 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, an Oracle JDeveloper plug-in, helps you get started with your 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

You can build the following portlets in Java:

- Discussion forum portlet
- E-mail portlet

2.2.4 Portlet Builder

Portlet Builder is a wizard-based tool to create data-driven portlets, where the data resides in an Oracle database. You can build interactive forms to insert, update, and delete database records. You can create flexible reports and HTML bar charts to display information from the database. Portlet Builder also enables you to pass parameters and navigate between your data-driven portlets by using dynamic links.

Examples of portlets you can build using the Portlet Builder

You can build the following portlets using Portlet Builder:

- Data entry portlet
- Dynamic list of partners portlet
- Sales results portlet

2.2.5 PL/SQL Portlets

Similar to Java portlets, PL/SQL portlets provide a flexible approach to build Web applications that cannot be satisfied by built-in portlets. For example, your application may require implementation of special business rules or logic or meet custom-designed authorization requirements. PL/SQL portlets are commonly used when you need to perform data intensive operations by using SQL and PL/SQL. Oracle Portal offers a rich set of PL/SQL APIs, such as programmatic provider registration, object level privilege management, user interface control, or multilingual support.

For example, any information provider can create custom portlets to display an application to users through Oracle Portal. Developers simply build their portlets according to Oracle Portal Developer Kit (PDK) specifications and register the provider with Oracle Portal. Developers can use the Oracle PDK to develop portlets to suit their needs.

Examples of portlets you can build using PL/SQL

You can build the following portlets using PL/SQL:

- Content upload portlet
- Site map portlet
- Sophisticated data entry and report portlet

2.3 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.

2.3.1 Web Clipping

Web Clipping is a tool that does not require any technical background at all. However, if you want to parameterize the Web page content that you clipped, you need to have an understanding of public portlet parameters and page parameters.

2.3.2 OmniPortlet

OmniPortlet requires you to have basic knowledge of the data source you want to leverage in your portlet. Table 2–2 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 2–2 OmniPortlet Data Sources

Data Source	Required Information
Spreadsheet	The URL that points to the spreadsheet containing the data that you want 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).

2.3.3 Java 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).

2.3.4 Portlet Builder

If you want to use Portlet Builder, you must have a good understanding of relational database concepts. Depending on what you want to achieve, SQL and PL/SQL knowledge may be required, as well. Using Portlet Builder, you can consume data from the local (Oracle Application Server infrastructure) database or remote databases using database links.

2.3.5 PL/SQL Portlets

To build PL/SQL portlets, you must know how to write SQL statements, code and debug PL/SQL program units using SQL*Plus or similar development tool that enables you to connect to the Oracle database. You should also know HTML and PL/SQL Web Toolkit to generate the portlet content. Experience of coding the PL/SQL Server Pages (PSP) is optional.

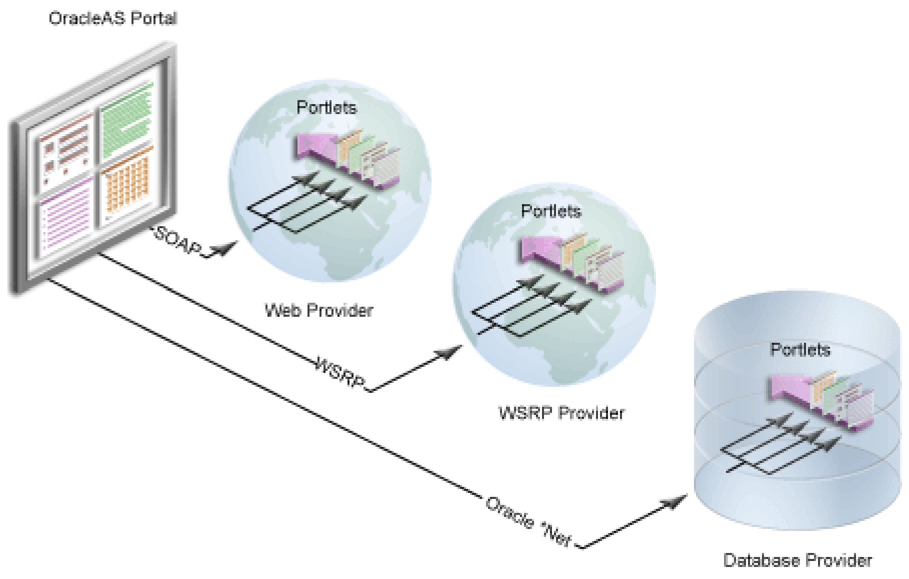
2.4 Deployment Type

Before a portlet can be consumed by an application, you must first deploy it, then register the provider you have deployed the portlet to. As shown in Figure 2–1, portlets can be deployed to Oracle Portal through three provider types:

- Web providers
- WSRP producers
- Database providers.

Web providers are deployed to a J2EE application server, which is often remote and communicates with Oracle Portal through Simple Object Access Protocol (SOAP) over HTTP. Web Services for Remote Portlets (WSRP), an OASIS standard, is supported in the Developer's Preview of Oracle Portal. Database providers are implemented in PL/SQL and deployed in the Oracle database where Oracle Portal is installed.

Figure 2–1 Portlet Provider Overview

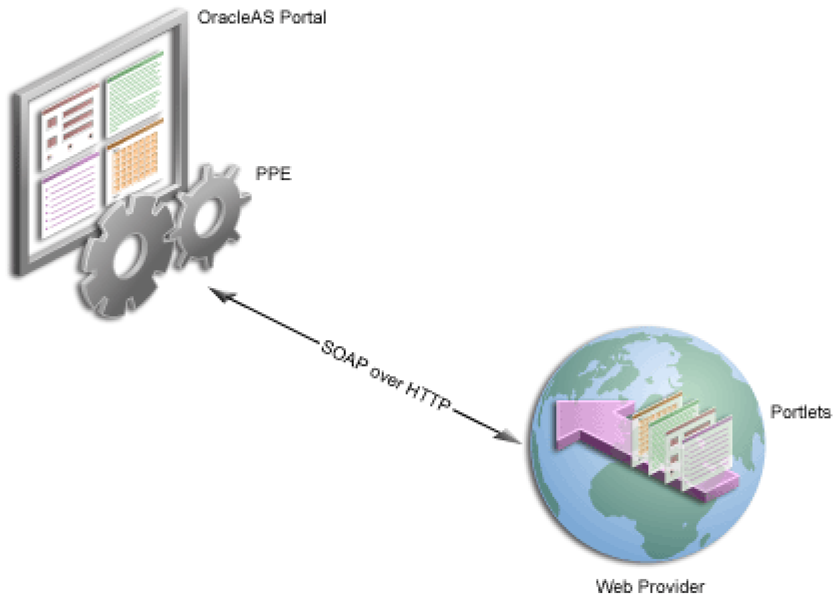


2.4.1 Web Providers

Web providers are the most commonly used and flexible type of provider. They may reside on the same application server as Oracle Portal, on a remote application server, or anywhere on the network (Figure 2–2). A Web provider could be implemented using virtually any Web technology. However, the Oracle Portal Developer Kit provides a Java framework that simplifies the task of building Web providers.

Web providers use open standards, such as XML, SOAP, HTTP, or J2EE for deployment, definition, and communication with Oracle Portal. Also, because Web providers can be deployed to a J2EE container, they do not put an additional load on the Oracle Portal Repository database.

Figure 2–2 Web Providers



There are several benefits when developing portlets and exposing them as Web providers. You can perform the following tasks:

- Deploy portlets remotely.
- Leverage existing Web application code to create portlets.
- Specify providers declaratively.
- Take advantage of more functionality than that with database providers.
- Use standard Java technologies (for example, servlets and JSPs) to develop portlets of Web providers.

To expose your portlets using a Web provider, you must create a provider that manages your portlets and can communicate with Oracle Portal using SOAP. To learn how to expose your portlets using a Web provider, see Section 6.3, "Building JPS-Compliant Portlets with Oracle JDeveloper."

2.4.2 WSRP Producers

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. So, a portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard. From an architecture perspective, WSRP producers are very similar to Web providers. For more information on the WSRP portal architecture, see "The Relationship Between WSRP and JPS".

To expose your portlets as a WSRP producer, you must create a producer that manages your portlets. To learn more about WSRP, see the WSRP and JSR 168 Standards page on the Oracle Technology Network. To learn how to expose your portlets as a WSRP producer, see Section 6.3.3, "Deploying Your JSR 168 Portlet to the Oracle WebLogic Server." You can also test your WSRP producers online using the Oracle Portal Verification Service:

http://portalstandards.oracle.com/portal/page/portal/OracleHostedWSRPPortal/Welcome

By default, the Portal session store maintains the user's profile information, which is obtained from Oracle Internet Directory, for the duration of a Portal session.

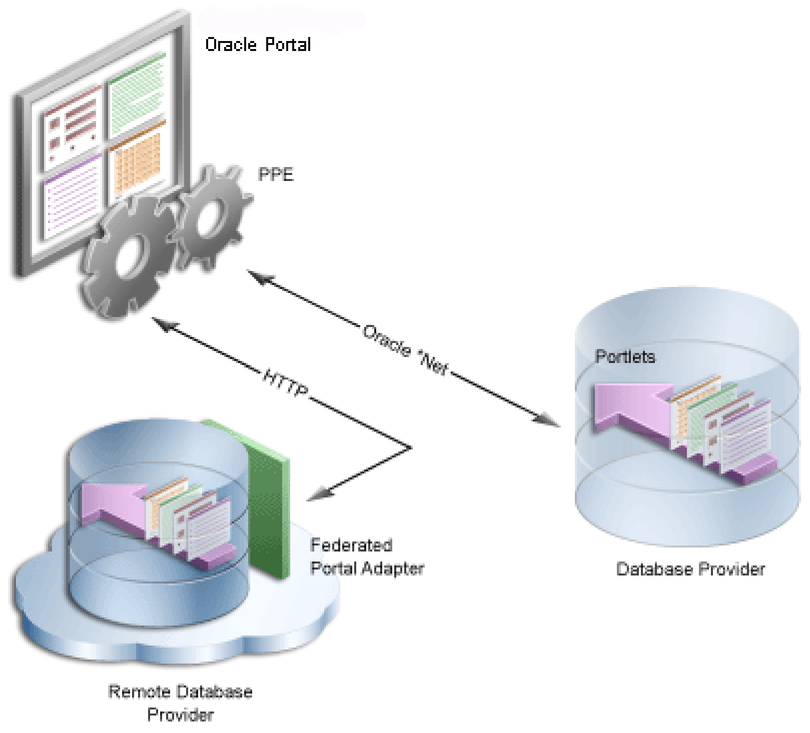
However, in some scenarios, WSRP producers require updates to the WSRP UserContext information sent to them. This information changes during the course of the session. In this scenario, to clear out the cached user information and obtain new information from Oracle Internet Directory when the target page is generated, the application or portlet producer redirects the browser to the following URL:

```
http(s)://server.domain.com:port/portal/pls/portal/portal.wwsec_app_user_
info.flush_cache?p_requested_url=<url-encoded-target-page>
```

2.4.3 Database Providers

You can also create a database provider that owns one or more PL/SQL portlets. Database providers and their PL/SQL portlets reside in the Oracle Metadata Repository database and are implemented as PL/SQL packages. To access database providers on remote servers, you can use the Federated Portal Adapter (Figure 2–3). For more information about the Federated Portal Adapter, see *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Figure 2–3 Database Providers



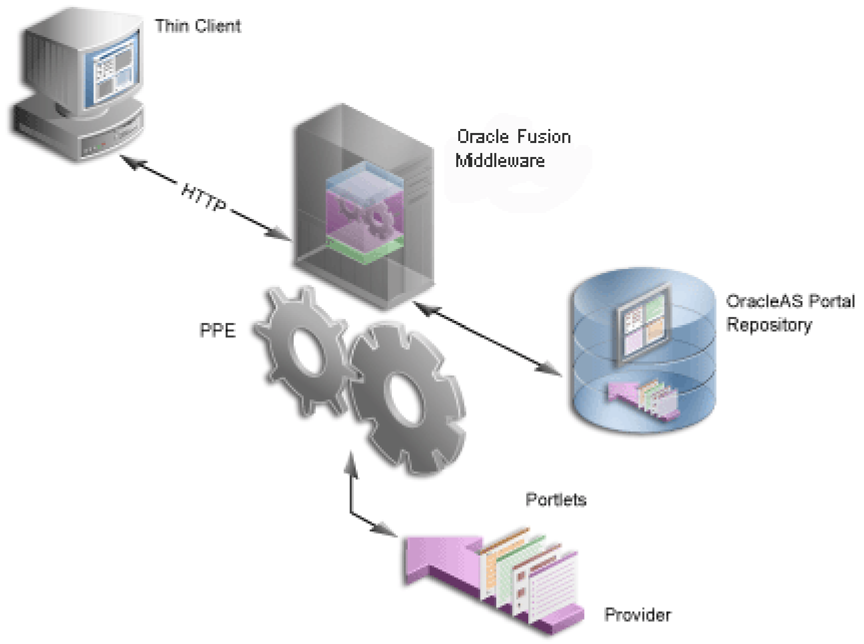
Database providers are ideal when you need to perform data-intensive operations using PL/SQL. An example of this is when you are building forms or charts with the Oracle Portal user interface or the PL/SQL APIs provided in the PDK.

To learn how to expose your PL/SQL portlets using a database provider, refer to Section 8.13, "Registering Providers Programmatically".

2.4.4 Provider Architecture

Figure 2–4 illustrates the basic architecture of portlet providers.

Figure 2–4 Provider Architecture



When users display the portal page in their Web browsers, the flow of the request works as follows:

1. The user requests a portal page from the Web browser by entering a URL in the browser's address field.
2. The Parallel Page Engine (PPE), which resides in the Oracle Application Server's middle tier, retrieves the portal page layout, portlet, and provider information (also called the page metadata) from the Oracle Portal Repository.

   Note: The PPE is responsible for constructing the requested portal page based on the page metadata.

3. The PPE contacts all the providers for the portlet content.
4. The providers 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 providers return the portlet content back to the PPE.
6. The PPE assembles the portal page, and the Oracle Application Server returns the page to the Web browser.

More On *OTN*

For more information about the portlet and provider architecture, visit the Portlet Development page on Portal Center:

`http://portalcenter.oracle.com`

Web Clipping, OmniPortlet, and Java portlets communicate with Oracle Portal through Web providers. After you install Oracle Portal, Web Clipping and OmniPortlet are ready to use; their providers are registered with Oracle Portal out of the box. You have to register the provider of your Java portlets explicitly.

Note: Web Clipping and OmniPortlet are developing very rapidly. The most recent versions of these portlets are available for download on OTN. If you decide to go with the downloaded version of these tools, you must deploy them to Oracle WebLogic Server and register them with Oracle Portal as Web providers. For more information, see *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Data-driven portlets, built with Portlet Builder, communicate with Oracle Portal through database providers. You do not need to register the Portlet Builder providers with Oracle Portal explicitly; they are automatically registered for you.

PL/SQL portlets communicate with Oracle Portal through a database provider. You have to register the database provider explicitly.

2.4.5 Provider Registration

Oracle Portal includes a provider registration wizard, accessible from the Providers tab in the Navigator. The registration screen contains the following sections:

- **Provider Information**: Contains the provider name, time out details, and the implementation style.
- **User/Session Information**: Contains information on how session information is communicated to the providers.
- **Database Providers**: Contains information specific to database providers, such as the implementation owner and name.
- **Web Providers**: Contains information specific to Web providers, such as the URL of the provider, the user's identity communicated to the provider, and proxy information.
- **WSRP Producers**: Contains information specific to WSRP producers, such as the WSDL URL and the session handling information supplied by the producer.

The sections that impact session handling are the User/Session Information section and the cookie domain check box on the Web provider registration page of the wizard. For more information on using the same cookie domain, refer to the "Sharing Session Cookies Not Allowed in PDK-Java Release 2" article, which can be accessed from the Portlet Development page on Portal Center (`http://www.oracle.com/technology/products/ias/portal/portlet_development_10gr2.html`).

User/Session Information

In the User/Session Information section, you can choose one of two options, depending on the session-related information you want the providers to receive from Oracle Portal. The options are as follows:

- **Public**: Choosing this option sets the name of the user to Public. The providers will not receive any session-related information like the session ID or the time the user logged in. This option is the equivalent of the LOGIN_FREQUENCY_PUBLIC in the provider registration API (see Section 8.13, "Registering Providers Programmatically").
- **User**: Choosing this option sends the name of the Oracle Portal user to the providers. This section contains the following two options:

- **Login Frequency**: Here, you can select one of three options (always, once for each user session, and never) to determine how often the session information must be sent to the provider, and thus how often the user needs to log in.
- **Require Portal user-specific session information**: Here, you can specify whether the session information will be sent in the provider calls.

2.5 Caching Style

Caching plays an essential role in ensuring that your portal is highly performant. Oracle Portal supports caching on various levels, such as caching pages, portlets, styles, and page metadata. Caching portlets is key to delivering accurate information in a timely manner to your users. All portlet building technologies, available with Oracle Portal, support caching.

As Oracle Portal supports user personalization of pages and portlets, the view of a page can vary from user to user. Oracle Portal's caching is designed to allow content to vary for each user. Therefore, portal objects, including portlets, can be cached at two levels, user level and system level, and can be described as follows:

- User-level caching is for a specific user; the cache entries stored are unique for that user and cannot be accessed by other users. Good candidates for user-level caching are portlets supporting personalization, such as e-mail or stock ticker portlets.
- System-level caching enables users to share a single cache entry and, therefore, there is no need to cache a copy of the object for every user. Examples of content that might be suitable for system-level caching are news portlets that are not personalizable, or custom-built navigation portlets.

When not using caching, you may find accessing various data sources with Web Clipping, OmniPortlet, and Portlet Builder to be time consuming. When you enable caching, you instruct Oracle Portal or Oracle Web Cache to maintain a copy of the portlet content. If the portlet is requested and the content was cached previously, the portlet does not have to spend time contacting the data source and regenerating its content again. Simply, the previously cached portlet content is returned. Different types of caching are as follows:

- **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 the cache content any time. Objects in Oracle Web Cache are considered valid until they are invalidated explicitly.

2.5.1 Web Clipping, OmniPortlet, and Portlet Builder

For portlets built with Web Clipping, OmniPortlet, and Portlet Builder you can specify a period of time for which they are cached (expiry-based caching). In addition to this, portlets built with Web Clipping and OmniPortlet are refreshed automatically when the end user personalizes them.

2.5.2 Java Portlets

Java portlets support three types of caching: expiry-, validation-, and invalidation-based caching. With Java portlets, you can combine invalidation-based caching with either expiry-based or validation-based caching.

In addition to caching all your portlet's content, you can also cache fragments of your portlets by using Edge Side Includes (ESI).

2.5.3 PL/SQL Portlets

Similar to Java portlets, PL/SQL portlets also support three types of caching: expiry-, validation-, and invalidation-based caching.

2.6 Development Tool

This section describes development tools you can use to build different types of portlets.

2.6.1 Web Clipping, OmniPortlet, and Portlet Builder

OmniPortlet, Web Clipping, and Portlet Builder use a browser-based wizard as the development tool.

2.6.2 Java Portlets

Although you can use any Java development environment to build Java portlets, it is highly recommended that you use Oracle JDeveloper, a professional, integrated development environment (IDE). While you can consider other IDEs, the PDK contains an Oracle JDeveloper plug-in that 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 Oracle WebLogic Server. Also, Oracle JDeveloper helps you test your portlet provider. Oracle recommends that you use the preconfigured Oracle WebLogic Server that is shipped withOracle JDeveloper as your development Java portlet runtime environment, if the version matches that of the platform on which you plan to deploy.

2.6.3 PL/SQL Portlets

When developing a PL/SQL portlet, you create PL/SQL program units that access Oracle Portal by calling Oracle Portal PL/SQL APIs. To enable this access, you create a schema, the provider schema, to store the provider and portlet PL/SQL packages in the same database in which Oracle Portal is installed. The provider schema must be granted execute privileges on the Oracle Portal PL/SQL APIs.

To facilitate the development of database providers and PL/SQL portlets, you can use the PL/SQL Generator, a hosted utility that creates installable PL/SQL code for a database provider and its PL/SQL portlets. The PL/SQL Generator is a Web application that receives the provider and portlet definitions in the form of an XML file. The syntax of the XML tags that are used for the provider and portlet definition is a subset of the XML tags that are used for defining Web providers with the PDK-Java. The output of the PL/SQL Generator is a SQL script that can be run from SQL*Plus. The script contains SQL commands for installing the provider and portlet packages.

More On OTN

The hosted PL/SQL Generator is available on the Oracle Portal Developer Kit page of OTN:

`http://www.oracle.com/technology/products/ias/portal/pdk.html`

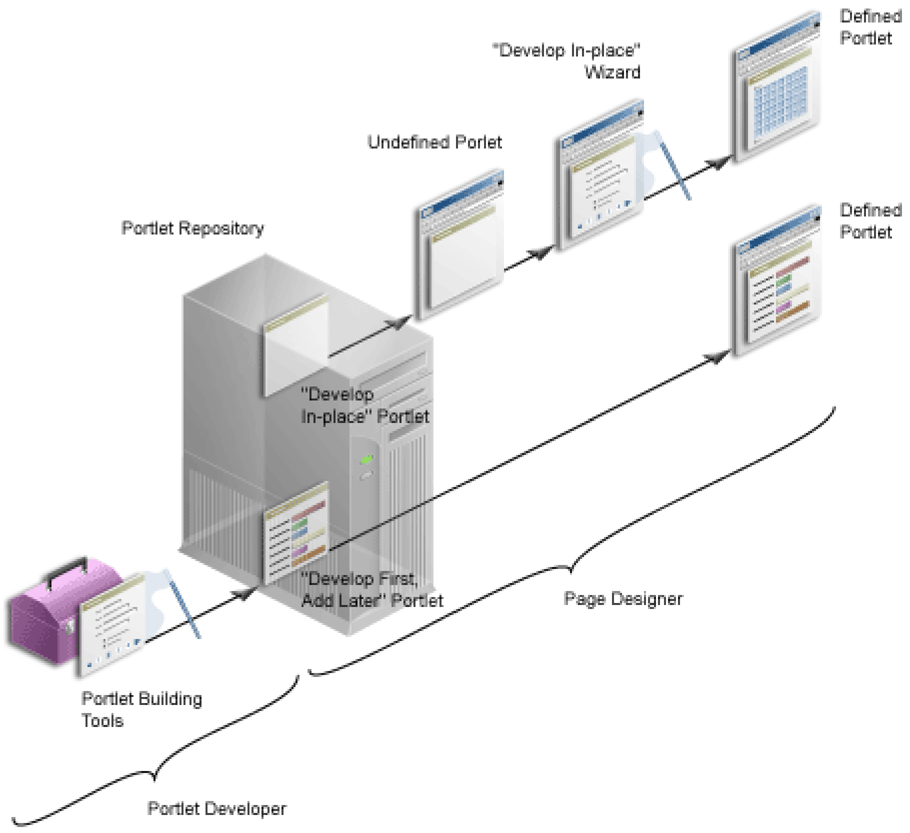
2.7 Portlet Creation Style

Oracle Portal supports the following two types of portlet creation as shown in Figure 2–5:

- Develop in-place
- Develop first, add later

The figure also indicates that the develop first, add later portlet creation is usually the task of the portlet developer, while the develop in-place portlet creation is the page designer's responsibility.

Figure 2–5 Portlet Creation Style



2.7.1 OmniPortlet and Web Clipping

OmniPortlet and Web Clipping both offer a develop in-place portlet creation style. First you add the portlets to a portal page and then you define them in place on the page.

2.7.2 Java Portlets

Typically Java portlets offer a develop first, add later portlet creation style. Two wizards are available through Oracle JDeveloper 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 Java portlets is to create the portlet, deploy it to a provider, register the provider with Oracle Portal, and then add the portlet to a page.

Note: With extensive coding, you can create develop in-place Java portlets. For example, Web Clipping and OmniPortlet are both Java portlets.

2.7.3 Portlet Builder

With Portlet Builder you define the portlets first. The previously defined portlets are then made available to you in the Portlet Repository so you can add them to your pages. For simple portlets, though, Portlet Builder offers you the develop in-place experience, similar to OmniPortlet and Web Clipping.

Note: Portlets built with Portlet Builder's develop in-place technology are somewhat limited as compared to those built using the Navigator.

2.7.4 PL/SQL Portlets

Similar to the Java portlets, PL/SQL portlets typically follow the develop first, add later creation path. Extensive coding is required to develop in-place PL/SQL portlets. For example, simple in-place portlets that are offered by Portlet Builder are written in PL/SQL.

2.8 User Interface Flexibility

This section describes the portlet building tools in terms of the control you have over the user interface.

2.8.1 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.

2.8.2 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.

2.8.3 Java Portlets and PL/SQL Portlets

In Java portlets and PL/SQL portlets, you have full control over your portlet's user interface. Your portlet is free to generate any HTML content that conforms the rendering rules for Oracle Portal pages.

2.8.4 Portlet Builder

While you can be very productive in building portlets with Portlet Builder, it is somewhat limiting with respect to the user interface.

2.9 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.

2.9.1 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, 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 end user personalization of HTML form values.

2.9.2 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.

2.9.3 Java Portlets

Java portlets can take advantage of the 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.

2.9.4 PL/SQL Portlets

PL/SQL portlets can communicate with Web servers to access data on the Internet by using procedures and functions from the UTL_HTTP package. The package makes HTTP callouts from SQL and PL/SQL. The package also supports HTTP over the Secured Socket Layer protocol (SSL), also known as HTTPS, directly or through an HTTP proxy. Other Internet-related data-access protocols (such as the File Transfer Protocol (FTP) or the Gopher protocol) are also supported using an HTTP proxy server that supports those protocols.

2.10 Ability to Render Content Inline

Active elements in your portlets, such as links or form buttons, enable your users to navigate to remote URLs. In a News portlet, for example, you 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 page, and lands on the news site.

You may have a requirement to keep your users within the context of the portal 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.

2.10.1 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.

You can choose from the following three options:

- Select not to rewrite the URLS within the portlet, in which case clicking the links takes users out of the portal 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, 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 Oracle Portal, 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 portal page so that all browsing within the Web Clipping portlet remains within Oracle Portal. If the Web Clipping provider is registered with an External Application, this will cause the Web Clipping provider to log itself into the External Application. In this case, the navigation within the portal through the Web Clipping provider is authenticated in the External Application.

2.10.2 OmniPortlet

OmniPortlet does not offer URL rewriting directly, but you can achieve inline rendering functionality by using public portlet parameters and events. Then you have to map the events to the same portal page where your OmniPortlet resides.

2.10.3 Java Portlets

Since you have full control over the links and buttons in Java portlets, you can easily implement 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, the PDK-Struts integration framework renders your content always in the same portlet container.

If your portlet consists of multiple JSPs (for example, several steps in a survey or wizard), your portlet can make use of a special parameter to specify at run time the JSP to use to render the content.

2.10.4 Portlet Builder

Portlets built with Portlet Builder do not have inherent inline rendering support. You can, however, construct your links in SQL-based reports and charts so that they point to specific portal pages. If required, you can also pass parameters to portal pages, which in turn can be mapped to portlet parameters.

2.10.5 PL/SQL Portlets

Similar to Java portlets, you have full control over the active elements in PL/SQL portlets and, therefore, you can achieve the inline rendering functionality programmatically by implementing private portlet parameters.

2.11 Charting Capability

This section describes the portlet building tools in terms of their charting functionality.

2.11.1 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.

2.11.2 OmniPortlet

OmniPortlet supports bar, line, and pie charts. Charts in OmniPortlet are dynamically generated images, which can include hyperlinks.

2.11.3 Java Portlets

You can create sophisticated chart portlets programmatically in your Java portlets using Oracle's Business Intelligence (BI) Beans.

Note: Oracle Reports and Oracle BI Discoverer portlets use BI Beans to create professional graphs.

2.11.4 Portlet Builder

With Portlet Builder, you can build HTML-based bar chart portlets. Among other features, you can specify the color and orientation of the bars.

2.11.5 PL/SQL Portlets

In PL/SQL portlets, HTML-based charting can be achieved by extensive coding.

2.12 Public Portlet Parameters Support

There are three types of parameters in Oracle Portal: page parameters, public portlet parameters, and private portlet parameters. These parameters can be described as follows:

- **Page parameters:** You can use a page parameter to pass a value to a page. Using page parameters, the information that is displayed on a page can vary depending on where the page is called from and who is viewing the page. Using page parameters, page designers can synchronize the portlets on a page by passing them the same values. This provides the ability to reuse and tailor portlets on pages by merely integrating them with page parameters. Without this functionality, you would have to code portlets individually to use different parameter values.
- **Public portlet parameters:** You can use a public portlet parameter to pass a value to a portlet. Using portlet parameters, the information that is displayed in a portlet can be specific to a particular page or a user. Portlet parameters are created by the

portlet developer and are exposed to the page designer, through the user interface. After adding a portlet to a page, page designers can assign values to the public portlet parameters to make the information displayed in the portlet specific to the page.

Page designers can assign values to public portlet parameters by providing a specific value (constant), a system variable (for example, the portal user name), or a page parameter. At run time, the portlet receives the values from the sources specified. In this way, page designers have complete control over the source of the parameter, whereas you have complete control over how the data is used after it is transmitted to the portlet.

- **Private portlet parameters:** You can use private portlet parameters to implement internal navigation in your portlet. You can pass parameters to your portlets 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. You can create a link in a portlet that can be used to refresh the portlet only, without triggering a full portal page refresh. By doing this, only the content of the affected portlet is updated and the rest of the page is not. Refer to "Partial Page Refresh" in Chapter 7, "Enhancing Java Portlets" for details about setting this programmatically.

Portlets supporting public portlet parameters enable page designers to tailor the portlets' data input for each portlet instance. In this case, the portlet developer can focus on the portlet logic, while page designers can easily reuse portlets and address the interaction between the page and the portlets.

All five portlet building technologies discussed in this chapter (OmniPortlet, Web Clipping, Java portlets, Portlet Builder, and PL/SQL portlets) support public portlet parameters. OmniPortlet, Web Clipping, and Portlet Builder provide complete support through their wizard interface. You can add public portlet parameter support to your Java portlets programmatically or with the Java Portlet Wizard. PL/SQL portlets support public parameters only programmatically.

Note: The JSR 168 standard does not cover the notion of public portlet parameters. If you want to utilize public portlet parameters in your Java portlets, you have to use PDK-Java.

2.13 Private Portlet Parameter Support

This section describes the portlet building tools in terms of their support for private parameters.

2.13.1 OmniPortlet, Web Clipping, and Portlet Builder

OmniPortlet, Web Clipping, and Portlet Builder do not provide access to the portlet developer to private portlet parameters.

2.13.2 Java Portlets and PL/SQL Portlets

In your Java portlets and PL/SQL portlets, you can implement internal navigation by using private portlet parameters.

Note: PL/SQL portlets do not support private and public parameters simultaneously. You need to decide which parameter type to support before coding your PL/SQL portlet.

2.14 Event Support

An event is a user action that you define to display a Portal page. User actions include clicking a link or a button in a portlet. Page designers specify what to do when an event occurs in a portlet on a page. When an event occurs, page designers can either redisplay the current page or navigate the user to another portal page, optionally passing values to that page's parameters.

2.14.1 Web Clipping, OmniPortlet, and Java Portlets

Web Clipping, OmniPortlet, and Java portlets support events.

2.14.2 Portlet Builder and PL/SQL Portlets

Portlet Builder and PL/SQL portlets do not support events.

2.15 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.

2.15.1 Web Clipping and OmniPortlet

You can hide and show portlets built with Web Clipping and OmniPortlet on portal pages dynamically by using security managers. Although Web Clipping and OmniPortlet do not expose security managers through the user interface, you can apply them by editing their XML provider definition file.

2.15.2 Java Portlets

The PDK provides a number of security managers for Java portlets. Following are two examples:

- **Group security manager:** The group security manager makes the portlet appear to users who are members of a specified group, while hiding it from those who are not members.
- **Authentication level security manager:** You can use the authentication level security manager to control access to the portlets based on the user's authentication level. For example you may hide the portlet from public users but display it to authenticated users.

JSR 168 portlets support the standard servlet mechanisms.

2.15.3 Portlet Builder

Portlet Builder provides a declarative user interface to control access to portlets.

2.15.4 PL/SQL Portlets

The PDK provides security APIs to implement hiding and showing content in PL/SQL portlets.

2.16 Multilingual Support

This section describes the portlet building tools in terms of their support for other languages.

2.16.1 Web Clipping, OmniPortlet, Java Portlets, and PL/SQL Portlets

Web Clipping, OmniPortlet, Java portlets, and PL/SQL portlets display textual information in the language selected by the portal user.

2.16.2 Portlet Builder

Portlets built with Portlet Builder support English only.

2.17 Pagination Support

Support for pagination is useful when a portlet must display a relatively large set of records.

2.17.1 Web Clipping

Pagination support is not applicable to Web Clipping.

2.17.2 OmniPortlet

OmniPortlet does not support pagination.

2.17.3 Java Portlets and PL/SQL Portlets

You can implement pagination in your Java portlets and PL/SQL portlets programmatically.

2.17.4 Portlet Builder

Portlet Builder has built-in support for pagination.

2.18 Single Sign-On and External Application Integration

This section describes the portlet building tools in terms of authentication for external application.

2.18.1 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 have to do is to associate an External Application ID to the Web Clipping provider when registering the provider.

2.18.2 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. The credentials are stored in the secured data repository of OmniPortlet, in an Oracle database.

2.18.3 Java Portlets

Java portlets support programmatic integration with the external application framework as well as any LDAP server, such as Oracle Internet Directory.

2.18.4 PL/SQL Portlets

You can build PL/SQL portlets that enable single sign-on by using mod_osso, an authentication module on the Oracle HTTP Server. mod_osso is a simple alternative to the single sign-on SDK, used in earlier releases to integrate partner applications. mod_osso simplifies the authentication process by serving as the sole partner application to the Single Sign-On server.

PL/SQL portlets can integrate with the external application framework programmatically.

Part II

Creating Portlets

Part II contains the following chapters:

3 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:

- Section 3.1, "Introduction to OmniPortlet"
- Section 3.2, "The OmniPortlet Wizard"
- Section 3.3, "Parameters and Events"

For information about using OmniPortlet to build example portlets, see Chapter 4, "Building Example Portlets with OmniPortlet." For troubleshooting information regarding OmniPortlet, see Section B.3, "Diagnosing OmniPortlet Problems." For information about registering and configuring OmniPortlet with Oracle Portal, see *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

3.1 Introduction to OmniPortlet

OmniPortlet is a subcomponent of Oracle Portal that enables page designers and 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 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 1, "Understanding Portlets," and information about providers and other portlet technologies in Chapter 2, "Portlet Technologies Matrix."

OmniPortlet enables page designers and content contributors to do the following:

- Display data from multiple sources (CSV, XML, 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
- Raise portlet events
- 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.

Note: To use OmniPortlet, the Simple Parameter Form, or the Web Clipping portlet with Windows 2000, you must use Netscape 7.0 or later, or Microsoft Internet Explorer 5.5 or later.

You can add an OmniPortlet to a portal page just as you would add any other portlet. OmniPortlet can be found under Portlet Builders in the Portlet Repository. If you have downloaded OmniPortlet as part of the Oracle Portal Developer Kit, for example, to upgrade to a later release, you must register it before you can use it.

You can find more information about building portal pages and adding portlets in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

Instructions for installing, configuring, and registering the OmniPortlet provider are provided within the `pdksoftware.zip` file containing the PDK-Java and Portal Tools. For specific information on configuring OmniPortlet, see *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

3.2 The OmniPortlet Wizard

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 re-enter the wizard by clicking Edit Defaults for the portlet. When you re-enter the wizard, you can change the definitions on the Source, Filter, View, and Layout tabs, as well as set up the event parameters on the Events tab.

Note: On the IBM Linux on Power platform, if the action buttons (Next, Previous, Finish, and Cancel) are minimized to dots when defining the OmniPortlet, increase the stack size shell limit to unlimited and restart the WLS_PORTAL instance. Run the following command to set the stack size shell limit to unlimited:

```
prompt> ulimit -s unlimited
```

This section provides a high-level overview of the six tabs, which are described in Table 3–1. 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 3–1 OmniPortlet Wizard and Edit Defaults

Step/Tab	Description
Type	Provides your data source options. Displays only in the initial definition of the portlet, and is not available when editing 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.
Filter	Provides sorting options at the Oracle Portal 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.

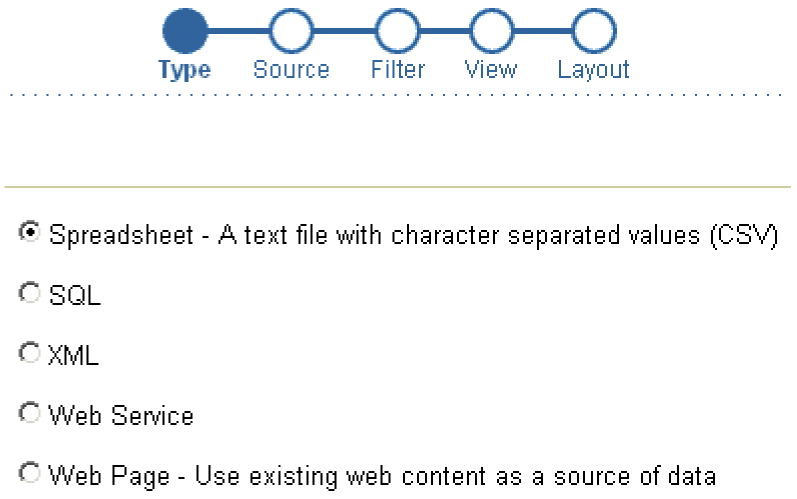
Table 3–1 (Cont.) OmniPortlet Wizard and Edit Defaults

Step/Tab	Description
Layout	Provides detailed options for customizing the layout. You can change these options later when editing the defaults of the portlet.
Events	Does not display in the initial definition of the portlet. Provides options for adding events to the portlet. Displays only after the portlet has been defined in the Edit Defaults mode of the wizard.

3.2.1 Type

When you first start OmniPortlet, the Type step displays, which enables you to choose your data source (Figure 3–1). Out of the box, OmniPortlet supports the data sources shown in Table 3–2.

Figure 3–1 Type Tab of the OmniPortlet Wizard



[✓ Spreadsheet - A text file with character separated values (CSV)]

SQL

XML

Web Service

Web Page - Use existing web content as a source of data

Note: If you've downloaded and installed an additional data source, the data source will display on the Type tab.

Table 3–2 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.

Table 3–2 (Cont.) Supported Data Source Types

Data Source Type	Description
J2EE Connector Architecture*	(Displays only if the Sample Provider is registered with Oracle Portal). 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).

After you complete the OmniPortlet Wizard and edit the defaults of the portlet, you cannot change the data source type.

3.2.2 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 provider 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:

- Section 3.2.2.1, "Proxy Authentication"
- Section 3.2.2.2, "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:

- Section 3.2.2.3, "Spreadsheet"
- Section 3.2.2.4, "SQL"
- Section 3.2.2.5, "XML"
- Section 3.2.2.6, "Web Service"
- Section 3.2.2.7, "Web Page"

Note: For more information on the Source tab options, click Help in the upper right corner of the page.

3.2.2.1 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 provider has been set up to use proxy authentication that requires your login, a Proxy Authentication section displays on the Source tab where you can enter this information.

The Proxy Authentication section only displays for the data sources that may require you to use a proxy server to access them: CSV (character-separated values), XML, Web Service, and Web Page. For more information on configuring the OmniPortlet provider to use proxy authentication, see the online Help topic that displays when you click Help on the Edit Providers: OmniPortlet Provider page. If the OmniPortlet provider is configured to "Require login for all users," each user must set his or her own login information:

- For page designers, set this in Edit Defaults: Source tab.

- For page viewers, set this on the Personalize screen.

You can also find more information on configuring OmniPortlet in *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

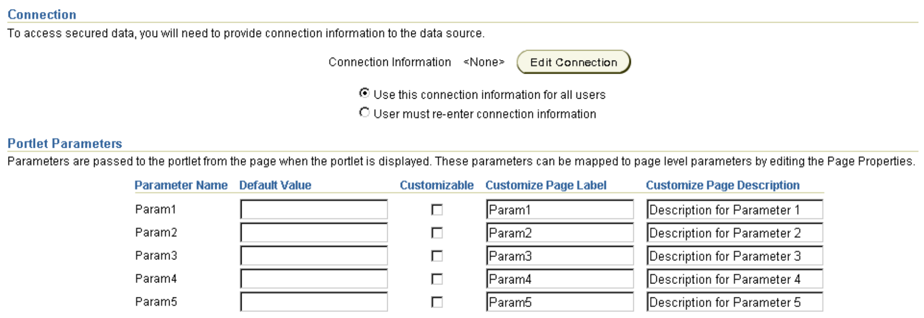
Note: If you are using the Web Page data source, the Proxy Authentication section displays in the Web Clipping Studio, after you have clicked the Select Web Page button on the Source tab.

3.2.2.2 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 includes a Portlet Parameters section, where you can define the parameters for the portlet (Figure 3–2). You can then map the portlet parameters to the page-level parameters.

Note: If you use parameters in place of Username, Password, or Connection String, the Test button will return an error, however the connection information is correct when the parameter values are substituted.

Figure 3–2 Source Tab: Connection and Portlet Parameters Section



Connection

To access secured data, you will need to provide connection information to the data source.

Connection Information <None>

[✓ Use this connection information for all users]

User must re-enter connection information

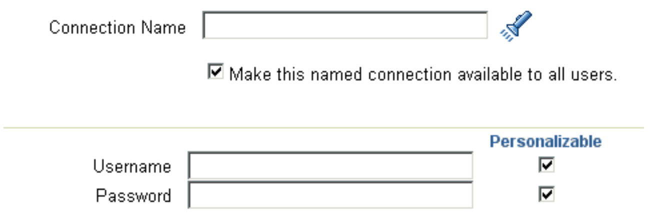
Portlet Parameters

Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page level parameters by editing the Page Properties.

Parameter Name	Default Value	Customizable	Customize Page Label	Customize Page Description
Param1		☐	Param1	Description for Parameter 1
Param2		☐	Param2	Description for Parameter 2
Param3		☐	Param3	Description for Parameter 3
Param4		☐	Param4	Description for Parameter 4
Param5		☐	Param5	Description for Parameter 5

To edit the connection information, click the Edit Connection button and fill out the information on the page shown in Figure 3–3. 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 wish to use to connect to the data source. For more information, see Section 3.2.2.4, "SQL."

Figure 3–3 Edit Connection Page



Connection Name

☑ Make this named connection available to all users.

		Personalizable
Username		☑
Password		☑

Note: For more information about the Connection section and the Edit Connection button, click Help on the Source tab of the OmniPortlet wizard.

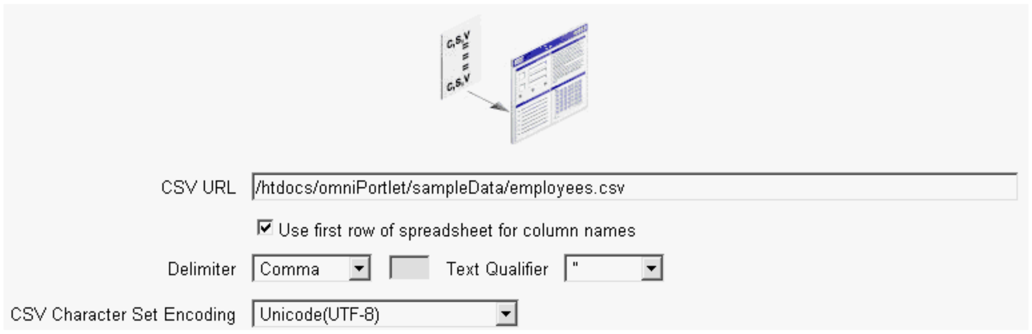
3.2.2.3 Spreadsheet

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 3–4). If the file is located on a secure server, you can specify the connection information in the Connection section described in Figure 3–2. You can also select the character set to use when Oracle Portal reads the file, as well as the delimiter and text qualifier.

Since the OmniPortlet provider exists and executes in a tier different from the Oracle Portal application and does not have access to the Oracle Portal session information, you must expose CSV files that are uploaded to Oracle Portal as PUBLIC in order for OmniPortlet to access them.

Note: For more information on using the CSV data source, see Section 4.3, "Building an OmniPortlet Based on a Spreadsheet (CSV)."

Figure 3–4 Source Tab: Spreadsheet



CSV URL /htdocs/omniPortlet/sampleData/employees.csv

[✓] Use first row of spreadsheet for column names

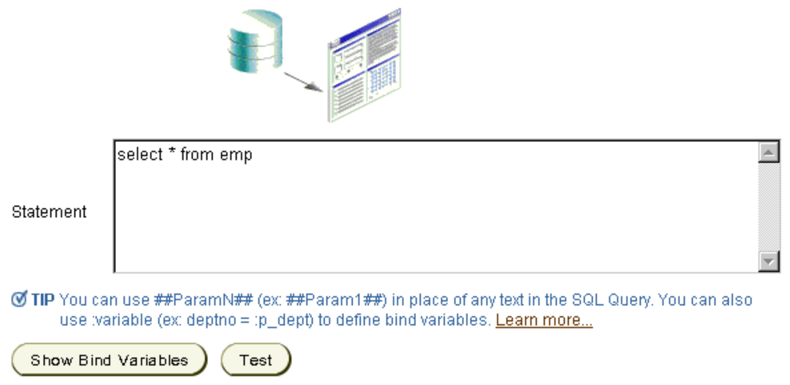
Delimiter Comma Text Qualifier "

CSV Character Set Encoding Unicode(UTF-8)

3.2.2.4 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 3–5 shows the Source tab for a SQL data source.

Figure 3–5 Source Tab: SQL



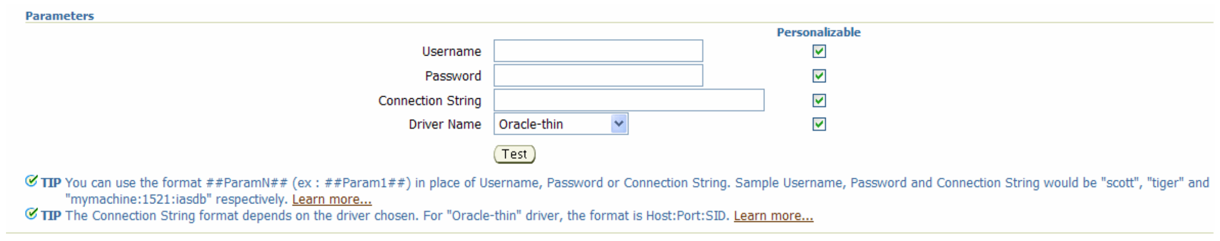
Statement: select * from emp

TIP You can use ##ParamN## (ex: ##Param1##) in place of any text in the SQL Query. You can also use :variable (ex: deptno = :p_dept) to define bind variables. Learn more...

3.2.2.4.1 SQL Connection Information You can use the DataDirect JDBC drivers to access other relational databases. To configure OmniPortlet to use these drivers, see *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

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 3–6.

Figure 3–6 Connection Information on the SQL Source Tab



Parameters

		Personalizable
Username		✓
Password		✓
Connection String		✓
Driver Name	Oracle-thin	✓

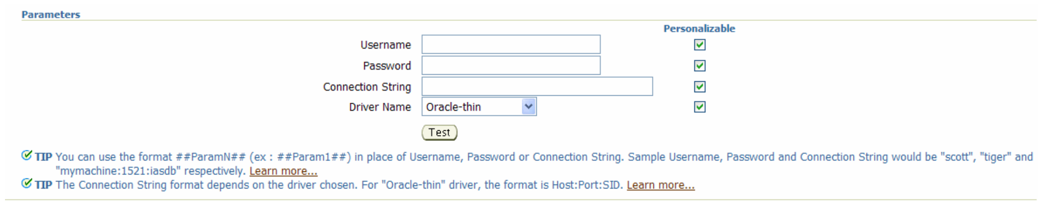
TIP You can use the format ##ParamN## (ex : ##Param1##) in place of Username, Password or Connection String. Sample Username, Password and Connection String would be "scott", "tiger" and "mymachine:1521:iasdb" respectively. Learn more...

TIP The Connection String format depends on the driver chosen. For "Oracle-thin" driver, the format is Host:Port:SID. Learn more...

Note: For more information on DataDirect drivers, see the *Certification Matrix for Oracle Fusion Middleware and DataDirect JDBC* (`http://www.oracle.com/technology/products/index.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. You can see an example in Figure 3–7.

Figure 3–7 Edit Connection Page with DataDirect Driver



Parameters

		Personalizable
Username		✓
Password		✓
Connection String		✓
Driver Name	Oracle-thin	✓

TIP You can use the format ##ParamN## (ex : ##Param1##) in place of Username, Password or Connection String. Sample Username, Password and Connection String would be "scott", "tiger" and "mymachine:1521:iasdb" respectively. Learn more...

TIP The Connection String format depends on the driver chosen. For "Oracle-thin" driver, the format is Host:Port:SID. Learn more...

3.2.2.4.2 Using Stored Procedures You can also make a call to Stored Procedures instead of SQL statements 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 the following procedure accepts `JOB` as a 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` with the connection information to your database. You can find the connection string in the `tnsnames.ora` file within your `ORACLE_INSTANCE\config` 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:

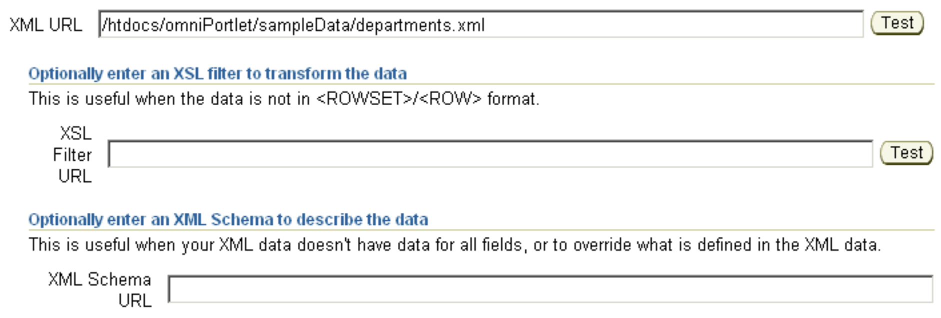
   ```
   call emp_proc('CLERK')
   ```

3.2.2.5 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 3–8.

Note: For more information on using the XML data source, see Section 4.4, "Building an OmniPortlet Based on an XML Data Source."

Figure 3–8 Source Tab: XML


XML URL /htdocs/omniPortlet/sampleData/departments.xml

Optionally enter an XSL filter to transform the data

This is useful when the data is not in <ROWSET>/<ROW> format.

XSL Filter URL

Optionally enter an XML Schema to describe the data

This is useful when your XML data doesn't have data for all fields, or to override what is defined in the XML data.

XML Schema URL

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 example shows 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, 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: Since the OmniPortlet provider exists and executes in a different tier from Oracle Portal and does not have access to the Oracle Portal session information, you must expose XML files that are uploaded to Oracle Portal as PUBLIC in order for OmniPortlet to access them.

3.2.2.6 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 a platform or language, 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.

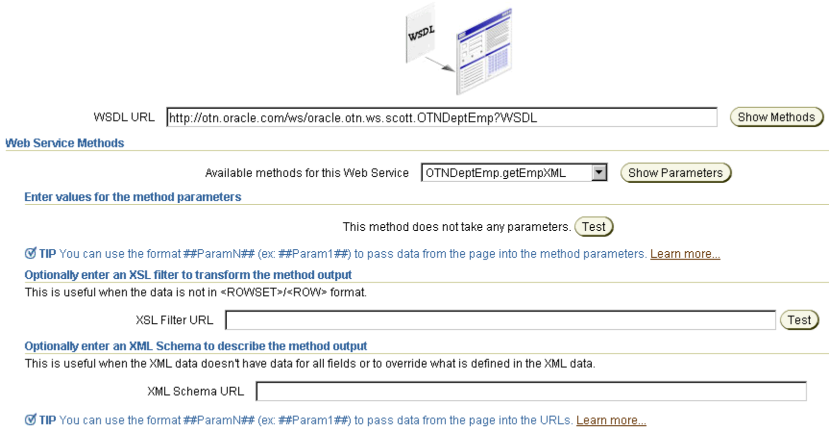
OmniPortlet supports both types of Web Services: Document and RPC (Remote Procedure Calls). After a WSDL document/file is supplied, it is parsed, and the available methods that can be called display on the Source tab.

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 3–9 shows the Source tab for a Web service.

Note: For more information on using the Web Service data source, see Section 4.2, "Building an OmniPortlet Based on a Web Service."

Figure 3–9 Source Tab: Web Service



WSDL URL http://otn.oracle.com/ws/oracle.otn.ws.scott.OTNDeptEmp?WSDL

Web Service Methods

Available methods for this Web Service OTNDeptEmp.getEmpXML

Enter values for the method parameters

This method does not take any parameters.

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the method parameters.

Optionally enter an XSL filter to transform the method output

This is useful when the data is not in <ROWSET>/<ROW> format.

XSL Filter URL

Optionally enter an XML Schema to describe the method output

This is useful when the XML data doesn't have data for all fields or to override what is defined in the XML data.

XML Schema URL

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URLs.

3.2.2.7 Web Page

OmniPortlet enables you to use existing Web content as a source of data to publish information to your portal. 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:

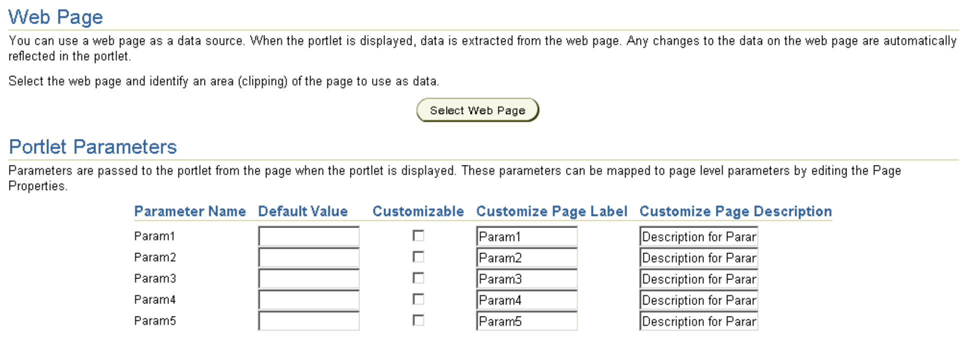
- **Navigation through various login mechanisms**, including form- and JavaScript-based submission, and HTTP Basic and Digest Authentication with cookie-based session management.
- **Fuzzy matching of clippings.** If a Web clipping gets reordered within the source page or if its character font, size, or style changes, 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 Fusion Middleware infrastructure database or on another Oracle database. 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 3–10) 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 Fusion Middleware Web Clipping online Help.

Note: For more information on using the Web Page data source, see Section 4.5, "Building an OmniPortlet Based on a Web Page Data Source."

Figure 3–10 Source Tab: Web Page



Web Page

You can use a web page as a data source. When the portlet is displayed, data is extracted from the web page. Any changes to the data on the web page are automatically reflected in the portlet.

Select the web page and identify an area (clipping) of the page to use as data.

Portlet Parameters

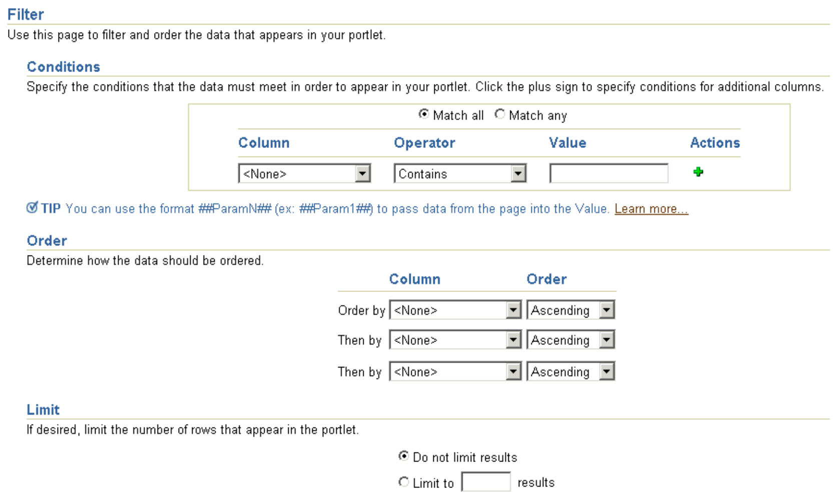
Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page level parameters by editing the Page Properties.

Parameter Name	Default Value	Customizable	Customize Page Label	Customize Page Description
Param1		☐	Param1	Description for Parar
Param2		☐	Param2	Description for Parar
Param3		☐	Param3	Description for Parar
Param4		☐	Param4	Description for Parar
Param5		☐	Param5	Description for Parar

3.2.3 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, 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, you can use the options on the Filter tab to sort your data (Figure 3–11).

Figure 3–11 Filter Tab



Filter

Use this page to filter and order the data that appears in your portlet.

Conditions

Specify the conditions that the data must meet in order to appear in your portlet. Click the plus sign to specify conditions for additional columns.

[✓ Match all] Match any

Column	Operator	Value	Actions
<None>	Contains		+

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the Value. Learn more...

Order

Determine how the data should be ordered.

	Column	Order
Order by	<None>	Ascending
Then by	<None>	Ascending
Then by	<None>	Ascending

Limit

If desired, limit the number of rows that appear in the portlet.

[✓ Do not limit results]

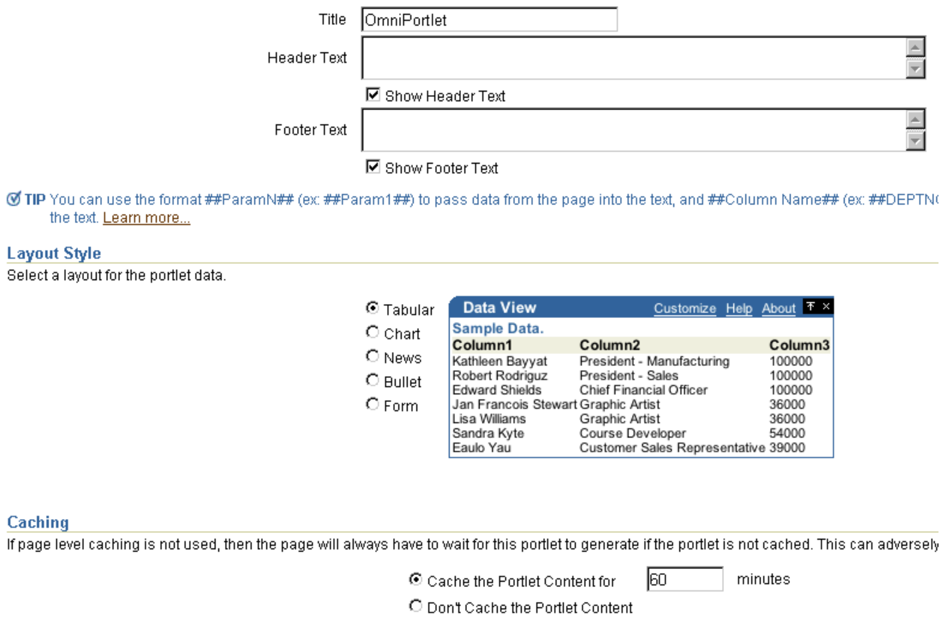
Limit to ____ results

3.2.4 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 3–12) 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 3–12 View Tab



Title: OmniPortlet

Header Text:

[✓ Show Header Text]

Footer Text:

[✓ Show Footer Text]

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the text, and ##Column Name## (ex: ##DEPTN
the text. Learn more...

Layout Style

Select a layout for the portlet data.

[✓ Tabular] Chart News Bullet Form

Data View — Customize Help About

Sample Data.

Column1	Column2	Column3
Kathleen Bayyat	President - Manufacturing	100000
Robert Rodriguz	President - Sales	100000
Edward Shields	Chief Financial Officer	100000
Jan Francois Stewart	Graphic Artist	36000
Lisa Williams	Graphic Artist	36000
Sandra Kyte	Course Developer	54000
Eaulo Yau	Customer Sales Representative	39000

Caching

If page level caching is not used, then the page will always have to wait for this portlet to generate if the portlet is not cached. This can adversely

[✓ Cache the Portlet Content for] 60 minutes
Don't Cache the Portlet Content

Note: For more information on 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.

3.2.5 Layout

The Layout tab changes depending on the Layout Style you chose on the View tab, and enables you to further personalize 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.

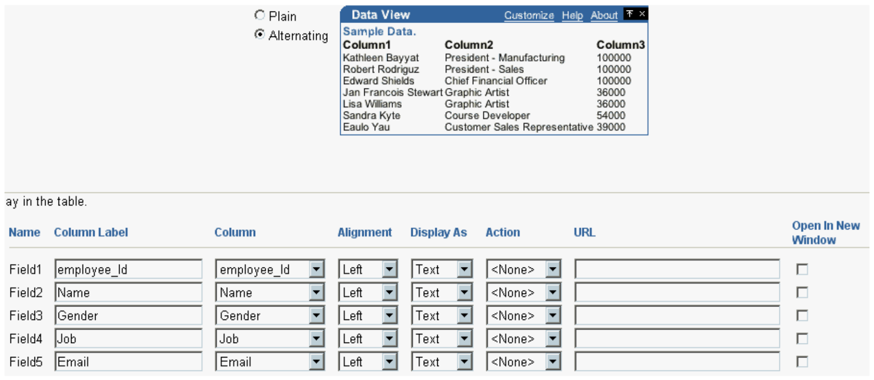
The following layout styles are available with OmniPortlet:

- Section 3.2.5.1, "Tabular Layout"
- Section 3.2.5.2, "Chart Layout"
- Section 3.2.5.3, "News Layout"
- Section 3.2.5.4, "Bullet Layout"
- Section 3.2.5.5, "Form Layout"
- Section 3.2.5.6, "HTML Layout"

3.2.5.1 Tabular Layout

Once you've chosen the tabular style on the View tab, you can refine the layout on the Layout tab (Figure 3–13). 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 3–13 Layout Tab: Tabular Style



( ) Plain [✓ Alternating]

Data View — Customize Help About

Sample Data.

Column1	Column2	Column3
Kathleen Bayyat	President - Manufacturing	100000
Robert Rodriguz	President - Sales	100000
Edward Shields	Chief Financial Officer	100000
Jan Francois Stewart	Graphic Artist	36000
Lisa Williams	Graphic Artist	36000
Sandra Kyte	Course Developer	54000
Eaulo Yau	Customer Sales Representative	39000

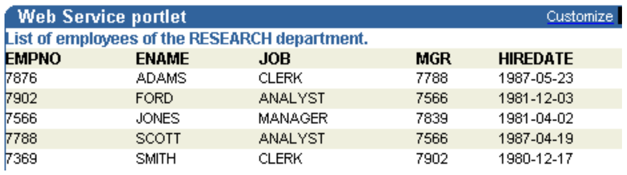
ay in the table.

Name	Column Label	Column	Alignment	Display As	Action	URL	Open In New Window
Field1	employee_Id	employee_Id	Left	Text	<None>		☐
Field2	Name	Name	Left	Text	<None>		☐
Field3	Gender	Gender	Left	Text	<None>		☐
Field4	Job	Job	Left	Text	<None>		☐
Field5	Email	Email	Left	Text	<None>		☐

Note: You can control the background color of a portlet through the portal page style. For more information on using portal page styles, see *Oracle Fusion Middleware User's Guide for Oracle Portal*.

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 3–14 shows an example of an OmniPortlet using a tabular format.

Figure 3–14 Example of an OmniPortlet Using a Tabular Layout



Web Service portlet — Customize

List of employees of the RESEARCH department.

EMPNO	ENAME	JOB	MGR	HIREDATE
7876	ADAMS	CLERK	7788	1987-05-23
7902	FORD	ANALYST	7566	1981-12-03
7566	JONES	MANAGER	7839	1981-04-02
7788	SCOTT	ANALYST	7566	1987-04-19
7369	SMITH	CLERK	7902	1980-12-17

Note: For more information on using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

3.2.5.2 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 3–15), 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 3–16 shows an example of the Layout tab for a pie chart layout.

Figure 3–15 Layout Tab: Chart

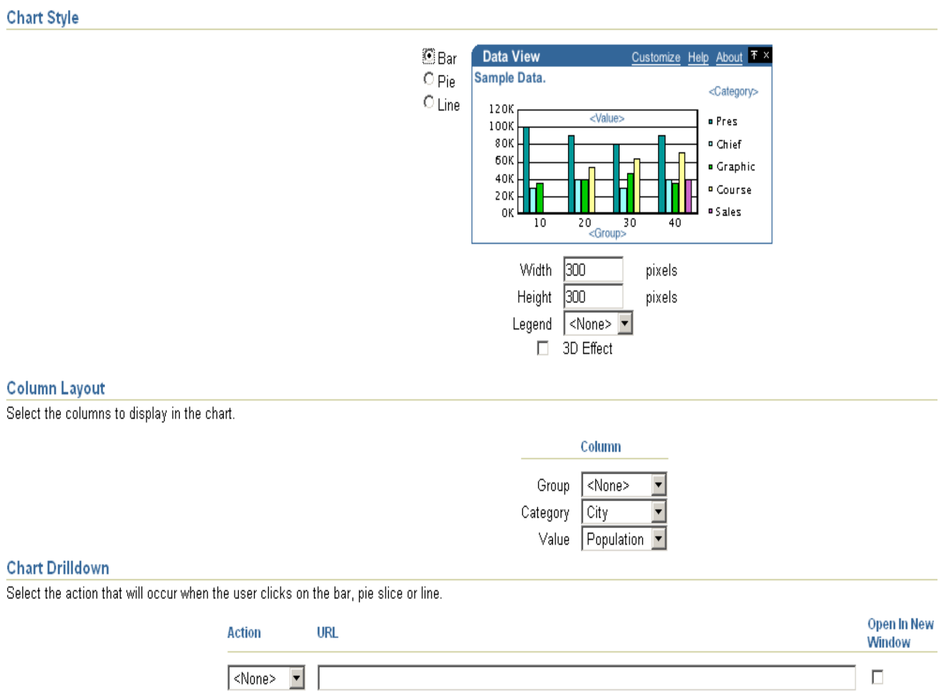


Chart Style

[✓ Bar] Pie Line

Width	300	pixels
Height	300	pixels
Legend	<None>	

☐ 3D Effect

Column Layout

Select the columns to display in the chart.

	Column
Group	<None>
Category	City
Value	Population

Chart Drilldown

Select the action that will occur when the user clicks on the bar, pie slice or line.

Action	URL	Open In New Window
<None>		☐

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), 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 3–16 Example of the Layout Tab for a Pie Chart Layout

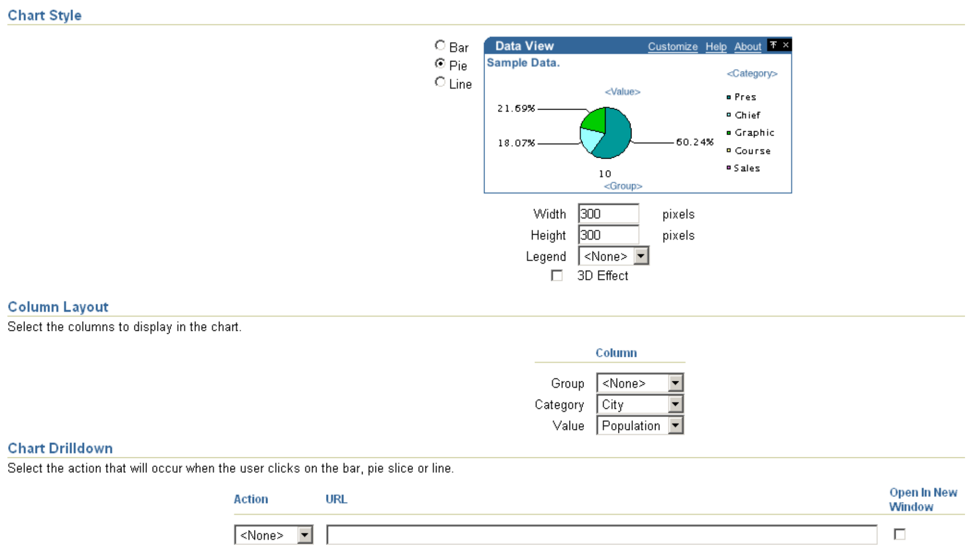


Chart Style: [✓ Pie] (Bar, Line)

Width 300 pixels
Height 300 pixels
Legend <None>
3D Effect

Column Layout
Select the columns to display in the chart.

	Column
Group	<None>
Category	City
Value	Population

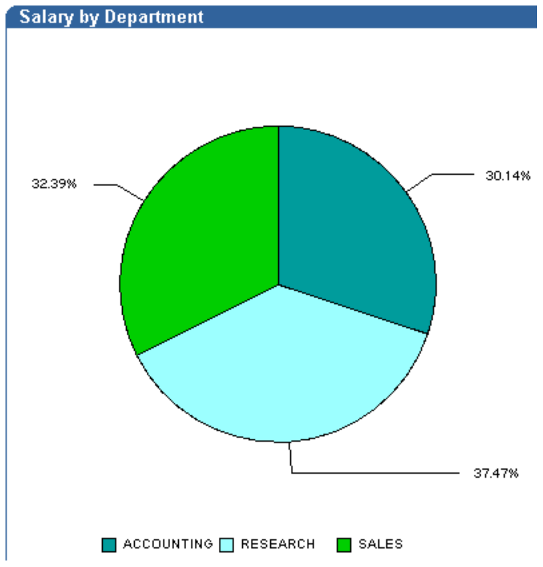
Chart Drilldown
Select the action that will occur when the user clicks on the bar, pie slice or line.

Action	URL	Open In New Window
<None>		

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 portal 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 3–17, you can see an example of a pie chart. Below the chart, you can see that the category, Department, is used for the legend.

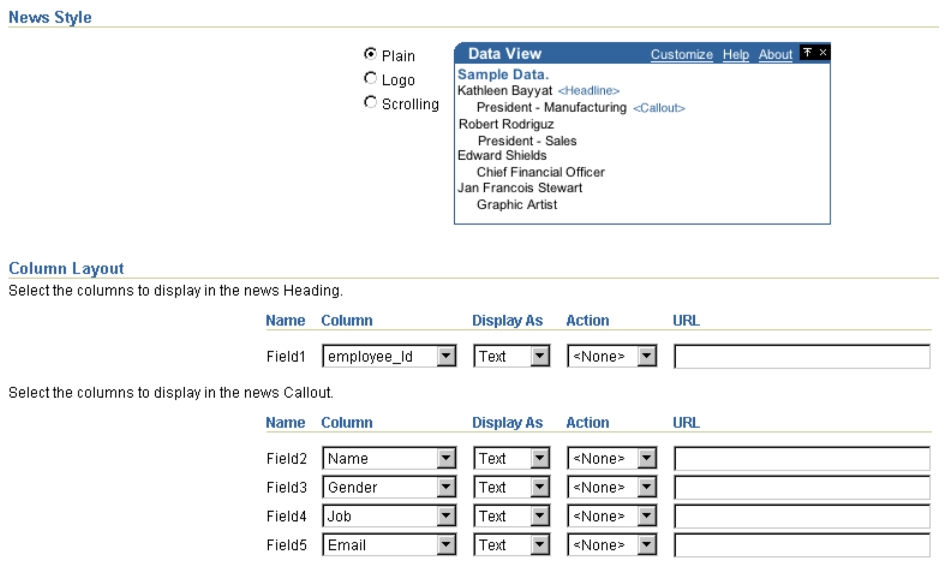
Figure 3–17 Example of an OmniPortlet Using a Pie Chart Layout



3.2.5.3 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 portal page. In the Column Layout section (Figure 3–18), 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 3–18 Layout Tab: News



News Style

(● Plain) ○ Logo ○ Scrolling

Sample Data.
Kathleen Bayyat <Headline>
President - Manufacturing <Callout>
Robert Rodriguz
President - Sales
Edward Shields
Chief Financial Officer
Jan Francois Stewart
Graphic Artist

Column Layout

Select the columns to display in the news Heading.

Name	Column	Display As	Action	URL
Field1	employee_Id	Text	<None>	

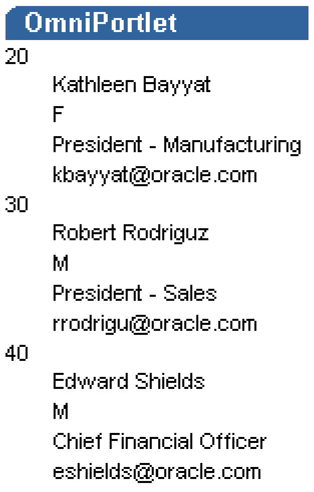
Select the columns to display in the news Callout.

Name	Column	Display As	Action	URL
Field2	Name	Text	<None>	
Field3	Gender	Text	<None>	
Field4	Job	Text	<None>	
Field5	Email	Text	<None>	

Figure 3–19 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 3–19 Example of an OmniPortlet Using a News Layout



OmniPortlet

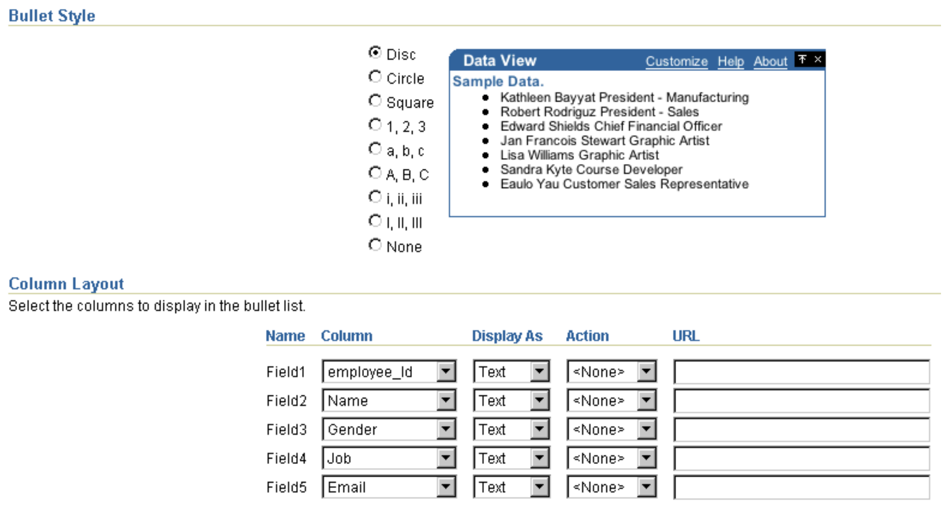
20
Kathleen Bayyat
F
President - Manufacturing
kbayyat@oracle.com
30
Robert Rodriguz
M
President - Sales
rrodrigu@oracle.com
40
Edward Shields
M
Chief Financial Officer
eshields@oracle.com

Note: For more information on using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

3.2.5.4 Bullet Layout

You can use the bullet layout to display your data in a bulleted list. The Layout tab (Figure 3–20) 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 3–20 Layout Tab: Bullet



Sample Data.
- Kathleen Bayyat President - Manufacturing
- Robert Rodriguz President - Sales
- Edward Shields Chief Financial Officer
- Jan Francois Stewart Graphic Artist
- Lisa Williams Graphic Artist
- Sandra Kyte Course Developer
- Eaulo Yau Customer Sales Representative

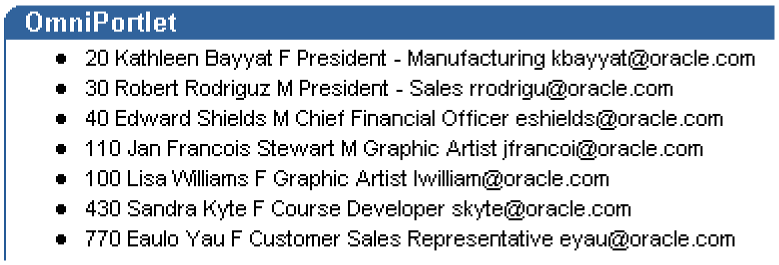
Column Layout

Select the columns to display in the bullet list.

Name	Column	Display As	Action	URL
Field1	employee_Id	Text	<None>	
Field2	Name	Text	<None>	
Field3	Gender	Text	<None>	
Field4	Job	Text	<None>	
Field5	Email	Text	<None>	

Figure 3–21 shows an example of an OmniPortlet using a bullet layout.

Figure 3–21 Example of an OmniPortlet Using a Bullet Layout



- 20 Kathleen Bayyat F President - Manufacturing kbayyat@oracle.com
- 30 Robert Rodriguz M President - Sales rrodrigu@oracle.com
- 40 Edward Shields M Chief Financial Officer eshields@oracle.com
- 110 Jan Francois Stewart M Graphic Artist jfrancoi@oracle.com
- 100 Lisa Williams F Graphic Artist lwilliam@oracle.com
- 430 Sandra Kyte F Course Developer skyte@oracle.com
- 770 Eaulo Yau F Customer Sales Representative eyau@oracle.com

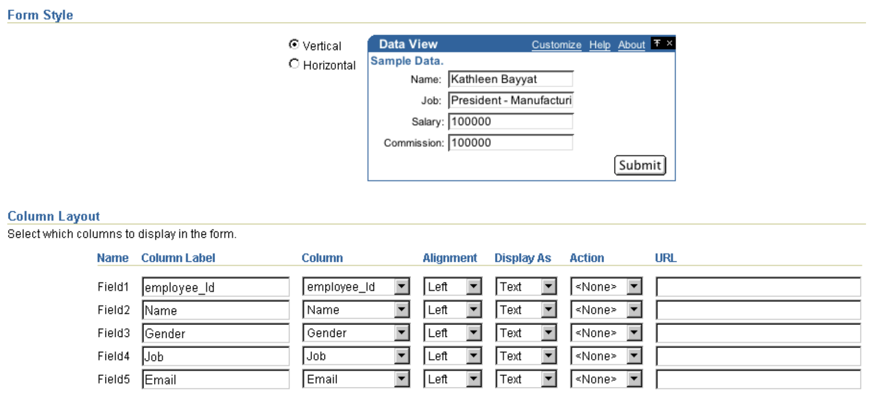
---

Note: For more information on using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

3.2.5.5 Form Layout

You can use the form layout (Figure 3–22) 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 and events to pass data to the selected row.

Figure 3–22 Layout Tab: Form


Form Style

[✓ Vertical] Horizontal

Sample Data.

Name:	Kathleen Bayyat
Job:	President - Manufacturi
Salary:	100000
Commission:	100000

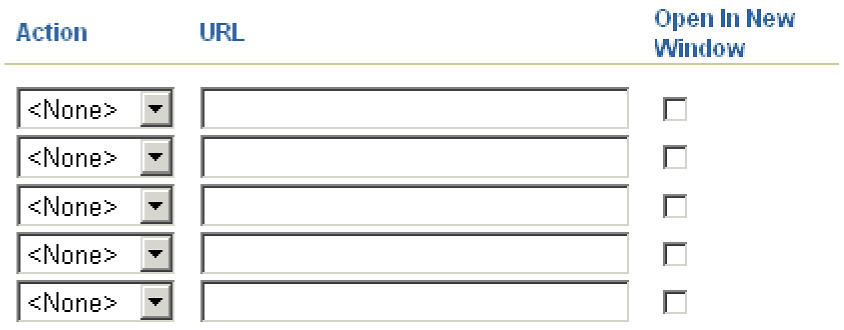
Column Layout

Select which columns to display in the form.

Name	Column Label	Column	Alignment	Display As	Action	URL
Field1	employee_Id	employee_Id	Left	Text	<None>	
Field2	Name	Name	Left	Text	<None>	
Field3	Gender	Gender	Left	Text	<None>	
Field4	Job	Job	Left	Text	<None>	
Field5	Email	Email	Left	Text	<None>	

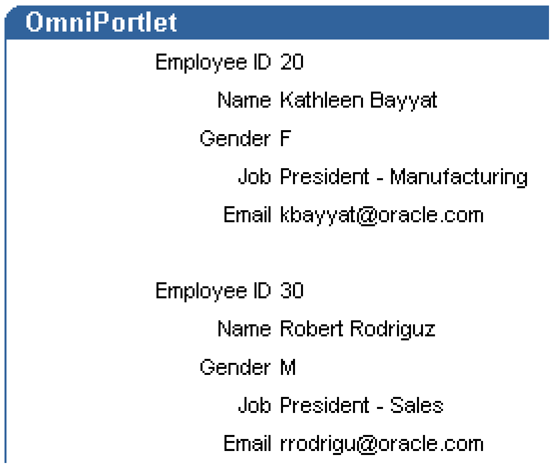
You can also specify whether to display the target of a URL in a new window (Figure 3–23). Figure 3–24 shows an example of an OmniPortlet using a Form layout.

Figure 3–23 Open In New Window Check Box



Action	URL	Open In New Window
<None>		☐
<None>		☐
<None>		☐
<None>		☐
<None>		☐

Figure 3–24 Example of an OmniPortlet Using a Form Layout


OmniPortlet

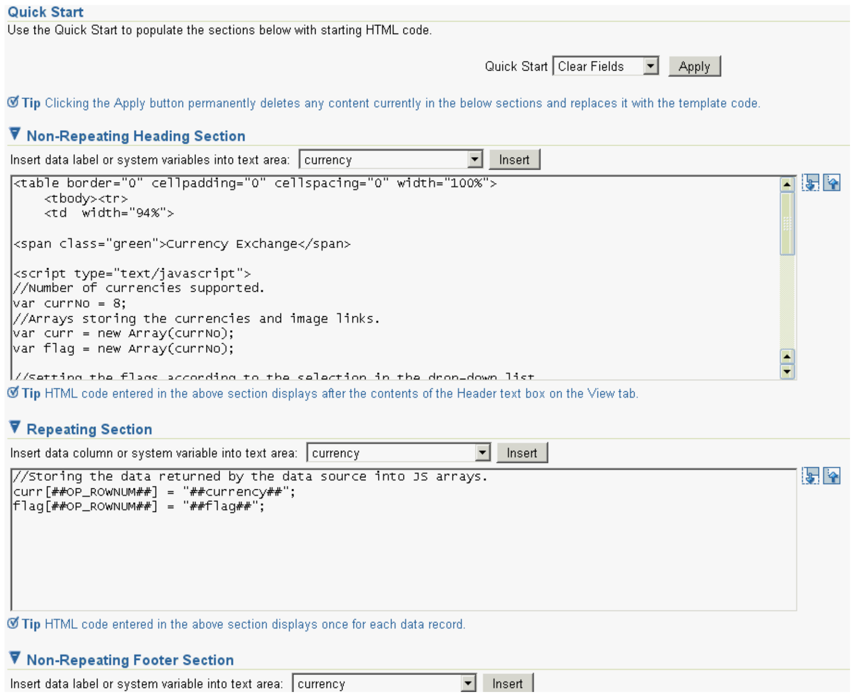
Employee ID	20
Name	Kathleen Bayyat
Gender	F
Job	President - Manufacturing
Email	kbayyat@oracle.com
Employee ID	30
Name	Robert Rodriguz
Gender	M
Job	President - Sales
Email	rrodrigu@oracle.com

Note: For more information on using the OmniPortlet Wizard, click the **Help** link in the upper right corner of the Layout tab.

3.2.5.6 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 3–25). By coding your own HTML and JavaScript, you have full control over the appearance and can develop a rich interface for your portlet. For an example of using JavaScript in the HTML layout, choose the Sortable Table layout from the Quick Start list on this tab.

Figure 3–25 Layout Tab: HTML



Quick Start

Use the Quick Start to populate the sections below with starting HTML code.

Quick Start: Clear Fields [Apply]

Tip Clicking the Apply button permanently deletes any content currently in the below sections and replaces it with the template code.

Non-Repeating Heading Section

Insert data label or system variables into text area: currency [Insert]

```
<table border="0" cellpadding="0" cellspacing="0" width="100%">
    <tbody><tr>
    <td  width="94%">

<span class="green">Currency Exchange</span>

<script type="text/javascript">
//Number of currencies supported.
var currNo = 8;
//Arrays storing the currencies and image links.
var curr = new Array(currNo);
var flag = new Array(currNo);

//Setting the flags according to the selection in the drop-down list
```

Tip HTML code entered in the above section displays after the contents of the Header text box on the View tab.

Repeating Section

Insert data column or system variable into text area: currency [Insert]

```
//Storing the data returned by the data source into JS arrays.
curr[##OP_ROWNUM##] = "##currency##";
flag[##OP_ROWNUM##] = "##flag##";
```

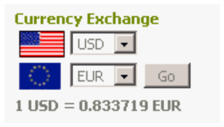
Tip HTML code entered in the above section displays once for each data record.

Non-Repeating Footer Section

Insert data label or system variable into text area: currency [Insert]

Figure 3–26 shows an example of an OmniPortlet using the HTML layout.

Figure 3–26 Example of an OmniPortlet Using the HTML Layout



3.2.6 Edit Defaults mode

After you have created your OmniPortlet and returned to your portal page, you can click the Edit Defaults icon to change the portlet options if required. You will notice that, in the Edit Defaults 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. There is also one extra tab, the Events tab, which is explained in the next section.

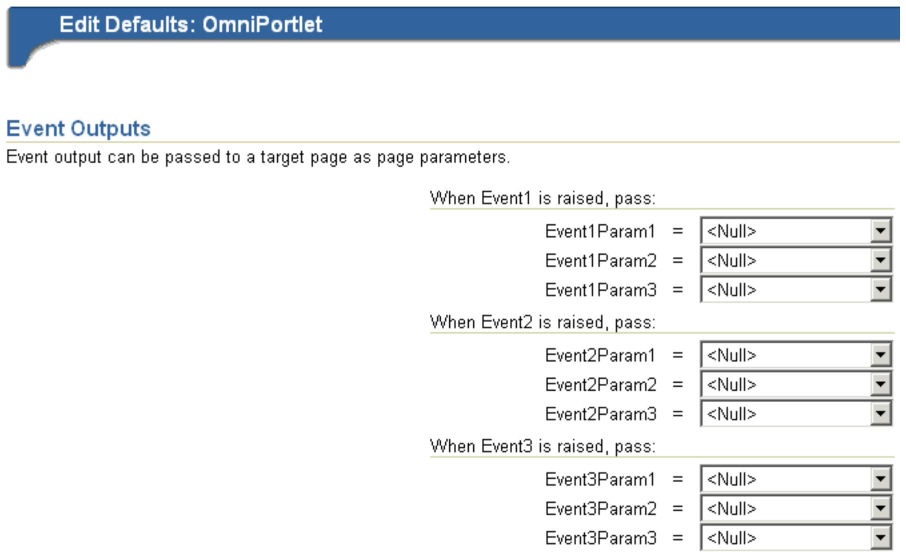
When you edit an OmniPortlet using the Edit Defaults mode, keep in mind the following notes:

- A new mode, "none," is the default setting for the Locale Personalization Level of OmniPortlet and the Simple Parameter form. This mode indicates that, when you edit the portlet defaults using the Edit Defaults mode, the changes apply to all users, regardless of the current Oracle Portal session language and the locale of your browser. For more information about these settings, see *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.
- You can personalize the portlet at runtime by clicking the **Personalize** link on the portlet. When you personalize the portlet, a complete copy of the personalization object is created. Since all properties are duplicated, subsequently modifying the portlet through Edit Defaults will not be reflected in the personalized version of the portlet. To ensure the latest changes are made to the portlet, you must click **Personalize** again (after the modifications from the Edit Defaults wizard are made), then select the **Reset to Defaults** option.
- By default, the OmniPortlet provider 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, you must use a shared Preference Store, such as the database Preference Store (DBPreferenceStore). To do so, you can choose to do one of the following:
    - Use a file-based Preference Store now, then migrate to the database Preference Store later using the PDK Preference Store Migration Utility.
    - Configure OmniPortlet to use the DBPreferenceStore, and follow the steps on configuring Portal Tools and Web providers in *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

3.2.7 Events

On the Events tab (Figure 3–27) in the Edit Defaults mode of the OmniPortlet Wizard, you can identify event parameters based on the portlet parameters you selected on the Source tab.

Figure 3–27 Events Tab of the OmniPortlet Wizard



Edit Defaults: OmniPortlet

Event Outputs

Event output can be passed to a target page as page parameters.

When Event1 is raised, pass:

Event1Param1	=	<Null>
Event1Param2	=	<Null>
Event1Param3	=	<Null>

When Event2 is raised, pass:

Event2Param1	=	<Null>
Event2Param2	=	<Null>
Event2Param3	=	<Null>

When Event3 is raised, pass:

Event3Param1	=	<Null>
Event3Param2	=	<Null>
Event3Param3	=	<Null>

3.3 Parameters and Events

Out of the box, OmniPortlet can receive up to five parameters and raise up to three events. Each of the events can send one or more parameters. For example, you can set up a chart that displays the employees in a department. When the user clicks one piece of the chart (for example, a department name), an event is raised that sends a parameter to the page. The page may then pass a parameter to all the portlets on that page that display information about the employees. Then, all the portlets on the page display information about the employees in the selected department.

Note: To learn how to use parameters and events with OmniPortlet, follow the steps in Chapter 4, "Building Example Portlets with OmniPortlet". If you are comfortable with the `provider.xml` file, you can add more parameters and events by editing the file.

To set up parameters and events, you must first enable the page group to accept parameters and events. In Oracle Portal, parameters and events are enabled by default. Then, you set up each portlet to accept the necessary parameters, and raise the required events. After you've set up the portlet parameters, you can link the portlets together by setting up the page-level parameters and events.

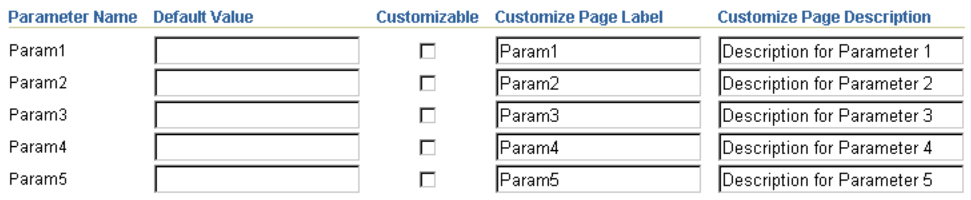
3.3.1 Portlet Parameters and Events

Out of the box, you can define up to five portlet parameters for an OmniPortlet. You can define parameters in the following places:

- On the Source tab of the wizard when you define the OmniPortlet
- On the Source tab when you select Edit Defaults for the OmniPortlet

Figure 3–28 shows the Portlet Parameters section on the Source tab.

Figure 3–28 Source Tab: Portlet Parameters Section



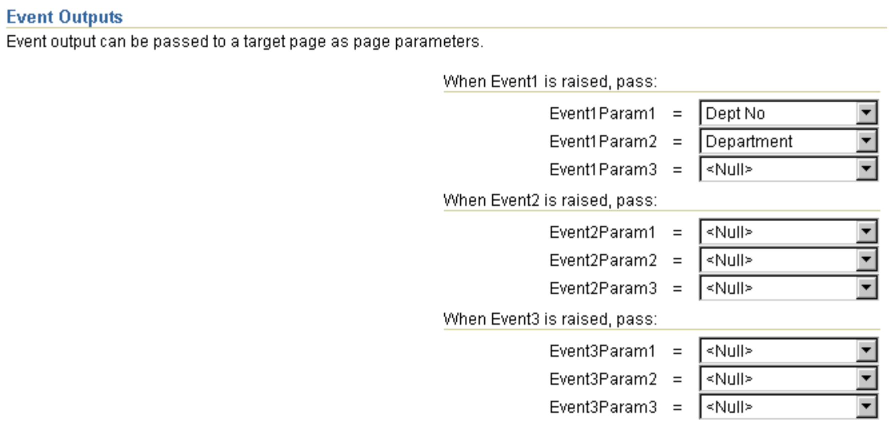
Parameter Name	Default Value	Customizable	Customize Page Label	Customize Page Description
Param1		☐	Param1	Description for Parameter 1
Param2		☐	Param2	Description for Parameter 2
Param3		☐	Param3	Description for Parameter 3
Param4		☐	Param4	Description for Parameter 4
Param5		☐	Param5	Description for Parameter 5

Parameter values determine what data is displayed in the portlet. You can also use a parameter to pass a value in a URL or to embed a value in the portlet text.

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. You can also refer to *Oracle Fusion Middleware User's Guide for Oracle Portal*.

You can set up each OmniPortlet to raise up to three events. Each event can pass up to three parameters. Each parameter can be a portlet parameter, such as Param1, or a data source column, such as Department_No. You set up events on the Events tab in the Edit Defaults mode of OmniPortlet (Figure 3–29).

Figure 3–29 Events Tab



Event Outputs

Event output can be passed to a target page as page parameters.

When Event1 is raised, pass:

Event1Param1	=	Dept No
Event1Param2	=	Department
Event1Param3	=	<Null>

When Event2 is raised, pass:

Event2Param1	=	<Null>
Event2Param2	=	<Null>
Event2Param3	=	<Null>

When Event3 is raised, pass:

Event3Param1	=	<Null>
Event3Param2	=	<Null>
Event3Param3	=	<Null>

3.3.2 Page Parameters and Events

After you've set up the parameters and events for each OmniPortlet on a portal page, you can map the portlet parameters and events to other portlets on the same page. For more information on using page parameters and events, see the Oracle Portal online help and *Oracle Fusion Middleware User's Guide for Oracle Portal*.

4

Building Example Portlets with OmniPortlet

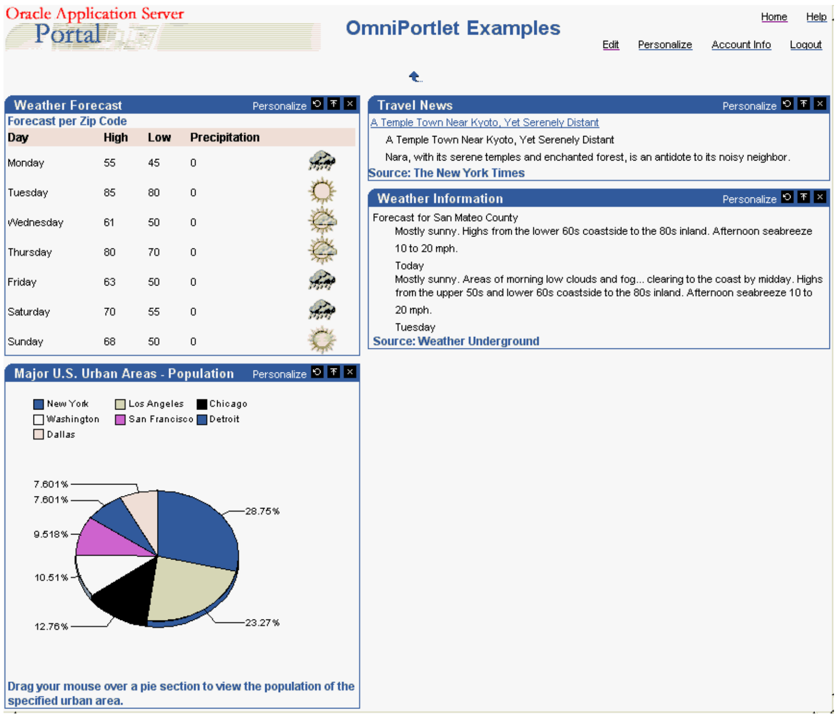
This chapter shows you how to use OmniPortlet to create four portlets based on different data sources: a Web service, a spreadsheet (CSV), XML, and an existing Web page. You will build these portlets based on the various built-in layouts, and will create a separate portlet based on an HTML layout. You will learn how to create and modify these portlets, as well as use page parameters and events to add interactivity to your portal page.

This chapter includes the following sections:

Note: To learn more about specific pages and tabs in OmniPortlet, click the **Help** link in the top right corner of the wizard. The online Help describes the contents of the selected page or tab. You can also find more information about using OmniPortlet in Chapter 3, "Creating Portlets with OmniPortlet." or about registering and configuring OmniPortlet with Oracle Portal in *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

At the end of this chapter, you will create a page that contains four portlets, as shown in Figure 4–1.

Figure 4–1 Portal Page with Four OmniPortlet Examples



Weather Forecast

Forecast per Zip Code

Day	High	Low	Precipitation
Monday	55	45	0
Tuesday	85	80	0
Wednesday	61	50	0
Thursday	80	70	0
Friday	63	50	0
Saturday	70	55	0
Sunday	68	50	0

Travel News

A Temple Town Near Kyoto, Yet Serenely Distant

A Temple Town Near Kyoto, Yet Serenely Distant

Nara, with its serene temples and enchanted forest, is an antidote to its noisy neighbor.

Source: The New York Times

Weather Information

Forecast for San Mateo County

Mostly sunny. Highs from the lower 60s coastside to the 80s inland. Afternoon seabreeze 10 to 20 mph.

Today

Mostly sunny. Areas of morning low clouds and fog... clearing to the coast by midday. Highs from the upper 50s and lower 60s coastside to the 80s inland. Afternoon seabreeze 10 to 20 mph.

Tuesday

Source: Weather Underground

Major U.S. Urban Areas - Population

Drag your mouse over a pie section to view the population of the specified urban area.

More On OTN

All four of these example portlets require you to be able to connect to the Internet. If you must use a proxy server to connect to the Internet, you will need to configure the HTTP proxy settings on the OmniPortlet Provider Test page to use proxy authentication. If the OmniPortlet provider has been set up to use proxy authentication that requires your login, you can enter your user information in the Proxy Authentication section of the Source tab in the OmniPortlet wizard. For more information on proxy authentication, refer to Section 3.2.2, "Source." For specific information on configuring the provider's proxy settings, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Note: The steps in this chapter assume that you are using the OmniPortlet provider that is available with Oracle Portal. If you installed the Oracle Portal Developer Kit separately, you may need to slightly modify the instructions when adding an OmniPortlet instance to the page. Instructions for registering and configuring OmniPortlet with Oracle Portal are located in *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

4.1 Adding an OmniPortlet Instance to a Portal Page

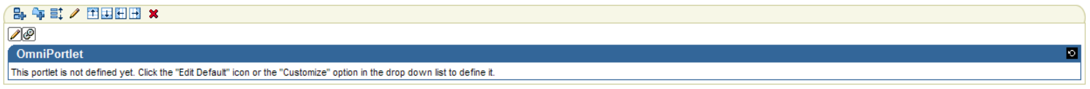
In this section, you will learn how to add an OmniPortlet instance to your portal page.

To add an OmniPortlet instance to a page, perform the following steps:

1. In the Edit mode of the page where you want to add the OmniPortlet, click the **Add Portlets** icon.
2. On the Add Portlets page, in the **Available Portlets** list, click the **Portlet Builders** link.
3. Click the **OmniPortlet** link.

4. Click **OK**. The new instance of OmniPortlet now displays on your portal page, as shown in Figure 4–2.

Figure 4–2 OmniPortlet Instance on a Portal Page



This portlet is not defined yet. Click the "Edit Default" icon or the "Customize" option in the drop down list to define it.

Note: On the IBM Linux on Power platform, if the action buttons (Next, Previous, Finish, and Cancel) are minimized to dots when defining the OmniPortlet, increase the stack size shell limit to unlimited and restart the WLS_PORTAL instance. Run the following command to set the stack size shell limit to unlimited:

```
prompt> ulimit -s unlimited
```

4.2 Building an OmniPortlet Based on a Web Service

The steps in this section will show you how to create a portlet that displays weather forecast information for a particular zip code. You will base this portlet on the Web Service available on the Oracle Technology Network. You will need Internet access to be able to complete this section.

To create an OmniPortlet based on a Web Service, perform the following steps:

1. In the new OmniPortlet instance on your portal page, click the **Edit Default** link to start the OmniPortlet Wizard.
2. On the Type page, select the **Web Service** radio button, then click **Next**.
3. On the Source page, in the **WSDL URL** field, enter the following URL:

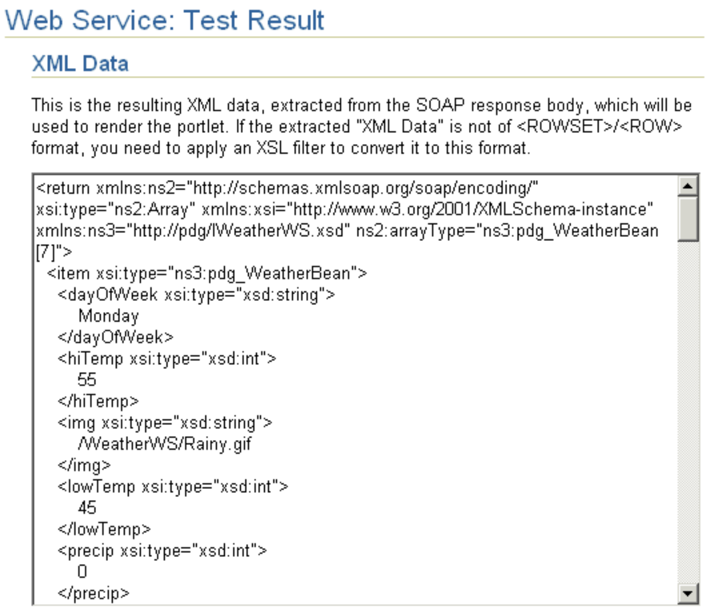
```
http://webservices.oracle.com/WeatherWS/WeatherWS?WSDL
```

Note: This Web Service has one method (`WeatherWS.giveMeSomeWeatherInfo`) and accepts one parameter (`param0`). If you use a method that has parameters, the parameters will display in this section of the tab. You can enter a sample value for the parameter, then click **Test** to view the sample XML data, the SOAP response, and the SOAP Request.

---

4. Click **Show Methods**.
5. In the Web Service Methods section, in the **param0** parameter field, enter a sample Zip code (for example, `94065`), then click **Test**. The Web Service: Test Result window displays, where you can verify the XML data returned by the Web Service, as shown in Figure 4–3.

Figure 4–3 OmniPortlet: Web Service Test Results Page



Web Service: Test Result

XML Data

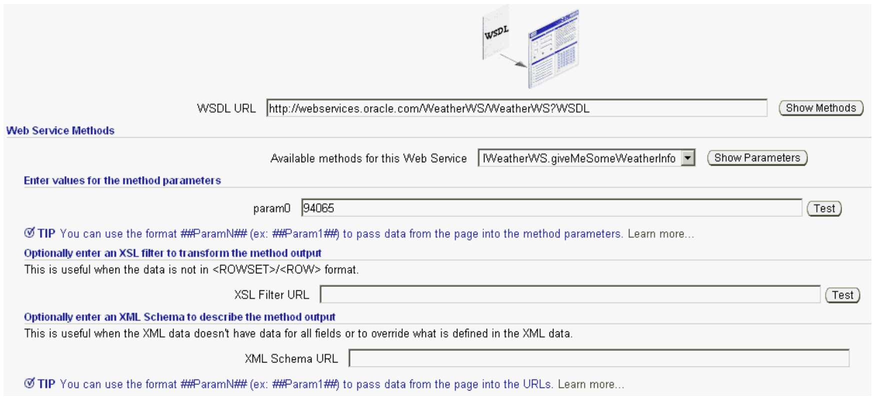
This is the resulting XML data, extracted from the SOAP response body, which will be used to render the portlet. If the extracted "XML Data" is not of <ROWSET>/<ROW> format, you need to apply an XSL filter to convert it to this format.

```
<return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:Array" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns3="http://pdg/IWeatherWS.xsd" ns2:arrayType="ns3:pdg_WeatherBean
[7]">
  <item xsi:type="ns3:pdg_WeatherBean">
    <dayOfWeek xsi:type="xsd:string">
      Monday
    </dayOfWeek>
    <hiTemp xsi:type="xsd:int">
      55
    </hiTemp>
    <img xsi:type="xsd:string">
      /WeatherWS/Rainy.gif
    </img>
    <lowTemp xsi:type="xsd:int">
      45
    </lowTemp>
    <precip xsi:type="xsd:int">
      0
    </precip>
```

6. Close the window.

   The Source page should look like the image shown in Figure 4–4.

Figure 4–4 OmniPortlet: Web Service Source Tab



WSDL URL: http://webservices.oracle.com/WeatherWS/WeatherWS?WSDL [Show Methods]

Web Service Methods

Available methods for this Web Service: IWeatherWS.giveMeSomeWeatherInfo [Show Parameters]

Enter values for the method parameters

param0: 94065 [Test]

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the method parameters. Learn more...

Optionally enter an XSL filter to transform the method output

This is useful when the data is not in <ROWSET>/<ROW> format.

XSL Filter URL: [Test]

Optionally enter an XML Schema to describe the method output

This is useful when the XML data doesn't have data for all fields or to override what is defined in the XML data.

XML Schema URL:

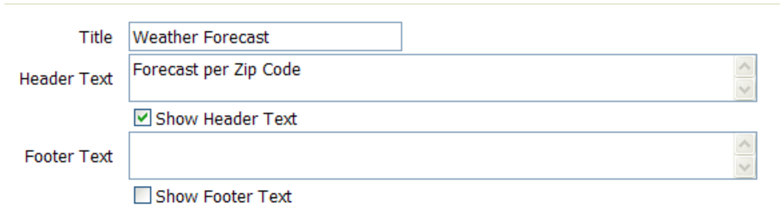
TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the URLs. Learn more...

Note: If you do not have access to the Internet, you can use a different Web Service, but keep in mind that your results will not match the example in this chapter.

---

7. Click **Next**.
8. On the Filter page, click **Next**.
9. On the View page, in the Title field, enter `Weather Forecast`.
10. In the Header Text field, enter `Forecast per Zip Code`.
11. Make sure the **Show Header Text** check box is selected, and clear the **Show Footer Text** check box, as shown in Figure 4–5.

Figure 4–5 OmniPortlet: Web Service View Tab



Title	Weather Forecast
Header Text	Forecast per Zip Code
	[✓] Show Header Text
Footer Text	
	[ ] Show Footer Text

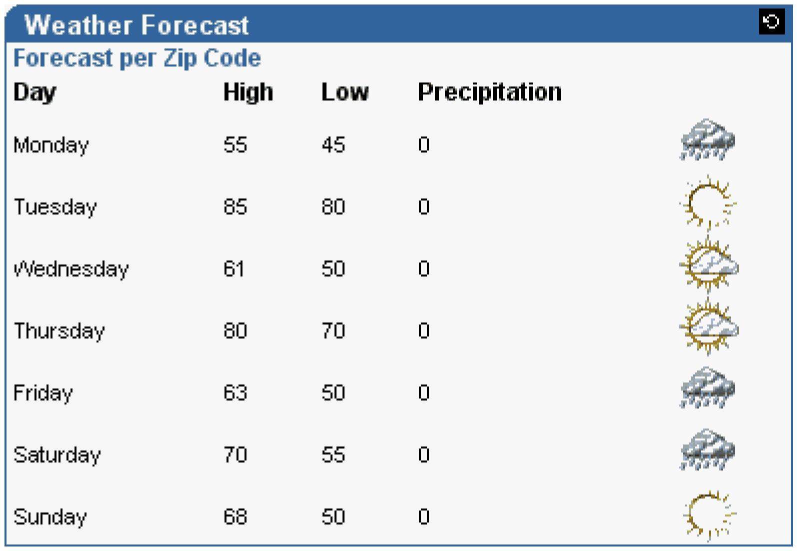
12. Make sure the **Tabular** radio button is selected, then click **Next**.
13. On the Layout page, select the **Plain** radio button.
14. Specify the **Column Label**, **Column**, and **Display As** properties for your data according to Table 4–1.

Table 4–1 Column Properties for the Weather Forecast Portlet

Column Label	Column	Display As
Day	dayOfWeek	Text
High	hiTemp	Text
Low	lowTemp	Text
Precipitation	precip	Text
(Blank)	img	Image

15. Now that you've completed defining the portlet, click **Finish**. Your portlet should look like Figure 4–6.

Figure 4–6 OmniPortlet: Web Service Portlet



Weather Forecast

Forecast per Zip Code

Day	High	Low	Precipitation
Monday	55	45	0
Tuesday	85	80	0
Wednesday	61	50	0
Thursday	80	70	0
Friday	63	50	0
Saturday	70	55	0
Sunday	68	50	0

4.3 Building an OmniPortlet Based on a Spreadsheet (CSV)

The steps in this section show you how to use OmniPortlet to define a portlet that displays regional information based on a spreadsheet (CSV) data source. This portlet will display the population of major U.S. urban areas in a pie chart.

To create an OmniPortlet based on a spreadsheet, perform the following steps:

1. Create a region below the existing region on your portal page.
2. Add an OmniPortlet to this new region.
3. Start the OmniPortlet Wizard by clicking the **Edit Default** link.
4. On the Type page, select the **Spreadsheet** radio button, then click **Next**.
5. On the Source page, in the **CSV URL** field, replace the existing text with the following CSV:

   ```
   http://webservices.oracle.com/WeatherWS/city_population.csv
   ```

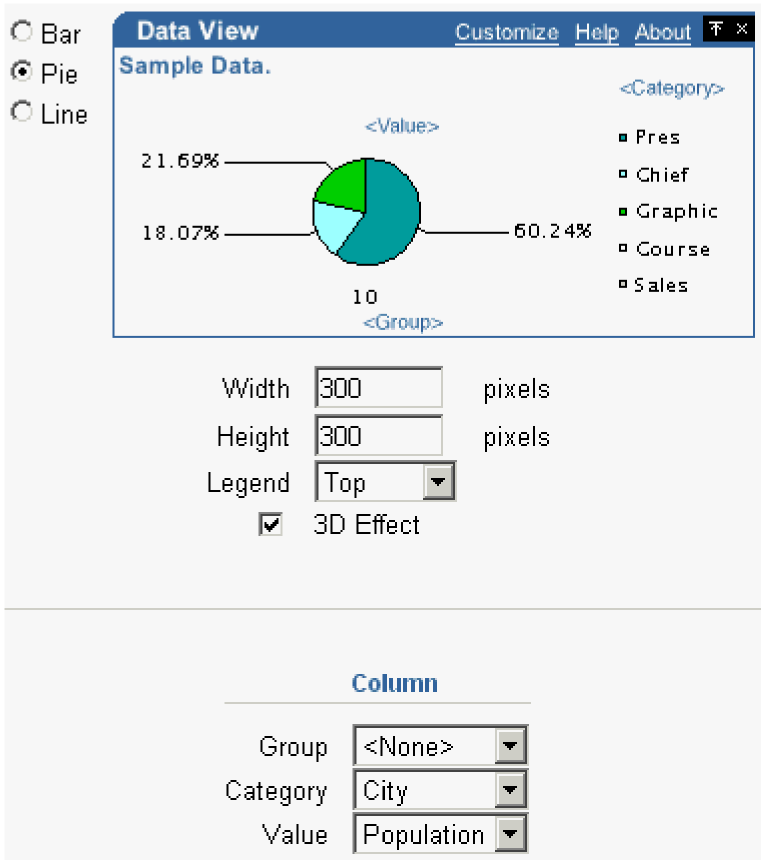
   Note: If you do not have access to the Internet, you can use the default URL, but keep in mind that your results will not match the example in this chapter.

6. Ensure that the **Use first row of spreadsheet for column names** check box is selected
7. Click **Next**.
8. On the Filter page, click **Next**.
9. On the View page, in the **Title** field, enter `Major U.S. Urban Areas - Population`.
10. Clear the **Show Header Text** check box.
11. In the Footer Text field, enter `Click a pie section to view the population of the specified urban area.`

    Note: The text you just entered in the Footer Text field instructs your end users to click a pie section to view more details about the selected population. To enable this feature, you will need to complete the steps in Section 4.6, "Setting Up Portlet Parameters and Events".

12. Ensure that the **Show Footer Text** check box is selected.
13. Under Layout Style, select the **Chart** radio button, then click **Next**.
14. On the Layout page, select the **Pie** radio button.
15. In the **Width** field, enter `300`.
16. In the **Height** field, enter `300`.
17. From the **Legend** list, choose **Top**.
18. Select the **3D Effect** check box.
19. Under Column Layout, from the **Group** list, choose **<None>**.
20. From the **Category** list, choose **City**.
21. From the **Value** list, choose **Population**. The Layout tab should now look like the image shown in Figure 4–7.

Figure 4–7 OmniPortlet: Character-Separated Values (CSV) Layout Tab



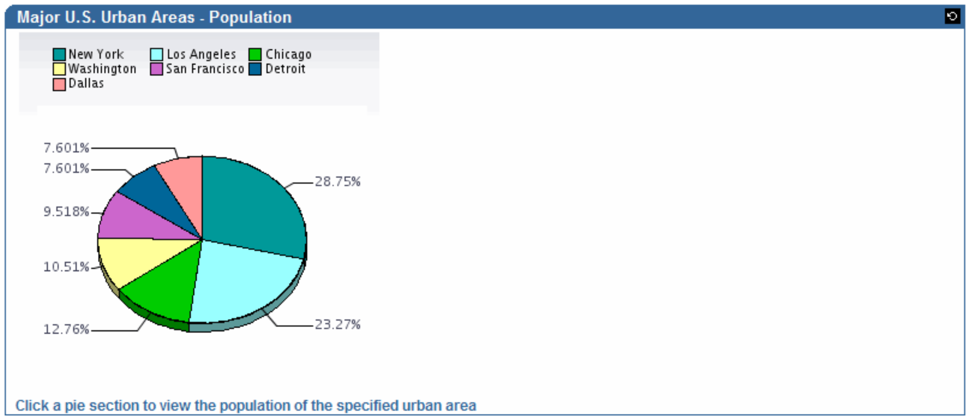
Width 300 pixels
Height 300 pixels
Legend Top
[✓] 3D Effect

Group <None>
Category City
Value Population

22. Click **Finish**.

   Your portlet now displays on your portal page below the Weather Forecast portlet, and should look like Figure 4–8.

Figure 4–8 OmniPortlet: CSV OmniPortlet on the Page


Major U.S. Urban Areas - Population
New York
Los Angeles
Chicago
Washington
San Francisco
Detroit
Dallas
7.601%
7.601%
9.518%
10.51%
12.76%
28.75%
23.27%
Click a pie section to view the population of the specified urban area

4.4 Building an OmniPortlet Based on an XML Data Source

The steps in this section will show you how to use OmniPortlet to define a portlet that displays news information in a scrolling layout, based on an XML data source.

To create an OmniPortlet based on an XML data source, perform the following steps:

1. Create a region next to the Web Services portlet (Weather Forecast) on your portal page.
2. Add an OmniPortlet to this new region.
3. Start the OmniPortlet Wizard by clicking the **Edit Default** link.
4. On the Type page, select the **XML** radio button, then click **Next**.
5. On the Source page, in the **XML URL** field, enter the following URL:

   ```
   http://www.nytimes.com/services/xml/rss/nyt/Travel.xml
   ```

   Note: If you do not have access to the Internet, you can use a different RSS feed, but keep in mind that your results will not match the images in this example.

6. In the **XSL Filter URL** field, enter the following URL for the XSL file:

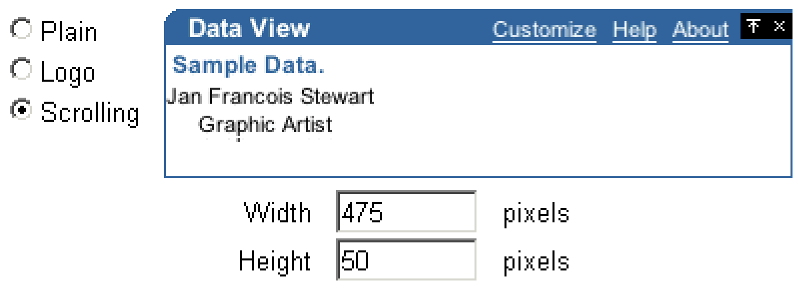
   ```
   http://www.oracle.com/technology/products/ias/portal/viewlets
   /omniportletnews.xsl
   ```

   Note: This XSL filter transforms the RSS news feed into the ROWSET/ROW structure that OmniPortlet can consume.

7. Click **Next**.
8. On the Filter page, click **Next**.
9. On the View page, in the **Title** field, enter `Travel News`.
10. Clear the **Show Header Text** check box.
11. In the Footer Text field, enter `Source: The New York Times`.
12. Ensure that the **Show Footer Text** check box is selected.
13. Under Layout Style, select the **News** radio button, then click **Next**.
14. On the Layout page, under News Style, select the **Scrolling** radio button.
15. In the **Width** field, enter `475`.
16. In the **Height** field, enter `50`.

    The News Style section of the Layout tab should now look like the image shown in Figure 4–9.

Figure 4–9 OmniPortlet: XML Layout Tab - News Style

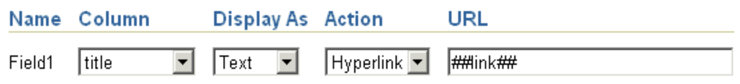


Plain
Logo
[✓ Scrolling]

Width 475 pixels
Height 50 pixels

17. Under Column Layout, next to Field 1 choose **title** from the **Column** list, **Hyperlink** from the Action list, and enter `##link##` in the URL field, as shown in Figure 4–10.

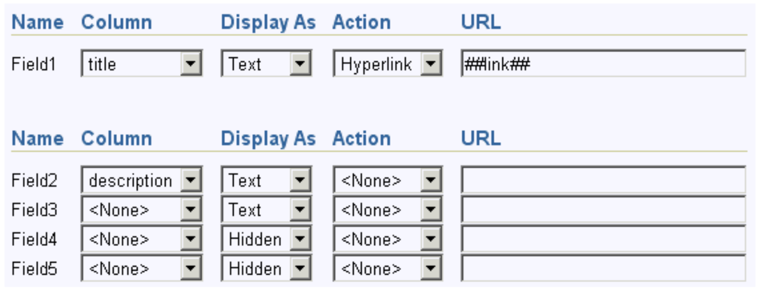
Figure 4–10 OmniPortlet: News Style Column Layout



Name	Column	Display As	Action	URL
Field1	title	Text	Hyperlink	##link##

18. Next to Field2, choose **description** from the **Column** list and leave the default values for the other settings. Ensure this section of the Layout tab looks like Figure 4–11.

Figure 4–11 Column Layout Section of the Layout Tab



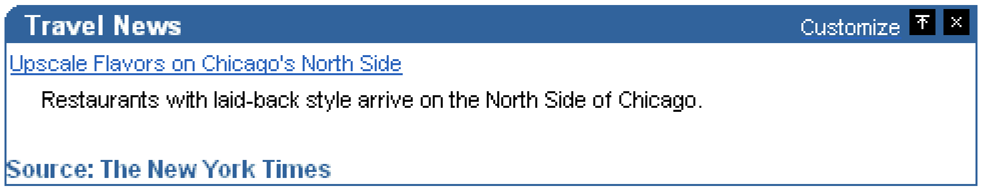
Name	Column	Display As	Action	URL
Field1	title	Text	Hyperlink	##link##

Name	Column	Display As	Action	URL
Field2	description	Text	<None>	
Field3	<None>	Text	<None>	
Field4	<None>	Hidden	<None>	
Field5	<None>	Hidden	<None>	

19. Click **Finish**.

    Your portlet now displays on your portal page, as shown in Figure 4–12.

Figure 4–12 OmniPortlet: XML Scrolling News OmniPortlet on the Page



Travel News
Customize

Upscale Flavors on Chicago's North Side

Restaurants with laid-back style arrive on the North Side of Chicago.

Source: The New York Times

4.5 Building an OmniPortlet Based on a Web Page Data Source

The steps in this section will show you how to use OmniPortlet to create a portlet that displays weather information in a text format based on an existing Web Page. In this section, you will use OmniPortlet to clip and scrape content from Web sites.

To create an OmniPortlet based on a Web page data source, perform the following steps:

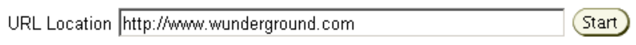
1. Create a region below the XML portlet (Travel News) on your portal page.
2. Add an OmniPortlet to this new region.
3. Start the OmniPortlet Wizard by clicking the **Edit Default** link.
4. On the Type tab of the OmniPortlet Wizard, select the **Web Page** radio button, then click **Next**.
5. On the Source tab, click **Select Web Page**.

6. On the Web Clipping Studio page that displays, in the **URL Location** field (Figure 4–13), enter the URL of the page you want to clip:

   ```
   http://www.wunderground.com
   ```

   Note: You can use a different Web page, but keep in mind that your results will not match the example in this chapter.

Figure 4–13 URL Location Field



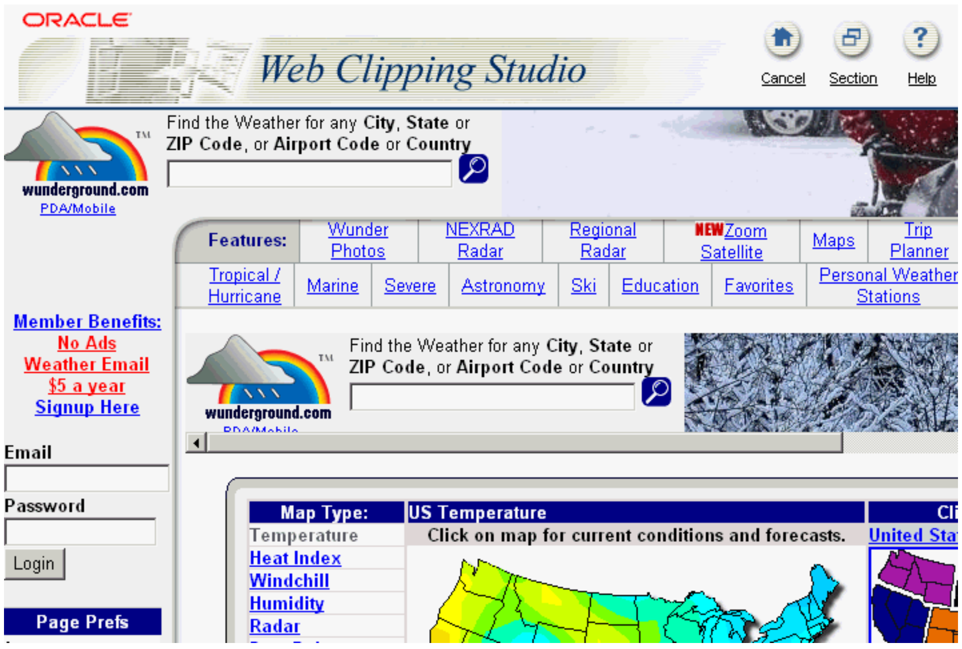
URL Location http://www.wunderground.com Start

Note: In this example, we use a third party Web site owned by The Weather Underground, Inc. Because this Web site is continually updated based on current weather forecasts, the images and steps included in this section may not reflect exactly what you see when you create this example. As you go further into this example, some of the steps may not work, as the owners may change the technology of this third party Web site.

---

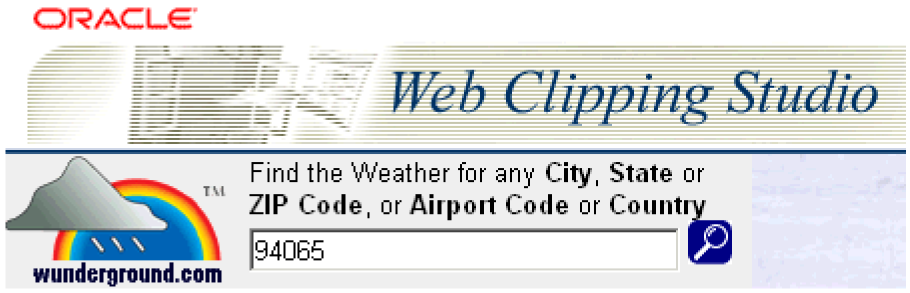
7. Click **Start** to display the Web Clipping Studio. You should see the Web page display in the Web Clipping Studio, as shown in Figure 4–14.

Figure 4–14 Web Clipping Studio Containing the www.wunderground.com Home Page



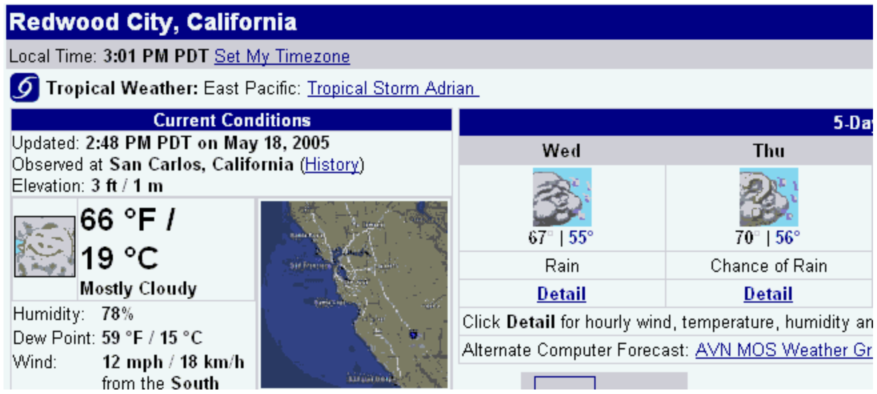
8. On the home page, enter a zip code (for example, `94065`) in the **Weather** field, as shown in Figure 4–15, then click the magnifying glass icon next to the field.

Figure 4–15 Entering the Zip Code 94065 into the Weather Field


ORACLE
Web Clipping Studio
Find the Weather for any City, State or ZIP Code, or Airport Code or Country
94065
wunderground.com

9. The weather information for Redwood City, California displays in the Web Clipping Studio. A segment of the page is shown in Figure 4–16.

Figure 4–16 Weather Information for Redwood City, California


Redwood City, California

Local Time: 3:01 PM PDT Set My Timezone

Tropical Weather: East Pacific: Tropical Storm Adrian

Current Conditions

Updated: 2:48 PM PDT on May 18, 2005
Observed at San Carlos, California (History)
Elevation: 3 ft / 1 m

66 °F /
19 °C
Mostly Cloudy

Humidity: 78%
Dew Point: 59 °F / 15 °C
Wind: 12 mph / 18 km/h from the South

5-Da

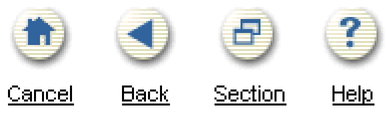
Wed	Thu
67° | 55°	70° | 56°
Rain	Chance of Rain
Detail	Detail

Click Detail for hourly wind, temperature, humidity an

Alternate Computer Forecast: AVN MOS Weather Gr

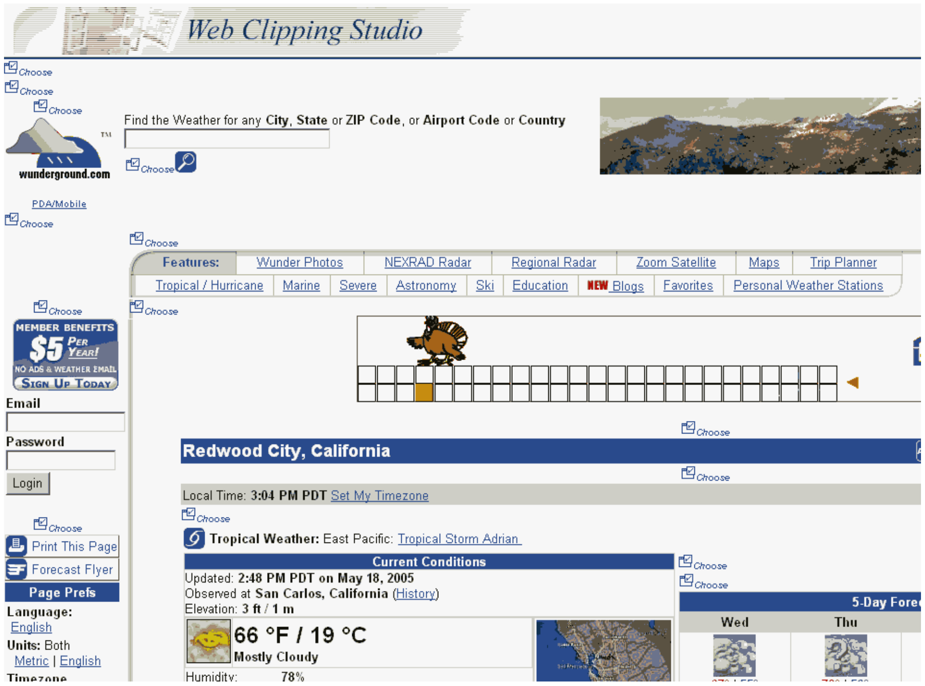
10. In the top right corner of the Web Clipping Studio, click the **Section** button, as shown in Figure 4–17.

Figure 4–17 The Section Button



11. After you click the Section button, you'll notice that the elements of the Web page are broken down, and that a new icon displays called Choose at the top of each section. Figure 4–18 shows a snapshot of the Web Clipping Studio.

