Create a portal application

This tutorial describes how to use Rational Software Development Platform's portal tools to create and test a portal application. The application includes portlets that you will create using JavaServer Faces components to design the pages in the user interface. The portlets in the application use session bean data to provide user administration functions and deliver item listings and details for a sample auction site. Finally, these portlets are added to portal pages in a site that you will design.

In the course of the tutorial, we will touch on key features of portal technology, including portlet programming, inter-portlet communication, and portal look and feel elements.

Create a portal application tutorial introduction

Time required

To complete the modules in this tutorial, you will need approximately **2 hours**. If you decide to explore other facets of the portal tools capabilities described in the exercises in this tutorial, it could take longer to finish.

Prerequisites

In order to complete this tutorial end to end, it will be helpful if you are familiar with:

- The Web perspective
- Page Designer
- JSP coding and tag libraries
- JavaServer Faces technology
- JavaBeans and Session EJBs

It will also be beneficial if you understand the basics of portlet programming.

Before attempting this tutorial, ensure that you have installed the optional Portal Tools feature from the Rational installation pad and the WebSphere Portal V5.1 Test Environment, which you must install and configure separately from Rational tools.

Learning objectives

Module 1: Developing portlet applications teaches you how to create, modify, link, and test JavaServer Faces portlets. In this module, you will:

- Learn about how portlet projects and applications are organized
- Examine the types of resources and code that are generated by the portal tools wizards
- Create JavaServer Faces portlets using a Web Diagram to create the application flow
- Connect portlets to a data source using Session EJBs
- Create user interfaces for pages that display, create, and update user records
- Create a basic search results page, and a page that provides a detailed description of any items retrieved by the search page
- Send data from one page to another using cooperative portlets
- Test portlets as you create them in the WebSphere Portal Test Environment

Module 2: Developing portal applications shows you how to create a portal project, update design elements, and run the AuctionPortal application in the WebSphere Portal Test Environment, simulating application user activities.

When you are ready, begin Module 1: Developing portlet applications.

Module 1: Developing portlet applications

This module teaches you how to create JavaServer Faces portlets that will later added to an auction portal site that you design. In this module, you will:

- Learn about how portlet projects are organized
- Examine the types of resources and code that are generated by the portal tools wizards
- Create JavaServer Faces portlets using a Web Diagram to create the application flow
- Connect portlets to JavaBean data and a Session EJB
- Create user interfaces for pages that display, create, and edit user records
- Create a basic search results page that is dynamically linked to another page that provides details about items located by a search
- Send data from one page to another using cooperative portlets
- Test the portlets that you create in the WebSphere Portal test environment

Remember: Before beginning this module, you should have the prerequisite knowledge outlined in the tutorial introduction.

Exercises

The exercises within this module must be completed in order:

- Exercise 1.1: Setup
- Exercise 1.2: Preparing for portlet development and defining the application flow
- Exercise 1.3: Developing the UserAdmin page
- Exercise 1.4: Creating pages for creating and editing user information
- Exercise 1.5: Adding portlets that search Auction site listings and provide listing details

Time required

This module will take approximately **1 hour and 30 minutes** to complete.

Exercise 1.1: Setup

Before beginning portlet and portal development in this tutorial, there are a series of setup steps that you must perform:

1. Creating a project and importing resources required for portlet development.
2. Installing and configuring the WebSphere Portal v5.1 test environment, which will allow us to test the application that we are building.
3. Configuring a database server that will supply data to run in the application.

The first step is to import pre-built EJB and EJB client projects into the portlet project that we will use in the tutorial. The AuctionPortletEAR created when you create the portlet project will contain the imported resources. These projects provide the back-end services that we will use to create the portlet included in the portlet project.

Creating a new portlet project

All portlet development is performed in the context of a portlet project. To create the portlet project for the UserAdmin portlet, follow these steps:

1. Open the Rational Software Development Platform.
2. Select **File** > **New** > **Project**.
3. Select the **Portlet Project** option. This option will generate project resources that support the IBM portlet API, as opposed to the JSR 168 portlet API. Click **Next**.
4. Click **OK** in the **Confirm Enablement** dialog. This dialog verifies that you want to enable Portal Development *capabilities*. Capabilities are the elements of your development environment, along with the appropriate set of tools, that automate many of the programming standards and code generation capabilities required for developing portal-based Web applications that will run on WebSphere Portal.

5. Name the project `AuctionPortlet`.
   When you type in the name of the project, notice that (if you click the **Show Advanced >>** button) the module for this project is automatically added to a like-named EAR project.
6. Note that the **WebSphere Portal version** defaults to `5.1`. Click **Next**.
7. Select the **Faces Portlet** radio button, because the portlets that will be created in this tutorial will leverage the speed and flexibility provided by JavaServer Faces and the portlet tools in building user interfaces for Web applications. Click **Next**
8. Click **Next** again.
9. In the Portlet Settings page, type `UserAdmin portlet` in **Portlet name** field, and `User Admin` in **Portlet title** field.
10. Select the **Change code generation options** check box. Type `UserAdminPortlet` in **Class prefix** field.
11. Click **Next**.
12. In the Miscellaneous page, notice that the **Initial page** value in the **View mode** group is `/UserAdminView.jsp`. This default page will be generated within the project, and will serve as the master page of the UserAdmin portlet. We will add additional pages in future exercises.
13. Click **Finish**.

    If you are prompted to switch to the Web Perspective, click **Yes**.

The default portlet view page, UserAdminView.jsp, is displayed in the editing area of your workspace.

Importing the resources used in this sample

To import AuctionPortal.zip, which contains all pre-built project resources necessary to create the interfaces and data access described in the tutorial:

1. From the menu bar, select **File > Import**. The Import dialog appears.
2. Under **Select an import source**, click **Project Interchange**.
3. Click **Next**. The **Import Project Interchange Contents** dialog appears.
4. Because different Rational products use different installation target locations, you must leave the product's user interface to locate the plug-in that contains the database content. Use a file search tool to locate the `com.ibm.etools.portal.examples.application_6.0.0.1` plug-in folder under the product installation path in your local file system.
5. Next to **From zip file**, click **Browse** and navigate to **AuctionPortal.zip** in the following location:

   x:`\com.ibm.etools.portal.examples.application_6.0.0.1/samples`

   where *x:* is the absolute path that contains the `com.ibm.etools.portal.examples.application_6.0.0.1` plug-in on your computer. Click **Open**.
6. Select the **AuctionEJB50** and **AuctionEJB50Client** check boxes.
7. Click **Finish**.

   The wizard imports the projects into your workspace.

8. You must also add the imported projects as modules to the AuctionPortletEAR project. In the Project Explorer, expand Enterprise Applications > AuctionPortletEAR and double-click **Deployment Descriptor: AuctionPortletEAR**.
9. Select the **Module** tab.
10. Click **Add** under Modules. Select **AuctionEJB50** and click **Finish**. If the **Change Target Server** dialog asks you to change the module target server to the EAR target server, click **Yes**.
11. Click **Add** under Project Utility JARs. Select **AuctionEJB50Client** and click **Finish**. If the **Change Target Server** dialog asks you to change the module target server to the EAR target server, click **Yes**.
12. Save and close the Deployment Descriptor.

Installing and configuring the WebSphere Portal v5.1 test environment

Although the WebSphere Portal v5.0 test environment is installable from the product launchpad, the resources in this tutorial are designed to run on WebSphere Portal 5.1. You must install the WebSphere Portal 5.1 test environment as described in the product *Installation Guide*. The installation requires that you use WebSphere Portal media that is provided with Rational tools.

After installing this version of the test environment, you must also configure it to run in your workspace:

1. Select **Window** > **Preferences** from the menu bar.
2. Expand the **Server** section, and click **Installed Runtimes**.
3. Click **WebSphere Portal v5.1 stub**, select the check box, and click **Edit**.
4. Change the name to `WebSphere Portal v5.1` and overwrite the **WebSphere Portal Location** and **WebSphere Application Server Location** values with the WebSphere Portal 5.1 test environment installation directories that you specified when you installed the test environment.
5. Click **Finish**.
6. Click **OK** in the Preferences dialog box.

After completing these steps, the WebSphere Portal v5.1 test environment server is available for testing, debugging, or profiling.

Next, you must configure a database server that contains the data used by the applications in this tutorial.

Creating and configuring a server that contains a sample application database

In this exercise, you must import and configure a Cloudscape database to provide data for the AuctionPortal application.

1. Display the Servers view. It is located near the Properties and Quick Edit views.
2. Select **New > Server** from the view's pop-up menu.
3. Select **WebSphere Portal v5.1 Test Environment** from the server type list.
4. Click **Finish**.
5. Double-click **WebSphere Portal v5.1 Test Environment** in the Servers view.
6. Select the **Data sources** tab in the server editor.
7. In the **Server Setting** section, click **Add** to add a listing to the JDBC provider list.
8. In the first page of the Create a JDBC Provider wizard, select `Cloudscape` as the **Database type**, and `Cloudscape JDBC Provider` as the **JDBC provider type**.
9. Click **Next**.
10. Type `Cloudscape JDBC Provider` in the **Name** field.
11. Click **Finish**.
12. Click the **Add** button next to the **Data source defined in the JDBC provider selected above** list.
13. In the first page of the Create a Data Source wizard, select `Cloudscape JDBC Provider`, and click **Next**.
14. Type `Auction` in the **Name** field.
15. Type `jdbc/auction` in the **JNDI name** field.
16. Click **Next**.
17. Select `databaseName` from the **Resource Properties** list.
18. Because different Rational products use different installation target locations, you must leave the product's user interface to locate the plug-in that contains the database content. Use a file search tool to locate the `com.ibm.etools.auction.sampledb_6.0.0` plug-in folder under the product installation path in your local file system.
19. Return to the Create a Data Source wizard, and type the following path in the **Value** field:

    x:`\com.ibm.etools.auction.sampledb_6.0.0\db\AUCTION60`

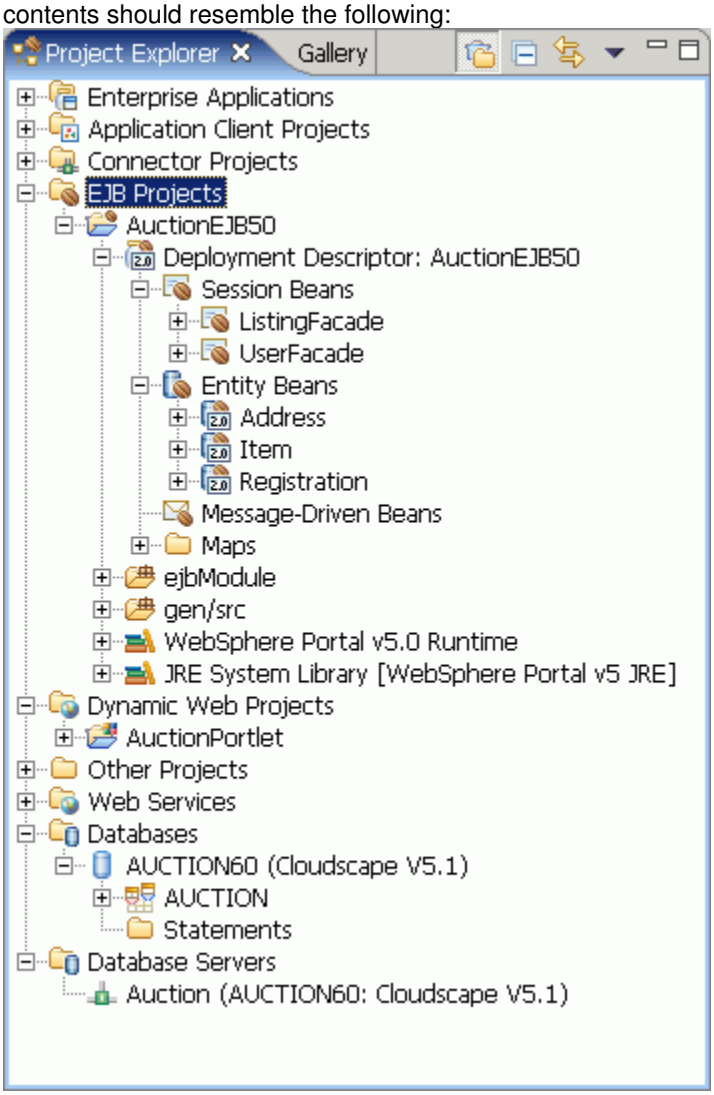
    where *x:* is the absolute path that contains the `com.ibm.etools.auction.sampledb_6.0.0` plug-in on your computer.
20. Click **Finish**.

21. Save and close the server configuration file.

Now you will be able to access data in the AUCTION60 database when you test the portlets in the AuctionPortlet project, as they are developed in Module 1.

About the files used in this tutorial

You can now browse the files in the AuctionEJB50 and AuctionEJB50Client projects, which includes the data sources and logic for the portlet pages. In the project Explorer view, when you expand the folders, the contents should resemble the following:



The resources in this project will be used to create the user interfaces for the portlets that we will create in this tutorial, and provide access to the data in the sample Cloudscape database. UserFacade and ListingFacade are the EJB session beans that will supply methods that will be called from the portlets created in the tutorial. RegistrationData and ItemData are data access beans that will be used to carry data between portlets and the Cloudscape database, using entity beans. The EJB project provides database mappings for the entity beans.

Now you are ready to begin Exercise 1.2: Preparing for portlet development and defining the application flow.

Exercise 1.2: Preparing for portlet development and defining the application flow

Before you begin, you must complete Exercise 1.1: Setup.

In the previous exercise, we created the AuctionPortlet project to contain the portlets that will be developed in this tutorial. To prepare for creating the user interface and adding data to specific Faces portlets, we will define the application flow using the Web Diagram editor. A Web diagram is only one method for defining the application flow, but it is especially suited for visual development of Faces-based portlet applications.

Creating a Web diagram to construct the application flow

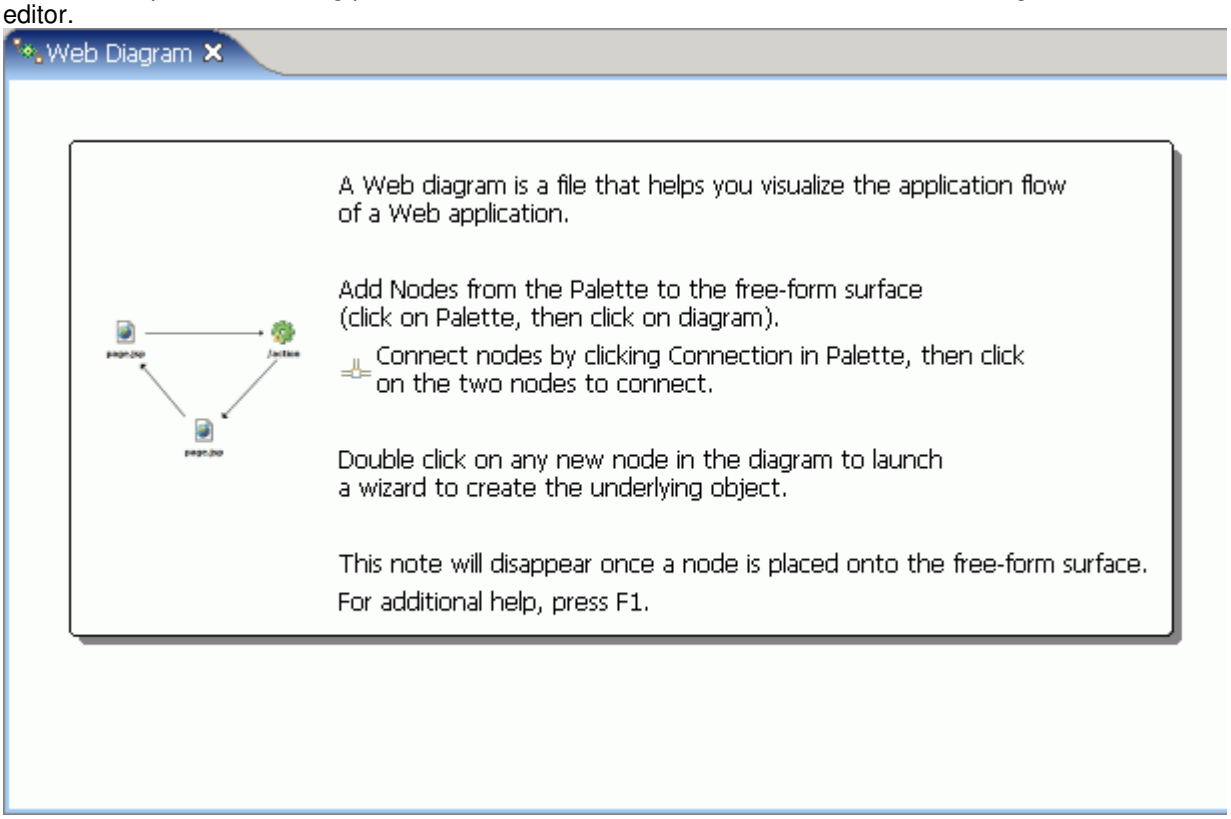
A Web diagram is a file that helps you visualize and change the flow of a Web application such as a Faces or Struts-based portlet application. The Web diagram editor is a visual editor for editing Web diagrams. When you add nodes, connections, and other components to a Web diagram, it is referred to generically as *drawing* the diagram.

