

Oracle® Fusion Middleware

Tutorial for Oracle WebCenter Developers

11g Release 1 (11.1.1)

E10273-05

January 2011

Oracle Fusion Middleware Tutorial for Oracle WebCenter Developers, 11g Release 1 (11.1.1)

E10273-05

Copyright © 2007, 2011, Oracle and/or its affiliates. All rights reserved.

Primary Author: Tom Maremaa

Contributor: Bill Witman, Peter Moskovits, Kundan Vyas, Robin Fisher, Fadi Hakim, Bob Fraser, Ingrid Snedecor

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	v
Audience	v
Documentation Accessibility	vi
Related Documents	vi
Conventions	vi
1 Introduction to WebCenter Portal Framework and the Tutorial	
What is WebCenter Portal Framework?	1-1
What Will I Create?	1-2
The Development Scenario	1-3
Tutorial Path	1-4
2 Preparing for the Tutorial	
Introduction	2-1
Step 1: Obtain the Software	2-1
Step 2: Verify the Correctly Installed JDeveloper Release and WebCenter Extension	2-1
Step 3: Work with the Integrated WebLogic Server (WLS)	2-3
Step 4: Download the Sample Tutorial Files	2-4
Step 5: Create a Content Repository Connection	2-4
3 Creating a WebCenter Portal Application	
Introduction	3-1
Step 1: Create a Custom WebCenter Portal Application	3-1
Step 2: Use Seeded Page Templates to Build Your Portal Application	3-8
4 Creating a New Page Template with a New Portal Skin	
Introduction	4-1
Step 1: Create a New Page Template	4-1
Step 2: Extract Setup Files and Replace the Existing Template	4-4
Step 3: Set the New Template and Create a Portal Resource	4-7
5 Changing the Look and Feel of Your Portal Application	
Introduction	5-1
Step 1: Change the Default Settings For Template and Skin	5-1

Step 2: Change the Default Page Template at Runtime	5-4
6 Connecting to and Managing Content Repositories	
Introduction.....	6-2
Step 1: Connect to Universal Content Management (UCM) Repository	6-2
Step 2: Add a Content Item to the Navigation Model.....	6-7
Step 3: Take Advantage of Iterative Development	6-9
Step 4: Add a New Content Query	6-12
7 Customizing Pages For Permissions and Runtime Editing	
Introduction.....	7-1
Step 1: Customize Pages and Set Permissions.....	7-2
Step 2: Edit Documents at Runtime.....	7-8
8 Conclusion	
Summary	8-1
Moving On.....	8-2
Index	

Preface

This Tutorial introduces you to Oracle WebCenter Portal Framework, a key component of Oracle WebCenter Suite that enables you to build your own WebCenter Portal 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* 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 applications.

This Tutorial does not assume any prior knowledge of Oracle JDeveloper or Oracle WebCenter Suite. 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 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

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

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

Related Documents

For more information 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*, which explains how to use Oracle JDeveloper and Oracle WebCenter Portal Framework to develop WebCenter Portal applications
- *Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces*, which explains how to use WebCenter Portal 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.

Convention	Meaning
<i>italic</i>	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.

Introduction to WebCenter Portal Framework and the Tutorial

Welcome to Oracle WebCenter Portal Framework! This chapter introduces you to key Oracle 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 applications.

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

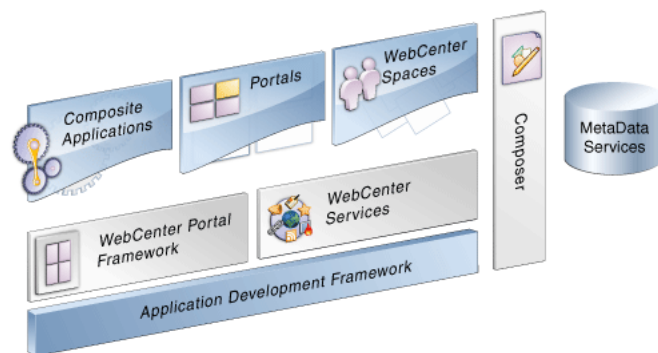
What is WebCenter Portal Framework?

The WebCenter Portal Framework provides portal-specific features to a WebCenter 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.

Oracle WebCenter 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 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.

[Figure 1-1](#) provides an overview of the Oracle WebCenter architecture, showing the major components that make up the product.

Figure 1–1 Overview of the Oracle WebCenter Architecture

In [Figure 1–1](#), notice WebCenter Services and Composer (or, Oracle 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” in the *Oracle Fusion Middleware Developer’s Guide for Oracle WebCenter*.

What Will I Create?

In this Tutorial, you will use WebCenter Portal Framework to build a WebCenter Portal 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 Oracle 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, Oracle 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 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 Universal Content Management (UCM) repository with access provided to the Oracle Content Server. You will set UCM as your primary connection and navigate to the WebCenterTutorial 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 UCM 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 WebCenter 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 UCM 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 UCM 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 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 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

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 Universal Content Repository (UCM), 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 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 (UCM) 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 application.

Preparing for the Tutorial

To prepare for this Tutorial, you need to obtain and install the current release of Oracle JDeveloper 11g Release 1 (11.1.1.4) 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 Universal Content Management (UCM) 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: Verify the Correctly Installed JDeveloper Release and WebCenter Extension](#)
- [Step 3: Work with the Integrated WebLogic Server \(WLS\)](#)
- [Step 4: Download the Sample Tutorial Files](#)
- [Step 5: 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: Verify the Correctly Installed JDeveloper Release and WebCenter Extension

Once you have obtained the software, ensure that you have installed Oracle JDeveloper 11g Release 1 (11.1.1.4), shown in [Figure 2-1](#), and the Oracle WebCenter extension (11.1.1).

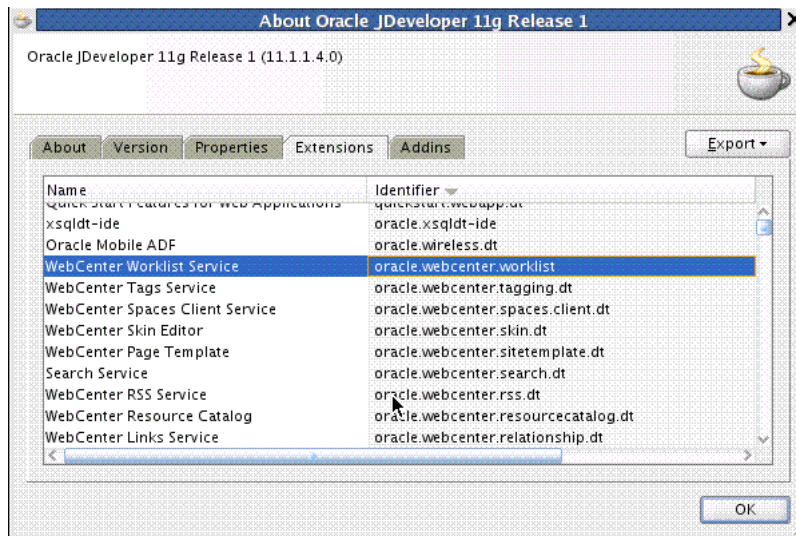
Figure 2–1 Oracle JDeveloper 11g Release 1 About Box



If you are not sure whether you have the WebCenter 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 11.1.1.4.0**. On the Extensions list, sort by **Identifier** to locate the `oracle.webcenter.*` components.

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

Figure 2–2 Oracle WebCenter Portal Framework in Oracle JDeveloper



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

To install the WebCenter extension to Oracle JDeveloper using the Update Center:

1. Launch Oracle JDeveloper.
2. If the Select Default Roles dialog displays, select **Default Role** to enable all technologies, and click **OK**.

3. If a dialog displays asking if you want to migrate settings from an earlier version, click **No**.
4. In Oracle JDeveloper, choose **Check for Updates** from the Help menu.
5. On the Welcome page, click **Next**.
6. Select **Search Update Centers**, and choose **Oracle Fusion Middleware Products**, then click **Next**.
7. On the Updates page, search for the WebCenter extension, select it, then click **Finish**.
8. When prompted, restart JDeveloper.

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

Step 3: 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.

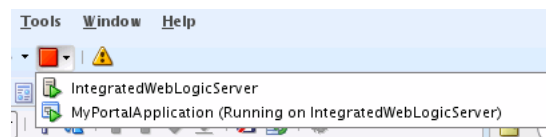
There are several options for starting Integrated WLS available in the **Run** menu in Oracle JDeveloper.

- 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 Fusion Middleware web page from your browser:
`http://localhost:7101`

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.