Figure 4–18 Web Clipping Studio Displaying the Web Page Sections



Find the Weather for any City, State or ZIP Code, or Airport Code or Country

Local Time: 3:04 PM PDT Set My Timezone

Tropical Weather: East Pacific: Tropical Storm Adrian

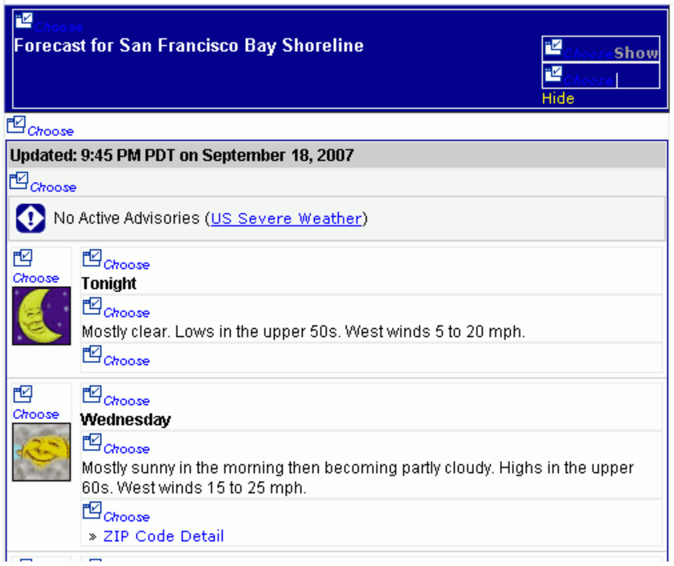
Updated: 2:48 PM PDT on May 18, 2005
Observed at San Carlos, California (History)
Elevation: 3 ft / 1 m

66 °F / 19 °C
Mostly Cloudy

Humidity: 78%

12. Find the section shown in Figure 4–19, which shows the weather information for the city you chose (in this example, San Francisco Bay Shoreline).

Figure 4–19 Weather Information for San Francisco Bay Shoreline with the Choose Icon



Forecast for San Francisco Bay Shoreline

Updated: 9:45 PM PDT on September 18, 2007

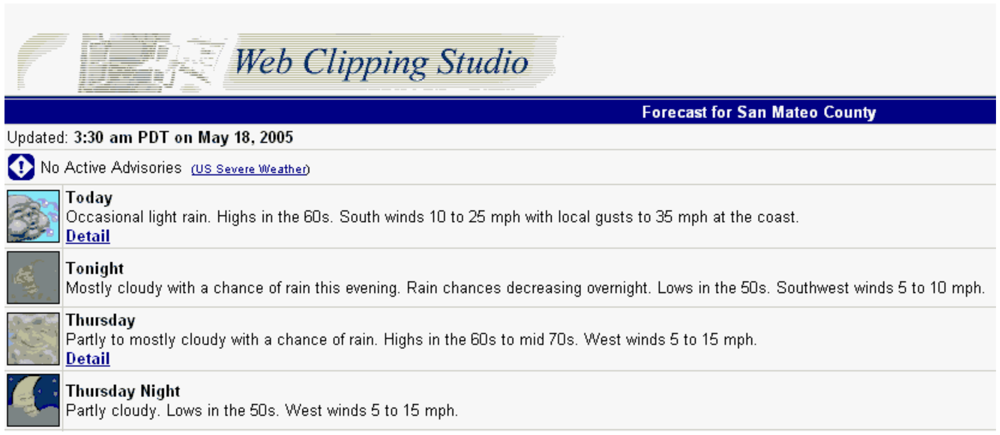
No Active Advisories (US Severe Weather)

Mostly clear. Lows in the upper 50s. West winds 5 to 20 mph.

Mostly sunny in the morning then becoming partly cloudy. Highs in the upper 60s. West winds 15 to 25 mph.

13. Click the **Choose** icon for this section, located directly above the "Forecast for San Francisco Bay Shoreline" title bar, as shown in Figure 4–19. The section displays in the Web Clipping Studio, as shown in Figure 4–20.

Figure 4–20 Weather Information Section for San Mateo County in the Web Clipping Studio



Forecast for San Mateo County

Updated: 3:30 am PDT on May 18, 2005

No Active Advisories (US Severe Weather)

Today
Occasional light rain. Highs in the 60s. South winds 10 to 25 mph with local gusts to 35 mph at the coast.
Detail

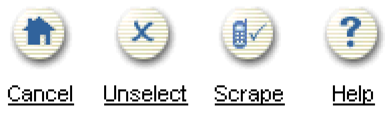
Tonight
Mostly cloudy with a chance of rain this evening. Rain chances decreasing overnight. Lows in the 50s. Southwest winds 5 to 10 mph.

Thursday
Partly to mostly cloudy with a chance of rain. Highs in the 60s to mid 70s. West winds 5 to 15 mph.
Detail

Thursday Night
Partly cloudy. Lows in the 50s. West winds 5 to 15 mph.

14. After you have chosen the clipping, you can refine your data further by scraping the data, that is, selecting specific cells you wish to display in your portlet. Click the **Scrape** button (Figure 4–21).

Figure 4–21 The Scrape Button

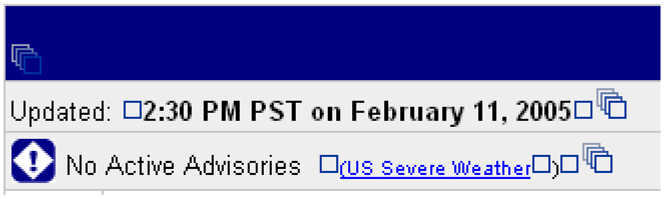


15. While in Scraping mode, you can identify the text pieces in the Web clipping by selecting the check boxes next to each item. You can repeat these items at the column level, row level, or table level.

    In this example, we want to show the title and the description of each resulting article from our search. We can repeat the title and description at the row level, so that each result returned by the search displays only the title and the description of every result. In general, you choose the text items in the first row that contains all the pieces you wish to repeat for each row.

    After you click the Scrape button, you'll notice that check boxes display next to each item on the Web page, as shown in Figure 4–22.

Figure 4–22 Scraping Check Boxes

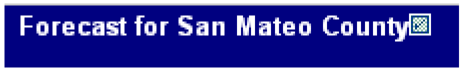


Updated: 2:30 PM PST on February 11, 2005

No Active Advisories (US Severe Weather)

16. Select the output you want by selecting the check boxes next to the items.Figure 4–23 shows an example of a check box.

Figure 4–23 Check Box for San Mateo County Title

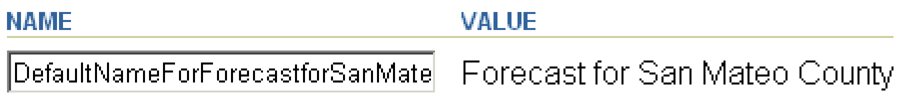


Note: Depending on the color of the section you want to select, the check boxes may not be prominently displayed.

---

17. After you select the check box for "Forecast for San Mateo County," notice that a corresponding label displays in the Data section of the Web Clipping Studio at the bottom of the screen, as shown in Figure 4–24.

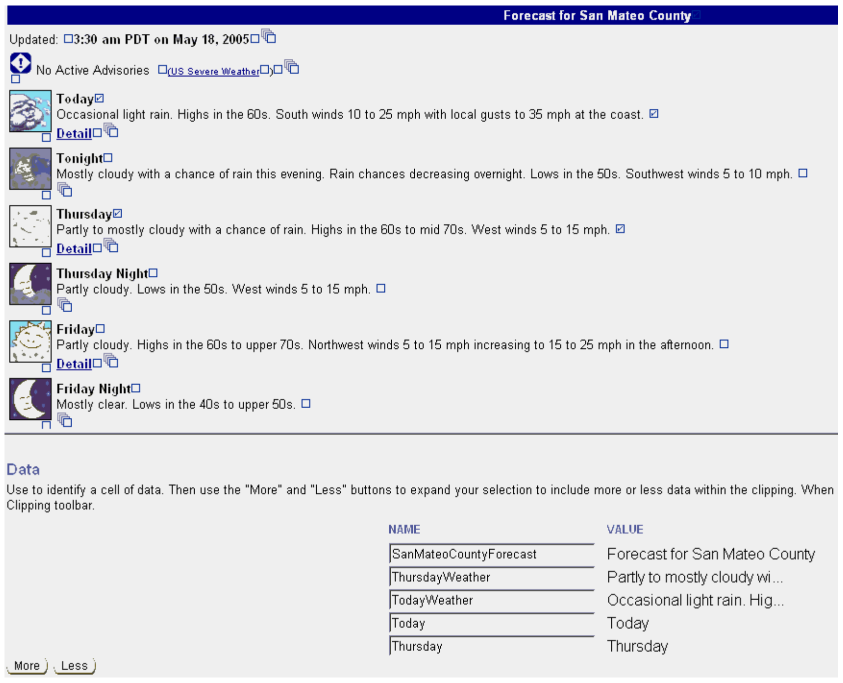
Figure 4–24 San Mateo County Label in Data Section



NAME	VALUE
DefaultNameForForecastforSanMate	Forecast for San Mateo County

18. In the Name field, enter a more meaningful name, such as "`SanMateoCountyForecast`." Do not include spaces in the name.
19. Select the fields you wish to display and add the corresponding labels as you select them. The following is a list of sample information we chose for this example, but you can choose your own examples depending on the current weather. You can see the following selected weather information in Figure 4–25:
   - Forecast for San Mateo County title
   - Today
   - Today's Weather
   - Thursday
   - Thursday's Weather

Figure 4–25 Selected Weather Information



Forecast for San Mateo County

Updated: 3:30 am PDT on May 18, 2005

No Active Advisories (US Severe Weather)

Today
Occasional light rain. Highs in the 60s. South winds 10 to 25 mph with local gusts to 35 mph at the coast.
Detail

Tonight
Mostly cloudy with a chance of rain this evening. Rain chances decreasing overnight. Lows in the 50s. Southwest winds 5 to 10 mph.

Thursday
Partly to mostly cloudy with a chance of rain. Highs in the 60s to mid 70s. West winds 5 to 15 mph.
Detail

Thursday Night
Partly cloudy. Lows in the 50s. West winds 5 to 15 mph.

Friday
Partly cloudy. Highs in the 60s to upper 70s. Northwest winds 5 to 15 mph increasing to 15 to 25 mph in the afternoon.
Detail

Friday Night
Mostly clear. Lows in the 40s to upper 50s.

Data

Use to identify a cell of data. Then use the "More" and "Less" buttons to expand your selection to include more or less data within the clipping. When
Clipping toolbar.

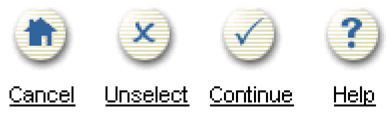
NAME	VALUE
SanMateoCountyForecast	Forecast for San Mateo County
ThursdayWeather	Partly to mostly cloudy wi...
TodayWeather	Occasional light rain. Hig...
Today	Today
Thursday	Thursday

Note: With the Web page data source, you can select URLs as part of your Web clipping. In Oracle Portal 10.1.2 and later, the context of the application is maintained. So, for example, any images that display on the hyperlinked page will be maintained.

---

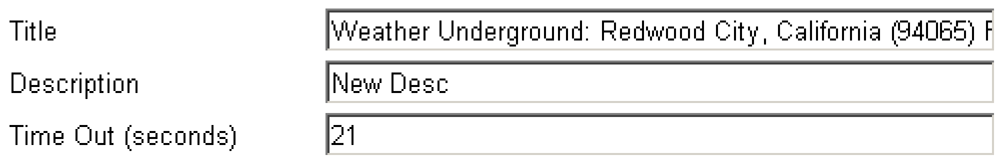
20. Now that you've selected the data you want to display, click the **Continue** button, as shown in Figure 4–26.

Figure 4–26 The Continue Button



21. On the page that displays, verify that the information in the Clipping Attributes section includes the title: "Weather Underground: Redwood City, California Forecast," as shown in Figure 4–27.

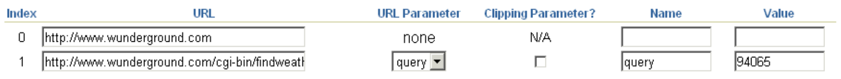
Figure 4–27 Clipping Attributes Section of the Web Clipping Studio



Title	Weather Underground: Redwood City, California (94065) F
Description	New Desc
Time Out (seconds)	21

22. Verify that the information in the Clipping Parameters section includes the parameters as shown in Figure 4–28.

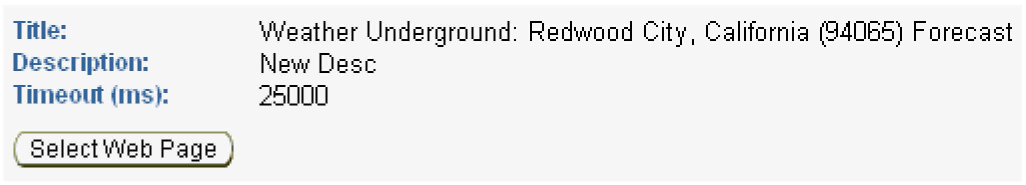
Figure 4–28 Clipping Parameters Section of the Web Clipping Studio



Index	URL	URL Parameter	Clipping Parameter?	Name	Value
0	http://www.wunderground.com	none	N/A		
1	http://www.wunderground.com/cgi-bin/findweatl	query	☐	query	94065

23. Select the **Clipping Parameter?** check box for the second parameter with the name "query."
24. Click **OK**.
25. On the Source tab of the OmniPortlet Wizard, the new title and description now display. To edit the Web clipping in the Web Clipping Studio, you can click the **Select Web Page** button again, as shown in Figure 4–29.

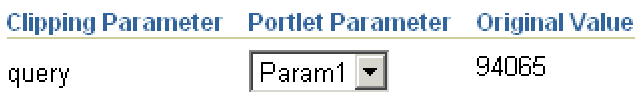
Figure 4–29 Web Page Source Tab



Title: Weather Underground: Redwood City, California (94065) Forecast
Description: New Desc
Timeout (ms): 25000

26. Under the Clipping Parameters heading, you should see the clipping parameter you set on the previous page, as shown in Figure 4–30:

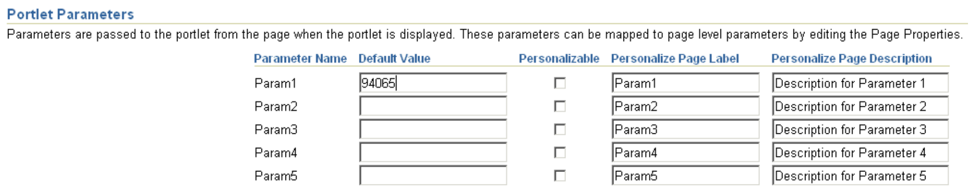
Figure 4–30 Clipping Parameter to Portlet Parameter Mapping



Clipping Parameter	Portlet Parameter	Original Value
query	Param1	94065

27. Under the Portlet Parameters heading, next to Param1, set the **Default Value** to the zip code, `94065`, as shown in Figure 4–31.

Figure 4–31 Portlet Parameters Section



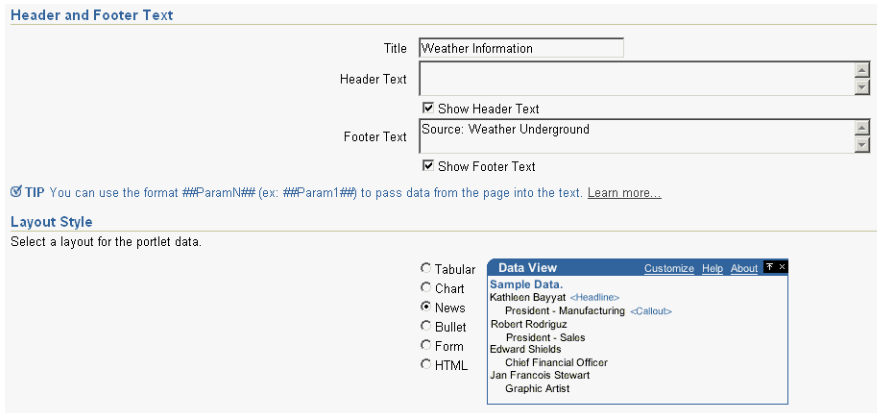
Portlet Parameters

Parameters are passed to the portlet from the page when the portlet is displayed. These parameters can be mapped to page level parameters by editing the Page Properties.

Parameter Name	Default Value	Personalizable	Personalize Page Label	Personalize Page Description
Param1	94065	☐	Param1	Description for Parameter 1
Param2		☐	Param2	Description for Parameter 2
Param3		☐	Param3	Description for Parameter 3
Param4		☐	Param4	Description for Parameter 4
Param5		☐	Param5	Description for Parameter 5

28. Click **Next**.
29. On the Filter tab, click **Next**.
30. On the View tab, in the Title field, enter `Weather Information`.
31. In the Footer Text field, enter `Source: Weather Underground` and make sure the **Show Footer Text** check box is selected.
32. Under Layout Style, select the **News** radio button. The View tab should look like Figure 4–32.

Figure 4–32 View Tab with the Options Selected



Header and Footer Text

Title: Weather Information

Header Text:

[✓ Show Header Text]

Footer Text: Source: Weather Underground

[✓ Show Footer Text]

TIP You can use the format ##ParamN## (ex: ##Param1##) to pass data from the page into the text. Learn more...

Layout Style

Select a layout for the portlet data.

Tabular Chart [✓ News] Bullet Form HTML

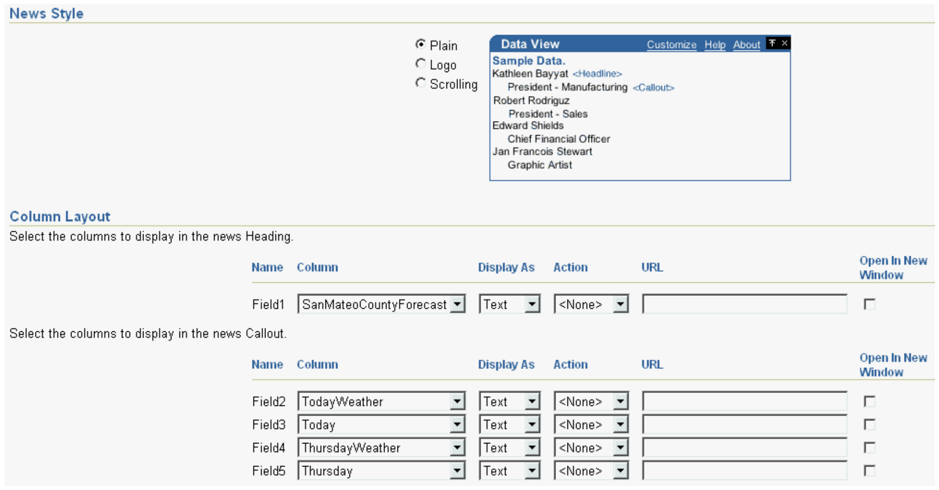
Data View — Customize Help About

Sample Data.
Kathleen Bayyat <Headline>
President - Manufacturing <Callout>
Robert Rodriguz
President - Sales
Edward Shields
Chief Financial Officer
Jan Francois Stewart
Graphic Artist

33. Click **Next**.

34. Verify that the Layout tab looks like Figure 4–33.

Figure 4–33 Layout Tab



News Style

[✓ Plain] Logo Scrolling

Data View — Customize Help About

Sample Data.
Kathleen Bayyat <Headline>
President - Manufacturing <Callout>
Robert Rodriguz
President - Sales
Edward Shields
Chief Financial Officer
Jan Francois Stewart
Graphic Artist

Column Layout

Select the columns to display in the news Heading.

Name	Column	Display As	Action	URL	Open In New Window
Field1	SanMateoCountyForecast	Text	<None>		☐

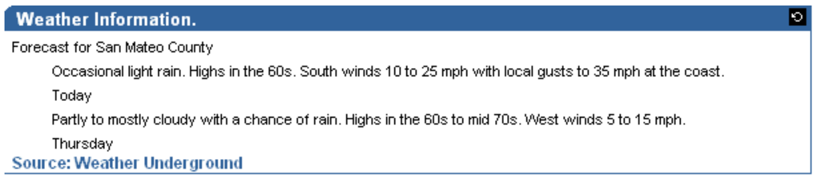
Select the columns to display in the news Callout.

Name	Column	Display As	Action	URL	Open In New Window
Field2	TodayWeather	Text	<None>		☐
Field3	Today	Text	<None>		☐
Field4	ThursdayWeather	Text	<None>		☐
Field5	Thursday	Text	<None>		☐

35. Click **Finish**.

36. Your new Web Page portlet displays on the portal page, and should look like Figure 4–34.

Figure 4–34 Weather Information Portlet on the Portal Page



Weather Information.

Forecast for San Mateo County

Occasional light rain. Highs in the 60s. South winds 10 to 25 mph with local gusts to 35 mph at the coast.

Today

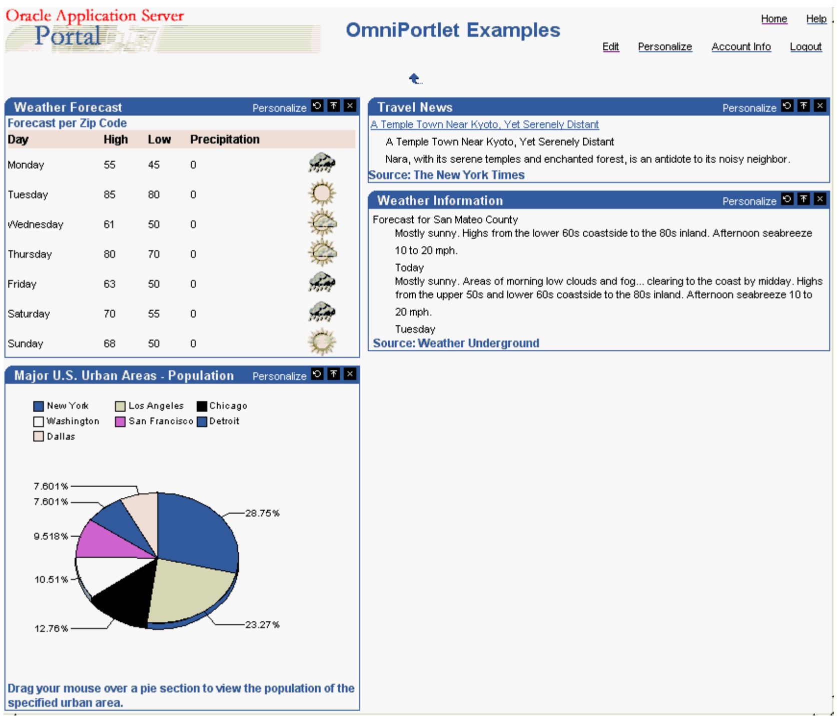
Partly to mostly cloudy with a chance of rain. Highs in the 60s to mid 70s. West winds 5 to 15 mph.

Thursday

Source: Weather Underground

Now that you have completed building all four example OmniPortlets, your page should look like Figure 4–35.

Figure 4–35 Portal Page Displaying the Four Example OmniPortlets



OmniPortlet Examples

Weather Forecast

Forecast per Zip Code

Day	High	Low	Precipitation
Monday	55	45	0
Tuesday	85	80	0
Wednesday	61	50	0
Thursday	80	70	0
Friday	63	50	0
Saturday	70	55	0
Sunday	68	50	0

Travel News

A Temple Town Near Kyoto, Yet Serenely Distant

A Temple Town Near Kyoto, Yet Serenely Distant

Nara, with its serene temples and enchanted forest, is an antidote to its noisy neighbor.

Source: The New York Times

Weather Information

Forecast for San Mateo County

Mostly sunny. Highs from the lower 60s coastside to the 80s inland. Afternoon seabreeze 10 to 20 mph.

Today

Mostly sunny. Areas of morning low clouds and fog... clearing to the coast by midday. Highs from the upper 50s and lower 60s coastside to the 80s inland. Afternoon seabreeze 10 to 20 mph.

Tuesday

Source: Weather Underground

Major U.S. Urban Areas - Population

Drag your mouse over a pie section to view the population of the specified urban area.

4.6 Setting Up Portlet Parameters and Events

The steps in this section show you how to set up the portlets and page you created to use parameters. Then, when a user clicks a slice of the pie chart (generating a portlet event) that corresponds to a region (for example, New York), the Weather Forecast per Zip Code portlet (based on a Web Service) and the Weather Information portlet (based on a Web page) will display the corresponding weather information for that region (New York).

To set up portlet parameters and events, you will need to do the following:

- Section 4.6.1, "Configure Portlets to Accept Parameters"
- Section 4.6.2, "Map the Page Parameter to the Portlet Parameters"
- Section 4.6.3, "Configure the Chart Portlet to Use Events"
- Section 4.6.4, "Map the Chart Event to the Page"

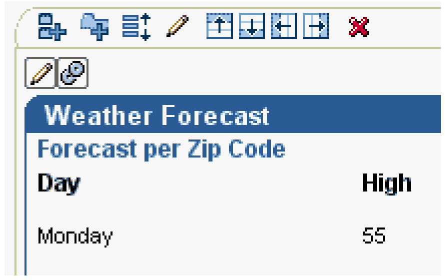
4.6.1 Configure Portlets to Accept Parameters

The steps in this section will show you how to configure the two portlets on your page that accept parameters (the Web Service portlet that displays Weather Forecast per Zip Code information and the Web page portlet that displays Weather Information).

To configure the two portlets to accept parameters, perform the following steps:

1. In Edit mode of the page, click the **Edit Defaults** icon in the top left corner of the Web Services portlet, as shown in Figure 4–36.

Figure 4–36 Edit Defaults Icon in the Web Services Portlet

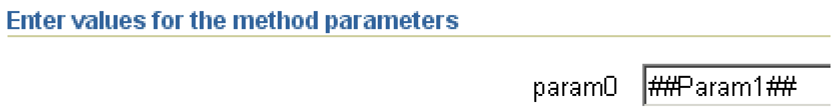

Weather Forecast

Forecast per Zip Code

Day	High
Monday	55

2. On the Source tab, replace the value of **param0** (94065) with `##Param1##`, as shown in Figure 4–37.

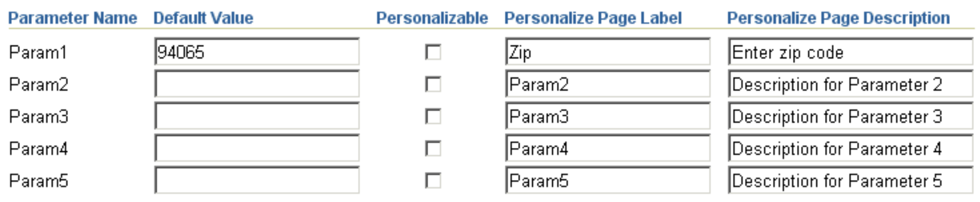
Figure 4–37 param0 Set to ##Param1##


Enter values for the method parameters

param0 ##Param1##

3. Under **Portlet Parameters**, set the default value of Param1 to `94065`.
4. In the **Personalize Page Label** field, enter `Zip`.
5. In the **Personalize Page Description** field, enter `Enter zip code`, as shown in Figure 4–38.

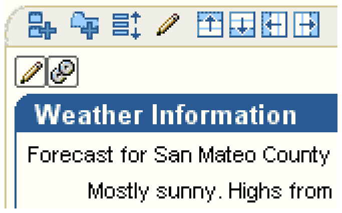
Figure 4–38 Portlet Parameters Section of the Web Services Source Tab



Parameter Name	Default Value	Personalizable	Personalize Page Label	Personalize Page Description
Param1	94065	☐	Zip	Enter zip code
Param2		☐	Param2	Description for Parameter 2
Param3		☐	Param3	Description for Parameter 3
Param4		☐	Param4	Description for Parameter 4
Param5		☐	Param5	Description for Parameter 5

6. Click **OK**.
7. On the portal page, in Edit mode, click the **Edit Defaults** icon for the Web page portlet, as shown in Figure 4–39.

Figure 4–39 Edit Defaults Icon for the Web Page Portlet

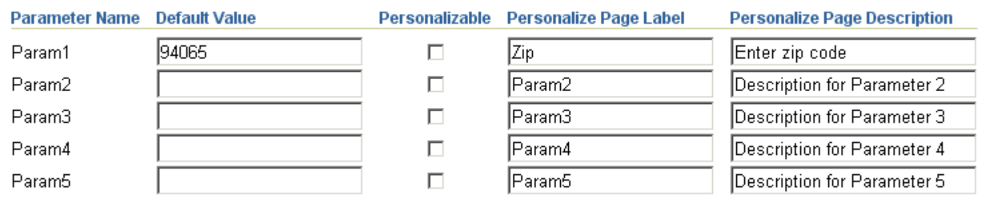

Weather Information

Forecast for San Mateo County

Mostly sunny. Highs from

8. On the Source tab, since you have already mapped a portlet parameter to the clipping parameter, you can simply add a label and description to the portlet parameter Param1, as shown in Figure 4–40.

Figure 4–40 Portlet Parameters Section of the Web Page Source Tab



Parameter Name	Default Value	Personalizable	Personalize Page Label	Personalize Page Description
Param1	94065	☐	Zip	Enter zip code
Param2		☐	Param2	Description for Parameter 2
Param3		☐	Param3	Description for Parameter 3
Param4		☐	Param4	Description for Parameter 4
Param5		☐	Param5	Description for Parameter 5

9. Click **OK**.

You have created two portlet parameters in the Web Services and Web page portlets to accept page parameters. Next, you will map a page parameter to these two portlet parameters.

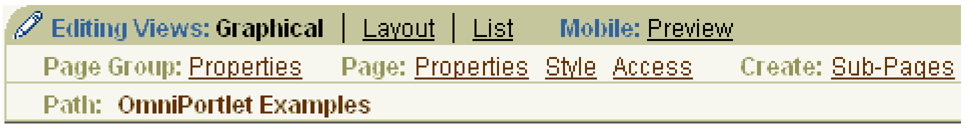
4.6.2 Map the Page Parameter to the Portlet Parameters

The steps in this section will show you how to map the page parameters to the two portlets you configured in the previous section.

To map the page parameter to the portlet parameters, perform the following steps:

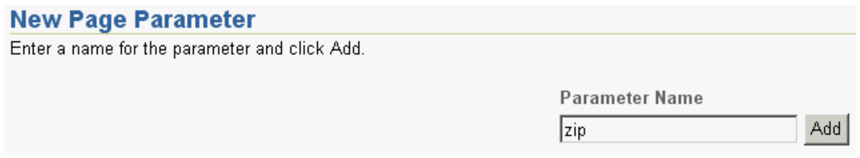
1. On the page, in Edit mode, click the Page: **Properties** link at the top of the screen, as shown in Figure 4–41.

Figure 4–41 Page: Properties Link


Editing Views: Graphical | Layout | List Mobile: Preview
Page Group: Properties Page: Properties Style Access Create: Sub-Pages
Path: OmniPortlet Examples

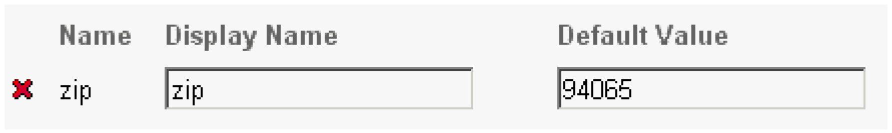
2. Click the **Parameters** tab.
3. Under New Page Parameter, in the **Parameter Name** field, enter `zip`, then click **Add**, as shown in Figure 4–42.

Figure 4–42 New Page Parameter



4. Under Page Parameter Properties, the new page parameter displays. In the **Default Value** field, enter `94065`, as shown in Figure 4–43.

Figure 4–43 Default Value for the "zip" Page Parameter



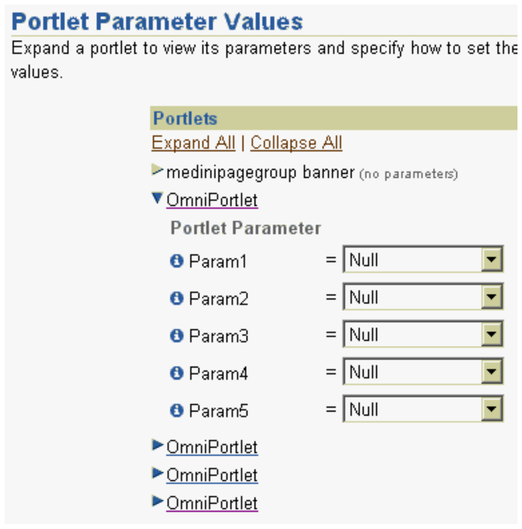
	Name	Display Name	Default Value
✖	zip	zip	94065

5. Under Portlet Parameter Values, you will see a list of portlets listed. If you have followed all the steps in this chapter, you will see four instances of OmniPortlet

listed. The Web Services portlet is the first in the list, and the Web Page data source is the fourth (or the bottom) instance of OmniPortlet in the list.

6. Click the arrow next to the first instance of OmniPortlet to expand the Portlet Parameters list, as shown in Figure 4–44.

Figure 4–44 Portlet Parameter Values Section



Portlet Parameter Values

Expand a portlet to view its parameters and specify how to set the values.

Portlets

Expand All | Collapse All

▶ medinipagegroup banner (no parameters)

▼ OmniPortlet

Portlet Parameter

Param1	=	Null
Param2	=	Null
Param3	=	Null
Param4	=	Null
Param5	=	Null

▶ OmniPortlet

▶ OmniPortlet

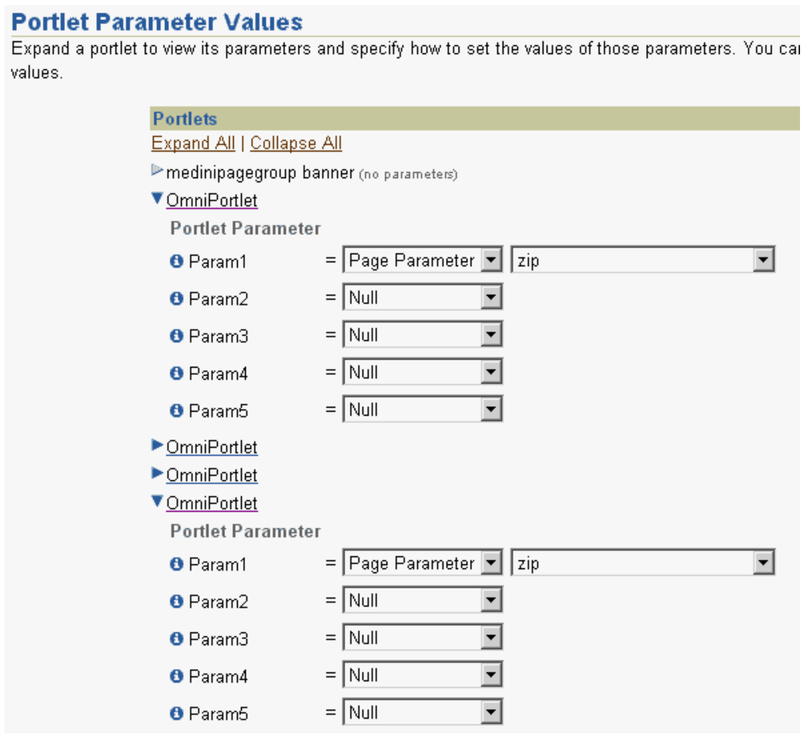
▶ OmniPortlet

7. Next to Param1, from the list, choose **Page Parameter**.

8. From the list that displays, ensure that **zip** is selected.

9. For the fourth OmniPortlet in the list, follow the same steps to set **Param1** to the Page Parameter of zip, as shown in Figure 4–45.

Figure 4–45 Portlet Parameters Section of the Page Parameters Tab



Portlet Parameter Values

Expand a portlet to view its parameters and specify how to set the values of those parameters. You ca
values.

Portlets

Expand All | Collapse All

▷ medinipagegroup banner (no parameters)

▼ OmniPortlet

Portlet Parameter			
Param1	=	Page Parameter	zip
Param2	=	Null	
Param3	=	Null	
Param4	=	Null	
Param5	=	Null	

▶ OmniPortlet

▶ OmniPortlet

▼ OmniPortlet

Portlet Parameter			
Param1	=	Page Parameter	zip
Param2	=	Null	
Param3	=	Null	
Param4	=	Null	
Param5	=	Null	

The page parameter `zip` is now mapped to the portlet parameters for the Web Services and Web page portlets. Next, you will set up the chart portlet so that when a slice of the pie chart is clicked, the selected zip code will be sent to the page.

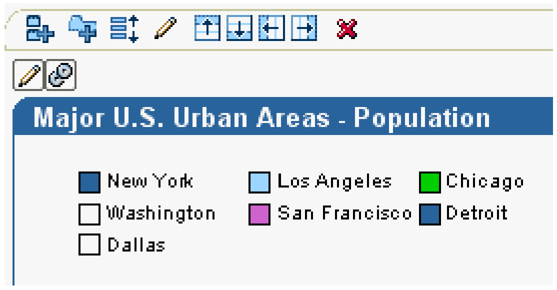
4.6.3 Configure the Chart Portlet to Use Events

The steps in this section will show you how to configure the chart portlet to use events. That is, based on an event in the portlet (such as clicking a slice in a pie chart), an event will occur. In this case, you will configure the portlet so that when a slice is clicked by an end user, the zip code associated with that slice will be sent to the page. Then, the data in the two portlets you configured in Section 4.6.1, "Configure Portlets to Accept Parameters" will refresh depending on the selected zip code.

To configure the chart portlet, perform the following steps:

1. In the Edit mode of the page, click the **Edit Defaults** icon for the chart portlet, as shown in Figure 4–46.

Figure 4–46 Edit Defaults Icon for the Chart Portlet

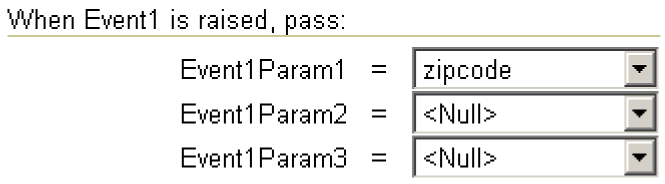


2. Click the **Layout** tab.

3. Under Chart Drilldown, choose **Event1** from the Action list.
4. Notice that in Edit Defaults mode, a new tab displays in the wizard called Events. Click the **Events** tab.
5. On this tab, you will configure Event1 (which you set on the Layout tab) to pass the zip code from the chart to the page as the event output.

   Set Event1Param1 to **zipcode**, as shown in Figure 4–47, then click **OK**.

Figure 4–47 Events Tab for the Chart Portlet



When Event1 is raised, pass:

Event1Param1	=	zipcode
Event1Param2	=	<Null>
Event1Param3	=	<Null>

You have configured the chart portlet so that a user can click a slice of the pie chart, and set up an event so that the zip code selected in the pie chart will be sent to the page. Next, you will set up the page to accept this event parameter.

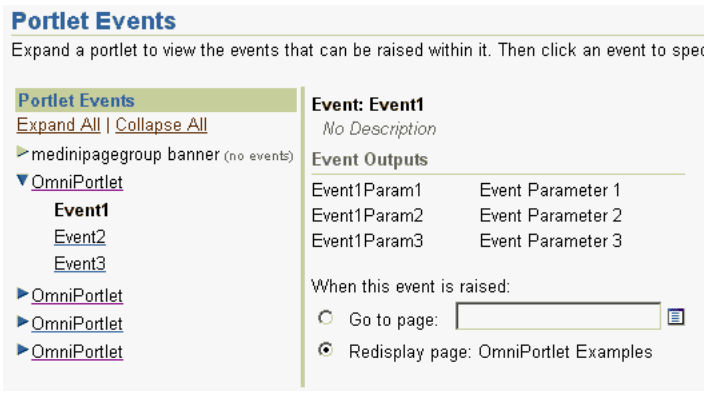
4.6.4 Map the Chart Event to the Page

The steps in this section will show you how to map the chart event you created in the previous section to the page, so that when a user clicks on a slice of the pie chart, the zip code selected in the pie chart will be accepted by the page as the page input. The page will then display the data that corresponds to the selected zip code in the Web Service and Web page portlets.

To map the chart event to the page, perform the following steps:

1. On the page, in Edit mode, click the Page: **Properties** link, then click the **Events** tab.
2. Expand the second OmniPortlet in the list to display the events below it, and select **Event1**, as shown in Figure 4–48.

Figure 4–48 Portlet Events Section of the Page Events Tab



Portlet Events

Expand a portlet to view the events that can be raised within it. Then click an event to spec

Portlet Events

Expand All | Collapse All

- medinipagegroup banner (no events)
- OmniPortlet
  - Event1
  - Event2
  - Event3
- OmniPortlet
- OmniPortlet
- OmniPortlet

Event: Event1

No Description

Event Outputs

Event1Param1	Event Parameter 1
Event1Param2	Event Parameter 2
Event1Param3	Event Parameter 3

When this event is raised:

Go to page:

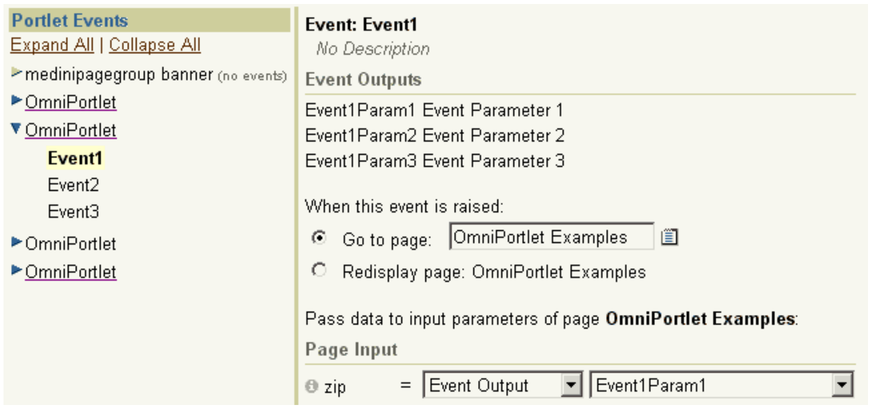
[✓ Redisplay page: OmniPortlet Examples]

3. Select the **Go to page** radio button, then, next to the field, click the **Browse Pages** icon to search for the name of your page (in this case, `OmniPortlet Examples`). Next to your page name, click **Return Object**.

Note: If you do not know the name of the page, you can return to the Edit mode of the page by clicking Cancel. Then, click the Page Group: **Properties** link to view the display name of the page group. When you return to the Events tab of the Page Properties, you can click the **Browse Pages** icon to search for the page group, under which you should see the page name.

---

4. Set the **Page Input** as shown in Figure 4–49.



Portlet Events
Expand All | Collapse All
- medinipagegroup banner (no events)
- OmniPortlet
- OmniPortlet
    - Event1
    - Event2
    - Event3
- OmniPortlet
- OmniPortlet

Event: Event1
No Description

Event Outputs

Event1Param1	Event Parameter 1
Event1Param2	Event Parameter 2
Event1Param3	Event Parameter 3

When this event is raised:
[✓ Go to page: OmniPortlet Examples]
Redisplay page: OmniPortlet Examples

Pass data to input parameters of page **OmniPortlet Examples**:

Page Input

zip = Event Output Event1Param1

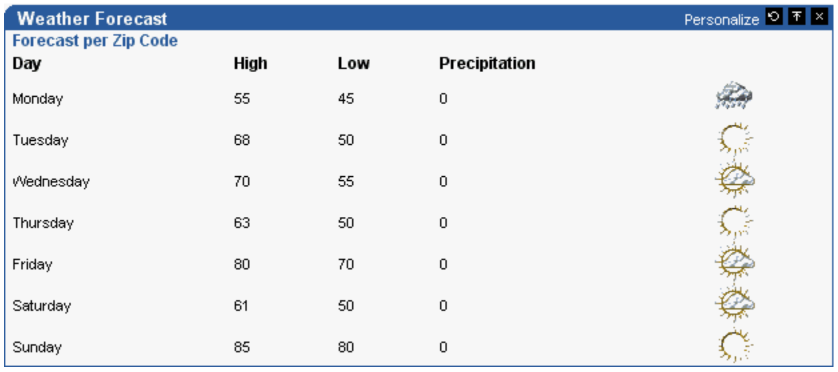
Figure 4–49 Page Input on the Events Tab

5. Click **OK**.

6. Now, when you drag your mouse over the pie chart in the Chart portlet, you should notice that you can click one of the sections. Try clicking on the largest slice (New York). You will notice that the page refreshes.

   In the URL of the page, you should see a parameter value set after the page name, for example: `OmniPortlet%20Examples?zip=10001`. The Weather Forecast information changes, and looks something like Figure 4–50.

Figure 4–50 Weather Forecast (Web Service) Portlet for New York

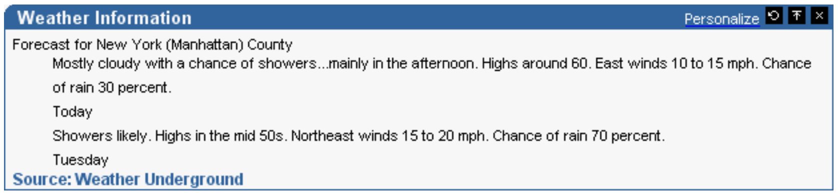


Forecast per Zip Code

Day	High	Low	Precipitation
Monday	55	45	0
Tuesday	68	50	0
Wednesday	70	55	0
Thursday	63	50	0
Friday	80	70	0
Saturday	61	50	0
Sunday	85	80	0

The Weather Information portlet changes, and looks something like Figure 4–51.

Figure 4–51 Weather Information (Web Page) Portlet for New York



Weather Information

Forecast for New York (Manhattan) County

Mostly cloudy with a chance of showers...mainly in the afternoon. Highs around 60. East winds 10 to 15 mph. Chance of rain 30 percent.

Today

Showers likely. Highs in the mid 50s. Northeast winds 15 to 20 mph. Chance of rain 70 percent.

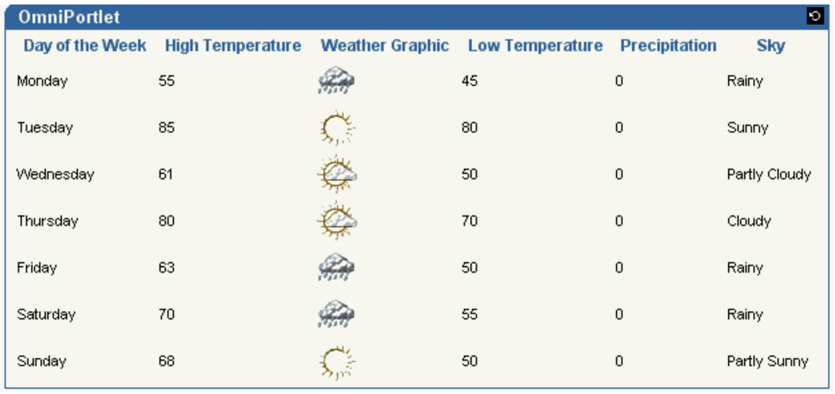
Tuesday

Source: Weather Underground

4.7 Building an OmniPortlet Using the HTML Layout

As you can see from the examples in this chapter, you can use any data source with any of the built-in layouts in OmniPortlet. This section will show you how to use the same Web Service data source you used in Section 4.2, "Building an OmniPortlet Based on a Web Service" to create a portlet using the HTML layout in OmniPortlet. At the end of this section, your portlet will look like Figure 4–52.

Figure 4–52 OmniPortlet Using the HTML Layout



OmniPortlet

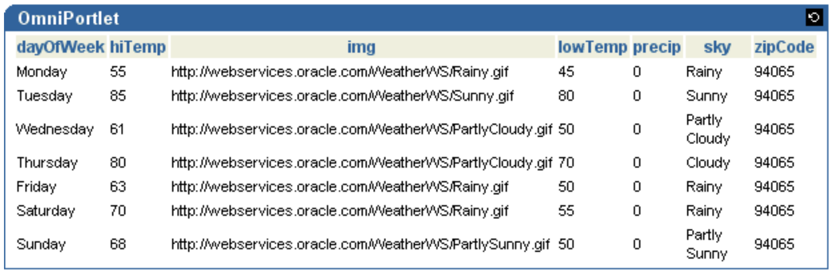
Day of the Week	High Temperature	Weather Graphic	Low Temperature	Precipitation	Sky
Monday	55		45	0	Rainy
Tuesday	85		80	0	Sunny
Wednesday	61		50	0	Partly Cloudy
Thursday	80		70	0	Cloudy
Friday	63		50	0	Rainy
Saturday	70		55	0	Rainy
Sunday	68		50	0	Partly Sunny

To create a portlet using the HTML layout, perform the following steps:

1. Follow steps 1-8 in Section 4.2, "Building an OmniPortlet Based on a Web Service" to use the Web Service data source.
2. On the View tab, under Layout Style, select **HTML**, then click **Next**.
3. On the Layout tab, the default layout is a Simple Table.

   You can switch the layout using the Quick Start drop-down list, or clear the fields to create your own layout. For this example, use the default Simple Table layout. In the Repeating Section, you can see how OmniPortlet automatically populates the columns from the data source into your layout. You can modify the HTML in any of the sections as you wish.
4. Click **Finish**. Your portlet looks like Figure 4–53.

Figure 4–53 Initial View of an OmniPortlet Using the HTML Layout



dayOfWeek	hiTemp	img	lowTemp	precip	sky	zipCode
Monday	55	http://webservices.oracle.com/WeatherWS/Rainy.gif	45	0	Rainy	94065
Tuesday	85	http://webservices.oracle.com/WeatherWS/Sunny.gif	80	0	Sunny	94065
Wednesday	61	http://webservices.oracle.com/WeatherWS/PartlyCloudy.gif	50	0	Partly Cloudy	94065
Thursday	80	http://webservices.oracle.com/WeatherWS/PartlyCloudy.gif	70	0	Cloudy	94065
Friday	63	http://webservices.oracle.com/WeatherWS/Rainy.gif	50	0	Rainy	94065
Saturday	70	http://webservices.oracle.com/WeatherWS/Rainy.gif	55	0	Rainy	94065
Sunday	68	http://webservices.oracle.com/WeatherWS/PartlySunny.gif	50	0	Partly Sunny	94065

5. You'll notice that under "img," the URL displays instead of an actual image. To change the HTML source, click the Edit Defaults (pencil) icon to edit your OmniPortlet.
6. Click the **Layout** tab.
7. In the Repeating Section, look for the row with the code: `<TD>##img##</TD>`.
8. Replace this code with: <TD><img src="##img##"></TD>.
9. You can also change the column headers in the Non-Repeating Heading Section. For example, replace:

   ```
   <TH CLASS='PortletHeading1'>img</TH>"
   ```

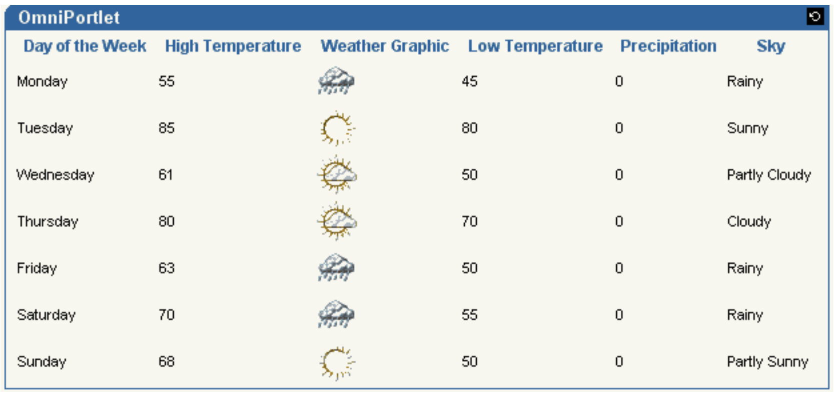
   with

   ```
   <TH CLASS='PortletHeading1'>Weather Graphic</TH>
   ```
10. In the Non-Repeating Heading section, delete the line:

    ```
      <TH CLASS='PortletHeading1'>Zip Code</TH>
    ```
11. In the Repeating Section, delete the line:

    ```
     <TD>##zipCode##</TD>
    ```
12. Perform the same action for any other headings you wish to change, then click **Finish**. Your portlet now looks like Figure 4–54.

Figure 4–54 Final View of an OmniPortlet Using the HTML Layout



Day of the Week	High Temperature	Weather Graphic	Low Temperature	Precipitation	Sky
Monday	55		45	0	Rainy
Tuesday	85		80	0	Sunny
Wednesday	61		50	0	Partly Cloudy
Thursday	80		70	0	Cloudy
Friday	63		50	0	Rainy
Saturday	70		55	0	Rainy
Sunday	68		50	0	Partly Sunny

5 Creating Content-Based Portlets with Web Clipping

Web Clipping is a browser-based declarative tool that enables you to integrate any Web application with Oracle Portal. It is designed to give you quick integration by leveraging the Web application's existing user interface. Web Clipping has been implemented as a Web provider using the Oracle Portal Developer Kit, which is a component of Oracle Portal.

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.

This chapter contains the following sections:

Note: Web Clipping is provided with Oracle Portal 11*g* Release 1 (11.1.1) and is also available as a download with the Oracle Portal Developer Kit. In Oracle Portal, you can add a Web Clipping portlet from the Oracle Portal Repository in the Portlet Builders folder. If you've downloaded Web Clipping as part of the Oracle Portal Developer Kit, you must register it before you can use it. To learn more about registering a Web provider, see *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*. You can then add an Web Clipping portlet to any portal page.

5.1 What Is Web Clipping?

Web Clipping allows 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 portal as a portlet. The Web Clipping portlet 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, 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 Oracle Portal 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 Application Server 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 per-user authentication. You can specify proxy server authentication details including type (Basic or Digest) and realm, through the Web Clipping Provider Test page. 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 Portal) automatically log in using a user name and password you provide, while valid users (authenticated into Portal) will need to log in using a user name and password they provide.

  See the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* for more information.
- **Migration from URL-based portlets,** enabling you to migrate your URL-based portlets to Web Clipping. See Section 5.6, "Migrating from URL-Based Portlets" for more information.
- **Navigation and clipping of HTTPS-based external Web sites,** if appropriate server certificates are acquired.
- **Open Transport API for customizing authentication mechanisms to clipped sites**. By default, Web Clipping provider supports only HTTP challenge-based authentication methods like Basic and Digest, and form submission logins. To support custom authentication methods, like Kerberos proxy authentication, users can use the Web Clipping Transport API. See Section 5.5, "Using Web Clipping Open Transport API" for more information.
- **Clipping of page content from HTML 4.0.1 pages,** including the following:
  - Clipping of <applet>, <body>, <div>, <embed>, <img>, <object>, <ol>, <span>, <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

- **National Language Sets (NLS)** in URLs and URL parameters. See Section 5.7, "Current Limitations for Web Clipping" for information about how Web Clipping determines the character set of clipped content.

By default, all Web clipping definitions are stored persistently in the Oracle Fusion Middleware infrastructure database. Any secure information, such as passwords, are stored in encrypted form, according to the Data Encryption Standard (DES), using Oracle encryption technology.

5.2 Adding Web Page Content to a Portal Page

To add Web page content to a portal page, follow the steps described in the following sections:

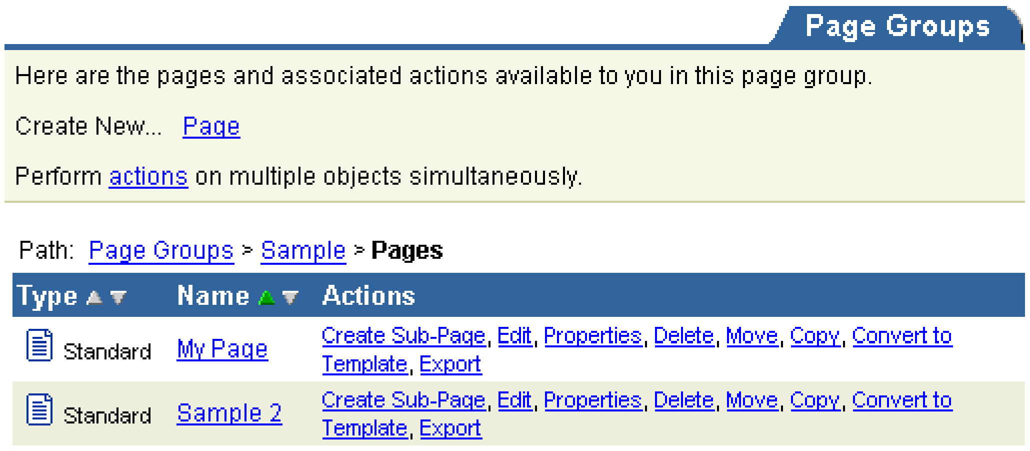
- Section 5.2.1, "Adding a Web Clipping Portlet to a Page"
- Section 5.2.2, "Selecting a Section of a Web Page to Display in the Web Clipping Portlet"
- Section 5.2.3, "Setting Web Clipping Portlet Properties"

5.2.1 Adding a Web Clipping Portlet to a Page

To add a Web Clipping portlet to an Oracle Portal page, perform the following steps:

1. Navigate to the Page Groups portlet. By default, the Page Groups portlet is located on the **Build** tab of the Portal Builder page.
2. In the **Layout & Appearance** section, for **Pages**, click **Browse.**
3. Figure 5–1 shows the Page Groups tab with the list of pages. For the page to which you want to add the Web Clipping portlet, click **Edit** in the **Actions** column.

Figure 5–1 Selecting a Page



Here are the pages and associated actions available to you in this page group.

Create New... Page

Perform actions on multiple objects simultaneously.

Path: Page Groups > Sample > **Pages**

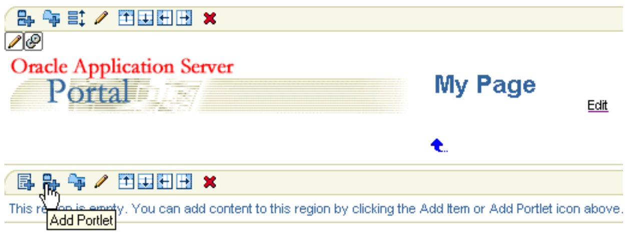
Type	Name	Actions
Standard	My Page	Create Sub-Page, Edit, Properties, Delete, Move, Copy, Convert to Template, Export
Standard	Sample 2	Create Sub-Page, Edit, Properties, Delete, Move, Copy, Convert to Template, Export

The page you want to edit is displayed.

4. In the region in which you want to add the Web Clipping portlet, click the **Add Portlet** icon.

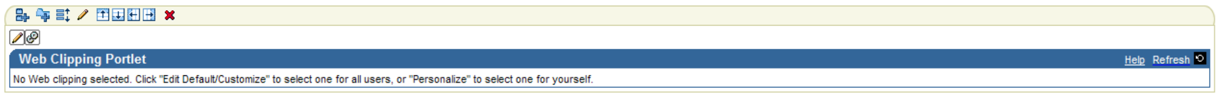
   Figure 5–2 shows a portion of the page.

Figure 5–2 Adding a Portlet to a Page


Oracle Application Server
Portal
My Page
Edit
This region is empty. You can add content to this region by clicking the Add Item or Add Portlet icon above.
Add Portlet

5. On the **Add Portlets** page, in the **Available Portlets** list, click the **Portlet Builders** link.
6. Click the **Web Clipping Portlet** link.
7. Click **OK**. The new instance of the Web Clipping portlet now displays on your portal page, as shown in Figure 5–3.

Figure 5–3 Web Clipping Portlet Added to a Page



No Web clipping selected. Click "Edit Default/Customize" to select one for all users, or "Personalize" to select one for yourself.

5.2.2 Selecting a Section of a Web Page to Display in the Web Clipping Portlet

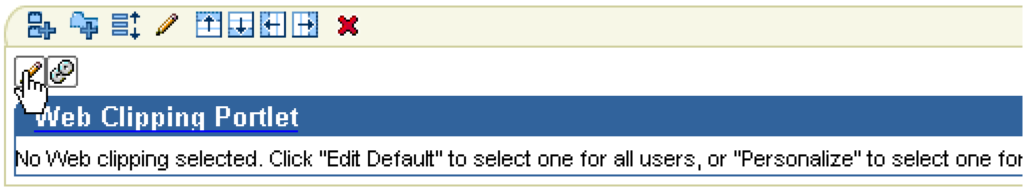
To select a section of a Web page to display in the Web Clipping portlet, you use the Web Clipping Studio. Using the Web Clipping Studio, you can do the following:

- 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. Above the Web Clipping portlet, click the **Edit Defaults** icon, as shown in Figure 5–4.

Figure 5–4 Editing Default Settings

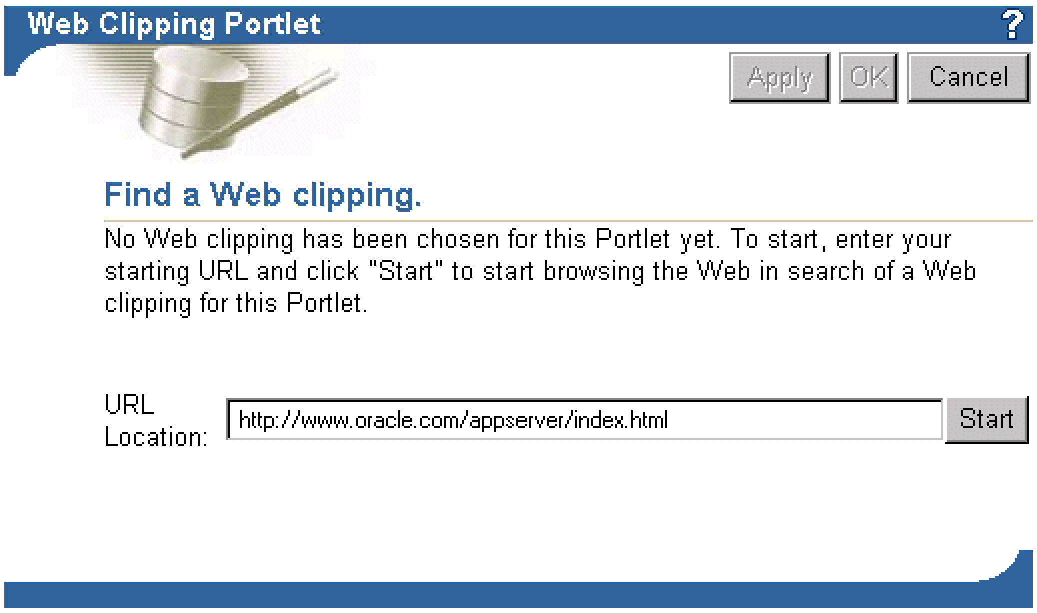


No Web clipping selected. Click "Edit Default" to select one for all users, or "Personalize" to select one for

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 5–5.

Figure 5–5 Specifying a URL



Find a Web clipping.

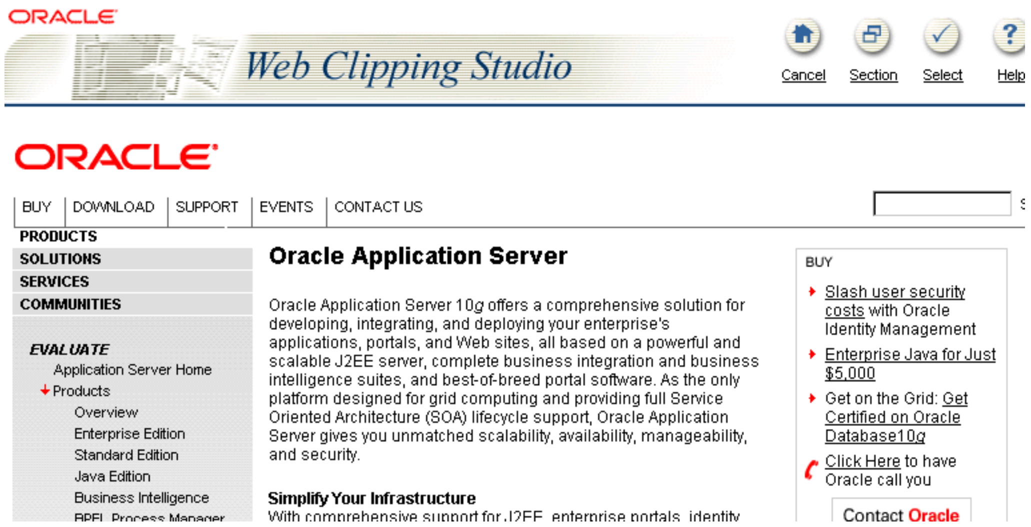
No Web clipping has been chosen for this Portlet yet. To start, enter your starting URL and click "Start" to start browsing the Web in search of a Web clipping for this Portlet.

URL Location: http://www.oracle.com/appserver/index.html

3. Click **Start**.

The Web Clipping Studio displays the page you specified, as shown in Figure 5–6.

Figure 5–6 Browsing to a Page Containing Content for a Web Clipping



Oracle Application Server

Oracle Application Server 10*g* offers a comprehensive solution for developing, integrating, and deploying your enterprise's applications, portals, and Web sites, all based on a powerful and scalable J2EE server, complete business integration and business intelligence suites, and best-of-breed portal software. As the only platform designed for grid computing and providing full Service Oriented Architecture (SOA) lifecycle support, Oracle Application Server gives you unmatched scalability, availability, manageability, and security.

Simplify Your Infrastructure

Slash user security costs with Oracle Identity Management

Enterprise Java for Just $5,000

Get on the Grid: Get Certified on Oracle Database10*g*

Click Here to have Oracle call you

Note that the URL in the browser bar changes from:

```
http://host:port/portal/page?_dad=portal&_schema=PORTAL...
```

To:

```
http://host:port/portalTools/webClipping...
```

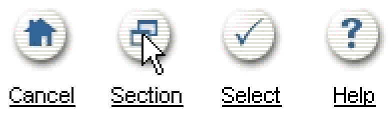
4. Browse to the page that contains the content you want to clip.

   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.

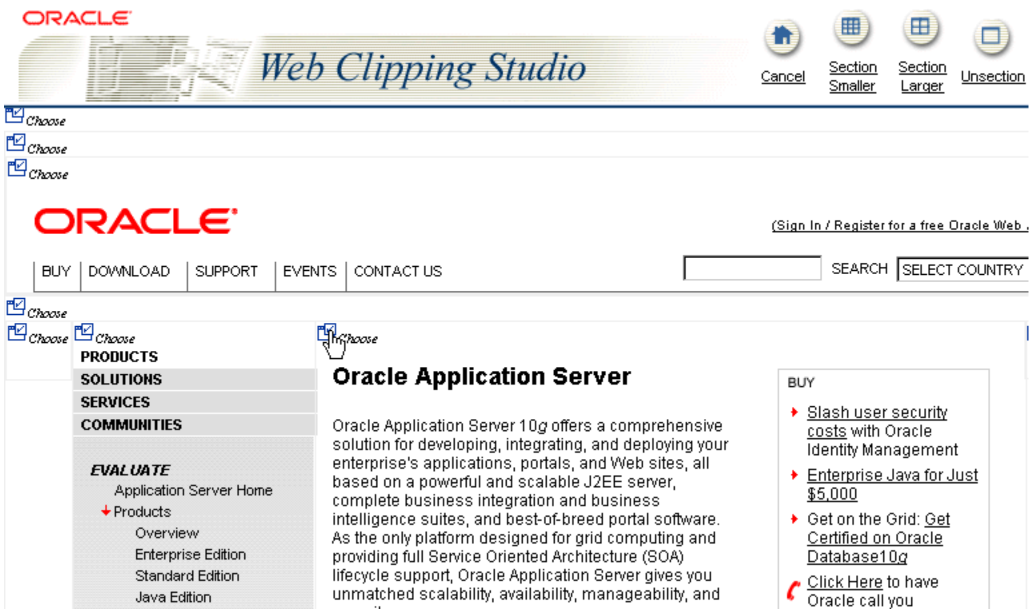
5. Once you display the page that contains the content you want to clip, in the Web Clipping Studio banner, click **Section,** as shown in Figure 5–7.

Figure 5–7 Sectioning the Target Web Page



Sectioning divides the target Web page into its clippable sections, as shown in Figure 5–8. After you click **Section**, you are no longer able to browse links in the displayed page. If you want to continue navigation, click **Unsection** in the Web Clipping Studio banner.

Figure 5–8 Sectioned Target Web Page



Oracle Application Server

Oracle Application Server 10*g* offers a comprehensive solution for developing, integrating, and deploying your enterprise's applications, portals, and Web sites, all based on a powerful and scalable J2EE server, complete business integration and business intelligence suites, and best-of-breed portal software. As the only platform designed for grid computing and providing full Service Oriented Architecture (SOA) lifecycle support, Oracle Application Server gives you unmatched scalability, availability, manageability, and

- Slash user security costs with Oracle Identity Management
- Enterprise Java for Just $5,000
- Get on the Grid: Get Certified on Oracle Database10*g*
- Click Here to have Oracle call you

6. At the top left 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, 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.

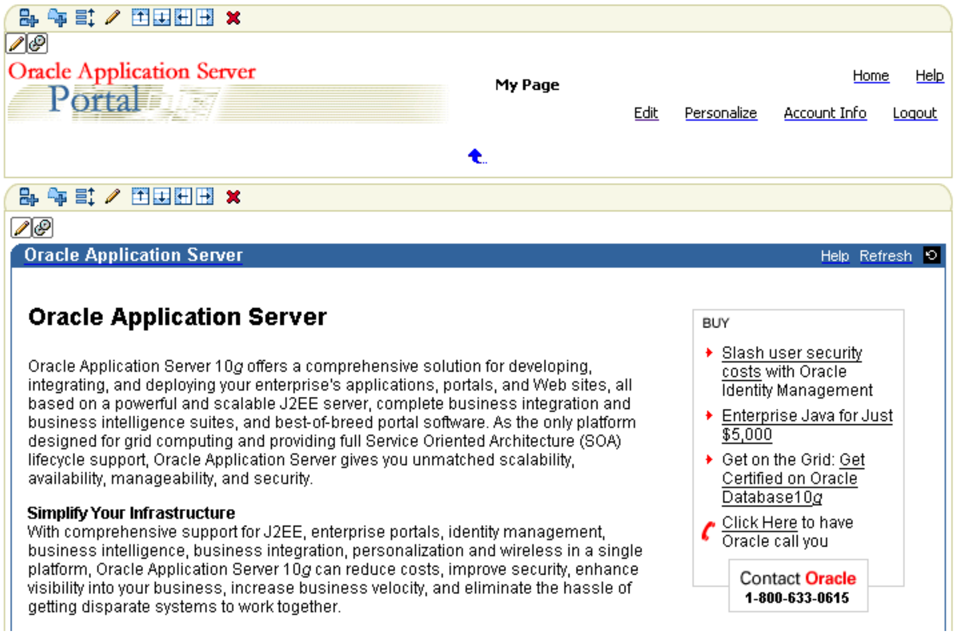
   If you do not want to use the section you clipped in your portlet, click **Unselect** to return to the page containing the section. You can choose another section on the page, or click **Unsection** to navigate to another page.

   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, 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 5.2.3, "Setting Web Clipping Portlet Properties" for more information.)

   Figure 5–9 shows the content added to the Web Clipping portlet.

Figure 5–9 Clipped Content Added to the Web Clipping Portlet on a Portal Page



Oracle Application Server

Oracle Application Server 10*g* offers a comprehensive solution for developing, integrating, and deploying your enterprise's applications, portals, and Web sites, all based on a powerful and scalable J2EE server, complete business integration and business intelligence suites, and best-of-breed portal software. As the only platform designed for grid computing and providing full Service Oriented Architecture (SOA) lifecycle support, Oracle Application Server gives you unmatched scalability, availability, manageability, and security.

Simplify Your Infrastructure
With comprehensive support for J2EE, enterprise portals, identity management, business intelligence, business integration, personalization and wireless in a single platform, Oracle Application Server 10*g* can reduce costs, improve security, enhance visibility into your business, increase business velocity, and eliminate the hassle of getting disparate systems to work together.

BUY
- Slash user security costs with Oracle Identity Management
- Enterprise Java for Just $5,000
- Get on the Grid: Get Certified on Oracle Database10*g*
- Click Here to have Oracle call you

Contact Oracle
1-800-633-0615

Note that the **Refresh** link in the Web Clipping Portlet retrieves fresh data from the originating Web site. The **Portlet Refresh** link reloads the portlet, but it may retrieve the data from the cache, depending upon the settings for expiration.

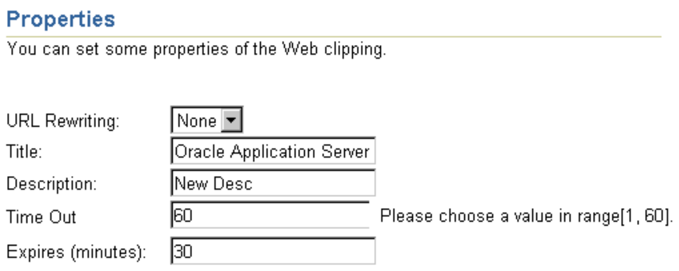
5.2.3 Setting Web Clipping Portlet Properties

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. Choose the section you want, click **Select** in the Web Clipping Studio banner. Web Clipping Studio displays the Find a Web Clipping page with a Properties section, as shown in Figure 5–10.

Figure 5–10 Properties Section of Find a Web Clipping Page



Properties

You can set some properties of the Web clipping.

URL Rewriting:	None	
Title:	Oracle Application Server	
Description:	New Desc	
Time Out	60	Please choose a value in range[1, 60].
Expires (minutes):	30	

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 Portal page in the browser.
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 provider 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, 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 allow end users to provide their own values for the parameters when they personalize the portlet. Select **None** if you do not want to allow 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 5.4.3, "Personalizing a Web Clipping Portlet" provides an example of personalizing parameters.
8. Click **OK**.

5.3 Integrating Authenticated Web Content Using Single Sign-On

This section walks you through an example that demonstrates how you can leverage OracleAS 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. Set up the external application in Oracle Portal, specifying the authentication information by performing the following steps. Refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* for more details.

   a. Log in to Oracle Portal as a user who has SSO Administration privileges, for example, the orcladmin user.

   b. Navigate to the Administer External Applications portlet. (Click the **Administer** tab, then click the **Portal** subtab. In the **SSO Server Administration** section, which is in the right column, select **Administer External Applications.**)

   c. Click **Add External Application.**

   d. In the Create External Application page, in the **Application Name** field, enter a name for the application, for example, `Metalink`.

   e. 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.

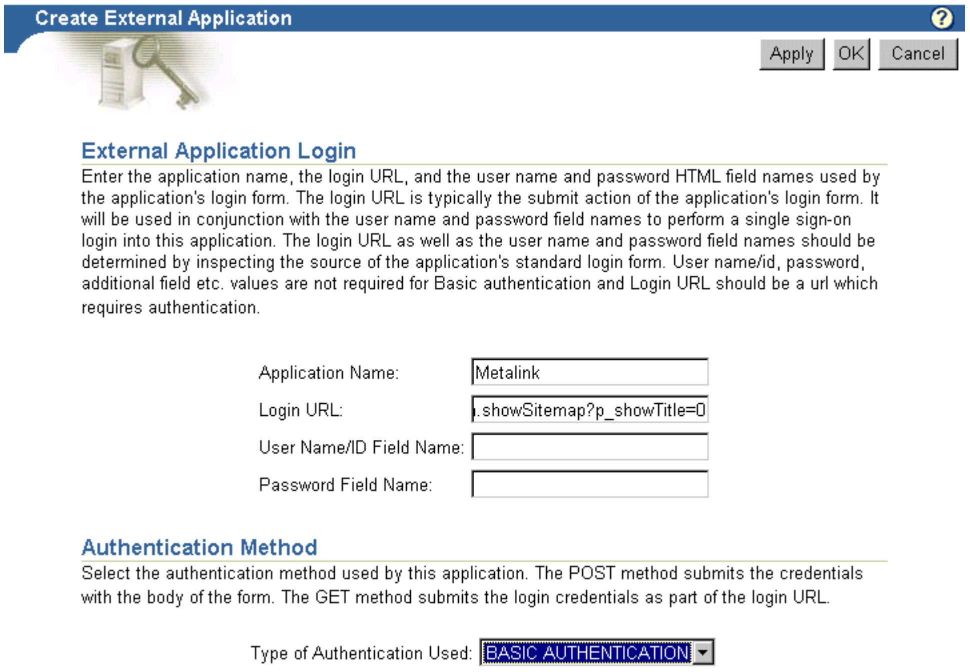
   f. For **User Name/ID Field Name,** enter the field name that the external application uses for the user name. Determine the field name by viewing the source for the desired page. If the Authentication method uses Basic Authentication, you do not need to enter a field name. For Metalink, you do not need to enter anything in this field.

   g. For **Password Field Name,** 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, you do not need to enter a field name. For Metalink, you do not need to enter anything in this field.

   h. Select **Basic Authentication** as the authentication method.

      Figure 5–11 shows a portion of the Create External Application page:

Figure 5–11 Creating an External Application



External Application Login

Enter the application name, the login URL, and the user name and password HTML field names used by the application's login form. The login URL is typically the submit action of the application's login form. It will be used in conjunction with the user name and password field names to perform a single sign-on login into this application. The login URL as well as the user name and password field names should be determined by inspecting the source of the application's standard login form. User name/id, password, additional field etc. values are not required for Basic authentication and Login URL should be a url which requires authentication.

Application Name:	Metalink
Login URL:	.showSitemap?p_showTitle=0
User Name/ID Field Name:	
Password Field Name:	

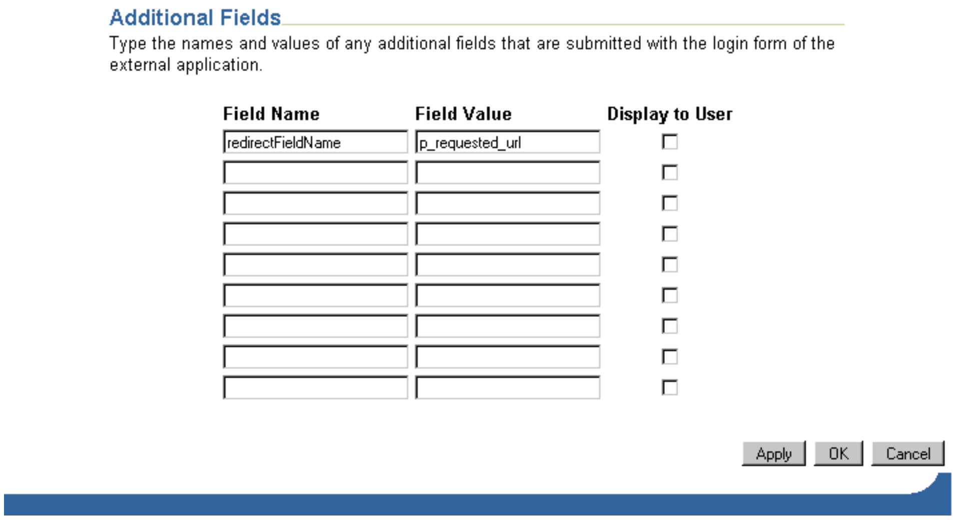
Authentication Method

Select the authentication method used by this application. The POST method submits the credentials with the body of the form. The GET method submits the login credentials as part of the login URL.

Type of Authentication Used: BASIC AUTHENTICATION

i. In the **Additional Fields** section, 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 5–12 shows the Additional Fields section.

Figure 5–12 Specifying Redirection



Additional Fields

Type the names and values of any additional fields that are submitted with the login form of the external application.

Field Name	Field Value	Display to User
redirectFieldName	p_requested_url	☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐

j. Click **OK.**

k. To test your credentials with Oracle Metalink, in the Administer External Applications page, click the name of the application you just created. Then, in the External Application Login page, log in to the application Oracle Metalink using your Oracle Metalink user name and password.

In the Administer External Applications page, click **Close.**

For more information about OracleAS Single Sign-On Server and external applications, see the *Oracle Application Server Single Sign-On Administrator's Guide*.

2. For the Web Clipping portlet, create a new Web Clipping provider by performing the following steps:

   a. Click the **Administer** tab, then click the **Portlets** tab.

   b. Select **Register a Provider.**

   c. In the Register a Provider page, enter `webClippingMetalink` for the **Name** and `webClipping Metalink` for the **Display Name.** Enter values for Timeout and Timeout Message. Choose **Web** for the **Implementation Style.**

   d. Click **Next.**

   e. In the **General Properties** section of the Define Connection page, for the **URL**, enter:

   ```
   http://host:port/portalTools/webClipping/providers/webClipping
   ```

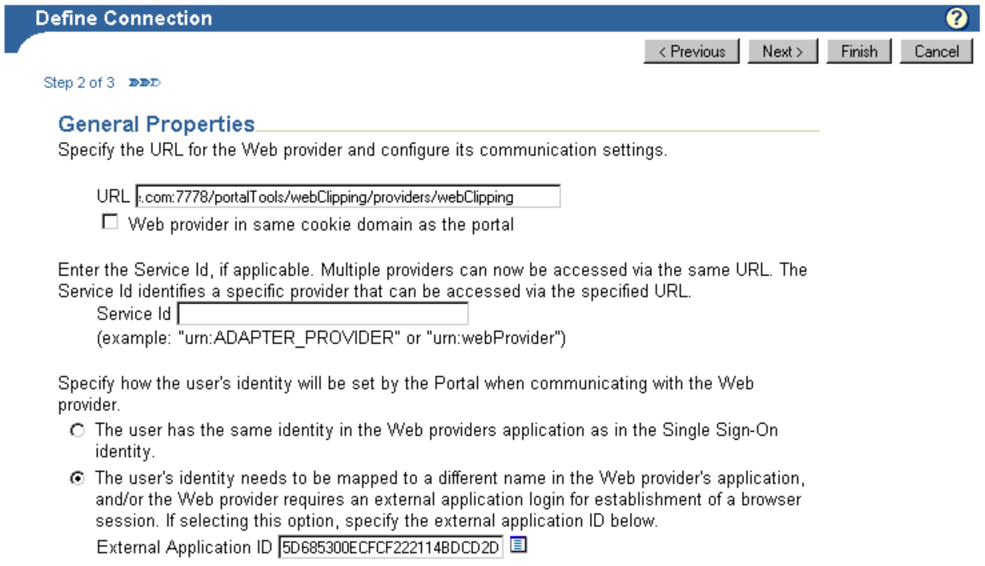
   Note that *`host:port`* refers to the host and port where the providers are located. This corresponds to the URL for Oracle Portal.

   f. For the user's identity, select **The user's identity needs to be mapped to a different name in the Web provider's application, and/or the Web provider requires an external application login for establishment of a browser session. If selecting this option, specify the external application ID below.**

   g. For **External Application ID**, click the **List of Values** icon and select the external application you added.

   Figure 5–13 shows the top part of the Define Connections page.

Figure 5–13 Specifying an External Application for a Web Clipping Provider



General Properties

Specify the URL for the Web provider and configure its communication settings.

URL :.com:7778/portalTools/webClipping/providers/webClipping

☐ Web provider in same cookie domain as the portal

Enter the Service Id, if applicable. Multiple providers can now be accessed via the same URL. The Service Id identifies a specific provider that can be accessed via the specified URL.

Service Id

(example: "urn:ADAPTER_PROVIDER" or "urn:webProvider")

Specify how the user's identity will be set by the Portal when communicating with the Web provider.

○ The user has the same identity in the Web providers application as in the Single Sign-On identity.

● The user's identity needs to be mapped to a different name in the Web provider's application, and/or the Web provider requires an external application login for establishment of a browser session. If selecting this option, specify the external application ID below.

External Application ID 5D685300ECFCF222114BDCD2D

   h. In the **User/Session Information** section, select **User** to send user specific information to the provider. For **Login Frequency,** select **Once Per User Session.**

i. Check that the proxy settings are correct. If you use a proxy server to contact the Web providers from the middle tier, enter the proxy server for **Middle Tier.**

   If you use a proxy server to contact the Web providers from the portal repository, enter the proxy server for **Portal Repository.**

   Usually, because the Oracle Portal and Web Clipping URLs point to the same middle tier, this step is not necessary.

j. Click **Finish.**

k. In the Registration Confirmation page, if the registration was successful, click **OK.**

3. Add a portlet to a page, using the Web Clipping Metalink provider that you just created, by performing the following steps:

   a. In Oracle Portal, navigate to the page in which you want to add the portlet.

   b. In the region in which you want to add the Web Clipping portlet, click the **Add Portlet** icon.

   c. In the Add Portlets to Region page, search for **Web Clipping Metalink.** It is located in the Portlet Staging Area. Click it to move it to the **Selected Portlets** box.

   d. Click **OK.**

   Section 5.2.1, "Adding a Web Clipping Portlet to a Page" describes in detail how to add a portlet.

4. If you have not entered your credentials for the External Application representing Metalink, the portlet will contain an **Update login information** link. Click the link and enter your credentials. Then, click **OK.**

5. Select a section of a page to display in the Web Clipping portlet, by performing the following steps:

   a. In the Web Clipping portlet, click the **Edit Defaults** icon.

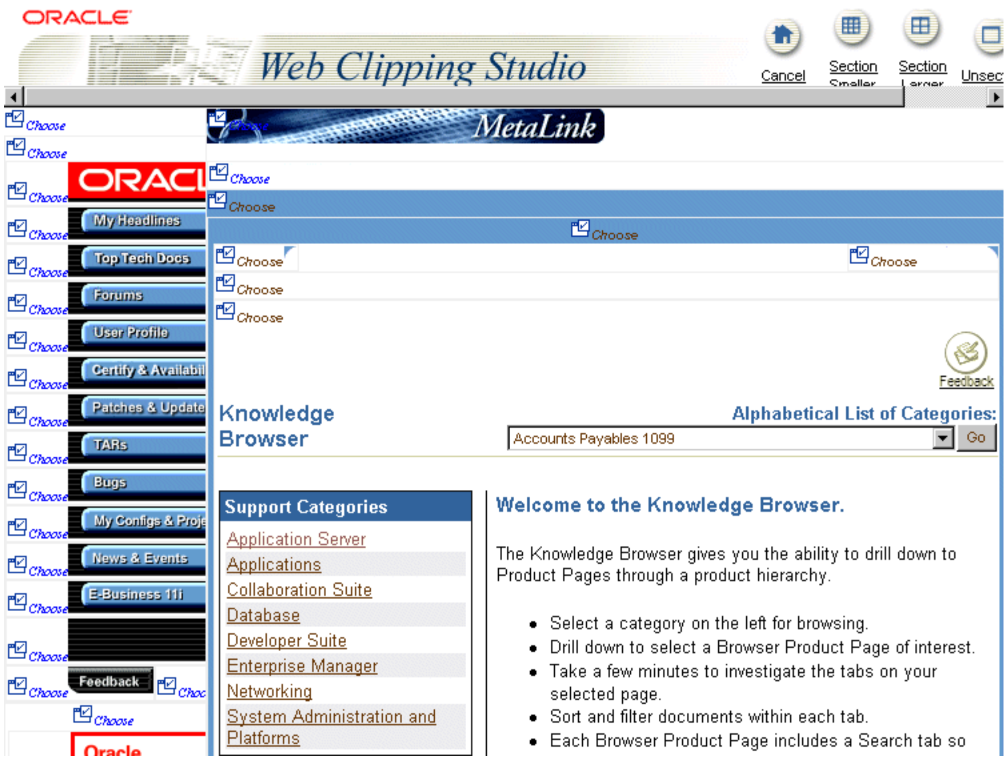
      The Find a Web Clipping page is displayed.

   b. In the **URL Location** field, the default URL for the External Application is displayed. Change it to the location of the starting Web page that links to the content you want to clip. In this case, enter `http://www.metalink.oracle.com`.

   c. Click **Start.** The Web Clipping Studio displays the page you specified. Log in to Oracle Metalink.

   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 5–14 shows the external application displayed in Web Clipping Studio.

Figure 5–14 External Application in Web Clipping Studio



Knowledge Browser

Alphabetical List of Categories: Accounts Payables 1099 Go

Support Categories
Application Server
Applications
Collaboration Suite
Database
Developer Suite
Enterprise Manager
Networking
System Administration and Platforms

Welcome to the Knowledge Browser.

The Knowledge Browser gives you the ability to drill down to Product Pages through a product hierarchy.

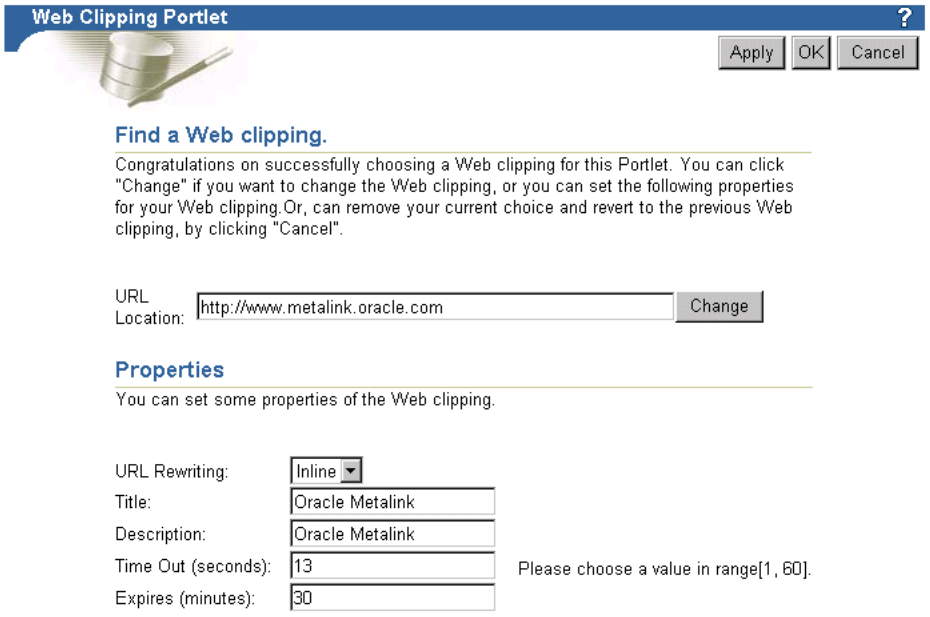
- Select a category on the left for browsing.
- Drill down to select a Browser Product Page of interest.
- Take a few minutes to investigate the tabs on your selected page.
- Sort and filter documents within each tab.
- Each Browser Product Page includes a Search tab so

e. At the top left 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, 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 5–15.

Figure 5–15 Properties of the External Application



Find a Web clipping.

Congratulations on successfully choosing a Web clipping for this Portlet. You can click "Change" if you want to change the Web clipping, or you can set the following properties for your Web clipping.Or, can remove your current choice and revert to the previous Web clipping, by clicking "Cancel".

URL Location: http://www.metalink.oracle.com

Properties

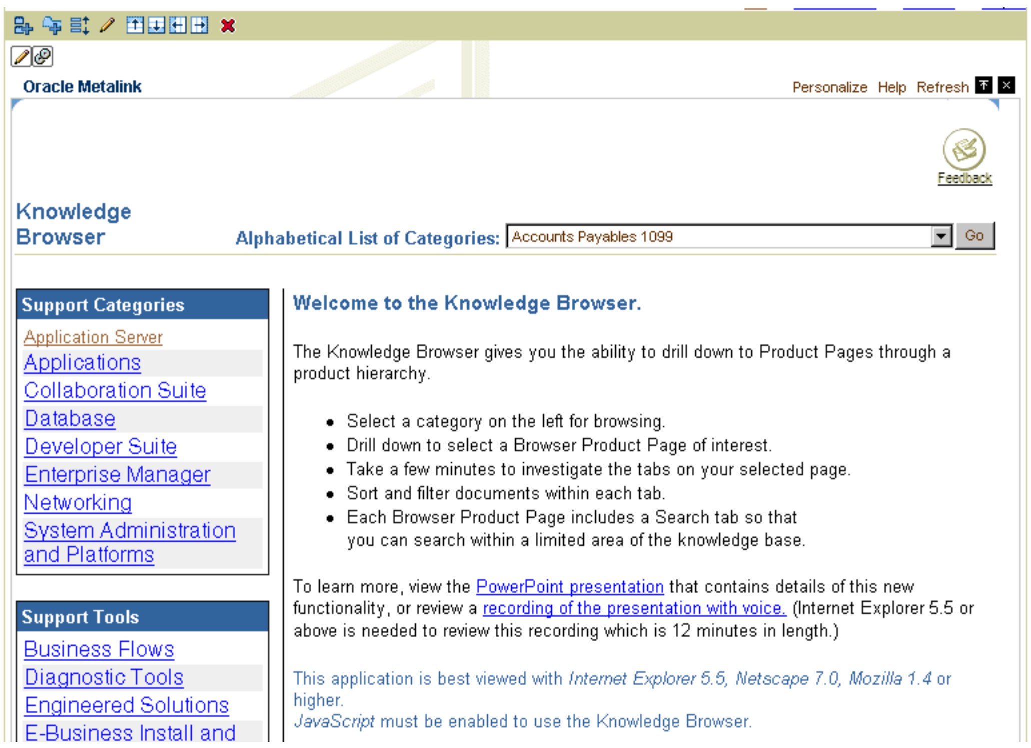
You can set some properties of the Web clipping.

URL Rewriting:	Inline
Title:	Oracle Metalink
Description:	Oracle Metalink
Time Out (seconds):	13
Expires (minutes):	30

Please choose a value in range[1, 60].

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. **OK** to display the selected Web clipping in the Web Clipping portlet on your page, as shown in Figure 5–16.

Figure 5–16 External Application Displayed in Portlet



Knowledge Browser

Alphabetical List of Categories: Accounts Payables 1099 Go

Support Categories

- Application Server
- Applications
- Collaboration Suite
- Database
- Developer Suite
- Enterprise Manager
- Networking
- System Administration and Platforms

Support Tools

- Business Flows
- Diagnostic Tools
- Engineered Solutions
- E-Business Install and

Welcome to the Knowledge Browser.

The Knowledge Browser gives you the ability to drill down to Product Pages through a product hierarchy.

- Select a category on the left for browsing.
- Drill down to select a Browser Product Page of interest.
- Take a few minutes to investigate the tabs on your selected page.
- Sort and filter documents within each tab.
- Each Browser Product Page includes a Search tab so that you can search within a limited area of the knowledge base.

To learn more, view the PowerPoint presentation that contains details of this new functionality, or review a recording of the presentation with voice. (Internet Explorer 5.5 or above is needed to review this recording which is 12 minutes in length.)

This application is best viewed with *Internet Explorer 5.5, Netscape 7.0, Mozilla 1.4* or higher.
JavaScript must be enabled to use the Knowledge Browser.

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 provider. For each external application, you must register a new provider. Each portal user accesses the authenticated content using their user name and password for that system, not the page designer's credentials.

5.4 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:

- Section 5.4.1, "Adding a Web Clipping Portlet to a Personal Page"
- Section 5.4.2, "Selecting a Clipping in OTN"
- Section 5.4.3, "Personalizing a Web Clipping Portlet"

5.4.1 Adding a Web Clipping Portlet to a Personal Page

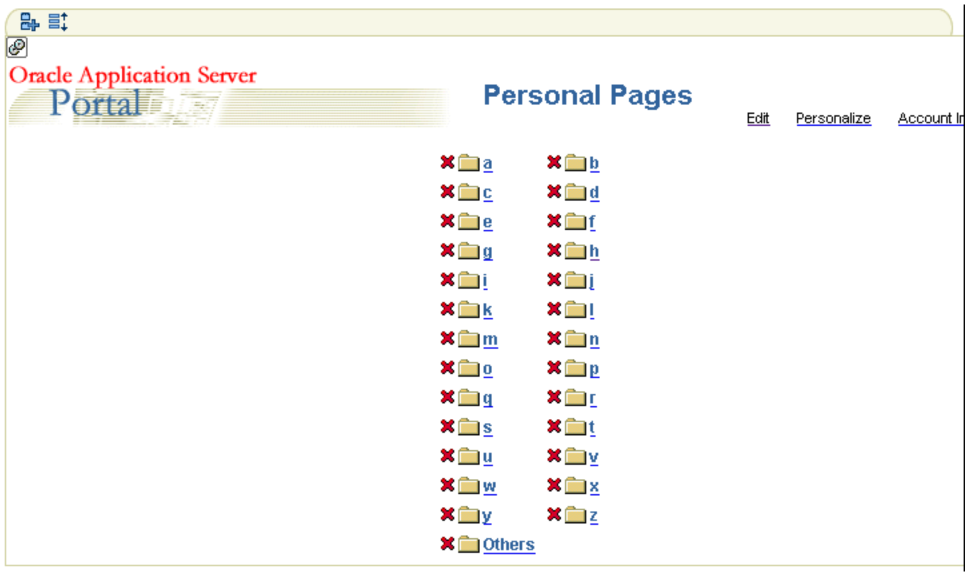
Administrators can set up personal pages for all users. This task assumes that the administrator has enabled this functionality. This task explains how to add the Web Clipping portlet to your personal page. To do this, perform the following steps:

1. In the **Work In** field of the Page Groups portlet, select the page group that contains personal pages.

   By default the Page Groups portlet is located on the Build tab of the Portal Builder page and the Personal Pages are located in Shared Objects page group.

2. In the **Pages** section, select Personal Pages. Expand the node for the first letter of your user name. To View the Personal pages page (Figure 5–17), click on View link in Page Group portlet.

Figure 5–17 Expanding Page Group Map Nodes



3. Click your Page Name. Your personal page is displayed.
4. In any portlet region, click the **Add Portlets** icon.
5. In the Add Portlets page, click the **Web Clipping Portlet** link.

   By default, the Web Clipping portlet is located in the Portal Builder page of the Portlet Repository. If you cannot find this page, use the **Search** field to find the portlet.

6. The Web Clipping portlet is added to the Selected Portlets list. Click **OK.**

5.4.2 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. In the Web Clipping portlet, click the **Edit Defaults** icon.
2. In the Web Clipping Studio's Find a Web Clipping page, in the **URL Location** field, enter:

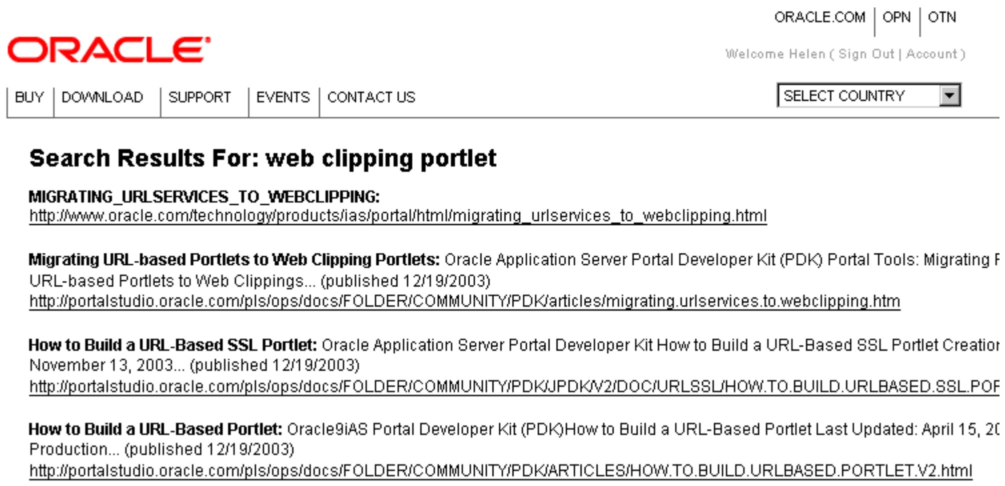
   ```
   http://www.oracle.com/technology/products/ias/portal/index.html
   ```

   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 **Search.**

The Search result is displayed in the Web Clipping Studio, as shown in Figure 5–18.

Figure 5–18 Searching for Information on OTN


ORACLE.COM | OPN | OTN
Welcome Helen (Sign Out | Account)
BUY | DOWNLOAD | SUPPORT | EVENTS | CONTACT US
SELECT COUNTRY

Search Results For: web clipping portlet

MIGRATING_URLSERVICES_TO_WEBCLIPPING:
http://www.oracle.com/technology/products/ias/portal/html/migrating_urlservices_to_webclipping.html

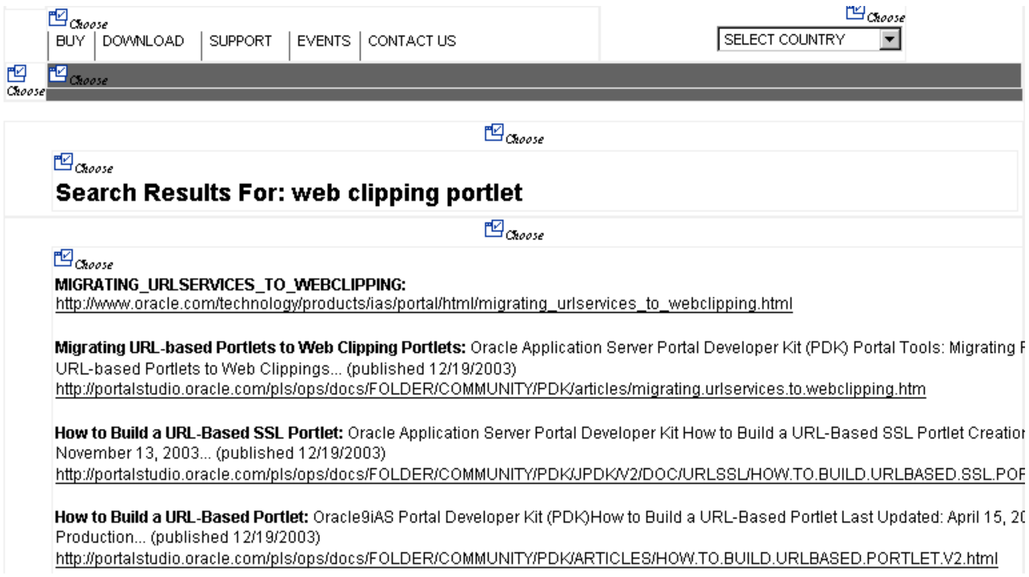
Migrating URL-based Portlets to Web Clipping Portlets: Oracle Application Server Portal Developer Kit (PDK) Portal Tools: Migrating F
URL-based Portlets to Web Clippings... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/articles/migrating.urlservices.to.webclipping.htm

How to Build a URL-Based SSL Portlet: Oracle Application Server Portal Developer Kit How to Build a URL-Based SSL Portlet Creatior
November 13, 2003... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/JPDK/V2/DOC/URLSSL/HOW.TO.BUILD.URLBASED.SSL.POF

How to Build a URL-Based Portlet: Oracle9iAS Portal Developer Kit (PDK)How to Build a URL-Based Portlet Last Updated: April 15, 20
Production... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/ARTICLES/HOW.TO.BUILD.URLBASED.PORTLET.V2.html

4. Click **Section.** Web Clipping Studio divides the target Web page into its clippable sections, as shown in Figure 5–19.

Figure 5–19 Sectioning the Target Web Page



Search Results For: web clipping portlet

MIGRATING_URLSERVICES_TO_WEBCLIPPING:
http://www.oracle.com/technology/products/ias/portal/html/migrating_urlservices_to_webclipping.html

Migrating URL-based Portlets to Web Clipping Portlets: Oracle Application Server Portal Developer Kit (PDK) Portal Tools: Migrating F
URL-based Portlets to Web Clippings... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/articles/migrating.urlservices.to.webclipping.htm

How to Build a URL-Based SSL Portlet: Oracle Application Server Portal Developer Kit How to Build a URL-Based SSL Portlet Creatior
November 13, 2003... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/JPDK/V2/DOC/URLSSL/HOW.TO.BUILD.URLBASED.SSL.POF

How to Build a URL-Based Portlet: Oracle9iAS Portal Developer Kit (PDK)How to Build a URL-Based Portlet Last Updated: April 15, 20
Production... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/ARTICLES/HOW.TO.BUILD.URLBASED.PORTLET.V2.html

5. At the top left corner of the search result, click **Choose.**

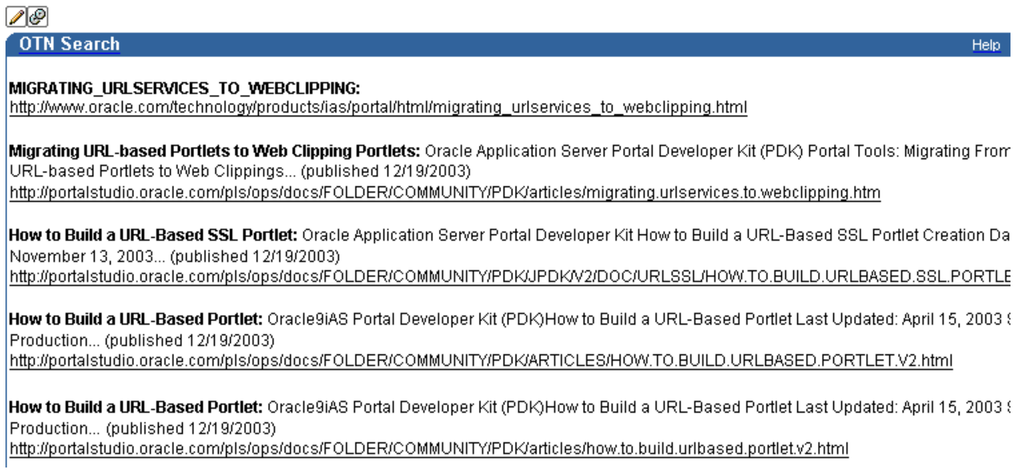
   A preview of the search result section displays.

   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, 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 5–20 shows the Web Clipping displayed in the page.

Figure 5–20 Selected Web Clipping Displayed in Web Clipping Portlet



OTN Search Help

MIGRATING_URLSERVICES_TO_WEBCLIPPING:
http://www.oracle.com/technology/products/ias/portal/html/migrating_urlservices_to_webclipping.html

Migrating URL-based Portlets to Web Clipping Portlets: Oracle Application Server Portal Developer Kit (PDK) Portal Tools: Migrating From URL-based Portlets to Web Clippings... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/articles/migrating.urlservices.to.webclipping.htm

How to Build a URL-Based SSL Portlet: Oracle Application Server Portal Developer Kit How to Build a URL-Based SSL Portlet Creation Da November 13, 2003... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/JPDK/V2/DOC/URLSSL/HOW.TO.BUILD.URLBASED.SSL.PORTLE

How to Build a URL-Based Portlet: Oracle9iAS Portal Developer Kit (PDK)How to Build a URL-Based Portlet Last Updated: April 15, 2003 Production... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/ARTICLES/HOW.TO.BUILD.URLBASED.PORTLET.V2.html

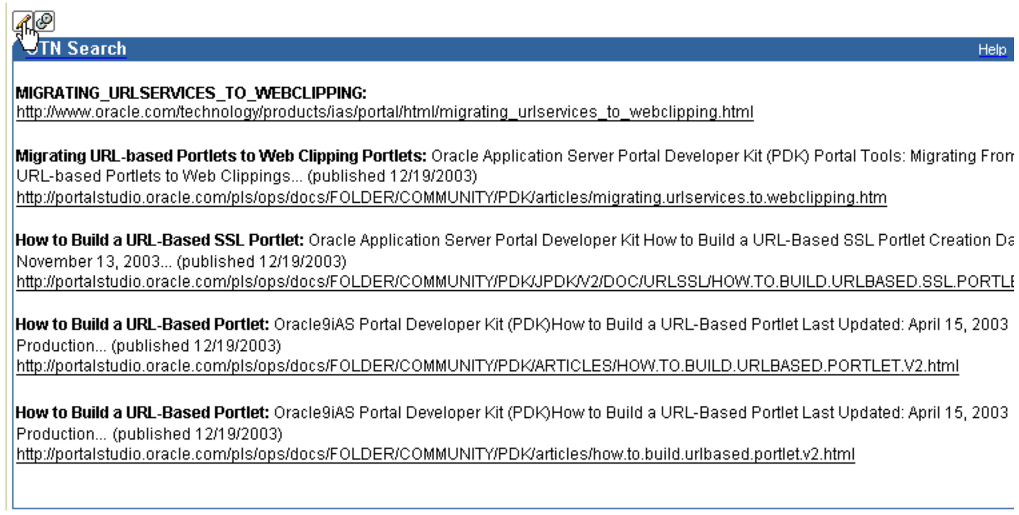
How to Build a URL-Based Portlet: Oracle9iAS Portal Developer Kit (PDK)How to Build a URL-Based Portlet Last Updated: April 15, 2003 Production... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/articles/how.to.build.urlbased.portlet.v2.html

5.4.3 Personalizing a Web Clipping Portlet

In this task, you edit the properties of the Web Clipping portlet to allow end users to display different search results in the portlet. To do this, perform the following tasks:

1. Above the Web Clipping portlet you just added, click the **Edit Defaults** icon, as shown in Figure 5–21.

Figure 5–21 Clicking Edit Defaults for the Web Clipping Portlet



TN Search Help

MIGRATING_URLSERVICES_TO_WEBCLIPPING:
http://www.oracle.com/technology/products/ias/portal/html/migrating_urlservices_to_webclipping.html

Migrating URL-based Portlets to Web Clipping Portlets: Oracle Application Server Portal Developer Kit (PDK) Portal Tools: Migrating From URL-based Portlets to Web Clippings... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/articles/migrating.urlservices.to.webclipping.htm

How to Build a URL-Based SSL Portlet: Oracle Application Server Portal Developer Kit How to Build a URL-Based SSL Portlet Creation Da November 13, 2003... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/JPDK/V2/DOC/URLSSL/HOW.TO.BUILD.URLBASED.SSL.PORTLE

How to Build a URL-Based Portlet: Oracle9iAS Portal Developer Kit (PDK)How to Build a URL-Based Portlet Last Updated: April 15, 2003 Production... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/ARTICLES/HOW.TO.BUILD.URLBASED.PORTLET.V2.html

How to Build a URL-Based Portlet: Oracle9iAS Portal Developer Kit (PDK)How to Build a URL-Based Portlet Last Updated: April 15, 2003 Production... (published 12/19/2003)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/articles/how.to.build.urlbased.portlet.v2.html

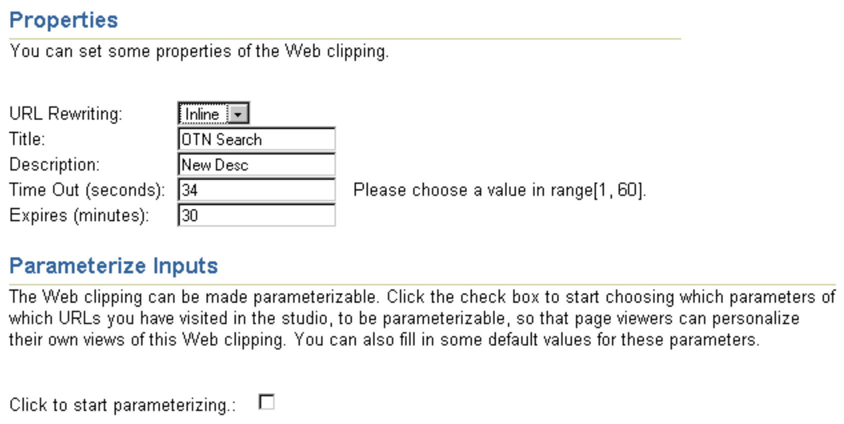
2. In the Find a Web Clipping page, modify the following items in the **Properties** section:

   - 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 5–22 shows the **Properties** and **Parameterize Inputs** sections of the Find a Web Clipping page.

Figure 5–22 Setting Properties for a Web Clipping



Properties

You can set some properties of the Web clipping.

URL Rewriting:	Inline	
Title:	OTN Search	
Description:	New Desc	
Time Out (seconds):	34	Please choose a value in range[1, 60].
Expires (minutes):	30	

Parameterize Inputs

The Web clipping can be made parameterizable. Click the check box to start choosing which parameters of which URLs you have visited in the studio, to be parameterizable, so that page viewers can personalize their own views of this Web clipping. You can also fill in some default values for these parameters.

Click to start parameterizing.: ☐

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 allow end users to specify their own search string.

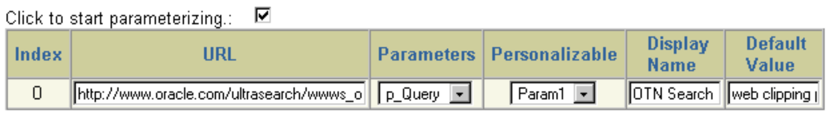
   In the **Parameterize Inputs** section, select the **Click to start parameterizing** check box.

4. In the parameters table, make the following changes:

   - In the **Parameters** column, choose **p_Query** from the list.
   - In the **Personalizable** column, choose **Param1** from the list. You can manually add more parameters in the provider.xml file if you need to.
   - 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.

   Figure 5–23 shows the parameters table.

Figure 5–23 Specifying Parameters for User Input

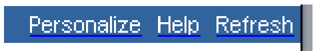


Click to start parameterizing.: ✓

Index	URL	Parameters	Personalizable	Display Name	Default Value
0	http://www.oracle.com/ultrasearch/wwws_o	p_Query	Param1	OTN Search	web clipping p

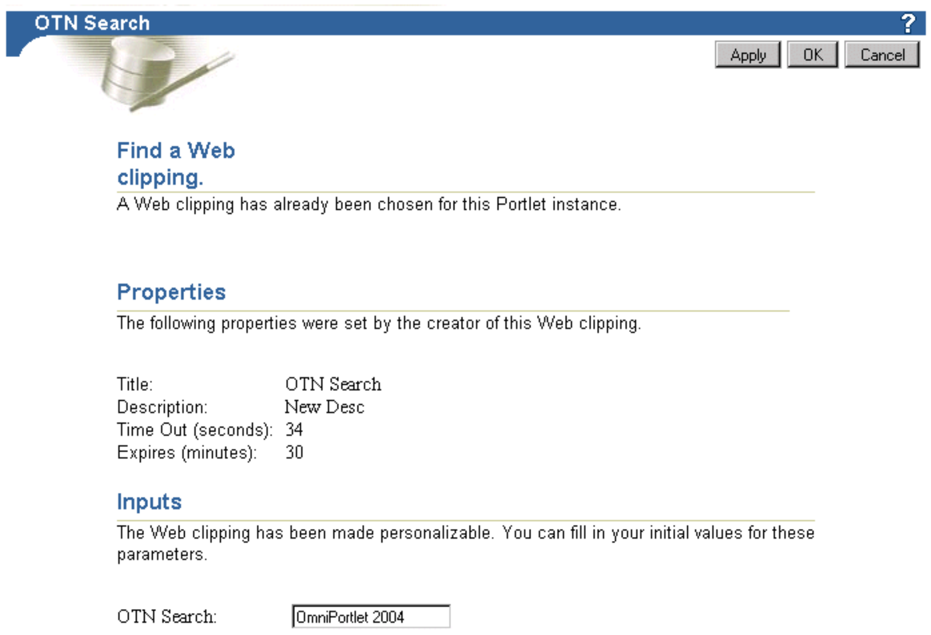
5. Click **OK** to display the default search results in the Web Clipping portlet on your page.
6. In the Editing Views section, click **View Page.**
7. In the Web Clipping portlet header, click **Personalize**, as shown in Figure 5–24.

Figure 5–24 Clicking Personalize in the Web Clipping Portlet Header



8. 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 5–25.

Figure 5–25 Specifying Input for Parameters



Find a Web clipping.

A Web clipping has already been chosen for this Portlet instance.

Properties

The following properties were set by the creator of this Web clipping.

Title:	OTN Search
Description:	New Desc
Time Out (seconds):	34
Expires (minutes):	30

Inputs

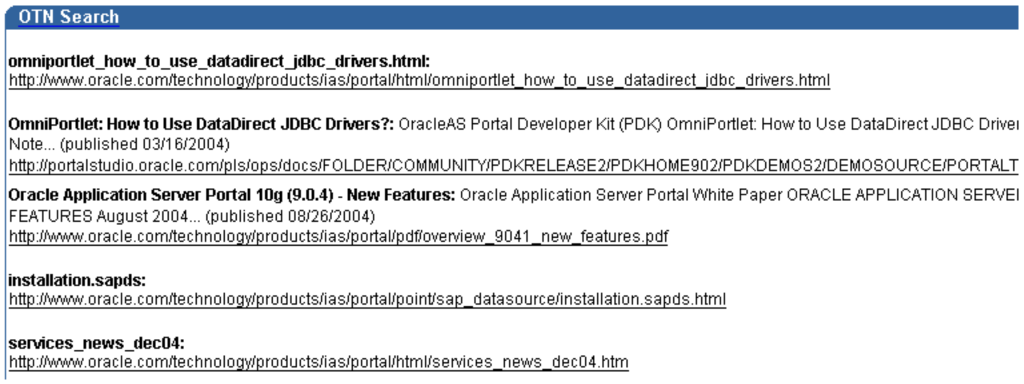
The Web clipping has been made personalizable. You can fill in your initial values for these parameters.

OTN Search: OmniPortlet 2004

9. Click **OK.**

   The Web Clipping portlet now displays the results of performing a search on OTN for OmniPortlet 2004 information, as shown in Figure 5–26.

Figure 5–26 New Web Clipping Result Based on Customer Input Parameter



omniportlet_how_to_use_datadirect_jdbc_drivers.html:
http://www.oracle.com/technology/products/ias/portal/html/omniportlet_how_to_use_datadirect_jdbc_drivers.html

OmniPortlet: How to Use DataDirect JDBC Drivers?: OracleAS Portal Developer Kit (PDK) OmniPortlet: How to Use DataDirect JDBC Drive
Note... (published 03/16/2004)
http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDKRELEASE2/PDKHOME902/PDKDEMOS2/DEMOSOURCE/PORTALT

Oracle Application Server Portal 10g (9.0.4) - New Features: Oracle Application Server Portal White Paper ORACLE APPLICATION SERVE
FEATURES August 2004... (published 08/26/2004)
http://www.oracle.com/technology/products/ias/portal/pdf/overview_9041_new_features.pdf

installation.sapds:
http://www.oracle.com/technology/products/ias/portal/point/sap_datasource/installation.sapds.html

services_news_dec04:
http://www.oracle.com/technology/products/ias/portal/html/services_news_dec04.htm

5.5 Using Web Clipping Open Transport API

To support custom authentication methods, users can use the Web Clipping Transport API. To extend the Web Clipping transport layer to support custom authentication methods, users must perform the following implementation and deployment procedures:

Implementation

Users can implement their own transport classes.

- Users can provide overrides for two use cases of the `oracle.portal.wcs.transport.http.HttpTransportLiaison` interface. In Web Clipping, this interface is used to abstract the HTTP transport layer. By default, the two use cases of this interface are manifested by following implementations:
  - `HttpClientStudioTransportLiaison`, which handles HTTP transport in Web Clipping Studio mode
  - `HttpClientProviderTransportLiaison`, which handles HTTP transport in Web Clipping Producer show mode

  To support more authentication methods, users must override the `addRequestHeaders` methods for both the Studio and Provider `HttpClientTransportLiaison` implementations to add their own authentication-specific headers. For more information about implementation, see Oracle WebLogic Server Web Clipping Transport API Reference.
- Users must compile the new subclasses and package them into a jar file. To compile the new subclasses, users can use the following command:

  ```
  javac -classpath path_to_wcejar -d classes/
  ```

  Where, path_to_wcejar refers to the path to the `wce.jar` file.

  To create the jar file, for example, users can use the following command from the `classes` directory:

  ```
  jar cvf ../mytransport.jar
  ```

  Where, *`mytransport.jar`* refers to t he jar file users want to create.

Deployment

Users must deploy the jar file to support the custom authentication method.

- Users must place the jar file into the class path or shared library that is used by the Web Clipping producer at runtime.
- Users must register the transport class in the `web.xml` file for Web Clipping producer by making the following modifications to the context parameters defined for `HttpClientProviderTranportiaison` and `HttpClientStudioTransportLiaison`:
  - Change the parameter value for `oracle.webclipping.provider.TransportLiaisonClass` to the name of the new class extended from the `HttpClientProviderTransportLiaison` class.
  - Change the parameter value for `oracle.webclipping.studio.TransportLiaisonClass` to the name of the new class extended from the `HttpClientStudioTransportLiaison` class.

- Finally, users must restart the producer server for the changes to take effect.

5.6 Migrating from URL-Based Portlets

You can migrate URL-based portlets that are stored in a `provider.xml` file to Web Clipping portlets. These Web Clipping portlets will exist in the Web Clipping Repository, but you can access the portlets through pointers defined in the `provider.xml` file.

Note that, during the migration process, no modifications will be done on the original files used by these URL-based portlets.

This section describes the following tasks:

- Section 5.6.1, "Preparing for Migration"
- Section 5.6.2, "Performing the Migration"
- Section 5.6.3, "Post-Migration Configuration"
- Section 5.6.4, "Maintaining Migrated Portlets"
- Section 5.6.5, "Limitations in Migrating URL-Based Portlets"

5.6.1 Preparing for Migration

Before you begin the migration, take the following steps:

1. Verify that the Web provider that contains the URL-based portlets you want to migrate is functional. The migration process will not succeed with non-functional URL-based portlets. To verify, make sure that the URL-based portlets are used by your Portal pages and that they appear correctly.
2. Find existing URL-based portlets.

   Before you run the migration tool, make sure that you know the file path to the service deployment properties file, which holds pointers to everything about the service, including the path to the provider.xml file that holds the URL services definitions.

   You must run the migration tool once for each service. If you have multiple services, find the entire list of service deployment properties files that will be used during migration.

   The location of these service deployment properties files may vary depending on individual deployment scenarios. Typically, with JPDK samples, they are located in:

   ```
   <Domain_home>/servers/WLS_PORTAL/tmp/_WL_user/jpdk
   ```

   JPDK samples that are shipped with the Oracle Portal are located in the same directory path.

   The list of sample service deployment properties files include the following:

   - urlbasicauth.properties
   - urlexternalauth.properties
   - urlnls.properties
   - urlparams.properties
   - urlsample.properties

- urlssl.properties

  In the migration process, the service deployment properties file as well as the provider.xml file used to contain old URL-based portlets will be copied and imported into the Web Clipping application, together with all the previous customizations based on the File Preference store. Because no modifications will be done on the original files used by these URL-based portlets, you do not need to back up any of these files. They can continue to be used as URL-based portlets.

3. Depending on the types of URL-based portlets you have, you may need to rearrange them as follows:

   - Because Web Clipping does not offer parallel functionality for the XML Filter capability that URL-based portlets offer, isolate all of the URL-based portlets that use the XML Filter into a separate service deployment (and `provider.xml` file) that will not be migrated.
   - Because you may face limitations when migrating URL-based portlets that use external applications, you may decide that the migration is too costly for that subset of your URL-based portlets. In this case, you can isolate them into a separate service deployment (and `provider.xml` file) that will not be migrated.

4. Make sure that the machine where you are planning to run the migration tool has local access to the service deployment files and files that the deployment files point to, for both the URL-based portlets provider and the Web Clipping provider.

5. Make sure that both the Web Clipping Repository and the HTTP Proxy are configured, and that the proxy settings concur with those used by your deployment of URL-based portlets. If you have a proxy server that requires authentication, the Web Clipping provider must be configured with **Requires Authentication** enabled. Set the Login scheme to use a global log in (**Use Login below for all users**).

   Check the Web Clipping Provider Test page at:

   ```
   http://host:port/portalTools/webClipping/providers/webClipping
   ```

   See the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* for more information about setting proxy authentication.

6. Shut down the WLS instance where your Portal Tools application is running. This step ensures that no modifications are made to the files that will be migrated. Do not restart the WLS instance until after the migration is completed.

   To shut down the WLS instance in a Portal instance, use the following command (from the *`Oracle\Middleware\user_projects\domains\<Domain Name>\bin`* directory):

   ```
   stopWebLogic.cmd
   ```

   Domain Name refers to the domain you have configured for your Portal.

5.6.2 Performing the Migration

The migration tool parses the service deployment properties file to find the `provider.xml` file that contains the XML definition of the URL-based portlets to be migrated.

The tool changes the Provider Definition class from URLProviderDefinition to WcProviderDefinition. For each of the URL-based portlets defined in the `provider.xml` file, an equivalent clipping definition is created and inserted into the

Web Clipping Repository. Then, the PortletDefinition XML snippet corresponding to the portlet being migrated is replaced with a new one that uses WcPortletDefinition as its class. The replacement XML snippet contains a Clip Id that is a pointer to the newly inserted clipping definition.

The new PortletDefinition XML snippet will no longer contain the same information as it did when it was a URLPortletDefinition. Namely, the migration tool uses the URL to visit as well as the headerTrimTag and footerTrimTag to generate an equivalent Web Clipping definition that will be stored within the Web Clipping Repository and only accessible from the `provider.xml` file through the Clip Id. The only information remaining will be the title and description. See Section 5.2.3, "Setting Web Clipping Portlet Properties" for instructions on modifying your clipping.

To migrate portlets, take the following steps:

1. Set environment variables as follows:
   - On UNIX, set LD_LIBRARY_PATH to either the *`Install_home`*`/lib` or *`Install_home`*`/lib32`, depending on whether the operating system is 32-bit or 64-bit.
   - On Windows, set PATH to *`Install_home`*`\bin`.
2. Change to the *`ORACLE_HOME`*`/portal/jlib` directory.
3. Use the following command to import the portlet to Web Clipping:

   ```
   java -Doracle.ons.oraclehome=ORACLE_HOME -jar wcwebdb.jar -import -from
   urlservices webclip_depprop urlsvc_depprop
   ```

   In the example, the parameters have the following meanings:

   - *ORACLE_HOME* refers to the directory where Oracle application is installed.
   - *webclip_depprop* refers to the path to your webClipping service deployment properties file. The path is typically:

     ```
     MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\tmp\_WL_
     user\portalTools_11.1.1.1.0\kjdcke\war\WEB-INF\deployment
     ```

     Inst_Home refers to the location where Portal is installed. *WLS_PORTAL* refers to the WLS instance where your PortalTools application is deployed.
   - *urlsvc_depprop* refers to the path for the URL-based portlets service deployment properties file that points to URL-based portlets that you wish to migrate.

   For example, assume your URL-based portlets are deployed in the following directory on Windows:

   ```
   MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\tmp\_WL_
   user\\portalTools_11.1.1.1.0\kjdcke\war\WEB-INF\deployment\providers
   ```

5.6.3 Post-Migration Configuration

After you migrate all of the URL-based portlets, take the following steps:

1. Start the WLS instance that hosts your provider.

   To start the WLS instance, use the following command:

   ```
   startWeblogic.cmd
   ```

2. Update all Portal instances where URL-Based portlets were previously registered. Go to Portal Navigator to find your previously registered URL-based Portlet provider. Click **Edit Registration** and then click the **Connection** tab to edit the connection information to the newly migrated provider. Depending on how you registered the portlet, perform either of the following steps:

   - If you previously registered your URL-based Portlet provider using the Service Id mechanism, your provider URL looked similar to the following:

     ```
     http://host:port/jpdk/jpdk/providers
     ```

     In this case, because the Service Id field already contains `urn:someservice`, you now need only to update the provider URL value to the following:

     ```
     http://host:port/portalTools/webClipping/providers
     ```

   - If you registered your URL-based Portlet provider using a provider URL that contained the Service Id, your provider URL looked similar to the following:

     ```
     http://host:port/jpdk/jpdk/providers/service_id
     ```

     In this case, you need to update the provider to the following:

     ```
     http://host:port/portalTools/webClipping/providers/service_id
     ```

   For the most part, the rest of the registration details can remain as they are. However, if you plan on using the in-line rendering functionality of the migrated portlets, you must select **Once Per User Session** for **Login Frequency.**

3. Restore HTTP proxy authentication setting that you altered in preparation for the migration. During the migration process, URL-based portlets are converted to Web Clipping portlets. If the Web Clipping provider has originally been configured with **Requires Authentication** enabled, your migrated URL-based portlets may not function properly.

   Check the test page at:

   ```
   http://host:port/portalTools/webClipping/providers/webClipping
   ```

   If the Login Scheme is not set to **Use login below for all users**, then authentication is accomplished by mapping the current portal user to a proxy server username and password. A portal user's proxy server credentials can only be entered through the **Personalize** link of the Web Clipping portlet. However this cannot be accomplished through the **Personalize** link of the migrated URL-based portlets. The workaround is to temporarily add a Web Clipping portlet to the page and personalize it to provide the security credentials. After this is complete, you can remove the temporary portlet.

4. Review each Portal page where you had used URL-based portlets and verify that the page still contains the content. Occasionally, you may find that, after migration, what used to be a "trimmed" Web page is now the entire Web page. This may be due to some of the limitations of the migration tool, which are discussed in Section 5.6.5, "Limitations in Migrating URL-Based Portlets." If this happens, edit the portlet by altering the Web Clipping associated with it.

5.6.4 Maintaining Migrated Portlets

After you have migrated the portlets, you no longer use the `provider.xml` file to edit the portlets. Instead, you use Web Clipping to edit the portlets, as described in Section 5.2.3, "Setting Web Clipping Portlet Properties."

Note the following:

- Whatever modifications you make to the clipping will affect it at the portlet definition level and therefore will affect all the instances of that portlet which may exist across Portal pages.
- When you modify the clipping and you have made the portlet rendering to be Inline, and have registered the provider with a Login Frequency of **Once per User Session**, then page viewers who have an instance of the particular portlet on their page, will need to log out of Oracle Portal and log in again to see the changes implemented. This feature avoids disruption of the page viewer's usage of a particular portlet when it is being modified.
- For portlets with Web Clippings that require authentication to an external application prior to accessing the page containing the clip, the "off-line" editing mechanism detailed in the previous item does not provide such authentication information. Consequently, there is no way to modify the migrated clipping if the original URL-based portlet uses external applications. There is currently no workaround for this limitation.

5.6.5 Limitations in Migrating URL-Based Portlets

This sections describes limitations in migrating URL-based portlets to Web Clipping and describes some differences between the two methods. Following are the differences:

- There is a fundamental difference between URL-based portlets and Web Clipping in the way that clippings are defined.

