# Oracle® Fusion Middleware

Tutorial for Oracle WebCenter Portal Developers

11*g* Release 1 (11.1.1.7.0)

**E10273-09**

April 2013

ORACLE®

Oracle Fusion Middleware Tutorial for Oracle WebCenter Portal Developers, 11*g* Release 1 (11.1.1.7.0)

E10273-09

Primary Author: Tom Maremaa

Contributor: Michele Cyran, Bill Witman, Robin Fisher, Fadi Hakim, Bob Fraser, Richard Nessel, Ingrid Snedecor, Savita Thakur, Joan Carter, Promila Chitkara

# Contents

# Preface

This Tutorial introduces you to Oracle WebCenter Portal: Framework, a key component of Oracle WebCenter Portal that enables you to build your own WebCenter Portal: Framework Applications. As you work through this Tutorial, you'll become familiar with Oracle JDeveloper and the components that have been added to support the new Oracle WebCenter Portal: Framework functionality. When you're ready to begin building your own application, you can move on to the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal* for assistance.

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

## Audience

This document is intended for users wishing to familiarize themselves with Oracle WebCenter Portal: Framework and learn how to develop WebCenter Portal: Framework Applications.

This Tutorial does not assume any prior knowledge of Oracle JDeveloper or Oracle WebCenter Portal. It does, however, assume that you are already somewhat familiar with the following:

- Oracle Application Development Framework (Oracle ADF)
- Oracle ADF Faces
- HTML coding experience (including CSS and JavaScript)
- XML, XSD, XSL syntax rules experience
- Some understanding of JSPs, JavaScript and/or Java
- Basic knowledge of content management tools and processes
- General web concepts and web site structures

The Tutorial is intended for the developer who wants to learn how to build a WebCenter Portal: Framework application. It is aimed specifically at WebCenter site developers, consultants, project managers, and site administrators who need to build and administer portal applications.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

### Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

## Related Documents

For more information on Oracle WebCenter Portal: Framework, see the following documents, which are available on the Oracle WebCenter Suite Documentation page on the Oracle Technology Network (OTN) at `http://www.oracle.com/technology/products/webcenter/documentation.html`:

- *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*, which explains how to use Oracle JDeveloper and Oracle WebCenter Portal Framework to *develop* WebCenter Portal: Framework applications
- *Oracle Fusion Middleware User's Guide for Oracle WebCenter Portal: Spaces*, which explains how to *use* WebCenter Portal: Framework applications at runtime (in a browser)

For more information on Application Development Framework, see the *Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework*.

## 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. |

# 1 Introduction to WebCenter Portal: Framework and the Tutorial

Welcome to Oracle WebCenter Portal: Framework! This chapter introduces you to key WebCenter Portal: Framework concepts, then explains what you will create following the steps in this Tutorial. The lessons are designed to familiarize you with different aspects of WebCenter Portal: Framework functionality, and to demonstrate enough about each feature so that you can create your own WebCenter Portal: Framework applications.

If you need additional information about a feature, you can always refer to the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal* and the *Oracle Fusion Middleware User's Guide for Oracle WebCenter Portal: Spaces*.

## What is WebCenter Portal: Framework?

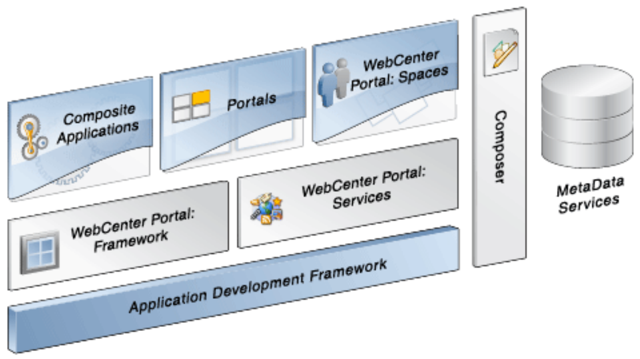
The WebCenter Portal: Framework provides portal-specific features to a WebCenter Portal application. Portals allow users to view and interact with information and to customize their experience to match their exact requirements. Portals typically include features like pages, navigation, security, and customization. Portals can also include features like portlets, content management system integration, personalization, social computing services, search, analytics, and more.

WebCenter Portal: Framework augments the Oracle ADF environment by providing additional integration and runtime customization options. In essence, the framework integrates capabilities historically included in portal products, such as navigation, page hierarchies, portlets, customization, personalization, and integration, directly into the fabric of the JSF environment. This eliminates artificial barriers for the user and provides the foundation for developing context-rich applications.

You can selectively add only desired Oracle WebCenter Portal components or services to your WebCenter Portal application. For example, you might only want to add the Instant Messaging and Presence (IMP) service. In this case, you could add just that service without adding all of the other services available with Oracle WebCenter Portal.

Figure 1–1 provides an overview of the Oracle WebCenter Portal architecture, showing the major components that make up the product.

***Figure 1–1 Overview of the Oracle WebCenter Architecture***


Composite Applications
Portals
WebCenter Portal: Spaces
Composer
MetaData Services
WebCenter Portal: Framework
WebCenter Portal: Services
Application Development Framework

In Figure 1–1, notice Oracle WebCenter Portal: Services and Composer. You will use both of these components in conjunction with WebCenter Portal: Framework in this Tutorial.

For more information about WebCenter Portal: Framework, WebCenter Services, and Composer, refer to "Understanding Oracle WebCenter Portal" in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

## What Will I Create?

In this Tutorial, you will use WebCenter Portal: Framework to build a WebCenter Portal: Framework application that is customizable at runtime, empowering you and your end users to edit application pages according to personal requirements and directly leveraging Oracle Metadata Services (MDS).

Building your portal application, you will also use Composer, which is an easy, browser-based environment whose components you can simply add at design time to a page in JDeveloper. In conjunction with Metadata Services, Composer provides a runtime editing tool that enables business users to edit application pages. Changes made to a page at runtime are then saved as metadata, separate from the base application definitions. This eliminates the need to revise your application and redeploy it to the production environment.

The goal is for you to complete the lessons in the Tutorial and build your WebCenter Portal: Framework application within a period of about two hours. In so doing, you will gain valuable hands-on experience working with Oracle JDeveloper and learning some of the important, high-level concepts you need to master in order to extend your knowledge of JDeveloper and the Framework.

The Tutorial is not intended to provide you with a complete guide to all the features and capabilities available in the WebCenter Portal: Framework. However, as you build your portal as a developer at design time, you ought to become familiar with some of the key concepts and paradigms in the Framework, such as

- The power of page templates
- Working with and applying skins to change the look and feel of your portal at runtime
- The power of the unified navigation model
- Iterative development so you can work more quickly and efficiently when building a portal application by disabling certain optimization features

- Customizing pages and site templates in your portal and setting permissions for user access
- Runtime in-context editing of HTML document content

You will create a content repository connection that is owned and deployed by your WebCenter Portal application. In this case, the connection will be to the Oracle WebCenter Content repository with access provided to the Oracle Content Server. You will set UCM as your primary connection and navigate to the UCM Content folder in the Contributions directory, where the HTML content for your application is stored. You will then retrieve this content and use it to customize portal pages by dragging and dropping the documents from the folder on to various components and rendering that content as Content Presenter task flows. This is a preferred and recommended method of working with the tools available in the Portal Framework.

By using Oracle WebCenter Portal: Services to integrate content from a content repository, you will be able to display that content in a user-friendly interface and enable users to "tag" and search the content.

## The Development Scenario

***Go Green Eat Fresh*** is a public facing website that offers restaurant customers a choice of healthy and fast foods to eat, including pastas, meat and salads. Customers can browse and select from a menu of choices on the Home page of the portal. Administrators and registered users with access can edit the content of HTML documents, such as menus, food listings and orders, at runtime. That content is stored in folders in Oracle WebCenter Content repository for easy access and can be retrieved when content pages need to be changed or modified.

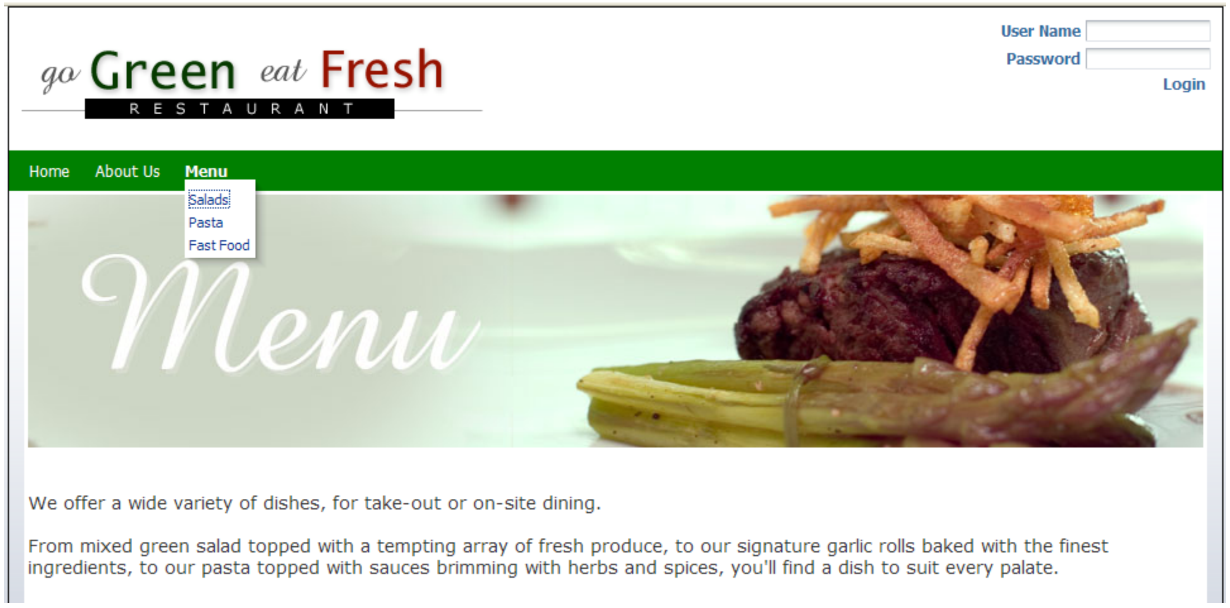
In this Tutorial you play the dual role of a portal developer and a website administrator who is tasked with building the ***Go Green Eat Fresh*** portal and managing its content repository, changing and updating its content based on customer needs and demands.

These will be your assigned tasks:

1. Create an application based on a WebCenter Portal: Framework application template.
2. Create users with Administration access to the portal.
3. Create site navigation for your portal.
4. Create a navigation link of type Content Item.
5. Create a page which renders HTML documents listed under a folder in UCM as tabs.
6. Create a page which acts as a template to create content type links.
7. Create a content type link (as a node in the navigation) to a folder in UCM, which will list all the documents under it as child links.
8. Create a content link (as a node in the navigation) which will render the result of Content Query (CMIS query) as links under the node.
9. Create a customizable page which is invisible in the page hierarchy.
10. Create an editable content page (rendering an HTML page)

Figure 1–2 shows a partial view of the WebCenter Portal: Framework Application you will create following the lessons in this Tutorial.

*Figure 1–2 A Partial View of the Go Green Eat Fresh Public Facing Restaurant Website*



We offer a wide variety of dishes, for take-out or on-site dining.

From mixed green salad topped with a tempting array of fresh produce, to our signature garlic rolls baked with the finest ingredients, to our pasta topped with sauces brimming with herbs and spices, you'll find a dish to suit every palate.

## Tutorial Path

This Tutorial is designed for the chapters to be completed in the same sequence as they are presented. Due to dependencies, completing them in a different order may result in missing resources or even errors.

The path through this Tutorial is as follows:

- Chapter 2, "Preparing for the Tutorial" tells you what you must do before you can complete the steps in this Tutorial, like verifying you have correctly installed JDeveloper and the required WebCenter Extensions. It also specifies that you will need to connect to a content repository, in this case, the Oracle WebCenter Content repository, to complete the lessons in the Tutorial.
- Chapter 3, "Creating a WebCenter Portal Application" introduces you to the steps you need to follow to create a WebCenter Portal: Framework Application, using Oracle JDeveloper.
- Chapter 4, "Creating a New Page Template with a New Portal Skin" discusses how to create a new JSF page template and set that template as a portal resource.
- Chapter 5, "Changing the Look and Feel of Your Portal Application" describes how to change the default settings for both your page template and skin at design time in JDeveloper, thus changing the application look and feel.
- Chapter 6, "Connecting to and Managing Content Repositories" discusses how to create a content repository connection owned and deployed by your WebCenter Portal application.
- Chapter 7, "Customizing Pages for Permissions and Runtime Editing" describes how to customize specific pages in the page hierarchy and set permissions for user access, as well as how to edit in-context HTML document content.
- Chapter 8, "Conclusion" recaps the lessons you learned in each chapter, discussing the sequence of steps that you followed to create, enhance and customize a WebCenter Portal: Framework Application.

# 2 Preparing for the Tutorial

To prepare for this Tutorial, you need to obtain and install the current release of Oracle JDeveloper 11*g* Release 1 (11.1.1.7) software on your system. You also need to verify if you have the correct Oracle WebCenter extension installed. Beyond that, you will need to copy and extract a folder with sample Tutorial files on your hard drive. This chapter explains what you need to install in order to successfully complete the lessons in the Tutorial.

In addition, you will need to create a connection to a content repository. This is a necessary and preferred way of working with a content-based portal, such as the one you will create by following the lessons in this Tutorial.

Chapter 6, "Connecting to and Managing Content Repositories" describes how you create a connection to the Oracle WebCenter Content repository. Note that you can also have portal applications that do not use content stored in a repository.

## Introduction

You will set up the environment for the Tutorial by following these steps:

- Step 1: Obtain the Software
- Step 2: Install the WebCenter Extension Bundle
- Step 3: Set the User Home Directory Environment Variable
- Step 4: Verify the Correctly Installed JDeveloper Release and WebCenter Extension
- Step 5: Work with the Integrated WebLogic Server (WLS)
- Step 6: Download the Sample Tutorial Files
- Step 7: Create a Content Repository Connection

## Step 1: Obtain the Software

Oracle JDeveloper provides an integrated development environment for developing WebCenter Portal applications. For information on obtaining and installing Oracle JDeveloper, see the Oracle JDeveloper page on OTN at:

```
http://www.oracle.com/technetwork/developer-tools/jdev/overview/
index.html
```

## Step 2: Install the WebCenter Extension Bundle

Before you can develop WebCenter Portal applications, you must install the WebCenter Portal extension bundle in Oracle JDeveloper. This extension bundle is a

JDeveloper add-in that provides the complete set of WebCenter Portal capabilities and features to the JDeveloper Studio Edition.

To check whether WebCenter Portal extension is installed for JDeveloper, click the **File** menu and select **New Application**. In the new window, select **General - Applications**. If you see **WebCenter Portal - Framework Applications** and **WebCenter Portal - Portlet Producter Applications**, then WenCenter Portal extension has been installed for JDeveloper. If they do not appear, then you must install the WebCenter Portal extension.

To install the WebCenter Portal extension bundle:

1. Start Oracle JDeveloper.
2. If the Select Default Roles dialog displays, select **Default Role** to enable all technologies, and click **OK**.
3. If a dialog opens asking if you want to migrate settings from an earlier version, click **No**.
4. From the **Help** menu, select **Check for Updates**.
5. Click **Next** in the Welcome page of the Check for Updates wizard.
6. On the Source page, under Search Update Centers, search for **webcenter**. Select any WebCenter extensions that appear, then click **Finish**.
7. When prompted, restart JDeveloper

JDeveloper is now configured to create the WebCenter Portal application for this Tutorial.

For more information on obtaining and installing Oracle WebCenter Framework, see the Oracle WebCenter page on OTN at:

`http://webcenter.oracle.com`

## Step 3: Set the User Home Directory Environment Variable

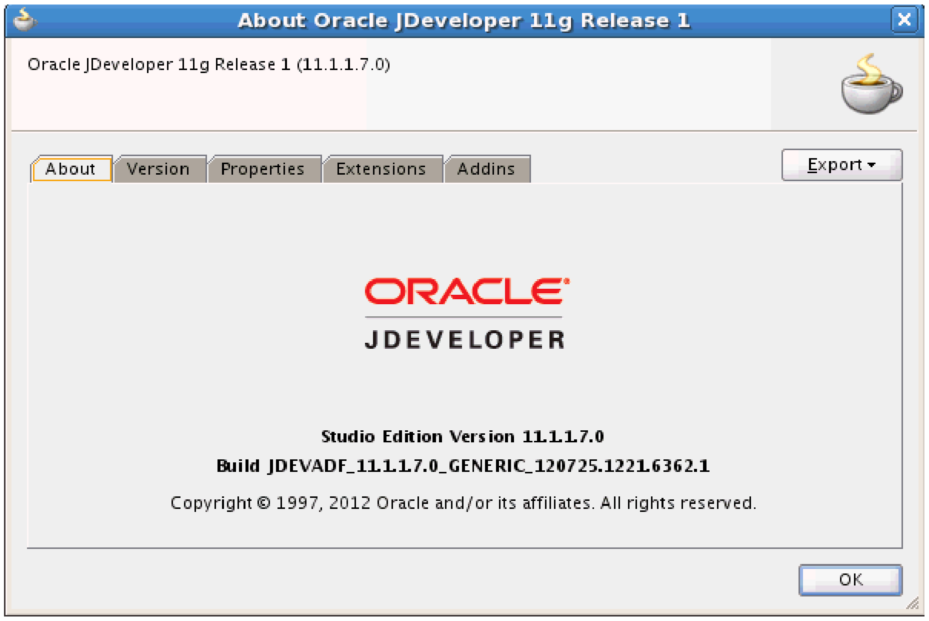
Oracle strongly recommends that you set an environment variable for the user home directory that is referenced by JDeveloper. By setting this variable, you can avoid receiving long pathname errors that are known to occur in some circumstances.

For detailed instructions on setting the user home directory on Windows, Linux, UNIX, and Mac OS X operating systems, see the section "Setting the User Home Directory" in *Oracle Fusion Middleware Installation Guide for Oracle JDeveloper*.

## Step 4: Verify the Correctly Installed JDeveloper Release and WebCenter Extension

Once you have obtained the software, ensure that you have installed Oracle JDeveloper 11*g* Release 1 (11.1.1.7), shown in Figure 2–1, and the Oracle WebCenter Portal extension (11.1.1).

***Figure 2–1 Oracle JDeveloper 11g Release 1 About Box***



Oracle JDeveloper 11g Release 1 (11.1.1.7.0)

Studio Edition Version 11.1.1.7.0

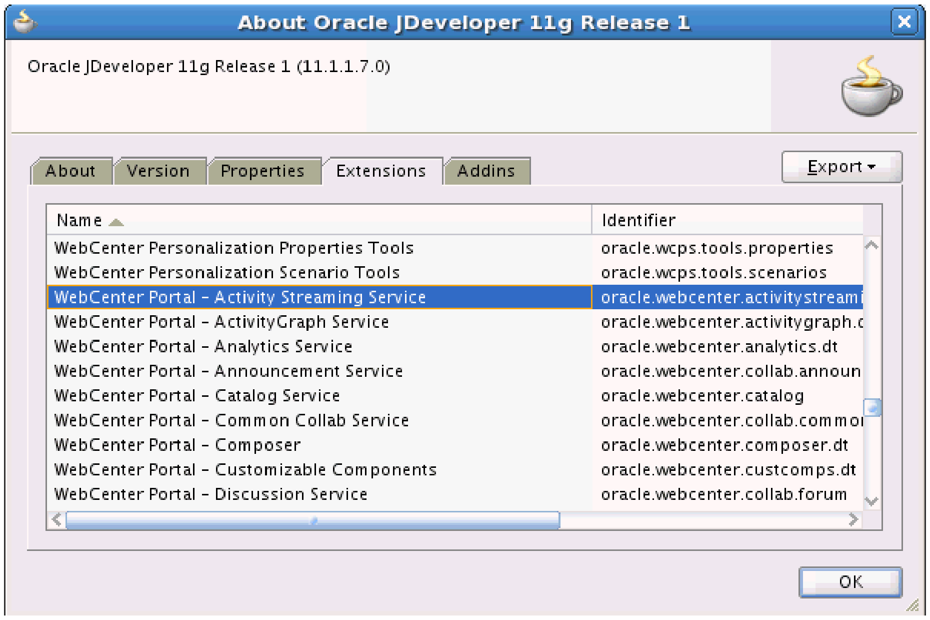
Build JDEVADF_11.1.1.7.0_GENERIC_120725.1221.6362.1

Copyright © 1997, 2012 Oracle and/or its affiliates. All rights reserved.

If you are not sure whether you have the WebCenter Portal extension, you can verify this by opening Oracle JDeveloper, then **About** from the Help menu, then click the **Extensions** tab. At the top of the About dialog, you should see **Oracle JDeveloper 11g Release 1 11.1.1.7.0**. On the Extensions list, sort by **Identifier** to locate the `oracle.webcenter.*` components.

Figure 2–2 shows the Oracle WebCenter Portal components listed in JDeveloper.

***Figure 2–2 Oracle WebCenter Portal Framework Extensions in Oracle JDeveloper***



Oracle JDeveloper 11g Release 1 (11.1.1.7.0)

| Name | Identifier |
|---|---|
| WebCenter Personalization Properties Tools | oracle.wcps.tools.properties |
| WebCenter Personalization Scenario Tools | oracle.wcps.tools.scenarios |
| WebCenter Portal - Activity Streaming Service | oracle.webcenter.activitystreami |
| WebCenter Portal - ActivityGraph Service | oracle.webcenter.activitygraph.c |
| WebCenter Portal - Analytics Service | oracle.webcenter.analytics.dt |
| WebCenter Portal - Announcement Service | oracle.webcenter.collab.announ |
| WebCenter Portal - Catalog Service | oracle.webcenter.catalog |
| WebCenter Portal - Common Collab Service | oracle.webcenter.collab.commo |
| WebCenter Portal - Composer | oracle.webcenter.composer.dt |
| WebCenter Portal - Customizable Components | oracle.webcenter.custcomps.dt |
| WebCenter Portal - Discussion Service | oracle.webcenter.collab.forum |

If you do not see these components (shown in Figure 2–2), you must install the WebCenter Portal extension, as described in the following steps.

To install the WebCenter Portal extension to Oracle JDeveloper using the Update Center, see "Step 2: Install the WebCenter Extension Bundle".

For more information on obtaining and installing Oracle WebCenter Portal Framework, see the Oracle WebCenter Portal page on OTN (`http://webcenter.oracle.com`).

## Step 5: Work with the Integrated WebLogic Server (WLS)

Installation of Oracle WebCenter Portal Framework reconfigures the Integrated WebLogic Server (WLS) domain in JDeveloper to include additional libraries and several prebuilt portlets. For this Tutorial, you may not need to work with the additional libraries or prebuilt built portlets. However, you do need to know how to start and stop Integrated WLS.

There are several options for starting Integrated WLS available in the **Run** menu in Oracle JDeveloper.

- To start Integrated WLS in debug mode, select **Debug Server Instance** from the **Run** menu.

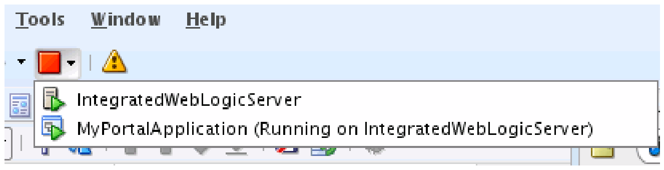
  Running the service in debug mode helps in debugging the service.

- To start Integrated WLS in the regular mode, select **Start Server Instance** from the **Run** menu.

There are several ways to determine if the integrated WLS is running and to stop it.

- The Terminate menu shows you a list of running server(s) and the deployed application(s), if any. (Figure 2–3). To stop a server (or to undeploy an application), select it from this menu.

***Figure 2–3 The Terminate Menu Shows What Is Running***



- Select **Terminate** from the **Run** menu, and select the server to stop it.
- Access the Integrated WLS console from your browser:

  ```
  http://localhost:7101/console
  ```

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

In working with Integrated WLS, it is important to understand the following concepts.

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

You can use Integrated WLS as a platform for pretesting WebCenter Portal application deployments on your local computer by establishing an application server connection to it from Oracle JDeveloper. When you run the application in Integrated WLS, it is actually deployed as if you were deploying it to a WebLogic Server instance in an application server. For more information about Integrated WLS, see Section 64.3, "Deploying a WebCenter Portal Application to a WebLogic Managed Server," in Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal.

---

**Note:** You can specify the Java Virtual Machine (JVM) settings for a WebCenter Portal application running on the Integrated WLS in the `setDomainEnv.sh` script located here:

```
JDEV_SYSTEM_DIRECTORY/DefaultDomain/bin/
```

The default memory values are:

```
-Xmx512m -XX:PermSize=128m -XX:MaxPermSize=512m
```

When creating or referring to the `JDEV_SYSTEM_DIRECTORY`, keep in mind that, on a Windows platform, a WebCenter domain name cannot contain spaces, and the domain cannot be created in a folder that has a space in its path. Also, pages in WebCenter Portal application are not rendered if there is a space in the path to the system directory in Oracle JDeveloper. Therefore, ensure that `DOMAIN_HOME` and `JDEV_SYSTEM_DIRECTORY` paths do not contain spaces.

---

## Step 6: Download the Sample Tutorial Files

As you work through the lessons in this Tutorial, you'll need to include certain content -- images, skins and templates -- in your portal application. This material is contained in a ZIP file, which you can download by following these instructions.

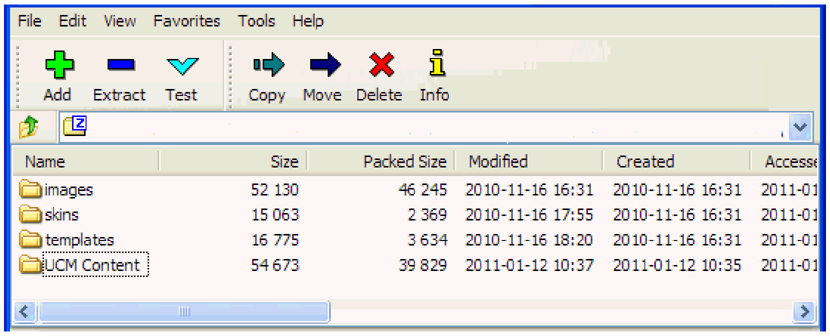
To download the sample Tutorial files:

1. Open a browser, and enter the following in the Address field:

   ```
   http://www.oracle.com/technetwork/middleware/webcenter/owcs-r11ps3-devtutsetup-254761.zip
   ```

2. Open the ZIP file (`owcs-r11ps3-devtutsetup-254761.zip`), as shown in Figure 2–4.

***Figure 2–4 The Downloaded Tutorial SetUp Zip File***



| Name | Size | Packed Size | Modified | Created | Accesse |
|---|---|---|---|---|---|
| images | 52 130 | 46 245 | 2010-11-16 16:31 | 2010-11-16 16:31 | 2011-01 |
| skins | 15 063 | 2 369 | 2010-11-16 17:55 | 2010-11-16 16:31 | 2011-01 |
| templates | 16 775 | 3 634 | 2010-11-16 18:20 | 2010-11-16 16:31 | 2011-01 |
| UCM Content | 54 673 | 39 829 | 2011-01-12 10:37 | 2011-01-12 10:35 | 2011-01 |

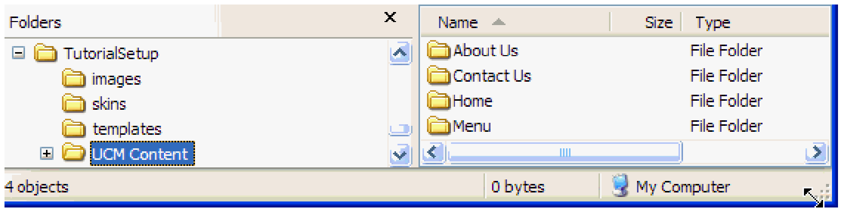
3. Unzip the file to a local drive, such as `C:\TutorialSetUp`.

In Chapter 4, "Creating a New Page Template with a New Portal Skin," you will extract the contents of these files and copy them step-by-step to their appropriate folders for use in building your portal application.

4. Once you've downloaded the zip file and unzipped it, extracting its contents, you need to upload the Oracle WebCenter Content folder to the Contributions folder in the Oracle WebCenter Content repository. The best way to upload the whole file folder to Oracle WebCenter Content is with Desktop Integration Suite. For detailed information, see the chapter "Integration into Windows Explorer" in *Oracle Fusion Middleware User's Guide for Desktop Integration Suite*.

   The UCM Content folder contains HTML content (Figure 2–5) you will need in creating your WebCenter Portal application, as you work through the lessons in this Tutorial.

***Figure 2–5 The UCM Content Folder with Sub Folders of HTML Content for Upload***



| Name | Size | Type |
|---|---|---|
| About Us | | File Folder |
| Contact Us | | File Folder |
| Home | | File Folder |
| Menu | | File Folder |

## Step 7: Create a Content Repository Connection

To complete the lessons in this Tutorial, you will need access to a content repository, specifically one that is owned and deployed by your WebCenter Portal application. In this case, for purposes of this Tutorial, you will need to create a connection to the Oracle WebCenter Content repository, which provides access to the Content Server.

Connecting to the Oracle WebCenter Content repository is a preferred use case and best practice for creating WebCenter Portal applications, if you need to work with content-based portal, as is the case with the lessons described in this Tutorial.

Chapter 6, "Connecting to and Managing Content Repositories" discusses in detail the steps you need to follow to create a connection to Oracle WebCenter Content. It is not necessary to create this connection before starting to create and build your portal application.

For more information about creating a content repository connection, see the chapter "Managing Content Repository Connections" in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal* and the section "Configuring Oracle WebCenter Content Server Repositories" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

3

# Creating a WebCenter Portal Application

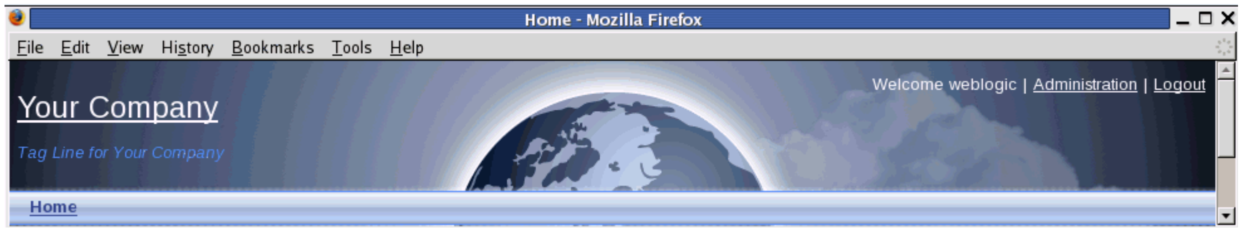
In this lesson, you will create a basic WebCenter Portal: Framework application. WebCenter provides an application template that provisions the new application with WebCenter Portal files and libraries – everything you need to develop and deploy a portal.

You will play the role of a portal developer assigned the task of developing the basic structure of the portal. By selecting the option to configure your application with standard portal features, you will ensure that all the necessary portal artifacts, like templates, catalogs, skins, default page, and the Resource Manger, are generated in the application. This will reduce the time required to develop your application.

As a developer, you will learn in this lesson how to work with the existing application template, and then in "Creating a New Page Template with a New Portal Skin" how to modify and change that default page template.

At the end of this lesson, the page you create will look like Figure 3–1 when you log in as Administrator and open the page in a web browser.

***Figure 3–1 The Home Page with Administrator Privileges Enabled after Successful Login***


Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Welcome weblogic | Administration | Logout
Your Company
Tag Line for Your Company
Home

## Introduction

This lesson contains the following steps:

- Step 1: Create a Custom WebCenter Portal Application
- Step 2: Use Seeded Page Templates to Build Your Portal Application

Before you begin the steps in this lesson, ensure you have followed the steps up to this point in the Tutorial.

## Step 1: Create a Custom WebCenter Portal Application

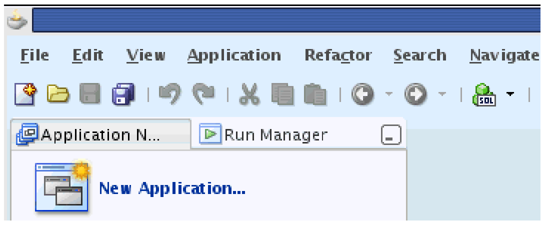
Let's begin by creating a WebCenter Portal: Framework application, using the WebCenter wizard for creating new portal applications. The wizard uses an out-of-the-box **Portal Application template** that ensures the appropriate application components are included.

After you create your portal application, you can then configure the necessary connections to a database and content repository, as described in Chapter 6, "Connecting to and Managing Content Repositories."

**To create a WebCenter Portal application:**

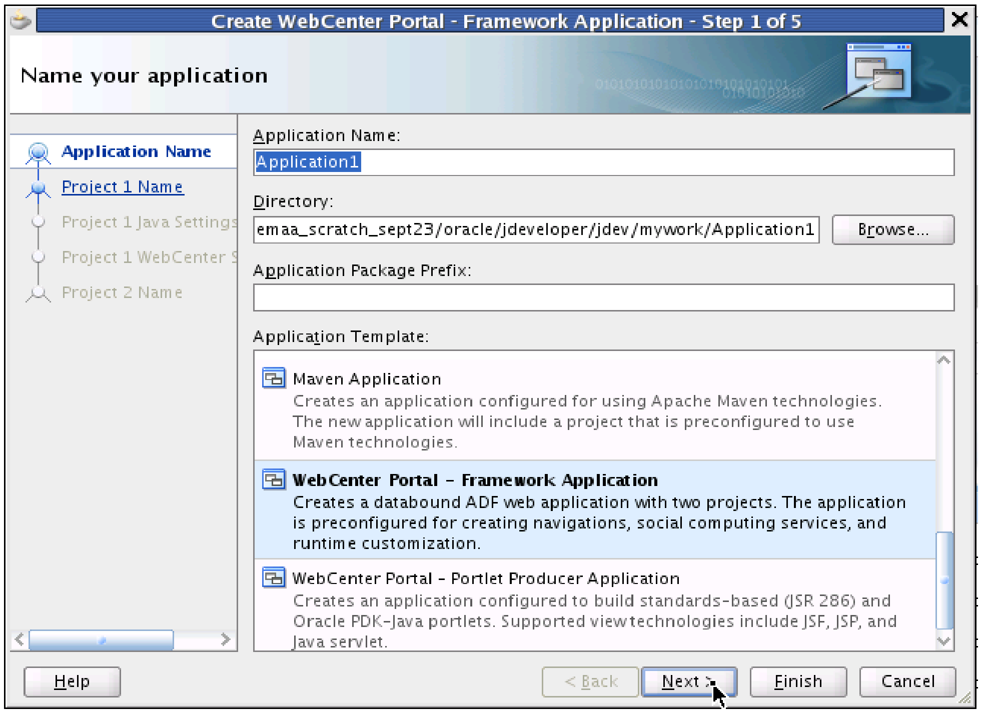
1. In Oracle JDeveloper, in the Application Navigator, choose the **New Application** icon (Figure 3–2) and click it to launch the application wizard.

