Portal Tutorials

The following tutorials highlight the processes and features of portal development with the WebLogic Workshop Portal Extensions.

Tutorial: Building Your First Portal

This tutorial guides you through the brief process of creating a portal with working portlets. The tutorial takes about 15 minutes to complete.

Tutorial Goals

At the end of this tutorial you will have built a fully functional portal in a short time with little effort.

Tutorial Overview

The WebLogic Workshop Portal Extensions include a graphical Portal Designer that lets you surface application functionality easily and quickly in a sophisticated portal interface. Sample portlets included with the WebLogic Workshop Portal Extensions provide instant, reusable functionality for a portal, as this tutorial illustrates.

Steps in This Tutorial

Step 1: Start WebLogic Workshop, Open an Application, and Start WebLogic Server – 5 minutes. In this step you start WebLogic Workshop, open the sample portal application, and start WebLogic Server.

Step 2: Create a Portal – 10 minutes. In this step you create a portal file for an existing portal project, add a page to the portal, add portlets to the pages, and view your finished portal.

Step 1: Start WebLogic Workshop, Open an Application, and Start WebLogic Server

In this step, you will start the development environment for building a portal.

The tasks in this step are:

- Start WebLogic Workshop
- Open the Sample Portal Application
- Start WebLogic Server

Start WebLogic Workshop

From the Windows Start Menu, choose **-->Programs-->BEA WebLogic Platform 8.1-->WebLogic Workshop**. The start script is located in the <BEA_HOME>/weblogic81/workshop directory.

Open the Sample Portal Application

The portal application you open contains all the necessary portal resources to complete the tutorial. In Java 2 Enterprise Edition (J2EE) terms, this application is an enterprise application that contains Web applications and related resources. Each Web application can contain multiple portals.

1. In the WebLogic Workshop menu, choose File-->Open-->Application.
2. In the Open Workshop Application dialog, select the <BEA_HOME>\weblogic81\samples\portal\portalApp\portalApp.work file and click Open.

The portalApp application directory tree appears in the Application window.

Start WebLogic Server

To develop portals and portal applications in WebLogic Workshop, WebLogic Server must be running on your development machine. For this tutorial, you will start the domain server used by the WebLogic Portal samples. The portalApp application you opened in the previous step contains all the necessary server configuration settings. To start WebLogic Server:

- In the WebLogic Workshop menu, choose Tools-->WebLogic Server-->Start WebLogic Server.

  On Windows systems, you can bring up the command window from the Windows task bar to watch the startup progress. When the server starts, the WebLogic Workshop status bar shows the message "Server Running."

Step 2: Create a Portal

In this step, you will create a portal file for an existing portal project, add a page to the portal, add portlets to the pages, and view your finished portal.

The tasks in this step are:

- Create a Portal File
- Add a Page to the Portal

- Add a Portlet to the Portal
- View the Portal

Create a Portal File

A portal file is an XML file that contains all configuration information for a portal. The XML file is rendered graphically in the Portal Designer of the WebLogic Workshop Portal Extensions. As you build the portal in the graphical interface, the XML is generated automatically behind the scenes. To create a portal file:

1. In the Application window, right-click the sampleportal project and choose New-->Portal.
2. In the New File dialog, enter my.portal in the File name field. You must keep the file extension.
3. Click Create.

   The new file is added to the sampleportal project, and a new portal file appears in the Portal Designer. The new portal has a header, footer, and a main body containing a default book and page. The Document Structure window displays the component hierarchy.

Add a Page to the Portal

A new portal already contains a page. In this step you will add a second page to your portal and rename the tabs of both pages.

1. In the Palette window, drag the Page control into the Portal Designer next to the existing page tab. A page tab called "New Page" appears.
2. Click the **New Page** tab.
3. In the Property Editor window, change the page's Title property to My Page 2.
4. In the Portal Designer, click the Page 1 tab.
5. In the Property Editor window, change the page's Title property to My Page 1 and press Enter.

Add a Portlet to the Portal

In this step you will add a pre-built sample portlet from the sampleportal project to each page.

1. In the Portal Designer, click the My Page 1 tab.

2. In the Data Palette window, drag the Login to Portal portlet into the left placeholder of My Page 1.
3. In the Portal Designer, click the My Page 2 tab.
4. In the Data Palette window, drag the RSS News Feed into the left placeholder of My Page 2 and the Dev2Dev portlet into the right placeholder.

Note: The RSS News Feed portlet requires an Internet connection to access the content feed.

5. Save the portal file.

View the Portal

You can view your portal with the WebLogic Test Browser or with your default browser.

- WebLogic Test Browser - In the WebLogic Workshop toolbar, click the Start button (or press Ctrl+F5). Navigate between the portal pages using the My Page 1 and My Page 2 links.
- Default Browser - In the WebLogic Workshop menu, choose Portal-->Open Current Portal. Navigate between the portal pages using the My Page 1 and My Page 2 links.

Congratulations! You have built a portal.

What You Can Do Next

The portal development lifecycle involves development with the WebLogic Workshop Portal Extensions and administration with the WebLogic Administration Portal. The tutorial you have just completed covers the development phase. The following steps instructions for starting a tutorial that covers the administration phase. The second phase of the portal-building process involves administering the portal you have created. To run a companion tutorial for administering a portal:

1. In the WebLogic Workshop menu, choose Portal-->Open Portal Administration.
2. Log in with Username: weblogic Password: weblogic.
3. When the Administration Portal appears, click Show Help in the upper left corner of the window.
4. In the Help window, click the Tutorials tab, select Build Your First Portal, and follow the tutorial.

Tutorial: Changing a Portal's Look & Feel and Navigation

This tutorial shows you how easy it is to modify the look and feel of a portal and modify the navigation style used for the portal pages. The tutorial takes about 10 minutes to complete.

Tutorial Goals

At the end of this tutorial you will have changed the look and feel and page navigation style of a portal.

Tutorial Overview

This tutorial involves modifying two elements of a portal's physical appearance and behavior: look and feel, and navigation.

The WebLogic Workshop Portal Extensions provide a flexible, extensible architecture for controlling the look and feel and page navigation in a portal. A portal look and feel is made up of a skin (graphics, a cascading style sheet, and JavaScript functions) and skeletons (JSP files that determine the rendering—the physical boundaries—of individual portal components, such as desktops, pages, and portlets).

Ultimately the look and feel of a portal, and its navigation style, are determined by the portal administrator and end users. All look and feel and navigation resources that are available to developers in the WebLogic Workshop Portal Extensions are also available to delegated administrators in the WebLogic Administration Portal and to end users in the Visitor Tools when the portal is put into production.

When an administrator creates a desktop based on a .portal file in the WebLogic Administration Portal, that portal is decoupled from the development environment and can be modified as needed by the administrator, and in turn by end users when the portal is put into production. The initial look and feel and navigation settings you provide in development serve as the default settings for portal administrators and end users. This tutorial merely shows how easy it is to change look and feel and navigation elements in the development environment.

Steps in This Tutorial

Step 1: Start WebLogic Workshop, Open an Application, and Start WebLogic Server — 5 minutes. In this step you start WebLogic Workshop, open the sample portal application, and start WebLogic Server.

Step 2: Change the Portal Look and Feel and Navigation — 5 minutes. In this step you change a portal's look and feel and page navigation style.

Step 1: Start WebLogic Workshop, Open an Application, and Start WebLogic Server

In this step, you will start the development environment for working with portals.

The tasks in this step are:

- Start WebLogic Workshop
- Open the Sample Portal Application
- Start WebLogic Server

Start WebLogic Workshop

From the Windows Start Menu, choose **-->Programs-->BEA WebLogic Platform 8.1-->WebLogic Workshop**. The start script is located in the <BEA_HOME>/weblogic81/workshop directory.

Open the Sample Portal Application

The portal application you open contains all the necessary portal resources to complete the tutorial. In Java 2 Enterprise Edition (J2EE) terms, this application is an enterprise application that contains Web applications and related resources.

1. In the WebLogic Workshop menu, choose File-->Open-->Application.
2. In the Open Workshop Application dialog, select the <BEA_HOME>\weblogic81\samples\portal\portalApp\portalApp.work file and click Open.
3. The portalApp application directory tree appears in the Application window.

Start WebLogic Server

To develop portals and portal applications in WebLogic Workshop, WebLogic Server must be running on your development machine. For this tutorial, you will start the domain server used by the WebLogic Portal samples. The portalApp application you opened in the previous step contains all the necessary server configuration settings. To start WebLogic Server:

- In the WebLogic Workshop menu, choose Tools-->WebLogic Server-->Start WebLogic Server.

- On Windows systems, you can bring up the command window from the Windows task bar to watch the startup progress. When the server starts, the WebLogic Workshop status bar shows the message "Server Running."

Step 2: Change the Portal Look & Feel and Navigation

In this step, you will change a portal's look and feel, change the navigation scheme for the portal pages, and view the results.

