IBM RATIONAL APPLICATION DEVELOPER 6.0 – LAB EXERCISE

Developing a Basic Portlet

What this exercise is about

The purpose of this lab is to demonstrate the use of the Portlet Adapter methods setVariable, getVariable, and removeVariable along with implementing the Portlet ActionListener interface to save information to the PortletData Object from the doEdit method. The Lab implements a Name List that has a name to company pairing. The name to company pairing is read from the config-params in the portlet.xml file and stored to the PortletData Object. We allow the edit of the Name List thru the doEdit () method and implement the ActionListener interface to store changed information to the PortletData Object.

Portlet data is saved, retrieved, or deleted to persistence using the PortletData object. Portlets can store values in the PortletData object only when the portlet is in edit mode. If the portlet is on a group page, then information saved in PortletData is available to all users of the portlet. Therefore, the PortletData should only be used for information pertaining to the portlet, not the user.

Lab Requirements

List of system and software required for the student to complete the lab.

- IBM Rational Application Developer v6.0 with the Portal Tools Additional Feature selected
- WebSphere Portal Server v5.0.2.2 Test Environment
- The lab source files LabFiles60.zip must be extracted to the root directory C:\

What you should be able to do

At the end of this lab you should be able to:

- Develop portlets using the Portal Tools within Rational Application Developer v6.0

Introduction

This lab will provide users a hands-on experience to create a name list portlet application. It will be divided into four parts. The first part will show the user how to create a portlet project by defining a portlet project with the project wizard. This will create a basic, deployable portlet in which to base the portlet application. The next part, part two, will add the appropriate methods and code to the name list portlet. In part three the user will add the capability to support the edit mode, which will allow the user to add names to the portlet. The final part will have the user deploy and test the portlet application on the portal test environment.

Exercise Instructions

Some instructions in this lab may be Windows operating-system specific. If you plan on running the lab on an operating-system other than Windows, you will need to execute the appropriate commands, and use appropriate files (.sh vs. .bat) for your operating system. The directory locations are specified in the lab instructions using symbolic references, as follows:

Reference Variable	Windows Location	AIX/UNIX Location
<WAS_HOME>	C:\WebSphere\AppServer	/usr/WebSphere/AppServer /opt/WebSphere60/AppServer
<IRAD_HOME>	C:\Program Files\IBM\RSDP\6.0	
<LAB_FILES>	C:\Labfiles60	/tmp/Labfiles60
<TEMP>	C:\temp	/tmp
<LAB_NAME>	IRAD_Portal_Basic	

Windows users please note: When directory locations are passed as parameters to a Java program such as EJBdeploy or wsadmin, it is necessary to replace the backslashes with forward slashes to follow the Java convention. For example, C:\LabFiles60\ would be replaced by C:/LabFiles60/.

Solution instructions are included at the end in case you are unable to complete the lab.

Part 1: Creating the Portlet Application Project

____ 1. Start Rational Application Developer.

__ a. Select **Start > Programs > IBM Rational > IBM Rational Application Developer V6.0 > Rational Application Developer**.

__ b. A dialog box will be displayed allowing you to select the location where you would like the workspace directory to be stored. Enter **<LAB_FILES>\<LAB_NAME>\workspace** for the location and select **OK**. Application Developer will start with an empty workspace. An empty workspace will leave your existing workspace untouched and help avoid name conflicts between what you may already have in your workspace and what you will be creating in this lab.