A Web diagram consists of *nodes* and *connections* between nodes. A node is an icon that represents a resource such as a portlet JSP page or a Java™ bean. If the resource exists, the node is called *realized*; if the resource does not exist, the node is called *unrealized*. Realized nodes are shown in color with their names in boldface. Unrealized nodes are shown as gray icons.

We will add three JSP nodes to the Web diagram. One will be considered a master node, which will allow the navigation to two linked details nodes.

1. When you created the Auction portlet project, the wizard automatically created a default Web diagram file called diagram.gph. In the Project Explorer, expand the Dynamic Web Projects and AuctionPortlet folders, and then double-click Web Diagram

   The file will open in the editing pane with some instructions about how to use the Web diagram editor.



2. The active view on the right-side of the workspace is a palette, which contains a number of drawers that contain objects that you can drop on the Web diagram. You will add **Web page** objects onto the UserAdmin.gph file from the palette. The palette provides "sticky" drag and drop behavior, meaning that after you click on the object in the palette, you don't have to hold down the mouse button during the dragging operation. Just move the cursor to the Web diagram, and click again to drop the object.

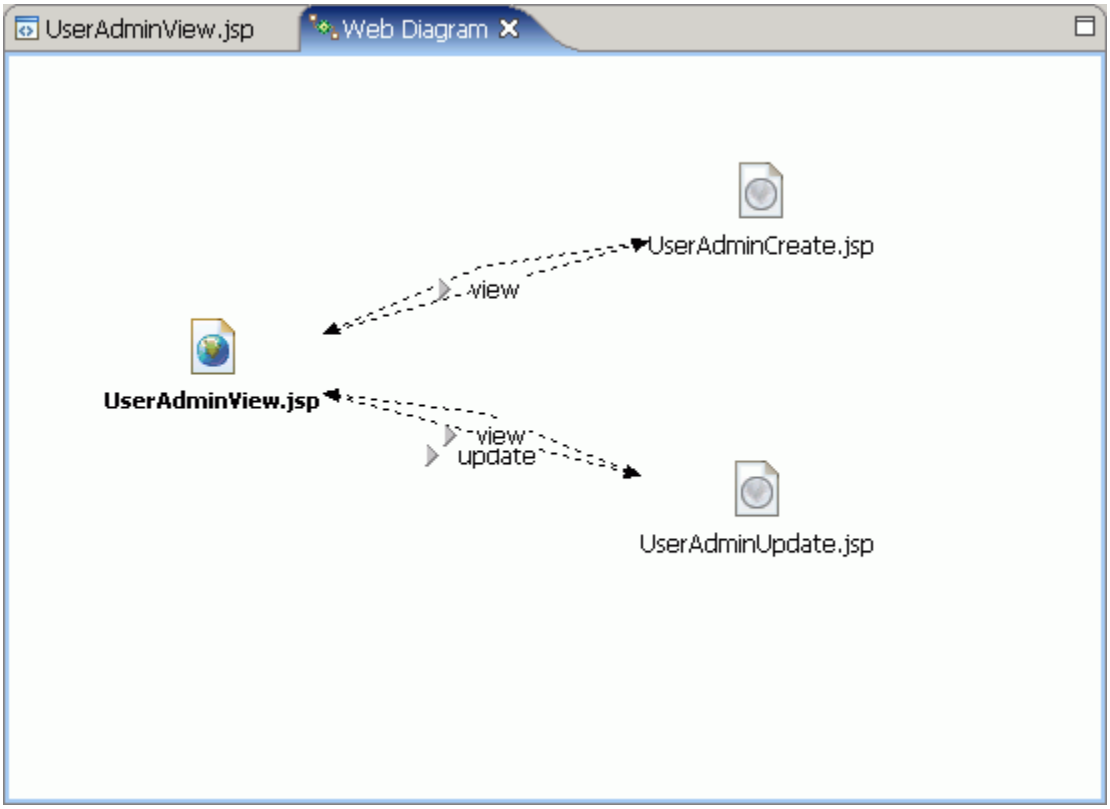
   To create the first node, you can drag `UserAdminView.jsp`, the initial JSP view page created with the AuctionPortlet project, from the Project Explorer to the editor. Because this file has already been created, it is displayed as a realized object.

3. Drag and drop a **Web page** object to the editor. Change the name of the Web page to `UserAdminCreate.jsp`.
4. Repeat the previous step to create a **Web page** called `UserAdminUpdate.jsp`.

   Note: You can drag the nodes to re-position them.

5. Next you will create connections between the Web page nodes to specify the flow of data in the portlets:
   a. Select **Connection** from the UserAdminView.jsp pop-up menu. From UserAdminView.jsp, drag the connection to the UserAdminCreate.jsp node.
   b. Select **Web Page Link** from the **Choose a connection** dialog, and click **OK**.
   c. Select **Connection** from the UserAdminCreate.jsp node pop-up menu, and drag the connection back to the UserAdminView.jsp node.
   d. Select **Faces Outcome** from the **Choose a connection** dialog, and click **OK**.
   e. Type `view` over the highlighted `<new>` to name the new connection.
   f. Select **Connection** from the UserAdminView.jsp pop-up menu. From UserAdminView.jsp, drag the connection to the UserAdminUpdate.jsp node.
   g. Select **Faces Outcome** from the **Choose a connection** dialog, and click **OK**.
   h. Type `update` over the highlighted `<new>` to name the new connection.
   i. Select **Connection** from the UserAdminUpdate.jsp node pop-up menu, and drag the connection back to the UserAdminView.jsp node.
   j. Select **Faces Outcome** from the **Choose a connection** dialog, and click **OK**.
   k. Type `view` over the highlighted `<new>` to name the new connection.
   l. Save the Web diagram.

   The diagram should look similar to this:



The links that have been defined will prime the portlet pages with the proper linkages, so that user-initiated data will flow through the application properly. When the UserAdminView page requests a user creation or update action, the appropriate form is opened. After the necessary information is submitted by these forms, the new user information becomes available to the UserAdmin portlet query function.

Now you are ready to begin Exercise 1.3: Developing the UserAdmin page.

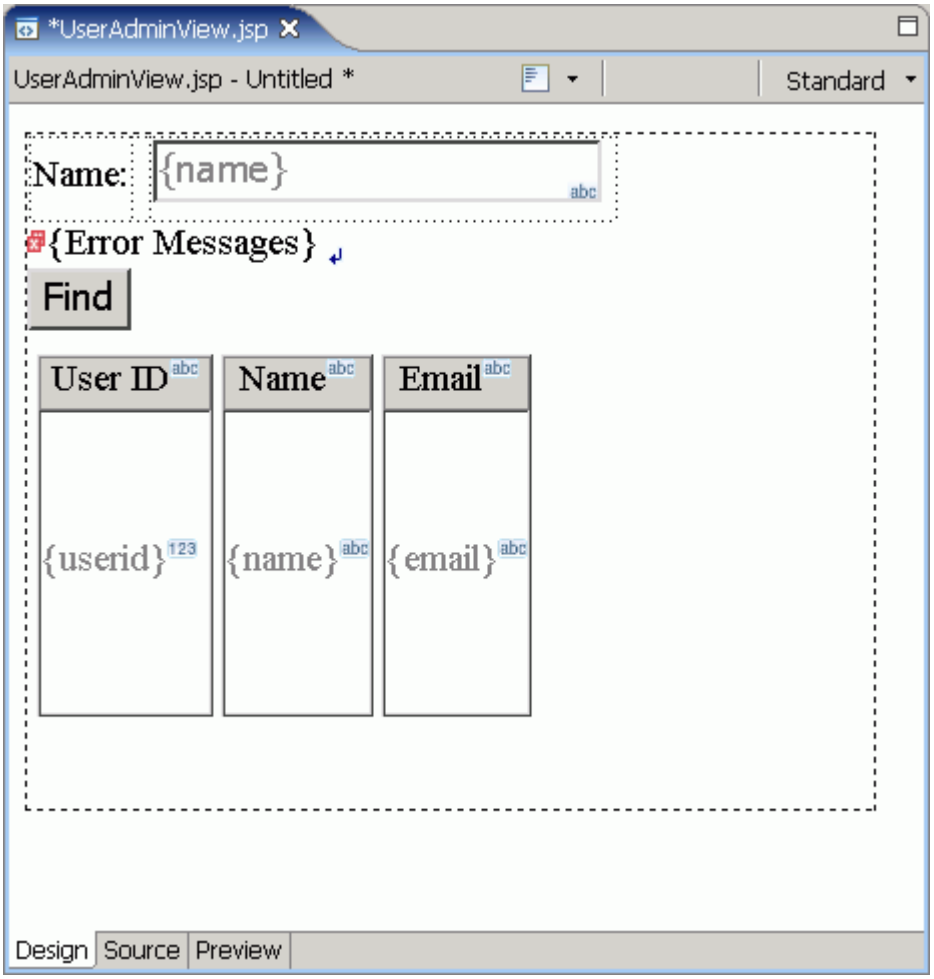
Exercise 1.3: Developing the UserAdmin page

Before you begin, you must complete Exercise 1.2: Preparing for portlet development and defining the application flow.

Create the UserAdminView.jsp page

Creating the UserAdminView.jsp page involves designing the user interface and adding a connection to dynamic data, an EJB session reference, into the application's logic.

1. If UserAdminView.jsp is not open in Page Designer, you can double-click the UserAdminView.jsp icon in the Web diagram editor to open it.
2. Delete the default `Place content here.` text.
3. Drag the **EJB Session bean** object from the **Data** drawer in the palette to the file.
4. When the Session Bean wizard opens, click the **New EJB Reference** button.
5. Expand the AuctionPortletEAR and AuctionEJB50 folders, and select `UserFacade` to create the enterprise bean reference.
6. Click **Finish**.
7. Click **Next** in the Session Bean wizard.
8. Select the `findByName(String name)` interface, which will be used for the input field on the portlet page.
9. Click **Next**.
10. Click the **Options** button in the Input Form page, and type `Find` in the **Label** field. Click **OK**.
11. Click **Next**, which should bring you to the Results Form page of the wizard. In this page, you will define the data table that will retrieve and display the data from the database.
12. Click **None** to deselect all of the columns, so that you can individually select, organize, and configure the appropriate columns for the data table to be used in the portlet page. Then, select the check boxes for the following columns:
    - userid
    - name
    - email
13. Using the up down arrow buttons, move the selected data columns into the order shown in the step above.
14. Select and change the **Label** value for the userid column to `User ID`.
15. Click **Finish** to generate the default user interface for the UserAdminView.jsp page. The user interface will look similar to the following:



16. Save UserAdminView.jsp.

Add Java page code to the UserAdmin page

In this portion of the exercise, you will add Java page code to accomplish the following:

- Store the `name` parameter in the session scope, so that it can be reused for any future refresh of the portlet content.
- Initialize the parameter to be displayed in the `Name` input field with the one stored in session scope.
- Initialize the result data using `name` parameter stored in the session scope.

You can add the EJB reference logic and the code to bind the invocation and results to the user interface, using the following steps:

1. Select **Edit Page Code** from the pop-up menu in Page Designer.
2. Type the following **bold** code into doUserFacadeLocalFindByNameAction():

```
public String doUserFacadeLocalFindByNameAction() {
    String name = getUserFacadeLocalFindByNameParamBean().getName();
    getSessionScope().put("name", name);
    try {
        userFacadeLocalFindByNameResultBean =
                getUserFacadeLocal().findByName(name);
    } catch (Exception e) {
        logException(e);
    }
    return null;
```

```
}
```

3. Type the following **bold** code into getUserFacadeLocalFindByNameParamBean():

```
public UserFacadeLocalFindByNameParamBean
            getUserFacadeLocalFindByNameParamBean() {
    if (userFacadeLocalFindByNameParamBean == null) {
        userFacadeLocalFindByNameParamBean =
               new UserFacadeLocalFindByNameParamBean();
        String name = (String)getSessionScope().get("name");
        userFacadeLocalFindByNameParamBean.setName(name);
    }
    return userFacadeLocalFindByNameParamBean;
}
```

4. Type the following **bold** code into getUserFacadeLocalFindByNameResultBean():

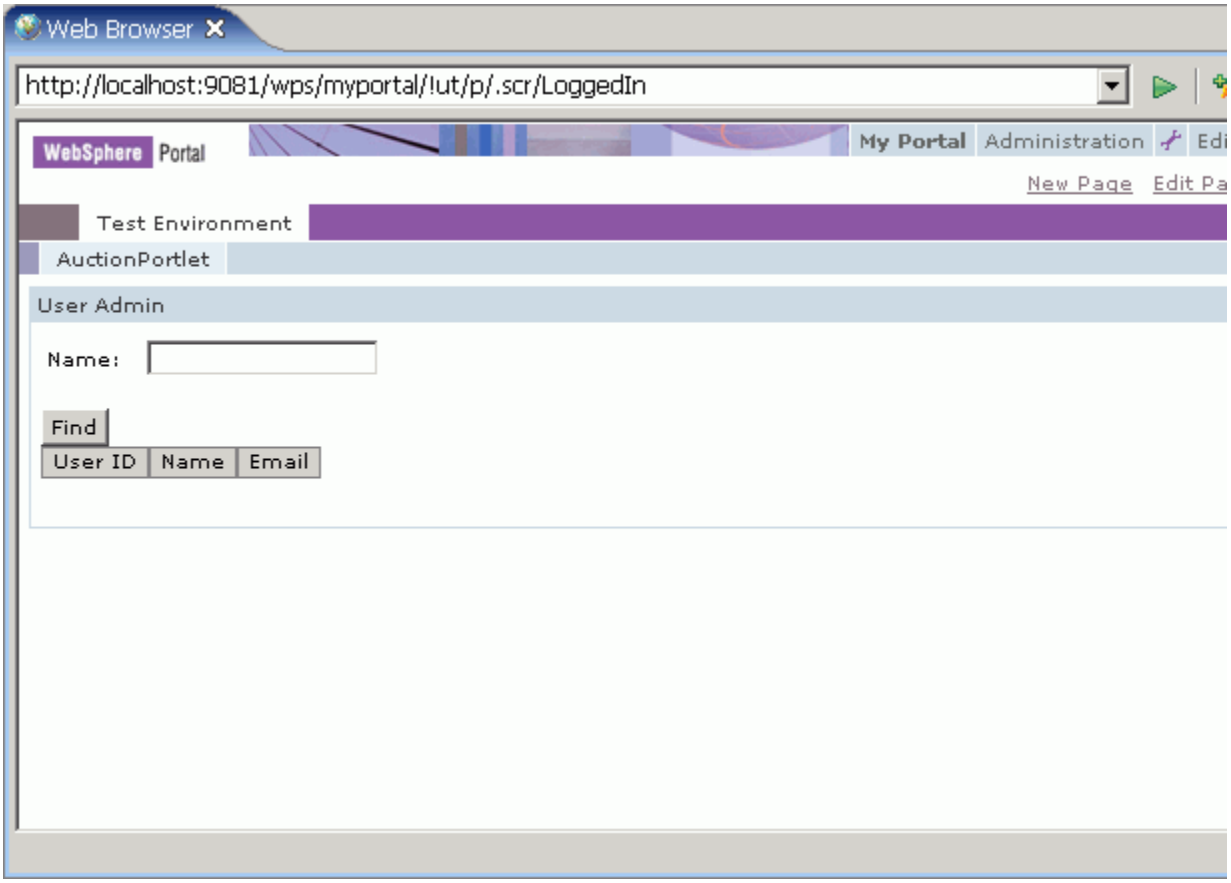
```
public RegistrationData[] getUserFacadeLocalFindByNameResultBean() {
    if (userFacadeLocalFindByNameResultBean == null) {
        String name = (String)getSessionScope().get("name");
        if (name != null) {
            try {
                userFacadeLocalFindByNameResultBean =
                 getUserFacadeLocal().findByName(name);
            } catch (Exception e) {
                logException(e);
            }
        }
    }
    return userFacadeLocalFindByNameResultBean;
}
```