The tasks in this step are:

- Open the Sample Portal
- View the Portal in a Browser
- Add a Book to the Main Page Book
- Change the Portal's Look and Feel
- Change the Portal's Navigation
- View the Modified Portal in a Browser

Open the Sample Portal

The Sample portal is included with the WebLogic Workshop Portal Extensions. It contains a set of pre-built sample portlets you can reuse in your own portals. You will change the Sample portal's look and feel and page navigation scheme.

1. In the Application window, expand the sampleportal folder.
2. Double-click sample.portal. The portal file opens in the Portal Designer.

View the Portal in a Browser

In this step, view the existing look and feel and navigation of the Sample portal to see what it looks like before you change it.

In this step, view the existing look and feel and navigation of the Sample portal to see what it looks like before you change it.

1. In the WebLogic Workshop toolbar, click the Start button (or press Ctrl+F5) to view the portal in the Workshop Test Browser.
2. Click the page navigation tabs and note the behavior.

3. Close the WebLogic Test Browser.

Add a Book to the Main Page Book

1. In the Palette window, drag the Book control into the Main Page Book area of the Portal Designer, next to My Page. A new book appears in the main page book.

   Do NOT drag the Book control into a placeholder on one of the pages.

2. Click the New Book tab. Notice that it automatically contains a new page as well.

Change the Portal's Look and Feel

1. In the Document Structure window, click Desktop.
2. In the Property Editor window, change the Look and Feel property from avitek to Classic.

Change the Portal's Navigation

1. In the Document Structure window, click Book: Main Page Book.
2. In the Property Editor window, change the Navigation property from Single Level Menu to Multi Level Menu. Multi-level menus display links to nested books and pages with drop-down navigation.
3. Save the portal file.

View the Modified Portal in a Browser

You can view the portal with the WebLogic Test Browser or with your default browser.

- WebLogic Test Browser - In the WebLogic Workshop toolbar, click the Start button (or press Ctrl+F5).
- Default Browser - In the WebLogic Workshop menu, choose Portal-->Open Current Portal.

Use the page and book navigation links and note the different navigation behavior. The New Book link should provide a drop-down navigation menu to access the page it contains.

Congratulations! You have changed a portals look and feel and page navigation style.

Tutorial: Showing Personalized Content in a Portlet

This tutorial has you develop personalization functionality and surface that functionality in a portlet. The tutorial takes about 30 minutes to complete.

Tutorial Goals

At the end of this tutorial you will have created a portlet that displays different personalized content to different users.

Tutorial Overview

The WebLogic Workshop Portal Extensions include a graphical Portal Designer that lets you create and surface application functionality easily and quickly in a sophisticated portal interface. In this tutorial you will use the WebLogic Workshop Portal Extensions and the WebLogic Administration Portal to create the users, properties, content, and code that results in a complete personalization solution. You will perform minimal JSP development using WebLogic Workshop Portal Extensions JSP tags in an easy-to-use, integrated interface.

Steps in This Tutorial

Step 1: Start WebLogic Workshop, Open an Application, and Start WebLogic Server — 5 minutes. In this step you start WebLogic Workshop, open the sample portal application, and start WebLogic Server.

Step 2: Create a User Profile Property Set — 5 minutes. In this step you will create the user properties that uniquely identify authenticated users and determine what the users see.

Step 3: Create Two Users — 5 minutes. In this step you will create two test users and assign property values to them.

Step 4: Load Content— 5 minutes. In this step you will load content into the BEA Virtual Content Repository.

Step 5: Create a JSP — 1 minute. In this step you will create a JSP file.

Step 6: Add Content Selectors to the JSP — 10 minutes. In this step you will add JSP tags to the JSP to enable personalization.

Step 7: Create a Portlet with the JSP — 1 minute. In this step you will create a portlet out of the JSP and add it to an existing portal.

Step 8: Test the Personalized Portlet — 5 minutes. In this step you will log in as both users to see the personalized portlet in action.

Step 1: Start WebLogic Workshop, Open an Application, and Start WebLogic Server

In this step, you will start the development environment for working with portals.

The tasks in this step are:

- Start WebLogic Workshop
- Open the Sample Portal Application
- Start WebLogic Server

Start WebLogic Workshop

From the Windows Start Menu, choose **-->Programs-->BEA WebLogic Platform 8.1-->WebLogic Workshop**. The start script is located in the <BEA_HOME>/weblogic81/workshop directory.

Open the Sample Portal Application

The portal application you open contains all the necessary portal resources to complete the tutorial. In Java 2 Enterprise Edition (J2EE) terms, this application is an enterprise application that contains Web applications and related resources.

1. In the WebLogic Workshop menu, choose File-->Open-->Application.
2. In the Open Workshop Application dialog, select the <BEA_HOME>\weblogic81\samples\portal\portalApp\portalApp.work file and click Open.
3. The portalApp application directory tree appears in the Application window.

Start WebLogic Server

To develop portals and portal applications in WebLogic Workshop, WebLogic Server must be running on your development machine. For this tutorial, you will start the domain server used by the WebLogic Portal samples. The portalApp application you opened in the previous step contains all the necessary server configuration settings. To start WebLogic Server:

- In the WebLogic Workshop menu, choose Tools-->WebLogic Server-->Start WebLogic Server.

On Windows systems, you can bring up the command window from the Windows task bar to watch the startup progress. When the server starts, the WebLogic Workshop status bar shows the message "Server Running."

Step 2: Create a User Profile Property Set

In this step, you will create a user profile property set. This property set will contain properties that can be set for all users. Each user can have different property values. In this tutorial, property values determine the personalized content users see.

The tasks in this step are:

- Create a Property Set File
- Add a Property to the Property Set

Create a Property Set File

1. In the Application window, expand the data project.
2. Right-click the userprofiles folder, and choose New-->User Profile Property Set.
3. In the New File dialog, enter userpreferences.usr in the File name field. You must keep the file extension.
4. Click Create. The Property Set Designer appears.

Add a Property to the Property Set

1. In the Palette window, drag the Single Restricted icon into the Property Set Designer.
2. In the Property Editor window, enter the following text in the Property Name field: Graphic Preference.
3. Click the ellipsis icon [...] in the Value(s) field.
4. In the Enter Property Value dialog, enter modern in the top field and click Add. Enter classic in the top field and click Add.
5. Click OK.
6. Save and close the property set file.

Step 3: Create Two Users

In this step, you will create two test users that have different preferences set in their user profiles. When you log in as each user at the end of the tutorial, each users will see different content displayed in the portlet based on their different user profile property values.

The tasks in this step are:

- Start the WebLogic Administration Portal
- Create Two Users and Set Their Profile Preferences

Start the WebLogic Administration Portal

1. In the WebLogic Workshop menu, choose Portal-->Open Portal Administration.
2. Log in with Username: weblogic Password: weblogic.
3. When the WebLogic Administration Portal appears, select Users & Groups under Users, Gropus, & Roles in the top Menu in the editor pane.

Create Two Users and Set Their Profile Preferences

1. In the Users & Groups resource tree, click everyone (All Users).
2. Select the Add Users page in the editor pane.
3. Click Create User in the upper right of the main window.
4. In the Add a New User dialog that appears, enter modernuser, enter a password of password (which is the default), and click Add New User. A confirmation message appears at the top of the Add Users page.
5. Create a second user called classicuser with a password of password.
6. Select the Edit Users page.
7. In section 1 of the page, enter an asterisk (*) in the Search field and click Search.
8. In section 2 of the page,click the classicuser name. The Editing User window appears.
9. Select the Edit User Profile Values page.
10. In the Properties from property set field, select userpreferences. This is the property set you created.

11. Expand the Graphic Preference property line. Change the Graphic Preference property value to classic, and click Update Value.
12. In the left resource tree, select everyone (All Users).
13. In section 1 of the page, enter an asterisk (*) in the Search field and click Search.
14. In section 2 of the page,click the modernuser name. The Editing User window appears.
15. On the Edit User Profile Values page, change the user's Graphic Preference to modern, and click Update Value.

You now have two users with different user profile preferences.

Step 4: Load Content

In this step, you will load sample content into the Virtual Content Repository that will be used later in this tutorial. This step requires WebLogic Server to be running, as described in Step 1.

1. Open a command window and change to the following directory:

   ```
   <BEA_HOME>\weblogic81\portal\bin
   ```

2. Enter the following command:

   ```
   load_cm_data.cmd
   ```

   or

   ```
   sh load_cm_data.sh
   ```

   The script loads sample content into the sample domain's Virtual Content Repository under the default "BEA Repository."

Preview the Content

You can view this sample content in the WebLogic Administration Portal using the following steps:

1. Choose Portal-->Open Portal Administration in WebLogic Workshop or by entering http://localhost:7001/portalAppAdmin in a browser.
2. Log in as weblogic/weblogic.
3. In the WebLogic Administration Portal, select Content in the top Menu in the editor pane.
4. In the left resource tree, expand the BEA Repository.