Step 4: 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 the 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/documentation/webcentertutorialcontent-11120-128869.zip
```

2. Open the ZIP file (`webcentertutorialcontent-11120-128869.zip`).
3. Unzip the file to a local drive, such as `C:\TutorialContent`.

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.

Step 5: 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 Universal Content Management (UCM) repository, which provides access to the Oracle Content Server.

Connecting to UCM 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 UCM. 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 "Configuring Content Repository Connections" in Oracle Fusion Middleware Developer's Guide for Oracle WebCenter. The chapter discusses how to configure connections to content repositories to provide access to the content.

For information about how to configure Oracle Content Server for Oracle WebCenter, see "Oracle Content Server Prerequisites" in Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter.

Creating a WebCenter Portal Application

In this lesson, you will create a basic WebCenter Portal 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 login as Administrator and open the page in a web browser.

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



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 application, using the WebCenter wizard for creating new portal applications. The wizard uses an out-of-the-box **Portal**

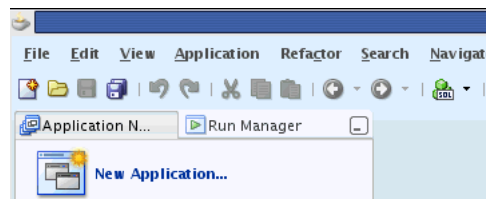
Application template that ensures the appropriate application components are included.

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:

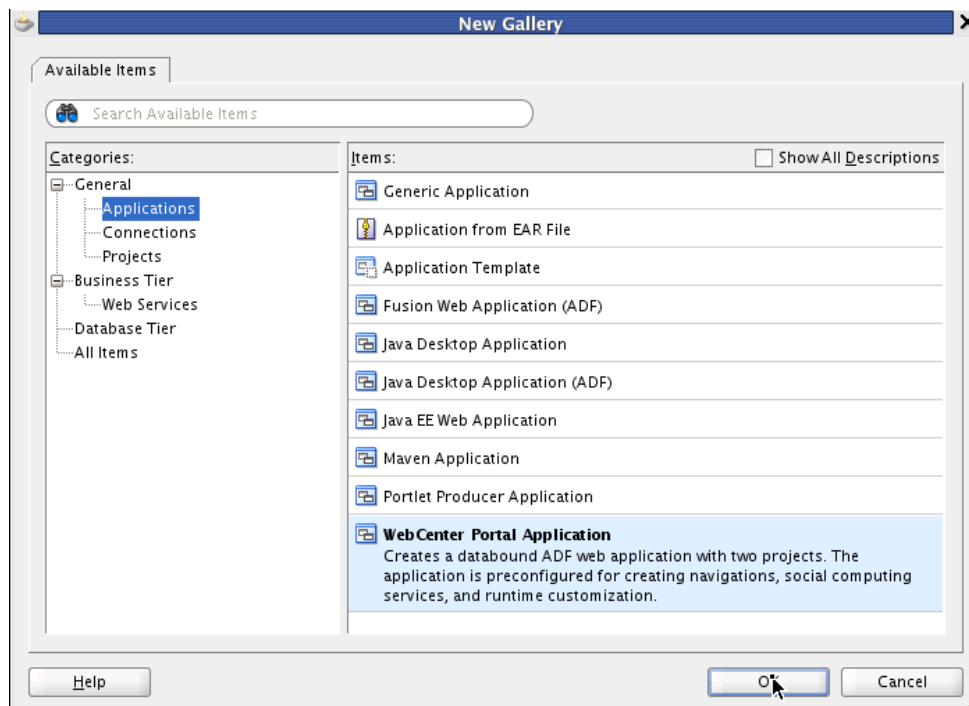
1. In Oracle JDeveloper, in the Application Navigator, choose the **New Application** icon ([Figure 3-2](#)) and click it to launch the application wizard.

Figure 3-2 The New Application Icon in Application Navigator



2. In the New Gallery, under the General category, select **Applications**.
3. Now in the Items list, navigate down the list and select **WebCenter Portal Application**, then click **OK** ([Figure 3-3](#)).

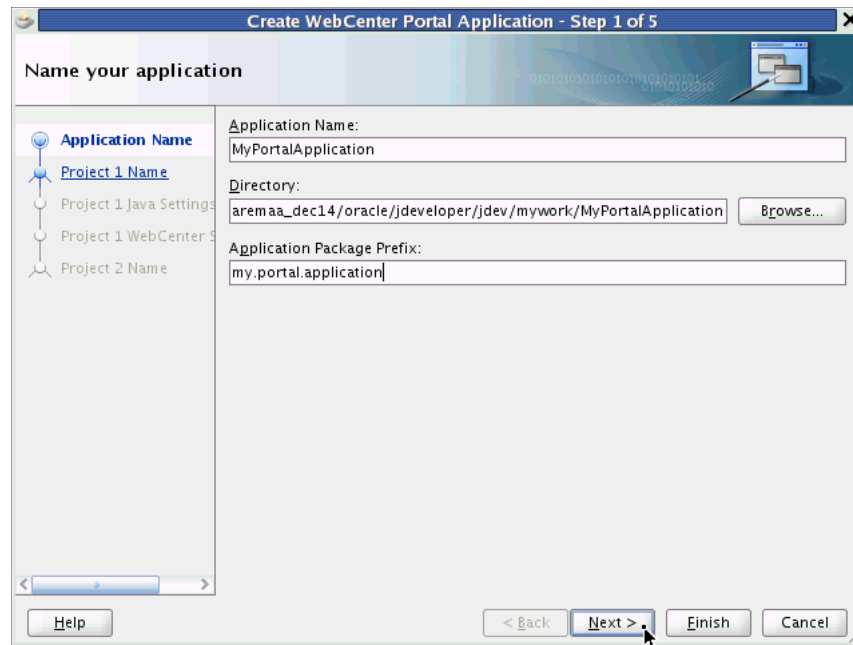
Figure 3-3 Create New WebCenter Portal Application



4. 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.

5. In the Application Package Prefix field (shown in [Figure 3-4](#)), enter `my.portal.application`.

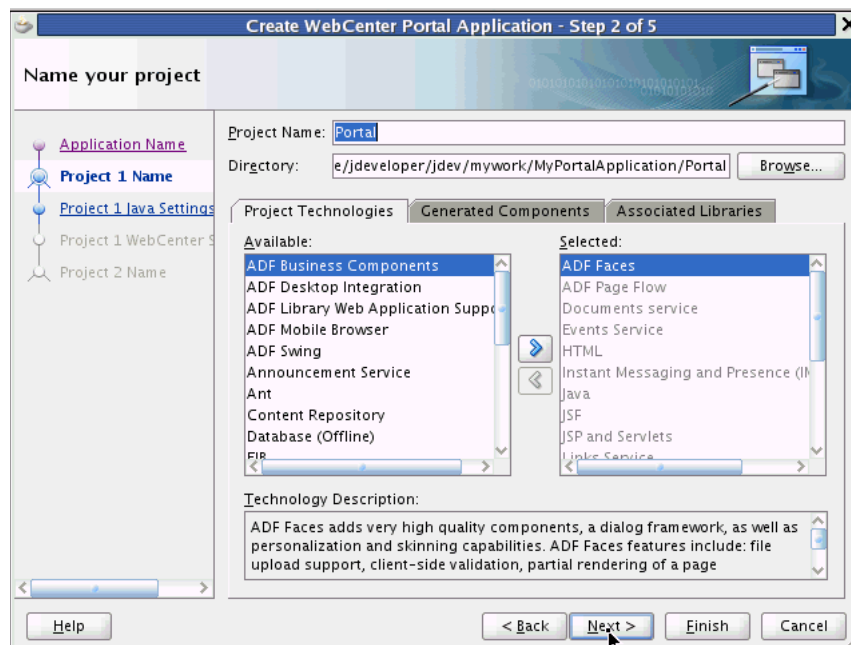
Figure 3-4 Naming Your Application - Step 1 of 5



6. 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.

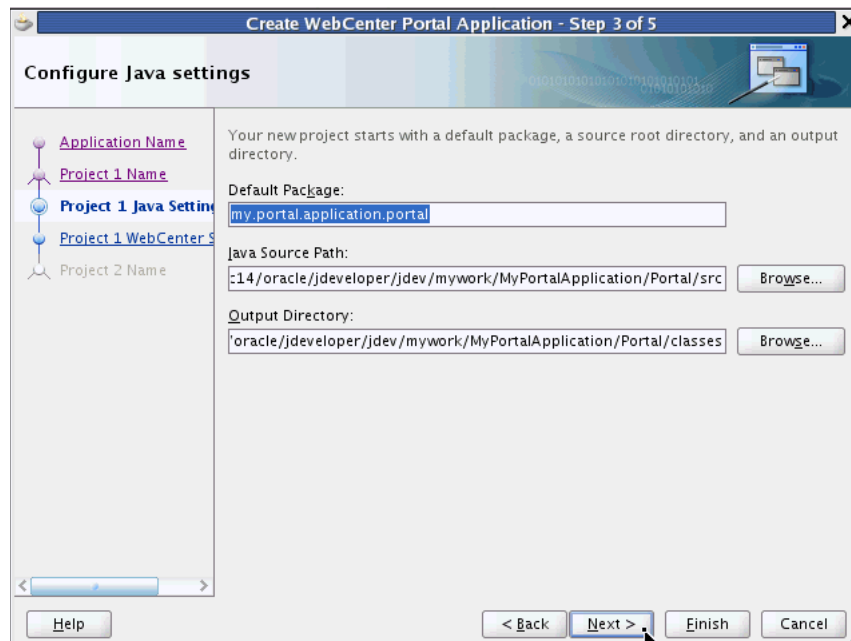
Figure 3–5 Name Your WebCenter Project - Step 2 of 5



7. 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.

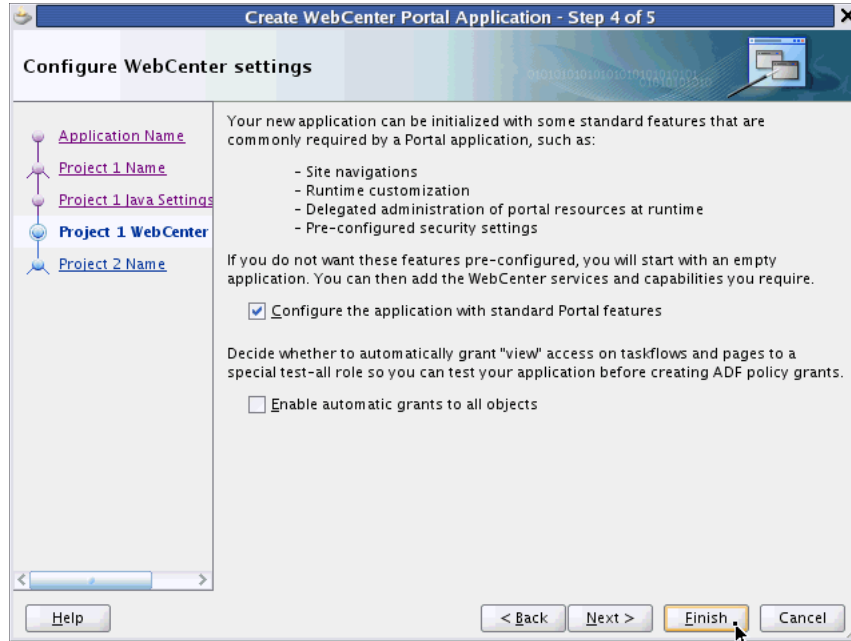
Figure 3–6 Configure Java Settings in WebCenter - Step 3 of 5



8. 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.

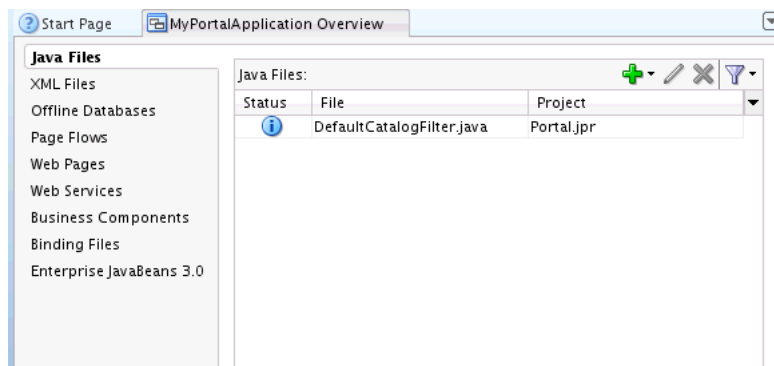
Figure 3-7 Configure WebCenter Settings - Step 4 of 5



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 3-8](#)).

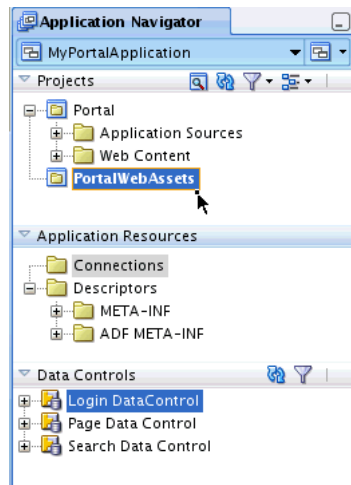
Figure 3-8 The MyPortalApplication Overview Window with Java Files Selected



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 3–9](#).

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.

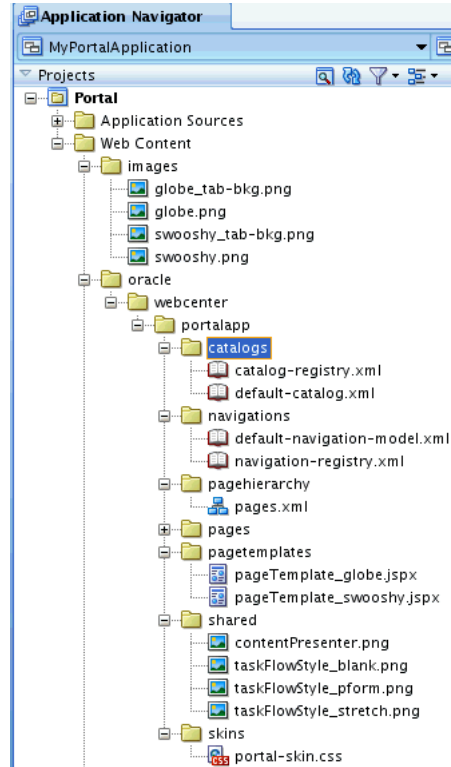
Figure 3–9 The MyPortalApplication Project in a Collapsed View in Application Navigator



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 3–10](#)), for a view of the logical structure and parent-child relationships created in your portal application.

Figure 3–10 Expanded Folders in MyPortalApplication Project Shown in Application Navigator

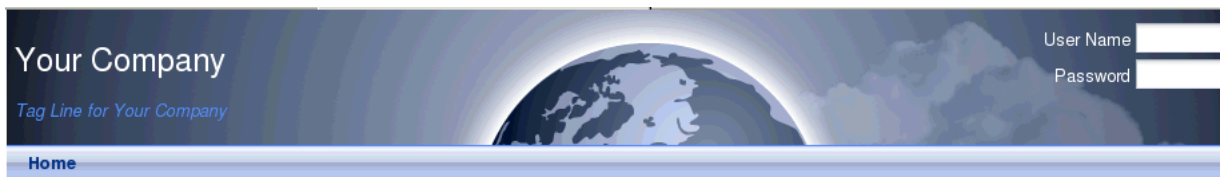


12. Right-click the **Portal** project and select **Run** to run the application.

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 3–11](#).

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 3–11 The Default Home Portal Page in a Web Browser



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

In the upper right corner of the Home page, in the **User Name** field ([Figure 3–12](#)), 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 3–12 Enter User Name and Password To Log in to Home Page

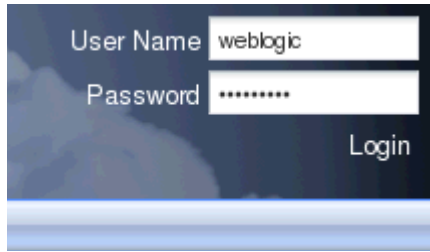


Figure 3–13 shows the portal Home page with Administrator privileges enabled after successfully logging in.

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 3–13 The Home Page with Administrator Privileges Enabled After Successful Login



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*.

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 3–15) and `pageTemplate_swooshy.jspx` (Figure 3–16).

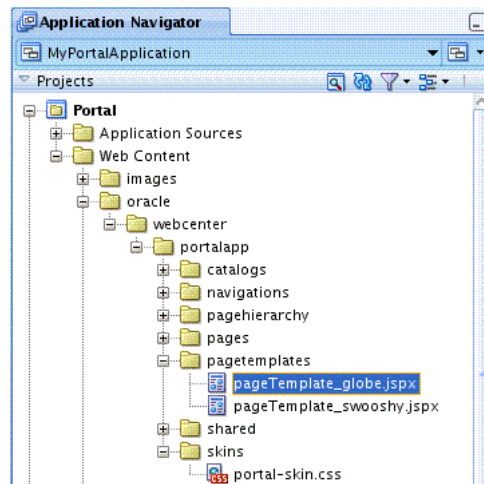
Both templates offer essentially the same functionality but with a different set of graphics.

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 3-14.

Figure 3-14 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 3-15.

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

Figure 3-15 The `pageTemplate_globe.jspx` Seeded Page Template

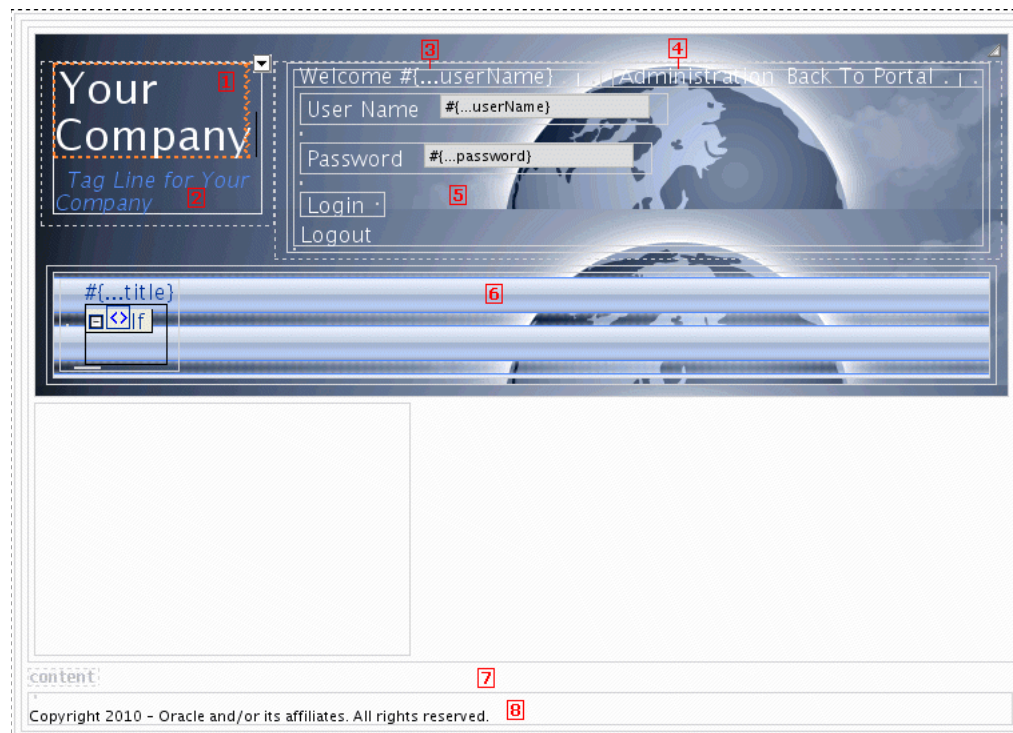
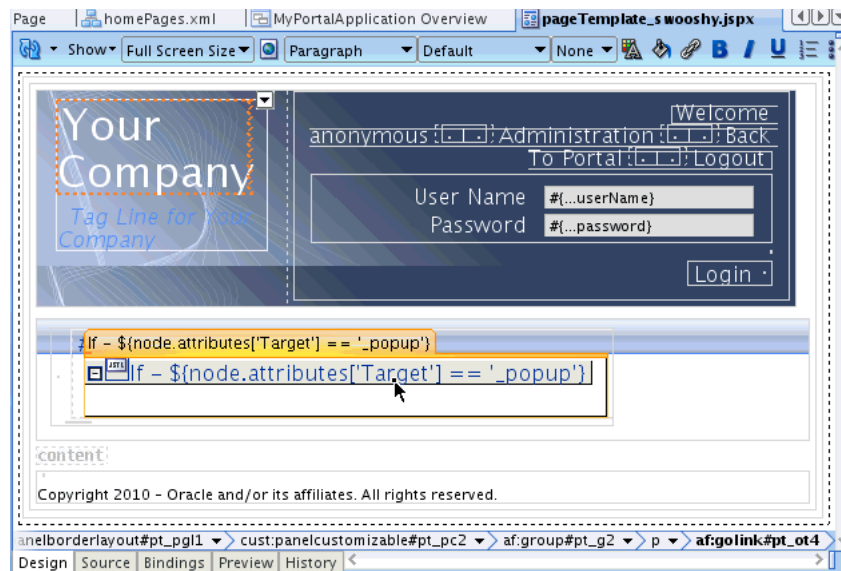


Figure 3–15 shows the `pageTemplate_globe.jspx` page template with each of its page features and artifacts (enumerated below) called out with their corresponding numbers in the illustration.

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 3–16 shows the `pageTemplate_swooshy.jspx` page template, but without each of its page features and artifacts enumerated.

Figure 3–16 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.

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 that goal, you will need to create the page template, then extract the setup files provided with this Tutorial from a folder residing on your hard drive. The folder contains a batch of files with graphic images, skins and templates. You will then copy these files to their respective folders in your application project and replace the existing *swooshy* page template, with a new page template that is provided.

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: Set the New Template and Create a Portal Resource](#)

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.

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*.

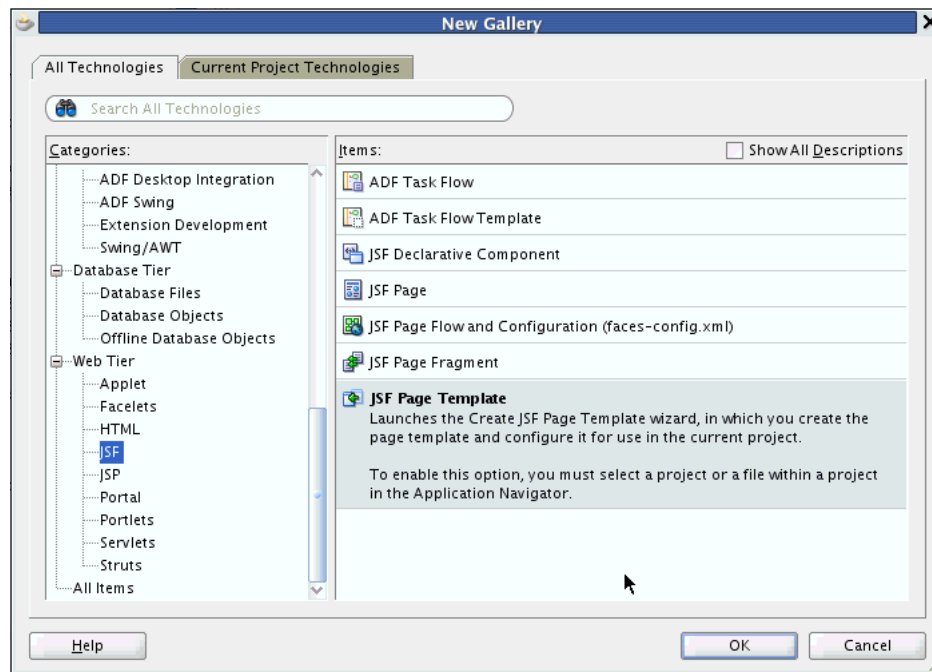
To create a new page template in our portal application:

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**.

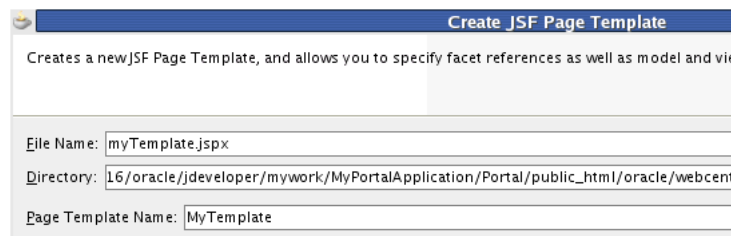
Figure 4-1 The JSF Page Template Selected in the New Gallery



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`.

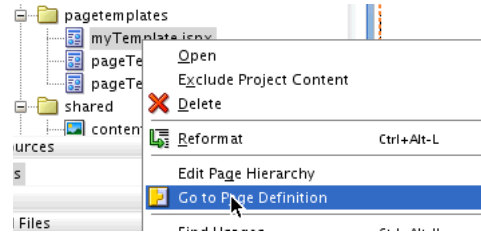
The file name identifies the page template in the Application Navigator.

Figure 4-2 The Create JSF Page Template



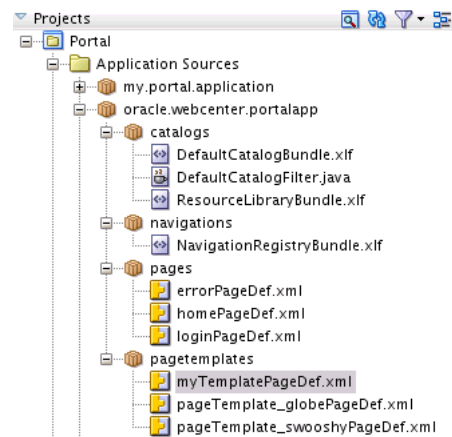
4. In the **Directory** field (Figure 4-2), enter the full directory path of the location under which to create the page template.
5. In the **Page Template Name** field (Figure 4-2), enter the display name for the page template, in this case `MyTemplate`.
6. Navigate in the Application Navigator to 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



7. When the dialog **Confirm Create New Page Definition** appears, click **Yes**.
8. Verify that the `myTemplatePageDef.xml` file now resides in the Application Sources sub folder `pagetemplates`, as shown in Figure 4-4.