5. Save and close UserAdminView.java.

Run UserAdminView.jsp

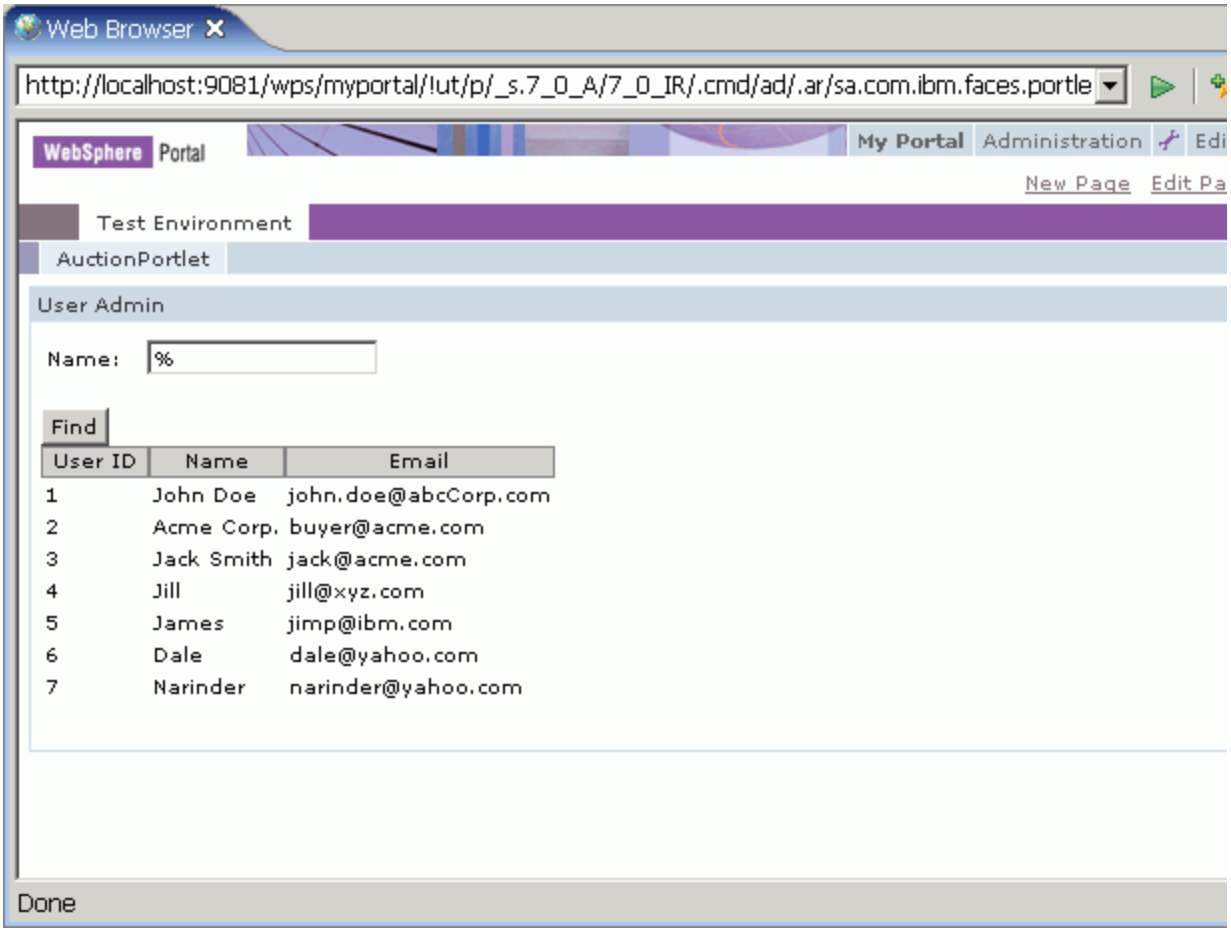
To verify that the UserAdmin portlet is working as intended up to this point, you should run the portlet on the internal browser provided with Rational tools. To run the portlet, do the following:

1. Select the AuctionPortlet project in the Project Explorer, and select **Run > Run on server** from its pop-up menu.
2. Because you have already defined the WebSphere Portal v5.1 test environment, select it, and click **Finish** in the Server Selection wizard.
3. The file will eventually display in the browser. Here you can see the input fields and layout that a user would see on a portal site:



4. To test the input form and the data table that you have just created, type `%` in **Name** field and click the **Find** button.

   The data table should display all users.



Before we move to the next exercise, it is advisable to stop the test environment server to improve performance during development. To stop the test environment server, simply select it in the Servers view, and click the **Stop the server** tool bar button  .

Now you are ready to begin Exercise 1.4: Creating pages for creating and editing user information.

Exercise 1.4: Creating pages for creating and updating user information

Before you begin, you must complete Exercise 1.3: Developing the UserAdmin page.

Based on the application flow that you defined the Web diagram, there should be separate UserAdmin pages for creating user information (UserAdminCreate.jsp) and updating existing user information (UserAdminUpdate.jsp).

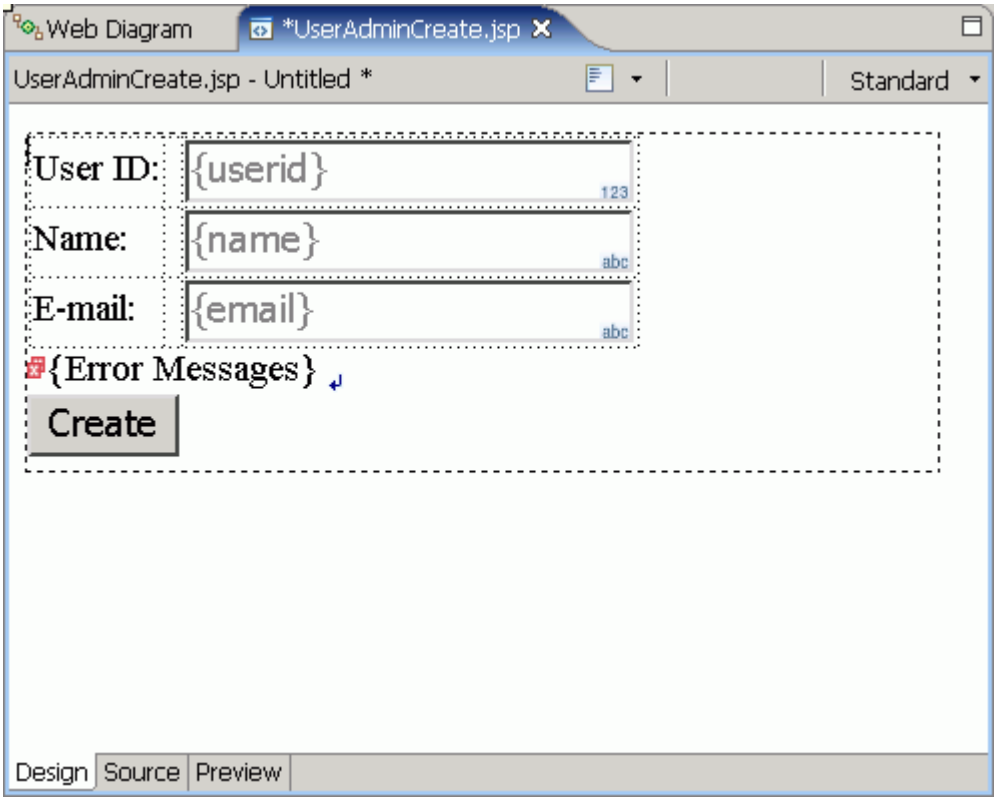
Create the UserAdminCreate.jsp page

In this portion of the exercise, using a session bean you will create data access in the UserAdminCreate page that enables the administrator to create new user records. Because a user ID value is mandatory for creating new user records, this page also incorporates validation logic. In addition, you will add a Cancel button to allow a user to end this process.

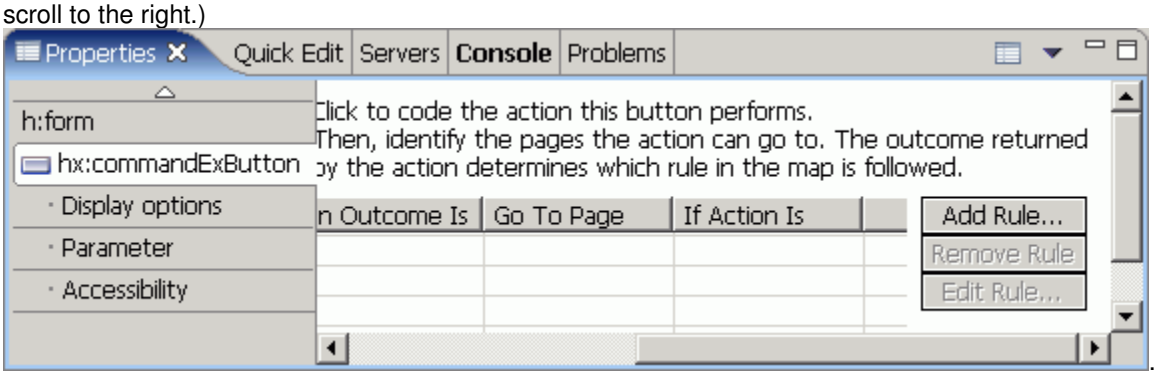
1. Return to the Web diagram editor and double-click the UserAdminCreate.jsp page.
2. Click **Finish** in the creation wizard.

   The empty file opens in the editing area.

3. Delete the default `Place content here.` text.
4. Drag the **EJB Session bean** object from the **Data** drawer in the palette to the file.
5. When the Session Bean wizard opens, select `ejb/UserFacade` and click **Next**.
6. Select the `create(RegistrationData data)` interface, which will be used for the input field on the portlet page.
7. Click **Next**.
8. In the Input Form page, you will define the input form that will create new data in the database. Click **None** to deselect all of the fields, so that you can individually select, organize, and configure the appropriate fields for the input form to be used in the portlet page. Then, select the check boxes for the following fields:
   - data.userid
   - data.name
   - data.email
9. Using the up down arrow buttons, move the selected data fields into the order shown in the step above.
10. Select and change the **Label** value for the data.userid field to `User ID:`.
11. Click the **Options** button, and type `Create` in the **Label** field. Click **OK**.
12. Click **Finish** to generate the input form in the UserAdminCreate.jsp page. The user interface will look similar to the following:



13. Select the **Create** button in the page.
14. Open the Properties view if it is not already active, and click the **Add Rule** button. (You may need to scroll to the right.)



15. In the **Add Navigation Rule** dialog, select `UserAdminView.jsp` from the **Page** list box.
16. Select the **The outcome named** radio button and type `view`.
17. Click **OK**.
18. Click the **Quick Edit** view tab. Click in the Quick Edit editing area, and type "view" in the return string:

```
return "view";
```

Add Validation logic

Because the user ID value is mandatory input for this page, we should add validation to check the user ID is specified.

1. Select the `User ID:` Input component.
2. Open the Properties view.
3. Select the **Validation** tab.

4. Click **Value is required** check box.
5. Save UserAdminCreate.jsp.

You do not have to write any of the code necessary to run this portion of the application. All of the EJB reference logic and the code to bind the invocation and results to the user interface is generated.

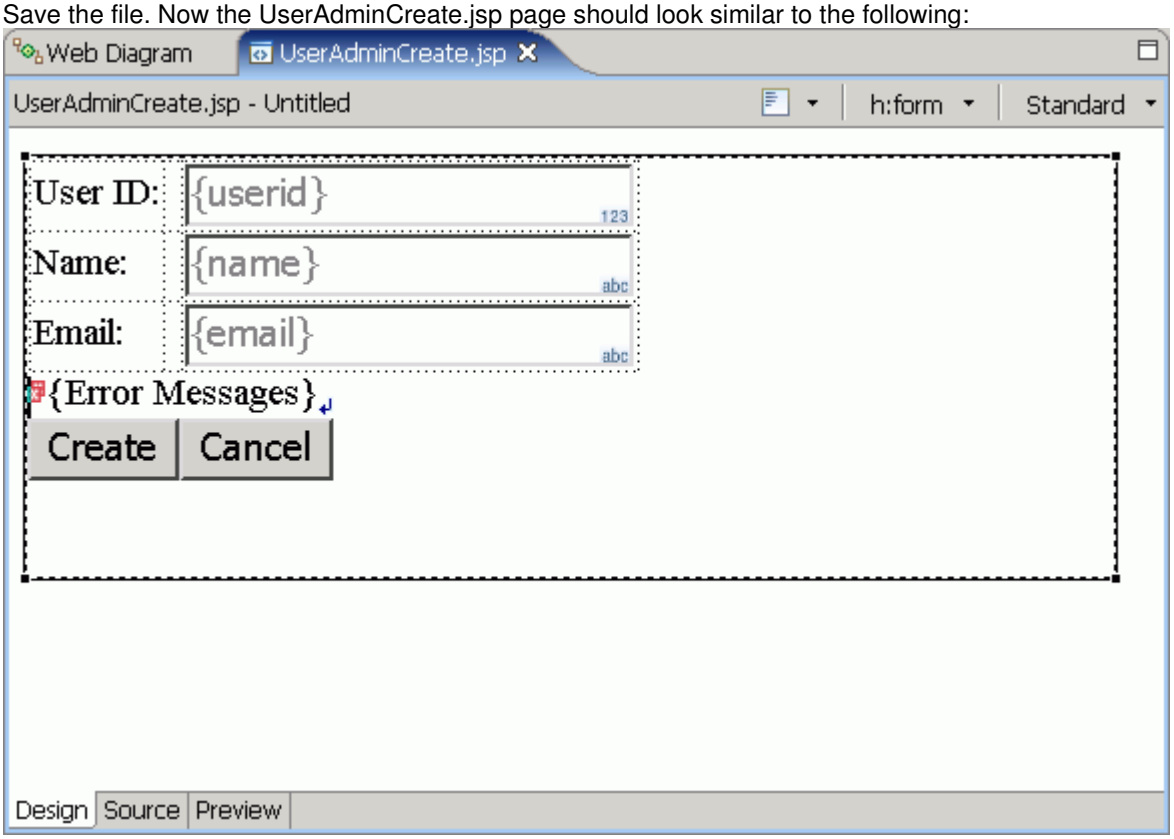
Add Cancel logic

In addition, you should add a Cancel button, along with the appropriate navigation wiring to the search page:

1. Drag a **Command - Button** from the **Faces Components** palette drawer, and drop it next to **Create** button.
2. Open the Properties view if it is not already active.
3. Click **Display options** tab in the set of properties, and type `Cancel` in the **Button label** field, and press Enter.
4. Click the **Quick Edit** view tab. Click in the Quick Edit editing area, and type "view" in the return string:

```
return "view";
```