***Figure 3–2 The New Application Icon in Application Navigator***



2. Now in the Items list, navigate down the list and select **WebCenter Portal Framework Application** (Figure 3–3).

***Figure 3–3 Create New WebCenter Portal Framework Application***



Application Name: Application1

Directory: emaa_scratch_sept23/oracle/jdeveloper/jdev/mywork/Application1

Application Package Prefix:

Application Template:

Maven Application
Creates an application configured for using Apache Maven technologies. The new application will include a project that is preconfigured to use Maven technologies.

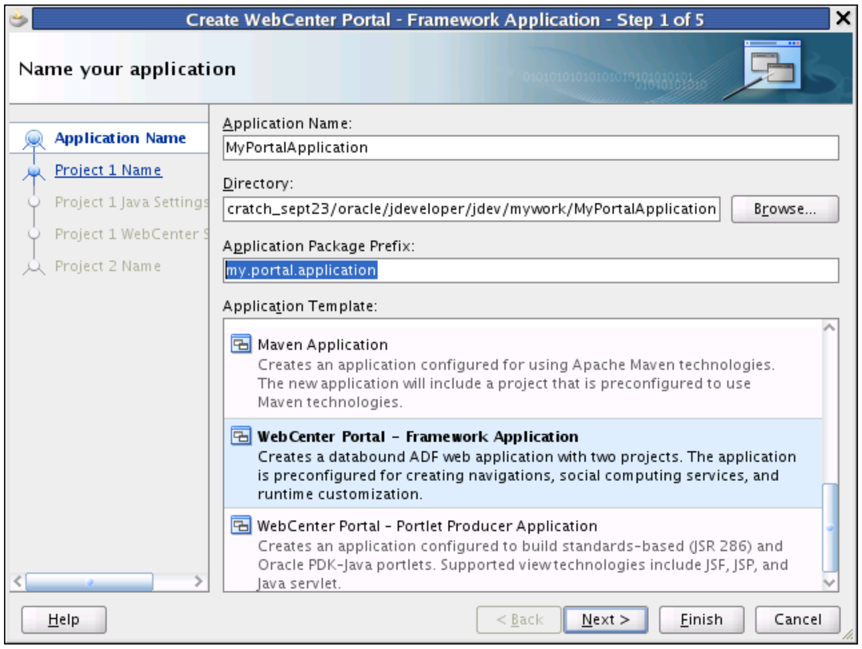
**WebCenter Portal - Framework Application**
Creates a databound ADF web application with two projects. The application is preconfigured for creating navigations, social computing services, and runtime customization.

WebCenter Portal - Portlet Producer Application
Creates an application configured to build standards-based (JSR 286) and Oracle PDK-Java portlets. Supported view technologies include JSF, JSP, and Java servlet.

3. On the Application Name tab, in the Application Name field, enter `MyPortalApplication`, as shown in Figure 3–4. Click the Browse button in the Directory field to specify the directory on your system where you want your portal application to reside.

4. In the Application Package Prefix field (shown in Figure 3–4), enter `my.portal.application`.

***Figure 3–4 Naming Your Application - Step 1 of 5***



Application Name: MyPortalApplication

Directory: cratch_sept23/oracle/jdeveloper/jdev/mywork/MyPortalApplication

Application Package Prefix: my.portal.application

Application Template:

Maven Application
Creates an application configured for using Apache Maven technologies. The new application will include a project that is preconfigured to use Maven technologies.

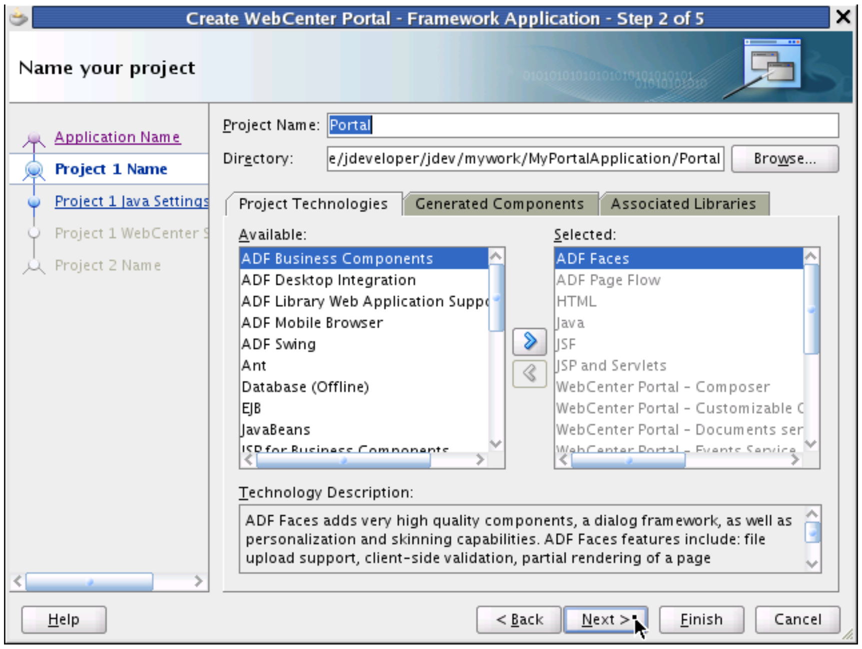
**WebCenter Portal - Framework Application**
Creates a databound ADF web application with two projects. The application is preconfigured for creating navigations, social computing services, and runtime customization.

WebCenter Portal - Portlet Producer Application
Creates an application configured to build standards-based (JSR 286) and Oracle PDK-Java portlets. Supported view technologies include JSF, JSP, and Java servlet.

5. Click **Next.**

   The Name your project dialog appears in the wizard, as shown in Figure 3–5. On the Project Name tab, in the Project Name field, note that the project is named `Portal` by default.

***Figure 3–5 Name Your WebCenter Project - Step 2 of 5***



Project Name: Portal

Directory: e/jdeveloper/jdev/mywork/MyPortalApplication/Portal

Available:
ADF Business Components
ADF Desktop Integration
ADF Library Web Application Suppo
ADF Mobile Browser
ADF Swing
Ant
Database (Offline)
EJB
JavaBeans
JSR for Business Components

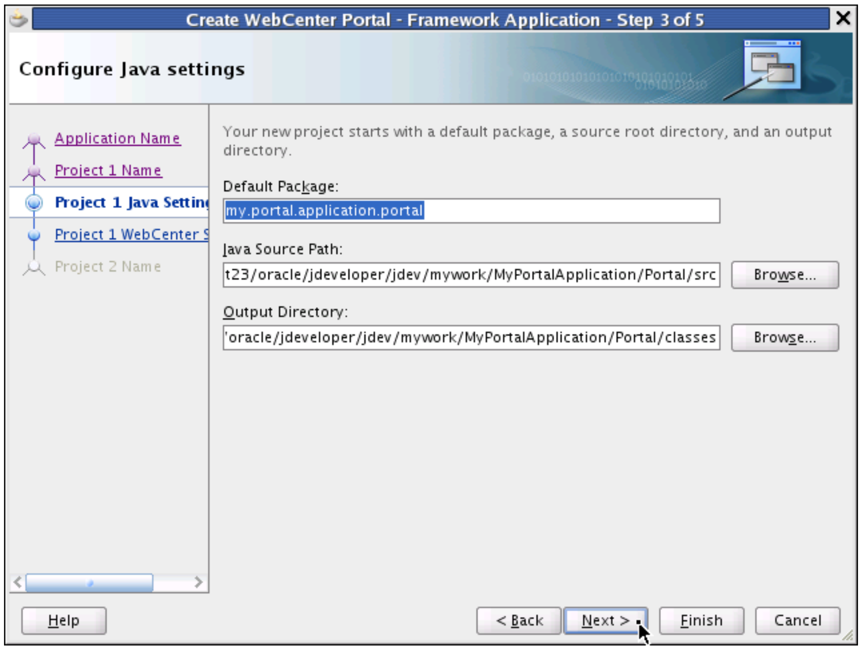
Selected:
ADF Faces
ADF Page Flow
HTML
Java
JSF
JSP and Servlets
WebCenter Portal - Composer
WebCenter Portal - Customizable C
WebCenter Portal - Documents ser
WebCenter Portal - Events Service

Technology Description:
ADF Faces adds very high quality components, a dialog framework, as well as personalization and skinning capabilities. ADF Faces features include: file upload support, client-side validation, partial rendering of a page

6. Click **Next**.

On the Project Java Settings tab, in the Default Package field (Figure 3–6), note that the project package is named `my.portal.application.portal` by default. A source root directory and an output directory are also specified by default.

***Figure 3–6 Configure Java Settings in WebCenter - Step 3 of 5***



Your new project starts with a default package, a source root directory, and an output directory.

Default Package: my.portal.application.portal

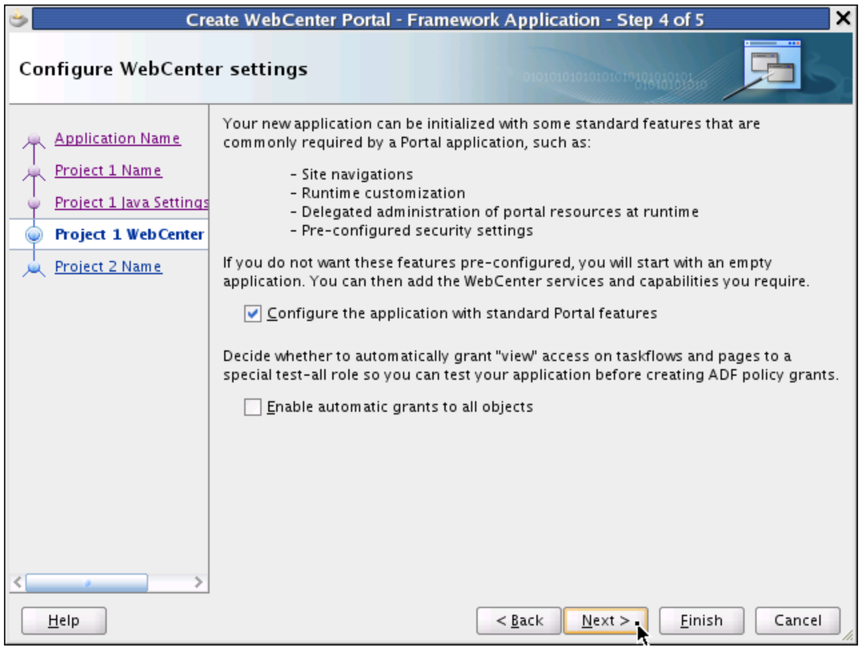
Java Source Path: t23/oracle/jdeveloper/jdev/mywork/MyPortalApplication/Portal/src

Output Directory: 'oracle/jdeveloper/jdev/mywork/MyPortalApplication/Portal/classes

**7.** Click **Next**.

The Configure WebCenter settings dialog appears, as shown in Figure 3–7. Ensure that the checkbox **Configure the application with standard Portal features** is checked.

***Figure 3–7 Configure WebCenter Settings - Step 4 of 5***



Your new application can be initialized with some standard features that are commonly required by a Portal application, such as:

- Site navigations
- Runtime customization
- Delegated administration of portal resources at runtime
- Pre-configured security settings

If you do not want these features pre-configured, you will start with an empty application. You can then add the WebCenter services and capabilities you require.

[✓] Configure the application with standard Portal features

Decide whether to automatically grant "view" access on taskflows and pages to a special test-all role so you can test your application before creating ADF policy grants.

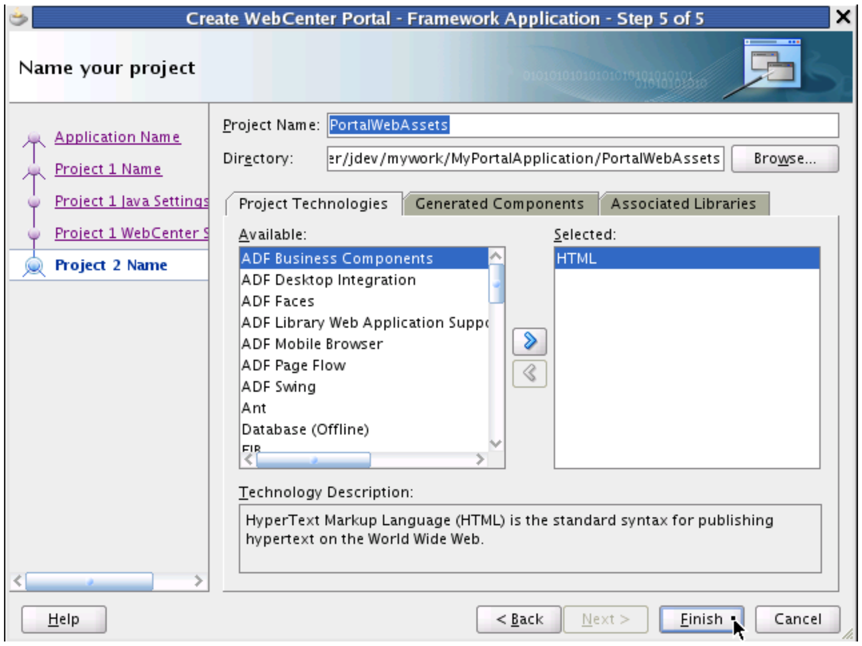
[ ] Enable automatic grants to all objects

********************************************************************************************

8. Click **Next**.

   In the wizard, Step 5 of 5 appears in the Create WebCenter Portal - Framework dialog. A second project name is created, entitled `PortalWebAssets`, as shown in Figure 3–8.

***Figure 3–8 Configure WebCenter Settings - Step 5 of 5***



Project Name: PortalWebAssets

Directory: er/jdev/mywork/MyPortalApplication/PortalWebAssets

Available:
ADF Business Components
ADF Desktop Integration
ADF Faces
ADF Library Web Application Suppo
ADF Mobile Browser
ADF Page Flow
ADF Swing
Ant
Database (Offline)

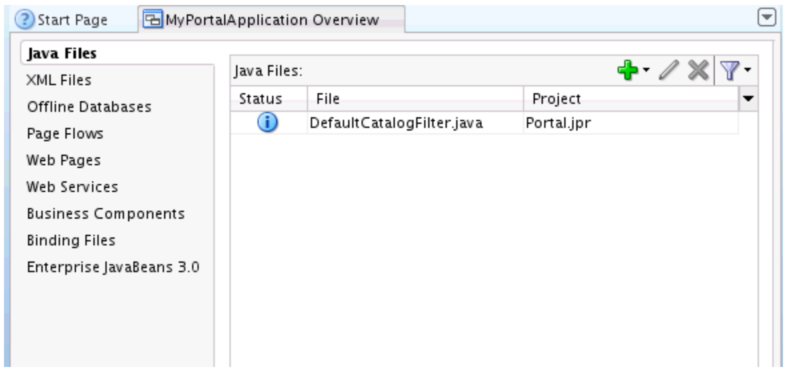
Selected:
HTML

Technology Description:
HyperText Markup Language (HTML) is the standard syntax for publishing hypertext on the World Wide Web.

9. Click **Finish**.

   Oracle JDeveloper now configures and generates the base XML files, offline databases, page flows, web pages, business components, web services, binding files, and enterprise Java Beans available for you to build and deploy your WebCenter Portal application, as shown in the MyPortalApplication Overview window (Figure 3–9).

***Figure 3–9 The MyPortalApplication Overview Window with Java Files Selected***

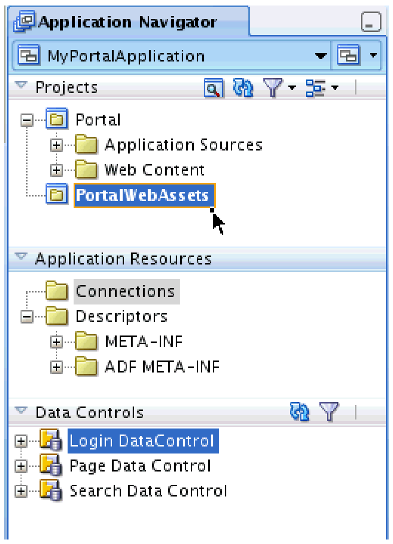


| Status | File | Project |
|---|---|---|
| ⓘ | DefaultCatalogFilter.java | Portal.jpr |

10. Return to the folders of your portal application in Application Navigator. A collapsed view of the folders shows your default project as named **Portal**, with Application Sources and Web Content as sub folders, shown in Figure 3–10.

    Note **PortalWebAssets** is also a project. PortalWebAssets are intended to include static resources, like HTML and image files, in a newly created portal web assets project.

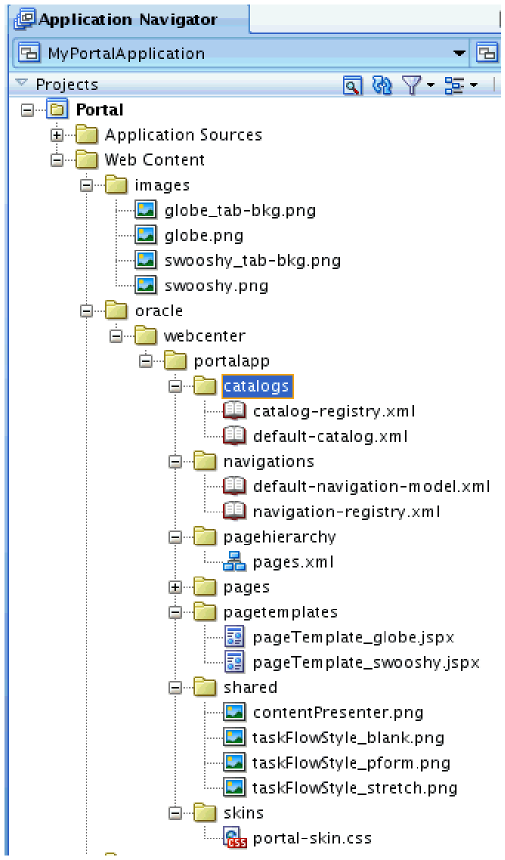
***Figure 3–10 The MyPortalApplication Project in a Collapsed View in Application Navigator***



Following the steps in the Wizard, your WebCenter Portal application is now populated with a portal project, named Portal, and a static application resources project called by default PortalWebAssets. Your portal project includes features like site navigation, page hierarchies, delegated administration, security, page templates, and runtime customizing. Your portal application can consume portlets, incorporate content management services, and include WebCenter social computing services. PortalWebAssets include static application resources like HTML and image files. By separating the static resources into a separate project, you can deploy those resources to a dedicated server.

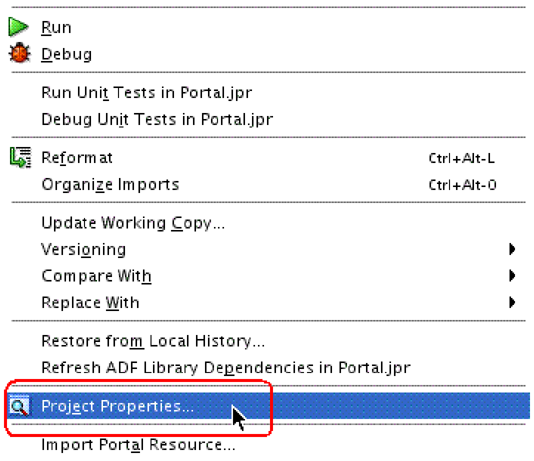
11. Now expand the various folders and sub folders, like catalogs, navigators, page hierarchy, pages and page templates (shown in Figure 3–11), for a view of the logical structure and parent-child relationships created in your portal application.

***Figure 3–11 Expanded Folders in MyPortalApplication Project Shown in Application Navigator***



12. Now you need to change the context root of the portal application. Right-click **Portal** project and choose the **Project Properties** menu item, as shown in Figure 3–12.

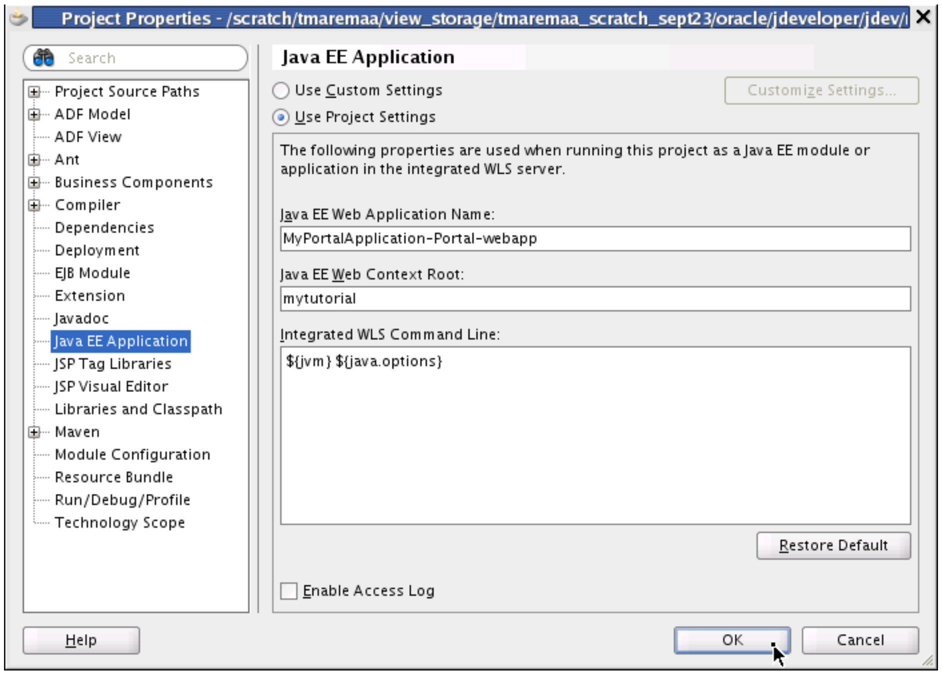
***Figure 3–12 The Project Properties Menu Item Selected to Change the Context Root of the Application***


Run
Debug
Run Unit Tests in Portal.jpr
Debug Unit Tests in Portal.jpr
Reformat Ctrl+Alt-L
Organize Imports Ctrl+Alt-O
Update Working Copy...
Versioning
Compare With
Replace With
Restore from Local History...
Refresh ADF Library Dependencies in Portal.jpr
Project Properties...
Import Portal Resource...

13. Select the **Java EE Application** node. In the Java EE Web Context Root field, enter `mytutorial`, as shown in Figure 3–13, and click **OK**.

Ensure that the Java EE web context root is set to `mytutorial`.

***Figure 3–13 The Java EE Application Dialog with mytutorial Entered in the Java EE Web Context Root Field***



Java EE Application

Use Custom Settings [✓ Use Project Settings]

The following properties are used when running this project as a Java EE module or application in the integrated WLS server.

Java EE Web Application Name: MyPortalApplication-Portal-webapp

Java EE Web Context Root: mytutorial

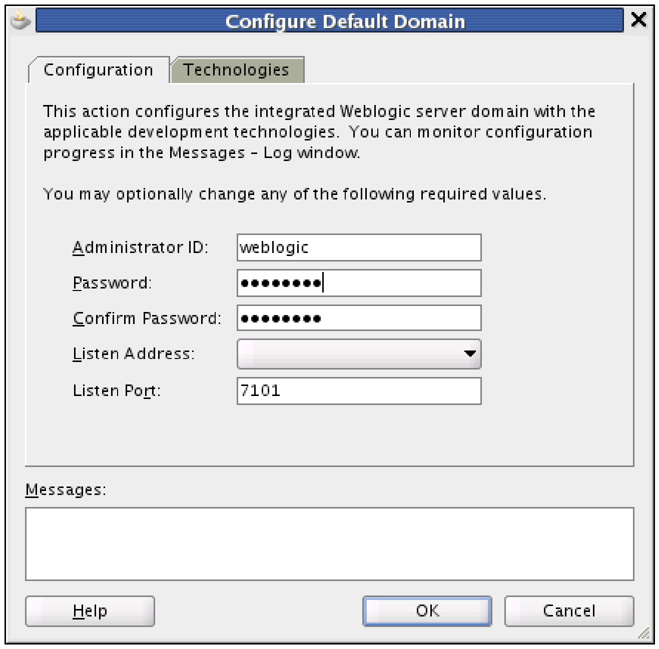
Integrated WLS Command Line: ${jvm} ${java.options}

Enable Access Log

**14.** In Application Navigator, right-click the **Portal** project and choose **Run** to run the application.

A Configure Default Domain dialog appears, as shown inFigure 3–14. In the Administrator ID field, enter `weblogic`. In the Passworld field, enter `weblogic1`. Note that you can change any of these values at your discretion.

***Figure 3–14 The Configure Default Domain Dialog to Enter Administrator ID and Password***



This action configures the integrated Weblogic server domain with the applicable development technologies. You can monitor configuration progress in the Messages - Log window.

You may optionally change any of the following required values.

| | |
|---|---|
| Administrator ID: | weblogic |
| Password: | •••••••• |
| Confirm Password: | •••••••• |
| Listen Address: | |
| Listen Port: | 7101 |

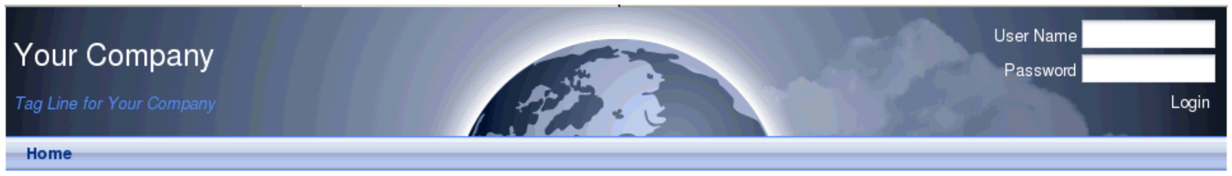
Messages:

**15.** Click **OK**.

Oracle JDeveloper now builds the application out of the box and displays the default portal page Home in a web browser, as shown in Figure 3–15.

Initially, this portal displays a single page, rendered as `Home`. The `Home` page is based on the seeded Globe page template (discussed in the next section), which provides all the initial functionality of the portal, including a banner, a login form with User Name and Password fields, and a navigation menu with a single link element -- `Home` -- displayed on the web page.

***Figure 3–15 The Default Home Portal Page in a Web Browser***



In the upper right corner of the `Home` page, in the **User Name** field (Figure 3–16), you can log into the Home page. Enter `weblogic` as the User Name. (Note that the weblogic user is seeded in the integrated WebLogic Server.) In the Password field, enter `weblogic1`.

***Figure 3–16 Enter User Name and Password To Log in to Home Page***

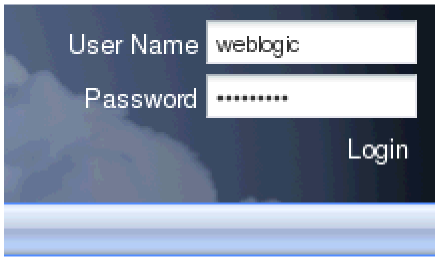
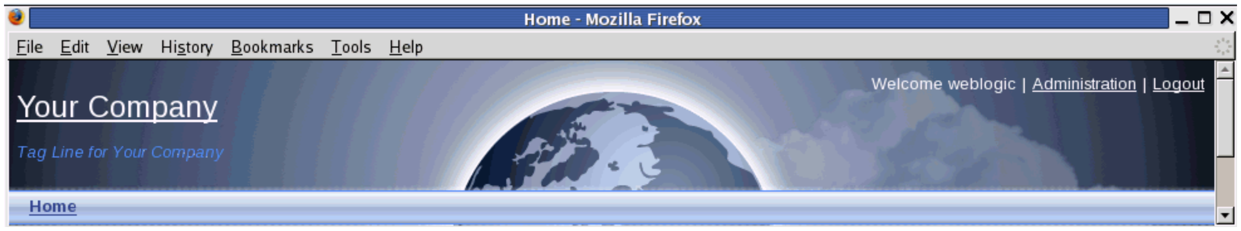


Figure 3–17 shows the portal Home page with Administrator privileges enabled after successfully logging in.

Note that Administrator privileges are now enabled because you can see the **Administration** link in the upper right corner of the web browser. This means the user, specified here as `weblogic`, has administration privileges for the portal.

***Figure 3–17 The Home Page with Administrator Privileges Enabled After Successful Login***


Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Welcome weblogic | Administration | Logout
Your Company
Tag Line for Your Company
Home

For more information on creating an application based on the WebCenter Portal Application template, see “Preparing Your Development Environment” in the *Oracle Fusion Middleware Developer’s Guide for Oracle WebCenter Portal*.

## Step 2: Use Seeded Page Templates to Build Your Portal Application

When you create a portal, you will base its look and feel on a **page template**. Page templates enable you to maintain a consistent look and feel across all the pages in your portal, and typically determine the artifacts, like banners, footers and navigation bars, that surround the main content of the page.

Using JDeveloper, you can create and publish page templates. In addition, you can also modify them to meet specific design or runtime requirements in your portal application.

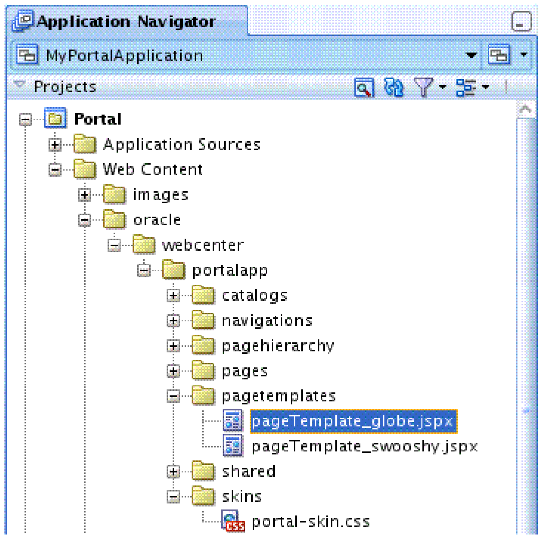
By selecting the **Configure the application with standard Portal features** option, as shown in Figure 3–7, "Configure WebCenter Settings - Step 4 of 5", two seeded, out-of-the-box templates are added by default to your portal application: `pageTemplate_globe.jspx` (shown in Figure 3–19) and `pageTemplate_swooshy.jspx` (Figure 3–20).

Both templates offer essentially the same functionality but with a different set of graphics.

To view the `pageTemplate_globe.jspx` template:

1. Navigate to the `pagetemplates` folder in your portal project.
2. Double-click the folder and select the `pageTemplate_globe.jspx` file, as shown in Figure 3–18.

***Figure 3–18 The pageTemplate_globe.jspx File in the Page Templates Folder***



3. Right-click the page template and choose **Open**. The file opens in JDeveloper, as shown in Figure 3–19.

Ensure that you select the **Design** tab in the lower left corner.

***Figure 3–19 The pageTemplate_globe.jspx Seeded Page Template***

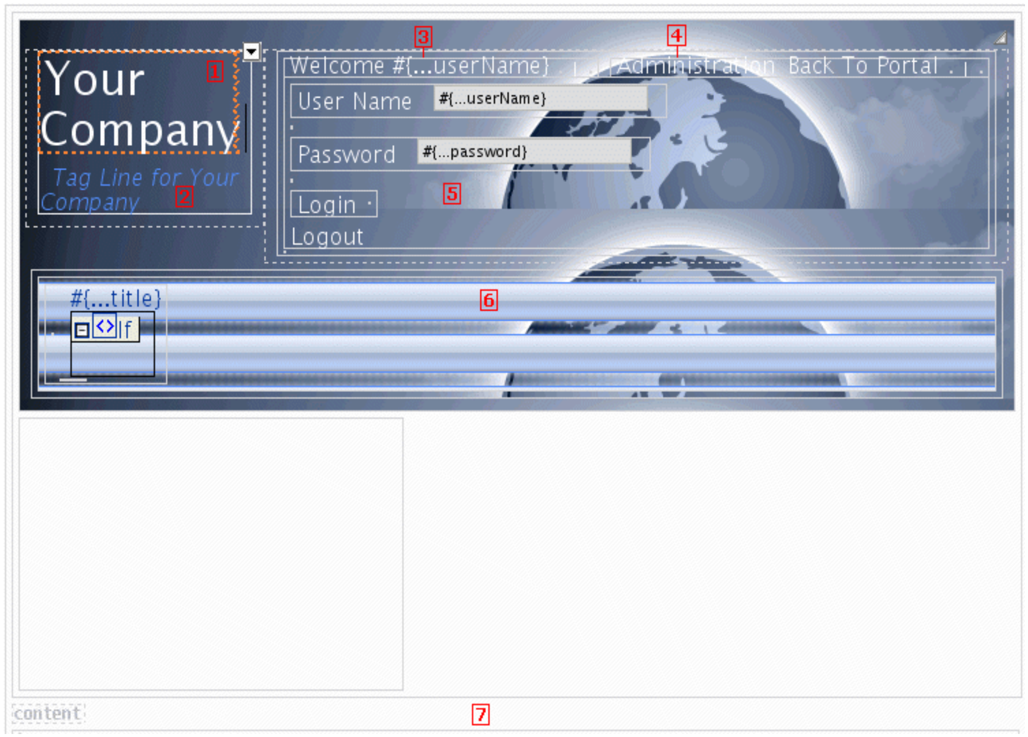


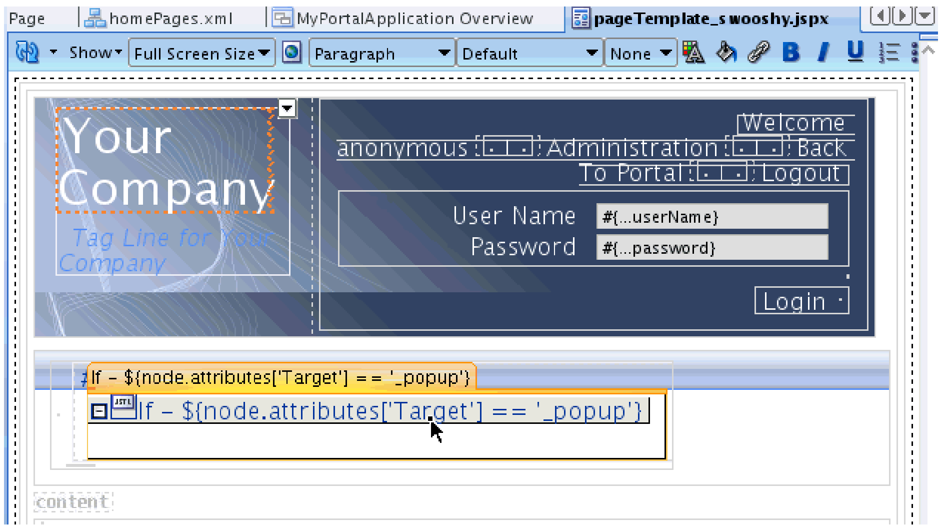
Figure 3–19 shows the `pageTemplate_globe.jspx` page template with each of its page features and artifacts (enumerated below) called out with their corresponding numbers in the illustration.