  URL-based portlets use the headerTrimTags and footerTrimTags tags to define the begin and end of a particular section in an HTML page. On the other hand, Web Clippings use the HTML tag path (for example, `html/body/table[2]/tr[2]/td[1]`) to denote the path to use when looking for an HTML tag that would contain the clipping within the page.

  While the header and footer trim tags for URL-based portlets offer greater flexibility (that is, the clipping does not need to reside within one single HTML tag), it is not as robust. When a clipping can no longer be found in the same location within a page, there is no way to apply fuzzy matching logic to find the clipping again. On the other hand, Web Clipping stores key pieces of information to locate the clipping within the page. If the clipping is not in the same location, the fuzzy match feature looks for other candidates on the same HTML page to account for the possibility that the clip may have moved.
- The migration tool tries to find the clipping (HTML Tag) that encapsulates the headerTrimTags and footerTrimTags. Often, this will not be the exact HTML that was extracted by the URL-based portlet. In these cases, the entire Web page will be returned as the content of the migrated Web Clipping Portlet. To amend this, simply edit the portlet.
- URL-based portlets that have been migrated to Web Clipping do not support proxy authentication by default. This is because URL-based portlets inherently do not support proxy authentication and Web Clipping preserves the edit mode of the portlets. Their edit mode does not provide an opportunity to enter authentication information. To work around this restriction, add an empty Web Clipping portlet to the same portal and use the Web Clipping portlet **Personalize** link to enter the user name and password for proxy authentication.
- URL-based portlets that used external applications are migrated as full-page Web Clipping portlets. This means that the URL previously specified by the URL-based

portlet definition will be the one used to display the full page after the log in process. Whatever headerTrimTags and footerTrimTags you have previously specified will be lost, because external application integration occurs at the Portal side while the migration tool only handles the provider side. Because you are not connected to any Portal instances at the time of the migration, and the external applications framework resides only on the Portal side, the migration tool cannot fetch the authentication information necessary to log in to parse the HTML and use the headerTrimTags and footerTrimTags that you have previously specified to compute an HTML Tag Path to store in the Web Clipping Definition.

In maintaining migrated portlets, you can edit the clipping through links in the test page. However, this does not include URL-based portlets that used external applications to authenticate themselves to the external sites. There is currently no workaround for this issue.

- URL-based portlets that use the XML Filter cannot be migrated to Web Clipping portlets because there is a lack of such a functionality in the Web Clipping Portlet. Before proceeding with migration, make sure that you have backed up the URL-based portlets that use the XML Filter into a separate URL-based portlet provider. You must do this because the migration occurs at the provider level. That is, upon interpreting that a particular `provider.xml` file has URLProviderDefinition for its provider definition class, all the portlet definitions contained within that provider definition are migrated.

5.7 Current Limitations for Web Clipping

This section describes current limitations for Web Clipping. For information about the latest limitations in a release, be sure to read the *Oracle Portal 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, 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 Portal pages. As a workaround, examine the portlet that is supplying the data and take the appropriate action, as follows:
  – For database provider portlets, use export/import to copy pages across portals.
  – For Web provider portlets, re-register the same provider in the destination portal and edit the portal manually.
- You cannot use the Web Clipping portlet to clip a Web page that contains more that one frame, that is, a frameset.
- 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, 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, the style that will be displayed depends on the browser.

- Web Clipping checks for NLS settings in the following way:
  1. Web Clipping checks the `Content-Type` in the HTTP header for the charset attribute. If this is present, it assumes that this is the character encoding of the HTML page.
  2. If the charset attribute is not present, it checks the HTML `META` tag on the page to determine the character encoding.
  3. If the HTML `META` tag is not found, Web Clipping uses the charset in the previous browsed page. If this is the first page, 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), 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 higher, or Microsoft Internet Explorer 5.5 or higher for Windows 2000, or Microsoft Internet Explorer 6.0 or higher for Windows XP.

  If you use browser versions older than these, you may encounter JavaScript errors.

For troubleshooting information, see Appendix B, "Troubleshooting Portlets and Providers."

6 Creating Java Portlets

This chapter explains how to create Java portlets based on the Java Portlet Specification (JSR 168) or the Oracle 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:

- Section 6.1, "Guidelines for Writing Java Portlets"
- Section 6.2, "Introduction to Java Portlet Specification (JPS) and WSRP"
- Section 6.3, "Building JPS-Compliant Portlets with Oracle JDeveloper"
- Section 6.4, "Introduction to Oracle PDK-Java"
- Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper"

More On OTN

The source code for many of the examples referenced in this chapter is available as part of PDK-Java. You can download PDK-Java from the Oracle Portal Developer Kit (PDK) 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`.

6.1 Guidelines for Writing 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 6.1.1, "Guidelines for Show Modes"
- Section 6.1.2, "Guidelines for Navigation within a Portlet"
- Section 6.1.3, "Guidelines for JavaScript"
- Section 6.1.4, "Guidelines for Mobile Portlets"

6.1.1 Guidelines for Show Modes

Show mode exhibits the runtime portlet functionality seen by users. JPS offers some modes not offered by PDK-Java and vice versa. If you are coding portlets to JPS, 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 call 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

The different Show modes that a portlet may have, each with its own visualization and behavior, are discussed in the following sections:

- Section 6.1.1.1, "Shared Screen Mode (View Mode for JPS)"
- Section 6.1.1.2, "Edit Mode (JPS and Pdk-Java)"
- Section 6.1.1.3, "Edit Defaults Mode (JPS and PDK-Java)"
- Section 6.1.1.4, "Preview Mode (JPS and PDK-Java)"
- Section 6.1.1.5, "Full Screen Mode (PDK-Java)"
- Section 6.1.1.6, "Help Mode (JPS and Oracle Portal)"
- Section 6.1.1.7, "About Mode (JPS and PDK-Java)"
- Section 6.1.1.8, "Link Mode (PDK-Java)"

6.1.1.1 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. A JPS portlet must have a view mode, the rest are optional.

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 a page. In Oracle Portal, portlets are rendered inside HTML table cells when in Shared Screen mode. This means a portlet can display only content that can be rendered within a table cell, including, among other technologies, HTML, plug-ins, and Java applets. The actual size of the table cell is variable depending on user settings, the browser width, and the amount and style of content in the portlet.

6.1.1.1.1 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, and 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.

- **Avoid elements that cannot be rendered properly in an HTML table cell.** Some constructs cannot be used simply because they do not display correctly in a table cell. Frames, for example, 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, 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 wish and they can change it at any time. Your portlet should handle these situations gracefully.

The HTML you use also impacts 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 deeply nested tables.** Deeply nested tables slow performance dramatically in some older browser versions. Oracle Portal draws several levels of tables to render portlets. If your portlets use tables within tables, your users may have to wait quite a while for those pages to render.
- **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, it can greatly slow the rendering of the page.

6.1.1.1.2 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 Oracle Portal page. The portlets access these settings for their fonts and colors, either directly or using the API.

While different browsers have implemented varying levels of the full CSS specification, Oracle Portal uses a very basic subset of this specification to allow 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, 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 Oracle Portal 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 portal pages, you cannot guarantee that your portlet can appear in a particular spot.
- **Follow Accessibility Standards.** You should ensure that you code you style sheets according to existing accessibility standards (more information available at `http://www.w3.org/TR/WCAG10-CSS-TECHS/`).

6.1.1.2 Edit Mode (JPS and Pdk-Java)

A portlet uses Edit mode to allow 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.

Portal users typically access a portlet's Edit mode by clicking **Personalize** on the portlet banner. When users click **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.

6.1.1.2.1 Guidelines for Edit Mode Operations The following guidelines should govern what you expose to users in Edit mode:

- **Allow users to personalize the title of the portlet.** The same portlet may be added to the same portal page several times. Allowing 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, 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 provider settings or do other administrative tasks, you should create secured portlets specifically for those tasks.
- **Only show mobile options when applicable.** The portlet can interrogate whether an Oracle Portal instance is enabled for mobile devices. If the instance is not mobile-enabled, then you need not show any mobile-specific options. Furthermore, if the page might serve both mobile and desktop users, you should consider delineating between mobile options and desktop options. Refer to Section 6.1.4.6, "Tailor Personalization Pages" for additional tips.

6.1.1.2.2 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 shared screen mode.
- **Apply** saves the user personalizations and reloads the current page.
- **Cancel** returns the portlet to shared screen mode without saving changes.

6.1.1.2.3 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, 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, 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 portal. PDK-Java makes this type of logic easy to implement.

6.1.1.3 Edit Defaults Mode (JPS and PDK-Java)

A portlet uses the Edit Defaults mode to enable page designers to customize the default behavior of a particular portlet instance. Edit Defaults mode provides a list of settings that the page designer 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.

Page designers access Edit Defaults mode, when editing a page, by clicking the Edit Defaults icon just above the portlet banner.

When users click the Edit Defaults icon, a new page appears in the same browser window. The portlet typically creates a Web page representing a dialog box to customize the portlet instance settings. After applying the settings, users are automatically returned to the original page.

6.1.1.3.1 Guidelines for Edit Defaults Mode Options The following guidelines should govern what you expose to page designers in Edit Defaults mode:

- **If using caching, invalidate the cache.** If customizations cause a change in portlet display or content, 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 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 provider settings or do other administrative tasks, you should create secured portlets specifically for those tasks.
- **Only show mobile options when applicable.** The portlet can interrogate whether an Oracle Portal instance is enabled for mobile devices. If the instance is not mobile-enabled, then you need not show any mobile-specific options. Furthermore, if the page might serve both mobile and desktop users, you should consider delineating between mobile options and desktop options. Refer to Section 6.1.4.6, "Tailor Personalization Pages" for additional tips.

6.1.1.3.2 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 customizions and returns the portlet to shared screen mode.

- **Apply** saves the user customizations and reloads the current page.
- **Cancel** returns the portlet to shared screen mode without saving changes.

6.1.1.3.3 Guidelines for Rendering Personalization Values When you show the forms used to change customization settings, you should default the values so that the page designer does not have to constantly re-enter settings. When rendering customization 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 customizations are found, 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 customizations to be presented in a predictable way, consistent with the other portlets in the portal.

6.1.1.4 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.

Portal users typically access a portlet's Preview mode by clicking its 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. Portal administrators may access Preview mode from the Portlet Repository.

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 behave as expected when rendered in Preview mode. Hyperlinks can be simulated using the underlined font.
- **Do not use active form buttons.** Forms may not behave as you expect them to when rendered in Preview mode. If you decide to render form elements, do not link them to anything.

6.1.1.5 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.

For example, if a portlet displays expense information, 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, it could link to the entire expense application from Full Screen mode.

Portal users access a portlet's Full Screen mode by clicking the title of the portlet.

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.

6.1.1.6 Help Mode (JPS and Oracle Portal)

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.

Portal users access a portlet's Help mode by clicking **Help** in the portlet header.

Guidelines for Help Mode

The following guidelines 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.

6.1.1.7 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.

Portal users access a portlet's About mode by clicking **About** on the portlet header. 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.

Guidelines for About Mode

The following guidelines 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.

6.1.1.8 Link Mode (PDK-Java)

A portlet uses Link mode to render a link to itself that displays on a mobile page. When the user clicks the link, the portlet is called in Show mode. The portlet then renders itself in the mobile View/Shared Screen mode.

For JPS portlets that declare support of the Oracle Mobile XML content type, Oracle Portal renders the link in one of two ways, as follows:

- Call the portlet's View mode with the MINIMIZED window state, if the portlet declares support for it.
- Otherwise, render a link using the portlet's title.

Guidelines for Link Mode

The following guidelines should govern what you expose to users in Link mode:

- **Limit content.** The purpose of Link mode is to render a link without extraneous material. Link mode should simply render the short title and possibly some relevant summary information (usually just a word or two).

6.1.2 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 the case of 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 the case of 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 portal. If a link or form within a portlet does not refer back to the portal, then following that link takes the user away from the portal, which is not typically the desired behavior.

The portlet developer does not need to know the detailed mechanics of how the parameters of a form or link get passed around between the user, portal, and portlet. However, they must understand that they cannot write links in a portlet the same way they do for typical, simple Web pages.

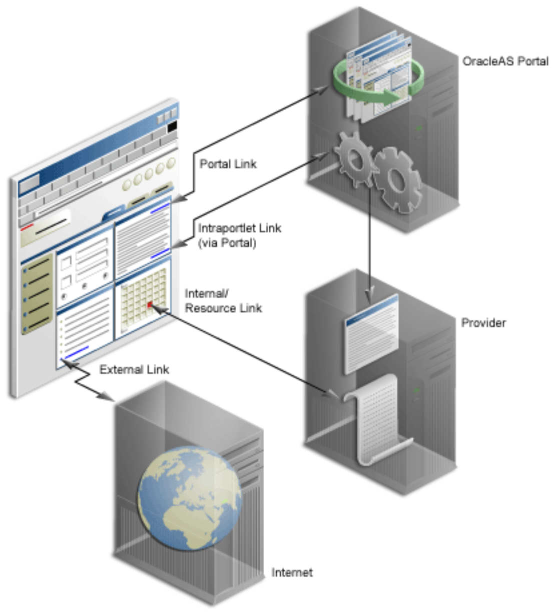
6.1.2.1 Types of Links for Portlets

A portlet may render links of four classes, as follows:

- **Intraportlet links** require the portlet to be aware of the address of Oracle Portal because they actually refer to it in some way.
- **Portal links**, like intraportlet links, must be aware of the address of Oracle Portal for the same reason.
- **External links** make no reference to Oracle Portal and behave in portlets as they would do in a normal Web page.
- **Internal/Resource links**, like external links, also make no reference to Oracle Portal.

Figure 6–1 contains a summary of these link types. The arrows indicate how the links reference the resources to which they logically refer.

Figure 6–1 Oracle Portal Link Types



6.1.2.1.1 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.

6.1.2.1.2 Portal Links Portal links refer to significant pages within Oracle Portal, such as the user's home page.

6.1.2.1.3 External Links External links refer neither to the portlet (through a page) nor to any part of the portal. If selected, these links take the user away from the portal, for example, `www.oracle.com`.

6.1.2.1.4 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.

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>
<%
     }
```

These calls cause the Parallel Page Engine to make the request to the resource and return it to the browser. For session-based providers, any cookies returned from the original `initSession` call to the provider are sent with the request back to the provider to maintain the right session context.

6.1.3 Guidelines for JavaScript

You may find the use of JavaScript useful within a portlet, but bear in mind the following guidelines within your portlets:

- You should never use JavaScript to redirect the page in which the portlet is rendered. If you need to direct users elsewhere, 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 other JavaScript on the page.

6.1.4 Guidelines for Mobile Portlets

Oracle Portal is capable of rendering its pages for both HTML and non-HTML (mobile) devices. When rendering for a mobile device, Oracle Portal requires portlets to generate content in a universal markup language called OracleAS Wireless XML.

Many portlets, known as desktop portlets, generate only HTML responses and as such can only render themselves in standard HTML browsers. Some portlets, known as mobile portlets, generate only OracleAS Wireless XML responses. These portlets can render themselves on any device, including standard HTML browsers. Many portlets, though, take a hybrid approach that renders either HTML or OracleAS Wireless XML depending on the environment. These hybrid portlets can render themselves on any device, but they render best on standard HTML browsers. Although OracleAS Wireless XML is sufficient for HTML responses, it is not as expressive as HTML. Since portlets running in both a desktop and mobile environment are typically accessed using the desktop, developers commonly choose to create hybrid portlets that can provide the best possible rendition in the desktop environment.

When building mobile portlets, you should adhere to the following guidelines:

- Section 6.1.4.1, "Declare Capabilities"
- Section 6.1.4.2, "Declare a Short Title"
- Section 6.1.4.3, "Implement Personalization of the Short Title"
- Section 6.1.4.4, "Implement Link Mode"
- Section 6.1.4.5, "Heed Device Information"
- Section 6.1.4.6, "Tailor Personalization Pages"

For information on how to build mobile-enabled portlets, refer to Section 7.2.10, "Enhancing Portlets for Mobile Devices".

6.1.4.1 Declare Capabilities

To properly manage portlets, Oracle Portal must know the set of content types a portlet generates. Oracle Portal uses this information in the following ways:

- To restrict the Portlet Repository view in the Add Portlet dialog. Only those portlets capable of being rendered on the targeted page will appear in the Add Portlet dialog. For example, when a user invokes the Add Portlet dialog from a mobile design page, only portlets that indicate they can generate OracleAS Wireless XML responses are displayed.

- To display an icon in the Portlet Repository view in the Add Portlet dialog that identifies portlets capable of being rendered on many devices. For example, when a user invokes the Add Portlet dialog from a standard design page, those portlets that are mobile capable are listed with the icon shown in Figure 6–2 to indicate they will also render on mobile devices.

Figure 6–2 Mobile-enabled Icon

- To display only those portlets registered with the capability of generating OracleAS Wireless XML when rendering a standard page on a mobile device.
- To include only those portlets registered with the capability of generating OracleAS Wireless XML when creating a new mobile page based on an existing standard page.

6.1.4.2 Declare a Short Title

The small screen size of the typical mobile device limits the number of characters it can display in a single line without scrolling. Portlet titles, which appear as the menu item label when Oracle Portal renders the mobile page in a menu structure, are often too long for mobile displays. Hence, you can define a short title for your portlet. The short title replaces the standard title where display space is limited.

6.1.4.3 Implement Personalization of the Short Title

The standard portlet title represents the default portlet instance name when rendered in the header of a portlet on a standard page. The portlet's short title represents the default portlet instance name when rendered as a menu item in a mobile page. Just as we recommend that portlets support personalizing the standard title, we also recommend that your portlets support personalizing the short title. This functionality enables the page designer or end user to give the instance a meaningful name.

6.1.4.4 Implement Link Mode

When Oracle Portal renders a standard page to the desktop, it assembles portlets on the page in the tabular layout defined by the page designer. Thus, Oracle Portal aggregates the content of many portlets on a single page. Because of their small displays, mobile devices cannot effectively display the content of multiple portlets on a single page. Instead, the page's portlets appear as links (menu items). Users view portlet content by navigating the menu one portlet at a time. The menu item links typically use the portlet's short name. Since well behaved portlets allow personalization of the short name and the portlet manages its own personalization data, the portlet must participate in rendering the menu item link. To enable this functionality, you can implement the Link mode for portlets. In response to a request to render a portlet in Link mode, a portlet generates a link to itself in the appropriate content type. For example, if the render requests HTML, the portlet returns an anchor tag. If the render requests OracleAS Wireless XML, the portlet returns a `SimpleHref` tag.

6.1.4.5 Heed Device Information

All requests, whether from mobile devices or the desktop, pass general device information. For example, one passed attribute identifies the device class, such as `pcbrowser`, `pdabrowser`, or `microbrowser` (cell phones). A portlet developer may use this attribute to adjust the response's layout or quantity of data.

6.1.4.6 Tailor Personalization Pages

A single portlet instance must maintain a single set of user personalizations spanning all devices, mobile and desktop. Therefore, the same personalization page appears even if the instance is shared between a standard and mobile page, and some fields apply only to one environment, desktop or mobile. In this situation, the portlet should identify these fields that pertain to only one environment. For example, a portlet might display a mobile-only section on its personalization page. Furthermore, because the mobile capability is configurable, a portlet could remove mobile-only references from its personalization page when it detects that the mobile functionality is disabled.

6.2 Introduction to Java Portlet Specification (JPS) and WSRP

Organizations engaged in enterprise portal 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 enterprise portals.

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.

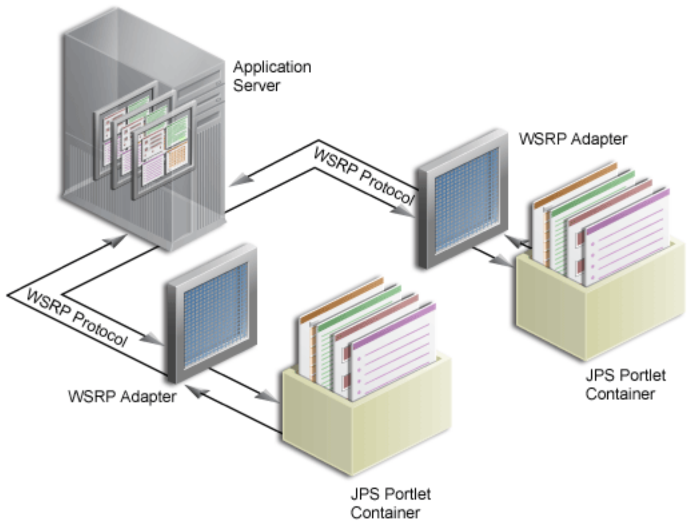
Note: HTML forms can be submitted using either the `get` or `post` method, but the WSRP standard only requires the consumer (portal) to use the `post` method. Support of the `get` method is optional according to the standard. Since portal consumers are not required to support the `get` method, Oracle recommends that you use the `post` method when developing your portlets.

The Relationship Between WSRP and JPS

WSRP is a communication protocol between portal 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 portal pages becomes as simple as selecting portlets from the Oracle Portal repository. Figure 6–3 shows the architecture of the WSRP specification.

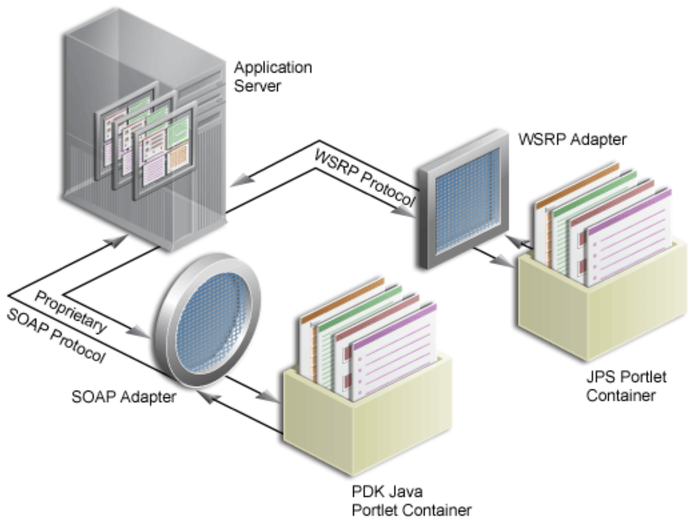
Note: Figure 6–3 illustrates the use of JPS portlets with WSRP, but it should be noted that WSRP can also work with non-JPS portlets.

Figure 6–3 WSRP Specification Architecture



Since Oracle Portal's existing architecture is so similar to the one specified by the WSRP committee, Oracle Portal is able to support communication between our portal and both the new Java Portlet APIs as well as our existing APIs (PDK-Java). Figure 6–4 shows the architecture of the WSRP portal. 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 6–4 Oracle Portal's WSRP Architecture



6.3 Building JPS-Compliant Portlets with Oracle JDeveloper

Using the JSR 168 Portlet Wizard in Oracle JDeveloper you can expose your portlet over WSRP quickly and easily.

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. See Chapter 1, "Understanding Portlets" and Section 6.1, "Guidelines for Writing 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 contains the following subsections:

- Section 6.3.1, "Creating a JSR 168 Portlet"
- Section 6.3.2, "Adding Portlet Logic to Your JSR 168 Portlet"
- Section 6.3.3, "Deploying Your JSR 168 Portlet to the Oracle WebLogic Server"
- Section 6.3.4, "Registering and Viewing Your JSR 168 Portlet"
- Section 6.3.5, "Registering WSRP Producers in Enterprise Configurations"

6.3.1 Creating a JSR 168 Portlet

This section walks you through the JSR 168 Java Portlet Wizard. You can choose which portlet Show 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.

The steps to create a portlet using the JSR 168 Java Portlet Wizard are as follows:

1. Start Oracle JDeveloper.
2. In the Application Navigator, expand the application under which you want to create your portlet.

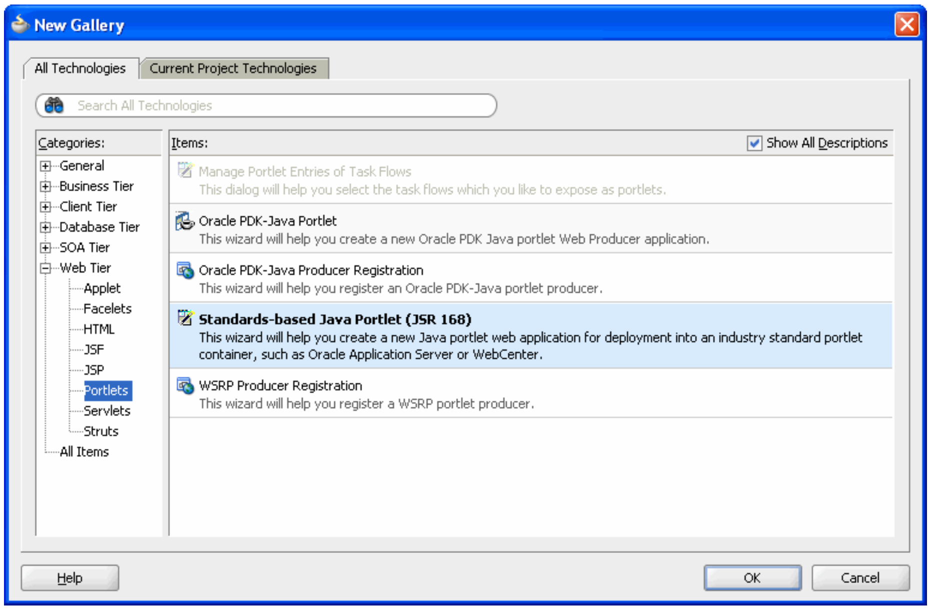
3. Right-click the project under which you want to create your portlet, and select **New.**

> **Note:** If you do not have a project yet, you can create one in one of the following two ways:
>
> - Right-click an existing application in the Application Navigator and choose **New Project**. Select Empty Project and click **OK.** Fill out the Create Project dialog box and click **OK**.
> - Right-click the **Applications** node and choose **New Application**. Fill out the Create Application dialog box and click **OK**. When the Create Project dialog box appears, fill it out and click **OK**.

4. In New Gallery, expand **Web Tier**, select **Portlets** and then **Standards-based Java Portlet (JSR 168)** (see Figure 6–5).

> **Note:** Selecting **Standards-based 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 6–5 New Gallery Dialog Box for Standards-based Java Portlet



Manage Portlet Entries of Task Flows
This dialog will help you select the task flows which you like to expose as portlets.

Oracle PDK-Java Portlet
This wizard will help you create a new Oracle PDK Java portlet Web Producer application.

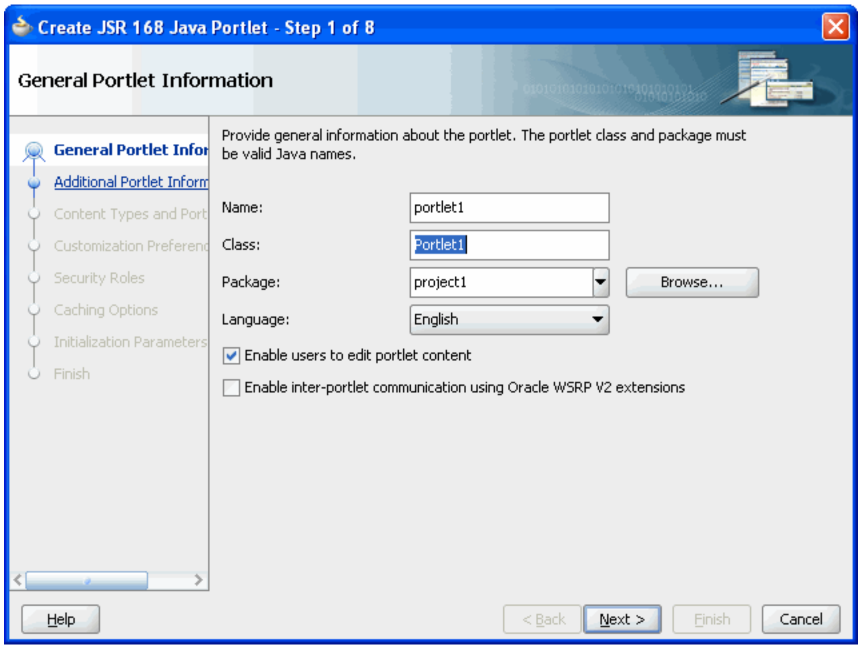
Oracle PDK-Java Producer Registration
This wizard will help you register an Oracle PDK-Java portlet producer.

Standards-based Java Portlet (JSR 168)
This wizard will help you create a new Java portlet web application for deployment into an industry standard portlet container, such as Oracle Application Server or WebCenter.

WSRP Producer Registration
This wizard will help you register a WSRP portlet producer.

5. Click **OK** to display the General Portlet Information page (see Figure 6–6).

Figure 6–6 General Portlet Information Page



Provide general information about the portlet. The portlet class and package must be valid Java names.

Name: portlet1
Class: Portlet1
Package: project1
Language: English
[✓] Enable users to edit portlet content
[ ] Enable inter-portlet communication using Oracle WSRP V2 extensions

6. In the **Name** field, enter a name for the portlet. You can accept the default name or enter your own.

7. 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 enter your own. If you supply your own name, it must be a valid Java name.

8. From 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, the wizard uses the default package of the project.

9. From the **Language** list, select the default language that your portlet will support. The wizard uses English by default.

10. If you want your portlet to support Edit mode, select **Enable users to edit portlet content**. In the wizard, this option is selected by default..

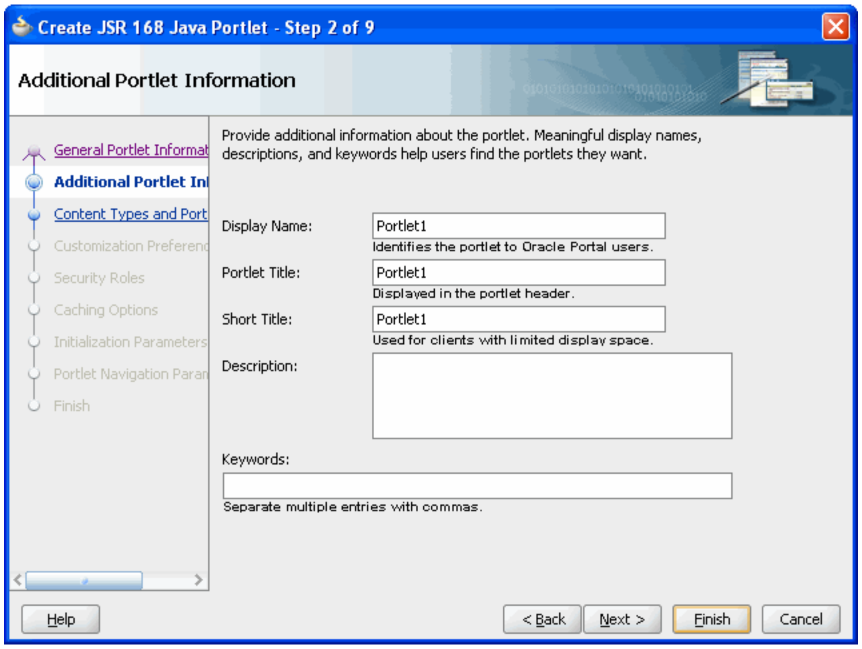
    If you select this option, you can specify the details for the Edit mode later on in the wizard.

11. If you want to create a portlet that supports Oracle WSRP 2.0 extensions, select **Enable inter-portlet communication using Oracle WSRP V2 extensions** and then click **Next**.

> **Note:** JSR 168 portlets built with 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 WebLogic Server.

12. In the Additional Portlet Information page (see Figure 6–7) In the **Display Name** field, enter a name for your portlet. This name will be displayed in the Oracle Portal catalog or repository.

Figure 6–7 The Additional Portlet Information page



Provide additional information about the portlet. Meaningful display names, descriptions, and keywords help users find the portlets they want.

Display Name: Portlet1
Identifies the portlet to Oracle Portal users.

Portlet Title: Portlet1
Displayed in the portlet header.

Short Title: Portlet1
Used for clients with limited display space.

Description:

Keywords:
Separate multiple entries with commas.

13. In the **Portlet Title** field, enter a descriptive title for your portlet.

    The portlet title is displayed in the Resource Palette or Application Resources panel, so make the title something that will help users decide whether the portlet is useful to them. The portlet title is also displayed on the portlet header when the portlet appears on a page.

14. 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.

15. In the **Short Title** field, enter a shorter title for your portlet. This short title will be displayed on the portlet header when the portlet appears on a page on a mobile device.

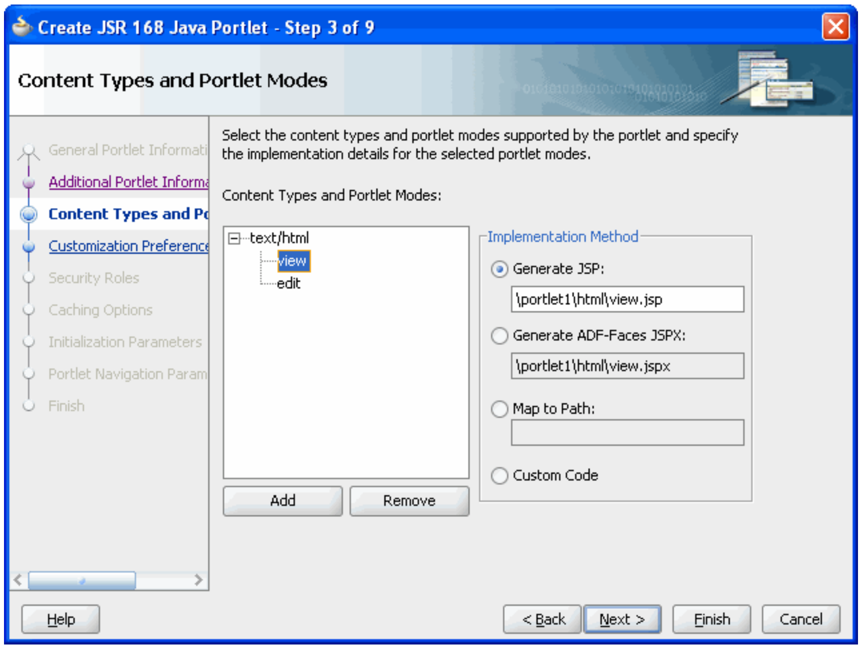
16. In the **Description** field, enter a description of your portlet. This description will be displayed beneath the portlet name in the Portlet Repository.

17. In the **Keywords** field, enter any additional keywords to help users find your portlet in a search.

18. Click **Next** to display the Content Types and Portlet Modes page shown in Figure 6–8.

    Alternatively, you can click **Finish** to create the portlet immediately, using the default values for all remaining settings.

Figure 6–8 Content Types and Portlet Modes Page


Create JSR 168 Java Portlet - Step 3 of 9
Content Types and Portlet Modes
General Portlet Informati
Additional Portlet Informa
Content Types and Po
Customization Preference
Security Roles
Caching Options
Initialization Parameters
Portlet Navigation Param
Finish
Add
Remove
Help
< Back
Next >
Finish
Cancel

Select the content types and portlet modes supported by the portlet and specify the implementation details for the selected portlet modes.

Content Types and Portlet Modes:

- text/html
  - view
  - edit

Implementation Method

(●) Generate JSP: \portlet1\html\view.jsp

( ) Generate ADF-Faces JSPX: \portlet1\html\view.jspx

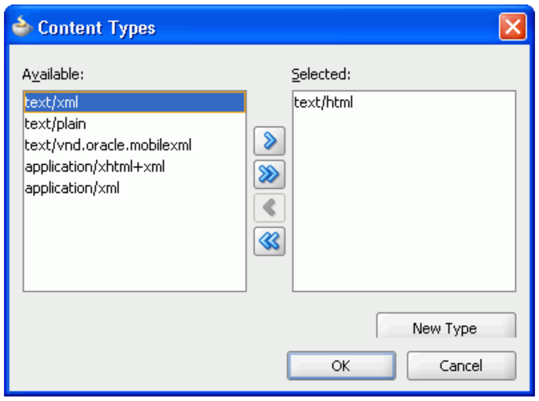
( ) Map to Path:

( ) Custom Code

19. By default, your portlet will display text/html as the content type. If you want to add other content types, select text/html, then click **Add.**

    The list of available content types displays (Figure 6–9). 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 6–9 Content Types Dialog Box



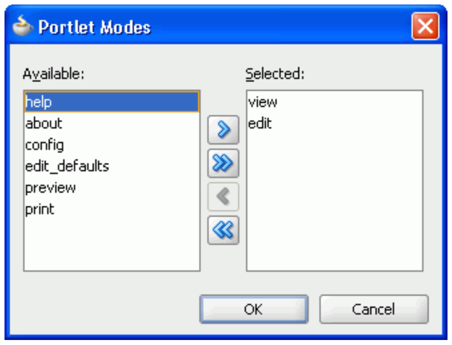
Available:
text/xml
text/plain
text/vnd.oracle.mobilexml
application/xhtml+xml
application/xml

Selected:
text/html

20. By default, your portlet includes view mode. If you selected Editable on the General Portlet Properties page, your portlet also includes Edit mode. If you want to include additional portlet Show modes, select an existing Show mode (for example, view), then click **Add.**

    The list of available portlet Show modes displays (Figure 6–10). You can add Show 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 Show modes, see Section 6.1, "Guidelines for Writing Java Portlets."

Figure 6–10 Portlet Modes Dialog Box



21. Once you have added all of the desired portlet Show modes, choose a function to be performed for each mode. For each portlet mode, click the portlet mode and select an 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 defaults.

  When you complete the wizard, the generated JSP displays in the Application 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 **Generate ADF-Faces JSPX** to generate `.jspx` files. The `.jspx` files will generate exactly the same markup as JSPs.

- Select **Map to Path** if you want to map the portlet to 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.

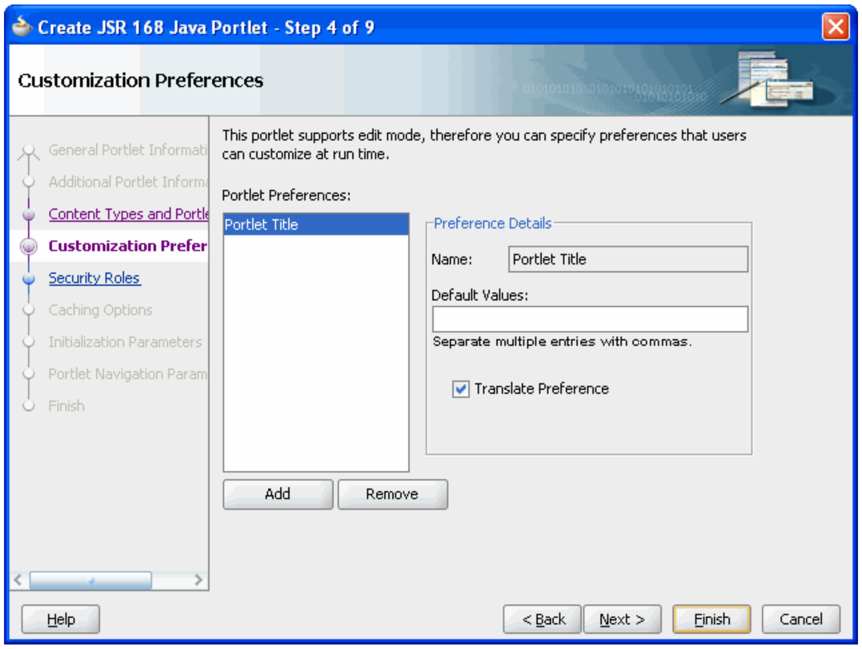
- Select **Custom Code** if you want to implement the portlet mode through a custom coded object. You will create this object later. This selection generates a skeleton method to render content (`private void do<MODE_NAME><CONTENT_TYPE>`) in the generated portlet Java class. You must update this code to render useful content.

22. Click **Next**.

If you selected Editable on the General Portlet Properties page earlier in the wizard, then the Customization Preferences page displays(Figure 6–11). Go to step 23.

If you did not select this option, then the Security Roles page displays (Figure 6–13). Go to step 31.

Figure 6–11 Customization Preferences Page


Create JSR 168 Java Portlet - Step 4 of 9
Customization Preferences
General Portlet Informati
Additional Portlet Informa
Content Types and Portle
Customization Prefer
Security Roles
Caching Options
Initialization Parameters
Portlet Navigation Param
Finish
Add
Remove
Help
< Back
Next >
Finish
Cancel

This portlet supports edit mode, therefore you can specify preferences that users can customize at run time.

Portlet Preferences: Portlet Title

Preference Details

Name: Portlet Title

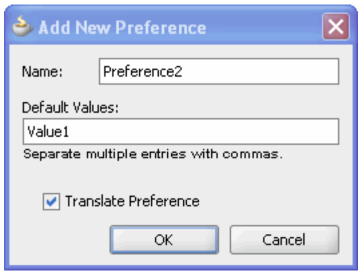
Default Values:

Separate multiple entries with commas.

☑ Translate Preference

23. If you want to include additional customization preferences, click Add. The Add New Preferences dialog box displays.

Figure 6–12 Add New Preference Dialog Box



Name: Preference2

Default Values: Value1

Separate multiple entries with commas.

☑ Translate Preference

24. In the **Name** field, enter a name for the new preference. The name must be unique in the portlet. Use only letters, numbers, and the underscore character.

25. In the **Default Value(s)** field, enter one or more default values for the new customization preference. Separate multiple values with commas.

26. Select the **Translate Preference** check box if you want the customization preference value to be translated. If you select this option, 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.

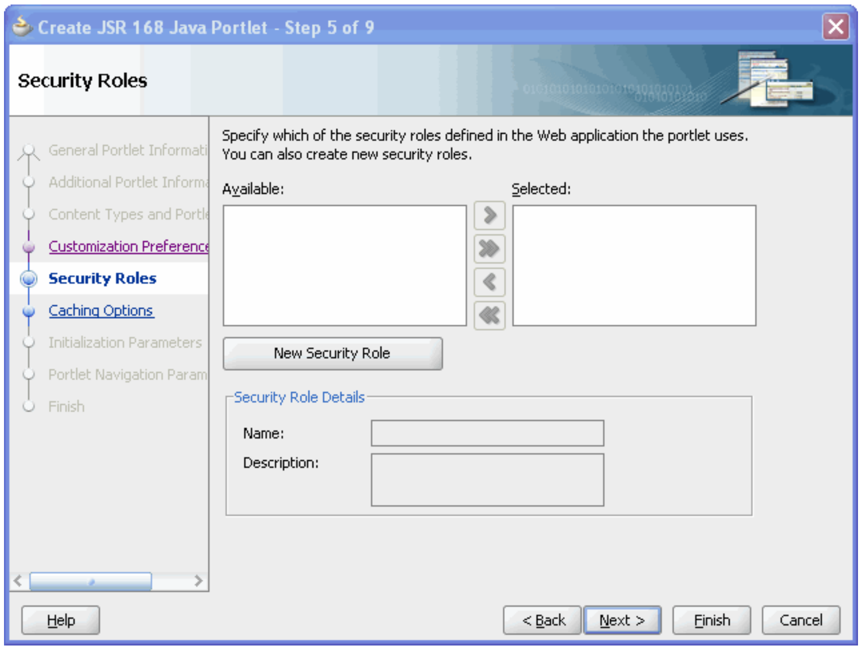
Note: The Name is always translated, but there is not always a need to translate the Default Value. For example, if the value is an integer, no translation is needed

---

27. Click **OK.** Repeat the preceding steps if you want to add more customization preferences.

28. To edit details for existing customization preferences, select the preference in the Portlet Preferences list and edit the fields in the Preference Details section.
29. To delete an existing customization preference, select the preference in the Portlet Preferences list and click Remove.
30. Click **Next** to display the Security Roles page (Figure 6–13).

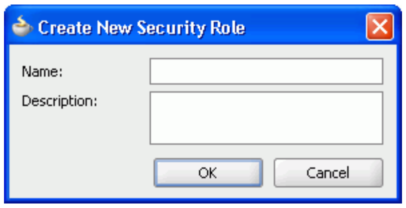
Figure 6–13 Security Roles Page



Specify which of the security roles defined in the Web application the portlet uses. You can also create new security roles.

31. 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 to the security role in the application's portlet deployment file (`portlet.xml`) that refers to the security role in `web.xml`.
32. If you want to define a new security role, click **New Security Role**. The Create New Security Role dialog box displays (Figure 6–14).

Figure 6–14 Create New Security Role Dialog Box

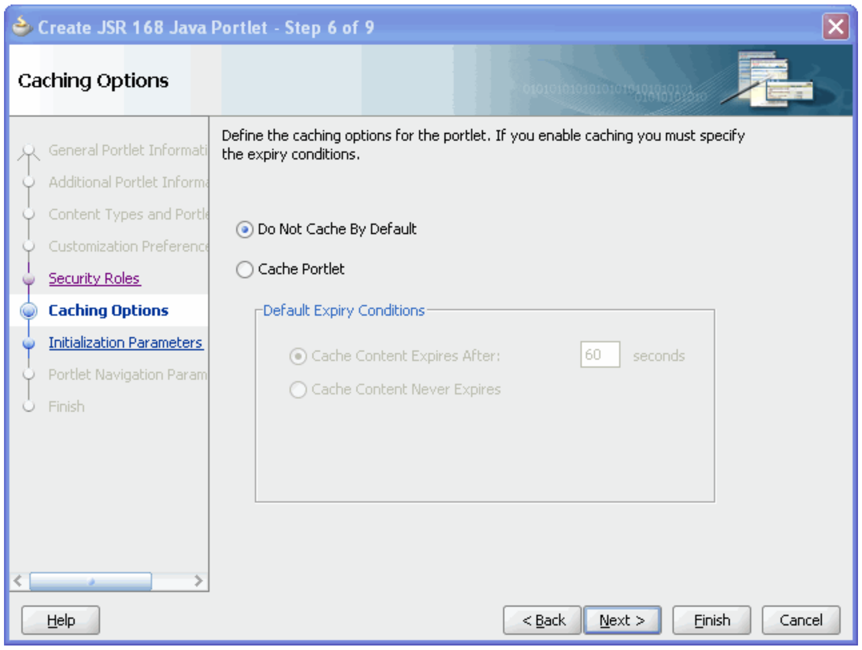


33. In the **Name** field, enter a unique name for the security role.
34. In the **Description** field, enter a description for the security role, explaining the access privileges and restrictions this role will have on the portlet.
35. Click **OK.**

The new security role is added to the **Available** list. You can also manually create security roles.

36. Click **Next** to display the Caching Option page (Figure 6–15).

Figure 6–15 Caching Option Page



Define the caching options for the portlet. If you enable caching you must specify the expiry conditions.

(●) Do Not Cache By Default
( ) Cache Portlet

Default Expiry Conditions
(●) Cache Content Expires After: 60 seconds
( ) Cache Content Never Expires

37. Select **Cache Portlet** if you want to enable caching for your 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.

38. 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.

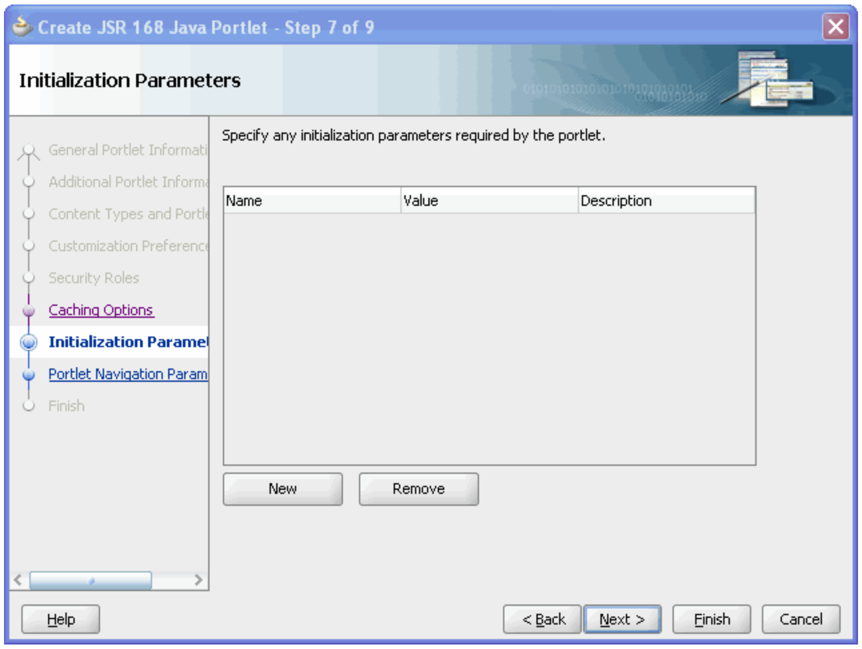
Note: If you do not want any default caching for this portlet, choose **Do Not Cache By Default**. In this case, the wizard actually sets a cache duration of 0 seconds. As stated earlier, this cache setting only comes into play when the portlet itself does not specify a caching condition for a response.

If you choose no caching here and you later decide that you want default caching for the portlet, 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.

---

39. Click **Next** to display the Initialization Parameters page (Figure 6–16).

Figure 6–16 Initialization Parameters Page



Specify any initialization parameters required by the portlet.

40. 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, click **New.** This adds a new row to the table of parameters. You can then double click the row to edit the details.

41. In the **Name** field, enter a unique name for the initialization parameter. Use only letters, numbers, and the underscore character.
42. In the **Value** field, enter a default value for the parameter.
43. In the **Description** field, enter a description for the parameter.
44. To delete an initialization parameter, select it in the table and click **Remove.**
45. Click **Next** to display the Finish page.

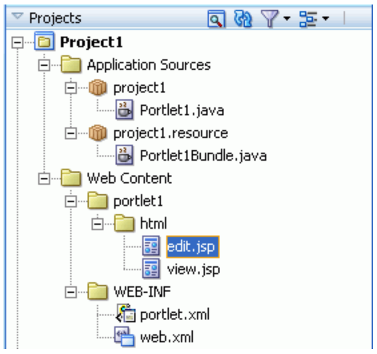
Note: If your Portal is WSRP2 enabled, then the Portal Navigation Parameters page appears. Click **Next** to display the Finish page.

46. 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 6–17):
    - If you selected **Generate JSP** for the portlet modes, generated code for each mode

      If you selected **Custom Code** instead, 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 6–17 Application Navigator



6.3.2 Adding Portlet Logic to Your JSR 168 Portlet

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 on adding functionality and features.

6.3.3 Deploying Your JSR 168 Portlet to the Oracle WebLogic Server

After you finish the wizard and successfully generate your portlet, you are ready to deploy it to the Oracle WebLogic Server. To create and deploy a WAR file, perform the following steps:

1. In the Application Navigator, right-click the project that contains your portlet and select **New.**
2. In the New Gallery, expand **General** and select **Deployment Profiles.**
3. In the Items list, select **WAR File** and click **OK.**

   The **Create Deployment Profile -- WAR File** dialog box opens.
4. In the **Deployment Profile Name** field, enter a meaningful name for the deployment profile (for example, `webapp`).
5. Click **OK**.

   The **Edit WAR Deployment Profile Properties** dialog box opens.
6. Under **Web Application's Context Root**, select **Specify J2EE Web Context Root** and enter the context root in the corresponding field, for example `my-portlet`.
7. Click **OK**.

   The **Project Properties** dialog box opens.
8. Click **OK.**

9. In the Application Navigator, right-click your project and select **Deploy,** then select the deployment profile, next select **to,** and finally select the **IntegratedWLSConnection**.
10. In the **Select deployment type** dialog box, select Yes and click **OK**.
11. When the Deployment Finished message displays in the **Deployment Log** at the bottom of Oracle JDeveloper, verify that no errors occurred.
12. Construct the WSDL URL for your JPS-compliant portlet as follows:

    ```
    http://host:port/context-root/portlets/wsrp2?WSDL
    ```

    where *host* is the server to which your producer has been deployed.

    port is the Oracle 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.
13. In a Web browser, enter the WSDL URL from the previous step to ensure that it is working. If the WSDL definition does not appear in the browser, then the deployment of your WAR file must have failed. Refer to Appendix B.2, "Diagnosing Java Portlet Problems".
14. If you are using SSL (HTTPS), you must modify your WSDL URL before registering the producer with Oracle Portal. If you are not using SSL, you may skip to the next step now.

    Note: In order for HTTPS to work with a producer, you must have previously configured the server certificate in the infrastructure server certificate store as described in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

    To modify your WSDL URL, perform the following steps:

    a. In a Web browser, enter the HTTPS WSDL URL. For example:

       ```
       http://host:port/context-root/portlets/WSRPBaseService?WSDL
       ```

       Each port in the definition should be displayed with an HTTPS location, for example:

       ```
       <wsdl:port binding="bind:WSRP_v1_Markup_Binding_SOAP"
       name="WSRPBaseService">
       <soap:address
       location="https://host:port/context-root/portlets/WSRPBaseService"/>
       </wsdl:port>
       ```

    b. If the ports are not listed with HTTPS locations, you must change them manually before proceeding. You can do this by saving the XML to a file from the browser and opening it in a text editor.

    c. Save a copy of the WSDL definition to a file on your application server in a location where it can be accessed externally over HTTP. For example, the htdocs directory of your Apache installation. When you register the producer in Oracle Portal, use the location of this WSDL for your **WSDL URL** on the Define Connection page of the registration.

15. You should now register your producer and view your portlet. Refer to Section 6.3.4, "Registering and Viewing Your JSR 168 Portlet".

    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, 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 portal 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.

6.3.4 Registering and Viewing Your JSR 168 Portlet

After you've created and deployed the provider and its portlets, you should register the provider with Oracle Portal. Registering your provider gives Oracle Portal the information it needs to locate and communicate with that provider. After you register a provider, the provider and its portlets become available in the Portlet Repository. They are also listed in the Oracle Portal Navigator.

Note: When you build portlets and providers with built-in tools, such as the Portlet Builder, Oracle Portal automatically registers the provider for you. Once you've created your portlet, it automatically displays in the Portlet Repository. Oracle Portal also offers built-in portlets that are contained in a preconfigured provider. For example, OmniPortlet and Web Clipping are portlets that you can use out of the box, and are already registered with Oracle Portal. You can view these portlets in the Add Portlets list. However, if you build the portlets and providers programmatically, you must then register these providers to make them available to the portal user.

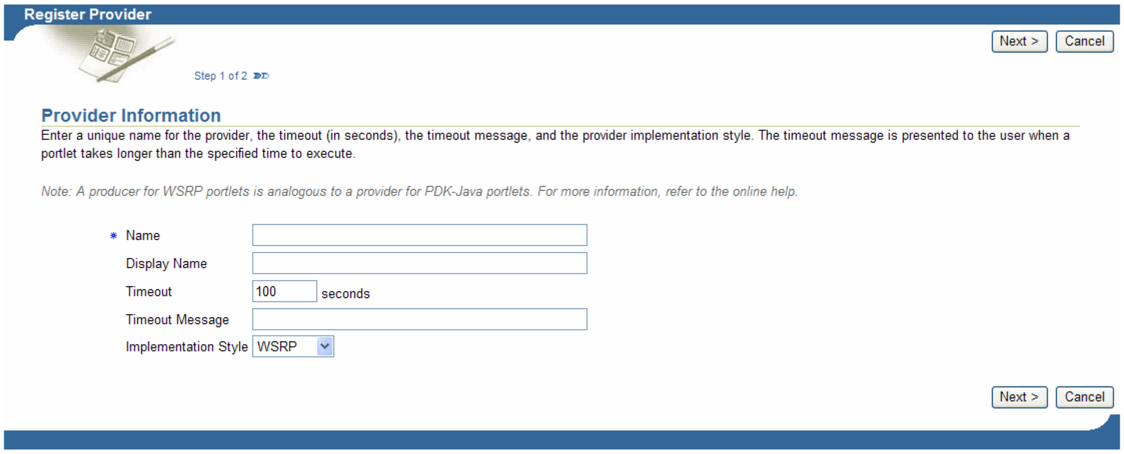
To register providers for your standards-based portlets:

1. Open Oracle Portal and log in. Note that to register your provider, you need to have Manage or Edit privileges on providers. If you do not have these privileges, you need to request them from your administrator.

   Note: If you do not want to install Oracle Portal but you still need to test your portlets, you may want to use the Oracle Portal Verification Service on OTN to validate your WSRP producers: `http://portalstandards.oracle.com/portal/page/portal/OracleHostedWSRPPortal/Welcome`

2. If you are not already on the Portal Builder page, click **Builder** in the upper right corner.
3. Click the **Administer** tab.
4. Click the **Portlets** subtab.
5. In the Remote Providers portlet, click **Register a Provider** to display the Register Provider page (Figure 6–18).

Figure 6–18 Register Provider Page



Provider Information

Enter a unique name for the provider, the timeout (in seconds), the timeout message, and the provider implementation style. The timeout message is presented to the user when a portlet takes longer than the specified time to execute.

Note: A producer for WSRP portlets is analogous to a provider for PDK-Java portlets. For more information, refer to the online help.

* Name	
Display Name	
Timeout	100 seconds
Timeout Message	
Implementation Style	WSRP

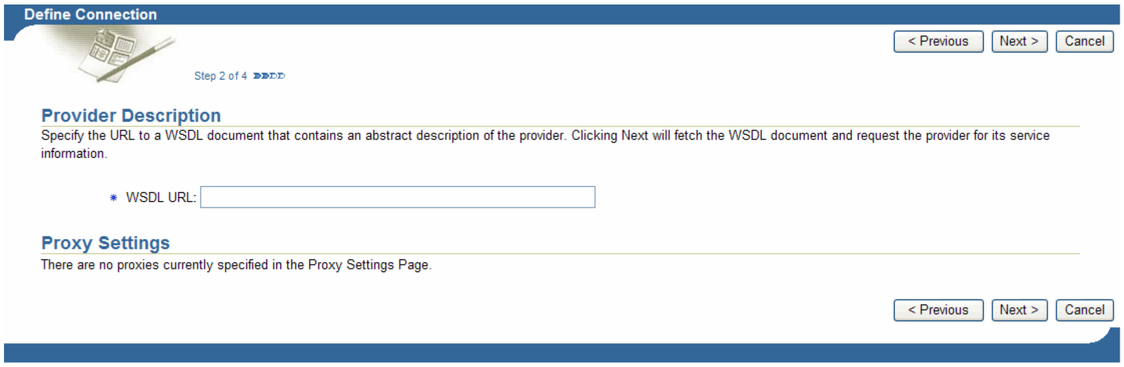
6. In the **Name** field, enter the name of the provider. This name must not be more than 200 characters or contain spaces or other special characters.
7. In the **Display Name** field, enter a name to display for the provider when it is referenced, for example in the Portlet Repository. The display name must not be more than 200 characters.
8. In the **Timeout** field, enter the number of seconds Oracle Portal should try to connect to the provider before displaying the timeout message.
9. In the **Timeout Message** field, enter the message to display when Oracle Portal cannot establish contact with the provider within the number of seconds specified in the Timeout field. The message displays within the body of the portlet. The message may not contain HTML or mobileXML.
10. From the **Implementation Style** list, select WSRP for your WSRP provider.

Note: If a provider is unavailable for some reason, it can slow down the rendering of pages that contain portlets from that provider. By default, Oracle Portal waits until all portlets are returned before completing the page assembly. To avoid delays, you can set a time limit using the Page Assembly Timeout on the Main tab of the page properties. If Oracle Portal cannot retrieve the portlet from the provider within the specified timeout period, it will render the page without the portlet. If, at a later time, the provider becomes available, Oracle Portal refreshes the page, adding the missing portlets. In this way, page rendering is never halted due to the unavailability of a particular provider. For more information about Page Assembly Timeout, see the *Oracle Fusion Middleware User's Guide for Oracle Portal*.

11. Click **Next** to display the Define Connection page (Figure 6–19).
12. In the WSDL URL field, enter the WSDL URL for your provider. For example:

```
http://myserver.com:8888/my-portlet/portlets/WSRPBaseService?WSDL
```

Figure 6–19 Define Connection Page



Provider Description

Specify the URL to a WSDL document that contains an abstract description of the provider. Clicking Next will fetch the WSDL document and request the provider for its service information.

* WSDL URL:

Proxy Settings

There are no proxies currently specified in the Proxy Settings Page.

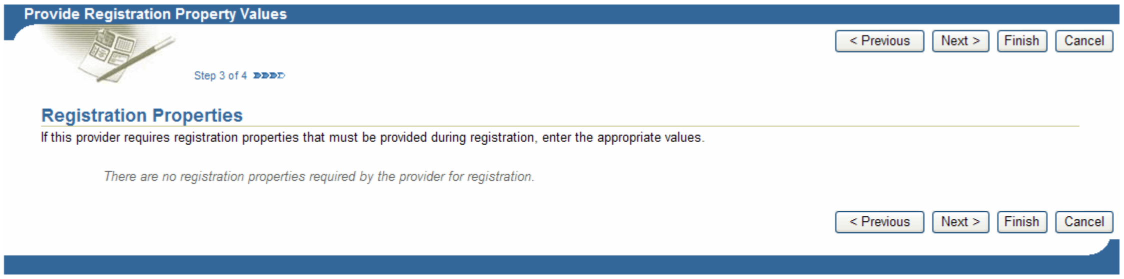
Note: A WSRP producer can contain portlets that work within the WLDL v1 or v2 framework. WLDL v1 portlets work in pre-11gR1 Oracle Portal releases. WSDL v2 portlets work in Oracle Portal 11gR1. When a WSRP producer containing v2 portlets is registered with its v1 URL, the portlet repository displays the v1 registration name and only the v1 portlets.

So, when a WSRP producer containing only v2 portlets is registered with its v1 URL, the portlet repository displays only the registration name; it does not display any portlet.

---

13. Click **Next** to display the Portal Registration Property Values page (Figure 6–20).
14. Provide any registration properties required by the provider. If there are none, you can proceed to the next step.

Figure 6–20 Provider Registration Property Values Page



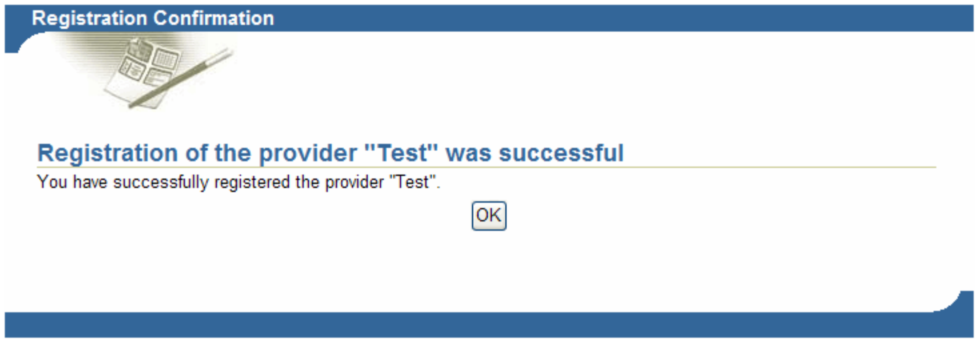
Registration Properties

If this provider requires registration properties that must be provided during registration, enter the appropriate values.

There are no registration properties required by the provider for registration.

15. Click **Finish**. You should see a Registration Confirmation page similar to the one in Figure 6–21.

Figure 6–21 Registration Confirmation Page



Registration of the provider "Test" was successful

You have successfully registered the provider "Test".

16. Your portlet should now be available for adding to pages just as any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in the *Oracle Fusion Middleware User's Guide for Oracle Portal*.

6.3.5 Registering WSRP Producers in Enterprise Configurations

When you register a WSRP producer in an Enterprise configuration, you must create a Web Seringy!!vices Definition Language (WSDL) document manually, then register the WSRP producer using that WSDL. This is because the dynamically generated WSDL creates URLs using the HTTPS protocol and the HTTPS port while WSRP producers use HTTP.

Note: A *producer* for WSRP portlets is analogous to a *provider* for PDK-Java portlets.

To create a WSDL document manually, perform the following tasks:

1. View the dynamically generated WSDL through your browser (preferably Internet Explorer).

   To view the WSDL for our WSRP samples, go to:

   ```
   http://host:external http port/portletapp/portlets?WSDL
   ```

2. Save the file from the browser to any externally available location.

   For our WSRP Samples, save the file from the browser into the following directory:

   `ORACLE_HOME/j2ee/home/applications/portletapp/wsrp-samples`

   Save the file as `wsrpsamples.wsdl`.

3. Edit the file, replacing `https` with `http` and correcting the ports to be the external http ports.

4. View the file through a browser.

   For example, for our WSRP Samples use the following URL:

   ```
   http://host:external http port/portletapp/wsrpsamples.wsdl
   ```

5. Use the URL to your `*.wsdl` file (such as the URL under Step 4) when you register the WSRP producer.

For more information about Enterprise configurations, see the *Oracle Fusion Middleware Enterprise Architecture and Deployment Guide*.

6.4 Introduction to Oracle PDK-Java

Oracle PDK-Java gives you a framework to simplify the development of Java portlets by providing commonly required utilities and allowing you to leverage existing development skills and application components such as JSPs, servlets, and static HTML pages. Oracle PDK-Java also enables you to create portlets without having to deal directly with the complexity of communications between Oracle Portal and providers.

The Oracle PDK-Java framework is divided into the following areas:

- The **Provider Adapter** insulates the developer from the HTTP syntax defined by Oracle Portal for communication with Web providers. It translates the information passed between Oracle Portal and your Java Web provider. Without an adapter, your provider would not only manage portlets, but it would also have to communicate this information directly to Oracle Portal in the expected language. The adapter eliminates the need for your Web provider to understand the portal language and vice-versa.
- The **Provider Interface** defines the APIs (functions) required by your Java implementation to integrate with the Provider Adapter. The Provider Adapter receives messages from the portal, translates them into calls to the Provider Interface, and translates the provider's response into a format that the portal can understand. The Provider Interface contains a set of Java classes that define the methods your provider 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 **Provider Runtime** provides a base implementation that follows the specification of the Provider Interface. The Provider Runtime includes a set of default classes that implement each one of the Provider 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 provider by implementing common functions for Oracle Portal requests and providing a declarative mechanism for configuring the provider. Using the Provider Runtime, you can focus your development efforts on the portlets themselves rather than the infrastructure needed to communicate with the portal. If the standard behavior of the Provider Runtime does not meet your requirements, 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 **Provider 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 portal page, and supporting portlet caching.

6.5 Building Oracle PDK-Java Portlets with Oracle JDeveloper

Using the Oracle PDK-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 Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 6.1, "Guidelines for Writing 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 contains the following sections:

- Section 6.5.1, "Creating an Oracle PDK-Java Portlet and Provider"
- Section 6.5.2, "Adding Portlet Logic to Your Oracle PDK-Java Portlet"
- Section 6.5.3, "Validating Your Oracle PDK-Java Portlet and Provider"
- Section 6.5.4, "Deploying Your Oracle PDK-Java Portlet to an Application Server"
- Section 6.5.5, "Registering and Viewing Your Oracle PDK-Java Portlet"

6.5.1 Creating an Oracle PDK-Java Portlet and Provider

This section walks you through the Oracle PDK-Java Portlet Wizard. You can choose which portlet Show 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.

The steps to create a portlet and provider are as follows:

1. Start Oracle JDeveloper.
2. In the Application 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 a project yet, you can create one in one of the following ways:

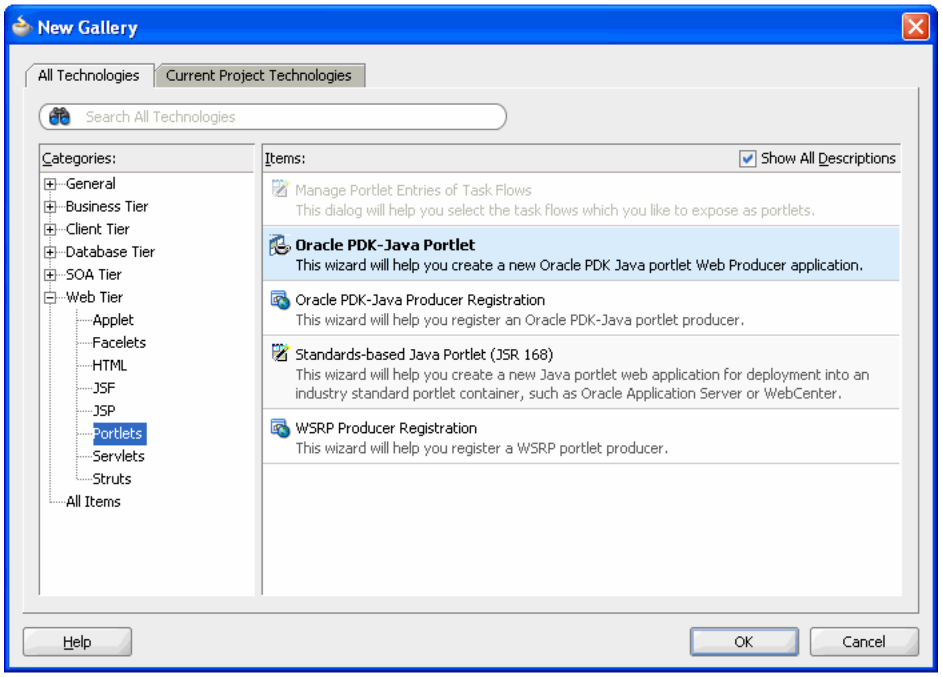
- Right-click an existing application in the Application Navigator and choose **New Project.** Select Empty Project and click **OK.** Fill out the Create Project dialog box and click **OK.**
- Right-click the Applications node and choose **New Application.** Fill out the Create Application dialog box and click **OK.** When the Create Project dialog box appears, fill it out and click **OK.**

---

4. In the New Gallery, expand the Web Tier category and select **Portlets.**
5. In the Items list, select **Oracle PDK-Java Portlet** (Figure 6–22).

Note: Selecting **Standards-based 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 6–22 New Gallery Dialog Box for Oracle PDK-Java Portlet



Manage Portlet Entries of Task Flows
This dialog will help you select the task flows which you like to expose as portlets.

Oracle PDK-Java Portlet
This wizard will help you create a new Oracle PDK Java portlet Web Producer application.

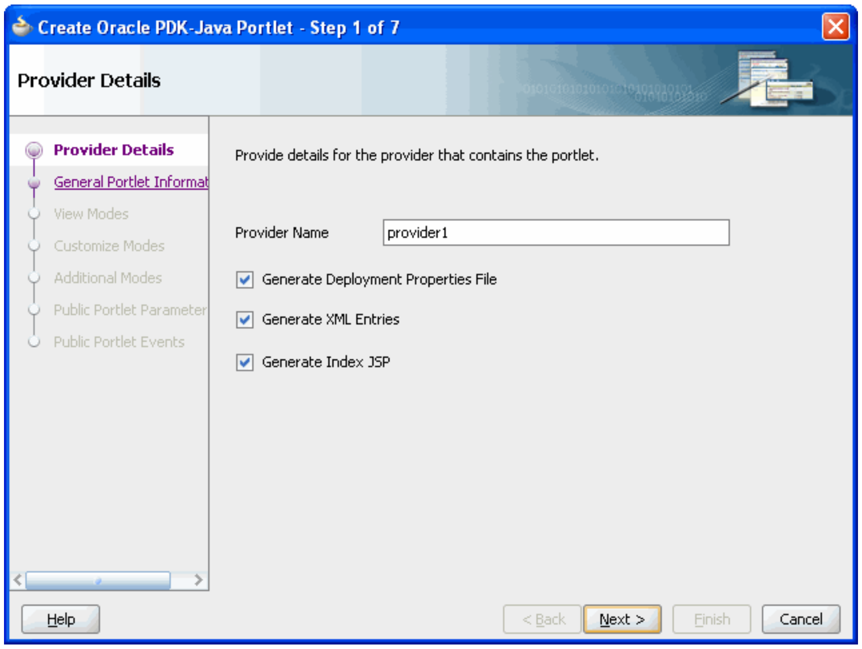
Oracle PDK-Java Producer Registration
This wizard will help you register an Oracle PDK-Java portlet producer.

Standards-based Java Portlet (JSR 168)
This wizard will help you create a new Java portlet web application for deployment into an industry standard portlet container, such as Oracle Application Server or WebCenter.

WSRP Producer Registration
This wizard will help you register a WSRP portlet producer.

6. Click **OK** to display the Provider Details Page (Figure 6–23).

Figure 6–23 Provider Details Page



Provide details for the provider that contains the portlet.

Provider Name: provider1

- [✓] Generate Deployment Properties File
- [✓] Generate XML Entries
- [✓] Generate Index JSP

7. In the Provider Details page, enter a name for the new producer that will contain your portlet. This name must be unique within the project. In the PDK-Java, the term provider is used instead of producer. A provider is exactly the same thing as a producer.

8. Select **Generate Deployment Properties File**.

   This automatically generates 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 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.

9. Select **Generate XML Entries**.

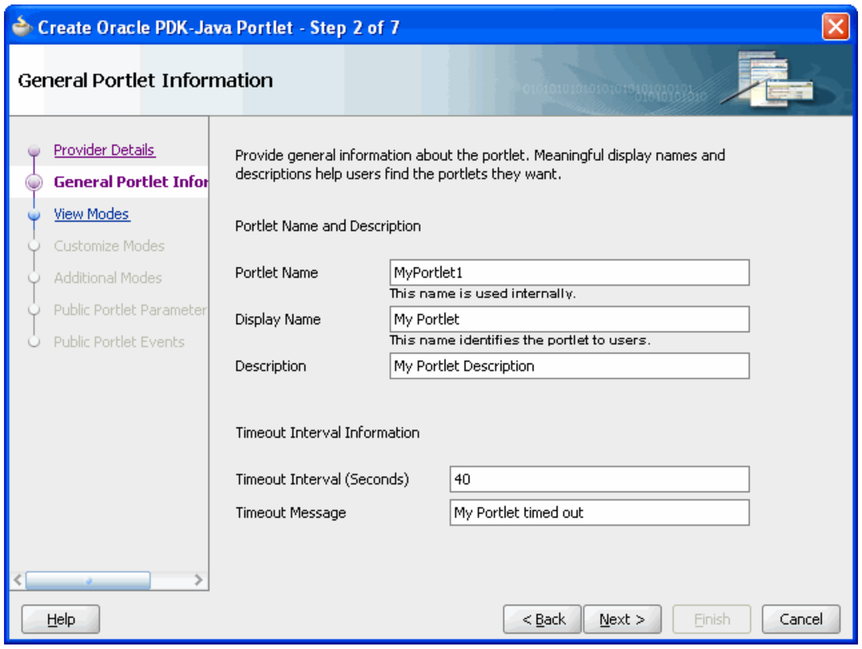
   This automatically generates a producer definition file (`provider.xml`) for the producer that contains details of the portlets belonging to the producer, including those generated by the wizard.

10. Select **Generate Index JSP**.

    This automatically generates an `index.jsp` file that lists all the producers that reside in the application with hyperlinks that enable easy access to producer test pages.

11. Click **Next** to display the General Portlet Information Page.

Figure 6–24 General Portlet Information Page



Provide general information about the portlet. Meaningful display names and descriptions help users find the portlets they want.

Portlet Name and Description

Portlet Name: MyPortlet1
This name is used internally.

Display Name: My Portlet
This name identifies the portlet to users.

Description: My Portlet Description

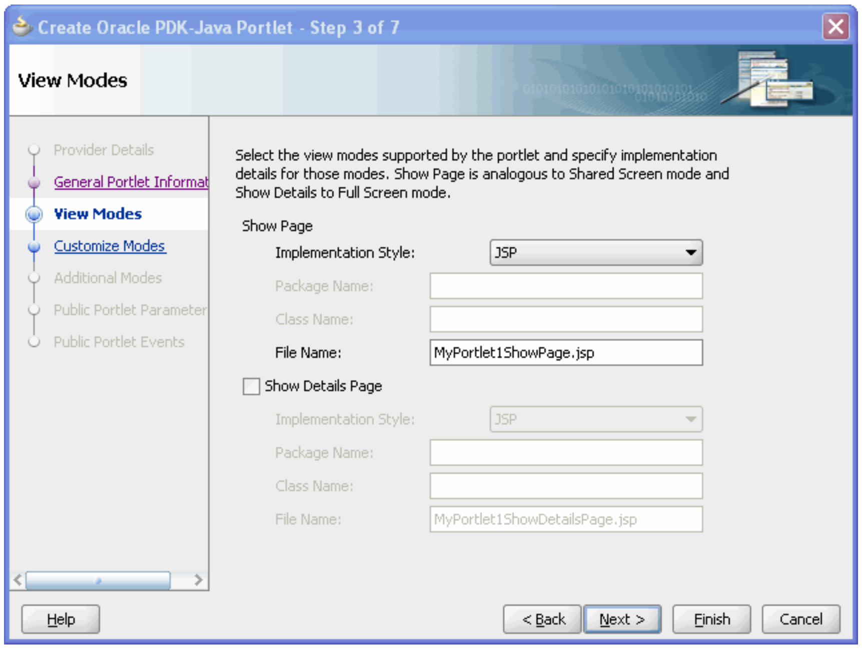
Timeout Interval Information

Timeout Interval (Seconds): 40

Timeout Message: My Portlet timed out

12. In the **Portlet Name** field, enter a meaningful name for your portlet. This name is used internally and is not exposed to users. The display name is displayed to users in portlet selection lists, such as the Component Palette.
13. In the **Display Name** field, enter a display name for your portlet. This name will be displayed in portlet selection lists, such as the Portlet Repository, where users choose which portlets to add to a page.
14. In the **Description** field, enter a description for your portlet. This description will be displayed beneath the portlet name in the Portlet Repository.
15. In the **Timeout** field, enter the number of seconds to allow for rendering the portlet.
16. In the **Timeout Message** field, enter a message to display if the rendering of the portlet exceeds the timeout value specified.
17. Click **Next** to display the View Modes page (Figure 6–25).

Figure 6–25 View Modes Page


Create Oracle PDK-Java Portlet - Step 3 of 7

View Modes

Select the view modes supported by the portlet and specify implementation details for those modes. Show Page is analogous to Shared Screen mode and Show Details to Full Screen mode.

Show Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1ShowPage.jsp

Show Details Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1ShowDetailsPage.jsp

18. In the Show page section, 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 provider 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, 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.

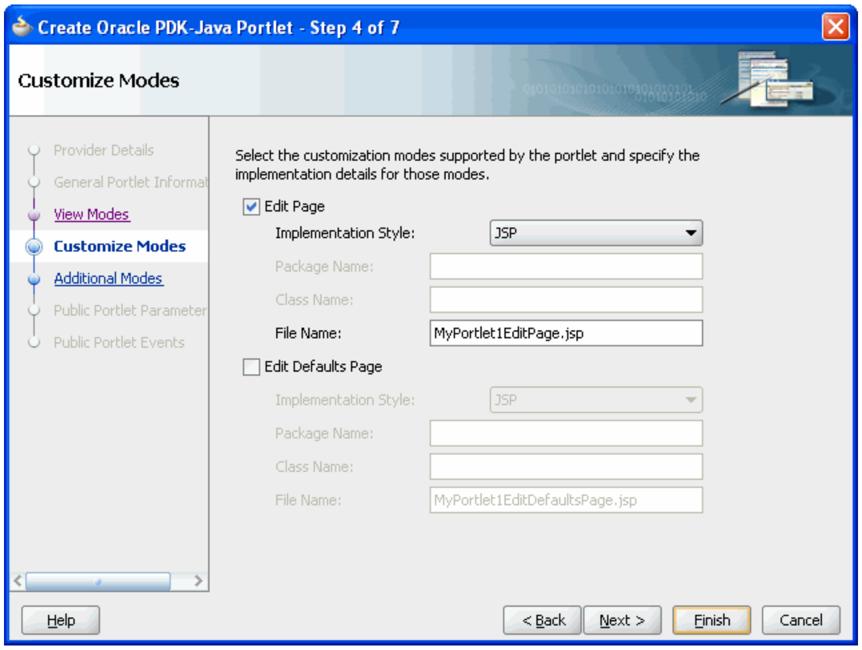
For more information about Shared Screen mode, see Section 6.1.1.1, "Shared Screen Mode (View Mode for JPS)."

19. If you want to implement Full Screen mode for your portlet, select **Show details page,** then select an **Implementation style** as described earlier for Show page.

    For more information about Full Screen mode, see Section 6.1.1.5, "Full Screen Mode (PDK-Java)."

20. Click **Next** to display the Customize Modes page (Figure 6–26)

Figure 6–26 Customize Modes Page



Select the customization modes supported by the portlet and specify the implementation details for those modes.

☑ Edit Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1EditPage.jsp

☐ Edit Defaults Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1EditDefaultsPage.jsp

21. **Edit page** is selected by default. If you want to implement Edit mode for your portlet, select an **Implementation style** as described earlier for Show page. If you do not want to implement Edit mode, clear the **Edit page** check box.

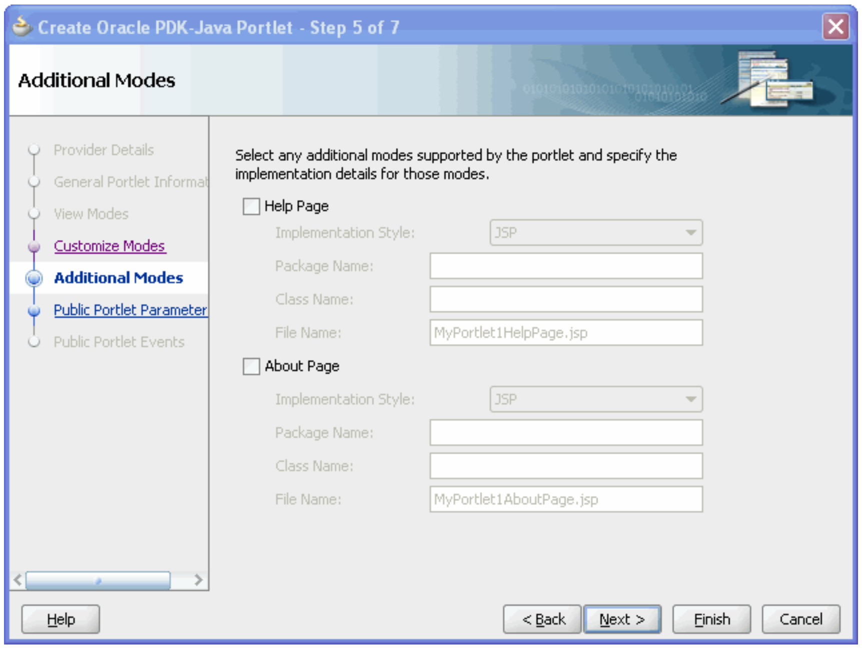
    For more information about Edit mode, see Section 6.1.1.2, "Edit Mode (JPS and Pdk-Java)."

22. 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 6.1.1.3, "Edit Defaults Mode (JPS and PDK-Java)."

23. Click **Next** to display the Additional Modes page (Figure 6–27).

Figure 6–27 Additional Modes Page


Create Oracle PDK-Java Portlet - Step 5 of 7

Additional Modes

Select any additional modes supported by the portlet and specify the implementation details for those modes.

Help Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1HelpPage.jsp

About Page
Implementation Style: JSP
Package Name:
Class Name:
File Name: MyPortlet1AboutPage.jsp

24. If you want to implement Help mode for your portlet, select **Help page,** then select an **Implementation style** as described earlier for Show page.

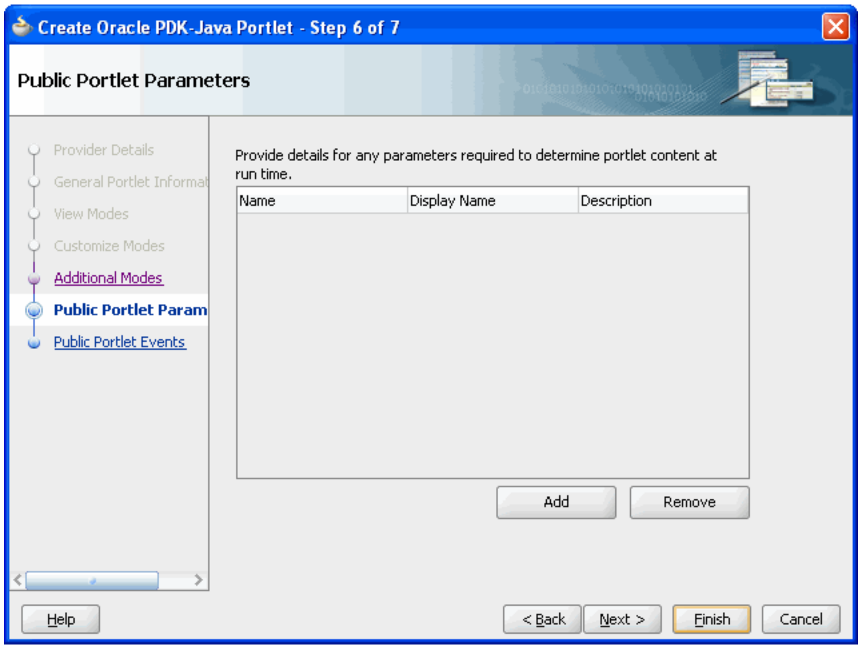
For more information about Help mode, see Section 6.1.1.6, "Help Mode (JPS and Oracle Portal)."

25. 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 6.1.1.7, "About Mode (JPS and PDK-Java)."

26. Click **Next** to display the Public Portlet Parameters page (Figure 6–28)

Figure 6–28 Public Portlet Parameters Page



Provide details for any parameters required to determine portlet content at run time.

Name	Display Name	Description

27. If you want to add public parameters to your portlet, click **Add** to create a blank row. For more information about using parameters, see Section 7.2.3, "Passing Parameters and Submitting Events."

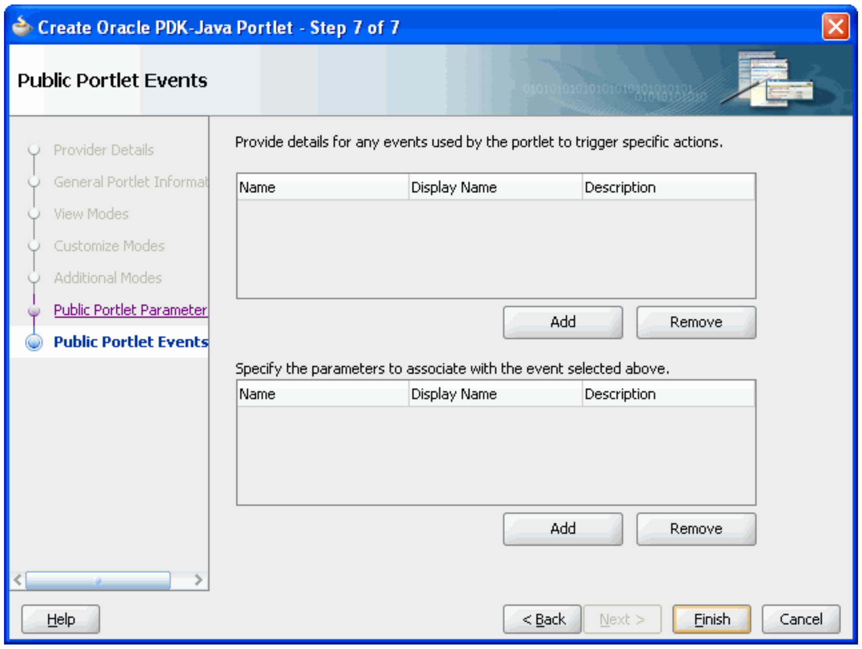
28. In the **Name** field, enter an internal name for the parameter, for example, `MyParam`.

29. In the **Display Name** field, enter a name to display to users, for example, `My Portlet Parameter`.

30. In the **Description** field, enter descriptive information about the parameter.

31. Click **Next** to display the Public Portlet Events page (Figure 6–29).

Figure 6–29 Public Portlet Events Page



Provide details for any events used by the portlet to trigger specific actions.

Name	Display Name	Description

Specify the parameters to associate with the event selected above.

Name	Display Name	Description

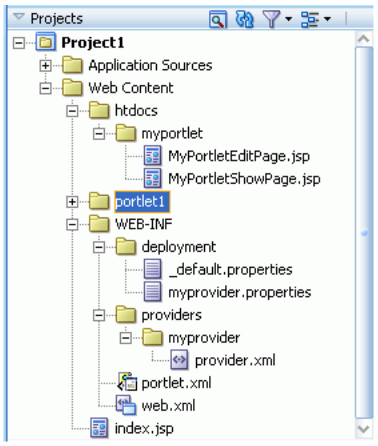
32. On this page you can map parameters to events. For more information about using events, see Section 7.2.3, "Passing Parameters and Submitting Events."

33. Click **Finish** to generate the files for your portlet. The following files should be generated for your project in the Application Navigator (see Figure 6–30):

    - 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 6–30 Application Navigator



6.5.2 Adding Portlet Logic to Your Oracle PDK-Java Portlet

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 on adding functionality and features.

6.5.3 Validating Your Oracle PDK-Java Portlet and Provider

After you have built your portlet, you need to check the configuration to ensure that the portlet and its provider operate correctly.

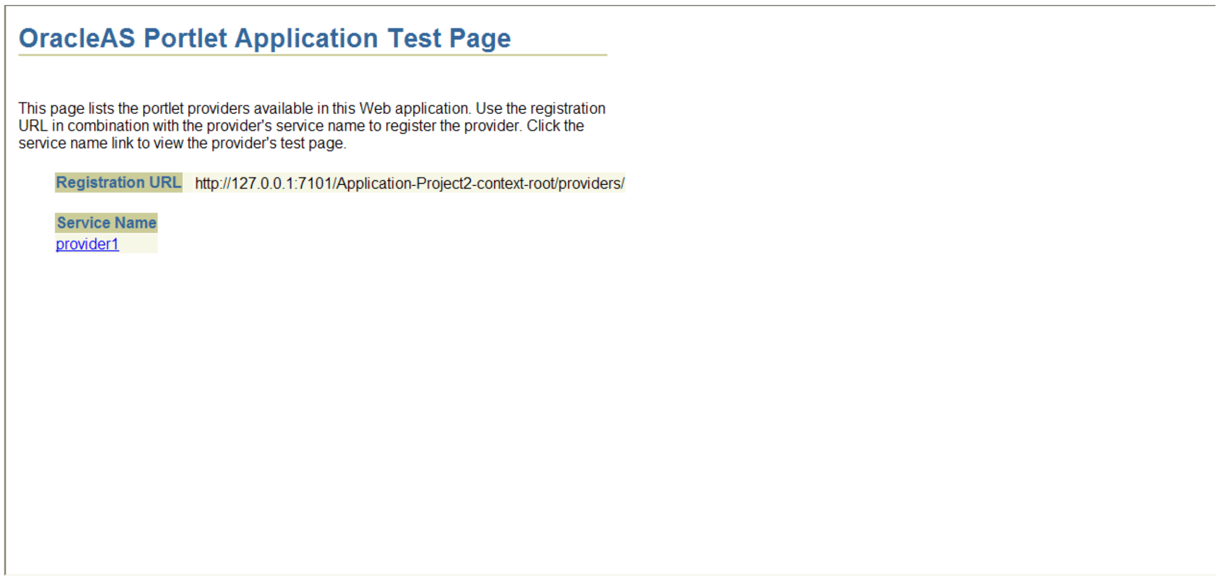
Note: This procedure is for testing purposes only. After this procedure, you still need to register your provider as described in Section 6.5.5, "Registering and Viewing Your Oracle PDK-Java Portlet". For development and production, you should always deploy your portlet to an application server as described in Section 6.5.4, "Deploying Your Oracle PDK-Java Portlet to an Application Server".

To validate your portlet and provider:

1. In Oracle JDeveloper, open the project you created in the previous sections.
2. In the Application Navigator, right-click the `index.jsp` file for you portlet and select **Run.**

   Your browser opens a page similar to the one shown in Figure 6–31.

Figure 6–31 Portlet Application Test Page



OracleAS Portlet Application Test Page

This page lists the portlet providers available in this Web application. Use the registration URL in combination with the provider's service name to register the provider. Click the service name link to view the provider's test page.

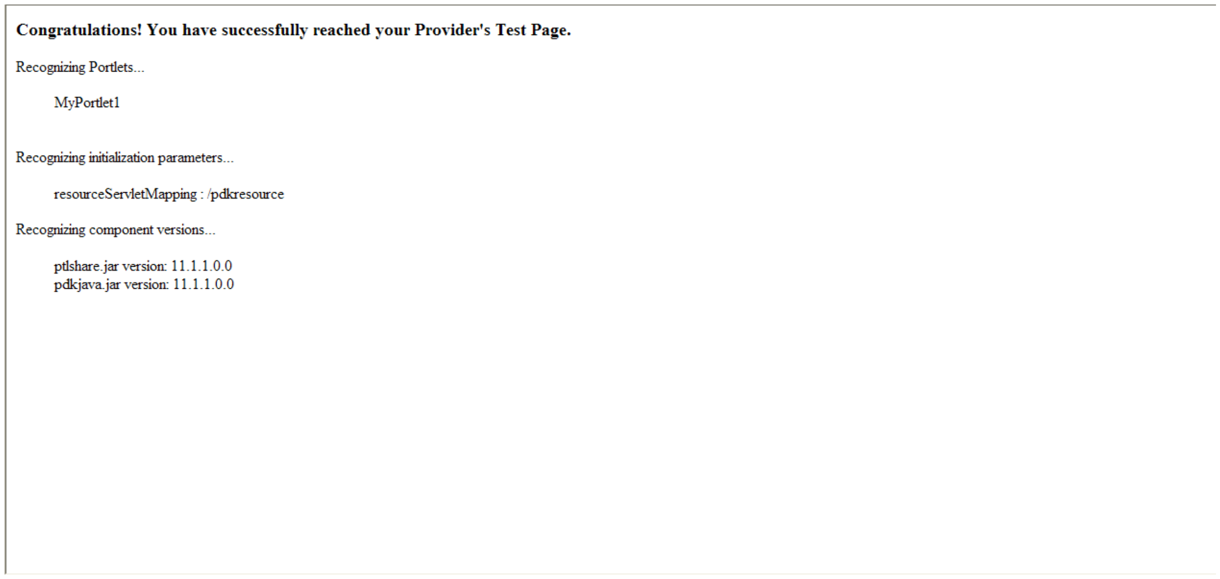
Registration URL http://127.0.0.1:7101/Application-Project2-context-root/providers/

Service Name
provider1

3. Click the link underneath Service Name.

   Your browser opens with a page similar to the one shown in Figure 6–32. Note that you need the URL from this page to register your provider, which is the next task.

Figure 6–32 Provider Test Page


Congratulations! You have successfully reached your Provider's Test Page.

Recognizing Portlets...

MyPortlet1

Recognizing initialization parameters...

resourceServletMapping : /pdkresource

Recognizing component versions...

ptlshare.jar version: 11.1.1.0.0
pdkjava.jar version: 11.1.1.0.0

6.5.4 Deploying Your Oracle PDK-Java Portlet to an Application Server

After you finish the wizard and successfully generate your portlet, you are ready to deploy it to the Oracle WebLogic Server (WLS). The following sections describe how to deploy an Oracle PDK-Java portlet to WLS.

To create and deploy the WAR file:

1. In the Application Navigator, right-click the project that contains your portlet and select **New**.
2. In the New Gallery, expand the General category and select **Deployment Profiles**.
3. In the Items list, select **WAR File** and click **OK.** The Create Deployment Profile -- WAR File dialog box opens.
4. In the Deployment Profile Name field, enter a meaningful name for the deployment profile (for example, `myj2eeportlet`).
5. Click **OK**. The WAR Deployment Profile Properties dialog box opens.
6. Under Web Application's Context Root, select **Specify Java EE Web Context Root** and enter the context root in the corresponding field, for example myj2eeportlet1.
7. Select the **Contributors** node under WEB-INF/lib.
8. Select **Portlet Development.**
9. Click **OK.** The Project Properties dialog opens.
10. Click **OK.**
11. In the Application Navigator, right-click your project and select **Deploy,** then select the deployment profile, next select **to,** and finally select the application server connection to which you want to deploy the portlet.
12. When the Deployment Finished message displays in the **Deployment Log** at the bottom of Oracle JDeveloper, verify that no errors occurred.
13. Construct the URL for your portlet as follows:

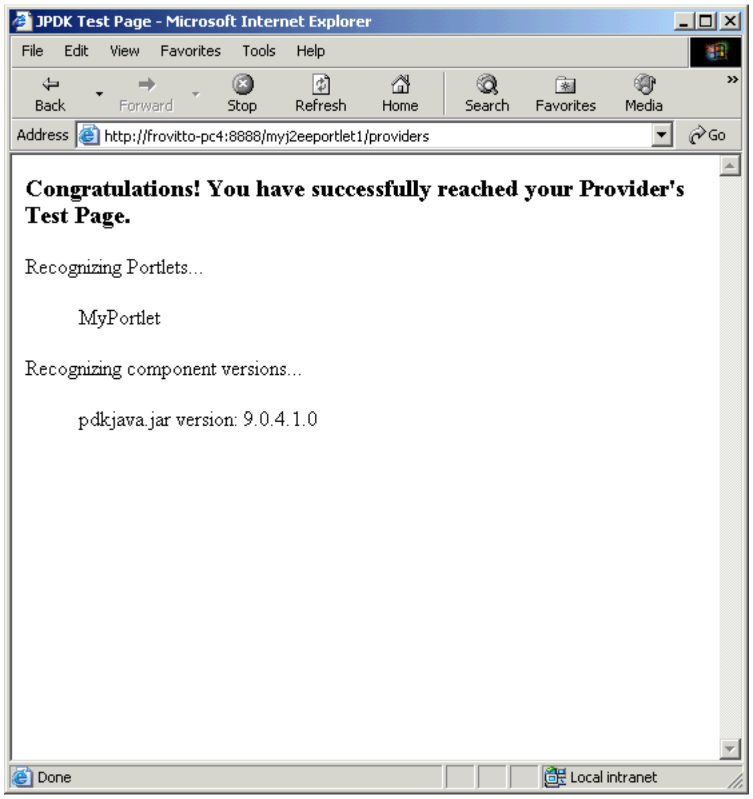
    http://*host*:*port*/*context-root*/providers where *host* is the server to which your provider has been deployed.

port is the Oracle 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.

14. In a Web browser, enter the URL from the previous step to ensure that it is working. You should see a page similar to the one in Figure 6–33.

Figure 6–33 PDK - Java Test Page for Portlets


JPDK Test Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media
Address http://frovitto-pc4:8888/myj2eeportlet1/providers
Go
Done
Local intranet

Congratulations! You have successfully reached your Provider's Test Page.

Recognizing Portlets...

MyPortlet

Recognizing component versions...

pdkjava.jar version: 9.0.4.1.0

6.5.5 Registering and Viewing Your Oracle PDK-Java Portlet

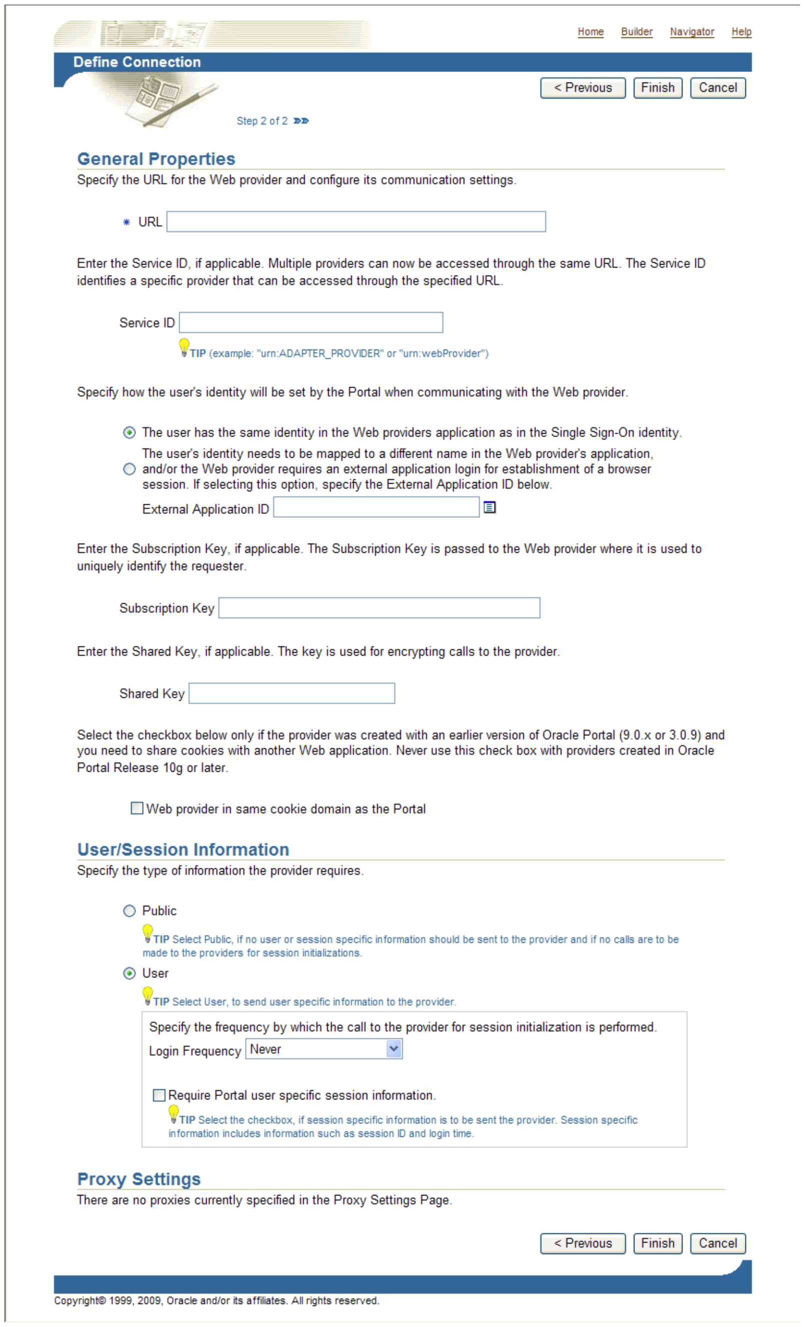
After you've created and deployed the provider and its portlets, you should register the provider with Oracle Portal. Registering your provider gives Oracle Portal the information it needs to locate and communicate with that provider. After you register a provider, the provider and its portlets become available in the Portlet Repository. They are also listed in the Oracle Portal Navigator.

Note: When you build portlets and providers with built-in tools, such as the Portlet Builder, Oracle Portal automatically registers the provider for you. Once you've created your portlet, it automatically displays in the Portlet Repository. Oracle Portal also offers built-in portlets that are contained in a preconfigured provider. For example, OmniPortlet and Web Clipping are portlets that you can use out of the box, and are already registered with Oracle Portal. You can view these portlets in the Add Portlets list. However, if you build the portlets and providers programmatically, you must then register these providers in order to make them available to the portal user.

To register providers for your Oracle PDK-Java portlets:

1. Open Oracle Portal and log in. Note that to register your provider, you need to have Manage or Edit privileges on providers. If you do not have these privileges, you need to request them from your administrator.
2. If you are not already on the Portal Builder page, click the **Builder** link in the upper right corner.
3. Click the **Administer** tab.
4. Click the **Portlets** sub tab.
5. In the Remote Providers portlet, click **Register a Provider** to display the Register Provider page.
6. In the **Name** field, enter the name of the provider. The name must not be more than 200 characters or contain spaces or other special characters.
7. In the **Display Name** field, enter a name to display for the provider when it is referenced, for example in the Portlet Repository. The display name must not be more than 200 characters.
8. In the **Timeout** field, enter the number of seconds Oracle Portal should try to connect to the provider before displaying the timeout message.
9. In the **Timeout Message** field, enter the message to display when Oracle Portal cannot establish contact with the provider within the number of seconds specified in the Timeout field. The message displays within the body of the portlet. The message may not contain HTML or mobileXML.
10. From the **Implementation Style** list, select **Web.**
11. Click **Next** to display the Define Connection page (Figure 6–34)

Figure 6–34 Define Connection Page



General Properties

Specify the URL for the Web provider and configure its communication settings.

* URL

Enter the Service ID, if applicable. Multiple providers can now be accessed through the same URL. The Service ID identifies a specific provider that can be accessed through the specified URL.

Service ID

TIP (example: "urn:ADAPTER_PROVIDER" or "urn:webProvider")

Specify how the user's identity will be set by the Portal when communicating with the Web provider.

[✓ The user has the same identity in the Web providers application as in the Single Sign-On identity.]

The user's identity needs to be mapped to a different name in the Web provider's application, and/or the Web provider requires an external application login for establishment of a browser session. If selecting this option, specify the External Application ID below.

External Application ID

Enter the Subscription Key, if applicable. The Subscription Key is passed to the Web provider where it is used to uniquely identify the requester.

Subscription Key

Enter the Shared Key, if applicable. The key is used for encrypting calls to the provider.

Shared Key

Select the checkbox below only if the provider was created with an earlier version of Oracle Portal (9.0.x or 3.0.9) and you need to share cookies with another Web application. Never use this check box with providers created in Oracle Portal Release 10g or later.

☐ Web provider in same cookie domain as the Portal

User/Session Information

Specify the type of information the provider requires.

Public

TIP Select Public, if no user or session specific information should be sent to the provider and if no calls are to be made to the providers for session initializations.

[✓ User]

TIP Select User, to send user specific information to the provider.

Specify the frequency by which the call to the provider for session initialization is performed.

Login Frequency: Never

☐ Require Portal user specific session information.

TIP Select the checkbox, if session specific information is to be sent the provider. Session specific information includes information such as session ID and login time.

Proxy Settings

There are no proxies currently specified in the Proxy Settings Page.

Copyright© 1999, 2009, Oracle and/or its affiliates. All rights reserved.

12. In the **URL** field, enter the URL for your provider. This URL is the one that you created at the end of Section 6.5.4, "Deploying Your Oracle PDK-Java Portlet to an Application Server."

13. In the **Service ID** field, enter the service ID for your provider.

Note: PDK-Java enables you to deploy multiple providers under a single adapter servlet. The providers are identified by the **Service ID** field. When you deploy a new provider, you must assign a service identifier to the provider and use that service identifier when creating your provider WAR file. The service identifier is used to look up a file called `service_id.properties`, which defines the characteristics of the provider, such as whether to display its test page.

For more information about service identifiers, refer to Section D.1.2, "Service Identifiers".

To access a sample provider test page, you must use the URL and service ID. An example of a URL for a sample provider is as follows:

```
http://myserver.com:8888/myj2eeportlet1/providers/sample
```

where, `http://myserver.com:8888/myj2eeportlet1/providers` is the URL value, and `sample` is the service ID.

14. Click **Finish**. You should see a Registration Confirmation page.
15. Your portlet should now be available for adding to pages just as any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in the *Oracle Fusion Middleware User's Guide for Oracle Portal*.

6.6 Setting up the WSRP and JPDK Applications in Oracle Portal

This section describes steps to manually configure the WSRP and JPDK application and deploying them on the Oracle WebLogic Server.

This section includes the following subsections:

- Configuring and Deploying the WSRP
- Configuring and Deploying the JPDK

6.6.1 Configuring and Deploying the WSRP

To configure the WSRP, complete the following steps:

- Creating WSRP Managed Server
- Adding the Required Libraries to the WSRP Managed Server
- Starting the WSRP Managed Server
- Configuring the Datasource
- Extending the Existing WebLogic Domain
- Adding wsm-pm to the WSRP Managed Server
- Deploying the Sample EAR File
- Configuring Oracle Web Cache
- Registering and Viewing Your Portlet

Creating WSRP Managed Server

You can create a WebLogic Managed Server on an existing domain using the Oracle WebLogic Server Administration Console to create the managed server instance and provision the shared libraries required to run a custom Oracle Portal application. To create the WSRP Managed Server, complete the following steps:

1. Log on to the Oracle WebLogic Server Administration Console.
2. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
3. In the Domain Structure tree, expand **Environment**, and then select **Servers**.

   The **Summary of Servers** page is displayed.
4. Click **New**.

   The **Create a New Server** page is displayed.
5. In the Create a New Server page enter the following information:
   - **Server Name**: Enter the name of the server, for example **WLS_WSRP**.
   - **Server Listen Port**: Enter the port number from which you want to access the server instance. For example, 9003.
   - Select the **No, this is a stand-alone server** radio button.
6. Click **Finish**.
7. In the **Summary of Servers page**, click on the Server (**WLS_WSRP**) you have created.
8. Select your machine name from the **Machine** drop-down list and click **Save**.
9. Click **Activate Changes**.

Adding the Required Libraries to the WSRP Managed Server

For a portlet producer application, you must deploy the following libraries to the new Managed Server or cluster:

- oracle.portlet-producer.wsrp(11.1.1,11.1.1)
- oracle.portlet-producer.jpdk(11.1.1,11.1.1)
- DMS Application (11.1.1.1.0)
- oracle.jrf.system.filter
- oracle.jsp.next(11.1.1,11.1.1)
- oracle.wsm.seedpolicies(11.1.1,11.1.1)
- wsil-wls

To add the libraries, complete the following steps:

1. Click **Lock & Edit**.
2. In the Domain Structure tree, select **Deployments**.

   The **Summary of Deployments** page is displayed.
3. Select **oracle.portlet-producer.wsrp(11.1.1,11.1.1)** from the Deployments table.

   The **Settings for oracle.portlet-producer.wsrp(11.1.1,11.1.1)** page is displayed.

Note: If the oracle.portlet-producer.wsrp(11.1.1,11.1.1) does not appear in the Deployment table, then click **Customize this table**, and disable the **Exclude libraries when displaying deployments** check box. Click **Apply**.

---

4. Click the **Targets** tab, and select **AdminServer** and **WLS_WSRP** from the Servers section.
5. Click **Save**.
6. Go to your **Summary of Deployments** page, and for each shared library, click the **Targets** tab, and then check **AdminServer** and **WLS_WSRP** from the Servers section.
7. To add the wsil-wls library, complete the following steps:
   a. Select **wsil-wls** from the Deployments table.
   b. Click the **Targets** tab, and select **wsil-wls** from the Component table.
   c. Click **Change Targets**, and then check **AdminServer** and **WLS_WSRP** from the Servers section.
   d. Click **Yes**.
8. Click **Activate Changes**.

Starting the WSRP Managed Server

To start the managed server, complete the following steps:

1. Click **Lock & Edit**.
2. In the Domain Structure tree, expand **Environment**, and then select **Servers**.

   The **Summary of Servers** page is displayed.
3. Click the **Control** tab, and check your created server (**WLS_WSRP**).
4. Click **Start**.

   The **Server Life Cycle Assistant page** is displayed.
5. Click **Yes**, to start the managed server.

Configuring the Datasource

To map the portletPrefs datasource targets to the managed server (**WLS_WSRP**), do the following:

1. In the Domain Structure tree, expand **Services**, and then select and expand **JDBC** and click **Data Sources**.

   The **Summary of JDBC Data Sources** page is displayed.
2. Click the Name associated with the `jdbc/portletPrefs` JNDI.
3. Click the **Targets** tab, and check **WLS_WSRP** from the Servers section.
4. Click **Save**.
5. Click **Activate Changes**.

Extending the Existing WebLogic Domain

To extend your existing Oracle WebLogic Server domain with the Oracle WSM Policy Manager, complete the following steps:

Note: You must stop all the servers running in the Oracle Portal domain.

1. Select **Extend an existing WebLogic Domain** and click **Next**.

   The **Select a WebLogic Domain Directory** screen is displayed.

2. Select your valid domain directory and click **Next**.

   The **Select Extension Source** screen is displayed.

3. Select **Oracle WSM Policy Manager - 11.1.1.0 [Oracle_Common]** (Select the one associated with your Oracle Portal) check box from the **Extend my domain automatically to support the following added products** option and click **Next**.

   The **Configure JDBC Data Sources** screen is displayed.

4. Click **Next**.

   The **Test JDBC Data Sources** screen is displayed.

5. Click **Next**.

   The **Configure JDBC Component Schema** screen is displayed.

6. Check the **OWSM MDS Schema** and enter the Host Name, Port, DBMS/ Service, Schema Password for the MDS schema. Click **Next**.

   Note: This is the schema created for MDS when you run the Repository Configuration Utility (RCU) at install time. If you did not created this schema already then you need to create this schema first using RCU.

   The **Test Component Schema** screen is displayed.

7. Confirm the test is successful and click **Next**.

   The **Select Optional Configuration** screen is displayed.

8. Select **Deployments and Service**s.

9. Click **Next**.

   The **Target Deployments to Clusters or Servers** screen is displayed.

10. Ensure **wsrp-pm** is targeted to servers including **WLS_WSRP** and **AdminServer**.

11. Click **Next**.

    The **Target Services to Cluster or Servers** screen is displayed

12. Click **Next**.

13. Review your settings in the **Configuration Summary** screen, and click **Extend**.

14. Click **Done**.

Start the Oracle WebLogic Administration Server, the managed server, and the WLS_WSRP managed server.

Adding wsm-pm to the WSRP Managed Server

You must add the wsm-pm to the WLS_WSRP managed server, by completing the following steps:

1. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
2. In the Domain Structure tree, select **Deployments**.

   The **Summary of Deployments** page is displayed.
3. Select **wsm-pm** from the Deployments table.
4. Click the **Targets** tab, and select **wsm-pm** from the Component table.
5. Click **Change Targets**, and then check **AdminServer** and **WLS_WSRP** from the Servers section.
6. Click **Yes**.

Deploying the Sample EAR File

To deploy the EAR file, download the **wsrp-samples.ear** file, from Download the Oracle Portlet Container at `http://www.oracle.com/technology/products/ias/portal/pdk.html` and do the following:

1. Click **Lock & Edit**.
2. In the Domain Structure tree, select **Deployments**.

   The **Summary of Deployments** page is displayed.
3. Click **Install**.
4. Select **wsrp-samples.ear** from your directory, click **Next**.
5. Select **Install this deployment as an application**, and click **Next**.
6. Check **WLS_WSRP** from the Servers section, and click **Next**.
7. Enter a name for the deployment, and click **Finish**.
8. Click **Activate Changes**.
9. In the **Summary of Deployments** page, select the application you have deployed, and then click **Start**.
10. Now that you have deployed the sample EAR file, you need to test its WSDL URL by entering it into a browser. The WSDL URL is of the form:

    ```
    http://host:port/portletapp/portlets?WSDL
    ```

    You should see a page similar to the one shown in Example 6–1.

Example 6–1 WSRP Producer WSDL Page

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions targetNamespace="urn:oasis:names:tc:wsrp:v1:wsdl"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:bind="urn:oasis:names:tc:wsrp:v1:bind">
<import namespace="urn:oasis:names:tc:wsrp:v1:bind"
 location="http://stamf10.us.abc.com:8090/wsrp-tools/portlets/wsrp1?WSDL=wsrp_v1_
bindings.wsdl" />
<service name="WSRP_v1_Service">
```

```
<port name="WSRPServiceDescriptionService"
 binding="bind:WSRP_v1_ServiceDescription_Binding_SOAP">
<soap:address
location="http://stamf10.us.oracle.com:8090/wsrp-tools/portlets/WSRPServiceDescrip
tionService" />
</port>
<port name="WSRPBaseService" binding="bind:WSRP_v1_Markup_Binding_SOAP">
<soap:address
location="http://stamf10.us.oracle.com:8090/wsrp-tools/portlets/WSRPBaseService"
/>
</port>
<port name="WSRPPortletManagementService"
 binding="bind:WSRP_v1_PortletManagement_Binding_SOAP">
<soap:address
location="http://stamf10.us.abc.com:8090/wsrp-tools/portlets/WSRPPortletManagement
Service" />
</port>
<port name="WSRPRegistrationService" binding="bind:WSRP_v1_Registration_Binding_
SOAP">
<soap:address
location="http://stamf10.us.abc.com:8090/wsrp-tools/portlets/WSRPRegistrationServi
ce"/>
</port>
</service>
</definitions>
```

Configuring Oracle Web Cache

To view the WSDL URL using the Oracle Web Cache port, edit the `portal.conf` file (Located at `ORACLE_INSTANCE\config\OHS\ohs1\moduleconf` in Windows) as follows:

```
<Location /portletapp>
    SetHandler weblogic-handler
    WebLogicHost servername.domain.com
    WebLogicPort 9003
</Location>
```

Save your file and restart the Oracle HTTP Server. Now you can access the WSDL URL through the Oracle Portal port (The default port is `8090`).

Registering and Viewing Your Portlet

After you've created and deployed the provider and its portlets, you should register the provider with Oracle Portal. For more information, see Registering and Viewing Your JSR 168 Portlet in the "*Oracle Fusion Middleware Developer's Guide for Oracle Portal*."

6.6.2 Configuring and Deploying the JPDK

To configure and deploying the JPDK, perform the following:

- Creating the JPDK Managed Server
- Adding the Required Libraries to the JPDK Managed Server
- Starting the JPDK Managed Server
- Configuring the Datasource
- Deploying the EAR File
- Configuring Oracle Web Cache

- Registering and Viewing Your Portlet

Creating the JPDK Managed Server

You can create a WebLogic Managed Server on an existing domain using the Oracle WebLogic Server Administration Console to create the managed server instance and provision the shared libraries required to run a custom Oracle Portal application. To create the JPDK Managed Server, do the following:

1. Log on to the WebLogic Server Administration Console.
2. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
3. In the Domain Structure tree, expand **Environment** and then select **Servers**.

   The **Summary of Servers** page is displayed.
4. Click **New**.

   The **Create a New Server** page is displayed.
5. In the Create a New Server page enter the following information:
   - **Server Name**: Enter the name of the server. For this step use **WLS_JPDK** as an example.
   - **Server Listen Port**: Enter the port number from which you want to access the server instance. For this step use **9004** as an example.
   - Select the **No, this is a stand-alone server** radio button.
6. Click **Finish**.
7. In the **Summary of Servers page**, click on the Server (**WLS_JPDK**) you have created.
8. Select your machine name from the **Machine** drop-down list and click **Save**.
9. Click **Activate Changes**.

Adding the Required Libraries to the JPDK Managed Server

For a portlet producer application, you must deploy the following libraries to the new Managed Server or cluster:

- oracle.portlet-producer.jpdk(11.1.1,11.1.1)
- DMS Application (11.1.1.1.0)
- oracle.jsp.next(11.1.1,11.1.1)

To add the libraries, do the following:

1. Click **Lock & Edit**.
2. In the Domain Structure tree, select **Deployments**.

   The **Summary of Deployments** page is displayed.
3. Select **oracle.portlet-producer.jpdk(11.1.1,11.1.1)** from the Deployments table.

   The **Settings for oracle.portlet-producer.jpdk(11.1.1,11.1.1)** page is displayed.

Note: If the oracle.portlet-producer.jpdk(11.1.1,11.1.1) does not appear in the Deployment table, then click **Customize this table**, and disable the **Exclude libraries when displaying deployments** check box. Click **Apply**.

---

4. Click the **Targets** tab, and select **AdminServer** and **WLS_JPDK** from the Servers section.
5. Click **Save**.
6. Go to your **Summary of Deployments** page, and for each shared library, click the **Targets** tab, and then check **AdminServer** and **WLS_JPDK** from the Servers section.
7. Click **Activate Changes**.

Starting the JPDK Managed Server

To start the managed server do the following:

1. Click **Lock & Edit**.
2. In the Domain Structure tree, expand **Environment** and then select **Servers**.

   The **Summary of Servers** page is displayed.
3. Click the **Control** tab, and select your created server (**WLS_JPDK**).
4. Click **Start**.

   The **Server Life Cycle Assistant page** is displayed.
5. Click **Yes**, to start the managed server.

Configuring the Datasource

To map the `portletPrefs` datasource targets to the managed server (WLS_JPDK), do the following:

1. In the Domain Structure tree, expand **Services**, and then select and expand **JDBC** and click **Data Sources**.

   The **Summary of JDBC Data Sources** page is displayed.
2. Click the Name associated with the **jdbc/portletPrefs** JNDI.
3. Click the **Targets** tab, and check **WLS_JPDK** from the Servers section.
4. Click **Save**.
5. Click **Activate Changes**.

Deploying the EAR File

To deploy the JPDk, do the following:

1. Click **Lock & Edit**.
2. In the Domain Structure tree, select **Deployments**.

   The **Summary of Deployments** page is displayed.
3. Click **Install**.

   The **Install Application Assistant** page is displayed.

4. In the **Path** field, enter the location of the jpdk.ear file (Located at `ORACLE_HOME/archives/applications` for Windows and `ORACLE_HOME\archives\applications` for UNIX).
5. Select **jpdk.ear**, and click **Next**.
6. Select **Install this deployment as an application**, and click **Next**.
7. Check **WLS_JPDK** from the Servers section, and click **Next**.
8. In the **Name** field, enter a name for the deployment.
9. Click **Finish**.
10. Click **Activate Changes**.
11. Click **Lock & Edit**.
12. From the **Summary of Deployments** page, select the application and click **Start** and then, Servicing all requests.
13. From the **Start Application Assistant**, click **Yes**.
14. Now that you have deployed the sample EAR file, you need to test its URL by entering it into a browser. The URL is of the form:

    ```
    http://host:port/jpdk/providers/sample/
    ```

For more information, see Creating Java Portlets in the *Oracle Fusion Middleware Developer's Guide for Oracle Portal*.

Configuring Oracle Web Cache

To view the JPDK sample URL using the Oracle Web Cache port, edit the `portal.conf` file (Located at `ORACLE_INSTANCE\config\OHS\ohs1\moduleconf` in Windows) as follows:

```
<Location /jpdk>
    SetHandler weblogic-handler
    WebLogicHost servername.domain.com
    WebLogicPort 9004
</Location>
```

Save your file and restart the Oracle HTTP Server. Now you can access the WSDL URL through the Oracle Portal port (The default port is `8090`).

Registering and Viewing Your Portlet

After you've created and deployed the provider and its portlets, you should register the provider with Oracle Portal. For more information, see "Registering and Viewing Your Oracle PDK-Java Portlet" section in the *Oracle Fusion Middleware Developer's Guide for Oracle Portal*.

7 Enhancing Java Portlets

This chapter explains how to enhance Java portlets you created with the Oracle JDeveloper Portal Add-In, and how to make a portlet out of your struts application. This chapter contains the following sections:

- Section 7.1, "Enhancing JPS Portlets"
- Section 7.2, "Enhancing PDK-Java Portlets"
- Section 7.3, "Building Struts Portlets with Oracle JDeveloper"

More On OTN

The source code for many of the examples referenced in this chapter is available as part of PDK-Java. You can download PDK-Java on Oracle Technology Network (OTN):

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

7.1 Enhancing JPS Portlets

Once you have built your initial portlet in the Portlet Wizard as described in Section 6.3.1, "Creating a JSR 168 Portlet", 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. One of the more important enhancements that you might wish to perform is Section 7.1.1, "Adding Personalization", which is described in this section.

7.1.1 Adding Personalization

In this section, you enhance the portlet you created in Section 6.3.1, "Creating a JSR 168 Portlet" with some code that allows 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.

7.1.1.1 Assumptions

To perform the tasks in this section, we are making the following assumption:

- You built a portlet using the wizard, successfully registered the producer, and added the portlet to the page.

7.1.1.2 Implementing Personalization

In this section, you add some code to My Java Portlet, redeploy the portlet, and then test it in Oracle Portal. To do this, perform the following steps:

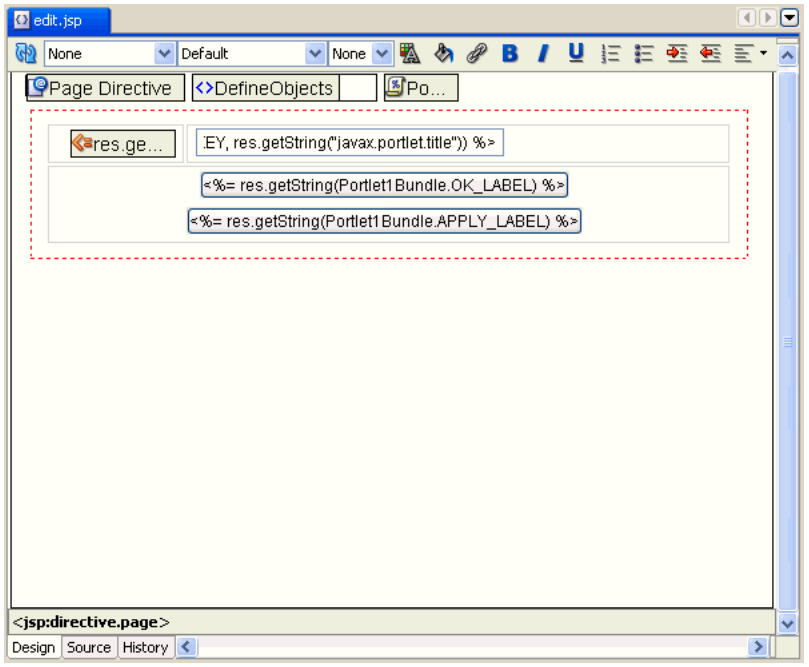
1. In Oracle JDeveloper, double-click the `view.jsp` file for your JPS-Standard portlet in the Application Navigator.

2. Add the code that is indicated in bold in the following snippet:

```
<%@ page contentType="text/html"
   import="javax.portlet.*,java.util.*,mypackage1.Portlet1,
    mypackage1.resource.Portlet1Bundle"%>"
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet"%>
<portlet:defineObjects/>
<%
   PortletPreferences prefs = renderRequest.getPreferences();
%>
<%= prefs.getValue("portletContent", "Portlet Content") %>
```

3. 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 7–1.

Figure 7–1 edit.jsp in the Design View

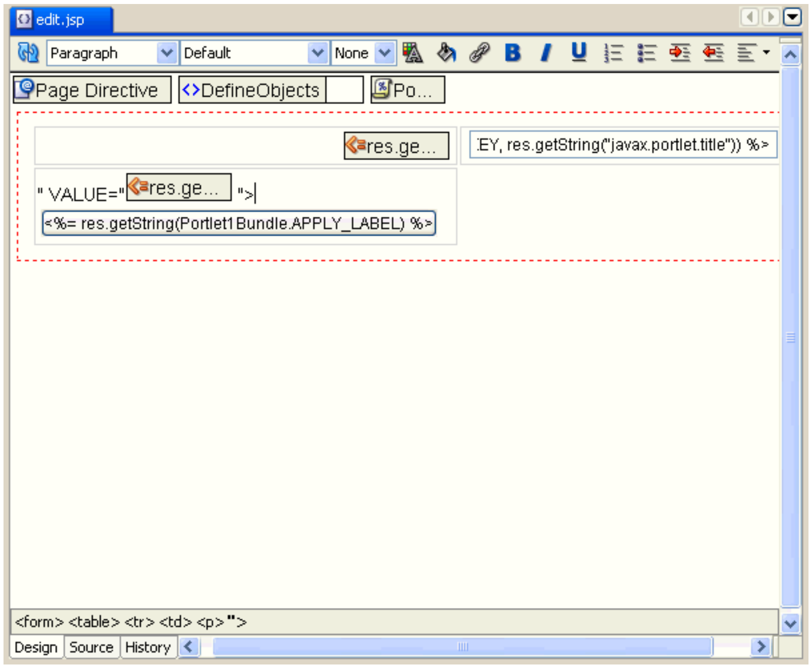

edit.jsp
None
Default
None
Page Directive
DefineObjects
Po...
res.ge...
EY, res.getString("javax.portlet.title")) %>
<%= res.getString(Portlet1Bundle.OK_LABEL) %>
<%= res.getString(Portlet1Bundle.APPLY_LABEL) %>
<jsp:directive.page>
Design
Source
History

4. Add the code that is indicated in bold in the following snippet to implement a form field called **Content**:

```
<FORM ACTION="<portlet:actionURL/>" METHOD="POST">
<TABLE BORDER="0">
<TR><TD WIDTH="20%">
<P CLASS="portlet-form-field" ALIGN="right">
<%= res.getString(Portlet1Bundle.PORTLETTITLE) %>
</P></TD><TD WIDTH="80%">
<INPUT CLASS="portlet-form-input-field" TYPE="TEXT"
  NAME="<%= Portlet1.PORTLETTITLE_KEY %>"
  VALUE="<%= prefs.getValue(Portlet1.PORTLETTITLE_KEY,
      res.getString("javax.portlet.title")) %>"
   SIZE="20">
</TD></TR>
<tr><td width="20%">
<p class="portlet-form-field" align="right">
Content
</p>
</td><td width="80%">
</td></tr>
<TR><TD COLSPAN="2" ALIGN="CENTER">
<INPUT CLASS="portlet-form-button" TYPE="SUBMIT" NAME=
   "<%= Portlet1.OK_ACTION%>"
   VALUE="<%= res.getString(Portlet1Bundle.OK_LABEL) %>">
<INPUT CLASS="portlet-form-button" TYPE="SUBMIT" NAME=
   ="<%=Portlet1.APPLY_ACTION %>"
   VALUE="<%= res.getString(Portlet1Bundle.APPLY_LABEL) %>">
</TD></TR>
</TABLE>
```

5. Click the **Design** tab to see the new form field that you just added (Figure 7–2).

Figure 7–2 Modified edit.jsp in the Design View


edit.jsp
Paragraph
Default
None
Page Directive
DefineObjects
Po...
res.ge...
EY, res.getString("javax.portlet.title")) %>
" VALUE="
res.ge...
">
<%= res.getString(Portlet1Bundle.APPLY_LABEL) %>
<form> <table> <tr> <td> <p> ">
Design
Source
History

6. 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();
```

7. Redeploy the portlet. Notice that Oracle JDeveloper automatically saves and compiles the code before deploying the portlet. Refer to Section 6.3.3, "Deploying

Your JSR 168 Portlet to the Oracle WebLogic Server" for a reminder of how to perform this step.

8. In Oracle Portal, 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.
9. Click the **Customize** link. You can see the new form field that you added in Oracle JDeveloper.
10. 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>
```

11. Click **Apply** and then click **Close**. The HTML is rendered in the portlet.

7.2 Enhancing PDK-Java Portlets

Once you have built your initial portlet in the Portlet Wizard as described in Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper", you may perform the following tasks to enhance it:

- Section 7.2.1, "Adding Show Modes"
- Section 7.2.2, "Adding Personalization"
- Section 7.2.3, "Passing Parameters and Submitting Events"
- Section 7.2.4, "Using JNDI Variables"
- Section 7.2.6, "Accessing Session Information"
- Section 7.2.7, "Implementing Portlet Security"
- Section 7.2.8, "Controlling the Export/Import of Portlet Personalizations"
- Section 7.2.9, "Enhancing Portlet Performance with Caching"
- Section 7.2.11, "Writing Multilingual Portlets"

This section assumes the following:

- You are familiar with portlet terminology such as portlet Show modes. Refer to Chapter 1, "Understanding Portlets" and Section 6.1, "Guidelines for Writing Java Portlets".
- You have already downloaded and installed the Java Portlet Container.
- 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`

7.2.1 Adding Show Modes

In the Portlet Wizard, you add Show modes by checking boxes on the wizard pages. Refer to Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper" for more information about using the wizard. For each Show mode that you select in the wizard, a basic HelloWorld skeleton is created. If you need to add a Show mode after creating the portlet or you are adding one of the modes (preview or link) not available through the wizard, you can do that manually by updating `provider.xml` and HTML or JSPs in Oracle JDeveloper. The following sections explain how to add Show modes to a PDK-Java portlet:

- Section 7.2.1.2, "Implementing Extra Show Modes"
- Section 7.2.1.3, "Updating the XML Provider Definition"
- Section 7.2.1.4, "Viewing the Portlet"

Once you have completed this section, you will be able to implement any Show 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 Help mode in detail, you will understand how to do it, as the process is the same as for Preview mode, which is described here.