5. Save the file. Now the UserAdminCreate.jsp page should look similar to the following:



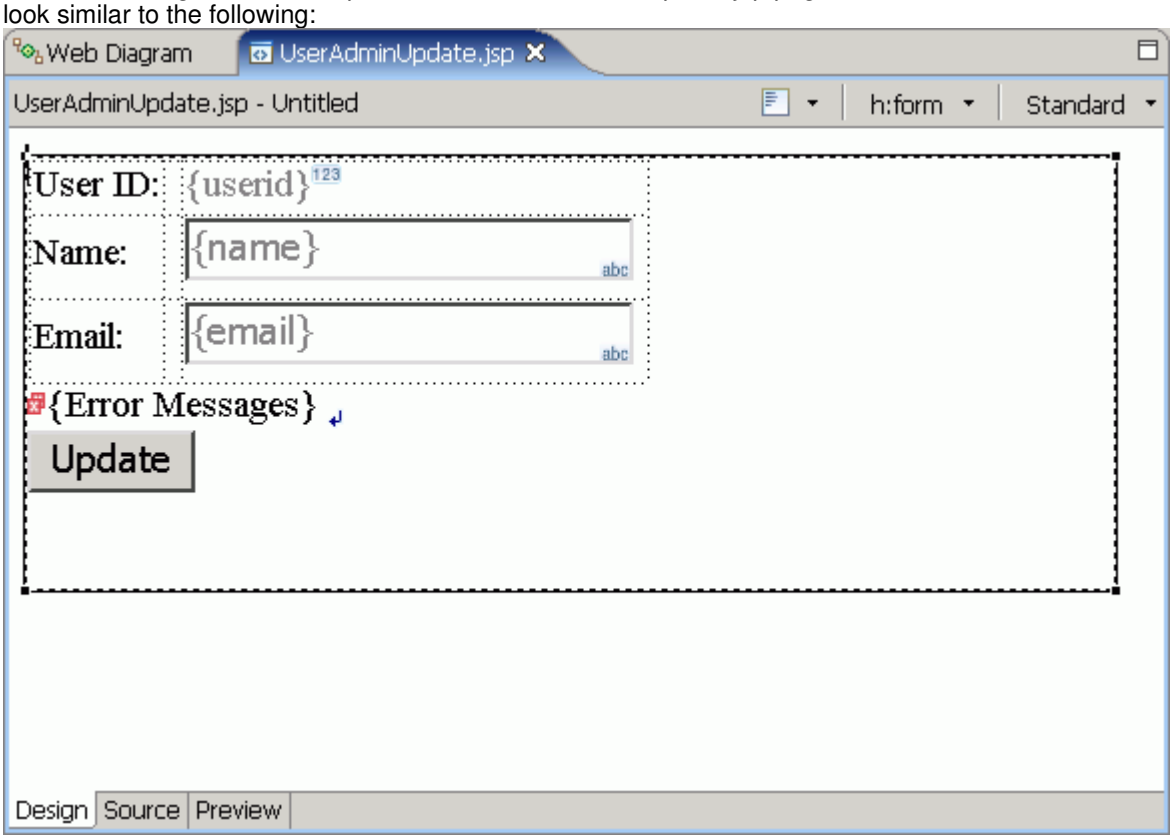
Create the UserAdminUpdate.jsp page

In this portion of the exercise, using a JavaBean you will create data access for updating user information in the UserAdminUpdate page. In addition, you will add a Cancel button to allow a user to end this process, and add additional code to obtain existing records so that they can be updated in this page.

1. Return to the Web diagram editor and double-click the UserAdminUpdate.jsp page.
2. Click **Finish** in the creation wizard.

The empty file opens in the editing area.

3. Delete the default `Place content here.` text.
4. Drag the **EJB Session bean** object from the **Data** drawer in the palette to the file.
5. When the Session Bean wizard opens, select `ejb/UserFacade` and click **Next**.
6. Select the `update(RegistrationData data)` interface, which will be used for the input field on the portlet page.
7. Click **Next**.
8. In the Input Form page, you will define the input form that will update data in the database. Click **None** to deselect all of the fields, so that you can individually select, organize, and configure the appropriate fields for the input fields to be used in the portlet page. Then, select the check boxes for the following fields:
    - data.userid
    - data.name
    - data.email
9. Using the up down arrow buttons, move the selected data fields into the order shown in the step above.
10. Select and change the **Label** value for the data.userid field to `User ID:`.
11. Select **Output Field** from the **Control Type** list for data.userid, because you want to avoid changing the ID when updating other information in the portlet.
12. Click the **Options** button, and type `Update` in the **Label** field. Click **OK**.
13. Click **Finish** to generate the input form in the UserAdminUpdate.jsp page. The user interface will look similar to the following:



14. Select the **Update** button in the page.
15. Open the Properties view if it is not already active, and click the **Add Rule** button. (You may need to scroll to the right.)
16. In the **Add Navigation Rule** dialog, select `UserAdminView.jsp` from the **Page** list box.
17. Select **The outcome named** radio button and type `view`.
18. Click **OK**.
19. Click the **Quick Edit** view tab. Click in the Quick Edit editing area, and type "view" in the return string:

```
return "view";
```

You do not have to write any of the code necessary to run this portion of the application. All of the EJB reference logic and the code to bind the invocation and results to the user interface is generated.

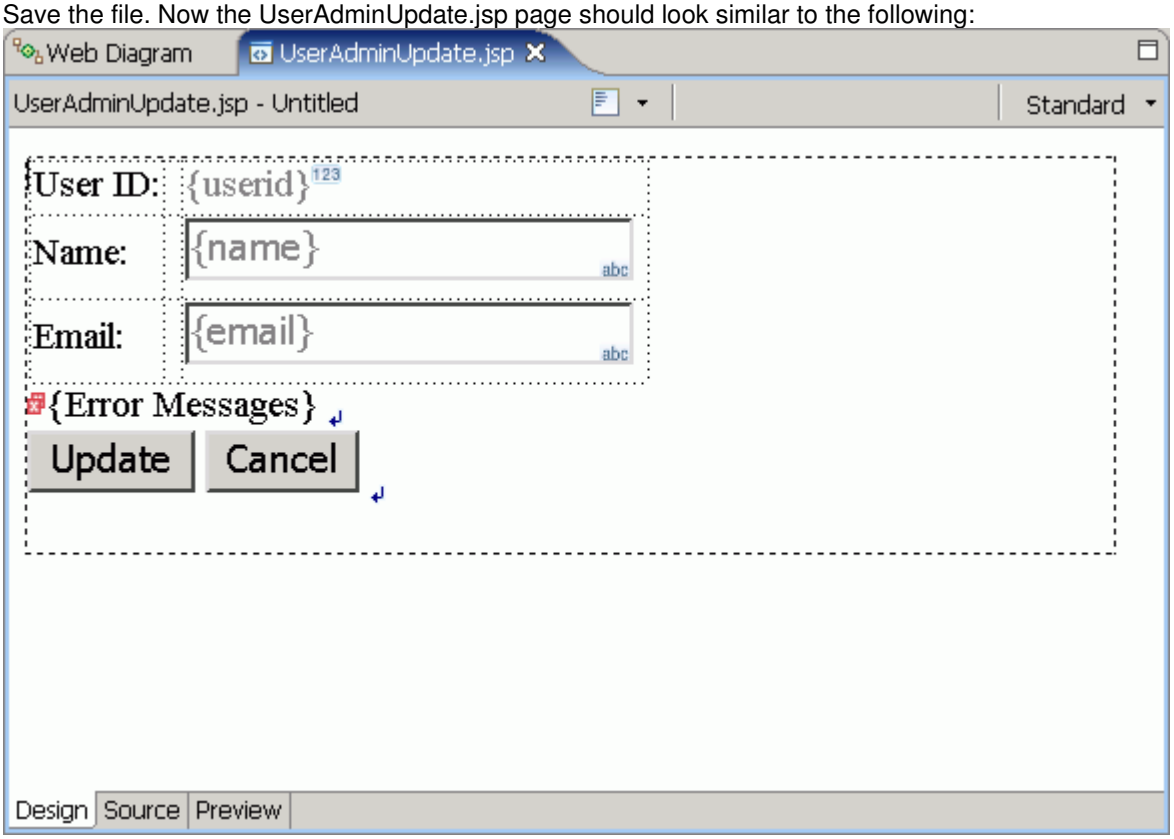
Add Cancel logic

In addition, you should add a Cancel button, along with the appropriate navigation wiring to the search page:

1. Drag a **Command - Button** from the **Faces Components** palette drawer, and drop it next to the **Update** button.
2. Open the Properties view if it is not already active.
3. Click **Display options** tab in the set of properties, and type `Cancel` in the **Button label** field, and press Enter.
4. Click the **Quick Edit** view tab. Click in the Quick Edit editing area, and type "view" in the return string:

```
return "view";
```

5. Save the file. Now the UserAdminUpdate.jsp page should look similar to the following:



Add additional code for updating existing records

The update logic added in this portion of the exercise enables the application to obtain existing records, so that a user can update them with this page. The form is filled with relevant data by initializing the parameter bean using a method provided by UserFacade, `findById()`.

1. Select **Edit Page Code** from the UserAdminUpdate.jsp pop-up menu. The page code is the Java file that contains the underlying logic for UserAdminUpdate.jsp. Insert the following code (the bold-faced

portion):

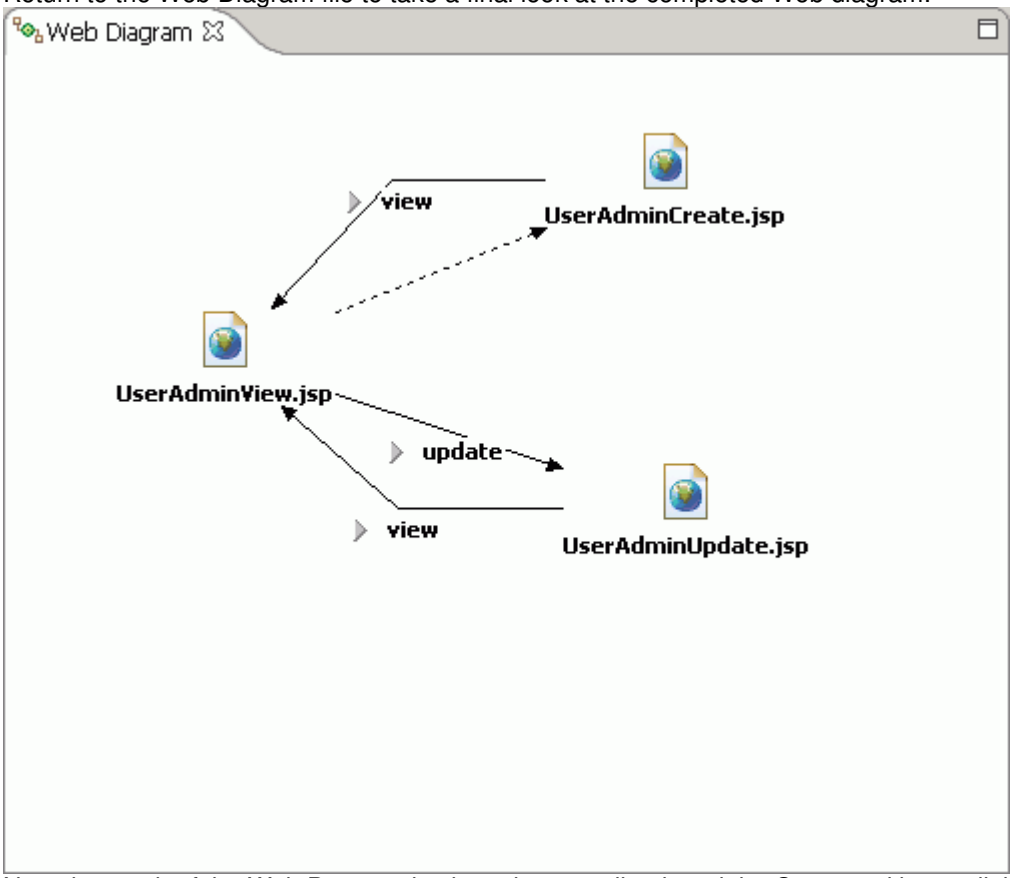
```
public UserFacadeLocalUpdateParamBean getUserFacadeLocalUpdateParamBean() {
  if (userFacadeLocalUpdateParamBean == null) {
     userFacadeLocalUpdateParamBean = new UserFacadeLocalUpdateParamBean();
     Integer userid = (Integer)getSessionScope().get("userid");
     try {
     userFacadeLocalUpdateParamBean.setData(getUserFacadeLocal()
                                            .findById(userid));
     } catch (Exception e) {
         logException(e);
     }
  }
  return userFacadeLocalUpdateParamBean;
}
```

2. Save and close UserAdminUpdate.java.

Add links for page navigation

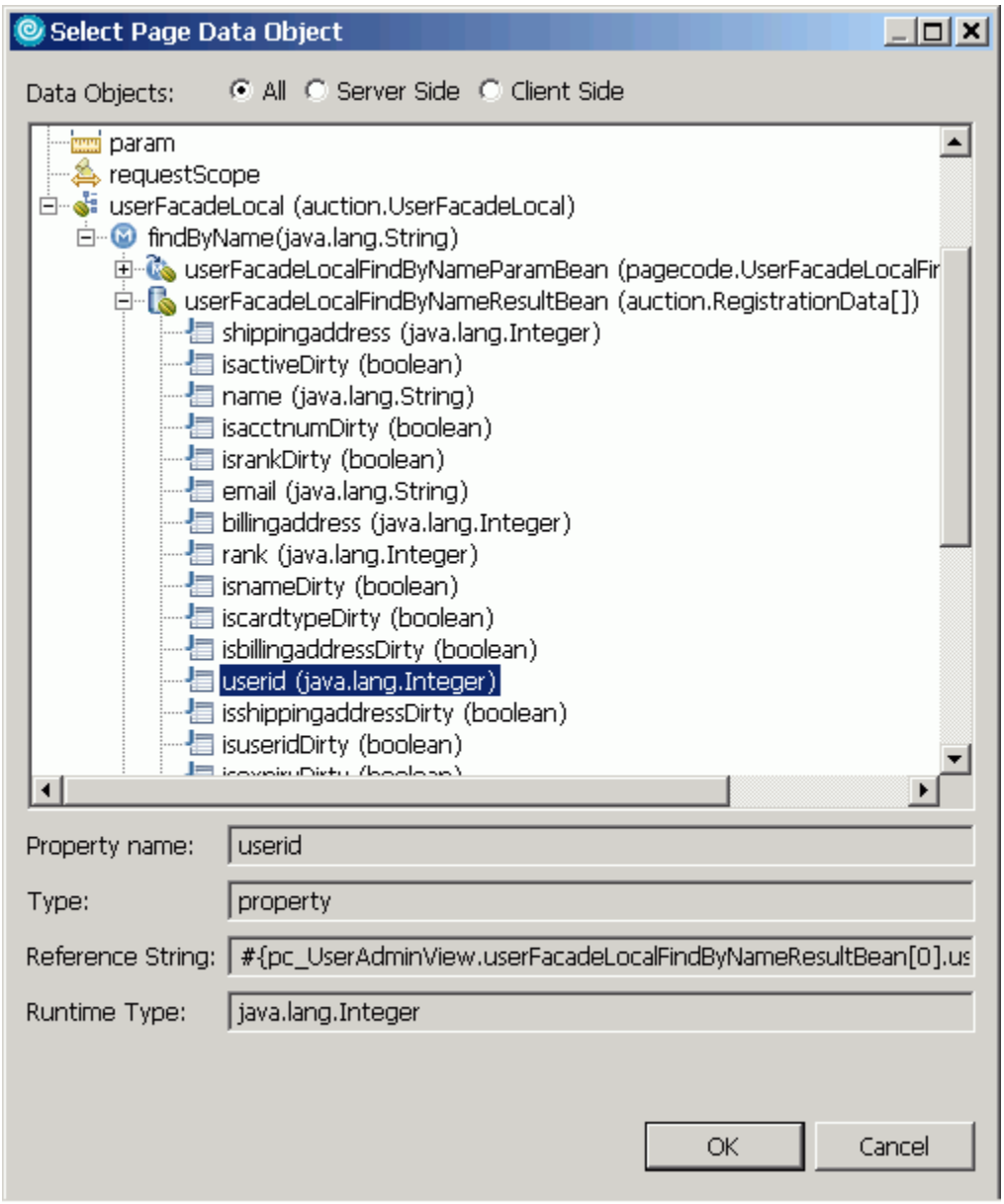
Finally, to complete the UserAdmin portlet, you must add links to provide navigation from the master page (UserAdminView.jsp) to the details pages (UserAdminCreate.jsp and UserAdminUpdate.jsp). To add the links, follow these steps:

1. Return to the Web Diagram file to take a final look at the completed Web diagram:



Note that each of the Web Page nodes have been realized, and the Command button links are in place.
2. Open UserAdminView.jsp.
3. Drag a **Link** component from the **Faces Components** palette drawer, and drop it next to the data table in the file.