1. A link to the portal home page
2. A tag line

3. A welcome message
4. A link to the seeded Administration page
5. A login area that converts to a logout link when users are logged in
6. A navigation bar
7. An area for adding content to pages based on the template
8. A copyright notice

Figure 3–20 shows the `pageTemplate_swooshy.jspx` page template, but without each of its page features and artifacts enumerated.

***Figure 3–20 The Seeded PageTemplate swooshy.jspx***



Using JDeveloper, you can modify and edit this default page template to meet your particular requirements. You can also create and build your own page template, as we discover in the next lesson in this Tutorial, ensuring that it has a common navigation bar, footer, and banner, then leave it up to your content contributors to populate the portal with content at runtime.

Following the steps outlined in this Tutorial, you have created a new portal application using Oracle JDeveloper.

In the next lesson, you will learn how to modify and edit an existing page template, with the goal of customizing its behavior to meet the particular needs of your end users. By completing that task, you will create a new page template in your portal application, further customizing its look and feel, and then set that template as an application resource.

4

# Creating a New Page Template with a New Portal Skin

In this lesson, working as a developer at design time, you will enhance the WebCenter portal application you constructed in the previous lesson and learn how to create a new JSF page template and register that template as a portal resource.

To achieve this goal, you will need to create a new page template, then extract the setup files provided with this Tutorial on your hard drive. The setup contains a batch of files with graphic images, skins and templates. You will then replace the new page template created by you in the application with the page template provided in the setup.

In the last step, you will register the new template and customize the site template, adding new images and a new skin to your portal application. When you run the application in a web browser, you will see a new home page with a new skin applied at runtime.

## Introduction

This lesson contains the following steps:

- Step 1: Create a New Page Template
- Step 2: Extract Setup Files and Replace the Existing Template
- Step 3: Create Portal Resources and Apply the New Template and Skin

---

**Note:** Before you begin the steps in this lesson, ensure you have followed the steps up to this point in the Tutorial.

---

## Step 1: Create a New Page Template

To extend the capabilities of our portal application, we need to create a new page template.

Note that in this step, we won't build a new JSF page template from scratch. But rather, we'll rely on a pre-configured, ready-made template which, following the steps in this lesson, you will extract into your application. To ensure that the template artifacts are correctly registered inside the application, we will create an empty template and then replace it with the one provided in the Tutorial Setup file.

You can use page templates to control the layout of your portal. A **page template** is a JSPX file that specifies the look and feel of your portal's pages. The template defines header, footer, content, and navigation regions within the page. You can apply the template to any number of pages, resulting in a consistent look and feel.

> **Tip:** The template is linked or referenced from the pages, so if you change the template, those changes are reflected on all the pages in your portal application.

For more information about page templates, see "Understanding Pages, Page Templates, and the Portal Page Hierarchy" in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

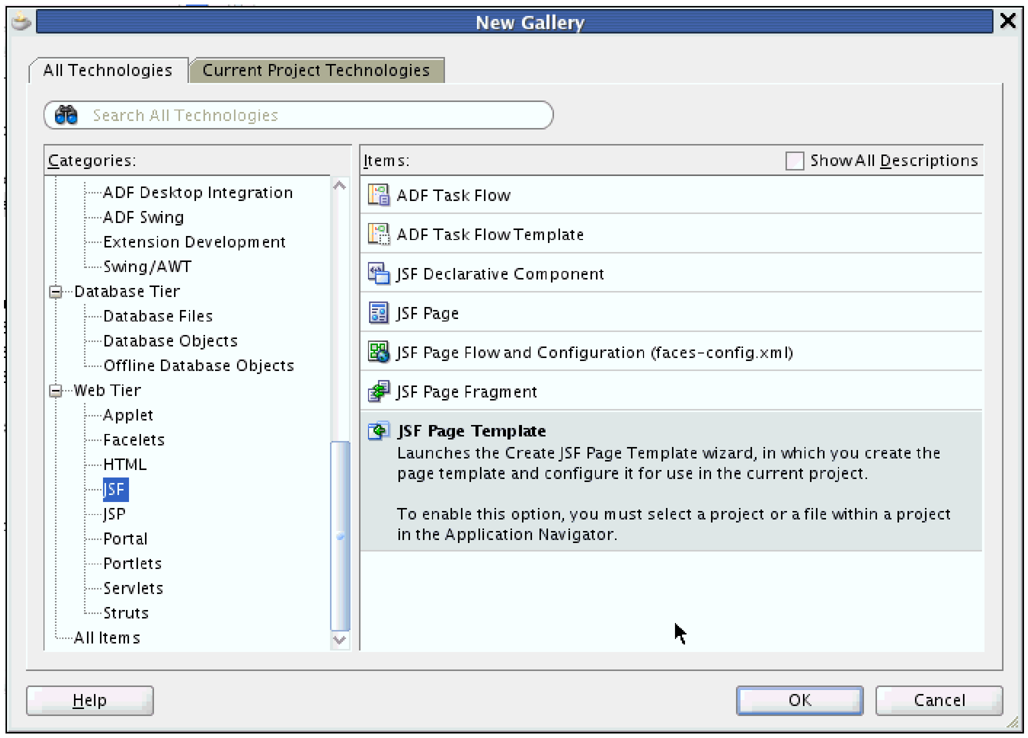
To create a new page template in our portal application:

1. In the Application Navigator of your portal application project, navigate to the page templates folder (`/oracle/webcenter/portalapp/pagetemplates`) and right-click the folder and choose New.

   A New Gallery dialog appears, as shown in Figure 4–1.

2. In the **New Gallery**, expand **Web Tier**, select **JSF** and then **JSF Page Template**, and click **OK**.

***Figure 4–1 The JSF Page Template Selected in the New Gallery***



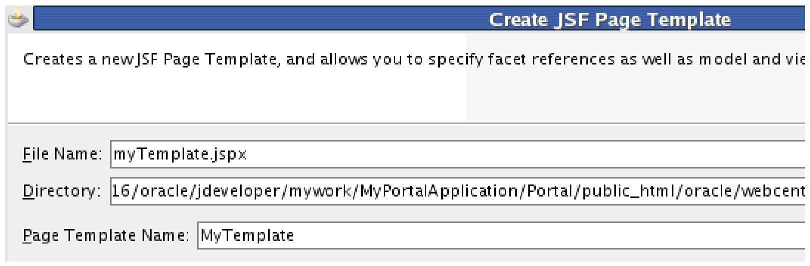
**JSF Page Template**
Launches the Create JSF Page Template wizard, in which you create the page template and configure it for use in the current project.

To enable this option, you must select a project or a file within a project in the Application Navigator.

3. In the Create JSF Page Template dialog (Figure 4–2), in the **File Name** field, enter the name for the JSPX file that represents the page template, in this case `myTemplate.jspx`.

   The file name identifies the page template in the Application Navigator.

***Figure 4–2 The Create JSF Page Template***


Create JSF Page Template

Creates a new JSF Page Template, and allows you to specify facet references as well as model and vie

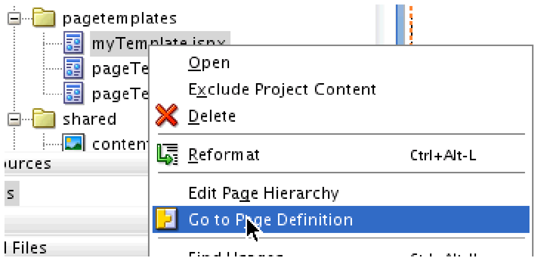
File Name: myTemplate.jspx

Directory: 16/oracle/jdeveloper/mywork/MyPortalApplication/Portal/public_html/oracle/webcent

Page Template Name: MyTemplate

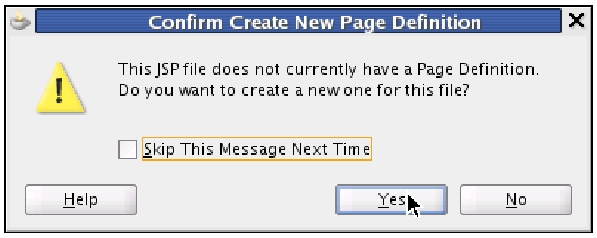
4. In the **Directory** field (Figure 4–2), enter the full directory path of the location under which to create the page template.
5. In the **Page Template Name** field (Figure 4–2), enter the display name for the page template, in this case `MyTemplate`.
6. Click **OK** to create the template.
7. Navigate in the Application Navigator to the `pagetemplates` folder, select `myTemplate.jspx`, then right-click the **Go to Page Definition** menu item, as shown in Figure 4–3.

***Figure 4–3 Creating a New Page Definition for the myTemplate.jspx File***



8. When the dialog Confirm Create New Page Definition appears (Figure 4–4), click **Yes**.

***Figure 4–4 The Confirm Create New Page Definition Dialog***


Confirm Create New Page Definition

This JSP file does not currently have a Page Definition.
Do you want to create a new one for this file?

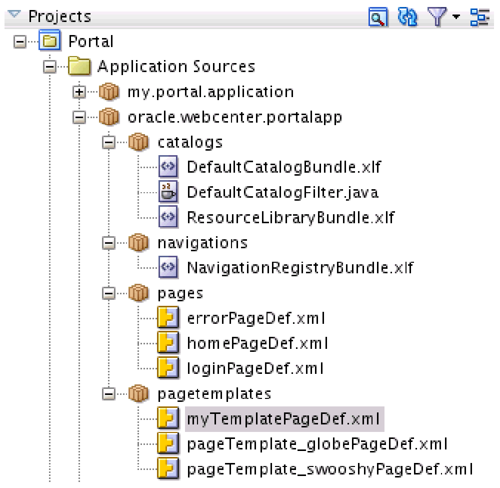
Skip This Message Next Time

Help Yes No

> **Tip:** A page definition file is an `XML` file that specifies ADF bindings, page parameters, and permission settings. Various mappings and bindings used by pages and page templates are also specified. In this case, the `myTemplatePageDef.xml` file specifies the task flow for navigation rendering of the site, as well as parameters defining site structure paths.
>
> The `Application Sources` folder is primarily a repository for page definition files, like the `myTemplatePageDef.xml` file, as well as for source code in a project.

9. Verify that the `myTemplatePageDef.xml` file now resides in the Application Sources sub folder `pagetemplates`, as shown in Figure 4–5.

***Figure 4–5 The myTemplatePageDef xml File in the Portal Application Sources Directory***



```
Projects
Portal
  Application Sources
    my.portal.application
    oracle.webcenter.portalapp
      catalogs
        DefaultCatalogBundle.xlf
        DefaultCatalogFilter.java
        ResourceLibraryBundle.xlf
      navigations
        NavigationRegistryBundle.xlf
      pages
        errorPageDef.xml
        homePageDef.xml
        loginPageDef.xml
      pagetemplates
        myTemplatePageDef.xml
        pageTemplate_globePageDef.xml
        pageTemplate_swooshyPageDef.xml
```

By associating a page definition with the page template, you will be able to include model objects, such as task flows and portlets, in the page template. Users can also switch to a different page template at runtime, if they choose.

It's important to note that within your portal application, page templates must either all have associated page definitions or none have associated page definitions. The reason for this is that if you have a combination of page templates with and without associated page definitions, users won't be able to switch templates at runtime.

For more information about templates and skins, see the chapter "Designing the Look and Feel of Your Portal" in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

## Step 2: Extract Setup Files and Replace the Existing Template

Now you want to extract the provided Tutorial setup files from a folder (`owcs-r11ps3-devtutsetup-254761.zip`) that resides on your local hard drive and then move those files to the appropriate folders in the WebCenter Portal application.

If you have not yet downloaded these setup files, as described in "Step 6: Download the Sample Tutorial Files" in Chapter 2, then do so now. You can download the files from your web browser. The files are available at this URL address:

`http://www.oracle.com/technetwork/middleware/webcenter/owcs-r11ps3-devtutsetup-254761.zip`

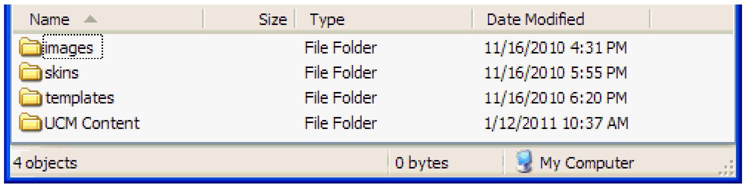
To extract the setup files and place them in the correct location in your newly created application in JDeveloper:

1. To begin with, you need to copy the `owcs-r11ps3-devtutsetup-254761.zip` file onto your hard drive in the directory of your choosing, and then proceed to unzip the files and extract their contents, as described in the following steps.

2. On your local drive navigate to, for example, `C:\...\USERS\Desktop\TutorialSetup\owcs-r11ps3-devtutsetup-254761.zip\`.

   Four folders reside in that directory: images, skins, templates, and UCM Content, as shown in Figure 4–6. Note that the UCM Content folder contains content that you need to upload to a UCM content repository, as discussed in Section , "Step 6: Download the Sample Tutorial Files."

***Figure 4–6 Images, Skins, Templates, and Content Folders for Setup on Local Hard Drive***

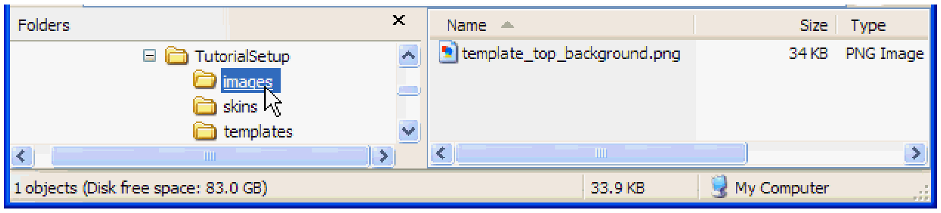


| Name | Size | Type | Date Modified |
|---|---|---|---|
| images | | File Folder | 11/16/2010 4:31 PM |
| skins | | File Folder | 11/16/2010 5:55 PM |
| templates | | File Folder | 11/16/2010 6:20 PM |
| UCM Content | | File Folder | 1/12/2011 10:37 AM |

4 objects | 0 bytes | My Computer

3. Extract the contents of the `images` folder (Figure 4–7) and move those contents to the `MyApplication/Portal/public_html/images` folder in your portal application.

***Figure 4–7 The Expanded Images Folder in the Tutorial Setup Directory***

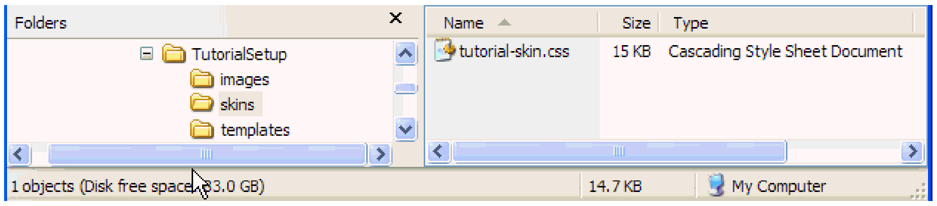


| Name | Size | Type |
|---|---|---|
| template_top_background.png | 34 KB | PNG Image |

1 objects (Disk free space: 83.0 GB) | 33.9 KB | My Computer

4. Repeat the same procedure for contents of the `skins` folder (Figure 4–8), moving those contents to the `MyPortalApplication/Portal/public_html/oracle/webcenter/portalapp/skins/` folder in your portal application. Note that the extracted skin is a Cascading Style Sheet (CSS) document.

***Figure 4–8 Expanded Skins Folder in the Tutorial Setup Directory***

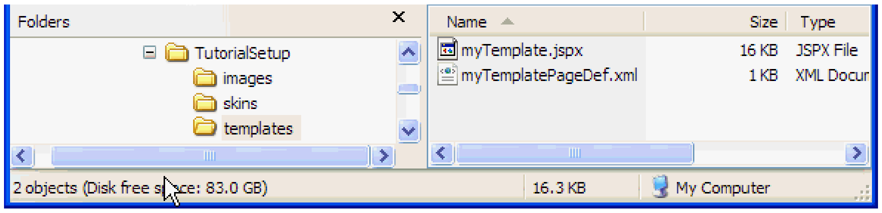


| Name | Size | Type |
|---|---|---|
| tutorial-skin.css | 15 KB | Cascading Style Sheet Document |

1 objects (Disk free space: 83.0 GB) | 14.7 KB | My Computer

5. Repeat again the same procedure for the contents of the `templates` folder (Figure 4–9), moving those contents to two separate locations, in this case moving the `myTemplate.jspx` template to the `MyPortalApplication/Portal/public_html/oracle/webcenter/portalapp/pagetemplates/` folder and the `myTemplatePageDef.xml` to the `MyPortalApplication/Portal/adfmsrc/oracle/webcenter/portalapp/pagetemplates` folder.

***Figure 4–9 Expanded Templates Folder in the Tutorial Setup Directory***



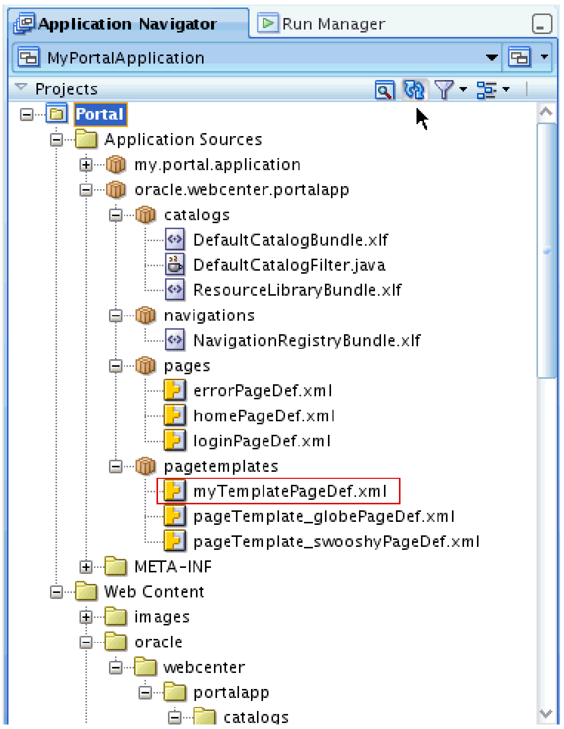
| Name | Size | Type |
|---|---|---|
| myTemplate.jspx | 16 KB | JSPX File |
| myTemplatePageDef.xml | 1 KB | XML Docur |

```
2 objects (Disk free space: 83.0 GB)    16.3 KB    My Computer
```

6. Now select the Portal folder at the top level of your Project and click the **Refresh** icon in JDeveloper (not your web browser). This will refresh and save each of the folders whose contents you have extracted and copied to your portal application in JDeveloper, as shown in Figure 4–10.

   The `myTemplatePageDef.xml` file now resides in the `pagetemplates` folder.

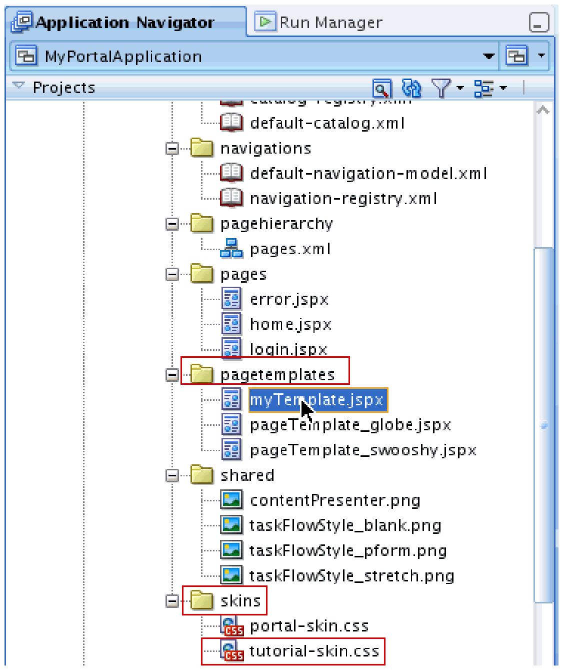
***Figure 4–10 The Portal Hierarchy Refreshed to Include the Extracted Files for Setup***



```
Portal
  Application Sources
    my.portal.application
    oracle.webcenter.portalapp
      catalogs
        DefaultCatalogBundle.xlf
        DefaultCatalogFilter.java
        ResourceLibraryBundle.xlf
      navigations
        NavigationRegistryBundle.xlf
      pages
        errorPageDef.xml
        homePageDef.xml
        loginPageDef.xml
      pagetemplates
        myTemplatePageDef.xml
        pageTemplate_globePageDef.xml
        pageTemplate_swooshyPageDef.xml
    META-INF
  Web Content
    images
    oracle
      webcenter
        portalapp
          catalogs
```

7. Once you refresh the page templates and skins folders in your portal application, the copied files, `myTemplate.jspx` and `tutorial-skin.css`, appear in their respective folders, as shown in Figure 4–11.

***Figure 4–11 myTemplate.jspx file and tutorial-skin.css File in the Portal Project Folders***



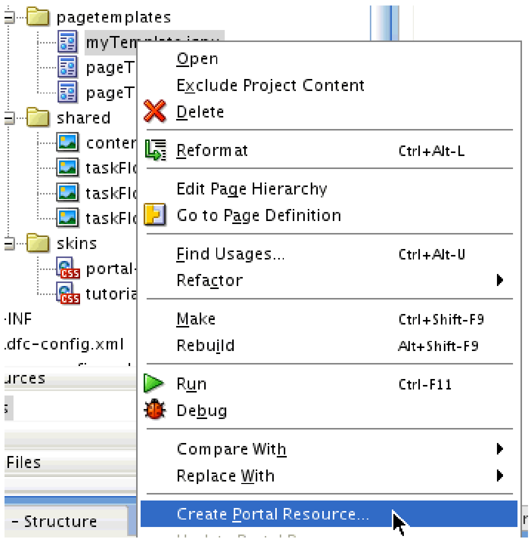
8. Close the Portal `Application Sources` folders and navigate to the `webcenter` folder in your project directory.

## Step 3: Create Portal Resources and Apply the New Template and Skin

In this next sequence of steps, you will create a portal resource at design time to customize both the site template and apply the newly provided tutorial skin.

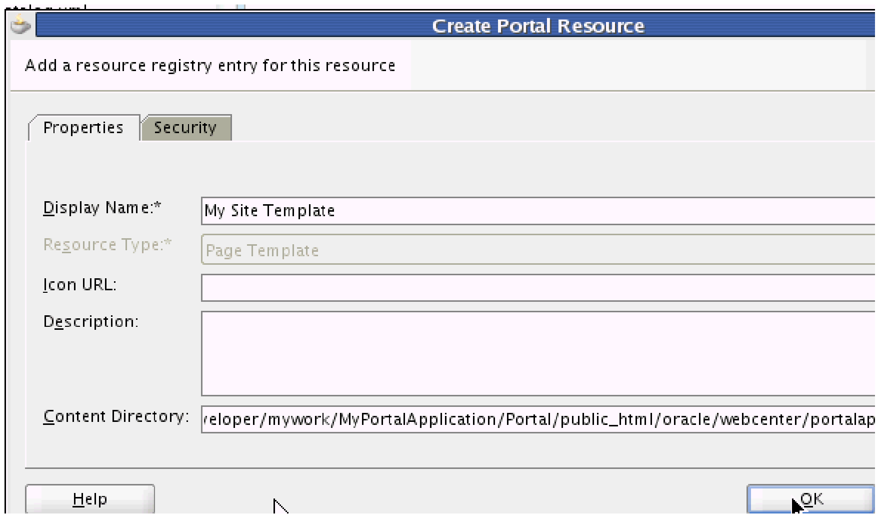
1. Open the `webcenter` folder in your portal project and navigate to the `pagetemplates` folder in the directory.
2. Select the `myTemplate.jspx` file and right-click the file.
3. Select the **Create Portal Resource** menu item, shown in Figure 4–12.

*Figure 4–12 The Create Portal Resource Menu Item for the myTemplate.jspx File*



4. In the Create Portal Resource dialog, enter in the Display Name field `My Site Template` (Figure 4–13) and click **OK**.

*Figure 4–13 The Create Portal Resource Dialog with the Display Name Specified as My Site Template*


Create Portal Resource

Add a resource registry entry for this resource

Properties | Security

Display Name:* My Site Template

Resource Type:* Page Template

Icon URL:

Description:

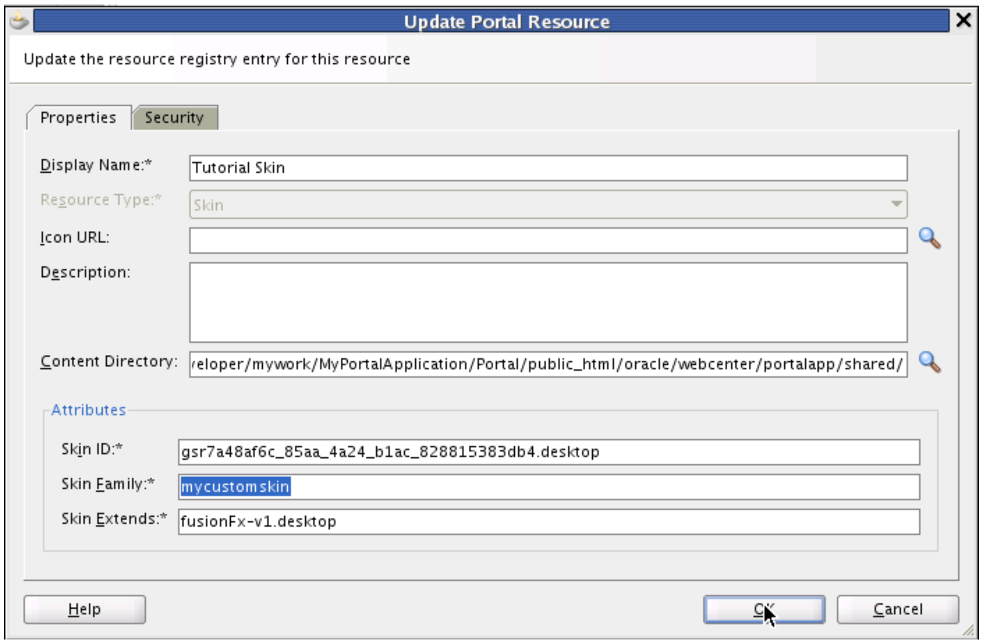
Content Directory: veloper/mywork/MyPortalApplication/Portal/public_html/oracle/webcenter/portalap

Help | OK

5. Navigate to the `skins` folder and open it. Select the `tutorial-skin` file and right-click the **Create Portal Resource** menu item, as performed in the previous step.

6. Change the Display Name to `Tutorial Skin`, and in the Skin Family field, enter `mycustomskin`, as shown in Figure 4–14.

***Figure 4–14 The Update Portal Resource Dialog with mycustomskin Specified as Skin Family Attribute***



Update the resource registry entry for this resource

| | |
|---|---|
| Display Name:* | Tutorial Skin |
| Resource Type:* | Skin |
| Icon URL: | |
| Description: | |
| Content Directory: | /eloper/mywork/MyPortalApplication/Portal/public_html/oracle/webcenter/portalapp/shared/ |

Attributes

| | |
|---|---|
| Skin ID:* | gsr7a48af6c_85aa_4a24_b1ac_828815383db4.desktop |
| Skin Family:* | mycustomskin |
| Skin Extends:* | fusionFx-v1.desktop |

7. Click **OK**.

8. Now select the **Portal** project in Application Navigator and right-click **Run** to run the portal application in JDeveloper. The portal displayed in the web browser shows the original template, with its default skin and standard portal application look-and-feel.

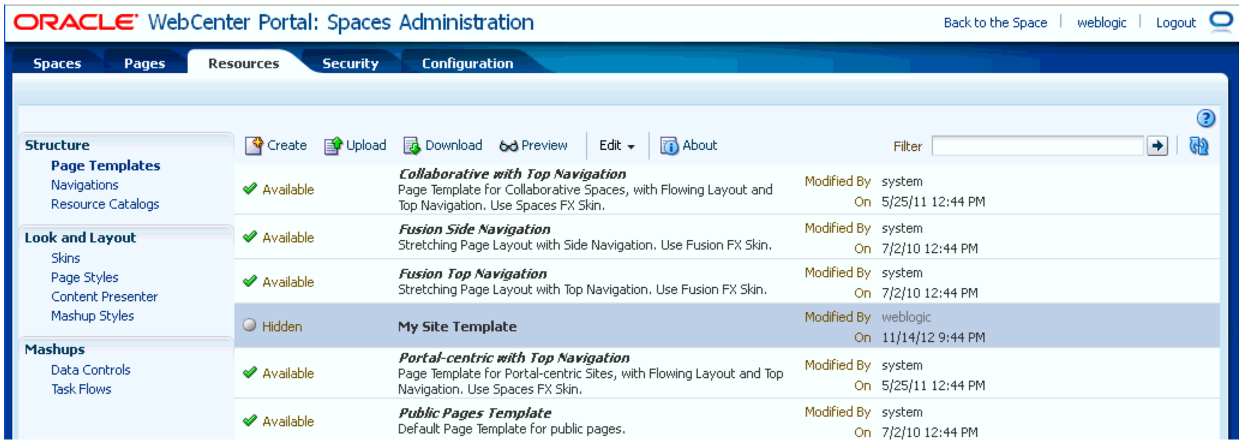
9. In the default Home portal page in the web browser, log in as `weblogic` (which enables you to have administrative privileges) and enter `weblogic1` as your password.

   Note that as discussed in Chapter 3, "Creating a WebCenter Portal Application," you must log in as a user with administrative privileges. In the Tutorial, the user "weblogic" has administrative privileges.

10. After logging in, click the **Administration** link in the upper right corner of the browser window.

11. When the Administration Console opens, select the **Resources** tab, and navigate to **Page Templates** in the **Structure** menu, as shown in Figure 4–15.

*Figure 4–15 The My Site Template as a Designated Resource in the Administration Console*


ORACLE WebCenter Portal: Spaces Administration — Back to the Space | weblogic | Logout

Spaces | Pages | Resources | Security | Configuration

Structure: Page Templates, Navigations, Resource Catalogs
Look and Layout: Skins, Page Styles, Content Presenter, Mashup Styles
Mashups: Data Controls, Task Flows

Create | Upload | Download | Preview | Edit | About | Filter

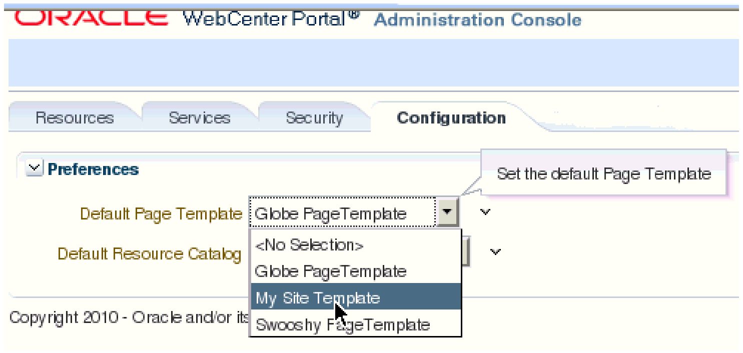
| | | | |
|---|---|---|---|
| ✓ Available | *Collaborative with Top Navigation* Page Template for Collaborative Spaces, with Flowing Layout and Top Navigation. Use Spaces FX Skin. | Modified By system | On 5/25/11 12:44 PM |
| ✓ Available | *Fusion Side Navigation* Stretching Page Layout with Side Navigation. Use Fusion FX Skin. | Modified By system | On 7/2/10 12:44 PM |
| ✓ Available | *Fusion Top Navigation* Stretching Page Layout with Top Navigation. Use Fusion FX Skin. | Modified By system | On 7/2/10 12:44 PM |
| ○ Hidden | My Site Template | Modified By weblogic | On 11/14/12 9:44 PM |
| ✓ Available | *Portal-centric with Top Navigation* Page Template for Portal-centric Sites, with Flowing Layout and Top Navigation. Use Spaces FX Skin. | Modified By system | On 5/25/11 12:44 PM |
| ✓ Available | *Public Pages Template* Default Page Template for public pages. | Modified By system | On 7/2/10 12:44 PM |

12. By default, `My Site Template` is hidden. With that row selected, from the **Edit** menu, choose **Show** to change its state to **Available**. A green checkmark and the word "Available" appear next to it now, and it is available for use in the application.