More On OTN

For more detailed information on the PDK runtime classes used in this section, refer to 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 on the syntax of `provider.xml`, refer to the provider Javadoc:

```
http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_
reference_v2.html
```

7.2.1.1 Assumptions

To perform the tasks in this section, we are making the following assumption:

- You built a portlet using the wizard, successfully registered the producer, and added the portlet to the page.

7.2.1.2 Implementing Extra Show Modes

Your first task when creating Show modes manually is to create an HTML file or JSP for each mode. For example, if you want to implement Preview mode, you need to create an HTML file to provide preview 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 `PreviewPage.html`. The content of the file could be something similar to the following:

   ```
   <p>This is the <i>preview</i> mode of your portlet!</p>
   ```

Once you have created the HTML file for previewing content, you are ready to update the XML provider definition.

7.2.1.3 Updating the XML Provider Definition

When you want to expose additional Show modes you must update your XML provider 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.

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider 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 Preview mode, perform the following steps:

1. Edit the provider definition file, `provider.xml` and add the tag to activate Preview mode within the `<portlet></portlet>` tags:

   ```
   <showPreview>true</showPreview>
   ```

2. Specify the preview page to be the HTML page that you created in Section 7.2.1.2, "Implementing Extra Show Modes":

   ```
   <previewPage>/htdocs/myportlet/MyPortletPreviewPage.html</previewPage>
   ```

3. Save the updates to `provider.xml`.
4. Redeploy your portlet. Refer to step 11 in Section 6.5.4, "Deploying Your Oracle PDK-Java Portlet to an Application Server.".

   When you redeploy, Oracle JDeveloper automatically saves and compiles the code before deploying the portlet.

7.2.1.4 Viewing the Portlet

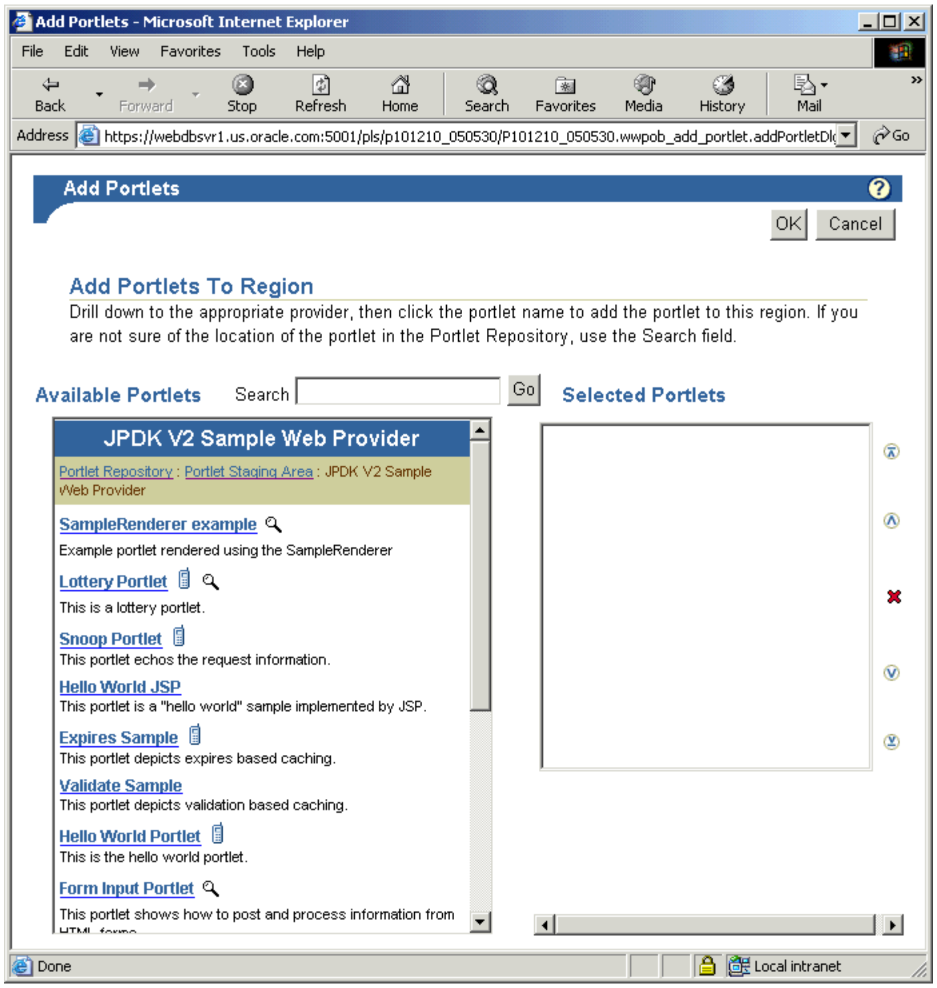
To view the new Show modes, you must ensure that your updated XML provider definition is re-parsed. To do this, perform the following steps:

1. Copy the HTML file you created in Section 7.2.1.2, "Implementing Extra Show Modes" and `provider.xml` to the WebLogic Server instance where you plan to deploy the portlet.
2. Refresh the provider.
3. Refresh the portal page containing your portlet.

To view Preview mode, do the following:

1. Edit a page or create a new page and choose **Add Portlet**.
2. Navigate to the location of your provider in the Portlet Repository (for example, Portlet Staging Area) and find your portlet. Note the magnifying glass icon next to the portlet shown in Figure 7–3

Figure 7–3 Add Portlet Page


Add Portlets - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail
Address https://webdbsvr1.us.oracle.com:5001/pls/p101210_050530/P101210_050530.wwpob_add_portlet.addPortletDl
Go
Add Portlets
OK Cancel

Add Portlets To Region

Drill down to the appropriate provider, then click the portlet name to add the portlet to this region. If you are not sure of the location of the portlet in the Portlet Repository, use the Search field.

Available Portlets
Search
Go
Selected Portlets

JPDK V2 Sample Web Provider

Portlet Repository : Portlet Staging Area : JPDK V2 Sample Web Provider

SampleRenderer example
Example portlet rendered using the SampleRenderer

Lottery Portlet
This is a lottery portlet.

Snoop Portlet
This portlet echos the request information.

Hello World JSP
This portlet is a "hello world" sample implemented by JSP.

Expires Sample
This portlet depicts expires based caching.

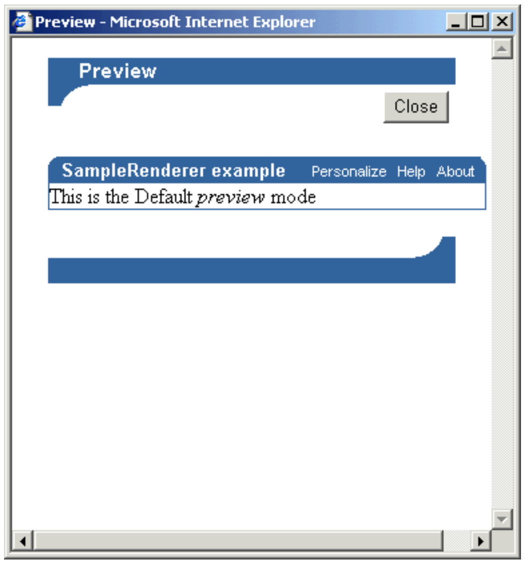
Validate Sample
This portlet depicts validation based caching.

Hello World Portlet
This is the hello world portlet.

Form Input Portlet
This portlet shows how to post and process information from

3. Click the magnifying glass icon next to the portlet and a preview window similar to the one in Figure 7–4 displays.

Figure 7–4 Preview Window



This is the Default *preview* mode

7.2.2 Adding Personalization

In Section 7.2.1, "Adding Show Modes" you learned how to use the PDK Provider Framework to activate and render additional Show modes that were either not activated when creating the portlet with the wizard or not available through the wizard (such as Link and Preview modes). This section describes the two Personalization modes (Edit and Edit Defaults) in more detail. When selected in the Java Portlet Wizard, Edit page and Edit Defaults page cause the generation of skeleton code for the two Personalization modes. The skeleton code enables you to access the personalization framework with a few lines of code rather than completely hand coding a personalization framework and a data store to hold the values.

To add personalization to your portlet, you need to do the following:

- Update the Edit page of your portlet to set and retrieve personalization changes.
- Update the Edit Defaults page of your portlet to set and retrieve personalization changes.
- Update the Show page of your portlets to use the personalization set by the user.

The Edit and Edit Defaults modes allow portlet users to change a set of customizable parameters supported by the portlet, which typically drive the way the portlet is rendered in other modes. For a particular instance of a portlet on an Oracle Portal page, the personalizations made in the Edit and Edit Defaults modes apply only to that instance of the portlet. This is explained as follows:

- **Edit** mode personalizations are specific to the individual user making the personalizations. This mode is activated by clicking the **Personalize** link on the portlet header in show mode.
- **Edit defaults** mode personalizations apply to all users in the same locale who have not yet made specific personalizations to that portlet instance. This mode is generally only available to page designers, and can be activated by following the **Edit** icon on the page.

When rendering Edit and Edit Defaults modes, a `PortletRenderer` can carry out either of the following tasks to support the personalization process:

- **Render the Edit Form**: For each of the portlet's customizable parameters, `PortletRenderer` uses a `PortletPersonalizationManager` to retrieve the current value and renders a control in an HTML form so the current value can be edited.
- **Handle Edit Form actions**: When an **OK** or **Apply** button is clicked on the standard edit form header, `PortletRenderer` uses a `PortletPersonalizationManager` to store the personalized parameters submitted by the edit form and redirects the browser to the appropriate portal page.

Therefore, the purpose of the `PortletPersonalizationManager` controller is to enable a `PortletRenderer` to store and retrieve the current values of customizable parameters that apply to a particular portlet instance and user. The PDK Framework uses the abstraction of a `PersonalizationObject` as a container for a set of personalized parameters and a `PortletReference` as the key under which a set of personalizations are stored. Thus, a `PortletPersonalizationManager` is simply a mechanism that allows the storage and retrieval of persisted `PersonalizationObjects` under a given `PortletReference`.

A preference store is a mechanism for storing information like user preference data, portlet/provider settings, or even portlet data, while using Oracle Portal. The information stored in the preference store is persistent in the sense that, even if you log out and log back in later, you can still access previously saved preferences. The preference store maintains the user preference information and invokes the user preferences whenever the user logs in again. PDK-Java provides the `PrefStorePersonalizationManager`, which uses a `PreferenceStore` implementation to persist personalized data. Currently, PDK-Java has two `PreferenceStore` implementations: `DBPreferenceStore` and `FilePreferenceStore`. The `DBPreferenceStore` persists data using a JDBC compatible relational database and `FilePreferenceStore` persists data using the file system.

For more details of these implementations, refer to 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`

Note: PDK-Java provides the Preference Store Migration/Upgrade Utility to help migrate the preference store from a file system to a database and upgrade personalizations from earlier releases. This utility is described more fully on OTN.

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

To add personalization functionality to your portlet you use `PrefStorePersonalizationManager` in conjunction with `NameValuePersonalizationObject`, that is, the default `PersonalizationObject` implementation. By default, the wizard generates a simple edit form for both the Edit and Edit Defaults modes to enable users to personalize the portlet title. This section describes how to update the existing code to enable portal users to personalize the portlet greeting.

7.2.2.1 Assumptions

To perform the tasks in this section, we are making the following assumptions:

1. You have followed through and understood the following sections:
   - Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper"
   - Section 7.2.1, "Adding Show Modes"
2. You built a portlet using the wizard, with **Edit page** and **Edit Defaults page** selected, and successfully added it to a page.

7.2.2.2 Implementing Personalization for Edit and Edit Defaults Pages

The Edit page of your portlet is called when a user personalizes the portlet. By default, the JSP generated by the wizard includes all of the required code to provide personalization of the portlet title. You just need to insert a few lines of code into the Edit page for additional personalization.

7.2.2.2.1 Reviewing the Generated Code The wizard creates the following code for you by default:

```
<%@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.personalize.NameValuePersonalizationObject"
   import="oracle.portal.provider.v2.render.PortletRendererUtil"
%>

<%
 PortletRenderRequest pReq = (PortletRenderRequest)
   request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
%>

<P>Hello <%=pReq.getUser().getName() %>.</P>
<P>This is the <b><i>Edit</i></b> render mode!</P>
<%-- This page both displays the personalization
     form and processes it,. Display the form if
     there is no action parameter, process it
     otherwise --%>

<%
 String actionParam = PortletRendererUtil.getEditFormParameter(pReq);
 String action = request.getParameter(actionParam);
 String title = request.getParameter("my2portlet_title");
 NameValuePersonalizationObject data = (NameValuePersonalizationObject)
    PortletRendererUtil.getEditData(pReq);
 // Cancel automatically redirects to the page, so
 // will only receive OK or APPLY
 if (action !=null)
 {
    data.setPortletTitle(title);
    PortletRendererUtil.submitEditData(pReq, data);
    return;
 }

 // Otherwise just render the form.
 title = data.getPortletTitle();
%>
<table border="0">
  <td width="20%">
   <p align="right">Title:</p>
  </td>
  <td width="80%">
```

```
      <input type="TEXT" name="my2portlet_title" value="<%= title %>">
    </td>
</table>
```

7.2.2.2.2 Modifying the Generated Code The JSP contains an input field for the portlet title. This field represents the Personalize page of the portlet where users can update the portlet title. To modify the generated code, perform the following steps:

1. Following the table in the generated code, add a second table containing a text field and a prompt, allowing users to enter a new greeting for the portlet:

   ```
   <table border="0">
     <tr>
       <td width="20%">
         <p align="right">Greeting:</p>
       </td>
       <td width="80%">
         <input type="TEXT" name="myportlet_greeting" value="<%= greeting %>">
       </td>
     </tr>
   </table>
   ```

2. This HTML simply specifies a field to enter a new greeting on the Edit page. This new greeting is displayed in the portlet's Shared Screen mode. Next, you add a string below `String title` that retrieves the value of the greeting:

   ```
   String title = request.getParameter("my2portlet_title");
   String greeting = request.getParameter("myportlet_greeting");
   ```

3. Generating an Edit page from the wizard automatically includes access to the personalization framework in the page code. At the top of the Edit page, you see the `NameValuePersonalizationObject` declared. This form of personalization in Oracle Portal allows easy storage of name/value pairs.

   The Edit page handles two cases: viewing the page or applying changes to it. The changes we have made so far affect the code for viewing the page. Applying changes to the Edit page is handled in the block of code beginning with `if (action !=null)`.

   In this block of code, you must store the new portlet greeting. You must also account for the case where the user decides to make no changes and you simply retrieve the existing greeting:

   ```
   if (action !=null)
   {
        data.setPortletTitle(title);
        //Put the new greeting.
        data.putString("myportlet_greeting", greeting);
        PortletRendererUtil.submitEditData(pReq, data);
        return;
   }
   //Otherwise just render the form.
   title = data.getPortletTitle();
   //Get the old greeting.
   greeting = data.getString("myportlet_greeting");
   ```

You are now done updating the Edit page.

You can simply duplicate these changes for the Edit Defaults page. The Edit Defaults page is called when a page designer or portal administrator clicks **Edit** on the page and then clicks the Edit Defaults icon for the portlet. This page sets the default personalization for this instance of the portlet. Even though the code in the JSP is identical, the PDK Framework and Oracle Portal automatically handle the personalization differently depending on the Show mode (Edit or Edit Defaults).

7.2.2.3 Implementing Personalization for Show Pages

To have access to the personalization data in the portlet's Shared Screen mode, you need to add a few lines of code to the Show page. By adding these lines you perform the following:

- Adding import statements.
- Declaring the `NameValuePersonalizationObject`.
- Retrieving the personalization data.

To implement personalization of the Show page, perform the following steps:

1. Edit your Show page and import `NameValuePersonalizationObject` and `PortletRendererUtil`. You can copy these from the Edit page if necessary.

   ```
   <%@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.personalize.
        NameValuePersonalizationObject"
      import="oracle.portal.provider.v2.render.PortletRendererUtil"
   %>
   ```

2. Declare the `NameValuePersonalizationObject` and retrieve the edit data from the portlet render request. You can copy this from the portlet's Edit page.

   ```
   <%
    PortletRenderRequest pReq = (PortletRenderRequest)
      request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
      NameValuePersonalizationObject data = (NameValuePersonalizationObject)
        PortletRendererUtil.getEditData(pReq);
   %>
   ```

3. Get the string information from the personalization framework:

   ```
   <%
   String greeting = data.getString("myportlet_greeting");
   %>
   ```

4. Add some text to the Show page that displays the greeting in the Shared Screen mode of the portlet.

   ```
   <P>Hello <%= pReq.getUser().getName() %>.</P>
   <P>This is the <b><i>show</i>,</b> render mode!</P>
   <P>Greeting: <%= greeting %></P>
   ```

You have now completed updating the Show page of the portlet.

7.2.2.4 Preference Information Within the XML Provider Definition

The Portlet Wizard generates all of the necessary tags for accessing the `PreferenceStore` in the XML provider definition file (`provider.xml`). By default, at the provider level, the wizard uses the `FilePreferenceStore` class to store preferences:

```
<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>
```

At the portlet level, tags are added to use `PrefStorePersonalizationManager` as the `personalizationManager` class and `NameValuePersonalizationObject` as the data class:

```
<personalizationManager class="oracle.portal.provider.v2.personalize.
   PrefStorePersonalizationManager">
    <dataClass>oracle.portal.provider.v2.NewValuePersonalizationObject</dataClass>
</personalizationManager>
```

You need not make any changes or updates to the XML Provider Definition if you choose to continue to use the `FilePreferenceStore` class. However, if you have a global environment for Oracle Portal (for example, you are running in a load balanced, multi-node cluster of Oracle Containers for Java EE instances) or would prefer to store preferences in the database, you can change the class from `FilePreferenceStore` to `DBPreferenceStore`.

For more information on using `DBPreferenceStore`, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

7.2.2.5 Viewing the Portlet

To view the personalization changes you made in the preceding sections, you need to deploy the portlet to your application server or Oracle Containers for Java EE and refresh the page containing your portlet. For more information on deploying your portlet, refer to Section 6.5.4, "Deploying Your Oracle PDK-Java Portlet to an Application Server".

You should now see that the portlet contains a null greeting. Click **Personalize** in the portlet title bar and update the greeting. When you return to the page, you should see your changes.

You can also test Edit Defaults by clicking **Edit** on the page and then clicking the Edit Defaults icon. Since you have already modified the portlet, the changes will not appear to you in Shared Screen mode unless you view the page as a public user or a different user.

7.2.3 Passing Parameters and Submitting Events

Oracle Portal and the PDK provide page parameters, public and private portlet parameters, and 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, refer to the following:

- Section 2.12, "Public Portlet Parameters Support"
- Section 2.13, "Private Portlet Parameter Support"
- Section 2.14, "Event Support"

7.2.3.1 Assumptions

To perform the tasks in this section, the following assumptions are made:

1. You have followed through and understood Section 6.5, "Building Oracle 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.

7.2.3.2 Adding Public Parameters

Using the wizard in Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper", you built a basic portlet and specified a parameter called `MyParam`. If you did not create a parameter, you can create a new portlet now by right clicking on `provider.xml` in the Applications - Navigator of Oracle JDeveloper, selecting **Add Portlet**, and following the steps in Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper".

By default, the wizard creates a portlet to which you can easily map page parameters without updating any code or files. In this section, you will use the default parameter created for you by the wizard.

To use the default parameter, you need only register the provider and add the portlet to a page. After that, you perform the following tasks:

- Create a page parameter.
- Wire the page parameter to your Java portlet.
- Enter parameter values in the URL or another portlet that passes this page parameter.

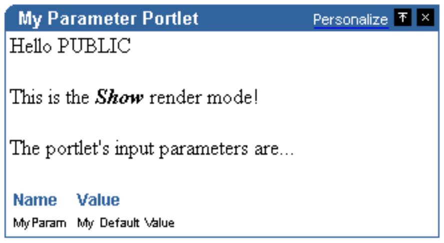
To add parameters to your portlet:

1. Go to the **Parameter** tab of the page properties. Note that parameters should be enabled by default, but, if not, you must enable them before proceeding.
2. Create a page parameter called `MyParameter` with a default value of `My Default Value`.
3. Expand your Java portlet and map the page parameter you just created to the portlet parameter. The portlet's parameter should map to the page parameter called `MyParameter`.
4. Go back to the page. Notice that, in the portlet, a value of `My Default Value` appears.
5. View the page and enter the parameter and a value at the end of the URL:

   ```
   &MyParameter=This%20portlet%20works
   ```

   Figure 7–5 shows an example of a parameter portlet.

Figure 7–5 Parameter Portlet



Hello PUBLIC

This is the ***Show*** render mode!

The portlet's input parameters are...

Name	Value
MyParam	My Default Value

If you have a portlet, such as the Simple Parameter Form included with OmniPortlet, that can pass parameters, you can map parameters from that portlet to your Java portlet using the Events tab.

If you now take a look at the code and tags generated by the wizard, you see that very little code was needed to enable parameters in the Java portlet.

Review `provider.xml`. Note that the wizard added one tag group called `inputParameter`, which includes the name of the parameter for which the portlet listens.

```
<inputParameter class="oracle.portal.provider.v2.DefaultParameterDefinition">
  <name>MyParam</name>
  <displayName>My Portlet Parameter</displayName>
</inputParameter>
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

The wizard also generated code in the JSP for your Show page that receives this parameter, and displays the parameter name and its value.

```
<%
ParameterDefinition params[] =
  pReq.GetPortletDefinition().getInputParameters();
%>

<p>This portlets input parameters are ...</p>
<table align="left" width="50%"><tr><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 < params.length; j++)
    {
       temp.append(values[j]);
       if (j + 1 != values.length)
       {
         temp.append(", ");
       }
    }
    value = temp.toString();
  }
  else
  {
    value = "No values submitted yet.";
  }
%>
<tr>
  <td><span class="PortletText2" <%= name %></span></td>
  <td><span class="PortletText2" <%= value %></span></td>
</tr>
<%
}
%>
</table>
```

7.2.3.3 Passing Private Portlet Parameters

Parameters that are used within a single portlet are known as private parameters. They are visible to a single portlet instance only. Portlet parameters are created and accessed using PDK-Java APIs described in this section. Parameter names are qualified so that it will be correctly handled by the portlet consumer. The following API will do this:

```
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 utilities for using private parameters are discussed in Section 7.2.3.3.2, "Building Links with the Portlet URL Types" and Section 7.2.3.3.3, "Building Forms with the Portlet URL Types."

7.2.3.3.1 Portlet URL Types

Intraportlet links refer to the Oracle Portal page on which the portlet resides, and that portlet is most likely running remotely from Oracle Portal. Hence, you must consider how the portlet can render a link to the correct page without some knowledge of the Oracle Portal page's URL. For more information about the types of links used by portlets, refer to Section 6.1.2, "Guidelines for Navigation within a Portlet".

When Oracle Portal requests that a portlet render itself, Oracle Portal passes it various URLs, which the portlet can then use to render links, including any intraportlet links it requires. You can fetch and manipulate these URLs to simplify the task of creating links among portlets and pages in Oracle Portal.

Oracle Portal provides the following URLs to its 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. This URL is useful to both desktop and mobile portlets.
- **DESIGN_LINK** is a URL to an Oracle Portal page that represents the portlet's personalization page. In Oracle Portal, a portlet's Edit and Customize modes are not rendered on the same page as the portlet. The Edit and Customize modes take over the entire browser window. Oracle Portal's portlet edit/customize page is not accessible to every user. It represents a minimal, static framework in which the portlet is free to render its personalization or edit options. This URL is only of use when rendering edit and customize links, which themselves are only supported in desktop clients.
- **LOGIN_LINK** is a URL to OracleAS Single Sign-On Server, should the portlet need to prompt the user (if PUBLIC) to login. This link is rarely used and only applicable to the desktop rendering of portlets.
- **BACK_LINK** is a URL to a page that Oracle Portal considers a useful return point from the current page where the portlet renders itself. For example, when the portlet is rendering it's Edit page, this link refers to the page on which the portlet resides and from which the user navigated to the Edit 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 Customize mode). Mobile portlets render a Back link automatically leaving the portlet to render just it's own content.
- **EVENT_LINK** is a URL that raises an event rather than explicitly navigate to some page. This link points to the Oracle Portal entry point to the event manager. This URL is useful to both desktop and mobile portlets.

7.2.3.3.2 Building Links with the Portlet URL Types To build links with the URL parameters, 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 the PDK:

```
portletRenderRequest.getRenderContext().getPageURL()
portletRenderRequest.getRenderContext().getEventURL()
portletRenderRequest.getRenderContext().getDesignURL()
portletRenderRequest.getRenderContext().getLoginServerURL()
portletRenderRequest.getRenderContext().getBackURL()
```

In the case of 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, 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 in order to get the thesaurus to operate on that new word. Refer to the example in Section 7.2.3.3.3, "Building Forms with the Portlet URL Types" to see the initial submission form that sets the value of `q`.

Note: When rendering attributes in `SimpleResult` (for a mobile portlet), you must escape the attribute value if it is likely to contain invalid XML characters. Most URLs contain `&` to separate the URL's parameters. Hence, you usually need to escape attributes that contain URLs with:

```
oracle.portal.utils.xml.v2.XMLUtil.escapeXMLAttribute
```

```
<%
    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 on 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>
```

7.2.3.3.3 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 7.2.3.3.2, "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, the developer needs 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. Refer to the example in Section 7.2.3.3.2, "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>
```

7.2.3.3.4 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 explicit 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 7.2.3.3.2, "Building Links with the Portlet URL Types" and Section 7.2.3.3.3, "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 subsequently.
- Use session storage to record the portlet state and URL 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 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. Refer to Section 7.2.6.2, "Implementing Session Storage" for more information about implementing session storage.

The following portlet code comes from the multi-page example in the sample provider 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 framework intercepts and interprets as an override to the value of the `showPage` parameter. If the PDK framework encounters the qualified version of the parameter when the multi-page portlet is requested, it will render the resource identified by `next_page` rather than `first.jsp`. Note that the PDK 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 on how to build portlets from struts applications, refer to Section 7.3, "Building Struts Portlets with Oracle JDeveloper".

`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, 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 on 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>
```

Partial Page Refresh

Portlets can refresh themselves without refreshing the entire page. For example, in the multi-page portlet sample, `firstpage.jsp` uses the API to specifically enable the portlet to link to and display the second page without refreshing the entire portal page. The text in bold in the following example shows how this can be set:

```
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ page language="java" session="false" %>
<%@ page import="oracle.portal.provider.v2.url.UrlUtils" %>
<%@ page import="oracle.portal.provider.v2.render.http.HttpPortletRendererUtil" %>
<%@ page import="oracle.portal.provider.v2.render.PortletRenderRequest" %>
<%@ page import="oracle.portal.provider.v2.http.HttpCommonConstants" %>
<%@ page import="oracle.portal.utils.NameValue" %>
<%
PortletRenderRequest prr = (PortletRenderRequest)
request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);

NameValue[] linkParams = new NameValue[1];
linkParams[0] = new NameValue(HttpPortletRendererUtil.portletParameter(request,
"next_page"), "/htdocs/multipage/second.jsp");
%>

<center>
Hello, this is the first page<p>
<%=UrlUtils.constructHTMLLink(prr, UrlUtils.REFRESH_LINK, "second page", "",
linkParams, true, true)%>
</center>
```

7.2.3.4 Submitting Events

In the previous section, you created a portlet that received parameters. Now you will create a portlet that passes parameters and events to other portlets on the same page or a different page. Some portlets, like the Simple Parameter Form in OmniPortlet, provide an easy, declarative interface to create a simple form to pass parameters to other portlets. If you want complete control over the events passed and the look of your portlet, though, you can add events to your Java portlet.

The Portlet Wizard does not create all of the code needed to pass parameters to other portlets. The wizard updates the tags in `provider.xml` and requires that you add the necessary business logic to your JSP code. To create a portlet that uses events, you perform the following tasks:

- Create a new portlet with the Portlet Wizard.
- Add code to your JSP page.
- Map this portlet's parameters to the portlet you created in Section 7.2.3.2, "Adding Public Parameters".

Creating an Events Portlet

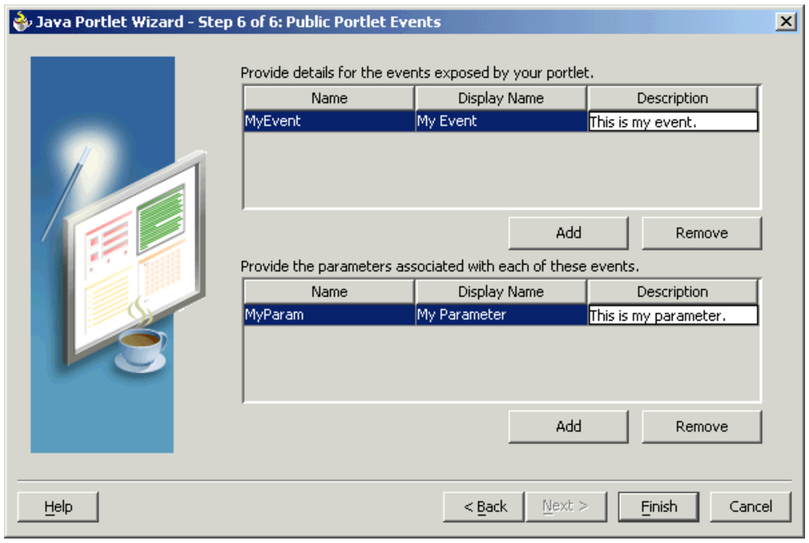
To create an events portlet, perform the following steps:

1. Create a new portlet called `MyEventsPortlet` in the same provider you used for the parameter portlet in Section 7.2.3.2, "Adding Public Parameters"by invoking the Portlet Wizard (Figure 7–6). Go through the wizard as normal. In step 5 of the wizard, create a parameter. In step 6 of the wizard, enter the information shown in Table 7–1.

Table 7–1 Events

Events Area	Name	Display Name	Description
Events Exposed	MyEvent	My Event	This is my event.
Parameters Associated	MyParam	My Parameter	This is my parameter

Figure 7–6 Public Portlet Events Page of Portlet Wizard



Provide details for the events exposed by your portlet.

Name	Display Name	Description
MyEvent	My Event	This is my event.

Provide the parameters associated with each of these events.

Name	Display Name	Description
MyParam	My Parameter	This is my parameter.

The wizard generates the following code in `provider.xml`:

Note: In the following example, notice that the input parameter and the event parameter have the same name, `MyParam`. They are two different parameters, even though they have the same name.

```
<showDetails>false</showDetails>
<inputParameter class="oracle.portal.provider.v2.
  DefaultParameterDefinition">
   <name>MyParam</name>
   <displayName>My Parameter</displayName>
</inputParameter>
<event class="oracle.portal.provider.v2.DefaultEventDefinition">
   <name>MyEvent</name>
   <displayName>My Event</displayName>
   <parameter class="oracle.portal.provider.v2.DefaultParameterDefinition">
     <name>MyParam</name>
     <displayName>My Parameter</displayName>
   </parameter>
</event>
```

```
<renderer class="oracle.portal.provider.v2.render.RenderManager">
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

2. Import the following necessary classes into `MyEventsPortlet`:
   - `oracle.portal.provider.v2.event.EventUtils`
   - `oracle.portal.utils.NameValue`
   - `oracle.portal.provider.v2.url.UrlUtils`
3. In `MyEventsPortlet`, add a link that passes the parameter value to another portlet. As shown in the following sample code, you receive the same page parameter as the previous portlet, but in addition you create a link that passes an event as well:

```
<%@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"
import="oracle.portal.provider.v2.event.EventUtils"
import="oracle.portal.utils.NameValue"
import="oracle.portal.provider.v2.url.UrlUtils"
%>
<%
PortletRenderRequest pReq = (PortletRenderRequest)
request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
%>
<%
  NameValue[] parameters = new NameValue[2];
  parameters[0] = new NameValue( EventUtils.eventName("MyEvent"),"");
  parameters[1] = new
NameValue(EventUtils.eventParameter("MyParam"),pReq.getParameter
  ("MyParam"));
%>
<span class="portletText1"><br>

<a href="<%= UrlUtils.constructLink
  (pReq, pReq.getRenderContext().getEventURL(), parameters , true, true)%>">
The value of the stock is <%= pReq.getParameter("MyParam") %>
</a>
<br><br></span>
```

Note: This sample code does not handle NULL values. When the portlet is initially added to the page, you may receive an error, but, after wiring the portlet to the page parameter, it should work fine.

4. Add the portlet to a different page (in the same page group) than the previous portlet (the Parameter Portlet). Expand the portlet and wire it to receive the same parameter as the previous portlet.

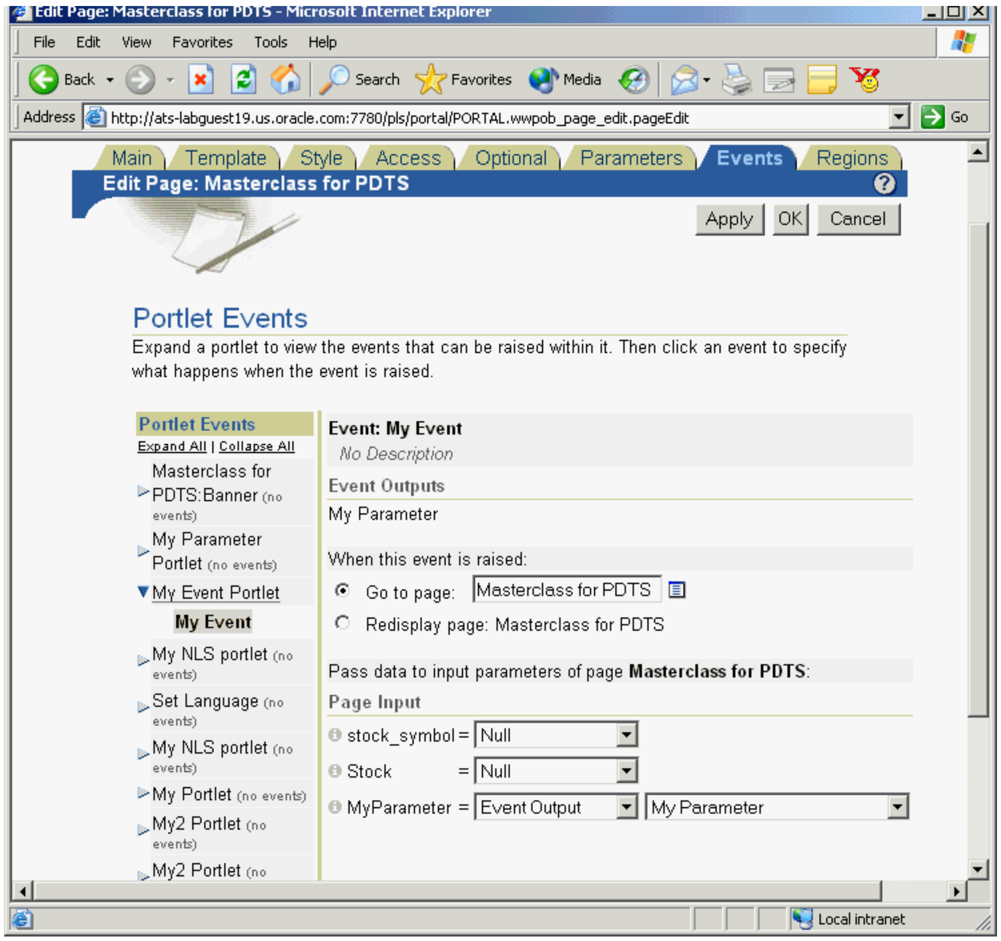
```
My Parameter = Page Parameter MyParameter
```

5. Apply your changes on the Parameter tab and go to the Events tab. Expand the Event portlet and select the event. Select **Go to Page** and find the page to which

you want to pass the event. Choose the page where the Parameter portlet is located. Configure this portlet to pass an event as the page parameter `MyParameter` as shown in Figure 7–7.

```
MyParameter = Event Output MyParameter
```

Figure 7–7   Portlet Events in the Edit Page


Edit Page: Masterclass for PDTS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://ats-labguest19.us.oracle.com:7780/pls/portal/PORTAL.wwpob_page_edit.pageEdit
Go
Main Template Style Access Optional Parameters Events Regions
Edit Page: Masterclass for PDTS
Apply OK Cancel
Local intranet

Portlet Events

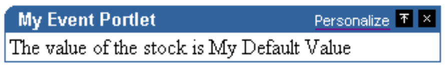
Expand a portlet to view the events that can be raised within it. Then click an event to specify what happens when the event is raised.

Portlet Events
Expand All | Collapse All
Masterclass for PDTS:Banner (no events)
My Parameter Portlet (no events)
My Event Portlet
My Event
My NLS portlet (no events)
Set Language (no events)
My NLS portlet (no events)
My Portlet (no events)
My2 Portlet (no events)
My2 Portlet (no

Event: My Event
No Description
Event Outputs
My Parameter
When this event is raised:
[✓ Go to page: Masterclass for PDTS]
Redisplay page: Masterclass for PDTS
Pass data to input parameters of page **Masterclass for PDTS**:
Page Input
stock_symbol = Null
Stock = Null
MyParameter = Event Output | My Parameter

6. Click **OK** to view the page. Your Event portlet should have a link that displays the value received from the page (Figure 7–8).

Figure 7–8   My Event Portlet Before Parameter Change



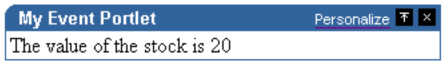
The value of the stock is My Default Value

7. You can append a parameter value to the URL and the portlet displays the value in the link.

```
&MyParameter=20
```

8. When you click the link, that value is passed to the Parameter portlet on its page (Figure 7–9).

Figure 7–9 My Event Portlet After Parameter Change



The value of the stock is 20

7.2.4 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 provider code. In this way, you can specify any property in a provider deployment and then easily access it anywhere in your provider code. PDK-Java provides utilities to enable the retrieval of both provider and non-provider JNDI variables within a J2EE container. To use JNDI variables, you need to perform the following tasks:

- Section 7.2.4.1, "Declaring JNDI Variables"
- Section 7.2.4.2, "Setting JNDI Variable Values"
- Section 7.2.4.3, "Retrieving JNDI Variables"

7.2.4.1 Declaring JNDI Variables

You declare JNDI variables in the `web.xml` file for your provider. 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.

7.2.4.1.1 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.

7.2.4.1.2 Variable Naming Conventions The PDK-Java defines a number of environment variables that can be set at the individual provider service level or at the Web application level. To avoid naming conflicts between different provider services or different application components packaged in the same Web application, we recommend you devise some naming convention.

Note: If you use the `EnvLookup` method, you must use `oracle/portal/provider/service/property`. You cannot substitute your own company name or component in this case.

For example:

- Provider service specific names should be of the form:

  ```
  {company}/{component name}/{provider name}/{variable name}
  ```

- Shared names should be of the form:

  ```
  {company}/{component name}/{provider 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 provider is associated.
- `{provider name}` is the service name of the provider.
- `{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 allow you to retrieve variables in this form without hard coding the service name of the provider into your servlets or JSPs. The service name need only be defined in the provider's WAR file. Refer to Section 7.2.4.3, "Retrieving JNDI Variables" for more information on retrieving JNDI variables.

7.2.4.1.3 Examples The following examples illustrate provider variable names:

```
oracle/portal/myProvider/myDeploymentProperty
oracle/portal/myprovider/myProperties/myProperty
```

The following example illustrates non-provider variable names:

```
oracle/portal/myOtherProperty
```

7.2.4.2 Setting JNDI Variable Values

In your provider deployment, you may want to set a new value for some or all of your JNDI variables. You can perform this task by setting the values manually in a Oracle WebLogic Server deployment plan as follows:

1. Go to the provider deployment in the Oracle WebLogic Administration Console, and create a new Deployment Plan if one hasn't been set against it.
2. Edit the Deployment Plan XML file. For each deployment property you want to set, add the following variable definition directly under the <deployment-plan> tag:

   ```
   <variable-definition>
         <variable>
           <name>jndi_var_def</name>
           <value>false</value>
         </variable>
     </variable-definition>
   ```

3. To tie this variable definition to and actual JNDI variable that you want to set, do the following for each property under the WEB-INF/web.xml module descriptor (oracle/portal/sample/rootDirectory is used as an example):

   ```
   <module-descriptor external="false">
         <root-element>web-app</root-element>
         <uri>WEB-INF/web.xml</uri>
         <variable-assignment>
           <name>jndi_var_def</name>
   <xpath>/web-app/env-entry/[env-entry-name=&quot;oracle/portal/sample/rootDirect
   ory&quot;]/env-entry-value</xpath>
         </variable-assignment>
       </module-descriptor>
   ```

4. Save the file.
5. Select 'Update' on the provider deployment to apply the Deployment Plan for the new settings to take effect.

7.2.4.3 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 7.2.4.1.2, "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 provider variables and the other retrieves non-provider 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 provider variable:

```
EnvLookup el = new EnvLookup();
String s = (String)el.lookup(myProviderName, myVariableName);
```

`myProviderName` represents the service name for your provider, which makes up part of the variable name. `myVariableName` represents the portion of the variable name that would come after the provider's service name. The example assumes the variable being retrieved is of type `java.lang.String`.

To retrieve a non-provider 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 7–2 shows the JNDI variables provided by default with PDK-Java. If you do not declare these variables, PDK-Java looks for their values in their original locations (`web.xml` and the deployment properties file).

Table 7–2 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 provider'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`	Provider's HMAC time difference.
`oracle/portal/provider/<service_name>/resourceUrlKey`	Authentication key for resource proxying through the Parallel Page Engine. Refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* for more information.
`oracle/portal/provider/`*`provider_name`*`/rootDirectory`	Location for provider personalizations. No default value.
`oracle/portal/provider/`*`provider_name`*`/sharedKey`	HMAC shared key. No default value.
`oracle/portal/provider/`*`provider_name`*`/showTestPage`	(non-provider) A boolean flag that determines if a provider'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. Refer to Section 7.2.8.3.2, "Disabling Export/Import of Personalizations" for more information.

7.2.5 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 takes place simultaneously with a refresh of the page, 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 allows an individual portlet to complete a transaction, such as updating a value, before allowing the page rendering to take place. 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, you could use the `form` tag and transaction tokens to avoid submitting the same parameter twice. Refer to Section 7.3, "Building Struts Portlets with Oracle JDeveloper" 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 Show 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 Show 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 Show 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 the PDK 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, we overload Edit mode to handle both the data submission and the portlet's rendering for personalizations. Note that any of the other full page Show modes (Edit, 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, 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**. Refer to `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. Since 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, 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.

7.2.6 Accessing Session Information

When a user accesses any portal page, Oracle Portal initiates a public unauthenticated session and maintains a cookie to track information about the session across requests. If the user logs in to Oracle Portal, this session becomes an authenticated session of the logged-in user. This portal session terminates when the 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, Oracle Portal contacts all of the providers that contribute portlets to the page, if they specify during registration that they get called for some special processing. This call allows providers to do processing based on the user session, log the user in the provider's application if needed, and establish provider sessions in Oracle Portal. For Database providers, this call is referred to as `do_login` and for Web providers it is `initSession`. Since most Web-enabled applications track sessions using cookies, this API call allows the provider of the application to return cookies.

You can utilize 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, 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).

Before you implement session storage, you should carefully consider the performance costs. Because portlets and providers 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.

If scalability is an important concern for you, a stateful application may cause you problems. Stateful applications can impact the load-balancing and failover mechanism for your Oracle Portal configuration. Even though you may deploy multiple middle-tiers accessing the same Oracle Portal instance, 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, impacting 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.

7.2.6.1 Assumptions

To perform the tasks in this section, we are making the following assumptions:

1. You have followed through and understood Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

7.2.6.2 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 from Oracle Portal, 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 provider 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.

More On OTN

For more detailed information on the PDK Framework classes, refer to 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 provider session.
- Read and write the session by accessing it from within your Java portlet.
- Set the session to true in `provider.xml`.

- Register the provider 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, you display the value, increment the count, and store the new value. If the session is valid but no previously stored value exists, 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, 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 provider to receive session information from the portal. You should only set this tag to true if you are using session information in

your provider or portlets. By setting this flag to true, extra load is added to the provider calls.

```
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
<session>true</session>
```

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

4. Register the provider in Oracle Portal. Ensure that you select the **User** radio button and choose a **Login Frequency** of **Once Per Session** on the Define Connections page of the wizard. For a reminder on how to register your portlet, refer to Section 6.5.5, "Registering and Viewing Your Oracle PDK-Java Portlet".

7.2.6.3 Viewing the Portlet

If you have not already added your Java portlet to a page, do so now. Ensure that you perform the following tasks:

- Set your provider to **Once per User Session** for the login frequency value.
- Refresh the provider to accept the new changes.
- Re-login in case your session is no longer valid.

7.2.7 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, refer to 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`

7.2.7.1 Assumptions

To perform the tasks in this section, we are making the following assumptions:

1. You have followed through and understood Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

7.2.7.2 Introduction to Portlet Security Features

This section introduces the major features that are available to secure your portlet providers.

7.2.7.2.1 Authentication When a user first logs in to an Oracle Portal instance, they must enter their password to verify their identity and obtain access. This authentication is performed by OracleAS Single Sign-On Server server. Refer to Section 7.2.7.3, "Single Sign-On" for more information.

Once the user’s identity is passed to the provider in shown requests, the provider 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();
```

Refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* for more information.

7.2.7.2.2 Authorization Authorization determines if a particular user may view or interact with a portlet. Oracle Portal provides the following two types of authorization checking:

- **Portal Access Control Lists (ACLs):** After a user is authenticated by OracleAS Single Sign-On Server, Oracle Portal uses ACLs to determine what privileges that user has to perform actions on portal objects, such as folders and portlets. The actions available to a user can range from simply viewing an object to performing administrative functions on it. If a user does not belong to a group that has been granted a specific privilege, Oracle Portal prevents that user from performing the actions associated with that privilege. Refer to Section 7.2.7.4, "Oracle Portal Access Control Lists (ACLs)" for more information.
- **Programmatic Portlet Security:** You can also implement your own security manager programmatically. Refer to Section 7.2.7.5, "Portlet Security Managers" for more information.

7.2.7.2.3 Communication Security To this point, we have covered user authentication and authorization, which do not check the authenticity of messages received by a provider. The following measures can be used to properly secure communication between a portal and a web provider:

- **Oracle Portal Server Authentication** restricts access to a provider to a small number of recognized machines. 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, the message is passed to the provider. If not, it is rejected before reaching the provider. Refer to Section 7.2.7.6, "Oracle Portal Server Security" for more information.
- **Message Authentication** uses a shared key to assert the Portal client identity and to prevent message tampering. Refer to Section 7.2.7.7, "Message Authentication" for more information.
- **Message Encryption** encrypts message contents. Refer to Section 7.2.7.8, "HTTPS Communication" for more information.
- **User Input Escape** causes Oracle Portal 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. Refer to Section 7.2.7.10, "User Input Escape" for more information.

For more information about communication security, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

7.2.7.3 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 expose application functionality directly in the portal or provide deep links that take you to the application itself to perform a task.

An application may need to authenticate the user accessing the application through the portlet. Following are the possible application authentication methods:

- Section 7.2.7.3.1, "Partner Application". In this case, the application user is the same authenticated user used by Oracle Portal.
- Section 7.2.7.3.2, "External Application". In this case, the Oracle Portal user is different from the application user, but the application user name and password are managed by the Oracle Portal user.
- Section 7.2.7.3.3, "No Application Authentication". In this case, the communication between provider and Oracle Portal is not protected at all.

For more information about Single Sign-On, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

7.2.7.3.1 Partner Application A partner application is an application that shares the same OracleAS Single Sign-On Server as Oracle Portal for its authentication. Thus, when a user is already logged in to Oracle Portal, their identity can be asserted to the partner application without them having to log in again.

Partner applications are tightly integrated with OracleAS Single Sign-On Server. When a user attempts to access a partner application, the partner application delegates the authentication of the user to OracleAS Single Sign-On Server. Once a user is authenticated (that is, has provided a valid user name and password) for one partner application, the user does not need to provide a user name or password when accessing other partner applications that share the same OracleAS Single Sign-On Server instance. OracleAS Single Sign-On Server determines that the user was successfully authenticated and indicates successful authentication to the new partner application.

The advantages of a partner application implementation are as follows:

- Provides the tightest integration with Oracle Portal and OracleAS Single Sign-On Server.
- Provides the best single sign-on experience to users.
- Provides the most secure form of integration because user names and passwords are not transmitted between Oracle Portal and the provider.

The disadvantages of a partner application implementation are as follows:

- The application must share the same user repository as Oracle Portal even though the application's user community may be a subset of the Oracle Portal user community. While worth some consideration, this issue is a minor one because the portal pages that expose the application can be easily restricted to the application's user community.
- The application can only be tightly integrated to one or more OracleAS Single Sign-On Server instances if they share the same user repository.
- The application must be written such that it delegates authentication to OracleAS Single Sign-On Server.
- You must have access to the application source code.

7.2.7.3.2 External Application An external application uses a different authentication server than Oracle Portal. The application may use a different instance of OracleAS Single Sign-On Server than that used by Oracle Portal or some other authentication method. However OracleAS Single Sign-On Server does store the user name and password of the external application for that user. This means that when a user is already logged into Oracle Portal, they will be logged into the external application without having to type in their user name or password.

Applications that manage the authentication of users can be loosely integrated with OracleAS Single Sign-On Server if the administrator registers them as external applications. When a user who was previously authenticated by OracleAS Single Sign-On Server accesses an external application for the first time, OracleAS Single Sign-On Server 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, OracleAS Single Sign-On Server 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, OracleAS Single Sign-On Server maps the new user name and password to the user's Oracle Portal 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:

- Allows integration with many portals. If, however, one of the portals is preferred over the others, 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.
- Allows integration with multiple portals independent of their user repositories and OracleAS Single Sign-On Server.
- 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 provider in plain text, unless you implement SSL.

7.2.7.3.3 No Application Authentication The provider trusts the Oracle Portal instance sending the request completely. The provider 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 Oracle Portal.

7.2.7.4 Oracle Portal Access Control Lists (ACLs)

When users log in to an Oracle Portal instance, they are authenticated by an OracleAS Single Sign-On Server instance. Having verified their identity, Oracle Portal uses ACLs to determine whether they are authorized to access particular portlets and add them to pages from the Portlet Repository.

Oracle Portal ACLs operate according to the following security characteristics:

- **Privileges** define the actions that can be performed on the object to which they are granted. Privileges include actions such as Manage and Execute.

- **Oracle Portal users and their privileges** are granted from the Administer tab of the Builder.
- **Oracle Portal user groups** are administered from the Administer tab of Oracle Portal Builder. Membership in the groups and privileges granted to the groups are all defined and maintained here. A privilege granted to a user group is inherited by all the users of that group.
- **Provider privileges** apply to the provider and all of its portlets. Provider ACLs are administered on the Provider tab of the Oracle Portal Navigator.
- **Portlet privileges** can override the privileges set for the provider of the portlet. Portlet ACLs are administered from the Provider tab of the Oracle Portal Navigator. Clicking **Open** for a provider takes you to a page that manages the portlets of the provider.

For more information on the available privileges for objects, users, and user groups in Oracle Portal, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

The advantages of ACLs are as follows:

- ACLs offer a simple, yet powerful, mechanism to secure Oracle Portal objects.
- Central management of user group membership simplifies the management of ACLs because it negates the necessity of modifying the ACLs associated with each object.

The disadvantages of ACLs are as follows:

- ACLs are applied at the provider or portlet level. You cannot vary the security rules for a portlet depending on the page where you place it.

7.2.7.5 Portlet Security Managers

Portlet security managers are implemented within a provider 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 provider 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 provider, then any user name may be passed in, even fictitious, unauthenticated ones.

A provider can implement two portlet security methods as follows:

- Get a list of portlets.
- Verify the accessibility of the portlet.

Portlets have access to the Oracle Portal 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 providers)

`AuthLevelSecurityManager` has access to the following information about authorization level:

- Strongly authenticated.

  The user has been authenticated by OracleAS Single Sign-On Server in the current Oracle Portal 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 Oracle Portal 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 Oracle Portal 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 provider.

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

The advantages of security methods are as follows:

- You can enable a portlet to produce different output depending on the level of authorization.

The disadvantages of security methods are 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 an Oracle Portal instance.

7.2.7.5.1 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 an Oracle Portal instance with privileges to create pages and add portlets to a page.
2. Create a new portal 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 Portal page.
5. Now log out of the Portal instance by clicking the **Logout** link.
6. Directly access the Portal 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 runtime detected this and allowed 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 allow rendering of the portlet's contents.

If you log in again and view the page, you will see that the portlet is still there.

7.2.7.5.2 Implementing Your Own Security Manager If your portlet requires special security arrangements which are not provided by the implementations shipped with the PDK, 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 provider definition with you new class name and configure child elements appropriately.

7.2.7.6 Oracle Portal Server Security

One way to prevent unauthorized access to providers is to restrict access to the provider to known client machines at the server level. Because only the identified clients may access the provider, this method defends against denial of service attacks.

In Oracle Application Server, you use the allow and deny directives in the `httpd.conf` file to control access to client machines based on their host names or IP addresses. If host names are used as discriminators, the server needs to look them up on its Domain Name Server (DNS), which adds extra overhead to the processing of each request. Using the IP address circumvents this problem, but the IP address of a remote client may change without warning.

The advantages of server security are as follows:

- It limits access to the provider 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 provider, 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 provider containing fake IP addresses and host names. This trick is difficult to perform effectively since return messages go to the machine whose IP address was copied, but it can still cause problems.

For more information on this topic, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

7.2.7.7 Message Authentication

Message authentication uses a shared key known to the client (Portal instance) and provider to restrict access to known clients. This may be used in SSL communication with a provider instead of client certificates.

Oracle Portal sends a digital signature, calculated using a Hashed Message Authentication Code (HMAC) algorithm, with each message to a provider. A provider may authenticate the message by checking the signature using its own copy of the shared key. This technique may be used in Secure Socket Layer (SSL) communication with a provider instead of client certificates.

A single provider 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, the provider administrator has to create a new key and distribute it to all of the Oracle Portal clients, who then must update their provider definitions. The way around this problem is to deploy different provider services, specifying a unique shared key for each service. Each provider service has its own deployment properties file so that each service is configured independently of the others. The overhead of deploying multiple provider services within the same provider adapter is relatively small.

In a provider without Oracle Web Cache in front of it, this use of the same signature cookie over the lifetime of a provider 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 provider. The shorter the provider 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 provider 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 provider open to show requests being illegally trapped and resent to the provider.

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, you should use message authentication in conjunction with SSL.

The advantages of message authentication are as follows:

- Ensures that the message received by a provider comes from a legitimate Oracle Portal instance.

The disadvantages of message authentication are as follows:

- Causes administration problems if a provider serves more than one portal.
- Entails performance implications if made very secure by having a short session timeout.

For more information on this topic, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Credential Store Share Key

The Producer's shared key is stored in the credential store. To store a shared key into the credential store do the following:

Create the Credential

To create the credentials, run the following WLST command:

```
createCred(map='PDK', key='pdk.omniPortlet.sharedKey', user='sharedKey',
password='1234567890abc')
```

Grant PDK Java Code Access to the Credential Store

To grant pdk java code access to the credential store permission :

```
grantPermission(appStripe=None,principalClass=None,principalName=None,codeBaseURL=
'file:${domain.home}/servers/WLS_Portal/tmp/_WL_
user/-',permClass='oracle.security.jps.service.credstore.CredentialAccessPermissio
n',permTarget='context=SYSTEM,mapName=PDK,keyName=*',permActions='read')
```

7.2.7.8 HTTPS Communication

Normal communication between Oracle Portal and a provider uses HTTP, a network protocol that transmits data as plain text using TCP as the transport layer. HTTPS uses an extra secured layer (SSL) on top of TCP to secure communication between a client and a server, making it difficult to intercept and read messages.

Each entity (for example, an Oracle Web Cache instance) receiving a communication using SSL has a freely available public key and a private key known only to the entity itself. Any messages sent to an entity are encrypted with its public key. A message encrypted by the public key may only be decrypted by the private key so that, even if a message is intercepted by a felonious third party, it cannot be decrypted.

Certificates used to sign communications ensure that the public key does in fact belong to the correct entity. These are issued by trusted third parties, known as Certification Authorities (CA). They contain an entity's name, public key, and other security credentials and are installed on the server end of an SSL communication to verify the identity of the server. Client certificates may also be installed on the client to verify the identity of a client.

Oracle Wallet Manager manages public key security credentials. It generates public and private key pairs, creates a certificate request to a CA, and installs the certificate on a server.

For more information on this topic, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Configuration of SSL

When a provider is registered from an Oracle Portal instance, only one URL is entered, which means either HTTP or HTTPS may be used but not both.

Each port on each server that may be used to receive SSL messages must have a server-side certificate installed (that is, an OracleAS Web Cache instance) in front of the Web provider and the server that hosts the provider. The certificate installed on a server port ensures that communication between two points is encrypted but does not authenticate the source of a message. Message authentication should be used as well to fully secure communication between a trusted Oracle Portal instance and a provider.

For more information about SSL configuration for Oracle Portal, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

7.2.7.9 LDAP (Oracle Internet Directory) Security

PDK-Java uses Portlet Security Managers for LDAP (Oracle Internet Directory) security. PDK-Java uses Oracle Internet Directory as a repository of users, groups, and permissions. It retrieves information about the logged-in user and determines whether the user has the required permissions to view the portlet and data within the portlet. By enabling Oracle Internet Directory security, your providers perform the following:

- Secure portlets based on groups.
- Restrict access to the administrative functions of your portlets (using your own security manager).
- Retrieve all of the user property information stored in the Oracle Internet Directory including first name, last name, title, e-mail, telephone number, groups, and photo.
- Create users and groups for Oracle Portal.

By default, Oracle Internet Directory security is disabled. You must make a change in the deployment properties file for a specific provider to enable this feature. Enabling and using Oracle Internet Directory to secure your portlets can be done quickly and easily. To do this, perform the following steps:

1. Enable the Oracle Internet Directory manager in the deployment properties files (`provider_name.properties`).

   ```
   oidManager=true
   oidAdminClass=class_that_extends_oracle.portal.provider.v2.oid.OidInfo
   ```

2. Provide the connection information for Oracle Internet Directory by extending the simple class called `OidInfo`.

3. Provide a list of groups that can view your portlet in the provider definition file.

   ```
   <group>cn=group1,cn=groups,dc=us,dc=oracle,dc=com</group>
   ```

   Your provider connects to Oracle Internet Directory using the information provided to the `OidInfo` class by you. The portlet accesses Oracle Internet Directory using the credentials provided (for example, user name and password) and performs the specified tasks. We recommend that you create an Oracle Internet Directory user specifically for your provider connection with the minimum set of privileges needed to complete the tasks requested by your portlets. For example, if your portlet only checks group information, do not connect to the Oracle Internet Directory as an administrator.

7.2.7.9.1 Implementing Oracle Internet Directory Security PDK-Java provides a set of default classes specifically for Oracle Internet Directory integration. These classes handle the connection from your portlets to Oracle Internet Directory, enable your portlets to be secured based on Oracle Portal groups, and provide access to user property information from within Oracle Internet Directory. The classes used by your Web provider for Oracle Internet Directory integration are as follows:

- `oracle.portal.provider.v2.oid.OidInfo` receives the Oracle Internet Directory connection information provided by the developer and connects to Oracle Internet Directory. When building your own portlets, you should extend this class to send secure connection details from the provider to Oracle Internet Directory.
- `oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo` is an extension of `OidInfo` and provides an easy way to test portlet security. This class is used by the Oracle Internet Directory samples in PDK-Java and parses the deployment properties file for the Oracle Internet Directory connection information (seen subsequently). This class should be used only for testing and development, it is not safe to use in a production scenario.
- `oidManager` is set to false by default. It must be set to true in `provider_name.properties` to enable Oracle Internet Directory. (If you have only one provider in your Web application, ensure that `provider_name.properties` is identical to `_default.properties`.) For example:

  ```
  serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
  loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
  showTestPage=true
  definition=providers/lab_provider/provider.xml
  autoReload=true
  oidManager=true
  oidAdminClass=oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo
  oidHost=myhost.mydomain.com
  ```

```
oidPort=oidPort
oidUser=oidUser
oidPasswd=oidPassword
```

- `oidAdminClass` is set to the class that extends `OidInfo`. PDK-Java provides `UnsafeOidInfo` by default, but as the name suggests, this class should not be used in production scenarios.
  - `oidHost` is the machine where Oracle Internet Directory is hosted.
  - `oidPort` is the port used by the Oracle Internet Directory.
  - `oidUser` is the Oracle Internet Directory account.
  - `oidPasswd` is the Oracle Internet Directory password.

  For example:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/lab_provider/provider.xml
autoReload=true
oidManager=true
oidAdminClass=oracle.portal.sample.v2.devguide.oid.UnsafeOidInfo
oidHost=myhost.mydomain.com
oidPort=oidPort
oidUser=oidUser
oidPasswd=oidPassword
```

- `oracle.portal.provider.v2.security.GroupSecurityManager` manages which groups have access to your provider and its portlets. It retrieves this information from the provider definition file and is portlet specific. Each portlet in a provider may have different group settings. There is no limit on the number of groups that can be set using this tag, but, since the Web provider parses and validates each group in turn, listing many groups may degrade performance.
- `<group>` is the tag in `provider.xml` that handles group management. It lists the groups allowed to access the portlet. The group information here follows the same case sensitivity as the Oracle Internet Directory.

Note: The following example refers to your *`portal_instance_id`*, which is specific to your installation. To find your instance identifier, refer to your *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory*.

```
<securityManager class="oracle.portal.provider.v2.security.
    GroupSecurityManager">
      <group>cn=DBA,cn=portal_instance_id,cn=groups,
             dc=us,dc=oracle,dc=com</group>
</securityManager>
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

The advantages of Oracle Internet Directory security are as follows:

- Offers a simple, powerful way to secure your portlets.

- Secures data within your portlets based on the user's group membership.
- Creates users and groups directly from your portlets exposed as Web providers.

The disadvantages of Oracle Internet Directory security are as follows:

- Slightly degrades performance when authorizing your portlet through Oracle Internet Directory. There is a cost associated with obtaining group information from any LDAP server, but this cost only happens the first time a user accesses a portlet in a session.
- Requires provider access to Oracle Internet Directory.
- Assumes all Oracle Portal instances served by the provider use the same Oracle Internet Directory instance.

More On OTN

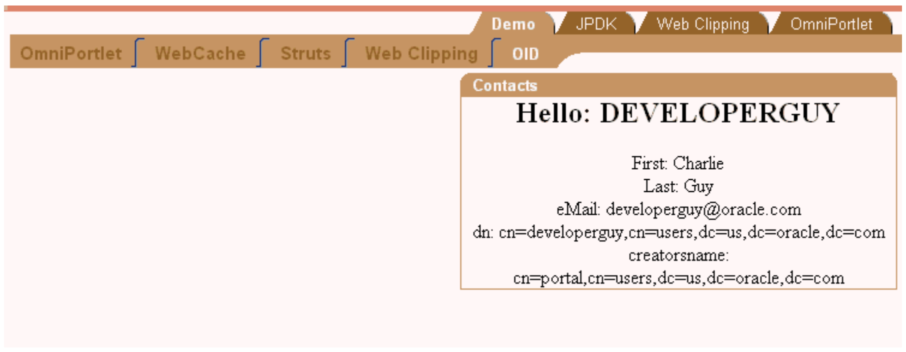
For more information on securing your providers using Oracle Internet Directory or to set up the sample portlets secured using Oracle Internet Directory, review the technical note, *Installing the Oracle Internet Directory Portlets* on OTN.

7.2.7.9.2 Viewing Your Portlets After you secure your provider with Oracle Internet Directory, you can validate its behavior by following these steps:

1. Ensure you are logged in to an Oracle Portal instance as a user who is a member of the group specified in the `<group>` tag in `provider.xml`.
2. Use an existing page or create a new one, 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. Click **Logout**.
6. Directly access the page by entering the URL noted in Step 4 in your browser's address bar or login to Oracle Portal using a user that is not part of the group listed in `provider.xml`.

You will see the page created in Step 2 but not the portlet added in Step 3, as shown in Figure 7–10. When you added the portlet to the page, you were logged in as a user authorized to view the portlet. The PDK runtime detected this and allowed you to add the portlet. When you logged out and viewed the page, you were no longer part of the group allowed to view the portlet and hence the PDK Framework did not allow rendering of the portlet's contents.

Figure 7–10 Page and Portlets for Developer

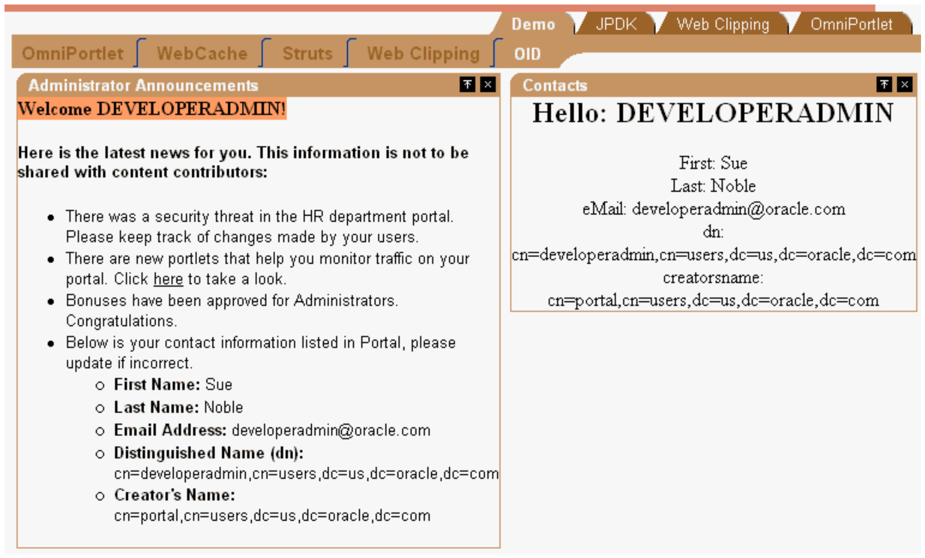


Hello: DEVELOPERGUY

First: Charlie
Last: Guy
eMail: developerguy@oracle.com
dn: cn=developerguy,cn=users,dc=us,dc=oracle,dc=com
creatorsname:
cn=portal,cn=users,dc=us,dc=oracle,dc=com

If you log in again and view the page, you will see that the portlet is still there (Figure 7–11).

Figure 7–11 Page and Portlets for Developer/Administrator



Welcome DEVELOPERADMIN!

Here is the latest news for you. This information is not to be shared with content contributors:

- There was a security threat in the HR department portal. Please keep track of changes made by your users.
- There are new portlets that help you monitor traffic on your portal. Click here to take a look.
- Bonuses have been approved for Administrators. Congratulations.
- Below is your contact information listed in Portal, please update if incorrect.
  - **First Name:** Sue
  - **Last Name:** Noble
  - **Email Address:** developeradmin@oracle.com
  - **Distinguished Name (dn):** cn=developeradmin,cn=users,dc=us,dc=oracle,dc=com
  - **Creator's Name:** cn=portal,cn=users,dc=us,dc=oracle,dc=com

Hello: DEVELOPERADMIN

First: Sue
Last: Noble
eMail: developeradmin@oracle.com
dn:
cn=developeradmin,cn=users,dc=us,dc=oracle,dc=com
creatorsname:
cn=portal,cn=users,dc=us,dc=oracle,dc=com

7.2.7.10 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, an attacker might attempt to pass scripts or commands to the portlet through the title string. Oracle Portal provides the following features to ensure that you can protect your portlets from such attacks:

- Section 7.2.7.10.1, "Default Container Encoding"
- Section 7.2.7.10.2, "Escape Methods"

7.2.7.10.1 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, you can set `escapeStrings` to `false` for that provider.

`escapeStrings` applies to all logical providers within a Web provider. You can set the value of `escapeStrings` from the WebLogic Server Administration Console as you would any other JNDI variable. Refer to Section 7.2.4.2, "Setting JNDI Variable Values" for more information.

7.2.7.10.2 Escape Methods If you have code that renders customized values, 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. Oracle Portal 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());
```

7.2.8 Controlling the Export/Import of Portlet Personalizations

The export/import facility of Oracle Portal is a multi-purpose tool for moving your portal objects, such as portlets, between instances of Oracle Portal. For example, you might use export/import to move objects from a development environment to a stage environment and then, finally, to a production environment. You might also use export/import to move pages and page groups between Oracle Portal instances, or to move Web providers from one machine to another. For more information about export/import in general, please refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Because portlet default settings can be set by the administrator and then changed by the user, they require some special consideration when you import and export them. To simplify the transport process, Oracle Portal provides default functionality that handles administrator personalization data (that is, data created using Edit Defaults mode) for you. When a portlet is exported, the default personalization data stored using PDK-Java's `PreferenceStore` mechanism is exported with the portlet by default. Hence, when the portlet is imported into a target instance of Oracle Portal, this data is imported along with it. As a result, the portlet instance's default settings are maintained when the portlet is moved from one portal instance to another.[1]

The aforementioned behavior is provided to you as a convenience and it requires no action on your part to leverage. You might, however, want to exercise more granular control over the export of personalization data than that provided by the default functionality. To implement your own requirements for export/import, you can make use of the programming interface to augment or override the default handling of personalizations.

If you use the PDK-Java preference store mechanism, the export/import of your Edit Default personalizations is built-in and requires no additional effort on your part. This default export/import of administrator personalizations relies on the PDK-Java preference store. If you have created your own preference store mechanism (for example, a file or database preference storage system), then you also must implement your own export/import support that performs the following functions:

- Exports personalizations. This functionality must at least export administrator personalizations, but it could optionally include user personalizations, too.
- Imports personalizations. Note that this functionality must reflect whatever you implemented for export. For example, if you allow the export of both administrator and user personalizations, then the import functionality must support both as well.

The export/import functionality for personalizations requires that your Oracle Portal instance and provider are on Release 10.1.2. Export/import of personalizations behaves the same regardless of the location of your provider, which can be either of the following:

- in the default Oracle Containers for Java EE of the Oracle Application Server, where the Oracle Portal instance is different.

[1] User personalization data for Oracle Portal objects is never exported. This restriction applies to portlets as well as other objects, such as pages.

- in a separate Oracle Containers for Java EE, where the Oracle Portal instance may be different, and the provider is the same but is not registered on the target Oracle Portal instance.

7.2.8.1 Import/Export Programming Interface

The PDK-Java's preference store mechanism allows data to be persisted by any number of application entities. The following three entities are the ones that persist data for the purposes of export/import:

1. The *portlet instance* is the portlet on a page with the default personalizations made to it by the administrator. The API for the portlet instance is as follows:
   - oracle.portal.provider.v2.PortletInstance
     - exportData

       ```
       public byte[] exportData
           (
               boolean exportUsers,
               String[] userNames,
               TransportLogger logger
           )
           throws PortletException
       ```

     - importData

       ```
       public void importData
           (
               byte[] data,
               TransportLogger logger
           )
           throws PortletException
       ```

2. The *portlet definition* is the base portlet without any personalizations applied to it. You might think of the portlet definition as the version of the portlet that exists in the Portlet Repository before it is placed on a particular page for use. The API for the portlet definition is as follows:
   - oracle.portal.provider.v2.PortletDefinition
     - exportData

       ```
       public byte[] exportData
           (
               ProviderInstance pi,
               boolean exportUsers,
               String[] userNames,
               TransportLogger logger
           )
           throws PortletException
       ```

     - importData

       ```
       public void importData
           (
               ProviderInstance pi,
               byte[] data,
               TransportLogger logger
           )
           throws PortletException
       ```

3. The *provider instance* is the entity that contains and communicates with a set of portlets. The API for the provider instance is as follows:

   - oracle.portal.provider.v2.ProviderInstance

     – exportData

     ```
     public byte[] exportData
         (
             boolean exportUsers,
             String[] userNames,
             TransportLogger logger
         )
         throws ProviderException
     ```

     – importData

     ```
     public void importData
         (
             byte[] data,
             TransportLogger logger
         )
         throws ProviderException
     ```

By default, each of these entities employs an instance of `oracle.portal.provider.v2.transport.PrefStoreTransporter` to transform the data from an `oracle.portal.provider.v2.preference.PreferenceStore` to a byte array for transport. For the default export/import behavior, though, only the portlet instance entity's personalization data is exported and imported. If you have persisted data at the portlet definition or provider instance level, you may want to export that data as well. For example, a billing handle that you persisted at the `ProviderInstance` level may need to be exported.

To change the behavior of `PrefStoreTransporter`, you can override its default implementation. The example in Section 7.2.8.3.7, "Exporting by Reference Example" illustrates how you can override `PrefStoreTransporter`.

Logging Interface

To simplify troubleshooting of your export/import transactions, you can send messages to both the calling Oracle Portal instance and the Web provider log. PDK-Java provides a transport logging class that enables you to add events to the log during export and import operations. In this way, you can better keep track of events that occur during the transport of portlet personalizations. The log can be a valuable troubleshooting tool if you encounter unexpected behavior in your portlets during or after transport. For example, you can log events when incompatibilities between PDK-Java versions are found.

You log events using the logger object, an instance of the `oracle.portal.provider.v2.transport.TransportLogger` class provided for each of the methods mentioned earlier. You log events with the calling portal through the instance provided for each method. You record events in the Web provider log with the normal logging mechanism, `oracle.portal.log.LogManager`. The log levels for export/import are as follows:

- `TransportLogger.SEVERITY_INFO`
- `TransportLogger.SEVERITY_WARNING`
- `TransportLogger.SEVERITY_ERROR`

7.2.8.2 Exporting Personalizations Example

This example illustrates the most basic case where you build a portlet and accept the default behavior for the export of personalizations. In the examples in Section 7.2.8.3.6, "Encrypting Personalization Data Example" and Section 7.2.8.3.7, "Exporting by Reference Example", you will see how to enhance the security of your personalizations during export and import. To implement the more basic form of exporting personalizations, do the following:

1. Create a stock portlet and implement the Show mode with the following `MyStockPortletShowRenderer.java` class. Note that this class does not incorporate any special code to enable export/import.

```
package oracle.portal.sample.v2.devguide.tx;
import java.util.StringTokenizer;
import oracle.portal.provider.v2.PortletException;
import oracle.portal.provider.v2.personalize.NameValuePersonalizationObject;
import oracle.portal.provider.v2.render.PortletRenderRequest;
import oracle.portal.provider.v2.render.PortletRendererUtil;
import oracle.portal.provider.v2.render.http.BaseManagedRenderer;
import java.io.PrintWriter;
import oracle.portal.sample.v2.devguide.webservices.
    NetXmethodsServicesStockquoteStockQuoteServiceStub;
public class MyStockPortletShowRenderer extends BaseManagedRenderer
{
    private String pid = null;
    private String userdata;
    private String stockList;
    private String stockCode;
  public void renderBody(PortletRenderRequest request) throws PortletException
  {
    // Use the PrintWriter from the PortletRenderRequest
    PrintWriter out = null;
    NetXmethodsServicesStockquoteStockQuoteServiceStub ns = new
      NetXmethodsServicesStockquoteStockQuoteServiceStub();
    try
    {
      out = request.getWriter();
      NameValuePersonalizationObject data = null;
      data = (NameValuePersonalizationObject)PortletRendererUtil.
         getEditDefaultData(request);
      stockList= data.getString("stock");
      if(stockList!=null) {
        StringTokenizer st = new  StringTokenizer(stockList,",");
        out.println("<table border='0'>");
        out.println("<thead>");
        out.println("<tr>");
        out.println("<th width='20%'>");
        out.println("<p align='left'> Stock Code</p></th><th width='20%'>");
        out.println("<p align='left'> Quote</p>");
        out.println("</th>");
        out.println("</tr>");
        out.println("<thead>");
            while(st.hasMoreElements()) {
              stockCode= st.nextElement().toString();
              out.println("<tr>");
              out.println("<td width='20%'>");
              out.println("<p align='left'>"+  stockCode +
                 "</p></td><td width='20%'>");
              out.println(ns.getQuote(stockCode));
              out.println("</td>");
            out.println("</tr>");
          }
        out.println("</table>");
      }
      else
      {
       out.println("<br> Click <b>Edit Defaults</b> to define stock codes.");
      }
    }
    catch(Exception ioe)
    {
      throw new PortletException(ioe);
    }
  }
}
```

2. Implement the Edit Defaults mode for your stock portlet with the following class, `MyStockPortletEditDefaultsRenderer.java`. This class enables the administrator to make and store default personalizations, which are then exported according to the default behavior.

```
package oracle.portal.sample.v2.devguide.tx;
import oracle.portal.provider.v2.PortletException;
import oracle.portal.provider.v2.http.HttpCommonConstants;
import oracle.portal.provider.v2.render.PortletRenderRequest;
import oracle.portal.provider.v2.render.http.BaseManagedRenderer;
import oracle.portal.provider.v2.render.PortletRendererUtil;
import oracle.portal.provider.v2.personalize.NameValuePersonalizationObject;
import java.io.PrintWriter;
import java.io.IOException;
import oracle.portal.provider.v2.render.http.HttpPortletRendererUtil;
public class MyStockPortletEditDefaultsRenderer extends BaseManagedRenderer
{
  public void renderBody(PortletRenderRequest request) throws PortletException
  {
    PrintWriter out = null;
    try
    {
      out = request.getWriter();
    }
    catch(IOException ioe)
    {
      throw new PortletException(ioe);
    }

    // Personalize the portlet title and stock
    String actionParam = PortletRendererUtil.getEditFormParameter(request);
    PortletRenderRequest prr = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String action = request.getParameter(actionParam);
    String title = prr.getQualifiedParameter("myportlet_title");
    String stock = prr.getQualifiedParameter("myportlet_stock");
    NameValuePersonalizationObject data = null;
    try
    {
      data = (NameValuePersonalizationObject)
              PortletRendererUtil.getEditDefaultData(request);
    }
    catch(IOException io)
    {
      throw new PortletException(io);
    }
    // Cancel automatically redirects to the page, so
    // will only recieve OK or APPLY
    if (action != null)
    {
      data.setPortletTitle(title);
      data.putString("stock",stock);
      try
      {
        PortletRendererUtil.submitEditData(request, data);
      }
      catch(IOException ioe)
      {
        throw new PortletException(ioe);
      }
      return;
    }
    // Otherwise just render the form
    title = data.getPortletTitle();
    stock = data.getString("stock");
    out.print("<table border='0'> <tr> ");
    out.println("<td width='20%'> <p align='right'>Title:</p></td>
                 <td width='80%'>");
    out.print("<input type='TEXT' name='" +
               HttpPortletRendererUtil.portletParameter(prr, "myportlet_title")
               + "' value='" + title + "'>");
    out.println("</td> </tr>");
    out.print("<tr> <td width='20%'> <p align='right'>Stock Codes:</p></td>
               <td width='80%'>");
    out.print("<input type='TEXT' name='" +
               HttpPortletRendererUtil.portletParameter(prr, "myportlet_stock")
               + "' value='" + stock + "'>");
    out.println("<br> For example use US Stock Codes separated by comma:
                 <i> SUNW,IBM,ORCL</i>");
    out.print("</td> </tr>");
    out.println("</table>");
  }
}
```

3. Create the following class, `NetXmethodsServicesStockquoteStockQuoteServiceStub.java`, for your stock portlet:

```
package oracle.portal.sample.v2.devguide.webservices;
import oracle.soap.transport.http.OracleSOAPHTTPConnection;
import org.apache.soap.encoding.SOAPMappingRegistry;
import java.net.URL;
import org.apache.soap.rpc.Call;
import org.apache.soap.Constants;
import java.util.Vector;
import org.apache.soap.rpc.Parameter;
import org.apache.soap.rpc.Response;
import org.apache.soap.Fault;
import org.apache.soap.SOAPException;
import java.util.Properties;
public class NetXmethodsServicesStockquoteStockQuoteServiceStub
{
  public NetXmethodsServicesStockquoteStockQuoteServiceStub()
  {
    m_httpConnection = new OracleSOAPHTTPConnection();
    m_smr = new SOAPMappingRegistry();
  }
  private String _endpoint = "http://64.124.140.30:9090/soap";
  public String getEndpoint()
  {
    return _endpoint;
  }
  public void setEndpoint(String endpoint)
  {
    _endpoint = endpoint;
  }
  private OracleSOAPHTTPConnection m_httpConnection = null;
  private SOAPMappingRegistry m_smr = null;
  public Float getQuote(String symbol) throws Exception
  {
    Float returnVal = null;
    URL endpointURL = new URL(_endpoint);
    Call call = new Call();
    call.setSOAPTransport(m_httpConnection);
    call.setTargetObjectURI("urn:xmethods-delayed-quotes");
    call.setMethodName("getQuote");
    call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
    Vector params = new Vector();
    params.addElement(new Parameter("symbol", String.class, symbol, null));
    call.setParams(params);
    call.setSOAPMappingRegistry(m_smr);
    Response response = call.invoke(endpointURL,
      "urn:xmethods-delayed-quotes#getQuote");
    if (!response.generatedFault())
    {
      Parameter result = response.getReturnValue();
      returnVal = (Float)result.getValue();
    }
    else
    {
      Fault fault = response.getFault();
      throw new SOAPException(fault.getFaultCode(), fault.getFaultString());
    }
    return returnVal;
  }
  public void setMaintainSession(boolean maintainSession)
  {
    m_httpConnection.setMaintainSession(maintainSession);
  }
  public boolean getMaintainSession()
  {
    return m_httpConnection.getMaintainSession();
  }
  public void setTransportProperties(Properties props)
  {
    m_httpConnection.setProperties(props);
  }
  public Properties getTransportProperties()
  {
    return m_httpConnection.getProperties();
  }
}
```

4. Create a Web provider through `provider.xml` for this portlet. Notice the use of the `<preferenceStore>` element to allow for the storing of personalizations:

```
<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>MyStockPortlet</name>
      <title>My Stock Portlet</title>
      <description>Simple Stock Portlet to show Export and Import
                   feature of web providers</description>
      <timeout>80</timeout>
      <showEditToPublic>false</showEditToPublic>
      <hasAbout>false</hasAbout>
      <showEdit>false</showEdit>
      <hasHelp>false</hasHelp>
      <showEditDefault>true</showEditDefault>
      <showDetails>false</showDetails>
      <renderer class="oracle.portal.provider.v2.render.RenderManager">
         <renderContainer>true</renderContainer>
         <renderCustomize>true</renderCustomize>
         <autoRedirect>true</autoRedirect>
         <contentType>text/html</contentType>
         <showPage class="oracle.portal.sample.v2.
                          devguide.tx.MyStockPortletShowRenderer"/>
         <editDefaultsPage class="oracle.portal.sample.v2.devguide.tx.
                                  MyStockPortletEditDefaultsRenderer"/>
      </renderer>
      <personalizationManager class="oracle.portal.provider.v2.personalize.
                                     PrefStorePersonalizationManager">
         <dataClass>oracle.portal.provider.v2.personalize.
                    NameValuePersonalizationObject
         </dataClass>
      </personalizationManager>
   </portlet>
</provider>
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

5. Register this export-enabled provider with the source Oracle Portal instance. For more information about registering Web providers, refer to Section 6.5.5, "Registering and Viewing Your Oracle PDK-Java Portlet".

Note: If the Web provider is running in a secured environment, remember to provide the proxy host and port while starting up Oracle WebLogic Server. For example:

```
JAVA_OPTIONS="-Dhttp.proxyHost=www-proxy.us.oracle.com
-Dhttp.proxyPort=80
```

6. Create two regions on a sample page and add **My Stock Portlet** to the first region. For information on creating regions and pages, refer to the *Oracle Fusion Middleware User's Guide for Oracle Portal*.
7. Edit the page and click the Edit Defaults icon for My Stock Portlet. Choose the stock codes `SUNW,IBM,ORCL`. For more information on how to edit defaults for a portlet on a page, refer to the *Oracle Fusion Middleware User's Guide for Oracle Portal*.
8. Add **My Stock Portlet** to a second region and again edit the defaults. Use a different stock code this time, `MSFT`.
9. Export the page group containing this page. For instructions on how to export a page group, refer to Chapter 10, "Exporting and Importing Content," in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.
10. Import the page group into a target Oracle Portal instance. For instructions on how to import a page group, refer to Chapter 10, "Exporting and Importing Content," in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.
11. View the page with My Stock Portlet in the target Oracle Portal instance and ensure that the personalizations were maintained.

7.2.8.3 Implementing Security for Export/Import

Transporting personalizations can present a security concern if your portlet stores sensitive data and is not operating in a secured environment. At the provider and portlet level, Oracle Portal provides several ways for you to secure the export and subsequent import of portlet personalizations. To better secure portlets and providers for exportation and importation, you can take the following actions:

- Section 7.2.8.3.1, "Securing Provider Communications". Using Oracle Portal configuration options, you can secure the communications between providers and Oracle Portal. This step in turn makes the export and import of portlets more secure.
- Section 7.2.8.3.2, "Disabling Export/Import of Personalizations". You can disable the export of all portlet personalization data for each Web application. This method provides the greatest security but only at a significant cost in functionality because it prevents administrators from retaining their default personalizations when the portlet is moved.
- Section 7.2.8.3.3, "Obfuscating Data for Transport (Automatic)". By default, Oracle Portal obfuscates but does not encrypt personalization data before transporting it.
- Section 7.2.8.3.4, "Encrypting Personalization Data for Transport". You may want to encrypt personalization data for transport if any of the following are true:
  - Your Web provider connection is not secured using HTTPS.
  - You want to ensure the data is secured during transit.
  - You want the data to remain secure while stored in the Oracle Portal instance.
- Section 7.2.8.3.5, "Exporting by Reference". Instead of including portlet personalization data directly in the transport set, you can include it by reference in the transport set. Because the data itself is not present in the transport set, export by reference is the most secure way of transporting personalizations.

7.2.8.3.1 Securing Provider Communications If the security of exporting/importing portlets is of concern to you, you should configure Oracle Portal to secure communications with your portlet providers. The chief mechanisms for securing provider communications in Oracle Portal are as follows:

- Message authentication through a Hashed Message Authentication Code (HMAC) algorithm. For more information on message authentication for providers, refer to Section 6.1.7.8, "Message Authentication", in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.
- HTTPS between providers and Oracle Portal. For more information on HTTPS for provider communications, refer to Section 6.1.7.9, "HTTPS Communication", in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

> **Note:** You cannot use certificates for the HTTPS communication with providers.

7.2.8.3.2 Disabling Export/Import of Personalizations The JNDI variable, `oracle/portal/provider/global/transportEnabled`, controls whether to allow the exportation and importation of personalizations. If you set the variable to true, personalizations are transported as part of export and import. If you set it to false, they are not transported. You can set JNDI variables for PDK-Java through a Deployment Plan set on the PDK-Java web application in the Oracle WebLogic Server. This can be done using the WebLogic Server Administration Console. Deployment Plans allow for easy modification of an application's configuration, without modifying the packaged deployment descriptor files. After setting up the Deployment Plan, you can make manual changes to it for the JNDI variable within the pre-existing WEB-INF/web.xml module descriptor, like the following:

```
<module-descriptor external="false">
      <root-element>web-app</root-element>
      <uri>WEB-INF/web.xml</uri>
      <variable-assignment>
        <name>provider_transportEnabled</name>
<xpath>/web-app/env-entry/[env-entry-name=&quot;oracle/portal/provider/global/tran
sportEnabled&quot;]/env-entry-value</xpath>
      </variable-assignment>
    </module-descriptor>
```

The variable definition of this variable assignment is made directly under the <deployment-plan> tag, and will look like:

```
<variable-definition>
      <variable>
        <name>provider_transportEnabled</name>
        <value>false</value>
      </variable>
    </variable-definition>
```

This sets `oracle/portal/provider/global/transportEnabled` to false.

> **See:** *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*

7.2.8.3.3 Obfuscating Data for Transport (Automatic) By default, personalization data is encoded (Base64). This encoding ensures that data is obfuscated during transport. You do not need to take any actions to leverage Base64 encoding as it is provided by default. However, if you want greater security, you can encrypt the data. Refer to Section 7.2.8.3.4, "Encrypting Personalization Data for Transport".

7.2.8.3.4 Encrypting Personalization Data for Transport By implementing the `oracle.portal.provider.v2.security.CipherManager` class for your provider, you can encrypt the personalization data prior to exporting it. Upon import, the cipher manager is invoked again to decrypt the data. Refer to Section 7.2.8.3.6, "Encrypting Personalization Data Example".

Note: If you choose to encrypt your Web providers for export using the cipher manager, you must also devise your own key management strategy for the encryption algorithm.

7.2.8.3.5 Exporting by Reference As mentioned previously, the default behavior for exporting of portlets is to include the actual personalization data in the transport set. For a more secure transport, you can code your portlet such that the personalizations are exported using pointers rather than by including the actual preference data. When the transport set is imported, the target Oracle Portal instance sends the pointer back to the Web provider, which then has the opportunity to reassociate the actual data with the new portlet instance. Refer to Section 7.2.8.3.7, "Exporting by Reference Example".

Note: When exporting across security zones, exporting by reference may not work effectively. In general, you should only employ export by reference when transporting within the same general security environment.

7.2.8.3.6 Encrypting Personalization Data Example To encrypt personalization data in your Web provider, you need to create your own cipher manager and associate it with your portlet provider. This example provides a simple, insecure cipher manager for illustrative purposes only. To implement a secure implementation of the cipher manager for your production system, you would need to significantly extend this sample. Some of the issues you would need to consider for a production implementation are as follows:

- Do not hold the key object in memory. Read it from a persistent store as necessary.
- Use the provider's `PreferenceStore` API supported by a `DBPreferenceStore` to work in the clustered case.
- On import, if the cipher manager instance obtained from `provider.xml` matches the class name returned in the SOAP message, that `CipherManager` instance is used to perform the decryption. Hence, the instance maintained in the portlet/provider definition may be configured using any applicable means (for example, tags in `provider.xml` or JNDI variable) and that configuration is reused on import.

To encrypt personalization data in your Web provider, do the following:

Note: The following sample is for illustrative purposes only. You would need to significantly enhance it for use in a production environment.

---

1. Create a cipher manager class, `InsecureCipherManager`. This class will be used for encryption and decryption of personalization data exported from or imported to a Web provider. A base64 encoded, hard coded secret key is used with the DES algorithm supplied by the default `javax.crypto` provider of the underlying Java Runtime Environment. As a result, this particular sample is insecure because the encoded key can be recovered by a malicious party simply by decompiling the byte code.

   Note: This sample makes use of the `javax.crypto` package, which is optional in Java 1.3 and must be installed manually. In Java 1.4, though, this package is present by default.

```
package oracle.portal.sample.v2.devguide.tx;
import java.io.IOException;
import java.security.GeneralSecurityException;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import oracle.portal.provider.v2.ProviderException;
import oracle.portal.provider.v2.security.CipherManager;
import sun.misc.BASE64Decoder;
public final class InsecureCipherManager implements CipherManager
{
    /**
     * Base64 encoded external form of a javax.crypto.SecretKey which was
     * generated for the DES algorithm. This is completely insecure! Anyone
     * can decompile the bytecode and recostitue the key. A more secure
     * implementation would implement a key management policy in a
     * java.security.KeyStore.
     */
    private static final String sEncodedKey = "UTJds807Arw=";
    /**
     * Generated from the (insecure) encoded form in sEncodedKey.
     */
    private SecretKey mKey;
    /**
     * Transforms the input data to a more secure form, in a single operation,
     * using the DES cryptographic algorithm along with a statically defined
     * secret key.
     * @param toEncode the input data.
     * @return an encoded form of the input data.
     * @throws ProviderException if an error occurs during transform.
     */
    public final byte[] encode(byte[] toEncode) throws ProviderException
    {
        try
        {
            Cipher c = Cipher.getInstance("DES");
            c.init(Cipher.ENCRYPT_MODE, getSecretKey());
            return c.doFinal(toEncode);
        }
        catch (GeneralSecurityException gse)
        {
            throw new ProviderException(gse);
        }
        catch (IOException ioe)
        {
            throw new ProviderException(ioe);
        }
    }
    /**
     * Transforms the input data to its original form, in a single operation,
     * using the DES cryptographic algorithm along with a statically defined
     * secret key.
     * @param toDecode the input data.
     * @return a decoded form of the input data.
     * @throws ProviderException if an error occurs during transform.
     */
    public final byte[] decode(byte[] toDecode) throws ProviderException
    {
        try
        {
            Cipher c = Cipher.getInstance("DES");
            c.init(Cipher.DECRYPT_MODE, getSecretKey());
            return c.doFinal(toDecode);
        }
        catch (GeneralSecurityException gse)
        {
            throw new ProviderException(gse);
        }
        catch (IOException ioe)
        {
            throw new ProviderException(ioe);
        }
    }
    /**
     * Returns a <code>javax.crypto.SecretKey</code> deserialized from the
     * obuscated form in sEncodedKey. Note, this is highly insecure!!
     */
    private SecretKey getSecretKey()
        throws GeneralSecurityException, IOException
    {
        if (mKey == null)
        {
            DESKeySpec ks = new DESKeySpec((new BASE64Decoder()).decodeBuffer(
                sEncodedKey));
            SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
            mKey = skf.generateSecret(ks);
        }
        return mKey;
    }
}
```

2. Modify your `provider.xml` to reference the cipher manager:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<?providerDefinition version="3.1"?>
<provider class="oracle.portal.provider.v2.DefaultProviderDefinition">
<providerInstanceClass>net.mzyg.tx.TxProviderInstance</providerInstanceClass>
   <session>false</session>
   <passAllUrlParams>false</passAllUrlParams>
   <preferenceStore class="oracle.portal.provider.v2.
      preference.DBPreferenceStore">
      <name>prefStore1</name>
      <connection>java:comp/env/jdbc/PortletPrefs</connection>
   </preferenceStore>
<cipherManager class="oracle.portal.sample.v2.devguide.tx.
   InsecureCipherManager"/>
```

7.2.8.3.7 Exporting by Reference Example To export by reference rather than exporting the actual personalization, do the following:

1. Override the `DefaultPortletInstance` with the following `ExportByRefDefaultPortletInstance`:

```
package oracle.portal.sample.v2.devguide.tx;
import oracle.portal.provider.v2.DefaultPortletInstance;
import oracle.portal.provider.v2.preference.PreferenceStore;
import oracle.portal.provider.v2.transport.PrefStoreTransporter;
public class ExportByRefDefaultPortletInstance extends DefaultPortletInstance
{
  /**
   * Returns a {@link oracle.portal.provider.v2.transport.PrefStoreTransporter}
   * capable of carrying out transport operations such as export/import on
   * data applicable to {@link oracle.portal.provider.v2.PortletInstance}
   * persisted in {@link oracle.portal.provider.v2.preference.PreferenceStore}.
   * This implementation returns an {@link ExportByRefPrefStoreTransporter}.
   * @param ps the {@link oracle.portal.provider.v2.preference.PreferenceStore}
   * containing the data to be transported.
   * @return a {@link oracle.portal.provider.v2.transport.PrefStoreTransporter}
   */
  protected PrefStoreTransporter getPrefStoreTransporter(PreferenceStore ps)
  {
      return new ExportByRefPrefStoreTransporter(ps);
  }
}
```

2. Create the `ExportByRefPrefStoreTransporter` class referenced in `ExportByRefDefaultPortletInstance`. This class implements an alternative preference store transporter that does not send preference data during the export operation. Instead, it writes the context path of the source preference to the stream. During the export, it reads the context path and uses that path to look up the preference data and copy it to the new instance. This method of exporting by reference assumes that the source and target providers have access to the same preference store. In fact, the best case for this example would be the situation where the source and target providers are the same.

```
package oracle.portal.sample.v2.devguide.tx;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import oracle.portal.provider.v2.transport.PrefStoreTransporter;
import oracle.portal.provider.v2.transport.TransportLogger;
import oracle.portal.provider.v2.preference.Preference;
import oracle.portal.provider.v2.preference.PreferenceStore;
import oracle.portal.provider.v2.preference.PreferenceStoreException;
public class ExportByRefPrefStoreTransporter extends PrefStoreTransporter
{
    public ExportByRefPrefStoreTransporter(PreferenceStore prefStore)
    {
        super(prefStore);
    }
   /**
     * Exports the context path of the supplied {@link
     * oracle.portal.provider.v2.preference.Preference} from the {@link
     * oracle.portal.provider.v2.preference.PreferenceStore}.
     * @param pref the source {@link
     * oracle.portal.provider.v2.preference.Preference}
     * @param out the <code>java.io.OutputStream</out> to which data will be
     * written.
     * @param logger
     */
   protected void exportPreference(Preference pref, OutputStream out,
        TransportLogger logger) throws PreferenceStoreException, IOException
    {
        // Get the context path of the preference we are exporting.
        String contextPath = pref.getContextPath();
        DataOutputStream dos = new DataOutputStream(out);
        // Write the context path in the export data. The import process
        // will use this context path to lookup this preference in the
        // preference store and copy it to the new context
        dos.writeUTF(contextPath);
    }
    /**
     * Reads a context path from the stream and copies preference data
     * from that location into the {@link
     * oracle.portal.provider.v2.preference.PreferenceStore}.
     * @param pref the target {@link
     * oracle.portal.provider.v2.preference.Preference}
     * @param in the <code>java.io.InputStream</code> from which to read data.
     * @param logger
     */
    protected void importPreference(Preference pref, InputStream in,
        TransportLogger logger) throws PreferenceStoreException, IOException
    {
        // Read the context path to copy the value for this
        // preference from.
        DataInputStream dis = new DataInputStream(in);
        String contextPath = dis.readUTF();
        // Create preference object to copy from (identical to the
        // target preference but with a different context path)
        Preference sourcePref = new Preference(contextPath,
           pref.getName(), pref.getType(), (String)null);
        // Copy across the preference
        getPrefStore().copy(sourcePref, pref, true);
    }
}
```

3. Update `provider.xml` to include the following element for your portlet:

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
...
<portletInstanceClass>oracle.portal.sample.v2.devguide.tx.
     ExportByRefDefaultPortletInstance</portletInstanceClass>
</portlet>
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

7.2.9 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, 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, 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.

Web providers generate dynamic content (that is, portlets) and they often reside remotely from the Oracle Portal instance on which they are deployed. As such, caching might improve their performance. The architecture of Oracle Portal lends itself well to caching, since all rendering requests originate from a single page assembling agent, known as the Parallel Page Engine (PPE), which resides on the middle tier. You can make the PPE cache the portlets rendered by your Web provider and reuse the cached copies to handle subsequent requests, minimizing the overhead your Web provider imposes on page assembly.

The Web provider 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 provider 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. Refer to Section 7.2.9.2, "Activating Caching" and Section 7.2.9.3, "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, 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 takes place, 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 Web provider 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 Web provider, is contacted only when new content needs to be generated, but you are not bound to a fixed regeneration schedule. Refer to Section 7.2.9.2, "Activating Caching" and Section 7.2.9.4, "Adding Invalidation Based Caching" for more information.

3. **Validation-based Caching**: When a provider 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 provider a render request that includes the version identifier of the cached content. The provider determines whether the version identifier remains valid. If the version identifier is still valid, the provider immediately sends a lightweight response to the PPE without any content, which indicates the cached version can be used. Otherwise, the provider 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 provider whether the content is up to date. The validity of the cached copy is determined by some logic in the provider. The advantage of this approach is that the provider controls the use of the cached content rather than relying on a fixed period of time. Refer to Section 7.2.9.2, "Activating Caching" and Section 7.2.9.5, "Adding Validation-Based Caching" for more information.

7.2.9.1 Assumptions

To perform the tasks in these sections, we are making the following assumptions:

1. You have followed through and understood Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

7.2.9.2 Activating Caching

To use the caching features of Oracle Portal in your Web providers, 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 providers, 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 Web provider. Bear in mind that remote Web providers often do not have Oracle Web Cache installed. For more information about Oracle Web Cache, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache*.

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 Web provider.
- Caching rules do not cause the caching of provider content. Content should be cached according to the surrogate control headers generated by the provider in its response.

7.2.9.3 Adding Expiry-Based Caching

Expiry-based caching is one of the simplest caching schemes to implement, and can be activated declaratively in your XML provider definition. You can set an expiry time for the output of any `ManagedRenderer` you utilize 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 6.5, "Building Oracle 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>
   ```

   More On OTN

   For more information on the syntax of `provider.xml`, refer to the provider 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.

7.2.9.4 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. Refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Web Cache* for more information.
- Section 7.2.9.4.1, "Configuring the Provider Servlet"
- Section 7.2.9.4.2, "Defining the Oracle Web Cache Invalidation Port"
- Section 7.2.9.4.3, "Configuring the XML Provider Definition"
- Section 7.2.9.4.4, "Manually Invalidating the Cache"

7.2.9.4.1 Configuring the Provider Servlet To enable invalidation-based caching, you must switch it on at the provider 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 provider's test page. If the `testPageURI` property is not set in the `sample.properties` file, then the provider 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

7.2.9.4.2 Defining the Oracle Web Cache Invalidation Port If you are using an Oracle Application Server installation type where PDK-Java was automatically pre-installed (for example, an Oracle Portal and Wireless type installation), 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 7.2.9.4.3, "Configuring the XML Provider 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 port(s) of the Oracle Web Cache instance(s) 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 Oracle Containers for Java EE installation, 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 file. To set this system property for a Oracle WebLogic Server, simply add an appropriate line to the `.product.properties` file for the particular instance in which PDK-Java is installed (for example, `MW_HOME/wlserver_10.3/.product.properties`) and then restart that instance.

Note: Since 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 Web Provider instances behind separate Oracle Web Cache configurations.

7.2.9.4.3 Configuring the XML Provider 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), 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 from Oracle Portal defined in the Cache tab of the Global Settings page.
- Web Provider default (24 hours) if no maximum expiry time is specified by Oracle Portal.
- 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 takes place in an Oracle Web Cache instance located in the same place as the Web provider. Pages stored using expiry based caching are cached in the middle tier of Oracle Portal.

More On OTN

For more information on the syntax of `provider.xml`, refer to the **provider Javadoc** on OTN:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

7.2.9.4.4 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 Web provider 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 provider that removes the portlet from the cache. The next request for the portlet is forwarded to the provider and the provider 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 provider. 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 provider for more information.

7.2.9.5 Adding Validation-Based Caching

Adding validation-based caching requires slightly more effort, but gives you explicit control over exactly which requests to your provider 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, 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, the portlet shows the new content. If the content has not changed, a cached version of the portlet is displayed.

7.2.10 Enhancing Portlets for Mobile Devices

This section explains how to go about enhancing a PDK-Java portlet for a mobile device. Before proceeding with this section, you should familiarize yourself with the guidelines for building mobile-enabled portlets, Section 6.1.4, "Guidelines for Mobile Portlets", and the methods of building portlets with PDK-Java, Section 7.2, "Enhancing PDK-Java Portlets".

To properly build a portlet for a mobile device, do the following:

1. Create a JSP to generate the OracleAS Wireless XML in response to the show request for the portlet. The portlet's JSPs are managed and controlled by the built in renderer, `oracle.portal.provider.v2.render.http.ResourceRenderer`.

   With this renderer, you can define distinct resources (JSPs) for each Show mode as well as which JSP to call for each Show mode. Hence, you can define one JSP to handle the Show mode when the requested content is HTML and another when the requested content is OracleAS Wireless XML. For example, this capability enables you to put the lottery portlet's mobile rendition in a separate JSP. In this sample, `mlotto.jsp` generates the mobile rendition.

   Note the `contentType` declaration in the first line of the source code. The content type of the JSPs response is set to `text/vnd.oracle.mobilexml`. This is the MIME type name for OracleAS Wireless XML. All mobile responses must set their content type to this value to work properly.

   ```
   <%@ page session="false" contentType="text/vnd.oracle.mobilexml" %>
   <%@ page import="oracle.portal.sample.v2.devguide.lottery.LottoPicker" %>
   <% LottoPicker picker = new LottoPicker();
      picker.setIdentity(request.getRemoteAddr() ); %>
   <SimpleText>
       <SimpleTextItem />
       <SimpleTextItem>Your numbers:</SimpleTextItem>
       <SimpleTextItem />
       <SimpleTextItem HALIGN="CENTER" >
           <%
           int [] picks = picker.getPicks();
           for (int i = 0; i < picks.length; i++) {
               out.print(picks[i] + " ");
           }
           out.println("");
           %>
       </SimpleTextItem>
   </SimpleText>
   ```

2. Modify `provider.xml` by adding the `acceptContentType` tag in the portlet definition to indicate that the portlet can generate OracleAS Wireless XML responses. Each `acceptContentType` tag defines a distinct content type that the portlet can render. The value is the MIME type of the content. In this case the lottery portlet declares that it can generate HTML (`text/html`) and OracleAS Wireless XML (`text/vnd.oracle.mobilexml`).

   ```
   <acceptContentType>text/html</acceptContentType>
   ```

```
<acceptContentType>text/vnd.oracle.mobilexml</acceptContentType>
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

3. Declare the mapping of JSP renderers to Show modes in `provider.xml`.

   You need to define a render handler for the pertinent Show modes by content type. One `showPage` declaration identifies a JSP that renders the HTML response and another one identifies the JSP that renders the OracleAS Wireless XML response. Note that the class attribute is now required, whereas, in a simpler case, it could allow the class to default. You express the association of a particular JSP to a particular content type through the `resourcePath` and `contentType` tags inside the `showPage` declaration as follows:

   - `resourcePath` defines the JSP used to render this mode.
   - `contentType` defines the response type of this JSP.

   When the `ResourceRenderer` receives a show request for the lottery portlet it invokes `lotto.jsp` to render the HTML response and `mlotto.jsp` for OracleAS Wireless XML. The following code illustrates how you express this relationship in `provider.xml`:

```
<renderer class="oracle.portal.provider.v2.render.RenderManager">
   <contentType>text/html</contentType>
   <renderContainer>true</renderContainer>
   <showPage
      class="oracle.portal.provider.v2.render.http.ResourceRenderer">
      <resourcePath>/htdocs/lottery/lotto.jsp</resourcePath>
      <contentType>text/html</contentType>
   </showPage>
   <showPage
      class="oracle.portal.provider.v2.render.http.ResourceRenderer">
      <resourcePath>/htdocs/lottery/mlotto.jsp</resourcePath>
      <contentType>text/vnd.oracle.mobilexml</contentType>
   </showPage>
   ...
</renderer>
```

Note: The `ResourceRenderer` (and any portlet) determines the type of request it has received by the request's Accept header. The Accept header is a standard HTTP header that defines the acceptable response types for a given request. It may be a list with multiple values if multiple content types are acceptable. When there are multiple values, they are listed in order from most to least preferred. Oracle Portal controls the values passed to the portlet in the Accept header. In the case of an HTML request, the Accept header is:

```
text/html, text/xml, text/vnd.oracle.mobilexml
```

These values indicate that Oracle Portal prefers an HTML response but also accepts XML and OracleAS Wireless XML.

For mobile requests, the Accept header is:

```
text/vnd.oracle.mobilexml, text/xml
```

These values indicate that Oracle Portal prefers OracleAS Wireless XML but also accepts general XML that contains a stylesheet reference that transforms to OracleAS Wireless XML.

The `ResourceRenderer` maps requested content type to the specific resource renderer by working through the Accept list in preference order. Once it finds a match between an acceptable response type and a registered renderer, it dispatches the request to the resource. For example, for HTML requests the `ResourceRenderer` will match the first entry, `text/html`, to the declaration that defines `lotto.jsp` as the `text/html` handler. Likewise, when a mobile request is received, the `ResourceRenderer` matches the first entry, `text/vnd.oracle.mobilexml`, to the declaration that defines `mlotto.jsp` as the `text/vnd.oracle.mobilexml` handler.

---

4. Declare a short title. A short title is the short form of a portlet's name and is for use where display space is limited. Specifically, Oracle Portal uses it when rendering portlets as menu items in the page menu generated in response to a mobile request. If the portlet has registered a short title, Oracle Portal uses that as the menu item label. Otherwise, it uses the portlet's standard title. To declare a short title, you include the `shortTitle` tag in the portlet's metadata section in `provider.xml`:

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
   <id>2</id>
   <name>Lottery</name>
   <title>Lottery Portlet</title>
   <shortTitle>Lottery</shortTitle>
   ...
</portlet>
```

5. Support short title personalization. Because the portlet's short title is presented to the user as the menu item label that references the portlet instance on the page, we recommend that all portlets allow users to personalize the short title. PDK-Java provides a base data class that manages both the short and standard title:

```
oracle.portal.provider.v2.personalize.NameValuePersonalizationObject
```

The `NameValuePersonalizationObject` manages both the title and short title. It contains methods for getting and setting the values. The personalization code is split into two parts, form display and form submit. The logic first acquires the current data personalization object. Next, it checks to see if this is a form submit. If so, it updates the values in the data personalization object, writes them back to the repository, and returns. PDK-Java subsequently redirects back to this same form but without the parameters that indicate it is a form submit causing the form to redisplay with the new values. In this situation, the JSP responds with the personalization page including the current personalization values.

We recommend that portlets use this class or subclass to inherit this base support. The only personalization the lottery sample supports is these two titles. Hence, it uses the `NameValuePersonalizationObject` class directly in `custom.jsp`:

```
<%@ page session="false" import="oracle.portal.provider.v2.*" %>
<%@ page import="oracle.portal.provider.v2.http.*" %>
<%@ page import="oracle.portal.provider.v2.render.*" %>
<%@ page
  import="oracle.portal.provider.v2.personalize.NameValuePersonalizationObject"
%>
<%-- This page both displays the personalization form and processes it.
     We display the form if there isn't a submitted title parameter,
     else we apply the changes --%>
<%
    PortletRenderRequest portletRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String title = request.getParameter("lotto_title");
    String sTitle = request.getParameter("lotto_short_title");
    String actionParam =
PortletRendererUtil.getEditFormParameter(portletRequest);
    String action = request.getParameter(actionParam);
    NameValuePersonalizationObject data = (NameValuePersonalizationObject)
        PortletRendererUtil.getEditData(portletRequest);
    // Cancel automatically redirects to the page
    //   -- so will only receive 'OK' or 'APPLY'
    if (action != null) {
        data.setPortletTitle(title);
        data.setPortletShortTitle(sTitle);
        PortletRendererUtil.submitEditData(portletRequest, data);
        return;
    }
    // otherwise just render the form
    title = data.getPortletTitle();
    sTitle = data.getPortletShortTitle();
%>
<TABLE BORDER="0">
    <TR>
        <TD WIDTH="20%">
            <P ALIGN="RIGHT">Title:
        </TD>
        <TD WIDTH="80%">
            <INPUT TYPE="TEXT" NAME="lotto_title"
                   VALUE="<%= title %>" SIZE="20">
        </TD>
    </TR>
    <TR>
    <TD WIDTH="20%">
        <P ALIGN="RIGHT">Short Title:
    </TD>
    <TD WIDTH="80%">
        <INPUT TYPE="TEXT" NAME="lotto_short_title" VALUE="<%= sTitle %>"
           SIZE="20" MAXLENGTH="20">
    </TD>
    </TR>
</TABLE>
```

6. Support the rendering of personalized short titles. Once you add short title personalization, a portlet must take responsibility for rendering the portlet as a menu item in the mobile page response. Because of the small screen displays on mobile devices, portlets aren't rendered together. Users are presented with a menu of links to portlets on a given page. The user navigates to each portlet through this menu to see the content. To better support this model, Oracle Portal provides Link mode, where portlets generate Link references to themselves.

   For HTML requests, Link mode generates an anchor tag, a `href`. For mobile requests, Link mode generates a `simpleHref` tag. Currently, Oracle Portal only sends a Link mode request when assembling a mobile page response. Hence, you only need to support rendering the `text/vnd.oracle.mobilexml` content type for Link mode. If your portlet also has an HTML rendition, we recommend you also support HTML Link mode.

   The lottery portlet example implements each Link rendition in a distinct JSP, as follows:

   - `milotto.jsp` contains the code to generate a `text/vnd.oracle.mobilexml` Link response.
   - `anchorlotto.jsp` contains the code to generate a `text/html` Link response.

   The code for each is almost identical. Since the purpose of supporting Link mode is to render the personalized short title, the code first acquires the short title from the repository. It then generates the appropriate link tag for the requested markup. The acquired short title is rendered as the link's label. Since Oracle Portal is responsible for creating the URL that references any particular usage of a portlet on a page, Oracle Portal creates the URL used in the link and passes it to the portlet, which then uses the URL to create the link. The URL for the link is retrieved from the request's `PageURL` parameter. The code for `milotto.jsp` is as follows:

```
<%@ page session="false" contentType="text/vnd.oracle.mobilexml" %>
<%@ page import="oracle.portal.provider.v2.http.HttpCommonConstants" %>
<%@ page import="oracle.portal.provider.v2.render.*" %>
<%@ page
  import="oracle.portal.provider.v2.personalize.NameValuePersonalizationObject"
%>
<%@ page import="oracle.portal.utils.xml.v2.XMLUtil" %>
<%
    PortletRenderRequest portletRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    NameValuePersonalizationObject data = (NameValuePersonalizationObject)
        PortletRendererUtil.getEditData(portletRequest);
    String title = data.getPortletShortTitle();
    // if short title is empty then use the title
    if (title == null || title.length() == 0)
        title = data.getPortletTitle();
%>
<SimpleHref target="<%= XMLUtil.escapeXMLAttribute(
         portletRequest.getRenderContext().getPageURL()) %>">
   <%= XMLUtil.escapeXMLText(title) %>
</SimpleHref>
```

Note: The text being output as the URL and the short title label are passed through special XML escape utilities. Because `text/vnd.oracle.mobilexml` is an XML content type, generated responses must adhere to XML rules, which require the escaping of specific characters. Unless generating static text, we recommend that all text be escaped using the two supplied utilities. The reason for two utility methods is that the set of characters requiring escape varies depending upon whether the text is a tag attribute value or a tag value.

---

7. Declare support for Link mode. Once you have implemented Link mode, you must update `provider.xml` to indicate the presence of Link mode. You declare Link mode by adding code similar to the following to `provider.xml`:

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
   <id>2</id>
   <name>Lottery</name>
   ...
   <showLink>true</showLink>
   <showEdit>true</showEdit>
   <showEditToPublic>false</showEditToPublic>
   ...
</portlet>
```

8. Bind the JSPs to the Link mode. The portlet must declare the mapping between the Show modes and the JSP renderers. The syntax for Link mode is similar to that for Show mode.

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
   <id>2</id>
   <name>Lottery</name>
   ...
   <renderer class="oracle.portal.provider.v2.render.RenderManager">
      <contentType>text/html</contentType>
      <renderContainer>true</renderContainer>
      <linkPage
         class="oracle.portal.provider.v2.render.http.ResourceRenderer" >
         <resourcePath>/htdocs/lottery/anchorlotto.jsp</resourcePath>
         <contentType>text/html</contentType>
      </linkPage>
      <linkPage
         class="oracle.portal.provider.v2.render.http.ResourceRenderer" >
         <resourcePath>/htdocs/lottery/milotto.jsp</resourcePath>
         <contentType>text/vnd.oracle.mobilexml</contentType>
      </linkPage>
     ...
   </renderer>
   ...
</portlet>
```

7.2.10.1 Accessing Configuration, User, and Device Information

To better support mobile devices, Oracle Portal passes extra information to the portlet for use in generating its response. This information falls into three major categories, as follows:

- Section 7.2.10.1.1, "Configuration Data"

- Section 7.2.10.1.2, "User Data"
- Section 7.2.10.1.3, "Device Information"

7.2.10.1.1 Configuration Data Oracle Portal sends a flag that indicates whether the mobile function is enabled when requesting that the portlet render its personalization or edit pages. Portlets can use this information to exclude or include mobile specific attributes in their personalization pages as appropriate.

The portlet accesses this information using the `PortletRenderRequest` object:

```
...
if (portletRequest.getPortalConfig().isMobileEnabled()) {
...
}
...
```

7.2.10.1.2 User Data OracleAS Wireless adds the user location to the requests it forwards to Oracle Portal for response. This user location information is determined by the user's actual location, if the user's device has this support, or a profiled location. The user location is exposed by the `ProviderUser` object, which you can access from the `PortletRenderRequest` object. Portlets that are location aware can use this information to adjust the content they generate.

```
...
UserLocation location = portletRequest.getUser().getLocation();
if (location != null) {
   ...
}
...
```

7.2.10.1.3 Device Information On each request, Oracle Portal sends characteristics about the requesting device to the portlet. This information is sent for all portlet requests, not just mobile requests. The information classifies the type of device making the request, its layout orientation, the maximum response size the device can handle, and an indication of whether the connection with the device is secure. Table 7–3 describes the available device types.

Table 7–3 Device Classes

Class	Description
`voice`	Indicates a voice-only device, such as a normal telephone calling a voice access number.
`micromessenger`	Indicates text messenging devices, such as SMS phones or pagers.
`messenger`	Indicates general messenging devices, such as e-mail.
`microbrowser`	Indicates a small size display device, which supports a markup browser, such as a WAP phone.
`pdabrowser`	Indicates a medium size display device, such as a Palm or PocketPC.
`pcbrowser`	Indicates a large size display device used with desktop browsers.

Portlets may choose to alter the layout or representation of the content in their response based on the type of device making the request. For example, a portlet may break a wide table into a series of screens that link column groups together for small screen devices.

The maximum response size is a hint that a portlet can use to constrain the amount of data it returns. The size is expressed in bytes. A maximum response size of 0 means the size is unknown.

The portlet accesses this information using the `PortletRenderRequest` object.

```
...
DeviceInfo deviceInfo = portletRequest.getDeviceInfo();
switch (deviceInfo.getDeviceClass()) {
    case DeviceInfo.DeviceClass.MICROBROWSER:
        renderMicroBrowser(portletRequest);
        break;
    default:
        renderDefault(portletRequest);
        break;
}
...
```

7.2.10.2 Modifying Navigation for Mobile Portlets

Much of the information in Section 7.2.3.3.4, "Implementing Navigation within a Portlet" is also relevant to the development of mobile portlets, but some of the utilities referenced are specific to portlets that render HTML. For portlets that render `SimpleResult`, the equivalent utilities shown in Table 7–4 are available.

Table 7–4 Equivalent HTML and SimpleResult Utilities

HTML Utilities	SimpleResult Utilities
`UrlUtils.constructHTMLLink`	`UrlUtils.constructMXMLLink`
`UrlUtils.htmlFormHiddenFields`	`UrlUtils.mxmlFormHiddenFields`
`UrlUtils.emitHiddenField`	`UrlUtils.emitMxmlHiddenField`

The following example illustrates how to adapt the thesaurus sample for a mobile-enabled portlet. Note that, when deployed as a mobile portlet, both sets of JSPs (HTML and `SimpleResult`) are deployed. These JSPs complement their HTML counterparts, they do not replace them. Notice also that the JSPs use `SimpleResult` as their markup and the value of the navigation parameter has changed such that it points to the next mobile JSP rather than the next desktop JSP.

`mThesaurusForm.jsp`:

```
<%
    PortletRenderRequest pRequest = (PortletRenderRequest)
        request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);
    String paramNameQ = "q";
    String qualParamNameQ =
        HttpPortletRendererUtil.portletParameter(pRequest, paramNameQ);
%>
<!-- Output the MXML content -->
<SimpleText>
    <SimpleTitle>Thesaurus</SimpleTitle>
    <SimpleTextItem>Enter the word you wish to search for:</SimpleTextItem>
    <SimpleForm
        method="POST"
        target="<%= UrlUtils.htmlFormActionLink(pRequest,UrlUtils.PAGE_LINK) %>">
        <%=UrlUtils.mxmlFormHiddenFields
           (pRequest.getRenderContext().getPageURL()) %>
        <%= UrlUtils.emitMxmlHiddenField(
                HttpPortletRendererUtil.portletParameter(request, "next_page"),
                "htdocs/path/mThesaurusLink.jsp" ) %>
        <SimpleFormItem type="text" size="20" name="<%= qualParamNameQ %>"
                 value="" />
    </SimpleForm>
</SimpleText>
```

mThesaurusLink.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 MXML content -->
<SimpleText>
    <SimpleTitle>Words similar to <%= paramValueQ %></SimpleTitle>
<%
        Thesaurus t = new Thesaurus();
        String[] relatedWords = t.getRelatedWords(paramValueQ);
        NameValue[] linkParams = new NameValue[2];
        linkParams[0] = new NameValue(
            qualParamNameNextPage, "htdocs/path/mThesaurusLink.jsp");
        for (int i=0; i<relatedWords.length; i++)
        {
            linkParams[1] = new NameValue(
                qualParamNameQ, relatedWords[i]);
%>
        <SimpleTextItem>
                <%= relatedWords[i] %>
                <SimpleBreak/>
                <%= UrlUtils.constructMXMLLink(
                    pRequest,
                    pRequest.getRenderContext().getPageURL(),
                    "(words related to " + relatedWords[i] + ")",
                    "",
                    linkParams,
                    true,
                    true)%>
        </SimpleTextItem>
<%
        }
%>
    <SimpleTextItem>
        <SimpleHref target="<%=XMLUtil.escapeXMLAttribute(
            pRequest.getRenderContext().getPageURL())%>">
            Reset Portlet
        </SimpleHref>
    </SimpleTextItem>
</SimpleText>
```

7.2.11 Writing Multilingual Portlets

This section shows you how to build a Java portlet that can be rendered in different languages. The language used in your portlet will depend upon on the language setting that has been chosen in the portal that is displaying it.

Once you have completed this section you will be able to write portlets that support as many or as few languages as you wish. You will also be able to convert your existing portlets to support multiple languages. Once a portlet is written to support multiple languages, it is easy to plug in new languages. The basic model for multilingual Java portlets is similar to the standard Java Internationalization model. If you already know about Java Internationalization, you should find this process very familiar.

7.2.11.1 Assumptions

To perform the tasks in this section, we are making the following assumptions:

1. You have followed through and understood Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper".
2. You built a portlet using the wizard and successfully added it to a page.

7.2.11.2 Internationalizing Your Portlet

This consists of the following two tasks:

- Section 7.2.11.2.1, "Providing Translations for Portlet Content"
- Section 7.2.11.2.2, "Providing Translation for Portlet Attributes"

7.2.11.2.1 Providing Translations for Portlet Content In Section 6.5, "Building Oracle PDK-Java Portlets with Oracle JDeveloper", you created a portlet using the Java Portlet Wizard. The basic message created by the wizard is only available in one language and the text displayed is hard-coded in to the portlet's renderer class. To make your portlets available in multiple languages, you have to store such language dependent elements in their own *resource bundles*.

Creating Resource Bundles

For each language you want your portlet to be available in, you will need a resource bundle. You will also need to create a resource bundle to use when there is no resource bundle corresponding to the language setting chosen in the portal. Perform the following tasks:

- **Create a Default Resource Bundle**

  Perform the following steps:

  1. In Oracle JDeveloper, create a Java class called `MyProviderBundle` that extends `ListResourceBundle` from the `java.util.package`. The class should contain a multi-dimensional array of objects that holds key-value pairs representing each of the language dependent elements from your JSP show page. This implementation is demonstrated in the following code:

```
package mypackage2;
import java.util.ListResourceBundle;
public class MyProviderBundle extends ListResourceBundle
{
public static String HELLO_MSG = "MyPortletHelloMessage";
public static String INFO_MSG = "MyPortletInfoMessage";
public Object[][] getContents()
{
```

```
return contents;
}
static private final Object[][] contents =
{
{HELLO_MSG, "Hello"},
{INFO_MSG, "This is the show page of the portlet and it is being generated
in the default language!"}
};
}
```

2. Save `MyProviderBundle`.

- **Creating Resource Bundles for Other Supported Languages**

  Now you must create a resource bundle class for each language you want your portlet to support. Each of these classes must be named the same as your default resource bundle class, but with a language code appended to the end. For example, if you want to support the French language, create a Java class named `MyProviderBundle_fr`. The language code `fr` is the same as the code that will be used by the *locale* object in the portal if the language setting is set to French

  More On OTN

  For more information on Locales, search for `java.util.Locale` in the Javadoc. Refer to 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`

  When you change the language setting in Oracle Portal, you change the value of the current locale object and therefore the locale object's language code. These language codes adhere to the ISO:639 codes for representation for names of languages. To create resource bundles for other supported languages, perform the following steps:

1. To create a French resource bundle, create a Java class named `MyProviderBundle_fr`, as described earlier.
2. Using your default resource bundle as a template, replace the English language strings with their French equivalents. An example is as follows:

```
package mypackage2;

import java.util.ListResourceBundle;
public class MyProviderBundle_fr extends
ListResourceBundle
{
public Object[][] getContents()
{
return contents;
}
static private final Object[][] contents =
{
{MyProviderBundle.HELLO_MSG, "Bonjour"},
{MyProviderBundle.INFO_MSG, "Cette page est le 'show mode' de la portlet
    et est generee dans la langue par defaut."}
};
}
```

3. Save `MyProviderBundle_fr`.

4. Repeat steps 1 through 3 for every language that you wish to create a resource bundle for, updating the class name with the appropriate language code and the message strings with their equivalent in the appropriate language.

Updating Your Renderer

To make use of the resource bundles you just created, you need to edit the JSP show page and replace the hard-coded messages with references that will pickup the messages at run time from the resource bundle that corresponds most closely with the locale object of the portal. To update your renderer, perform the following steps:

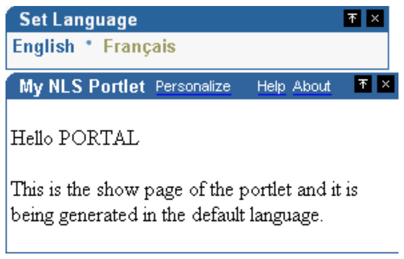
1. Open the JSP that represents your show page and change the following:

```
<%@page contentType="text/html; charset=windows-1252"
   import="oracle.portal.provider.v2.render.PortletRenderRequest"
   import="oracle.portal.provider.v2.http.HttpCommonConstants"
   import="java.util.ResourceBundle"
%>

<%
 PortletRenderRequest pReq = (PortletRenderRequest)
   request.getAttribute(HttpCommonConstants.PORTLET_RENDER_REQUEST);

<!-- Get a resource bundle object for the current language. -->
ResourceBundle b =
ResourceBundle.getBundle("mypackage2.MyProviderBundle",pReq.getLocale());
%>

<!--  Pull the message from the appropriate resource bundle. -->
<P> <%= b.getString(mypackage2.MyProviderBundle.HELLO_MSG) %>
   <%= pReq.getUser().getName() %>.</P>
<P> <%= b.getString(mypackage2.MyProviderBundle.INFO_MSG) %></P>
```

2. Save your JSP page.

Now you can refresh your portlet and view the changes (Figure 7–12).

Figure 7–12 Portlet in English

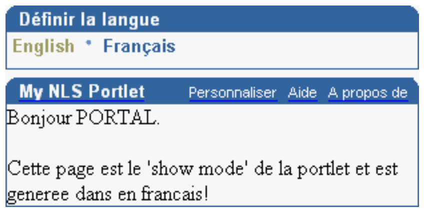


Hello PORTAL

This is the show page of the portlet and it is being generated in the default language.

To view the French greeting, you set the language in the Set Language portlet to French instead of English (Figure 7–13).

Figure 7–13 Portlet in French



Bonjour PORTAL.

Cette page est le 'show mode' de la portlet et est generee dans en francais!