4. Type `/UserAdminCreate.jsp` in the **URL** field, and `Create` in the **Label** field. Click **OK**.
5. Drag a **Command - Hyperlink** component from the **Faces Components** palette drawer, and drop it onto the output field labeled `{userid}` in the data table.
6. Open the Properties view if it is not already active, and click the **Add Rule** button. (You may need to scroll to the right.)
7. In the **Add Navigation Rule** dialog, select `UserAdminUpdate.jsp` from the **Page** list box.
8. Select the **The outcome named** radio button and type `update`.
9. Click **OK**.
10. Select the **Parameter** tab in the Properties view.
11. Click the **Add Parameter** button.
12. Type `userid` in the **Name** field.
13. Click the cell in the **Value** field, and then click the browse icon to open the **Select Page Data Object** dialog. Expand **userFacadeLocal > findByName(java.lang.String) > userFacadeLocalFindByNameResultBean**, and select **userid (java.lang.Integer)** in the **Select Page Data Object** dialog.



14. Click **OK**.

Next, you must add code for storing the parameter in the session scope, so that the update page can use

the user ID to supply the user information in the update page's input form.

1. Click the **Quick Edit** view tab. Click in the Quick Edit editing area, which will create an empty template for the action code associated with the `Update` button, and type the following code:

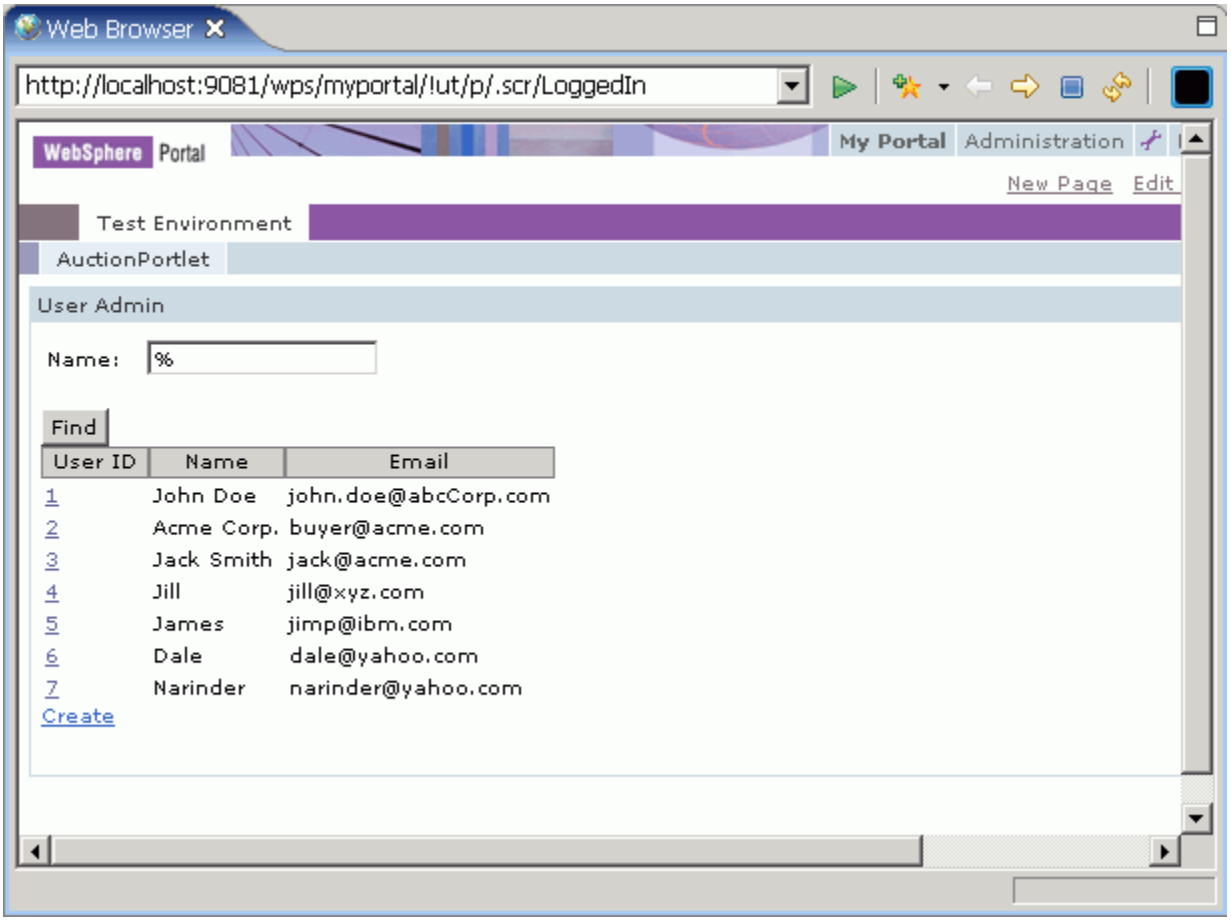
```
String userid = (String)getRequestParam().get("userid");
getSessionScope().put("userid", new Integer(userid));
return "update";
```

2. Save and close the file.

Run the UserAdmin portlet

To verify that pages in the UserAdmin portlet are working as intended up to this point, you should run the JSP file on the internal browser provided with Rational tools. To run the UserAdmin portlet, do the following:

1. Select the AuctionPortlet project in the Project Explorer, and select **Run > Run on server** from its pop-up menu.
2. Because you have already defined the WebSphere Portal v5.1 test environment, select it, and click **Finish** in the Server Selection wizard.
3. The file will eventually display in the browser. Here you can see the input fields, links, and layout that a user would see on a portal site.
4. Type `%` in the Name field and click the **Find** button to retrieve existing users from the database.



Notice that pressing the **Create** link or selecting a user in the data table opens the UserAdminCreate or UserAdminUpdate page, respectively. If you supply values in either of these pages and click the **Create** or

Update button, the additions and updates will be accepted. Updated data should be reflected immediately in the UserAdminView page. If your search name matches a user name that you created (for example, you can use a wildcard character % to retrieve all users), the new user should be displayed when you return from the create page.

Before we move to the next exercise, stop the test environment server. To stop the test environment server, simply select it in the Servers view, and click the **Stop the server** tool bar button  .

Now you are ready to begin Exercise 1.5: Adding portlets that search Auction site listings and provide listing details.

Exercise 1.5: Adding portlets that search Auction site listings and provide listing details

Before you begin, you must complete Exercise 1.4: Creating pages for creating and updating user information.

In this exercise, you will create additional portlets that provide cooperative behavior, using the Click-to-Action mechanism to send data from a source portlet to a target portlet. The source portlet (ListingSearch) uses a session bean to access Auction item data. After locating specific listings with the ListingSearch portlet, the target ListingDetail portlet displays detailed information for the items located by a search. In each case, you must first create the portlet.

Create the ListingSearch portlet

To create the ListingSearch portlet, follow these steps:

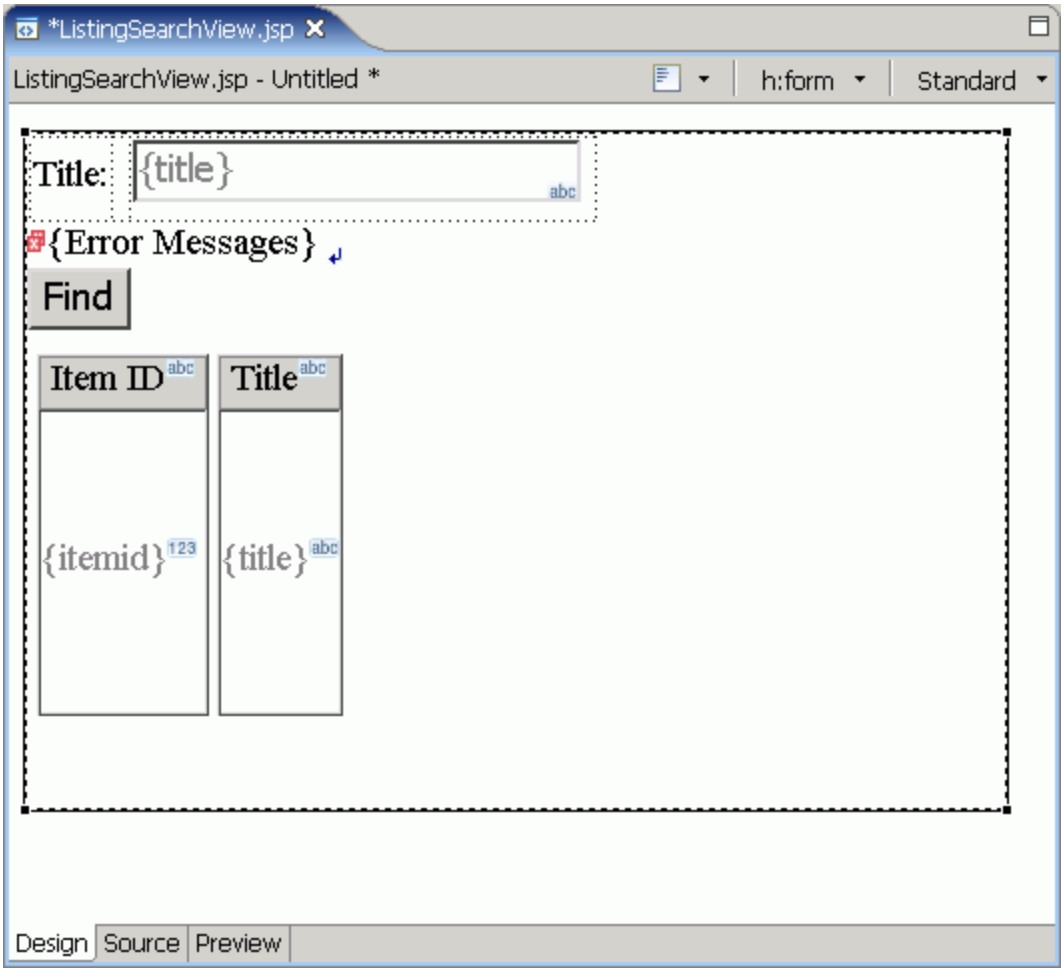
1. In the Project Explorer, expand the Dynamic Web Project folder, and locate the AuctionPortlet project folder. Right click on the project folder, and Select **New** > **Portlet**.
2. In the initial page of the New Portlet wizard, change the **Default name prefix** to `ListingSearch`.
3. Select the **Faces portlet** radio button.
4. Click **Next**.
5. Select `AuctionPortlet application` from **Application name** drop-down list.
6. Type `Listing Search` in the **Portlet title** field.
7. Click **Finish**.

ListingSearchView.jsp opens in the editing pane.

Add a data table to the ListingSearch portlet page

With the ListingSearchView.jsp file open in Page Designer, follow these steps to add session bean data as the data source for the ListingSearch page:

1. Delete the default `Place content here.` text.
2. Drag the **EJB Session bean** object from the **Data** drawer in the palette to the file.
3. When the Session Bean wizard opens, click the **New EJB Reference** button.
4. Expand the AuctionPortletEAR and AuctionEJB50 folders, and select `ListingFacade` to create the enterprise bean reference. Click **Finish**.
5. Click **Next** in the Session Bean wizard.
6. Select the `findByTitle(String title)` interface, which will be used for the input field on the portlet page, and click **Next**.
7. Note that the title field is selected. Click **Options**.
8. Type `Find` in the **Label** field. Click **OK**.
9. Click **Next**, which should bring you to the Results Form page of the wizard. In this page, you will define the data table that will retrieve and display the data from the database.
10. Click **None** to deselect all of the columns, so that you can individually select, organize, and configure the appropriate columns for the data table to be used in the portlet page. Then, select the check boxes for the following columns:
    - itemid
    - title
11. Using the up down arrow buttons, move the selected data columns into the order shown in the step above.
12. Select and change the **Label** value for the itemid column to `Item ID`.
13. Click **Finish** to generate the default user interface for the UserAdminView.jsp page. The user interface will look similar to the following:



14. Save the file.

Add Java page code to the UserAdmin page

In this portion of the exercise, you will add Java page code to accomplish the following:

- Store the `title` parameter in the session scope, so that it can be reused for any future refresh of the portlet content.
- Initialize the parameter to be displayed in the `Title` input field with the one stored in session scope.
- Initialize the result data using `title` parameter stored in the session scope.

You can add the EJB reference logic and the code to bind the invocation and results to the user interface, using the following steps:

1. Select **Edit Page Code** from the pop-up menu in Page Designer.
2. Type the following **bold** code into doListingFacadeLocalFindByTitleAction():

```
public String doListingFacadeLocalFindByTitleAction() {
    String title = getListingFacadeLocalFindByTitleParamBean().getTitle();
    getSessionScope().put("title", title);
    try {
        listingFacadeLocalFindByTitleResultBean =
        getListingFacadeLocal().findByTitle(title);
    } catch (Exception e) {
        logException(e);
    }
    return null;
```

```
}
```

3. Type the following **bold** code into getListingFacadeLocalFindByTitleParamBean():

```
public ListingFacadeLocalFindByTitleParamBean
        getListingFacadeLocalFindByTitleParamBean() {
    if (listingFacadeLocalFindByTitleParamBean == null) {
        listingFacadeLocalFindByTitleParamBean =
        new ListingFacadeLocalFindByTitleParamBean();
        String title = (String)getSessionScope().get("title");
        listingFacadeLocalFindByTitleParamBean.setTitle(title);
    }
    return listingFacadeLocalFindByTitleParamBean;
}
```

4. Type the following **bold** code into getListingFacadeLocalFindByTitleResultBean():

```
public ItemData[] getListingFacadeLocalFindByTitleResultBean() {
    if (listingFacadeLocalFindByTitleResultBean == null) {
        String title = (String)getSessionScope().get("title");
        if (title != null) {
            try {
                listingFacadeLocalFindByTitleResultBean =
                    getListingFacadeLocal().findByTitle(title);
            } catch (Exception e) {
                logException(e);
            }
        }
    }
    return listingFacadeLocalFindByTitleResultBean;
}
```

5. Save ListingSearchView.java.

Create the ListingDetail portlet

To create the ListingDetail portlet, follow these steps:

1. In the Project Explorer, expand the Dynamic Web Project folder, and locate the AuctionPortlet project folder. Right click on the project folder, and Select **New** > **Portlet**.
2. In the initial page of the New Portlet wizard, change the **Default name prefix** to `ListingDetail`.
3. Select the **Faces portlet** radio button.
4. Click **Next**.
5. Select `AuctionPortlet application` from **Application name** drop-down list.
6. Type `Listing Detail` in the **Portlet title** field.
7. Click **Finish**.