Step 5: Create a JSP

In this step, you will create a Java Server Page (JSP) that will display the personalized content in a portlet.

Create a JSP

1. In WebLogic Workshop, right-click the sampleportal project in the Application window and choose New-->JSP File.
2. In the New File dialog, name new JSP **myp13n.jsp**. You must keep the file extension.
3. Click Create. A new JSP file appears.
4. Click the Source View tab at the bottom of the JSP Designer, and delete the text and tags that appear inside the <body> tag.
5. Click the Design View tab at the bottom of the JSP Designer.

Step 6: Add Content Selectors to the JSP

In this step, you will add content selector JSP tags to the JSP, along with two other JSP tags to display the personalized content. In the process of adding the content selector JSP tags, you will also create content selectors (XML files) that contain the queries for retrieving content from the BEA Virtual Content Repository and the rules that trigger the queries to run.

The tasks in this step are:

- Add the <pz:contentSelector> Tag to the JSP
- Add the <es:forEachInArray> Tag to the JSP
- Add an Image Tag
- Create a Second Content Selector

Add the <pz:contentSelector> Tag to the JSP

1. In the JSP Designer, make sure the Design View tab at the bottom of the window is selected.
2. In the Palette window, locate the Portal Personalization category and drag the Content Selector tag into the JSP Designer.
3. In the Property Editor window, enter nodes in the id field.

4. In the rule field, click the ellipsis icon [...]. The Select content selector dialog appears.
5. Click New content selector, and click Select.
6. In the New content selector dialog, enter modern and click OK.
7. In the Property Editor window, click the -> icon in the rule field. The Content Selector Designer appears.

   Now you will define the query the content selector uses to retrieve content from the Virtual Content Repository.
8. In the Content Selector Designer, click the [empty content search] link. The Content Search window appears.
9. In the Property set field, select Standard.
10. In the Property field, make sure cm_binaryName is selected, and click Add. The Content Search Values window appears.
11. In the Comparison field, select contains.
12. In the Value field, enter college and click Add.
13. Click OK. You are returned to the Content Search window. Click OK.
14. In the Available Conditions area of the Content Selector Designer, deselect any checkboxes that are selected, then select The visitor has specific characteristics.
15. In the top of the Content Selector Designer, click the [characteristics] link. The Visitor Characteristics window appears.
16. In the Visitor property set field, select userpreferences.
17. In the Visitor property field, make sure Graphic Preference is selected, and click Add. The Visitor Characteristic Values window appears.
18. In the Comparison field, make sure is equal to is selected.
19. In the Value field, select modern, click Add, and click OK.
20. In the Visitor Characteristics window, click OK.
21. Save and close the file.

Add the <es:forEachInArray> Tag to the JSP

1. In the JSP Designer, open the myp13n.jsp file if you are not already there.
2. In the Palette window, locate the Portal Utilities category, and drag the For Each In Array tag into the JSP Designer, to the right of the <pz:contentSelector> tag.
3. Click the Source View tab in the JSP designer and set the following properties on the For Each In Array tag:

```
<es:forEachInArray array="<%=nodes%>" id="node"
type="com.bea.content.Node">
```

Add an Image Tag

1. Switch back to Design View in the JSP designer.
2. In the Palette window, locate the HTML category, and drag the Image tag on top of the forEachInArray tag. In the ImageWizard dialog that appears, click OK. This should put the image tag inside the forEachInArray tag.
3. Go back to Source View in the JSP designer and add the following path to the image:

```
<img src="<%=request.getContextPath() + "/ShowBinary" +
node.getPath()%>">
```

Create a Second Content Selector

1. In Source View, copy the block of JSP tags and the image reference you created, and paste the block just below itself.
2. In the copy of the JSP tags, change the <pz:contentSelector> tag's rule attribute to classic. You should now have two content selector blocks of code. The only difference between the two is the rule attribute value. One value is modern and the other is classic.
3. Switch to Design View in the JSP designer.
4. Select the second <pz:contentSelector> tag. It appears as classic.
5. In the Property Editor window, click the ellipsis icon [...] in the rule field. The Select content selector window appears.
6. Click New content selector, and click Select. The New content selector window appears.
7. Enter the name classic, and click OK.

8. In the Property Editor window, click the -> icon in the rule field. The Content Selector Designer appears.
9. In the Content Selector Designer, click the [empty content search] link. The Content Search window appears.
10. In the Property set field, select Standard.
11. In the Property field, make sure cm_binaryName is selected, and click Add. The Content Search Values window appears.
12. In the Comparison field, select contains.
13. In the Value field, enter IRACampaign and click Add.
14. Click OK. You are returned to the Content Search window. Click OK.
15. In the Available Conditions area of the Content Selector Designer, deselect any checkboxes that are selected, then select The visitor has specific characteristics.
16. In the top of the Content Selector Designer, click the [characteristics] link. The Visitor Characteristics window appears.
17. In the Visitor property set field, select userpreferences.
18. In the Visitor property field, make sure Graphic Preference is selected, and click Add. The Visitor Characteristic Values window appears.
19. In the Comparison field, make sure is equal to is selected.
20. In the Value field, select classic, click Add, and click OK.
21. In the Visitor Characteristics window, click OK.
22. Save and close the content selector file and the JSP file.

Step 7: Create a Portlet with the JSP

In this step, you will drag the JSP into an existing portal to create a portlet out of it with the Portlet Wizard.

Create a Portlet by Adding the JSP to a Portal

1. In the Application window, expand the sampleportal project, and double-click sample.portal to open it in the Portal Designer.

2. In the Application window, drag myp13n.jsp below the Login portlet in the Portal Designer, and click Yes in the Create Portlet dialog that appears. The Portlet Wizard appears.
3. Click Next.In the Portlet Details window, enter the following text in the Title field: My
4. Personalized Portlet.
5. Click Finish. The new portlet appears below the Login to Portal Portlet.
6. Save the portal file.

Step 8: Test the Personalized Portlet

In this step, you will test the portlet to see personalization in action.

The tasks in this step are:

- View the Sample Portal
- Log in as One User and View the Portlet
- Log in as the Other User and View the Portlet

View the Sample Portal

You can view the portal with the WebLogic Test Browser or with your default browser.

1. WebLogic Test Browser - In the WebLogic Workshop toolbar, click the Start button (or press Ctrl+F5).
2. Default Browser - In the WebLogic Workshop menu, choose Portal-->Open Current Portal.

Log in as One User and View the Portlet

1. In the Login portlet, log in with Username: classicuser Password: password.
2. View the updated contents of the personalized portlet.
3. Click Logout.

Log in as the Other User and View the Portlet

1. Log in with Username: modernuser Password: password.
2. View the contents of the personalized portlet.

3. Congratulations! You have created a portlet that uses personalization.

What You Can Do Next

The portal development lifecycle involves development with the WebLogic Workshop Portal Extensions and administration with the WebLogic Administration Portal. The tutorial you have just completed covers the development phase. The following steps instructions for starting a tutorial that covers the administration phase. The second phase of the portal personalization process involves administering the content selectors you have created. To run a companion tutorial for administering content selectors:

1. In the WebLogic Workshop menu, choose Portal-->Open Portal Administration.
2. Log in with Username: weblogic Password: weblogic.
3. When the Administration Portal appears, click Show Help in the upper left corner of the window.
4. In the Help window, click the Tutorials tab, select Add Personalized Content to Your Portal, and follow the tutorial.

Tutorial: Creating a Login Control Page Flow

This tutorial shows you how easy it is to add a Portal Control into a Page Flow. The tutorial takes about 20 minutes to complete.

Tutorial Goals

At the end of this tutorial you will have created a Page Flow that includes a Portal Control, and that can be placed inside a portlet to expose the login functionality.

Tutorial Overview

This tutorial introduces the use of Portal Controls with Page Flows, and should help familiarize you with databinding in forms.

Note: For instructions on adding the User Login Control without the wizard, consult the User Login Control help.

Steps in This Tutorial

Step 1: Create a Portal Control Page Flow — 10 minutes. In this step you create a new Page Flow, which includes selecting the Login Control.

Step 2: Place the Page Flow in a Portlet— 10 minutes. In this step the new Page Flow is used to create a new portlet.