Notice that the text inside the portlet has changed, but the portlet title remains in the default language, English. You can also have the portlet set the appropriate portlet attributes (such as portlet name, portlet title, and portlet description) by pointing to a resource bundle from `provider.xml`, as described in the next section.

7.2.11.2.2 Providing Translation for Portlet Attributes In your provider's definition file, `provider.xml`, a number of attributes describing your portlet are defined such as the portlet's name and description, these are used in places, for example in your portlet's title bar in Show mode and so should be translated, too. There are two different ways of providing these translations, which one you choose is up to you. Both of these methods are outlined in the following sections:

- "Method 1: Using Resource Bundles at the Provider Level"
- "Method 2: Creating Resource Bundles at Portlet Level"

Method 1: Using Resource Bundles at the Provider Level

You can provide translations for your portlet attributes in your resource bundle(s), then specify that you want to use these resource bundles in `provider.xml`, specifying the keys you have used in your resource bundles. Using this method you can use the keys you want to, and as long as you use different keys for each corresponding attribute in your provider's various portlets you can have just one set of resource bundles that all of your provider's portlets can use. This section consists of the following tasks:

- **Updating Your Resource Bundles**

  Perform the following steps:

  1. Open your default resource bundle, `MyProviderBundle.java`.
  2. Add additional strings to your resource bundle that represent your portlet attributes and then add text for those strings:

```
package mypackage2;

import java.util.ListResourceBundle;
public class MyProviderBundle extends ListResourceBundle
{
public static String HELLO_MSG = "MyPortletHelloMessage";
public static String INFO_MSG = "MyPortletInfoMessage";
public static String PORTLET_NAME = "FirstPortletName";
public static String PORTLET_TITLE = "FirstPortletTitle";
public static String PORTLET_SHORT_TITLE = "FirstPortletShortTitle";
public static String PORTLET_DESCRIPTION = "FirstPortletDescription";
public static String TIMEOUT_MESSAGE = "FirstPortletTimeoutMessage";

public Object[][] getContents()
{
return contents;
}
static private final Object[][] contents =
{
{HELLO_MSG, "Hi"},
{INFO_MSG, "This is the show page of the portlet and it is being generated
   in the default language!"},
{PORTLET_NAME, "MyNLSPortlet"},
{PORTLET_TITLE, "My NLS Portlet"},
{PORTLET_SHORT_TITLE, "MyNLSPortlet"},
{PORTLET_DESCRIPTION, "My first ever NLS portlet, using my
   MyPortletShowPage.jsp"},
{TIMEOUT_MESSAGE, "Timed out waiting for MyNLSPortlet"}
};
}
```

3. Save `MyProviderBundle.java`.
4. Open `MyProviderBundle_fr.java`. Change it so that it contains the French strings that match the strings declared in `MyProviderBundle`.

```
package mypackage2;

import java.util.ListResourceBundle;
public class MyProviderBundle_fr extends ListResourceBundle
{
public Object[][] getContents()
{
return contents;
}
static private final Object[][] contents =
{
{MyProviderBundle.HELLO_MSG, "Bonjour"},
{MyProviderBundle.INFO_MSG, "Cette page est le 'show mode' de la portlet
   et est generee en francais!"},
{MyProviderBundle.PORTLET_NAME, "MaPremierePortlet"},
{MyProviderBundle.PORTLET_TITLE, "Ma Portlet Multi-Langue"},
{MyProviderBundle.PORTLET_SHORT_TITLE, "Ma NLS Portlet"},
{MyProviderBundle.PORTLET_DESCRIPTION, "Ma premiere portlet
   multi-langue, utilisant mon renderer"},
{MyProviderBundle.TIMEOUT_MESSAGE, "Temps d'acces a la portlet
   demandee expire"}
};
}
```

5. Save `MyProviderBundle_fr.java`.

- **Updating provider.xml**

  Perform the following steps:

  1. Open the XML provider definition file and update it to point to the resource bundle instead of using the hard-coded portlet attribute values.

```
<portlet class="oracle.portal.provider.v2.DefaultPortletDefinition">
 <id>3</id>
 <resource>mypackage2.MyProviderBundle</resource>
 <nameKey>FirstPortletName</nameKey>
 <titleKey>FirstPortletTitle</titleKey>
 <ShortTitleKey>FirstPortletShortTitle</ShortTitleKey>
 <descriptionKey>FirstPortletDescription</descriptionKey>
<timeout>10</timeout>
 <timeoutMessageKey>FirstPortletTimeoutMessage</timeoutMessageKey>
 <showEditToPublic>false</showEditToPublic>
 <hasAbout>true</hasAbout>
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html

Method 2: Creating Resource Bundles at Portlet Level

PDK-Java defines a set of resource bundle keys that you can use for providing translations for your portlet attributes. Making use of these keys means that you don't have to specify the resource bundle keys in your `provider.xml` file, as we did in "Method 1: Using Resource Bundles at the Provider Level". However, you do have to provide a separate set of resource bundles for each portlet in your provider as the keys you use for each portlet need to be the same, but their values will differ. You must perform the following tasks:

- **Updating Your Resource Bundles**

  Perform the following steps:

  1. Open your default resource bundle, `MyProviderBundle.java`.
  2. Remove any changes you made from the previous section, and import `oracle.portal.provider.v2.PortletConstants`. You can then reference the following constants instead of the strings. You do not have to declare static strings when using `PortletConstants`:

     ```
     {PortletConstants.NAME, "MyNLSPortlet"},
     {PortletConstants.TITLE, "My NLS portlet"},
     {PortletConstants.SHORTTITLE, "MyNLSPortlet"},
     {PortletConstants.DESCRIPTION, "My first ever NLS portlet"},
     {PortletConstants.TIMEOUTMSG, "Timed out waiting for MyNLSPortlet"}
     ```

  3. Save `MyProviderBundle.java`.
  4. Open `MyProviderBundle_fr.java`. Remove the portlet attributes added in the previous section, import `oracle.portal.provider.v2.PortletConstants`, and reference the constants instead of the strings.

     ```
     {PortletConstants.NAME, "MaPremierePortlet"},
     {PortletConstants.TITLE, "Ma Portlet Multi-Langue"},
     {PortletConstants.SHORTTITLE, "Ma NLS Portlet"},
     {PortletConstants.DESCRIPTION, "Ma premiere portlet multi-langue,
        utilisant mon renderer"},
     {PortletConstants.TIMEOUTMSG, "Temps d'acces a la portlet demandee
        expire"}
     ```

  5. Save `MyProviderBundle_fr.java`.

- **Updating provider.xml**

  Perform the following steps:

  1. In `provider.xml`, you need to use only one tag instead of one tag for each string as you did in "Method 1: Using Resource Bundles at the Provider Level". Delete all translated strings in the file, for example, the `<nameKey>`, `<titleKey>`, `<ShortTitleKey>`, and `<descriptionKey>` tags. Then add the following tag between the portlet id and the timeout number value:

     ```
     <resource>mypackage2.MyProviderBundle</resource>
     ```

     More On OTN

     For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

     `http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

  For more information on Java Internationalization see the *Internationalization trail* of the *Java Tutorial*.

7.2.11.3 Viewing the Portlet

Once you have updated your provider and deployed it to Oracle Containers for Java EE, refresh the provider and portal page containing your portlet. To see your resource bundles working, add the "Set Language" portlet to your page and try changing the language setting to French. Remember that the default resource bundle is English, and that selecting any other language that doesn't have a corresponding resource bundle will result in the portlet being displayed in English.

7.3 Building Struts Portlets with Oracle JDeveloper

This section describes the framework for building Struts portlets with Oracle JDeveloper for use in Oracle Portal. You will learn how to build a Struts portlet from an existing application by adding the Struts Tag Library from the Oracle Portal Developer Kit (version 9.0.4.0.2 or higher) to Oracle JDeveloper, then use the Oracle PDK Java Portlet wizard to create the Java portlet itself. This sections covers the following tasks:

- Section 7.3.1, "Oracle Portal and the Apache Struts Framework"
- Section 7.3.2, "Creating a Struts Portlet"

7.3.1 Oracle Portal and the Apache Struts Framework

This section discusses the use of the Apache Struts with Oracle Portal. Struts is an implementation of the Model-View-Controller (MVC) design pattern. The following topics are discussed in this section:

- Section 7.3.1.1, "Model View Controller Overview"
- Section 7.3.1.2, "Apache Struts Overview"
- Section 7.3.1.3, "Oracle Portal Integration with Struts"
- Section 7.3.1.4, "Summary"

7.3.1.1 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 MVC (Model View Controller) 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 allows 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.

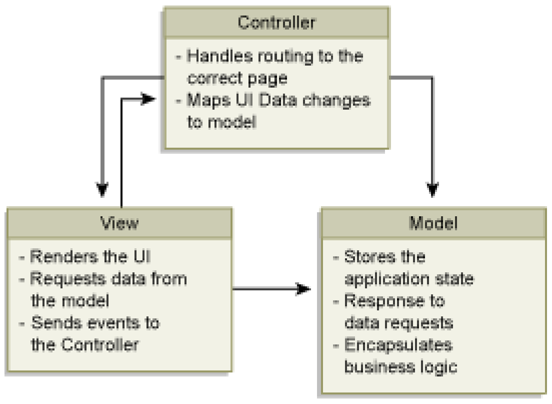
The Controller can be driven in several different ways. For example, you can use URL arguments to route the requests to the correct code. The MVC pattern itself determines the function of the Controller, not how it should work.

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 allows 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 7–14 shows how the MVC pattern applies to a conventional thin-client Web application:

Figure 7–14 The MVC Pattern


Controller
- Handles routing to the correct page
- Maps UI Data changes to model
View
- Renders the UI
- Requests data from the model
- Sends events to the Controller
Model
- Stores the application state
- Response to data requests
- Encapsulates business logic

7.3.1.2 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 allows 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`).

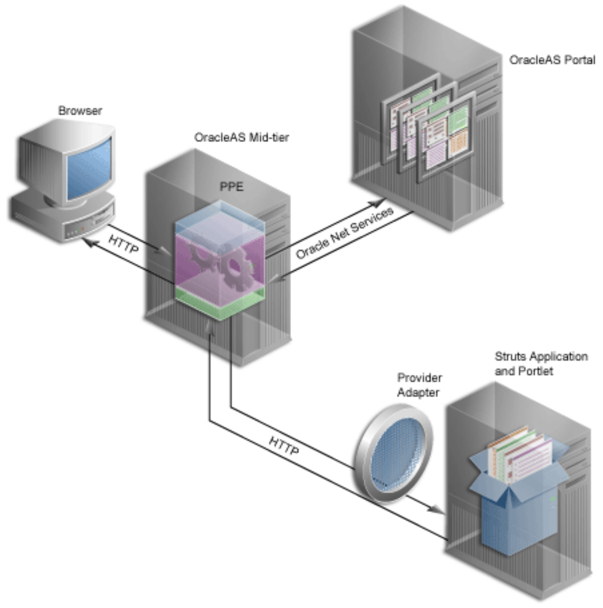
7.3.1.3 Oracle Portal Integration with Struts

The Oracle Portal Developer Kit contains numerous examples and documents regarding the usage of the Oracle 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 Oracle 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 7–15.

Figure 7–15 Integrating Struts Applications with Oracle Portal



The following code shows a portion of the provider 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>
...
```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider 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 Show 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 allows the end user to use the application within the portal.

To create a Struts portlet, you must use the Oracle 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, refer to Section 7.3.2.1, "Creating a Struts Portlet". Also, since 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, refer to the next section, Section 7.3.2.1, "Creating a Struts Portlet".

7.3.1.4 Summary

Apache Struts has become the de facto 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 Oracle Portal without completely changing the existing business logic.

Note: For more information on the Oracle Portal Developer Kit, see Portal Center (`http://www.oracle.com/technology/products/ias/portal/pdk.html`)

7.3.2 Creating a Struts Portlet

Oracle 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, refer to Section 7.3.1, "Oracle Portal and 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 7.3.2.1, "Creating a Struts Portlet"
- Section 7.3.2.2, "Registering the Provider"
- Section 7.3.2.3, "Summary"

7.3.2.1 Creating a Struts Portlet

To publish a part of an existing Struts application as portlet, we recommend 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 we recommend 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 7–16 and Figure 7–17 show how you submit a blog and save a blog entry.

Figure 7–16 Submitting a Blog

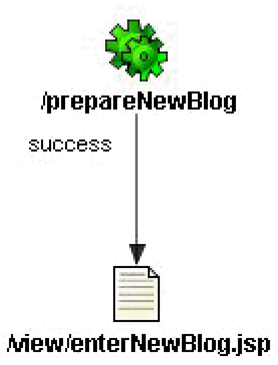
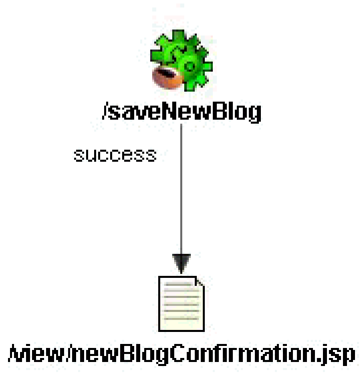


Figure 7–17 Saving a Blog Entry



`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>
```

7.3.2.1.1 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.

7.3.2.1.2 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 allowed 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 Oracle 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 Web provider. If you want to use this tag in the context of Internet Web sites that have firewalls, you must make sure the provider is directly accessible from the Internet. If it is not possible, you can deploy the images to the Oracle 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 Oracle 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.

7.3.2.1.3 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 utilize 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 on opening the wizard, see Section 6.5.1, "Creating an Oracle PDK-Java Portlet and Provider".

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. Since 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 choose **File > Erase from Disk** in Oracle JDeveloper.

6. Edit the `provider.xml` and change the following properties:

   At the provider level, perform the following:

   - If you want users to always return to the same portlet state as when they left the portal page, 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>
     ```

   - If the Struts application uses sessions (for example, the form sysnchronizer token mechanism is used or `<actionInSession>` is set to true), 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>
     ```

More On OTN

For more information on the syntax of `provider.xml`, refer to the provider Javadoc on OTN:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

7.3.2.1.4 Extending the portlet to add Portal Business Logic In your application, you should add code specific to your portal, such as the user's information, personalization, and localization. To do so, you can create a new `Action` class that is only called in the Portal context, and handles all Portal-specific business logic.

7.3.2.2 Registering the Provider

Now that your portlet is ready to be used by Oracle Portal, you must make it accessible to Oracle Portal by registering it. For information on how to register your PDK-Java portlet, refer to Section 6.5.5, "Registering and Viewing Your Oracle PDK-Java Portlet". If you chose to save the struts action in the session context, `<actionInSession>true</actionInSession>`, then you must specify the provider login frequency as **Once per user session** during registration. Setting the login frequency this way ensures the session information is passed to your struts portlet.

7.3.2.3 Summary

Oracle Application Server enables you to easily create Struts portlets using Oracle JDeveloper and publish existing Struts applications to Oracle Portal. For more information on using the Oracle JDeveloper Java Portlet wizards, refer to the beginning of this chapter. For more information on using Oracle Portal, refer to the *Oracle Fusion Middleware User's Guide for Oracle Portal* and the *Oracle Portal Online Help*.

7.3.3 Creating an Oracle Application Development Framework (ADF) 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 to Oracle Portal the same way as Struts applications. Refer to Section 7.3.2, "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 provider 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>
```

8 Creating PL/SQL Portlets

Note: In general, Oracle recommends that you build your portlets using Java rather than PL/SQL. For more information on choosing a technology for building your portlets, refer to Chapter 2, "Portlet Technologies Matrix". For more information on building your portlets using Java, refer to Chapter 6, "Creating Java Portlets".

The Oracle Portal PL/SQL APIs are implemented as a set of PL/SQL packages and objects. Database providers and portlets are deployed to a database schema as PL/SQL packages. This chapter explains how to create PL/SQL portlets based on the Oracle Portal Developer Kit-PL/SQL (PDK-PL/SQL). To make effective use of this chapter, you should already know PL/SQL and have some familiarity with the PL/SQL Web Toolkit.

This chapter contains the following sections:

More On OTN

The source code for many of the examples referenced in this chapter is available as part of PDK-PL/SQL. You can download PDK-PL/SQL from the Oracle Portal Developer Kit (PDK) page on Oracle Technology Network (OTN):

`http://www.oracle.com/technology/products/ias/portal/pdk.html`

When you unzip PDK-PL/SQL, you will find the examples in:

```
../pdk/plsql/starter
../pdk/plsql/sample
../pdk/plsql/cache
../pdk/plsql/sso
../pdk/plsql/svcex
```

You can find the reference for PDK-PL/SQL in:

```
../pdk/plsql/doc
```

8.1 Guidelines for Creating PL/SQL Portlets

When you write your portlets in PL/SQL, you should follow the best practices described in this section:

- Section 8.1.1, "Portlet Show Modes"
- Section 8.1.2, "Recommended Portlet Procedures and Functions"
- Section 8.1.3, "Guidelines for Mobile Portlets"

8.1.1 Portlet Show Modes

Just like a Java portlet, a PL/SQL portlet has a variety of Show modes available to it. A Show mode is an area of functionality provided by a portlet. The following available Show modes are described more fully in Chapter 6, "Creating Java Portlets":

- Shared Screen mode is described in Section 6.1.1.1, "Shared Screen Mode (View Mode for JPS)"
- Edit mode is described in Section 6.1.1.2, "Edit Mode (JPS and Pdk-Java)".
- Edit Defaults mode is described in Section 6.1.1.3, "Edit Defaults Mode (JPS and PDK-Java)".
- Preview mode is describe in Section 6.1.1.4, "Preview Mode (JPS and PDK-Java)".
- Full Screen mode is described in Section 6.1.1.5, "Full Screen Mode (PDK-Java)".
- Help mode is described in Section 6.1.1.6, "Help Mode (JPS and Oracle Portal)".
- About mode is described in Section 6.1.1.7, "About Mode (JPS and PDK-Java)".
- Link mode is described in Section 6.1.1.8, "Link Mode (PDK-Java)".

To check for the selected Show mode, you can use the constants in the `wwpro_api_provider` package. These constants are listed with their corresponding Show mode in Table 8–1.

Table 8–1 Show Mode Constants in wwpro_api_provider

Show mode	Constant
Shared Screen	`MODE_SHOW`
Edit	`MODE_SHOW_EDIT`
Edit Defaults	`MODE_SHOW_EDIT_DEFAULTS`
Preview	`MODE_SHOW_PREVIEW`
Full Screen	`MODE_SHOW_DETAILS`
Help	`MODE_SHOW_HELP`
About	`MODE_SHOW_ABOUT`

Table 8–1 (Cont.) Show Mode Constants in wwpro_api_provider

Show mode	Constant
Link	`MODE_SHOW_LINK`

8.1.2 Recommended Portlet Procedures and Functions

The primary goal of the portlet's code is to generate the HTML output that displays on a page for all of the Show modes required by Oracle Portal. Although it is possible to implement the portlet as a set of separate PL/SQL stored program units, organizing the portlet's code into a PL/SQL package is the best way of encapsulating related portlet code and data as a single unit in the database. You also achieve better database performance and ease of portlet maintenance.

As you may recall from Section 2.4, "Deployment Type", requests from Oracle Portal for a particular portlet go through the portlet's provider. To communicate with its portlets, the provider contains a set of required methods that make calls to the portlet code.

When implementing a portlet as a PL/SQL package, it is a good idea to organize the portlet code in parallel with the provider code. For example, when the provider needs to retrieve information about one of its portlets, it uses its `get_portlet` function. Hence, it makes sense for the portlet to contain a `get_portlet_info` function that returns the requested information when called by the provider's `get_portlet` function. Similarly, it is logical for the provider's `show_portlet` procedure to call the portlet's `show` procedure, which produces the HTML output for a requested Show mode and returns it to the provider.

Table 8–2 describes the recommended procedures and functions for a PL/SQL portlet to communicate effectively with the database provider.

Table 8–2 Recommended Functions and Procedures for PL/SQL Portlets

Procedure/Function Name	Purpose
`get_portlet_info`	Returns the portlet record to the provider.
`show`	Produces HTML output for a requested Show mode and returns it to the provider.
`register`	Initializes the portlet at the instance level. The register procedure should not contain any transaction closing statements, such as COMMIT, ROLLBACK, or SAVEPOINT. OracleAS Portal handles the closing of the transactions itself.
`deregister`	Enables cleanups at the instance level. The deregister procedure should not contain any transaction closing statements, such as COMMIT, ROLLBACK, or SAVEPOINT. OracleAS Portal handles the closing of the transactions itself.
`is_runnable`	Determines whether the portlet can be run. Security checks can be performed in this function.
`copy`	Copies the personalized and default values of portlet preferences from one portlet instance to a new portlet instance when Oracle Portal makes a copy of the page.
`describe_parameters`	Returns a list of public portlet parameters.

8.1.3 Guidelines for Mobile Portlets

Oracle Portal is capable of rendering its pages for both HTML and non-HTML (mobile) devices. When designing a portlet for a mobile device, you must consider some additional guidelines. The guidelines for mobile portlets are described fully in Section 6.1.4, "Guidelines for Mobile Portlets".

For information on how to build mobile-enabled portlets, refer to Section 8.12, "Enhancing Portlets for Mobile Devices".

8.2 Building PL/SQL Portlets with the PL/SQL Generator

To facilitate the development of database providers and PL/SQL portlets, you can use the PL/SQL Generator, a utility that creates installable PL/SQL code for a database provider and its portlets. The PL/SQL Generator is a standalone Web application that receives the provider and portlet definitions in the form of an XML file (similar in format to the `provider.xml` file). The XML tags used for the provider and portlet definition are a subset of the XML tags used for defining Web providers with PDK-Java. The output of the PL/SQL Generator is a SQL script that can be run from SQL*Plus. The script contains SQL commands for installing the provider and portlet packages in the correct order.

You can download the PL/SQL Generator along with its installation instructions from:

```
http://www.oracle.com/technology/products/ias/portal/files/plsqlgenerat
or.zip
```

The general model for working with the PL/SQL Generator is as follows:

1. Create an XML file that defines the provider and portlets that you want to build, as described in Section 8.2.1, "Creating the Input XML File".
2. Run the PL/SQL Generator using the XML file as input, as described in Section 8.2.2, "Running the PL/SQL Generator".
3. Publish the generated PL/SQL portlet, which includes the following steps:
   - Install the provider generated by the PL/SQL Generator into the database, as described in Section 8.2.3.1, "Installing the Packages in the Database".
   - Register the database provider with the Oracle Application Server, as described in Section 8.2.3.2, "Registering the Database Provider".
   - Add the generated portlet to a page, as described in Section 8.2.3.3, "Adding Your Portlet to a Page".

8.2.1 Creating the Input XML File

The source XML file starts and ends with the `<provider>` and `</provider>` tags, and can include one or more portlet definitions. Each portlet definition is bracketed by the `<portlet>` and `</portlet>` tags. A portlet definition includes the XML tags that specify values for the portlet record attributes and enable the links in the portlet header. For example, the `<name>` tag specifies the portlet name in the provider domain and the `<title>` tag specifies the portlet display name or title. When set to true, the `<showEdit>` tag enables the Edit mode for the portlet and the corresponding link in the portlet header. Table 8–3 lists the available XML tags for PL/SQL Generator input.

Table 8–3 XML Tags for PL/SQL Generator Input

XML Tag	Definition	Value Type
`provider`	Encloses provider definition tags.	Not applicable
`portlet`	Encloses portlet definition tags.	Not applicable

Table 8–3 (Cont.) XML Tags for PL/SQL Generator Input

XML Tag	Definition	Value Type
`id`	Specifies the portlet ID in the provider. This value must be unique within the provider.	string
`name`	Specifies the portlet name. The name should not contain any spaces. The generator uses the information provided in the `name` tag for the portlet package name.	string
`title`	Specifies the portlet display name.	string
`shortTitle`	Specifies the portlet short display name. This tag is useful for mobile portlets.	string
`description`	Specifies the portlet description.	string
`defaultLocale`	Specifies the language the portlet renders by default. The value is the two letter ISO language and country codes expressed as *`language.country`*.	string
`timeout`	Specifies the portlet's timeout interval in seconds.	number
`timeoutMsg`	Specifies the message to display when the portlet times out.	string
`showEdit`	Indicates whether the portlet supports Edit mode, which enables the user to personalize the portlet's properties.	Boolean
`showEditDefault`	Indicates whether the portlet supports the Edit Defaults mode, which enables page administrators to personalize the default values of the portlet's properties.	Boolean
`showDetails`	Indicates whether the portlet can be viewed in Full Screen mode. In this mode, the entire browser window is dedicated to the portlet. Full screen mode enables the portlet to show more details than when it shares the page with other portlets.	Boolean
`showPreview`	Indicates whether the portlet supports the Preview mode.	Boolean
`hasHelp`	Indicates whether the portlet supports the Help mode.	Boolean
`hasAbout`	Indicates whether the portlet supports the About mode.	Boolean
`language`	Defines the portlet's default language (for example, `en`).	string
`contentType`	Indicates the default content type supported by the portlet. The tag can take one of the following values: `wwpro_api_provider.CONTENT_TYPE_HTML` `wwpro_api_provider.CONTENT_TYPE_XML` `wwpro_api_provider.CONTENT_TYPE_MOBILE`	string
`apiVersion`	Specifies the version of the Oracle Portal PL/SQL API to which the portlet conforms. The value should be `wwpro_api_provider.API_VERSION_1`.	string
`callIsRunnable`	Indicates whether OracleAS Portal must check for the user's credentials before displaying the portlet. The default value is `true`.	Boolean

Table 8–3 (Cont.) XML Tags for PL/SQL Generator Input

XML Tag	Definition	Value Type
`callGetPortlet`	Indicates whether the portal can use the portlet record data stored in the Portlet Metadata Repository (PMR) instead of contacting the provider for the portlet record. If the portlet record (specified by provider `id`, portlet `id`, and `language`) returned by a provider does not change, then the provider should set the value for `call_get_portlet` to false. This tells the portal to use the PMR instead of making calls to the provider's `get_portlet` and `get_portlet_list` functions. An example of when a provider would not want the portal to use portlet metadata from the PMR is when the value of the portlet records is different for logged on users. The default value is `true`.	Boolean
`acceptContentType`	Specifies a comma-delimited list of content types that the portlet can produce. For example, if a portlet can produce content of both HTML and MOBILEXML type, then the tag value is: `text/html,text/vnd.oracle.mobilexml`	string
`hasShowLinkMode`	Indicates whether the portlet implements the Link mode. If the value is `false`, the portlet uses its short or full title to display a link label that references the portlet content in a mobile device. Otherwise, a personalized link can be generated in the portlet code. The default value is `false`.	Boolean
`mobileOnly`	Indicates whether the portlet is available only to mobile devices. The default value is `false`.	Boolean
`preferenceStorePath`	Specifies the base preference store path where the provider has stored the portlet personalization information. This path is used when exporting portlets.	string
`createdOn`	Defines the portlet creation date. The default value is `sysdate`.	date
`createdBy`	Identifies the user who created the portlet record.	string
`lastUpdatedOn`	Defines the most recent date on which the portlet record was changed. The default value is `sysdate`.	date
`lastUpdatedBy`	Identifies the user who most recently changed the portlet record.	string
`passAllUrlParams`	Indicates parameter passing behavior in the portlet. If the tag value is `true`, then Oracle Portal passes all parameters in the URL to the portlet. If the tag value is `false`, then the portlet receives only those parameters that are intended for the portlet. The default value is `true`.	Boolean
`cacheLevel`	Indicates a portlet's cache level. It can take one of the following values: `wwpro_api_provider.CACHE_LEVEL_SYSTEM` `wwpro_api_provider.CACHE_LEVEL_USER`	string
`rewriteUrls`	Indicates whether or not URL rewriting will be performed on the output from a portlet render request. The default value is `false`.	Boolean

Following is a sample of the input XML for the PL/SQL Generator. Mandatory information is shown in bold.

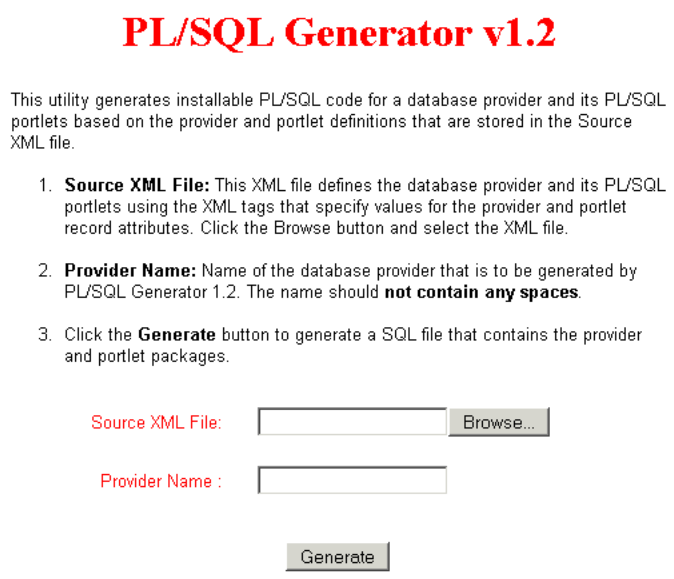
```
<!-- This is a sample provider.xml file for the PLSQL Generator 1.2 -->
<provider>
    <portlet>
        <id>1</id>
        <name>Test_Portlet</name>
        <title>Test Portlet Title</title>
        <shortTitle>Short portlet title</shortTitle>
        <description>This is a Test portlet</description>
        <timeout>30</timeout>
        <timeoutMsg>Test Portlet Timed Out</timeoutMsg>
        <showEdit>true</showEdit>
        <showEditDefault>true</showEditDefault>
        <showDetails>true</showDetails>
        <showPreview>true</showPreview>
        <hasHelp>true</hasHelp>
        <hasAbout>true</hasAbout>
        <language>en</language>
        <contentType>wwpro_api_provider.CONTENT_TYPE_HTML</contentType>
        <apiVersion>wwpro_api_provider.API_VERSION_1</apiVersion>
        <callIsRunnable>true</callIsRunnable>
        <callGetPortlet>true</callGetPortlet>
        <acceptContentType>'text/html'</acceptContentType>
        <hasShowLinkMode>false</hasShowLinkMode>
        <mobileOnly>false</mobileOnly>
        <passAllUrlParams>true</passAllUrlParams>
        <cacheLevel>wwpro_api_provider.CACHE_LEVEL_USER</cacheLevel>
        <rewriteUrls>true</rewriteUrls>
    </portlet>
</provider>
```

8.2.2 Running the PL/SQL Generator

After you have created a valid XML input file, you can run the PL/SQL Generator to generate the provider and portlet packages in the form of a SQL file as follows:

1. If you have not already done so, install the PL/SQL Generator according to the instructions that came with the download.
2. From your browser, go to the URL for the PL/SQL Generator. It should look something like the page shown in Figure 8–1.

Figure 8–1 PL/SQL Generator Page


PL/SQL Generator v1.2

This utility generates installable PL/SQL code for a database provider and its PL/SQL portlets based on the provider and portlet definitions that are stored in the Source XML file.

1. **Source XML File:** This XML file defines the database provider and its PL/SQL portlets using the XML tags that specify values for the provider and portlet record attributes. Click the Browse button and select the XML file.
2. **Provider Name:** Name of the database provider that is to be generated by PL/SQL Generator 1.2. The name should **not contain any spaces**.
3. Click the **Generate** button to generate a SQL file that contains the provider and portlet packages.

Source XML File: Browse...

Provider Name :

Generate

3. Click **Browse** and select the source XML file for the **Source XML File** field. Refer to Section 8.2.1, "Creating the Input XML File" for more information on creating the XML file.
4. In the **Provider Name** field, enter the name of the provider. The provider name must not contain any spaces. The generator uses the value entered in this field for the provider package name.
5. Click **Generate** to generate the SQL file that contains the installable PL/SQL code for the provider and portlet packages. When the browser prompts you to save or open the file, choose **Save**.
6. In the Save dialog box, change the file extension to `.sql` and revise the file name as you wish.
7. Save the file.

8.2.3 Publishing the Generated PL/SQL Portlet

After you have run the PL/SQL Generator and obtained a SQL file, you still need to perform the following tasks to make the provider and portlets available to Oracle Portal:

- Section 8.2.3.1, "Installing the Packages in the Database"
- Section 8.2.3.2, "Registering the Database Provider"
- Section 8.2.3.3, "Adding Your Portlet to a Page"

8.2.3.1 Installing the Packages in the Database

To install the generated provider and portlet packages into the database where you installed Oracle Portal, perform the following steps:

1. Start a SQL*Plus session and log in to the PORTAL schema.
2. Create a new database schema, the provider schema, to store the generated provider and portlet packages by entering the following commands in SQL*Plus:

```
create user provider_schema identified by provider_schema_password;
grant resource, connect to provider_schema;
```

3. Grant the EXECUTE privilege for the Oracle Portal APIs to the provider schema by running the `provsyns.sql` script that is located in the `ORACLE_HOME/portal/admin/plsql/wwc` directory as follows:

   ```
   @provsyns.sql provider_schema
   ```

4. Log in to the provider schema and run the generated SQL file. It will create the provider and portlet packages in the database.

8.2.3.2 Registering the Database Provider

After creating the provider and portlet packages in the database, you must register the provider with Oracle Portal before adding the PL/SQL portlet to a portal page, by performing the following steps:

1. Log in to Oracle Portal as an administrator.
2. From the Portal Builder, click the **Administer** tab then the **Portlets** tab.
3. In the **Remote Providers** portlet, click **Register a Provider**.
4. Fill in the **Name**, **Display Name**, **Timeout**, and **Timeout Message** as desired.
5. From the **Implementation Style**, list choose **Database**.
6. Click **Next** and complete the remainder of the wizard.
7. When you complete the wizard, click **Finish**.
8. From the Portlet Repository portlet, click **Display Portlet Repository**.
9. Browse the repository and find the provider that you just registered. Typically, new providers appear in the Portlet Staging Area of the repository.
10. Once you find the provider, confirm that it contains all of the portlets you created in the provider. If the provider or its portlets do not appear, then retrace the steps in this section and the preceding sections (Section 8.2.3.1, "Installing the Packages in the Database", Section 8.2.1, "Creating the Input XML File", and Section 8.2.2, "Running the PL/SQL Generator") to ensure that you correctly created and registered your provider and portlet.

8.2.3.3 Adding Your Portlet to a Page

Once your provider and its portlets appear in the repository, you can add it to a page. To add your portlet to a page, follow the instructions in the *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.3 Building PL/SQL Portlets Manually

This section describes how to build a basic PL/SQL portlet using the `hello world` sample contained in the `starter` provider sample. The `starter` provider sample, located in `..\pdkplsql\pdk\plsql\starter` in PDK-PL/SQL (`pdkplsql.zip`), consists of the following files:

- `starter_provider.pks` is the package specification of the `starter` provider.
- `starter_provider.pkb` is the package body of the `starter` provider.
- `helloworld_portlet.pks` is the package specification of the `hello world` portlet.

- `helloworld_portlet.pkb` is the package body of the `hello world` portlet.
- `snoop_portlet.pks` is the package specification of the `snoop` portlet.
- `snoop_portlet.pkb` is the package body of the `snoop` portlet.
- `insintpr.sql` is the installation script for the `starter` provider.

The general model for building PL/SQL portlets manually is as follows:

1. Modify the `hello world` portlet package specification and body to create your own portlet package, as described in Section 8.3.1, "Implementing the Portlet Package".
2. Modify the `starter` provider package specification and body to add your new portlet to a provider, as described in Section 8.3.2, "Implementing the Provider Package".
3. Add your portlet to a page, as described in Section 8.3.3, "Adding Your Portlet to a Page".

8.3.1 Implementing the Portlet Package

To modify `helloworld_portlet.pks` and `helloworld_portlet.pkb` to create your own portlet package, perform the following steps:

1. Make copies of the package specification, `helloworld_portlet.pks`, and body, `helloworld_portlet.pkb`.
2. Rename the copies to `my_first_portlet.pks` and `my_first_portlet.pkb`, respectively.
3. Open `my_first_portlet.pks` in an editor and change the name of the package to `my_first_portlet`:

```
CREATE OR REPLACE
package my_first_portlet
is
...

end my_first_portlet;
```

4. Open `my_first_portlet.pkb` in an editor and repeat the change that you made in the previous step; that is, change the name of the package to `my_first_portlet`.
5. In `my_first_portlet.pkb`, find the function named `get_portlet_info` and modify it as follows:

```
function get_portlet_info
(
     p_provider_id in integer
    ,p_language in varchar2
)
return wwpro_api_provider.portlet_record
is
    l_portlet      wwpro_api_provider.portlet_record;
begin
    l_portlet.id := starter_provider.PORTLET_FIRST;
    l_portlet.provider_id := p_provider_id;
    l_portlet.title := 'My First Portlet';
    l_portlet.name := 'My_First_Portlet';
    ...
```

6. In `my_first_portlet.pkb`, find the procedure named `show` and modify it as follows:

```
procedure show
(
   p_portlet_record        wwpro_api_provider.portlet_runtime_record
)
is
    l_portlet      wwpro_api_provider.portlet_record;
    l_text_name in varchar2(100);
    l_text in varchar2(200);
begin
...
        /*
        Display the content of the portlet in the show mode.
        Use the wwui_api_portlet.portlet_text() API when
        generating the content of the portlet so that the
        output uses the portlet CSS.
        */
        htp.p(wwui_api_portlet.portlet_text(
             p_string   =>  'Hello World - Mode Show'
            ,p_level    =>  1
            ));
        /*
        Add the functionality you want here. In this case we are adding
        a welcome message addressed to the current user.
        */
        l_text_name := 'Welcome to my first portlet ' || wwctx_api.get_user;
        l_text := wwui_api_portlet.portlet_text(
            p_string   =>  l_text_name,
            p_level    =>  1        );
        htp.p(l_text);       htp.para;
        if (p_portlet_record.has_border) then
            wwui_api_portlet.close_portlet;
        end if;
...
```

7. Save `my_first_portlet.pkb`.

8.3.2 Implementing the Provider Package

After you implement the portlet package, you must add your portlet to a provider. To modify `starter_provider.pks` and `starter_provider.pkb` to add your new portlet to a provider, perform the following steps:

1. Make copies of the package specification, `starter_provider.pks`, and body, `starter_provider.pkb`.
2. Rename the copies to `starter_provider2.pks` and `starter_provider2.pkb`, respectively.

> **Note:** If you want to create a new, empty provider, remove all references to the `hello world` and `snoop` portlets from `starter_provider2.pks` and `starter_provider2.pkb` before performing the steps that follow.

3. Open `starter_provider2.pks` in an editor.

4. Add a constant called `PORTLET_FIRST`. This constant is used as the identifier for the portlet within the provider. Hence, the constant's value must be unique within the provider.

```
CREATE OR REPLACE
package STARTER_PROVIDER
is
  /**
    * This package is used as an example to show how providers can be created
    * in the portal system.
    *
    * This provider contains the following portlets:
    *
    * Hello World (PORTLET_HELLOWORLD)
    * Snoop (PORTLET_SNOOP)
    *
    */
    PORTLET_HELLOWORLD constant integer := 1;
    PORTLET_SNOOP      constant integer := 2;
    PORTLET_FIRST      constant integer := 3;
```

5. Save `starter_provider2.pks`.
6. Open `starter_provider2.pkb` in an editor.
7. In `starter_provider2.pkb`, add a call for the new portlet's `get_portlet_info` function in the `get_portlet` function of the provider package. This step entails adding the call `my_first_portlet.get_portlet_info` in the `get_portlet` function. The `get_portlet` function allows the portal to retrieve information for the portlet when necessary.

```
function get_portlet
     p_provider_id in integer
    ,p_portlet_id in integer
    ,p_language in varchar2
)
return wwpro_api_provider.portlet_record
is
begin
    if (p_portlet_id = PORTLET_HELLOWORLD) then
        return helloworld_portlet.get_portlet_info(
             p_provider_id  => p_provider_id
            ,p_language     => p_language
            );
    elsif (p_portlet_id = PORTLET_SNOOP) then
        return snoop_portlet.get_portlet_info(
             p_provider_id  => p_provider_id
            ,p_language     => p_language
            );
    elsif (p_portlet_id = PORTLET_FIRST) then
        return my_first_portlet.get_portlet_info(
             p_provider_id  => p_provider_id
            ,p_language     => p_language
            );
    else
        raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
    end if;
end get_portlet;
```

8. In `starter_provider2.pkb`, add the new portlet to the list of portlets returned by the provider. This step entails adding the new portlet to the `get_portlet_`

`list` function of the provider. The `get_portlet_list` function tells the portal which portlets the provider implements.

```
function get_portlet_list
...
begin
    l_cnt := 0;
    if (p_security_level = false ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_HELLOWORLD
                ,p_language     => p_language
                );
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_SNOOP
                ,p_language     => p_language
                );
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_FIRST
                ,p_language     => p_language
                );
    else
        if (helloworld_portlet.is_runnable(
             p_provider_id     =>  p_provider_id
            ,p_reference_path  =>  null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_HELLOWORLD
                ,p_language     => p_language
                );
        end if;
        if (snoop_portlet.is_runnable
             p_provider_id     =>  p_provider_id
            ,p_reference_path  =>  null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                    ,p_portlet_id   => PORTLET_SNOOP
                    ,p_language     => p_language
                    );
        end if;
        if (my_first_portlet.is_runnable(
             p_provider_id     =>  p_provider_id
            ,p_reference_path  =>  null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => PORTLET_FIRST
                ,p_language     => p_language
                );
        end if;
    end if;
    return l_portlet_list;
end get_portlet_list;
```

9. In `starter_provider2.pkb`, modify the `is_portlet_runnable` function to add a call to the `is_runnable` function of the new portlet.

```
function is_portlet_runnable
(
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
)
return boolean
is
begin
    if (p_portlet_instance.portlet_id = PORTLET_HELLOWORLD) then
        return helloworld_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    elsif (p_portlet_instance.portlet_id = PORTLET_SNOOP) then
        return snoop_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then
        return my_first_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    else
        raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
    end if;
end is_portlet_runnable;
```

10. Repeat step 9 according to the information in Table 8–4.

Table 8–4 Changes to starter_provider2.pkb

Procedure/Function	Addition
`procedure register_portlet`	`elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then my_first_portlet.register(p_portlet_instance)`
`procedure deregister_portlet`	`elsif (p_portlet_instance.portlet_id = PORTLET_FIRST) then my_first_portlet.deregister (p_portlet_instance)`
`function describe_portlet_ parameters`	`elsif (p_portlet_id = PORTLET_FIRST) then return my_first_portlet.describe_parameters (p_provider_id, p_language);`
`procedure show_portlet`	`elsif (p_portlet_record.portlet_id = PORTLET_FIRST) then my_first_portlet.show(p_portlet_record)`
`procedure copy_portlet`	`elsif (p_copy_portlet_info.portlet_id = PORTLET_FIRST) then my_first_portlet.copy(p_portlet_record)`

11. Save and close `starter_provider2.pkb`.

12. Log in to Oracle Portal as you normally would.
13. From the Portal Builder, click the **Administer** tab then the **Portlets** tab.
14. From the Portlet Repository portlet, click **Display Portlet Repository**.
15. Browse the repository and find the starter provider (typically it will appear in the Portlet Staging Area of the repository). It should contain its two original portlets: `hello world` and `snoop`.
16. From a command line prompt, start SQL*Plus and connect as the owner of the `starter` provider schema.
17. Compile the new and modified PL/SQL packages in the following order:
    - `starter_provider2.pks`
    - `my_first_portlet.pks`
    - `starter_provider2.pkb`
    - `my_first_portlet.pkb`
18. If any compilation errors occur, fix and recompile them until all of the packages compile successfully.
19. From the Portlet Repository portlet, click **Display Portlet Repository**.
20. Browse the repository and find the `starter` provider again. It should now contain your new portlet, `my_first_portlet`, in addition to its original portlets.

Note: If you make changes to an existing provider or the portlet record, you need to refresh your provider before seeing the changes reflected in your Oracle Portal instance.

8.3.3 Adding Your Portlet to a Page

Your portlet should now be available for adding to pages like any other portlet in the Portlet Repository. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.4 Implementing Information Storage

Oracle Portal provides APIs for storing and retrieving individual portlet preferences, and storing and manipulating temporary data for the current session. Implementing information storage consists of the following:

- Section 8.4.1, "Implementing a Preference Store"
- Section 8.4.2, "Implementing a Session Store"

8.4.1 Implementing a Preference Store

Oracle Portal provides a set of APIs for storing and retrieving individual preferences for each unique portlet instance in a persistent manner. It provides a unique identifier for each individual, a preference store automatically mapped by user, and access mechanisms for storing and retrieving personalization information in your PL/SQL portlets.

By default, when you enable end-user personalization, **Personalize** appears on the title bar of your portlet. This link displays a form where users can choose settings for that portlet.

End-user personalization options are available through the wwpre_api_name and wwpre_api_value packages.

8.4.1.1 Using a Preference Store

In general, you can set up preference storage as follows:

1. Create the preference path using wwpre_api_name.create_path.
2. Create the preference using wwpre_api_name.create_name.
3. Set the preference values by providing the preference name and scoping level for which you want to set the value. Use wwpre_api_value.set_value_as_varchar2, set_value_as_number, or set_value_as_date for this purpose.
4. Get preference values by providing the preference name and path whenever you want to retrieve the preference value. Use wwpre_api_value.get_value_as_varchar2, get_value_as_number, or get_value_as_date for this purpose.

8.4.1.2 Creating and Accessing a Preference Store

The services example, located in ..\pdkplsql\pdk\plsql\svcex in PDK-PL/SQL (pdkplsql.zip), illustrates how you can implement preference storage. The objective is to achieve the following functionality:

- When a user clicks **Personalize**, they can enter text in two fields.
- The first field prompts for personalized text. The second prompts for a personalized portlet title.
- The values the user enters for these two fields are stored in the preference store.
- The personalized text and portlet titles are retrieved whenever that user invokes the portlet instance.

You can browse through this example as follows to see how to create the preference store, store values in it, and retrieve values from it:

1. Open the services_portlet.pkb file in an editor.

   The portlet path and preference names are provided with aliases in the constants part of your portlet definition.

   ```
   DOMAIN          constant varchar2(30) := 'provider';
   SUBDOMAIN       constant varchar2(32) := 'services';
   PORTLET_PATH    constant varchar2(256):= 'oracle.portal.pdk.servicesportlet.';
   PREFNAME_STRING constant varchar2(30) := 'services_string';
   PREFNAME_TITLE  constant varchar2(30) := 'services_title';
   ```

2. Find the register procedure. Your portlet needs to create a path for storing preferences. To do so, it calls wwpre_api_name.create_path for creating the preference path. It then calls wwpre_api_name.create_name for creating the preference name, taking the portlet path, name, and description as input parameters. Another input parameter is the p_type_name that indicates special value types. The NLSID type indicates that the value stored is an NLS id. The functions for setting and retrieving this type treat it as a number value. Apart from that, when a preference store value of this type is exported or copied, so are its associated strings. The last input parameter, the language, is obtained from a context API.

   ```
   procedure register
   (
       p_portlet_instance in wwpro_api_provider.portlet_instance_record
   ```

```
)
is
begin
    --
    -- Create a path for the portlet instance.  This is used to create
    -- the preferences for the portlet instance in the preference store.
    --
    wwpre_api_name.create_path(
        p_path => PORTLET_PATH || p_portlet_instance.reference_path
        );
    --
    -- Create the names to store the portlet preferences.
    --
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_STRING,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_TITLE,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
exception
    when others then
        raise;
end register;
```

3. The deregister procedure must eliminate the preference store with a call to `wwpre_api_name.delete_name`.

```
procedure deregister
(
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
)
is
begin
     --
     -- Delete the path used by the portlet instance.  This will delete
     -- all the names and all the values associated with the path.
     --
     wwpre_api_name.delete_path(
        p_path => PORTLET_PATH || p_portlet_instance.reference_path
        );
exception
    when others then
        raise;
end deregister;
```

4. The portlet must also get and set the values in the preference store using `wwpre_api_value.set_value` and `wwpre_api_value.get_value`. Find the `get_default_preference` function. Notice how this function loads the system level default values from the preference store. The default preferences are associated with an instance. The language strings are set in the database.

```
function get_default_preference
...
begin
    --
    -- Try to find a previously entered portlet instance string preference,
    -- if any.
    -- A portlet instance string preference is stored in the preference
    -- store and has a level of SYSTEM_LEVEL_TYPE.
    --
        p_path       => PORTLET_PATH || p_reference_path,
    l_prefs.string_id := to_char(wwpre_api_value.get_value_as_number(
        p_name       => PREFNAME_STRING,
        p_level_type => wwpre_api_value.SYSTEM_LEVEL_TYPE
        ));
    --
    -- If the value returned above is null it is an indication that there
    -- is no default string yet.  Initialize the string id to 0 to indicate
    -- this and load the default string value.
    --
    if (l_prefs.string_id is null or to_number(l_prefs.string_id) = 0) then
        wwpre_api_value.set_value_as_number(
            p_path       => PORTLET_PATH || p_reference_path,
            p_name       => PREFNAME_STRING,
            p_level_type => wwpre_api_value.SYSTEM_LEVEL_TYPE,
            p_level_name => null,
            p_value      => 0
            );
...
end get_default_preference;
```

5. Find the `show` procedure. Notice the behavior when the portlet is in Edit Defaults or Edit mode. Note also how `p_action` is populated when the user clicks **APPLY**, **CANCEL**, or **OK**. Once the form is submitted, the `show` procedure of the portlet is called again and, if the `p_action` parameter is not null, then the `save_prefs` procedure is called to save the personalizations and redirect to the relevant page.

```
procedure show
(
    p_portlet_record        wwpro_api_provider.portlet_runtime_record
)
is
    l_str varchar2(32000);
    l_pref_record preference_record;
    l_action   varchar2(10);
    l_names    owa.vc_arr;
    l_values   owa.vc_arr;
begin
...
    elsif (p_portlet_record.exec_mode =
                wwpro_api_provider.MODE_SHOW_EDIT)
        or (p_portlet_record.exec_mode =
                wwpro_api_provider.MODE_SHOW_EDIT_DEFAULTS)
    then
        wwpro_api_parameters.retrieve(l_names, l_values);
        for i in 1..l_names.count loop
            if (upper(l_names(i)) = upper('p_string')) then
                l_pref_record.string := l_values(i);
            elsif l_names(i) = 'p_title' then
                l_pref_record.title_string := l_values(i);
            elsif l_names(i) = 'p_action' then
```

```
                l_action := l_values(i);
            end if;
        end loop;
        if (l_action in (ACTION_OK,ACTION_APPLY,ACTION_CANCEL)) then
            if (p_portlet_record.exec_mode =
                                  wwpro_api_provider.MODE_SHOW_EDIT) then
                save_prefs(p_string => l_pref_record.string,
                           p_title  => l_pref_record.title_string,
                           p_action => l_action,
                           p_level  => wwpre_api_value.USER_LEVEL_TYPE,
                           p_portlet_record  => p_portlet_record);
            else
                save_prefs(p_string => l_pref_record.string,
                           p_title  => l_pref_record.title_string,
                           p_action => l_action,
                           p_level  => wwpre_api_value.SYSTEM_LEVEL_TYPE,
                           p_portlet_record  => p_portlet_record);
            end if;
        else
            show_edit(p_portlet_record  => p_portlet_record);
        end if;
...
end show;
```

6. The show_edit procedure renders the page for Edit or Edit Defaults mode. It renders two text fields that allow the user to change the personalizable values in a form with three buttons (**Apply**, **OK**, and **Cancel**). Note that this function uses the wwpro_api_adapter.open_form to create the HTML form with the correct action attribute for the <FORM> tag and with the correct hidden fields. It is important to use this procedure to create the <FORM> tag if you want to use the portlet with the Federated Portal Adapter from remote Oracle Portal instances.

```
procedure show_edit
(
    p_portlet_record in wwpro_api_provider.portlet_runtime_record
)
is
    l_prefs    preference_record;
    l_text_prompt_string  varchar2(30);
    l_title_prompt_string varchar2(30);
begin
...
    htp.centeropen;
    htp.tableOpen(cattributes => 'BORDER="1" WIDTH=90%');
    htp.tableRowOpen;
    htp.p('<TD>');
    --
    --  This procedure call creates the <FORM> tags with a set of
    --  standard parameters.  Using this procedure makes the
    --  personalization page work through the pl/sql http adapter.
    --
    wwpro_api_adapter.open_form(p_formattr => 'NAME="services"',
                                p_prr      => p_portlet_record);
    htp.p('</TD>');
    htp.tableRowClose;
    htp.tableClose;
    htp.centerclose;
    htp.formclose;
end show_edit;
```

7. Review the following procedures and functions, which are related to the preference storage implementation in this example:
   - `get_user_preference` retrieves the user personalized string and title for the portlet.
   - `save_prefs` is invoked to save the preferences to the preference store when the user clicks **OK** or **Apply** after making personalization changes.
   - `entered_text_is_valid` checks to see if the text entered in the personalizable text fields is valid.
8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.4.2 Implementing a Session Store

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement a session store. The objective is to achieve the following functionality:

- When a user invokes this portlet, it displays text that reads: "This portlet has rendered *x* times in this session." *x* is the number of times the portlet has been rendered.
- Every time the user invokes the portlet, the counter increases by 1.
- Clicking **Details** in the portlet enables the user to reset the counter using **Clear**. After clearing the counter, the counter starts again from zero.

Creating and Accessing a Session Store

You can browse through this example as follows to see how to create the session store, store values in it, and retrieve values from it:

1. Open the `services_portlet.pkb` file in an editor.

   The domain and subdomain definitions for your session object are provided with aliases in the constants part of your portlet definition.

```
DOMAIN          constant varchar2(30) := 'provider';
SUBDOMAIN       constant varchar2(32) := 'services';
PORTLET_PATH    constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING constant varchar2(30) := 'services_string';
PREFNAME_TITLE  constant varchar2(30) := 'services_title';
```

2. Find the `clear_count` procedure. `clear_count` is called from the `show` procedure when the user clicks **Clear** to reset the counter. `clear_count` calls `wwsto_api_session.load_session` to load the session object. Then, it calls `wwsto_api_session.set_attribute` to set the counter to zero. Lastly, it saves the session object by calling `save_session`.

```
procedure clear_count
(
    p_action          in varchar2,
    p_back_url        in varchar2,
    p_reference_path  in varchar2
)
```

```
is
    ex_counter integer;
    session_parms &&1..wwsto_api_session;
begin
    --
    -- Clear the display counter.
    --
    if (p_action = ACTION_CLEAR) then
        --
        -- Load the session object that contains the display counter
        --
        session_parms :=
            &&1..wwsto_api_session.load_session (DOMAIN,SUBDOMAIN);
        ex_counter :=
            session_parms.get_attribute_as_number(
                'ex_counter' || p_reference_path);
        --
        -- Reset the display counter.
        --
        ex_counter := 0;
        session_parms.set_attribute(
        'ex_counter' || p_reference_path, ex_counter);
        --
        -- Save the changes to the database immediately to avoid any
        -- data consistency problems with the data stored in the
        -- session object.
        --
        session_parms.save_session;
    end if;
    owa_util.redirect_url(curl=>p_back_url);
end clear_count;
```

3. Find the `show_contents` procedure. `show_contents` is called from the `show` procedure to retrieve the counter, increment it by one, and save the value in the session store. Notice how it retrieves the session object to display the number of times the user has rendered the portlet. It also retrieves the counter value with `get_attribute_as_number` and increments the counter for every invocation of this procedure.

```
procedure show_contents
(
    p_portlet_record  wwpro_api_provider.portlet_runtime_record
)
is
    l_prefs preference_record;
    session_parms &&1..wwsto_api_session;
    ex_counter integer;
    l_portlet wwpro_api_provider.portlet_record;
    l_str varchar2(32000);
begin
    --
    -- In this mode a session counter is used to indicate
    -- the number of invocations of this portlet during the
    -- current session.  The counter is stored in the session
    -- store.
    --
    session_parms :=
        &&1..wwsto_api_session.load_session(DOMAIN,SUBDOMAIN);
    ex_counter    :=
        session_parms.get_attribute_as_number(
            'ex_counter' || p_portlet_record.reference_path);
    if (ex_counter is null) then -- first invocation
        session_parms.set_attribute(
            'ex_counter' || p_portlet_record.reference_path,1);
        ex_counter := session_parms.get_attribute_as_number(
            'ex_counter' || p_portlet_record.reference_path);
    else -- on every invocation increase by 1
        ex_counter := ex_counter + 1;
        session_parms.set_attribute(
            'ex_counter'
            || p_portlet_record.reference_path, ex_counter);
    end if;
    session_parms.save_session;
...
end show_contents;
```

4. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
5. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.5 Using Parameters

The functionality of portlets can be extended with the help of parameters. The business logic implemented by portlets may produce different HTML output depending on the parameters passed to the page. By using portlet parameters, you can navigate within the portlet in Shared Screen mode without changing the current page. Portlets can also communicate with each other through parameters.

Portlet parameters are structured as name-value pairs. These pairs map directly to the URL parameter passing format by using the `GET` submission method or can use the HTTP message body by using the `POST` submission method. Portlets can also expose their parameters to Oracle Portal. When added to a page, these portlets can accept values in the form of page parameters created by the page designer.

Note: Portlet parameter names should not start with an underscore (_) because those parameters are reserved for internal use by Oracle Portal and are not passed to the portlet.

Portlets do not have direct access to the URL, the HTTP message body, or the page parameters. To retrieve the parameter values, portlets must call the Oracle Portal PL/SQL parameter APIs provided in the `wwpro_api_parameters` package.

Oracle Portal offers the following types of parameters:

Caution: You cannot mix the usage of public and private parameters in a portlet. To enable public parameters for your portlet, you must take steps that preclude the usage of private parameters and vice versa.

- **Private portlet parameters** enable the implementation of internal navigation in your portlet.

- **Public portlet parameters** let you pass control over the data flow of your portlet to the page designer. The page designer can map the public portlet parameters to their page parameters, provide default values, and allow users to personalize those values.
- **Page parameters** are defined in a simple user interface by page designers. These page parameters can be mapped to public portlet parameters in order for the page designer to pass parameter values from the page to the portlets on it.

For more information about parameters, refer to Section 2.12, "Public Portlet Parameters Support" and Section 2.13, "Private Portlet Parameter Support".

8.5.1 Passing Private Parameters

You can use either `GET` or `POST` HTML submission methods when passing private portlet parameters. The `GET` method uses the URL to pass the parameters, whereas the `POST` method places the parameters in an HTTP message body. For both methods, you must specify the portlet instance on the portal page, how the parameter is called, and the value of the parameter.

There are the following two types of private portlet parameters:

- **Qualified parameters** ensure that a private portlet parameter is not read by any other portlet on the page. The reference path, which is assigned when the portlet is added to a page, is the unique prefix of the parameter. For example, `http://page_url?277_MAP_368673.region=Europe`. The qualified parameter's reference path is `277_MAP_368673`, the name is `region`, and the value is `Europe`. For private parameters, we strongly recommend that you always use qualified parameters.
- **Unqualified parameters** have no information about the portlet instance and can be read by any portlet on the page. For example, `http://page_url?region=Europe`. The unqualified parameter's name is `region` and its value is `Europe`. For private parameters, we strongly recommend that you avoid unqualified parameters.

8.5.2 Passing Page Parameters and Mapping Public Portlet Parameters

Public portlet parameters enhance the flexibility of your portlets by enabling page designers to reuse your portlets on multiple pages. As a result, page designers do not have to ask you to make changes to the portlet code when adding the portlet to different pages. By using public portlet parameters, any portlet on a page can easily receive its value from the mapped page parameter, regardless of the portlet parameter name.

For example, suppose you have a page parameter named `dept_id`. Three portlets need this value, but one portlet calls it `dept`, another calls it `deptno`, and still another `department_id`. Mapping the page parameter enables all three portlets to receive the value from the `dept_id` parameter and place it in the appropriate portlet parameter. Furthermore, the page designer may set a default value (for example, department 20) that can be personalized by users (for example, department 30) and applied to all three portlets.

The general model for passing public and page parameters is as follows:

1. Enable public parameters in the portlet record by setting `pass_all_url_params` to `false`. This ensures that the portlet is only passed parameters intended for that portlet.

2. Declare the public parameters in the provider's `describe_portlet_parameters` function. For each of the portlets that belong to the provider, this procedure should return a list of the parameters that the portlet accepts in the form of a PL/SQL table of records of the type:

```
type portlet_parameter_table is table of
portlet_parameter_record index by binary_integer;
```

3. Provide descriptive information for the parameters in the portlet's `describe_parameters` function. For example:

```
function describe_parameters
   (p_provider_id in integer, p_language in varchar2)
return wwpro_api_provider.portlet_parameter_table
   is
l_params wwpro_api_provider.portlet_parameter_table;
   begin
     l_params(1).name := 'dept_id';
     l_params(1).datatype := wwpro_api_provider.STRING_TYPE;
     l_params(1).description := 'Defines a department ID';
     l_params(1).display_name := 'Department ID';
     return l_params;
end describe_parameters;
```

4. Assign values to the public parameters. Public parameters typically get their values through page parameters. Page parameters are usually assigned default values by the page designer and the user can then personalize the value at runtime. Alternatively, page parameter values can be assigned in the calling URL. For more information about how page designers can use page parameters, refer to the *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.5.3 Retrieving Parameter Values

Regardless of whether you are using private or public parameters, you use the same APIs to retrieve their values. Portlets obtain their parameters by calling the following PL/SQL parameter APIs in the `wwpro_api_parameters` package:

- `wwpro_api_parameters.get_value` returns the parameter value that is specified by a given parameter name. Parameter names are not case sensitive, whereas parameter values are case sensitive. For example:

```
l_region := wwpro_api_parameters.get_value
   (p_name => 'region',
    p_reference_path => p_portlet_record.reference_path);
```

- `wwpro_api_parameters.get_values` returns an array of parameter values. This function returns all the values that are associated with a single parameter name or an empty list if no matches are found. Some business logic may require multiple selections, when multiple values are passed to the portlet by using the same parameter name. Portlets can take one or more values of the same parameter. For example:

```
l_region_values owa.vc_arr;
...
l_region_values := wwpro_api_parameters.get_values
   (p_name = 'region',
    p_reference_path => p_portlet_record.reference_path);
```

- `wwpro_api_parameters.get_names` returns the names of the parameters that are passed on to a specified portlet that is identified by the reference path. The returned list is a PL/SQL table of the `owa.vc_ar` type that is defined as follows:

  ```
  type vc_arr is table of varchar2(32000) index by binary_integer;
  ```

  > **Note:** Portlet parameter names should not start with an underscore (_) because those parameters are reserved for internal use by Oracle Portal and are not passed to the portlet.

  For example:

  ```
  l_names owa.vc_arr;
  ...
  l_names := wwpro_api_parameters.get_names
     (p_reference_path => p_portlet_record.reference_path);
  ```

- `wwpro_api_parameters.retrieve` returns the names and values of all of the portlet's parameters. For example:

  ```
  procedure show_portlet
     ( p_portlet_record in out
           wwpro_api_provider.portlet_runtime_record )
  is
     l_names owa.vc_arr;
     l_values owa.vc_arr;
  ...
  begin
  ...
     wwpro_api_parameters.retrieve (l_names, l_values);
     for i in 1..l_names.count loop
       htp.p('Parameter Name: '||l_names(i));
       htp.p('Parameter Value: '||l_values(i));
       htp.br;
     end loop;
  ...
  end show_portlet;
  ```

8.6 Accessing Context Information

Whenever a user accesses a page in Oracle Portal, a public session is established. When the user logs in to Oracle Portal, the public session becomes an authenticated session. This session contains several pieces of context information about the user, such as user name, current session ID, IP address, and language preference. It also includes supporting information such as the Oracle Portal schema currently in use.

Session context services return information about a user's session and are available through the `wwctx_api` package.

8.6.1 Using Context Information

The general model for working with the session context is as follows:

1. Identify the piece of information you require for your functionality.
2. Use the appropriate method from `wwctx_api` to get and optionally set this value.

Table 8–5 lists the function calls used to obtain the various pieces of session information.

Note: For more information on the context APIs, see the PL/SQL API Reference. The API Reference can be found on Portal Center (`http://portalcenter.oracle.com`) or, if you downloaded PDK-PL/SQL (`pdkplsql.zip`), in `..\pdkplsql\pdk\plsql\doc`.

Table 8–5 Context Information Function Calls

Session Information	Function Call
Current user	`wwctx_api.get_user`
Login status of user	`wwctx_api.is_logged_on`
Login time	`wwctx_api.get_login_time`
Language	`wwctx_api.get_nls_language`
Current session id	`wwctx_api.get_sessionid`
IP address of user client	`wwctx_api.get_ip_address`
User schema	`wwctx_api.get_db_user`
Oracle Portal schema	`wwctx_api.get_product_schema`
Oracle Portal version	`wwctx_api.get_product_version`

8.6.2 Using wwctx_api to Obtain Context Information

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can obtain session information using the `wwwctx_api` package. You can browse through this example as follows to see how the function calls are implemented in a portlet:

1. Open the `services_portlet.pkb` file in an editor.
2. Find the `get_portlet_info` function.
3. Notice the usage of `wwctx_api.get_user` to derive the user information and set that value in the portlet information record:

```
...
    l_portlet.timeout               := null;
    l_portlet.timeout_msg           := null;
    l_portlet.created_on            := to_date('10/19/2000', 'MM/DD/YYYY');
    l_portlet.created_by            := wwctx_api.get_user;
    l_portlet.last_updated_on       := to_date('10/19/2000', 'MM/DD/YYYY');
    l_portlet.last_updated_by       := wwctx_api.get_user;
    l_portlet.has_show_edit_defaults := true;
    l_portlet.has_show_preview      := true;
    l_portlet.preference_store_path := PORTLET_PATH;
...
```

4. `wwctx_api.get_user` is used similarly in various places throughout `services_portlet.pkb`. Search the code for other occurrences of `wwctx_api.get_user`.
5. Another example of getting context information occurs in the `is_runnable` function:

```
function is_runnable
(
     p_provider_id in integer
    ,p_reference_path in varchar2
)
return boolean
is
begin
    --
    -- Portlet security check.  It allows the portlet to be visible
    -- if the user is logged on, that is, the current session is not a
    -- public session.
    --
    return wwctx_api.is_logged_on;
end is_runnable;
```

6. In the `register` procedure, `wwctx_api.get_nls_language` is used to get the language:

```
    --
    -- Create the names to store the portlet preferences.
    --
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_STRING,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
    wwpre_api_name.create_name(
        p_path        => PORTLET_PATH
            || p_portlet_instance.reference_path,
        p_name        => PREFNAME_TITLE,
        p_description => 'Single custom row in '
            || 'Introductory Example portlet.',
        p_type_name   => 'NLSID',
        p_language    => wwctx_api.get_nls_language);
```

7. Close `services_portlet.pkb`. You can implement session context similarly but based upon your own functional requirements.
8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.7 Implementing Portlet Security

Portlet security refers to the techniques and methods used by portlets to control their access by end users. The portlets leave authentication to Oracle Portal and trust that the portal will return them to the correct, validated user upon request.

Oracle Portal strictly controls access to information and applications by assigning specific privileges to users and groups. Portal security services allow you to specify access control programmatically and check for the appropriate privileges at runtime. Security mechanisms used by portlets ensure that only authorized users gain access to these portlets. These security services are available through the `wwsec_api` package.

Portlet security is invoked when a portlet is displayed on a portal page and when a portlet is returned in a portlet list by the `get_portlet_list` function for database providers. Security services in the Portal framework have the following key features:

- **Portlet Display:** Before a portlet is displayed on a page, the provider checks for the portlet's access privileges. The provider needs to define the `is_portlet_runnable` function which calls the portlet's `is_runnable` function to check access privileges.
- **User Group:** You can find which default group a user belongs to by using the `wwsec_api.get_defaultgroup` function.
- **Check Privileges:** You can find whether a user or group has the required privileges to personalize a portlet by using the `wwsec_api.has_privilege` function.
- **Highest Privilege:** You can find the highest available privilege of a user across all groups by using the `wwsec_api.get_privilege_level` function.
- **Accessible Objects:** You can find all the objects to which a user has access, given a privilege level, by using the `wwsec_api.accessible_objects` function. You can find other similar associated functions in the API documentation. The API Reference can be found on Portal Center (`http://portalcenter.oracle.com`) or, if you downloaded PDK-PL/SQL (`pdkplsql.zip`), in `..\pdkplsql\pdk\plsql\doc`.

8.7.1 Using Security

To implement PL/SQL portlet security, the portal requires the function `is_portlet_runnable` be implemented by database providers. The actual implementation of this function is up to the application; that is, the security scheme that determines whether the current user has enough privileges to access the portlet is defined by the individual portlet implementation. The portal also requires the function `get_portlet_list` for database providers to return the set of portlets that are accessible by the current user.

Guidelines for Using the Security APIs

The portlet security mechanism may use the context and security subsystem APIs and infrastructure. The context APIs can be used to retrieve information about the current user. The security subsystem can be used to check the privileges of the current user.

Note: For more information on the context and security subsystem APIs, see the PL/SQL API Reference. The API Reference can be found on Portal Center (`http://portalcenter.oracle.com`) or, if you downloaded PDK-PL/SQL (`pdkplsql.zip`), in `..\pdkplsql\pdk\plsql\doc`.

While using these APIs, keep in mind the following:

- Only authorized users should be able to see your portlet in the Add Portlet dialog. This objective can be accomplished by implementing the `is_portlet_runnable` function in the provider. You can also allow public access to your portlet.
- If a portlet does not want to render itself to a user, it should return no HTML or return an exception that the page engine will ignore. It should not return an error message. Doing so adds unnecessarily to the error stack, which has its limits. Refer to Section 8.9, "Implementing Error Handling" for more information.

- Portlet security allows the portlet to perform a runtime security check to ensure that the current user has the necessary authorization to access the portlet.
- When a portlet is rendered in Show mode, it may call the `is_runnable` function for database providers to determine whether the portlet should be displayed for the currently logged on user. The portal does not make the call to this function directly. It is not a requirement, however, for the portlet to make this call. The portlet should make this call in its Show mode only if it implements portlet security.
- The result of the call to `is_runnable` determines whether the portlet is actually displayed. If the result is `true`, the portlet displays; otherwise it does not display. The portlet is rendered in Show mode when it is displayed in a portal page.
- When a portlet is returned in a portlet list by a call to the provider function `get_portlet_list`, the value of the `p_security_level` parameter determines the purpose of the function call. When the call is made from the Portlet Repository refresh operation in order to retrieve the master list of portlets that the provider implements, the parameter `p_security_level` has a value of `false`. This setting indicates to the provider that no portlet security check should be made and a master list of all the portlets that the provider implements must be returned. The master list of portlets returned in this case is used to populate the Portlet Repository for that provider.
- If the value of `p_security_level` is `true`, then it is up to the provider implementation to decide whether portlet security should be performed. If portlet security is implemented, the provider may return a different list of portlets depending on the current user.
- When the Portlet Repository is displayed, Oracle Portal calls the `is_portlet_runnable` function for database providers for each of the portlets that exist in the Portlet Repository. This step is done to display only the portlets that the currently logged on user is authorized to see. One example where the Portlet Repository is displayed is in the Add Portlets dialog.

8.7.2 Coding Security

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement security. You can browse through this example as follows to see how the security functions are implemented in a portlet:

1. Open the `services_provider.pkb` file in an editor.
2. Find the `is_portlet_runnable` function. This function calls the security implementation through the portlet's `is_runnable` function to check portlet access privileges.

```
function is_portlet_runnable
(
    p_portlet_instance in wwpro_api_provider.portlet_instance_record
)
return boolean
is
begin
    if (p_portlet_instance.portlet_id = SERVICES_PORTLET_ID) then
        return services_portlet.is_runnable(
             p_provider_id     =>  p_portlet_instance.provider_id
            ,p_reference_path  =>  p_portlet_instance.reference_path
            );
    else
        raise wwpro_api_provider.PORTLET_NOT_FOUND_EXCEPTION;
    end if;
end is_portlet_runnable;
```

3. Find the `get_portlet_list` procedure. `get_portlet_list` allows the portlet to be included in the list of portlets implemented by this provider. `get_portlet_list` first checks the security flag (`p_security_level`) to find out whether security is enabled. If the flag is set to true, `get_portlet_list` uses `is_runnable` to check whether the portlet is accessible. The value of the `p_security_level` parameter indicates whether to perform security checks before returning a portlet in the list. When a portlet repository refresh operation retrieves the master list of portlets implemented by the provider, `p_security_level` has a value of `false`. A value of `false` means the provider does not need to perform a security check and that a master list of all of the portlets implemented by the provider must be returned. The master list of portlets returned is used to populate the portlet repository for that provider. If the value of `p_security_level` is `true`, then the provider implementation decides whether to perform portlet security checks. If portlet security is implemented, the provider may return a different list of portlets depending on the currently logged on user.

```
function get_portlet_list
...
    if (p_security_level = false) then
        l_cnt := l_cnt + 1;
        l_portlet_list(l_cnt) := get_portlet(
             p_provider_id  => p_provider_id
            ,p_portlet_id   => SERVICES_PORTLET_ID
            ,p_language     => p_language
            );
    else
        if (services_portlet.is_runnable(
             p_provider_id     =>  p_provider_id
            ,p_reference_path  =>  null)
        ) then
            l_cnt := l_cnt + 1;
            l_portlet_list(l_cnt) := get_portlet(
                 p_provider_id  => p_provider_id
                ,p_portlet_id   => SERVICES_PORTLET_ID
                ,p_language     => p_language
                );
        end if;
...
end get_portlet_list;
```

4. Open the `services_portlet.pkb` file in an editor.

5. Find the `show` procedure. Before displaying a portlet, the `show` procedure runs a security check to determine whether the current user is allowed to see the portlet.

```
procedure show
...
    -- Perform a security check
    if (not is_runnable(
         p_provider_id     =>  p_portlet_record.provider_id
        ,p_reference_path  =>  p_portlet_record.reference_path)
    ) then
        wwerr_api_error.add(
                  DOMAIN, SUBDOMAIN,
                  'securityerr', 'services_portlet.show');
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
...
end show;
```

6. Find the `is_runnable` function. `is_runnable` is the place where you implement your security checks. In this example, the security check is quite simple. If the user is logged on (that is, not in a public session), then the function returns `true` and the portlet is displayed to the user. For your own purposes, you could, of course, code much more complex security checks in the `is_runnable` function.

```
function is_runnable
(
     p_provider_id in integer
    ,p_reference_path in varchar2
)
return boolean
is
begin
    --
    -- Portlet security check.  It allows the portlet to be visible
    -- if the user is logged on, that is, the current session is not a
    -- public session.
    --
    return wwctx_api.is_logged_on;
end is_runnable;
```

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.

8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.8 Improving Portlet Performance with Caching

Oracle Portal provides for the caching of PL/SQL portlets. This functionality permits PL/SQL portlets to cache their Web content on the middle tier. Subsequent requests for the content may be retrieved from the cache, with or without validation from the database, decreasing the database workload.

Oracle Portal provides three types of caching for your PL/SQL portlets:

- **Validation-based caching** compares a key value to check whether the contents of the cache are still valid. If the key value does not change, it uses the cached content. Otherwise, it makes a round trip to the portal node to fetch the portlet content.

- **Expiry-based caching** uses a given expiration period for the contents of the cache when rendering the portlet. This form of caching is useful for content that changes infrequently or at very regular intervals (for example, every day at the close of business).

- **Invalidation-based caching** is the most complex form of caching but also the most flexible. The objects in Oracle Web Cache are considered valid as long as they are not invalidated explicitly. You can also combine invalidation-based caching with either expiry-based or validation-based caching.

Because Oracle Portal supports user personalization of pages and portlets, the view of a page can vary from one user to another. Oracle Portal's caching is designed to allow content to vary on a per-user basis, even if the URL is the same across all users. Therefore, portal objects can be cached at either the user level or the system level and can be described as follows:

- **User-level caching** is for a specific user. The cache entries are unique for that user and cannot be accessed by other users.
- **System-level caching** is for all users. One cache entry is used for all users. Examples of content that might be suitable for system-level caching are page banners and news portlets.

When a database provider issues a request for a portlet, the request is sent to the portlets's `show` procedure. This procedure accepts the `portlet_runtime_record` as a parameter. This record structure contains fields that can be examined and set by the portlet to enable caching. The caching control fields of this record are as follows:

- **`caching_key`**: This value is communicated in the `ETAG` header for this request and returned back to the portlet provider in subsequent requests. Setting this field enables validation-based caching.
- **`caching_period`**: This field enables expiry-based caching. The value is the number of minutes the content should be held in the cache. This mode overrides validation-based caching. If a value is set for this field, then the `caching_key` field is ignored.
- **`caching_level`**: This field defines whether the content is meant for general use or for a specific user. The valid values are `SYSTEM` and `USER`.

8.8.1 Using Caching

The general model for working with portlet caching varies according to the type of caching you choose. To a great extent, the type of caching you choose depends on the portlet content. If the portlet content changes at fairly regular intervals (for example, at the close of business every day), then it probably makes sense to use expiry-based caching. If the portlet content changes at irregular intervals, then validation- or invalidation-based caching is probably best.

8.8.1.1 Validation-Based Caching

If you choose validation-based caching, the general model is as follows:

1. Set the `caching_key` field of the `portlet_runtime_record` parameter. Add a check to compare the value of the current key with the value of the `caching_key` field of the `portlet_runtime_record` parameter. Note that the first time the `show` procedure is called, the key is null and its value must be set.
2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.

8.8.1.2 Expiry-Based Caching

If you choose expiry-based caching, the general model is as follows:

1. Set the `caching_period` field of the `portlet_runtime_record` parameter to the desired interval for the cache (in minutes).

2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.

8.8.1.3 Invalidation-Based Caching

If you choose invalidation-based caching, the general model is as follows:

1. Indicate to Oracle Portal that it must generate specific headers for Oracle Web Cache by calling `wwpro_api_provider.USE_INVALIDATION`.
2. Determine whether you want to use system or user level caching. Set the `caching_level` field of the `portlet_runtime_record` parameter accordingly.
3. Optionally, set up validation- or expiry-based caching as well.
4. Add invalidation logic to your portlet where needed (for example, when the portlet is personalized) and make appropriate calls to `wwpro_api_invalidation`.

8.8.2 Configuring and Monitoring the Cache

The *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* describes how to configure caching as well as how to monitor and tune performance.

8.8.3 Implementing Validation-Based Caching

The caching example, located in `..\pdkplsql\pdk\plsql\cache` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement validation and expiry-based caching. You can browse through this example as follows to see how the validation-based functions are implemented in a portlet:

1. Open the `validcache_portlet.pkb` file in an editor.
2. At the very top of the file, notice the aliases for the caching level constants.

```
CREATE OR REPLACE
package body VALIDCACHE_PORTLET
is
    -- Caching Constants
    CACHE_LEVEL_SYSTEM constant varchar2(10) := 'SYSTEM';
    CACHE_LEVEL_USER   constant varchar2(10) := 'USER';
```

3. Find the `show` procedure. Notice first that the `p_portlet_record` is an `in` and `out` parameter for this procedure.

```
procedure show
(
    p_portlet_record   in out     wwpro_api_provider.portlet_runtime_record
)
```

4. In the procedure's security check, the caching fields of `p_portlet_record` are set to null if the security check fails.

```
begin
    if (not is_runnable(
         p_provider_id      => p_portlet_record.provider_id
        ,p_reference_path   => p_portlet_record.reference_path)
    ) then
        -- Set it to null so that cache does not get used even if exists
        p_portlet_record.caching_level := null;
        p_portlet_record.caching_key := null;
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
```

5. After that, the procedure calls the `get_cache_key` function to get the cache key's value and assign it to a temporary value:

```
    --
    -- CACHE IS VALID?
    --
    l_cache_key := get_cache_key();
```

6. Find the `get_cache_key` function, which is referenced from the `show` procedure. This function generates a key for the portlet. You can implement your own logic here based upon your portlet's requirements.

```
function get_cache_key
return varchar2
is
    l_date date;
begin
    select sysdate into l_date from  dual;
    return trim(substr(to_char(l_date, 'YYYY:MM:DD:HH:MI:SS'),1,18));
exception
    when others then
        null;
end get_cache_key;
```

7. Now return to the `show` procedure. Notice how the code checks your `portlet_runtime_record` parameter for the current values of the `caching_key` and the `caching_level`. This same piece of code can compare your `caching_key` values.

```
    if p_portlet_record.caching_level = CACHE_LEVEL_SYSTEM then
        if l_cache_key is not null then
            -- Cache exists for the user, overwrite it
            p_portlet_record.caching_level := CACHE_LEVEL_USER;
            p_portlet_record.caching_key := l_cache_key;
        else
            return; -- System cache is still valid.
        end if;
    elsif p_portlet_record.caching_level = CACHE_LEVEL_USER then
        if p_portlet_record.caching_key != l_cache_key then
            -- cache has expired. reset it
            p_portlet_record.caching_key := l_cache_key;
        else
            return; -- User cache is good as gold
        end if;
    elsif p_portlet_record.caching_level is null then
        if p_portlet_record.caching_key is not null then
            -- Cache does not exists for the user, create it
            p_portlet_record.caching_level := CACHE_LEVEL_USER;
            p_portlet_record.caching_key := l_cache_key;
        else
            -- Define a sytem cache. This can happen only once!
            -- the first time the portlet is rendered.
            p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
            p_portlet_record.caching_key := 'MY_INITIAL_CACHE_KEY';
        end if;
    else
        p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
        p_portlet_record.caching_key := 'MY_INITIAL_CACHE_KEY';
    end if;
```

8. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
9. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.8.4 Implementing Expiry-Based Caching

The caching example, located in `..\pdkplsql\pdk\plsql\cache` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement expiry-based caching. You can browse through this example as follows to see how the expiry-based functions are implemented in a portlet:

1. Open the `expirycache_portlet.pkb` file in an editor.
2. At the very top of the file, notice the aliases for the caching level constants.

```
CREATE OR REPLACE
package body VALIDCACHE_PORTLET
is
    -- Caching Constants
    CACHE_LEVEL_SYSTEM constant varchar2(10) := 'SYSTEM';
    CACHE_LEVEL_USER   constant varchar2(10) := 'USER';
```

3. Find the `show` procedure. Notice first that the `p_portlet_record` is an `in` and `out` parameter for this procedure.

```
procedure show
(
    p_portlet_record   in out     wwpro_api_provider.portlet_runtime_record
)
```

4. In the procedure's security check, the caching fields of `p_portlet_record` are set to null if the security check fails.

```
begin
    if (not is_runnable(
         p_provider_id      => p_portlet_record.provider_id
        ,p_reference_path   => p_portlet_record.reference_path)
    ) then
        -- Set it to null so that cache does not get used even if exists
        p_portlet_record.caching_level := null;
        p_portlet_record.caching_key := null;
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
```

5. After that, the procedure sets the value of the caching period in minutes in a temporary variable:

```
-- Set the Caching Period to one minute
l_cache_period := 1;
```

6. Next, notice how the code checks `your portlet_runtime_record` parameter for the current values of the `caching_period` and sets the `caching_period` accordingly. This same piece of code can compare your `caching_period` values.

```
if p_portlet_record.caching_level = CACHE_LEVEL_SYSTEM then
```

```
            -- Cache does not exists for the user, create it
            p_portlet_record.caching_level := CACHE_LEVEL_USER;
            p_portlet_record.caching_period := l_cache_period;
    elsif p_portlet_record.caching_level = CACHE_LEVEL_USER then
            -- Cache exists for the user, overwrite it
            p_portlet_record.caching_period := l_cache_period;
    elsif p_portlet_record.caching_level is null then
         if p_portlet_record.caching_period  is not null then
            -- Cache does not exists for the user, create it
            p_portlet_record.caching_level := CACHE_LEVEL_USER;
            p_portlet_record.caching_period := l_cache_period;
         else
            -- Define a sytem cache. This can happen only once!
            p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
            p_portlet_record.caching_period := l_cache_period;
         end if;
    else -- p_portlet_record.caching_level value is messed up!
         p_portlet_record.caching_level := CACHE_LEVEL_SYSTEM;
         p_portlet_record.caching_period := l_cache_period;
    end if;
```

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.8.5 Implementing Invalidation-Based Caching

Suppose you have a portlet that displays a map of the world, `map_portlet.pkb` and `map_portlet.pks`. You would go about adding invalidation-based functions to it as follows:

1. In the `show` procedure, you need to add a call to `wwpro_api_provider.use_invalidation`. This call indicates to Oracle Portal that the portlet content should be cached by Oracle Web Cache. Note that we have also specified that the content be cached at the user level and that expiry-based caching be used as well (that is, an expiration interval of one minute has been set).

```
procedure show
...
    if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
        p_portlet_record.caching_invalidation :=
          wwpro_api_provider.use_invalidation;
        p_portlet_record.caching_level := 'USER';
        p_portlet_record.caching_period := 1;
...
```

2. Create a procedure in your `map_portlet.pkb` file that invalidates the cache. For example:

```
procedure map_invalidation
(
p_provider_id in number,
p_portlet_id in number,
p_instance_id in varchar2,
p_page_url in varchar2
)
```

```
is
begin
 wwpro_api_invalidation.invalidate_by_instance
  (p_provider_id => p_provider_id,
   p_portlet_id =>  p_portlet_id,
   p_instance_id => p_instance_id,
   p_user => wwctx_api.get_user);
 owa_util.redirect_url(p_page_url);
end map_invalidation;
```

3. In the show procedure, add a link for refreshing the portlet before the code that draws the map. For example:

```
/* Draw the Refresh Me link */
htp.anchor(
  curl => wwctx_api.get_user||
    '.map_invalidation?p_provider_id='||p_portlet_record.provider_id||
    '&p_portlet_id='||p_portlet_record.portlet_id||
    '&p_instance_id='||p_portlet_record.reference_path||
    '&p_page_url='||utl_url.escape(
                    url => p_portlet_record.page_url,
                    escape_reserved_chars => TRUE),
  ctext => wwui_api_portlet.portlet_text(
    p_string =>'Refresh Me',
    p_level => 1)
);
```

4. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
5. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.9 Implementing Error Handling

Oracle Portal provides the capability for you to trap erroneous input and return meaningful error messages. It manages the internal error stack by tracking the raised exceptions and retaining information about them. Oracle Portal also includes a set of APIs for presenting errors in a standardized way.

Error handling services are available through the wwerr_api_error and wwerr_api_error_ui packages. These error handling services include the following key features:

- **Error stack**. Oracle Portal uses an error stack to keep track of the error messages. When an error occurs, the error message is pushed onto an error stack. Whenever procedures or function calls are nested, the error stack keeps track of all the error messages. You can choose to retrieve only the top error (or most recent error) by using the wwerr_api_error.get_top method. Alternatively, you can get all the error messages on the stack using the wwerr_api_error.get_errors function. The stack can also be checked for the presence of any errors by calling the wwerr_api_error.is_empty function.
- **Error messages**. Error handling services provide a way to define meaningful error messages. To define your own error messages, you need to define their name space. The name space consists of the following:
  - **Name** is the error name.

- **Domain** is the area of the product where the error occurred.
- **Subdomain** is the subsystem where the error occurred.
- **Context** is the name of the function where the error occurred.

The name space uniquely identifies your error message. If it does not do so, a `wwc-0000` error message is generated.

The default domains include the portal (`WWC`), application (`WWV`), and page groups (`WWS`). Each domain is further classified into subdomains, which define the object types. The portal domain includes the portlet, page, and document object types. The application domain includes object types such as forms, menus, reports, and charts. The page group domain includes object types such as pages, items, categories, and perspectives. If you need to define an error that does not fall within these classifications, you can define your own domain with subdomains for your errors.

- **Message parameters**. The other language strings that you create for your errors can take substitution parameters for your messages. The `p1`, `p2`, `p3`... parameters can be used to pass substitution parameters to the error messages. For example, for this string:

```
(domain='yahoo', subdomain='provider', name='generalerror', string='Error: %1')
```

an error can be added as follows:

```
wwerr_api_error.add(p_domain=>'yahoo',  p_sub_domain=>'provider',
  p_name=>'generalerror',  p_context=>'yahoo.show',  p1=> sqlerrm);
```

- **Error display**. The `wwerr_api_error_ui` package provides a means to generate a standard user interface for displaying the errors in Oracle Portal. The error messages can be displayed in the following two ways:
  - **Full screen user interface:** These error messages are displayed in a full screen mode. You may want to display full screen errors when the system encounters fatal or show-stopper errors.
  - **Inline user interface:** These error messages are displayed within the current page itself. You may use inline errors for minor errors or warnings.

Additionally, you can choose the output format of the display (HTML, XML, or ASCII text).

8.9.1 Using Error Handling

In general, you set up error handling as follows:

1. On detecting error conditions, add the error message, with an appropriate domain and sub-domain combination, to the stack using the `wwerr_api_error.add` procedure.
2. When necessary (for example, at the end of a routine), expose the error messages using the `wwerr_api_error_ui` procedures. To display full screen messages, use the procedures `show_html`, `show_xml`, or `show_text` depending on your preferred output type. To display inline messages, use the procedures `show_inline_html`, `show_inline_xml`, or `show_inline_text`, depending on the output type you desire.

Guidelines for Error Handling

While implementing error handling, keep in mind the following:

- While defining your own error messages, use your own error domain for these messages. Never use the `WWC`, `WWV`, or `WWS` domain for your error messages. You will need to write a small loader script to load these into the other language tables.
- Avoid unnecessary error messages. If you do not want to do anything in a function, just return `null` rather than an error. For example, suppose you are coding a `copy_portlet` procedure for your portlet because the provider calls it for all of its other portlets. If you do not wish the `copy_portlet` procedure for this particular portlet to do anything, then simply have it return `null`. If you return errors, it will unnecessarily disrupt the portlet functionality.
- A maximum of ten error messages is kept on the stack. Beyond ten, messages are ignored when a call to `wwerr_api_error.add` is made.
- Use the API as a programmatic way of finding the problem. You can use the non-user-interface format for this purpose. For example, when you programmatically register a provider, the exception block can use `get_text_stack` to get the error messages and print them. This approach helps when debugging calls to public APIs since all of them add errors to the stack for exceptions.
- Remember to seed the other language strings for your error messages. For more information, refer to Section 8.11, "Writing Multilingual Portlets".
- The standard user interface for error messages provides a navigation link back to the previous page. It also includes a Help icon for the specified help URL.

8.9.2 Adding Error Handling

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement error handling. You can browse through the following example to see how the error handling functions are implemented in a portlet:

1. Open the `services_portlet.pkb` file in an editor.
2. The domain and subdomain definitions for your error messages are provided with aliases in the constants part of your portlet definition.

```
DOMAIN           constant varchar2(30) := 'provider';
SUBDOMAIN        constant varchar2(32) := 'services';
PORTLET_PATH     constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
PREFNAME_STRING  constant varchar2(30) := 'services_string';
PREFNAME_TITLE   constant varchar2(30) := 'services_title';
```

3. Find the `show` procedure. This procedure performs a security check and, if an error condition arises, it calls `wwerr_api_error.add` to push the `securityerr` error message onto the stack.

```
procedure show
(
    p_portlet_record        wwpro_api_provider.portlet_runtime_record
)
is
...
begin
    -- Perform a security check
    if (not is_runnable(
         p_provider_id      =>  p_portlet_record.provider_id
        ,p_reference_path   =>  p_portlet_record.reference_path)
    ) then
```

```
        wwerr_api_error.add(
                DOMAIN, SUBDOMAIN,
                'securityerr', 'services_portlet.show');
        raise wwpro_api_provider.PORTLET_SECURITY_EXCEPTION;
    end if;
```

4. The `show` procedure also checks for any other kind of execution mode and generates an appropriate error message for an invalid display mode.

```
if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
...
elsif (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW_EDIT)
...
else
    wwerr_api_error.add(DOMAIN, SUBDOMAIN,
        'invaliddispmode', 'services_portlet.show');
    raise wwpro_api_provider.PORTLET_EXECUTION_EXCEPTION;
end if;
```

5. Lastly, the `show` procedure implements a general error message in the exception handler to catch any errors not trapped by the preceding conditions.

```
exception
    when others then
        wwerr_api_error.add(
            DOMAIN, SUBDOMAIN,
            'generalerr', 'services_portlet.show');
        raise wwpro_api_provider.PORTLET_EXECUTION_EXCEPTION;
end show;
```

6. Error handling is also implemented in the `save_prefs` and `save_default_prefs` procedures. They check whether the error stack is empty and, if it is not, the portlet makes a call to `wwerr_api_error.show_html` to display the error in full screen mode.

```
exception
    when INVALID_TEXT_EXCEPTION then
         l_information := l_user||'%'||l_time
             ||'%INVALID_TEXT_EXCEPTION%'||p_string;
         l_action      := LOG_FAILED;
         wwlog_api.log (p_domain      => DOMAIN,
                        p_subdomain   => SUBDOMAIN,
                        p_name        => l_user,
                        p_action      => l_action,
                        p_information => l_information,
                        p_url         => l_url,
                        p_row_count   => 0,
                        p_elapsed_time=> l_elapsed_time);
         wwerr_api_error.add(DOMAIN, SUBDOMAIN,
             'invalid_text', 'services_portlet.save_prefs');
     if (not wwerr_api_error.is_empty) then
         wwerr_api_error_ui.show_html;
     end if;
end save_prefs;
```

7. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.

8. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.10 Implementing Event Logging

Oracle Portal can log events that occur during transactions with its objects. It stores these logs in the database, which makes them available through standard SQL calls and reporting tools.

You can choose the events you would like to log and organize them categorically based on user-defined domains and subdomains. For the logged events, you can view information about the event, the time the event started and stopped, the host or IP address of the remote user, the browser type, and the language.

Event logging services are available through the `wwlog_api` and `wwlog_api_admin` packages. These services include the following key features:

- Event logs are useful for tracking specific usage of the portal. To track such information, you create a log event. Log events require a name space that consists of the following:
  - **Name**, which is the event name.
  - **Domain**, which is the area of the product where the event occurred.
  - **Subdomain**, which is the subsystem that generated the event.

  The default domains include the portal (`WWC`), application (`WWV`), and page group (`WWS`). Each domain is further classified into subdomains that define the object types. The portal domain includes the portlet, page, and document object types. The application domain includes object types such as forms, menus, reports, and charts. The page group domain includes object types such as folders, items, categories, and perspectives. Events themselves could be of types such as add, delete, personalize, hide, copy, execute, and export. If you need to define an event that does not fall within these classifications, you can define your own domain with subdomains for your events.
- Logs can track information in the following two ways:
  - Interval logging calculates the elapsed time for the action performed (for example, the time taken to render a portlet).
  - Event logging logs the occurrence of a single step event you care about (for example, whenever a user personalizes a portlet).
- Log switching enables you to set a switch interval that defines how long you want to maintain your existing log records. The log information stored in the database uses two different tables. The log records are purged based on the value entered for the **Activity Log Interval** in the **Configuration** tab of **Global Settings** (accessible from the Services portlet in the **Portal** subtab of the **Administer** tab). When the log interval (in days) is reached, the logging switches between the two logging tables in the database (for example, A and B). Logs first go into A. When the log interval is reached the first time, the logs are written to B. When the log interval is reached again, the logs go back to A. A is emptied in preparation to store the new log records. If you set your log interval to 14 (the default setting), the logs will switch every 14 days, thus preserving for you, at any point in time, records dated between 14 and 28 days old.

8.10.1 Using Event Logging

In general, you can set up event logging as follows:

1. Add the event object, with an appropriate domain and subdomain combination, using `wwlog_api_admin.add_log_event`. Adding the event ensures that lists of values and other user interface components invoked when the user is monitoring the events show this new event in their lists.
2. Register the log event record by using `wwlog_api_admin.add_log_registry`. The log registry record represents the events you want to log in the future and provides a means to filter the events that need to be logged.
3. Use `start_log` and `stop_log` to mark the events you want to log in your code. Alternatively, for entering single step event log information, just call the log method to mark that event.

Guidelines for Event Logging

While implementing event logging, keep in mind the following:

- Log only what you really care about to improve performance. You don't want to flood the system with log messages that are irrelevant to you. If events are logged in Show mode, then multiple instances of these portlets mean additional hits to the database.
- Choose your domain, subdomain, and log events carefully. While using the log APIs, do not use the Oracle Portal domains such as `WWC`, `WWV`, or `WWS` for your log messages. Organize your domains and subdomains hierarchically ensuring that they are unique across portlets. If other portlets happen to use the same domains or subdomains, you will see those log messages interspersed with your own.
- Create log events that show up in the pop-up lists of values monitoring the logs. You can simply create log registry records that filter the events that would actually be logged, either by specifying particular events or using the generic filters with wild cards (`%`). Apart from creating log registry records, we recommend that you create log events for events that you want to monitor. This way the lists of values in the user interface show these records for additional functions such as monitoring.
- Provide required privileges to users or user groups who need to monitor the logs. Any logs created by a user can be viewed by that user, the Portal Administrator, and any user with the Edit privilege on the `ANY_LOGS` object type.

8.10.2 Adding Event Logging

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement event logging. You can browse through this example as follows to see how the event logging functions are implemented in a portlet:

1. Open the `services_portlet.pkb` file in an editor.

   The domain and subdomain definitions for your log messages are provided with aliases in the constants part of your portlet definition.

   ```
   DOMAIN          constant varchar2(30) := 'provider';
   SUBDOMAIN       constant varchar2(32) := 'services';
   PORTLET_PATH    constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
   PREFNAME_STRING constant varchar2(30) := 'services_string';
   PREFNAME_TITLE  constant varchar2(30) := 'services_title';
   ```

2. Find the `save_prefs` procedure. This procedure provides personalizable functionality where you can personalize text and the portlet title in Edit mode. `save_prefs` stores these personalizations in the database. While saving the changes, you should also log them. Hence, this procedure provides an ideal example of implementing the logging service. A single step event is logged using `wwlog_api.log`. The first instance of `wwlog_api.log` logs the event of personalizing text. The second instance logs the event of personalizing the portlet title.

```
procedure save_prefs
...
begin
...
    if (l_prefs.string_id is null or to_number(l_prefs.string_id) = 0)
    then
        l_action := LOG_INSERT;
...
    else  -- string exists in at least one language so update it
        l_action := LOG_UPDATE;
...
    end if;
-- Log this transaction
l_information := l_user||'%'||l_time||'%completed%'||p_string;
        wwlog_api.log (p_domain      => DOMAIN,
                       p_subdomain   => SUBDOMAIN,
                       p_name        => l_user,
                       p_action      => l_action,
                       p_information => l_information,
                       p_url         => l_url,
                       p_row_count   => l_row_count,
                       p_elapsed_time=> l_elapsed_time);
...
    if (l_prefs.title_id is null or to_number(l_prefs.title_id) = 0)
    then
       l_action := LOG_INSERT;
...
    else
       l_action := LOG_UPDATE;
...
-- Log this transaction
       l_information := l_user||'%'||l_time||'%completed%'||p_title;
       wwlog_api.log (p_domain      => DOMAIN,
                       p_subdomain   => SUBDOMAIN,
                       p_name        => l_user,
                       p_action      => l_action,
                       p_information => l_information,
                       p_url         => l_url,
                       p_row_count   => l_row_count,
                       p_elapsed_time=> l_elapsed_time);
...
end save_prefs;
```

3. The `save_prefs` procedure also logs an event with `wwlog_api.log` when an exception occurs.

```
exception
    when INVALID_TEXT_EXCEPTION then
         l_information := l_user||'%'||l_time
             ||'%INVALID_TEXT_EXCEPTION%'||p_string;
         l_action      := LOG_FAILED;
```

```
              wwlog_api.log (p_domain      => DOMAIN,
                             p_subdomain   => SUBDOMAIN,
                             p_name        => l_user,
                             p_action      => l_action,
                             p_information => l_information,
                             p_url         => l_url,
                             p_row_count   => 0,
                             p_elapsed_time=> l_elapsed_time);
...
```

4. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
5. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.11 Writing Multilingual Portlets

Oracle Portal has a robust set of APIs for interacting with Oracle Portal multilingual storage facility. This storage facility provides a mechanism for storing and retrieving strings in different languages. These APIs abstract the native multilingual functionality and provide developers with a powerful storage mechanism for developing providers that support different language environments.

Multilingual services are available through the `wwnls_api` package. These services include the following key features:

- The multilingual APIs enable the provider to load several translations for the strings displayed in their portlets. Once the strings have been loaded, the provider can call the APIs to retrieve the strings from the multilingual table as needed.
- Context APIs retrieve the user's language and the appropriate translation for that language. The Context APIs determine the user's language environment from the language setting in the browser. When a requested translation does not exist, the APIs return the base language translation.

For example, assume that the provider's `register` procedure loads US and French translations for the portlet title. When the portlet is rendered, the provider implementation retrieves the portlet title string from the table and displays the following results:

- A request for a French string causes the portlet title to appear in French.
- A request for a US string causes the portlet title to appear in US English.
- A request for a Chinese string causes the portlet title to appear in US English because we did not load a translation for the Chinese language.

8.11.1 Using Multilingual Support

In general, you can set up multilingual support as follows:

1. Load your string definitions into the database using the string equivalents for each language you intend to use. For this purpose, call the `wwnls_api.add_string` or `wwnls_api.set_string` with an appropriate domain, subdomain, error message name, and error text combination.
2. Retrieve the strings you require with `wwnls_api.get_string` for the language that you desire.

8.11.2 Adding Multilingual Support

To add multilingual support, you need to perform the following tasks:

- Section 8.11.2.1, "Loading Language Strings"
- Section 8.11.2.2, "Retrieving Language Strings"

8.11.2.1 Loading Language Strings

Language strings can be loaded by a script that is part of the provider installation. This script calls `add_string` and `set_string` to create equivalent strings for different languages.

Oracle Portal uniquely identifies language strings using a combination of domain, subdomain, and name. The domain and subdomain provide a way to categorize the strings. The domain and subdomain should be unique enough to reasonably preclude conflicts with other users of the APIs.

- A *domain* is a particular area of the product. An example of a domain could be provider or page group.
- A *subdomain* is a subsystem of the domain. For example, the subdomain could be the provider name (for example, HelloProvider) or subpage name (for example, HelloPage).

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement multilingual support. You can browse through this example as follows to see how to load strings for multilingual support:

1. Open the `services_seed.sql` file in an editor.
2. Notice the `add_string` call with the parameters for domain name, subdomain name, string name, language, and the actual string text. It returns the String ID for the language string. For setting equivalent strings in other languages, `set_string` is called with the same parameters.

```
set serveroutput on size 1000000
set define off

declare
    l_string_id  integer;
    l_person_id  integer;
    l_group_id   integer;
begin
...
-- strings for portlet record fields
l_string_id := wwnls_api.add_string(
 'provider','services','ptldefname','us','DatabaseServicesPortlet');
wwnls_api.set_string(
 'provider','services','ptldefname','d','DatenbankServicesPortlet-d');
l_string_id := wwnls_api.add_string(
 'provider','services','ptldeftitle','us','Database Services Portlet');
wwnls_api.set_string(
 'provider','services','ptldeftitle','d','Datenbank Services Portlet - d');
l_string_id := wwnls_api.add_string(
 'provider','services','ptldefdesc','us','This is the database services
portlet implemented in PL/SQL. It displays 6 show modes.');
wwnls_api.set_string(
 'provider','services','ptldefdesc','d','Dies ist das Datenbank Service
Portlet, erstellt in PL/SQL. Es stellt 6 Anzeigemodi dar. - d');
l_string_id := wwnls_api.add_string(
 'provider','services','ptldevtmmsg','us','Web Services Portlet Timed
Out.');
wwnls_api.set_string(
 'provider','services','ptldevtmmsg','d','Zeitüeberschreitung aufgetreten
in Web Services Portlet. -d');
```

8.11.2.2 Retrieving Language Strings

The services example, located in `..\pdkplsql\pdk\plsql\svcex` in PDK-PL/SQL (`pdkplsql.zip`), illustrates how you can implement multilingual support. You can browse through this example as follows to see how to retrieve strings for multilingual support:

1. Open the `services_portlet.pkb` file in an editor.
2. The domain and subdomain definitions for your language strings are provided with aliases in the constants part of your portlet definition.

   ```
   DOMAIN           constant varchar2(30) := 'provider';
   SUBDOMAIN        constant varchar2(32) := 'services';
   PORTLET_PATH     constant varchar2(256):= 'oracle.portal.pdk.servicesportlet';
   PREFNAME_STRING  constant varchar2(30) := 'services_string';
   PREFNAME_TITLE   constant varchar2(30) := 'services_title';
   ```

3. Find the `get_portlet_info` function. Notice the calls to `wwnls_api.get_string` to populate the portlet title, name, and description.

   ```
   function get_portlet_info
   (
        p_provider_id in integer
       ,p_language in varchar2
   )
   return wwpro_api_provider.portlet_record
   is
        l_portlet  wwpro_api_provider.portlet_record;
   begin
       l_portlet.id := services_provider.SERVICES_PORTLET_ID;
       l_portlet.provider_id := p_provider_id;
       l_portlet.language := p_language;
       l_portlet.title :=
           wwnls_api.get_string(
                p_domain       =>  DOMAIN
               ,p_sub_domain   =>  SUBDOMAIN
               ,p_name         =>  'ptldeftitle'
               ,p_language     =>  p_language
               );
       l_portlet.description :=
           wwnls_api.get_string(
                p_domain       =>  DOMAIN
               ,p_sub_domain   =>  SUBDOMAIN
               ,p_name         =>  'ptldefdesc'
               ,p_language     =>  p_language
               );
       l_portlet.name :=
           wwnls_api.get_string(
                p_domain       =>  DOMAIN
               ,p_sub_domain   =>  SUBDOMAIN
               ,p_name         =>  'ptldefname'
               ,p_language     =>  p_language
               );
   ...
   ```

4. Browse the rest of the file to examine other usage examples of `wwnls_api.get_string`, which is used in several other places in `services_portlet.pkb`.
5. Optionally, if you want to see this portlet on a page and it is not already in the Portlet Repository, refer to the instructions in Section 8.3.2, "Implementing the Provider Package" for information on how to add it.
6. Once your portlet appears in the repository, you can add it to a page to test it. To add your portlet to a page, follow the instructions in *Oracle Fusion Middleware User's Guide for Oracle Portal*.

8.12 Enhancing Portlets for Mobile Devices

This section explains how to go about enhancing a portlet with PDK-PL/SQL for a mobile device. Before proceeding with this section, you should familiarize yourself with the guidelines for building mobile-enabled portlets, Section 8.1.3, "Guidelines for Mobile Portlets", and the methods of building portlets with PDK-PL/SQL, Section 8.2, "Building PL/SQL Portlets with the PL/SQL Generator" and Section 8.3, "Building PL/SQL Portlets Manually".

To properly build a portlet for a mobile device, do the following:

1. Set the portlet record attributes to support mobile output. Requests arriving from mobile devices through the mobile gateway need to be answered with OracleAS Wireless XML using the `text/vnd.oracle.mobilexml` content type in the header. A mobile-enabled portlet must specify that it can handle these types of requests and produce the mobile content accordingly.

   Table 8–6 lists the portlet record attributes pertinent to mobile portlets and explains how you should specify them.

Table 8–6 Portlet Record Attributes

Attribute	Description

`accept_content_type`	Specify a comma-delimited list of the content types that the portlet produces. If the portlet can produce both HTML and OracleAS Wireless XML, the string would be:
	`text/html, text/vnd.oracle.mobilexml`
	If the portlet can produce only OracleAS Wireless XML, the string would be:
	`text/vnd.oracle.mobilexml`

`mobile_only`	Indicate whether the portlet is available only to mobile pages. TRUE declares the portlet as mobile only, and it will not appear in the Add Portlets page for a standard page. FALSE declares the portlet as mobile and standard, and it will appear in the Add Portlets page for both standard and mobile pages. If the portlet only produces OracleAS Wireless XML and you set this flag to FALSE, the OracleAS Wireless XML is automatically transformed into HTML for desktop devices (such as a normal PC Web browser). This functionality enables you to develop portlets that output only OracleAS Wireless XML but can be viewed on standard pages for desktop access as well as for mobile access.
`short_title`	Enter a shorter version of the portlet title. This title is used on mobile devices because it is more likely to fit the smaller screen. If you do not set a short title, then the portlet title is used instead.

Table 8–6 (Cont.) Portlet Record Attributes

Attribute	Description
`has_show_link_mode`	Indicate whether you have implemented Link mode for the portlet. We recommend that all mobile portlets enable Link mode. The rationale for using Link mode is more fully explained when you reach step 3.

2. If the portlet can produce both HTML and OracleAS Wireless XML, then, during execution, it must determine whether the request is for a mobile or a desktop device. If it is a desktop request, then the portlet must product HTML output. If it is a mobile request, then it must produce OracleAS Wireless XML output. You can determine the request type with the `wwctx_api.get_http_accept()`. It fetches the HTTP accept header, which indicates the request type. In the portlet response, you must set the MIME header to the appropriate value in the HTTP header before the portlet content is produced. If not, then the portlet response is rejected when the resulting page is built by Oracle Portal. If the call to get `wwctx_api.get_http_accept()` returns a string starting with `text/vnd.oracle.mobilexml`, then you can assume it is a mobile request. Otherwise, it is a desktop request. In the case of a mobile request, you should set the MIME header to `text/vnd.oracle.mobilexml`. In the case of a desktop request, you can explicitly set the MIME header to `text/html`, but it is not required that you do so because it's the default setting.

   If you want to produce HTML for desktop requests and OracleAS Wireless XML for mobile requests, the `show` procedure for your portlet should look similar to the following:

```
procedure show
    (
        p_portlet_record in out wwpro_api_provider.portlet_runtime_record
    )
    is
    begin
      --
      -- Does the portal want us to render the portlet contents?
      --
      if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
          --
          -- Is this a mobile request?
          --
          if owa_util.get_cgi_env('HTTP_ACCEPT') like
            wwpro_login.CONTENT_TYPE_MOBILEXML || '%' then
              --
              -- This is a mobile request for the portlet contents
              -- call the mobile show contents procedure
              --
              render_mobile_show_contents(p_portlet_record);
          else
              --
              -- This is a desktop request for the portlet contents,
              -- call the desktop show contents procedure
              --
              render_desktop_show_contents(p_portlet_record);
          end if;
      elsif -- check for other show modes, and handle them
      ...
end show;
```

If you want to produce only OracleAS Wireless XML for all requests, the `show` procedure for your portlet should look similar to the following:

```
procedure show
    (
        p_portlet_record in out wwpro_api_provider.portlet_runtime_record
    )
    is
    begin
      --
      -- Does the portal want us to render the portlet contents?
      --
      if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
          --
          -- This is either a desktop or mobile request for the portlet
          -- contents call the show contents procedure to render the
          -- OracleAS Wireless XML output
          --
          render_show_contents(p_portlet_record);
      elsif -- check for other modes
      ...
end show;
```

In order to generate OracleAS Wireless XML, the `render_show_contents` or `render_show_mobile_contents` procedure would need to contain something similar to the following:

```
...
    (
        p_portlet_record in out wwpro_api_provider.portlet_runtime_record
    )
    is
    begin
      --
      -- Set the MIME type to be Oracle9iAS Wireless XML
      --
      owa_util.mime_header('text/vnd.oracle.mobilexml');
      --
      -- Output the Oracle9iAS Wireless XML markup
      --
      htp.p('<SimpleText><SimpleTextItem>');
      htp.p('Hello World!');
      htp.p('</SimpleTextItem></SimpleText>');
...
```

Note: As this is a mobile request the MIME header is set to `text/vnd.oracle.mobilexml`. In the OracleAS Wireless XML markup, you only need to render the actual content. Oracle Portal fits it into the complete OracleAS Wireless XML document. You do not need `<SimpleResult>` or `<SimpleContainer>` tags. These tags are rendered by Oracle Portal along with a back link, which, by default, is assigned to one of the buttons on the mobile device.

3. If you want your portlet to allow personalization of titles for mobile rendering, you need to implement Link mode. Link mode is only called for mobile requests and it renders a link to the portlet content. This link appears in the menu of page contents when the user first navigates to the page. If a Link mode is not present for

a portlet, then Oracle Portal renders a default link mode using the short title from the portlet record. You can also use Link mode to render more than just a title. For example, in a stock portlet, you could render the stock price of a user's favorite stock. Thus, they could see the current stock price without drilling down further.

The link for Link mode rendition is provided by Oracle Portal and passed to the portlet through the `page_url` parameter on the portlet record. The portlet can extend or completely replace this behavior. If this link is rewritten as a URL that takes the user away from Oracle Portal (it does not point to the Oracle Portal middle tier server), then the portlet should set the `show_behaviour_style` field to `wwpro_api_provider.EXTERNAL_PORTLET` on the portlet record.

The following example illustrates how to code your `show` procedure to include Link mode:

```
procedure show
    (
        p_portlet_record in out wwpro_api_provider.portlet_runtime_record
    )
    is
    begin
      --
      -- Does the portal want us to render the portlet contents?
      --
      if (p_portlet_record.exec_mode = wwpro_api_provider.MODE_SHOW) then
          --
          -- Is this a mobile request?
          --
          if owa_util.get_cgi_env('HTTP_ACCEPT') like
            wwpro_login.CONTENT_TYPE_MOBILEXML || '%' then
              --
              -- This is a mobile request for the portlet contents,
              -- call the mobile show contents procedure
              --
              render_mobile_show_contents(p_portlet_record);
          else
              --
              -- This is a desktop request for the portlet contents,
              -- call the desktop show contents procedure
              --
              render_desktop_show_contents(p_portlet_record);
          end if;
      --
      -- Does the portal want us to render the LINK mode?
      --
      elsif (p_portlet_record.exec_mode = wwpro_api_provider.MODE_LINK) then
          --
          -- This is a mobile request for the portlet link mode,
          -- call the mobile link procedure
          --
          render_mobile_link(p_portlet_record);
      elsif -- check for other show modes, and handle them
      ...
end show;
```

The following example illustrates what you must implement in the `render_mobile_link` procedure for Link mode. In particular, notice the setting of the MIME type, the usage of `get_custom_short_title`, and the OracleAS Wireless XML output.

```
procedure render_mobile_link
```

```
    (
        p_portlet_record in out wwpro_api_provider.portlet_runtime_record
    )
    is
      l_short_title varchar2(80);
    begin
      --
      -- Set the MIME type to be Oracle9iAS Wireless XML
      --
      owa_util.mime_header('text/vnd.oracle.mobilexml');
      --
      -- Get the personalized short title for this portlet instance
      -- using the language and reference path held in the portlet
      -- record
      --
      l_short_title := get_custom_short_title(p_portlet_record);
      --
      -- Output the OracleAS Wireless XML markup
      --
      if l_short_title is not null then
            htp.p('<SimpleHref target="'
                || htf.escape_sc(p_portlet_record.page_url)
                || '">'
                || l_short_title
                || '</SimpleHref>');
        else
            htp.p('<SimpleHref target="'
                || htf.escape_sc(p_portlet_record.page_url)
                || '">'
                || get_custom_title(p_portlet_record)
                || '</SimpleHref>');
        end if;
end render_mobile_link;
```

Accessing the DeviceClass Header

To further facilitate the creation of mobile-enabled portlets, you can access information in the DeviceClass header, which is sent to all portlets through owa_util.get_cgi_env('x-oracle-device.class'). You can use the values of this header to determine the complexity of the OracleAS Wireless XML rendition of the portlet. For example, the PDA rendition could contain considerably richer content than the microbrowser rendition. The voice rendition could contain extra tags for voice commands and links to sound files, which play instead of the text-to-speech system.

Table 8–7 describes the values available in the header.

Table 8–7 DeviceClass Header Values

Value	Description
voice	Indicates a voice-only device, such as a normal telephone calling a voice access number.
microbrowser	Indicates a small size display device, which supports a markup browser, such as a WAP phone.
pdabrowser	Indicates a medium size display device, such as a Palm or PocketPC.
pcbrowser	Indicates a large size display device used with desktop browsers

8.13 Registering Providers Programmatically

In most cases, you use the Oracle Portal user interface to register providers as described in Section 8.2.3.2, "Registering the Database Provider". In some instances, though, you may wish to register a provider programmatically rather than through the user interface. This section describes as follows how to use `wwpro_api_provider_registry.register_provider` to register your providers:

- Section 8.13.1, "Registration Prerequisites"
- Section 8.13.2, "Provider Record Input"
- Section 8.13.3, "Registration Example"

8.13.1 Registration Prerequisites

In order to register a provider programmatically with `wwpro_api_provider_registry.register_provider`, the following requirements must be met:

- You must first install the provider as described in Section 8.2.3.1, "Installing the Packages in the Database".
- You must have the necessary privileges to execute `wwpro_api_provider_registry.register_provider`. As it is a secure API, a security check determines whether the user calling it has the necessary privileges. At a minimum, you must have `wwsec_api.ANYPROVIDER_PUBLISH`. If you have the privileges to execute the API, you also have sufficient privileges to set the Oracle Portal session using `wwctx_api.set_context`.

Note: If the database user of the SQL*Plus session is the Oracle Portal schema owner, then, by default, the user is the Oracle Portal schema owner (unless `wwctx_api.set_context` is called with a different user). The schema owner already has the necessary privileges to execute `wwpro_api_provider_registry.register_provider`.

8.13.2 Provider Record Input

The `wwpro_api_provider_registry.register_provider` API requires the `provider_record` as input. When you pass the `provider_record` to `register_provider`, all of the fields are not required. Table 8–8 indicates which fields are required for Web and database providers. If a field is not applicable for a particular type of provider, it is shown as NA.

Table 8–8 Required `provider_record` Fields

Field	Required for database providers	Required for Web providers	Comments
`name`	Yes	Yes	None
`implementation_style`	Yes	Yes	This field is always the following for PL/SQL portlets: `wwpro_api_provider_registry.DATABASE_IMPL;`
`implementation_owner`	NA	Yes	This field is the schema where the provider/portlet is located.

Table 8–8 (Cont.) Required `provider_record` Fields

Field	Required for database providers	Required for Web providers	Comments
`implementation_name`	NA	Yes	When registering a database provider the implementation package of the provider must be valid for the registration to be successful.
`login_frequency`	Yes	Yes	This field specifies how often to grab the session information.
`http_app_type`	Yes	NA	For external application Web providers: `wwpro_api_provider_registry.HTTP_APP_TYPE_EXTERNAL` For regular Web providers: `wwpro_api_provider_registry.HTTP_APP_TYPE_PORTAL`
`http_url`	Yes	NA	None
`require_url`	Yes	NA	None
`encryption_key`	If provider_key is not null,yes. If provider_key is null, no.	If provider_key is not null,yes. If provider_key is null, no.	None

8.13.3 Registration Example

The following sample SQL script illustrates the usage of `wwpro_api_provider_registry.register_provider`.

```
set serveroutput on size 1000000
set define on
declare
    l_prov_rec     wwpro_api_provider_registry.provider_record;
    l_prov_id      integer;
begin
    l_prov_rec.name                 := 'CacheProvider';
    l_prov_rec.display_name         := 'Cache Provider';
    l_prov_rec.timeout              := 10;
    l_prov_rec.timeout_msg          := 'The provider timed out';
    l_prov_rec.implementation_style := wwpro_api_provider_registry.DATABASE_IMPL;
    l_prov_rec.implementation_owner := '&&2';
    l_prov_rec.implementation_name  := 'cache_provider';
    l_prov_rec.language             := wwctx_api.get_nls_language;
    l_prov_rec.enable_distribution  := true;
    l_prov_rec.login_frequency      := wwpro_api_provider_registry.LOGIN_
                                         FREQUENCY_NEVER;
    l_prov_rec.created_on           := sysdate;
    l_prov_rec.created_by           := '&&1';
    l_prov_rec.last_updated_on      := sysdate;
    l_prov_rec.last_updated_by      := '&&1';
    l_prov_id := wwpro_api_provider_registry.register_provider(l_prov_rec);
    commit;
    dbms_output.put_line('Cache Provider successfully registered');
```

```
exception
    when others then
        dbms_output.put_line('ERROR: Could not register Cache Provider');
        dbms_output.put_line('SQLERRM: ' || SQLERRM);
        rollback;
end;
/
```

Note: After you have registered your provider, you may also refresh the Portlet Repository programmatically using `wwpro_api_provider_registry.refresh_portlet_repository` or through the Oracle Portal user interface.

Part III

Content Management APIs

Part III contains the following chapters:

9 Content Management API Introduction

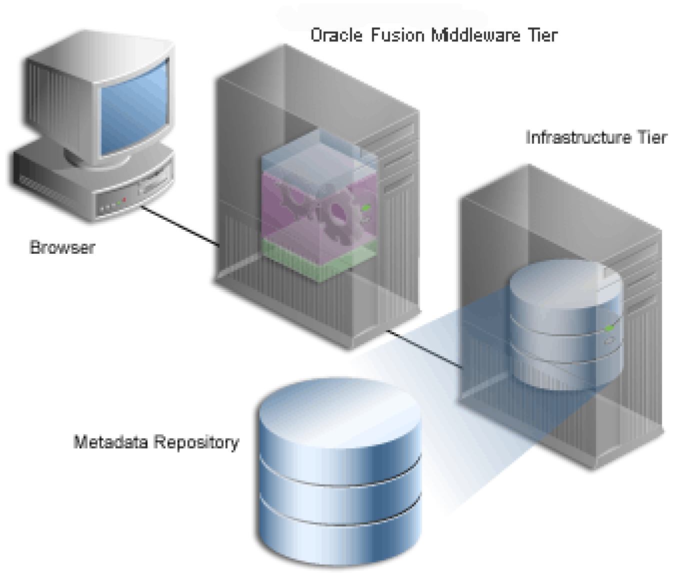
This chapter provides an overview of a selection of the APIs provided with Oracle Portal. You can use these particular APIs to write code for performing content management tasks. It contains the following sections:

- Section 9.1, "Overview"
- Section 9.2, "Content Management APIs"
- Section 9.3, "Providing Access to the APIs and Secure Views"
- Section 9.4, "Guidelines for Using the APIs"
- Section 9.5, "Guidelines for Using the Secure Views"
- Section 9.6, "Code Samples"

9.1 Overview

Oracle Portal uses a schema within the Oracle Metadata Repository, shown in Figure 9–1, to store the content and metadata associated with the portal instance. This schema is sometimes referred to as the content repository. For example, when a contributor adds a file item to a portal page, the file is uploaded to a table in the portal schema of the MDS Repository along with the metadata supplied.

Figure 9–1 The MDS Repository



For more information about the MDS Repository and Oracle Portal architecture in general, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Oracle Portal provides many built-in tools to help you manage the content in the portal schema of the MDS Repository right from your Web browser. However, sometimes you may find that you need to work with your content in an environment outside of the Oracle Portal browser-based user interface.

The following are some examples of when you might want to do this:

- Building an alternative user interface when the product user interface does not quite meet your requirements, for instance:
  - to change the look and feel of the wizards to meet your own corporate design
  - to add custom validations
  - to provide a custom search form or search results page
- Bulk loading content with metadata
- Integrating with external applications and content management systems
- Archiving items (for example, moving them to an archive page or offline storage)
- Managing versions (for example, deleting or purging noncurrent versions, limiting the number of versions, and so on)
- Integrating with external workflow systems

The content management APIs enable you to interact with the portal schema of the MDS Repository programmatically, rather than by using the Oracle Portal user interface. For more information, refer to Section 9.2, "Content Management APIs".

You can also query the data in the content repository using a set of secure views. For more information, refer to Section 9.2.1, "Secure Content Repository Views".

9.2 Content Management APIs

There are multiple public APIs that enable you to programmatically perform many content management tasks, such as adding items and creating pages.

The majority of the APIs for content management are contained within the WWSBR_API package. You may also find the following API packages useful when writing code to perform content management tasks:

- The WWSRC_API package contains APIs for performing searches on content in the portal schema of the MDS Repository.
- The WWSEC_API package contains APIs for controlling access to content in the portal schema of the MDS Repository.
- The WWCTX_API package contains APIs for managing a session context for a specific user.
- The WWPRO_API_INVALIDATION package contains APIs for invalidating content in Oracle Web Cache.

Note: Only use public APIs in your code. The use of non-public APIs is not supported and may cause your code to break when upgrading to new releases.

More On OTN

For more information about these packages, refer to Section F.1, "Supported APIs". For a full list of the supported PL/SQL APIs, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

```
http://portalcenter.oracle.com
```

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

9.2.1 Secure Content Repository Views

Held within the portal schema are a number of content repository views. You can use these views to query back data on the documents and items stored in the content repository. For a list of the secure views, refer to Section F.2, "Secure Views".

Many of the content management APIs that are discussed in this manual require you to pass object IDs as parameters. To do this, you need to know the IDs of the objects with which you want to work. You can use the secure content repository views to find the IDs of portal objects. For more information, refer to Section 10.3, "Finding an Object ID".

Note: Direct access to other tables and views in the portal schema of the MDS Repository is not supported, as the definition of those tables and views may change between releases.

9.2.2 Terminology

To maintain backward compatibility with the Oracle9*i*AS Portal Release 1 (3.0.9) content area APIs and views, many of the API procedure and parameter names and the view and column names continue to use Release 1 terminology. You may find Table 9–1 useful to map the Release 1 terminology to the current terminology.

Table 9–1 Mapping of Release 1 Terminology to Current Terminology

Release 1 Terminology	Current Terminology
content area	page group
folder	page
navigation bar	navigation page

9.3 Providing Access to the APIs and Secure Views

The portal schema automatically has the appropriate access to the public APIs and secure views. To enable another schema to access the public APIs and secure views, use the following script:

```
ORACLE_HOME/portal/admin/plsql/wwc/provsyns.sql
```

To provide access to the APIs, perform the following steps:

1. Change to the directory containing the `provsyns.sql` script:

   ```
   Windows: cd <ORACLE_HOME>\portal\admin\plsql\wwc
   Linux/Unix: cd <ORACLE_HOME>/portal/admin/plsql/wwc
   ```

2. Log in to SQL*Plus as the portal schema owner. For example:

   ```
   sqlplus portal/oracle1
   ```

You must run `provsyns.sql` as the portal schema owner. By default the portal schema is called PORTAL. An administrator can use Oracle Internet Directory to obtain the portal schema password as follows:

a. Navigate to:
   - Entry Management
   - cn=OracleContext
   - cn=Products
   - cn=IAS
   - cn=Infrastructure Databases
   - OrclReferenceName=Infrastructure Database (for example, iasdb.server.domain.com)
   - OrclResourceName=Schema Name (for example, PORTAL)

b. Click this entry.

c. Look for the orclpasswordattribute value in the right panel. This is the schema password.

3. Run the `provsyns.sql` script:

```
SQL>@provsyns.sql <schema>
```

Where `schema` is the name of the schema to which you want to grant access.

Tip: Ignore any messages related to a missing file when running `provsyns.sql`.

9.4 Guidelines for Using the APIs

When using the APIs described in this manual, you should follow the best practice guidelines described in the following sections.

9.4.1 Using a Separate Schema

When creating procedures and packages that use the content management APIs, create them in a separate schema of the database in which Oracle Portal is installed. Do not create them in the portal schema. Creating additional procedures and packages in the portal schema is not supported and they may be lost when upgrading to a new release.

Once you have created the schema for your procedures and packages, you must grant it access to the APIs and secure content repository views. For information about how to do this, refer to Section 9.3, "Providing Access to the APIs and Secure Views".

9.4.2 Using Constants

All the API packages include predefined constants that you can use to easily refer to Oracle Portal objects and metadata. For example, the WWSBR_API package includes constants for base attributes (such as ATTRIBUTE_AUTHOR and ATTRIBUTE_TEXT), seeded item types (such as ITEM_TYPE_FILE and ITEM_TYPE_URL), and image alignment options (such as ALIGN_BOTTOM and ALIGN RIGHT). To make your code robust, you should use these constants wherever possible for object IDs and attribute values. That way, if the actual ID or values change, your code will still work because the name of the constant will stay the same.

For a full list of the constants available in each API package, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

http://portalcenter.oracle.com

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

9.4.3 Invalidating the Cache

Many of the APIs automatically generate cache invalidation messages for the pages that are affected by the changes. Therefore, you should always call `wwpro_api_invalidation.execute_cache_invalidation` at the end of your routine to process these messages (see Example 11–3 for an example of how to do this). If you do not call this procedure, your changes may not be visible until the affected pages are invalidated through other means.

There is no need to call `execute_cache_invalidation` more than once. For example, if you are adding or updating multiple items in a loop, call `execute_cache_invalidation` when the loop is complete.

9.4.4 Issuing Commits

The APIs do not issue commits. However, if you are calling the APIs from a browser session through Portal Services, Portal Services will perform an automatic commit before returning control to the browser.

The `wwpro_api_invalidation.execute_cache_invalidation` API also issues a commit. If you are calling the APIs from an external environment (for example, SQL*Plus or a Web provider), and you need to commit before processing the cache invalidations, you must explicitly issue a commit statement in your code. Similarly, if you need to rollback, do so before calling `execute_cache_invalidation.`

9.4.5 Resetting CMEF Global Variables

If you are using the content management APIs in conjunction with the Content Management Event Framework (CMEF) you need to make sure that you reset the CMEF global variables before or after each API call.