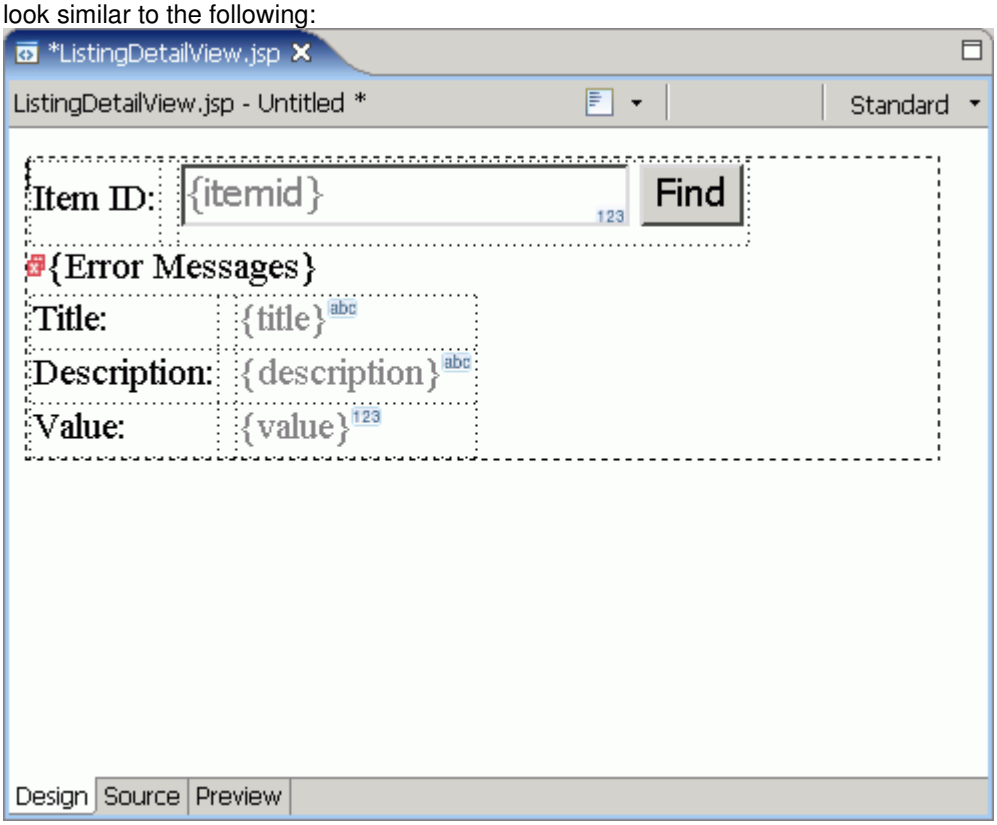
The ListingDetailView.jsp file will open in the editing pane.

Add a data form to the ListingDetail portlet page

With the ListingDetailView.jsp file open in Page Designer, follow these steps to add session bean data as the data source for the ListingDetail page:

1. Delete the default `Place content here.` text.
2. Drag the **EJB Session bean** object from the **Data** drawer in the palette to the file.

3. When the Session Bean wizard opens, select `ejb/ListingFacade` and click **Next**.
4. Select the `findById(Integer itemid)` interface, which will be used for the input field on the portlet page.
5. Click **Next**.
6. Note that the itemid field is selected. Select and change the **Label** value for the itemid field to `Item ID:`.
7. Click **Options**.
8. Type `Find` in the **Label** field. Click **OK**.
9. Click **Next**, which should bring you to the Results Form page of the wizard. In this page, you will define the data form that will retrieve and display the data from the database.
10. Click **None** to deselect all of the fields, so that you can individually select, organize, and configure the appropriate fields for the data table to be used in the portlet page. Then, select the check boxes for the following fields:
    - title
    - description
    - value
11. Using the up down arrow buttons, move the selected data fields into the order shown in the step above.
12. Click **Finish** to generate the user interface for the ListingDetailView.jsp page. The user interface will look similar to the following:



13. Save the file.

Add Java page code to the UserAdmin page

In this portion of the exercise, you will add Java page code to accomplish the following:

- Store the `itemid` parameter in the session scope, so that it can be reused for any future refresh of the portlet content.
- Initialize the parameter to be displayed in the `Item ID` input field with the one stored in session scope.
- Initialize the result data using `itemid` parameter stored in the session scope.

You can add the EJB reference logic and the code to bind the invocation and results to the user interface, using the following steps:

1. Select **Edit Page Code** from the pop-up menu in Page Designer.
2. Type the following **bold** code into doListingFacadeLocalFindByIdAction():

```
public String doListingFacadeLocalFindByIdAction() {
    Integer itemid = getListingFacadeLocalFindByIdParamBean().getItemid();
    getSessionScope().put("itemid", itemid);
    try {
        listingFacadeLocalFindByIdResultBean =
        getListingFacadeLocal().findById(itemid);
    } catch (Exception e) {
        logException(e);
    }
    return null;
}
```

3. Type the following **bold** code into getListingFacadeLocalFindByIdParamBean():

```
public ListingFacadeLocalFindByIdParamBean
        getListingFacadeLocalFindByIdParamBean() {
    if (listingFacadeLocalFindByIdParamBean == null) {
        listingFacadeLocalFindByIdParamBean =
        new ListingFacadeLocalFindByIdParamBean();
        Integer itemid = (Integer)getSessionScope().get("itemid");
        listingFacadeLocalFindByIdParamBean.setItemid(itemid);
    }
    return listingFacadeLocalFindByIdParamBean;
}
```

4. Type the following **bold** code into getListingFacadeLocalFindByIdResultBean():

```
public ItemData getListingFacadeLocalFindByIdResultBean() {
    if (listingFacadeLocalFindByIdResultBean == null) {
        Integer itemid = (Integer)getSessionScope().get("itemid");
        if (itemid != null) {
            try {
                listingFacadeLocalFindByIdResultBean =
                     getListingFacadeLocal().findById(itemid);
            } catch (Exception e) {
                logException(e);
            }
        }
    }
    return listingFacadeLocalFindByIdResultBean;
}
```

5. Save ListingDetailView.java.

Add cooperative linking to the ListingSearch portlet

The term *cooperative* portlets refers to the capability of portlets on a page to interact with each other by sharing information. One or more cooperative portlets on a portal page can automatically react to changes from a source portlet triggered by an action or event in the source portlet. Portlets that are targets of the event can react so that users are not required to make repetitive changes or actions in other portlets on the page. The mechanism used to implement cooperative behavior is called a *Click-to-Action event.*

You launch a Click-to-Action event from an icon on the source portlet. The icon presents a pop-up menu containing the list of targets for the action. After the user selects a specific target, the property broker delivers the data to the target in the form of the corresponding portlet action. Using the Click-to-Action

delivery method, users can transfer data with a simple click from a source portlet to one or more target portlets, causing the target react to the action and display a new view with the results.

Use the following steps to set up cooperative behavior between the ListingSearch and ListingDetail portlets:

1. Open ListingSearchView.jsp in Page Designer. (In the Project Explorer, expand the AuctionPortlet and WebContent folders, and double-click the file.)
2. Drag a **Click-to-Action Output Property** object from the **Portlet** palette drawer and drop it onto the output field in the data table labeled `{itemid}`.

   Note: Ensure that you drop the **Click-to-Action Output Property** object *onto* the output field, not before or after it. The output field should be highlighted in a rectangular box, and look similar to this:
   
   (You may need to move the dialog box aside to see the output field selection.)

3. In the **Insert Click-to-Action Output Property** dialog, supply the following values:
   - Type `itemid` in **Data type** field.
   - Select `ListingSearch portlet` from the **Source portlet** drop-down list, and click **OK**.

   Save the file.
4. To verify that the correct Click-to-Action code has been added, open the Source view, and ensure that the following code is included in the file:

```
<h:outputText id="text3"
        value="#{varlistingFacadeLocalFindByTitleResultBean.itemid}"
        styleClass="outputText">
        <f:convertNumber />
</h:outputText>
        <c2a:encodeProperty type="itemid"
                namespace="http://auctionportlet" name="itemid"
                value="#{varlistingFacadeLocalFindByTitleResultBean.itemid}
        </c2a:encodeProperty>
```

   Note the highlighted `value` attribute. This attribute will be missing if the Output Property has not been added to the page correctly.

These steps identify ListingSearch as the source portlet. Next, you must enable ListingDetail as the target portlet:

1. Expand AuctionPortlet > Portlet Deployment Descriptor in the Project Explorer. Select **Cooperative > Enable Target** from the ListingDetail portlet's pop-up menu.
2. In the **Enable Cooperative Portlet** dialog, type `itemid` in **Data type** field.
3. Click the **Browse** button next to the **Action** field, and select `/ListingDetailView.jsp > form1 > button1`. Click **OK**.
4. In the **Enable Cooperative Portlet** dialog, type `Show Detail` in **Label** field. Click **OK**.

To verify that the Click-to-Action source are correctly identified, return to the Portlet Deployment Descriptor folder in the Project Explorer. The icons that signify a source portlet (ListingSearch) and a target portlet (ListingDetails) should look like this:

Test cooperative behavior in the Listing portlets

To verify that the ListingSearch and ListingDetail portlets are working as intended, you should run them in the test environment.

To run the portlets, do the following:

1. Select the AuctionPortlet project in the Project Explorer, and select **Run > Run on server** from its pop-up menu.
2. Because you have already defined the WebSphere Portal v5.1 test environment, select it, and click **Finish** in the Server Selection wizard.
3. The portlets will display in the browser. Here you can see the input fields, buttons, and layout that a user would see on a portal site.
4. In the ListingSearch portlet, type in the wildcard search character % to list all items, and click the **Find** button. The table below the **Submit** button will display the auction items that match the search string.
5. Click the Click-to-Action icon in the **Item ID** column to send the listing ID to the ListingDetail portlet. The ListingDetail portlet displays detailed information about the item that was located by the search.

Before we move to the next exercise, stop the test environment server. To stop the test environment server, simply select it in the Servers view, and click the **Stop the server** tool bar button  .

Now you are ready to begin Exercise 2.1: Creating a new portal to display the portlet application.

Module 2: Developing portal applications

The new portal application created in this module uses the portlet applications that you created in Module 1: Developing portlet applications as content. In this module, you will:

- Design a basic portal site, and experiment with changing the look and feel of the site using Portal Designer
- Run the portal application in the WebSphere Portal test environment, so that you can see how users will use the application, and how data flows through the application.

Exercises

The exercises within this module must be completed in order:

- Exercise 2.1: Creating a new portal to display the portlet application
- Exercise 2.2: Customizing the portal site
- Exercise 2.3: Run the portal application in the WebSphere Portal test environment

Time required

This module will take approximately **30 minutes** to complete.

Exercise 2.1: Creating a new portal to display the portlet application

Before you begin, you must complete Exercise 1.5: Add portlets that search Auction site listings and provide listing details.

In this exercise, you will create a new portal application with pages that will contain the portlets that have been created up to this point.

Create the portal project

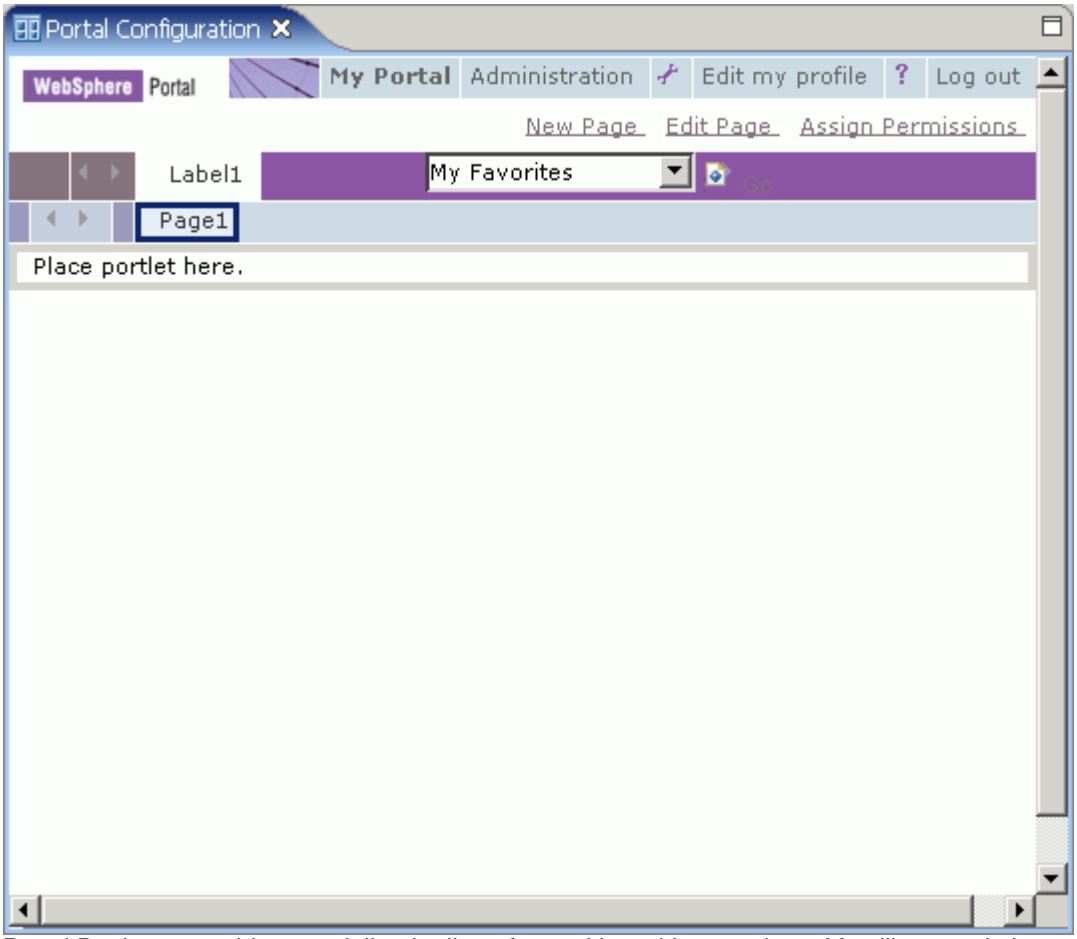
To create the Auction portal project, follow these steps:

1. From the workspace menu, select **File > New > Portal Project**.
2. In the initial page of the New Portal Project wizard, type `AuctionPortal` into the **Name** field.
3. Note that the **WebSphere Portal version** defaults to `5.1`. Click **Next**.
4. Select a default theme for the site. For purposes of this tutorial, select the **WebSphere** theme.
5. Select a default skin associated with the the WebSphere theme. Note that default skins for a portal site are always associated with a default theme. For purposes of this tutorial, select the **Outline** skin.

   There are two types of default skin. One is for the overall portal site, and the other is for a theme. In this wizard, you are specifying only the default skin for the selected default theme. (Only one skin can be a default skin for a portal site.) Typically, each theme has a default skin defined, and the default skin for a portal site is rarely used.

6. Click **Finish**. If you are asked to switch to the Web Perspective, click **OK**.

Portal Designer displays the project's Portal Configuration file, which shows the default navigation structure with one label and a child page:



Portal Designer provides specialized editors for working with a number of familiar portal elements, including labels, pages, a navigation, and themes and skins, which, along with cascading style sheets, allow you to control the look and feel of a portal site.

Create a label and pages to contain the portlet applications

Use Portal Designer to create the portal layout elements for the portlets that you have created:

1. Select **Label1** and then select the Properties view tab. Click the **Title** tab in the Properties view, and change `Label1` to `Auction`. Press Enter.
2. Select **Page1** and then select the Properties view tab. Click the **Title** tab in the Properties view, and change `Page1` to `Registration`. Press Enter.
3. Drag a **Page** component from the **Portal** drawer in the palette onto the `Auction` label. This will add a new page next to the Registration page. Change the new page's title to `Listing`, as you did in the previous step.