Figure 4-4 The `myTemplatePageDef.xml` File in the Portal Application Sources Directory



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 "Designing the Look and Feel of Your Portal" in *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*.

Step 2: Extract Setup Files and Replace the Existing Template

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

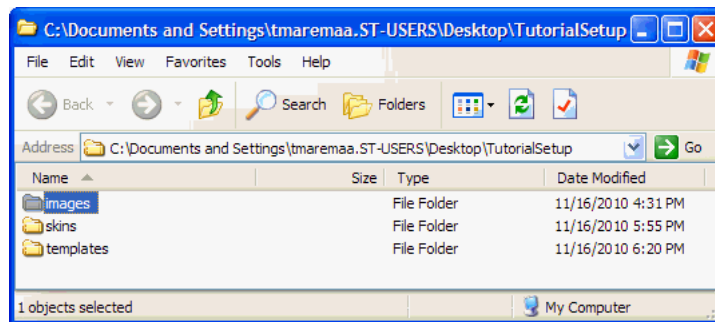
If you have not yet downloaded these Tutorial setup files, as described in [Section](#) , "Step 4: Download the Sample Tutorial Files," do so now. You can download the files from your web browser. The files are available at this URL address:

<http://www.oracle.com/technetwork/middleware/webcenter/documentation/webcentertutorialcontent-11120-128869.zip>

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

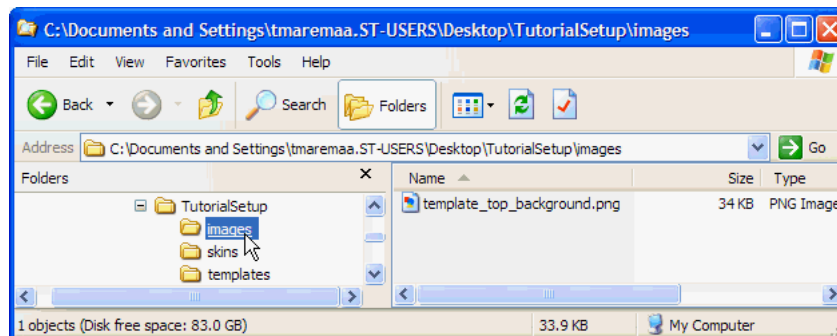
1. To begin with, you need to copy the TutorialSetup.zip file onto your hard drive in the directory of your choosing, and then proceed to unzip the files and extract their contents, as described in the following steps.
2. On your local drive navigate to, for example, C:\...\USERS\Desktop\TutorialSetup\images. Three folders reside in that directory: images, skins and templates, as shown in [Figure 4-5](#).

Figure 4-5 The Images, Skins and Templates Folders for Setup Residing on Your Local Hard Drive



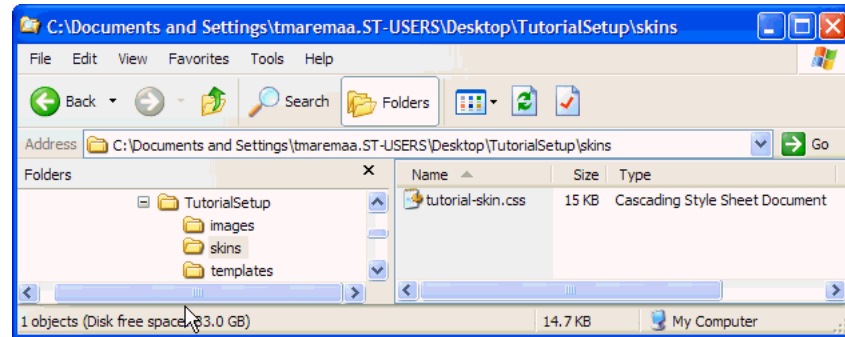
3. Extract the contents of the images folder ([Figure 4-6](#)) and move those contents to the MyApplication/Portal/public_html/images folder in your portal application.

Figure 4-6 The Expanded Images Folder in the Tutorial Setup Directory



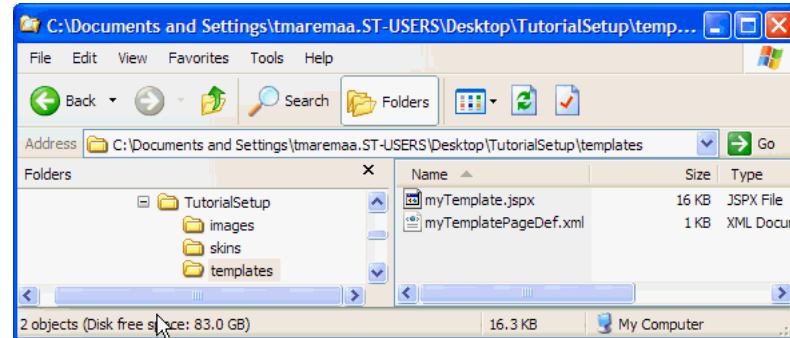
- Repeat the same procedure for contents of the skins folder (Figure 4-7), moving those contents to the `MyPortalApplication/Portal/oracle/webcenter/portalapp/skins/` folder in your portal application. Note that the extracted skin is a Cascading Style Sheet (CSS) document.

Figure 4-7 Expanded Skins Folder in the Tutorial Setup Directory



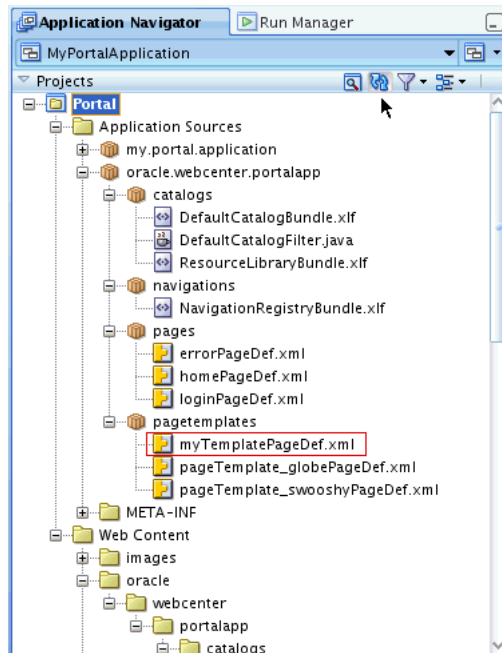
- Repeat again the same procedure for the contents of the templates folder (Figure 4-8), moving those contents to the `MyPortalApplication/Portal/oracle/webcenter/portalapp/templates/` folder in your portal application.

Figure 4-8 Expanded Templates Folder in the Tutorial Setup Directory



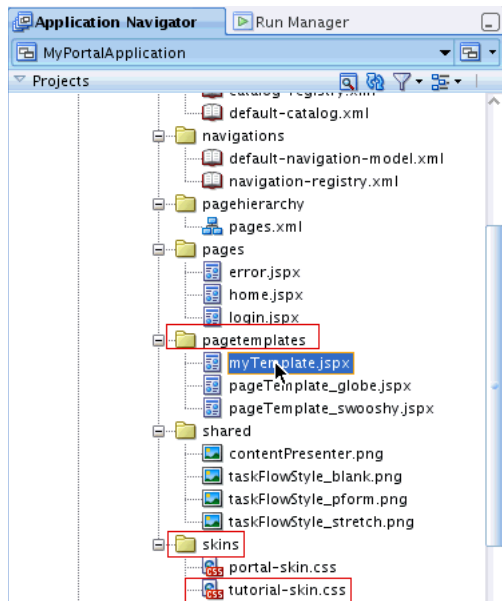
- Now select the Portal folder at the top level of your Project and click the **Refresh** icon. This will refresh and save each of the folders whose contents you have extracted and copied to your portal application in JDeveloper, as shown in Figure 4-9.

Figure 4–9 The Portal Hierarchy Refreshed to Include the Extracted Files for Setup



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 4–10](#).

Figure 4–10 The `myTemplate.jspx` file and `tutorial-skin.css` File in the Portal Project Folders



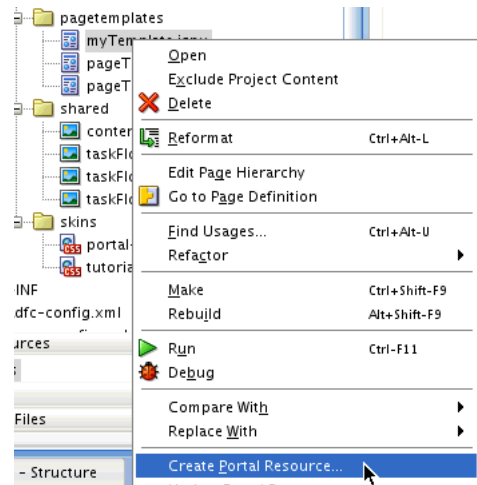
8. Close the Portal Application Sources folders and navigate to the `webcenter` folder in your project directory.

Step 3: Set the New Template and Create a Portal Resource

In this next sequence of steps, you will set the new template and the new tutorial skin that you've copied into your project, and then create a portal resource at design time to customize both the site template and the newly provided skin.

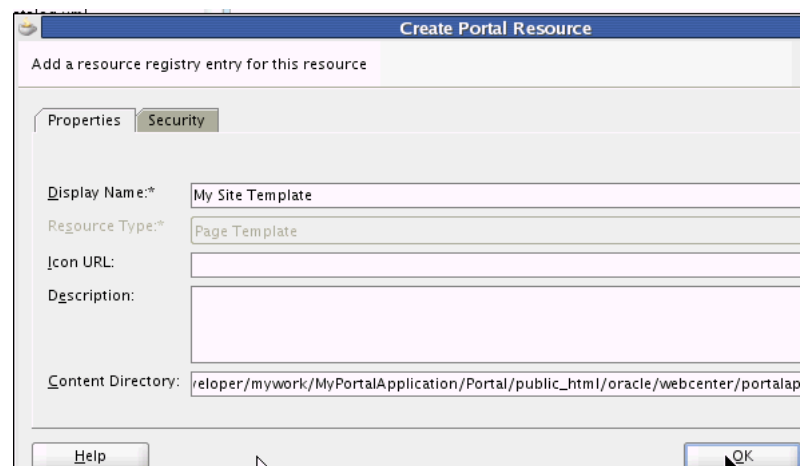
1. Open the `webcenter` folder in your portal project and navigate to the `pagetemplates` folder in the directory.
2. Select the `myTemplate.jspx` file and right-click the file.
3. Select the **Create Portal Resource** menu item, shown in [Figure 4-11](#).

Figure 4-11 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 4-12](#)) and click **OK**.

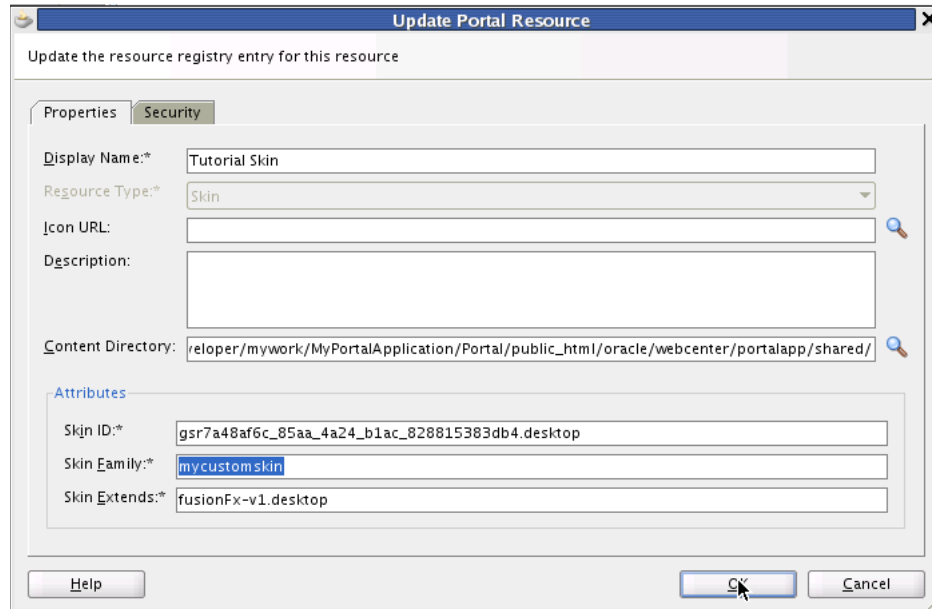
Figure 4-12 The *Create Portal Resource* Dialog with the `Display Name` Specified as `My Site Template`



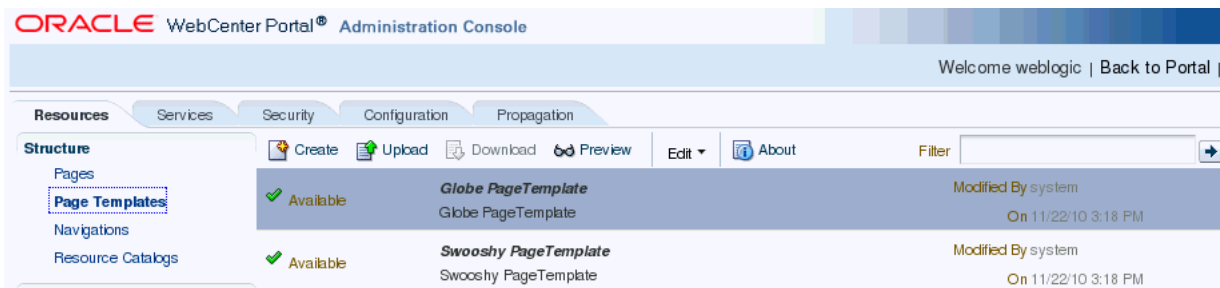
5. Navigate to the `skins` folder and open it. Select the `tutorial-skin` file and right-click the **Create Portal Resource** menu item, as performed in the previous step.

6. Change the Display Name to `Tutorial Skin`, and in the Skin Family field, enter `mycustomskin`, as shown in [Figure 4-13](#).

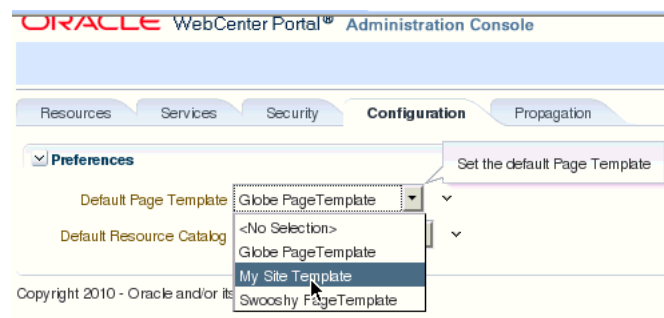
Figure 4-13 The Update Portal Resource Dialog with `mycustomskin` Specified as Skin Family Attribute



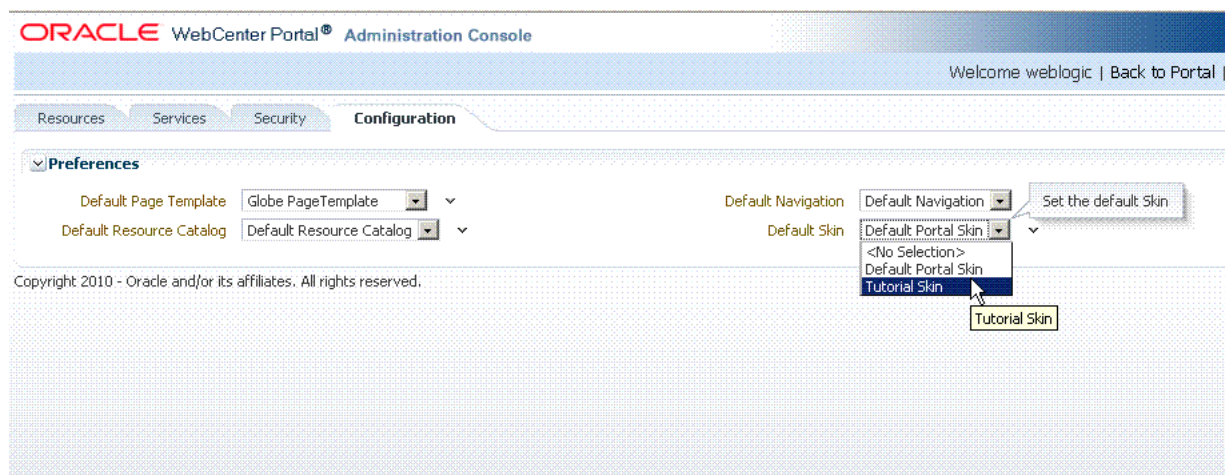
7. Click **OK**.
8. Navigate up the directory and click the **Refresh** button in the Application Navigator to refresh the contents of the skins folder.
9. Now select the **Portal** project in Application Navigator and right-click **Run** to run the portal application in JDeveloper. The portal displayed in the web browser shows the original template, with its default skin and standard portal application look-and-feel.
10. In the default Home portal page in the web browser, log in as `weblogic` (which enables you to have administrative privileges) and enter `welcome1` 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
11. After logging in, click the **Administration** link in the upper right corner of the browser window.
12. When the Administration Console opens, select the **Resources** tab, and navigate to the Page Templates item in the **Structure** menu, as shown in [Figure 4-14](#).

Figure 4–14 The My Site Template as a Designated Resource in the Administration Console

13. In the Administration Console, navigate to the **Configuration** tab and select it. From the **Look and Layout** menu, navigate to the skins item in the list. Open and select the tutorial skin.
14. In the Default Page Template menu, select My Site Template as the default Page Template, as shown in Figure 4–15.

Figure 4–15 The Default Template Changed to My Site Template

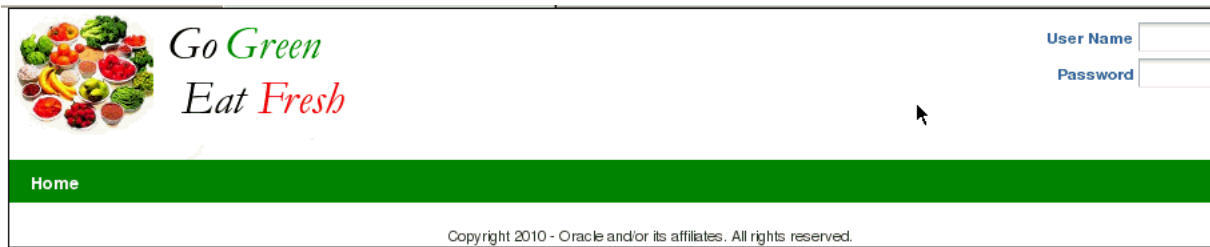
15. Now set the Default Portal Skin to Tutorial Skin (Figure 4–16).

Figure 4–16 The Default Portal Skin Changed to Tutorial Skin

16. Select the **Back to Portal** link in the Administration Console and click it.
17. Refresh the web browser page to reload the new tutorial skin.

Now the **Go Green, Eat Fresh** Home page appears, as shown in [Figure 4-17](#).

Figure 4-17 *The Home Portal in a Web Browser with a New Tutorial Skin Applied*



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.

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've learned 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 change 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.

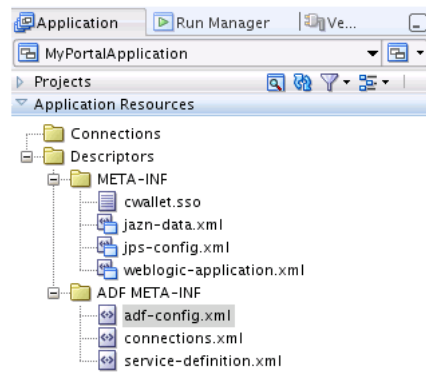
To change the default preferences of your portal application at design time, you will need to directly edit the `adf-config.xml` file in your project. The steps to accomplish this task are described in this section, as follows.

To change the default settings for the skin and template:

1. Open `adf-config.xml`.

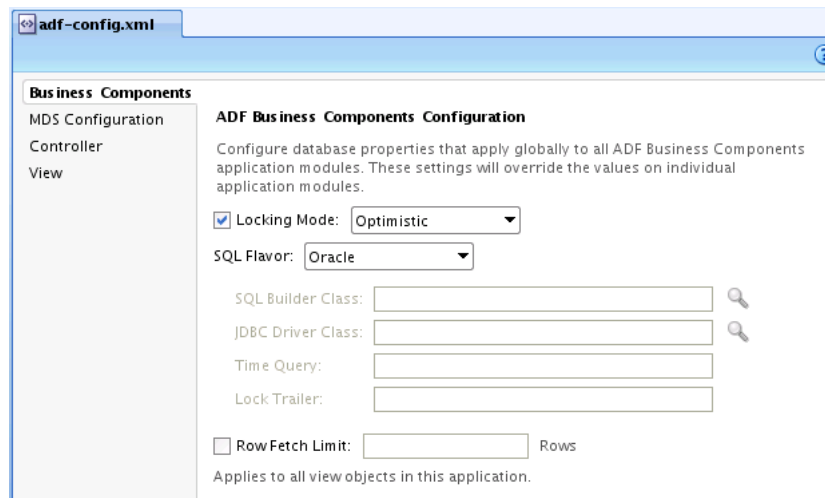
To locate this file in JDeveloper, open the **Application Resources** part of the Application Navigator. Then, open the **Descriptors** folder and the **ADF META-INF** folder, as shown in [Figure 5-1](#).

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



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. Return to the **Source** view of the `adf-config.xml` file and note that the value attribute is now updated as "mycustomskin", as shown in [Figure 5-3](#).