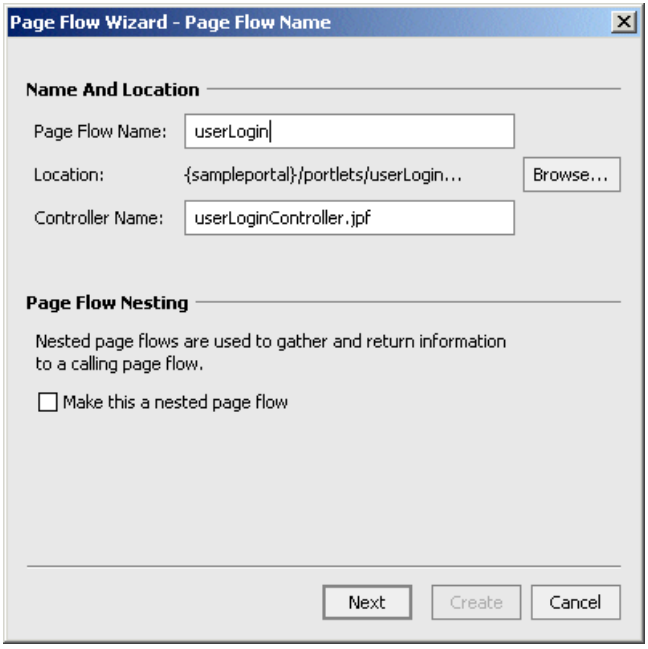
Step 1: Create a Page Flow Using the Wizard

In this step, you will create a Portal Control Page Flow using the Page Flow Wizard.

Create the Page Flow

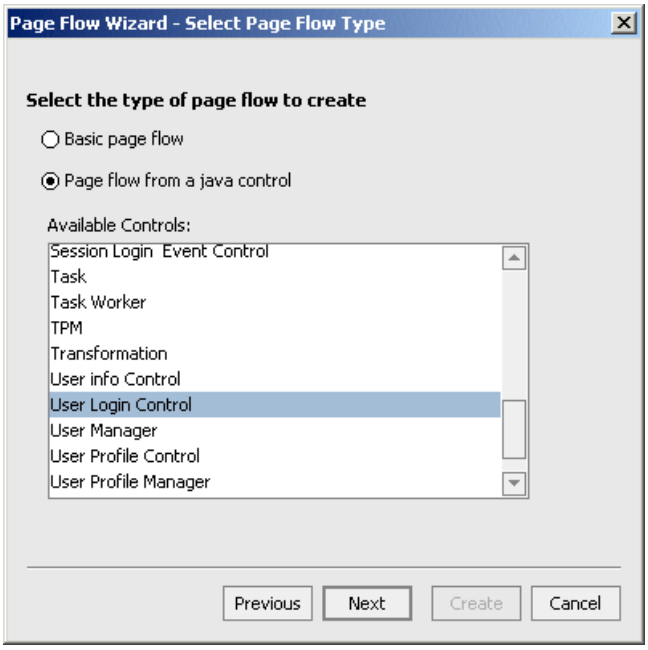
1. From a Portal Web application project within WebLogic Workshop, create a new Page Flow by right-clicking on the portlets directory within the current Portal project and selecting New -> Page Flow.
2. The Page Flow Wizard appears. Name this new Page Flow "userLogin", and click Next.

Figure 1-1 Page Flow Wizard



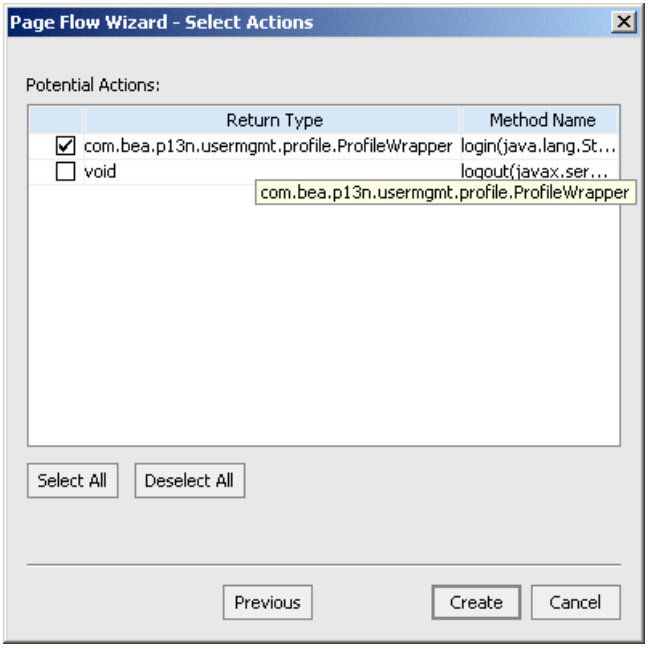
3. At the Select Page Flow Type prompt, select Page Flow from a Java Control, select the User Login Control, and Click Next.

Figure 1-2 Create User Login Control



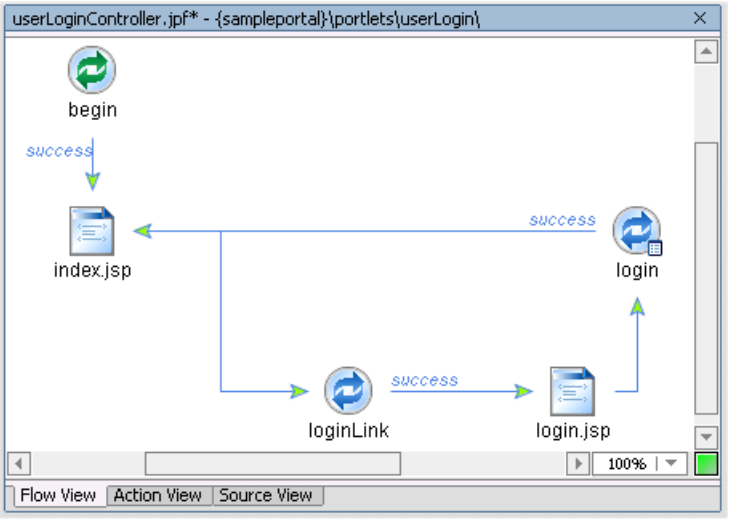
4. From the Select Actions prompt, select the login() method, which returns the ProfileWrapper type. Click Create.

Figure 1-3 Select Form Actions



5. The resulting Page Flow should appear in Flow View.

Figure 1-4 View Page Flow



6. Click Ctrl+S to save your work. Notice that in addition to the userLoginController page flow file, the wizard has also generated index.jsp and login.jsp.

   The wizard has created a page flow with all the necessary elements in place, but they must be configured to fit your application. The pageflow needs to be modified to pass in the "request" to the methods. By default, the pageflow will create a formbean property for every parameter.

7. Open the userLoginController.jpf in Source View, and within the control method login, replace this code:

   ```
   com.bea.p13n.usermgmt.profile.ProfileWrapper var = myControl.login(
   aForm.username, aForm.password, aForm.request );
   ```

   with this:

   ```
   com.bea.p13n.usermgmt.profile.ProfileWrapper var = myControl.login(
   aForm.username, aForm.password, super.getRequest() );
   ```

8. Now open the userLoginController.jpf in Flow View, and from the Data Palette, open the control (it should be called myControl), and drag the logout() method into the Page Flow.

   Double-clicking on the logout action will open the Source View to the code:

   ```
   /**
   ```

```
* @jpf:action
*/
protected Forward logout(LogoutForm form)
{
myControl.logout(form.getRequest());
return new Forward( "success" );
}
```

9. To the action method definition for this logout action, add the following line:

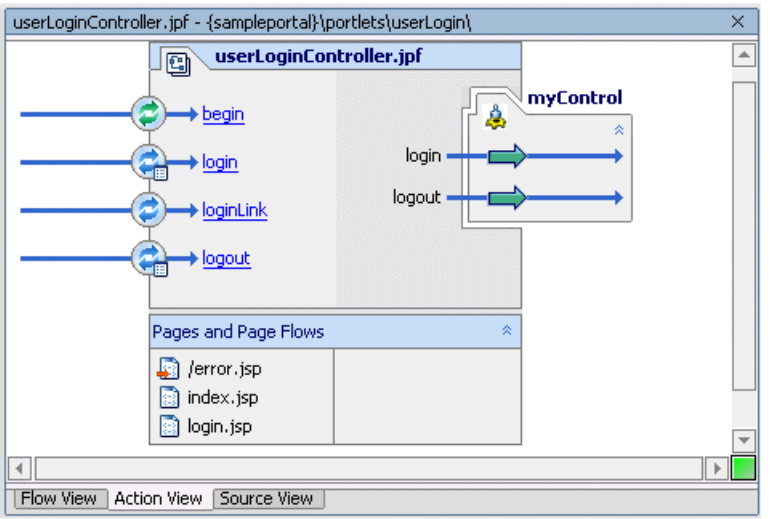
```
* @jpf:forward name="success" path="index.jsp"
```

10. In the same action code, change the (form.getRequest()); to (this.getRequest());
11. The resulting action method definition should look like this:

```
/**
* @jpf:action
* @jpf:forward name="success" path="index.jsp"
*/
protected Forward logout(LogoutForm form)
{
myControl.logout(this.getRequest());
return new Forward( "success" );
}
```

12. From the Flow View, the userLoginController.jpf should now look like this:

Figure 1-5 userLoginController.jpf



13. Next, double-click on the index.jsp and open Source View. Just below the login form, add a logout button:

    ```
    <netui:anchor action="loginLink">
    login
    </netui:anchor>
    <netui:form action="logout">
    <netui:button type="submit" value="logout"
    action="logout"></netui:button>
    </netui:form>
    ```

    The code in bold above places a button of type "submit" in the bottom of the page, associates it with the logout action, wraps it inside a netui:form that invokes the action="logout".