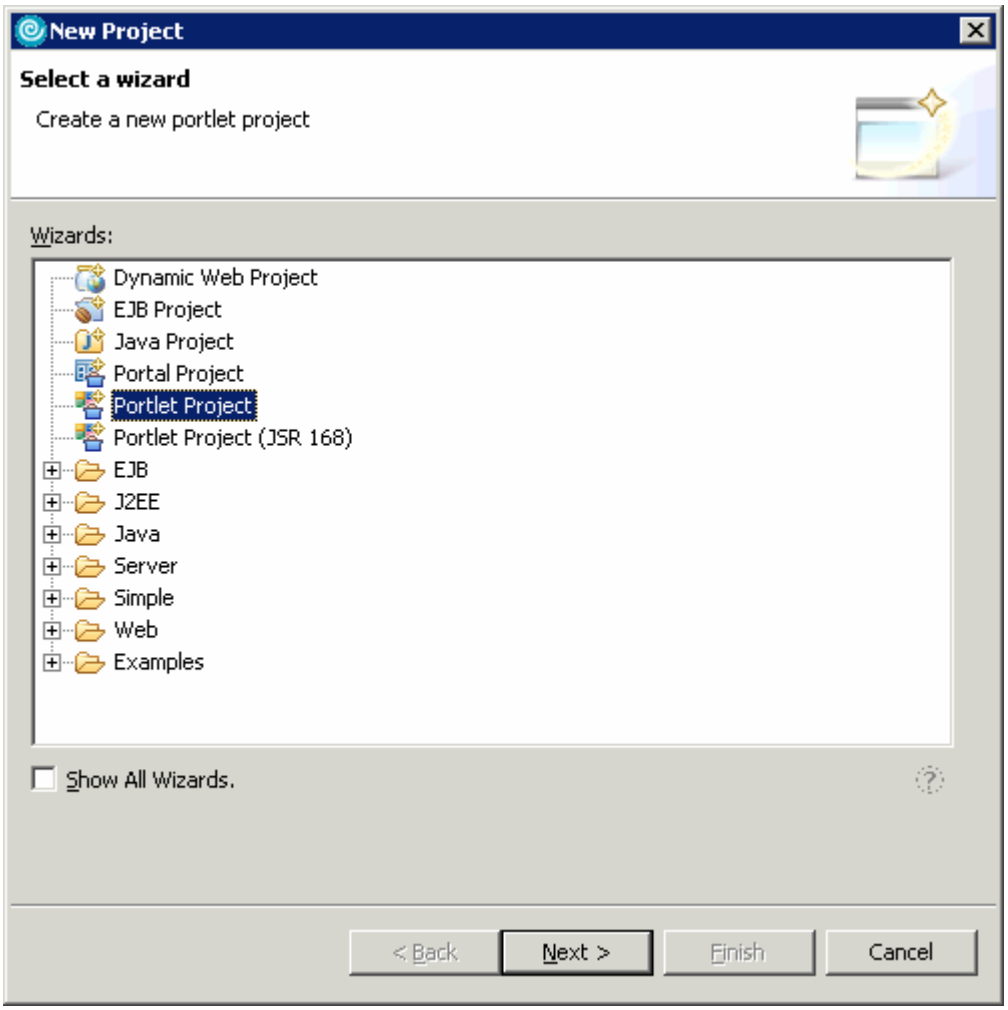
NOTE: If the Auto Launch Configuration Change Alert window appears click the **Yes** button to change the auto launch eclipse instance to use when opening IBM Rational Software Development Platform in the future.

____ 2. When Rational Application Developer v6.0 opens, close the welcome page.

____ 3. Create a new portlet project named NameList.

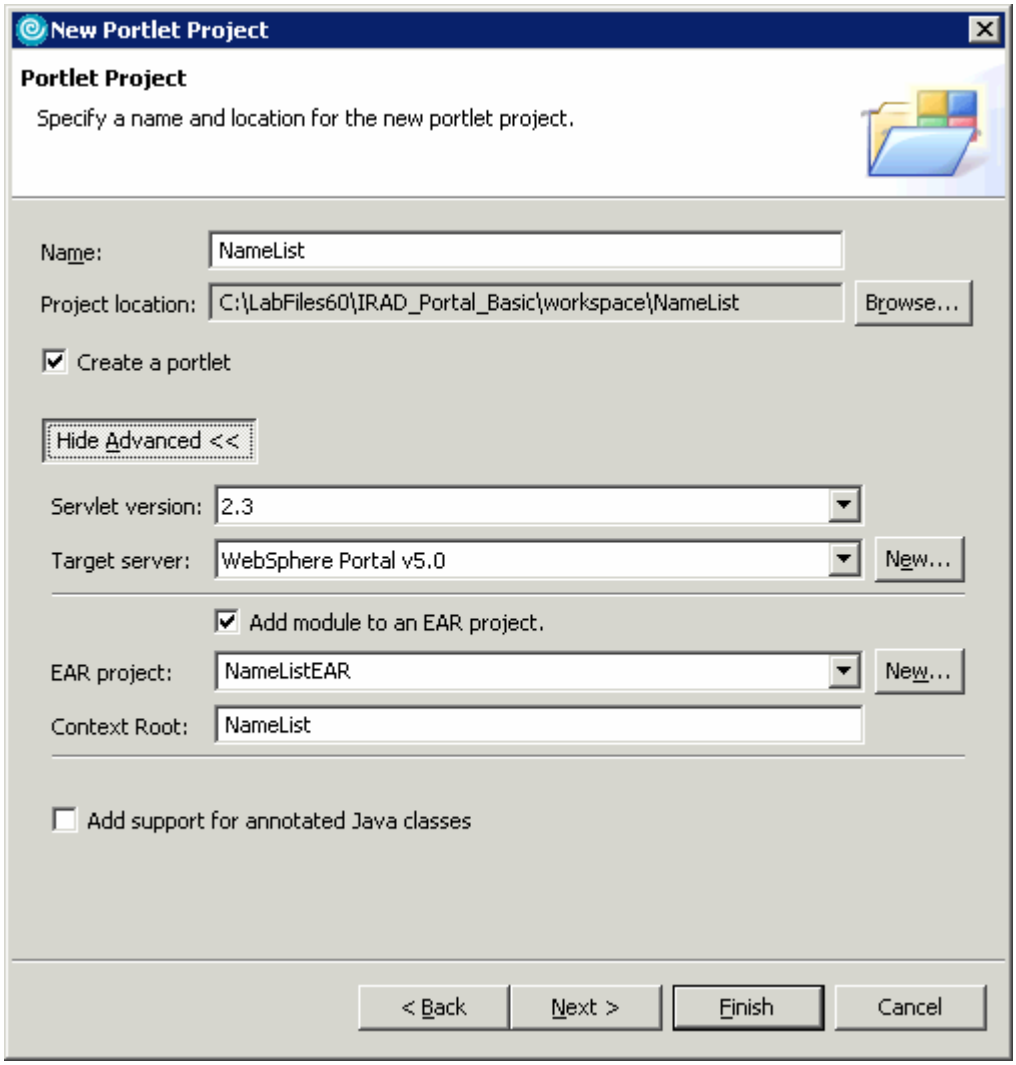
__ a. Create a new Portlet project. Select **File > New > Project**.