13. Similarly, under the **Look and Layout** heading, choose **Skins**. With `Tutorial Skin` selected, from the **Edit** menu, choose **Show** to change its state to **Available**.

14. On the Administration Console, navigate to the **Configuration** tab. For Default Page Template menu, select `My Site Template` as the default page template, as shown in Figure 4–16.

*Figure 4–16 The Default Template Changed to My Site Template*


ORACLE WebCenter Portal® Administration Console

Resources | Services | Security | Configuration

Preferences

Set the default Page Template

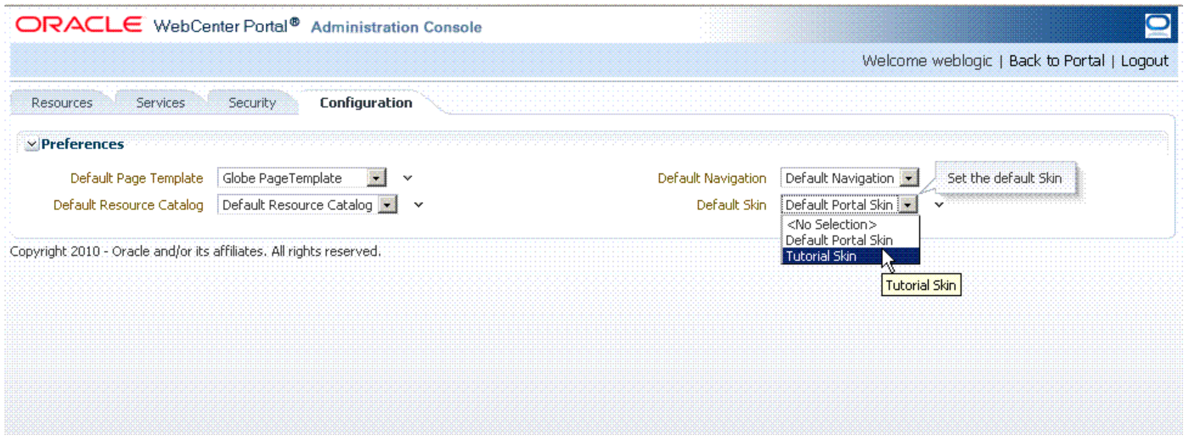
Default Page Template: Globe PageTemplate
<No Selection>
Globe PageTemplate
My Site Template
Swooshy PageTemplate

Default Resource Catalog

Copyright 2010 - Oracle and/or its

15. For **Default Skin**, select `Tutorial Skin` from the list. Now set the default portal skin to `Tutorial Skin` (Figure 4–17).

*Figure 4–17 The Default Portal Skin Changed to Tutorial Skin*


ORACLE WebCenter Portal® Administration Console
Welcome weblogic | Back to Portal | Logout
Resources
Services
Security
Configuration
Preferences

Default Page Template: Globe PageTemplate
Default Resource Catalog: Default Resource Catalog
Default Navigation: Default Navigation
Default Skin: Default Portal Skin
<No Selection>
Default Portal Skin
Tutorial Skin
Set the default Skin
Tutorial Skin

Copyright 2010 - Oracle and/or its affiliates. All rights reserved.

**16.** Click the **Back to Portal** link in the Administration Console.

In this lesson, you have learned how to enhance your portal application by creating a new page template, setting that template as a portal resource and applying a new skin (extracted from the Tutorial Setup files and copied into the skins folder in your project) to your portal to change its look and feel at runtime.

In the next lesson, you move ahead to further customize your application portal by changing the default settings of your template at design time in JDeveloper.

# 5 Changing the Look and Feel of Your Portal Application

In the previous lesson, you changed the default template in your portal application to `myTemplate` and proceeded to change the default skin to the tutorial skin provided in the folder on your hard drive whose contents you extracted. These changes then appeared in your web browser as a new template and a new skin when you built and ran your application in JDeveloper.

Now in this lesson, you will move ahead to change the default settings for both your template and skin at design time in JDeveloper. When you launch your portal application again in a web browser, these changes will show the new default settings with changed preferences, as well as the new template and skin. In so doing, you'll learn how to change the look and feel of your portal application at design time and how to apply skins to your portal.

A **skin** is essentially a global style sheet (based on the Cascading Style Sheet specification [CSS]) that you can apply to your entire application. Once you do that, every layout component automatically uses the styles assigned by the skin. You cannot edit that skin at runtime or post-deployment, however.

Skins are important because they enable you to define the appearance of your application and achieve some degree of consistency across multiple pages, so that you can more effectively communicate your company's preferred look and feel.

## Introduction

This lesson contains the following steps:

- Step 1: Change the Default Settings For Template and Skin
- Step 2: Change the Default Page Template at Runtime

Before you begin the steps in this lesson, ensure you have followed the steps up to this point in the Tutorial.

## Step 1: Change the Default Settings For Template and Skin

When you create a WebCenter Portal Application using the WebCenter Application template, a skin is included by default. In this Tutorial, you've extracted a custom skin provided for you, which you've then applied in place of the default skin. Now you need to change the default settings for both the skin and the provided template by changing their preference entries.

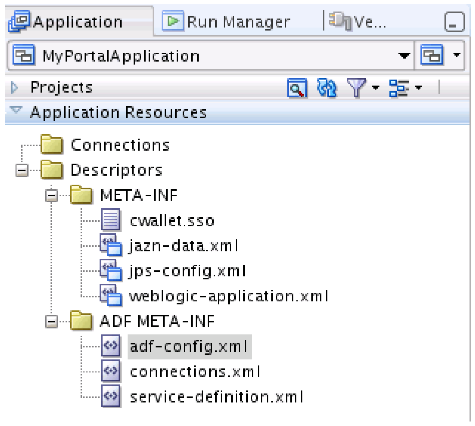
To change the default preferences of your portal application at design time, you will need to directly edit the `adf-config.xml` file in your project. The steps to accomplish this task are described in this section, as follows.

To change the default settings for the skin and template:

1. Open `adf-config.xml`.

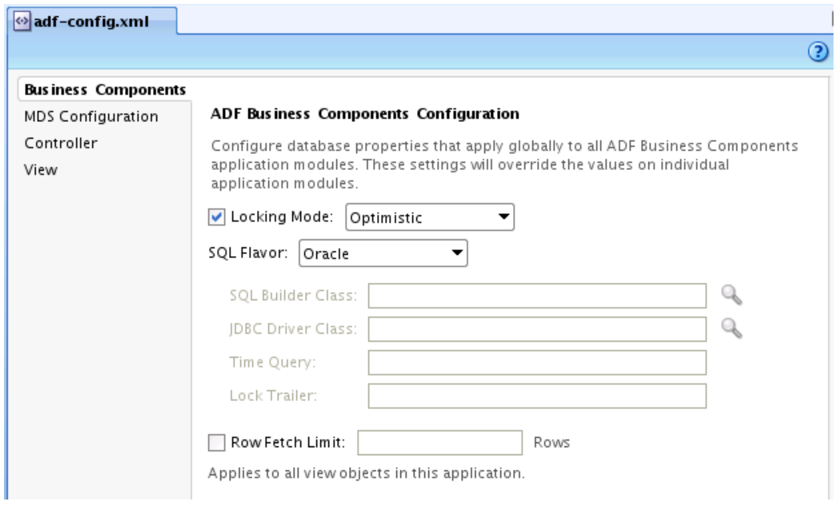
   To locate this file in JDeveloper, open the **Application Resources** part of the Application Navigator. Then, open the **Descriptors** folder and the **ADF META-INF** folder, as shown in Figure 5–1.

***Figure 5–1 Location of the adf-config.xml File in JDeveloper***



2. In the **ADF META-INF** folder, select the `adf-config.xml` file and open it. The file appears in the **Overview** tab, as shown in Figure 5–2.

***Figure 5–2 The adf-config.xml file Specifying Component Configuration***



Configure database properties that apply globally to all ADF Business Components application modules. These settings will override the values on individual application modules.

Locking Mode: Optimistic

SQL Flavor: Oracle

Row Fetch Limit:

Applies to all view objects in this application.

3. Now click the **Source** view tab in the JDeveloper window to view the XML source contents of the file.

4. In the Search field of the `adf-config.xml` file, enter the word `preferences`. Navigate in the XML schema to this code (Example 5–1):

***Example 5–1 The XML Code Specifying the Default Page Template***

```
<portal:preference id="oracle.webcenter.portalapp.pagetemplate.pageTemplate"
                   desc="Default Page Template"
         value="/oracle/webcenter/portalapp/pagetemplates/pageTemplate_globe.jspx"
                   resourceType="Template" display="true"/>
```

5. Change the `value` attribute to `myTemplate.jspx` and change the `desc` attribute to "My Site Template", as shown in Example 5–2.

***Example 5–2 Changed XML Template Code***

```
       value="/oracle/webcenter/portalapp/pagetemplates/myTemplate.jspx"
             desc="My Site Template"
```

6. Navigate in the preference schema to the `desc` attribute `"Default Portal Skin"` and the `value` attribute "`portal`", shown in Example 5–3. Select "`portal`" and change it to "`mycustomskin`".

***Example 5–3 The Value Attribute of the Default Portal Skin***

```
<portal:preference id="oracle.webcenter.portalapp.skin"
                   desc="Default Portal Skin" value="portal"
```

7. In the **Source** view of the `adf-config.xml` file, note that the `value` attribute is now updated as "`mycustomskin`", as shown in Example 5–4.

***Example 5–4 The Portal Skin Value Attribute Changed to "mycustomskin"***

```
<portal:preference id="oracle.webcenter.portalapp.skin"
                   desc="Default Portal Skin" value="mycustomskin"
```

8. Change the `desc` attribute from "`Default Portal Skin`" to "`Tutorial Skin`", shown in Example 5–5.

***Example 5–5 The Changed desc Attribute to Tutorial Skin***

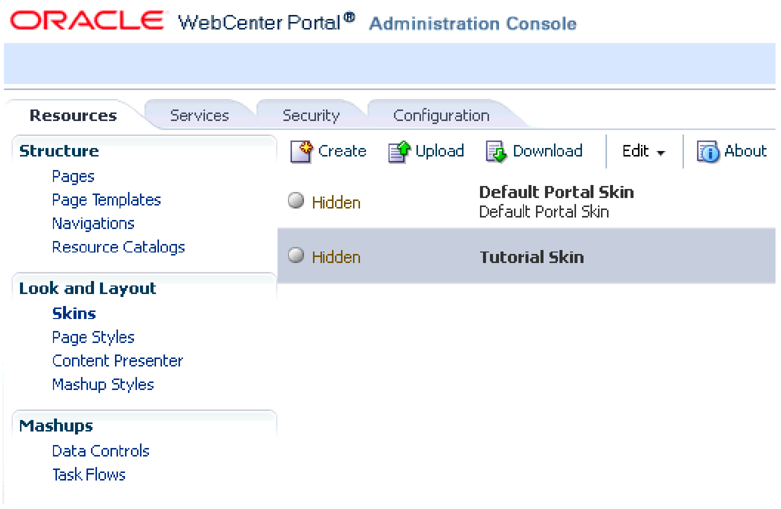
```
<portal:preference id="oracle.webcenter.portalapp.skin"
                   desc="Tutorial Skin" value="mycustomskin"
                   resourceType="Skin" display="true"/>
```

9. Save the `adf-config.xml` file.
10. Right-click the Portal project in Application Navigator and select **Run** to build and launch the application in JDeveloper.
11. When the `Home` page appears in a web browser, log in as User `weblogic` and Password as `weblogic1` to log in and enable Administrator privileges.
12. In the Administration Console, click the **Resources** tab and click the **Skins** item in the Look and Layout list. As a result of the changes you've made through steps 5 and 8, the `Tutorial Skin`, when checked, will be available to the application (Figure 5–3).

**Tip:** The Administration Console lets you work with resources, services, security, and portal configurations at runtime. The Administration Console includes a **Resources** tab that lets you work with several portal-specific features at runtime, like pages, page templates, navigation models, resource catalogs, skins, page style, task flows, and so on.

The Resource Manager enables portal administrators to manage these resources at runtime. Using the Resource Manager, portal users can also download resources, or an entire application, from the runtime environment, edit them in JDeveloper, and then upload them back into the deployed application.

***Figure 5–3 The Tutorial Skin Available When Checked as Skin Resource***



Hidden Default Portal Skin
Default Portal Skin

Hidden Tutorial Skin

Since you are still developing your application (and have not yet deployed it), you can continue to switch back and forth between the runtime view and design time in Oracle JDeveloper to modify the look and feel.

For more information about changing default templates and applying different skins at design time, see *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

## Step 2: Change the Default Page Template at Runtime

The steps that follow describe how you can change the default page template at runtime. You accomplish this by logging in as a user with administrative privileges and clicking the **Administration** link in the web browser to access the Administration Console. Once in the Administration Console, you will select the **Resources** tab that lets you work with portal-specific features at runtime, like page templates.

The values you enter in the Administration Console, modifying or changing page templates, will be lost the next time you choose **Run** from JDeveloper, however.

**Tip:** It's important to understand that if you are using the Integrated WebLogic Server in a development environment, running your portal application through JDeveloper as we've been doing in this Tutorial, then any changes you make to the portal at runtime, using the Resource Manager, will be discarded upon redeployment by default. If you use the Resource Manager, for example, to make changes like adding entitlements to a page, changing the layout, or modifying the navigation model, those changes will not be preserved the next time you redeploy your application.

---

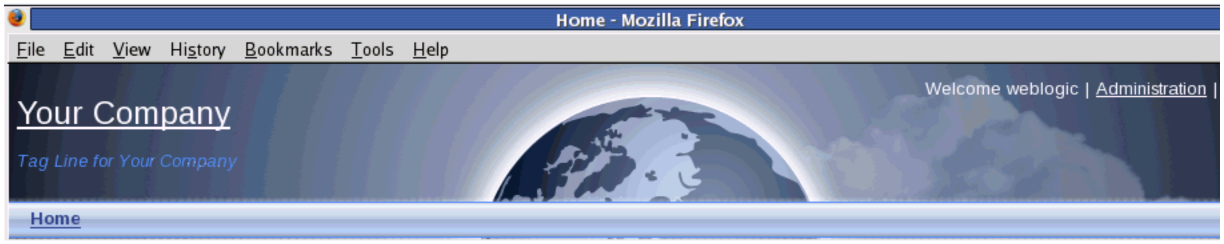
**Note:** You can also change the default page template by modifying the `adf-config.xml` file, as you did with the skin, described in Section , "Step 1: Change the Default Settings For Template and Skin."

---

To change the default page template from `Globe` to `Swooshy`:

1. In the Application Navigator in JDeveloper, right-click the Portal project and choose **Run** to run the application. The application opens in your web browser, as shown in Figure 5–4.

***Figure 5–4 The Home Portal Page with Administration Privileges Enabled***


Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Welcome weblogic | Administration |
Your Company
Tag Line for Your Company
Home

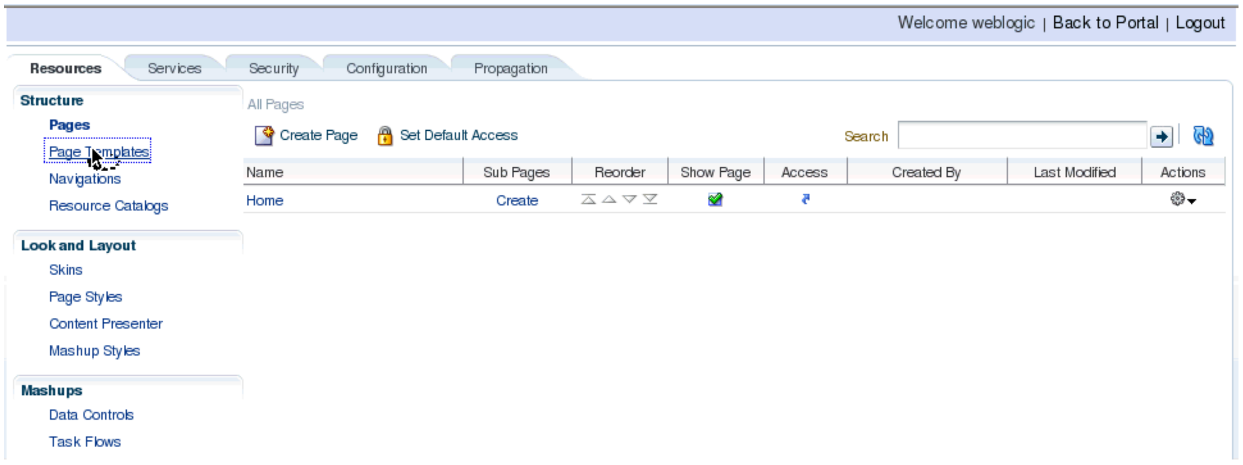
2. In the default `Home` portal page in the web browser, log in as `weblogic` and `weblogic1` as your password.

   Note that you must log in as a user with administrative privileges. In the Tutorial, the user "weblogic" has administrative privileges.

3. Click the **Administration** link in the upper right corner of the web page. The Administration Console appears (Figure 5–5).

4. Select the **Resources** tab and navigate to the Page Templates item in the **Structure** menu, as shown in Figure 5–5. Note that the **Propagation** tab will only appear if you have defined the appropriate connection for propagating from stage to production.

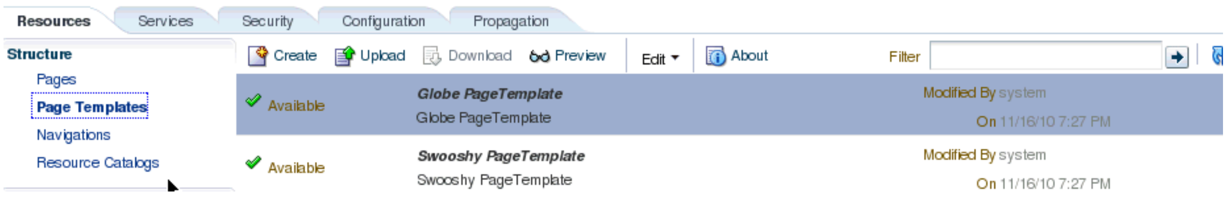
*Figure 5–5 Administration Console with the Resource Tab and Page Templates Item Selected*



| Name | Sub Pages | Reorder | Show Page | Access | Created By | Last Modified | Actions |
|---|---|---|---|---|---|---|---|
| Home | Create | | ✓ | | | | |

5. Click the **Page Templates** item. Both the `Globe PageTemplate` and the `Swooshy PageTemplate` appear, as shown in Figure 5–6, with the `Globe PageTemplate` field highlighted. Note that both templates are marked as available, so you can apply them.

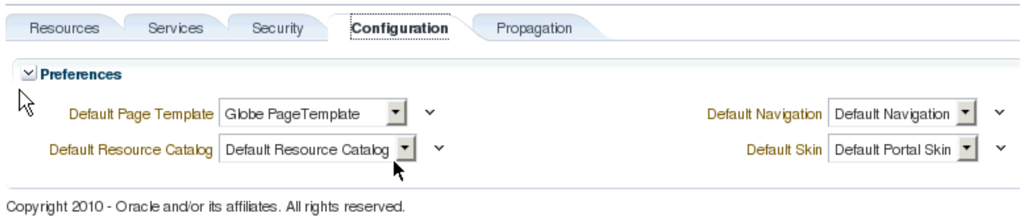
*Figure 5–6 The Globe PageTemplate Selected in the Page Templates Item*



| | | |
|---|---|---|
| ✓ Available | *Globe PageTemplate*<br>Globe PageTemplate | Modified By system<br>On 11/16/10 7:27 PM |
| ✓ Available | *Swooshy PageTemplate*<br>Swooshy PageTemplate | Modified By system<br>On 11/16/10 7:27 PM |

6. Select the **Configuration** tab in the Administration Console. In the **Preferences** menu, the `Default Page Template` is specified as the `Globe PageTemplate` (Figure 5–7).

*Figure 5–7 The Globe Template Specified as the Default Page Template in Preferences*

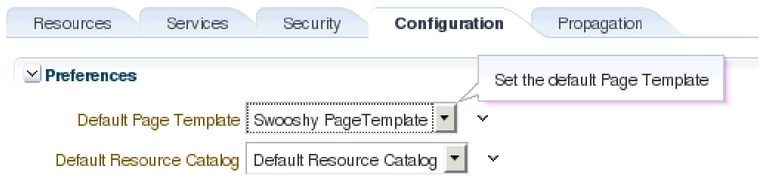


Default Page Template: Globe PageTemplate
Default Resource Catalog: Default Resource Catalog
Default Navigation: Default Navigation
Default Skin: Default Portal Skin

Copyright 2010 - Oracle and/or its affiliates. All rights reserved.

7. Change the `Default Page Template` and set it to the `Swooshy PageTemplate`, as shown in Figure 5–8.

*Figure 5–8 The Default Page Template Changed to Swooshy Page Template*



Default Page Template: Swooshy PageTemplate

Default Resource Catalog: Default Resource Catalog

8. Click the **Back to Portal** link. The new page template is applied to the portal.

In this lesson, you've learned how to change the default settings for both your template and skin at design time in JDeveloper by changing their preferences and updating portal resources. With those changes in effect, your portal application will have a different look and feel.

You also learned how to change the default page template at runtime from Globe to Swooshy by accessing the Administration Console and modifying preferences.

# 6 Connecting to and Managing Content Repositories

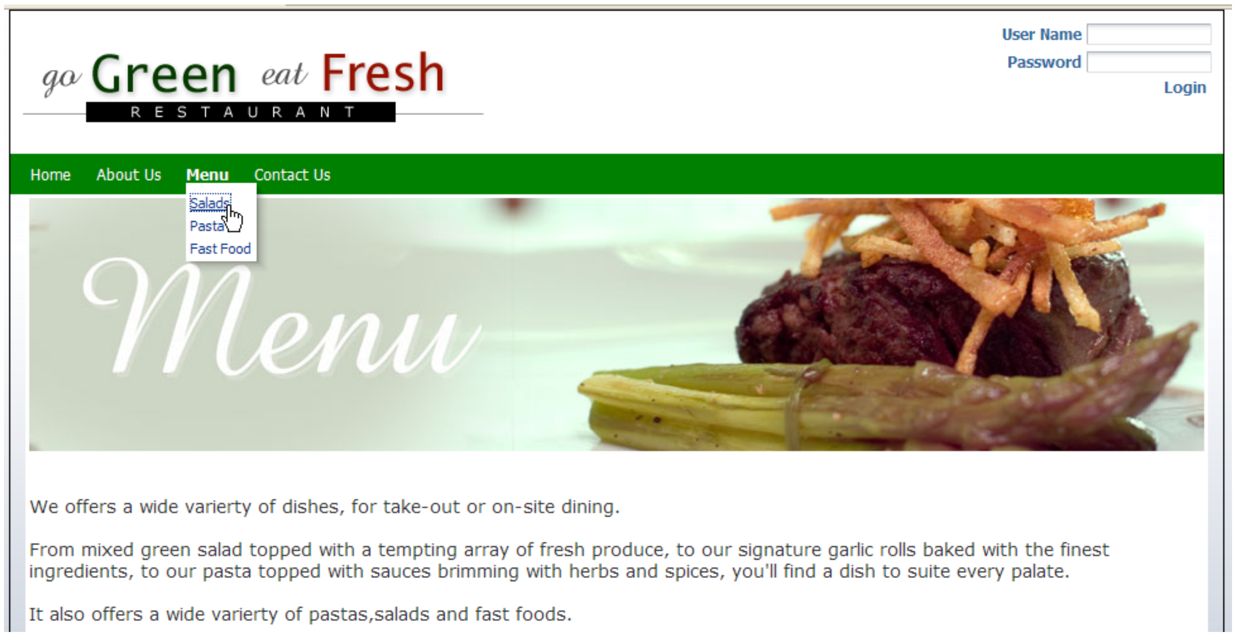
In this lesson, you will create a content repository connection that is owned and deployed by your WebCenter Portal application. In this case, the connection will be to the Oracle WebCenter Content repository with access provided to the Oracle WebCenter Portal's Content Service for Oracle WebCenter Content. You will set Oracle WebCenter Content as your primary connection and navigate to the Contributions directory, where HTML content files for your application, like **About Us**, **Contact Us**, **Home** and **Menu**, are stored in content sub folders.

You will then work with these files and the Documents - Content Presenter service to create task flow bindings for the application. For example, in the `home.jspx` file, you will drag and drop the `home.html` file as your Content Presenter. By enabling a connection to the Oracle WebCenter Content repository, you will be able to manage more efficiently the content you need while optimizing the development of your application.

The other tasks described in this lesson include learning how to add a content item to the default navigation model, as well as how to take advantage of Iterative Development, which allows you to make changes to your application while it is still running on the Integrated WebLogic Server and immediately see the effects of those changes when you refresh the pages in your web browser. You will also learn how to add a new Content Query that will fetch all the documents you need in your portal application that are based on specified metadata field tags.

At the end of this lesson, the page you created in the previous lesson will look like Figure 6–1.

*Figure 6–1 The MyPortalApplication in a Web Browser with Menu Items Selected*



We offers a wide varierty of dishes, for take-out or on-site dining.

From mixed green salad topped with a tempting array of fresh produce, to our signature garlic rolls baked with the finest ingredients, to our pasta topped with sauces brimming with herbs and spices, you'll find a dish to suite every palate.

It also offers a wide varierty of pastas,salads and fast foods.

For more information about adding content items to the navigation model and new content queries, see "Building a Navigation Model for Your Portal" in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*. For more information about managing content repositories and UCM, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*.

## Introduction

This lesson contains the following steps:

- Step 1: Connect to Oracle WebCenter Content Repository
- Step 2: Create a User With Privileges To Edit Repository Content
- Step 3: Add a Content Item to the Navigation Model
- Step 4: Take Advantage of Iterative Development
- Step 5: Add a New Content Query

Before you begin the steps in this lesson, ensure you have followed the steps up to this point in the Tutorial.

## Step 1: Connect to Oracle WebCenter Content Repository

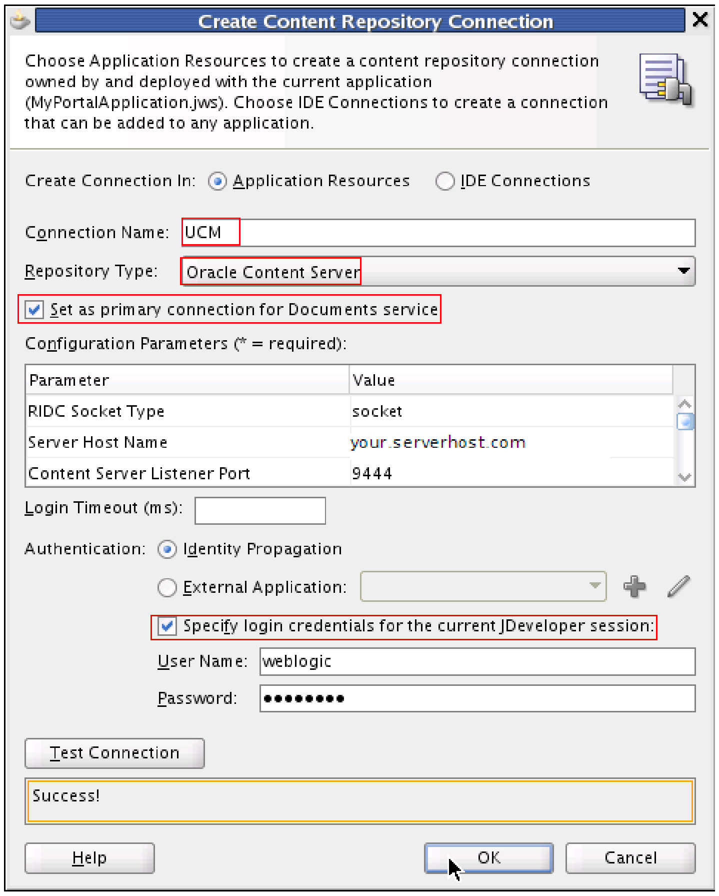
Oracle JDeveloper enables you to manage and handle document content stored on the Oracle WebCenter Portal's Content Service for Oracle WebCenter Content by creating a connection to the content repository, in this case to Oracle WebCenter Content. This connection is then owned and deployed by your portal application. You create this connection in **Application Resources**, as described in the following steps.

To connect to the content repository:

1. In Application Navigator, navigate down to Application Resources and right-click the `Connections` folder.

2. Choose **New Connection** and the **Content Repository** item. The Create Content Repository Connection dialog appears, as shown in Figure 6–2.

***Figure 6–2 The Create Content Repository Connection Dialog***


Create Content Repository Connection

Choose Application Resources to create a content repository connection owned by and deployed with the current application (MyPortalApplication.jws). Choose IDE Connections to create a connection that can be added to any application.

Create Connection In: [✓ Application Resources] IDE Connections

Connection Name: UCM

Repository Type: Oracle Content Server

✓ Set as primary connection for Documents service

Configuration Parameters (* = required):

| Parameter | Value |
|---|---|
| RIDC Socket Type | socket |
| Server Host Name | your.serverhost.com |
| Content Server Listener Port | 9444 |

Login Timeout (ms):

Authentication: [✓ Identity Propagation] External Application:

✓ Specify login credentials for the current JDeveloper session:

User Name: weblogic

Password: ●●●●●●●●

Test Connection

Success!

Help OK Cancel

3. In the Connection Name field, enter `UCM`. Ensure that you enter `UCM` in all caps. Note that you should *not* use any other designation for the Oracle WebCenter Content connection, like `myUCM`, for example. The Oracle WebCenter Content content for the Tutorial relies on this specific connection name. Ensure that the connection name is `UCM`.

4. From the Repository Type field, select `Oracle Content Server`. Ensure that you check **Set as primary connection for Documents service**.

5. In the Configuration Parameters pane (Figure 6–2), enter the parameters and values shown in Table 6–1. The Server Host Name should be your server host, that is, the host name of the system where your Oracle Content Server is running. The Content Server Listener Port is the port of your Oracle Server.

> **See Also:** For detailed information, see the section, "Configuring Oracle WebCenter Content Server Repositories," in particular, "Configuration Prerequisites" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

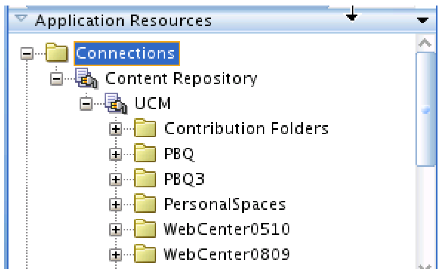
*Table 6–1 Configuration Parameters and Values*

| Parameter | Value |
|---|---|
| RIDC Socket Type | socket |
| Server Host Name | `mycontentserver.serverhost.com` |
| Content Server Listener Port | The port of your Oracle Server. Default is `4444`. |

6. Ensure that you check **Specify login credentials for the current JDeveloper session** (Figure 6–2).
7. In the User Name field, enter `weblogic` and in the Password field, enter weblogic1.
8. Click **Test Connection**, and if successful, click **OK**.
9. When you create a connection to a repository, the contents in the main directory of the repository display under the Content Repository connection in the Application Resources panel, as shown in Figure 6–3. You can double-click folders and files to view them.

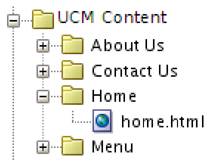
   Expand the `Connections` folder.

*Figure 6–3 The Connection Established to the Content Repository*



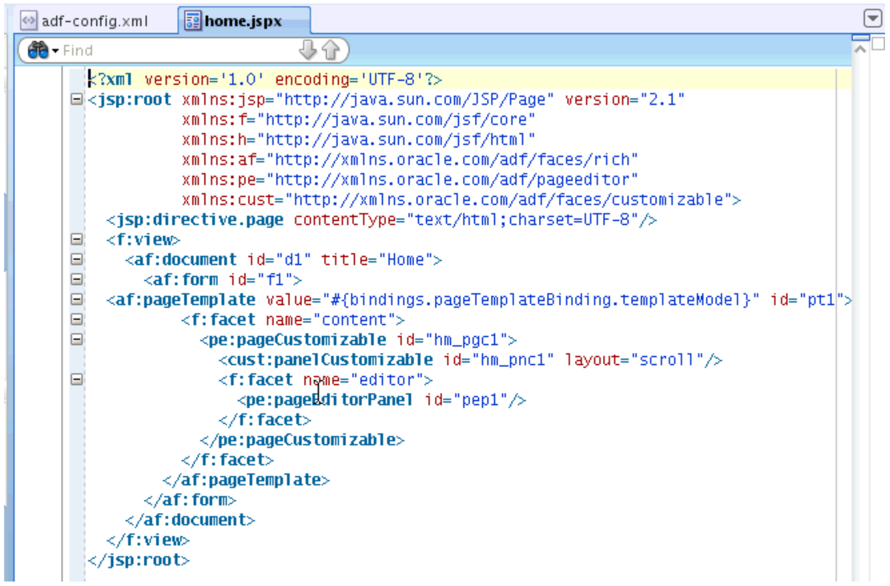
10. Under your UCM connection, expand the `Contributions` folder and navigate to the `UCM Content` folder, where your uploaded content is stored. (This was set up in "Step 6: Download the Sample Tutorial Files".) Now open the `Home` folder with the `home.html` file shown in Figure 6–4.

*Figure 6–4 The Home Folder Opened with the home.html file*



11. Return to your portal project directory, navigate up to the `pages` folder. Open the `home.jspx` file, then view it in **Source** view by clicking the **Source** tab. The `home.jspx` file appears in **Source** view, as shown in Figure 6–5.