Figure 5-3 The Portal Skin Value Attribute Changed to "mycustomskin"

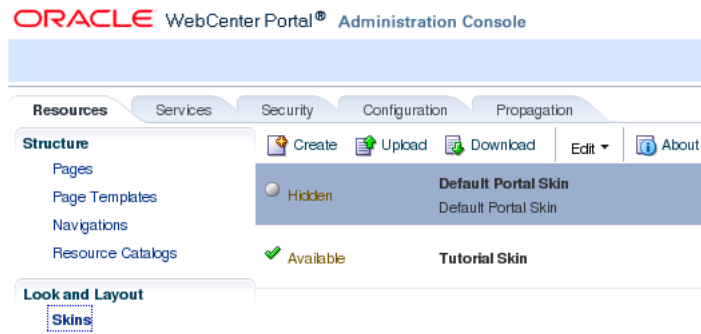
```
display="true"/>
<portal:preference id="oracle.webcenter.portalapp.skin"
  desc="Default Portal Skin" value="mycustomskin"
  resourceType="Skin" display="true"/>
```

8. Change the desc attribute from "Default Portal Skin" to "Tutorial Skin", shown in [Figure 5-4](#).

Figure 5-4 The Changed desc Attribute to Tutorial Skin

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

9. Now return to the MyPortalApplication project, select the Portal project in Application Navigator and right-click **Run** to build and launch the application in JDeveloper.
10. When the Home page appears in a web browser, login as User `weblogic` and Password as `weblogic1` to login and enable Administrator privileges.
11. In the Administration Console, click the **Resources** tab and click the **Skins** item in the Look and Layout list. Note that the Default Portal Skin is unchecked, while the Tutorial Skin is checked and now available ([Figure 5-5](#)).

Figure 5–5 The Tutorial Skin Checked as Available as a Skin Resource

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*.

Step 2: Change the Default Page Template at Runtime

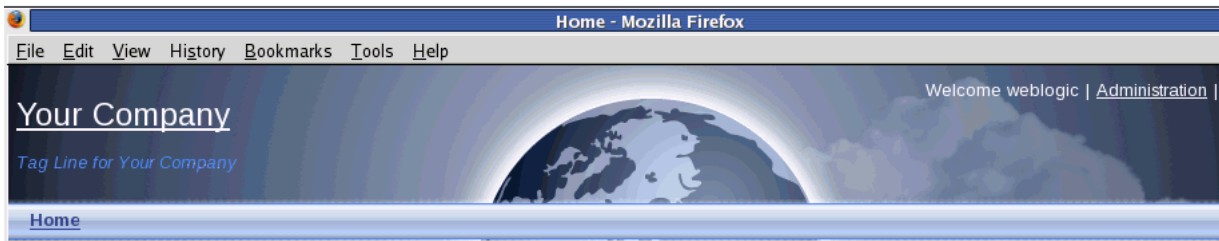
The steps that follow describe how you can change the default page template at runtime. You accomplish this by logging in as user with administrative privileges and clicking the **Administration** link in the web browser to access the Administration Console. Once in the Administration Console, you will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

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

1. Return to the Application Navigator in JDeveloper. Right-click the Portal project and select **Run** to run the application. The application opens in your web browser, as shown in [Figure 5–6](#).

Figure 5–6 The Home Portal Page with Administration Privileges Enabled

2. In the default Home portal page in the web browser, log in as `weblogic` and `welcome1` as your password.

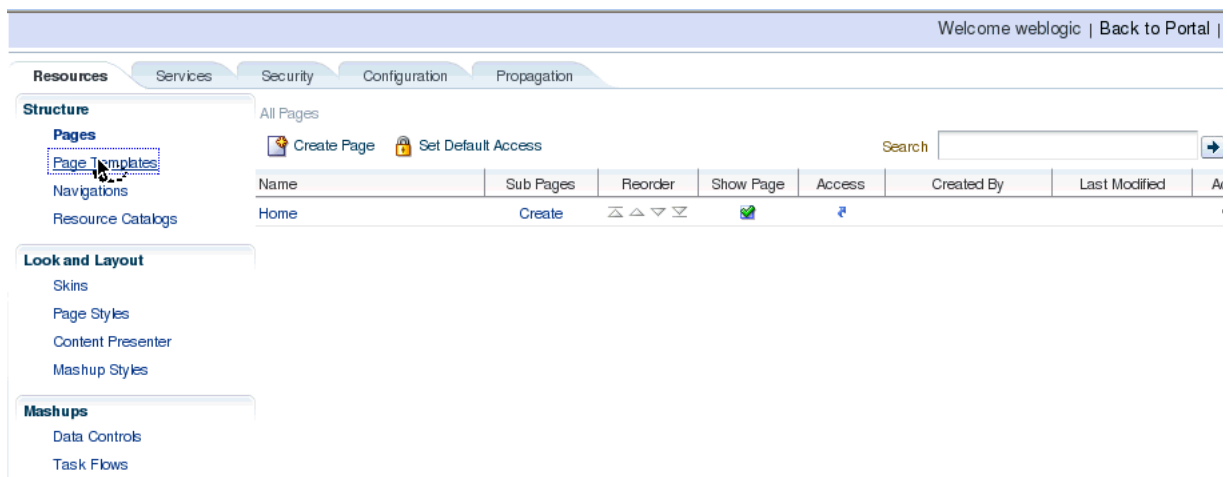
Note that you must log in as a user with administrative privileges. In the Tutorial, the user "weblogic" has administrative privileges.

3. Click the **Administration** link in the upper right corner of the web page. The Administration Console appears (Figure 5–7).

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.

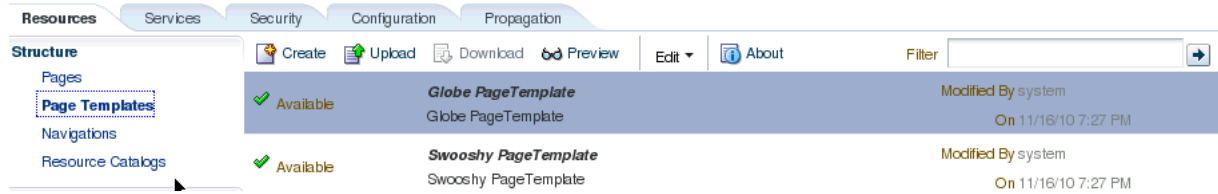
By adding the Resource Manager to a page in your portal, you enable portal administrators to manage these resources at runtime. Using the Resource Manager, portal users can also download resources, or an entire application, from the runtime environment, edit them in JDeveloper, and then upload them back into the deployed application.

4. Select the **Resources** tab and navigate to the Page Templates item in the **Structure** menu, as shown in Figure 5–7. Note that the **Propagation** tab will only appear if you have defined the appropriate connection for propagating from stage to production.

Figure 5–7 Administration Console with the Resource Tab and Page Templates Item Selected

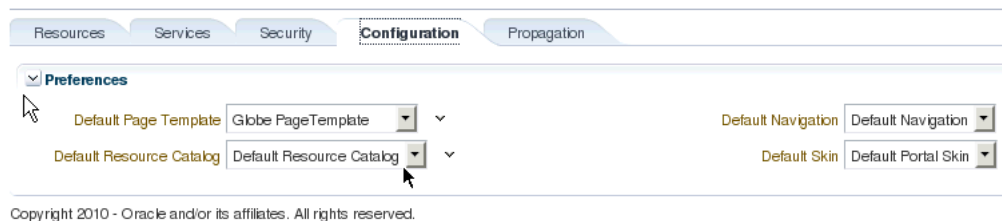
- Click the **Page Templates** item. Both the *Globe PageTemplate* and the *Swooshy PageTemplate* appear, as shown in [Figure 5-8](#), with the *Globe PageTemplate* field highlighted.

Figure 5-8 *The Globe PageTemplate selected in the Page Templates item*



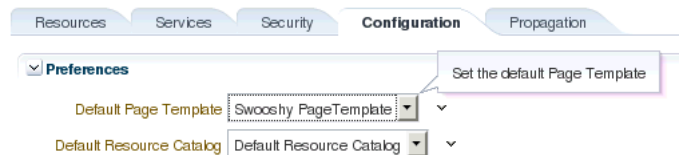
- Select the **Configuration** tab in the Administration Console. In the **Preferences** menu, the *Default Page Template* is specified as the *Globe PageTemplate* ([Figure 5-9](#)).

Figure 5-9 *The Globe Template Specified as the Default Page Template in Preferences*



- Change the *Default Page Template* and set it to the *Swooshy PageTemplate*, as shown in [Figure 5-10](#).

Figure 5-10 *The Default Page Template Changed to Swooshy Page Template*



- Click the **Back to Portal** link and return to the Application Navigator in JDeveloper.

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.

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 Universal Content Management (UCM) repository with access provided to the Oracle Content Server. You will set UCM as your primary connection and navigate to the WebCenterTutorial directory, where HTML content files for your application, like **About Us**, **Contact Us**, **Home** and **Menu**, are stored in 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 UCM, 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 in UCM.

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



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*. For more information about

managing content repositories and UCM, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Introduction

This lesson contains the following steps:

- [Step 1: Connect to Universal Content Management \(UCM\) Repository](#)
- [Step 2: Add a Content Item to the Navigation Model](#)
- [Step 3: Take Advantage of Iterative Development](#)
- [Step 4: 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 Universal Content Management (UCM) Repository

Oracle JDeveloper enables you to manage and handle document content stored on the Oracle Content Server by creating a connection to the content repository, in this case to UCM. 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 to Connect to UCM

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	stan118.us.oracle.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

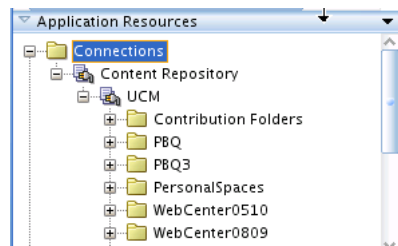
3. In the Connection Name field, enter UCM, then in the Repository Type field, enter Oracle Content Server. Ensure that you check **Set as primary connection for Documents service**.
4. In the Configuration Parameters pane (Figure 6–2), enter the parameters and values shown in Table 6–1. The Server Host Name should be your server host.

Table 6–1 Configuration Parameters and Values

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

5. Ensure that you check **Specify login credentials** (Figure 6–2) for the current JDeveloper session.
6. In the User Name field, enter weblogic and in the Password field, enter weblogic1.
7. Click **Test Connection**, and if successful, click OK.
8. Return to your MyPortalApplication in Application Navigator and navigate to the Application Resources folder. Select and open the Connections folder, as shown in Figure 6–3.

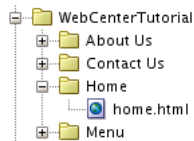
Figure 6–3 The Connection to the Content Repository and UCM established



Note the expanded Connections folder. You are now connected to the UCM repository.

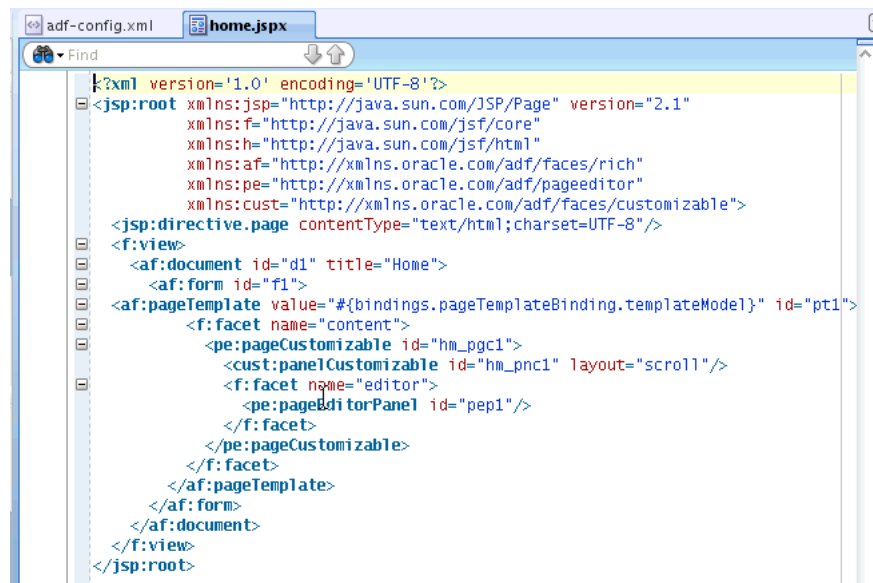
- Open the Contributions folder and navigate to the WebCenterTutorial folder, then open the Home folder with the home.html file shown in Figure 6–4.

Figure 6–4 The Home Folder Opened with the home.html file Shown in the WebCenterTutorial directory



- Return to your portal project directory, navigate up to the pages folder and select the home.jspx file. Click to open the 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



- 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.

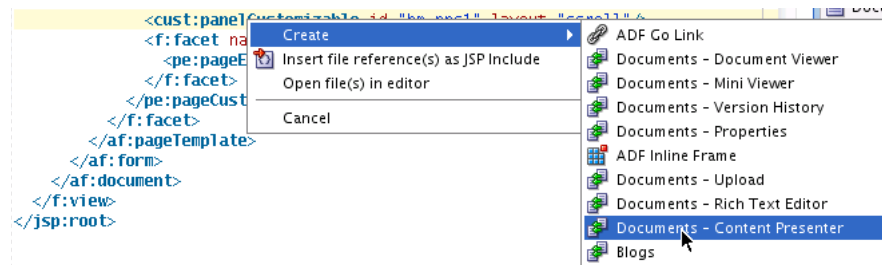
Figure 6–6 The Customizable Panel XML Code Snippet