When a user performs an action using the user interface, Oracle Portal uses various global variables to determine which CMEF messages get logged. After each action, these global variables are automatically reset ready for the next action. When you use the APIs to perform portal actions, these global variables are not automatically reset. Therefore, to ensure that CMEF messages are logged correctly, you must reset the CMEF variables explicitly by calling the `wwsbr_api.clear_cmef_context` API before or after each API call.

For example, if you use the APIs to create several items on a page, you need to call the `clear_cmef_context` API between each of the `add_item` calls (see Example 9–1).

Example 9–1 Calling the clear_cmef_context API

```
...
l_new_item_master_id1 := wwsbr_api.add_item(
  p_caid              => l_caid,
  p_folder_id         => l_folder_id,
  ...
  );
wwsbr_api.clear_cmef_context;
```

```
l_new_item_master_id2 := wwsbr_api.add_item(
  p_caid              => l_caid,
  p_folder_id         => l_folder_id,
  ...
  );
wwsbr_api.clear_cmef_context;
...
```

For more information about CMEF, refer to Chapter 16, "Using the Content Management Event Framework".

9.4.6 Using Predefined Exceptions

The API packages contain many predefined exceptions. When coding with the content management APIs, it is good practice to include the appropriate predefined exceptions rather than relying on the WHEN OTHERS exception to pick up all errors. This also has the advantage that any error messages generated by the code can be more specific to the actual problem. Example 9–2 shows some of the exceptions you might include when calling the `wwsbr_api.set_attribute` API.

Example 9–2 Using Predefined Exceptions

```
begin
 wwsbr_api.set_attribute(
   p_site_id           => 37,
   p_thing_id          => 8056,
   p_attribute_site_id => wwsbr_api.SHARED_OBJECTS,
   p_attribute_id      => wwsbr_api.ATTRIBUTE_TITLE,
   p_attribute_value   => 'New Display Name'
 );
 -- Process cache invalidation messages.
 wwpro_api_invalidation.execute_cache_invalidation;
exception
 when wwsbr_api.ITEM_NOT_FOUND_ERROR then
   dbms_output.put_line('Item does not exist');
 when wwsbr_api.ATTRIBUTE_NOT_FOUND then
   dbms_output.put_line('Attribute does not exist');
 when wwsbr_api.ITEM_NOT_FOR_UPDATE then
   dbms_output.put_line('Cannot update an item with a status of Rejected, Deleted
or Marked for Delete');
 when wwsbr_api.NOT_AUTHORIZED_USER then
   dbms_output.put_line('User trying to update the item is not the current user or
the user who checked the item out');
 when wwsbr_api.EDIT_CUSTOM_ATTR then
   dbms_output.put_line('Error while trying to update a custom attribute');
 when OTHERS then
   dbms_output.put_line('Error '||to_char(sqlcode)||': '||sqlerrm);
end;
/
```

Tip: When using predefined exceptions, remember to include the name of the package that owns it, for example wwsbr_api.PAGE_NOT_FOUND.

More On OTN

For a full list of the exceptions available in each API package, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

9.4.7 Naming Objects

When creating page groups and page group objects, you specify a unique internal name for the object (`p_name`). Internal names must:

- be no more than 60 characters in length.
- not contain spaces or special characters other than the underscore character (_).

9.5 Guidelines for Using the Secure Views

When using the secure views described in this manual, you should follow the best practice guidelines described in the following sections.

Note: In this release, there is no supported view of the document table in the repository. A secure view of the document table is planned for a future release.

9.5.1 Identifying Primary Keys

The primary key for most objects contains the object id, the page group id (caid), and the language. When joining between views, always use these columns in the JOIN clause. Example 9–3 shows the JOIN when joining an item to its page.

Example 9–3   Joining an Item to Its Page

```
select ...
from wwsbr_all_items i,
     wwsbr_all_folders p
where i.folder_id = p.id
  and i.caid = p.caid
  and i.language = p.language
```

9.5.2 Querying Translatable Objects

If an object is translatable (that is, it resides in a page group for which translations are enabled and it includes languages in its key), you must observe the following rules:

- If the object (or its current version) is translated, a row will exist for the translation. To select the row for the current session language, compare the value of the language column to the function `wwctx_api.get_nls_language()`.
- If the object is not translated, select the row for the page group's default language.

Example 9–4 selects the translated page display name for all pages in a given page group.

Example 9–4   Selecting a Translated Page Display Name

```
select p.display_name title
from wwsbr_all_folders p
where p.caid = 53
and (p.language = wwctx_api.get_nls_language -- The current language.
  or (exists -- A row for the page in the page group default language.
      (select pg.id
       from wwsbr_all_content_areas pg
       where pg.id = p.caid
         and pg.default_language = p.language
      )
      and not exists -- A row for the page in the current language.
       (select p2.id
        from wwsbr_all_folders p2
        where p2.id = p.id
          and p2.language = wwctx_api.get_nls_language
       )
     )
    )
/
```

Tip: When writing PL/SQL routines that refer to the current session language, call `wwctx_api.get_nls_language` once and store it in a variable. Referring to the variable in any following SQL statements, rather than calling the API multiple times, results in better performing code.

9.5.3 Selecting Data for the Current User

To select data for the current user, use the function `wwctx_api.get_user`. Example 9–5 selects items created by the current user.

Example 9–5 Selecting Items Created by the Current User

```
declare
  l_user varchar2(60);
  ...
begin
  ...
  l_user := wwctx_api.get_user;
  select ...
  from wwsbr_all_items
  where creator = l_user
  ...
end;
/
```

9.6 Code Samples

The code samples provided in the next few chapters are intended to provide some examples of how you can use the content management and associated APIs. Not all of the APIs are used in these samples, and the samples provided do not utilize all the parameters, constants, exceptions, and so on available to the APIs.

For a full list of the public APIs available with Oracle Portal and their parameters, constants, exceptions and so on, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

The examples are written to be run from SQL*Plus. If running from other environments, for example Portal Services, you may need to make changes to the code given the context in which it is executing.

The examples, with the exception of Chapter 14, "Creating Multi-Lingual Content", which deals with translations, assume that the language context is set to the default language of the page group being manipulated. This simplifies the WHERE clause on any SQL statements. Objects always have a translation in the default language therefore the queries will return a single row as appropriate.

If this assumption on the language code does not hold true for your code, refer back to Section 9.5.2, "Querying Translatable Objects" for an example of how to extend the WHERE clause to deal with multiple translations or the non-existence of a translation in the given language context.

10 Getting Started with Content Management APIs

This chapter introduces some tasks and concepts that you may find useful before you start using the content management APIs provided with Oracle Portal. It contains the following sections:

- Section 10.1, "Setting the Session Context"
- Section 10.2, "API Parameters"
- Section 10.3, "Finding an Object ID"

10.1 Setting the Session Context

All of the content management APIs and the secure views assume that the context has been set for the functions to execute as a specific portal user. If you call an API from within a browser session (for example, a database provider or PL/SQL item), the context is set automatically to the user who is currently logged in to the portal. If you call an API from a Web provider or an external application (for example, SQL*Plus or Portal Services), you must set the context to a specific user by using the `wwctx_api.set_context` API.

This API allows an application to assert an identity, by providing a user name and password that the portal can verify with the Oracle Application Server Single Sign-On Server before establishing a session for the asserted user. If the assertion fails, due to an invalid password, for example, an exception is raised.

Example 10–1 sets the context to user `JOE.BLOGS` with the password `welcome1`.

Example 10–1 Setting the Session Context (set_context API)

```
declare
  p_user_name varchar2(60) := 'JOE.BLOGGS';
  p_password  varchar2(60) := 'welcome1';
  p_company   varchar2(60) := null;
begin
  wwctx_api.set_context(p_user_name,p_password,p_company);
end;
/
```

10.2 API Parameters

- The following parameters refer to the page group ID:

  - `p_site_id`
  - `p_siteid`
  - `p_caid`

  For backward compatibility purposes these parameter names have not been changed, and will not be changed, to use the most recent terminology. For more information, refer to Section 9.2.2, "Terminology".

- The following parameters refer to the page ID:
  - `p_folder_id`
  - `p_id`

  For backward compatibility purposes these parameter names have not been changed, and will not be changed, to use the most recent terminology. For more information, refer to Section 9.2.2, "Terminology".

- An item has two IDs. All versions of the item have the same master item ID (GUID). However, when you create a new version of an item it is assigned its own ID that uniquely identifies that particular version of the item. Some APIs simply need the master item ID to identify an item (`p_master_item_id`). However, some APIs (for example, `modify_item`) also need to know which particular version of the item you want to work with. In this case you need to pass the unique item ID (`p_item_id`). In some APIs, you will find that the item ID is referred to as `p_thing_id`.
- The primary key for most objects contains the ID of the actual object and the ID of the page group to which the object belongs. When referencing an existing object in a parameter, you must reference both the object ID and its page group ID. The referenced object's page group must either be the same as the page group of the referencing object or the Shared Objects page group. For example, when adding an item, you need to specify the item type (for example, file, text, image, and so on). To do this you must pass the item type ID (`p_type_id`) and the ID of the page group which contains the item type (`p_type_caid`).

10.3 Finding an Object ID

Many of the content management APIs require you to pass object IDs to their parameters. To do this, you need to know the IDs of the objects with which you want to work. You can obtain these IDs by using the secured content repository views.

10.3.1 Finding a Page Group ID

Many of the content management APIs include the ID of a page group as one of the parameters. For example, when creating a page using the `add_folder` API, you need to specify the ID of the page group in which you want to create the page using the `p_caid` parameter. Example 10–2 shows a query that you could use to find out this ID.

Example 10–2 Finding the ID of a page group

```
select id "Page Group ID"
from wwsbr_all_content_areas
where name = '&pagegroup'
  and language = '&language'
/
```

Example 10–3 shows how you might use this query in a function.

Example 10–3 Function to Find the ID of a Page Group

```
create or replace function get_page_group_id (p_name in varchar2) return number is
  l_page_group number;
  l_language   varchar2(30);
begin
  l_language := wwctx_api.get_nls_language;
  select id
  into l_page_group
  from wwsbr_all_content_areas
  where name = p_name
    and language = l_language;
  return l_page_group;
exception
  ...
end;
/
```

10.3.2 Finding a Page ID

When working with pages and items using the content management APIs, you need to know the ID of the page in which you want to work. For example, if you want to delete a page using the `delete_page` API, you need to identify the page that you want to delete by its ID. Example 10–4 shows how to find out the ID of a page if you know the ID of the page group to which it belongs.

Example 10–4 Finding the ID of Page in a Known Page Group

```
select id "Page ID"
from wwsbr_all_folders
where name = '&page'
  and caid = 53
  and language = '&language'
/
```

Sometimes you may not know the ID of the page group to which a particular page belongs. Example 10–5 shows how to find out the ID of a page and its owning page group.

Example 10–5 Finding the IDs for a Page and Its Page Group Given Page and Page Group Names

```
column 'Page Group Name' format a25;
column 'Page Name' format a25;
select f.caid "Page Group ID",
       c.name "Page Group Name",
       f.id   "Page ID",
       f.name "Page Name"
from wwsbr_all_content_areas c,
     wwsbr_all_folders f
where f.name = '&page'
  and c.id = f.caid
  and c.name = '&pagegroup'
  and c.language = '&language'
  and c.language = f.language
/
```

10.3.3 Finding Region IDs

When adding an item to a page using the `add_item` API, you need to know the ID of the region in which you want to place the item. You can add items only to item regions that have been set up to allow content to be added to them. Example 10–6 shows a query that finds out the IDs and types of the insertable regions on a particular page. Given this information, you can choose the region ID of an appropriate region on the page in which to place your new item.

Example 10–6 Finding the IDs and Types of Insertable Regions on a Given Page

```
select distinct r.id "Region ID",
                r.type "Type"
from wwsbr_all_content_areas c,
     wwsbr_all_folders f,
     wwsbr_all_folder_regions r
where f.name = '&page'
  and c.name = '&pagegroup'
  and c.id = f.caid
  and f.id = r.folder.id
  and f.caid = r.folder.caid
  and r.allow_content = 1
  and c.language = '&language'
  and c.language = f.language
  and (f.language = r.language
    or r.language is null)
/
```

10.3.4 Finding an Item ID

Example 10–7 illustrates how to find the ID of an item by querying the WWSBR_ALL_ITEMS view.

Example 10–7 Finding the ID of the Current Version of an Item Given its Master Item ID

```
select id
from wwsbr_all_items
where masterid = 513
  and caid = 53
  and active = 1
  and is_current_version = 1
  and language = '&language'
/
```

To avoid duplicate rows when querying for currently published items, always include the LANGUAGE, ACTIVE (`active=1`), and IS_CURRENT_VERSION (`is_current_version=1`) columns. Example 10–8 shows how to select all items on a given page (`folder_id=1`) and a given page group (`caid=75`).

Example 10–8 Finding Item IDs

```
select i.display_name title, i.id latestversion
from wwsbr_all_items i
where i.folder_id = 1
  and i.caid = 75
  and i.active = 1
  and i.is_current_version = 1
  and (i.language = '&language' -- The current session language.
      or (    exists -- A row for the item in the page group default language.
              (select pg.id
               from wwsbr_all_content_areas pg
               where pg.id = i.caid
                 and pg.default_language = i.language
              )
           and not exists -- A row for the item in the current language.
              (select i2.id
               from wwsbr_all_items i2
               where i2.id = i.id
                 and i2.language = '&language'
                 and i2.active = i.active
                 and i2.is_current_version = i.is_current_version
              )
         )
      )
order by i.id
/
```

11 Performing Simple Content Management Tasks

This chapter describes how to use the APIs provided with Oracle Portal to perform simple content management tasks. It contains the following sections:

- Section 11.1, "Editing Page Properties"
- Section 11.2, "Editing Content"
- Section 11.3, "Reorganizing Content"
- Section 11.4, "Copying Content"
- Section 11.5, "Deleting Content"

More On OTN

For more information about any of the APIs mentioned in this chapter, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

> **Tip:** Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

11.1 Editing Page Properties

If you want to edit page properties, you need to perform the following steps:

1. Query the WWSBR_USER_PAGES view to populate a page record.
2. Modify the properties you want to update.
3. Pass the updated page record to the `modify_folder` API.

You can update the properties shown inTable 11–1 in the page record.

Table 11–1 Editable Page Record Properties

Property	Data Type	Description
NAME	VARCHAR2(60)	Name of the page. This name is used in path based URLs.
TITLE	VARCHAR(256)	Display name, or title, of the page.

Table 11–1 (Cont.) Editable Page Record Properties

Property	Data Type	Description
SETTINGSSETID	NUMBER	ID of the style used by the page.
SETTINGSSETSITEID	NUMBER	Page group ID of the style used by the page.
ISPUBLIC	NUMBER(1)	Indicates that the page is viewable by public users. Valid values: ■ 0 - not public ■ 1 - is public
IMAGE	VARCHAR2(350)	Unique document name of the image associated with the page, for example 6001.JPG.
ROLLOVERIMAGE	VARCHAR2(350)	Unique document name of the rollover image associated with the page, or the inactive tab image for the tab, for example 6001.JPG.
TITLEIMAGE	VARCHAR2(350)	Unique document name of the active tab image for the tab, for example 6001.JPG.
LEADER	VARCHAR2(256)	E-mail address of the page contact.
DESCRIPTION	VARCHAR2(2000)	Description of the page.
CREATEDATE	DATE	Date the page was created.
CREATOR	VARCHAR2((256)	User name of the person who created the page.
HAVEITEMSECURITY	NUMBER(1)	Indicates that item level security is enabled for the page. Valid values: ■ 0 - ILS disabled ■ 1 - ILS enabled
ITEMVERSIONING	VARCHAR2(30)	Indicates the level of item versioning for the page. Valid values: ■ versionnone - no versioning ■ versionsimple - simple versioning ■ versionaudit - audit versioning
TOPICID	NUMBER	ID of the category assigned to the page.
TOPIC_SITEID	NUMBER	Page group ID of the category assigned to the page.

Table 11–1 (Cont.) Editable Page Record Properties

Property	Data Type	Description
VALUE	VARCHAR2(2000)	For PL/SQL pages, the PL/SQL code. For JSP pages, the JSP source document name. Do not change this value for JSP pages.
IS_PORTLET	NUMBER(1)	Indicates if the page is published as a portlet. Valid values: ■ 0 - not a portlet ■ 1 - is a portlet
PLSQL_EXECUTOR	VARCHAR2(30)	For PL/SQL type pages, the database schema used to execute the PL/SQL code. Valid values: ■ $PUBLIC$ ■ $CREATOR$ ■ <database user name>
KEYWORDS	VARCHAR2(2000)	Keywords for the page.
IS_READY	NUMBER(1)	Indicates that page creation is complete. Valid values: ■ 1 - page creation is complete
INHERIT_PRIV	VARCHAR2(200)	The page from which this page inherits its privileges. Use the following format: `<page group id>/<page id>`
CACHE_MODE	NUMBER(1)	Caching mode for the page. Valid values: ■ 2 - no caching ■ 1 - cache page definition only ■ 0 - cache page definition and content for x minutes ■ 4 - cache page definition only at system level ■ 3 - cache page definition and content at system level for x minutes
CACHE_EXPIRES	NUMBER(38)	Cache period in minutes.
ALLOW_PAGE_STYLE	NUMBER(1)	For templates, indicates if pages can use a different style. Valid values: ■ 1 - allow pages to use different style.

Table 11–1 (Cont.) Editable Page Record Properties

Property	Data Type	Description
ALLOW_PAGE_ACL	NUMBER(!)	For templates, indicates if pages have different access settings. Valid values: ▪ 1 - allow pages to have different access settings
INIT_JSPFILE	VARCHAR2(256)	For JSP pages, initial JSP file if the JSP source of the page is a JAR or WAR file.
UI_TEMPLATE_ID	NUMBER(38)	ID of HTML page skin.
TEMPLATE_ISPUBLIC	NUMBER(1)	Indicates if the template is ready to use. Valid values: ▪ 1 - template is ready to use
CONTAINER_ID	NUMBER	ID of the container page.
DEFAULT_ITEM_REGION_ ID	NUMBER	ID of the default item region for the page.
DEFAULT_PORTLET_ REGION_ID	NUMBER	ID of the default portlet region for the page.
ITEMTYPE_INHERIT_FLAGS	NUMBER(1)	For WebDAV, indicates if default item types are inherited from parent page. Valid values: ▪ 7 - inherit all item types from parent page ▪ 0 - specify all types on this page
REGFILE_ITEMTYPE	RAW(32)	For WebDAV, GUID of default item type for regular files.
ZIPFILE_ITEMTYPE	RAW(32)	For WebDAV, GUID of default item type for zip files.
IMAGEFILE_ITEMTYPE	RAW(32)	For WebDAV, GUID of default item type for image files.
DISPLAYINPARENT	NUMBER(1)	Indicates if the page is displayed in its parent page's sub page region. Valid values: ▪ 1 - display page in parent's sub page region
SEQ	NUMBER	The sequence (order) of the page in its parent page's sub page region.
ALPHABETICAL_SORT	NUMBER(1)	Indicates that sub pages are displayed in alphabetical order. Valid values: ▪ 1 - display sub pages in alphabetical order
ITEM_PAGE_ID	NUMBER(38)	ID of the item template.

Table 11–1 (Cont.) Editable Page Record Properties

Property	Data Type	Description
ITEM_PAGE_SITE_ID	NUMBER(28)	Page group ID of the item template.
ITEM_PAGE_TABSTRING	VARCHAR2(512)	Tab strings of the item template. Use the following format: `<tab name>:<sub tab name>:...:<sub tab name>`
INHERIT_ITEM_PAGE	NUMBER(1)	For item template only, indicates that items inherit the parent page's item template. Valid values: ▪ 1 - inherit parent page's item template
ALLOW_ITEM_PAGE_ OVERRIDE	NUMBER(1)	For item template only, indicates that items can have their own item template. Valid values: ▪ 1 - allow items on the page to have their own item template
HAS_INPLACE_ITEM	NUMBER(1)	Indicates if the page or tab has a placeholder item. Valid values: ▪ 1 - has a placeholder item
TIMEOUT	NUMBER	Limit time, in seconds, used to fetch portlets.

Users need to have *Manage* privileges on a page to edit its properties.

When updating translatable page attributes, `modify_folder` updates the translation determined by the session language setting. Therefore to make sure you update the translation with the correct values for all the translatable attributes, you must query the page attributes of the same language first.

Example 11–1 shows how to use the `modify_folder` API to edit the English translation of the display name of a page.

Example 11–1 Editing Page Properties

```
declare
  l_page     wwsbr_api.page_record;
begin
  wwctx_api.set_nls_language(
    p_nls_language => wwnls_api.AMERICAN
  );
  select *
  into l_page
  from   <schema>.wwsbr_user_pages
  where  siteid = 33
    and  id = 1
    and  language = wwnls_api.AMERICAN;
  l_page.title := 'New Page Display Name';
  wwsbr_api.modify_folder(
    p_page => l_page
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

p_page is the page record for the page you want to modify.

schema is the portal schema (that is, the schema in which Oracle Portal is installed).

11.2 Editing Content

This section contains the following examples:

- Section 11.2.1, "Setting Item Attributes"
- Section 11.2.2, "Editing an Item"
- Section 11.2.3, "Checking Items Out and In"
- Section 11.2.4, "Using Version Control"

Note: When editing an item, it is good practice to check the item out first to ensure that any changes made by your code are not overwritten by other users. Don't forget to check the item back in after you have completed your edits. For information about checking items out and in, refer to Section 11.2.3, "Checking Items Out and In".

11.2.1 Setting Item Attributes

If you want to set a particular attribute for an item, use the `set_attribute` API. This API updates the value of an attribute for a particular version of an item. The values provided are validated against the data type of the attribute. You can use this API to set the value of a base attribute (see Table 11–2) or a custom attribute. You cannot use this API to edit rejected items, items marked for deletion, and rejected items that have been marked for deletion.

Example 11–2 shows how you can use the `set_attribute` API to set a particular attribute for an item.

Tip: Use the `set_attribute` API when you want to set a single item attribute. If you want to set multiple item attributes, use the `modify_item` API (described in Section 11.2.2, "Editing an Item").

Example 11–2 Setting the Display Name of an Item (set_attribute API)

```
begin
  wwsbr_api.set_attribute(
    p_site_id           => 37,
    p_thing_id          => 8056,
    p_attribute_site_id => wwsbr_api.SHARED_OBJECTS,
    p_attribute_id      => wwsbr_api.ATTRIBUTE_TITLE,
    p_attribute_value   => 'New Display Name'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
```

```
  ...
end;
/
```

- **p_site_id** is the ID of the page group to which the item belongs.
- **p_thing_id** is the unique item ID. The item ID identifies the particular version of the item that you want to edit.
- **p_attribute_site_id** is the ID of the page group to which the attribute belongs. For base attributes, use the wwsbr_api.SHARED_OBJECTS constant.
- **p_attribute_id** is the ID of the attribute. For base attributes, use the constants from the WWSBR_API package as shown in Table 11–2.
- **p_attribute_value** is the new value that you want to assign to the attribute. For base attributes, see Table 11–2.

For information about how to obtain an object ID, for example a page group ID or item ID, refer to Section 10.3, "Finding an Object ID".

Table 11–2 provides guidelines for setting the values of base attributes.

Table 11–2 Attribute Values for Base Attributes

Attribute	Constant	Value (p_attribute_value)
Author	wwsbr_api.ATTRIBUTE_AUTHOR	Any varchar2(50) value
Category	wwsbr_api.ATTRIBUTE_CATEGORY	<pagegroupid>_<categoryid>
Character Set	wwsbr_api.ATTRIBUTE_CHARSET	■ wwsbr_api.VALUE_BIG5 ■ wwsbr_api.VALUE_EUC_JP ■ wwsbr_api.VALUE_GBK ■ wwsbr_api.VALUE_ISO_8859 ■ wwsbr_api.VALUE_SHIFT_JIS ■ wwsbr_api.VALUE_US_ASCII ■ wwsbr_api.VALUE_UTF8 ■ wwsbr_api.VALUE_WINDOWS_1252
Description	wwsbr_api.ATTRIBUTE_DESCRIPTION	Any varchar2(2000) value
Display Name	wwsbr_api.ATTRIBUTE_TITLE	Any varchar2(256) value
Display Option	wwsbr_api.ATTRIBUTE_DISPLAYOPTION	■ wwsbr_api.IN_PLACE to display the item directly in place ■ wwsbr_api.FULL_SCREEN to display the item in the full browser window ■ wwsbr_api.NEW_WINDOW to display the item in a new browser window
Enable Item Check-out	wwsbr_api.ATTRIBUTE_ITEMCHECKOUT	■ wwsbr_api.ENABLE_ITEM_FOR_CHECK_OUT ■ wwsbr_api.DISABLE_ITEM_FOR_CHECK_OUT

Table 11–2 (Cont.) Attribute Values for Base Attributes

Attribute	Constant	Value (p_attribute_value)
Expiration Period	wwsbr_api.ATTRIBUTE_ EXPIRATIONPER	■ wwsbr_api.PERMANENT to un-expire the item ■ Any numeric value to set the expire mode to NUMBER ■ Any date value greater than the publish date to set the expire date to the value passed and the expire mode to DATE
File	wwsbr_api.ATTRIBUTE_ FILE	The path and file name of the file that you want to upload
Image	wwsbr_api.ATTRIBUTE_ IMAGE	The path and file name of the image that you want to upload
ImageMap	wwsbr_api.ATTRIBUTE_ IMAGEMAP	Any varchar2(4000) value
Image Alignment	wwsbr_api.ATTRIBUTE_ IMAGEALIGN	■ wwsbr_api.ALIGN_TEXT_TOP ■ wwsbr_api.ALIGN_ABSOLUTE_ BOTTOM ■ wwsbr_api.ALIGN_ABSOLUTE_ MIDDLE ■ wwsbr_api.ALIGN_BOTTOM ■ wwsbr_api.ALIGN_RIGHT ■ wwsbr_api.ALIGN_TOP ■ wwsbr_api.ALIGN_LEFT ■ wwsbr_api.ALIGN_MIDDLE ■ wwsbr_api.ALIGN_BASELINE ■ wwsbr_api.ALIGN_IMAGE_ ABOVE_LINK
Item Link	wwsbr_api.ATTRIBUTE_ ITEM_LINK	<pagegroupid>_<itemid>
Item Template	wwsbr_api.ATTRIBUTE_ ITEM_TEMPLATE	<pagegroupid>_<itemtemplateid>
Keywords	wwsbr_api.ATTRIBUTE_ KEYWORDS	Any varchar2(2000) value
Mime Type	wwsbr_api.ATTRIBUTE_ MIME_TYPE	Any of the mime type constants in the WWSBR_API package. For example: ■ wwsbr_api.VALUE_TEXT_HTML ■ wwsbr_api.VALUE_TEXT_PLAIN ■ wwsbr_api.VALUE_IMAGE_GIF For a list of the available mime type constants, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center: `http://portalcenter.oracle.com` In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

Table 11–2 (Cont.) Attribute Values for Base Attributes

Attribute	Constant	Value (p_attribute_value)
Name	wwsbr_api.ATTRIBUTE_ NAME	Any varchar2(256) value Specify a name that is unique within the page and all its tabs and sub-pages
Page Link	wwsbr_api.ATTRIBUTE_ PAGE_LINK	<pagegroupid>_<pageid>
Perspectives	wwsbr_api.ATTRIBUTE_ PERSPECTIVES	<pagegroupid>_<perspectiveid> Separate multiple perspectives with commas. Non-existing perspective IDs are identified by the API. Only those perspectives that are available to the page group are added to the item.
PL/SQL	wwsbr_api.ATTRIBUTE_ PLSQL	Any varchar2(2000) value
Publish Date	wwsbr_api.ATTRIBUTE_ PUBLISHDATE	Any date value This is a required attribute.
Rollover Image	wwsbr_api.ATTRIBUTE_ ROLLOVERIMAGE	The path and file name of the image file that you want to upload

Table 11–2 (Cont.) Attribute Values for Base Attributes

Attribute	Constant	Value (p_attribute_value)
Smart Link	wwsbr_api.ATTRIBUTE_ SMARTLINK	▪ wwsbr_api.VALUE_ACCOUNT_ INFO ▪ wwsbr_api.VALUE_ADVANCED_ SEARCHE ▪ wwsbr_api.VALUE_BUILDER ▪ wwsbr_api.VALUE_COMMUNITY ▪ wwsbr_api.VALUE_CONTACT ▪ wwsbr_api.VALUE_EDIT_PAGE ▪ wwsbr_api.VALUE_FAVOURITES ▪ wwsbr_api.VALUE_HELP ▪ wwsbr_api.VALUE_PORTAL_ HOME ▪ wwsbr_api.VALUE_CUST_ MOBILE_HOME_PAGE ▪ wwsbr_api.VALUE_NAVIGATOR ▪ wwsbr_api.VALUE_PAGE_GROUP_ HOME ▪ wwsbr_api.VALUE_PERSONAL_ PAGE ▪ wwsbr_api.VALUE_CUSTOMIZE_ PAGE ▪ wwsbr_api.VALUE_PORTLET_ REPOS_REF_STATUS ▪ wwsbr_api.VALUE_PORTLET_ REPOS ▪ wwsbr_api.VALUE_PROPERTY_ SHEET ▪ wwsbr_api.VALUE_REF_PORTLET_ REPOS ▪ wwsbr_api.VALUE_REFRESH_ PAGE ▪ wwsbr_api.VALUE_SUBSCRIBE
Smart Text	wwsbr_api.ATTRIBUTE_ SMARTTEXT	▪ wwsbr_api.VALUE_SMART_TEXT_ CURRENT_DATE ▪ wwsbr_api.VALUE_SMART_TEXT_ CURRENT_PAGE ▪ wwsbr_api.VALUE_SMART_TEXT_ CURRENT_USER
Text	wwsbr_api.ATTRIBUTE_ TEXT	Any varchar2(4000) value
URL	wwsbr_api.ATTRIBUTE_ URL	Any URL value up to varchar2(2000)

Table 11–2 (Cont.) Attribute Values for Base Attributes

Attribute	**Constant**	**Value (p_attribute_value)**
Version Number	wwsbr_api.ATTRIBUTE_ VERSION_NUMBER	Any positive number. The number does not have to be an integer. Specify a version number that is unique, that is, it should not be the same as any existing version of the item. If you supply a value that is the same as an existing version number for the item, then it will be set to one more than the highest version number for the item.

11.2.2 Editing an Item

To edit an existing item, use the `modify_item` API. This API enables you to set the properties of an item and also enables you to perform the following:

- Add new versions of the item (if item version is enabled for the page). See Section 11.2.4, "Using Version Control".
- Enable item level security for the item (if item level security (ILS) is enabled for the page). See Section 15.3, "Setting Item Level Privileges" for an alternative way of doing this.
- Upload files that are associated with the item. If the file that you want to associate with the item has already been uploaded, use the `modify_item_post_upload` API instead.

You should use this API to edit only those items with an ACTIVE value of 1. That is, do not use it to edit pending, rejected, or deleted items.

You must pass both the item's master ID to identify the item and its ID to identify the specific version of the item that you want to update. The specified ID does not have to be the current version of the item. For information about obtaining an object ID, refer to Section 10.3, "Finding an Object ID".

The `modify_item` API modifies the value of every attribute, even if the attribute is not passed in the parameter list. If an attribute is not passed, its value will revert to the default value of the parameter. If you want to preserve the current value of attributes, you must retrieve those values for the item and pass them to the API. The value of attributes can be retrieved from the following views and APIs:

- WWSBR_ALL_ITEMS for built-in attributes
- WWSBR_ITEM_ATTRIBUTES for custom attributes
- WWSBR_ITEM_TYPE_ATTRIBUTES for a list of attributes for an item type
- WWSBR_ITEM_PERSPECTIVES for perspectives
- `wwsec_api.grantee.list` for a list of privileges on the item type. See Section 15.3, "Setting Item Level Privileges" for more information on using this API.

Example 11–3 updates the display name and text of a text item.

Example 11–3 Editing an Item (modify_item API)

```
declare
  l_item_master_id number;
begin
  l_item_master_id := wwsbr_api.modify_item(
    p_master_item_id => 453,
    p_item_id        => 454,
    p_caid           => 33,
    p_folder_id      => 45,
    p_display_name   => 'Movie Review',
    p_text           => 'This is the text of the review.'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_master_item_id** is the master ID of the item. You can look up this value in the MASTERID column of the WWSBR_ALL_ITEMS view.
- **p_item_id** is the ID of the item. This identifies the version of the item and does not have to be the item's current version.

  Tip: To identify the item's current version, use the following query:

  ```
  select id
  from wwsbr_all_items
  where master_id = 453
    and is_current_version = 1;
  ```

- **p_caid** is the ID of the page group to which the item belongs.
- **p_folder_id** is the ID of the page on which the item appears.
- **p_display_name** is the display name (title) of the item.
- **p_text** is the text for the item (the item in this example is a text item).

11.2.3 Checking Items Out and In

If an item is enabled for check-out, before editing it, you should check the item out so that other users cannot make changes at the same time (Example 11–4).

Example 11–4 Checking an Item Out and In (check_out_item and check_in_item)

```
begin
  -- Check out the item.
  wwsbr_api.check_out_item(
    p_master_item_id => 12345, -- Master ID is the same for all versions.
    p_caid           => 33
  );
  -- Update the display name of the item.
  wwsbr_api.set_attribute(
    p_site_id => 33,
    p_thing_id => 8056, -- Unique item ID.
    p_attribute_site_id => wwsbr_api.SHARED_OBJECTS,
    p_attribute_id      => wwsbr_api.ATTRIBUTE_TITLE,
    p_attribute_value   => 'New Display Name'
  );
  -- Check the item back in.
  wwsbr_api.check_in_item(
    p_master_item_id => 12345, -- Master ID is the same for all versions.
    p_caid           => 33
  );
```

```
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_ache_invalidation;
exception
  ...
end;
/
```

- **p_master_item_id** is the master ID of the item being checked out or in.
- **p_caid** is the ID of the page group to which the item belongs.

For information about the `set_attribute` API used in Example 11–4, refer to Section 11.2.1, "Setting Item Attributes".

Note: To check whether an item is already checked out, query the IS_ITEM_CHECKEDOUT column of the WWSBR_ALL_ITEMS view. For more information about this view, refer to Section F.2.5, "WWSBR_ALL_ITEMS".

11.2.4 Using Version Control

When you edit an item using the `modify_item` API, if you set `p_addnewversion = TRUE`, a new version is created for the item. The new version is given a new ID (the master ID stays the same). If you do not want the new version of the item to be immediately published as the current version, set `p_add_as_current_version = FALSE`.

Example 11–5 shows how you can use the modify_item API to create a new version of an item.

Example 11–5 Creating a New Version of an Item (modify_item API)

```
declare
  l_item_master_id number;
begin
  l_item_master_id := wwsbr_api.modify_item(
    p_master_item_id => 453,
    p_item_id        => 454,
    p_caid           => 33,
    p_folder_id      => 45,
    p_display_name   => 'Movie Review',
    p_text           => 'This is the text of the review.',
    p_addnewversion  => true
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

For a description of the parameters used here, see Example 11–3.

11.3 Reorganizing Content

Content is very rarely static, so you may find that you need to move content around within your portal.

This section contains the following examples:

- Section 11.3.1, "Moving an Item to a Different Page"
- Section 11.3.2, "Moving Pages"

11.3.1 Moving an Item to a Different Page

You can use the `move_item` API to move an item to a different page in the same page group or to a page in a different page group. Moving an item preserves the item's metadata. To move an item to a different page group, the item must not be:

- based on a local item type.
- associated with a local category.
- associated with any local perspectives.

Example 11–6 moves an item from one page group to a page in a different page group.

Example 11–6 Moving an Item to a Different Page Group (move_item API)

```
begin
  wwsbr_api.move_item(
    p_caid           => 33,
    p_master_item_id => 12345,
    p_dest_caid      => 53,
    p_dest_page_id   => 1,
    p_dest_region_id => 5
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_caid** is the ID of the page group to which the item currently belongs.
- **p_master_item_id** is the master ID of the item that you want to move.
- **p_dest_caid** is the ID of the page group to which you want to move the item.
- **p_dest_page_id** is the ID of the page to which you want to move the item.
- **p_dest_region_id** is the ID of the item region of the page in which you want to move the item. If you do not specify a region ID, the item is moved to the page's default item region, which may not be desirable.

11.3.2 Moving Pages

You can use the `move_folder` API to move a page within the same page group. You cannot move a page to a different page group.

To move a page, users must have *Manage* privileges on the page being moved *and* the page under which the page is being moved.

Example 11–7 shows how to move a page.

Example 11–7 Moving a Page (move_folder API)

```
begin
  wwsbr_api.move_folder(
    p_id        => 12345,
    p_siteid    => 33,
    p_parent_id => 10000
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_id** is the ID of the page that you want to move.
- **p_siteid** is the ID of the page group to which the page belongs.
- **p_parent_id** is the ID of the page under which you want to move the page.

11.3.3 Moving Categories and Perspectives

You can, if necessary, move categories and perspectives around within a page group. You cannot move a category or perspective to a different page group.

To move a category or perspective, a user must have *Manage Classifications* privileges or higher on the owning page group.

To move a category, use the `move_category` API, as shown in Example 11–8.

Example 11–8 Moving a Category (move_category API)

```
begin
  wwsbr_api.move_category(
    p_src_id => 2000,
    p_dest_id => 3000,
    p_siteid  => 33
  );
exception
  ...
end;
/
```

To move a perspective, use the `move_perspective` API, as shown in Example 11–9.

Example 11–9 Moving a Perspective (move_perspective API)

```
begin
  wwsbr_api.move_perspective(
    p_src_id => 2000,
    p_dest_id => 3000,
    p_siteid  => 33
  );
exception
  ...
end;
/
```

- **p_src_id** is the ID of the category or perspective that you want to move.
- **p_dest_id** is the ID of the category or perspective under which you want to move the category or perspective.
- **p_siteid** is the ID of the page group to which the category or perspective belongs.

11.4 Copying Content

If you want to add content that is the same as, or similar to, some content that already exists, instead of creating it all over again, you can copy the existing content.

This section contains the following examples:

- Section 11.4.1, "Copying Items"
- Section 11.4.2, "Copying Pages"

11.4.1 Copying Items

You can use the `copy_item` API to copy an item to a different page in the same page group or to a page in a different page group. To copy an item to a different page group, the item must not be any of the following:

- based on a local item type
- associated with a local category
- associated with any local perspectives

Copying an item preserves the item's metadata.

Example 11–10 copies an item to a page in a different page group:

Example 11–10 Creating a Copy of an Item in a Different Page Group (copy_item API)

```
begin
  wwsbr_api.copy_item(
    p_caid           => 33,
    p_master_item_id => 12345,
    p_dest_caid      => 53,
    p_dest_page_id   => 1,
    p_dest_region_id => 5
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_caid** is the ID of the page group to which the item currently belongs.
- **p_master_item_id** is the master ID of the item that you want to copy.
- **p_dest_caid** is the ID of the page group to which you want to copy the item.
- **p_dest_page_id** is the ID of the page to which you want to copy the item.
- **p_dest_region_id** is the ID of the item region of the page in which you want to copy the item. If you do not specify a region ID, the item is copied to the page's default item region, which may not be desirable.

11.4.2 Copying Pages

You can use the `copy_folder` API to copy a page within the same page group. You cannot copy a page to a different page group.

To copy a page, users must have *View* privileges on the page being copied and *Manage* privileges on the page under which the page is being copied.

Example 11–11 shows how to copy a page.

Example 11–11 Creating a Copy of a Page (copy_folder API)

```
declare
  l_new_pageid number;
begin
  l_new_pageid := wwsbr_api.copy_folder(
    p_id        => 12345,
    p_siteid    => 33,
    p_parent_id => 10000
    p_name      => 'page1',
    p_title     => 'page1'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_id** is the ID of the page that you want to copy.
- **p_siteid** is the ID of the page group to which the page belongs.
- **p_parent_id** is the ID of the page under which you want to copy the page.
- **p_name** is the name for the new page created by the copy operation.
- **p_title** is the display name for the new page created by the copy operation.

11.5 Deleting Content

When content is no longer required, you can use APIs to remove it from the portal and free up space.

This section contains the following examples:

- Section 11.5.1, "Deleting Items"
- Section 11.5.2, "Deleting Pages"

11.5.1 Deleting Items

You can use the `delete_item` API to delete all versions of an item or just a single specific version. This API also deletes all sub items associated with the versions being deleted. When you delete an item using this API, you can choose to accept the default delete mode for the page group or override this setting.

Example 11–12 deletes version 1 of an item permanently from the page group. The item version is permanently removed, even if the page group is set up to retain deleted items.

Example 11–12 Deleting an Item (delete_item API)

```
begin
  procedure delete_item(
    p_master_item_id => 48037,
    p_caid           => 54,
    p_version_number => 1,
    p_mode           => wwsbr_api.DELETE_ITEM_PURGE
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_master_item_id** is the master ID of the item that you want to delete.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_version_number** is the version of the item that you want to delete. If you attempt to delete the current version of the item, the ITEM_ACTIVE_VERSION exception is raised. If you do not specify an item version, all versions of the item are deleted.

  Tip: If you want to delete the current version of the item, you must either delete all versions, or use the Oracle Portal user interface to revert to a previous version.

- **p_mode** is the mode to use when deleting the item. It can take the following values:
  - wwsbr_api.DELETE_ITEM_DEFAULT uses the page group setting.
  - wwsbr_api.DELETE_ITEM_PURGE deletes the item immediately, regardless of the page group setting.
  - wwsbr_api.DELETE_ITEM_LAZYDELETE marks the item as deleted, regardless of the page group setting.

To restore a previously deleted item, use the `undelete_item` API (Example 11–13). If the item had any sub-items, those sub-items are also restored.

Note: You can restore a deleted item only if the page group that owns the item is set up to retain deleted items, or the item was deleted using the wwsbr_api.DELETE_ITEM_LAZYDELETE mode, and the item has not been purged from the content repository.

Example 11–13 Restoring a Previously Deleted Item (undelete_item API)

```
begin
  wwsbr_api.undelete_item(
    p_thing_id => 12345,
    p_caid     => 33
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_thing_id** is the unique ID of the deleted item that you want to restore.
- **p_caid** is the ID of the page group to which the item belongs.

11.5.2 Deleting Pages

Use the `delete_folder` API to delete a page. Users need to have *Manage* privileges on the page to be able to delete it.

Note: Deleted pages cannot be restored by another API call.

Example 11–14 shows how to delete a page.

Example 11–14 Deleting a Page (delete_folder API)

```
begin
  wwsbr_api.delete_folder(
    p_id     => 12345,
    p_siteid => 33
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_id** is the ID of the page that you want to delete.
- **p_siteid** is the ID of the page group to which the page belongs.

12 Extending Your Portal

This chapter describes how to use the APIs provided with Oracle Portal to extend your portal. It contains the following sections:

- Section 12.1, "Creating a Page Group"
- Section 12.2, "Creating Pages"
- Section 12.3, "Creating Categories and Perspectives"
- Section 12.4, "Creating Items"
- Section 12.5, "Setting Perspectives Attributes of Pages and Items"
- Section 12.6, "Approving and Rejecting Items"

More On OTN

For more information about any of these APIs, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

> **Tip:** Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

12.1 Creating a Page Group

To create a new page group, use the `add_content_area` API as shown in Example 12–1.

Example 12–1 Creating a Page Group (add_content_area API)

```
declare
  l_new_page_group_id number;
begin
  -- Create the page group.
  l_new_page_group_id := wwsbr_api.add_content_area(
    p_name             => 'ENTERTAINMENT',
    p_display_name     => 'Entertainment Page Group',
    p_versioning       => wwsbr_api.VERSIONING_AUDIT,
    p_default_language => 'us'
  );
  -- process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
```

```
  ...
end;
/
```

- **p_name** is the internal name for the new page group. This name is used in path based URLs and must be unique.
- **p_display_name** is the display name for the new page group.
- **p_versioning** is the version level for items in the page group. It can take the following values:
  - wwsbr_api.VERSIONING_NONE
  - wwsbr_api.VERSIONING_SIMPLE
  - wwsbr_api.VERSIONING_AUDIT
- **p_default_language** is the default language for the page group.

12.2 Creating Pages

If you want to create a new page, use the `add_folder` API. By default, the new page has two regions: a portlet region containing the default navigation page of the page group, and an item region. This means that pages created programmatically, rather than through the Oracle Portal user interface, may be quite limited in their layout.

However, you can programmatically create pages with a more sophisticated layout by basing the parent page of the new page on a Portal Template with the desired layout. Then if you configure the page group to automatically copy parent page properties, your new page will use the same template as its parent. Using this method means that your pages can have any layout that you require.

You cannot use this API to create JSP pages or navigation pages. To create these types of pages, use the Create Page Wizard in Oracle Portal.

If the page type on which you base the page has default values set for any page properties, these values override any values set using this API.

To define privileges for your new page, use the APIs in the WWSEC_API package (Example 12–2). For more information, refer to Section 15.2, "Setting Page Level Privileges".

Example 12–2 Creating a Page (add_folder API)

```
declare
  l_new_page_id       number;
  l_caid              number := 33;
begin
  -- create the page.
  l_new_page_id := wwsbr_api.add_folder(
    p_caid              => l_caid,
    p_name              => 'ENTERTAINMENT',
    p_display_name      => 'Entertainment Page',
    p_type_id           => wwsbr_api.FOLDER_TYPE_CONTAINER,
    p_type_caid         => wwsbr_api.SHARED_OBJECTS,
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
```

```
/
```

- **p_caid** is the ID of the page group to which you want to add the page.
- **p_name** is the internal name for the new page. This name is used in path based URLs.
- **p_display_name** is the display name (title) for the new page.
- **p_type_id** is the ID of the page type on which you want to base the page. The page type must be available to the page group in which you are creating the page. Seeded page types have the following constants defined:
  - wwsbr_api.FOLDER_TYPE_CONTAINER
  - wwsbr_api.FOLDER_TYPE_URL
  - wwsbr_api.FOLDER_TYPE_PLSQL
  - A value from the ID column of the WWSBR_FOLDER_TYPES view

  You can find the IDs for custom page types by querying the ID column of the WWSBR_FOLDER_TYPES view.
- **p_type_caid** is the ID of the page group to which the page type used for the page belongs. This value must be the same as the page group in which you are creating the page (p_caid) or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).

12.3 Creating Categories and Perspectives

If you have already defined a quite large taxonomy for your portal, you might prefer to set it up programmatically, rather than use the Oracle Portal user interface.

Use the `add_category` API to create a new category (Example 12–3). Use the `add_perspective` API to create a new perspective (Example 12–4). To create a category or perspective, users must have *Manage Classifications* privileges or higher on the page group.

Example 12–3 Creating a Category (add_category API)

```
declare
  l_new_category_id number;
  l_caid            number := 33;
begin
  l_new_category_id := wwsbr_api.add_category(
    p_caid         => l_caid,
    p_parent_id    => 0,
    p_name         => 'newcategory1',
    p_display_name => 'New Category 1'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

Example 12–4 Creating a Perspective (add_perspective API)

```
declare
```

```
  l_new_perspective_id number;
  l_caid               number := 33;
begin
  l_new_perspective_id := wwsbr_api.add_perspective(
    p_caid         => 33,
    p_parent_id    => 0,
    p_name         => 'newperspective1',
    p_display_name => 'New Perspective 1'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_caid** is the ID of the page group in which you want to create the category or perspective.
- **p_parent_id** is the ID of the category or perspective under which you want to create the new category or perspective. If this is a top-level category or perspective, use 0 (zero).
- **p_name** is the internal name for the new category or perspective.
- **p_display_name** is the display name for the new category or perspective.

12.4 Creating Items

To create an item on a page, use the `add_item` API (Example 12–5).

This API returns the master ID of the item, which is not the item's unique ID. To look up the ID of the item after it is created, query the WWSBR_ALL_ITEMS view as shown in Example 10–7. For more information about the difference between an item's unique ID and its master item ID, refer to Section 10.2, "API Parameters".

Example 12–5 Creating a Text Item (add_item API)

```
declare
  l_new_item_master_id number;
begin
  l_new_item_master_id := wwsbr_api.add_item(
    p_caid              => 33,
    p_folder_id         => 13923,
    p_display_name      => 'Movie Review',
    p_type_id           => wwsbr_api.ITEM_TYPE_TEXT,
    p_type_caid         => wwsbr_api.SHARED_OBEJCTS,
    p_region_id         => 5,
    p_text              => 'This is the text of the review.',
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_caid** is the ID of the page group that owns the page in which you want to create the item.
- **p_folder_id** is the ID of the page in which you want to create the item.

- **p_display_name** is the display name (title) of the new item.
- **p_type_id** is the ID of the item type on which the item is based. Use the constants defined in the WWSBR_API package. Example are as follows:
  - wwsbr_api.ITEM_TYPE_FILE
  - wwsbr_api.ITEM_TYPE_TEXT
  - wwsbr_api.ITEM_TYPE_URL

  For a full list of constants for the seeded item types, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

  http://portalcenter.oracle.com

  In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

  You can find the IDs for custom item types by querying the ID column of the WWSBR_ITEM_TYPES view.

  The item type must be available in the page group on which you are creating the item, and therefore listed in the WWSBR_CONTENT_AREA_ITEM_TYPES view.
- **p_type_caid** is the ID of the page group to which the item type used for the item belongs. This value must be the same as the page group to which you are adding the item (p_caid), or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).
- **p_region_id** is the ID of the region in which you want to create the item. If you do not specify a region, or if the region ID is invalid, the item is placed in the default item region for the page. Use the WWSBR_ALL_FOLDER_REGIONS view to look up the region ID.
- **p_text** is the text for a text item.

Creating File Items

If an object is associated with one or more files (for example, images, file items, and so on), its APIs allow you to upload these files when creating or modifying the object. You can pass a file location consisting of a directory path and file name to the file and image parameters for these APIs. The directory in which you place the files must be visible to your database (that is, local to the database server) and the files themselves must have proper permissions. If you see strange exceptions resulting from API calls in which you are trying to upload files, double check the permissions on the directories and files.

When the file that you want to upload resides on the same server as the database in which Oracle Portal is installed, you can upload the file at the same time as you create or update the item. Example 12–6 shows how you might do this.

Example 12–6 Creating a File Item (add_item API)

```
declare
  l_item_masterthing_id number;
begin
  -- Add an item that resides on the same server as the OracleAS Portal
  -- content repository database.
  l_item_masterthing_id := wwsbr_api.add_item(
    p_caid          => 53,  -- A known page group ID.
    p_folder_id     => 1,   -- A known page ID.
    p_display_name  => 'My File',
    p_type_id       => wwsbr_api.ITEM_TYPE_FILE,
    p_type_caid     => wwsbr_api.SHARED_OBJECTS,
    p_region_id     => 513, -- A known region on the page.
    p_description   => 'Description of my file',
    p_file_filename => '/tmp/myfile.txt'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_caid** is the ID of the page group that owns the page in which you want to create the item.
- **p_folder_id** is the ID of the page in which you want to create the item.
- **p_display_name** is the display name (title) of the new item.
- **p_type_id** is the ID of the item type on which the item is based.
- **p_type_caid** is the ID of the page group to which the item type used for the item belongs. This value must be the same as the page group to which you are adding the item (p_caid), or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).
- **p_region_id** is the ID of the region in which you want to create the item. If you do not specify a region, or if the region ID is invalid, the item is placed in the default item region for the page. Use the WWSBR_ALL_FOLDER_REGIONS view to look up the region ID.
- **p_description** is a description of the item.
- **p_file_filename** is the directory path and file name of the file associated with this item. The file must be located on the same server as database in which Oracle Portal is installed.

If the file does not reside on the same server as the database in which Oracle Portal is installed, you must first load the file into the content repository document tables using the `upload_blob` API (Example 12–7). The `upload_blob` API returns a unique document name. Use the `add_item_post_upload` API to create a new item that claims the file specified by the returned document name.

Note: Your calling application must declare a BLOB and assign it to an object compatible with the database BLOB data type (refer to *Application Developer's Guide - Large Objects* in your Oracle Database documentation set). You can then pass the BLOB to the `p_blob` parameter of the `upload_blob` API.

Example 12–7 Uploading a File to the Content Repository and Creating a File Item Using the Uploaded File (upload_blob API and add_item_post_upload API)

```
declare
  l_blob                blob;
  l_blob_filename       varchar2(250);
  l_mime_type           varchar2(100) := 'text/html';
  l_doc_name            varchar2(500);
  l_display_name        varchar2(250);
  l_region_id           number        := 2013;
  l_file_name           varchar2(100);
  l_image_name          varchar2(100);
  l_site_id             number        := 73;
  l_page_id             number        := 1;
  l_item_type_id        number        := wwsbr_api.ITEM_FILE_TYPE;
  l_item_type_siteid    number        := wwsbr_api.SHARED_OBJECTS;
  l_description         varchar2(1000);
  l_item_masterthing_id number;
begin
  -- Your calling application must define the my_get_blob() function and retrieve
  -- your document into a blob so that it can be uploaded in the subsequent step.
  l_blob := my_get_blob('8001.HTML');
  -- Upload the BLOB to the Oracle Portal document table.
  l_blob_filename := 'index2.html';
  l_doc_name := wwsbr_api.upload_blob(
    p_file_name => l_blob_filename,
    p_blob      => l_blob,
    p_mime_type => l_mime_type
  );
  l_display_name := l_blob_filename;
  l_file_name := l_doc_name;
  l_description := 'File uploaded to portal = ' || l_doc_name;
  -- Use add_item_post_upload() to claim the document and add the item to a page.
  l_item_masterthing_id := wwsbr_api.add_item_post_upload(
    p_caid          => l_site_id,
    p_folder_id     => l_page_id,
    p_display_name  => l_display_name,
    p_file_name     => l_file_name,
    p_type_id       => l_item_type_id,
    p_type_caid     => l_item_type_siteid,
    p_region_id     => l_region_id,
    p_description   => l_description,
    p_display_option => wwsbr_api.FULL_SCREEN
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

The following are parameters for `upload_blob`:

- **p_file_name** is the file name that you want to assign to the BLOB. This is not the value returned by the function. Usually this is the file name by which the file is known on the source file system.
- **p_blob** is the BLOB containing the content.
- **p_mime_type** is the MIME type for the BLOB. Use the predefined constants provided in the WWSBR_API package.

The following are parameters for `add_item_post_upload`:

- **p_caid** is the ID of the page group that owns the page in which you want to create the item.
- **p_folder_id** is the ID of the page in which you want to create the item.
- **p_display_name** is the display name (title) of the new item.
- **p_file_name** is the internal document name for the file item, matching a document in the document table. Pass the value returned by `upload_blob` to this parameter to reference the uploaded document as the file for this item.

- **p_type_id** is the ID of the item type on which the item is based.
- **p_type_caid** is the ID of the page group to which the item type used for the item belongs. This value must be the same as the page group to which you are adding the item (p_caid), or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).
- **p_region_id** is the ID of the region in which you want to create the item. If you do not specify a region, or if the region ID is invalid, the item is placed in the default item region for the page. Use the WWSBR_ALL_FOLDER_REGIONS view to look up the region ID.
- **p_description** is a description of the item.
- **p_file_filename** is the directory path and file name of the file associated with this item. The file must be located on the same server as database in which Oracle Portal is installed.
- **p_display_option** is how the item should be displayed. Use the following predefined constants:
  - wwsbr_api.FULL_SCREEN to display a link that, when clicked, displays the item in the same browser window.
  - wwsbr_api.NEW_WINDOW to display a link that, when click, displays the item in a new browser window.
  - wwsbr_api.IN_PLACE to display the item in the region (this applies to text and PL/SQL items only).

If you have already created the item that you want to use to claim the uploaded document, use the `modify_item_post_upload` API.

12.5 Setting Perspectives Attributes of Pages and Items

When creating a page or item you can, optionally, specify the perspectives that apply to that page or item.

Example 12–8 shows how you specify the perspectives that apply to a page when creating the page. Example 12–9 shows how you specify the perspectives that apply to an item when creating the item.

Example 12–8 Specifying Perspectives When Creating a Page

```
declare
  l_new_page_id       number;
  l_perspective_ids   wwsbr_api.g_perspectiveidarray;
  l_perspective_caids wwsbr_api.g_caid_array;
begin
  select id, caid
  bulk collect into l_perspective_ids, l_perspective_caids
  from wwsbr_all_perspectives
  where caid in (l_caid, wwsbr_api.shared_objects);
  l_new_page_id := wwsbr_api.add_folder(
    p_caid             => 33,
    p_name             => 'ENTERTAINMENT',
    p_display_name     => 'Entertainment Page',
    p_type_id          => wwsbr_api.FOLDER_TYPE_CONTAINER,
    p_type_caid        => wwsbr_api.SHARED_OBJECTS,
    p_perspectives     => l_perspective_ids,
    p_perspective_caid => l_perspective_caids
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

Example 12–9   Specifying Perspectives When Creating an Item

```
declare
  l_new_item_master_id number;
  l_perspective_ids    wwsbr_api.g_perspectiveidarray;
  l_perspective_caids  wwsbr_api.g_caid_array;
begin
  -- Select perspectives with the name prefix = 'ENTERTAINMENT'.
  select id, caid
  bulk collect into l_perspective_ids, l_perspective_caids
  from wwsbr_all_perspectives
  where caid in (l_caid, wwsbr_api.shared_objects) and
        display_name like 'ENTERTAINMENT%';
  l_new_item_master_id := wwsbr_api.add_item(
    p_caid             => 33,
    p_folder_id        => 13923,
    p_display_name     => 'Movie Review',
    p_type_id          => wwsbr_api.ITEM_TYPE_TEXT,
    p_type_caid        => wwsbr_api.SHARED_OBEJCTS,
    p_region_id        => 5,
    p_text             => 'This is the text of the review.',
    p_perspectives     => l_perspective_ids,
    p_perspectives_caid => l_perspective_caids
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_perspectives** is an array of perspective IDs.
- **p_perspective_caid** is an array of page group IDs for the perspectives identified in `p_perspectives`. The position of each element in this array must match the position of the corresponding perspective ID in `p_perspectives`. The values in `p_perspective_caid` must be the same as the page group in which you are creating the page or item (p_caid) or the Shared Objects page group (use the wwsbr_api.SHARED_OBJECTS constant).

For descriptions of the other parameters in these examples, refer to Example 12–2 and Example 12–5.

12.6 Approving and Rejecting Items

The WWSBR_API package includes two APIs to help with managing approvals. To see how you might use these APIs in conjunction with the Content Management Event Framework, refer to Section 16.7, "Example: Item Validation" and Section 16.8, "Example: Integrating External Workflow"

Example 12–10 shows how to approve a pending item. Example 12–11 shows how to reject a pending item.

Example 12–10 Approving an Item

```
begin
  wwsbr_api.approve(
    p_item_id => 8056,
    p_site_id => 53,
    p_comment => 'Item approved'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
end;
/
```

Example 12–11 Rejecting an Item

```
begin
  wwsbr_api.reject(
    p_item_id => 8056,
    p_site_id => 53,
    p_comment => 'Item rejected'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
end;
/
```

- **p_item_id** is the unique ID of the item being approved or rejected.
- **p_site_id** is the ID of the page group to which the item belongs.
- **p_comment** provides additional information about the approval or rejection of the item.

13 Searching Portal Content

This chapter describes how to use the public search APIs provided with Oracle Portal. The search APIs are available in the WWSRC_API package.

The public search APIs enable you to search your portal programmatically and return search results as content records or an XML document. Using these APIs, you can design completely custom search forms that seamlessly integrate with Oracle Portal pages. You can also customize how portal search results get displayed on a page; you are no longer limited to the search results portlet provided with Oracle Portal. As the public search APIs can return the results in XML, you may process or format the search results however you require.

You can also use the search APIs to display portal content in any custom application. For example, if you have a Web-based application you could easily provide a search form in the application and display the search results from Oracle Portal there as well.

The WWSRC_API package provides the following two types of API:

- Search submission/execution APIs
  - Item search
  - Page search
  - Category search
  - Perspective search
- Search results APIs
  - Get item results as XML document
  - Get page results as XML document

To determine the structure of the search results returned by the search APIs, refer to the appropriate secure content repository view, as shown in Table 13–1.

Table 13–1 Search Results to Secure View Mapping

Search API	Secure Content Repository View
wwsrc_api.item_search	WWSBR_ALL_ITEMS
wwsrc_api.page_search	WWSBR_ALL_FOLDERS
wwsrc_api.category_search	WWSBR_ALL_CATEGORIES
wwsrc_api.perspective_search	WWSBR_ALL_PERSPECTIVES

For example, the results returned by the `item_search` API have the same structure as the WWSBR_ALL_ITEMS view. For information about the structure of the secure views, refer to Chapter F, "Content Management APIs and Views".

This chapter contains the following sections:

- Section 13.1, "Searching For Items Across All Page Groups"
- Section 13.2, "Searching For Pages in Specific Page Groups"
- Section 13.3, "Searching For Items By Attribute"
- Section 13.4, "Transforming Search Results into XML"
- Section 13.5, "Displaying Search Results"

For the examples in the following sections, you need to enable output in SQL*Plus. To do this log on to SQL*Plus as the portal schema owner and enter the following command:

```
SQL> SET SERVEROUTPUT ON
```

More On OTN

For more information about the public search APIs, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, and then click **PL/SQL API Reference** (APIs and References section).

> **Tip:** Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

13.1 Searching For Items Across All Page Groups

Example 13–1 uses the `item_search` API to search all the page groups in your portal for items that contain the term 'portal'.

Example 13–1 Searching Across All Page Groups (item_search API)

```
declare
  l_results    wwsrc_api.items_result_array_type;
  l_count      number;
  l_scores     wwsrc_api.number_list_type;
begin
  l_results := wwsrc_api.item_search(
    p_mainsearch => 'portal',
    p_out_count  => l_count,
    p_out_scores => l_scores
  );
  dbms_output.put_line('Number of results: ' || l_count);
exception
  ...
end;
/
```

- **p_mainsearch** is the keyword or term for which you want to search. This may be any value, including an Oracle Text query expression.
- **p_out_count** is the number of search hits.

- **p_out_scores** is an array of search results scores. This is the Oracle Text relevancy score, rating how well each result matches the search term (or any other textual search criteria). The index of the array matches the index of the results array returned by the function.

In Example 13–1, the maximum number of search results that may be returned is determined by the `wwsrc_api` constant MAX_ROWS (default is 1000). This avoids the possibility of a search query with many hits taking too long to run if a specific number of rows is not specified. For an example of how to specify the number of rows to return see Example 13–2.

13.2 Searching For Pages in Specific Page Groups

Example 13–2 uses the `page_search` API to search in two specific page groups (MyPageGroup and Shared) for pages that contain the term 'Oracle'. In this example, only the first ten results are returned.

Example 13–2 Searching in Specific Page Groups (page_search API)

```
declare
  l_results   wwsrc_api.pages_result_array_type;
  l_count     number;
  l_scores    wwsrc_api.number_list_type;
  l_pggroups  wwsrc_api.number_list_type;
begin
  l_pggroups(1) := 0;  -- Page group  0 (shared).
  l_pggroups(2) := 33; -- Page group 33 (mypagegroup).
  l_results := wwsrc_api.page_search(
    p_mainsearch  => 'Oracle',
    p_page_groups => l_pggroups,
    p_rows        => 10,
    p_out_count   => l_count,
    p_out_scores  => l_scores
  );
  dbms_output.put_line('Number of results:' || l_count);
exception
  ...
end;
/
```

- **p_mainsearch** is the keyword or term for which you want to search. This may be any value, including an Oracle Text query expression.
- **p_page_groups** is an array of the IDs of the page groups that you want to search. If you do not include this parameter, the value defaults to `wwsrc_api.EMPTY_NUMBER_LIST` (that is, all page groups).
- **p_rows** is the maximum number of search results to return. This defaults to the `wwsrc_api` constant value `MAX_ROWS`.
- **p_out_count** is the number of search hits.
- **p_out_scores** is an array of search result scores. This is the Oracle Text relevancy score, rating how well each result matches the search term (or any other textual search criteria). The index of the array matches the index of the results array returned by the function.

13.3 Searching For Items By Attribute

Example 13–3 uses the `specify_attributes` and `item_search` APIs to search for all file items created by Joe Bloggs since 01-Jan-2004 that contain the term 'Oracle'.

Example 13–3 Searching Specific Attributes (specify_attributes API)

```
declare
  l_results           wwsrc_api.items_result_array_type;
  l_count             number;
  l_scores            wwsrc_api.number_list_type;
  l_attributes        wwsrc_runtime_attr_varray;
  l_createdate_attrid wwsbr_attributes.id%type;
  l_createdate_caid   wwsbr_attributes.caid%type;
  l_createdate_type   wwsbr_attributes.data_type%type;
begin
  -- Build up attribute object with author criteria.
  wwsrc_api.specify_attributes(
    p_id                 => wwsbr_api.ATTRIBUTE_AUTHOR,
    p_siteid             => wwsbr_api.SHARED_OBJECTS,
    p_value              => 'Joe Bloggs',
    p_operator           => wwsrc_api.CONTAINS_ALL,
    p_datatype           => wwsrc_api.DATA_TYPE_TEXT,
    p_in_out_attr_varray => l_attributes
  );
  -- Build up attribute object with create date criteria
  -- using wwsbr_attributes view.
  select id, caid, data_type
  into l_createdate_attrid, l_createdate_caid, l_createdate_type
  from wwsbr_attributes
  where name = 'createdate'
  and rownum = 1; -- Ignore translations.
  wwsrc_api.specify_attributes(
    p_id                 => l_createdate_attrid,
    p_siteid             => l_createdate_caid,
    p_value              => '01-JAN-2004',
    p_operator           => wwsrc_api.GREATER_THAN,
    p_datatype           => l_createdate_type,
    p_in_out_attr_varray => l_attributes
  );
  -- Perform the search.
  l_results := wwsrc_api.item_search(
    p_mainsearch     => 'Oracle',
    p_itemtypeid     => wwsbr_api.ITEM_TYPE_FILE,
    p_itemtypesiteid => wwsbr_api.SHARED_OBJECTS,
    p_attributes     => l_attributes,
    p_out_count      => l_count,
    p_out_scores     => l_scores
  );
  dbms_output.put_line('Number of results: ' || l_count);
exception
  ...
end;
/
```

The following are parameters for `specify_attributes`:

- **p_id** is the ID of the attribute. Use the `wwsbr_api` constants or query the WWSBR_ATTRIBUTES view to find this ID.
- **p_siteid** is the ID of the page group to which the attribute belongs.

- **p_value** is the attribute value to include as search criteria. This must be a text value.
- **p_operator** is the search operator that you want to use.
- **p_datatype** is the datatype of the attribute value.
- **p_in_out_attr_varray** is the existing varray of attributes (if any).

The following are parameters for `item_search`:

- **p_mainsearch** is the keyword or term for which you want to search. This may be any value, including an Oracle Text query expression.
- **p_itemtypeid** is the ID of the type of item for which you want to search. Use the `wwsbr_api` constants or query the WWSBR_ITEM_TYPES view to find this ID. Note that this is the actual subtype of the item, not the base type.
- **p_itemtypesiteid** is the ID of the page group to which the item type belongs.
- **p_attributes** is an array of attributes to search together with their operators and values. You can build up the values for this parameter by calling the `specify_attributes` API.
- **p_out_count** is the number of search hits.
- **p_out_scores** is an array of search result scores. This is the Oracle Text relevancy score, rating how well each result matches the search term (or any other textual search criteria). The index of this array matches the index of the results array returned by the function.

13.4 Transforming Search Results into XML

The APIs mentioned so far return search results in an array. To provide you with more flexibility about how to display your search results, Oracle Portal also provides several APIs that return search results as XML. These APIs return the XML results as a CLOB.

The following sections describe how you can generate a physical XML file from a CLOB produced by the search APIs and store the XML file in a directory on the Oracle Portal middle tier:

1. Section 13.4.1, "Creating a Directory for the XML File".
2. Section 13.4.2, "Creating an XML File from a CLOB"
3. Section 13.4.3, "Generating Search Results in XML"

These examples assume that the database is on the same machine as the Oracle Portal middle tier.

13.4.1 Creating a Directory for the XML File

If you decide to transform your search results into an XML file, you must first perform some set up tasks to define the physical directory in which to write the file.

To create a directory on the Oracle Portal middle tier, perform the following steps:

1. In SQL*Plus, connect as SYSTEM or SYS and enter the following command:

   ```
   create directory <dirname> as '<physical_location>';
   ```

   For example:

   ```
   create directory RESULTDIR as '/u02/ora/OraHome_101202_
   PortalMF/Apache/Apache/htdocs/searchapi';
   ```

In this example, we use the `htdocs` directory of the Oracle Portal middle tier. This is a good location to use as you can then access the XML results file through HTTP, in the format:

```
http://<Portal_Mid_Tier>:<Port>/<directory>/<XML_Output_File>
```

For example:

```
http://my.portal.com:7778/searchapi/results.xml
```

You can then use this to specify the XML file as the data source for OmniPortlet. For an example of how to do this, refer to Section 13.5.1, "Displaying XML Search Results in OmniPortlet".

2. Still in SQL*Plus, grant write privileges to the directory from PL/SQL programs. In our example, we would do this by entering the following command:

```
grant write on directory RESULTDIR to public;
```

3. Check that you can write files to the new directory by running the following procedure:

```
declare
 2 v_output_file1 utl_file.file_type;
 3 begin
 4 v_output_file1 := utl_file.fopen('RESULTDIR', 'NEW.txt', 'a');
 5 utl_file.put_line(v_output_file1, 'NATURE and Beauty');
 6 utl_file.fclose_all;
 7 end;
 8 /

PL/SQL procedure successfully completed.
```

Now, when you go to the directory on the file system, you should see the NEW.txt file.

13.4.2 Creating an XML File from a CLOB

When you have specified the location for the XML output, you can write the procedure that creates it. Example 13–4 shows a generic procedure that takes the CLOB produced by the `get_all_items_xml` API and writes an XML file to a specific location. It takes two input parameters: the file name and the CLOB.

Example 13–4 Writing an XML File to a Specific File System Location

```
create or replace procedure results_xml_to_file(xml_filename varchar2, result_xml
clob) as
  l_amount binary_integer;
  l_offset number := 1;
  l_text   varchar2(32767);
  l_file   utl_file.file_type;
  l_clob   clob;
begin
  l_clob := result_xml;
  l_file := utl_file.fopen(
    location  => 'RESULTDIR', -- Directory name is case-sensitive.
    filename  => xml_filename,
    open_mode => 'w'
  );
  l_amount := 32767;
  l_offset := 1;
  begin
    dbms_lob.open(l_clob, dbms_lob.file_readonly);
    loop
      dbms_lob.read(l_clob, l_amount, l_offset, l_text);
      utl_file.put(
        file   => l_file,
        buffer => l_text
      );
      l_offset := l_offset + l_amount;
      exit when l_amount < 32767;
    end loop;
    utl_file.new_line(file => l_file);
    dbms_lob.close(l_clob);
  end;
  utl_file.fclose(file => l_file);
exception
  ...
end;
/
```

13.4.3 Generating Search Results in XML

Now, when you use the APIs to produce search results as XML, you can call the procedure in Example 13–4 to write the resulting CLOB to an XML file on the file system. Example 13–5 shows how you might do this.

Example 13–5 Generating Search Results in XML (get_all_items_xml API)

```
declare
  l_results   wwsrc_api.items_result_array_type;
  l_scores    wwsrc_api.number_list_type;
  l_count     number;
  l_clob      clob;
begin
  l_results := wwsrc_api.item_search(
    p_mainsearch => 'portal',
    p_out_count  => l_count,
    p_out_scores => l_scores
  );
  l_clob := wwsrc_api.get_all_items_xml(l_results);
  results_xml_to_file('results12.xml', l_clob);
exception
  ...
end;
/
```

For information about the `item_search` API used inExample 13–5, refer to Section 13.1, "Searching For Items Across All Page Groups".

You can use the XML file generated by this API as a data source for OmniPortlet. For an example of how you might do this, refer to Section 13.5.1, "Displaying XML Search Results in OmniPortlet".

13.4.4 Workaround for get_item_xml

When you use `get_item_xml` to transform search results to XML, the API produces a `<?xml version="1.0"?>` element for each search result. This produces invalid

XML. Example 13–6 shows a workaround that removes these tags and also adds opening and closing `<ResultSet>` tags to produce an XML file with the ROWSET/ROW structure. If you want to use the XML file as a data source for OmniPortlet, you should do something similar.

Note: Example 13–6 loops through the search results and writes them all to XML (producing the same results as using `get_all_items_xml`). Typically, you would use `get_item_xml` to filter the search results somehow.

Example 13–6 Using get_item_xml to Write Search Results to a File

```
declare
  l_amount    binary_integer;
  l_offset    number       := 1;
  l_text      varchar2(32767);
  l_file      utl_file.file_type;
  l_clob      clob;
  l_results   wwsrc_api.items_result_array_type;
  l_scores    wwsrc_api.number_list_type;
  l_count     number;
begin
  l_results := wwsrc_api.item_search(
    p_mainsearch => 'portal',
    p_out_count  => l_count,
    p_out_scores => l_scores
  );
  l_file := utl_file.fopen(
    location  => 'RESULTDIR', -- Directory name is case-sensitive.
    filename  => 'results14.xml',
    open_mode => 'w'
  );
  utl_file.put(
    file   => l_file,
    buffer => '<ResultSet>'
  );
    for i in 1..l_results.count loop
      l_amount := 32767;
      l_offset := 1;
    begin
      l_clob := wwsrc_api.get_item_xml(l_results(i));
      dbms_lob.open(l_clob, dbms_lob.file_readonly);
    loop
      dbms_lob.read(l_clob, l_amount, l_offset, l_text);
      -- Workaround for XML generated with get_item_xml.
      if instr(l_text, '?>') != 0 then
        l_text := substr(l_text, instr(l_text, '?>') + 2);
      end if;
      -- End of workaround.
      utl_file.put(
        file   => l_file,
        buffer => l_text
      );
      l_offset := l_offset + l_amount;
      exit when l_amount < 32767;
    end loop;
    utl_file.new_line(file => l_file);
    dbms_lob.close(l_clob);
    end;
    end loop;
    utl_file.put(
      file   => l_file,
      buffer => '</ResultSet>'
    );
    utl_file.fclose(file => l_file);
exception
  ...
end;
/
```

13.5 Displaying Search Results

For your search query to be useful, you need to display the results somewhere. You can choose to display the results of the search directly from the array, for example, in a dynamic page. For more flexibility, you can transform the search results into XML first and then display the XML, for example in OmniPortlet.

13.5.1 Displaying XML Search Results in OmniPortlet

The steps in this section provide an example of how to use the XML generated by the search APIs as the data source for OmniPortlet. For example, you could schedule a job to regularly execute an API to produce search results as XML and then automatically display those results in an OmniPortlet.

For more information about OmniPortlet, refer to Chapter 3, "Creating Portlets with OmniPortlet".

Note: The XML produced by the search APIs is returned in a CLOB. Therefore, if you want to use the XML as a data source for OmniPortlet, you must first write the CLOB to a file. For an example of how to do this, refer to Section 13.4, "Transforming Search Results into XML".

To display XML search results in OmniPortlet, perform the following steps:

1. First you need to add a new OmniPortlet instance to a page.

   Tip: OmniPortlet is usually available in the Portlet Builders area of the Portlet Repository.

2. Click the **Define** link to launch the OmniPortlet Wizard.
3. On the Type page of the wizard, select **XML**, then click **Next**.
4. On the Source page, in the **XML URL** field, enter the URL of your XML file, for example:

   ```
   http://my.portal.com:7778/searchapi/results.xml
   ```

   Because the XML data produced by the search APIs uses a ROWSET/ROW structure, you do not need to specify an XSL filter.
5. Click **Next**.
6. On the View page, select **Tabular** for the **Layout Style**, then click **Next**.

7. On the Layout page, choose the columns you want to display in the portlet and how you want them to appear.
8. Click **Finish** to create the OmniPortlet instance.

13.5.2 Displaying Search Results in a Dynamic Page

The following two examples show how to display search results in a dynamic page portlet. Users can then add this portlet to any page within your portal. To do this, perform the following steps:

1. Create a procedure that the dynamic page can call to perform the search.

   The procedure in Example 13–7 uses the `item_search` API to search for a specified term (`searchterm`) and displays only those results with a score higher than a specified value (`score`).

   Note: You must create the procedure in the database in which your portal resides.

Example 13–7 Procedure to Perform the Search

```
create or replace procedure search_results(searchterm varchar2, score number) as
  x varchar2(100);
  y number;
  l_results   wwsrc_api.items_result_array_type;
  l_count     number;
  l_scores    wwsrc_api.number_list_type;
begin
  x := searchterm;
  y := score;
  l_results := wwsrc_api.item_search(
    p_mainsearch => x,
    p_out_count  => l_count,
    p_out_scores => l_scores
  );
  htp.p('Number of total hits: ' || l_count);
  htp.p('<br>');
  htp.p('<br>');
  for i in 1..l_results.count loop
    if (l_scores(i) > y) then
      htp.p('<b>' || i || '</b> - <a href="' || l_results(i).url ||
        '">' || l_results(i).display_name || '</a>');
      htp.p('<br>');
      htp.p('score = ' || l_scores(i));
      htp.p('<br>');
    end if;
  end loop;
exception
  ...
end;
/
grant execute on search_results to public;
```

   For information about the `item_search` API used inExample 13–7, refer to Section 13.1, "Searching For Items Across All Page Groups".

2. Create the dynamic page.

A dynamic page is one of the portlets you can build using the Oracle Portal Portlet Builder. For more information about Portlet Builder, refer to the *Oracle Fusion Middleware Developer's Guide for Oracle Portal*. For more information about building dynamic pages, refer to the Oracle Portal online Help.

Example 13–8 shows the code to use for the dynamic page.

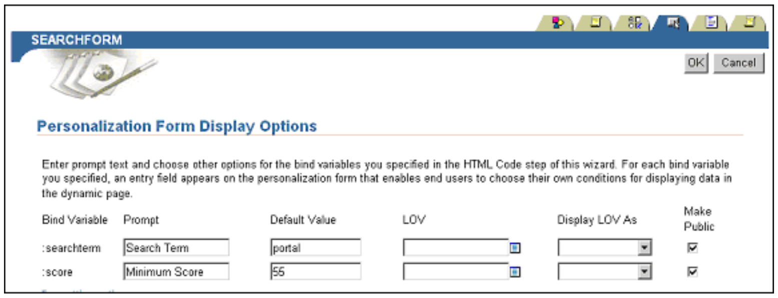
> **Tip:** Replace `<schema>` with the name of the schema in which you created your procedure.

Example 13–8 Code for Dynamic Page to Display Results

```
<html>
  <head>
    <title>Example for Search API UI</title>
  </head>
  <body>
    <h2>My Search Form</h2>
    <oracle>
      declare
        x varchar2(100);
        y number;
      begin
        x := :searchterm;
        y := :score;
        <schemaname>.search_results(x,y);
      end;
    </oracle>
  </body>
</html>
```

3. Provide default values for the two bind variables defined in the HTML code so that the dynamic page displays some results (Figure 13–1).

Figure 13–1 Providing Default Values for Bind Variables



Personalization Form Display Options

Enter prompt text and choose other options for the bind variables you specified in the HTML Code step of this wizard. For each bind variable you specified, an entry field appears on the personalization form that enables end users to choose their own conditions for displaying data in the dynamic page.

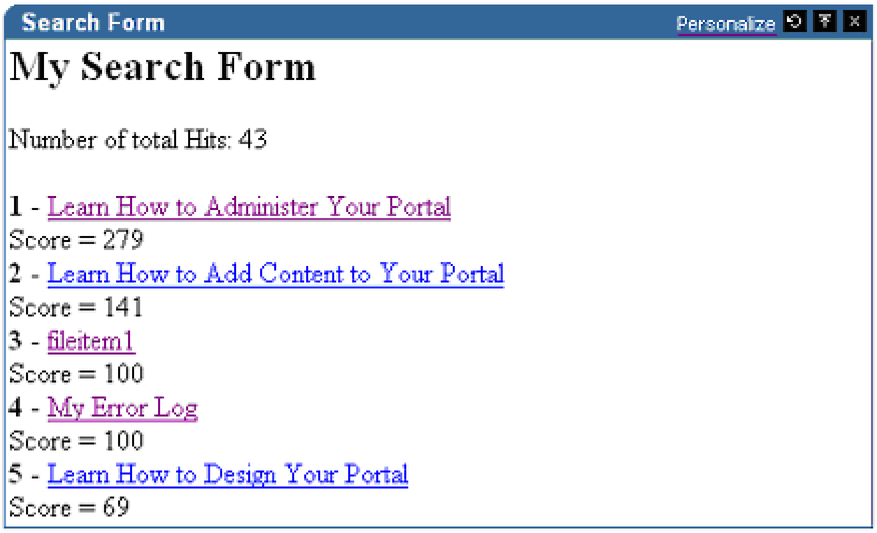
Bind Variable	Prompt	Default Value	LOV	Display LOV As	Make Public
:searchterm	Search Term	portal			✓
:score	Minimum Score	55			✓

4. Make sure the dynamic page is available as a portlet.
5. You can now add this dynamic page portlet to any page in your portal.

> **Tip:** You should find the portlet in the Staging Area of the Portlet Repository under the name of the provider in which you created it.

The dynamic page portlet displays search results based on the default values provided when you created the dynamic page, as shown in Figure 13–2.

Figure 13–2 Default Search Results Displayed in the Dynamic Page Portlet



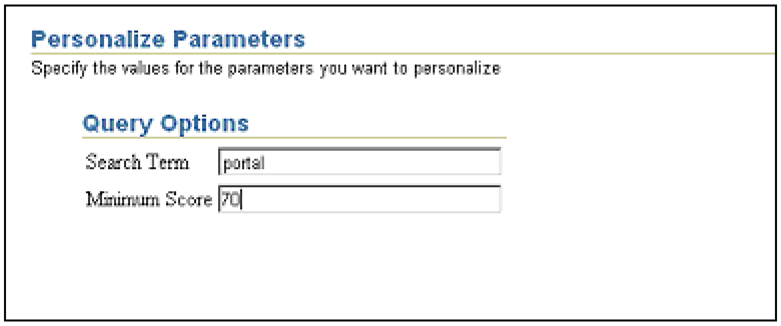
My Search Form

Number of total Hits: 43

1 - Learn How to Administer Your Portal
Score = 279
2 - Learn How to Add Content to Your Portal
Score = 141
3 - fileitem1
Score = 100
4 - My Error Log
Score = 100
5 - Learn How to Design Your Portal
Score = 69

6. Users can personalize this portlet to provide their own search term and minimum score, as shown in Figure 13–3.

Figure 13–3 Personalizing Search Criteria



Personalize Parameters

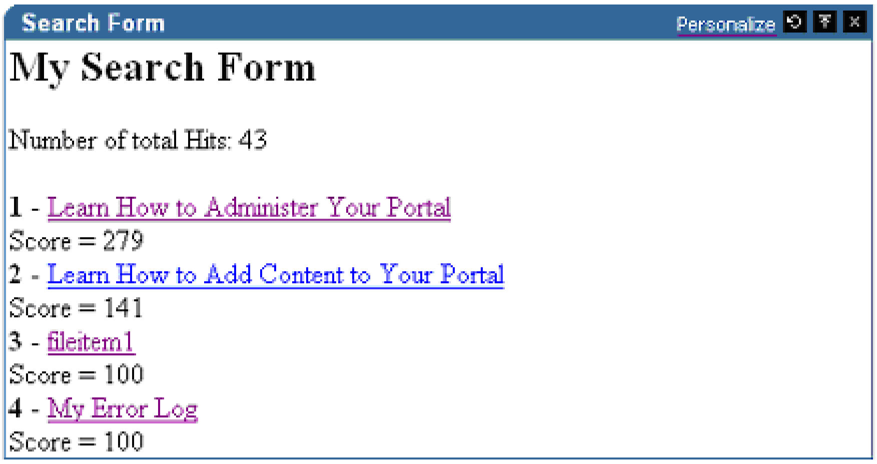
Specify the values for the parameters you want to personalize

Query Options

Search Term	portal
Minimum Score	70

In Figure 13–4 you can see the changes in the search results when a user personalizes the portlet and changes the minimum score to 70.

Figure 13–4 Personalized Search Results Displayed in the Dynamic Page Portlet



My Search Form

Number of total Hits: 43

1 - Learn How to Administer Your Portal
Score = 279
2 - Learn How to Add Content to Your Portal
Score = 141
3 - fileitem1
Score = 100
4 - My Error Log
Score = 100

14 Creating Multi-Lingual Content

This chapter describes how to use the APIs provided with Oracle Portal to create content in different languages. It contains the following sections:

- Section 14.1, "Introduction to Multi-Lingual Support"
- Section 14.2, "Querying the Default Language"
- Section 14.3, "Setting the Session Language"
- Section 14.4, "Modifying an Existing Translation"
- Section 14.5, "Creating a Translation for an Item"
- Section 14.6, "Translations and Item Versioning"

More On OTN

For more information about any of these APIs, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

> **Tip:** Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

14.1 Introduction to Multi-Lingual Support

The page group administrator must have configured the page group to support multiple languages before you can use the APIs to create translations of portal objects. Before using the APIs, it is recommended that you read Chapter 20 "Translating Portal Content" of the *Oracle Fusion Middleware User's Guide for Oracle Portal* to understand how to configure a page group and how objects with translations behave.

The basic approach to creating or modifying a translation of a portal object is to set the language context and then use the appropriate content management API to create or modify the item.

14.2 Querying the Default Language

To determine the default language of a page group, run the following query:

```
select distinct wwnls_api.get_language(default_language) "Page Group Default
Language"
from wwsbr_all_content_areas pg
where name = 'MyPageGroup';
```

To determine the language context of a SQL*Plus session:

- The SQL statement which is used to get the default language should be executed as Portal schema user. To connect to the protal schema, enter the portal schema password and the `SERVICE_NAME/SID` of the database where the portal repository is installed.
- Run the following query:

```
select wwctx_api.get_nls_language "Session Context Language Code",
       wwnls_api.get_language(wwctx_api.get_nls_language) "Language
Description"
from dual;
```

> **Note:** When you log in to SQL*Plus and set your user context, you are initially in the language specified by your NLS_LANG environment variable.

14.3 Setting the Session Language

Before creating content in a different language, use the `wwctx_api.set_nls_language` API to set the session language, as shown in Example 14–1.

Example 14–1 Setting the Session Language (set_nls_language API)

```
begin
  wwctx_api.set_nls_language(
    p_nls_language => wwnls_api.GREEK,
    p_nls_territory => wwnls_api.TER_GREECE
  );
exception
  ...
end;
/
```

- **p_nls_language** is the Globalization Support language abbreviation of the language that you want to set for the session.
- **p_nls_territory** is the Globalization Support territory abbreviation of the territory that you want to set for the session.

More On OTN

For a list of available language and territory constants, refer to the documentation for the WWNLS_API package in the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

14.4 Modifying an Existing Translation

Modifying an existing translation is simply a matter of setting the language context and using the appropriate API to modify the object. Example 14–2 modifies the text of an existing text item that has a translation in French.

Example 14–2 Modifying a Translation

```
declare
```

```
  l_item_master_id number;
begin
  -- Set the language context.
  wwctx_api.set_nls_language(
    p_nls_language => wwnls_api.FRENCH
  );
  -- Edit the item (this edits the French translation).
  l_item_master_id := wwsbr_api.modify_item(
    p_master_item_id => 453,
    p_item_id        => 454,
    p_caid           => 33,
    p_folder_id      => 45,
    p_display_name   => 'Revue de Film');
    p_text           => 'C'est le texte de la revue');
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

For information about the `modify_item` API used in Example 14–2, refer to Section 11.2.2, "Editing an Item".

14.5 Creating a Translation for an Item

If the session is using a different language from the default language for the page group *and* translations for the session language are enabled for the page group, the `add_item` API creates a translation for an item when the item is created. For example, if the default language for the page group is English, the session language is French, and French translations are enabled for the page group, the API adds the following two rows to the item table:

- One with a value of 'us' (English) for the default language.
- One with a value of 'f' (French) for the session language.

Other than the language, both rows will have identical values for all attributes, including the item ID. To avoid this, it is recommended that you always add an item in the default language first before creating any of its translations.

In similar circumstances, the `modify_item` API updates the translation for the session language (if it exists). If the translation does not exist, as long as translations are enabled for the session language, the API creates a new translation. Therefore, if you want to add a translation for an existing item, use the `modify_item` (or `modify_item_post_upload`) API. Example 14–3 creates a file item then changes the language context and adds a translation for the item.

Example 14–3 Creating a Translation

```
declare
  l_item_masterthing_id  number;
  l_item_masterthing_id2 number;
  l_item_id              number;
begin
  -- Set the language context.
  wwctx_api.set_nls_language(
    p_nls_language => wwnls_api.AMERICAN
  );
  -- Add an item that resides on the same server as the portal content repository
  -- database.
  l_item_masterthing_id := wwsbr_api.add_item(
    p_caid          => 53,  - A known page group ID.
    p_folder_id     => 1,   - A known page ID.
    p_display_name  => 'My File',
    p_type_id       => wwsbr_api.ITEM_TYPE_FILE,
    p_type_caid     => wwsbr_api.SHARED_OBJECTS,
    p_region_id     => 513, - A known region on the page.
    p_description   => 'Description of my file',
    p_file_filename => '/docs_for_upload/English_File.txt'
  );
  -- Note that if the default language was not American then OracleAS Portal
  -- would in addition automatically create a translation of the item in the
  -- default language.
  --
  -- Determine the item id.
  select id
  into l_item_id
  from wwsbr_all_items
  where masterid = l_item_masterthing_id
    and caid = l_site_id
    and language = wwctx_api.get_nls_language
    and is_current_version = 1;
  -- Change the language context to French.
  wwctx_api.set_nls_language(
    p_nls_language => wwnls_api.FRENCH
  );
  -- Modify item adding its translation and any translated attributes.
  l_item_masterthing_id2 := wwsbr_api.modify_item(
    p_master_item_id => l_item_masterthing_id,
    p_item_id        => l_item_id,
    p_caid           => l_site_id,
    p_folder_id      => l_page_id,
    p_display_name   => 'Mon Fichier',
    p_description    => 'Description du fichier',
    p_file_filename  => '/docs_for_upload/French_File.txt'
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

For information about the add_item API used in Example 14–3, refer to Section 12.4, "Creating Items". For information about the modify_item API, refer to Section 11.2.2, "Editing an Item".

14.6 Translations and Item Versioning

If you attempt to create a new version of an item in a nondefault language, by setting `p_addnewversion` to TRUE using `modify_item` API, a new version is not created; a new translation is added instead.

When using versioning in conjunction with translations, we recommend that you create new versions of items in the default language and then translate the new version of the item.

For a more detailed discussion of the effect of item versioning on translations, refer to Section 20.3 "Creating Translatable Content" in the *Oracle Fusion Middleware User's Guide for Oracle Portal*.

15 Implementing Content Security

This chapter describes how to use the APIs provided with Oracle Portal to ensure that your content is secure. It contains the following sections:

- Section 15.1, "Retrieving Object Privileges"
- Section 15.2, "Setting Page Level Privileges"
- Section 15.3, "Setting Item Level Privileges"

More On OTN

For more information about any of these APIs, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

> **Tip:** Remember, if you are calling the APIs from a Web provider or external application, you need to set the session context first. For more information, refer to Section 10.1, "Setting the Session Context".

15.1 Retrieving Object Privileges

To retrieve a list of privileges that are currently defined for an object, use the `wwsec_api.grantee_list` API. Example 15–1 prints the values of the grantee array for a page with an ID of 17623 in page group with an ID of 33. Example 15–2 prints the values of the grantee array for an item with a master ID of 32919 in a page group with an ID of 53.

Example 15–1 Retrieving the List of Privileges for a Page (grantee_list API)

```
declare
  l_grantees         wwsec_api.grantee_array;
  l_object_type_name varchar2(5)  := wwsec_api.PAGE_OBJ;
  l_name             varchar2(60) := '33/17623';
begin
  -- Call the function.
  l_grantees := wwsec_api.grantee_list(
    p_object_type_name => l_object_type_name,
    p_name             => l_name
  );
  -- Output the results
  if l_grantees is not null then
    if l_grantees.count > 0 then
      for i in l_grantees.first..l_grantees.last loop
        if l_grantees.exists(i) then
```

```
          dbms_output.put_line('GRANTEE_TYPE '||to_char(i)||'=
'||l_grantees(i).GRANTEE_TYPE);
          dbms_output.put_line('GRANTEE_ID '||to_char(i)||'=
'||l_grantees(i).GRANTEE_ID);
          dbms_output.put_line('GRANTEE_NAME '||to_char(i)||'=
'||l_grantees(i).GRANTEE_NAME);
          dbms_output.put_line('PRIVILEGE '||to_char(i)||'=
'||l_grantees(i).PRIVILEGE);
        end if;
      end loop;
    end if;
  end if;
exception
  ...
end;
/
```

Example 15–2   Retrieving the List of Privileges for an Item

```
declare
  l_grantees          wwsec_api.grantee_array;
  p_object_type_name varchar2(5)  := wwsec_api.ITEM_OBJ;
  p_name             varchar2(60) := '53/32919';
begin
  -- Call the function.
  l_grantees := wwsec_api.grantee_list(p_object_type_name, p_name);
  -- Output the results.
  if l_grantees is not null then
    if l_grantees.count > 0 then
      for i in l_grantees.first..l_grantees.last loop
        if l_grantees.exists(i) then
          dbms_output.put.line('GRANTEE_TYPE '||to_char(i)||'= '||l_
grantees(i).GRANTEE_TYPE);
          dbms_output.put.line('GRANTEE_ID '||to_char(i)||'= '||l_
grantees(i).GRANTEE_ID);
          dbms_output.put.line('GRANTEE_NAME '||to_char(i)||'= '||l_
grantees(i).GRANTEE_NAME);
          dbms_output.put.line('PRIVILEGE '||to_char(i)||'= '||l_
grantees(i).PRIVILEGE);
        end if;
      end loop;
    end if;
  end if;
exception
  ...
end;
/
```

The `grantee_list` API takes the following three parameters:

- **p_object_type_name** is the type of the object. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.PAGE_OBJ or wwsec_api.ITEM_OBJ.
- **p_name** is the reference to the object. Use the format `'<page group ID>/<object ID>'`. So for items, use `'<page group ID>/<master item ID>'`, for example `'53/32919'`.
- **p_owner** is the name of the schema that owns the object. For items, do not pass a value to this parameter as it defaults to the portal schema owner.

The API returns an array (WWSEC_API.GRANTEE_ARRAY) with the following columns:

- **grantee_type** is either USER or GROUP
- **grantee_id** is the unique ID of the user or group
- **grantee_name** is the user name or group name
- **privilege** is the privilege granted to the user or group

15.2 Setting Page Level Privileges

This section shows how to use APIs in the `WWSEC_API` package to set page level privileges.

Note: You can also use the APIs listed in the following sections to set tab level access by using the following format for the `p_name` parameter:

```
<page group ID>/<tab ID>
```

You do not need to specify the ID of the container page, as the tab ID is enough to uniquely identify the tab within the page group.

15.2.1 Granting Page Level Privileges

Example 15–3 shows how to use the `set_group_acl` API to grant privileges to a group. Example 15–4 shows how to use the `set_user_acl` API to grant privileges to a user.

Example 15–3 Granting Page Privileges to a Group (set_group_acl API)

```
declare
  l_group_id number       := wwsec_api.group_id('MYGROUP');
  l_name     varchar2(60) := '33/17623';
BEGIN
  wwsec_api.set_group_acl(
    p_group_id         => l_group_id,
    p_object_type_name => wwsec_api.PAGE_OBJ,
    p_name             => l_name,
    p_privilege        => wwsec_api.VIEW_PRIV
  );
end;
/
```

Example 15–4 Granting Page Privileges to a User (set_user_acl API)

```
declare
  l_person_id number      := wwsec_api.id('JOHN.SMITH');
  l_name     varchar2(60) := '33/17623';
begin
  wwsec_api.set_user_acl(
    p_person_id        => l_person_id,
    p_object_type_name => wwsec_api.PAGE_OBJ,
    p_name             => l_name,
    p_privilege        => wwsec_api.VIEW_PRIV
  );
```

```
end;
/
```

These two APIs take the following parameters:

- **p_group_id** is the ID of the group to which you want to grant privileges (`set_group_acl` only)
- **p_person_id** is the ID of the user to whom you want to grant privileges (`set_user_acl` only)
- **p_object_type_name** is type of the object on which you want to grant privileges. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.PAGE_OBJ.
- **p_name** is the reference to the object. Use the format `'<page group ID>/<page ID>'`, for example `'33/17623'`.
- **p_privilege** is the level of privilege you want to grant to the user or group. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.VIEW_PRIV.

15.2.2 Removing Page Level Privileges

At some point, it may become necessary to remove a user or group's privileges from a page. Example 15–5 shows how to use the `remove_group_acl` API to remove a group's privileges. Example 15–6 shows how to use the `remove_user_acl` API to remove a user's privileges.

Example 15–5 Removing Page Privileges from a Group (remove_group_acl API)

```
declare
  l_group_id number       := wwsec_api.group_id('MYGROUP');
  l_name     varchar2(60) := '33/17623';
BEGIN
  wwsec_api.remove_group_acl(
    p_object_type_name => wwsec_api.PAGE_OBJ,
    p_name              => l_name,
    p_group_id         => l_group_id,
    p_privilege         => wwsec_api.MANAGE_PRIV
  );
end;
/
```

Example 15–6 Removing Page Privileges from a User (remove_user_acl API)

```
declare
  l_person_id number       := wwsec_api.id('JOHN.SMITH');
  l_name     varchar2(60)  := '33/17623';
BEGIN
  wwsec_api.remove_user_acl(
    p_object_type_name => wwsec_api.GROUP_OBJ,
    p_name             => l_name,
    p_person_id        => l_person_id,
    p_privilege        => wwsec_api.MANAGE_PRIV
  );
end;
/
```

These two APIs take the following parameters:

- **p_object_type_name** is type of the object from which you want to remove privileges. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.PAGE_OBJ.
- **p_name** is the reference to the object. Use the format `'<page group ID>/<page ID>'`, for example `'33/17623'`.
- **p_group_id** is the ID of the group whose privileges you want to remove (`remove_group_acl` only). Set this parameter to NULL if you want to remove the specified privilege on this page from all groups.
- **p_person_id** is the ID of the user whose privileges you want to remove (`remove_user_acl` only). Set this parameter to NULL if you want to remove the specified privilege on this page from all users.
- **p_privilege** is the level of privilege you want to remove from the user or group. Use the predefined constants in the WWSEC_API package to specify the value of this parameter, for example wwsec_api.VIEW_PRIV. Set this parameter to NULL if you want to remove all privileges on the page from the user or group.

15.3 Setting Item Level Privileges

If item level security (ILS) is enabled for a page, you can specify access privileges for individual items on the page.

Example 15–7 shows how to use the `modify_folder` API to enable ILS for a page.

Example 15–7 Enabling Item Level Security for a Page

```
declare
  l_page     wwsbr_api.page_record;
begin
  select *
  into l_page
  from   <schema>.wwsbr_user_pages
  where  siteid = 33
    and  id = 1
    and  rownum = 1;
  l_page.haveitemsecurity := 1;
  wwsbr_api.modify_folder(
    p_page => l_page
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

For information about the `modify_folder` API and WWSBR_USER_PAGES view used in Example 15–7, refer to Section 11.1, "Editing Page Properties".

Example 15–8 shows how to enable ILS for an individual item (this is the same as selecting **Define Item Level Access Privileges** in the Oracle Portal user interface):

Example 15–8 Enabling Item Level Security for an Item (enable_ils_for_item API)

```
begin
  wwsbr_api.enable_ils_for_item(
    p_master_item_id => 453,
    p_caid           => 33,
    p_folder_id      => 45
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_master_item_id** is the master ID of the item. You can find this value in the MASTERID column of the WWSBR_ALL_ITEMS view.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_folder_id** is the ID of the page on which the item appears.

> **Tip:** If you want to edit other attributes for the item as well as the ILS setting, you can use the `modify_item` or `modify_item_post_upload` APIs instead. To enable ILS set the `p_access_level` parameter to wwsbr_api.ITEM_ACCESS, to disable ILS set the parameter to wwsbr_api.FOLDER_ACCESS.

15.3.1 Granting Item Level Privileges

After enabling ILS for the item, you can define access privileges for one or more users or groups.

When setting item level privileges, the type of privileges that are granted is dependent on which of the following parameters are passed rather than the parameter values:

- Pass an array of user IDs to `p_itemown_user` to grant the *Manage* privilege to a list of users.
- Pass an array of user IDs to `p_itemmanage_user` to grant the *Edit* privilege to a list of users.
- Pass an array of user IDs to `p_itemview_user` to grant the *View* privilege to a list of users.
- Pass an array of group IDs to `p_itemown_group` to grant the *Manage* privilege to a list of groups.
- Pass an array of group IDs to `p_itemmanage_group` to grant the *Edit* privilege to a list of groups.
- Pass an array of group IDs to `p_itemview_group` to grant the *View* privilege to a list of groups.

You can pass values to any combination of these parameters in the same procedure call to set a range of privileges across different users and groups.

Example 15–9 shows how you can use the `add_item_ils_privileges` API to grant item-level privileges to users.

Example 15–9 Granting Item Level Privileges to Users (add_item_ils_privileges API)

```
declare
  l_itemown_username_array wwsbr_type.array;
  l_itemown_userid_array   wwsbr_type.array;
begin
  l_itemown_username_array(1) := 'jsmith';
  l_itemown_username_array(2) := 'janesmith';
  l_itemown_username_array(3) := 'joedoe';
  for i in 1 .. l_itemown_username_array.count loop
    -- Get the user ID from the wwsec_api.id_sso API.
    l_itemown_userid_array(i) := wwsec_api.id_sso(
      p_username => l_itemown_username_array(i)
    );
  end loop;
  wwsbr_api.add_item_ils_privileges(
    p_master_item_id => 453,
    p_caid           => 33,
    p_folder_id      => 45,
    p_itemown_user   => l_itemown_userid_array
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_master_item_id** is the master ID of the item. You can find this value in the MASTERID column of the WWSBR_ALL_ITEMS view.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_folder_id** is the ID of the page on which the item appears.
- **p_itemown_user** is an array of user IDs to which you want to grant *Manage* privileges.

If you pass the same user or group in more than one of the privilege arrays, the user or group is granted the highest privilege level specified.

> **Tip:** You can also update the access privileges for an item using the `add_item_ils_privileges` API.

15.3.2 Removing Item Level Privileges

If for some reason, you need to remove a user or group's privileges on an item, use the `delete_ils_privilege` API, as shown in Example 15–10.

Example 15–10 Removing Item Level Privileges (delete_ils_privilege API)

```
declare
  l_user_id       number := 334;
  l_page_group_id number := 75;
  l_page_id       number := 1;
  l_item_id       number := 74637;
begin
  wwsbr_api.delete_ils_privilege(
    p_user_or_group_id => l_user_id,
    p_caid             => l_page_group_id,
    p_folder_id        => l_page_id,
    p_master_item_id   => l_item_id
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_user_or_group_id** is the ID of the user or group for which you want to remove privileges.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_folder_id** is the ID of the page on which the item appears.
- **p_master_item_id** is the master ID of the item. You can find this value in the MASTERID column of the WWSBR_ALL_ITEMS view.

15.3.3 Inheriting Item Level Privileges from the Page

If you decide that an item that has its own privileges defined should, in fact, simply inherit its privileges from the page instead, use the `inherit_folder_privileges` API to disable ILS (this is the same as selecting **Inherit Parent Page Access Privileges** in the Oracle Portal user interface). Example 15–11 shows how to use the `inherit_folder_privileges` API.

Example 15–11 Inheriting Item Privileges from the Parent Page (inherit_folder_privileges API)

```
begin
  wwsbr_api.inherit_folder_privileges(
    p_master_item_id => 453,
    p_caid           => 33,
    p_folder_id      => 45
  );
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  ...
end;
/
```

- **p_master_item_id** is the master ID of the item. You can find this value in the MASTERID column of the WWSBR_ALL_ITEMS view.
- **p_caid** is the ID of the page group to which the item belongs.
- **p_folder_id** is the ID of the page on which the item appears.

> **Tip:** If you want to edit other attributes for the item as well as the ILS setting, you can use the `modify_item` or `modify_item_post_upload` APIs instead. To enable ILS set the `p_access_level` parameter to wwsbr_api.ITEM_ACCESS, to disable ILS set the parameter to wwsbr_api.FOLDER_ACCESS.

16

Using the Content Management Event Framework

This chapter describes the Oracle Portal Content Management Event Framework (CMEF). It contains the following sections:

16.1 What Is the Content Management Event Framework?

CMEF enables you to extend Oracle Portal's content management functionality by adding programmatic hooks to certain pre-defined portal events. The framework publishes these events to an Oracle database queue. This allows third party programs to subscribe to these events and to use the APIs to extend your portal. In this way, you can use page and item related events within a portal to trigger actions within one or more external applications.

16.2 How Does the Content Management Event Framework Work?

CMEF uses Oracle Streams Advanced Queuing (AQ) technology. Oracle Streams AQ is an Oracle database component that provides a message-queue system with a rich and industry standard feature set. Oracle Streams AQ offers the following features:

- Multiple ways for applications (producers) to place messages in a queue (enqueue).
- Multiple ways for applications (consumers) to get messages from a queue (dequeue).
- A publish/subscribe model that enables the producer application to be independent of the consumer applications.
- Propagation of messages between queues on different machines and databases.

- Guaranteed delivery of messages along with exception handling in case messages cannot be delivered.
- Persistent storage of messages.
- Message prioritization.
- Time properties for messages such as expiration and delays.

More On OTN

For more information, refer to the Oracle Streams Advanced Queuing page on the Oracle Technology Network (OTN):

`http://www.oracle.com/technology/products/aq`

Actions within a portal (through the Oracle Portal user interface, the Web-based Distributed Authoring and Versioning (WebDAV) protocol, or PL/SQL APIs) trigger CMEF events. These events cause Oracle Portal to publish CMEF messages to a queue. You can create subscribers to consume the messages on the queue and perform actions based on them. For example, you could create a subscriber to verify that when a user adds an item to a portal page, its display name is less than 80 characters. The subscriber may process events as they occur or process them based on some time interval.

There are three major types of CMEF events: ADD, UPDATE, and DELETE. Every content management action falls under one of these events. An event can be in one of several different states. For example, when a user adds an item to a page, the state of the ADD event indicates whether the item is available immediately (PUBLISHED) or at some later date (NOT_PUBLISHED). Refer to Section 16.3, "Using the Content Management Event Framework" to learn more about how subscribers can use this information to retrieve messages of interest to them.

Using the events published by CMEF involves the following five basic queuing operations:

- **Enqueuing Messages:** Oracle Portal publishes messages (known as enqueuing) to the multiconsumer queue named WWSBR_EVENT_Q. Messages to this queue never expire. For more information, refer to Section 16.2.1, "Enqueuing Messages".
- **Dequeuing Messages:** Subscribing applications pick up messages (known as dequeuing) from the queue. For more information, refer to Section 16.2.2, "Subscribers and Dequeuing Messages".
- **Exception Handling:** A message is said to be processed normally if it is consumed within the specified time interval and within the specified number of attempts. Messages not consumed normally are placed in a separate queue called the exception queue. For more information, refer to Section 16.2.3, "Exception Handling".
- **Listening for Messages:** An application can use LISTEN to wait for messages for multiple subscriptions without having to repeatedly poll the queue. For more information, refer to Section 16.2.4, "Listening for Messages".
- **Notifications:** This Oracle Streams AQ feature enables users or clients to receive notification of a message of interest. For more information, refer to the Oracle Streams AQ documentation.

You use the DBMS_AQ package to perform queuing operations.

16.2.1 Enqueuing Messages

CMEF enqueues messages to the WWSBR_EVENT_Q queue, specifying AQ message properties that subscriber applications can then use at dequeue time (Table 16–1).

Table 16–1 AQ Message Properties Set by CMEF

Message Property	Type	Value
PRIORITY	BINARY_INTEGER	1
DELAY	BINARY_INTEGER	NO_DELAY
EXPIRATION	BINARY_INTEGER	NEVER
CORRELATION	VARCHAR2(128)	NULL
RECIPIENT_LIST	AQ$_RECIPIENT_LIST_T	NULL
EXCEPTION_QUEUE	VARCHAR2(51)	WWSBR_EVENT_ERR_Q
ORIGINAL_MSGID	RAW(16)	NULL

The WWSBR_EVENT_Q queue is a multiconsumer queue. This enables more than one subscriber to consume a single message, allowing for multiple subscribers without having to make multiple copies of each message. Since all messages in the queue have the same priority, this queue works on a first-in first-out basis.

Each message that CMEF enqueues to the WWSBR_EVENT_Q queue contains a payload that contains information about the portal object to which the event relates, such as page ID or page group ID. Subscribers can use this payload information to perform actions on the portal object. The contents of the CMEF message payload are described in more detail in Section 16.3.6, "CMEF Message Payload".

CMEF enqueues messages for immediate consumption, that is, a message is posted on the WWSBR_EVENT_Q queue as soon as an action occurs in the portal; there is no delay. Messages created by CMEF do not have an expiration time.

16.2.2 Subscribers and Dequeuing Messages

On the consuming end, subscribers have various mechanisms to consume the messages produced by CMEF. Subscribers can process messages as they arrive, and thus must wait for the messages to arrive. Alternatively, subscribers can choose to be notified when the messages arrive. These notifications can be Oracle Call Interface (OCI) callback functions, PL/SQL functions, or even e-mails.

16.2.2.1 Adding a Subscriber to the WWSBR_EVENT_Q Queue

For a subscriber to be able to consume messages produced by CMEF, you need to add it to the WWSBR_EVENT_Q queue using the DBMS_AQADM.ADD_SUBSCRIBER procedure and you need to login as the Portal schema user:

```
GRANT EXECUTE ON DBMS_AQADM TO <<PORTAL_SCHEMA>>;
GRANT Aq_administrator_role TO <<PORTAL_SCHEMA>>;
```

Example 16–1 adds the JAY subscriber to the WWSBR_EVENT_Q queue.

Example 16–1 Adding a Subscriber to WWSBR_EVENT_Q

```
subscriber := sys.aq$_agent('JAY', null, null);
dbms_aqadm.add_subscriber(
  queue_name => 'portal.wwsbr_event_q',
  subscriber => subscriber
);
```

Since Oracle Streams AQ supports a maximum of 1024 subscribers for each multiconsumer queue, you can add up to a maximum of 1024 subscribers to the

WWSBR_EVENT_Q queue. All consumers that are added as subscribers to this queue must have unique values for the AQ$_AGENT parameter.

You can remove a subscriber using the DBMS_AQADM.REMOVE_SUBSCRIBER procedure.

For an example of a simple subscriber, refer to Section 16.3.1, "Creating Subscriber Code".

16.2.2.2 Subscriber Queue Management

Oracle Enterprise Manager DBA Studio enables you to manage Oracle Streams AQ. You can use DBA Studio to create queue tables, create queues, browse AQ messages, archive or purge AQ messages, add AQ subscribers, and manage propagation. DBA Studio also shows the topology for the propagation of messages between queues at database level and queue level.

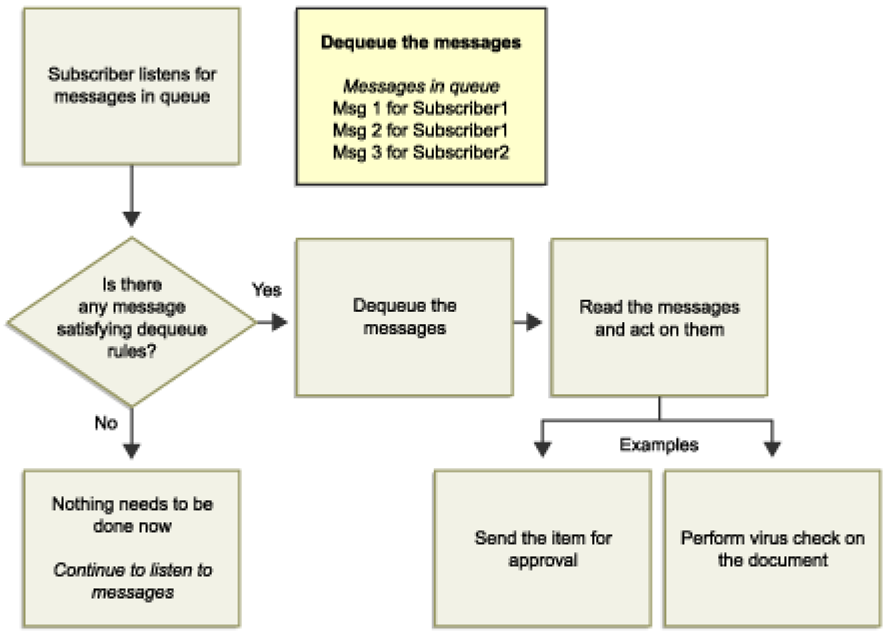
The Oracle Diagnostics and Tuning pack supports alerts and monitoring for AQ queues. You can set up alerts for when the number of messages for a particular subscriber exceeds a threshold, or when there is an error in propagation. In addition, you can monitor queues for the number of messages in a ready state or the number of messages for each subscriber, and so on.

You can also manage the subscriber queue using the standard AQ APIs. For more information, refer to your Oracle Enterprise Manager documentation.

16.2.2.3 Dequeuing Messages

The operation of retrieving messages from a queue is known as dequeuing (Figure 16–1).

Figure 16–1 The Dequeuing Process


Subscriber listens for messages in queue
Dequeue the messages
Messages in queue
Msg 1 for Subscriber1
Msg 2 for Subscriber1
Msg 3 for Subscriber2
Is there any message satisfying dequeue rules?
Yes
Dequeue the messages
Read the messages and act on them
No
Examples
Nothing needs to be done now
Continue to listen to messages
Send the item for approval
Perform virus check on the document

You use the DBMS_AQ.DEQUEUE procedure for dequeuing messages from the WWSBR_EVENT_Q queue. Example 16–2 illustrates dequeuing for the subscriber JAY.

Example 16–2 Dequeuing Messages

```
...
dequeue_options.wait          := dbms_aq.NO_WAIT;
dequeue_options.consumer_name := 'JAY';
dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
dequeue_options.dequeue_mode  := dbms_aq.BROWSE;
dbms_aq.dequeue(
  queue_name         => 'WWSBR_EVENT_Q',
  dequeue_options    => dequeue_options,
  message_properties => message_properties,
  payload            => message,
  msgid              => message_handle
);
...
```

- **NAVIGATION:** Use the NAVIGATION parameter of the DBMS_AQ.DEQUEUE operation to determine the sequence in which you want to dequeue the messages. The default NAVIGATION parameter for the dequeue request is NEXT_MESSAGE. This means that the subsequent dequeue operation will retrieve the messages from the queue based on the snapshot obtained in the first dequeue. In particular, a message that is enqueued after the dequeue command will be processed only after processing all the messages already enqueued before in the queue. This is sufficient for messages enqueued for the WWSBR_EVENT_Q queue since it does not have priority-based ordering.

  Note: NEXT_MESSAGE with some delay is the optimal way of processing AQ messages. When the first message in the queue needs to be processed by every dequeue command, subscribers must explicitly use the FIRST_MESSAGE navigation option.

- **DEQUEUE_MODE:** A dequeue request can either view a message or delete a message. To view a message, the subscriber can use the BROWSE or LOCK modes. To consume a message, the subscriber can use the REMOVE or REMOVE_NODATA modes. If a subscriber browses a message, the message remains available for further processing. Similarly a locked message remains available for further processing after the subscriber releases the lock by performing a transaction commit or rollback. To prevent a viewed message from being dequeued by a concurrent user, you should view the message in locked mode. After a subscriber consumes a message using either of the REMOVE modes, the message is no longer available for dequeue requests.

  When a subscriber dequeues a message using REMOVE_NODATA mode, the request does not retrieve the payload of the message. This mode is useful when the user has already examined the message payload, possibly by means of a previous BROWSE dequeue. In this way, you can avoid the overhead of payload retrieval that can be substantial for large payloads.

  Note: One event is enqueued for each subscriber. Thus removing an event from one subscriber's queue does not remove it from the queues of other subscribers.

- **CONSUMER_NAME:** A subscriber can dequeue a message from the WWSBR_EVENT_Q queue by supplying this queue name:

– In PL/SQL you supply the consumer name using the CONSUMER_NAME field in the DEQUEUE_OPTIONS_T record.

– In OCI you supply the consumer name using the OCISetAttr procedure to specify a text string as the OCI_ATTR_CONSUMER_NAME of an OCI_DTYPE_AQDEQ_OPTIONS descriptor.

– In Visual Basic (OO4O) you supply the consumer name by setting the consumer property of the OraAQ object.

Multiple processes or operating system threads can use the same to dequeue concurrently from a queue. Unless the message ID of a specific message is specified during dequeue, the consumers can dequeue messages that are in the READY state.

Note: You should not need to use the Search parameters to dequeue CMEF events.

16.2.3 Exception Handling

A message is considered processed when all intended consumers have successfully dequeued the message. If a message cannot be processed for some reason, it moves to an exception queue.

A message is considered expired if one or more consumers does not dequeue it before the expiration time. Expired messages also move to an exception queue. An exception queue is a repository for all expired or unserviceable messages.

Applications cannot directly enqueue into exception queues. Also, an exception queue cannot have subscribers associated with it. However, an application that intends to handle these expired or unserviceable messages must dequeue from the exception queue.

CMEF exceptions are sent to the WWSBR_EVENT_ERR_Q exception queue. Expired messages from the WWSBR_EVENT_Q multiconsumer queue cannot be dequeued by the intended recipients of the message. However, they can be dequeued in REMOVE mode once by specifying a NULL consumer name in the dequeue options. The queue monitor removes expired messages from multiconsumer queues. This allows dequeuers to complete the dequeue operation by not locking the message in the queue table.

Since the queue monitor removes messages that have been processed by all consumers from multiconsumer queues at regular intervals, users may see a delay between when the messages have been completely processed and when they are physically removed from the queue.

Note: The WWSBR_EVENT_ERR_Q exception queue, like all exception queues, is a single-consumer queue.

16.2.4 Listening for Messages

Oracle Streams AQ can monitor multiple queues for messages with a single LISTEN call. A subscriber can use LISTEN to wait for messages for multiple subscriptions. It can also be used by gateway applications to monitor multiple queues. If the LISTEN call returns successfully, a dequeue must be used to retrieve the message. Without the LISTEN call, an application which sought to dequeue from a set of queues would have to continuously poll the WWSBR_EVENT_Q queue to determine if there is a message.

Alternatively, you could design your subscriber to have a separate dequeue process for each queue. However, if there are long periods with no traffic in any of the queues, including WWSBR_EVENT_Q, these approaches will create unacceptable overhead. The LISTEN call is well suited for such subscribers. When there are messages for multiple agents in the agent list, LISTEN returns with the first agent for whom there is a message.

You can use the LISTEN call to monitor receipt of messages on one or more queues on behalf of a list of agents. The call takes a list of agents as an argument. You specify the queue to be monitored in the address field of each agent listed. You also must specify the name of the agent when monitoring multiconsumer queues. Example 16–3 shows how to use the LISTEN call to listen to messages on multiple queues.

Example 16–3 Listening to Messages on Multiple Queues

```
declare
  agent_w_msg   aq$agent;
  qlist         dbms_aq.agent_list_t;
begin
  -- MYQ1, MYQ2, MYQ3 are multiconsumer queues in the SCOTT schema.
  qlist(1) := aq$agent('agent1', 'scott.MYQ1', null);
  qlist(2) := aq$agent(null, 'scott.MYQ2', null);
  qlist(3) := aq$agent('agent3', 'scott.MYQ3', null);
  -- Listen with a timeout of 100 seconds.
  dbms_aq.listen(
    agent_list => qlist,
    wait       => 100,
    agent      => agent_w_msg
  );
  dbms_output.put_line('MSG in Q: '||agent_w_msg.address||'for '
    ||agent_w_msg.name);
  dbms_output.put_line('');
end;
/
```

This is a blocking call that returns when there is a message ready for consumption for an agent in the list. If there are messages for more than one agent, only the first agent listed is returned. If there are no messages found when the wait time expires, an error is raised.

A successful return from the call is only an indication that there is a message for one of the listed agents in one of the specified queues. The interested agent should dequeue the relevant message. Example 16–4 illustrates the dequeue process combined with listening. Here, we dequeue the messages for the subscriber, JAY, for a certain time period.

Example 16–4 Listening and Dequeuing Messages

```
begin
  agent_list(1)     := sys.aq$_agent('JAY', 'WWSBR_EVENT_Q', null);
  wait_time integer := 100;
  loop
    -- Wait for order status messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait       => wait_time,
      agent      => agent_w_message
    );
    -- If there are messages for JAY, dequeue them.
    if (agent_w_message.name = 'JAY') then
```

```
      dequeue_options.wait          := dbms_aq.NO_WAIT;
      dequeue_options.consumer_name := 'JAY';
      dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
      dequeue_options.dequeue_mode  := dbms_aq.BROWSE;
      dbms_aq.dequeue(
        queue_name         => 'item_queue',
        dequeue_options    => dequeue_options,
        message_properties => message_properties,
        payload            => message,
        msgid              => message_handle
      );
    end if;
  end loop;
exception
  when NO_MESSAGES then
    dbms_output.put_line('No more messages for Jay');
end;
/
```

16.3 Using the Content Management Event Framework

Every portal item and page action generates a CMEF message that is enqueued to the WWSBR_EVENT_Q queue. A subscriber can use the information contained within this message to perform various actions using the Oracle Portal PL/SQL APIs.

There are three basic steps in handling CMEF events, each of which is described and illustrated later in this section. These steps are as follows:

1. Section 16.3.1, "Creating Subscriber Code"
2. Section 16.3.2, "Adding a Subscriber to WWSBR_EVENT_Q"
3. Section 16.3.3, "Enabling CMEF Events at the Page Group Level"

This section also provides a description of the message payload, followed by several examples of common portal actions and the events they generate.

16.3.1 Creating Subscriber Code

Oracle Streams AQ offers a content-based subscription model. Subscriber applications can specify interest based on message content. You should execute CMEF event subscriber code in the Oracle Portal schema. Example 16–5 shows a sample subscriber.

Example 16–5 An Example Subscriber

```
create or replace <procedure name> as
  agent_list         dbms_aq.aq$_agent_list_t;
  wait_time          integer       := <time in seconds>;
  agent_w_message    sys.aq$_agent;
  dequeue_options    dbms_aq.dequeue_options_t;
  message_properties dbms_aq.message_properties_t;
  message_handle     raw(16);
  message            <portal schema>.wwsbr_event;
  l_subscriber       varchar2(30)  := '<subscriber name>';
  l_queue            varchar2(30)  := 'PORTAL.WWSBR_EVENT_Q';
  l_mode             binary_integer := dbms_aq.[BROWSE|LOCK|REMOVE|REMOVE_NODATA];
  ...
  <additional parameters>
  ...
BEGIN
```

```
  agent_list(1) := sys.aq$_agent(l_subscriber, l_queue, null);
  loop
    -- Listen for messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait       => wait_time,
      agent      => agent_w_message
    );
    -- If there are messages for the subscriber then dequeue them.
    if (agent_w_message.name = l_subscriber) then
      dequeue_options.wait          := dbms_aq.NO_WAIT;
      dequeue_options.consumer_name := l_subscriber;
      dequeue_options.navigation    := dbms_aq.[NEXT_MESSAGE|FIRST_MESSAGE];
      dequeue_options.dequeue_mode  := l_mode;
      -- Dequeue messages.
      dbms_aq.dequeue(
        queue_name         => l_queue,
        dequeue_options    => dequeue_options,
        message_properties => message_properties,
        payload            => message,
        msgid              => message_handle
      );
      -- Determine the type of event that occurred and act accordingly.
      ...
      <your code here>
      ...
    end if;
  end loop;
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
end;
/
```

16.3.2 Adding a Subscriber to WWSBR_EVENT_Q

A subscriber subscribes to a queue from where it consumes messages. You have to add a subscriber to the WWSBR_EVENT_Q queue in order to process CMEF event messages, as shown in Example 16–6.

Example 16–6 Adding a Subscriber to WWSBR_EVENT_Q

```
declare
  subscriber sys.aq$_agent;
begin
  subscriber := sys.aq$_agent('<subscriber>', null, null);
  dbms_aqadm.add_subscriber(
    queue_name => '<portal schema>.wwsbr_event_q',
    subscriber => subscriber
  );
end;
/
```

16.3.3 Enabling CMEF Events at the Page Group Level

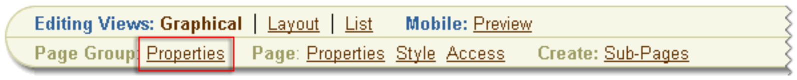
In Oracle Portal, CMEF is enabled or disabled at the page group level. By default, CMEF is enabled when a user creates a page group, and thus, events are triggered whenever changes occur within the Oracle Portal user interface or WebDAV.

To enable or disable CMEF for a page group, perform the following steps:

1. Go to any page of the page group and switch to Edit mode.
2. In the toolbar at the top of the page, click the **Properties** link next to Page Group.

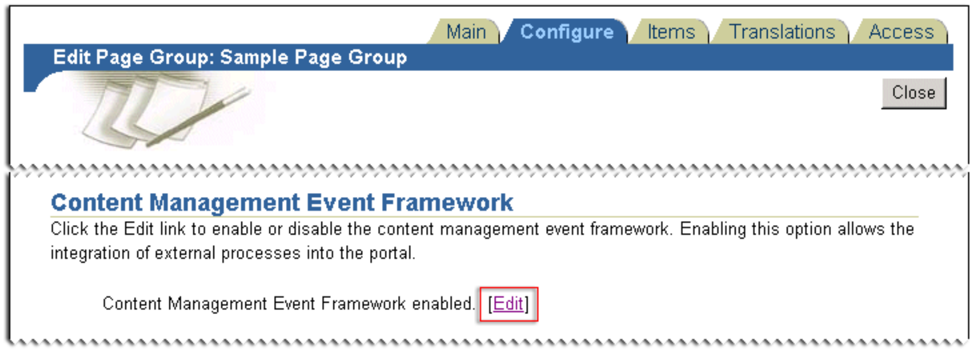
> **Note:** Make sure you click the link next to Page Group and not the one next to Page (Figure 16–2).

Figure 16–2 The Page Group Properties Link on the Edit Mode Toolbar


Editing Views: Graphical | Layout | List Mobile: Preview
Page Group: Properties Page: Properties Style Access Create: Sub-Pages

3. Click the **Configure** tab to bring it forward.
4. In the Content Management Event Framework section you can see whether CMEF is enabled or disabled. If you want to change this setting, click the **Edit** link (Figure 16–3).

Figure 16–3 Status of CMEF for a Page Group



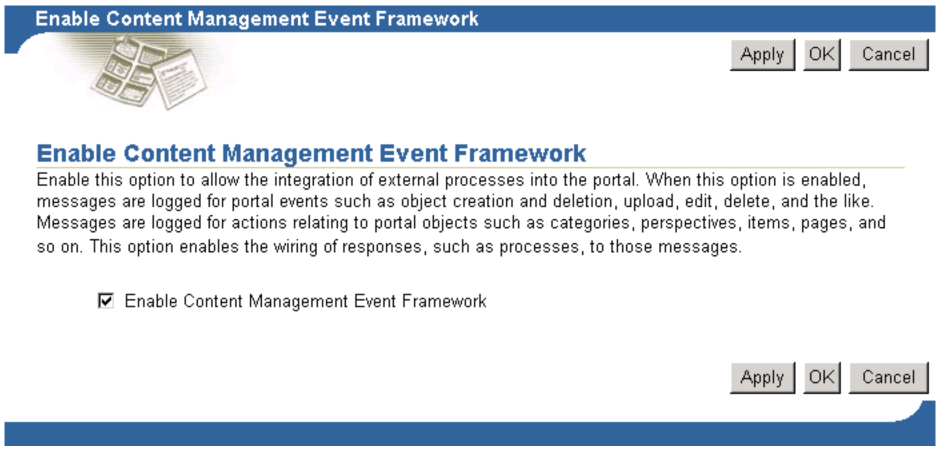
Content Management Event Framework

Click the Edit link to enable or disable the content management event framework. Enabling this option allows the integration of external processes into the portal.

Content Management Event Framework enabled. [Edit]

5. To enable CMEF, select the **Enable Content Management Event Framework** check box (Figure 16–4). To disable CMEF, clear the check box.

Figure 16–4 Enabling or Disabling CMEF for a Page Group



Enable Content Management Event Framework

Enable this option to allow the integration of external processes into the portal. When this option is enabled, messages are logged for portal events such as object creation and deletion, upload, edit, delete, and the like. Messages are logged for actions relating to portal objects such as categories, perspectives, items, pages, and so on. This option enables the wiring of responses, such as processes, to those messages.

☑ Enable Content Management Event Framework

6. Click **OK** to save your changes.
7. Click **Close** to return to the page.

16.3.4 Examining CMEF Events

Use the CMEF Events page (Figure 16–5) to examine the subscribers that have been added to the WWSBR_EVENTS_Q queue.

To access the CMEF Events page, perform the following steps:

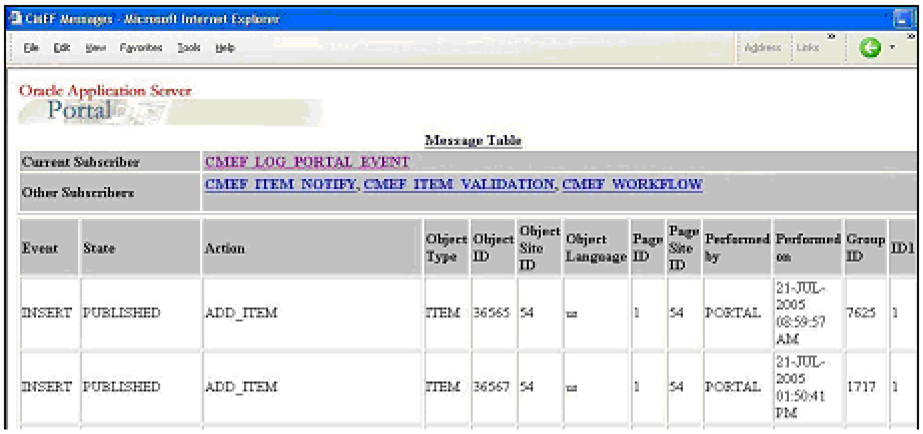
1. Login to your portal as the portal schema owner.
2. Enter the following URL in the browser Address field:

```
http://<host>:<port>/portal/pls/<dad>/<schema>.wwsbr_event_dbg.show
```

   - **host** is the machine on which your portal middle tier is installed.
   - **port** is the port used by your portal
   - **dad** is the Database Access Descriptor (DAD) for your Oracle Portal installation.
   - **schema** is the schema in which Oracle Portal is installed.

Note: The CMEF Events page is not supported by Oracle, but it is included within Oracle Portal for debugging purposes. It is only accessible by the portal schema owner.

Figure 16–5 CMEF Events Page


CMEF Messages - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address Links
Oracle Application Server
Portal

Message Table

Current Subscriber	CMEF_LOG_PORTAL_EVENT
Other Subscribers	CMEF_ITEM_NOTIFY, CMEF_ITEM_VALIDATION, CMEF_WORKFLOW

Event	State	Action	Object Type	Object ID	Object Site ID	Object Language	Page ID	Page Site ID	Performed by	Performed on	Group ID	ID1
INSERT	PUBLISHED	ADD_ITEM	ITEM	36565	54	us	1	54	PORTAL	21-JUL-2005 08:59:57 AM	7625	1
INSERT	PUBLISHED	ADD_ITEM	ITEM	36567	54	us	1	54	PORTAL	21-JUL-2005 01:50:41 PM	1717	1

16.3.5 Running a CMEF Subscriber

To run a subscriber, issue the command shown in Example 16–7 at a SQL prompt.

Example 16–7 Running a Subscriber

```
begin
  <subscriber procedure>();
end;
/
```

16.3.6 CMEF Message Payload

Every Oracle Portal user interface or PL/SQL content management action falls under one of the three main CMEF events: INSERT, UPDATE, DELETE. States provide more meaning to events. For example, the following types of ADD/INSERT ITEM events:

- Add an item and publish it immediately.
- Add an item and publish it at a later date.
- Add an item that requires approval, because the user has *Manage Items With Approval* privileges.

For a detailed list of the actions and related events and states, refer to Appendix G, "Content Management Event Framework Events".

Each CMEF event has an associated CMEF message payload as shown in Table 16–2.

Table 16–2 CMEF Message Payload Properties

Message Property	Type	Description
ACTION	VARCHAR2(30)	The portal action that triggered the event.
RAW_EVENT	VARCHAR2(30)	The event produced as a result of the portal action.
STATE	VARCHAR2(30)	Provides additional information related to the event.
OBJECT_ID	NUMBER	The ID of the object to which the event relates (for example, item, page, category, and so on).
OBJECT_SITE_ID	NUMBER	The ID of the page group to which the object belongs.
OBJECT_LANGUAGE	VARCHAR2(30)	The session language. NULL if the action is independent of language.
PAGE_ID	NUMBER	For items, the ID of the page on which the item appears. NULL for other objects.
PAGE_SITE_ID	NUMBER	For items, the ID of the page group to which the page identified in PAGE_ID belongs. NULL for non-item related events.
OBJECT_CLASS	VARCHAR2(30)	Records the class of object about which an event has been raised.
EVENTS_USER	VARCHAR2(256)	The name of the user who performed the portal action.
EVENTS_DATE	VARCHAR2(60)	The date on which the event occurred. Format: dd-mon-yyyy HH12:MI PM
ID1 (OVERLOADED)	NUMBER	For items, the item type ID. For pages, the page type ID. For item and page types, the base type ID.

Table 16–2 (Cont.) CMEF Message Payload Properties

Message Property	Type	Description
SITE_ID1 (OVERLOADED)	NUMBER	For item types, the ID of the page group to which the item type belongs. For pages, the ID of the page group to which the page type belongs. For item and page types, the ID of the page group to which the base type belongs.
GROUP_ID	NUMBER	When multiple messages are associated with a particular event, related messages have the same group ID.
OBJECT PATH	VARCHAR2(4000)	A unique path to the portal object being referenced by this message on the queue (this can be NULL). It is used only for pages, items, categories, perspectives, item types, and page types.
OBJECT UID	VARCHAR2(4000)	A unique immutable identifier to the portal object being referenced by this message. It is used only for pages, items, categories, perspectives, item types, and page types.

16.3.7 Oracle Portal Actions and CMEF Events

This section describes some of the most common portal actions and shows how to include code in your subscriber to detect these actions. These actions may occur through the Oracle Portal user interface, the Oracle Portal PL/SQL APIs, or WebDAV.

For a more detailed list of portal actions and the events and message payloads that they generate, refer to Appendix G, "Content Management Event Framework Events".

16.3.7.1 Page and Page Group Actions

In Oracle Portal, a portal is a collection of one or more page groups. A page group is a hierarchical collection of pages for which common attributes and mechanisms can be established.

16.3.7.1.1 Creating a Page Creating a page produces the following CMEF message payload:

Action	Event	State	Object Class
ADD_PAGE	INSERT	PUBLISHED	PAGE
ADD_ITEM	INSERT	PUBLISHED	ITEM

The first message is for the page itself, and the second is for the portlet instance that displays the default navigation page on the page.

If you want your subscriber to respond to the creation of a page, perform the following check:

```
if ((message.object_class = 'PAGE') and
    (message.raw_event    = wwsbr_event_q_access.EVENT_INSERT) then
. . .
end if;
```

16.3.7.1.2 Updating the Access Control List of a Page Changing the ACL of a page so that it does not inherit from that of its page group, then clicking **Apply** or **OK** produces the following message payload:

Action	Event	State	Object Class
SPECIFY_PAGE_ACL	UPDATE	GENERAL	PAGE

Now, adding a user to the ACL of a page, then clicking **Add** produces the following message payload:

Action	Event	State	Object Class
ADD_PAGE_ACL	UPDATE	GENERAL	PAGE

Changing the ACL of a page so that it does not inherit from that of its page group, immediately adding a user or group to the ACL, and then clicking **Apply** or **OK** produces the following message payload:

Action	Event	State	Object Class
SPECIFY_AND_ ADD_PAGE_ACL	UPDATE	GENERAL	PAGE

Changing the ACL of a page so that it inherits that of the page group then clicking **Apply** or **OK** produces the following message payload:

Action	Event	State	Object Class
INHERIT_PAGE_ ACL	UPDATE	GENERAL	PAGE

Clicking **Apply** or **OK** on the Page Properties page, produces an additional message:

Action	Event	State	Object Class
UPDATE_PAGE_ACL	UPDATE	GENERAL	PAGE

16.3.7.1.3 Updating the Access Control List of a Page Group Updating the access control list (ACL) of a page group by adding a user or group, or deleting a user or group produces the following message payload:

Action	Event	State	Object Class
ADD_PAGEGROUP_ ACL or DELETE_PAGE_ GROUP_ACL	UPDATE	GENERAL	PAGE_GROUP
UPDATE_PAGE_ GROUP_ACL	UPDATE	GENERAL	PAGE_GROUP

If you want your subscriber to respond to general ACL updates on a page group, perform the following check:

```
if ((message.action       = 'UPDATE_PAGE_GROUP_ACL') and
    (message.object_class = 'PAGE_GROUP') and
    (message.raw.event    = wwsbr_event_q_access.EVENT_UPDATE)) then
. . .
end if;
```

However, if you are more interested in filtering for actual update and delete events on the page group's ACL, then your subscriber should perform the following checks:

```
if ((message.object_class = 'PAGE_GROUP') and
    (message.raw_event    = wwsbr_event_q_access.EVENT_UPDATE)) then
  if (message.action = 'ADD_PAGE_GROUP_ACL') then
    . . .
  end if;
  if (message.action = 'DELETE_PAGE_GROUP_ACL') then
    . . .
  end if;
  . . .
end if;
```

Note: The ADD_PAGE_GROUP_ACL and DELETE_PAGE_GROUP_ACL events are triggered when the user clicks **Add** for each user that is added to/deleted from the ACL. The UPDATE_PAGE_GROUP_ACL is generated when the user clicks **Apply** or **OK**.

16.3.7.1.4 Deleting a Page Deleting a page produces the following message payload:

Action	Event	State	Object Class
DELETE_PAGE	DELETE	PURGED	PAGE

Note: Events are not generated for the sub-pages or items that are on the page.

16.3.7.2 Item Actions

Items are one of the basic components of a portal page. Items in a portal are based on item types. An action on an item triggers a CMEF event irrespective of the item type.

For example, an ADD_ITEM action occurs whenever a user adds an item of a base, extended, or custom item type. This ensures that there is consistent CMEF messaging behavior when an item action occurs.

16.3.7.2.1 Creating an Item and Publishing it at the Same Time If a user has *Manage* privileges, creating an item of any type on a page produces the following message payload:

Action	Event	State	Object Class
ADD_ITEM	INSERT	PUBLISHED	ITEM

If you want you subscriber to respond to this action, perform the following check:

```
if ((message.action       = 'ADD_ITEM') and
    (message.object_class = 'ITEM') and
    (message.raw_event    = 'INSERT') and
    (message.state        = 'PUBLISHED')) then
   . . .
end if;
```

16.3.7.2.2 Adding an Item That Requires Approval

If approvals and notifications are enabled for a page (or page group), and a user with *Manage Items With Approval* privileges, adding an item to a page produces the following message payload:

Action	Event	State	Object Class
SUBMIT_ITEM_FOR_APPROVAL	INSERT	NOT_PUBLISHED	ITEM

If you want your subscriber to respond to this action, perform the following check:

```
if ((message.action       = 'SUBMIT_ITEM_FOR_APPROVAL') and
    (message.object_class = 'ITEM') and
    (message.raw_event    = 'INSERT') and
    (message.state        = 'NOT_PUBLISHED')) then
   . . .
end if;
```

16.3.7.2.3 Approving an Item

Approving an item triggers either an INSERT or UPDATE event, which may also be followed by either a DELETE or UPDATE event depending upon whether or not versioning is enabled.

Item approved; versioning disabled

This is the simplest case. If the item still has approval steps to pass through, then an UPDATE event is triggered:

Action	Event	State	Object Class
APPROVE_ITEM_STEP	UPDATE	GENERAL	ITEM

If the item has completed all approval steps, then an INSERT event is triggered:

Action	Event	State	Object Class
APPROVE_ITEM	INSERT	PUBLISHED	ITEM

Item approved; versioning enabled; current version overwritten

If item versioning is set to **Simple** at the page or page group level, and a user selects **Overwrite Current Version** when editing an item, on approval of the item, two events are generated. The first event is for the item that is approved, and a second DELETE event for the item that is overwritten as a result of the item being approved:

Action	Event	State	Object Class
APPROVE_ITEM	INSERT	PUBLISHED	ITEM
APPROVE_ITEM	DELETE	PURGED	ITEM

Item approved; versioning enabled; new and current version

If item versioning is set to **Audit** at the page or page group level, or it is set to **Simple** and a user selects **Add Item As New And Current Version**, on approval of the item an INSERT event is generated. An UPDATE event is also generated that is related to marking the previous version of the item as UNPUBLISHED:

Action	Event	State	Object Class
APPROVE_ITEM	INSERT	PUBLISHED	ITEM
APPROVE_ITEM	UPDATE	UNPUBLISHED	ITEM

If you are writing a subscriber that sends a notification when an item is approved or has passed through a stage of being approved, you should perform the following check:

```
-- If an item is approved.
if ((message.action       = 'APPROVE_ITEM') and
    (message.object.class = 'ITEM') and
    (message.raw_event    = 'INSERT') and
    (message.state        = 'PUBLISHED')) then
   . . .
-- If an item has passed an approval step.
elsif ((message.action       = 'APPROVE_ITEM_STEP') and
       (message.object_class = 'ITEM') and
       (message.raw_event    = 'UPDATE') and
       (message.state        = 'PUBLISHED')) then
   . . .
end if;
```

Item approved, versioning enabled; new but not current version

An INSERT event occurs for the item that is added, but the state of the item is marked as NOT_PUBLISHED to indicate that it is not published as the current version:

Action	Event	State	Object Class
APPROVE_ITEM	INSERT	NOT_PUBLISHED	ITEM

16.3.7.2.4 Applying a Category or Perspective to an Item Applying a different category or perspective to an item produces the same message payload as editing an item:

Action	Event	State	Object Class
EDIT_ITEM	UPDATE	GENERAL	ITEM

Note: No specific event is generated when the category or perspectives applied to an item are changed, and no additional information is provided.

If you want your subscriber to respond to this action, perform the following check:

```
if ((message.action       = 'EDIT_ITEM') and
    (message.object_class = 'ITEM') and
    (message.raw_event    = 'UPDATE') and
    (message.state        = 'GENERAL') then
```

```
   . . .
end if;
```

16.3.7.2.5 Deleting an Item Deleting an item from a page group that retains deleted items (that is, items are marked for deletion, but not actually deleted) produces the following message payload:

Action	Event	State	Object Class
DELETE_ITEM	DELETE	MARKED_FOR_ DELETE	ITEM

Deleting an item from a page group that does not retain deleted items (that is, deleted items are immediately and permanently removed) produces the following message payload:

Action	Event	State	Object Class
DELETE_ITEM	DELETE	PURGED	ITEM

Your subscriber can use the state value within the message payload to determine what type of delete action occurred:

```
if ((message.action       = 'DELETE_ITEM') and
    (message.object_class = 'ITEM') and
    (message.raw_event    = 'DELETE')) then
  . . .
  -- If item is in a page group that does not actually delete items.
  if (message.state = 'MARKED_FOR_DELETE') then
      . . .
  -- If item is in a page group that actually deletes items.
  elsif (message.state = 'PURGED') then
      . . .
  end if;
end if;
```

16.4 Installing the Examples

If you would like to deploy and use the examples within the next few sections, we recommend that you create them in a separate schema, called CMEFSAMPLES. To create this schema, use the following steps:

1. Create the database schema CMEFSAMPLES. You need to do this as the SYS user, since you need to grant permissions packages to which only SYS has access. This database schema must be in the same database as that in which the Oracle Portal repository resides. For example:

```
@connect "/ as sysdba"
drop user cmefsamples cascade;
create user cmefsamples identified by oracle1
default tablespace users temporary tablespace temp;
grant connect, resource to cmefsamples;
```

2. As the SYS schema, grant the following privileges to CMEFSAMPLES:

```
@connect "/ as sysdba"
grant create table                  to cmefsamples;
grant create sequence              to cmefsamples;
grant create view                  to cmefsamples;
grant create procedure             to cmefsamples;
grant create trigger               to cmefsamples;
grant create indextype             to cmefsamples;

grant create synonym               to cmefsamples;
grant create public synonym        to cmefsamples;

grant create database link         to cmefsamples;
grant create public database link  to cmefsamples;
grant execute on dbms_utility      to cmefsamples;
grant aq_administrator_role        to cmefsamples;
grant aq_user_role                 to cmefsamples;
grant execute  on dbms_aqadm       to cmefsamples;
grant execute  on dbms_aq          to cmefsamples;
grant execute  on aq$_agent        to cmefsamples;
grant execute  on dbms_job         to cmefsamples;

execute dbms_aqadm.grant_type_access('cmefsamples');
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','cmefsamples',FALSE);
execute dbms_aqadm.grant_system_privilege('DEQUEUE_ANY','cmefsamples',FALSE);
execute dbms_aqadm.grant_system_privilege('MANAGE_ANY', 'cmefsamples', FALSE);
EXECUTE DBMS_AQADM.GRANT_QUEUE_PRIVILEGE('ENQUEUE','portal.WWSBR_EVENT_
Q','cmefsamples', FALSE);
EXECUTE DBMS_AQADM.GRANT_QUEUE_PRIVILEGE('DEQUEUE','portal.WWSBR_EVENT_
Q','cmefsamples', FALSE);
```

3. Grant the CMEFSAMPLES schema the permission to call the Oracle Portal PL/SQL APIs. For information about how to do this, refer to Section 9.3, "Providing Access to the APIs and Secure Views".
4. Log in to the portal schema and grant permissions on the following:

```
EXECUTE DBMS_AQADM.GRANT_QUEUE_PRIVILEGE('DEQUEUE','WWSBR_EVENT_
Q','cmefsamples', FALSE);
grant execute on wwsbr_event_q_access to cmefsamples;
grant execute on wwpob_page_util to cmefsamples;
grant select on wwsbr_all_folders to cmefsamples;
grant execute on wwsbr_thing_types to cmefsamples;
grant execute on wwv_thingdb to cmefsamples;
grant execute on wwsbr_event to cmefsamples;
```

5. Log in to the CMEFSAMPLES schema and run the following:

```
sqlplus cmefsamples/<password>
create synonym aq$_agent for sys.aq$_agent;
```

More On OTN

You can download the code for the following examples from OTN:

http://www.oracle.com/technology/products/ias/portal/files/cm_overview_10g1014_cmef_samples.zip

16.5 Example: Portal Object Event Logging

The LOG_PORTAL_EVENT subscriber in Example 16–9 listens to CMEF events and then writes them to a log database table called CMEF_LOG_TABLE (Example 16–8).

Example 16–8 The CMEF_LOG_TABLE

```
create table cmef_log_table(
  action          varchar2(30),
  event           varchar2(30),
  state           varchar2(30),
  object_type     varchar2(30),
  object_id       number,
  object_site_id  number,
  object_language varchar2(30),
  page_id         number,
  page_site_id    number,
  performed_by    varchar2(30)
)
```

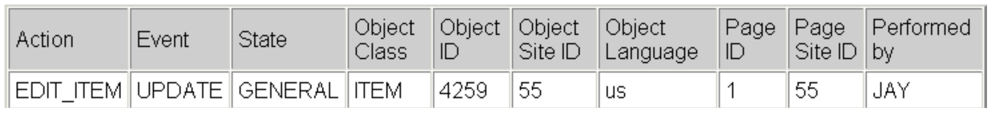
Example 16–9 The LOG_PORTAL_EVENT Subscriber

```
create or replace procedure log_portal_event as
  agent_list         dbms_aq.aq$_agent_list_t;
  wait_time          integer        := 5;
  agent_w_message    sys.aq$_agent;
  dequeue_options    dbms_aq.dequeue_options_t;
  message_properties dbms_aq.message_properties_t;
  message_handle     raw(16);
  message            portal.wwsbr_event;
  l_subscriber       varchar2(30)   := 'CMEF_LOG_PORTAL_EVENT';
  l_queue            varchar2(30)   := 'PORTAL.WWSBR_EVENT_Q';
  l_mode             binary_integer := dbms_aq.REMOVE;
begin
  agent_list(1) := sys.aq$_agent(l_subscriber, l_queue, null);
  loop
    -- Wait for messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait       => wait_time,
      agent      => agent_w_message
    );
    if (agent_w_message.name = l_subscriber) then
      dequeue_options.wait          := dbms_aq.NO_WAIT;
      dequeue_options.consumer_name := l_subscriber;
      dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
      dequeue_options.dequeue_mode  := l_mode;
      dbms_aq.dequeue(
        queue_name         => l_queue,
        dequeue_options    => dequeue_options,
        message_properties => message_properties,
        payload            => message,
        msgid              => message_handle
      );
      insert into cmef_log_table values(
        message.action,
        message.raw_event,
        message.state,
        message.object_class,
        message.object_id,
        message.object_site_id,
        message.object_language,
        message.page_id,
        message.page_site_id,
        message.events_user
      );
      commit;
    end if;
  end loop;
end;
/
```

The LOG_PORTAL_EVENT subscriber continuously listens for events on the WWSBR_EVENT_Q queue until the wait time of 5 seconds (as specified in the `wait_time` variable) is reached. It then dequeues CMEF events in REMOVE mode and then inserts the message payload values into the log table.

You could use the code in Example 16–9 to build an HTML page that displays the results of the log table. For example, the table in Figure 16–6 shows the CMEF message payload for an edit item action:

Figure 16–6 HTML Table Displaying CMEF Message Payload Values



Action	Event	State	Object Class	Object ID	Object Site ID	Object Language	Page ID	Page Site ID	Performed by
EDIT_ITEM	UPDATE	GENERAL	ITEM	4259	55	us	1	55	JAY

The properties of the CMEF message payload (for example, `message.raw_event`) are described in Section 16.3.6, "CMEF Message Payload".

If you want the LOG_PORTAL_EVENT subscriber to continually remove messages off the WWSBR_EVENT_Q queue, then you need to remove the following from Example 16–9:

```
dbms_aq.listen(
  agent_list => agent_list,
  wait       => wait_time,
  agent      => agent_w_message
);
if (agent_w_message.name = l_subscriber) then
  dequeue_options.wait           := dbms_aq.NO_WAIT;
  dequeue_options.consumer_name := l_subscriber;
  dequeue_options.navigation     := dbms_aq.FIRST_MESSAGE;
  dequeue_options.dequeue_mode   := l_mode;
END IF;
```

Example 16–10 shows how to add the LOG_PORTAL_EVENT subscriber created in the previous section to the WWSBR_EVENT_Q queue.

Example 16–10 Adding the LOG_PORTAL_EVENT Subscriber to WWSBR_EVENT_Q

```
declare
  subscriber sys.aq$_agent;
begin
  subscriber := sys.aq$_agent('CMEF_LOG_PORTAL_EVENT', null, null);
  dbms_aqadm.add_subscriber(
    queue_name => 'portal.wwsbr_event_q',
    subscriber => subscriber
  );
end;
/
```

To run the LOG_PORTAL_EVENT subscriber, issue the command shown in Example 16–11.

Example 16–11 Running the LOG_PORTAL_EVENT CMEF Subscriber

```
begin
  log_portal_event();
end;
/
```

16.6 Example: Item Notification

The item_notify subscriber in Example 16–12 sends an e-mail notification whenever a user adds, updates, or deletes an item on a specified page:

Example 16–12 The CMEF_ITEM_NOTIFY Subscriber

```
create or replace procedure item_notify as
  MIME_TYPE_TEXT      constant varchar(30)  := 'text/plain';
  CUSTOM_ATTRIBUTE_ID constant number := 1020;
  ADDED               constant varchar2(20) := 'added';
  UPDATED             constant varchar2(20) := 'updated';
  DELETED             constant varchar2(20) := 'deleted';
  ITEM                constant varchar2(10) := 'ITEM';
  agent_list          dbms_aq.aq$_agent_list_t;
  wait_time           integer               := 5;
  begin_time          pls_integer;
  end_time            pls_integer;
  agent_w_message     sys.aq$_agent;
  dequeue_options     dbms_aq.dequeue_options_t;
  message_properties  dbms_aq.message_properties_t;
  message_handle      raw(16);
  message             portal.wwsbr_event;
  l_subscriber        varchar2(30)          := 'CMEF_ITEM_NOTIFY';
  l_queue             varchar2(30)          := 'PORTAL.WWSBR_EVENT_Q';
  l_page_id           number                := 33; -- Page ID of page.
  l_mode              binary_integer        := dbms_aq.REMOVE;
  l_rec               varchar2(256);
  l_body              varchar2(4000)        := null;
  l_from_user         varchar2(30)          := '<from-email-address>';
  l_to_user           varchar2(30)          := '<to-email-address>';
  l_event             varchar2(30)          := '';
  l_portal_user_name  varchar2(30)          := 'portal';
  l_portal_password   varchar2(30)          := '<portal password>';
  l_display_name      varchar2(256)         := '';

begin
  begin_time :=  dbms_utility.get_time;
  agent_list(1) := sys.aq$_agent(l_subscriber, l_queue, null);
  loop
    -- Wait for messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait       => wait_time,
      agent      => agent_w_message
    );
    if (agent_w_message.name = l_subscriber) then
      dequeue_options.wait          := DBMS_AQ.NO_WAIT;
      dequeue_options.consumer_name := l_subscriber;
      dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
      dequeue_options.dequeue_mode  := l_mode;
      dbms_aq.dequeue(
        queue_name         => l_queue,
        dequeue_options    => dequeue_options,
        message_properties => message_properties,
        payload            => message,
        msgid              => message_handle
      );
      if ((message.object_class = ITEM)) then
        -- Determine the type of event that occurred.
        if (message.raw_event = portal.wwsbr_event_q_access.EVENT_INSERT) then
          l_event  := ADDED;
        elsif (message.raw_event = portal.wwsbr_event_q_access.EVENT_UPDATE) then
          l_event  := UPDATED;
        elsif (message.raw_event = portal.wwsbr_event_q_access.EVENT_DELETE) then
          l_event  := DELETED;
        end if;
        if ((l_event = ADDED) or
            (l_event = UPDATED) or
            (l_event = DELETED)) then
          -- Set the Portal Context.
          portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);

          begin
            -- Get the Item Display name from the all items view.
            select display_name
            into l_display_name
            from portal.wwsbr_all_items
            where id=message.object_id
            and caid=message.object_site_id;
          exception
            when NO_DATA_FOUND then
              dbms_output.put_line(sqlerrm);
              exit;
          end;

          -- Only send an e-mail if the item has a display name.
          if (l_display_name is not null) then
            -- Construct the body of the message.
            l_body := l_body || '<br>An item titled '
            || CSBR_UTIL.file_viewer_hyperlink(
            message.object_id, message.page_site_id, l_display_name)
            || ' was '
            || l_event || ' on '
            || to_char(message.events_date,'dd-mon-yyyy hh12:mi pm')
            || ' by ' || message.events_user || '. ';
            -- Send the message.
            html_email(
              p_from => l_from_user,
              p_to => l_to_user,
              p_subject => 'Item ' || l_event || ' Notification',
              p_text => 'Text',
              p_html => l_body
            );
          end if;
        end if;
      end if;
    end if;
    end_time :=  dbms_utility.get_time;
    if (end_time - begin_time) > 3000 then
      exit;
    end if;
  end loop;
end;
/
```

The ITEM_NOTIFY CMEF subscriber removes the actual events from the queue so that multiple e-mails are not sent to the user.

Note: For descriptions of the `file_viewer_hyperlink` function, which gets the URL of the item, and the `html_email` procedure, which handles the actual sending of the message, refer to "Additional Code".

More On OTN

For further information on the performance and timing of AQ events, refer to the Streams Advanced Queuing page on OTN:

`http://www.oracle.com/technology/products/aq`

For an example of how to add a subscriber to the WWSBR_EVENT_Q queue, refer to Example 16–10.

For an example of how to run a subscriber, refer to Example 16–11.

The ITEM_NOTIFY CMEF subscriber runs for 3 seconds. This ensures that the subscriber does not continuously run and consume system resources. Alternatively, you could modify the `wait_time` value used in the subscriber to a higher value, say 60 seconds, so that the subscriber listens for new events every minute. You could also use a DBMS_JOB to have the subscriber run at a specified interval, for example, 30 minutes, as shown in Example 16–13.

Example 16–13 Using DMBS_JOB

```
declare
  v_job number;
begin
  dbms_job.submit(
    job      => v_job,
    what     => 'item_notify()',
    interval => 'SYSDATE + (30/(24*60))'
  );
end;
/
```

Additional Code

Example 16–14 shows the CSBR_UTIL package.

Example 16–14 CSBR_UTIL Package

```
create or replace package CSBR_UTIL as
  function file_viewer_hyperlink(
    p_item_id in number,
    p_caid    in number,
    p_text    in varchar2) return varchar2;
end;

create or replace package body CSBR_UTIL as

  function file_viewer_url(
    p_item_id in number,
    p_caid    in number) return varchar2 is

    l_return      varchar2(10000);
    l_item_name   varchar2(100);
    l_folder_id   number;
    l_folder_name varchar2(100);
    l_portal_url  varchar2(1000);
  begin
    select name, folder_id
    into l_return, l_folder_id
    from portal.wwsbr_all_items
    where id   = p_item_id
    and caid   = p_caid
    and active = 1;
    begin
      while 1=1 loop
        select parent_id, name
        into l_folder_id, l_folder_name
        from portal.wwsbr_all_folders
        where id = l_folder_id
        and caid = p_caid;
        l_return := l_folder_name||'/'||l_return;
      end loop;
    exception
      when NO_DATA_FOUND then
        null; -- Exit loop at no rows found, this is expected.
    END;
    -- Set the Portal URL.
    l_portal_url := 'http://<host>:<port>/portal/page/<dad>/';
    return l_portal_url||l_return;
    exception
      when OTHERS then
        dbms_output.put_line(sqlerrm);
        return 'Error';
  end file_viewer_url;

  function file_viewer_hyperlink(
    p_item_id in number,
    p_caid    in number,
    p_text    in varchar2) return varchar2 is
  begin
    return ('<a href="'||file_viewer_url(
      p_item_id => p_item_id,
      p_caid    => p_caid)||'">'||p_text||'</a>');
  end file_viewer_hyperlink;

end;
/
```

Example 16–15 shows the html_email procedure.

Note: For this procedure to work you will need to deploy the Oracle Database Sendmail package on your database.

Example 16–15 The HTML E-mail Procedure

```
create or replace procedure html_email(
  p_to      in varchar2,
  p_from    in varchar2 default '',
  p_subject in varchar2,
  p_text    in varchar2 default null,
  p_html    in varchar2 default null
)
is
  l_boundary   varchar2(255) default 'a1b2c3d4e3f2g1';
  l_connection utl_smtp.connection;
  l_body_html  clob := empty_clob; -- This LOB will be the e-mail message.
  l_offset     number;
  l_ammount    number;
  l_temp       varchar2(32767) default null;
begin
  l_connection := utl_smtp.open_connection('<Mail Server here>');
  utl_smtp.helo( l_connection, '<Mail Server here>' );
  utl_smtp.mail( l_connection, p_from );
  utl_smtp.rcpt( l_connection, p_to );
  l_temp := l_temp || 'MIME-Version: 1.0' || chr(13) || chr(10);
  l_temp := l_temp || 'To: ' || p_to || chr(13) || chr(10);
  l_temp := l_temp || 'From: ' || p_from || chr(13) || chr(10);
  l_temp := l_temp || 'Subject: ' || p_subject || chr(13) || chr(10);
  l_temp := l_temp || 'Reply-To: ' || p_from || chr(13) || chr(10);
  l_temp := l_temp || 'Content-Type: multipart/alternative; boundary=' || chr(34)
|| l_boundary || chr(34) || chr(13) || chr(10);
  --------------------------------------------------------------------------------
  -- Write the headers.
  dbms_lob.createtemporary( l_body_html, false, 10 );
  dbms_lob.write(l_body_html, length(l_temp), 1,l_temp);

  ---------------------------------------------------------------------------------
  -- Write the text boundary.
  l_offset := dbms_lob.getlength(l_body_html) + 1;
  l_temp   := '--' || l_boundary || chr(13) || chr(10);
  l_temp   := l_temp || 'content-type: text/plain; charset=us-ascii' || chr(13) ||
chr(10) || chr(13) || chr(10);
  dbms_lob.write(l_body_html, length(l_temp), l_offset, l_temp);

  --------------------------------------------------------------------------------
  -- Write the plain text portion of the e-mail.
  l_offset := dbms_lob.getlength(l_body_html) + 1;
  dbms_lob.write(l_body_html, length(p_text), l_offset, p_text);

  --------------------------------------------------------------------------------
  -- Write the HTML boundary.
  l_temp := chr(13) || chr(10) || chr(13) || chr(10) || '--' || l_boundary ||
chr(13) || chr(10);
  l_temp := l_temp || 'content-type: text/html;' || chr(13) || chr(10) || chr(13)
|| chr(10);
  l_offset := dbms_lob.getlength(l_body_html) + 1;
  dbms_lob.write(l_body_html, length(l_temp), l_offset, l_temp);

  --------------------------------------------------------------------------------
  -- Write the HTML portion of the message.
  l_offset := dbms_lob.getlength(l_body_html) + 1;
  dbms_lob.write(l_body_html, length(p_html), l_offset, p_html);

  --------------------------------------------------------------------------------
  -- Write the final HTML boundary.
  l_temp := chr(13) || chr(10) || '--' || l_boundary || '--' || chr(13);
  l_offset := dbms_lob.getlength(l_body_html) + 1;
  dbms_lob.write(l_body_html, length(l_temp), l_offset, l_temp);

  ----------------------------------------------------------------------------
  -- Send the e-mail in 1900 byte chunks to UTL_SMTP.
  l_offset := 1;
  l_ammount := 1900;
  utl_smtp.open_data(l_connection);
  while l_offset < dbms_lob.getlength(l_body_html) loop
    utl_smtp.write_data(l_connection, dbms_lob.substr(l_body_html,l_ammount,l_
offset));
    l_offset := l_offset + l_ammount;
    l_ammount := least(1900, dbms_lob.getlength(l_body_html) - l_ammount);
  end loop;
  utl_smtp.close_data(l_connection);
  utl_smtp.quit( l_connection );
  dbms_lob.freetemporary(l_body_html);
end;
/
```

16.7 Example: Item Validation

The item_valication CMEF subscriber in Example 16–16 validates that a URL item added to a specified portal page includes a URL. If the item includes a URL, then the item is approved using the `wwsbr_api.approve` API. Otherwise the item is rejected using the `wwsbr_api.reject` API.

Example 16–16 The ITEM_VALIDATION CMEF Subscriber

```
create or replace procedure item_validation as
  MIME_TYPE_TEXT      constant varchar2(30) := 'text/plain';
  agent_list          dbms_aq.aq$_agent_list_t;
  wait_time           integer               := 5;
  begin_time          pls_integer;
  end_time            pls_integer;
  agent_w_message     sys.aq$_agent;
  dequeue_options     dbms_aq.dequeue_options_t;
  message_properties  dbms_aq.message_properties_t;
  message_handle      raw(16);
  message             portal.wwsbr_event;
  l_subscriber        varchar2(30)          := 'CMEF_ITEM_VALIDATION';
  l_queue             varchar2(30)          := 'PORTAL.WWSBR_EVENT_Q';
  l_mode              binary_integer        := dbms_aq.REMOVE;
  l_attribute_site_id number                := 0;
  l_event             varchar2(30);
  l_doc               wwdoc_api.document_record;
  l_desc              varchar2(4000);
  l_portal_user_name  varchar2(30)          := 'portal';
  l_portal_password   varchar2(30)          := ' <portal password>';
  l_itemtype          varchar2(30)          := '';
  l_url               varchar2(4000)        := '';
begin
  begin_time :=  dbms_utility.get_time;
  agent_list(1) := sys.aq$_agent(l_subscriber, l_queue, null);
  loop
    -- Wait for messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait       => wait_time,
      agent      => agent_w_message
    );
    if (agent_w_message.name = l_subscriber) then
      dequeue_options.wait := DBMS_AQ.NO_WAIT;
      dequeue_options.consumer_name := l_subscriber;
      dequeue_options.navigation := dbms_aq.FIRST_MESSAGE;
      dequeue_options.dequeue_mode := l_mode;
      dbms_aq.dequeue(
        queue_name         => l_queue,
        dequeue_options    => dequeue_options,
        message_properties => message_properties,
        payload            => message,
        msgid              => message_handle
      );
      -- If the event is an ITEM INSERT event and marked for later publication.
      if ((message.action       = 'SUBMIT_ITEM_FOR_APPROVAL') and
          (message.object_class = 'ITEM') and
          (message.raw_event    = 'INSERT') and
          (message.state        = 'NOT_PUBLISHED')) then

        -- Set the Portal Context.
        portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);
        -- Get the Object Information.
        begin
          select itemtype, url
          into l_itemtype, l_url
          from portal.wwsbr_all_items
          where id=message.object_id
            and caid=message.object_site_id;
        exception
          when NO_DATA_FOUND then
            dbms_output.put_line(sqlerrm);
            exit;
        end;
        -- If the item type is a URL and no URL is specified.
        if (l_itemtype = portal.wwsbr_thing_types.BASE_ITEM_TYPE_URL) and
           (l_url is null) then
          begin
            -- Update item to indicate that a URL needs to be specified.
            l_desc := '<font color=RED>NO URL SPECIFIED</font>';
            -- Reset the CMEF global variables.
            wwsbr_api.clear_cmef_context;
            wwsbr_api.set_attribute(
              p_site_id           => message.object_site_id,
              p_thing_id          => message.object_id,
              p_attribute_site_id => l_attribute_site_id,
              p_attribute_id      => wwsbr_api.ATTRIBUTE_DESCRIPTION,
              p_attribute_value   => l_desc
              p_log_cmef_message  => FALSE
            );
            -- Reject the item.
            wwsbr_api.reject(
              p_item_id => message.object_id,
              p_site_id => message.object_site_id,
              p_comment => 'Rejected'
            );
            -- Reset the CMEF global variables.
            wwsbr_api.clear_cmef_context;
          exception
            when OTHERS then
```

```
                dbms_output.put_line(sqlerrm);
            end;
          else
            -- Approve the item.
            begin
              wwsbr_api.approve(
                p_item_id => message.object_id,
                p_site_id => message.object_site_id,
                p_comment => 'Approved'
              );
              -- Reset the CMEF global variables.
              wwsbr_api.clear_cmef_context;
            exception
              when OTHERS then
                dbms_output.put_line(sqlerrm);
            end;
          end if;
          commit;
        end if;
      end if;
      end_time :=  dbms_utility.get_time;
      if (end_time - begin_time) > 3000 then
        exit;
      end if;
    end loop;
    -- Process cache invalidation messages.
    wwpro_api_invalidation.execute_cache_invalidation;
  end;
  /
```

The following check in the item_validation CMEF subscriber determines whether the item requires approval:

```
-- If the event is an ITEM INSERT event and marked for later publication.
if ((message.action       = 'SUBMIT_ITEM_FOR_APPROVAL') and
    (message.object_class = 'ITEM') and
    (message.raw_event    = 'INSERT') and
    (message.state        = 'NOT_PUBLISHED')) then
   . . .
end if;
```

The item_validation CMEF subscriber uses the `wwsbr_api.set_attribute` API to set the item description to indicate that the user did not specify a URL. The subscriber also chooses not to log CMEF messages when calling this API (`p_log_cmef_messages = FALSE`).

```
-- Update item to indicate that a URL needs to be specified.
l_desc := '<font color=RED>No FILE NAME SPECIFIED</font>';
wwsbr_api.set_attribute(
  p_site_id           => l_thing.siteid,
  p_thing_id          => l_thing.id,
  p_attribute_site_id => l_attribute_site_id,
  p_attribute_id      => wwsbr_api.ATTRIBUTE_DESCRIPTION,
  p_attribute_value   => l_desc
  p_log_cmef_message  => FALSE
);
```

Approving or rejecting an item with the `approve` or `reject` APIs triggers an event just the same as if the item were approved or rejected through the Oracle Portal UI.

For example, calling the `wwsbr_api.approve` API triggers an INSERT event with an action of APPROVE_ITEM.

```
wwsbr_api.approve(
  p_item_id => l_thing.id,
  p_site_id => l_thing.siteid,
  p_comment => 'Approved'
);
```

For an example of how to add a subscriber to the WWSBR_EVENT_Q queue, refer to Example 16–10.

For an example of how to run a subscriber, refer to Example 16–11.

16.8 Example: Integrating External Workflow

Organizations often find that they want to integrate their business processes into their enterprise portal. Using the capabilities of Oracle Workflow, you can implement portals to route information of any type according to the compliance rules defined for your organization. With CMEF, you can integrate both traditional, applications-based workflow and e-business integration workflow with your Oracle Portal content management system.

This section provides an example of how to integrate external workflow with the Oracle Portal content management system.

16.8.1 Integrating Workflow with Oracle Portal

Oracle Workflow is a component of Oracle Application Server and Oracle E-Business Suite that enables you to design internal business processes and store them in a central repository. You can use Oracle Workflow to support a wide variety of compliance mandates, designing processes that are both auditable and repeatable, and enforce pre-set approvals and limits. Oracle's newest compliance solution, Oracle Internal Controls Manager (Oracle ICM) works in conjunction with Oracle Workflow to monitor internal business processes and ensure they are performed as designed.

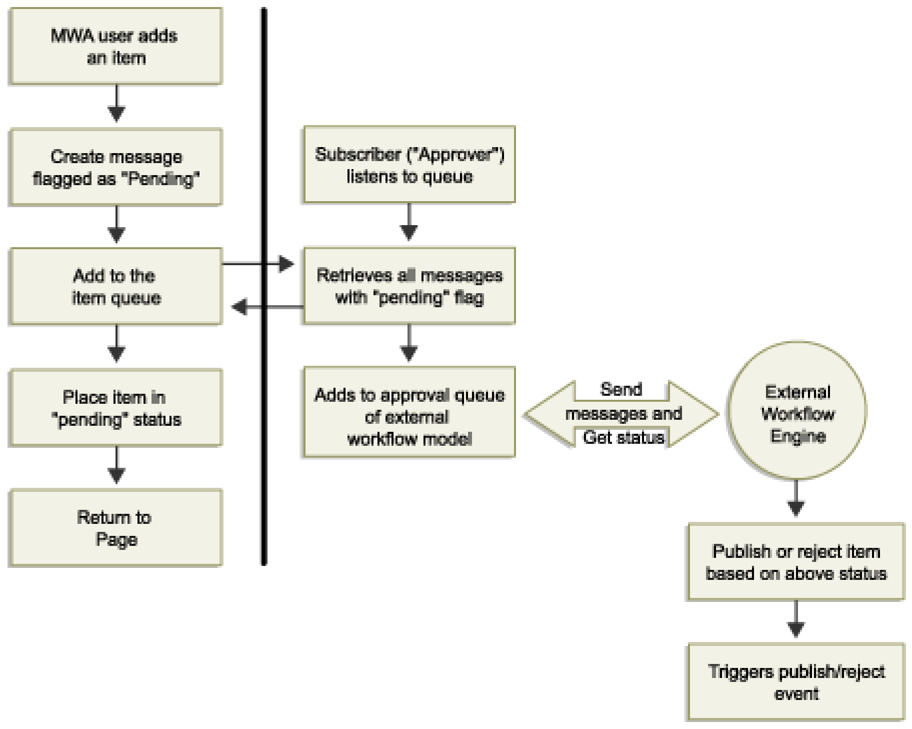
In general, there are five steps involved in integrating Oracle Workflow with the Oracle Portal content management system for content approval, and these steps are as follows:

1. Enable approvals and notifications in Oracle Portal.
2. Grant users the *Manage Items With Approval* privileges.
3. Create a portal user that will be used by the Oracle Workflow process to either approve or reject items and add that user to the portal approval process.
4. Register the workflow process with Oracle Workflow. This workflow process calls your compliance process, or it may perform your actual compliance process.
5. Create subscriber code and add the subscriber to WWSBR_EVENT_Q to process CMEF events. This subscriber initializes the workflow engine and then calls the workflow process.

We can see how this works by modifying the item validation example described in Section 16.7, "Example: Item Validation" to call a workflow process to perform the item validation.

Let's take a look at how this process would work, based on the diagram in Figure 16–7.

Figure 16–7 Process Flow for Workflow Integration with Oracle Portal


MWA user adds an item
Create message flagged as "Pending"
Add to the item queue
Place item in "pending" status
Return to Page
Subscriber ("Approver") listens to queue
Retrieves all messages with "pending" flag
Adds to approval queue of external workflow model
Send messages and Get status
External Workflow Engine
Publish or reject item based on above status
Triggers publish/reject event

16.8.2 Example Overview

Figure 16–7 shows the process flow diagram for using CMEF to integrate Oracle Workflow with the Oracle Portal content management system. When a user with *Manage Items With Approval* privileges adds an item to the portal, that item is marked as pending. Just as with the ITEM_VALIDATION subscriber in Example 16–16, an Oracle Streams Advanced Queuing subscriber is required to listen for events on the WWSBR_EVENT_Q queue (see Example 16–17). If the event is of type SUBMIT_ ITEM_FOR_APPROVAL, the subscriber launches the workflow engine process, WF_ CHECKURL, and passes the parameters for the portal item to the workflow process. The workflow process calls an external PL/SQL procedure (see Example 16–18) to perform its business logic, which in this case includes either approving or rejecting the portal item.

If the user specifies a URL, then Oracle Workflow approves the item using the `wwsbr_ api.approve` API. Otherwise, Oracle Workflow rejects the item using the `wwsbr_ api.reject` API.

This example assumes the following prerequisites:

- The CMEFSAMPLES schema exists in the same database as the portal schema.
- The Oracle OWF_MGR workflow schema is installed in the same database as the portal schema.
- The AQ_TM_PROCESSES parameter should be set to at least five. To check this, log in to the database as the SYS user and execute the following query:

```
select value
from   v$parameter
where  name = 'aq_tm_processes';
```

  If the value is less than five, then set the value as follows:

```
alter system set aq_tm_processes=5;
```

- Restart your database.
- Oracle Portal 11g has been installed.

16.8.3 Detailed Example Description

When a portal user with *Manage Items With Approval* privileges adds an item to a page, a SUBMIT_ITEM_FOR_APPROVAL event is added to the WWSBR_EVENT_Q queue. The item is marked as pending on the portal page. This user's items must be approved before they are made visible on the page.

The Streams Advanced Queuing subscriber CMEF_WORKFLOW listens for events on WWSBR_EVENT_Q. If the event is of type SUBMIT_ITEM_FOR_APPROVAL, then it launches the workflow engine process and passes the parameters for the portal item to the workflow process.

The workflow process WF_CHECKURL performs its business logic, which in this case includes either approving or rejecting portal items.WF_ CHECKURL uses the views and APIs to approve or reject the item. If it approves the item, then an APPROVE_ ITEM event is added to WWSBR_EVENT_Q. If it rejects the item, then a REJECT_ ITEM event is added to WWSBR_EVENT_Q.

When the user refreshes the page, he or she will either see the item published on the page (if it was approved), or removed (if it was rejected).

The following six steps occur within this example:

1. Section 16.8.3.1, "Enable Approvals and Notifications in Oracle Portal"
2. Section 16.8.3.2, "Grant Users the Manage Items With Approval Privileges"
3. Section 16.8.3.3, "Run Scripts Required for the CMEF Workflow Integration Example"
4. Section 16.8.3.4, "Create Subscriber and Check Procedures"
5. Section 16.8.3.5, "Register the WF_CHECKURL Process with Oracle Workflow"
6. Section 16.8.3.6, "Add the CMEF_WORKFLOW Subscriber to the WWSBR_ EVENT_Q Queue"

Each of these steps is described in more detail in the following sections.

16.8.3.1 Enable Approvals and Notifications in Oracle Portal

The first thing you need to do is enable approvals and notifications in the page group.

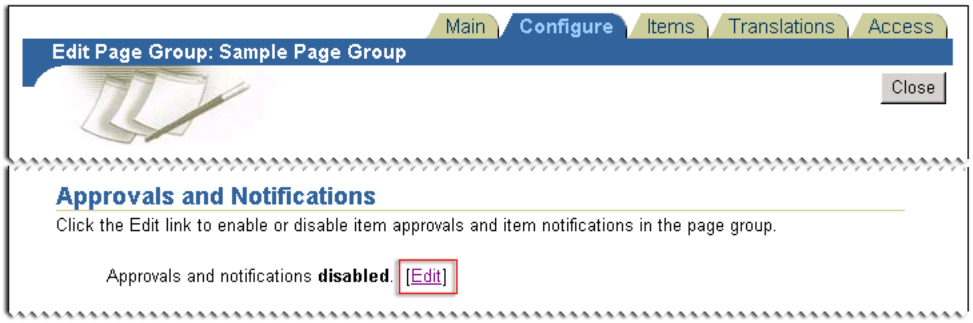
To enable approvals and notifications, perform the following steps:

1. Go to any page of the page group and switch to Edit mode.
2. In the toolbar at the top of the page, click the **Properties** link next to Page Group.

> **Note:** Make sure you click the link next to Page Group and not the one next to Page (Figure 16–2).

3. Click the **Configure** tab to bring it forward.
4. In the Approvals and Notifications section you can see whether approvals and notifications are enabled or disabled. If you want to change this setting, click the **Edit** link (Figure 16–8).

Figure 16–8 Status of Approvals and Notifications for a Page Group



Edit Page Group: Sample Page Group

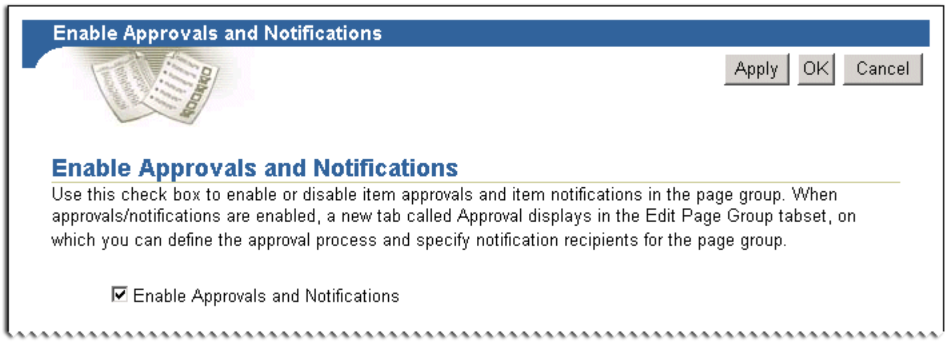
Approvals and Notifications

Click the Edit link to enable or disable item approvals and item notifications in the page group.

Approvals and notifications **disabled**. [Edit]

5. To enable approvals and notifications, select the **Enable Approvals and Notifications** check box (Figure 16–9).

Figure 16–9 Enabling or Disabling Approvals and Notifications for a Page Group



Enable Approvals and Notifications

Enable Approvals and Notifications

Use this check box to enable or disable item approvals and item notifications in the page group. When approvals/notifications are enabled, a new tab called Approval displays in the Edit Page Group tabset, on which you can define the approval process and specify notification recipients for the page group.

☑ Enable Approvals and Notifications

6. Click **OK** to save your changes.
7. Click **Close** to return to the page.

16.8.3.2 Grant Users the Manage Items With Approval Privileges

---

Note: You can skip this step if your external workflow process does not require the approval or rejection of portal items.

The next step is to grant *Manage Items With Approval* privileges to all users who need approval for their items. For the purposes of this example, let's specify that all users require approval for their items.

To specify that all users require approval for their items, perform the following steps:

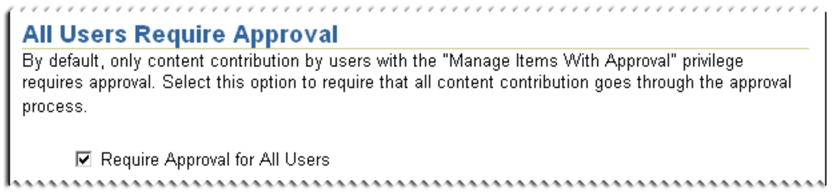
1. Go to any page in the page group and switch to Edit mode.
2. In the toolbar at the top of the page, click the **Properties** link next to Page Group.

Note: Make sure you click the link next to Page Group and not the one next to Page (Figure 16–2).

---

3. Click the **Approval** tab to bring it forward.
4. Select the **Require Approval for All Users** check box (Figure 16–10).

Figure 16–10   Specifying That All Users Require Approval for Their Items



All Users Require Approval

By default, only content contribution by users with the "Manage Items With Approval" privilege requires approval. Select this option to require that all content contribution goes through the approval process.

☑ Require Approval for All Users

5. Click **OK**.

16.8.3.3 Run Scripts Required for the CMEF Workflow Integration Example

Before you proceed to the next step, there are several tasks that you need to perform. These steps are as follows:

1. Log in to the SYS schema and run the following command:

   ```
   GRANT EXECUTE ON owf_mgr.wf_engine to cmefsamples;
   ```

2. Log in to the portal schema and run `provsyns.sql`. When prompted for the schema/provider name enter `owf_mgr`.
3. Log in to the CMEFSAMPLES schema and run the following:

   ```
   drop sequence WF_SEQ

   create sequence WF_SEQ
   increment by 1
   start with 1
   maxvalue 1000000000
   minvalue 1
   cache 20

   grant select on wf_seq to owf_mgr;
   ```

   This creates the sequence required by workflow, and allows the OWF_MGR workflow schema SELECT privileges on the sequence.
4. Create the synonym WF_ENGINE to the workflow schema:

   ```
   create synonym wf_engine for owf_mgr.wf_engine;
   ```

16.8.3.4 Create Subscriber and Check Procedures

You need to create a subscriber that listens to the WWSBR_EVENT_Q queue, waiting until a user adds an item that requires approval. The subscriber calls the WF_CHECKURL workflow process to perform the item validation and to send the e-mail notification. Example 16–17 shows the WORKFLOW_APPROVAL subscriber code.

Example 16–17   The WORKFLOW_APPROVAL Subscriber

```
create or replace procedure workflow_approval is
  MIME_TYPE_TEXT      constant varchar2(30) := 'text/plain';
  agent_list          dbms_aq.aq$_agent_list_t;
  wait_time           integer               := 30;
  begin_time          pls_integer;
  end_time            pls_integer;
  agent_w_message     aq$_agent;
  dequeue_options     dbms_aq.dequeue_options_t;
  message_properties  dbms_aq.message_properties_t;
```

```
  message_handle      raw(16);
  message             portal.wwsbr_event;
  l_subscriber        varchar2(30)          := 'CMEF_WORKFLOW';
  l_queue             varchar2(30)          := 'PORTAL.WWSBR_EVENT_Q';
  l_mode              binary_integer        := dbms_aq.REMOVE;
  l_attribute_site_id number                := 0;
  l_event             varchar2(30);
  l_doc               wwdoc_api.document_record;
  l_desc              varchar2(4000);
  l_itemkey           varchar2(100);
  l_job_nr            number;
  l_itemtype          varchar2(100)         := 'WF';
  l_wf_process        varchar2(100)         := 'WF_CHECKURL';
  l_is_indirect       boolean               := TRUE;
  l_portal_user_name  varchar2(30)          := 'portal';
  l_portal_password   varchar2(30)          := '<portal password>';
  l_url               varchar2(4000);       := '';

begin
  begin_time := dbms_utility.get_time;
  agent_list(1) := aq$_agent(l_subscriber, l_queue, null);
  loop
    -- Wait for messages.
    dbms_aq.listen(
      agent_list => agent_list,
      wait       => wait_time,
      agent      => agent_w_message
    );
    if (agent_w_message.name = l_subscriber) then
      dequeue_options.wait          := dbms_aq.NO_WAIT;
      dequeue_options.consumer_name := l_subscriber;
      dequeue_options.navigation    := dbms_aq.FIRST_MESSAGE;
      dequeue_options.dequeue_mode  := l_mode;
      dbms_aq.dequeue(
        queue_name         => l_queue,
        dequeue_options    => dequeue_options,
        message_properties => message_properties,
        payload            => message,
        msgid              => message_handle
      );
      -- If the event is an ITEM INSERT event and marked for later publication.
        if ((message.action       = 'SUBMIT_ITEM_FOR_APPROVAL') and
            (message.object_class = 'ITEM') and
            (message.raw_event    = 'INSERT') and
            (message.state        = 'NOT_PUBLISHED') then
          portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);
          -- Get the URL property for the object.
          select url
          into l_url
          from portal.wwsbr_all_items
          where id = message.object_id
          and caid = message.object_site_id;
          -- Get the nextval of the workflow sequence for the itemkey.
          select wf_seq.nextval
          into l_job_nr
          from dual;
          l_itemkey := lpad(to_char(l_job_nr), 5, '0');
          -- Launch the workflow engine process and pass the paramters for
          -- the portal item to the workflow process.
          wf_engine.createprocess(l_itemtype, l_itemkey, l_wf_process);
          -- Set the value for the portal URL.
          wf_engine.setitemattrtext(l_itemtype, l_itemkey, 'PORTAL_URL', l__url);
          -- Set the value for the portal item ID.
          wf_engine.setitemattrnumber(l_itemtype, l_itemkey, 'PORTAL_ITEM_ID',
message.object_id);
          -- Set the value for the portal page group ID.
          wf_engine.setitemattrnumber(l_itemtype, l_itemkey, 'PORTAL_SITE_ID',
message.object_site_id);
          -- Start the workflow process.
          wf_engine.startprocess(l_itemtype, l_itemkey);
        end if;
    end if;
    end_time := dbms_utility.get_time;
    if (end_time - begin_time) > 3000 then
      exit;
    end if;
  end loop;
exception
  when OTHERS then
    dbms_output.put_line(sqlerrm);
end workflow_approval;
/
```

You also need to create a procedure to check whether the user specified a file name for the item to determine whether or not it can be approved. The workflow process delegates the actual validation and approval of the item to this CHECK_URL procedure. Example 16–18 shows how to create the CHECK_URL procedure.

Example 16–18 The CHECK_URL Procedure

```
create or replace procedure check_url(
  itemtype  in  varchar2,
  itemkey   in  varchar2,
  actid     in  number,
  funcmode  in  varchar2,
  resultout out varchar2) is

  MIME_TYPE_TEXT      constant varchar2(30) := 'text/plain';
  agent_list          dbms_aq.aq$_agent_list_t;
  wait_time           integer               := 30;
  begin_time          pls_integer;
  end_time            pls_integer;
  agent_w_message     aq$_agent;
  dequeue_options     dbms_aq.dequeue_options_t;
  message_properties  dbms_aq.message_properties_t;
  message_handle      RAW(16);
  message             portal.wwsbr_event;
  l_subscriber        varchar2(30)      := 'CMEF_WORKFLOW';
  l_queue             varchar2(30)      := 'PORTAL.WWSBR_EVENT_Q';
  l_mode              BINARY_INTEGER    := dbms_aq.REMOVE;
  l_attribute_site_id number            := 0;
  l_event             varchar2(30);
  l_doc               wwdoc_api.document_record;
  l_desc              varchar2(4000);
  l_itemtype          varchar2(100)    := 'WF';
  l_itemkey           varchar2(100);
  l_url               varchar2(100);
  l_siteid            number;
  l_itemid            number;
  l_ignore            boolean          := FALSE;
  l_is_indirect       boolean          := TRUE;
  l_portal_user_name  varchar2(30)     := 'portal';
  l_portal_password   varchar2(30)     := '<portal password>');
begin
  -- Get the values for the attributes stored in the procedure.
  l_url := wf_engine.getitemattrtext(itemtype,itemkey,'PORTAL_URL',l_ignore);
  l_itemid := wf_engine.getitemattrnumber(itemtype,itemkey,'PORTAL_ITEM_ID',l_
ignore);
  l_siteid := wf_engine.getitemattrnumber(itemtype,itemkey,'PORTAL_SITE_ID',l_
ignore);
  -- Set the portal context.
  portal.wwctx_api.set_context(l_portal_user_name, l_portal_password);
  -- If the item type is a URL item and no URL is specified.
  if l_url is null then
    begin
      -- Update the item to indicate that the URL needs to be specified.
      l_desc := '<font color=RED>NO URL SPECIFIED</font>';
      -- Reset the CMEF global variables.
      wwsbr_api.clear_cmef_context;
      wwsbr_api.set_attribute(
        p_site_id           => l_siteid,
        p_thing_id          => l_itemid,
        p_attribute_site_id => l_attribute_site_id,
        p_attribute_id      => wwsbr_api.attribute_description,
        p_attribute_value   => l_desc
        p_log_cmef_message  => FALSE
      );
      -- Reject the item.
      wwsbr_api.reject(
        p_item_id     => l_itemid,
        p_site_id     => l_siteid,
        p_is_indirect => l_is_indirect,
        p_comment     => 'Rejected'
      );
      -- Reset the CMEF global variables.
      wwsbr_api.clear_cmef_context;
    end;
  else
    -- Approve the item.
    begin
      wwsbr_api.approve(
        p_item_id => l_itemid,
        p_site_id => l_siteid,
        p_comment => 'Approved'
      );
      -- Reset the CMEF global variables.
      wwsbr_api.clear_cmef_context;
    end;
  end if;
  commit;
  -- Process cache invalidation messages.
  wwpro_api_invalidation.execute_cache_invalidation;
exception
  when OTHERS then
    dbms_output.put_line(sqlerrm);
end check_url;
/
```

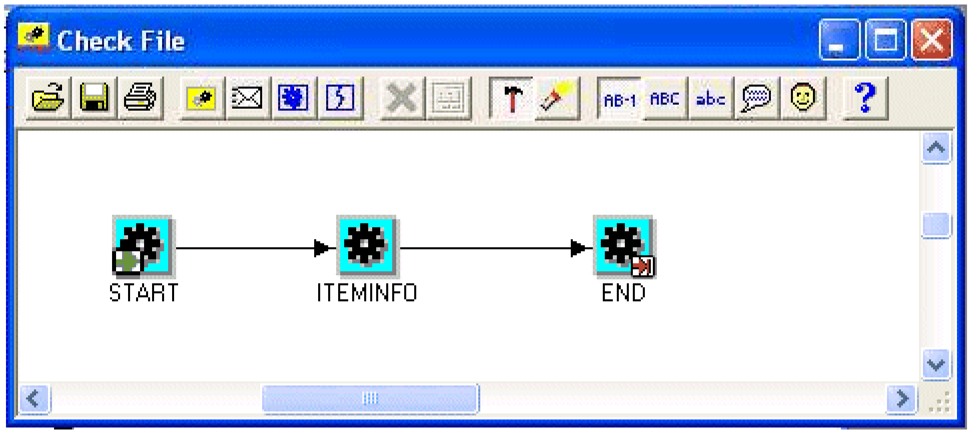
Note: CMEFSAMPLES will need to grant execute on WORKFLOW_APPROVAL and CHECK_URL to OWF_MGR:

```
grant execute on workflow_approval to owf_mgr;
grant execute on check_filename to owf_mgr;
```

16.8.3.5 Register the WF_CHECKURL Process with Oracle Workflow

The WF_CHECKURL process actually approves or rejects the portal item and sends the e-mail notification. You need to register this process with Oracle Workflow using the Oracle Workflow Builder (Figure 16–11).

Figure 16–11 The CHECK_FILE Oracle Workflow Process



Use the following steps to install the WF_CHECKURL process into the OWF_MGR schema:

1. Open the Oracle Workflow Builder.

   Tip: After installing the Workflow Builder, you may have to run it from the command line because you will have to set the ORACLE_HOME environment variable to that of the Workflow Builder. For example:

   ```
   set ORACLE_HOME=c:\oraclewf
   ```

2. Start a new workflow project.
3. Save the workflow project as `wf_checkurl.wft`.
4. Close the Oracle Workflow Builder.
5. Open `wf_checkurl.wft` in a text editor, for example, Notepad.
6. Copy and paste the WF_CHECKURL code into the file. You can download this code from Portal Center:

   `http://www.oracle.com/technology/products/ias/portal/files/cm_overview_10g1014_cmef_samples.zip`

7. Save the file and close the text editor.
8. Open `wf_checkurl.wft` in Oracle Workflow Builder.
9. Save `wf_checkurl.wft` into your portal database using the OWF_MGR schema.

Tip: If you do not know the OWF_MGR password, see MetaLink Note 198800.1.

16.8.3.6 Add the CMEF_WORKFLOW Subscriber to the WWSBR_EVENT_Q Queue

Next, you need to add the CMEF_WORKFLOW subscriber to the WWSBR_EVENT_Q queue so that it can process the CMEF events and trigger the WF_CHECKURL Oracle Workflow process, as shown in Example 16–19.

Example 16–19 Adding the CMEF_WORKFLOW Subscriber to WWSBR_EVENT_Q

```
declare
  subscriber sys.aq$_agent;
begin
  1_subscriber := sys.aq$_agent('CMEF_WORKFLOW', null, null);
  dbms_aqadm.add_subscriber(
    queue_name => 'portal.wwsbr_event_q',
    1_subscriber => subscriber
  );
end;
/
```

Now, when a user with *Manage Items With Approval* privileges add an item to a page, it is marked as Pending and a single INSERT event occurs, with a state of NOT_PUBLISHED:

Action	Event	State	Object Class
SUBMIT_ITEM_FOR_APPROVAL	INSERT	NOT_PUBLISHED	ITEM

The CMEF_WORKFLOW_APPROVAL subscriber responds to this action and invokes the PORTAL_ITEM_WORKFLOW Oracle Workflow process.

The PORTAL_ITEM_APPROVAL Oracle Workflow process performs its business logic and calls the Oracle Portal `approve` or `reject` API to approve or reject the item. It also sends an e-mail notification to the specified e-mail ID indicating the approval status.

Part IV

Appendixes

Part IV contains the following appendixes:

A

Creating Portlets with the Portlet Builder

You can find information on using Portlet Builder in Appendix A "Creating Portlets with the Portlet Builder" of the *Oracle Fusion Middleware Developer's Guide for Oracle Portal Release 2 (10.1.4)* in the Oracle Application Server Release 2 (10.1.2.0.2) library located on the Oracle Technology Network (`http://www.oracle.com/technology/documentation/appserver1012.html`).

B Troubleshooting Portlets and Providers

This appendix describes common problems that you might encounter when using Oracle Portal and explains how to solve them. It contains the following topics:

- Section B.1, "Diagnosing General Portlet Problems"
- Section B.2, "Diagnosing Java Portlet Problems"
- Section B.3, "Diagnosing OmniPortlet Problems"
- Section B.4, "Diagnosing Web Clipping Problems"
- Section B.5, "Need More Help?"

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 Metadata Repository Oracle home:

- MW_HOME, represents the full path of the Oracle Fusion Middleware home.
- ORACLE_HOME, represents the full path of the middle-tier Oracle home.
- ORACLE_INSTANCE, represents the full path of the instance home associated with 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 Oracle Infrastructure home containing the Oracle Metadata Repository.

B.1 Diagnosing General Portlet Problems

This section describes common problems and solutions for all portlets. It contains the following topic:

- Section B.1.1, "Portlet Refresh Failure"
- Section B.1.2, "HTML Tags Appearing in Portlet"

B.1.1 Portlet Refresh Failure

When using the JavaScript `document.write()` method in conjunction with a Mozilla browser, you may find that portlet refresh does not work.

Problem

When using a Mozilla browser, such as Firefox, the `document.write()` method causes portlet refresh to no longer work as expected.

Solution

Avoid using `document.write()` in your output if you also want portlet refresh to work on Firefox. If this is not possible, you can choose to have the portlet not display the Refresh and Restore icons during rendering. You can also turn off portlet refresh and restore at the region level.

B.1.2 HTML Tags Appearing in Portlet

After upgrading from an earlier release to 11*g* Release 1 (11.1.1), you may find unformatted HTML appearing in your portlets, for example, in the portlet title.

Problem

When viewing a portlet, you see raw HTML. For example, you might see the following in a portlet title:

```
<b>Oracle Application Server 10<i>g</i> Collateral</b>
```

when what you expected to see was:

Oracle Application Server 10*g* Collateral

To protect against cross site scripting (XSS) attacks, Oracle Portal Release 10.1.4 escapes all user input by default. Thus, some HTML that was formatted in earlier releases now appears unformatted because it has been escaped to plain text.

Solution

You can resolve this issue in one of the following two ways:

- Tell your users to stop entering HTML tags as part of their input and have them enter only plain text.
- Turn on compatibility mode to make your Oracle Portal instance behave the way it did in releases prior to Release 10.1.4, that is, stop the default escaping of HTML to plain text. Note that turning on compatibility mode makes your portal instance less secure.

For more information on these options, refer to *Oracle Fusion Middleware Upgrade Guide for Oracle Portal, Forms, Reports, and Discoverer*.

B.2 Diagnosing Java Portlet Problems

This section describes common problems and solutions for Java portlets. It contains the following topics:

- Section B.2.1, "Portlet Logging"
- Section B.2.2, "Installation and Deployment Problems"
- Section B.2.3, "Portlet Code Does Not Compile"

- Section B.2.4, "Application Server Connection Test Fails"
- Section B.2.5, "Provider Test Page Shows Error"
- Section B.2.7, "Portlet Does Not Display on Page"
- Section B.2.8, "After Initial Successful Display, Portlet Does Not Display on Page"

B.2.1 Portlet Logging

In addition to specific issues listed in the other sections of this appendix, the following general techniques can help you to troubleshoot portlet problems.

Problem 1

When accessing a portal page, the portlet does not display or displays an error message.

Solution 1

Check the diagnostic log file look for errors.

For Oracle Portal:

```
MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\logs\WLS_
PORTAL-diagnostic.log
```

For a custom Web provider:

```
MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\logs\WLS_
PORTAL-diagnostic.log
```

Problem 2

Pertinent information that may help solve the problem is not being recorded in the log file.

Solution 2

Increase the provider's log level to produce more detailed logging information by adding the following entry to the `web.xml` file:

```
<env-entry>
  <env-entry-name>oracle/portal/provider/global/log/logLevel</env-entry-name>
  <env-entry-value>6</env-entry-value>
  <env-entry-type>java.lang.Integer</env-entry-type>
</env-entry>
```

B.2.2 Installation and Deployment Problems

This section describes problems that you may encounter when installing and deploying the PDK-Java framework and samples.

B.2.2.1 Cannot Find a Java Class Object

You receive an error message about a Java class that does not exist. For example:

```
An unexpected error occurred-29540 : class
  oracle/webdb/provider/web/HttpProviderDispatcher does not exist (WWC-43000)
```

Problem
The referenced Java class object is not in the database or it is invalid. Oracle Portal, not PDK-Java, generates the error message when it cannot execute the class.

Solution
Log in to the database as your main Oracle Portal schema. Execute a SELECT statement to verify that the object is in the schema. For example:

```
SELECT object_name, object_type, status FROM user_objects
  WHERE object_name like '%HttpProvider%'
```

Note that the name between quotes is case sensitive.

You should receive the following back:

```
/3334f18_HttpProviderDispatcher
```

Check for invalid or missing JAVA CLASS objects. Something could be wrong with the Oracle Portal installation. If so, then recompiling invalid objects might help resolve it.

B.2.2.2 Cannot Deploy the template.ear File
You receive the following error message when deploying your EAR file based on `template.ear`:

```
Invalid J2EE application file specified - Base Exception:
Cannot get xml document by parsing WEB-INF/web.xml in template.war: <Line 88,
Column 14> : '--' is not allowed in comments.
Resolution:
```

Problem
The `web.xml` file contains a syntax error.

Solution
Verify that the syntax of `web.xml` is correct. Some versions of `web.xml` shipped in `template.war` have a misplaced comment tag. Make sure that all opening comment tags, `<!--`, have a matching closing tag, `-->`, before starting a new comment tag.

B.2.2.3 Error When Attempting to Register Provider
A number of errors may occur when you attempt to register your provider. This section explains what can go wrong and how you can correct it. If your scenario is not covered in this section, then check the `application.log` of your provider or, if that does not help, use `logcfg.sql` for Portal Repository logging. For more information `logcfg.sql` and `application.log`, refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Problem 1
You receive the following error message:

```
Internal error (WWC-00006)
The provider URL specified may be wrong or the provider is not running.
(WWC-43176)
```

When OracleAS Portal attempts to register your provider, it must contact the listener that serves your Web portlets. If you have already accessed the provider's test page, `http://`*`host.domain`*`:`*`port`*`/`*`context`*`/providers/`*`servicename`*, according to the installation instructions, Oracle Portal may not recognize your machine.

Solution 1

Make sure the machine where Oracle Portal resides can access your Web provider listener. The easiest way to test this setup is to start a browser on the host of the Oracle Portal repository database and try to access the provider's test page. If the browser needs a proxy setting to reach the provider, then you should set the same proxy for Oracle Portal on the Administer tab. You should also use this proxy when registering the provider.

If your Web provider server is not part of the DNS, then add an entry in the Oracle Portal server's host file.

If a firewall exists between the machine with the Oracle Portal database and the machine with the provider, then make sure that the necessary port is open.

Problem 2

You receive the following error:

```
Internal error (WWC-00006)
The provider URL specified may be wrong or the provider is not running.
(WWC-43176)
The following error occurred during the call to Web provider: Can't read
deployment properties for service: _default (WWC-43147)
```

When you register a provider, you have the option of specifying a Service Id. If you do not specify anything in the Service Id field, the registration URL will not contain the service name part, `http://`*`host.domain`*`:`*`port`*`/`*`context`*`/providers/`. Therefore, Oracle Portal is looking for a default service, which is stored in a deployment file called `_default.properties`. When this file is not found, you receive this error.

Solution 2

Make sure that you bundle a file called `_default.properties` into your WAR file with the following path:

```
/Web-inf/deployment/
```

The `_default.properties` file should define the default service to use when none is specified for the provider. The file should look similar to the following:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/test/provider.xml
autoReload=true
testPageURI=/htdocs/testpage/TestPage.jsp
```

Problem 3

You receive the following error message:

```
Internal error (WWC-00006)
The provider URL specified may be wrong or the provider is not running.
(WWC-43176)
The following error occurred during the call to Web provider: Can't read
deployment properties for service: test (WWC-43147)
```

Oracle Portal cannot find the deployment properties file for the service you specified in the Service Id field or the service name part of the registration URL, `http://`*`host.domain`*`:`*`port`*`/`*`context`*`/providers/`*`servicename`*. For example, suppose you received this error in response to your attempt to register your provider with the following URL:

```
http://host.domain:port/context/providers/test
```

In this case, you probably do not have a file called `test.properties` in your WAR file in `/Web-inf/deployment/`.

Solution 3

Make sure that you bundle a deployment properties file (for example, `test.properties`) into your WAR file with the following path:

```
/Web-inf/deployment/
```

The file should look similar to the following:

```
serviceClass=oracle.webdb.provider.v2.adapter.soapV1.ProviderAdapter
loaderClass=oracle.portal.provider.v2.http.DefaultProviderLoader
showTestPage=true
definition=providers/test/provider.xml
autoReload=true
testPageURI=/htdocs/testpage/TestPage.jsp
```

Problem 4

You receive the following error message:

```
An error occurred when attempting to call the providers register function.
(WWC-43134)
The provider URL specified may be wrong or the provider is not running.
(WWC-43176)
The following error occurred during the call to Web provider: Class
oracle.portal.provider.v2.render.RenderManager has no set or add method for tag
"createdOn" (WWC-43147)
```

Solution 4

If you are using `DefaultProvider`, `provider.xml` includes the element `<createdOn>`. When the XML is parsed, the initialization process looks for a method called `setCreatedOn` in the class representing the containing element. For example:

```
<renderer class="my.local.Renderer">
<createdOn>12-Mar-2001</createdOn>
...
</renderer>
```

In this example, the initialization process looks for `my.local.Renderer.setCreatedOn()`. In your case, the appropriate method does not exist, causing this error.

Problem 5

You receive an error like the following:

```
(WWC-00006)
An unexpected error occurred: User-Defined Exception
(WWC-43000)
wwpro_api_provider_registry.register_provider
An unexpected error occurred: ava.sql.SQLException: Inserted value too large for
```

```
column:
```

This error indicates that your portlet has exceeded the data limit for a portlet. Each portlet is limited to 4K of data. The lengths of all of the following contribute toward this data limit:

- provider name, display name, and description
- parameter name, display name, and description
- event name, display name, and description

Solution 5

You can take the following two actions to reduce the amount of data used for the portlet:

- Reduce the length of the portlet's parameter and event names, display names, and descriptions.
- Reduce the number of parameters and events in your portlet

B.2.2.4 Error Adding a Portlet to a Provider

You added a portlet to your provider, but, when you view the provider in Oracle Portal, you do not see the new portlet.

Problem

The most common problem is a syntactical error in `provider.xml`, but it could also be that the updated `provider.xml` file is not being found for some reason.

Solution

Try the following:

- Examine `provider.xml`. Ensure that you have two syntactically correct opening and closing portlet tags within the provider tag. For example:

  ```
  <provider class="oracle.portal.provider.v2.DefaultProviderDefinition"
     session="true">
    <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition"
      version="1">
      <id>1</id>
      ...
    </portlet>
    <portlet class="oracle.portal.provider.v2.DefaultPortletDefinition"
      version= "1">
      <id>2</id>
      ...
    </portlet>
  </provider>
  ```

- Check the test page for errors. If no errors are found, then refresh the provider.
- Check the `application.log` of the provider for errors.

B.2.2.5 Portlet Does Not Exist

You received the following error message:

```
Error: The listener returned the following Message: 500 Portlet with id $1 doesn't
 exist
```

Problem

You removed a portlet from `provider.xml` without first removing the portlet from the page. The portlet no longer appears on the Customization page and therefore you cannot remove it from the page.

Solution

Delete the region that contains the portlet. Remember, when deleting a portlet from `provider.xml`, delete the portlet from the page first.

B.2.2.6 File Not Found

You receive the following error message:

```
Request URI:/jpdk/snoop/snoopcustom.jsp Exception: javax.servlet.ServletException:
java.io.FileNotFoundException:
C:\iAS\Apache\Apache\htdocs\jpdk\snoop\snoopcustom.jsp (The system cannot find the
path specified)
```

Problem

You added a JSP portlet to your page. Oracle Portal cannot locate the JSP in the file system through `provider.xml`.

Solution

Confirm the file name and location and verify the information within `provider.xml`.

B.2.2.7 XML Parser Error

The XML Parser may fail at times when parsing your provider.xml. This section explains what can go wrong and how you can correct it.

Problem 1

You receive the following error message:

```
Error: The XML parser encountered an Error, and could not complete the conversion
for portlet id=150, it returned the following message: XML Parsing Error
```

You attempted to register your provider or display portlets on a page, and received this error message. When altering `provider.xml`, one or more of the tags were corrupt and the file cannot be parsed.

Solution 1

Review `provider.xml` for errors.

Problem 2

You receive the following error message:

```
Error: The XML parser encountered an Error, and could not complete the conversion
for portlet id=146, it returned the following message: XSL reference value missing
in XML document.
```

You attempt to add your portlet to a page or register your provider, and you receive this error. The portlet information within `provider.xml` may be incorrect. Another possibility is that your classes cannot be located.

Solution 2

Check that no tags are missing and that they appear in the correct sequence. Verify that your class files exist and are specified correctly within `provider.xml`.

B.2.2.8 Error Adding Portlets

You receive the following error message when trying to add portlets to a page:

```
Error: An unexpected error occurred: ORA-29532: Java call terminated by uncaught
Java exception: Attribute value should start with quote.
(WWC-43000) An unexpected error occurred: Attribute value should start with quote.
at oracle.xml.parser.v2.XMLError.flushErrors(XMLError.java:205)
```

Problem

Oracle Portal has an issue with `provider.xml`. Most likely, an attribute within the file is corrupt.

Solution

Review `provider.xml` for errors.

B.2.2.9 Content Request Timed Out

You receive the following error message:

```
ERROR TIMEOUT FOR CONTENT=timeout
```

Problem

You successfully registered your Web provider. When adding portlets to a page or refreshing a page with existing portlets, you receive the timeout error message. This error indicates that Oracle Portal timed out trying to contact your Java portlet listener.

Solution

Verify that you can still access the sample page:

```
http://host:port/context/providers/servicename
```

You set the timeout period by adding the following `init` parameters to the page servlet of Oracle Portal's `appConfig.xml` at:

```
MW_HOME\user_projects\domains\<DomainName>\config\fmwconfig\servers\WLS_
PORTAL\applications\portal\configuration
```

Note that this `appConfig.xml` file is for the Parallel Page Engine. For example:

```
<requesttime>1000</requesttime>
<minTimeout>100</minTimeout>
<stall>500</stall>
```

For more information about `appConfig.xml` and its configuration parameters, refer to *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

You may also set a timeout value for individual providers when you register them with Oracle Portal. Note that the provider timeout value must match or exceed the `minTimeout` parameter value in `appConfig.xml` in order to have any effect. For information about registering PDK-Java providers, refer to Section 6.5.5, "Registering and Viewing Your Oracle PDK-Java Portlet". For information about registering WSRP providers, refer to Section 6.3.4, "Registering and Viewing Your JSR 168 Portlet."

B.2.2.10 Message 500 Returned

You receive the following error message:

```
Error: The listener returned the following Message: 500
```

Problem

This generic message displays when Oracle Portal receives an Internal Server Error while attempting to display your portlet. This problem might have any one of several causes.

Solution

Review the `WLS_PORTAL-diagnostic.log` file located at `MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\logs` to discover the cause of the error.

B.2.2.11 JPS Portlets with the get Method not Working

HTML forms can be submitted using either the `get` or `post` method, but the WSRP standard only requires the consumer (portal) to use the `post` method. Support of the `get` method is optional according to the standard. Since portal consumers are not required to support the `get` method, we highly recommended that you use the `post` method when developing your portlets.

Problem

You have JPS portlets that use the `get` method and they are not working correctly in Oracle Portal. Forms with `get` lose the query string in the action URL when the form is submitted. The query string is needed for the portlets to return back to the portal and for the WSRP state.

Solution

Replace the `get` method with the `post` method.

B.2.2.12 Portlet Displays Session Expired Message After Redeployment

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, it may receive an error message the first time it tries to contact the producer after redeployment.

Problem

You receive an error message in your portlet the first time you access it after redeployment.

```
Error: Could not get markup. The cookie or session is invalid or there is a
runtime exception.
```

Solution

To reestablish the producer's session, refresh the portal 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.

B.2.3 Portlet Code Does Not Compile

When you try to compile the code for your portlet, you receive a compilation error.

Problem

The Portlet Development library is not selected.

Solution

Make sure that Portlet Development library is selected for the project. Edit your project's properties and select the **Profiles > Development > Libraries** entry in the pane on the right. Make sure that the Portlet Development library is listed under **Selected Libraries**.

B.2.4 Application Server Connection Test Fails

In Oracle JDeveloper, the Application Server Connection Test fails with the message `Connection refused: connect`.

Problem 1

WebLogic Server is not running.

Solution 1

Make sure WebLogic Server is up and running by typing the following URL in your browser: `http://`*`yourhostyourdomain`*`:7001`

Problem 2

The connection information is incorrect.

Solution 2

Verify the connection information that you provided in the Connection Setup wizard.

B.2.5 Provider Test Page Shows Error

When accessing the provider test page with your browser, an error is shown.

Problem 1

The `provider.xml` syntax is incorrect.

Solution 1

More On OTN

Correct the `provider.xml` syntax. Refer to the *PDK-Java XML Provider Definition Tag Reference* document on Portal Studio:

`http://www.oracle.com/technology/products/ias/portal/html/javadoc/xml_tag_reference_v2.html`

Problem 2

Needed JAR files are missing from the deployment environment.

Solution 2

Search the JAR files for the missing class. When the missing JAR file is identified, you can add to the `lib` directory of the application from the Oracle JDeveloper Deployment Profile. Refer to the *Oracle JDeveloper Online Help System* for more information about the deployment profile.

B.2.6 Web Provider Not Appearing in Portlet Repository

Users cannot see a Web provider, such as Omniportlet, in the Portlet Repository even though it has been registered successfully and is visible in the navigator.

Problem

The Web provider uses `DBPreferenceStore` and the database information stored in `data-sources.xml` is incorrect.

Solution

Correct the information in `data-sources.xml`. Refresh the provider and invalidate the Portlet Repository cache.

B.2.7 Portlet Does Not Display on Page

When you access a portal page, the portlet does not display on the page.

Problem 1

The provider is not running.

Solution 1

Make sure that the provider is up and running by entering the provider registration URL in your browser's address bar.

Problem 2

The security manager for the provider is preventing the portlet from displaying.

Solution 2

In the provider definition file, `provider.xml`, delete or comment out the security manager, if any.

Problem 3

The user may not have the Execute privilege for the portlet as defined on the portlet's Access page.

Solution 3

Check the privileges for the portlet and, if the user or a group the user belongs to does not have the Execute privilege, grant them access. Portlets sometimes inherit their access privileges from the provider. You can use the Navigator in Oracle Portal to find the portlet's provider and check its access privileges, then drill down to the portlet to see its access privileges.

Problem 4

You get an error trying to display the portlet after you redeployed it.

Solution 4

Refer to Section B.2.2.12, "Portlet Displays Session Expired Message After Redeployment".

B.2.8 After Initial Successful Display, Portlet Does Not Display on Page

When you access a portal page, the portlet initially displays on the page but returns error messages in subsequent display attempts.

Problem

The provider session information is incorrect.

Solution

Check whether or not the provider uses sessions. If it does, edit the provider registration information to make sure that you registered it accordingly:

```
Login Frequency: Once per User Sessions
```

You should also confirm that `<session>` is set to true in `provider.xml`.

B.3 Diagnosing OmniPortlet Problems

This section provides information to help you troubleshoot problems you may encounter while using OmniPortlet.

To view errors that occur during the execution of OmniPortlet, perform the following:

- Open the `WLS_PORTAL.log` file:

  ```
  MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\logs
  ```

- Display the HTML source (for example, in Microsoft Internet Explorer browser, choose **View > Source**), and locate the errors embedded in the output HTML as comments.

To alter the logging level of the OmniPortlet Provider, perform the following action:

- Open the `web.xml` file and modify the `context-param` value of `oracle.portal.log.LogLevel` to the possible values ranging from `1` to `8` (where `8` means `debug`). The `web.xml` file is located at:

  ```
  MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\tmp\_WL_
  user\portalTools_11.1.1.1.0\kjdcke\war\WEB-INF\web.xml
  ```

The OmniPortlet errors that you are most likely to encounter, and possible solutions, are as follows:

- Section B.3.1, "OmniPortlet Cannot Access the Specified URL"
- Section B.3.2, "Portlet Content Is Not Refreshed"

B.3.1 OmniPortlet Cannot Access the Specified URL

Your OmniPortlet displays errors or does not display the correct content.

Problem 1

The URL is not active.

Solution 1

Type the URL directly in your browser address field to test its validity.

Problem 2

If a proxy server is required to reach the site, the proxy settings are not valid. The following messages may display:

```
Failed to open specified URL.
Cannot open the URL specified because of connection timeout.
```

Solution 2

Check that your proxy settings are valid by clicking **Edit** for the HTTP Proxy Setting in the OmniPortlet Provider Test Page.

Problem 3

The message `OmniPortlet timed out` displays in your OmniPortlet.

Solution 3

Perform either of the following tasks:

1. If a proxy server is required to reach the site, see "Solution 2".
2. If the URL request takes a long time to process (for example, it executes a long running query), try increasing the timeout value (in seconds) in the OmniPortlet `provider.xml` file in:

   ```
   MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\tmp\_WL_
   user\portalTools_11.1.1.1.0\1pwj8k\war\WEB-INF\providers\omniPortlet
   ```

   If you change the timeout value, you also need to do the following:

   - Bounce your middle-tier.
   - Increase the provider registration Timeout value of the Oracle Portal instances with which this provider is registered.

Problem 4

If HTTP authentication is required and the user name and password are missing or not valid, the following error message displays:

```
Authorization failed when connecting to the URL specified.
Provide correct user name and password to connect.
```

Solution 4

To configure the Secured Data (Web Clipping) Repository Setting, click **Edit** on the OmniPortlet Provider Test Page. On the Source tab, click **Edit Connection** and enter a valid user name and password on the Connection Information page. To enter the connection information, the Secured Data Repository must be already be configured. To configure its settings, click Edit on the OmniPortlet Provider Test Page.

Problem 5

If opening a URL to an HTTPS site with a certificate and the certificate is identified as not valid, the following error message displays in the portlet:

```
SSL handshake failed for HTTPS connection to the specified URL.
The certificate file needs to be augmented.
```

Solution 5

See the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Problem 6

If the proxy server requires authentication and the user name and password are missing or not valid, the following error message displays:

```
Invalid or missing user proxy login information.
```

Solution 6
Check that your proxy server user name and password are valid. Refer to Section B.4.5, "HTTP Error Code 407 When Clipping Outside Firewall" for more information.

B.3.2 Portlet Content Is Not Refreshed
After changing portlet properties in the Edit Defaults page, your portlet content is not refreshed.

Problem 1
Oracle Web Cache invalidation is not configured properly.

Solution 1
See Section I.2.1.3 "Configuring Caching (PDK Only)" in the Oracle Fusion Middleware Administrator's Guide for Oracle Portal.

Problem 2
You have personalized the portlet.

Solution 2
See Section B.3.3, "Edit Defaults Changes are Not Reflected in the Personalized Portlet".

B.3.3 Edit Defaults Changes are Not Reflected in the Personalized Portlet
When you personalize the portlet at runtime using the **Personalize** link, the new property values are not reflected in the personalized version of the portlet.

Problem
When you personalize the portlet, a complete copy of the personalization object file is created. Since all properties are duplicated, subsequent modifications through Edit Defaults are not reflected in the personalized version.

Solution
To ensure the latest changes are made to the portlet, click **Personalize** again (after the modifications using Edit Defaults), then select the **Reset to Defaults** option.

B.4 Diagnosing Web Clipping Problems
This section provides information to help you troubleshoot the following problems you may encounter while using the Web Clipping provider or Web Clipping Studio:

- Section B.4.1, "Setting Logging Levels"
- Section B.4.2, "Reviewing Error Messages"
- Section B.4.3, "Checking the Status of the Provider with the Test Page"
- Section B.4.4, "Problem Connecting to the Web Site for Clipping"
- Section B.4.5, "HTTP Error Code 407 When Clipping Outside Firewall"
- Section B.4.6, "Cannot Clip a Page"
- Section B.4.7, "Images Not Retrieved with Clipping"

- Section B.4.8, "Resolving Problems with Migration of URL-based Portlets"

B.4.1 Setting Logging Levels

By default, the logging level of Web Clipping is set to level 3, which provides information about configuration, severe errors, and warnings. This level is reasonable for day-to-day operations. To view information useful for debugging, you should set the logging level to 8.

To set the logging level, edit the `web.xml` file, which is located at:

```
MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\tmp\_WL_
user\portalTools_11.1.1.1.0\kjdcke\war\WEB-INF\web.xml
```

To set the level to 8 and display debugging information, set the value of the parameter `oracle.portal.log.LogLevel` as follows:

```
<context-param>
        <param-name>oracle.portal.log.LogLevel</param-name>
        <param-value>8</param-value>
</context-param>
```

After you make the change, restart the Web application.

B.4.2 Reviewing Error Messages

Errors that occur when accessing the Test Page or executing of the Web Clipping portlet are written to:

```
MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\logs\WLS_PORTAL.log
```

B.4.3 Checking the Status of the Provider with the Test Page

You can use the Web Clipping Provider Test Page to determine if the provider is functioning properly. To access the Test Page, click **Web Clipping Provider** from the Portal Tools Application Welcome Page, which is located at:

```
http://host:port/portalTools
```

The Provider Test Page: Web Clipping is displayed. It provides the following information:

- Portlet information about the Web Clipping portlet. The Web Clipping provider contains only one portlet.
- Provider initialization parameters and values.
- Provider status, with links to pages for editing the configuration.

For more information about using the Test Page, see the "Administering Web Clipping" appendix in the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

B.4.4 Problem Connecting to the Web Site for Clipping

You encounter difficulties making or maintaining connections to the Web site containing the clipping.

Problem 1

If you cannot access the Web site using Web Clipping Studio, you may be using an incorrect URL.

Solution 1

To be sure a URL is correct, test the URL that you want to clip in a browser before you attempt to clip it. Also test the URL to be sure that it is accessible from the provider middle tier.

Problem 2

The connection times out when attempting to browse to any Web site and your environment uses a proxy server to connect to HTTP servers outside a firewall.

Solution 2

Make sure that the proxy servers are configured correctly.

To configure the proxy servers, go to the Web Clipping Provider Test Page, as described in Section B.4.3, "Checking the Status of the Provider with the Test Page". In the Web Clipping Provider Test Page, click **Edit** in the **Actions** column of the **HTTP Proxy** row. In the Edit Provider page, specify the **HTTP Proxy Host** and the **HTTP Proxy Port** for the HTTP Proxy.

For access to servers inside the firewall, you can specify a list of domain names that need not go through the firewall by selecting **No Proxy for Domains beginning with** and entering the URL. You do not need to restart the managed server for the new settings to take effect.

For more information about configuring proxy servers, see the "Administering Web Clipping" appendix of the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Problem 3

Your configuration includes a load balancer and you experience difficulty making or maintaining connections while attempting to add a clip to a Web Clipping portlet.

Solution 3

The configuration was not set up correctly. To solve this problem, perform the following:

- If multiple Oracle WebLogic Server instances are set up behind a load balancer, the Web Clipping Repository and HTTP proxy must be configured identically on all Oracle WebLogic Server instances before you join them to the load balancer.

  Web clippings have definitions that must be stored persistently in the Web Clipping Repository hosted by an Oracle Database server. In a multiple middle-tier environment, all instances of Oracle WebLogic Server must store definitions in the same repository.

- The Load Balancer must be session-enabled. If it is not, the first request connects, but subsequent requests, which may be routed to a different instance, fail.

For more information about configuring with a load balancer, see the "Performing Advanced Configuration" chapter of the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

Problem 4

You use a reverse proxy, but cannot make a connection.

Solution 4

Make sure that the reverse proxy server is configured correctly. See the "Performing Advanced Configuration" chapter of the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal* for information about configuring a reverse proxy.

Problem 5

You cannot connect and the error log contains a message about denying logon to the database. (See Section B.4.1, "Setting Logging Levels" for information about the error log.)

Solution 5

The PORTAL schema password for the infrastructure database may have been modified manually and it may not match the password stored in Oracle Internet Directory. Refer to the *Oracle Fusion Middleware Administrator's Guide for Oracle Internet Directory* for more information about setting the password.

B.4.5 HTTP Error Code 407 When Clipping Outside Firewall

The proxy servers are configured for proxy authentication and you receive HTTP error code 407 when you attempt to clip a page outside the firewall.

Problem

The proxy servers are configured for proxy authentication, but you have not configured proxy authentication.

Solution

You must manually configure proxy authentication. Web Clipping supports both global proxy and per-user authentication. You can specify the realm of the proxy server and whether all users login automatically with a provided user name and password, each user logs in with an individual user name and password, or all users log in with a specified user name and password. For more information about configuring proxy authentication, see the "Administering Web Clipping" appendix of the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

B.4.6 Cannot Clip a Page

You can connect to the Web site, but you cannot clip the page.

Problem

The page may be overpopulated with IFrames.

Solution

View the page in a browser and look at the page source. If it contains IFrames, start browsing in Web Clipping Studio using the URL pointed to by the IFrame `src` attribute.

B.4.7 Images Not Retrieved with Clipping

Images in a Web clipping are not retrieved with the rest of the clipping.

Problem

Because images are treated as links (using the `src` attribute of the `IMG` tag), images from clipped sites are served directly from the original sites. If the images require that you enable proxy settings during creation of the clipping, then disabling the browser proxy setting disables the viewing of the images in a clipping.

Solution

Enable proxy settings in the browser.

B.4.8 Resolving Problems with Migration of URL-based Portlets

You can migrate URL-based portlets that are stored in a `provider.xml` file to Web Clipping portlets. This section explains how to troubleshoot issues that arise as you migrate your portlets to Web Clipping portlets. For more information on the migration process itself, refer to Section 5.6, "Migrating from URL-Based Portlets".

B.4.8.1 File Not Found Exception When Running Migration Tool

When you run the migration tool, you receive the following exception:

```
Exception in thread "main" java.io.FileNotFoundException:
 /wrongpath/somservice.properties (No such file or directory)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:103)
        at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.importService
                                             (UrlServicesMigrator.java:631)
        at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.main
                                             (UrlServicesMigrator.java:724)
        at oracle.webdb.wcs.WcWebdbMain.main(WcWebdbMain.java:60)
```

Problem

The files you specified for the deployment properties file for either the URL-based portlets service or the Web Clipping Portlet service do not exist.

Solution

Verify the path to both service deployment properties files in the argument list to see if they point to the correct location.

B.4.8.2 Null Pointer Exception When Running Migration Tool

When you run the migration tool, you receive the following exception:

```
Exception in thread "main" java.lang.NullPointerException
        at java.util.Hashtable.put(Hashtable.java:375)
        at java.util.Properties.setProperty(Properties.java:97)
        at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.setRepository
                                                  (UrlServicesMigrator.java:415)
        at oracle.webdb.wcs.client.urlservices.UrlServicesMigrator.main
                                                  (UrlServicesMigrator.java:733)
        at oracle.webdb.wcs.WcWebdbMain.main(WcWebdbMain.java:58)
```

Problem

The Repository settings are not configured for the Web Clipping Provider.

Solution

See Section 5.6.1, "Preparing for Migration" for information about steps you must take before running the migration tool.

B.4.8.3 Target provider.xml is Already Migrated Error

When you run the migration tool, you receive the following error:

```
Error: Target provider.xml is already migrated.
```

Problem

The `provider.xml` file pointed to by the URL-based portlets service deployment properties file has already been migrated. The provider definition class specified in the `provider.xml` file is already in WcProviderDefinition.

Solution

If you want to redo the migration, rename your service in both the deployment properties file and the entire directory name under WEB-INF/providers and then run the migration again.

B.4.8.4 Cannot Migrate provider.xml with Class Error

When you run the migration tool, you receive the following error:

```
Error: Can't migrate provider.xml with class <someclass>
```

Problem

The `provider.xml` file pointed to by the URL-based portlets service deployment properties file contains a provider class name that does not match the one used by URL-based portlets. Specifically, the Provider class name specified in the file is not `oracle.portal.provider.v2.http.URLProviderDefinition`. The migration tool is intended to migrate URL-based Portlet provider definitions, not any other kind of provider definitions. Even those classes which sub-class from URL-based portlets provider definition will not be correctly migrated.

Solution

You cannot migrate this service.

B.5 Need More Help?

You can find more solutions on Oracle *MetaLink*, `http://metalink.oracle.com`. If you do not find a solution for your problem, log a service request.

> **See Also:** *Oracle Portal Release Notes*, available on the Oracle Technology Network.

C Mapping Profile Items to Attributes

When developing portlet functionality that requires access to a user's identity, you often need to map between attributes in Oracle Portal and Oracle Internet Directory, and `userProfileItems` in WSRP. Note that for Oracle Portal, these attributes are static and cannot be configured. This appendix provides a table that shows the mapping between various attributes.

C.1 Mapping userProfileItems to Attributes

Table C–1 provides a mapping of `userProfileItems` to Oracle Internet Directory and Oracle Portal attributes.

Note: In cases where the Oracle Internet Directory attribute holds a complex string, like `homePostalAddress`, it is parsed into the needed component strings. The delimiter is assumed to be the dollar sign (`$`). For example, suppose you had the following value for `homePostalAddress` in Oracle Internet Directory:

```
123 Main St$$$Smallville$NY$10001
```

In the Oracle Portal profile attributes, this string would be parsed as follows:

- 123 Main St goes into `HOME_ADDR1`.
- Smallville goes into `HOME_CITY`.
- NY goes into `HOME_STATE`.
- 10001 goes into `HOME_ZIP`.

Table C–1 userProfileItems Mapping

Oracle Portal Profile Attributes	Oracle Internet Directory Attributes	WSRP userProfileItems
FIRST_NAME	givenName	name/given
LAST_NAME	sn	name/family
MIDDLE_NAME	middleName	name/middle
KNOWN_AS	displayName	name/nickname
DATE_OF_BIRTH	orclDateOfBirth	bDate

Table C–1 (Cont.) userProfileItems Mapping

Oracle Portal Profile Attributes	Oracle Internet Directory Attributes	WSRP userProfileItems
ORGANIZATION	o	employerInfo/employer
TITLE	title	employerInfo/jobtitle
HOME_ADDR1	homePostalAddress	homeInfo/postal/name
HOME_ADDR2		homeInfo/postal/street
HOME_ADDR3		homeInfo/postal/organization
HOME_CITY		homeInfo/postal/city
HOME_STATE		homeInfo/postal/stateprov
HOME_ZIP		homeInfo/postal/postalcode
HOME_COUNTRY		homeInfo/postal/country
HOME_PHONE	homePhone	homeInfo/telecom/telephone/number
OFFICE_ADDR1	street	businessInfo/postal/name
OFFICE_ADDR2		businessInfo/postal/street
OFFICE_ADDR3		businessInfo/postal/organization
OFFICE_CITY	l	businessInfo/postal/city
OFFICE_STATE	st	businessInfo/postal/stateprov
OFFICE_ZIP	postalCode	businessInfo/postal/postalcode
OFFICE_ COUNTRY	c	businessInfo/postal/country
WORK_PHONE	telephoneNumber	businessInfo/telecom/telephone/number
FAX	facsimileTelephoneNumbe r	businessInfo/telecom/fax/number
MOBILE_PHONE	mobile	businessInfo/telecom/mobile/number
EMAIL	mail	businessInfo/online/email

D
Manually Packaging and Deploying PDK-Java Providers

This appendix explains how to manually package your PDK-Java provider implementation into a portable format suitable for deployment on the Oracle WebLogic Server or another J2EE application server. It then explains how to deploy the resulting EAR file in an Oracle WebLogic Server environment and subsequently register it with one or more Oracle Portal instances.

Note: In general, we recommend that you package and deploy your providers using the tools available in Oracle JDeveloper. However, if you find that you must package and deploy your providers manually for some reason, we provide the information in this appendix for your reference.

- Section D.1, "Introduction"
- Section D.2, "Packaging and Deploying Your Providers"

D.1 Introduction

Before preceding with packaging and deploying your provider, you must understand the following basic concepts:

- Section D.1.1, "WAR and EAR files"
- Section D.1.2, "Service Identifiers"

D.1.1 WAR and EAR files

WAR and EAR files are used to deploy applications on a J2EE application server, such as Oracle WebLogic 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 providers under a single adapter servlet. The providers are identified by a service identifier. When you deploy a new provider, you must assign a service identifier to the provider and use that service identifier when creating your provider WAR file. The service name is used to look up a file called `service_id.properties`, which defines the characteristics of the provider, such as whether to display its test page.

For example, you can register the PDK-Java samples provider 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 provider name (sample) is appended to the URL of the PDK-Java samples provider. In this case, you should leave the Service Id field blank when registering the provider.

You can specify the service identifier separately in cases where multiple portals are sharing the same provider. By registering each portal with a different service identifier, you can specify the provider properties for each consumer independently.

Once your provider has been deployed, you must use the correct service identifier to register your provider with Oracle Portal, which ensures that requests are routed to the correct provider. If the adapter servlet receives a request without a service identifier, the request goes to the default provider.

Note: If you do not know the service identifier, check the provider test page or contact the administrator of the provider. If you are using the Federated Portal Adapter, the URL points to the adapter, not the provider, 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 Providers

The following sections show the steps you must perform to package and deploy a provider manually:

- Section D.2.1, "Packaging Your Provider"
- Section D.2.2, "Deploying Your EAR File Using Fusion Middleware Control"
- Section D.2.3, "Testing Deployment"
- Section D.2.4, "Setting Deployment Properties"
- Section D.2.5, "Securing Your Provider"
- Section D.2.6, "Registering Your Provider"

D.2.1 Packaging Your Provider

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, you are free to assemble your WAR file that way.

- Section D.2.1.1, "Preparing Your Directories"
- Section D.2.1.2, "Specifying Your Default Service"
- Section D.2.1.3, "Creating Your WAR File"
- Section D.2.1.4, "Creating Your EAR File"

D.2.1.1 Preparing Your Directories

In preparation for creating your WAR file, you need to 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 provider needs any additional JAR files, add them to the `WEB-INF/lib` directory.
4. If your provider needs any additional Java classes not contained in a JAR file, 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 provider under the providers directory.
7. Copy the `_default.properties` file to *`service_name`*`.properties` and edit it to reflect your provider'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 provider definition file in the subdirectory you just created.
10. Edit `_default.properties` to reflect the configuration settings of your default provider. The default provider is accessed if a service identifier is not specified in a request. Refer to Section D.2.1.2, "Specifying Your Default Service" for more information on this step.
11. If you use servlets to render content, edit `WEB-INF/web.xml` to add your servlets to the list of pre-defined 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 provider that receives any request without a service name. You specify a default provider by editing the `_default.properties` file in the deployment directory of your WAR file.

Edit the definition entry to point to the provider definition file that represents your default provider. 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 Using Fusion Middleware Control

You can deploy your EAR file to your WebLogic Managed Server (WLS_PORTAL) using Fusion Middleware Control. To deploy the EAR file:

1. Start the Oracle Fusion Middleware Control and navigate to the home page of the Portal instance in which you configured PDK-Java (for example, `WLS_PORTAL`).
2. From the WebLogic Server menu, choose **Application Deployment**, and then **Deploy**.

   The **Select Archive** page is displayed.
3. In the Archive or Exploded Directory section, you can select one of the following:
   - **Archive is on the machine where this browser is running.** Then, enter the location of the archive or click **Browse** to find the archive file.
   - **Archive or exploded directory is on the server where Enterprise Manager is running.** Then, enter the location of the archive or click **Browse** to find the archive file.
4. Click **Continue**. The URL mapping for Web Modules displays. 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. In the Deployment Plan section, you can select one of the following:
   - **Automatically create a new deployment plan.**
   - **Deployment plan is on the machine where this web browser is running.** If you select this option, enter the path to the plan.
   - **Deployment plan is on the server where Enterprise Manager is running.** I If you select this option, enter the path to the plan.
6. Click **Next.**

   The **Select Target** page is displayed.
7. Select the target to which you want to deploy the application. The Administration Server, Managed Servers, and clusters are listed. You can select a cluster, one or more Managed Server in the cluster, or a Managed Server that is not in a cluster. Although the Administration Server is shown in the list of targets, you should not deploy an application to it. The Administration Server is intended only for administrative applications such as the Oracle WebLogic Server Administration Console.
8. Click **Next.**

   The Application Attributes page is displayed.
9. In the Application Attributes section, for **Application Name,** enter the application name.
10. In the Context Root of Web Modules section, if the web module does not have the context root configured in the application.xml file, you can specify the context root for your application. The context root is the URI for the web module. Each web module or EJB module that contains web services may have a context root.

11. If the application's adf-config.xml file archive contains MDS configuration, the Target Metadata Repository section is displayed. It allows you to choose the repository and partition for this application:
    - To change the repository, click the icon next to the **Repository Name.** In the Metadata Repositories dialog box, select the repository and click **OK.**
    - To change the partition, enter the partition name in **Partition Name.**
12. In the Distribution section, you can select one of the following:
    - **Distribute and start application (servicing all requests)**
    - **Distribute and start application in admin mode (servicing only admin requests)**
    - **Distribute only**
13. Click **Next.**

    The **Deployment Wizard, Deployment Settings** page is displayed.
14. On this page, you can perform common tasks before deploying your application or you can edit the deployment plan or save it to a disk. You can:
    - Configure Web modules
    - Configure application security
15. Expand **Deployment Plan.**

    You can edit and save the deployment plan, if you choose.
16. Click **Deploy.**

    Application Server Control displays processing messages.
17. When the deployment is completed, click **Close.**

D.2.3 Testing Deployment

To test your provider deployment, you access the provider 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 Oracle WebLogic Server instance. In an Oracle Application Server installation with Oracle Web Cache installed, `port` should be the Oracle Web Cache listener port (for example, 8090). In the Oracle WebLogic Server installation, the default HTTP port number is 8888.

`context_root` is the URI path prefix you mapped to the provider Web application on deployment (for example, `/myapp`) or the default one specified in `application.xml` in the case of a manual deployment with `dcmctl`.

For example:

```
http://my.host.com:8090/newProvider/providers
```

If your `.properties` file specifies `showTestPage=true`, you should see the familiar test page for your default provider. To view the test page for a specific provider service, you can append the service name to the URL. For example:

```
http://my.host.com:8090/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–2 provides a list of these variables with descriptions.

Table D–2 JNDI Variables for Provider Deployment

Variable	Description
`oracle/portal/provider/global/log/logLevel`	Is the logging level (0-8) used by PDK-Java and applies to all providers.
`oracle/portal/`*`service_name`*`/showTestPage`	Is a Boolean flag that specifies whether a provider's test page is accessible. The default value is true.
`oracle/portal/`*`service_name`*`/maxTimeDifference`	Is the provider's HMAC time difference.
`oracle/portal/`*`service_name`*`/definition`	Is the location of the provider'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 provider customizations. It has no default value.

Setting the Variables

You can set the values of the variables in Table D–2 as you would any other JNDI variables. Refer to Section 7.2.4.2, "Setting JNDI Variable Values" for information on how to set JNDI variables.

D.2.5 Securing Your Provider

When using the PDK-Java framework in a production environment, you should secure your providers. For more information on securing providers, refer to Section 7.2.7, "Implementing Portlet Security" and the *Oracle Fusion Middleware Administrator's Guide for Oracle Portal*.

D.2.6 Registering Your Provider

Once you have successfully deployed and verified your provider, you can register it as you would any other provider. Refer to Section 6.5.5, "Registering and Viewing Your Oracle PDK-Java Portlet" for more information about registering your provider.

E Oracle Portal Provider Test Suite

The Oracle Portal Provider Test Suite performs sanity, performance, and unit tests on a Web provider without an installation of Oracle Portal or Internet access from the provider machine. The following two types of utilities are available to the user in the test suite:

- Section E.1, "Provider Test Page"
- Section E.2, "Test Harness"

More On OTN

You can download the Oracle Portal Provider Test Suite from the Oracle Portal Developer Kit (PDK) page on Oracle Technology Network (OTN):

`http://www.oracle.com/technology/products/ias/portal/pdk.html`

From the Oracle Portlet Developer Kit (PDK) home page, select **Download the PDK-Java TestSuite**.

E.1 Provider Test Page

The provider test page provides a basic sanity test for the provider. It contains a list of portlets, servlet initialization arguments, and the version numbers of the `ptlshare` and `pdkjava` libraries. The provider test page is the simplest utility available for testing any Web provider. You access the test page by a URL after deploying the enterprise application or Web application on Oracle WebLogic Server. You test the Web provider by accessing this URL from a browser:

```
http://server:port/application_name/providers/provider_name
```

For example, the PDK-Java comes with a sample application and portlets. The application is encapsulated in a WAR file, which in turn is encapsulated in an EAR file. When you deploy it, Oracle WebLogic Server extracts the files and creates a directory structure with the sample portlets under:

```
MW_HOME\user_projects\domains\<DomainName>\servers\WLS_PORTAL\tmp\_WL_user\jpdk
```

To view the test page for this provider, you would use this URL:

```
http://server:port/jpdk/providers/sample
```

When you access the test page, the SOAP servlet validates the XML provider definition, `provider.xml`, ensuring that the corresponding provider is well formed. This validation is useful for debugging deployment issues with your provider before attempting to register it with a Oracle Portal. If you successfully deploy your Web application on WLS, you receive a success message on the test page.

You can turn the test page on and off by using the JNDI variable `oracle/portal/provider/`*`provider_name`*`/showTestPage` to true or false. Once a provider is tested and in service you might want to restrict access to the test page. For more information about retrieving and setting JNDI variables, refer to Section 7.2.4, "Using JNDI Variables".

E.2 Test Harness

The test harness is a command line utility for unit- and performance-testing your providers without accessing an Oracle Portal instance. The test harness sends HTTP requests to the target Web provider and records the responses for further analysis. The responses are logged into an XML file. Performance statistics are logged into another file for analysis.

The test harness provides considerable flexibility in the following ways:

- You create your own test definition. Based on the information in the test definition, the harness sends requests to the provider. The test definition file is in XML format and lists request instances to send to the target Web provider.
- The information returned by the provider is stored in a standard XML file, which makes it easier to understand and analyze.
- You can perform load testing of the Web provider.

E.2.1 Test Definition File

The test definition file is an XML file that lists the request instances for a particular test. You can optionally subdivide the request instances into request groups within the test definition file. You can include the details of the request instances in the test definition file or refer to a request library XML file that defines the instance details.

In addition to the request instances, the test definition file also includes information about the host and port of the target Web provider and any pre-processors to which it needs to be sent. A pre-processor enables you to include application-specific logic in the request instances at runtime in the test harness. For example, a pre-processor might check the validity of XML in the SOAP messages being sent to the target Web provider. The test harness provides the following three built-in pre-processors:

- Oracle Portal pre-processor
- Validation-based caching pre-processor
- HMAC (Hashed Message Authentication Checksum) pre-processor

The following sample test definition file illustrates the format and content of the file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?testDefinition version="0.1"?>
<testDefinition>
 <defaultHost>machine.name.com</defaultHost>
 <defaultPort>80</defaultPort>
 <defaultPath>/jpdk/providers/sample</defaultPath>
 <property name="portletId" value="1"/>
 <preProcessorDefinitions>
   <preProcessor
       class="oracle.webdb.testharness.preprocessor.PortalPreProcessor">
     <name>portal</name>
   </preProcessor>
   <preProcessor class="oracle.webdb.testharness.preprocessor.HMACPreProcessor">
     <name>hmac</name>
```

```
      <sharedKey>1234567890aBcDeFgHiJ</sharedKey>
    </preProcessor>
  </preProcessorDefinitions>
  <requestGroup id="register">
    <description>Carries out necessary registrations</description>
    <cycles>1</cycles>
    <threads>1</threads> -- increase the number to load test the Web provider.
    <requestInstance id="register_provider">
      <libraryId>ptl902</libraryId>
      <definitionId>registerProvider</definitionId>
      <runs>1</runs>
    </requestInstance>
    <requestInstance id="register_portlet">
      <libraryId>ptl902</libraryId>
      <definitionId>registerPortlet</definitionId>
      <runs>1</runs>
    </requestInstance>
  </requestGroup>
  <requestGroup id="show">
    <description>Carries out work necessary to show the portlet</description>
     ...
     ...
  </requestGroup>
     ...
</testDefinition>
```

E.2.2 runTest Command

The `runTest` command invokes the test harness with the specified options using the specified test definition.

runTest *options test_definition*

Table E–1 describes the command options for `runTest`.

Table E–1 runTest Options

Option	Description
-n *testname*	Assigns the specified name to this run. Defaults to a date-time name.
-g *groupId*	Is a comma separated list of the identifiers of the request groups in the test definition file to be run for this test. To run all groups, specify `--all-groups`.
-p *perfLogFile*	Is the name of the performance log file. Defaults to the test name. To disable performance testing, use `--no-perf`.
-l *resLogFile*	Is the name of the response log file. Defaults to the test name.
-v *resLogLevel*	Is the level of response logging (MIN | HEADERS | ALL). Default is ALL.
-c *csvFile*	Is the name of the response data CSV file. Default is to not generate it.
-s *ctlFile*	Is the name of the response data ctl file. Default is to not generate it.
--no-perf	Switches off all logging of performance data.
--all-groups	Runs all groups in the test definition.
-verbose	Produces verbose informational logging.

E.2.3 Running a Test with Test Harness

In order to test Web providers using the test harness, do the following:

1. Extract the `pdktestsuite.zip` file to a convenient location. This will create a directory called `pdk\testsuite\pdktest` to which all the test harness files are extracted.
2. Set an environment variable called `PDKTEST_HOME` to point to the location created in the previous step. For example, if you extracted to your D drive, you would now have a directory called `D:\pdk\testsuite\pdktest`. Set the value of `PDKTEST_HOME`, to `D:\pdk\testsuite\pdktest`. The Test Harness provides executable scripts in the `bin` directory.
3. Create a test definition file similar to the one in Section E.2.1, "Test Definition File".
4. Invoke the `runTest` command as described in Section E.2.2, "runTest Command".

F Content Management APIs and Views

This appendix lists the supported Oracle Portal content management APIs and views. It contains the following sections:

- Section F.1, "Supported APIs"
- Section F.2, "Secure Views"

F.1 Supported APIs

This section provides an overview of the APIs available to you for performing content management tasks.

More On OTN

For more information about the contents of the different packages, refer to the *Oracle Portal PL/SQL API Reference* on Portal Center:

`http://portalcenter.oracle.com`

In the Portal Focus Areas section, click **Portlet Development**, then in the APIs and References section, click **PL/SQL API Reference**.

F.1.1 The WWSBR_API Package

This package contains APIs for accessing and manipulating the contents in the portal schema of the MDS Repository. You can perform the following actions using these APIs:

- Create, edit, and delete portal objects, such as pages, categories, and items.
- Copy and move items and pages.
- Check items in and out.
- Approve or reject pending items.

F.1.2 The WWSRC_API Package

This package contains APIs for searching content in the MDS Repository. You can perform the following actions using these APIs:

- Search for items, pages, categories, and perspectives.
- Filter search results based on specific attribute values.
- Return search results as an XML document.

F.1.3 The WWSEC_API Package

This package contains APIs for administering Oracle Portal security. You can perform the following actions using these APIs:

- User maintenance (create user profile entries, activate portal access, update user properties, delete user profile entries, and other associated tasks).
- Group maintenance (create, activate, update, delete groups, and other associated tasks).
- Access control (grant, check, copy, update, remove user/group privileges, and other associated tasks).

F.1.4 The WWCTX_API Package

This package contains APIs for enabling access to a session context. These APIs provide the methods necessary to manage a session context for a specific user.

A session is established when a user accesses the portal. A public session is created upon initial access. It is converted to an authenticated session after the user logs in and exists until the user logs out. Old sessions are also periodically cleaned up by the portal.

F.1.5 The WWPRO_API_INVALIDATION Package

This package contains APIs for invalidating content cached in the Oracle Web Cache. Use these APIs when your code causes content to change to make sure that those changes are immediately visible to users. For example, you may need to call an API to invalidate the cache in code that adds an item to a page.

F.2 Secure Views

The views in `wwsbr_api_views` enable you to safely build custom queries against the portal schema of the MDS Repository, without having to worry about the definitions changing between releases. Table F–1 lists the currently supported views:

Table F–1 Secure Content Repository Views

View	Description
WWSBR_ALL_CATEGORIES	Describes all categories.
WWSBR_ALL_CONTENT_AREAS	Describes all page groups.
WWSBR_ALL_FOLDER_REGIONS	Describes all page regions on pages that the current user is able to view.
WWSBR_ALL_FOLDERS	Describes pages on which the current user has *View* privileges or higher.
WWSBR_ALL_ITEMS	Describes the items that the current user is able to view.
WWSBR_ALL_NAVIGATION_BARS	Describes all navigation pages that the current user is able to view.
WWSBR_ALL_PERSPECTIVES	Describes all perspectives.
WWSBR_ALL_STYLES	Describes all styles.
WWSBR_APPROVER	Describes the approvers for every page group approval step.
WWSBR_ATTRIBUTES	Describes all attributes.

Table F–1 (Cont.) Secure Content Repository Views

View	Description
WWSBR_CONTENT_AREA_APPROVAL	Describes the approval definition for a page group.
WWSBR_CONTENT_AREA_ITEM_TYPES	Describes which item types are available to which page groups.
WWSBR_FOLDER_ATTRIBUTES	Describes the attributes associated with pages that the current user is able to view
WWSBR_FOLDER_PERSPECTIVES	Describes the perspectives associated with pages that the current user is able to view.
WWSBR_FOLDER_TYPE_ATTRIBUTES	Describes the attributes that are available within each page type.
WWSBR_FOLDER_TYPES	Describes all page types.
WWSBR_ITEM_APPROVAL	Describes item approval information for every item that goes through approval.
WWSBR_ITEM_ATTRIBUTES	Describes the attributes and attribute values associated with items that the current user is able to view.
WWSBR_ITEM_PERSPECTIVES	Describes the perspectives associated with items that the current user is able to view.
WWSBR_ITEM_TYPE_ATTRIBUTES	Describes the attributes that are available within each item type.
WWSBR_ITEM_TYPES	Describes all item types.
WWSBR_SUBSCRIPTION	Describes user subscription to portal pages and items.
WWSBR_USER_FOLDERS	Describes pages on which the current user has *Manage* privileges or higher.
WWSBR_USER_PAGES	Describes pages on which the current user has *Manage* privileges or higher.

Note: In the following tables, an asterisk (*) indicates that the column is part of the view's primary key.

F.2.1 WWSBR_ALL_CATEGORIES

Table F–2 lists the columns in the WWSBR_ALL_CATEGORIES view, which describes all categories.

Table F–2 WWSBR_ALL_CATEGORIES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the category.	
CAID*	NOT NULL	NUMBER	Page group ID for the category.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
PARENTID		NUMBER	ID of the parent category.	0 (zero) if the category has no parent.

Table F–2 (Cont.) WWSBR_ALL_CATEGORIES

Column	Null?	Data Type	Description	Notes
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME		VARCHAR2(60)	Internal name of the category. Unique within parent category.	
DISPLAY_NAME	NOT NULL	VARCHAR2(240)	Display name, or title, of the category.	

F.2.2 WWSBR_ALL_CONTENT_AREAS

Table F–3 lists the columns in the WWSBR_ALL_CONTENT_AREAS view, which describes all page groups. This view shows all page groups in the repository, regardless of the current user's privileges.

Note: An alternative way of uniquely identifying a row in this view is to use the NAME and LANGUAGE columns.

Table F–3 WWSBR_ALL_CONTENT_AREAS

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the page group.	
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME	NOT NULL	VARCHAR2(60)	Internal name of the page group.	
DISPLAY_NAME		VARCHAR2(256)	Display name, or title, of the page group.	
DEFAULT_ LANGUAGE	NOT NULL	VARCHAR2(30)	Language code for the default language of the page group.	

F.2.3 WWSBR_ALL_FOLDER_REGIONS

Table F–4 lists the columns in the WWSBR_ALL_FOLDER_REGIONS view, which describes all page regions on pages viewable by the current user.

Table F–4 WWSBR_ALL_FOLDER_REGIONS

Column	Null?	Data Type	Description	Notes
ID*		NUMBER(38)	ID of the region.	
DISPLAY_NAME*		VARCHAR2(4000)	Display name, or title, of the region.	
LANGUAGE		VARCHAR(7)	Language code.	
FOLDER_ID		NUMBER(38)	ID of the page containing the region.	Foreign key to: ▪ WWSBR_ALL_ FOLDERS.ID ▪ WSBR_USER_ FOLDERS.ID

Table F–4 (Cont.) WWSBR_ALL_FOLDER_REGIONS

Column	Null?	Data Type	Description	Notes
FOLDER_CAID		NUMBER(38)	Page group ID for the page containing the region.	Foreign key to: ■ WWSBR_ALL_ FOLDERS.CAID ■ WWSBR_USER_ FOLDERS.CAID
PARENT_ID		NUMBER(38)	ID of the parent region, if the region is nested.	
TYPE		VARCHAR2(30)	The region type.	Valid values: ■ NONE - undefined ■ TAB - tab region ■ SUBPAGE - sub page region ■ ITEM - item region ■ PORTLET - portlet region
SEQ		NUMBER	Sequence of a region within the parent region.	
ALLOW_CONTENT		NUMBER	Indicates if the region allows content to be added.	Valid values: ■ 0 - does not allow content to be added ■ 1 - allows content to be added

F.2.4 WWSBR_ALL_FOLDERS

Table F–5 lists the columns in the WWSBR_ALL_FOLDERS view, which describes the pages on which the current user has *View* privileges or higher.

Table F–5 WWSBR_ALL_FOLDERS

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	Unique identifier of the page within the page group.	
CAID*	NOT NULL	NUMBER	ID of the page group.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
PARENT_ID		NUMBER	ID of the parent page.	0 (zero) if root page of the page group.
NAME	NOT NULL	VARCHAR2(60)	Internal name of the page.	Must be unique within the parent page.
DISPLAY_NAME	NOT NULL	VARCHAR2(256)	Display name, or title, of the page.	
CATEGORY_ID		NUMBER	ID of the category assigned to the page.	Foreign key to WWSBR_ ALL_CATEGORIES.ID
CATEGORY_SITEID		NUMBER	Page group ID for the category assigned to the page.	Foreign key to WWSBR_ ALL_ CATEGORIES.CAID

Table F–5 (Cont.) WWSBR_ALL_FOLDERS

Column	Null?	Data Type	Description	Notes
DESCRIPTION		VARCHAR2(2000)	Description of the page.	
TYPE_ID	NOT NULL	NUMBER	ID of the page type.	Foreign key to WWSBR_ FOLDER_TYPES.ID
TYPE_SITEID	NOT NULL	NUMBER	Page group ID for the page type.	Foreign key to WWSBR_ FOLDER_TYPES.CAID
BASE_TYPE_ID	NOT NULL	NUMBER	The ID of the page base type	Foreign key to WWSBR_ FOLDER_TYPES.ID where WWSBR_ FOLDER_TYPES.IS_ BASE_FOLDER_TYPE=1
IS_PORTLET	NOT NULL	NUMBER(1)	Indicates if the page is published as a portlet.	Valid values: ▪ 0 - not a portlet ▪ 1 - is a portlet
IS_CACHING_ON		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
CACHE_MODE		NUMBER(1)	The caching mode for the page.	Valid values: ▪ 2 - no caching ▪ 1 - cache page definition only ▪ 0 - cache page definition and content for x minutes ▪ 4 - cache page definition only at system level ▪ 3 - cache page definition and content at system level for x minutes
SUB_FOLDER_ SEQUENCE		NUMBER	The sequence (order) of the page in its parent page's sub-page region.	
DISPLAY_IN_ PARENT_FOLDER	NOT NULL	NUMBER(1)	Indicates if the page is displayed in its parent page's sub page region.	
ITEMVERSIONING		VARCHAR2(30)	Determines the level of item versioning for the page.	Valid values: ▪ versionnone - no versioning ▪ versionsimple - simple versioning ▪ versionaudit - audit versioning.
STYLE_ID	NOT NULL	NUMBER	ID for the style used by the page.	Foreign key to WWSBR_ ALL_STYLES.ID
STYLE_CAID	NOT NULL	NUMBER	Page group ID for the style used by the page.	Foreign key to WWSBR_ ALL_STYLES.CAID

Table F–5 (Cont.) WWSBR_ALL_FOLDERS

Column	Null?	Data Type	Description	Notes
URL_VALUE		VARCHAR2(4000)	URL value for a URL type page.	
SEARCH_VALUE		VARCHAR2(2000)	DEPRECATED. This column is retained for backward compatibility only.	
PLSQL_VALUE		VARCHAR2(2000)	PL/SQL value for a PL/SQL page.	
IMAGE		VARCHAR2(350)	Unique name of the image associated with the page.	Matches the NAME column in the document table.
TITLE_IMAGE_NAME		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
ROLLOVER_IMAGE_NAME		VARCHAR2(350)	The unique name of the rollover image associated with the page.	Matches the NAME column in the document table.
BANNER_IMAGE_NAME		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
NAVIGATION_BAR_ID		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
NAVIGATION_BAR_CAID		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
IS_PUBLIC	NOT NULL	NUMBER(1)	Indicates that the page is viewable by public users.	Valid values: ▪ 0 - not public ▪ 1 - is public
ITEM_LEVEL_SECURITY	NOT NULL	NUMBER(1)	Indicates that item level security is enabled for the page.	Valid values: ▪ 0 - ILS disabled ▪ 1 - ILS enabled
DISPLAY_FULL_SCREEN		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
PLSQL_FOLDER_EXECUTOR		VARCHAR2(30)	For PL/SQL type pages, the database schema used to execute the PL/SQL code.	
CREATEDATE		DATE	Date the page was created.	

Table F–5 (Cont.) WWSBR_ALL_FOLDERS

Column	Null?	Data Type	Description	Notes
CREATOR		VARCHAR2(256)	User name of the person who created the page.	
UPDATEDATE		DATE	Date the page was last updated.	
UPDATOR		VARCHAR2(256)	User name of the person who last updated the page.	

F.2.5 WWSBR_ALL_ITEMS

Table F–6 lists the columns in the WWSBR_ALL_ITEMS view, which describes the items viewable by the current user.

For an example of how to use this view, refer to Section 10.3.4, "Finding an Item ID".