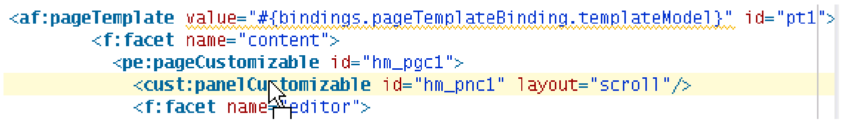
*Figure 6–5 The home.jspx File in Source View*



```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1" title="Home">
      <af:form id="f1">
  <af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}" id="pt1">
          <f:facet name="content">
            <pe:pageCustomizable id="hm_pgc1">
              <cust:panelCustomizable id="hm_pnc1" layout="scroll"/>
              <f:facet name="editor">
                <pe:pageEditorPanel id="pep1"/>
              </f:facet>
            </pe:pageCustomizable>
          </f:facet>
        </af:pageTemplate>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

12. Select the XML code snippet `<cust:panelCustomizable id="hmpnc1" layout="scroll"/>` at the center of the file, as shown in Figure 6–6. You will add another line of code after the selected snippet.

*Figure 6–6 The Customizable Panel XML Code Snippet*

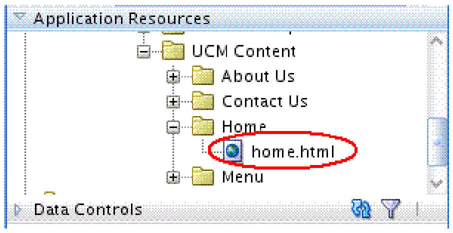


```
<af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}" id="pt1">
        <f:facet name="content">
          <pe:pageCustomizable id="hm_pgc1">
            <cust:panelCustomizable id="hm_pnc1" layout="scroll"/>
            <f:facet name="editor">
```

*******************************************************************************************

13. In the `UCM Content` folder, open the `Home` folder and select the `home.html` file, shown in Figure 6–7.

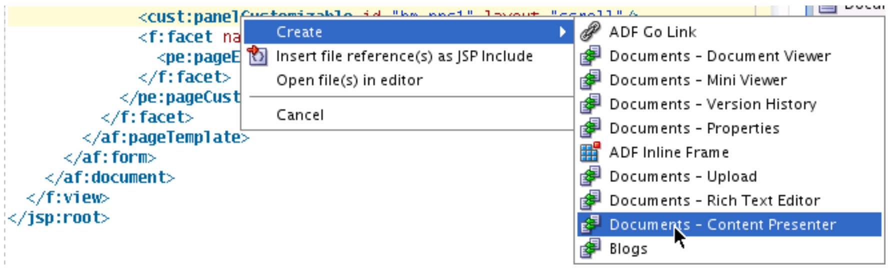
*Figure 6–7 The home.html File in the UCM Content Folder*



14. In JDeveloper, return to the **Source** view of the `home.jspx` file. Select the XML code snippet `<cust:panelCustomizable id="hmpnc1" layout="scroll"/>` shown in Figure 6–6, and drag and drop the `home.html` from the connection on to the `panelCustomizable`.

15. Choose the **Create** menu item, then scroll down the sub menu list to select the **Documents - Content Presenter** item, as shown in Figure 6–8, and click the item.

***Figure 6–8 The Create Documents - Content Presenter Menu Item Selected***

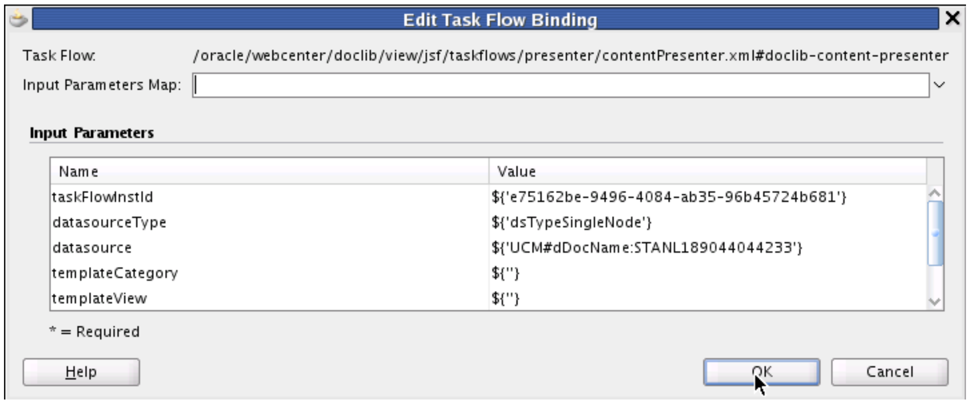