```

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

```

12. In the WebCenterTutorial folder, open the Home folder and select the home.html file.
13. 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 right-click the selected code snippet to add another line of code in the next step. A pop-up list appears, with **Create** at the top of the list.
14. Choose the **Create** menu item, then scroll down the sub menu list to select the **Documents - Content Presenter** item, as shown in [Figure 6–7](#).

Figure 6–7 The Create Documents - Content Presenter Menu Item Selected

15. Drag and drop the **Documents - Content Presenter** task flow into the panelCustomizable component in the home.jspx file, adding a facet region specifying a value attribute for your task flow binding in the XML code, as shown in [Figure 6–8](#). In so doing, you are rendering the content stored in your UCM repository through the Content Presenter task flow.

Figure 6–8 The Documents - Content Presenter Task Flow Added as a Region

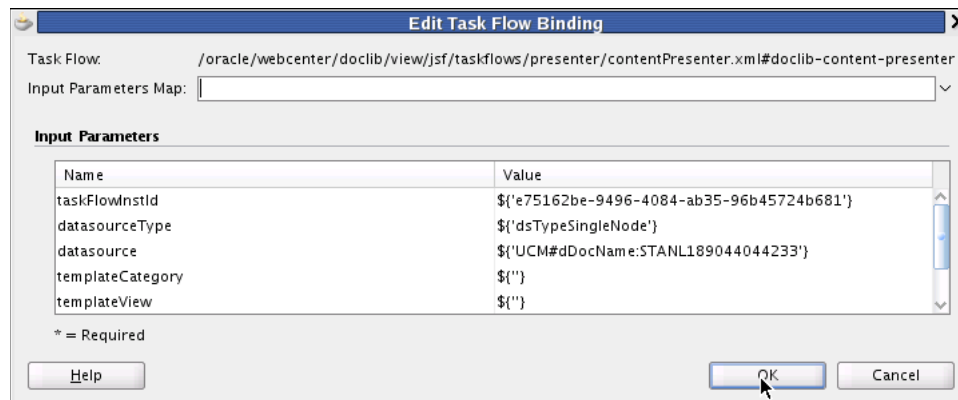
```