Add rows and columns to layout portal pages

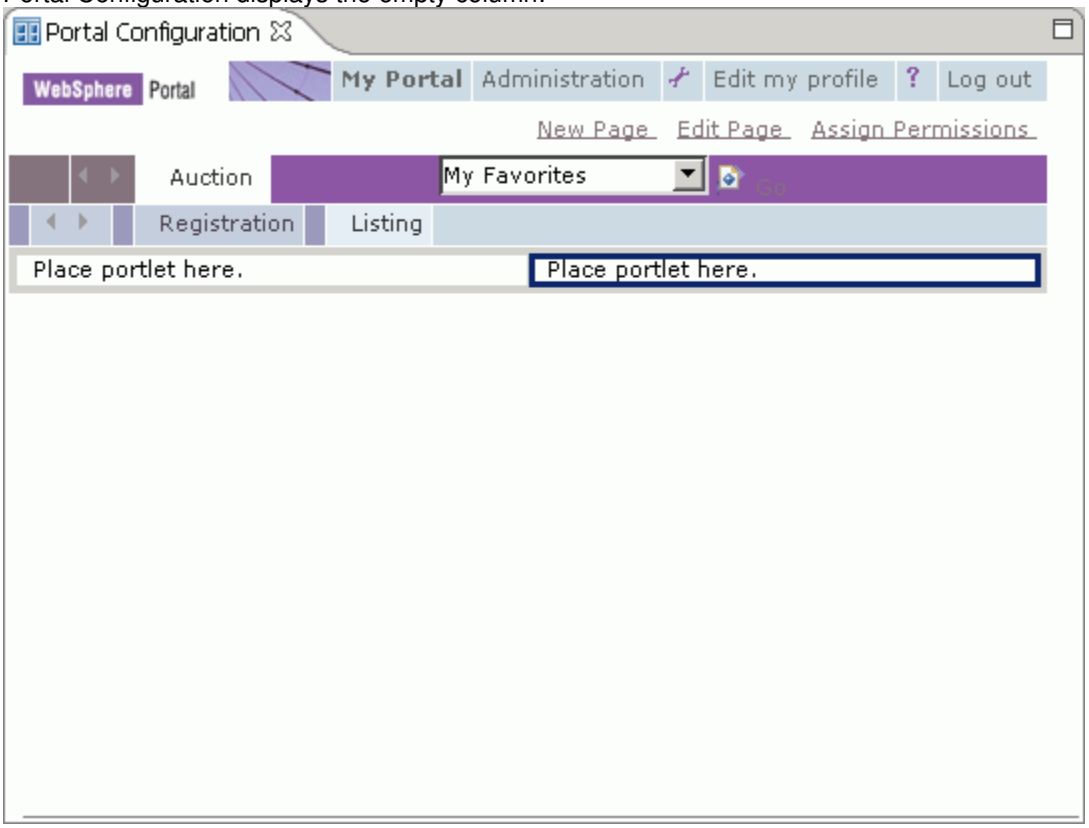
The portal pages provide a framework for adding content to your portal site. You have already created this content in the form of the portlets defined in previous exercises. In order to add multiple portlets to individual pages, you can define a spatial layout with some combination of rows and columns that allow you to place portlets on the page in a logical fashion.

By default, each page that you create includes a single column and a single row. It is a simple process to add columns and rows as you would for any "table" in a Web application. (Note that the *table* metaphor is not part of the standard portal terminology--it is only being used for instructional purposes.) Because you will be adding two portlets to the listing page, you can add another "cell" to the Listing page layout, in this case

a column. You can do this in one of two ways:

- Select the Listing page. Drag a **Column** item from the palette to the Listing page. A new column will be added to the right of the default column, in the existing row.
- With the Listing page selected, click the **Outline** tab in the group of views in the lower left-hand area of your workspace. Expand the Layout node, and then the Listing (page) node. Then, you can either right-click the Row node and select **Insert Column > As Child**, or right-click the existing Column node and select **Insert Column > To Right**.

In either case, the additional column is created, so that the ListingDetail portlet can be inserted later. The Portal Configuration displays the empty column:

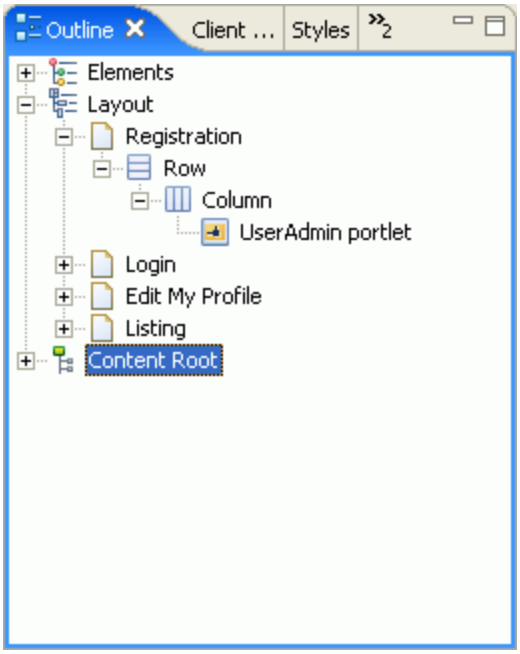


Add portlets to the portal pages

Now that there are new pages with layout elements under the Auction label, you can add the portlets that were created in earlier exercises in the following way:

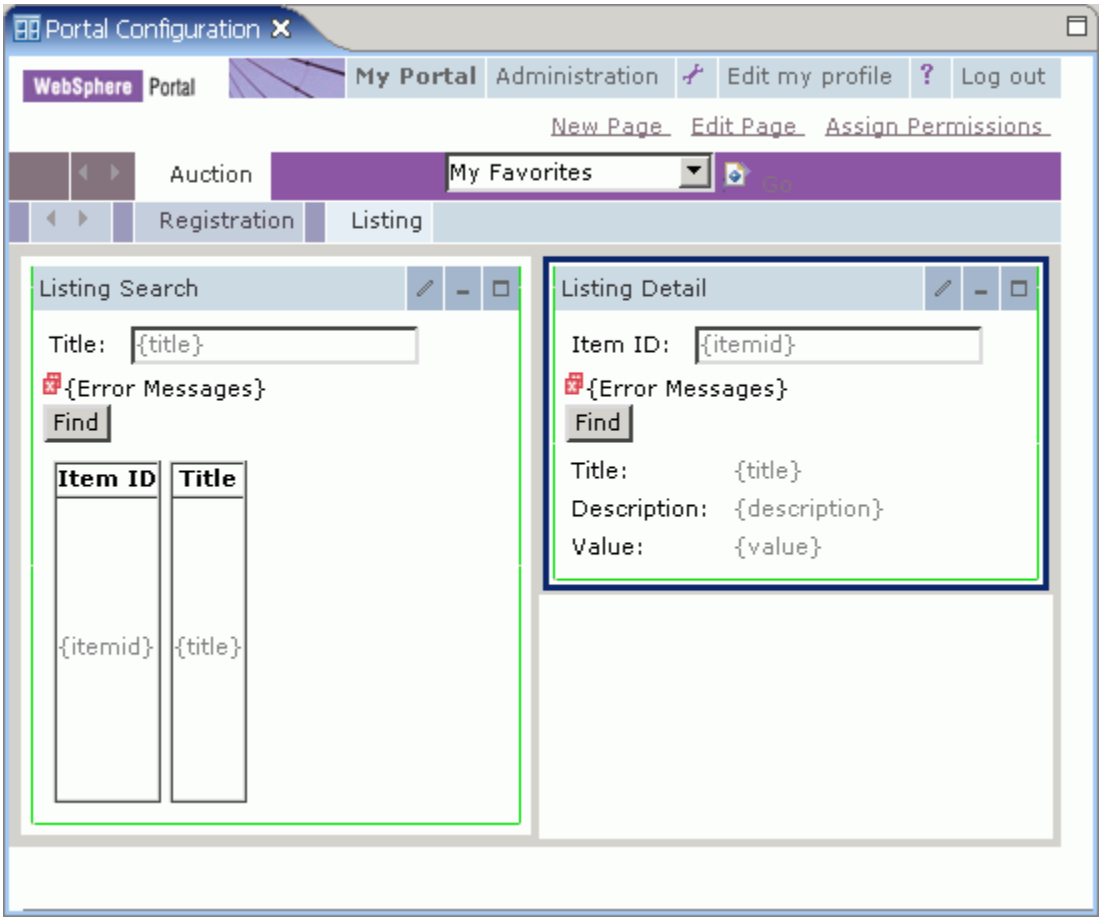
1. Select the Registration page.
2. Drag a **Portlet** item from the palette to the column in the Registration page. Select the User Admin portlet from the options in the **Insert Portlet** dialog, and click **OK**.

   Note that you can also perform the same action using the Outline view. In the Outline view, expand the Layout node, and then the Registration node. Under the Column, you will see the UserAdmin portlet icon. Select the column and right-click to open the pop-up menu. Observe that there are cascading menus to add additional portlets to the page.



3. Select the Listing page, by clicking `Listing` in the navigation area. You are now ready to add the cooperative Listing portlets to the Listing page:
    1. Using one of the methods described in the previous step, insert the ListingSearch portlet into the first column of the Listing page.
    2. Select the column to the right of the ListingSearch portlet, and insert the ListingDetail portlet.
4. Save the file.

    The updated Listing page should look similar to this:



The portal site is nearly completed. In the exercise that follows, you will learn how to customize the look and feel of the portal site using some of the specialized editors that work in conjunction with Portal Designer.

Now you are ready to begin Exercise 2.2: Customizing the portal site.

Exercise 2.2: Customizing the portal site

Before you begin, you must complete Exercise 2.1: Create a new portal to display the portlet application.

Create a new theme

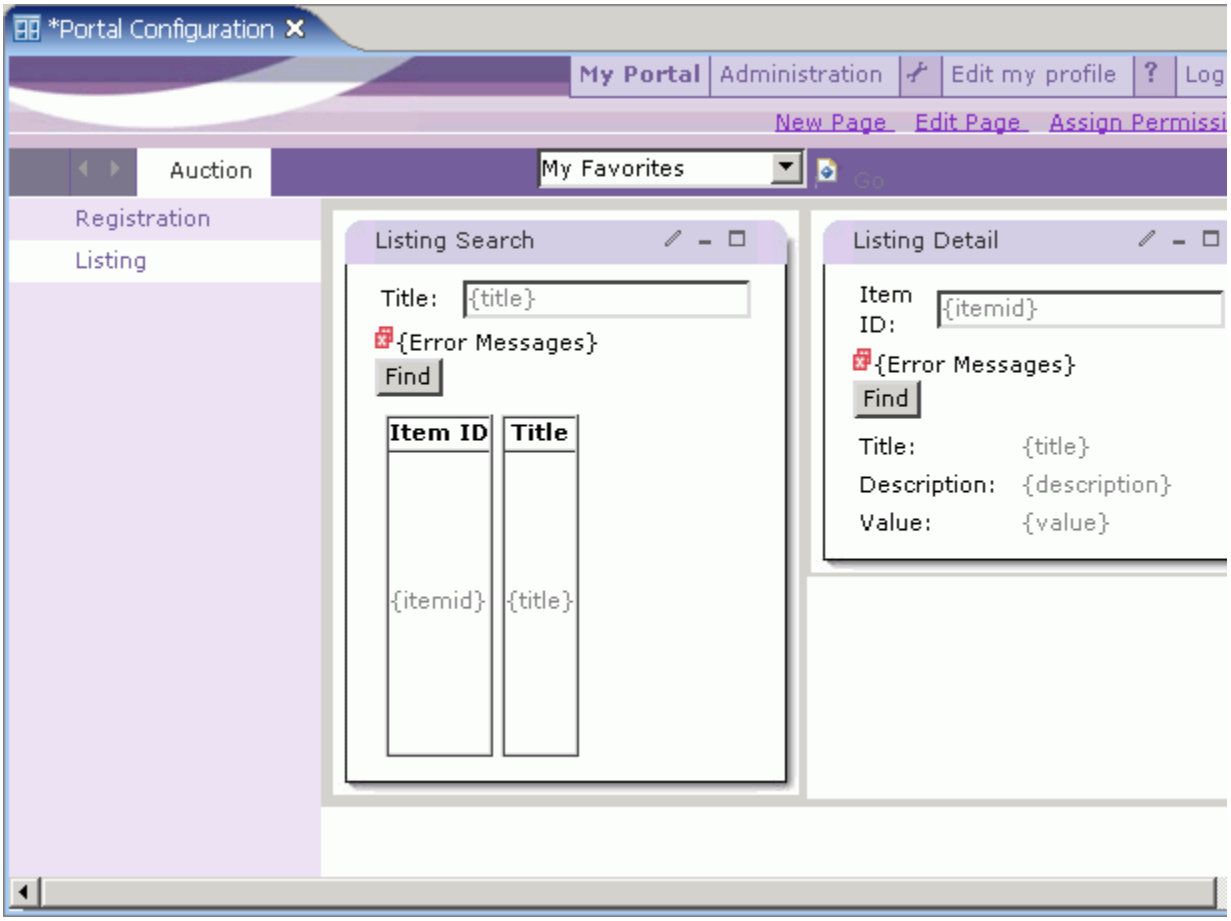
In the process of creating a portal application, you can choose to create a brand new theme in Portal Designer. Themes provide the overall look and feel of your portal application, and incorporate many elements of portal design, including images, navigation, tool bars, and page level visual effects. Follow these steps to create a new theme for the portal site that we have developed in this tutorial:

1. Select **File > New > Theme** from the menu bar.
2. Type `Auction` in **Title** field.
3. Scroll and select the `Corporate` theme to be the source theme. It is much easier to base a new theme on an existing theme, so that you do not have to create all of the necessary theme elements from scratch.
4. Click **Next**.
5. Select the Shadow skin from the list of available skins, and click the **Set as Default Skin** to make Shadow the default skin for the new Auction theme.

   A skin is the border around each portlet within a portal page. Unlike themes, which apply to the overall look and feel of the portal, skins are limited to the look and feel of each portlet that you insert into your portal application. By default, only a limited selection of skins are available for each theme.

6. Click **Finish**.
7. `Elements > Themes` in the Outline view, and select Auction.
8. In the Properties view, select the **Default** check box to apply the new theme to the portal application.

   The change will be immediately applied to the Portal Configuration:



9. Save the Portal Configuration.

In this portion of the exercise, you will update styles, themes, and skins in the Auction portal application, using CSS Designer and Page Designer.

Change the banner image of the current theme

To replace the banner image in the default theme for the Auction portal, follow these steps:

1. First, you will import a new banner image into the project:
    1. From the menu bar, select **File > Import**. The Import dialog appears.
    2. Under **Select an import source**, click **File System**.
    3. Click **Next**.
    4. Because different Rational products use different installation target locations, you must leave the product's user interface to locate the plug-in that contains the new banner image. Use a file search tool to locate the `com.ibm.etools.portal.examples.application_6.0.0.1` plug-in folder under the product installation path in your local file system.
    5. Return to the Import wizard, and click **Browse** next to the **From directory** field. Navigate to the following directory:

        `x:\com.ibm.etools.portal.examples.application_6.0.0.1\samples`

        where *x:* is the path that contains the `com.ibm.etools.portal.examples.application_6.0.0.1` plug-in on your computer.

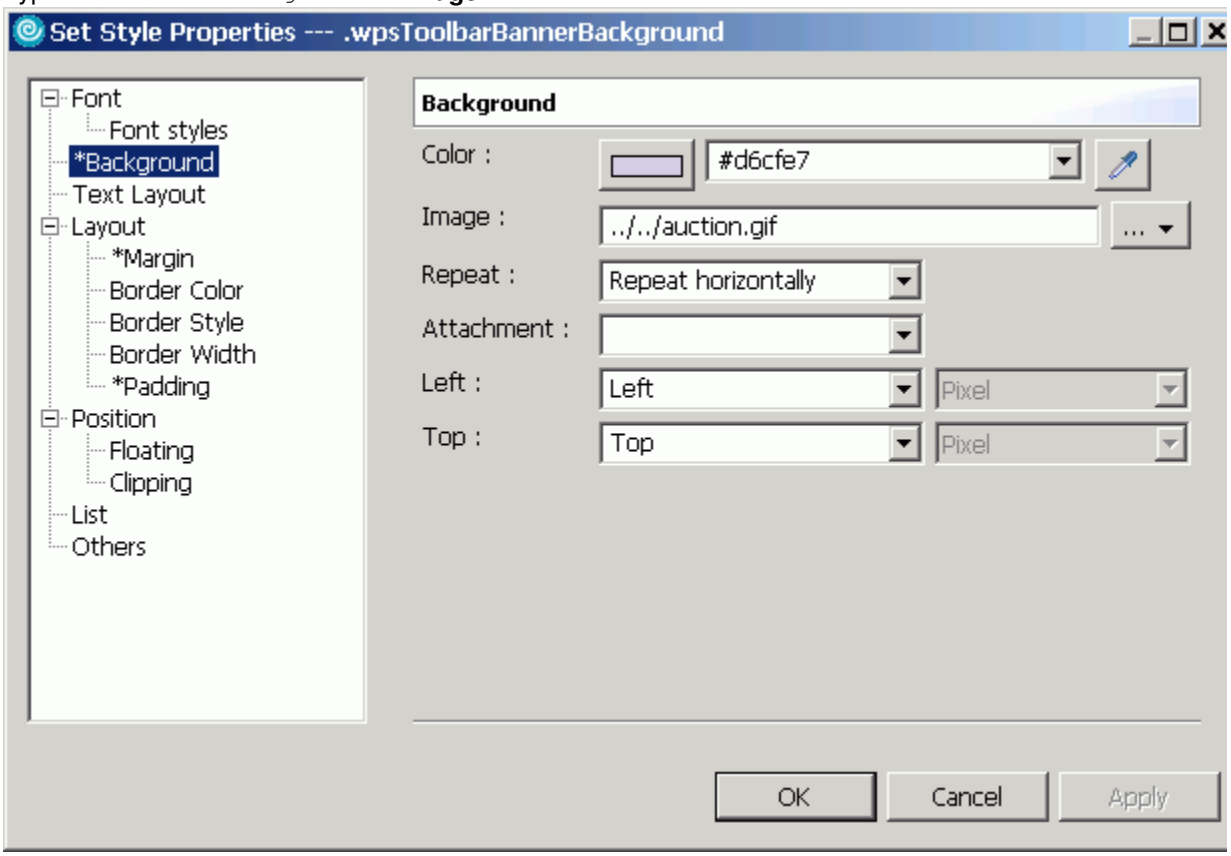
6. Select **auction.gif** as the target of the import, and click **OK**.
7. Click **Browse** next to the **Into folder** field and specify `AuctionPortal/PortalContent/themes/html/Auction`.
8. Click **Finish**.