```
<cust:panelCustomizable id="hm_pnc1" layout="scroll">
  <f:facet na
    <pe:pageE
  </f:facet>
</pe:pageCust
</f:facet>
</af:pageTemplate>
</af:form>
</af:document>
</f:view>
</jsp:root>
```

Create
Insert file reference(s) as JSP Include
Open file(s) in editor
Cancel
ADF Go Link
Documents - Document Viewer
Documents - Mini Viewer
Documents - Version History
Documents - Properties
ADF Inline Frame
Documents - Upload
Documents - Rich Text Editor
Documents - Content Presenter
Blogs

**16.** The Edit Task Flow Binding dialog now appears, shown in Figure 6–9. The task flow input parameters are automatically assigned specific values. Click **OK**.

***Figure 6–9 The Edit Task Flow Binding Dialog with Specific Values Assigned***



Task Flow: /oracle/webcenter/doclib/view/jsf/taskflows/presenter/contentPresenter.xml#doclib-content-presenter

Input Parameters Map:

Input Parameters

| Name | Value |
|---|---|
| taskFlowInstId | ${'e75162be-9496-4084-ab35-96b45724b681'} |
| datasourceType | ${'dsTypeSingleNode'} |
| datasource | ${'UCM#dDocName:STANL189044044233'} |
| templateCategory | ${''} |
| templateView | ${''} |

* = Required

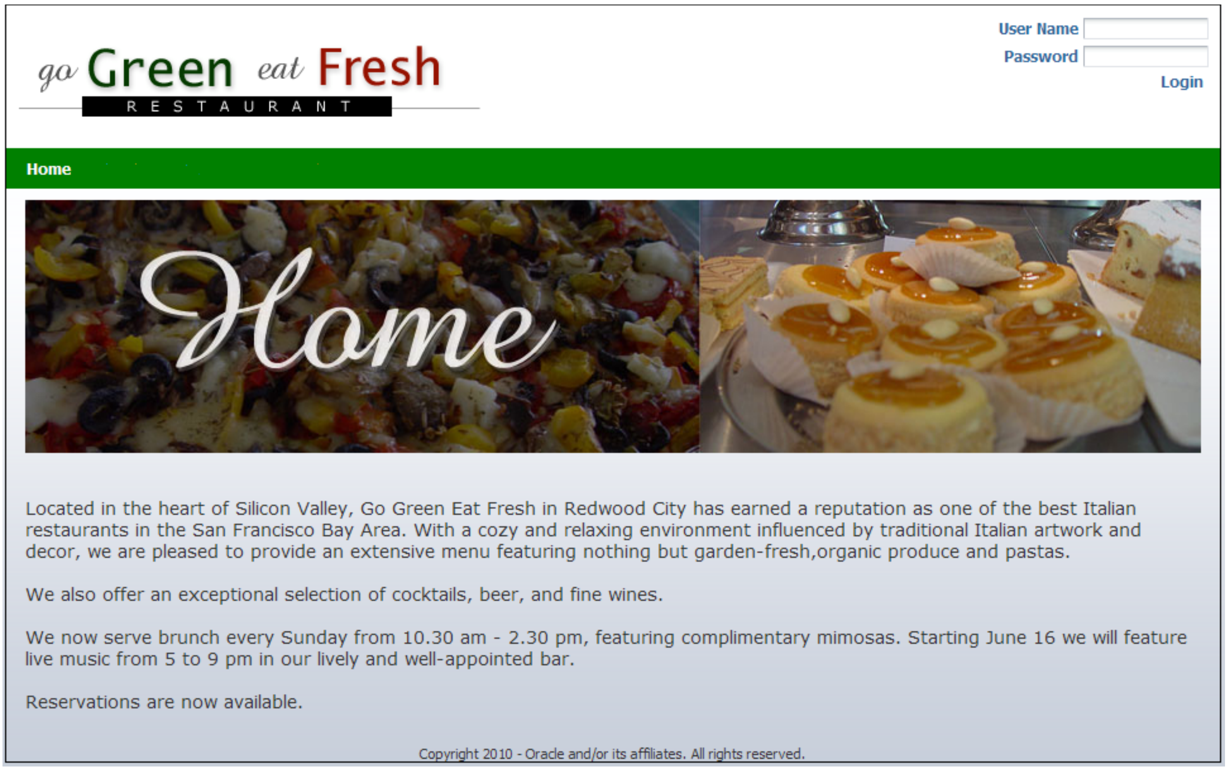
It's important to understand that in the last two steps, you have added a Content Presenter task flow and set the task flow parameters to read the values from the navigation link parameters. When you run your portal application, the **Home** node will automatically appear in the navigation menu because the `home.jspx` file has already been added to the page hierarchy and the page hierarchy has been added to the default navigation model.

**17.** Save your changes.

**18.** Refresh the web browser.

The new **Home** link appears in the navigation, with the text for document content at the center of the Home page, as shown in Figure 6–10.

***Figure 6–10 The MyPortalApplication in a Web Browser with the Home Page and Home Content Defined***



Located in the heart of Silicon Valley, Go Green Eat Fresh in Redwood City has earned a reputation as one of the best Italian restaurants in the San Francisco Bay Area. With a cozy and relaxing environment influenced by traditional Italian artwork and decor, we are pleased to provide an extensive menu featuring nothing but garden-fresh,organic produce and pastas.

We also offer an exceptional selection of cocktails, beer, and fine wines.

We now serve brunch every Sunday from 10.30 am - 2.30 pm, featuring complimentary mimosas. Starting June 16 we will feature live music from 5 to 9 pm in our lively and well-appointed bar.

Reservations are now available.

Copyright 2010 - Oracle and/or its affiliates. All rights reserved.

In this sequence of steps, you have learned how you can take advantage of the Content Presenter task flow in order to display and render documents under a UCM Content folder as a tab.

Using the Content Presenter task flow, you are able to drag and drop a task flow onto a panel component as a region in the XML code for purposes of binding the task flow to that region. You can then set task flow parameters to read the values from the navigation link parameters.

Up to this point, you have created and built a page that can be used as a template, if you choose, onto which you can add multiple navigational links.

## Step 2: Create a User With Privileges To Edit Repository Content

Now that you have successfully created a connection to the Oracle WebCenter Content repository for your WebCenter Portal application and performed the necessary steps to drag and drop the `home.html` file as your Content Presenter, we will create a user in the portal application who will have privileges to edit the content stored in your repository.

There are two steps involved to accomplish this particular task:

- Create a user (in this case, named `contentadmin`) in the application
- Create a user with the same name (`contentadmin`) in the Embedded LDAP WebLogic Server and add that user as a part of the Administrators group

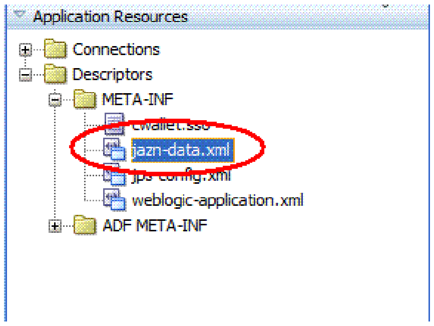
When you run the application after completing these steps and log in as `contentadmin`, the identity of this user is propagated to the Embedded LDAP of Oracle WebLogic Server.

Since the UCM instance uses the same Embedded LDAP as an identity store, and since all the users who are a part of the Administrators users group have by default Read, Write, Delete and Administration privileges, the user you've added -- `contentadmin` -- will be able to edit content.

To create a user in the portal application with privileges to edit content stored in your repository:

1. In Application Navigator, navigate down to Application Resources, expand the `META-INF` folder and open the `jazn-data.xml` file, shown in Figure 6–11.

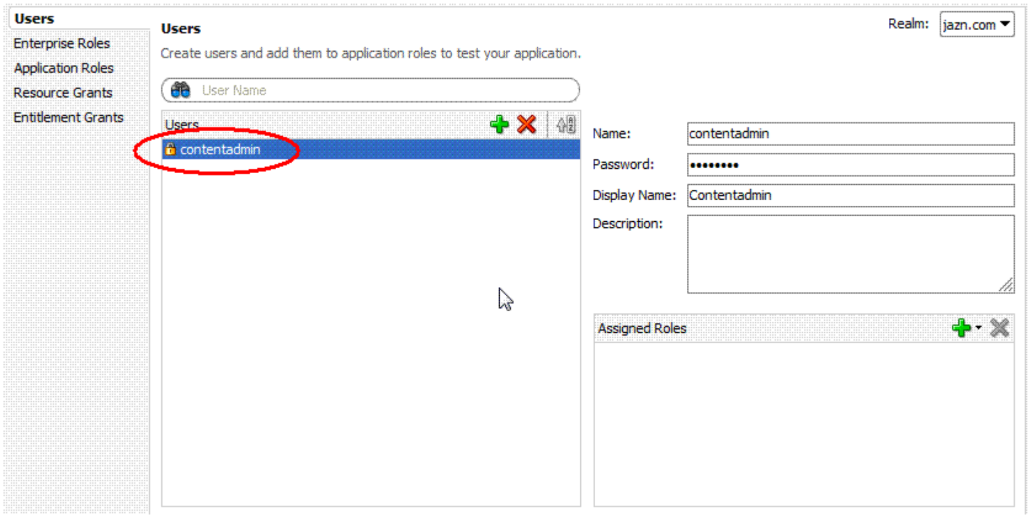
***Figure 6–11 The jazn-data.xml File Selected in the META-INF Folder***



2. Create a user named `contentadmin` in your portal application (Figure 6–12).

   Once you have created a user in the application, you will create a user on the embedded LDAP of Oracle WebLogic Server.

***Figure 6–12 The contentadmin Created as a User In Oracle WebCenter Content with Name and Password***

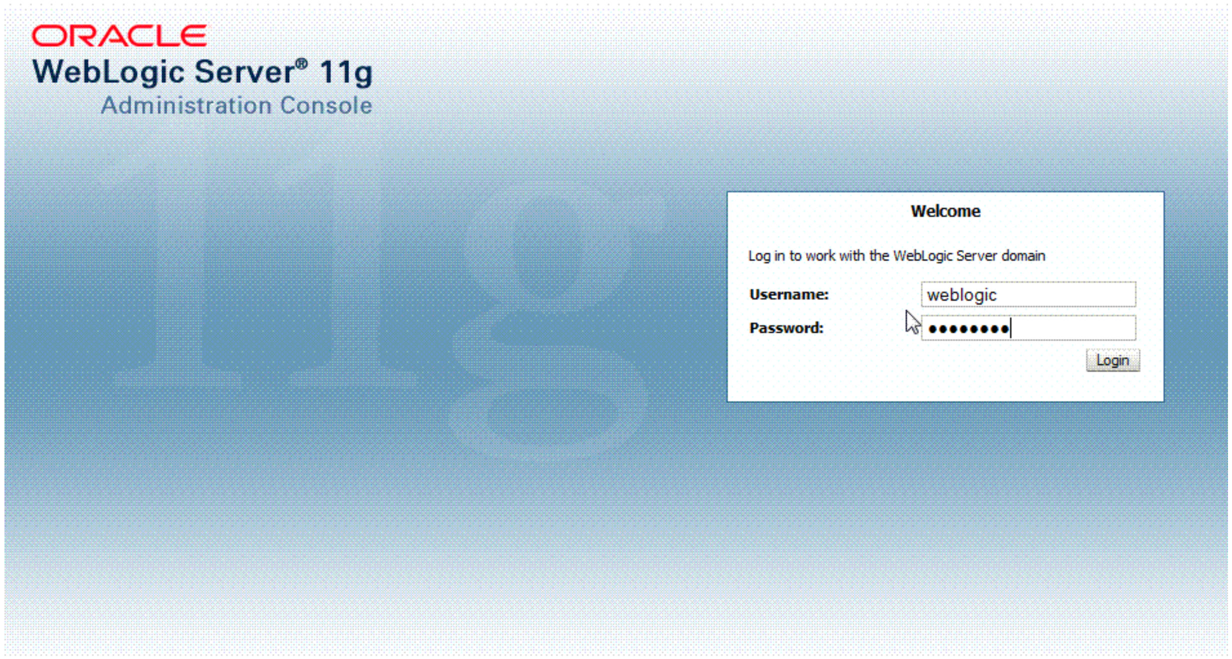


Create users and add them to application roles to test your application.

| | |
|---|---|
| Name: | contentadmin |
| Password: | •••••••• |
| Display Name: | Contentadmin |
| Description: | |

3. Log in to the Oracle WebLogic Server Administration Console where the Oracle WebCenter Content server is installed and enter `weblogic` as the user name and the password you used when installing the product (Figure 6–13).

***Figure 6–13 Logging in to the Administration Console for Oracle WebLogic Server***



Welcome

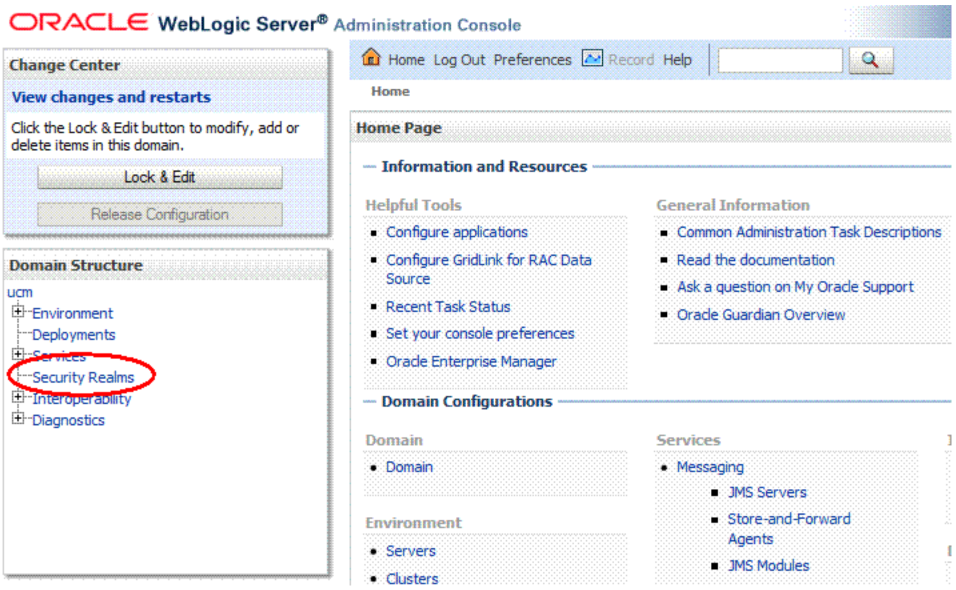
Log in to work with the WebLogic Server domain

**Username:** weblogic
**Password:** ••••••••

Login

4. Select the Security Realms item in the Oracle WebCenter Content Domain Structure pane (Figure 6–14) and click it.

***Figure 6–14 The Security Realms Item in Oracle WebCenter Content***


ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help

Home

**Change Center**

**View changes and restarts**

Click the Lock & Edit button to modify, add or delete items in this domain.

Lock & Edit

Release Configuration

**Domain Structure**

ucm
- Environment
- Deployments
- Services
- Security Realms
- Interoperability
- Diagnostics

**Home Page**

**Information and Resources**

Helpful Tools
- Configure applications
- Configure GridLink for RAC Data Source
- Recent Task Status
- Set your console preferences
- Oracle Enterprise Manager

General Information
- Common Administration Task Descriptions
- Read the documentation
- Ask a question on My Oracle Support
- Oracle Guardian Overview

**Domain Configurations**

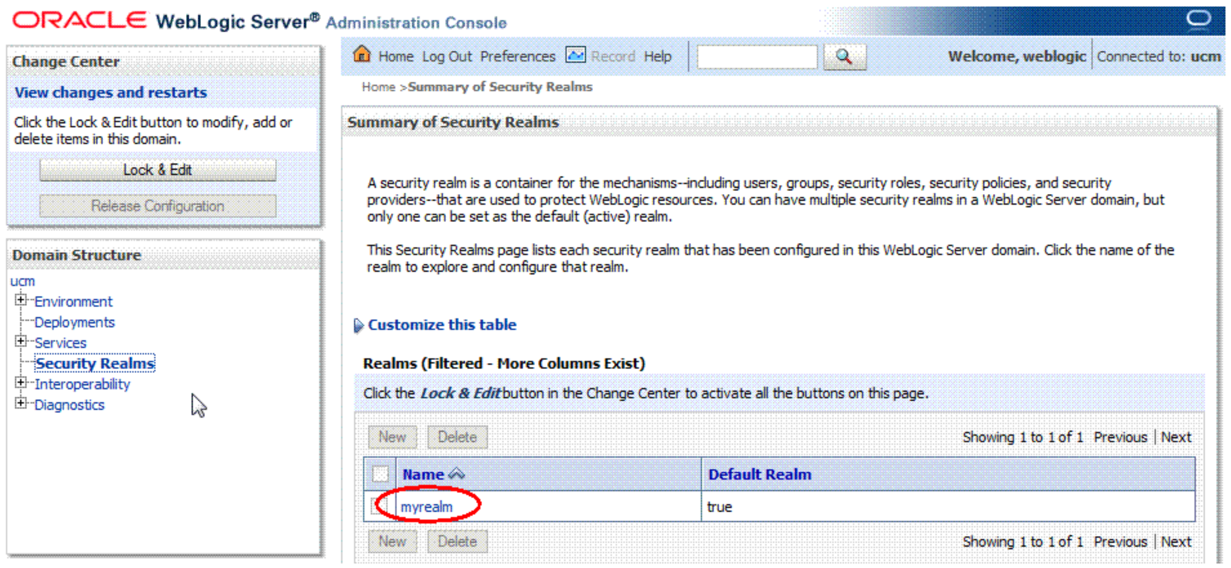
Domain
- Domain

Environment
- Servers
- Clusters

Services
- Messaging
  - JMS Servers
  - Store-and-Forward Agents
  - JMS Modules

5. Now select the `myrealm` name in the **Realms** table (Figure 6–15) and click it.

***Figure 6–15 The Name myrealm Selected in the Realms Table***


ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: ucm
Home >Summary of Security Realms
Change Center
View changes and restarts
Lock & Edit
Release Configuration
Domain Structure
ucm
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
Summary of Security Realms
Customize this table
New
Delete
Showing 1 to 1 of 1 Previous | Next

Click the Lock & Edit button to modify, add or delete items in this domain.

A security realm is a container for the mechanisms--including users, groups, security roles, security policies, and security providers--that are used to protect WebLogic resources. You can have multiple security realms in a WebLogic Server domain, but only one can be set as the default (active) realm.

This Security Realms page lists each security realm that has been configured in this WebLogic Server domain. Click the name of the realm to explore and configure that realm.

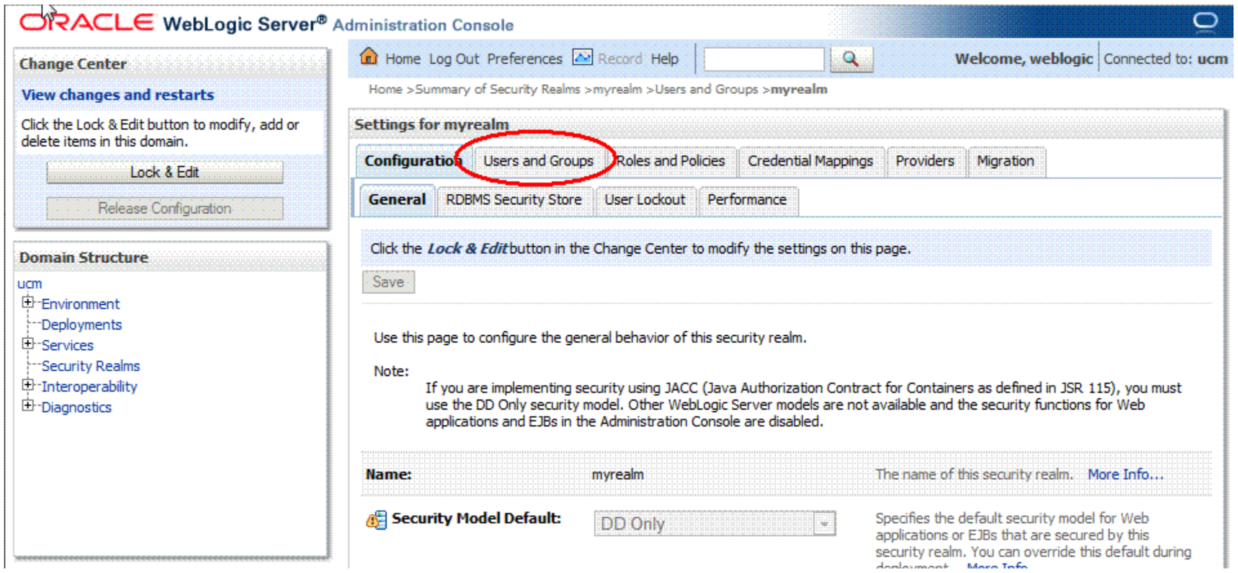
Realms (Filtered - More Columns Exist)

Click the *Lock & Edit* button in the Change Center to activate all the buttons on this page.

| Name | Default Realm |
|---|---|
| myrealm | true |

6. In the Settings for myrealm pane (Figure 6–16), select and then click the **Users and Groups** tab.

***Figure 6–16 Users and Groups Selected in the Settings for myrealm Pane***



Click the Lock & Edit button to modify, add or delete items in this domain.

Click the *Lock & Edit* button in the Change Center to modify the settings on this page.

Use this page to configure the general behavior of this security realm.

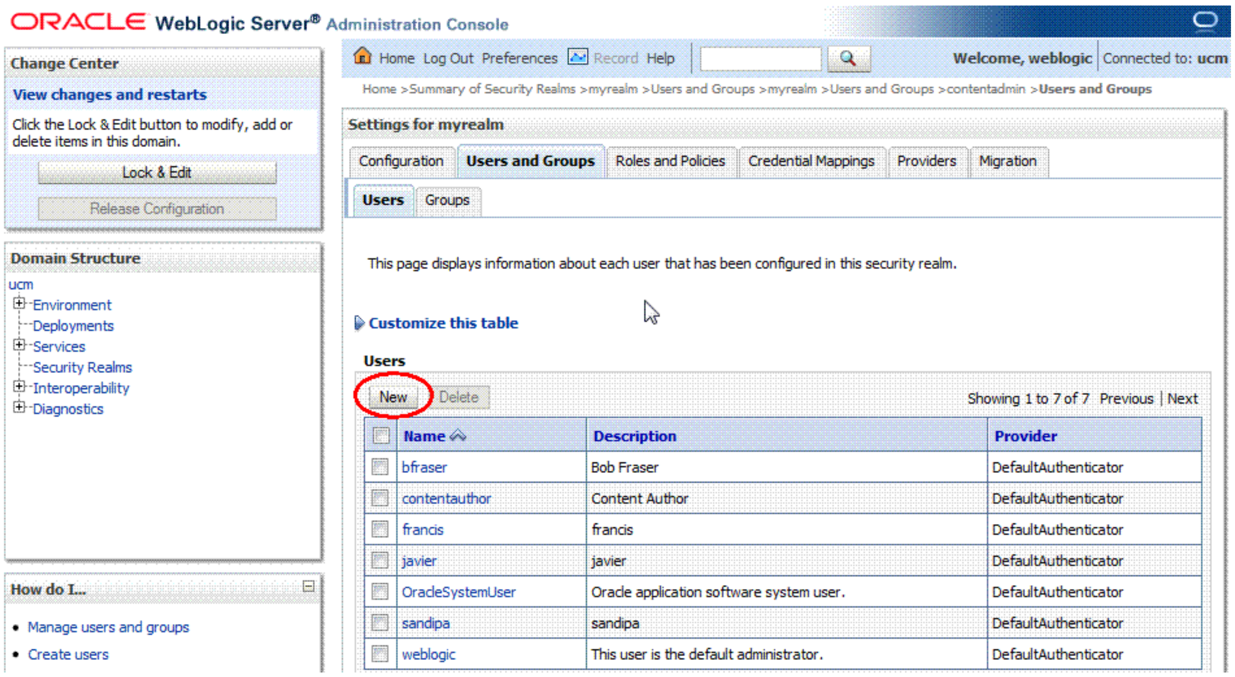
Note:
If you are implementing security using JACC (Java Authorization Contract for Containers as defined in JSR 115), you must use the DD Only security model. Other WebLogic Server models are not available and the security functions for Web applications and EJBs in the Administration Console are disabled.

Name: myrealm — The name of this security realm. More Info...

Security Model Default: DD Only — Specifies the default security model for Web applications or EJBs that are secured by this security realm. You can override this default during

7. In the **Users** table, click the **New** button (Figure 6–17).

***Figure 6–17 The New User Button Selected in the Users and Groups Pane***



Click the Lock & Edit button to modify, add or delete items in this domain.

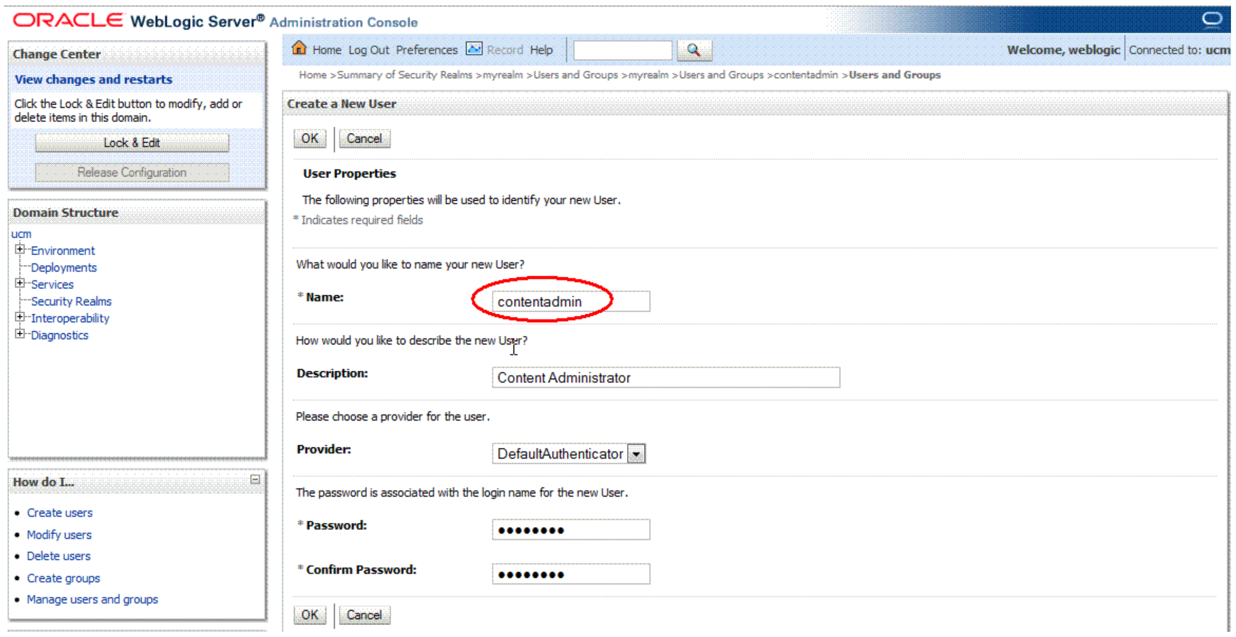
This page displays information about each user that has been configured in this security realm.

Users

| Name | Description | Provider |
|---|---|---|
| bfraser | Bob Fraser | DefaultAuthenticator |
| contentauthor | Content Author | DefaultAuthenticator |
| francis | francis | DefaultAuthenticator |
| javier | javier | DefaultAuthenticator |
| OracleSystemUser | Oracle application software system user. | DefaultAuthenticator |
| sandipa | sandipa | DefaultAuthenticator |
| weblogic | This user is the default administrator. | DefaultAuthenticator |

8. Now create a user named `contentadmin` when asked what you would like to name your new user (Figure 6–18). Specify this new user as a Content Administrator when asked how you would like to describe this new user.

***Figure 6–18 The User contentadmin Created as New User***


ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic Connected to: ucm
Home >Summary of Security Realms >myrealm >Users and Groups >myrealm >Users and Groups >contentadmin >Users and Groups
Change Center
View changes and restarts
Lock & Edit
Release Configuration
Domain Structure
ucm
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Create users
Modify users
Delete users
Create groups
Manage users and groups
Create a New User
OK
Cancel

Click the Lock & Edit button to modify, add or delete items in this domain.

User Properties

The following properties will be used to identify your new User.

* Indicates required fields

What would you like to name your new User?

* Name: contentadmin

How would you like to describe the new User?

Description: Content Administrator

Please choose a provider for the user.

Provider: DefaultAuthenticator

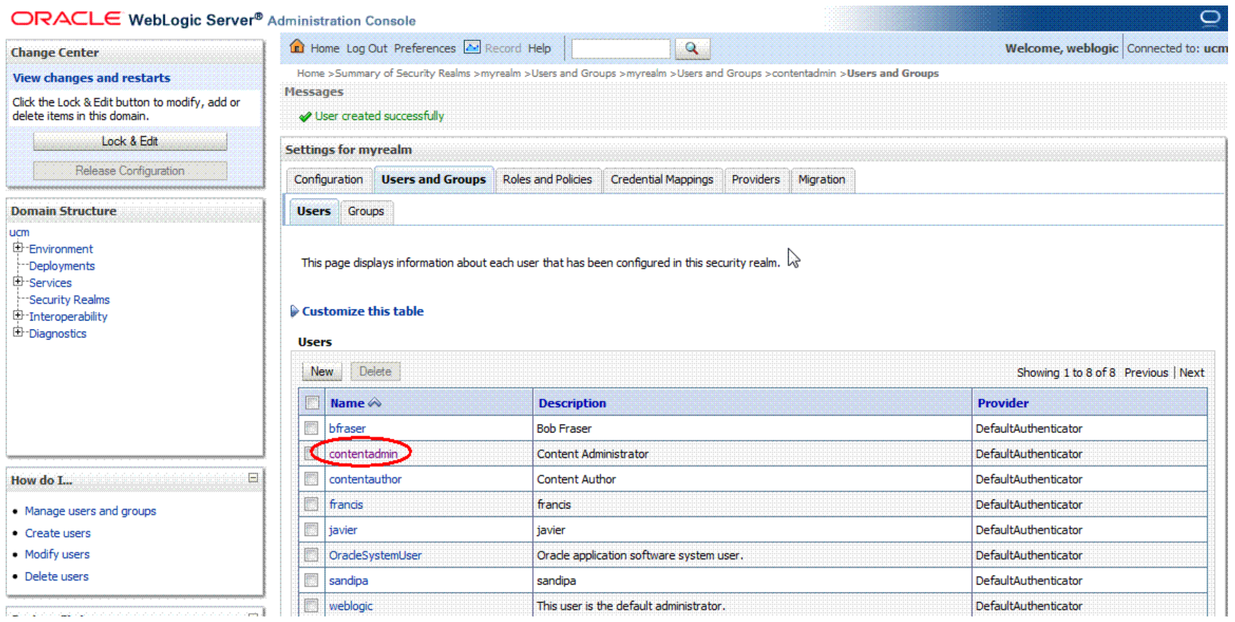
The password is associated with the login name for the new User.

* Password: ••••••••

* Confirm Password: ••••••••

9. Select and then click the newly created user in the **Users** table (Figure 6–19). Content Administrator appears in the **Description** field of the table. Note that the page displays information about each user that has been configured in this security realm.

***Figure 6–19 The Newly Created User contentadmin in the Users Table***



Home >Summary of Security Realms >myrealm >Users and Groups >myrealm >Users and Groups >contentadmin >**Users and Groups**

Messages

✓ User created successfully

Click the Lock & Edit button to modify, add or delete items in this domain.

**Settings for myrealm**

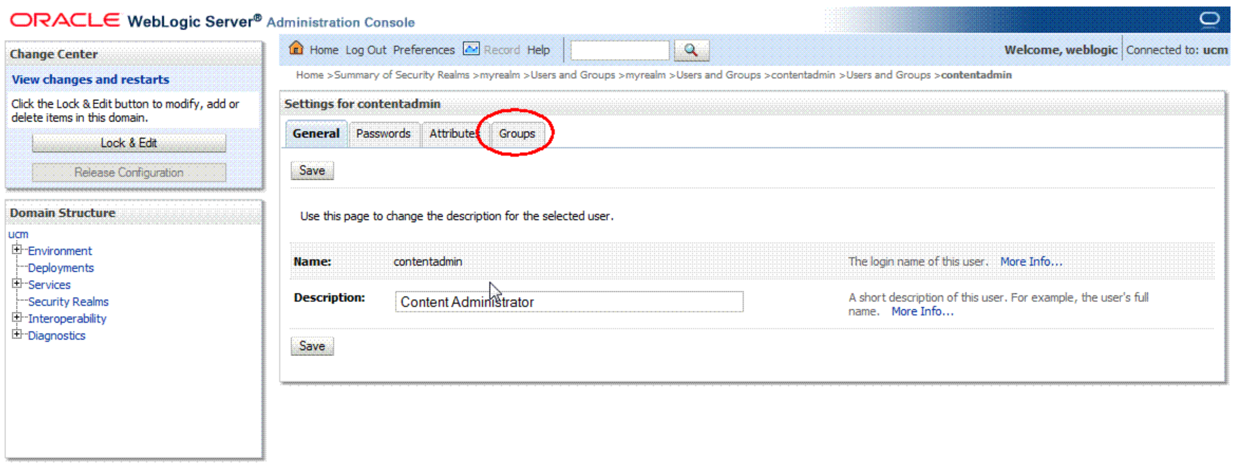
This page displays information about each user that has been configured in this security realm.

**Users**

| Name | Description | Provider |
|---|---|---|
| bfraser | Bob Fraser | DefaultAuthenticator |
| contentadmin | Content Administrator | DefaultAuthenticator |
| contentauthor | Content Author | DefaultAuthenticator |
| francis | francis | DefaultAuthenticator |
| javier | javier | DefaultAuthenticator |
| OracleSystemUser | Oracle application software system user. | DefaultAuthenticator |
| sandipa | sandipa | DefaultAuthenticator |
| weblogic | This user is the default administrator. | DefaultAuthenticator |

**10.** Select and click the **Groups** tab in the Settings for contentadmin pane (Figure 6–20).

***Figure 6–20 Groups Tab Selected in the Settings for contentadmin Pane***



Home >Summary of Security Realms >myrealm >Users and Groups >myrealm >Users and Groups >contentadmin >Users and Groups >**contentadmin**

Click the Lock & Edit button to modify, add or delete items in this domain.

**Settings for contentadmin**

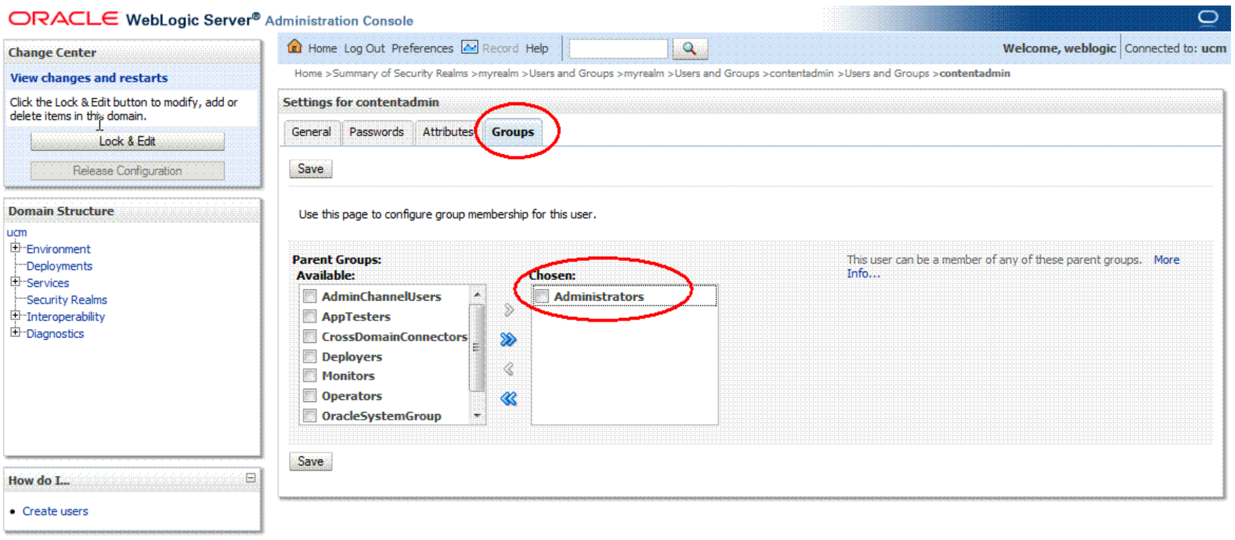
Use this page to change the description for the selected user.

**Name:** contentadmin — The login name of this user.

**Description:** Content Administrator — A short description of this user. For example, the user's full name.

**11.** Select the **Groups** tab and choose the Administrators group from the **Parent Groups: Available** list and add it to Groups (Figure 6–20). Note that you can use this page to configure group membership for a particular user.

*Figure 6–21 Administrators in Available Parent Groups Added to Groups*


ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic | Connected to: ucm
Home >Summary of Security Realms >myrealm >Users and Groups >myrealm >Users and Groups >contentadmin >Users and Groups >contentadmin
Change Center
View changes and restarts
Click the Lock & Edit button to modify, add or delete items in this domain.
Lock & Edit
Release Configuration
Domain Structure
ucm
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
How do I...
Create users
Settings for contentadmin
General
Passwords
Attributes
Groups
Save
Use this page to configure group membership for this user.
Parent Groups:
Available:
AdminChannelUsers
AppTesters
CrossDomainConnectors
Deployers
Monitors
Operators
OracleSystemGroup
Chosen:
Administrators
This user can be a member of any of these parent groups. More Info...
Save

12. Click **Save**.

    Once you've completed these steps, the user you've added -- `contentadmin` -- will now be able to edit the UCM content in the content repository.

## Step 3: Add a Content Item to the Navigation Model

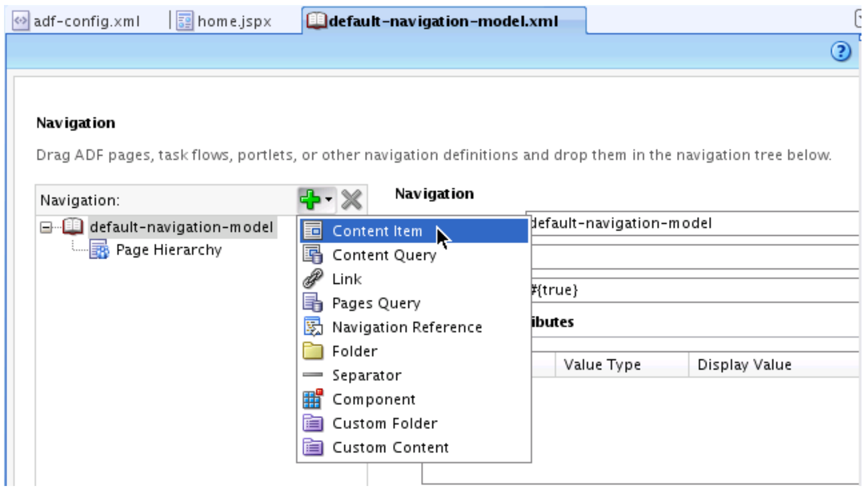
In this next sequence of steps, we move ahead to add a content item to the default navigation model XML file for the application.

To add a content item to the default navigation model:

1. In the Application Navigator of your portal application project, go to the navigations folder (`/oracle/webcenter/portalapp/navigations`). Right-click `default-navigation-model.xml`, and choose Open.

2. In the Design view for the navigation model, in the Navigation column on the left side, click the Add new node icon (the plus icon) and choose the **Content Item**, shown in Figure 6–22.

*Figure 6–22 Adding a Content Item to the Default Navigation Model*

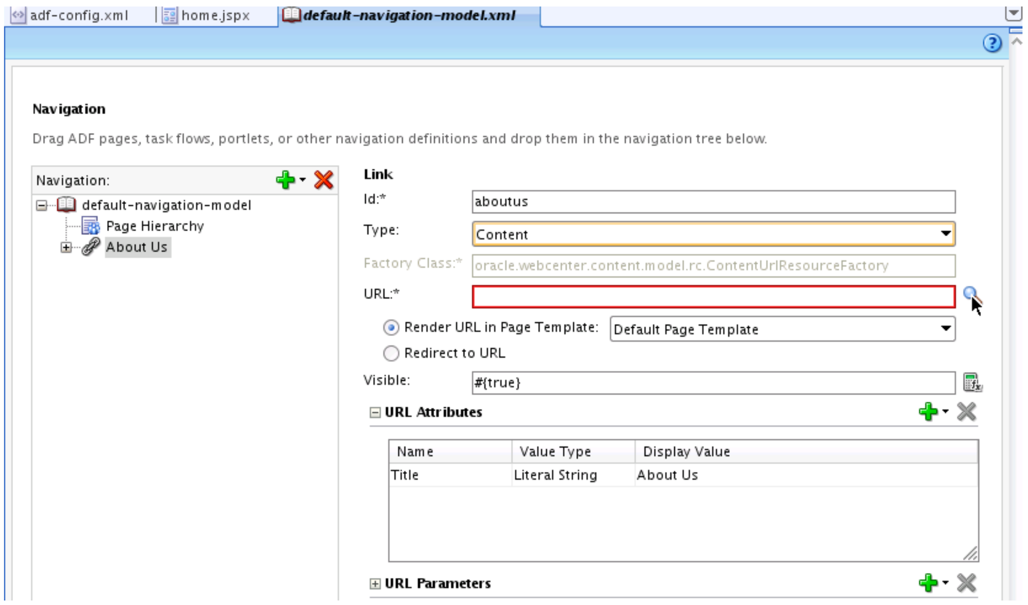


**Tip:** When you first create a WebCenter Portal application, the seeded navigation model, `default-navigation-model.xml`, is set as the default navigation model. The default navigation model provides a convenient way to select a navigation model that can be used by default by your application. Page template designers, for example, can then reference this default navigation model without having to know its actual name.

You can set the default navigation model for your application, if you want to create your own model, by editing the `oracle.webcenter.portalapp.navigation.model` preference in the `adf-config.xml` file.

3. The navigation window for the `default-navigation-model.xml` file appears with a link specified as `contentItem` and with the Id specified also as `contentItem`. You will need to change these entries, as shown in Figure 6–23. Note that the Type field is set as `Content`.
4. Select the **About Us** file, enter in the Link Id field: `aboutus`. In the URL Attributes pane, enter the Display Value as `About Us`.

***Figure 6–23 Id and Display Value Specified for Content Item***

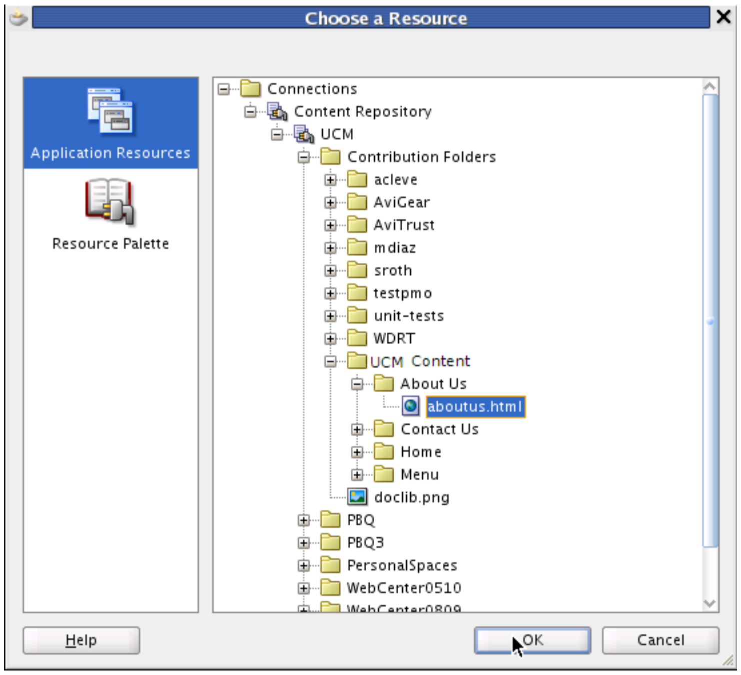

adf-config.xml
home.jspx
default-navigation-model.xml
Navigation
Drag ADF pages, task flows, portlets, or other navigation definitions and drop them in the navigation tree below.
Navigation:
default-navigation-model
Page Hierarchy
About Us
Link
URL Attributes
URL Parameters

| | |
|---|---|
| Id:* | aboutus |
| Type: | Content |
| Factory Class:* | oracle.webcenter.content.model.rc.ContentUrlResourceFactory |
| URL:* | |
| (● Render URL in Page Template) | Default Page Template |
| (○ Redirect to URL) | |
| Visible: | #{true} |

| Name | Value Type | Display Value |
|---|---|---|
| Title | Literal String | About Us |

5. Click the Browse icon (in the shape of a magnifying glass) next to the URL field to browse the contents.
6. In the Choose a Resource dialog, open and expand the UCM sub folders, then navigate to the `UCM Content` folder and select the `aboutus.html` file shown in Figure 6–24.

***Figure 6–24 The aboutus.html file Selected in the Choose a Resource Dialog***



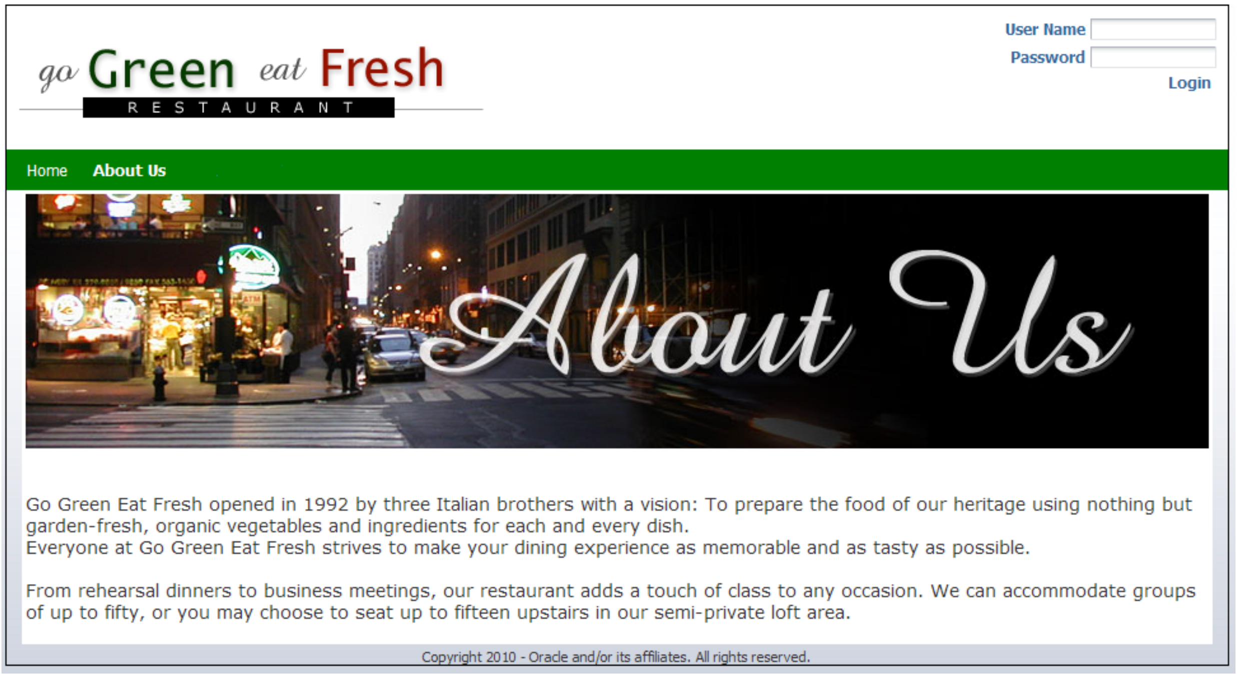
**7.** Click **OK**. The URL of the selected resource appears in the URL field.

Note that you can also simply drag and drop the file from the content connection into the navigation and it will create the content item for you.

**8.** Save your files.

**9.** Now return to your web browser, reload the browser page and log in as the weblogic user. The **About Us** link appears in the navigation model next to **Home**, as shown in Figure 6–25.

*Figure 6–25 The About Us Link in the Web Browser of Your Portal Application*



Go Green Eat Fresh opened in 1992 by three Italian brothers with a vision: To prepare the food of our heritage using nothing but garden-fresh, organic vegetables and ingredients for each and every dish.
Everyone at Go Green Eat Fresh strives to make your dining experience as memorable and as tasty as possible.

From rehearsal dinners to business meetings, our restaurant adds a touch of class to any occasion. We can accommodate groups of up to fifty, or you may choose to seat up to fifteen upstairs in our semi-private loft area.

Copyright 2010 - Oracle and/or its affiliates. All rights reserved.

## Step 4: Take Advantage of Iterative Development

The Iterative Development option is enabled by default in your WebCenter Portal application. There are several advantages to this option.

For one thing, iterative development lets you speed up your development process by allowing you to make changes to your application while it is still running on the Integrated WebLogic Server and then immediately see the effects of those changes when you refresh the current pages in your web browser.

> **Tip:** On a browser refresh, you will see changes almost instantly to page definitions and page hierarchy, existing JSPX files and the navigation model, page templates, the resource catalog and task flows or portlets you may have added to pages. For example, you can add a task flow to a page and simply refresh the browser or you can change the values in task flows and right away see the results.
>
> Other operations are not supported by iterative development, however, and require you to re-run the application if you create any new file explicitly (for example, a new page definition or page hierarchy), or implicitly. For instance, when you add a sub-page to a node in the page hierarchy, a new `*pages.xml` file is created, or if you edit any configuration file, like `web.xml` or `adfc-config.xml`.
>
> Basically, iterative development works by disabling certain optimization features. Note that iterative development only applies when running from JDeveloper using the built-in server. This option has no effect once your portal application is deployed to a staging or production server.

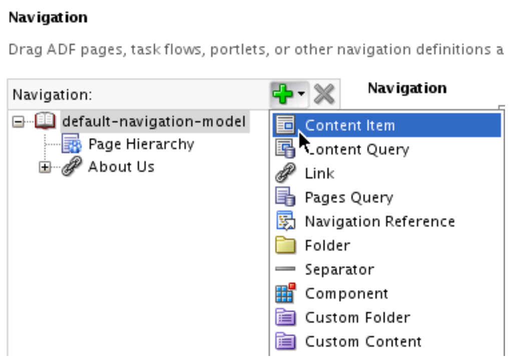
In the previous step Section , "Step 3: Add a Content Item to the Navigation Model", you added a new content link to your application, which appeared in the navigation hierarchy when you refreshed the contents of your web browser.

With Iterative Development enabled in your application, you will add a new link in JDeveloper of type `Content` in the default navigation model under Root, and then choose the `menu.html` document for that link. As you save the changes to your application and refresh the browser page, you will see the Menu node on the navigation menu. Clicking it will enable you to see the content of the Menu.

To add a new link in the default navigation model in your application:

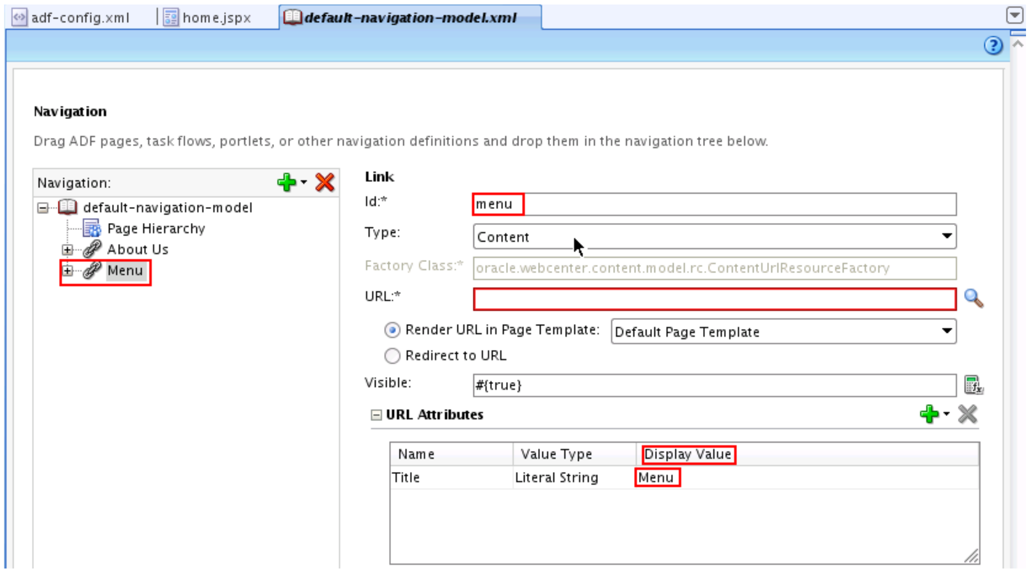
1. In Application Navigator, select the `default-navigation-model.xml` file. Right-click the **Application Properties** menu item.
2. In the Application Properties dialog, select WebCenter in the **Run** node and ensure that **Enable Iterative Development** is checked. (Note that this is checked by default.)
3. Click **OK**.
4. In the Navigation pane with the `default-navigation-model.xml` file selected, click the plus icon to add a Content Item in the navigation, as shown in Figure 6–26.

***Figure 6–26 The Content Item To Be Added to the Default Navigation Model XML File***



5. The `contentItem` appears as a node in the `default-navigation-model.xml` navigation and in the Id field as `contentItem`. Change the Id to `menu` and the Display Value to `Menu`, as shown in Figure 6–27.

***Figure 6–27 The contentItem Changed to menu with a New Display Value***



**Navigation**

Drag ADF pages, task flows, portlets, or other navigation definitions and drop them in the navigation tree below.

Navigation:
- default-navigation-model
  - Page Hierarchy
  - About Us
  - Menu

**Link**

| | |
|---|---|
| Id:* | menu |
| Type: | Content |
| Factory Class:* | oracle.webcenter.content.model.rc.ContentUrlResourceFactory |
| URL:* | |
| Render URL in Page Template: | Default Page Template |
| Redirect to URL | |
| Visible: | #{true} |

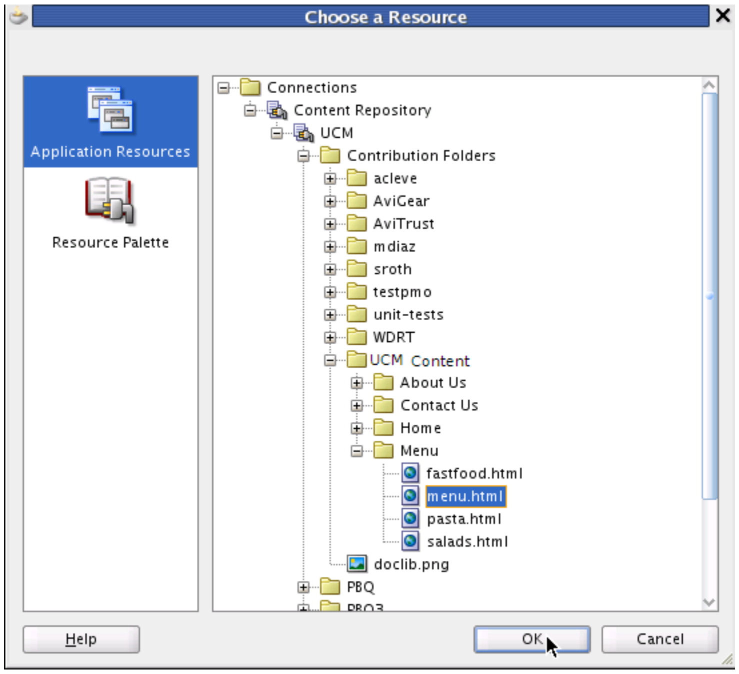
**URL Attributes**

| Name | Value Type | Display Value |
|---|---|---|
| Title | Literal String | Menu |

6. Click the Browse icon next to the URL field.

7. Navigate to the `Menu` folder in your UCM connection and select the `menu.html` file, as shown in Figure 6–28.

***Figure 6–28 The Choose a Resource Dialog with menu.html Selected***

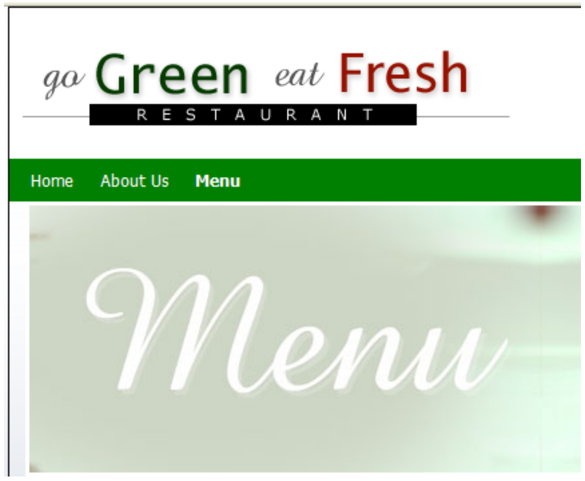


- Connections
  - Content Repository
    - UCM
      - Contribution Folders
        - acleve
        - AviGear
        - AviTrust
        - mdiaz
        - sroth
        - testpmo
        - unit-tests
        - WDRT
        - UCM Content
          - About Us
          - Contact Us
          - Home
          - Menu
            - fastfood.html
            - menu.html
            - pasta.html
            - salads.html
        - doclib.png
      - PBQ
      - PBQ3

8. Click **OK**.

9. Refresh the **Home** page in your web browser and note that the **Menu** link appears in the navigation of the **Home** page, as shown in a partial view in Figure 6–29. This is the result of enabling the Iterative Development feature in your portal application.

***Figure 6–29 The Refreshed Web Browser Page with the Menu Link Added in the Navigation Model***



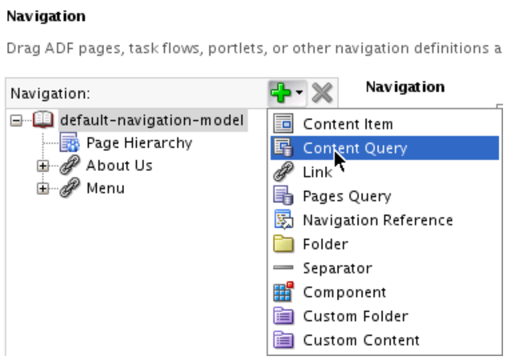
## Step 5: Add a New Content Query

In this next step, you will add a new content query to your portal application. This query will fetch all the documents that are based on metadata field tags in the Oracle WebCenter Content repository.

To add a new content query:

1. Return to your portal application in JDeveloper and open the `default-navigation-model.xml` file.
2. Add the new node icon by clicking the plus button.
3. Select the **Content Query** menu item in the list, as shown in Figure 6–30.

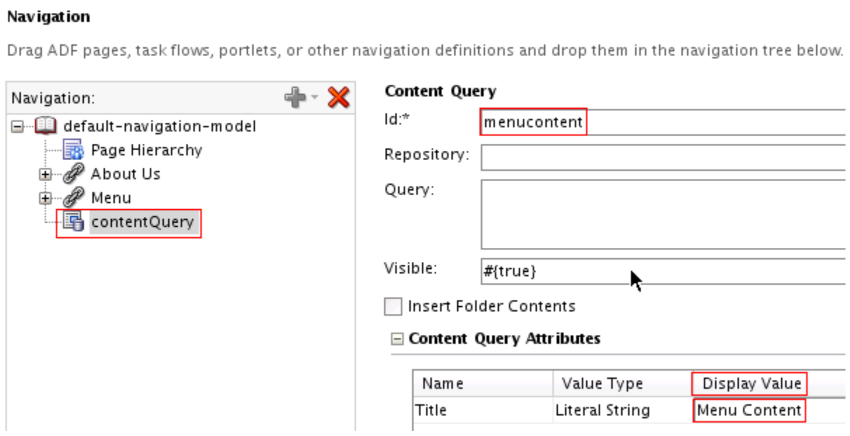
***Figure 6–30 Adding a New Content Query to the Default Navigation Model***



*************************************************************************************************

4. In the **Content Query** pane, enter in the Id field `menucontent`. In the **Content Query Attributes** pane, enter as the Display Value `Menu Content` (Figure 6–31).

*Figure 6–31 The Content Query Pane with Id and Display Value Changed*


Navigation

Drag ADF pages, task flows, portlets, or other navigation definitions and drop them in the navigation tree below.

Navigation:
- default-navigation-model
  - Page Hierarchy
  - About Us
  - Menu
  - contentQuery

Content Query

Id:* menucontent

Repository:

Query:

Visible: #{true}

Insert Folder Contents

Content Query Attributes

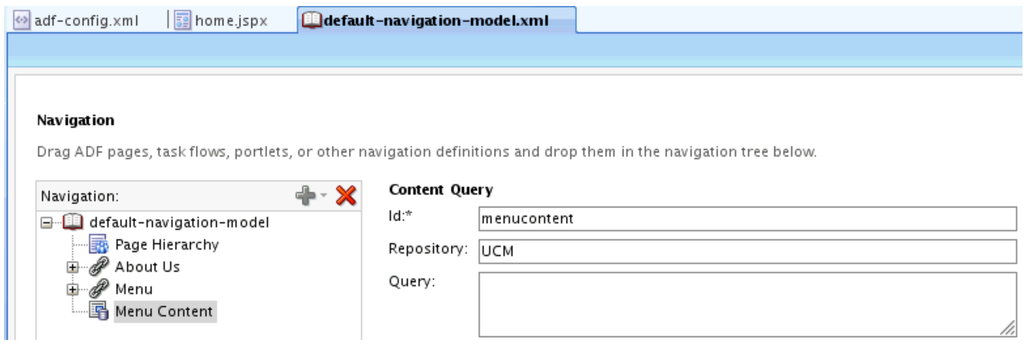
| Name | Value Type | Display Value |
|---|---|---|
| Title | Literal String | Menu Content |

**************************************************************************************************

5. Select **Menu Content** in the navigation node, and click the Browse icon in the **URL** field. The Choose a Resource dialog now appears.

6. In the Choose a Resource dialog, expand Connections, and select UCM, then click **OK**. Note that your connection named UCM appears in the **Repository** field, shown in Figure 6–32.

*Figure 6–32 The Repository Field with UCM Entered*



Navigation

Drag ADF pages, task flows, portlets, or other navigation definitions and drop them in the navigation tree below.

Navigation:
- default-navigation-model
  - Page Hierarchy
  - About Us
  - Menu
  - Menu Content

Content Query

Id:* menucontent

Repository: UCM

Query:

**************************************************************************************************

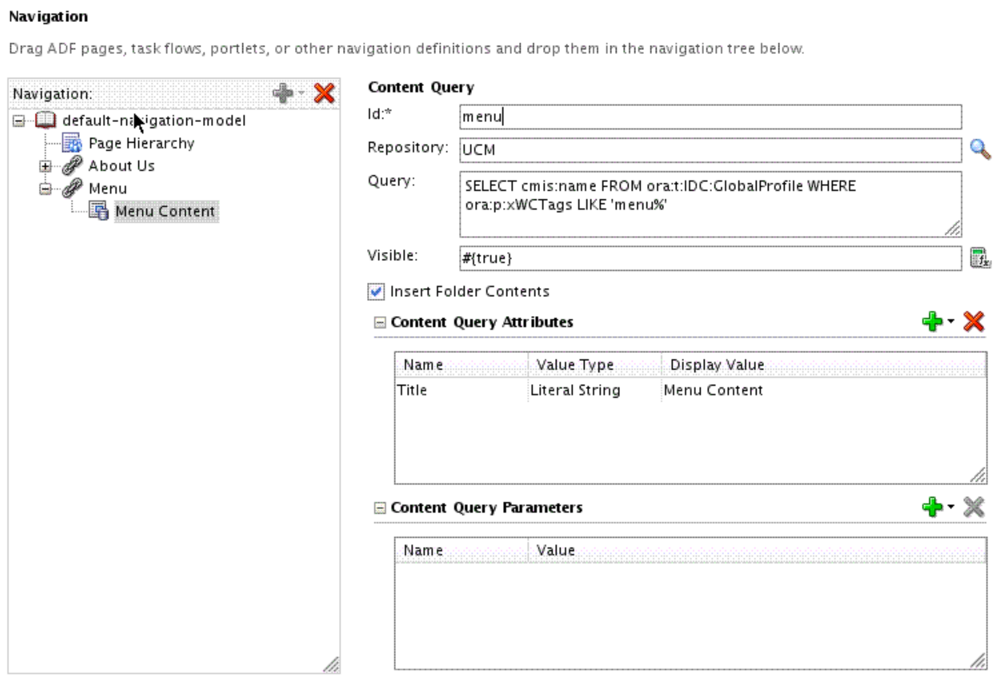
7. Drag and drop the Menu Content item, which is a content query, on the Menu item. This query, based on a metadata field tag in UCM, will fetch all the documents that match the specified criteria for the query in UCM, as shown in Figure 6–33.

8. Enter the Content Query text in the Query field:

```
SELECT cmis:name FROM ora:t:IDC:GlobalProfile WHERE ora:p:xWCTags LIKE 'menu%'
```

Note that documents have to be tagged with the keyword `menu` for the content query to work properly.

***Figure 6–33 The Menu Contact Content Query Text Entered***



Id:* menu

Repository: UCM

Query:

```
SELECT cmis:name FROM ora:t:IDC:GlobalProfile WHERE
ora:p:xWCTags LIKE 'menu%'
```

Visible: #{true}

Insert Folder Contents ✓

Content Query Attributes

| Name | Value Type | Display Value |
|---|---|---|
| Title | Literal String | Menu Content |

Content Query Parameters

| Name | Value |
|---|---|

*****************************************************************************************************

9. Ensure that you check **Insert Folder Contents**.

   **Tip:** You need to perform this step, which is necessary and required in this case, to ensure that the result of the menu content query is published under the menu node because the menu content query is a child of the menu node. The step is not required in all use cases, however.

10. To ensure that your Content Query executes properly and returns the correct set of results, you need to set its metadata on the content.

    As shown in Step 8, the Content Management Interoperability Services (CMIS) query looks like this:

    ```
    SELECT cmis:name FROM ora:t:IDC:GlobalProfile WHERE
    ora:p:xWCTags LIKE 'menu%'
    ```

    Note that UCM relies on the concept of a “display” name and an “internal” name for each metadata field. For example, in this CMIS query `xWCtags` is the internal name for the metadata field while Tags is the display name for the same field. When you build a CMIS query, you need to use the UCM *internal* name for the field.

**Tip:** The CMIS query used in this example fetches all the documents based on the metadata field `Tags`. For the content query to work properly, you must enter the metadata next to the `Tags` field in the Oracle WebCenter Content Administration Console for Content Server. In this example, the `Tags` field is set to a value specified as `menu`.

Keep in mind that the `Tags` field described here is strictly one of the metadata fields in Oracle WebCenter Content repository, and not related to any other Oracle WebCenter Tags service or tagging capability.

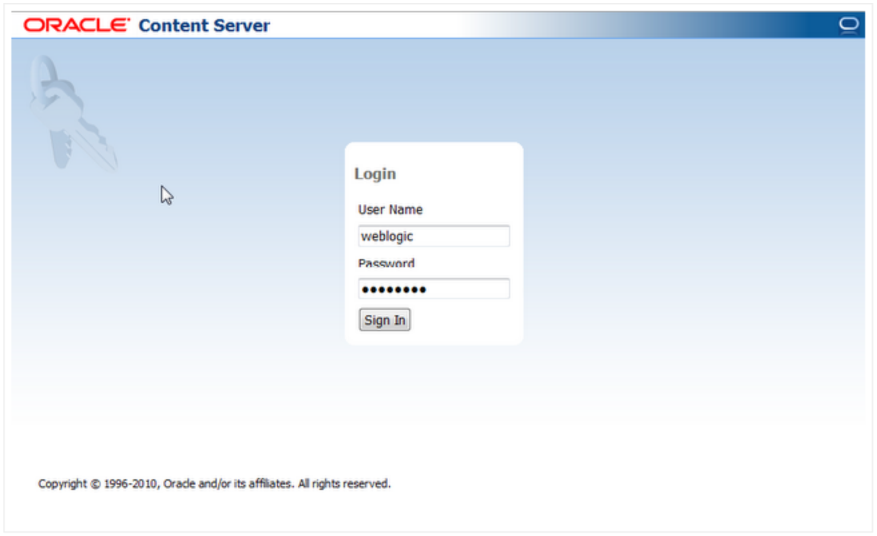
For more information on how to view content metadata -- in particular, the `Tags` field -- see the section "Viewing Content Information" in the Chapter "Working with Files" in *Oracle Fusion Middleware User's Guide for Content Server.*

For more information on how to update content metadata, see *Oracle Fusion Middleware User's Guide for Content Server.*

The following steps will enable you to set the correct values on the Tags field.

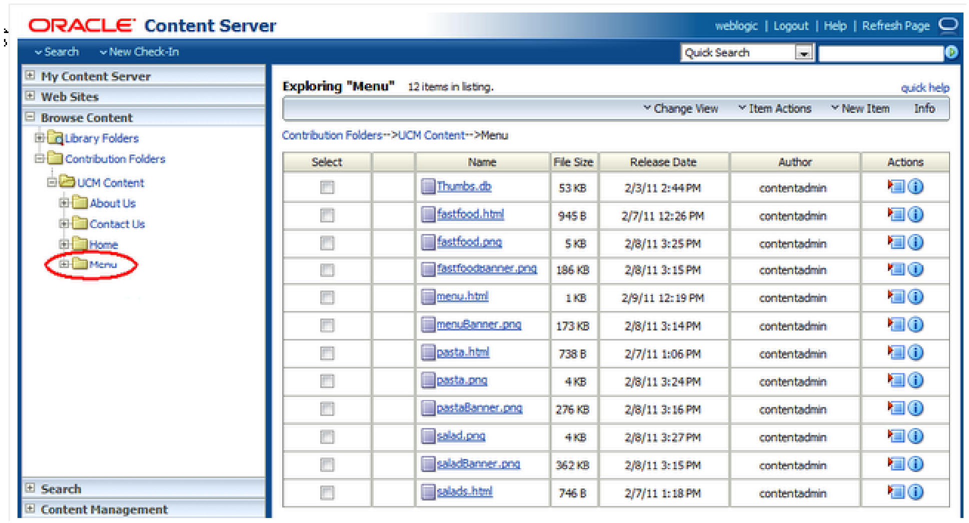
11. Log into the UCM instance with Administrator privileges enabled, as shown in Figure 6–34.

***Figure 6–34 Login With Administrator Privileges***


ORACLE Content Server
Login
User Name
weblogic
Password
••••••••
Sign In
Copyright © 1996-2010, Oracle and/or its affiliates. All rights reserved.

12. Navigate to the `Menu` folder in the `Contribution Folders` directory in the UCM repository, as shown in Figure 6–35.

*Figure 6–35 The Menu Folder For Oracle Content Server in UCM*



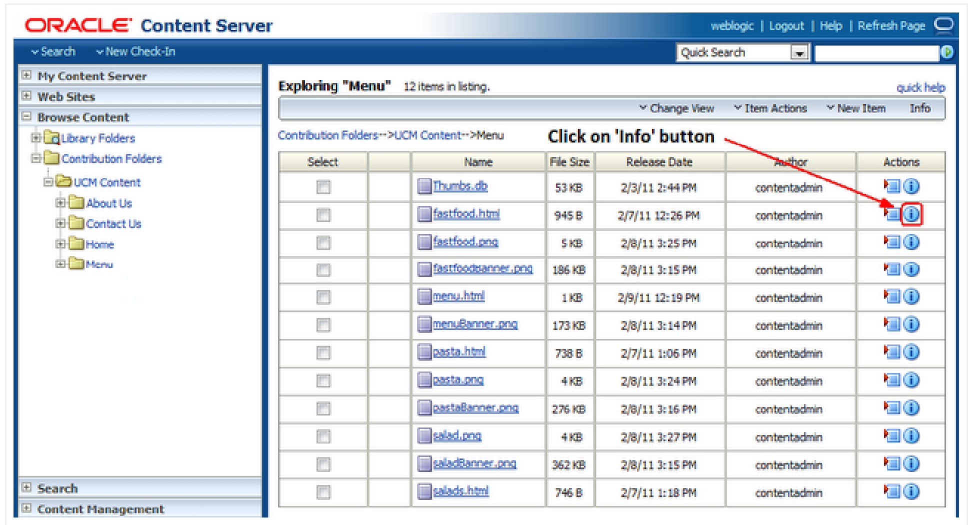
Exploring "Menu" 12 items in listing.

Contribution Folders-->UCM Content-->Menu

| Select | | Name | File Size | Release Date | Author | Actions |
|---|---|---|---|---|---|---|
| ☐ | | Thumbs.db | 53 KB | 2/3/11 2:44 PM | contentadmin | |
| ☐ | | fastfood.html | 945 B | 2/7/11 12:26 PM | contentadmin | |
| ☐ | | fastfood.png | 5 KB | 2/8/11 3:25 PM | contentadmin | |
| ☐ | | fastfoodBanner.png | 186 KB | 2/8/11 3:15 PM | contentadmin | |
| ☐ | | menu.html | 1 KB | 2/9/11 12:19 PM | contentadmin | |
| ☐ | | menuBanner.png | 173 KB | 2/8/11 3:14 PM | contentadmin | |
| ☐ | | pasta.html | 738 B | 2/7/11 1:06 PM | contentadmin | |
| ☐ | | pasta.png | 4 KB | 2/8/11 3:24 PM | contentadmin | |
| ☐ | | pastaBanner.png | 276 KB | 2/8/11 3:16 PM | contentadmin | |
| ☐ | | salad.png | 4 KB | 2/8/11 3:27 PM | contentadmin | |
| ☐ | | saladBanner.png | 362 KB | 2/8/11 3:15 PM | contentadmin | |
| ☐ | | salads.html | 746 B | 2/7/11 1:18 PM | contentadmin | |

13. Click the **Info** button under the **Actions** menu item for the `fastfood.html`file, shown in Figure 6–36. This action takes you to the Content Information page in Oracle Content Server (Figure 6–37).

*Figure 6–36 Clicking The Info Button for the fastfood.html File*



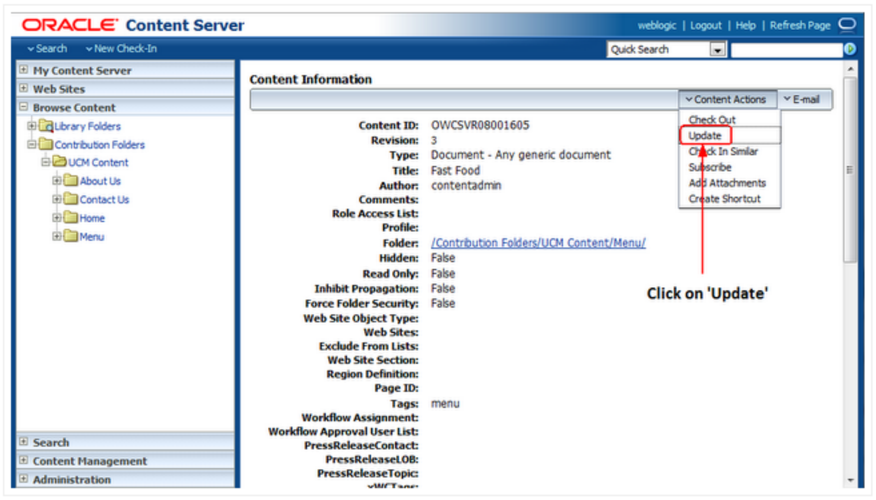
Exploring "Menu" 12 items in listing.

Contribution Folders-->UCM Content-->Menu

| Select | | Name | File Size | Release Date | Author | Actions |
|---|---|---|---|---|---|---|
| ☐ | | Thumbs.db | 53 KB | 2/3/11 2:44 PM | contentadmin | |
| ☐ | | fastfood.html | 945 B | 2/7/11 12:26 PM | contentadmin | |
| ☐ | | fastfood.png | 5 KB | 2/8/11 3:25 PM | contentadmin | |
| ☐ | | fastfoodBanner.png | 186 KB | 2/8/11 3:15 PM | contentadmin | |
| ☐ | | menu.html | 1 KB | 2/9/11 12:19 PM | contentadmin | |
| ☐ | | menuBanner.png | 173 KB | 2/8/11 3:14 PM | contentadmin | |
| ☐ | | pasta.html | 738 B | 2/7/11 1:06 PM | contentadmin | |
| ☐ | | pasta.png | 4 KB | 2/8/11 3:24 PM | contentadmin | |
| ☐ | | pastaBanner.png | 276 KB | 2/8/11 3:16 PM | contentadmin | |
| ☐ | | salad.png | 4 KB | 2/8/11 3:27 PM | contentadmin | |
| ☐ | | saladBanner.png | 362 KB | 2/8/11 3:15 PM | contentadmin | |
| ☐ | | salads.html | 746 B | 2/7/11 1:18 PM | contentadmin | |

14. On the Content Information page (Figure 6–37), choose the Content Actions menu, and click the **Update** menu item. In so doing, you will land on the Info Update Form for the `fastfood.html` page (Figure 6–38).

*Figure 6–37 Content Actions Menu With Click On Update Selected*

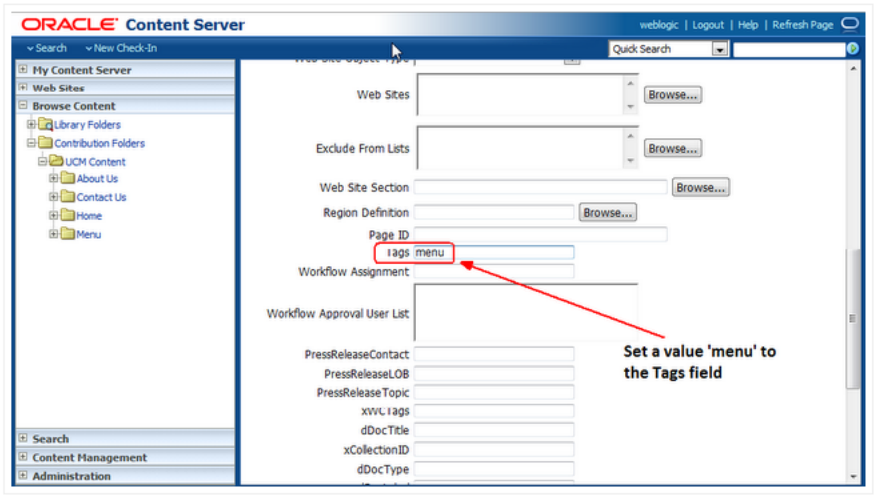


Content Information

| | |
|---|---|
| Content ID: | OWCSVR08001605 |
| Revision: | 3 |
| Type: | Document - Any generic document |
| Title: | Fast Food |
| Author: | contentadmin |
| Comments: | |
| Role Access List: | |
| Profile: | |
| Folder: | /Contribution Folders/UCM Content/Menu/ |
| Hidden: | False |
| Read Only: | False |
| Inhibit Propagation: | False |
| Force Folder Security: | False |
| Web Site Object Type: | |
| Web Sites: | |
| Exclude From Lists: | |
| Web Site Section: | |
| Region Definition: | |
| Page ID: | |
| Tags: | menu |
| Workflow Assignment: | |
| Workflow Approval User List: | |
| PressReleaseContact: | |
| PressReleaseLOB: | |
| PressReleaseTopic: | |

15. In the Info Update Form, now set a value defined as `menu` in the Tags field, as shown in Figure 6–38.

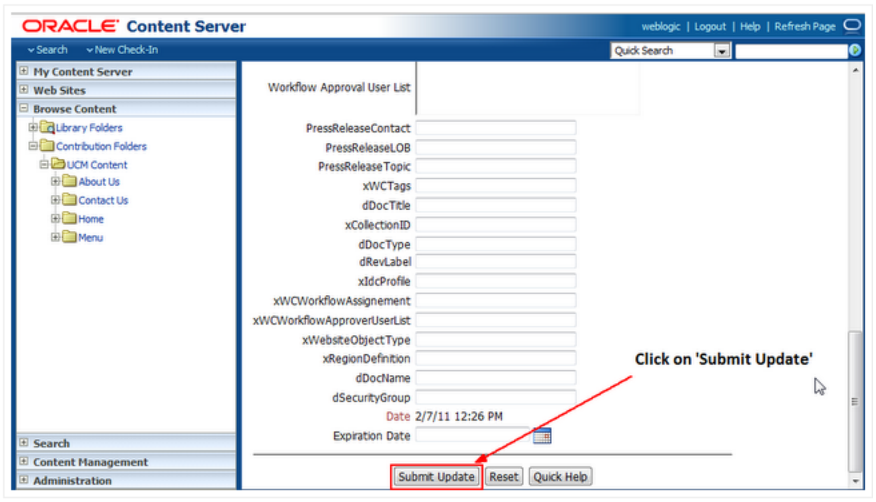
*Figure 6–38 Setting the Value menu in the Tags Field*



| | |
|---|---|
| Web Sites | |
| Exclude From Lists | |
| Web Site Section | |
| Region Definition | |
| Page ID | |
| Tags | menu |
| Workflow Assignment | |
| Workflow Approval User List | |
| PressReleaseContact | |
| PressReleaseLOB | |
| PressReleaseTopic | |
| xWCTags | |
| dDocTitle | |
| xCollectionID | |
| dDocType | |

16. Click the **Submit Update** button at the bottom of the form (Figure 6–39). In so doing, you will set the metadata for the content query.

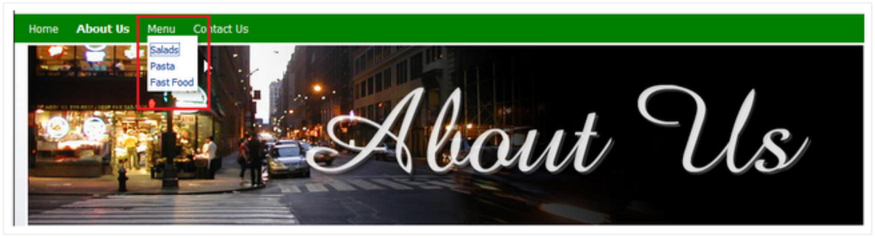
*Figure 6–39 Clicking Submit Update to Set the Metadata For The Content Query*



Workflow Approval User List
PressReleaseContact
PressReleaseLOB
PressReleaseTopic
xWCTags
dDocTitle
xCollectionID
dDocType
dRevLabel
xIdcProfile
xWCWorkflowAssignement
xWCWorkflowApproverUserList
xWebsiteObjectType
xRegionDefinition
dDocName
dSecurityGroup
Date 2/7/11 12:26 PM
Expiration Date

17. Now save your changes, return to your portal application and refresh the home page in your web browser. The **Fast Food** link appears under the Menu node in the navigation bar (Figure 6–40).

*Figure 6–40 The Fast Food Link Displayed Under the Menu Node*



18. Repeat steps 14-18 for other HTML files, like `salad.html` and `pasta.html`, which reside in the `Menu` folder in your repository. Once you set the metadata on the Tags field for all the HTML files, the results of each CMIS query will appear under the Menu node when you refresh your web browser.

In this lesson, you've learned the importance of working with and managing your document contents in the Oracle WebCenter Content repository. In so doing, you can optimize your development efforts, enable Iterative Development and add new content to your navigation model. You can also take advantage of adding content queries to your navigation model, which enable you to query and fetch documents based on specific metadata tags that are used in Oracle WebCenter Content.

In the next lesson, you will move ahead to extend your portal development skills by learning how to customize portal pages and add them to your page hierarchy, setting permissions on user access. In addition, you'll learn how you can easily edit HTML content in-context at runtime in your portal application.

# 7 Customizing Pages for Permissions and Runtime Editing

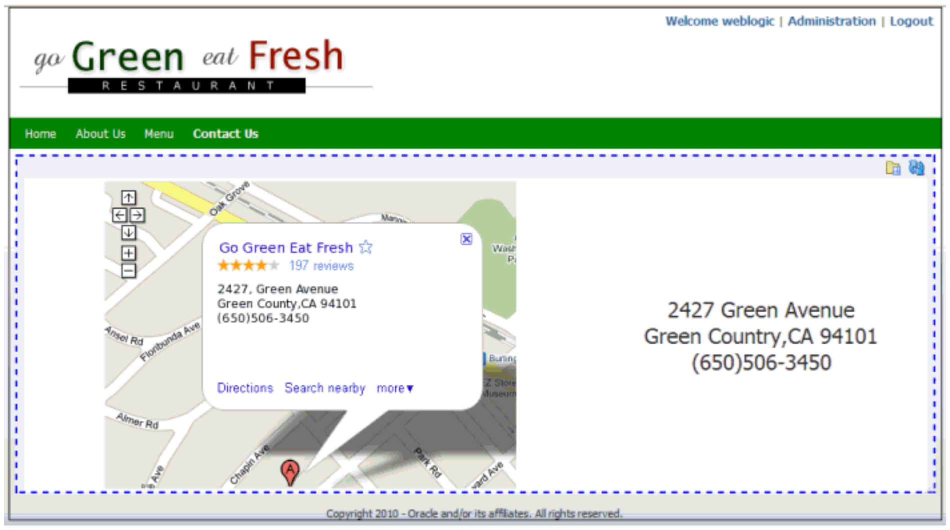
In this lesson, as a developer at design time, you will customize the content on a specific page in the page hierarchy of your WebCenter Portal: Framework application, adding that page and its content to the hierarchy so you can set permissions for user access and editing, or modification of document content.

You will also drag and drop documents in HTML format from your UCM repository on to a consignable panel component in a new `.jspx` page and render that page as a Content Presenter task flow. The Content Presenter task flow lets you add content to your portal application, in this case, using the Documents service to display that document content stored in your content repository. The Documents service provides a user-friendly interface to manage, display, and search documents at runtime.

In the last step, you will enable authenticated users to edit the content of those HTML pages in-context at runtime. A pop-up window will open an editing session for adding or modifying content stored in your UCM repository.

At the end of this lesson, the home page you created in the previous lesson will look like Figure 7–1 when in-context editing of HTML content is enabled on the **Contact Us** page at runtime.

***Figure 7–1 The Portal Application Contact Us Page Enabled for Editing at Runtime***



2427, Green Avenue
Green County,CA 94101
(650)506-3450

2427 Green Avenue
Green Country,CA 94101
(650)506-3450

Copyright 2010 - Oracle and/or its affiliates. All rights reserved.

## Introduction

This lesson contains the following steps:

- Step 1: Customize Pages and Set Permissions
- Step 2: Edit Documents at Runtime

Before you begin the steps in this lesson, ensure you have followed the steps up to this point in the Tutorial.

## Step 1: Customize Pages and Set Permissions

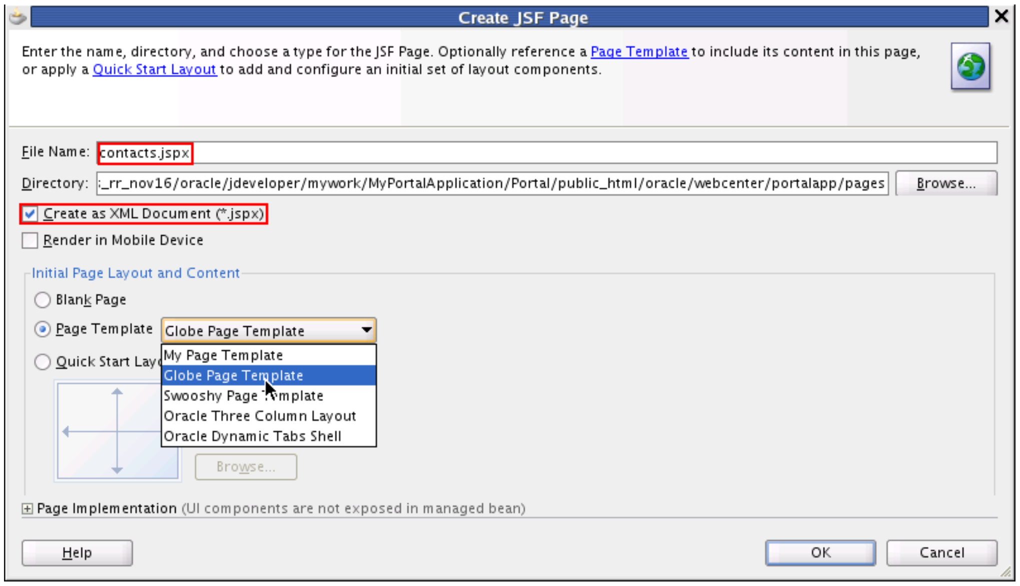
Building on the steps of the previous lesson (Chapter 6), we can now move ahead to customize specific pages in our WebCenter Portal application, thus setting page permissions and user access.

To accomplish this task, we first need to create a new `.jspx` page, in this case, `contacts.jspx`, and then add that page to the Page Hierarchy. By adding the page to the hierarchy, we can set page permissions, and in so doing, we can specify user access to those pages for security purposes.

To customize pages and set page permissions in our portal application:

1. In JDeveloper, navigate to the `pages` folder in Application Navigator.
2. Right-click **New**, expand **Web Tier**, select **JSF** and select the **JSF Page**, which launches the Create JSF Page dialog, as shown in Figure 7–2.
3. Enter `contacts.jspx` in the File Name field and ensure that the checkbox **Create as XML Document (*.jspx)** is checked.

***Figure 7–2 The Create JSF Page Dialog with the Globe Page Template Selected***


Create JSF Page

Enter the name, directory, and choose a type for the JSF Page. Optionally reference a Page Template to include its content in this page, or apply a Quick Start Layout to add and configure an initial set of layout components.

File Name: contacts.jspx

Directory: :_rr_nov16/oracle/jdeveloper/mywork/MyPortalApplication/Portal/public_html/oracle/webcenter/portalapp/pages

[✓] Create as XML Document (*.jspx)

[ ] Render in Mobile Device

Initial Page Layout and Content

( ) Blank Page

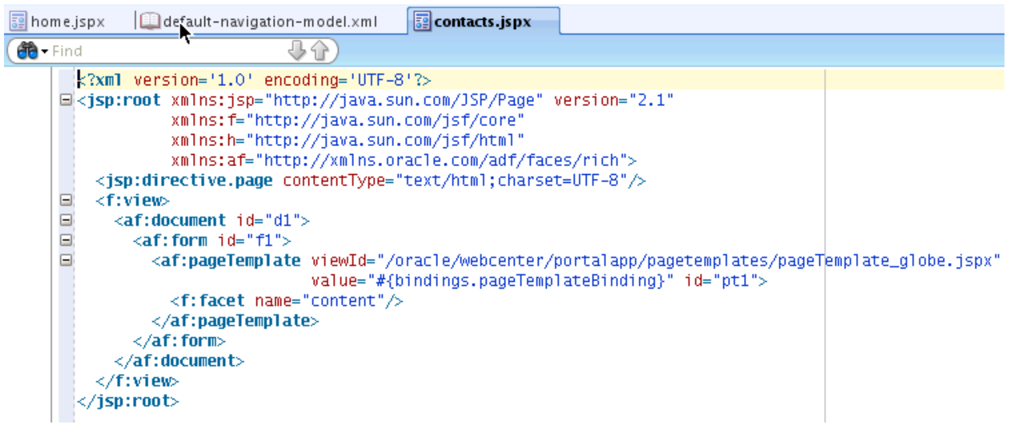
(●) Page Template: Globe Page Template

( ) Quick Start Layout

Page Implementation (UI components are not exposed in managed bean)

4. Select **Globe Page Template** from the list of available templates as the Page Template.
5. Click **OK**.
6. Select `contacts.jspx` and open it in **Source** view (Figure 7–3).

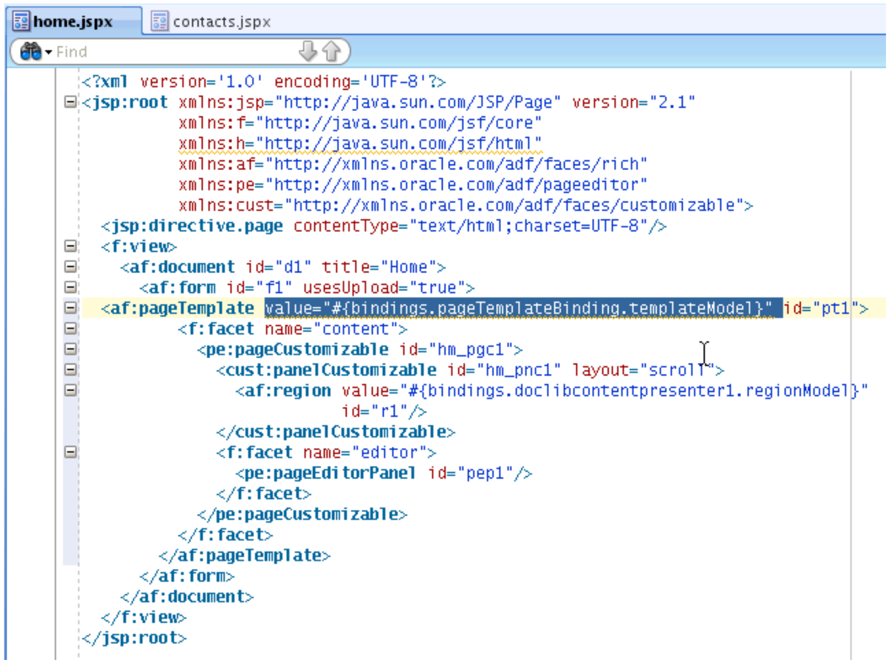
***Figure 7–3 The contacts.jspx File Shown in Source View***



```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1">
      <af:form id="f1">
        <af:pageTemplate viewId="/oracle/webcenter/portalapp/pagetemplates/pageTemplate_globe.jspx"
                         value="#{bindings.pageTemplateBinding}" id="pt1">
          <f:facet name="content"/>
        </af:pageTemplate>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

7. Now in Application Navigator, select the `home.jspx` file, which resides in the `pages` folder, open it in **Source** view and select the code line, which specifies the `value` attribute, shown in Figure 7–4. Note that this is needed so the page picks up the template changes that occur at runtime.

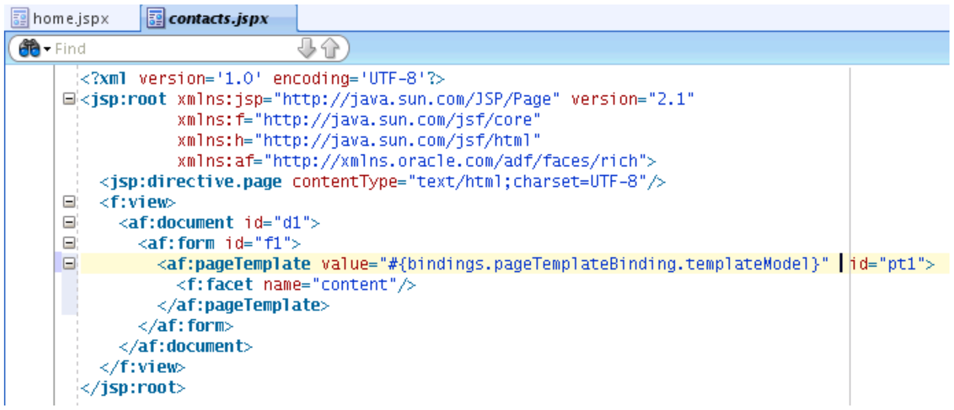
***Figure 7–4 The value Attribute Selected in the home.jspx File in Source View***



```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1" title="Home">
      <af:form id="f1" usesUpload="true">
  <af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}" id="pt1">
          <f:facet name="content">
            <pe:pageCustomizable id="hm_pgc1">
              <cust:panelCustomizable id="hm_pnc1" layout="scroll">
                <af:region value="#{bindings.doclibcontentpresenter1.regionModel}"
                           id="r1"/>
              </cust:panelCustomizable>
              <f:facet name="editor">
                <pe:pageEditorPanel id="pep1"/>
              </f:facet>
            </pe:pageCustomizable>
          </f:facet>
        </af:pageTemplate>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

8. Now copy the code line highlighted in Figure 7–4, which displays the `value` attribute that specifies the page template bindings, and paste this code line into the `contacts.jspx` file where the value attribute is located, as shown in Figure 7–5.

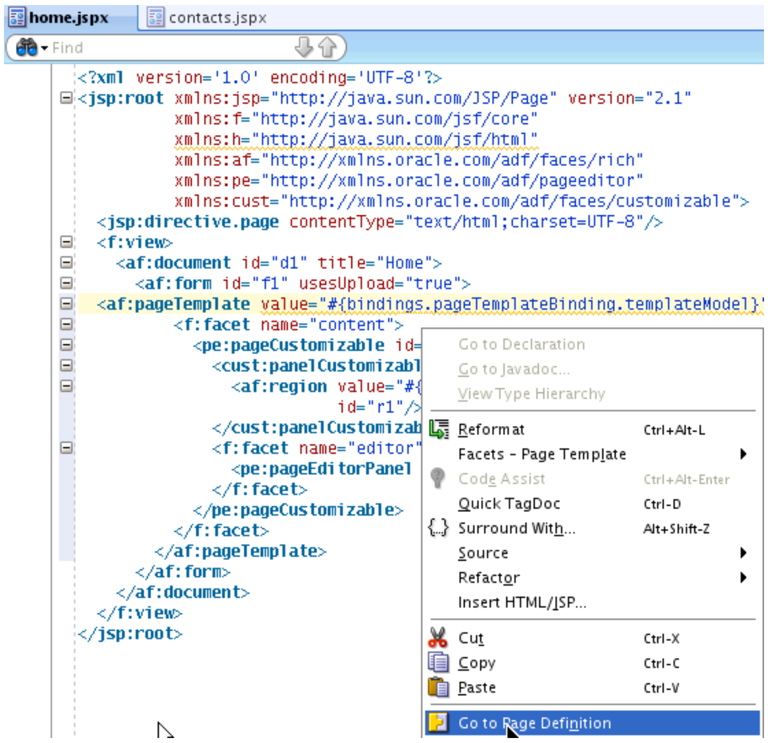
***Figure 7–5 The value Attribute Pasted into the contacts.jspx File***



```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1">
      <af:form id="f1">
        <af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}" id="pt1">
          <f:facet name="content"/>
        </af:pageTemplate>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

**9.** Select the `home.jspx` file, right-click the file, and choose the **Go to Page Definition** menu item, shown in Figure 7–6. This opens the `homePageDef.xml` file.