<f:facet name="content">
  <pe:pageCustomizable id="hm_pgcl">
    <cust:panelCustomizable id="hm_pnc1" layout="scroll">
      <af:region value="#{bindings.doclibcontentpresenter1.regionModel}"
        id="r1"/>
    </cust:panelCustomizable>
  </pe:pageCustomizable>
</f:facet>

```

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



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. In your **MyPortalApplication** project, select `index.html`, and right-click **Run** to run the application in JDeveloper.
19. 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



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 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.

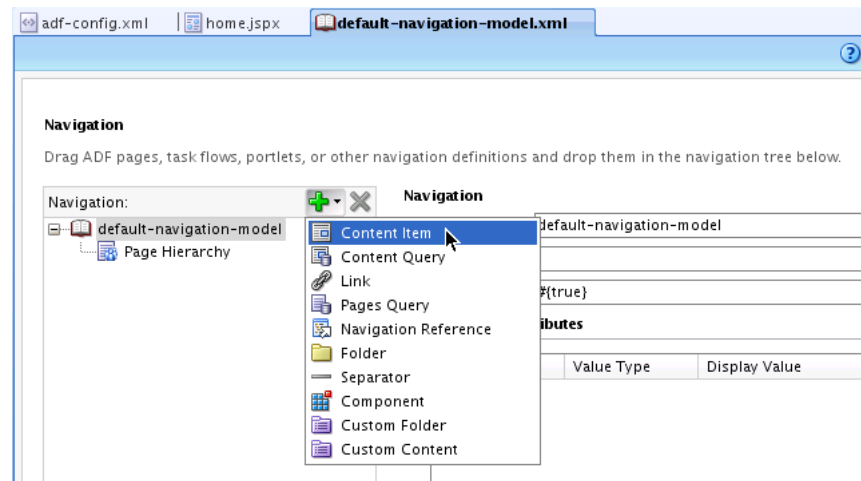
Step 2: Add a Content Item to the Navigation Model

Now that we have created a connection to the UCM content repository for our WebCenter portal application and dragged and dropped the `home.html` file as our Content Presenter, we can 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. Return to JDeveloper, select the `default-navigation-model.xml` file from the navigations folder, open it and click the plus icon to add an item, in this case, the **Content Item**, shown in [Figure 6–11](#).

Figure 6–11 Adding a Content Item to the Default Navigation Model

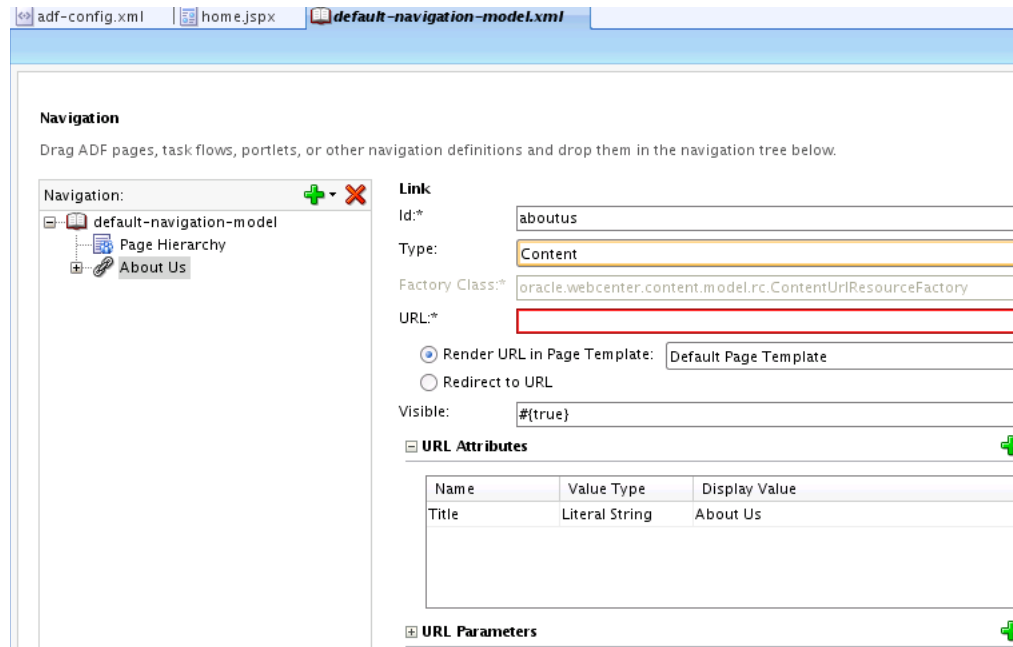


Tip: When you first create a WebCenter Portal application, the seeded navigation model, `default-navigation-model.xml`, is set as the default navigation model. The default navigation model provides a convenient way to select a navigation model that can be used by default by your application. Page template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

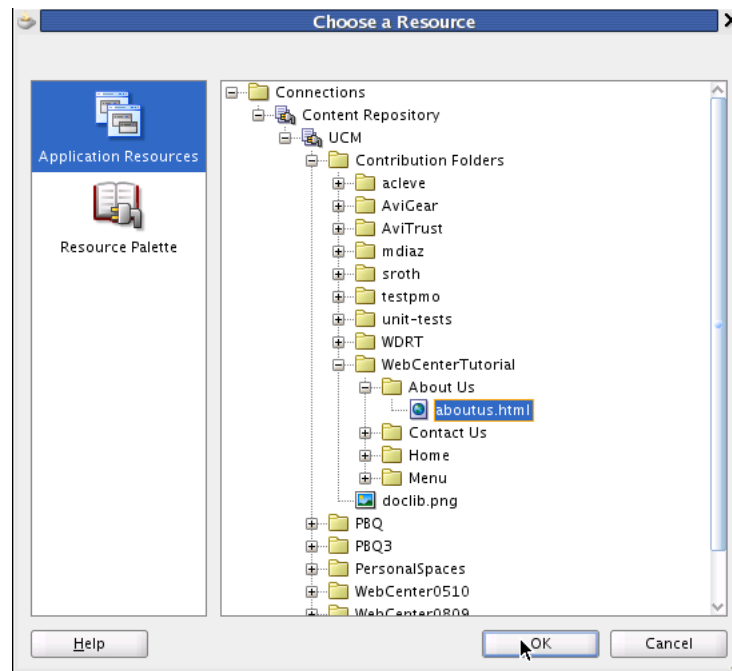
You can set the default navigation model for your application, if you want to create your own model, by editing the `oracle.webcenter.portalapp.navigation.model` preference in the `adf-config.xml` file.

2. 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 6–12.
3. Select the **About Us** item, enter in the Id field: aboutus. In the Type field, enter Content. In the URL Attributes pane, enter the Display Value as About Us.

Figure 6–12 Navigation and Link Specified with Display Value



4. Select the **URL** field and click the tool item (in the shape of a magnifying glass) at the far right of the field to browse the contents.
5. In the Choose a Resource dialog, open and expand the UCM sub folders, then navigate to the WebCenterTutorial folder and select the aboutus.html file shown in Figure 6–13.

Figure 6–13 The *aboutus.html* file Selected in the Choose a Resource Dialog

6. Click **OK**. The Resource appears in the URL field of the navigation window as a URL address.

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.

7. Save your files.
8. Now return to your web browser and reload the browser page. The **About Us** link appears in the navigation model next to **Home**, as shown in [Figure 6–14](#).

Figure 6–14 The *About Us* Link in the Web Browser of Your Portal Application

Step 3: 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.

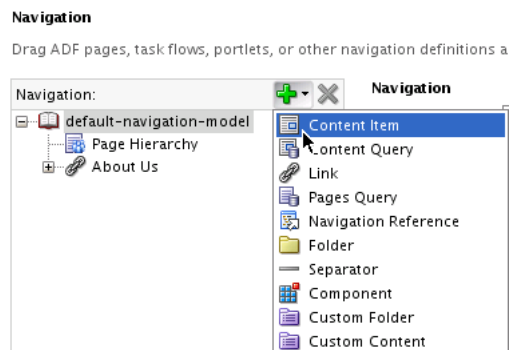
In the previous step [Section , "Step 2: Add a Content Item to the Navigation Model"](#), you added a new content link to your application, which appeared in the navigation hierarchy when you refreshed the contents of your web browser.

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:

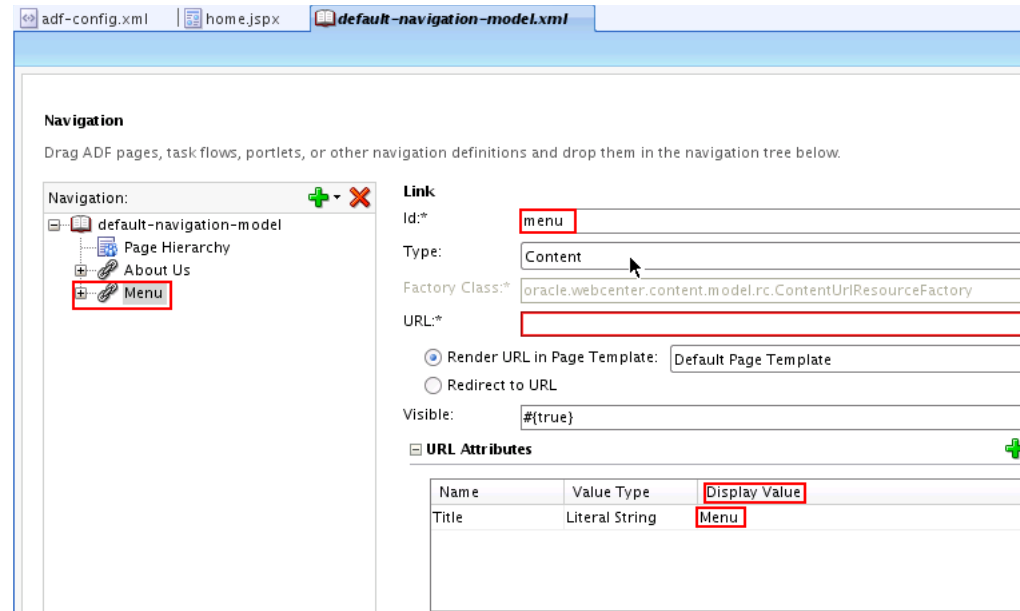
1. In Application Navigator, select `default-navigation-model.xml` file. Right-click the **Application Properties** menu item.
2. Select `WebCenter` in the **Run** node and ensure that **Enable Iterative Development** is checked. (Note that this is checked by default.)
3. Click **OK**.
4. In the Navigation, select `default-navigation-model.xml`, and click the plus icon to add a `Content Item` in the navigation, as shown in [Figure 6–15](#).

Figure 6–15 The Content Item Added to the Default Navigation Model XML file



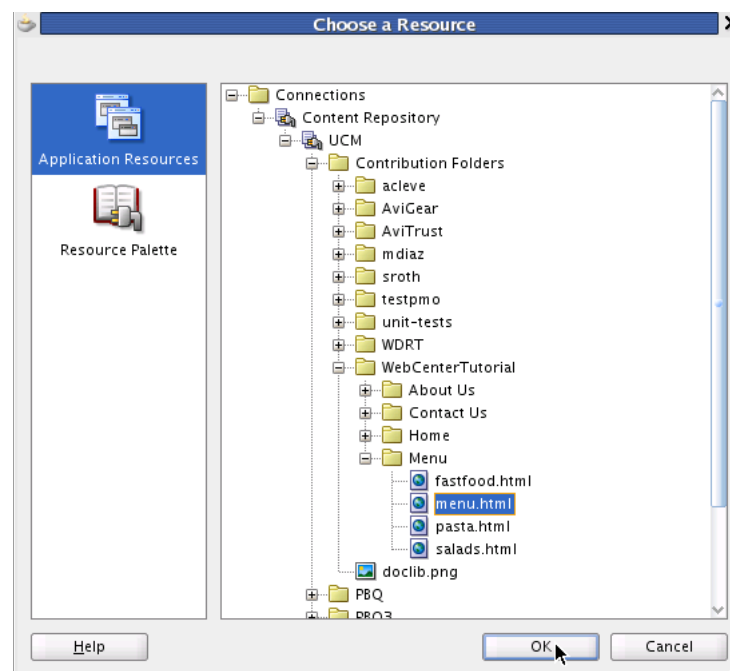
- The `contentItem` appears as a node in the `default-navigation-model.xml` navigation and in the `Id` field as `contentItem`. Change the `Id` to `menu` and the `Display Value` to `Menu`, as shown in Figure 6-16.

Figure 6-16 The `contentItem` Changed to `menu` with a New `Display Value`



- Choose a Resource. Right-click **Application Resources** and navigate down to the `Menu` folder, select `menu.html`, as shown in Figure 6-17.

Figure 6-17 The *Choose a Resource* Dialog with `menu.html` Selected



- Click **OK**.

8. 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 [Figure 6–18](#). This is the result of enabling the Iterative Development feature in your portal application.

Figure 6–18 The Refreshed Web Browser Page with the Menu Link Added in the Navigation Model



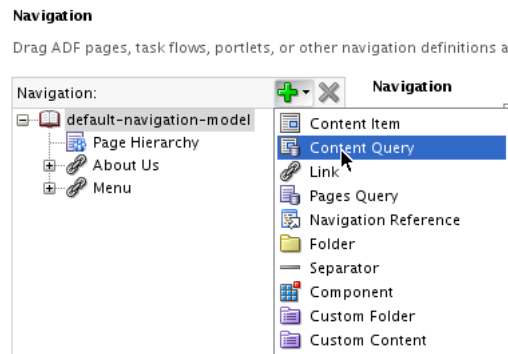
Step 4: Add a New Content Query

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 UCM.

To add a new content query:

1. Return to your portal application in JDeveloper and select the `default-navigation-model.xml` file.
2. Add a new link to the menu node. Select the **Content Query** menu item, as shown in [Figure 6–19](#).

Figure 6–19 Adding a New Content Query to the Default Navigation Model

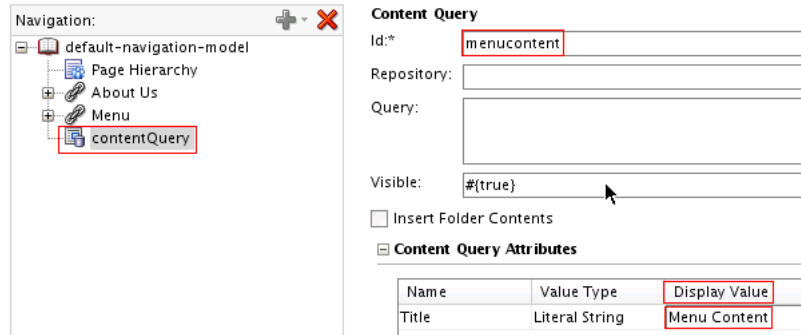


3. In the **Content Query** pane, enter in the Id field `menucontent`. In the **Content Query Attributes** pane, enter as the Display Value `Menu Content` ([Figure 6–20](#)).

Figure 6–20 The Content Query Pane with Id and Display Value Changed

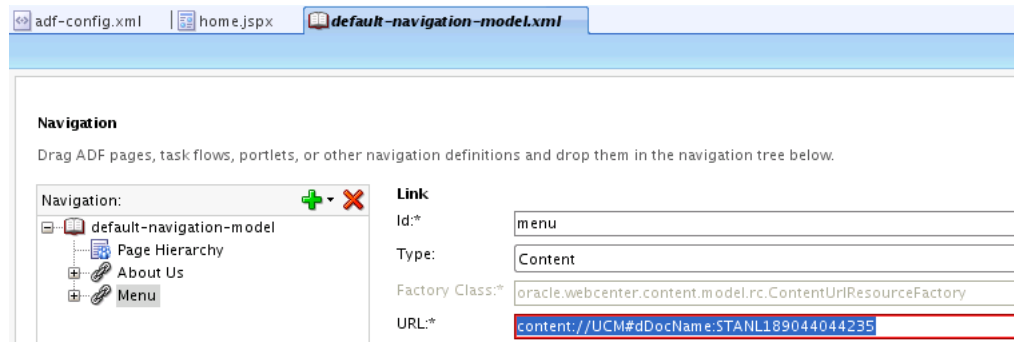
Navigation

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



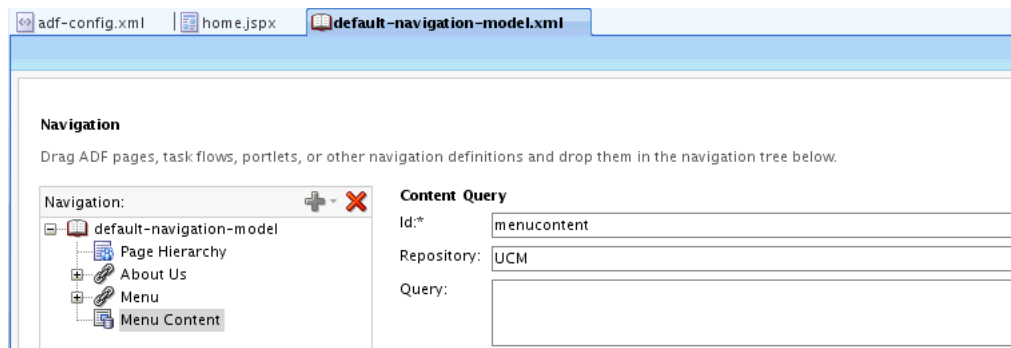
4. Select **Menu** in the navigation node, and click the browsing tool in the **URL** field. The Application Connection dialog appears. Select menu .html.
5. Return to the Navigation pane and note the updated **URL** field, as shown in [Figure 6–21](#).

Figure 6–21 The URL for the Menu Link Selected with the UCM Content



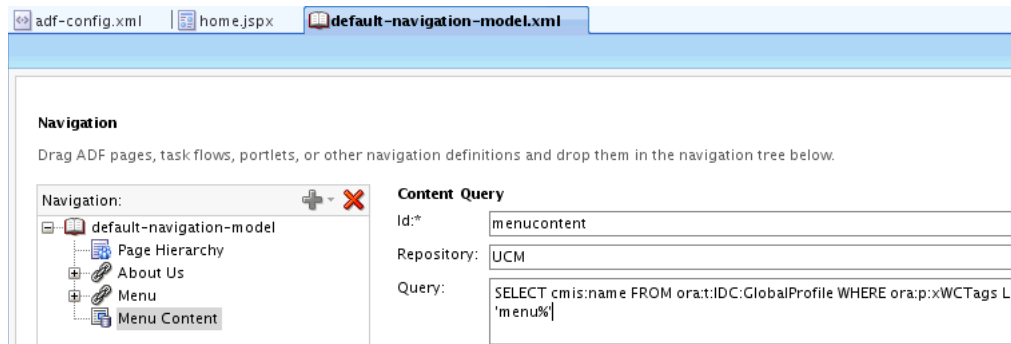
6. In the **Repository** field, click the tool which opens a Connections window, select UCM, and note that UCM appears in the **Repository** field, shown in [Figure 6–22](#).

Figure 6–22 The Repository Field with UCM Entered



7. Drag and drop the **Content Query** on the Menu Content node. 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.
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 6–23 The Menu Contact Content Query Text Entered



9. Ensure that you check **Insert Folder Contents**.
10. Save your changes.
11. Refresh the home page of your web browser.

The results of the query are shown under the **Menu** link as **Fast Food, Pasta and Salads** links, as shown in [Figure 6–24](#).

Figure 6–24 The Menu Item Links Listed in the Web Browser

In this lesson, you've learned the importance of working with and managing your document contents in the Universal Content Management (UCM) 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 UCM.

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.

Customizing Pages For Permissions and Runtime Editing

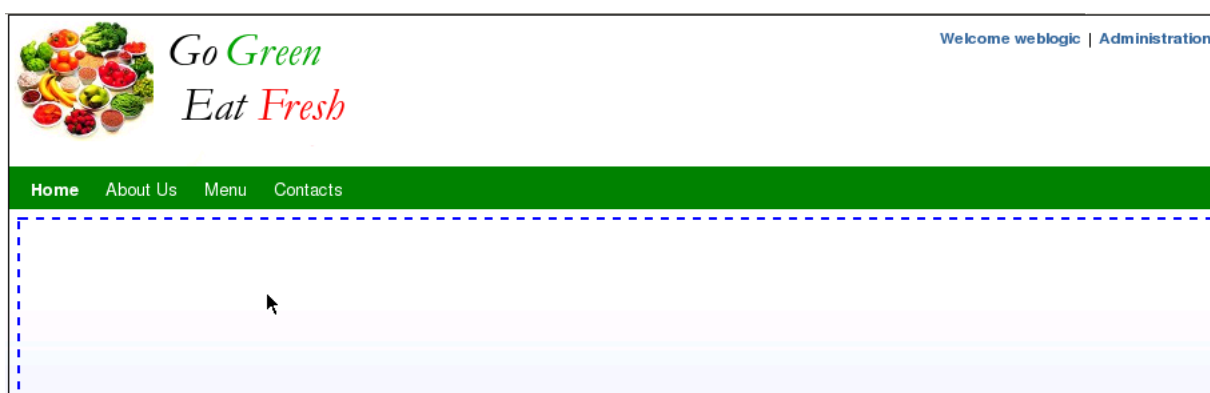
In this lesson, as a developer at design time, you will customize a specific page in the page hierarchy of your WebCenter Portal application, adding that page to the hierarchy so you can set permissions for user access and editing or modification of document content.

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 content editing is enabled at runtime.

Figure 7-1 The Portal Application Home Page Enabled for Editing at Runtime



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

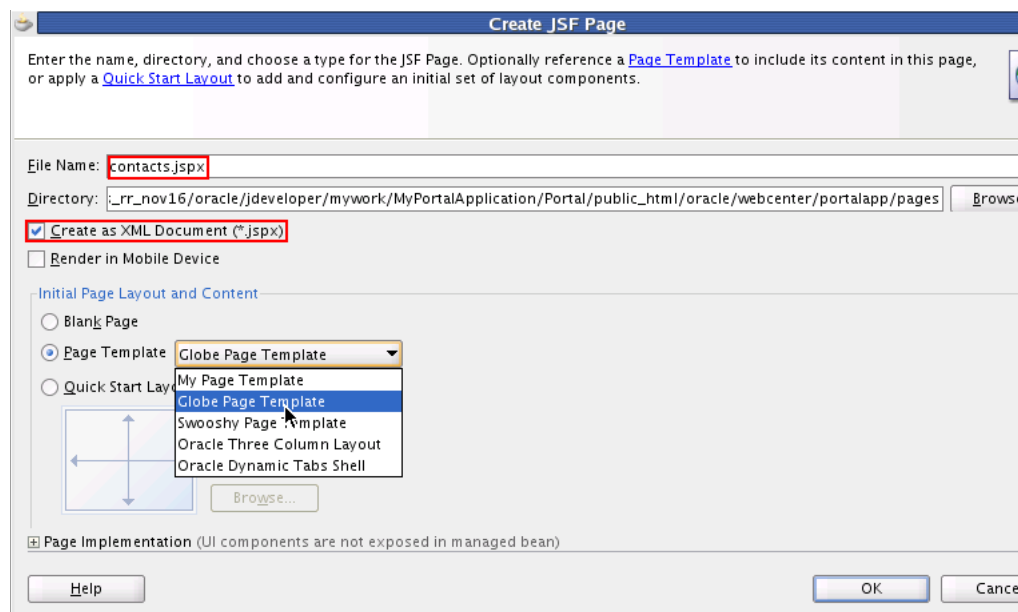
Building on the steps of the previous lesson ([Chapter 6](#)), we can now move ahead to customize specific pages in our WebCenter Portal application, thus setting page permissions and user access.

To accomplish this task, we first need to create a new `.jspx` page, in this case, `contacts.jsx`, 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](#).

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



3. Enter `contacts.jsx` in the File Name field and select **Globe Page Template** from the list of available templates as the Page Template.
4. Click **OK**.
5. Select `contacts.jsx` and open it in **Source** view.
6. Navigate to the code snippet shown in [Figure 7-3](#) and set the `value` attribute `value="#{bindings.pageTemplateBinding.templateModel}"`. Note that this is needed so the page picks up the template changes that occur at runtime.

Figure 7-3 The Value Attribute Selected in the contacts.jspx File

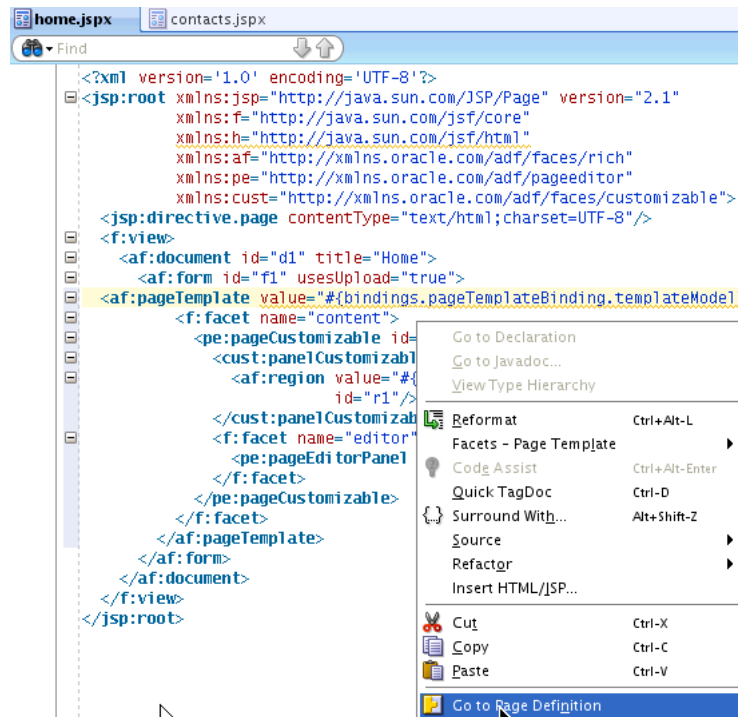
```
<af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}"
```

- Now in Application Navigator, select `home.jspx` and open it in **Source** view.
- Copy the code line shown in [Example 7-1](#) setting the `value` attribute which specifies page template bindings to the `contacts.jspx` file.

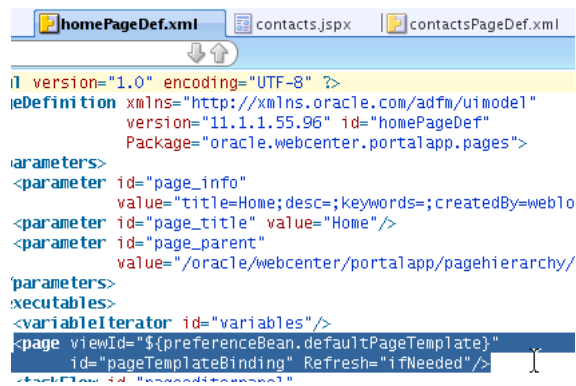
Example 7-1 The Code Snippet for the value Attribute