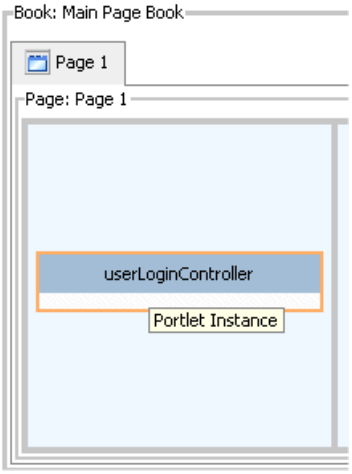
14. Save the index.jsp file.

Step 2: Place the Page Flow in a Portlet

In this step, you will use the Portlet Wizard to create a Portlet from the new Page Flow.

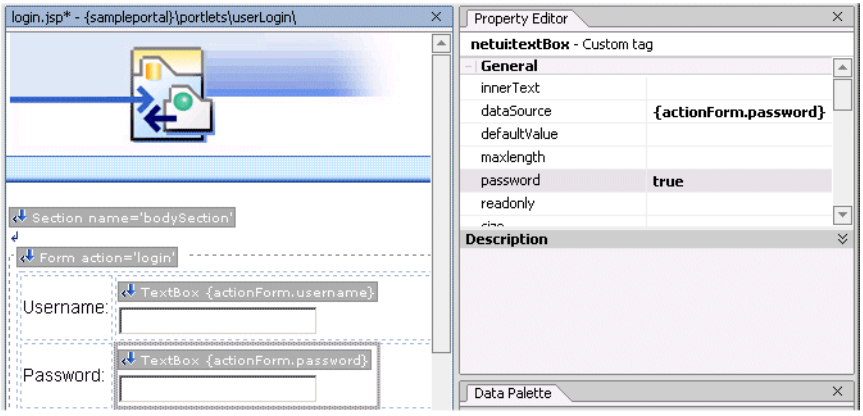
1. In the Portal Designer, drag the Page Flow file, called userLoginController.jpf, into a place holder in a portal. The Portlet Wizard appears.
2. From the Portlet Details screen, click Finish to create the userLoginController portlet and place it in the portal.

Figure 1-6 Place userLoginController Portlet in Placeholder



3. Open **login.jsp** in the Design View and select the Password text box. Then, in the Properties Editor, scroll down the list of General Properties to set Password equal to Yes. This will cause any input in this field to be masked.

Figure 1-7 Set Properties on Login.jsp



4. For debugging purposes, the next step shows how to add some code to show which user is logged on. Open index.jsp in the Source View, adding the following code to the end, just before the last two lines:

```
<%Object res = request.getAttribute ( "results" );%><%=(res == null ?
"<i>none</i><br/>" : ( res + "<br/>"))%>

<% if (request.getRemoteUser() != null) { %>

<BR>you are logged in as: <%=request.getRemoteUser()%>

<br>

<%

}

else

{

%>

<BR>you are not logged in

<%

}

%>
```

The index.jsp should read as follows:

```
<!--Generated by WebLogic Workshop-->
```

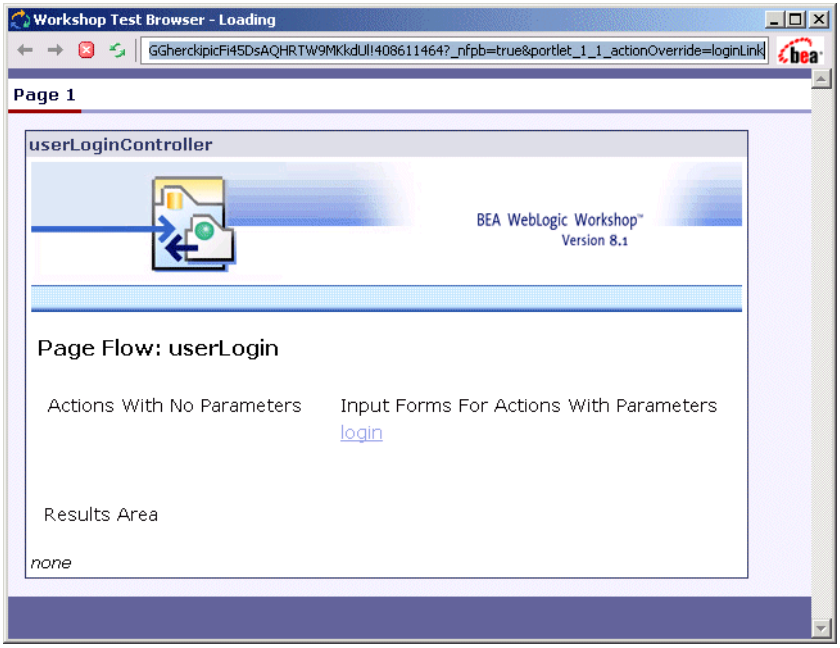
```
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@ taglib uri="netui-tags-databinding.tld" prefix="netui-data"%>
<%@ taglib uri="netui-tags-html.tld" prefix="netui"%>
<%@ taglib uri="netui-tags-template.tld" prefix="netui-template"%>
<netui-template:template templatePage="/resources/jsp/template.jsp">
<netui-template:setAttribute value="Index" name="title"/>
<netui-template:section name="bodySection">
<p class="pagehead">
 Page Flow: userLogin
</p>
<table width="100%" cellpadding="0" class="tablebody" cellspacing="0">
<tr>
<td valign="top">
<table width="100%" class="tablebody">
<tr class="tablehead">
<td>Actions With No Parameters</td>
</tr>
</table>
</td>
<td valign="top">
<table width="100%" class="tablebody">
<tr class="tablehead">
<td>Input Forms For Actions With Parameters</td>
</tr>
<tr>
<td>
<netui:anchor action="loginLink">
login
</netui:anchor>
<br>
```

```
</td>
</tr>
</table>
<br/>
<br/>
</td>
</tr>
<tr class="tablehead">
<td align="left" colspan="2">
Results Area
</td>
</tr>
</table>
<br/>
<%Object res = request.getAttribute ( "results" );%><%=(res == null ?
"<i>none</i><br/>" : ( res + "<br/>"))%>
<% if (request.getRemoteUser() != null) { %>
<BR>you are logged in as: <%=request.getRemoteUser()%>
<br>
<netui:form action="logout">
<netui:button type="submit" value="logout"
action="logout"></netui:button>
</netui:form>



<%
}
else
{
%>
<BR>you are not logged in
```

```
<%
}
%>
</netui-template:section>
</netui-template:template>
```

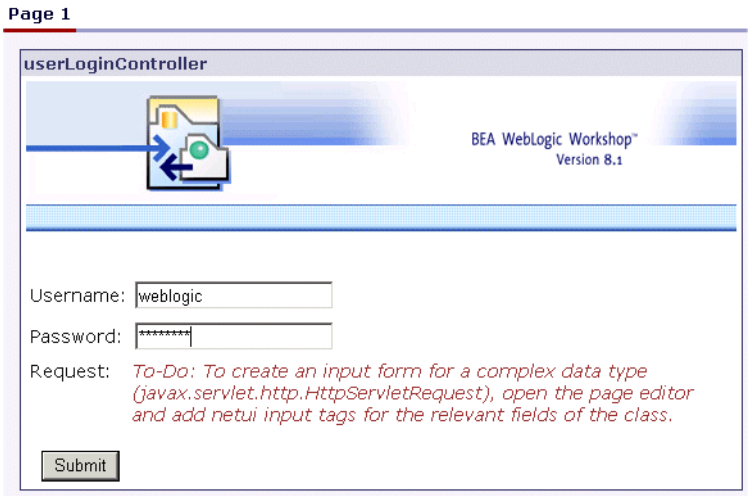
5. Click Ctrl+S to save your work.
6. In the WebLogic Workshop menu, choose Portal-->Open Current Portal.