   The wizard imports the file into your workspace.

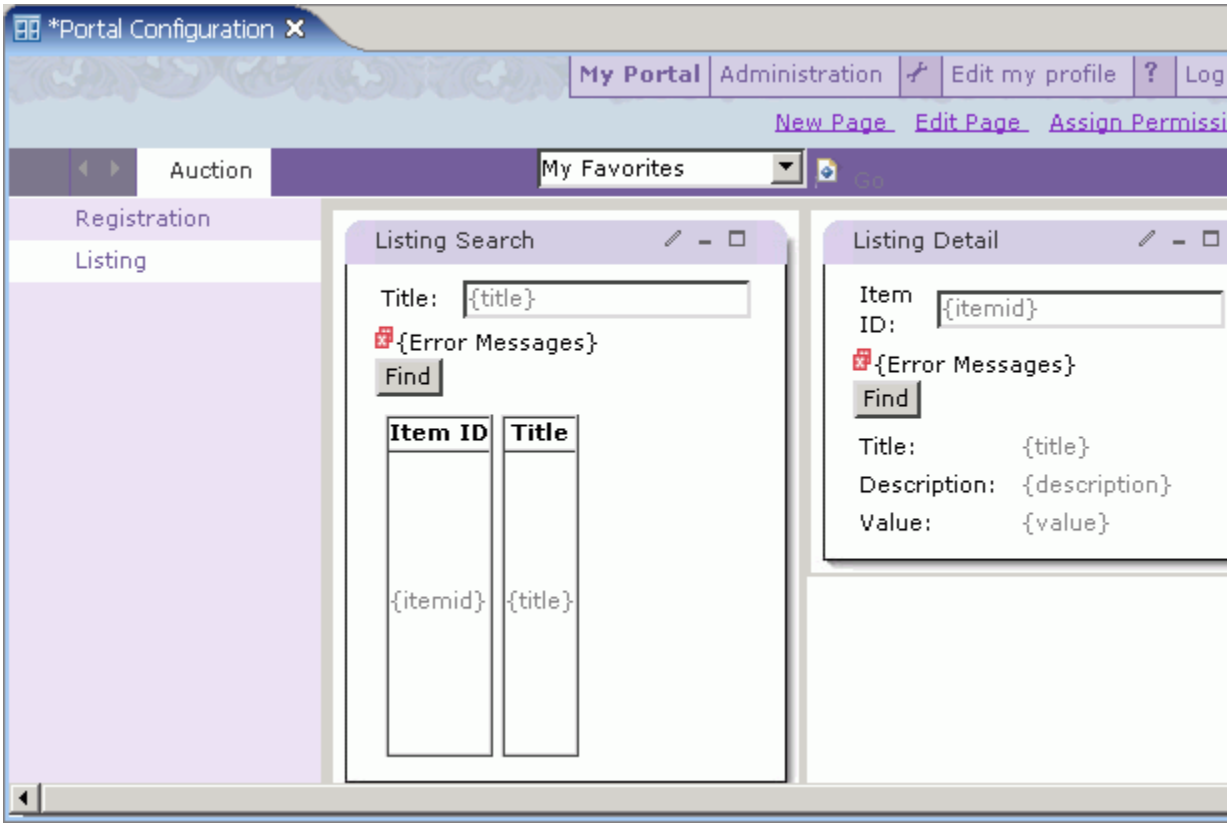
2. With the Portal Configuration file for the AuctionPortal project open, select **Edit Style** from the Portal Designer pop-up menu. This will open the Styles.css file in CSS Designer. Styles.css is the default style sheet for the default theme in the application.

   The CSS Designer provides two views of the styles defined for CSS files: the Preview (on the left side), which provides visualized examples of the style rules as they appear in a browser rendering of a Web resource, and the Source view (on the right side), which displays a text version of the CSS file. You can edit styles using either of these views.

3. Scroll and select the **Banner Background** icon in the Preview.
4. Select **Edit Style Rule [.wpsToolbarBannerBackground]** from the pop-up menu.
5. Click the **Background** property from the left side of the **Set Style Properties** dialog.
6. Type `../../auction.gif` in the **Image** field.



7. Click **OK**.
8. Save the CSS file and close CSS Designer. Note that the new banner image is applied to the open page in Portal Designer.



9. Save and close the Portal Configuration file.

You can make a large variety changes to a theme, such as the layout of a toolbar in the header area, using Page Designer. You can edit the layout (and style) of a theme and its associated skins. Changes are stored in the theme's default.jsp file, associated skins' control.jsp files, and other related JSP files. Also, any changes that you make in the editor will apply to all uses of this theme within your portal application.

Now you are ready to begin Exercise 2.3: Run the portal application in the WebSphere Portal Test Environment.

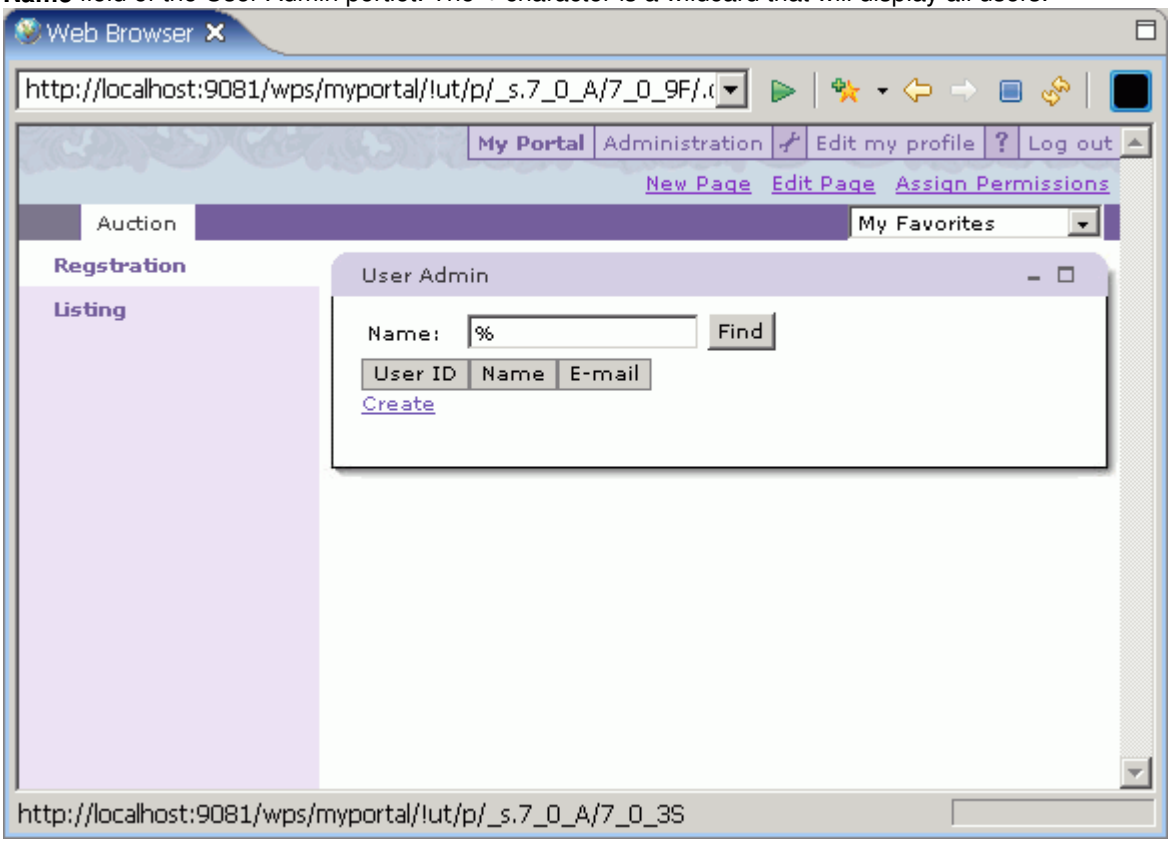
Exercise 2.3: Run the portal application in the WebSphere Portal Test Environment

Before you begin, you must complete Exercise 2.2: Customizing the portal site.

Test the portal application in WebSphere Portal test environment

To verify that the Auction is working as intended, you should run the Auction portal on the WebSphere Portal test environment. To run the portal project and test the application with live values, do the following:

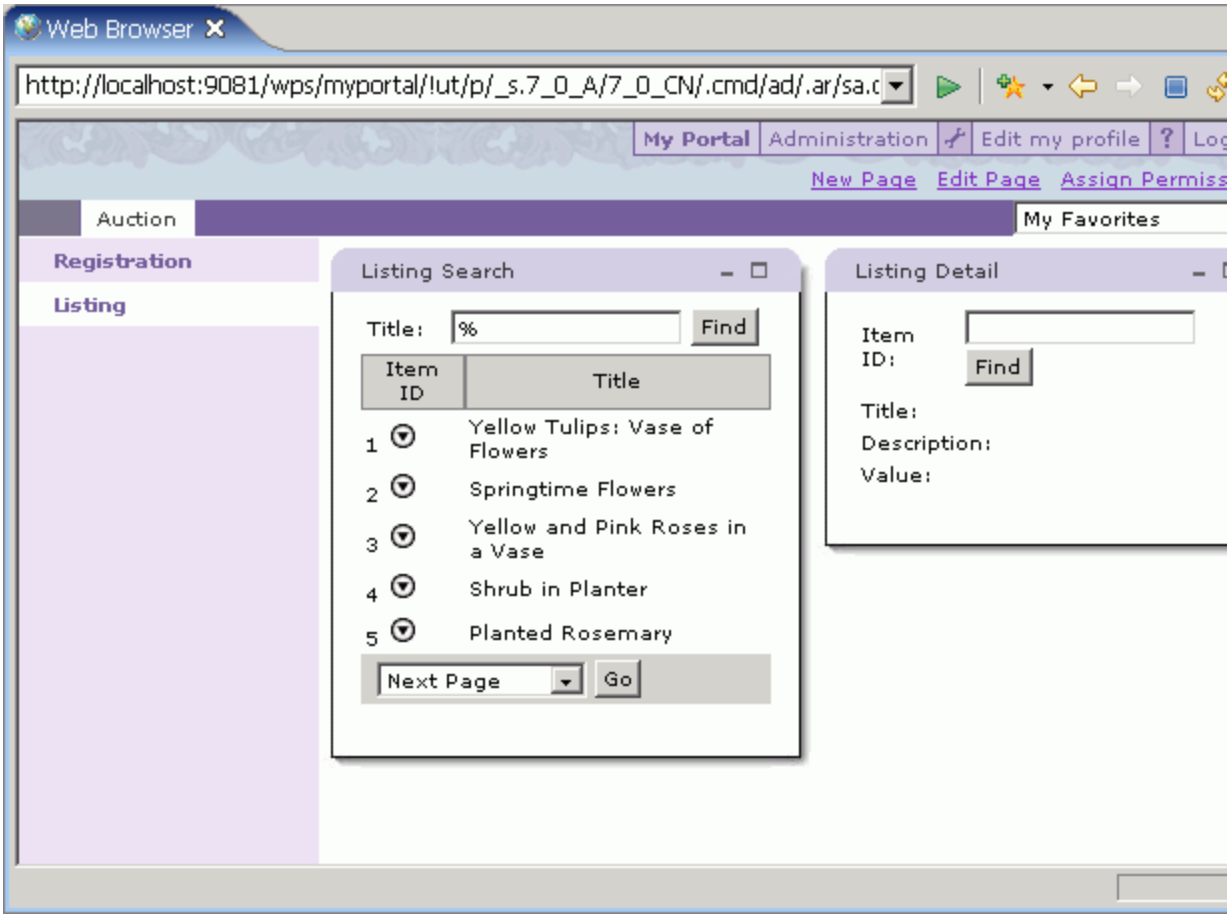
1. Select the AuctionPortal project in Project Explorer.
2. From the AuctionPortal pop-up menu, Select **Run > Run on Server**.
3. From the list of available server types, select **Select WebSphere Portal v5.1 Test Environment**.
4. Click **Finish**.
5. After a few moments, when the portal application is displayed in a browser window, type % in the **Name** field of the User Admin portlet. The % character is a wildcard that will display all users.



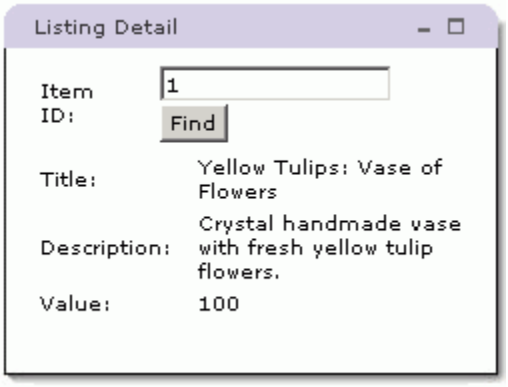
6. Click the **Find** button.

   The page will return a list users in the data table.

7. Click one of the **ID** values in the table to update user information.
8. Click **Create** to create a new user.
9. Next, click on the Listing page on the portlet.
10. Type % in the **Title** field of the Listing Search portlet to display all available listings.



11. Click the down arrow icon next to one of the available items, and select **Show Detail**. Observe the detail information provided in the Listing Detail portlet:



You have now completed all of the exercises in this tutorial. To review the content that we have covered, see the Create a portal application tutorial summary.

Create a portal application tutorial summary

Congratulations! This tutorial has taught you the basics of using Rational Software Development Platform's portal tools to create and test a portal application that includes portlets that you have created using JavaServer Faces to design the user interface.

Completed learning objectives

If you have completed all of the modules, you should now be able to:

- Explain how portal and portlet projects and applications are organized
- Recognize the types of resources and code that are generated by the portal tools wizards
- Create the application flow for JavaServer Faces portlets using the Web Diagram editor
- Connect portlets to JavaBean data and Session EJBs
- Create user interfaces for pages that display, create, and edit user records
- Create a basic search results page, and a page that provides a detailed description of any items retrieved by the search
- Send data from one page to another using cooperative portlets
- Design a basic portal site, and change the look and feel of the site using Portal Designer
- Test the portal and portlet applications that you create as you develop them.

More information

If you want to learn more about the topics covered in this tutorial, consider the following sources:

- Rational tools help topics, samples in the **Samples Gallery**, and additional tutorials provided in the **Tutorials Gallery**
- WebSphere Portal Product Documentation
- WebSphere Developer Domain - Portal Zone
- IBM Redbooks WebSphere Domain
- Using Cooperative Portlets in WebSphere Portal V5