```
<af:pageTemplate value="#{bindings.pageTemplateBinding.templateModel}"
```

- In `home.jspx`, right-click and select the **Go to Page Definition** menu item, shown in [Figure 7-4](#).

Figure 7-4 Specifying the Page Definition in the home.jspx File in Source View

- Open the `homePageDef.xml` file in **Source** view and select the code snippet (the `viewId` attribute selected), as shown in [Figure 7-5](#).

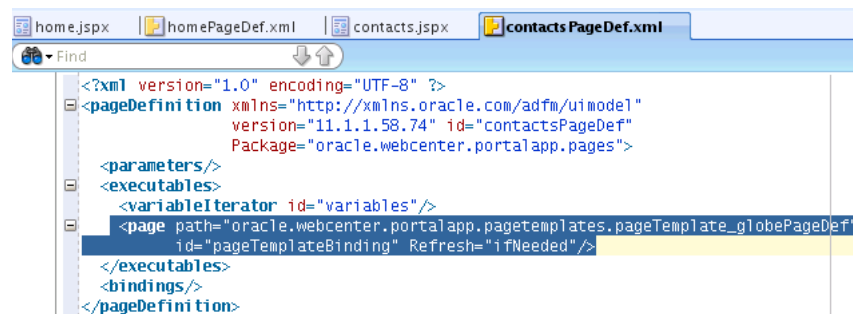
Figure 7-5 The homePageDef.xml File in Source View with the viewId Attribute Selected


```

<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uiModel"
  version="11.1.1.55.96" id="homePageDef"
  Package="oracle.webcenter.portalapp.pages">
  <parameters>
    <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>
  <executables>
    <variableIterator id="variables"/>
    <page viewId="{preferenceBean.defaultPageTemplate}"
      id="pageTemplateBinding" Refresh="ifNeeded"/>
  </executables>
</pageDefinition>

```

- Now open the `contactsPageDef.xml` file in **Source** view and select the code snippet shown in [Figure 7-6](#). The snippet specifies the `path` and `id` attributes for the contacts page definition file.

Figure 7-6 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>

```

- Select and copy the code snippet shown in [Example 7-2](#), which specifies the `viewId` attribute into the `contactsPageDef.xml` file.

Example 7-2 Code Snippet to Copy into contactsPageDef.xml

```

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

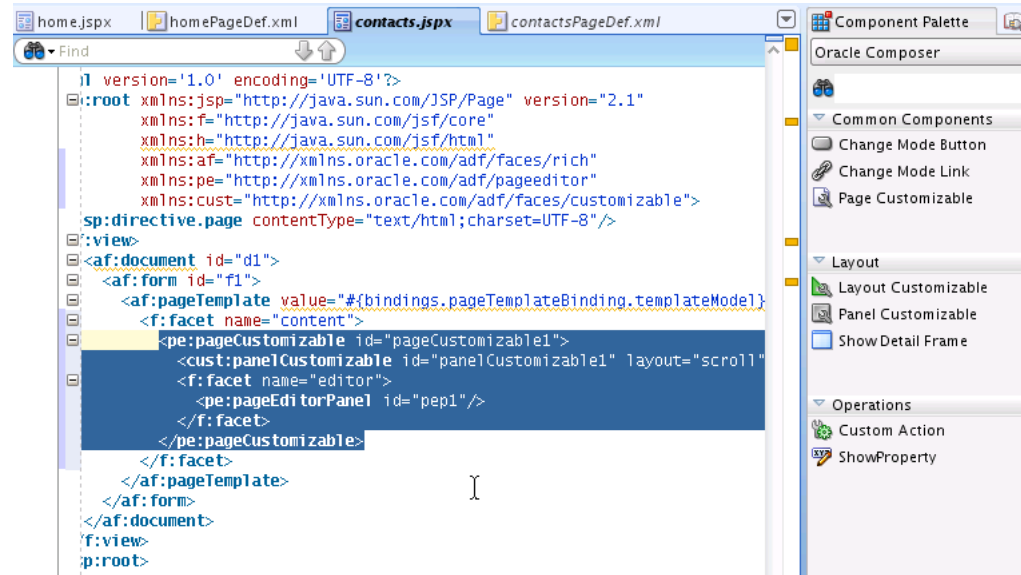
```

- In JDeveloper, in the **Component Palette**, select Oracle Composer. Navigate down to the **Page Customizable** component and drag and drop that component on to the `contacts.jspx` **Source** view.

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

[Figure 7-7](#) shows the code changed after dragging and dropping into the `contacts.jspx` file.

Figure 7-7 The New Code Snippet Dragged and Dropped into contacts.jspx



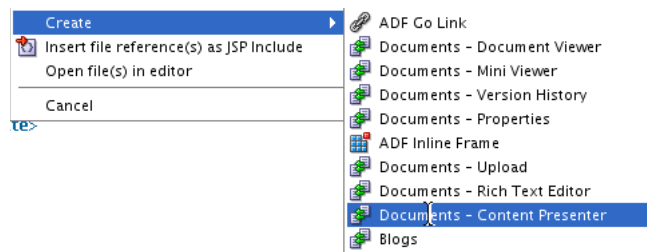
14. Now open the Contact Us folder in the UCM repository folder called WebCenterTutorial and select `contacts.jspx` to open it.
15. Right-click the `contacts.jspx` file and select the **Create** menu item and the **Documents - Content Presenter** task flow menu item shown in Figure 7-8.