Figure 1-8 Previewing the Portal



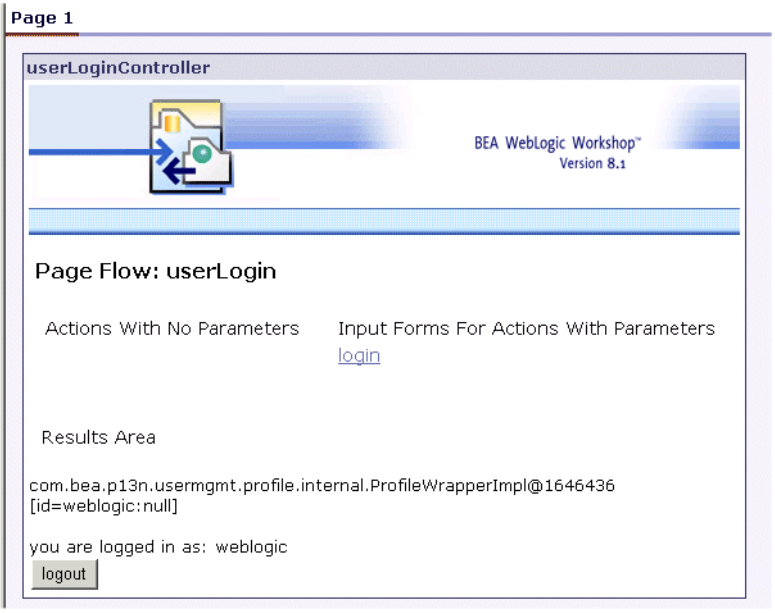
7. Click on Login, and at the login page, enter weblogic/weblogic. (The password field should be masked.

Figure 1-9 Loggin Into the Control Portal



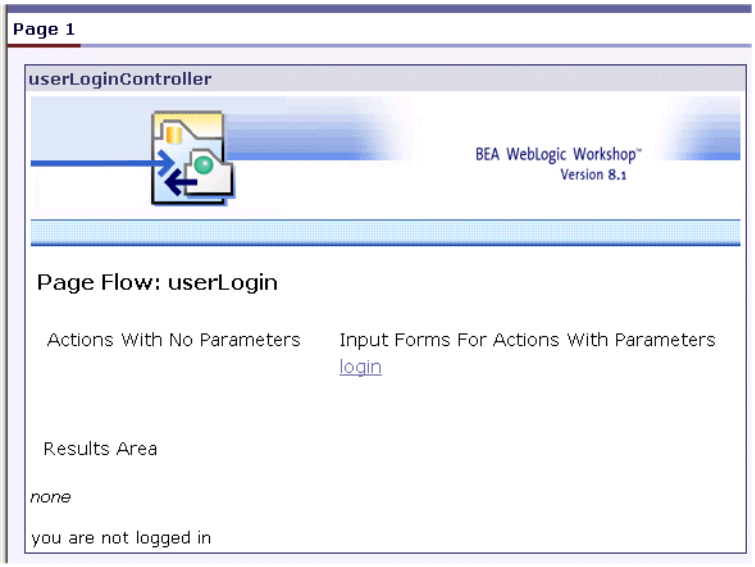
8. Verify that the next page displays the current logged-in user.

Figure 1-10 Verify the Login Action Works



9. Click logout, and verify the page displays the "you are not logged in" message.

Figure 1-11 Verify Logout Action Works



Congratulations! You have created a portlet that uses a Page Flow and an EJB control.

Tutorial: Using Page Flows Inside Portlets

This tutorial guides you through the process of learning how to use Page Flows inside portlets. The tutorial takes about 45 minutes to complete.

Tutorial Goals

At the end of this tutorial you will have some familiarity with how to use Page Flows inside portlets.

Tutorial Overview

The WebLogic Workshop Portal Extensions include a graphical Portal Designer that lets you surface application functionality easily and quickly in a sophisticated portal interface. Sample portlets included with the WebLogic Workshop Portal Extensions provide instant, reusable functionality for a portal, as this tutorial illustrates.

The portal development lifecycle involves development with the WebLogic Workshop Portal Extensions and administration with the WebLogic Administration Portal. This tutorial covers the development phase. After you have completed this tutorial, you will see instructions for starting a tutorial that covers the administration phase.

Steps in This Tutorial

Step 1: Create a Portal Application — 5 minutes. In this step you Create a Portal Application, add a Portal Web Project, and add a Portal, and start WebLogic Server.

Step 2: Create a Simple Navigation Page Flow portlet - 10 minutes. In this step you create two simple .jpf files and their corresponding portlets.

Step 3: Create portlets that communicate. - 10 minutes. In this step you create a Page Flow portlet that listens to input from another Page Flow portlet.

Step 4: Create an EJB Control Page Flow portlet. - 10 minutes. In this step you create a Page Flow portlet, add a Personalization Control, and edit properties on the Control.

Step 1: Create a Portal Application

In this step, you will create a new Portal Application, add a Portal Web Project, and add a Portal.

The tasks in this step are:

- Create a new Portal Application

- Add a Portal

Create a new Portal Application

To create the necessary framework and resources for portal development, you must do one of two things:

- Create a new portal application and add a Portal Web Project to it.
- Install Portal into an existing application and add a Portal Web Project to it.
- Add a Portal
- Create a Portal File to use as the layout framework to view your portlets. Name thisportal.portal.

Step 2: Create a Simple Navigation Page Flow

In this step, you will create a Page Flow that provides navigation from one JSP to another. Then, instead of designating the JSP files as the content url for the portlet, you will designate the .JPF file as the portlet's content node. The Page Flow supports JSP transitions within the portlet.

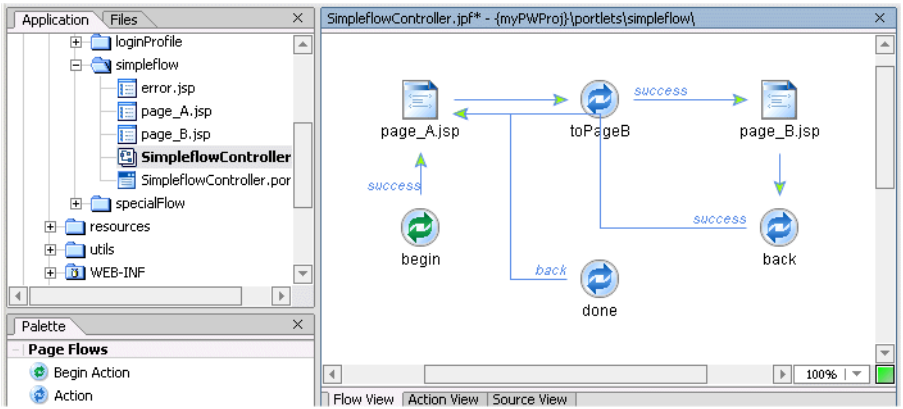
The tasks in this step are:

- Create a Page Flow Called simpleflow
- Insert this Page Flow into a portlet
- View the navigation portlets

Create a Page Flow Called simpleflow

1. From within the Project tab in your Web application, create a new folder called portlets.
1. Right-click on the portlets directory and create a new Page Flow. Name the Page Flow Simpleflow.
2. Add two actions: back and toPageB.

Figure 1-12 Adding Actions to a Page Flow



The code for this JPF should look something like this:

```
package portlets.simpleflow;
import com.bea.wlw.netui.pageflow.Forward;
import com.bea.wlw.netui.pageflow.PageFlowController;
public class SimpleflowController extends PageFlowController
{

    /**
     * This method represents the point of entry into the page group
     * @jpf:action
     * @jpf:forward name="success" path="page_A.jsp"
     *
     */
    protected Forward begin()
    {
        return new Forward( "success" );
    }

    /**
```

```
     * @jpf:action
     * @jpf:forward name="success" path="page_B.jsp"
     */
    public Forward toPageB()
    {
        return new Forward( "success" );
    }
    /**
     * @jpf:action
     * @jpf:forward name="success" path="page_A.jsp"
     */
    protected Forward back()
    {
        return new Forward("success");
    }
}
```

Insert this Page Flow into a portlet

1. Open the portal in the Portal designer, and drag the .JPF file into a placeholder on a page. The Portlet Wizard appears. Name and save the portal.
2. Open the Page Flow in the Page Flow editor, the following nodes will appear greyed out: page_B.jsp and page_A.jsp. Right-click on each of these to create the files.