Table F–6 WWSBR_ALL_ITEMS

Column	Null?	Data Type	Description	Notes
MASTERID	NOT NULL	NUMBER	Master ID of the item.	All versions and translations of an item have the same master ID.
ID*	NOT NULL	NUMBER	ID of the item.	Each version of an item receives a new ID. If translations and approvals are enabled, multiple rows may exist for the same ID.
CAID*	NOT NULL	NUMBER	Page group ID for the item.	Foreign key to: ▪ WWSBR_ALL_FOLDERS.CAID ▪ WWSBR_USER_FOLDERS.CAID ▪ WWSBR_ALL_CONTENT_AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
IS_CURRENT_VERSION*	NOT NULL	NUMBER(1)	Indicates that the item is the current version.	Valid values: ▪ 0 - not current version ▪ 1 - is current version
FOLDER_ID	NOT NULL	NUMBER	ID of the page containing the item.	Foreign key to: ▪ WWSBR_ALL_FOLDERS.ID ▪ WWSBR_USER_FOLDERS.ID ▪ WWSBR_ALL_FOLDER_REGIONS.FOLDER_ID
FOLDER_REGION_ID	NOT NULL	NUMBER	ID of the region containing the item.	Foreign key to WWSBR_ALL_FOLDER_REGIONS.ID

Table F–6 (Cont.) WWSBR_ALL_ITEMS

Column	Null?	Data Type	Description	Notes
NAME	NOT NULL	VARCHAR2(256)	Internal name of the item.	The name is a system generated GUID value that can be used as an alternate ID. The name is also used in path based URLs.
DISPLAY_NAME		VARCHAR2(256)	Display name, or title, of the item.	
ITEMTYPE	NOT NULL	VARCHAR2(30)	The base item type, for example, BASEFILE, BASETEXT, BASEURL, and so on.	
SUBTYPE		VARCHAR2(40)	ID of the item type.	Foreign key to WWSBR_ ITEM_TYPES.ID
SUBTYPE_CAID		NUMBER	Page group ID of the item type.	Foreign key to WWSBR_ ITEM_TYPES.CAID
PARENT_ITEM_ID		NUMBER	ID of the parent item, if the item is a sub item.	0 (zero) if the item is not a sub item.
CATEGORY_ID		NUMBER	ID for the category assigned to the item.	Foreign key to WWSBR_ ALL_CATEGORIES.ID
CATEGORY_CAID		NUMBER	page group ID for the category assigned to the item.	Foreign key to WWSBR_ ALL_ CATEGORIES.CAID
AUTHOR		VARCHAR2(50)	Author of the item.	
DESCRIPTION		VARCHAR2(2000)	Description of the item.	
PUBLISH_DATE	NOT NULL	DATE	Date the item will be published.	
EXPIREMODE		VARCHAR2(90)	Expiration mode for the item.	Valid values: ■ 'PERMANENT' - item never expires ■ 'NUMBER' - item expires in EXPIRENUMBER days from CREATEDATE ■ 'DATE' - item expires on EXPIREDATE
EXPIRENUMBER		NUMBER	Number of days after CREATEDATE when item will expire.	Only valid when EXPIREMODE = 'NUMBER'.
EXPIREDATE		DATE	Date when item will expire.	Only valid when EXPIREMODE = 'DATE'.

Table F–6 (Cont.) WWSBR_ALL_ITEMS

Column	Null?	Data Type	Description	Notes
IMAGE		VARCHAR2(350)	Image associated with the item.	This value may be a reference (for example, a path to an image stored on the file system or in the content repository) or a unique document name that matches the NAME column in the document table.
KEYWORDS		VARCHAR2(2000)	Keywords for the item.	
URL		VARCHAR2(4000)	URL for a URL type item.	
FILENAME		VARCHAR2(350)	Unique name of the file associated with a file type item.	Matches the NAME column in the document table.
TEXT		CLOB	Text for a text type item.	
FOLDER_LINK_ID		NUMBER	Page ID for a page link type item	Foreign key to: ■ WWSBR_ALL_FOLDERS.ID ■ WWSBR_USER_FOLDERS.ID
FOLDER_LINK_CAID		NUMBER	Page group ID for a page link type item.	Foreign key to: ■ WWSBR_ALL_FOLDERS.CAID ■ WWSBR_USER_FOLDERS.CAID
ACTIVE*	NOT NULL	NUMBER(1)	Indicates the active status of the item.	Valid values: ■ 1 - active ■ 2 - pending approval as new and current version ■ 3 - pending approval as new but not current version ■ 0 - pending approval ■ -1 - deleted ■ -7 - rejected ■ -9 - reject deleted
CAN_BE_CHECKEDOUT		NUMBER(1)	Indicates if the item can be checked out.	Valid values: ■ 0 - cannot be checked out ■ 1 - can be checked out
IS_ITEM_CHECKEDOUT		NUMBER(1)	Indicates if the item is checked out.	Valid values: ■ 0 - item is not checked out ■ 1 - item is checked out

Table F–6 (Cont.) WWSBR_ALL_ITEMS

Column	Null?	Data Type	Description	Notes
CHECKER_ USERNAME		VARCHAR2(256)	User name of the user who checked out the item.	
CHECKOUT_DATE		DATE	Date the item was checked out.	
FULLSCREEN	NOT NULL	NUMBER(1)	Indicates the rendering behavior for an item shown as a link.	Valid values: ▪ 0 - clicking the link displays the item in a new browser window ▪ 1 - clicking the link displays the item in the same browser window
INPLACE	NOT NULL	NUMBER(1)	Indicates the rendering behavior for a text or PL/SQL item.	Valid values: ▪ 0 - a link to the item is displayed; the rendering behavior of the link depends on the value of the FULLSCREEN column ▪ 1 - the item is rendered in-place in the region
CREATEDATE	NOT NULL	DATE	Date the item was created.	
CREATOR	NOT NULL	VARCHAR2(256)	User name of the person who created the item.	
UPDATEDATE		DATE	Date the item was last updated.	
UPDATOR		VARCHAR2(256)	User name of the person who last updated the item.	
SECURITY		VARCHAR2(25)	Indicates whether security has been set for the item.	Valid values: ▪ item - ILS is enabled on the page and security is set at the item level ▪ folder - security is set at the page level
VISIBLE	NOT NULL	NUMBER(1)	Indicates whether the item is hidden.	Valid values: ▪ 0 - item is hidden ▪ 1 - item is shown
SEQUENCE	NOT NULL	NUMBER	Regular sequence of the item if there is no grouping.	
CATEGORY_ SEQUENCE	NOT NULL	NUMBER	Sequence of the item when items are grouped by category.	

Table F–6 (Cont.) WWSBR_ALL_ITEMS

Column	Null?	Data Type	Description	Notes
AUTHOR_ SEQUENCE	NOT NULL	NUMBER	Sequence of the item when items are grouped by author.	
CREATE_DATE_ SEQUENCE	NOT NULL	NUMBER	Sequence of the item when items are grouped by create date.	
ITEMTYPE_ SEQUENCE	NOT NULL	NUMBER	Sequence of the item when items are grouped by item type.	

F.2.6 WWSBR_ALL_NAVIGATION_BARS

Table F–7 lists the columns in the WWSBR_ALL_NAVIGATION_BARS view, which describes all navigation pages viewable by the current user.

Note: An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

Table F–7 WWSBR_ALL_NAVIGATION_BARS

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the navigation page.	
CAID*	NOT NULL	NUMBER	Page group ID for the navigation page.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME	NOT NULL	VARCHAR2(60)	Internal name of the navigation page.	Unique within page group.
DISPLAY_NAME	NOT NULL	VARCHAR2(256)	Display name, or title, of the navigation page.	
STYLE_ID	NOT NULL	NUMBER	ID for the style used by the navigation page.	Foreign key to WWSBR_ ALL_STYLES.ID
STYLE_CAID	NOT NULL	NUMBER	Page group ID for the style used by the navigation page.	Foreign key to WWSBR_ ALL_STYLES.CAID
IS_PORTLET	NOT NULL	NUMBER(1)	Indicates if the page is published as a portlet.	Valid values: ■ 0 - not a portlet ■ 1 - is a portlet
ALIGNMENT		VARCHAR2	DEPRECATED. This column is retained for backward compatibility.	

F.2.7 WWSBR_ALL_PERSPECTIVES

Table F–8 lists the columns in the WWSBR_ALL_PERSPECTIVES view, which describes all perspectives.

Table F–8 WWSBR_ALL_PERSPECTIVES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the perspective.	
CAID*	NOT NULL	NUMBER	Page group ID for the perspective.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
PARENTID		NUMBER	ID of the parent perspective.	0 (zero) if the perspective has no parent
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME		VARCHAR2(60)	Internal name of the perspective.	Unique within parent perspective.
DISPLAY_NAME	NOT NULL	VARCHAR2(240)	Display name, or title, of the perspective.	

F.2.8 WWSBR_ALL_STYLES

Table F–9 lists the columns in the WWSBR_ALL_STYLES view, which describes all styles.

Note: An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

Table F–9 WWSBR_ALL_STYLES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the style.	
CAID*	NOT NULL	NUMBER	Page group ID for the style.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME	NOT NULL	VARCHAR2(65)	Internal name of the style.	Unique within the page group.
DISPLAY_NAME		VARCHAR2(350)	Display name, or title, of the style.	

F.2.9 WWSBR_APPROVER

Table F–10 lists the columns in the WWSBR_APPROVER view, which describes the approvers for every page group approval step. The STEP_ID column corresponds to the STEP_ID column in the WWSBR_CONTENT_AREA_APPROVAL view. For every step, there could be one or more approvers. An approver can be a user or group and this information is stored in the APPROVER_TYPE column.

Table F–10 WWSBR_APPROVER

Column	Null?	Data Type	Description	Notes
ID	NOT NULL	NUMBER	Internal ID, unique to each row.	Foreign key to WWSBR_CONTENT_AREA_APPROVAL.ID
STEP_ID*	NOT NULL	NUMBER	Sequential step ID of an approval process.	
APPROVER_ID*	NOT NULL	NUMBER	User ID of the approver.	
APPROVER_TYPE*	NOT NULL	VARCHAR2(10)	The type of the approver.	Valid values: ▪ USER - approver is a user ▪ GROUP - approver is a group

F.2.10 WWSBR_ATTRIBUTES

Table F–11 lists the columns in the WWSBR_ATTRIBUTES view, which describes all attributes.

Note: An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

Table F–11 WWSBR_ATTRIBUTES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the attribute.	
CAID*	NOT NULL	NUMBER	Page group ID for the attribute.	Foreign key to WWSBR_ALL_CONTENT_AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME	NOT NULL	VARCHAR2(256)	Internal name of the attribute.	Functions as alternate key (unique within page group).
DISPLAY_NAME		VARCHAR2(256)	Display name, or title, of the attribute.	
IS_BASE_ATTRIBUTE	NOT NULL	NUMBER(1)	Indicates that the attribute is a base attribute.	Valid values: ▪ 0 - not a base attribute ▪ 1 - is a base attribute
DATA_TYPE	NOT NULL	VARCHAR2(30)	Data type of the attribute.	
LENGTH		NUMBER(5)	Length of the attribute.	
IS_TRANSLATABLE	NOT NULL	NUBMER(1)	Indicates that the attribute is translatable, meaning that a different value can be stored for each translation. If the attribute is not translatable, the same value is stored for all translations.	Valid values: ▪ 0 - not translatable ▪ 1 - is translatable

F.2.11 WWSBR_CONTENT_AREA_APPROVAL

Table F–12 lists the columns in the WWSBR_CONTENT_AREA_APPROVAL view, which describes the approval definition for a page group or a page. For every approval definition there are x rows corresponding to the x steps defined in the approval process. To determine the approvers for each step, see the WWSBR_APPROVER view.

Table F–12 WWSBR_CONTENT_AREA_APPROVAL

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the approval process	
STEP_ID		NUMBER	ID of the step in the approval process	
CAID	NOT NULL	NUMBER	ID of the page group against which the approval process is defined.	
FOLDER_ID		NUMBER	ID of the page against which the approval process is defined.	
ALL_OR_ANY		VARCHAR2(10)	Indicates if all the approvers need to approve the item for it to be published, or just one.	Valid values: ■ ALL - all approvers must approve the item ■ ANY - only one approver needs to approve the item
PARALLEL_OR_SERIAL		VARCHAR2(10)	Indicates if all the approvers are notified at the same time, or one after the other.	Valid values: ■ P - all approvers are notified at the same time ■ S - approvers are notified one after the other

F.2.12 WWSBR_CONTENT_AREA_ITEM_TYPES

Table F–13 lists the columns in the WWSBR_CONTENT_AREA_ITEM_TYPES view, which describes which item types are available to which page groups. An item type cannot be assigned to an item unless the item type is available within the item's page group.

Table F–13 WWSBR_CONTENT_AREA_ITEM_TYPES

Column	Null?	Data Type	Description	Notes
CAID*	NOT NULL	NUMBER	ID of the page group.	Foreign key to WWSBR_ALL_CONTENT_AREAS.ID
ITEM_TYPE_ID*	NOT NULL	NUMBER	ID of the item type.	Foreign key to WWSBR_ITEM_TYPES.ID
ITEM_TYPE_CAID*	NOT NULL	NUMBER	Page group ID for the item type.	Foreign key to WWSBR_ITEM_TYPES.CAID

F.2.13 WWSBR_FOLDER_ATTRIBUTES

Table F–14 lists the columns in the WWSBR_FOLDER_ATTRIBUTES view, which describes the attributes associated with pages viewable by the current user.

Table F–14 WWSBR_FOLDER_ATTRIBUTES

Column	Null?	Data Type	Description	Notes
FOLDER_CAID*	NOT NULL	NUMBER	Page group ID of the page.	Foreign key to: ▪ WWSBR_ALL_ FOLDERS.CAID ▪ WWSBR_USER_ FOLDERS.CAID
FOLDER_ID*	NOT NULL	NUMBER	ID of the page.	Foreign key to: ▪ WWSBR_ALL_ FOLDERS.ID ▪ WWSBR_USER_ FOLDERS.ID
ATTRIBUTE_ID*		NUMBER	ID of the attribute.	Foreign key to WWSBR_ ATTRIBUTES.ID
ATTRIBUTE_CAID*		NUMBER	Page group ID for the attribute.	Foreign key to WWSBR_ ATTRIBUTES.CAID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	Foreign key to: ▪ WWSBR_ALL_ FOLDERS.LANGUA GE ▪ WWSBR_USER_ FOLDERS.LANGUA GE ▪ WWSBR_ ATTRIBUTES.LAN GUAGE
VALUETYPE	NOT NULL	VARCHAR2(30)	Data type of the attribute.	
VALUE		VARCHAR2(2000)	Value of the attribute.	

F.2.14 WWSBR_FOLDER_PERSPECTIVES

Table F–15 lists the columns in the WWSBR_FOLDER_PERSPECTIVES view, which describes the perspectives associated with pages viewable by the current user.

Table F–15 WWSBR_FOLDER_PERSPECTIVES

Column	Null?	Data Type	Description	Notes
FOLDER_CAID*	NOT NULL	NUMBER	ID of the page's page group.	Foreign key to: ▪ WWSBR_ALL_ FOLDERS.CAID ▪ WWSBR_USER_ FOLDERS.CAID
FOLDER_ID*	NOT NULL	NUMBER	Page group ID for the page.	Foreign key to: ▪ WWSBR_ALL_ FOLDERS.CAID ▪ WWSBR_USER_ FOLDERS.CAID
PERSPECTIVE_ID*	NOT NULL	NUMBER	ID of the perspective.	Foreign key to WWSBR_ ALL_PERSPECTIVES.ID
PERSPECTIVE_ CAID*	NOT NULL	NUMBER	Page group ID for the perspective.	Foreign key to WWSBR_ ALL_ PERSPECTIVES.CAID
PERSPECTIVE_ NAME		VARCHAR2(60)	Internal name of the perspective.	Foreign key to WWSBR_ ALL_ PERSPECTIVES.NAME

F.2.15 WWSBR_FOLDER_TYPE_ATTRIBUTES

Table F–16 lists the columns in the WWSBR_FOLDER_TYPE_ATTRIBUTES view, which describes the attributes that are available within each page type.

Note: An alternative way of uniquely identifying a row in this view is to use the FOLDER_TYPE_ID, FOLDER_TYPE_CAID, ATTRIBUTE_ID, and ATTRIBUTE_CAID columns.

Table F–16 WWSBR_FOLDER_TYPE_ATTRIBUTES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the page type attribute row.	
CAID*	NOT NULL	NUMBER	Page group ID.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
FOLDER_TYPE_ID	NOT NULL	NUMBER	ID of the folder type.	Foreign key to WWSBR_ FOLDER_TYPES.ID
FOLDER_TYPE_ CAID	NOT NULL	NUMBER	Page group ID for the page type.	Foreign key to WWSBR_ FOLDER_TYPES.CAID

Table F–16 (Cont.) WWSBR_FOLDER_TYPE_ATTRIBUTES

Column	Null?	Data Type	Description	Notes
ATTRIBUTE_ID	NOT NULL	NUMBER	ID of the attribute.	Foreign key to WWSBR_ ATTRIBUTES.ID
ATTRIBUTE_CAID	NOT NULL	NUMBER	Page group ID for the attribute.	Foreign key to WWSBR_ ATTRIBUTES.CAID
DEFAULT_VALUE		VARCHAR2(2000)	Default value of the attribute.	The default value is always stored as VARCHAR2, regardless of the attribute's base type.

F.2.16 WWSBR_FOLDER_TYPES

Table F–17 lists the columns in the WWSBR_FOLDER_TYPES view, which describes all page types.

Note: An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

Table F–17 WWSBR_FOLDER_TYPES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the page type.	
CAID*	NOT NULL	NUMBER	Page group ID for the page type.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME		VARCHAR2(30)	Internal name of the page type.	Functions as alternate key (unique within page group).
DISPLAY_NAME		VARCHAR2(256)	Display name, or title, of the page type.	
DESCRIPTION		VARCHAR2(2000)	Description of the page type.	
IS_BASE_FOLDER_ TYPE	NOT NULL	NUMBER	Indicates that the item is a base page type.	Valid values: ■ 0 - not a base page type ■ 1 - is a base page type
BASE_FOLDER_ TYPE_ID		NUMBER	ID of the base page type on which this page type is based.	

F.2.17 WWSBR_ITEM_APPROVAL

Table F–18 lists the columns in the WWSBR_ITEM_APPROVAL view, which describes item approval information for every item that goes through approval. You can use this view to find information like:

- Where is item x in the approval chain?
- Who is currently reviewing item x?

- Which approver rejected item x?

In addition to the information about pending items, you can also use this view to query details on approvals, including approval date, comments, approver, and so on.

Table F–18 WWSBR_ITEM_APPROVAL

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	Internal ID.	
APPROVAL_ INSTANCE_ID	NOT NULL	NUMBER	Internal ID.	
ITEM_ID	NOT NULL	NUMBER	ID of the item pending approval.	
CAID	NOT NULL	NUMBER	ID of the page group that contains the item pending approval.	
LANGUAGE	NOT NULL	VARCHAR2(30)	Language code.	
FOLDER_ID		NUMBER	ID of the page on which the item pending approval appears.	
STEP_ID		NUMBER	ID of the step for which the item requires approval.	
APPROVAL_ID		NUMBER	ID of the approval process for the item.	Foreign key to WWSBR_ CONTENT_AREA_ APPROVAL.ID
CURRENT_ APPROVER_ID	NOT NULL	NUMBER	ID of the approver of the approval step.	For an ANY approval step this value will be the same for all rows of the approval step. For an ALL approval step this value will be the same as APPROVER_ID.
CURRENT_ APPROVER_TYPE		VARCHAR2(10)	The type of the current approver.	Valid values: ▪ USER - approver is a user ▪ GROUP - approver is a group
APPROVAL_DATE		DATE	Date when the item was approved.	
APPROVER_ID		NUMBER	ID of the approver.	Use `wwsec_api.user_ name` to get the user name of the approver.
APPROVER_ COMMENT		VARCHAR2(4000)	Comment specified at the time of approval or rejection.	

Table F–18 (Cont.) WWSBR_ITEM_APPROVAL

Column	Null?	Data Type	Description	Notes
STATUS	NOT NULL	NUMBER(1)	Status of the item.	Valid values: ■ 1 - active ■ 2 - pending approval as new and current version ■ 3 - pending approval as new but not current version ■ 9 - draft ■ 0 - pending approval ■ -1 - deleted ■ -7 - rejected ■ -9 - reject deleted
SUBMITTER		VARCHAR2(256)	ID of the user who submitted the item for approval.	
VALUE1		VARCHAR2(256)	Not used.	

F.2.18 WWSBR_ITEM_ATTRIBUTES

Table F–19 lists the columns in the WWSBR_ITEM_ATTRIBUTES view, which describes the attributes and attribute values associated with items viewable by the current user.

Table F–19 WWSBR_ITEM_ATTRIBUTES

Column	Null?	Data Type	Description	Notes
ITEM_CAID	NOT NULL	NUMBER	Page group ID for the item.	Foreign key to WWSBR_ALL_ITEMS.CAID
ITEM_MASTERID	NOT NULL	NUMBER	Master ID of the item.	Foreign key to WWSBR_ALL_ITEMS.MASTERID
ATTRIBUTE_ID		NUMBER	ID of the attribute.	Foreign key to WWSBR_ATTRIBUTES.ID
ATTRIBUTE_CAID		NUMBER	Page group ID for the attribute.	Foreign key to WWSBR_ATTRIBUTES.CAID
LANGUAGE	NOT NULL	VARCHAR2(30)	Language code.	Foreign key to WWSBR_ALL_ITEMS.LANGUAGE
VALUETYPE	NOT NULL	VARCHAR2(30)	Date type of the attribute.	
VALUE		VARCHAR2(4000)	Value of the attribute.	

F.2.19 WWSBR_ITEM_PERSPECTIVES

Table F–20 lists the columns in the WWSBR_ITEM_PERSPECTIVES view, which describes the perspectives associated with items viewable by the current user.

Table F–20 WWSBR_ITEM_PERSPECTIVES

Column	Null?	Data Type	Description	Notes
ITEM_CAID*	NOT NULL	NUMBER	Page group ID for the item.	Foreign key to WWSBR_ ALL_ITEMS.CAID
ITEM_ID*	NOT NULL	NUMBER	ID of the item.	Foreign key to WWSBR_ ALL_ITEMS.ID
ITEM_MASTERID*		NUMBER	Master ID of the item.	Foreign key to WWSBR_ ALL_ITEMS.MASTERID
PERSPECTIVE_ID*	NOT NULL	NUMBER	ID of the perspective.	Foreign key to WWSBR_ ALL_PERSPECTIVES.ID
PERSPECTIVE_ CAID*	NOT NULL	NUMBER	Page group ID for the perspective.	Foreign key to WWSBR_ ALL_ PERSPECTIVES.CAID
PERSPECTIVE_ NAME		VARCHAR2(60)	Internal name of the perspective.	Foreign key to WWSBR_ ALL_ PERSPECTIVES.NAME

F.2.20 WWSBR_ITEM_TYPE_ATTRIBUTES

Table F–21 lists the columns in the WWSBR_ITEM_TYPE_ATTRIBUTES view, which describes the attributes that are available within each item type.

Note: An alternative way of uniquely identifying a row in this view is to use the ITEM_TYPE_ID, ITEM_TYPE_CAID, ATTRIBUTE_ID, and ATTRIBUTE_CAID columns.

Table F–21 WWSBR_ITEM_TYPE_ATTRIBUTES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the item type attribute row.	
CAID*	NOT NULL	NUMBER	Page group ID.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
ITEM_TYPE_ID	NOT NULL	NUMBER	ID of the item type.	Foreign key to WWSBR_ ITEM_TYPES.ID
ITEM_TYPE_CAID	NOT NULL	NUMBER	Page group ID for the item type.	Foreign key to WWSBR_ ITEM_TYPES.CAID
ATTRIBUTE_ID	NOT NULL	NUMBER	ID of the attribute.	Foreign key to WWSBR_ ATTRIBUTES.ID

Table F–21 (Cont.) WWSBR_ITEM_TYPE_ATTRIBUTES

Column	Null?	Data Type	Description	Notes
ATTRIBUTE_CAID	NOT NULL	NUMBER	Page group ID for the attribute.	Foreign key to WWSBR_ ATTRIBUTES.CAID
HIDDEN_ ATTRIBUTE	NOT NULL	NUMBER(1)	Indicates if the attribute should be hidden.	This column is used for certain built-in item types and is not intended for customer use. Value values: ▪ 0 - is hidden ▪ 1 - not hidden
DEFAULT_VALUE		VARCHAR2(2000)	Default value of the attribute.	The default value is always stored as VARCHAR2, regardless of the attribute's base type.

F.2.21 WWSBR_ITEM_TYPES

Table F–22 lists the columns in the WWSBR_ITEM_TYPES view, which describes all item types.

Note: An alternative way of uniquely identifying a row in this view is to use the NAME, CAID, and LANGUAGE columns.

Table F–22 WWSBR_ITEM_TYPES

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	ID of the item type.	
CAID*	NOT NULL	NUMBER	Page group ID for the item type.	Foreign key to WWSBR_ ALL_CONTENT_ AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
NAME		VARCHAR2(30)	Internal name of the item type.	Functions as alternate key (unique within page group).
DISPLAY_NAME		VARCHAR2(256)	Display name, or title, of the item type.	
DESCRIPTION		VARCHAR2(2000)	Description of the item type.	
IS_BASE_ITEM_TYPE	NOT NULL	NUMBER	Indicates that the item is a base item type.	Only non-base item types can be assigned to items. Valid values: ▪ 0 - not a base item type ▪ 1 - is a base item type
BASE_ITEM_TYPE_ ID		NUMBER	ID of the base item type on which this item type is based.	

F.2.22 WWSBR_SUBSCRIPTION

Table F–23 lists the columns in the WWSBR_SUBSCRIPTION view, which describes user subscription to portal pages and items.

Table F–23 WWSBR_SUBSCRIPTION

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	Internal ID.	
OBJECT_CONTEXT_ID	NOT NULL	NUMBER	ID of the page group to which the subscribed object belongs.	
FOLDER_ID	NOT NULL	NUMBER	ID of the page to which the subscribed object belongs.	
OBJECT_ID		NUMBER	ID of the subscribed item.	In the case of a page subscription, this value is NULL.
GROUP_ID		NUMBER	Not used.	
RECIPIENT_ID	NOT NULL	NUMBER	ID of the user or group who has subscribed to the object.	
RECIPIENT_TYPE		VARCHAR2(10)	Not used.	

F.2.23 WWSBR_USER_FOLDERS

Table F–24 lists the columns in the WWSBR_USER_FOLDERS view, which describes the pages on which the current user has *Manage* privileges or higher.

Table F–24 WWSBR_USER_FOLDERS

Column	Null?	Data Type	Description	Notes
ID*	NOT NULL	NUMBER	Unique identifier of the page within the page group.	
CAID*	NOT NULL	NUMBER	ID of the page group.	Foreign key to WWSBR_ALL_CONTENT_AREAS.ID
LANGUAGE*	NOT NULL	VARCHAR2(30)	Language code.	
PARENT_ID		NUMBER	ID of the parent page.	0 (zero) if root page of the page group.
NAME	NOT NULL	VARCHAR2(60)	Internal name of the page.	Must be unique within parent page.
DISPLAY_NAME	NOT NULL	VARCHAR2(256)	Display name, or title, of the page.	
CATEGORY_ID		NUMBER	ID of the category assigned to the page.	Foreign key to WWSBR_ALL_CATEGORIES.ID
CATEGORY_CAID		NUMBER	Page group ID for the category assigned to the page.	Foreign key to WWSBR_ALL_CATEGORIES.CAID
DESCRIPTION		VARCHAR2(2000)	Description of the page.	
TYPE_ID	NOT NULL	NUMBER	ID of the page type.	Foreign key to WWSBR_FOLDER_TYPES.ID

Table F–24 (Cont.) WWSBR_USER_FOLDERS

Column	Null?	Data Type	Description	Notes
TYPE_CAID	NOT NULL	NUMBER	Page group ID for the page type.	Foreign key to WWSBR_ FOLDER_TYPES.CAID
BASE_TYPE_ID	NOT NULL	NUMBER	The ID of the page base type	Foreign key to WWSBR_ FOLDER_TYPES.ID where WWSBR_ FOLDER_TYPES.IS_ BASE_FOLDER_TYPE=1
IS_PORTLET	NOT NULL	NUMBER(1)	Indicates if the page is published as a portlet.	Valid values: ▪ 0 - not a portlet ▪ 1- is a portlet
IS_CACHING_ON		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
CACHE_MODE		NUMBER(1)	The caching mode for the page.	Valid values: ▪ 2 - no caching ▪ 1 - cache page definition only ▪ 0 - cache page definition and content for x minutes ▪ 4 - cache page definition only at system level ▪ 3 - cache page definition and content at system level for x minutes
SUB_FOLDER_ SEQUENCE		NUMBER	The sequence (order) of the page in its parent page's sub-page region.	
DISPLAY_IN_ PARENT_FOLDER	NOT NULL	NUMBER(1)	Indicates if the page is displayed in its parent page's sub page region.	
ITEMVERSIONING		VARCHAR2(30)	Determines the level of item versioning for the page.	Valid values: ▪ versionnone - no versioning ▪ versionsimple - simple versioning ▪ versionaudit - audit versioning.
STYLE_ID	NOT NULL	NUMBER	ID for the style used by the page.	Foreign key to WWSBR_ ALL_STYLES.ID
STYLE_CAID	NOT NULL	NUMBER	Page group ID for the style used by the page.	Foreign key to WWSBR_ ALL_STYLES.CAID
URL_VALUE		VARCHAR2(4000)	URL value for a URL type page.	

Table F–24 (Cont.) WWSBR_USER_FOLDERS

Column	Null?	Data Type	Description	Notes
SEARCH_VALUE		VARCHAR2(2000)	DEPRECATED. This column is retained for backward compatibility only.	
PLSQL_VALUE		VARCHAR2(2000)	PL/SQL value for a PL/SQL page.	
IMAGE		VARCHAR2(350)	Unique name of the image associated with the page.	Matches the NAME column in the document table.
TITLE_IMAGE_ NAME		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
ROLLOVER_IMAGE_ NAME		VARCHAR2(350)	The unique name of the rollover image associated with the page.	Matches the NAME column in the document table.
BANNER_IMAGE_ NAME		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	.
NAVIGATION_BAR_ ID		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
NAVIGATION_BAR_ CAID		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
IS_PUBLIC	NOT NULL	NUMBER(1)	Indicates that the page is viewable by public users.	Valid values: ■ 0 - not public ■ 1 - is public
ITEM_LEVEL_ SECURITY	NOT NULL	NUMBER(1)	Indicates that item level security is enabled for the page.	Valid values: ■ 0 - ILS disabled ■ 1 - ILS enabled
DISPLAY_FULL_ SCREEN		VARCHAR2	DEPRECATED. This column is retained for backward compatibility only.	
PLSQL_FOLDER_ EXECUTOR		VARCHAR2(30)	For PL/SQL type pages, the database schema used to execute the PL/SQL code.	
CREATEDATE		DATE	Date the page was created.	
CREATOR		VARCHAR2(256)	User name of the person who created the page.	
UPDATEDATE		DATE	Date the page was last updated.	
UPDATOR		VARCHAR2(256)	User name of the person who last updated the page.	

F.2.24 WWSBR_USER_PAGES

Table F–25 lists the columns in the WWSBR_USER_PAGES view, which describes the pages on which the current user has *Manage* privileges or higher. This view is for use with the wwsbr_api.modify folder API and contains more columns that the WWSBR_USER_FOLDERS view.

Table F–25 WWSBR_USER_PAGES

Column	Null?	Data Type	Description	Notes
GUID	NOT NULL	RAW(32)	Globally unique ID of the page.	
ID	NOT NULL	NUMBER	ID of the page.	
SITEID	NOT NULL	NUMBER	Page group ID of the page.	
LANGUAGE	NOT NULL	VARCHAR2(30)	Language code.	
PARENTID		NUMBER	ID of the parent page.	
NAME	NOT NULL	VARCHAR2(60)	Name of the page.	This name is used in path based URLs.
TITLE	NOT NULL	VARCHAR(256)	Display name, or title, of the page.	
SETTINGSSETID	NOT NULL	NUMBER	ID of the style used by the page.	
SETTINGSSETSITEID	NOT NULL	NUMBER	Page group ID of the style used by the page.	
ISPUBLIC	NOT NULL	NUMBER(1)	Indicates that the page is viewable by public users.	Valid values: ▪ 0 - not public ▪ 1 - is public
IMAGE		VARCHAR2(350)	Unique document name of the image associated with the page, for example 6001.JPG.	
ROLLOVERIMAGE		VARCHAR2(350)	Unique document name of the rollover image associated with the page, or the inactive tab image for the tab, for example 6001.JPG.	
TITLEIMAGE		VARCHAR2(350)	Unique document name of the active tab image for the tab, for example 6001.JPG.	
LEADER		VARCHAR2(256)	E-mail address of the page contact.	
DESCRIPTION		VARCHAR2(2000)	Description of the page.	
PRODUCTION		NUMBER(1)	Not used.	
CREATEDATE		DATE	Date the page was created.	
CREATOR		VARCHAR2((256)	User name of the person who created the page.	
UPDATEDATE		DATE	Date the page was last updated.	

Table F–25 (Cont.) WWSBR_USER_PAGES

Column	Null?	Data Type	Description	Notes
UPDATOR		VARCHAR2(256)	User name of the person who last updated the page.	
PUBLISHDATE		DATE	Not used.	
ICON		VARCHAR2(256)	Not used.	
CTXTXT		VARCHAR2(1)	Used internally by search.	
HAVEITEMSECURITY	NOT NULL	NUMBER(1)	Indicates that item level security is enabled for the page.	Valid values: ■ 0 - ILS disabled ■ 1 - ILS enabled
ITEMVERSIONING		VARCHAR2(30)	Indicates the level of item versioning for the page.	Valid values: ■ versionnone - no versioning ■ versionsimple - simple versioning ■ versionaudit - audit versioning
TOPICID		NUMBER	ID of the category assigned to the page.	
TOPIC_SITEID		NUMBER	Page group ID of the category assigned to the page.	
VALUE		VARCHAR2(2000)	For PL/SQL pages, the PL/SQL code. For JSP pages, the JSP source document name. Do not change this value for JSP pages.	
TYPE	NOT NULL	NUMBER	ID of the page type for the page.	
TYPE_SITEID	NOT NULL	NUMBER	Page group ID of the page type for the page.	
BASE_TYPE	NOT NULL	NUMBER	ID of the base page type.	
IS_PORTLET	NOT NULL	NUMBER(1)	Indicates if the page is published as a portlet.	Valid values: ■ 0 - not a portlet ■ 1 - is a portlet
QUOTA		VARCHAR2(5)	Not used.	
SYSPRIV_NAME	NOT NULL	VARCHAR2(200)	The root page from which the page inherits access settings.	
PLSQL_EXECUTOR		VARCHAR2(30)	For PL/SQL type pages, the database schema used to execute the PL/SQL code.	Valid values: ■ $PUBLIC$ ■ $CREATOR$ ■ <database user name>
KEYWORDS		VARCHAR2(2000)	Keywords for the page.	

Table F–25 (Cont.) WWSBR_USER_PAGES

Column	Null?	Data Type	Description	Notes
IS_READY		NUMBER(1)	Indicates that page creation is complete.	Valid values: ▪ 1 - page creation is complete
INHERIT_PRIV		VARCHAR2(200)	The page from which this page inherits its privileges.	Use the following format: `<page group id>/<page id>`
CACHE_MODE		NUMBER(1)	Caching mode for the page.	Valid values: ▪ 2 - no caching ▪ 1 - cache page definition only ▪ 0 - cache page definition and content for x minutes ▪ 4 - cache page definition only at system level ▪ 3 - cache page definition and content at system level for x minutes
CACHE_EXPIRES		NUMBER(38)	Cache period in minutes.	
TEMPLATE_ID		NUMBER(38)	ID of the template on which the page is based.	
TEMPLATE_SITEID		NUMBER(38)	Page group ID of the template on which the page is based.	
ALLOW_PAGE_STYLE		NUMBER(1)	For templates, indicates if pages can use a different style.	Valid values ▪ 1 - allow pages to use different style.
ALLOW_PAGE_ACL		NUMBER(!)	For templates, indicates if pages have different access settings.	Valid values: ▪ 1 - allow pages to have different access settings
DAV_ID	NOT NULL	VARCHAR2(60)		
DAV_LOCK_TOKEN		VARCHAR2(36)		
IS_TEMPLATE	NOT NULL	NUMBER(1)	Indicates that the page is a template.	Valid values: ▪ 1 - is template
INIT_JSPFILE		VARCHAR2(256)	For JSP pages, initial JSP file if the JSP source of the page is a JAR or WAR file.	
UI_TEMPLATE_ID		NUMBER(38)	ID of HTML page skin.	
TEMPLATE_ISPUBLIC	NOT NULL	NUMBER(1)	Indicates if the template is ready to use.	Valid values: ▪ 1 - template is ready to use
CONTAINER_ID	NOT NULL	NUMBER	ID of the container page.	

Table F–25 (Cont.) WWSBR_USER_PAGES

Column	Null?	Data Type	Description	Notes
DEFAULT_ITEM_REGION_ID		NUMBER	ID of the default item region for the page.	
DEFAULT_PORTLET_REGION_ID		NUMBER	ID of the default portlet region for the page.	
ITEMTYPE_INHERIT_FLAGS		NUMBER(1)	For WebDAV, indicates if default item types are inherited from parent page.	Valid values: ▪ 7 - inherit all item types from parent page ▪ 0 - specify all types on this page
REGFILE_ITEMTYPE		RAW(32)	For WebDAV, GUID of default item type for regular files.	
ZIPFILE_ITEMTYPE		RAW(32)	For WebDAV, GUID of default item type for zip files.	
IMAGEFILE_ITEMTYPE		RAW(32)	For WebDAV, GUID of default item type for image files.	
DISPLAYINPARENT	NOT NULL	NUMBER(1)	Indicates if the page is displayed in its parent page's sub page region.	Valid values: ▪ 1 - display page in parent's sub page region
SEQ		NUMBER	The sequence (order) of the page in its parent page's sub page region.	
ALPHABETICAL_SORT	NOT NULL	NUMBER(1)	Indicates that sub pages are displayed in alphabetical order.	Valid values: ▪ 1 - display sub pages in alphabetical order
IS_ITEM_PAGE		NUMBER(1)	Indicates that the template is for items (with item placeholder item)	Valid values: ▪ 1 - is item template
ITEM_PAGE_ID		NUMBER(38)	ID of the item template.	
ITEM_PAGE_SITE_ID		NUMBER(28)	Page group ID of the item template.	
ITEM_PAGE_TABSTRING		VARCHAR2(512)	Tab strings of the item template.	Use the following format: `<tab name>:<sub tab name>:...:<sub tab name>`
INHERIT_ITEM_PAGE		NUMBER(1)	For item template only, indicates that items inherit the parent page's item template.	Valid values: ▪ 1 - inherit parent page's item template
ALLOW_ITEM_PAGE_OVERRIDE		NUMBER(1)	For item template only, indicates that items can have their own item template.	Valid values: ▪ 1 - allow items on the page to have their own item template

Table F–25 (Cont.) WWSBR_USER_PAGES

Column	Null?	Data Type	Description	Notes
HAS_INPLACE_ ITEM	NOT NULL	NUMBER(1)	Indicates if the page or tab has a placeholder item.	Valid values: ■ 1 - has a placeholder item
TIMEOUT		NUMBER	Limit time, in seconds, used to fetch portlets.	
LAST_ITEM_ TEMPLATE_ ASSIGNMENT		DATE	Set by trigger.	

G

Content Management Event Framework Events

This appendix lists and describes the Oracle Portal actions and the CMEF events that they generate. It contains the following sections:

G.1 Actions and Events for Items

Table G–1 lists the actions performed when adding and updating items and the events raised by those actions.

Table G–1 Adding and Editing Items

Description	Action	Event	State	Object Class
An item is added and published immediately	ADD_ITEM	INSERT	PUBLISHED	ITEM
An item is added but set to be published some time in the future	ADD_ITEM	INSERT	NOT_PUBLISHED	ITEM
A portlet instance is added	ADD_ITEM	INSERT	PUBLISHED	ITEM
An item is updated but there is no change to the state of the item (that is, it stays as published or unpublished)	EDIT_ITEM	UPDATE	GENERAL	ITEM

Table G–1 (Cont.) Adding and Editing Items

Description	Action	Event	State	Object Class
An item that was originally set to be published later is edited, and chosen to be published now	EDIT_ITEM	UPDATE	PUBLISHED	ITEM
A published item is updated and set to be published some time in the future	EDIT_ITEM	UPDATE	UNPUBLISHED	ITEM
An expired item is updated and unexpired	EDIT_ITEM	UPDATE	PUBLISHED	ITEM
An item is hidden	HIDE_ITEM	UPDATE	UNPUBLISHED	ITEM
An item that was hidden is shown	SHOW_ITEM	UPDATE	PUBLISHED	ITEM
An item is expired	EXPIRE_ITEM	UPDATE	UNPUBLISHED	ITEM
An item is unexpired	UNEXPIRE_ITEM	UPDATE	PUBLISHED	ITEM
An item is checked-out	CHECK_OUT_ITEM	UPDATE	CHECKED_OUT	ITEM
An item is checked-in	CHECK_IN_ITEM	UPDATE	CHECKED_IN	ITEM

Table G–2 lists the actions performed when copying and moving items and the events raised by those actions.

Table G–2 Copying and Moving Items

Description	Action	Event	State	Object Class
An item is copied - the copied item is in the published state	COPY_ITEM	INSERT	PUBLISHED	ITEM
An item is copied - the copied item is in the unpublished state	COPY_ITEM	INSERT	NOT_PUBLISHED	ITEM
An item is moved (within a page or to a different page or page group) - the item is in the published state	MOVE_ITEM	DELETE	PURGED	ITEM
--	MOVE_ITEM	INSERT	PUBLISHED	ITEM
An item is moved (within a page or to a different page or page group) - the item is in the unpublished state	MOVE_ITEM	DELETE	PURGED	ITEM
--	MOVE_ITEM	INSERT	NOT_PUBLISHED	ITEM

Table G–3 lists the actions performed when deleting and restoring items and the events raised by those actions.

Table G–3 Deleting and Restoring Items

Description	Action	Event	State	Object Class
An item is deleted in a page group that is set to retain deleted items	DELETE_ITEM	DELETE	MARKED_FOR_ DELETE	ITEM
An item is deleted in a page group that is not set to retain deleted items	DELETE_ITEM	DELETE	PURGED	ITEM
An item in the unpublished state is undeleted (item is either hidden or expired or is set to be published in the future)	UNDELETE_ITEM	UPDATE	NOT_PUBLISHED	ITEM
An item in the published state is undeleted	UNDELETE_ITEM	UPDATE	PUBLISHED	ITEM

Table G–4 lists the actions performed when defining privileges for items and the events raised by those actions.

Table G–4 Defining Item Privileges

Description	Action	Event	State	Object Class
Specify item level access privileges	SPECIFY_ITEM_ ACL	UPDATE	GENERAL	ITEM
Add user to item ACL	ADD_ITEM_ACL	UPDATE	GENERAL	ITEM
User's privilege is updated on an item	UPDATE_ITEM_ ACL	UPDATE	GENERAL	ITEM
Delete user from item ACL	DELETE_ITEM_ ACL	UPDATE	GENERAL	ITEM
Choose to inherit security from parent page	INHERIT_ITEM_ ACL	UPDATE	GENERAL	ITEM

Table G–5 lists the actions performed when working with item versions and the events raised by those actions.

Table G–5 Item Versioning

Description	Action	Event	State	Object Class
A new version of an item is added and is set to be the current version	ADD_ITEM_ VERSION	INSERT	PUBLISHED	ITEM
A new version of an item is added but is not set to be the current version	ADD_ITEM_ VERSION	INSERT	NOT_PUBLISHED	ITEM
A version of an item is deleted in a page group that is set to retain deleted items	DELETE_ITEM_ VERSION	DELETE	MARKED_FOR_ DELETE	ITEM
A version of an item is deleted in a site that is not set to retain deleted items	DELETE_ITEM_ VERSION	DELETE	PURGED	ITEM

Table G–5 (Cont.) Item Versioning

Description	Action	Event	State	Object Class
The current version of an item is changed (the previous current version and the new current version are in the published state)	SWITCH_ITEM_ VERSION	UPDATE	UNPUBLISHED	ITEM
--	SWITCH_ITEM_ VERSION	UPDATE	PUBLISHED	ITEM
The current version of an item is changed (the previous current version and the new current version are in the unpublished state)	SWITCH_ITEM_ VERSION	UPDATE	GENERAL	ITEM
--	SWITCH_ITEM_ VERSION	UPDATE	NOT_PUBLISHED	ITEM
The current version of an item is changed (the previous current version was in the published state and the new current version is in the unpublished state)	SWITCH_ITEM_ VERSION	UPDATE	UNPUBLISHED	ITEM
--	SWITCH_ITEM_ VERSION	UPDATE	NOT_PUBLISHED	ITEM
The current version of an item is changed (the previous current version was in the unpublished state and the new current version is in the published state)	SWITCH_ITEM_ VERSION	UPDATE	GENERAL	ITEM
--	SWITCH_ITEM_ VERSION	UPDATE	PUBLISHED	ITEM

Table G–6 lists the actions performed when working with item approvals and the events raised by those actions.

Table G–6 Item Approvals

Description	Action	Event	State	Object Class
An item is added in pending mode	SUBMIT_ITEM_ FOR_APPROVAL	INSERT	NOT_PUBLISHED	ITEM
An item is added in draft mode	ADD_DRAFT_ ITEM	INSERT	NOT_PUBLISHED	ITEM
A pending item is edited by an approver	EDIT_ITEM_BY_ APPROVER	UPDATE	GENERAL	ITEM
An item is approved but is pending approval from the next approver	APPROVE_ITEM_ STEP	UPDATE	GENERAL	ITEM
An item is approved by all approvers (the approved item overwrites the current, published version)	APPROVE_ITEM	UPDATE	PUBLISHED	ITEM

Table G–6 (Cont.) Item Approvals

Description	Action	Event	State	Object Class
An item is approved by all approvers (the approved item becomes the new current version)	APPROVE_ITEM	INSERT	PUBLISHED	ITEM
--	APPROVE_ITEM	UPDATE	UNPUBLISHED	ITEM
An item is approved by all approvers (the approved item becomes a new, but not current version)	APPROVE_ITEM	INSERT	NOT_PUBLISHED	ITEM
An item is updated and submitted for approval	SUBMIT_ITEM_ FOR_APPROVAL	UPDATE	NOT_PUBLISHED	ITEM
An item is rejected	REJECT_ITEM	UPDATE	GENERAL	ITEM
A rejected item is resubmitted for approval	SUBMIT_ITEM_ FOR_APPROVAL	UPDATE	GENERAL	ITEM
A draft item is updated (attributes such as author, description, and so on are updated)	EDIT_DRAFT_ ITEM	UPDATE	GENERAL	ITEM
A draft item is updated (the publish date is updated to publish the item some time in the future)	EDIT_DRAFT_ ITEM	UPDATE	UNPUBLISHED	ITEM
A draft item is updated (the publish date is updated to publish the item immediately)	EDIT_DRAFT_ ITEM	UPDATE	PUBLISHED	ITEM
A draft item is submitted for approval	SUBMIT_ITEM_ FOR_APPROVAL	UPDATE	NOT_PUBLISHED	ITEM
An item is copied into a page where draft items are enabled	COPY_DRAFT_ ITEM	INSERT	NOT_PUBLISHED	ITEM
An item is copied to a page and is submitted for approval	SUBMIT_ITEM_ FOR_APPROVAL	INSERT	NOT_PUBLISHED	ITEM

G.2 Actions and Events for Pages

Table G–7 lists the actions performed when adding and editing pages and the events raised by those actions.

Table G–7 Adding and Editing Pages

Description	Action	Event	State	Object Class
A page is created - there are two messages: one for the page that is added and one for the portlet instance for the default navigation page	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
A page based on a template is created. The template contains one item and one portlet	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM

Table G–7 (Cont.) Adding and Editing Pages

Description	Action	Event	State	Object Class
--	DELETE_PAGE[1]	DELETE	PURGED	PAGE
--	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
A page is renamed. If the root page is renamed there are two messages: one for the page and one for the page group.	RENAME_PAGE	UPDATE	GENERAL	PAGE
A page property is updated (for example the description)	EDIT_PAGE	UPDATE	GENERAL	PAGE
A page is copied (the page contains one portlet instance and one item)	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	COPY_ITEM	INSERT	PUBLISHED	ITEM
--	COPY_ITEM	INSERT	PUBLISHED	ITEM
A page is moved under a different parent	MOVE_PAGE	DELETE	PURGED	PAGE
--	MOVE_PAGE	INSERT	PUBLISHED	PAGE
A page is translated	EDIT_PAGE	UPDATE	GENERAL	PAGE
--	EDIT_PAGE	INSERT[2]	PUBLISHED	PAGE
--	EDIT_PAGE	UPDATE	GENERAL	PAGE
Versioning is enabled or disabled for a page	EDIT_PAGE	UPDATE	GENERAL	PAGE

[1] As the page designer steps through the wizard, a page is created that includes the default navigation page. However, as soon as it is determined that the page is based on a template, this page is deleted and a new page (based on the template) is created instead.

[2] The first update is for the default language, the last is for the current translation. The message of interest is the insert for the current translation.

Table G–8 lists the actions performed when deleting pages and the events raised by those actions.

Table G–8 Deleting Pages

Description	Action	Event	State	Object Class
A page is deleted (if the page has sub-pages, a message is logged for each deleted sub-page, however no messages are logged for deleted items)	DELETE_PAGE	DELETE	PURGED	PAGE

Table G–9 lists the actions performed when defining privileges for pages and the events raised by those actions.

Table G–9 Defining Page Privileges

Description	Action	Event	State	Object Class
ILS is enabled or disabled on a page	UPDATE_PAGE_ ACL	UPDATE	GENERAL	PAGE
--	UPDATE_PAGE_ ILS	UPDATE	GENERAL	PAGE
A page is displayed to public users	UPDATE_PAGE_ ACL	UPDATE	GENERAL	PAGE
--	UPDATE_PAGE_ PUBLIC_SETTING	UPDATE	GENERAL	PAGE
A user is added to a page ACL	ADD_PAGE_ACL	UPDATE	GENERAL	PAGE
A user is deleted from a page ACL	DELETE_PAGE_ ACL	UPDATE	GENERAL	PAGE
A user's privilege is changed in a page ACL	UPDATE_PAGE_ ACL	UPDATE	GENERAL	PAGE
A page is set to inherit its ACL from its parent	INHERIT_PAGE_ ACL	UPDATE	GENERAL	PAGE
An ACL is specified at the page level (the page does not inherit its ACL from its parent)	UPDATE_PAGE_ ACL	UPDATE	GENERAL	PAGE
--	SPECIFY_PAGE_ ACL	UPDATE	GENERAL	PAGE

G.3 Actions and Events for Tabs

Table G–10 lists the actions performed when adding and editing tabs and the events raised by those actions.

Table G–10 Adding and Editing Tabs

Description	Action	Event	State	Object Class
A tab is added	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
A tab is added in a region that contains two items	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
--	DELETE_ITEM[1]	DELETE	PURGED	ITEM
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
--	DELETE_ITEM	DELETE	PURGED	ITEM
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
A tab is hidden	HIDE_ITEM	UPDATE	UNPUBLISHED	ITEM
A hidden tab is shown	SHOW_ITEM	UPDATE	PUBLISHED	ITEM
A tab is deleted	DELETE_PAGE	DELETE	PURGED	PAGE
--	DELETE_ITEM	DELETE	PURGED	ITEM
A tab is renamed	RENAME_PAGE	UPDATE	GENERAL	PAGE
A tab property is updated (for example, the display name)	EDIT_PAGE	UPDATE	GENERAL	PAGE

[1] Items on the original page are moved to the tab. So the item is first deleted from the original page and then added to the new tab.

G.4 Actions and Events for Page Groups

Table G–11 lists the actions performed when adding and editing page groups and the events raised by those actions.

Table G–11 Adding and Editing Page Groups

Description	Action	Event	State	Object Class
A page group is added	ADD_PAGE_GROUP	INSERT	GENERAL	PAGE_GROUP
A page group is renamed	RENAME_PAGE_GROUP	UPDATE	GENERAL	PAGE_GROUP

Table G–12 lists the actions performed when deleting and purging page groups and the events raised by those actions.

Table G–12 Deleting and Purging Page Groups

Description	Action	Event	State	Object Class
A page group is deleted. Messages are not logged for any items or sub-pages.	DELETE_PAGE_GROUP	DELETE	PURGED	PAGE_GROUP
Deleted items are purged from a page group (one message for each item and one message for the actual purge operation)	DELETE_ITEM	DELETE	PURGED	ITEM
--	PURGE_DELETED_ITEM	DELETE	PURGED	PAGE_GROUP
Expired items are purged from a page group (one message for each item and one message for the actual purge operation)	DELETE_ITEM	DELETE	PURGED	ITEM
--	DELETE_ITEM	DELETE	PURGED	ITEM
--	PURGE_EXPIRED_ITEM	DELETE	PURGED	PAGE_GROUP
Choose to not retain deleted items at the page group level, thus purging all existing retained deleted items (one message for each item and one message for the actual purge operation)	DELETE_ITEM	DELETE	PURGED	ITEM
--	PURGE_DELETED_ITEM	DELETE	PURGED	PAGE_GROUP

Table G–13 lists the actions performed when defining privileges for page groups and the events raised by those actions.

Table G–13 Defining Page Group Privileges

Description	Action	Event	State	Object Class
A user is added to a page group ACL	ADD_PAGE_ GROUP_ACL	UPDATE	GENERAL	PAGE_GROUP
An user's privilege is changed in a page group ACL	UPDATE_PAGE_ GROUP_ACL	UPDATE	GENERAL	PAGE_GROUP
A user is deleted from a page group ACL	DELETE_PAGE_ GROUP_ACL	UPDATE	GENERAL	PAGE_GROUP

Table G–14 lists the actions performed when installing translations for page groups and the events raised by those actions.

Table G–14 Page Group Translations

Description	Action	Event	State	Object Class
A translation is installed	TRANSLATE_ PAGE	INSERT	PUBLISHED	PAGE

G.5 Actions and Events for Attributes

Table G–15 lists the actions performed when working with attributes and the events raised by those actions.

Table G–15 Attributes

Description	Action	Event	State	Object Class
An attribute is added. The data type of the attribute is stored in the STATE field, for example, DATA_TYPE_TEXT.	ADD_ATTRIBUTE	INSERT	DATA_TYPE_ TEXT	ATTRIBUTE
An attribute is updated	EDIT_ATTRIBUTE	UPDATE	DATA_TYPE_ TEXT	ATTRIBUTE
An attribute is promoted (the message is the same regardless of how many items use the attribute)	MOVE_ ATTRIBUTE	UPDATE	GENERAL	ATTRIBUTE
An attribute is deleted (the message is the same even if there are items using the attribute)	DELETE_ ATTRIBUTE	DELETE	DATA_TYPE_ TEXT	ATTRIBUTE

G.6 Actions and Events for Item Types

Table G–16 lists the actions performed when working with item types and the events raised by those actions.

Table G–16 Item Types

Description	Action	Event	State	Object Class
A custom item type is created	ADD_ITEM_TYPE	INSERT	GENERAL	ITEM_TYPE
An item type is edited and changes are made in the Main tab (if you edit the name, a different message is generated, see following description)	EDIT_ITEM_TYPE	UPDATE	MAIN_TAB	ITEM_TYPE
An item type is renamed	RENAME_ITEM_ TYPE	UPDATE	MAIN_TAB	ITEM_TYPE
An attribute is attached to an item type	EDIT_ITEM_TYPE	UPDATE	ATTRIBUTES_ TAB	ITEM_TYPE
An attribute is detached from an item type	DETACH_ ATTRIBUTE	UPDATE	ATTRIBUTES_ TAB	ITEM_TYPE
Attribute default values are updated	EDIT_ITEM_TYPE	UPDATE	ATTRIBUTES_ TAB	ITEM_TYPE
A procedure is attached to an item type	EDIT_ITEM_TYPE	UPDATE	PROCEDURES_ TAB	ITEM_TYPE
A procedure attached to an item type is edited and new parameters are added	EDIT_ITEM_TYPE	UPDATE	PROCEDURES_ TAB	ITEM_TYPE
An item type is promoted	MOVE_ITEM_ TYPE	UPDATE	GENERAL	ITEM_TYPE
An item type is deleted (the message is the same even if there are items based on the item type)	DELETE_ITEM_ TYPE	DELETE	PURGED	ITEM_TYPE

G.7 Actions and Events for Page Types

Table G–17 lists the actions performed when working with page types and the events raised by those actions.

Table G–17 Page Types

Description	Action	Event	State	Object Class
A custom page type is created	ADD_PAGE_ TYPE	INSERT	GENERAL	PAGE_TYPE
A page type is edited and changes are made in the Main tab (if you edit the name, a different message is generated, see following description)	EDIT_PAGE_ TYPE	UPDATE	MAIN_TAB	PAGE_TYPE
A page type is renamed	RENAME_PAGE_ TYPE	UPDATE	MAIN_TAB	PAGE_TYPE
An attribute is attached to a page type	EDIT_PAGE_ TYPE	UPDATE	ATTRIBUTES_ TAB	PAGE_TYPE
Attribute default values are added	EDIT_PAGE_ TYPE	UPDATE	ATTRIBUTES_ TAB	PAGE_TYPE
An attribute is detached from a page type	DETACH_ ATTRIBUTE	UPDATE	ATTRIBUTES_ TAB	PAGE_TYPE

Table G–17 (Cont.) Page Types

Description	Action	Event	State	Object Class
A procedure is attached to a page type	EDIT_PAGE_ TYPE	UPDATE	PROCEDURES_ TAB	PAGE_TYPE
A procedure attached to a page type is edited and new parameters are added	EDIT_PAGE_ TYPE	UPDATE	PROCEDURES_ TAB	PAGE_TYPE
A page type is promoted	MOVE_PAGE_ TYPE	UPDATE	GENERAL	PAGE_TYPE
A page type is deleted	DELETE_PAGE_ TYPE	DELETE	PURGED	PAGE_TYPE

G.8 Actions and Events for Categories

Table G–18 lists the actions performed when working with categories and the events raised by those actions.

Table G–18 Categories

Description	Action	Event	State	Object Class
A category is added	ADD_CATEGORY	INSERT	GENERAL	CATEGORY
A category is renamed	RENAME_ CATEGORY	UPDATE	GENERAL	CATEGORY
--	RENAME_PAGE	UPDATE	GENERAL	PAGE
A category is edited and changes are made in the Main tab	EDIT_CATEGORY	UPDATE	GENERAL	CATEGORY
An image is attached to a category	EDIT_ CATEGORY_ IMAGE	UPDATE	GENERAL	CATEGORY
An image is detached from a category	DELETE_ CATEGORY_ IMAGE	UPDATE	GENERAL	CATEGORY
A category is promoted	MOVE_ CATEGORY	UPDATE	GENERAL	CATEGORY
A category is deleted	DELETE_ CATEGORY	DELETE	GENERAL	CATEGORY

G.9 Actions and Events for Perspectives

Table G–19 lists the actions performed when working with perspectives and the events raised by those actions.

Table G–19 Perspectives

Description	Action	Event	State	Object Class
A perspective is added	ADD_ PERSPECTIVE	INSERT	GENERAL	PERSPECTIVE
A perspective is renamed	RENAME_ PERSPECTIVE	UPDATE	GENERAL	PERSPECTIVE
--	RENAME_PAGE	UPDATE	GENERAL	PAGE

Table G–19 (Cont.) Perspectives

Description	Action	Event	State	Object Class
A perspective is edited and changes are made in the Main tab	EDIT_ PERSPECTIVE	UPDATE	GENERAL	PERSPECTIVE
An image is attached to a perspective	EDIT_ PERSPECTIVE_ IMAGE	UPDATE	GENERAL	PERSPECTIVE
An image is detached from a perspective	DELETE_ PERSPECTIVE_ IMAGE	UPDATE	GENERAL	PERSPECTIVE
A perspective is promoted	MOVE_ PERSPECTIVE	UPDATE	GENERAL	PERSPECTIVE
A perspective is deleted	DELETE_ PERSPECTIVE	DELETE	GENERAL	PERSPECTIVE

G.10 Actions and Events for Templates

Table G–20 lists the actions performed when working with templates and the events raised by those actions.

Table G–20 Templates

Description	Action	Event	State	Object Class
A page based on a template is created[1]	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
--	DELETE_PAGE	DELETE	PURGED	PAGE
--	ADD_PAGE	INSERT	PUBLISHED	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
--	ADD_ITEM	INSERT'	PUBLISHED	ITEM
A page is edited to be based on a template, resulting in content from one region being deleted	DELETE_ITEM	DELETE	PURGED	ITEM
--	EDIT_PAGE	UPDATE	GENERAL	PAGE
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
--	ADD_ITEM	INSERT	PUBLISHED	ITEM
An item is added to a template. There is a message for the item on the template itself and then additional messages for each time the item is replicated on pages based on the template	ADD_ITEM	INSERT	PUBLISHED	ITEM
An item is deleted from a template. There is a message for the item on the template itself and then additional messages (with different page IDs and page group IDs) for each time the item is deleted from pages based on the template.	DELETE_ITEM	DELETE	PURGED	ITEM

Table G–20 (Cont.) Templates

Description	Action	Event	State	Object Class
A page is detached from a template. One or more messages, as shown, is produced for each item and portlet that was on the template	COPY_ITEM	INSERT	PUBLISHED	ITEM
A template, on which one or more pages are based, is deleted.[2]	DELETE_PAGE	DELETE	PURGED	PAGE
--	COPY_ITEM	INSERT	PUBLISHED	ITEM
--	COPY_ITEM	INSERT	PUBLISHED	ITEM
--	DELETE_ITEM	DELETE	PURGED	ITEM
--	COPY_ITEM	INSERT	PUBLISHED	ITEM

[1] The first two inserts are for the page creation. The next delete is because that page gets deleted when you attach the template. The next insert is for the new page based on the template. The following two item inserts are for item replication from the template. There will be an insert event for every item and portlet on the template.

[2] The delete event is for the deletion of the actual template. The first two inserts are when the replicated items on a page based on the template are copied over. There will be an insert message for each item on the template. The following delete and insert events are for a non-replicated item on the page based on the template. There will be a delete and corresponding insert event for every non-replicated item on the page.

Glossary

About mode

An optional **portlet Show mode** that displays information about the portlet's copyright, version, and author.

access control list

See **ACL**.

ACL

Access Control List. A list of groups and users authorized for specific access to an **object**.

action

In the context of **CMEF**, an action occurs when something happens in a **portal** (either through the Oracle Portal user interface, **WebDAV**, or PL/SQL **API**s). Actions within a portal trigger CMEF **event**s, which cause Oracle Portal to write messages to the WWSBR_EVENT_Q **queue**. A **subscriber** can consume these messages and perform its own actions depending on what those messages are.

add-in

See **extension**.

advanced search

A search engine that enables users to:

- Find content that contains any or all terms in the search string.
- Search selected **page group**s, or search across all page groups.
- Restrict a search to a particular **page**, **category**, **perspective**, **item type**, or **attribute**.

If **Oracle Text** is installed and enabled, all text attributes and the actual content of documents and URLs are searched. If Oracle Text is not installed, or is disabled, the following **metadata** is searched: item attributes (Display Name, Description, Keywords, and Author), page attributes (Display Name, Description, and Keywords) and category/perspective attributes (Display Name and Description).

If Oracle Text is installed and enabled, you can also use advanced search to perform near, soundex, and fuzzy searches.

See also **search portlet**. Contrast with **basic search** and **custom search**.

API

Application Programming Interface. A set of exposed data structures and functions that an application can use to invoke services on a portal **object**, such as a **portlet**, **page**, or **page group**. Note that Oracle Portal APIs are exposed through the **PDK** available from the Oracle Portal Developer Kit (PDK) page on OTN:

`http://www.oracle.com/technology/products/ias/portal/pdk.html`

application

INTERNAL: Obsolete terminology. DO NOT USE to refer to a provider of portlets built using the **Portlet Builder** (for example, forms, reports, charts). Use **database provider** instead.

Application Programming Interface

See **API**.

Application Service Provider

See **ASP**.

approval notification

A message in the Notification portlet indicating that a user whose content requires approval has created or updated an **item**. The intended recipients of approval notifications (**approver**s) relating to a particular page group or page are identified in an **approval process**. An approver may respond to the notification by approving or rejecting the item in question.

approval process

A series of one or more steps that determines which users (or **group**s) need to review content that requires approval before it can be published. Each step in an approval process must have one or more **approver**s. Routing to the approvers can be in serial (one at a time) or in parallel (all at once), and each step can be defined to require a response (either an approval or a rejection) by any one member or by all members. Once the required number of responses is received during a step, the process continues to the next step. The process ends when the item is rejected, or the final step is reached and the document is approved.

approver

A user identified in a page group or page **approval process**, either explicitly or implicitly through group membership, as someone who needs to review content that requires approval before it can be published. An approver may choose to approve or reject such content.

ASP

Application Service Provider. A service that provides remote hosting of applications, maintaining and operating the hardware, software, and other resources required to run the applications.

attribute

A portal **object** that stores information (or **metadata**) about an **item** or **page**: for example, Create Date, Expire Date, or Author. A **page group administrator** can create custom attributes to extend the functionality of **item type**s and **page type**s. For example, a base attribute on a file is Display Name; a custom attribute might be a check box to indicate whether the file is confidential. Custom attributes are useful for assigning unique, searchable identifiers to items.

authenticated user

A user who is logged on to a **portal**. By default, authenticated users can access and, based on privileges granted to the user, act on certain portal **object**s, such as **page**s.

Contrast with **public user**s, who can access public content only.

authorization

The evaluation of security constraints to send a message or make a request. Authorization uses specific criteria—authentication and restriction—to determine whether the request should be permitted.

authorized user

See **authenticated user**.

banner

See **region banner**. See also **navigation page**.

base attribute

See **attribute**.

base item type

See **item type**.

base page type

See **page type**.

basic search

A search engine that enables users to find content that contains a specific search string.

If **Oracle Text** is installed and enabled, all text **attribute**s and the actual content of documents and URLs are searched. If Oracle Text is not installed, or is disabled, the following **metadata** is searched: item attributes (Display Name, Description, Keywords, and Author), page attributes (Display Name, Description, and Keywords) and category/perspective attributes (Display Name and Description).

See also **search portlet**. Contrast with **advanced search** and **custom search**.

basic search box item

A navigation item that users can add to a **page** to enable other users to perform basic searches. The search box can initiate a search of all **page group**s or a specified subset of page groups.

batch job

The process of running a **Portlet Builder portlet** in the background using the Oracle Portal batch job facility. An end user can run a portlet in batch mode by selecting options on the portlet's **personalization form**. Batch processing is useful if the portlet is based on a large amount of data, if the portlet displays many rows of data, or if the job may take a long time to run.

bind variable

A variable in a SQL statement that must be replaced with a valid value or address of a value in order for the statement to execute successfully. Portlet developers typically use bind variables (for example, dept) to display a **parameter entry field** in a **portlet**'s **personalization form**. The entry field enables end users to choose the data that the portlet will display.

bookmark

See **favorite**.

breadcrumbs

See **page path item**.

Builder page

See **Portal Builder page**.

bulk load

See **zip file item**.

caching

The act of storing frequently accessed information, typically Web pages or **portlet**s in Oracle Portal, in a location where it can be accessed quickly to avoid frequent content generation. For example, **Oracle Web Cache** stores dynamically-generated portlets in its memory, then serves them to the **PPE** when there is a request for the specified portlet. This storage reduces the total time spent handling the request by avoiding connections to the back-end database and other Web sites.

See also **expiry-based caching**, **invalidation-based caching**, **system level caching**, **user level caching**, and **validation-based caching**.

calendar

A **portlet** created with the **Portlet Builder** that displays the results of a SQL **query** in calendar format.

call interface

The call interface displays the arguments that were selected when a **Portlet Builder portlet** was originally created or last edited.

category

A predefined **attribute** used to group or classify **page**s, **item**s, and **portlet**s in a **page group**. A category helps users answer the question: *What is this item or page?* For example, in a travel page group, you might have categories for maps, snapshots, and hotel reviews. Users can assign only one category to a particular item or page.

See also **perspective**.

chart

A **portlet** created with the **Portlet Builder** that displays the results of a SQL **query** as a chart, such as a bar chart, pie chart, or line chart. Bar charts are based on at least two table or view columns: one that identifies the bars on the chart and another that calculates the size of the bars on the chart.

check out/check in

A mechanism that allows a user to lock an item, by checking it out, so that other users cannot edit that same item. This prevents users from overwriting each others changes. After editing the item, the user releases it by checking it back in, making it available again for other users to edit.

child object

An object that is part of a hierarchy. For example, sub-pages, sub-categories, and sub-perspectives are child objects of a **page**, **category**, and **perspective** respectively.

See also **manifest**.

child page

In **Oracle Instant Portal**, a page created beneath a **top-level page**. Child pages are listed along the left side of the top-level page to which they belong, in the **navigation area**, and usually support the top-level page's main theme. You can add, delete, re-position, or edit child pages right in the navigation area, assuming you have the appropriate privileges.

CHTML

Compact HTML. A subset of **HTML** recommendations, designed for small devices.

classification

Categories and perspectives are used to classify the content of a **page** or **item** so that it is easy for users to locate that content during a search.

See also **category** and **perspective**.

CLOB

Character Large Object. A Large Object (LOB) datatype whose value is composed of character data corresponding to the database character set. A CLOB can be indexed and searched by the **Oracle Text** search engine.

cluster

A database **object** used to store tables that are related to one another and that are often joined together in the same area on a disk.

CMEF

Content Management Event Framework. A feature that enables you to extend Oracle Portal's content management event functionality by adding programmatic hooks to certain pre-defined portal **event**s. The framework publishes these events to an Oracle Streams Advanced Queuing (AQ) **queue**. This allows third party programs to subscribe to these events and to use the **API**s to extend the functionality of the portal. CMEF should not be confused with the portlet events feature.

CMEF event

An action within a portal triggers a corresponding **CMEF** event. A developer can write code that responds to CMEF events.

community

See **Portal Community**.

compact HTML

See **CHTML**.

content area

In **Oracle Instant Portal**, refers to the area in which a **page**'s **item**s (or content) appear, on the right side of the screen.

content contributor

A user who has the appropriate privileges to add **item**s to a **page**. Appropriate page privileges include *Manage Content* and *Manage Items With Approval*. Appropriate item privileges include *Manage*, *Edit*, and *View*.

content item type

A means of classifying the actual content of an **item** that is being uploaded to a **page**, such as a document, text, or an image.

Built-in content item types are:

- **file item** and **simple file item**
- **text item** and **simple text item**
- **URL item** and **simple URL item**
- **image item** and **simple image item**
- **image map item**
- **PL/SQL item** and **simple PL/SQL item**
- **page link item** and **simple page link item**
- **zip file item**

See also **item type**. Contrast with **navigation item type**.

Content Management Event Framework

See **CMEF**.

content repository

The Oracle Portal schema within the **Oracle Metadata Repository** that contains the content and **metadata** associated with a particular portal instance.

Contribute privileges

A privilege level on an **Oracle Instant Portal** page that allows the user to add, edit, move, or delete items on the page. A user with Contribute privileges also has all **View privileges**.

CSS

Cascading Style Sheet. A simple mechanism for adding style, such as fonts, colors, and spacing, to Web documents.

current version

The version of an **item** that is displayed on the **page**. The current version is not necessarily the most recent version of the item.

See also **versioning**.

custom attribute

See **attribute**.

custom item type

See **item type**.

custom page type

See **page type**.

custom provider

A type of **provider** that enables you to create and maintain **portlet**s that access customer-specific content or applications. You can build custom portlets in Oracle Portal either declaratively or programmatically.

custom search

A search engine that enables users to define a variety of searches against information stored in the Oracle Portal schema of the **Oracle Metadata Repository**. By editing the defaults of the Custom Search portlet, you can define unique search forms and search results pages that meet the specific search requirements, or configure portlets that execute and return results based on predefined search criteria.

If **Oracle Text** is installed and enabled, all text attributes and the actual content of documents and URLs are searched. If Oracle Text is not installed, or is disabled, the following **metadata** is searched: item attributes (Display Name, Description, Keywords, and Author), page attributes (Display Name, Description, and Keywords) and category/perspective attributes (Display Name and Description).

See also **search portlet**. Contrast with **advanced search** and **basic search**.

DAD

Database Access Descriptor. A set of values that specify how an application connects to an Oracle database to fulfill an HTTP request. The information in the DAD includes the user name (which also specifies the schema and the privileges), password, connect-string, error log file, standard error message, and Globalization Support parameters such as language, date format, date language, and currency.

Database Access Descriptor

See **DAD**.

database administrator

See **DBA**.

database object

See **object**.

database provider

A type of **provider** that is written as a PL/SQL **stored procedure** and is used to create **portlet**s that reside in the database. One example of a database provider is a provider built using the **Portlet Builder** to provide form, report, and chart portlets.

Contrast with **Web provider**.

data portlet

A **portlet** built using the **Portlet Builder** that displays data in a spreadsheet format.

DAV

See **WebDAV**.

DBA

Database Administrator. A user belonging to the DBA **group**. By default, members in the DBA group have access to all Oracle Portal product pages, and have the Manage privilege for all **page**s, **page group**s, **database provider**s, and administration.

default language

The language in which a **page group** is originally created. After creating the page group, you can add **translation**s to it to enable users to add content in different languages.

default subscriber

The base **subscriber** that is installed along with the install of Oracle Portal.

de-militarized zone

See **DMZ**.

dequeue

The operation of retrieving a CMEF message from an Oracle Streams AQ **queue**.

Contrast with **enqueue**.

developer

A user who builds **portlet**s for others to include on their pages. Relies heavily on the APIs to extend the capabilities of Oracle Portal; may frequently consult the **Portal Knowledge Exchange** or the forums for advice or inspiration.

Developer Services

See **Portal Developer Services**.

DIP

Directory Integration Platform. The provisioning platform provided by **Oracle Internet Directory** to synchronize different directories and directory enabled applications.

direct access URL

Obsolete terminology. See **path-based URL**.

Directory Information Tree

See **DIT**.

Directory Integration Platform

See **DIP**.

display name

An **object**'s external name used throughout Oracle Portal, for example, in the **Navigator**, on **page**s, and in the page editor. When the object is published as a **portlet**, the display name is used as the title of the portlet in the **Portlet Repository**.

Distinguished Name

See **DN**.

DIT

Directory Information Tree. A hierarchical tree-like structure in **Oracle Internet Directory** consisting of the **DN**s of the entries.

DN

Distinguished Name. The unique name of a directory entry in **Oracle Internet Directory**. It includes all the individual names of the parent entries back to the root. The DN tells you exactly where the entry resides in the directory's hierarchy. This hierarchy is represented by a directory information tree (**DIT**).

DMZ

De-militarized Zone. A computer host or small network inserted as a "neutral zone" between a company's private network and the outside public network. It prevents outside users from getting direct access to a server that has company data. A DMZ is an optional and more secure approach to a **firewall** and effectively acts as a **proxy server** as well. (The term comes from the geographic buffer zone that was set up between North Korea and South Korea following the UN police action in the early 1950s.)

document control

See **check out/check in**.

draft item

An **item** that has been added to a **page** but has not yet been submitted for approval. In **View mode**, a draft item is visible only to its author. When a draft item is ready for approval, the author can submit it, at which point the **approval process** is triggered. Draft items can be added to a page only if approvals are enabled for the **page group**.

durable URL

A URL that uses the item's GUID (globally unique ID) to uniquely identify it. An item's GUID does not change so its durable URL will not break when the item is edited, renamed, moved, or imported to a different portal instance.

Contrast with **path-based URL**.

dynamic page

A **portlet** created with the **Portlet Builder** that displays dynamic content on a page. The dynamic page wizard enables you to specify one or many PL/SQL blocks within HTML code to create a page. This code executes every time an end user requests the page.

dynamic URL

A URL that contains a query string (one or more parameters and the characters ? and &).

Contrast with **path-based URL**

edge side includes

See **ESI**.

Edit Defaults mode

An optional **portlet Show mode** that enables administrators to set the defaults of a portlet for all users.

Contrast with **Edit mode**.

Edit mode

1. Page editing: Edit mode enables an authenticated user with appropriate privileges to set **page** properties and to add, modify, or delete **portlet**s and **item**s on the page. To switch to Edit mode, the user clicks an **Edit** link on the page. There are three Edit mode views: **Graphical view**, **Layout view**, and **List view**.

   See also **Mobile Preview mode** and **Pending Items Preview mode**.

2. Portlets: An optional portlet **Show mode** that enables personalization of the portlet for each user, for each instance.

Contrast with **Edit Defaults mode**.

3. **Oracle Instant Portal**: Opposite of **View mode**. When the **Edit mode handle bar** is clicked, thus displaying the edit toolbar and the Add Item and Add Page buttons, the page is said to be in Edit mode. This is the only state in which action may be taken upon the page.

Edit mode handle bar

In **Oracle Instant Portal**, the gray icon that one clicks to display the edit toolbar. When the edit toolbar is displayed, the page is in Edit mode. Users with View privileges on the page do not see the Edit mode handle bar.

email item

In **Oracle Instant Portal**, an item represented by a yellow envelope and, optionally, a title on the page. When the user clicks the envelope, an e-mail editor is opened displaying a blank e-mail. The e-mail is pre-populated with the address specified by the item's creator.

enqueue

The operation of publishing messages to an Oracle Streams AQ **queue**.

Contrast with **dequeue**.

Oracle Enterprise Manager

Oracle Enterprise Manager is a component of the Oracle Fusion Middleware that enables administrators to manage Oracle Fusion Middleware services through a single environment. For example, an administrator can use Oracle Enterprise Manager to monitor the services that make up an Oracle Portal instance, including **HTTP** services, the **PPE**, the Oracle database, **provider**s, and **Oracle Ultra Search**.

enterprise portal

A common, integrated starting point that provides personalized access to relevant enterprise information sources. Enterprise portals enable site visitors to personalize their view of the resources available on the public Internet.

ESI

Edge Side Includes. A markup language to enable partial page caching (**PPC**) of HTML fragments.

event

An action within a portal triggers a corresponding event. Page designers can specify what should happen when these events occur, for example, by specifying that a particular event forces the reloading of the current **page**, and passes **parameter**s to the newly loaded page.

Event servlet

The Event servlet implements the functionality of Oracle Portal to allow for dynamic page navigation when accessing an event enabled **portlet**. The Event servlet runs in the same container as the **PPE**.

expandable rich text item

In **Oracle Instant Portal**, a type of text item designed to conserve space on the page. When in **View mode**, an expandable rich text item is represented by a white envelope, a title, and a summary. When the user clicks the envelope, the item expands to reveal the text associated with the item. Compare to a **rich text item**, in which the full text of the item is never hidden. Both expandable rich text and rich text items may contain images, hyperlinks, and tables, as well as any valid HTML code.

expiration period

The number of days after which, or an exact date on which, an **item** expires. After an item expires, it is viewable only by the item's or **page**'s owner and the **page group administrator** in **Edit mode**. Expired items are removed from the database during a **system purge** of all expired items.

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.

Oracle Web Cache uses both expiry-based caching and invalidation-based caching. Any data saved in Oracle Web Cache is considered valid until it is invalidated or it expires. For example, if expiry-based caching is specified for a fully assembled page, the page content remains valid in the cache for the specified retention period before it needs to be regenerated.

See also **invalidation-based caching** and **validation-based caching**.

expiry notification

A message automatically sent to a user or **group** indicating that an **item** on the **page** is about to expire. The notification is set up by the **page group administrator**.

explicit object

An object which is explicitly selected, from the Navigator or Bulk Actions, for export.

See also **manifest** and **referenced object**.

export

A method of creating a set of files (**transport set**) that contains **page group**s, **page**s, **portlet**s, and other content from a single Oracle Portal instance. You can then **import** this set of files into another Oracle Fusion Middleware instance.

eXtensible Markup Language

See **XML**.

extension

A Java class that extends the functionality of **Oracle JDeveloper**. For example the **PDK** includes an extension to aid with the development of portlets with Oracle JDeveloper.

external application

An application external to Oracle Portal that is typically launched from the External Applications **portlet**. As each external application is configured by a **portal administrator**, users simply supply their user name and password information. The **Oracle Application Server Single Sign-On** will present these credentials for future authentication challenges.

external object

An object which is an external dependency of an **explicit object**. External objects ensure that the explicit objects perform on the target portal.

See also **manifest**.

favorite

A hyperlink in the Favorites **portlet** that provides quick access to a frequently visited **URL**, either inside or outside your company firewall. An **authenticated user** can personalize the Favorites portlet with his or her own preferred set of frequently accessed URLs.

Favorite Content area

An area on the **Oracle Instant Portal** home page in which a user's favorite content appears. Users select items for this area by clicking an icon that looks like a house beside the item. The Favorite Content area is different for each user.

favorite group

A collection of **favorite**s (and favorite groups) that are usually logically related.

Federated Portal Adapter

See **FPA**.

file item

A type of **item** that a user can add to a **page**. When a user adds a file item to a page, the file is uploaded into the Oracle Portal schema of the **Oracle Metadata Repository** and is displayed as a hyperlink on the page. When a user clicks the **display name** link, the file may be downloaded to the user's computer or displayed in the user's Web browser, depending on the file type and the configuration of the browser.

In **Oracle Instant Portal** file items are represented by a rectangle with a red asterisk at the top and, optionally, a title and summary. When the user clicks the icon, the file associated with the item opens in a second browser window.

firewall

A system (either hardware or software) that acts as an intermediary to protect a set of computers or networks from outside attack. It regulates access to computers on a local area network from outside, and regulates access to outside computers from within the local area network. A firewall can work either by acting as a **proxy server** that forwards requests so that the requests behave as though they were issued by the firewall machine, or by examining requests and attempting to eliminate suspect calls.

form

A **portlet** created with the **Portlet Builder** that provides a transactional interface to one or more database tables, views, or procedures. For example, you can the Portlet Builder to build a form for entering new employee information into your Human Resources database.

See also **master-detail form**.

FPA

Federated Portal Adapter. The Federated Portal Adapter is a module in the portal instance (written in both Java and PL/SQL) that receives **SOAP** messages for a **Web provider**, parses the SOAP, and then dispatches the messages to a **database provider** as PL/SQL procedure calls. In effect, the Federated Portal Adapter makes a database provider behave exactly the same way as a Web provider, allowing users to distribute their database providers across database servers. All remote providers can be treated as Web providers, hiding their implementation (database or Web) from the user. The most common use is to share database providers (including page groups) owned by one portal instance among other portal instances.

frame driver

A **portlet** created with the **Portlet Builder** consisting of a Web page divided into two frames. A driving frame contains a SQL **query** that drives the contents of the second (target) frame.

Full Screen mode

An optional **portlet Show mode** that provides more content than can be shown in the portlet when it is sharing a page with other portlets.

function

A PL/SQL subprogram that performs a specified sequence of actions and then returns a value. Functions are usually small blocks of code written to perform a specific task within the scope of a larger application.

In a **page**, end users execute functions by clicking the title of a PL/SQL or custom item.

gist

An **Oracle Text** summary consisting of the document paragraphs that best represent the overall subject matter. You can use such summaries to skim the main content of the text or assess your interest in the text's subject matter.

global privilege

A **privilege** that grants a certain level of access to a user or **group** on all **object**s of a particular type. For example, you could grant a Web Designer group Manage privileges on all **style**s.

grantee

A user who is given privileges on an **object** by another user.

Graphical view

A page editing view that renders **page** content in-place on the page. Graphical view enables you to view pages and **item**s as they appear on the finished page as you edit.

Contrast with **Layout view** and **List view**.

group

A collection of Oracle Portal users who typically share a common need or interest; for example, Human Resources, Accounting, and so on. Groups make it easy to grant access to an **object** (such as a **page** or **portlet**) to several users at once. You can also use groups to implement user roles by assigning role-related privileges to a group, then adding users in that role. **Oracle Internet Directory** tracks the membership of Oracle Portal groups.

group owner

A user who has the privilege to add or delete members from a **group**, or to delete the group itself. Groups can have more than one owner.

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.

Handheld Device Markup Language

See **HDML**.

HDML

Handheld Device Markup Language. A simple language to define hypertext-like markup content and applications for handheld devices with small display.

Help mode

An optional **portlet Show mode** that displays usage information about the functionality of the portlet.

hierarchy

A **portlet** created with the **Portlet Builder** that displays data from a self-referencing table or view. At least two columns in the table must share a recursive relationship. A hierarchy can contain up to three levels and display data such as employees in an organization chart or the hierarchical relationship between menus in a Web site.

home page

The **page**, defined within Oracle Portal, that typically displays when logging on or when a user clicks a Home **smart link item**. The **portal administrator** chooses this page for **public user**s; **authenticated user**s may choose their own. If the portal administrator enables **mobile page** design, he or she can specify a separate mobile home page to display when the portal is accessed from a mobile device.

hosted site

See **stripe**.

HTML

Hyper Text Markup Language. A format for encoding hypertext documents that may contain text, graphics, and references to programs and other hypertext documents.

HTML content layout

A type of **HTML Template** that uses item-level **substitution tag**s to define a formatting scheme for individual **region**s. The HTML content layout repeats itself for each **item** or **portlet** in the region.

Contrast with **HTML page skin**.

HTML page skin

A type of **HTML Template** that uses page-level **substitution tag**s to control the appearance of the area surrounding page content. You can apply HTML page skins to **page**s or **Portal Template**s.

Contrast with **HTML content layout**.

HTML Template

A portal **object** built using your own **HTML** code that wraps around your **page** or **region** content. You can use an Oracle Portal wizard or any third party HTML editor to build HTML Templates.

See also **HTML content layout** and **HTML page skin**. Contrast with **Portal Template**.

HTTP

Hyper Text 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 WebLogic Server and its clients.

Hyper Text Markup Language

See **HTML**.

Hyper Text Transfer Protocol

See **HTTP**.

IDE

Integrated Development Environment. A visual tool containing editors, debuggers, screen painters, object browsers, and the like.

ILS

Item Level Security. A mechanism that controls granular access to **item**s on a given **page**. ILS authorizes item managers to grant explicit item access to users and **group**s that take precedence over page-level privileges.

image item

A type of **item** that a user can add to a **page**. You can add images in JPEG, GIF, or PNG formats.

In **Oracle Instant Portal**, as opposed to the images that are added as part of a **rich text item** or **expandable rich text item**, image items are designed to standalone on the page. An image item is added through the Add Item icon; an image that is part of a text item is added through the Insert Image window.

image map item

A type of **item** that a user can add to a **page**. An image map is a single image with hotspots that, when clicked, link to other **URL**s. For example, you can create an image map of the world in which each continent is hyperlinked to more information about the continent.

import

A method of transporting content and **object**s (for example, **page group**s, **page**s, and **portlet**s) into an Oracle Portal instance. For example, you can import a page, its associated **style**, and its contents from one instance of Oracle Portal to another.

index

An optional structure associated with a table used to locate rows of the table quickly, and (optionally) to guarantee that every row is unique.

in-place editing

In **Oracle Instant Portal**, the ability to add items or pages in context, as opposed to having to create these objects elsewhere and then add them to the page or portal.

Integrated Development Environment

See **IDE**.

internal image name

The name used to identify an image that has been uploaded to the portal. Uploaded images can be reused within the portal by referencing their internal names.

internal provider

A type of **provider** that makes **page group object**s (**page**s, **navigation page**s, and so on) available to instances of Oracle Portal.

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

Oracle Web Cache uses both expiry-based caching and invalidation-based caching. Any data saved in Oracle Web Cache is considered valid until it is invalidated or it expires. When the information cached in Oracle Web Cache becomes inaccurate, it must be invalidated. For example, **page metadata** saved in Oracle Web Cache is invalidated when a page designer changes the page structure or when user privileges change. Likewise, a **portlet instance** is invalidated whenever it is personalized by an end user.

See also **expiry-based caching** and **validation-based caching**.

item

1. An individual piece of content (text, hyperlink, image, and so on) that resides 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 Oracle Portal schema of the **Oracle Metadata Repository**. Items are rendered on the page according to the layout, **style**, and **attribute** display defined for the item region.

   See also **item type**.

2. In **Oracle Instant Portal**, the means through which content is added to a page. Available item types are **rich text item**, **expandable rich text item**, **file item**, **URL item**, **email item**, and **image item**.

item ID

The local database reference to the content of an **item**. An item ID value is used in custom **item type**s to pass items to PL/SQL procedures. The function uses the item ID to access the content of the item.

item level security

See **ILS**.

item metadata

The stored information or **attribute**s of an **item**.

item placeholder item

A type of **item** that a user can add to a **page**. An item placeholder identifies where the content from items that use a **Portal Template** for items will display in relation to the rest of the template content.

item type

An object that defines the contents of an **item** and the **attribute**s that are stored (**metadata**) about an item. Base item types included with Oracle Portal are categorized as **content item type**s and **navigation item type**s.

Custom item types are item types created by **page group administrator**s to extend the functionality provided by base item types and store additional attribute information about items.

item versioning

See **versioning**.

J2EE

Java 2 Platform, Enterprise Edition. A platform that enables application developers to develop, deploy, and manage multitier, server-centric, enterprise level applications. The J2EE 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 J2EE **portlet**s and expose them through **Web provider**s.

See also **Oracle WebLogic Server**.

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.

Java 2 Platform, Enterprise Edition

See **J2EE**.

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**. The **Portlet Builder** enables you to use JavaScript to create routines that validate entry fields in **form**s and **personalization form**s. You can also create JavaScript event handlers for entry fields and buttons on forms.

Java Specification Request

See **JSR 168**.

JavaServer Page

See **JSP**.

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**.

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`.

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.

keyword

An **attribute** used to provide additional information about a **page** or **item** so that users can easily locate the page or item during a search.

Layout view

A page editing view that enables you to add, arrange, and remove **region**s on the **page**. You can also hide, show, delete, or move content in this view.

Contrast with **Graphical view** and **List view**.

LBR

Load-balancing router. A very fast network device that distributes Web requests to a large number of servers. It provides portal users with a single published address, without their having to send each request to a specific **middle tier** server.

LDAP

Lightweight Directory Access Protocol. A standard for representing and accessing user and group profile information.

level

An object used to provide structure to **mobile page**s and as a way to limit the amount of content displayed on the smaller screens of mobile devices. Users drill down into the levels on a mobile page to view more content.

library

A collection of one or more PL/SQL or Java program units. Libraries can be referenced by several applications simultaneously.

Lightweight Directory Access Protocol

See **LDAP**.

Link mode

An optional **portlet Show mode** that enables portlets to render themselves on mobile devices, such as cellular telephones.

List view

A page editing view that displays a listing of all **page** content and provides options that enable you to perform actions (delete, move, copy, and so on) on multiple **object**s.

Contrast with **Graphical view** and **Layout view**.

list of objects item

A type of navigation **item** that a user can add to a **page** to list objects (for example, pages and **perspective**s) as a drop-down list or as links (with or without associated images).

list of values

See **LOV**.

load-balancing router

See **LBR**.

local provider group

The collection of **provider**s that are defined within an instance of Oracle Portal. Provider groups make it easier to share providers defined or registered within one instance of Oracle Portal with other Oracle Portal instances.

See also **provider group**. Contrast with **remote provider group**.

lock

1. A setting automatically applied to a **Portlet Builder portlet** when it is being edited. The setting prevents other users from editing the portlet.
2. In **WebDAV** the action of preventing other users from editing a file. Locking a file in a WebDAV client checks out the corresponding **item** in the portal itself.

login/logout link item

A type of navigation **item** that a user can add to a **page** to enable other users to log in or log out of the portal.

LOV

List of values. A **portlet** created with the **Portlet Builder** that enables developers to add selectable values to entry fields in **form**s. A single list of values can be displayed in different formats, such as combo boxes, radio buttons, or check boxes.

Manage privileges

A privilege level on an **Oracle Instant Portal** page that allows the user to edit, move, or delete a top-level page and all its child pages. A user with Manage privileges on a portal's home page is considered an administrator of that portal. A user with Manage privileges also has all Contribute and View privileges.

manifest

The list of objects in a **transport set** and their dependents, which provides a granular level of control over the import mode.

master-detail form

A **portlet** created with the **Portlet Builder** that displays a master table row and multiple detail rows within a single HTML page. Values in the master row determine which detail rows are displayed for querying, updating, inserting, and deleting.

See also **form**.

master item ID

An identifier for an **item** that is the same for all versions of that item.

menu

A **portlet** created with **Portlet Builder** that displays a Web page containing options that end users can click to navigate to other menus, other Portlet Builder portlets, or **URL**s.

message payload

Provides details about **CMEF event**s such as the portal **action** that triggered the event, the type of event (INSERT, UPDATE, or DELETE), and the **state** of the event.

metadata

Data that describes data. For example **item attribute**s store item metadata, such as display name, description, and author.

middle tier

Part of the Oracle Fusion Middleware architecture that handles **HTTP** user requests by forwarding them to the appropriate portal database or **provider**, assembles portal **page**s, and manages **caching** of portal content.

MIME type

Multipurpose Internet Mail Extension type. A message format used on the Internet to describe the contents of a message. MIME is used by **HTTP** servers to describe the type of content being delivered.

mobile page

A type of **page** that enables **page designer**s to produce pages specifically for mobile devices, for example, cellular phones.

Mobile Preview mode

A preview mode that enables you to preview how your page will look on a mobile device.

mobile XML

See **Oracle Application Server Wireless XML**.

Model-View-Controller

See **MVC**.

mod_weblogic

The **Oracle HTTP Server** module that manages the communication between the Oracle HTTP Server and **Oracle WebLogic Server**.

mod_plsql

The **Oracle HTTP Server** module that handles the database connections made from the Oracle HTTP Server. It enables PL/SQL database procedures to generate **HTTP** responses containing formatted data and **HTML** code that can display in a Web browser.

MVC

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 area

In **Oracle Instant Portal**, the area along the left side of the page in which the search box and a list of the page's child pages appear.

navigation item type

A means of providing navigation and access to portal-specific functions.

Built-in navigation item types include:

- **smart link item**
- **smart text item**
- **login/logout link item**
- **basic search box item**
- **list of objects item**
- **object map link item**
- **page path item**
- **page function item**

See also **item type**. Contrast with **content item type**.

navigation page

A special purpose **page** within a **page group** that is typically embedded on other pages or **Portal Template**s to implement standard user interface effects such as navigation bars and banners. Often contains **navigation item type**s for navigation within the portal.

Navigator

A feature for locating **object**s and interacting with Oracle Portal. Provides access to **object**s to which the user has privileges, such as **page group**s, **provider**s, and database objects.

New Content area

An area on the **Oracle Instant Portal** home page in which content added anywhere in the portal over the last 24 hours is gathered.

nondefault subscriber

A **subscriber** that has a **stripe** on a hosted Oracle Portal provided by an **ASP**.

object

1. Portal object: A structure such as a **page group**, **portlet**, **page**, or **style**.
2. Database object: An Oracle database structure such as a **table**, procedure, or **trigger**. These objects can be created using Oracle Portal wizards or Oracle database commands.

object map link item

A type of navigation **item** that a user can add to a page to display a map of **object**s that are available in the portal.

OEM

See **Oracle Enterprise Manager**.

OID

See **Oracle Internet Directory**.

OmniPortlet

A **Web provider** that provides portlets that can display spreadsheet, XML, and Web Service data as tabular, chart, news, bullet, and form layouts.

Oracle Metadata Repository

An Oracle database that contains schemas and business logic used by Fusion Middleware components (including Oracle Portal) and other pieces of the infrastructure.

Oracle Portal uses a schema within the Oracle Metadata Repository to store and manage the content and **metadata** associated with the portal instance. This is sometimes referred to as the **content repository**.

Oracle Portal

A component of Oracle Fussion Middleware used for the development, deployment, administration, and configuration of enterprise class **portal**s. Oracle Portal incorporates a portal building framework with self-service publishing features to enable you to create and manage information accessed within your portal.

Oracle Portal Developer Kit

See **PDK**.

Oracle Application Server Single Sign-On

A component of Oracle Fussion Middleware that enables users to log in to all features of the Oracle Fusion Middleware product suite, as well as to other Web applications, using a single user name and password. Oracle Portal is integrated with Oracle Application Server Single Sign-On as a **partner application** and delegates authentication to it.

Oracle Web Cache

A component of Oracle Fussion Middleware that improves the performance, scalability, and availability of frequently used Web sites. By storing frequently accessed URLs in memory, Oracle Web Cache eliminates the need to repeatedly process requests for those URLs on the Web server. Oracle Web Cache uses **invalidation-based caching** and is integrated with Oracle Portal for improved performance.

See also **portal cache**.

Oracle Application Server Wireless

A component of Portal used to deliver information and applications to mobile devices. Using Oracle Application Server Wireless, you can create custom portal sites that use different kinds of content, including Web pages, custom Java applications, and **XML**-based applications. Oracle Application Server Wireless sites make this diverse information accessible to mobile devices without your having to rewrite the content for each target device platform.

Oracle Application Server Wireless XML

A device independent markup language used for communication between Oracle Portal and **Oracle Application Server Wireless**.

Oracle Portal

See **Oracle Portal**.

Oracle Portal Verification Service

(Previously known as Portal Studio). A major component of Portal Center (`http://www.oracle.com/technology/products/ias/portal`) that is specifically tuned to the needs of the portal developer. This site provides developers a way to test and display remote portlets exposed as Web or WSRP providers without having to install their own copy of Oracle Portal. Developer's portlets reside on their servers and must be accessible over the Internet. You can access Oracle Portal Verification Service directly at `http://portalstandards.oracle.com/portal/page/portal/OracleHostedWSRPPortal/Welcome`.

Oracle Enterprise Manager

See **Oracle Enterprise Manager**.

Oracle HTTP Server

The **Web server** component of Oracle Fusion Middleware, 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 Portal. Extensions to the Oracle HTTP Server support Java **servlet**s, **JSP**s, Perl, PL/SQL, and CGI applications.

Oracle Instant Portal

A product built on top of OracleAS Portal, designed to provide a common place to share and exchange content for smaller groups of users.

Oracle Instant Portal administrator

A user who has full privileges over the entire portal. Any user with **Manage privileges** on the portal's home page is considered an **Oracle Instant Portal** administrator for that portal. The PORTAL and ORCLADMIN users, created during the installation process, are Oracle Instant Portal administrators for *all* portals. Only an Oracle Instant Portal administrator can change the banner or style, manage users, and create new **top-level page**s.

Oracle Internet Directory

The repository for storing Oracle Portal user credentials and **group** memberships. By default, the **Oracle Application Server 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 PartnerNetwork Solutions Catalog

(http://solutions.oracle.com/) A collection of information on Oracle Partner joint products and services. The Solutions Catalog includes information on partners who offer Oracle Portal related products and services.

Oracle Streams Advanced Queuing

Oracle Streams Advanced Queuing (AQ) provides database-integrated message queuing functionality. It is built on top of Oracle Streams and leverages the functions of the Oracle database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S).

Oracle Technology Network

See **OTN**.

Oracle Text

A feature of Oracle9*i* and later that provides advanced search and retrieval services on content stored in an Oracle repository. It is fully integrated into Oracle Portal to provide users with the ability to perform a full text search and retrieve content managed within the Oracle Portal schema of the **Oracle Metadata Repository**. It also provides automatic grouping and classification of results by **gist** and **theme**.

Oracle Ultra Search

An **Oracle Text**-based application that supports crawling, indexing, and federated searching of multiple, heterogeneous repositories including databases, file systems, **Web server**s, and e-mailing list archives.

Contrast with **search portlet**.

Oracle WebLogic Server

A scalable, enterprise-ready Java Platform, Enterprise Edition (Java EE) application server. The WebLogic Server (WLS) infrastructure supports the deployment of many types of distributed applications and is an ideal foundation for building applications based on Service Oriented Architectures (SOA). SOA is a design methodology aimed at maximizing the reuse of application services.

The WebLogic Server complete implementation of The Sun Microsystems Java EE 5.0 specification provides a standard set of APIs for creating distributed Java applications that can access a wide variety of services, such as databases, messaging services, and connections to external enterprise systems. End-user clients access these applications using Web browser clients or Java clients.

Oracle Workflow

Oracle Workflow enables you to automate and continuously improve business processes, routing information of any type according to easily-changeable business rules to users both inside and outside a company's enterprise.

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/.

Overwrite mode

See **Replace on Import mode**.

package

A database **object** consisting of a PL/SQL specification and a body. The specification includes the data types and subprograms that can be referenced by other program units. The body includes the actual implementation of the package.

page

A portal **object** that contains **portlet**s and **item**s. Each time you display a page, it is dynamically assembled and formatted according to the portlets and layout chosen for that page.

See also **page type**.

page designer

A user with the Manage privilege on a page (also known as a page manager). A user with this privilege can perform any action on the page and can create sub-pages under the page. The page designer is often responsible for designing the layout (or region configuration) of the page and assigning privileges on the page to other users (for example, to determine who can add content to the page).

The scope of a page designer's control over a page may be limited if the page is based on a template.

page function item

A type of navigation **item** that a user can add to a **page**. A page function is a procedure call that a user can add to a custom **page type**. If there are no page functions associated with the current page, the page function item does not display.

page group

A portal **object** that groups and sets properties of related portal objects, such as **page**s, **style**s, **navigation page**s, and **perspective**s. Page groups typically contain a hierarchy of pages and sub-pages for organizing content.

page group administrator

A user who has full privileges over an entire **page group**. Page group administrators set up and maintain the page group; designate page owners; and create a taxonomy. Page group administrators can also view and manage all the **page**s in the page group.

page group map

A hierarchical representation of all **page group**s in a portal, which enables users to access individual **page**s within the page group. The page group map is tailored for each user; only the pages the user is authorized to view or edit are displayed.

page group quota

See **quota**.

page link item

A type of **item** that a user can add to a **page**. A page link provides a route using a hyperlink to another page within the **portal**. When the user clicks the **display name** link, the page referenced by the item is displayed in the user's browser.

page manager

See **page designer**.

page metadata

Stored information or attributes about a **page**, which is used by Oracle Portal to set its layout and cache.

page path item

A type of navigation **item** that a user can add to a **page**. A page path is a chain of page reference names. Page paths, often called breadcrumbs, describe the complete directory path.

page toolbar

In page **Edit mode**, the links at the top of the page that enable you to edit various aspects of the **page**, switch editing views (for example, from **Graphical view** to **Layout view**), edit **page group** properties, and so on.

page type

A portal **object** that defines the content of a **page** and the information that is stored about a page. Base page types included with Oracle Portal are: **standard page**, **mobile page**, **PL/SQL page**, **JSP**, and **URL page**. Custom page types are page types created by **page group administrator**s to extend the functionality provided by base page types and store additional information about pages.

Parallel Page Engine

See **PPE**.

parameter

A value passed between **page**s and **portlet**s, or between portlets.

A page parameter is a page level parameter (created by a page designer) whose values can be mapped to portlet parameters.

A portlet parameter is declared by a provider. Page designers map page parameters to portlet parameters. When the **PPE** requests a portlet from a provider, only the portlet parameters that the portlet declared and mapped to page parameters are sent.

parameter entry field

A field on a **personalization form** that enables end users to enter values that will be passed to an Oracle Portal **portlet**.

partial page caching

See **PPC**.

partner application

An application that has delegated its authentication to **Oracle Application Server Single Sign-On**. If registered with the OracleAS Single Sign-On Server, users can log in to multiple **partner application**s using a single log in page. In a given session, once users have been authenticated by the OracleAS Single Sign-On Server, they won't need to log in again to access additional partner applications.

path aliasing

See **path-based URL**.

path-based URL

A path-based URL identifies the path taken through the **portal** to get to a particular **object**. It is an easy-to-read URL but as it contains the names of portal objects, the URL becomes invalid if the name of any object within the path changes.

For example, the path-based URL to access a top-level **page** (sample_page) of the **page group** MyPageGroup:

```
http://mymachine.mycompany.com:5000/portal/page/mydad/MyPageGroup/sample_page
```

Contrast with **durable URL**.

PDK

Oracle Portal Developer Kit. The development framework used to build and integrate Web content and applications with Oracle Portal. 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. The PDK is regularly updated on **Portal Center** (`http://portalcenter.oracle.com/`) to provide developers with the latest tools and techniques.

See also **PDK-Java**, **PDK-PL/SQL**, and **PDK-URL Services**.

PDK-Java

A toolkit for implementing **portlet**s in Java and adding portal features. Used to declaratively turn your existing Java **servlet**s, **JSP**s, and Web services into portlets.

See also **PDK**. Contrast with **PDK-PL/SQL**.

PDK-PL/SQL

A set of articles, samples, and services that enable PL/SQL programmers to easily create portlets and extend them by using PL/SQL **API**s.

See also **PDK**. Contrast with **PDK-Java**.

PDK-URL Services

A utility for declaratively turning secured and public Web content into **portlet**s. These services are capable of dynamically passing parameters to target **URL**s, clipping and reformatting content, and providing Oracle Application Server Single Sign-On for applications requiring form-based or basic authentication. These services also allow developers to take any application written in any language and easily create integrated portlets. PDK-URL Services takes the URL of an application, parses the content, and uses the **PDK-Java** framework to create a portlet.

See also **PDK**.

Pending Approvals Monitor

A portlet that enables you to list pending approvals in the page groups that you administer. You can list the pending approvals by approver, date, page group, or submitter.

Pending Items Preview mode

A preview mode that enables you to view items that are awaiting approval. This mode can be used by content contributors to preview items they have added before they are approved, and by approvers to preview items before they approve or reject them.

See also **Edit mode**.

personalization form

A page that prompts end users for values to pass to a **Portlet Builder portlet**. End users can view the personalization form for a portlet, if one has been created, by clicking the portlet's **Personalize** link.

personal page

An area within Oracle Portal where **authenticated user**s can store personal content and share it with other users. The **portal administrator** can choose to create a personal page for a user when creating a user account.

perspective

A cross-category grouping of **item**s. Perspectives help users answer the question, *Who will be interested in this item?* For example, you can add links to diverse vacation spots around the world and assign perspectives like *Vacations for Nordic Enthusiasts*, *Archeology Expeditions*, and *Extreme Vacations for Adventurers* to items about vacation types. Users publishing content using an item type that includes a perspective attribute may specify none, one, or many values.

Contrast with **category**.

PL/SQL item

A type of **item** that a user can add to a **page**. A PL/SQL item contains a block of PL/SQL code. When a user clicks the item, the code is executed. The result displays in the user's browser. PL/SQL items can also be displayed directly on the page.

PL/SQL function

See **function**.

PL/SQL page

A type of **page**. PL/SQL pages contain PL/SQL code that generates **HTML** when the page is rendered.

plug-in

See **extension**.

poll

A set of questions used to find out information from users.

Contrast with **survey** and **test**.

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. Portals built using Oracle Portal are made up of **page**s managed within **page group**s, containing **portlet**s and **item**s.

portal administrator

A user with the highest level of privileges in Oracle Portal. Portal administrators can view and modify anything in Oracle Portal, even **page**s and **database provider**s marked private. (The only exception is **group**s: although portal administrators can modify the PORTAL_ADMINISTRATORS and PORTAL_PUBLISHERS groups, they cannot modify any other group unless they have been named **group owner**.)

Portal Builder page

A predefined **page** that contains development and administrative **portlet**s used to build and manage portal **object**s and services.

portal cache

A mechanism for storing cache entries for objects that use **validation-based caching**. It also acts as a backup to the memory-based **Oracle Web Cache** when objects use both validation-based caching and **invalidation-based caching**.

Portal Catalog

See **Oracle PartnerNetwork Solutions Catalog**.

Portal Center

(`http://portalcenter.oracle.com/`) A Web site where you can find out everything you want to know about Oracle Portal. Updated frequently, it always has the latest product information, and is home to the **Portal Developer Services** and **Oracle Portal Verification Service**. This Web site contains all information about the product (including documentation, demonstrations, and so on), and provides access to Oracle Portal expertise.

Portal Community

A network of people dedicated to creating and exchanging information about Oracle Portal. This community includes anyone who uses Oracle Portal; it can leverage the **Portal Knowledge Exchange** for sharing Portal-related information and take advantage of the **Portal Developer Services**.

Portal DB provider

See **database provider**.

Portal Developer Kit

See **PDK**.

Portal Developer Services

The network of people dedicated to creating and exchanging Oracle Portal expertise. This program provides online testing tools through **Portal Center** (`http://portalcenter.oracle.com/`), as well as other venues, and interaction with the product team and other portal developers, using newsletters, surveys, and the **Portal Knowledge Exchange**.

See also **Portal Community**.

Portal Knowledge Exchange

A self-service Web site where subscribers to the **Portal Developer Services** can share white papers, techniques, and portlets with others in the **Portal Community**. Contributions can also be rated so that the most valuable contributions can be easily located.

portal page

See **page**.

portal repository

See **Oracle Metadata Repository**.

Portal Services

A set of services running in the Oracle Portal instance that are used to assemble portal pages and to access portal and page metadata. The Parallel Page Engine (PPE) is one of the Portal Services that assembles portal pages. Other services, like those previously provided by mod_plsql, are incorporated into the Portal Services as well.

portal session

A period of interaction between a browser and Oracle Portal, from the initial access to log off, closure of the browser window, or expiration of the session after a period of inactivity.

Portal smart link item

See **smart link item**.

Portal smart text item

See **smart text item**.

Portal Studio

See **Oracle Portal Verification Service**.

Portal Template

A portal **object** built declaratively using a wizard that enforces specific layouts, colors, fonts, and backgrounds for **page**s and **item**s. You can use Portal Templates with **navigation page**s, **standard page**s, and pages based on custom page types that are based on the standard page type. You can also use Portal Templates with **text item**s, **PL/SQL item**s, **URL item**s, and all **file item**s with a **MIME type** of text/html or text/plain.

Contrast with **HTML Template**.

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. Oracle 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** covers the standard-based portlet development options that Oracle Portal provides.

Portlet Builder

A collection of portlet-building wizards that are accessible through the Provider tab in the Portal Navigator. You can use these wizards to build **chart**s, **form**s, **report**s, **calendar**s, and lists of values.

portlet instance

A **portlet** placed on a particular **page**.

portlet provider

See **provider**.

portlet publisher

A user who can publish portal **object**s (**page**s or otherwise) as **portlet**s so that they can be included on pages.

portlet record

A programmatic structure that contains detailed information about a **portlet**, such as its implementation style and **Show mode** (PL/SQL).

Portlet Repository

A special **page group** that contains the **portlet**s available from the local **provider**s and any registered remote providers. When you register a provider, the provider and its portlets are added to the Portlet Repository.

PPC

Partial Page Caching. A feature that enables **Oracle Web Cache** to independently cache and manage fragments of HTML documents. A template page is configured with Edge Side Includes (**ESI**) markup tags that tell Oracle Web Cache to fetch and include the HTML fragments. The fragments themselves are HTML files containing discrete text or other objects.

PPE

Parallel Page Engine. A multithreaded **servlet** engine that runs in the Portal and services **page** requests. The PPE reads **page metadata**, calls **provider**s for **portlet** content, accepts provider responses, and assembles the requested page in the specified page layout. The Parallel Page Engine is part of the Portal Services, which run on the Oracle Fusion Middleware **middle tier**.

pretty URL

See **path-based URL**.

Preview mode

An optional **portlet Show mode** that provides users with a preview of the portlet before they add it to a page.

primary key

One or more columns in a database table that, in combination, uniquely identify a row in a table.

privilege

In Oracle Portal, the right to perform an action. Privileges are either global (set through in the User or Group Profile) or specific to particular **object**s (usually set through the object's Access tab). When building applications, access can also be granted to database objects, shared portlets, portlets, and applications.

producer

See **provider**.

profile

The information stored about an Oracle Portal user or group, such as password, user ID, and **privilege**s.

property sheet

A built-in **attribute** that displays a summary of an **item**'s attributes or a **page**'s properties.

provider

The communication link between Oracle Portal and a **portlet**. There are two types of providers: **Web provider**s and **database provider**s. Web providers may reside anywhere on the network and are addressed through **SOAP**. Web providers may be implemented using any Web technology. You can build your own Web providers by using the **PDK-Java** and the **PDK-URL Services**. Database providers reside within an Oracle database and manage portlets while performing data-intensive operations.

Providers act as containers for portlets; each portlet communicates with Oracle Portal through its provider. Providers also manage the portlets they contain.

provider definition

A declarative, **XML**-based configuration file (`provider.xml`) that describes a **Web provider**, its **portlet**s, and the location of the content to be displayed in the portlets. This configuration file also describes the behavior of the provider and its portlets.

provider group

A logical collection of **Web provider**s defined by a provider group service. A **portal administrator** can register provider groups for use with their portal. Once registered, a provider group simplifies the process of registering individual **provider**s in the group. This enables organizations that create Web providers to publish registration details of their providers and facilitate automatic registration with any Oracle Portal instance. The only information that must be given to the portal administrator is the name and location of the provider group.

See also **local provider group** and **remote provider group**.

provider record

The record returned by a **database provider** containing specified information about a **portlet**.

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 machines, while allowing Web access.

public page

Any **page** in a **page group** that is viewable by **public user**s (users who are not logged onto Oracle Portal). The page designer or **page group administrator** must explicitly designate a page as public.

public user

A user who can access, but is not logged onto, Oracle Portal. When users first access Oracle Portal, they do so as public users, whether or not they have the ability to log on. 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**.

purge

See **system purge**.

query

A SQL SELECT statement that specifies which data to retrieve from one or more tables or views in a database.

queue

The abstract storage unit used by a messaging system, such as **CMEF**, to store messages.

quota

The amount of space provided in a page group or in the Oracle Portal schema of the **Oracle Metadata Repository** to store uploaded documents.

recent object

A portal **object**, such as a **page** or **portlet**, that has recently been displayed or edited. Each **authenticated user** has his or her own Recent Objects portlet that provides links to the last *n* objects accessed.

referenced object

An object which is directly or indirectly referenced by an **explicit object**.

See also **manifest**.

reference path

The path that uniquely identifies a **portlet instance** on a **page**. A reference path can be used to target **parameter**s to individual portlets on a given page.

region

A carved-out area on a **standard page** used to define the page layout, define page content (**portlet**s and **items**), and control the **style** and **attribute**s for content displayed in a region. A standard page can have one or multiple regions. Regions can be created above, beneath, or beside other regions.

You can create the following types of regions:

- Undefined regions are regions that have not been assigned a particular type.
- Item regions allow you to add items such as text, images, files, and so forth.
- Portlet regions allow you to include portlets in a region.
- Sub-Page Links regions allow you to display a list of the current page's sub-pages in a region.
- Tab regions allow you to include **tab**s in a region.

region banner

A colored, horizontal bar with a title displayed in a **region** of a portal **page**. A banner breaks up the visual flow of a page and groups related **item**s that appear beneath it.

remote database

A database running on a separate machine that can be accessed over the network through a connect string or database link.

remote provider group

The collection of **provider**s that are defined outside of your local instance of Oracle Portal.

See **provider group**. Contrast with **local provider group**.

Replace on Import mode

An import mode. When this option is selected, if the object exists on the target, then it is replaced. If the object does not exist then it is created. When this option is not selected, if the object exists on the target, it is referenced. If it does not exist on the target, it is created.

report

A **portlet** created with the **Portlet Builder** that displays the results of a SQL **query** in a tabular format.

Reuse mode

See **Replace on Import mode**.

rich text editor

A WYSIWIG editor that enables **content contributor**s to easily apply formatting to **text item**s.

rich text item

One of the available items in **Oracle Instant Portal**, designed for smaller blocks of text. A rich text item appears on the page as a title, a summary, and the full text of the item. Compare to **expandable rich text item**, in which the text is hidden until the user clicks an icon. Both expandable rich text and rich text items may contain images, hyperlinks, and tables, as well as any valid HTML code.

root page

The top level of the **page** hierarchy in a **page group**; it contains all other sub-pages in the page group. Also known as the page group's **home page**.

routing method

A mechanism for organizing an **approval process**. Oracle Portal provides three approval routing methods:

- All, Parallel enables Oracle Portal to send the approval to recipients in the step all at the same time. All of the recipients must respond to the approval before the item approval can move to the next step.
- All, Serial enables Oracle Portal to send the approval to recipients in the step one at a time in the sequence specified. All of the recipients must respond to the approval before the item approval can move to the next step.
- Any, Parallel enables Oracle Portal to send the approval to recipients in the step all at the same time. However, only one of the recipients must respond to the approval before the item approval can move to the next step.

row

A set of values in a table; for example, the values representing one employee in the SCOTT.EMP table.

saved search

A mechanism for saving search criteria under a single name. This feature enables you to repeat the search quickly, by choosing the saved search name rather than re-entering the criteria manually. You can save the results of a **basic search**, **advanced**

search, or **custom search**. The Saved Searches portlet lists all the saved searches in a page group.

schema

A collection of **database object**s, including logical structures such as **table**s, **view**s, **sequence**s, **stored procedure**s, **synonym**s, **index**es, **cluster**s, and database links. A schema has the name of the user who controls it.

search portlet

A portlet that enables users to search for **page**s and content within the Oracle Portal schema of the **Oracle Metadata Repository**. Users can also search based on text strings, categories, perspectives, and attributes, and using operators such as CONTAINS, GREATER THAN, LESS THAN, and EQUAL TO.

Contrast with **Oracle Ultra Search**.

secure view

A **view** on the data in the **content repository** that is guaranteed to not change between releases. You should use the secure views to query data on the documents and items stored in the content repository to ensure that those queries will continue to work with later releases of **Oracle Portal**.

self registration

A mechanism that allows users to create new accounts for themselves through a link in the Login portlet.

sequence

A database **object** used to automatically generate numbers for table rows.

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.

session

See **portal session**.

shared object

A portal **object** such as a **personal page**, **navigation page**, **style**, **Portal Template**, **perspective**, **category**, or custom type that can be shared across **page group**s.

shared portlet

A **portlet** created with the **Portlet Builder** that is shared between other Portlet Builder portlets. Each shared portlet can be displayed on multiple **page**s with the same personalization.

Shared Screen mode

A **portlet Show mode** that renders the body of the portlet. Every portlet must have at least a Shared Screen mode.

Show mode

The ways by which a **portlet** can be called to display information. These methods include:

- **Shared Screen mode**
- **Edit mode**
- **Edit Defaults mode**
- **Preview mode**
- **Help mode**
- **About mode**
- **Link mode**
- **Full Screen mode**

simple file item

Similar to a **file item** except with fewer **attribute**s, which makes it quicker and easier to create.

simple image item

Similar to an **image item** except with fewer **attribute**s, which makes it quicker and easier to create.

Simple Object Access Protocol

See **SOAP**.

simple page link item

Similar to a **page link item** except with fewer **attribute**s, which makes it quicker and easier to create.

simple PL/SQL item

Similar to a **PL/SQL item** except with fewer **attribute**s, which makes it quicker and easier to create.

simple text item

Similar to a **text item** except with fewer **attribute**s, which makes it quicker and easier to create.

simple URL item

Similar to a **URL item** except with fewer **attribute**s, which makes it quicker and easier to create.

Single Sign-On

See **Oracle Application Server Single Sign-On**.

smart link item

A type of navigation **item** that is self-configuring. For example, if you add a Home smart link to a **navigation page**, when a user clicks the Home link, he or she is automatically taken to his or her **home page**.

smart text item

A type of navigation **item** that is self-configuring. For example, if you add a Current Date smart text item to a **navigation page**, the current date is automatically pulled from the server and does not need to be specified (through complex coding) by you.

snapshot

A **table** that contains the results of a **query** on one or more tables, called master tables, in a remote database.

snapshot log

A **table** associated with the master table of a **snapshot** tracking changes to the master table.

SOAP

Simple Object Access Protocol. A lightweight, **XML**-based protocol for exchanging information in a decentralized, distributed environment. SOAP supports different styles of information exchange, including: Remote Procedure Call style (RPC) and Message-oriented exchange. RPC style information exchange allows for request-response processing, where an endpoint receives a procedure-oriented message and replies with a correlated response message. Message-oriented information exchange supports organizations and applications that need to exchange business or other types of documents where a message is sent but the sender may not expect or wait for an immediate response.

SSL accelerator card

A hardware device that handles traffic much faster than regular SSL software.

See also **LBR**.

SSO

See **Oracle Application Server Single Sign-On**.

standard page

A type of **page** used to contain and manage **item**s and **portlet**s.

state

In the context of **CMEF**, the state of an **event** provides additional information about that event so that developers can correctly determine what a **subscriber** should do when a portal **action** triggers an event with that particular state.

stored procedure

A set of PL/SQL procedures that are stored in a database.

stripe

A secure slice on a **virtual private portal** that is assigned to a particular **subscriber**.

structured UI template

A **shared portlet** that controls the look and feel of **Portlet Builder portlet**s and runs in standalone mode. Structured UI templates display the same image and text in the same location around every portlet that uses the template.

See **user interface (UI) template**. Contrast with **unstructured UI template**.

struts

A development framework for Java servlet applications based upon the **MVC** design paradigm.

style

A set of values and parameters that controls the colors and fonts of **page**s and **region**s. Style settings include font style, size, color, alignment, and background color. Styles can be created for a specific **page group** or as a **shared object** that is used by pages within multiple page groups.

sub-category

A **category** that appears hierarchically beneath another (parent) category. This provides a way of grouping closely related categories together.

sub-item

An **item** that appears hierarchically beneath another (parent) item. This provides a way of grouping closely related items together, for example, the spreadsheet that is used by a particular HTML file item could be added as a sub-item of the HTML file item.

sub-page

A **page** that appears hierarchically beneath another (parent) page. Every page in a **page group** (except the root page) is a sub-page.

sub-perspective

A **perspective** that appears hierarchically beneath another (parent) perspective. This provides a way of grouping closely related perspectives together.

subscriber

In the context of **CMEF**, a subscriber consumes messages from the WWSBR_EVENT_Q **queue** and performs actions based on those messages.

In the context of hosted portals, a subscriber is a company that signs up with an **ASP** and receives a **stripe** on a hosted Oracle Portal.

subscription notification

A method by which end users can subscribe to a particular **page** or **item** so that they are notified (through the My Notifications portlet) when that page or item is updated. The **page designer** must include a Subscribe **Portal smart link item** for users to be able to subscribe to a page, and must display the Subscribe **attribute** for users to be able to subscribe to items. Additionally, the **page group administrator** must enable approvals and notifications for the **page group**.

substitution tag

A special portal tag used within **HTML Template**s and **unstructured UI template**s to dynamically embed titles, headings, and other elements into the template.

survey

A set of questions used to find out information from users. Surveys can redirect users to different sections of the survey depending on their answers to particular questions.

Contrast with **poll** and **test**.

synonym

An additional name assigned to a **table** or **view** that can thereafter be used to refer to it.

system level caching

A **caching** method where a single copy of an item is stored in the cache (on the middle tier) for all users. Consequently, such items cannot be user specific. All users see the same content for the item.

Contrast with **user level caching**.

system purge

A process that deletes all items in a **page group** from the Oracle Portal schema of the **Oracle Metadata Repository** that are marked as deleted or expired. System purges are performed by the **page group administrator** or **portal administrator**.

tab

An area on a **page** used to increase the amount of content that the page can display by effectively doubling (or tripling, quadrupling, and so on) the amount of real estate available. Tabs also allow you to group content that is common to a subject area, organization, specific role, and so forth.

table

The basic storage structure in a relational database.

tablespace

The allocation of space in the database.

template

See **HTML Template**, **Portal Template** or **user interface (UI) template**.

temporary tablespace

The allocation of space in the database used for the creation of temporary table segments for operations such as sorting table rows.

test

A set of questions used to assess a user's understanding of a particular subject or subjects. You can provide the correct answer for questions and assign each question a score. You can also hand score essay-type answers.

Contrast with **poll** and **survey**.

text item

A type of **item** that a user can add to a **page**. When you create a text item, you enter text (up to 32KB) in the Item Wizard. The text block is then stored in the Oracle Portal schema of the **Oracle Metadata Repository**.

theme

A snapshot generated by **Oracle Text** that describes a document. Rather than searching for documents that contain specific words or phrases, users can use Oracle Text to search for documents that are about a certain subject, even if that subject is not mentioned explicitly in the document.

title

See **display name**.

top-level page

An **Oracle Instant Portal** page represented by a tab in the portal's tab set. May contain one or more **child page**s. Page privileges applied to a top-level page are also applied to all its child pages.

translation

A **page group** rendered in another language. When a **page group administrator** creates a translation, **content contributor**s can add content in that language. Page group users can also view the translated content by setting their language to one of the supported languages.

transport set

A collection of portal **object**s for **export** or **import**. It can contain more than one object of a particular type, such as multiple **page group**s and multiple **page**s.

trigger

A database **object** associated with a table. It executes before or after one or more specified events.

Ultra Search

See **Oracle Ultra Search**.

Uniform Resource Locator

See **URL**.

unstructured UI template

A **shared portlet** that is used to insert content and **HTML** code to control the look and feel of **Portlet Builder portlet**s. Unstructured UI templates are based on HTML code that, when executed, dynamically embeds titles, headings, and other elements that make up a page.

See also **substitution tag** and **user interface (UI) template**. Contrast with **structured UI template**.

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 WebLogic Server.

URL item

A type of **item** that a user can add to a **page**. A URL item, when clicked, provides a route to another Web page. When a user clicks the URL item's **display name**, the Web page referenced by the **URL** displays.

In **Oracle Instant Portal**, a URL item is represented on the page by a globe and, optionally, a title and summary. When the user clicks the globe, the location specified in the item opens in a secondary browser window.

URL page

A type of **page** that provides a route to another Web page, identified by its **URL**. When a user clicks the page link, the Web page referenced by the link is displayed.

URL portlet

A **portlet** created with the **Portlet Builder** that displays the contents of a Web page specified by a **URL**.

user interface (UI) template

A **shared portlet** that controls the look and feel of **Portlet Builder portlet**s in full page display mode. Selecting a UI template when you are building a portlet automatically selects a title on the page where the portlet is displayed, a title background, links to other Web pages, and background colors and images.

See also **structured UI template** and **unstructured UI template**. Contrast with **HTML Template** and **Portal Template**.

user level caching

A **caching** method where a copy of an item is stored in the cache (on the middle tier) for each user. Consequently, users may see different content for the same item.

Contrast with **system level caching**.

validation-based caching

A **caching** method that uses a validation check to determine if the cached item is still valid. This is the caching method used by the **portal cache**. Before an item in the portal cache is used, the **PPE** contacts the portal repository or the **provider** that the content came from to determine if the cached item is still valid.

Contrast with **expiry-based caching** and **invalidation-based caching**.

versioning

A mechanism that allows multiple versions of an **item** to simultaneously exist in the Oracle Portal schema of the **Oracle Metadata Repository**. This feature is useful for tracking document changes from one version to the next or for reverting to a previous version if necessary.

view

A virtual **table** whose rows do not actually exist in the database, but which is based on a table that is physically stored in the database.

View mode

The runtime view of a **page**.

Contrast with **Edit mode**.

View privileges

A privilege level on an **Oracle Instant Portal** page that allows the user to view content on the page. All users may navigate pages, select items for the **Favorite Content area** of the home page, and search for content.

virtual private database

See **VPD**.

virtual private portal

Features for hosting multiple companies or multiple organizations securely within the same portal instance.

VPD

Virtual Private Database. A feature for **ASP**s that want to leverage the Oracle database to host their customers. Essentially, it uses one physical database instance for all customers, but to each customer it looks as if they have their own database. Users cannot see any information that is not meant for them and complete customer isolation is achieved. It requires little to no changes in the core application to take effect as most of the work is done at the database level. Implementing VPD basically requires two key steps: adding a context column (for example, company name) to all the database tables, and implementing a policy to restrict queries on each table based on the context of the logged in user. VPD provides highly secure, full subscriber isolation using this method.

WAP

Wireless Application Protocol. A set of open, global protocols for developing applications and services that use wireless networks.

Web Cache

See **Oracle Web Cache**.

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.

WebDAV

Web-based Distributed Authoring and Versioning. A protocol extension to **HTTP** 1.1 that supports distributed authoring and versioning. With WebDAV, the Internet becomes a transparent read and write medium, where content can be checked out, edited, and checked in to a **URL** address.

Web Services for Remote Portlets

See **WSRP**.

Web provider

An entity that is called, using an **HTTP** request, by Oracle Portal and returns **portlet** content in **HTML**, **XML**, or **WSRP**. A Web provider acts as a proxy for one or more portlets that are defined within a particular application environment (for example, Java, ASP, or Perl) and executed as applications external to Oracle Portal. Web providers are particularly appropriate for Web-accessible information sources.

See also **provider**. Contrast with **database provider**.

Web server

A program that delivers Web pages.

Wireless Application Protocol

See **WAP**.

Wireless Markup Language

See **WML**.

wireless portal

A **portal** accessible from wireless devices, such as cellular telephones.

See also **Oracle Application Server Wireless**.

wizard

A graphical interface that guides a user step-by-step through a process. In Oracle Portal, wizards are used for creating **database provider**s, **portlet**s, database **object**s, **page**s, **page group**s, and **item**s.

WML

Wireless Markup Language. An **XML**-based markup language used to define hypertext-like content and applications for handheld devices.

WSRP

Web Services for Remote Portlets (WSRP). A Web services standard that allows 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. So, a portlet (regardless of language) deployed to a WSRP-enabled container can be rendered on any portal that supports this standard.

XML

Extensible Markup Language. An open standard for describing data using a subset of the SGML syntax.

XML portlet

An Oracle Portal **portlet** that displays the executed results of **XML** code. To create the portlet, you either specify XML code or a **URL** that points to the XML code.

XSL

Extensible Stylesheet Language. The language used within stylesheets to transform or render XML documents.

zip file item

A type of **item** that a user can add to a **page**. Zip file items enable you to upload many files in a single operation. You can use them to migrate the contents of a file system or Web site into the Oracle Portal schema of the **Oracle Metadata Repository**. When you upload a zip file to Oracle Portal, then unzip the uploaded file, a page is created for each directory and an item is created for each file. The items are published in the target page.

Index