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*.

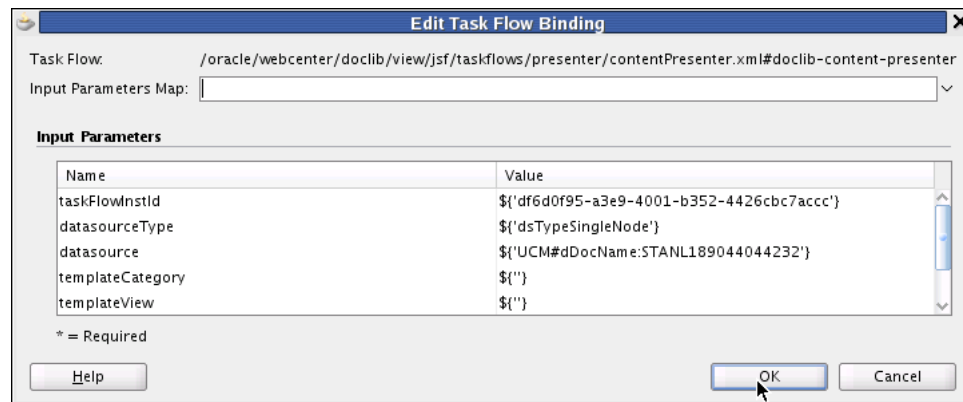
The **Documents** service enables you to display content from a content server or file system directly within your application. End users can then view and manage documents, as well as other types of content stored in content repositories.

Figure 7-8 The Documents - Content Presenter Task Flow Selected for Creation



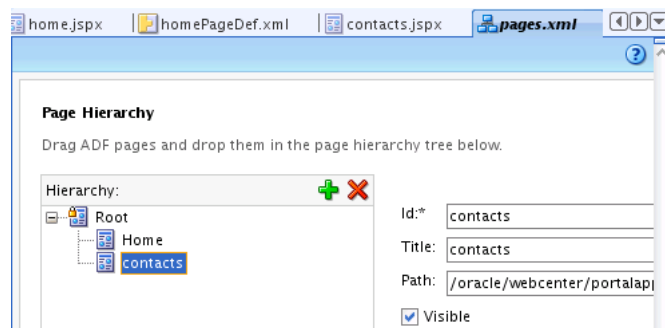
16. The Edit Task Flow Binding dialog appears, as shown in Figure 7-9. Note that in the Name field of Input Parameters, there is a UCM value for the data source.

Figure 7-9 The Edit Task Flow Binding Dialog with a Data Source Value as a UCM Document Name



17. Click OK.
18. Return to the portal application. Navigate to the pagehierarchy folder and select pages.xml.
19. When the pages.xml Page Hierarchy pane opens, drag and drop the contacts.jspx file into the Root node, and uncheck the Visible checkbox, which is shown as checked in Figure 7-10.

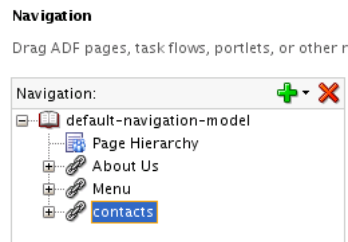
Figure 7-10 The pages.xml Page Hierarchy with Contacts Selected in the Root node



20. Now navigate to the default-navigation-model.xml folder and open the navigation dialog.

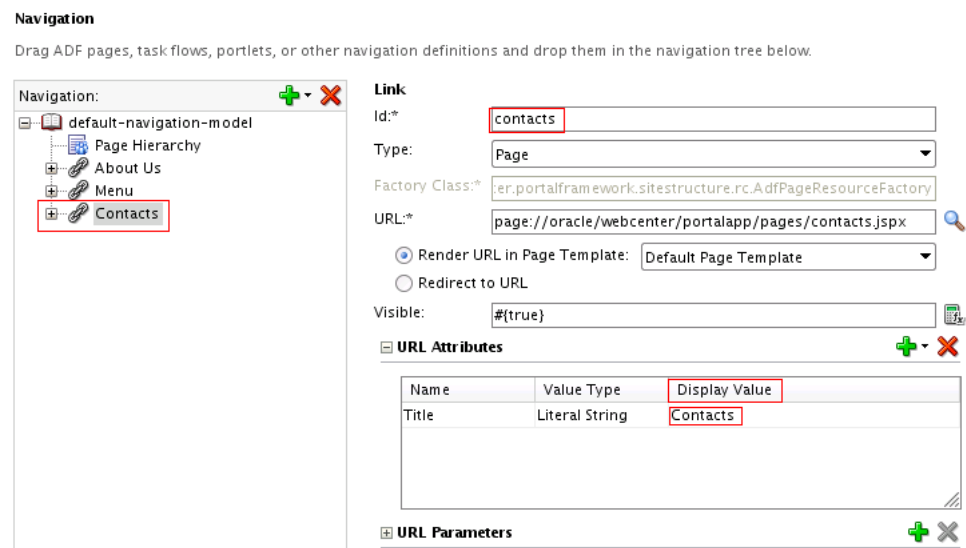
- Drag the `contacts.jspx` file to the `default-navigation-model` node (Figure 7–11), select the plus icon to add a link.

Figure 7–11 The Contacts Link Added in the Navigation Model



- In the Link ID field, enter `contacts`. In the URL Attributes pane, enter the Display Value as `Contacts`, as shown in Figure 7–12.

Figure 7–12 The Navigation Pane with the Link Id for Contacts and the Display Value Specified as Contacts



- 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 **Contacts** as part of the navigation model, as shown in Figure 7–13. The page will be populated with content using Oracle Composer, shown in Figure 7–14.

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

Figure 7-13 The MyPortalApplication shown in a Web Browser with the Contacts 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. Once the application opens in a web browser with **Contacts** as part of the navigation model, click the **Contacts** link in the banner to the right of the **Menu** link. The **Contacts** page displays the page on which you can enter document content, as shown in [Figure 7-14](#).

Figure 7-14 The Contact Page For Users to Enter Document Content



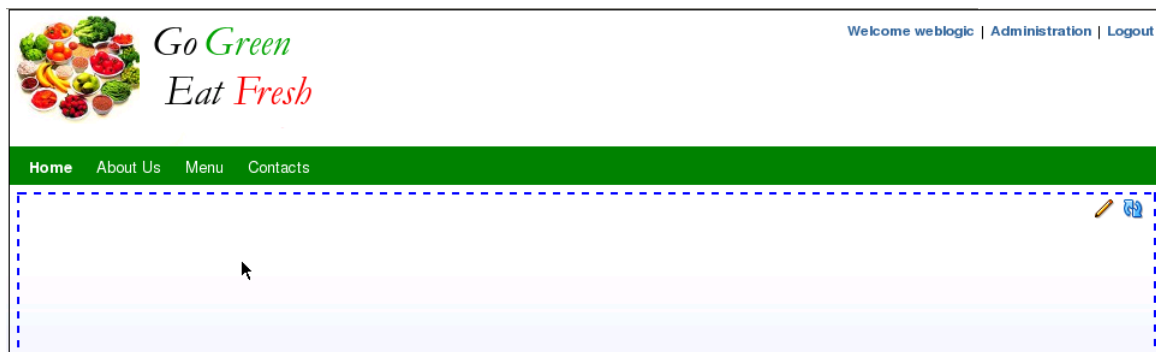
3. Login to the **Home** page as `weblogic` with `weblogic1` as your password (Figure 7–15).

Figure 7–15 Logging into the Home Web Page



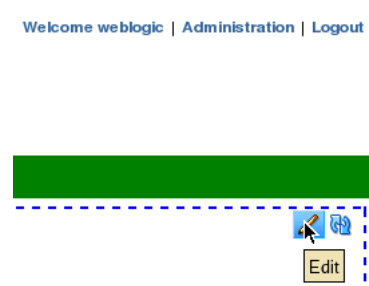
4. Once you have successfully logged in, click the **Administration** link in the browser. The **Home** page in the web browser is refreshed and the page appears at the center with a hash-marked outline and a pencil icon in the upper right corner, indicating that the outlined portion of the **Home** page can be edited for document content, as shown in Figure 7–16.

Figure 7–16 The MyPortalApplication in a Web Browser with Content Editing Enabled



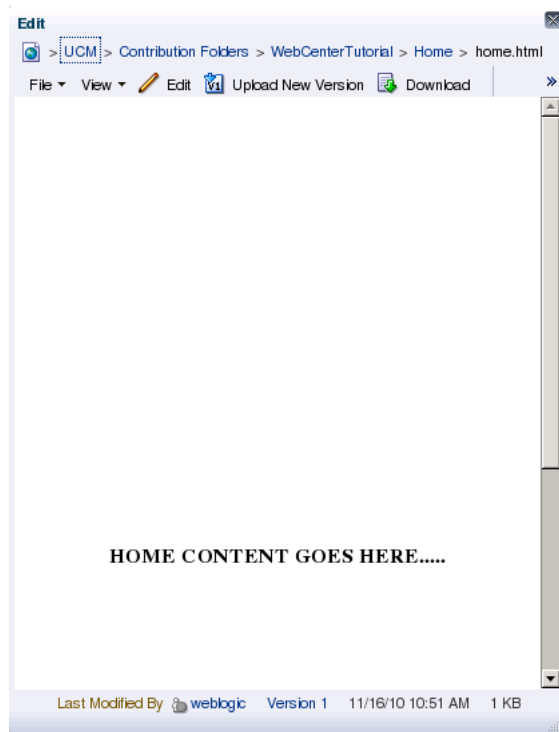
5. Navigate in the outlined portion of the browser page to the pencil icon and click the icon to enable editing, as shown in Figure 7–17.

Figure 7–17 The Pencil Icon Selected to Enable Editing of Home Page Document Content



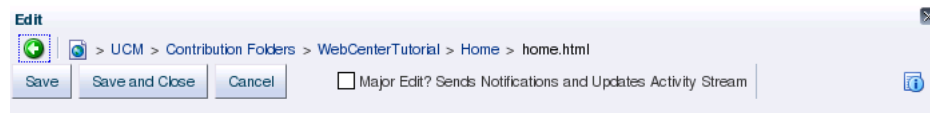
6. An Edit window pops up at the center of the page where authenticated users can add content to the `home.html` page and edit that content using a rich text editor, as shown in Figure 7–18. Note that the document content is stored in the UCM content repository for easy access and retrieval.

Figure 7–18 The Edit Window for In-context Editing of HTML Content



7. Users can then click the edit icon in the upper left top row and save their edits (Figure 7–19).

Figure 7–19 The Edit-Save Window for Saving of In-context HTML Document Content



In this lesson, you have learned how to customize specific pages in the page hierarchy of your WebCenter Portal 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.

Conclusion

Congratulations! You have created a WebCenter Portal 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 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 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 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 applications, including using Oracle Composer, WebCenter Services, and portlets, in the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter*.

To learn more about what you can do at runtime, including using Oracle 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 Spaces*.

You can find all Oracle WebCenter Suite 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 Suite, and view demonstrations and see examples of WebCenter Portal applications, portlets, and services in action on the Oracle WebCenter Suite home page on the Oracle Technology Network at:

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

Index

A

- Administration Console
 - selecting Resources, 4-8
 - selecting Resources tab, 5-4
 - values entered, 5-4
 - working with resources, services, security, and portal configurations at runtime, 5-5
- Administration link
 - in web browser, 7-9
- Administrator privileges, 3-8
- Application Navigator
 - Application Resources in portal project, 5-2
- applications
 - creating WebCenter applications, 3-1

C

- changing preference entries, 5-1
- content item
 - added to navigation model, 6-7
 - adding to default navigation model, 6-1
- Content Presenter
 - task flow, 7-1, 7-5
- Content Query
 - adding, 6-1
- content query
 - adding to portal application, 6-12
- content repository
 - access, 2-4
- custom WebCenter applications
 - creating, 3-1
- Customize
 - pages, 7-2

D

- default page template
 - changing at runtime, 5-4
- default-navigation-model
 - about, 6-7
- document editing at runtime, 7-8
- documents
 - drag and drop in HTML format, 7-1
 - edit at runtime, 7-8
- Documents - Content Presenter service

- creating task flow bindings, 6-1
- Documents service
 - about, 7-1, 7-5

E

- edit
 - adf-config.xml file to change default preferences, 5-2

G

- Globe Page Template, 7-2
- Globe page template, 3-7

H

- HTML editing
 - in-context, 7-8

I

- Integrated WebLogic Server (WLS)
 - installation, start and stop, 2-3
- Iterative Development
 - advantages, 6-9
 - effects, 6-1

L

- look and feel
 - defining, 4-1
 - how to change at design time, 5-1

M

- metadata field tags
 - in UCM, 6-14
- modify and edit an existing page template, 3-10

O

- Oracle Composer
 - Component Palette, 7-4
- Oracle JDeveloper
 - installing, 2-1
 - verifying correct release, 2-1

Oracle Technology Network, vi
Oracle WebCenter
 about, 1-1
OTN
 see Oracle Technology Network, vi

P

Page Customizable component, 7-4
page definition
 associating, 4-3
Page Hierarchy, 7-2
Page Template
 setting, 4-7
Page template
 creating, 4-1
 referenced, 4-1
 swooshy, 4-1
page template
 artifacts and page features, 3-10
 seeded, 3-8
PageTemplate
 Globe, 5-6
 Swooshy, 5-6
pageTemplate
 swooshy, 3-10
panelCustomizable component, 7-8
permissions, 7-2
Portal Application template, 3-1
Portal resource
 creating, 4-7
preferences
 change default, 5-2

R

Resource Manager, 5-5
 using at runtime, 5-4
Resource Manger
 necessary portal artifacts, 3-1

S

sample files
 downloading, 2-4
Sample Tutorial Files
 download, 2-4
set permissions
 user access and editing, 7-1
Site template
 customizing, 4-7
skin
 custom, 5-1
 global style sheet, 5-1
skins
 defining the appearance of your application, 5-1

T

template
 default, 5-1

Tutorial setup files
 extracting, 4-4
tutorial skin
 provided in tutorial setup zip file, 5-1

U

UCM, 2-4
 connecting to a content repository, 6-2
 metadata field tags, 6-1
UCM repository, 7-1, 7-8
Universal Content Management
 setting as primary connection, 6-1
users
 control over document content, 7-8

W

web address for zip files, 2-4
WebCenter applications
 creating, 3-1
WebCenter extension
 steps to install correct components, 2-2
 verifying components, 2-2
WebCenter Framework
 about, 1-1