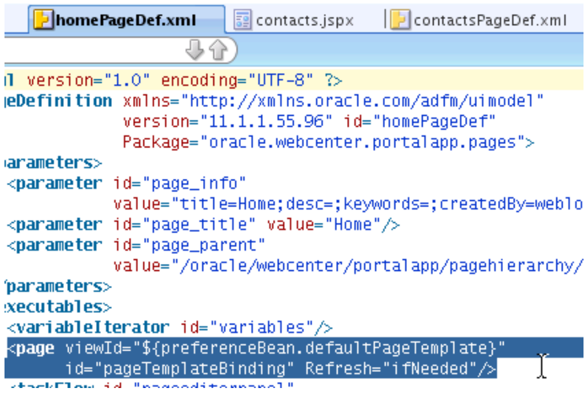
***Figure 7–6 Specifying the Page Definition in the home.jspx File in Source View***



```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1" title="Home">
      <af:form id="f1" usesUpload="true">
  <af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}'
        <f:facet name="content">
          <pe:pageCustomizable id=
            <cust:panelCustomizabl
              <af:region value="#{
                         id="r1"/>
            </cust:panelCustomizab
            <f:facet name="editor"
              <pe:pageEditorPanel
            </f:facet>
          </pe:pageCustomizable>
        </f:facet>
      </af:pageTemplate>
    </af:form>
  </af:document>
</f:view>
</jsp:root>
```

**10.** In the `homePageDef.xml` file in **Source** view, select and copy the code snippet specifying the `viewID` attribute for the page template and the `id` for the page template binding, as shown in Figure 7–7 and in Example 7–1.

*Figure 7–7 The homePageDef.xml File in Source View with the viewID Attribute Selected*



```
ıl version="1.0" encoding="UTF-8" ?>
ıeDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
             version="11.1.1.55.96" id="homePageDef"
             Package="oracle.webcenter.portalapp.pages">
ıarameters>
 <parameter id="page_info"
            value="title=Home;desc=;keywords=;createdBy=weblo
 <parameter id="page_title" value="Home"/>
 <parameter id="page_parent"
            value="/oracle/webcenter/portalapp/pagehierarchy/
'parameters>
:xecutables>
 <variableIterator id="variables"/>
 <page viewId="${preferenceBean.defaultPageTemplate}"
       id="pageTemplateBinding" Refresh="ifNeeded"/>
```

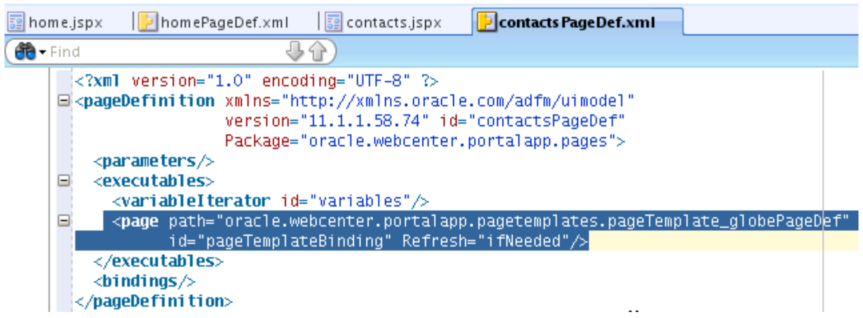
*Example 7–1 Code Snippet to Copy into contactsPageDef.xml*

```
<page viewID="${preferenceBean.defaultPageTemplate}"
      id="pageTemplateBinding" Refresh="ifNeeded"/>
```

**11.** Select the `contacts.jspx` file, right-click the file, and choose the **Go to Page Definition** menu item, as shown in Figure 7–6, when you opened the `homePageDef.xml` file.

**12.** Now open the `contactsPageDef.xml` file in **Source** view and paste the code snippet you copied from the previous step, replacing the code lines shown in Figure 7–8. These code lines specify the `path` and `id` attributes for the contacts page definition file.

*Figure 7–8 The path and id Attributes Selected in contactsPageDef.xml*



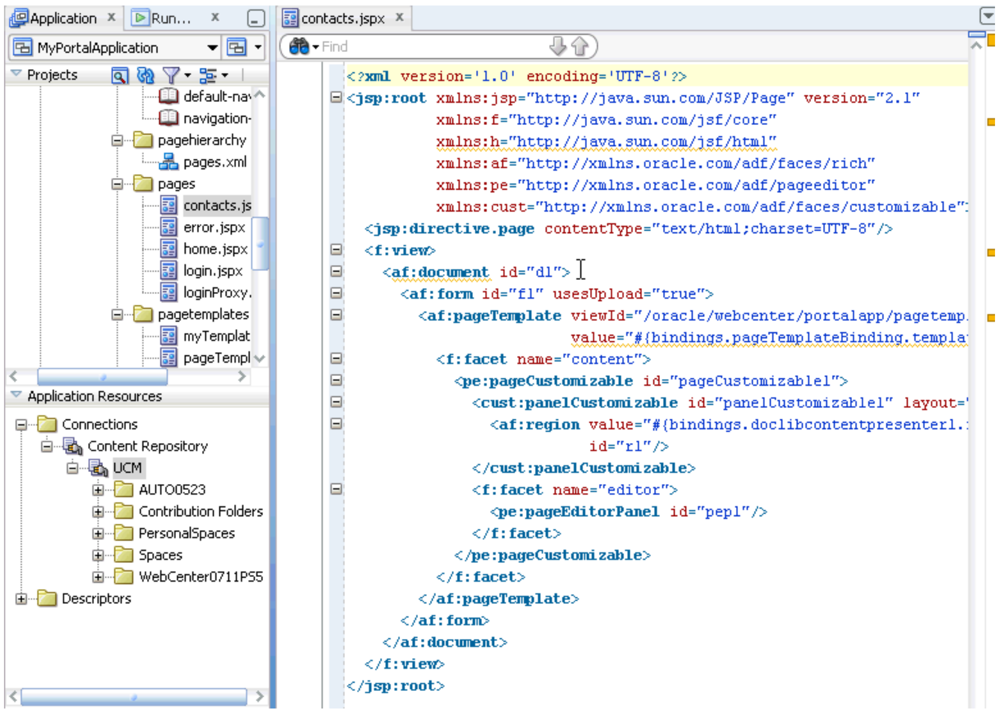
```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="11.1.1.58.74" id="contactsPageDef"
                Package="oracle.webcenter.portalapp.pages">
  <parameters/>
  <executables>
    <variableIterator id="variables"/>
    <page path="oracle.webcenter.portalapp.pagetemplates.pageTemplate_globePageDef"
          id="pageTemplateBinding" Refresh="ifNeeded"/>
  </executables>
  <bindings/>
</pageDefinition>
```

**13.** Now choose the `contacts.jspx` file and open it in Source view. Then in JDeveloper, navigate to the **Component Palette** (choose **View->Component Palette**), which resides in the right-hand column of the application, and select Composer.

**14.** In Composer, navigate down the items in the list to the `Page Customizable` component and drag and drop that component on to the `contacts.jspx` file after the code line `<f:facet name="content">`, as shown in Figure 7–9.

> **Tip:** The `Page Customizable` component denotes the customizable part of a page and appears in the Composer toolbar in Edit mode at runtime. Components enclosed within a `Page Customizable` component can be customized and edited.

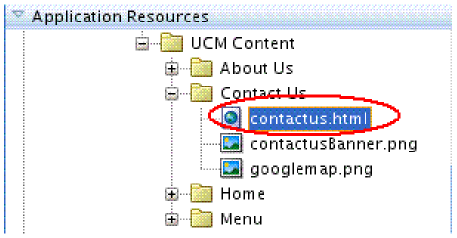
*Figure 7–9 The Code Snippet for Panel Customizable Added to contacts.jspx*



```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:f="http://java.sun.com/jsf/core"
          xmlns:h="http://java.sun.com/jsf/html"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
          xmlns:pe="http://xmlns.oracle.com/adf/pageeditor"
          xmlns:cust="http://xmlns.oracle.com/adf/faces/customizable"
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <f:view>
    <af:document id="d1">
      <af:form id="f1" usesUpload="true">
        <af:pageTemplate viewId="/oracle/webcenter/portalapp/pagetemp
                         value="#{bindings.pageTemplateBinding.templa
          <f:facet name="content">
            <pe:pageCustomizable id="pageCustomizable1">
              <cust:panelCustomizable id="panelCustomizable1" layout='
                <af:region value="#{bindings.doclibcontentpresenter1.
                           id="r1"/>
              </cust:panelCustomizable>
              <f:facet name="editor">
                <pe:pageEditorPanel id="pep1"/>
              </f:facet>
            </pe:pageCustomizable>
          </f:facet>
        </af:pageTemplate>
      </af:form>
    </af:document>
  </f:view>
</jsp:root>
```

15. Now expand the UCM connection in JDeveloper, as shown in Figure 7–10, and drag and drop the `contactus.html` file on to the `panelCustomizable` component.

*Figure 7–10 The contactus.html File Selected in the Contact Us Folder*



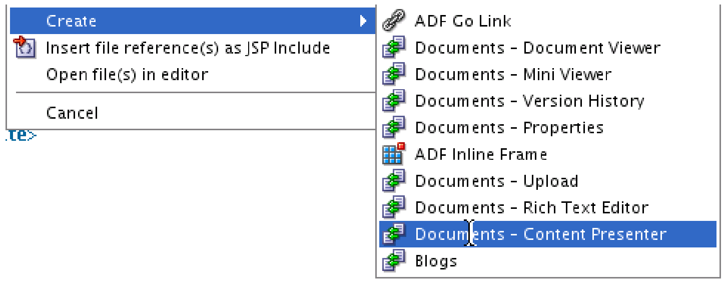
16. When the pop-up menu appears, choose the **Create** menu item and navigate to the **Documents - Content Presenter** task flow sub menu item shown in Figure 7–11 to render the HTML content.

**Tip:** The **Content Presenter** task flow lets you conveniently add content to your portal application. You can select a single item of content, multiple content items, or query for content, and then select a template to render that content on a page in your application. In so doing, Content Presenter enables you to precisely customize the selection and presentation of your content in your WebCenter Portal application.

Note that the **Content Presenter** task flow is available only when the connected content repository is Oracle Content Server and your WebCenter administrator has completed the prerequisite configuration. For more information about managing content repositories and UCM, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter Portal*

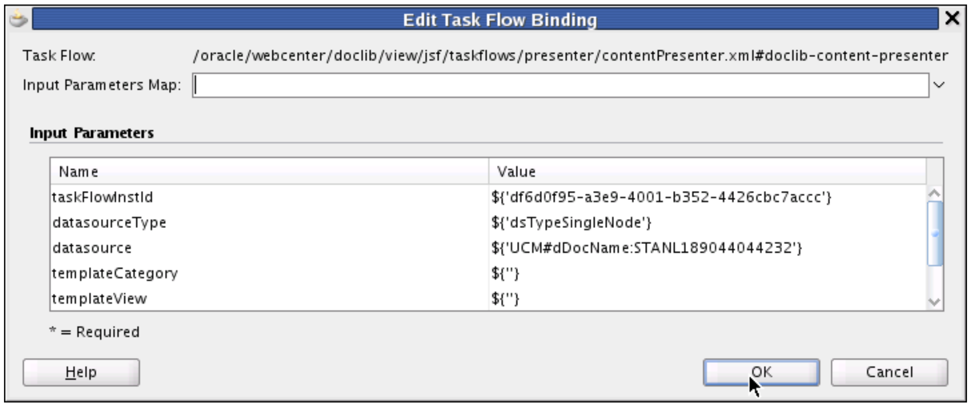
The **Documents** service enables you to display content from a content server or file system directly within your application. End users can then view and manage documents, as well as other types of content stored in content repositories.

***Figure 7–11 The Documents - Content Presenter Task Flow Selected for Creation***


Create
Insert file reference(s) as JSP Include
Open file(s) in editor
Cancel
ADF Go Link
Documents - Document Viewer
Documents - Mini Viewer
Documents - Version History
Documents - Properties
ADF Inline Frame
Documents - Upload
Documents - Rich Text Editor
Documents - Content Presenter
Blogs

**17.** The Edit Task Flow Binding dialog appears, as shown in Figure 7–12. Note that in the Input Parameters section, there is a UCM value for the `datasource` field.

***Figure 7–12 The Edit Task Flow Binding Dialog with a Data Source Value as a UCM Document Name***



Task Flow: /oracle/webcenter/doclib/view/jsf/taskflows/presenter/contentPresenter.xml#doclib-content-presenter

Input Parameters Map:

**Input Parameters**

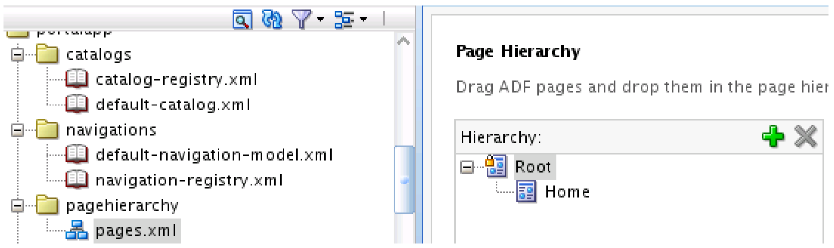
| Name | Value |
|---|---|
| taskFlowInstId | ${'df6d0f95-a3e9-4001-b352-4426cbc7accc'} |
| datasourceType | ${'dsTypeSingleNode'} |
| datasource | ${'UCM#dDocName:STANL189044044232'} |
| templateCategory | ${''} |
| templateView | ${''} |

* = Required

**18.** Click **OK**.

**19.** Return to the portal application and navigate to the `pagehierarchy` folder. Now open the `pages.xml` file, shown in Figure 7–13.

***Figure 7–13 The Page Hierarchy Root Node For the page.xml File***

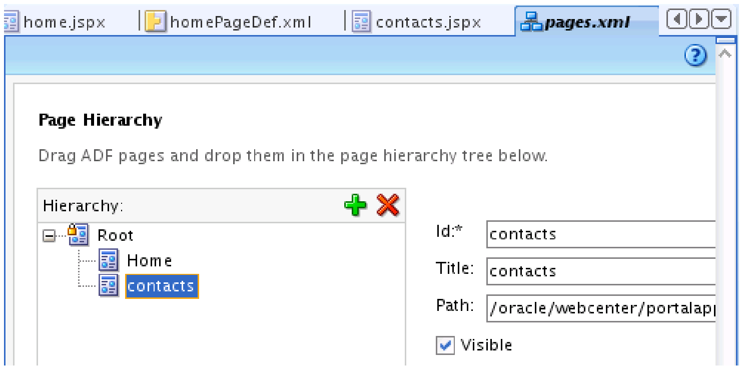


**Page Hierarchy**

Drag ADF pages and drop them in the page hier

**20.** When the `pages.xml` Page Hierarchy pane opens, drag and drop the `contacts.jspx` file into the Root node (Figure 7–14).

***Figure 7–14 The pages.xml Page Hierarchy with contacts Selected in the Root Node***



**Page Hierarchy**

Drag ADF pages and drop them in the page hierarchy tree below.

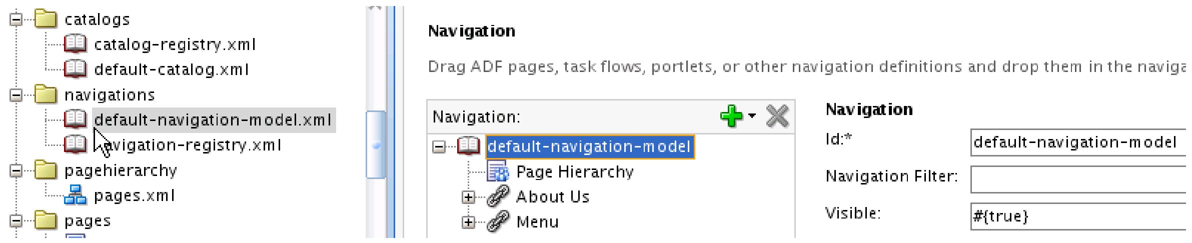
| | |
|---|---|
| Id:* | contacts |
| Title: | contacts |
| Path: | /oracle/webcenter/portalap |

✓ Visible

**21.** Select `contacts` in the Root node and in the Page Hierarchy pane, uncheck the **Visible** checkbox, which is shown as checked in Figure 7–14.

**22.** Now navigate to the `default-navigation-model.xml` file in the navigations folder and open it in **Design** view, as shown in Figure 7–15.

***Figure 7–15 The default-navigation-model.xml File Pane in Design View***



**Navigation**

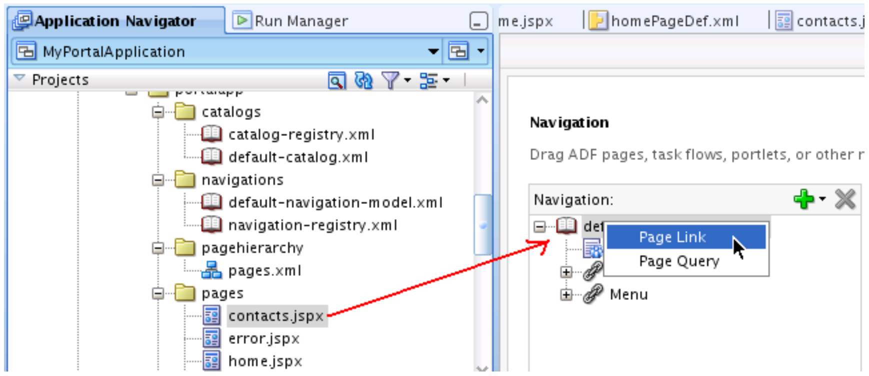
Drag ADF pages, task flows, portlets, or other navigation definitions and drop them in the naviga

**Navigation**

| | |
|---|---|
| Id:* | default-navigation-model |
| Navigation Filter: | |
| Visible: | #{true} |

**23.** Navigate to the `pages` folder, select the `contacts.jspx` file and drag it to the default-navigation-model node (Figure 7–16),

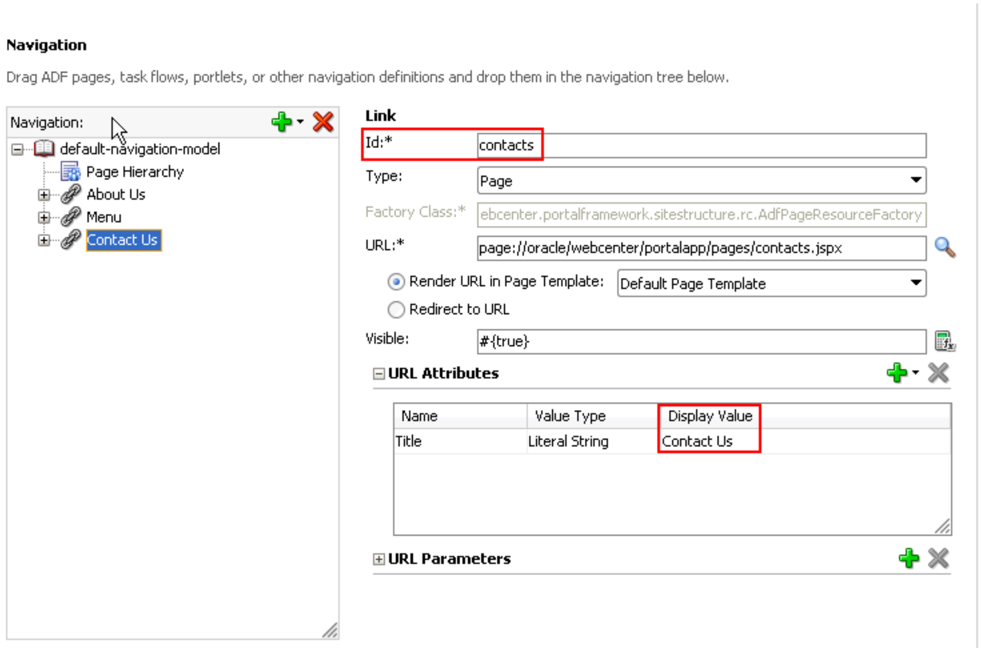
*Figure 7–16 The Contacts Link Added in the Navigation Model*



24. Choose the **Page Link** item and select the Add new node icon to add a link to the navigation model. and then click the new generated item **Contacts** under **Default-navigation-model**.

25. The navigation dialog opens, as shown in Figure 7–17, with `contacts` entered in the Id field. In the URL Attributes pane, enter the Display Value as `Contact Us`.

*Figure 7–17 The Navigation Pane with the Id for Contacts and the Display Value Specified as Contact Us*


Navigation
Drag ADF pages, task flows, portlets, or other navigation definitions and drop them in the navigation tree below.
Navigation:
default-navigation-model
Page Hierarchy
About Us
Menu
Contact Us
Link
URL Attributes
URL Parameters

| | |
|---|---|
| Id:* | contacts |
| Type: | Page |
| Factory Class:* | ebcenter.portalframework.sitestructure.rc.AdfPageResourceFactory |
| URL:* | page://oracle/webcenter/portalapp/pages/contacts.jspx |
| Render URL in Page Template: | Default Page Template |
| Redirect to URL | |
| Visible: | #{true} |

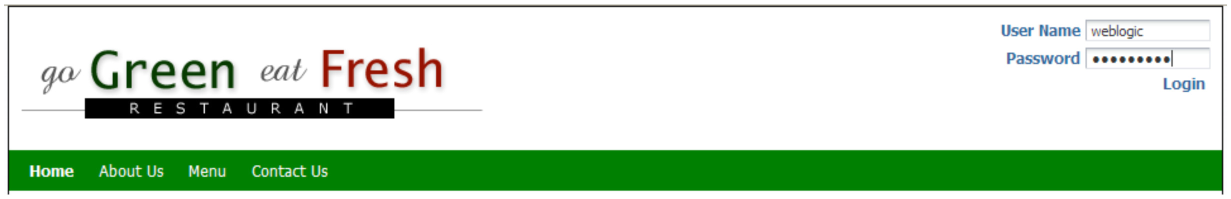
| Name | Value Type | Display Value |
|---|---|---|
| Title | Literal String | Contact Us |

26. Return to Application Navigator, select the Portal project and right-click **Run** to launch the application in a web browser.

    The application opens in a web browser with **Contact Us** as part of the navigation model, as shown in Figure 7–18. The page is populated with the content from Composer.

    For more information about creating and adding Content Presenter task flows and display templates, see the chapter "Creating Content Presenter Display Templates" in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

*Figure 7–18 MyPortalApplication in a Web Browser with the Contact Us Link Added to Navigation*



## Step 2: Edit Documents at Runtime

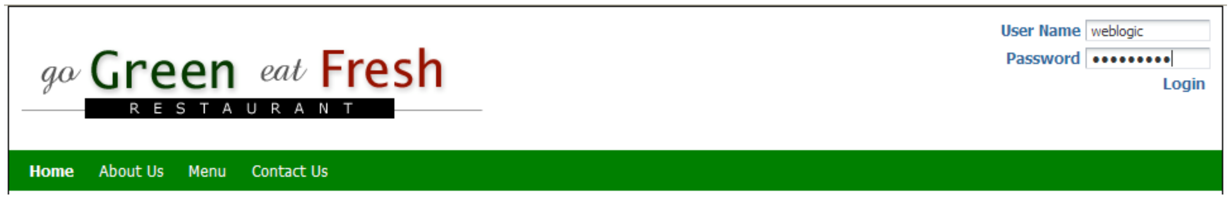
Now that we have customized our Contacts Page in our application, dragging and dropping the `contacts.html` document from our UCM repository on to the `panelCustomizable` component in `contacts.jspx` and rendering it as a Content Presenter task flow, we can move on to the task of enabling in-context HTML editing capability to our documents.

In-context document editing offers authenticated users easier access and control over document content. This feature is only available at runtime, however.

To edit documents at runtime:

1. In Application Navigator, select the Portal project and right-click **Run** to launch the application in a web browser.
2. Log in as `weblogic` (which enables you to have administrative privileges) with `weblogic1` as your password (Figure 7–19), and navigate to the Contact Us page by clicking in **Contact Us** link in the banner.

*Figure 7–19 Logging into the Home Web Page*

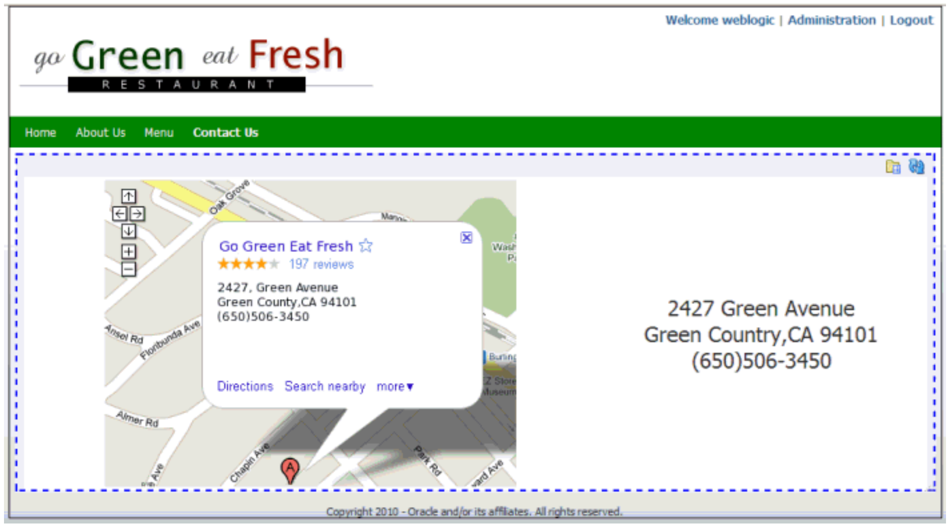


3. Once you areon the Contact Us page, click the content of the page and press `CTRL + SHIFT + C` on your keyboard. The Contact Us page in the web browser appears at the center with a hash-marked outline and an edit icon below the navigation bar in the upper right corner, indicating that the outlined portion of the Contact Us page can be edited for document content, as shown in Figure 7–20.

   **Tip:** By entering the `CTRL + SHIFT + C` keys, you will enable the content contributions mode, which allows you to edit in-context HTML content stored in the UCM content repository. You won't need to edit the page using Composer.

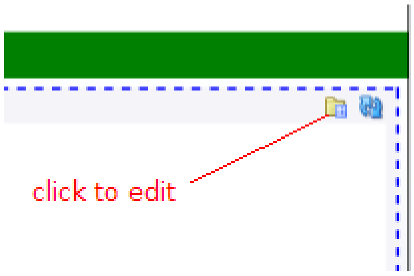
   Note that editing or customizing pages requires using a different set of tools and procedures, which enable you to add taskflows and portlets, for example, or to otherwise modify the page itself, and not just the document content on the page at runtime.

***Figure 7–20 The MyPortalApplication in a Web Browser with Content Editing Enabled***



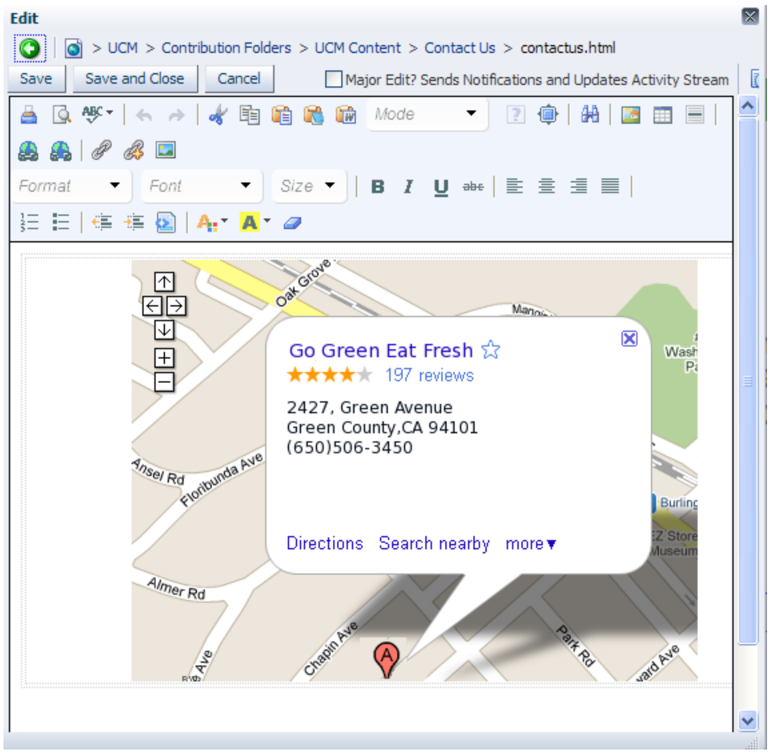
4. Navigate in the outlined portion of the browser page to the edit icon and click the icon to enable editing, as shown in Figure 7–21.

***Figure 7–21 The Edit Icon Selected to Enable Editing of Home Page Document Content***



5. An Edit window pops up at the center of the page where authenticated users can add content to the `contactus.html` page and edit that content using a rich text editor, as shown in Figure 7–22. Note that the document content is stored in the UCM content repository for easy access and retrieval.

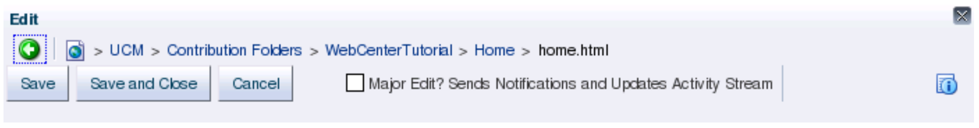
***Figure 7–22 The Edit Window for In-context Editing of HTML Content***


Edit
> UCM > Contribution Folders > UCM Content > Contact Us > contactus.html
Save
Save and Close
Cancel
Major Edit? Sends Notifications and Updates Activity Stream
Mode
Format
Font
Size
Oak Grove
Ansel Rd
Floribunda Ave
Almer Rd
Chapin Ave
Park Rd

Go Green Eat Fresh
197 reviews
2427, Green Avenue
Green County,CA 94101
(650)506-3450
Directions Search nearby more

6. Users can then save their edits by clicking the **Save** button in the upper left corner of the window (Figure 7–23).

***Figure 7–23 The Edit-Save Window for Saving of In-context HTML Document Content***


Edit
> UCM > Contribution Folders > WebCenterTutorial > Home > home.html
Save
Save and Close
Cancel
Major Edit? Sends Notifications and Updates Activity Stream

In this lesson, you have learned how to customize specific pages in the page hierarchy of your WebCenter Portal: Framework application and how to add those pages to the hierarchy so you can set permissions for user access. You have also learned how easy it is to enable runtime HTML editing of your document content stored in the UCM repository by adding new content to the page and making it editable at runtime.

# 8 Conclusion

Congratulations! You have created a WebCenter Portal: Framework application and learned about the fundamentals of Oracle WebCenter Portal: Framework.

## Summary

In this Tutorial, you learned how to perform a few quick and easy steps to create a WebCenter Portal: Framework application. You also learned how to modify and customize your application to meet the specific needs of end users coming to your web portal.

Specifically, you learned how to:

- Create a connection to a content repository (UCM), which allowed you to access the content you needed for building your portal application. As you move on and develop more complex WebCenter Portal: Framework applications, you may want to connect to other content repositories for various content, and so on. You can use the same methodology to create a connection to your other repositories.
- Create a simple WebCenter Portal: Framework application, which allowed you to check out how to use the built-in WebCenter application template to create a basic JSF application. You logged as Administrator in the Administration Console and changed the default page template from `Globe` to `swooshy`.
- Create and register a new page template, which enabled you to create a page definition for the template. You were then able to extract the contents of a setup file into your portal application. The file included images, templates and skins. You registered the new template and new skin as application resources in Oracle JDeveloper.
- Change the look and feel of the application using a new skin. At design time, you changed the default settings for both the template and the skin in the adf-config.xml file, and changed the entry preferences.
- Add pages to the page hierarchy in the default navigation model, and within that model, create links to your documents from UCM.
- Enable the process of Iterative Development in your portal application, thus enabling you to make changes to your application while it is still running on the Integrated WebLogic Server and then immediately see the effects of those changes when you refresh the pages in your web browser.
- Add new content queries in the default navigation model, with each query fetching documents based on metadata field tags in UCM.
- Customize pages and site templates in your portal and set permissions for user access.

- Edit HTML documents in-context at runtime, using a provided rich text editor.

You should now have a basic working knowledge of the fundamentals of Oracle WebCenter Portal Framework.

## Moving On

You can learn more about designing your own WebCenter Portal: Framework applications, including using  Composer, WebCenter Services, and portlets, in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter Portal*.

To learn more about what you can do at runtime, including using Composer to customize pages, and how the various components behave and can be configured at runtime, see the *Oracle Fusion Middleware User's Guide for Oracle WebCenter Portal: Spaces*.

You can find all Oracle WebCenter Portal documentation on the WebCenter Documentation page on the Oracle Technology Network, at `http://www.oracle.com/technology/products/webcenter/documentation.html`.

You can learn more about other features of Oracle WebCenter Portal, and view demonstrations and see examples of WebCenter Portal: Framework applications, portlets, and services in action on the Oracle WebCenter Portal home page on the Oracle Technology Network at:

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

# Index