   The JSPs should appear roughly as follows:

   page_A.jsp

```
<!--Generated by Weblogic Workshop-->
<%@ page language="java" contentType="text/html;charset=UTF-8"%>
<%@ taglib uri="netui-tags-databinding.tld" prefix="netui-data"%>
<%@ taglib uri="netui-tags-html.tld" prefix="netui"%>
<%@ taglib uri="netui-tags-template.tld" prefix="netui-template"%>
<netui-template:template templatePage="/resources/jsp/template.jsp">
<netui-template:setAttribute value="Index" name="page_A.jsp"/>
```

```
<netui-template:section name="bodySection">
<br/>
<blockquote>
  <p>
  <h3>This is page_A.jsp</h3>
  <p>
  <netui:anchor action="toPageB">Link to page_B.jsp by calling
  the "toPageB" action on the SimpleFlowController.jpf Page
flow.</netui:anchor>
  </blockquote>


  </netui-template:section>
  </netui-template:template>
```

page_B.jsp

```
<!--Generated by Weblogic Workshop--><%@ page language="java"
contentType="text/html;charset=UTF-8"%><%@ taglib
uri="netui-tags-databinding.tld" prefix="netui-data"%><%@ taglib
uri="netui-tags-html.tld" prefix="netui"%><%@ taglib
uri="netui-tags-template.tld"
prefix="netui-template"%><netui-template:template
templatePage="/resources/jsp/template.jsp">
<netui-template:setAttribute value="Index" name="page_B.jsp"/>
<netui-template:section name="bodySection">           <br/>
<blockquote>       <p>    <h3>This is page_B.jsp</h3>    <p>
<netui:anchor action="back">navigate to page_A.jsp by calling "back" on
the SimpleFlowController.jpf page flow. </netui:anchor>
</netui-template:section>
</netui-template:template>
```

View the Navigation Portlet

1. Save all files and start the server.
2. Preview the portlet by navigating to:

   ```
   http://<host>:<port>YourWebappp/portal.portal.
   ```

Step 3: Create Portlets that Communicate

In this step, you will use Page Flows to create one portlet that listens to another portlet. This particular example shows two ways of passing information from one portlet to the next: using a Form or using the Request Parameter. They both use the Page Flow as the means of connecting two portlets. By configuring the portlets to listen to one another, forms and requests instantiated by a JSP in one Page Flow can be sent to both Page Flows. This allows the listening portlet to update itself.

The tasks in this step are:

- Create two Page Flow portlets
- Create Actions, Additional JPS
- Place Portlets in Portal
- Set Properties on Portlets

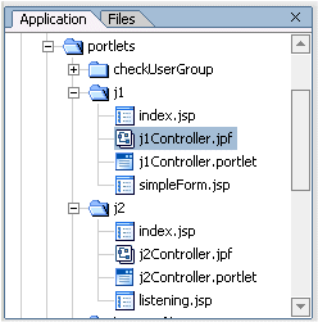
Create two Page Flow portlets

In this task, two Page Flows are created within the portlets directory; one called j1, the other called j2.

1. Right-click on the portlets directory, and select Create New Page Flow.
2. Name the Page Flow j1, and select Basic Page Flow (with no Java Controls.)
3. Repeat this step, creating a Page Flow called j2.

Note: WebLogic Workshop automatically creates the Page Flow controller class, prepending the name of the Page Flow to the Controller.jpf filename.

Figure 1-13 Viewing Page Flows in Application Tab



Edit Page Flows

This example requires modifying the Page Flows so that they can be notified by the listening portlet. The j1 Page Flow will receive form data from its simpleForm.jsp. The j2 Page Flow (in a listening portlet) will also receive notifications, and store the data, so its portlet can retrieve and display it. \

Note: Use Flow View to set up actions, then touch up the code using the Source View.

Edit j1 Page Flow

1. Open the j1Controller.jpf and make sure the package declaration is correct:

   ```
   package portlets.j1;
   ```

2. Edit the begin method, adding a forward named simpleForm:

   ```
   /**
     * @jpf:action
     * @jpf:forward name="simpleForm" path="simpleForm.jsp"
     */
     public Forward begin()
     {
     return new Forward( "simpleForm" );
     }
   ```

3. Add the following three actions to the Page Flow:

   ```
   /**
   ```

```
        * @jpf:action
        * @jpf:forward name="simpleForm" path="simpleForm.jsp"
        */
        public Forward passString1( Form form )
        {
        passedText = form.getText();
        return new Forward( "simpleForm" );
        }


    /**
        * @jpf:action
        * @jpf:forward name="simpleForm" path="simpleForm.jsp"
        */
        public Forward passString2()
        {
        return new Forward( "simpleForm" );
        }


        /**
        * @jpf:action
        * @jpf:forward name="simpleForm" path="simpleForm.jsp"
        */
        public Forward passString3()
        {
        return new Forward( "simpleForm" );
        }
```

4. Make sure the following Form inner class appears at the end of the Page Flow file:

```
public static class Form extends FormData
    {
private String text;
public void setText( String text )
        {
        this.text = text;
        }
        public String getText()
        {
        return this.text;
        }
        }
```

Edit j2 Page Flow

1. Open the j2Controller.jpf and make sure the package declaration is correct:

   ```
   package portlets.j2;
   ```

2. This Page Flow receives data from the j1 portlet and stores it so it can be displayed in the j2 portlet. directs the j2 portlet to listen to the j1 portlet. Add a variable declaration to the beginning of the class to hold the text from the j1 Page Flow:

   ```
   public class j2Controller extends PageFlowController
     {
     public String thePassedText = "";
   ```

3. Edit the begin method, adding a forward named listening, which points to the listening.jsp we'll create later. The listening.jsp will display in the j2 portlet the data received from the j2 Page Flow.

   ```
   /**
     * @jpf:action
     * @jpf:forward name="listening" path="listening.jsp"
     */
     public Forward begin()
     {
   ```

```
        return new Forward( "listening" );
         }
```

4. The first action we'll now add to the Page Flow has the same signature as the j1 passString1 method. When the j2 portlet is listening to j1, both the j1.passString1 method and the j2.passString2 methods are called.

```
/**
* @jpf:action
* @jpf:forward name="listening" path="listening.jsp"
*/
public Forward passString1(portlets.j1.j1Controller.Form form)
{
thePassedText = form.getText();
return new Forward( "listening" );
}
```

5. To illustrate other ways to pass strings between Page Flows, add passString2 and passString3 to the Page Flow for portlet j2.
   - The passString2 method uses a request to send the data.
   - The passString3 method uses the same portlet communication approach as passString2; the only difference is that the j2 Page Flow sends data to its JSP portlet using an attribute.

```
/**
        * @jpf:action
        * @jpf:forward name="listening" path="listening.jsp"
        */
        public Forward passString2()
        {
thePassedText = getRequest().getParameter("string2");
        return new Forward( "listening" );
        }

        /**
```

```
        * @jpf:action
        * @jpf:forward name="listening" path="listening.jsp"
        */
        public Forward passString3()
        {
HttpServletRequest request = getRequest();
String attribValue = request.getParameter("string3");
request.setAttribute("string3", attribValue);

return new Forward( "listening" );
        }
```

Create Additional JSPs

In this step, we'll create the "From" JSP for the j1 portlet, and the "To" JSP used by the j2 portlet.

Create listening.jsp

1. In the Application palette, right-click on the portlets/j2 folder and select New JSP file.
2. Name the file listening.jsp.
3. Open the source view, and insert the following code:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %><%@
taglib uri="netui-tags-html.tld" prefix="netui" %>

<h3>"To" JSP</h3>

Get text from jpf form = <b><netui:label
value="{pageFlow.thePassedText}"/></b>

<br/>

<br/>

Using {request.string3} databinding = <b> <netui:label
value="{request.string3}"/>

</b> <br/>

<% String attribName = "string3"; %>

Using request.getAttribute() = <%=request.getAttribute(attribName)%>

<br/>
```

```
<br/>
```

4. Save listening.jsp.

Create simpleForm.jsp

1. In the Application palette, right-click on the j1 folder and select New JSP file.
2. Name this new JSP file simpleForm.jsp.
3. Switch to Design View.
4. From the Netui Palette, drag a TextBox object onto the form, then a Button object right below it. Use the Form object to wrap around both of them. Initially, the source for this portion of the jsp will look something like this:

```
<netui:form action="none">
<netui:textBox></netui:textBox>
<netui:button>Submit</netui:button>
</netui:form>
```

5. Edit the source of the fragment you created: Add the passString1 action to the form, and designate a dataSource attribute for the textBox:

```
<netui:form action="passString1">
<netui:textBox dataSource="text"/>
<netui:button>Submit</netui:button>
</netui:form>
```

6. From the Netui Palette, drag an anchor element just below the textBox. The Form Wizard appears. Invoke the passString2 action, labelling it "PassString2", or edit the source code as follows:

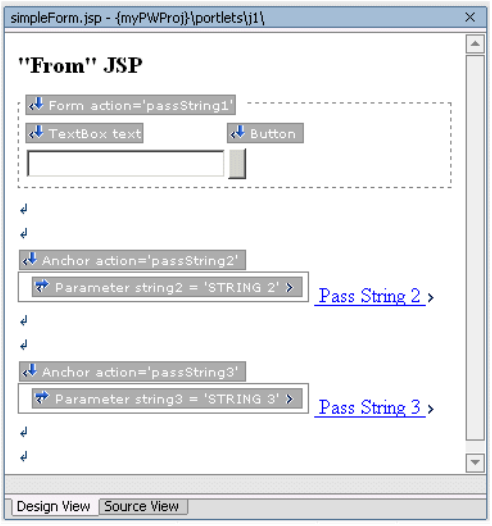
```
<netui:anchor action="passString2">Pass String 2
<netui:parameter name="string2" value="STRING 2"/>
</netui:anchor>
```

7. From the Netui Palette, drag one more anchor element just below the previous one, and edit the resulting code as follows:

```
<netui:anchor action="passString3">Pass String 3
<netui:parameter name="string3" value="STRING 3"/>
</netui:anchor>
```

8. The Design View of simpleForm.jsp should now look something like this:

Figure 1-14 Viewing the From JSP in Design View



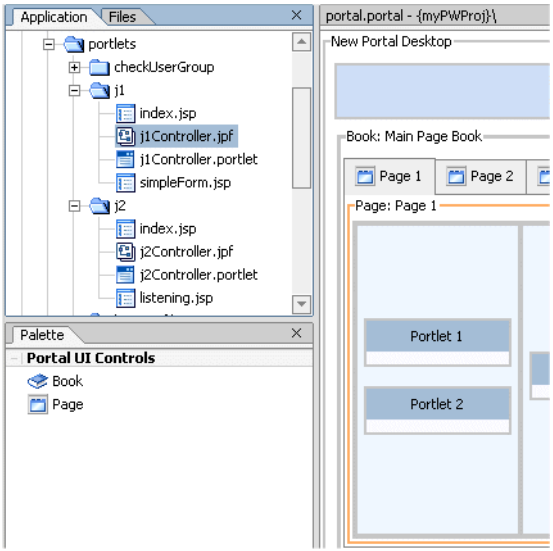
Note: As you edit the page flow, you can verify the navigation by opening a viewer that will preview the pages without the portal. To do this, click on the Start arrow from the WebLogic Workshop toolbar, or press **CTRL+F5**.

Place Portlets in Portal

To view the portlets, they need to be placed in a portal.

1. Open the portal created in Step 1 in Design View, select a placeholder, and drag the Page Flows into placeholders on a page, using the Portlet Wizard to create new portlets from each Page Flow.
2. Use the Property designer to edit the Title attribute for each portlet.
3. From the Portal designer, the portal should now look something like this:

Figure 1-15 Portlets Placed on a Page

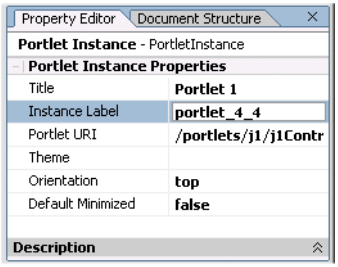


Set Properties on Portlets

When a portlet is placed inside a portal, it is assigned an instanceLabel which the framework uses to keep track of individual portlet placement. In order to make this sample work, the listenTo attribute on portlet j2 needs to be set to the instanceLabel of the j1 portlet on your portal.

1. Open the portal, select the j1 portlet and verify its instanceLabel. In this example, it is portlet_4_4.

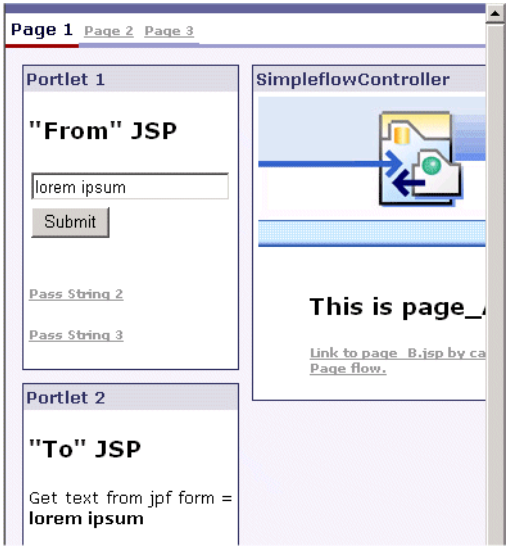
Figure 1-16 Portlet Instance Label Property



2. Now open the j2 portlet by double-clicking on it within the portal. The listenTo attribute for this portlet needs to be set to the instanceLabel for the j1 portlet in your portal.

3. Save all files and start the server.
4. Preview the portal by navigating to http://<host>:<port>YourWebappp/portal.portal.
5. The resulting portal should look something like this:

Figure 1-17 Previewing the Navigation Page Flow Portlets



Step 4: Create an EJB Control Page Flow portlet

In this step you create a Page Flow portlet, add a Personalization Control, and edit properties on the Control.

The tasks in this step are:

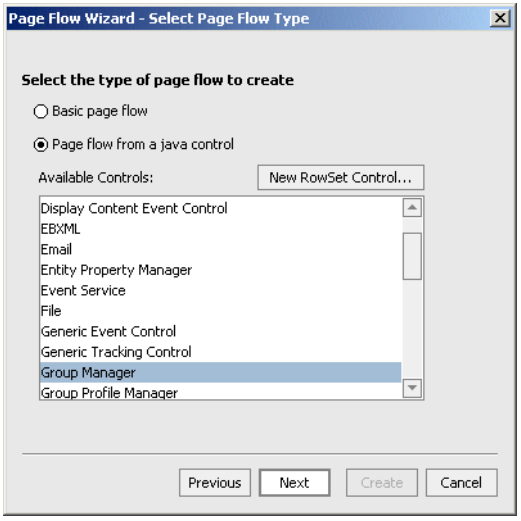
- Create a New EJB Control Page Flow
- Place the Page Flow into a Portlet, Place Portlet in a Portal

Create a New EJB Control Page Flow

1. From within the Project tab in your Web application, right-click on the portlets directory and create a new Page Flow. Name the Page Flow checkuserGroup.
2. In the Page Flow Wizard, select Page flow from a java control.

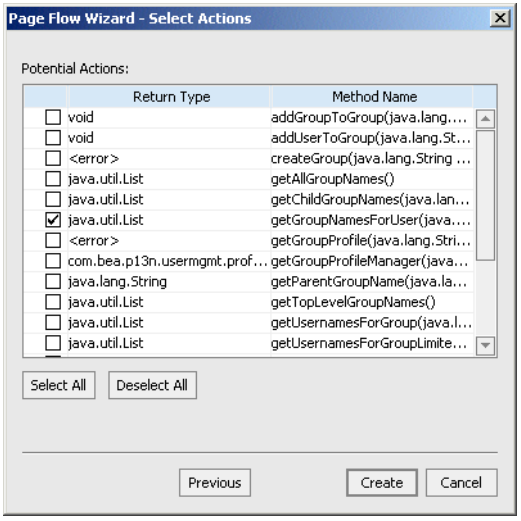
3. Select the GroupManager control. Click Next.

Figure 1-18 Select Control



4. From the list of Potential Actions, select getGroupNamesForUser. Click **Create**.

Figure 1-19 Select Actions

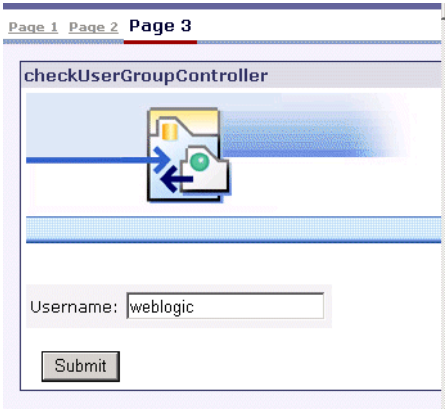


5. The wizard creates an index.jsp page with a link to the getGroupNamesForUser action that invokes the control.

Place the Page Flow into a Portlet, Place Portlet in a Portal

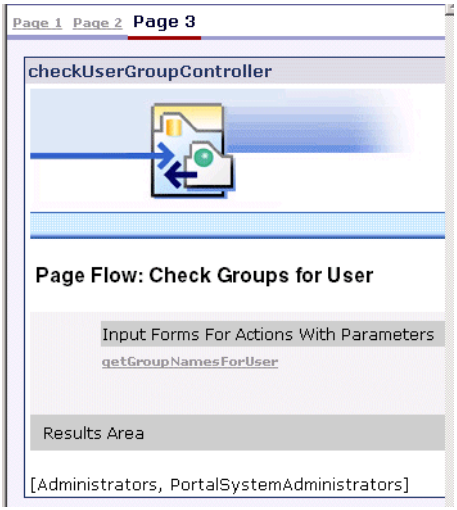
1. With the Portal open in Design View, drag the checkUserGroupController.jpf into a placeholder in the portal.
2. The Portlet Wizard presents a prompt offering to create a portlet from this resource. Click Yes.
3. The Portlet Details screen appears. Click Finish.
4. Drag the resulting portlet into a placeholder in your portal.
5. Preview the portlet by navigating to **http://<host>:<port>YourWebappp/portal.portal.**
6. Click on the getGroupNamesForUser link, enter a username and click **Submit**.

Figure 1-20 Entering User Name



7. In this example, user weblogic is a member of the groups Administrators and PortalSystemAdministrators.

Figure 1-21 Results of checkUserGroupController



Congratulations! You have succesfully placed a Portal EJB control inside a portlet.