__ b. In the New Project window, select **Portlet Project** and click **Next >**.

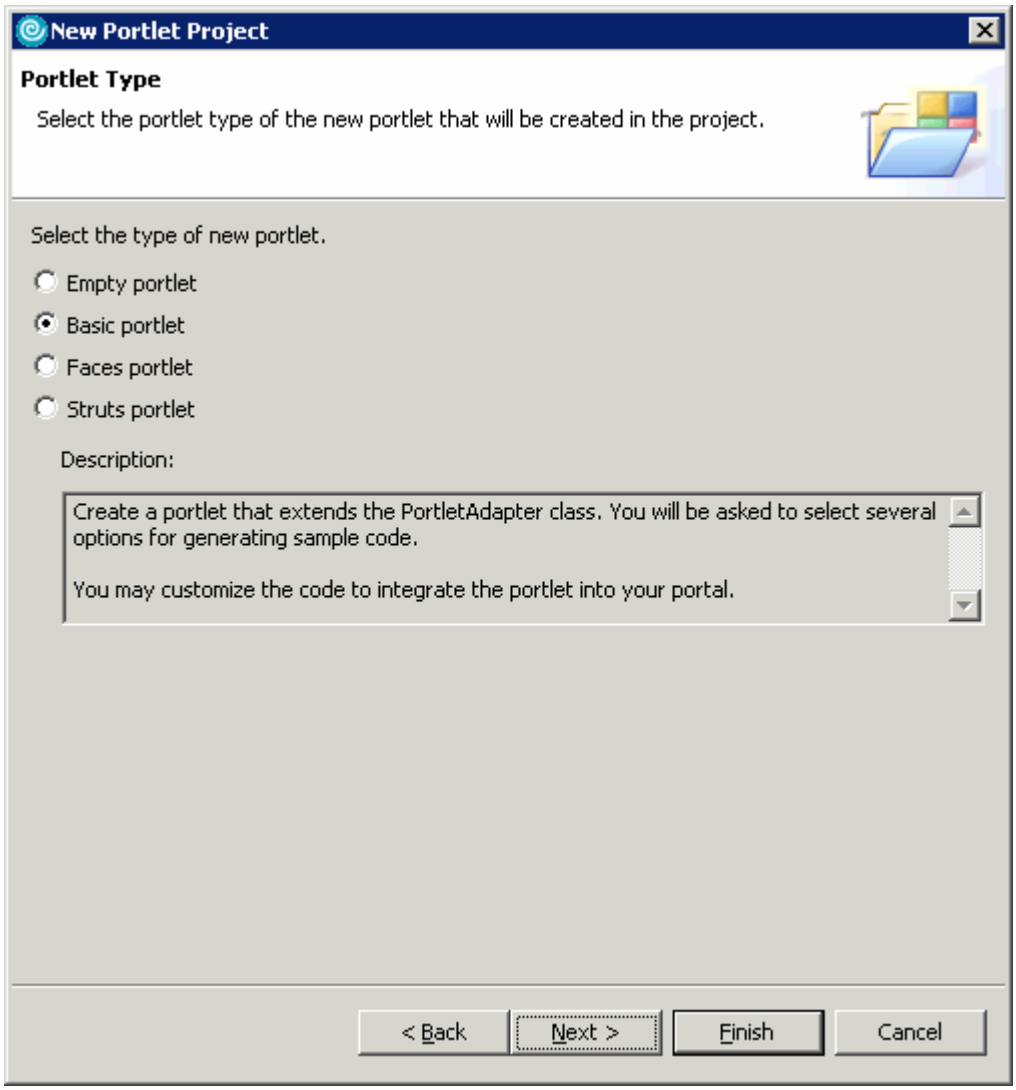


__ c. In the Confirm Enablement window, read the message and click **OK**. Creating a Portlet project requires that advanced Web Development capabilities be enabled within Application Developer.

__ d. In the New Portlet Project window, enter **NameList** in the Name field. Click **Show Advanced >>** to show the advanced properties of the portlet project wizard. Verify that the Target server field is set to **WebSphere Portal v5.0**. Click **Next >**.



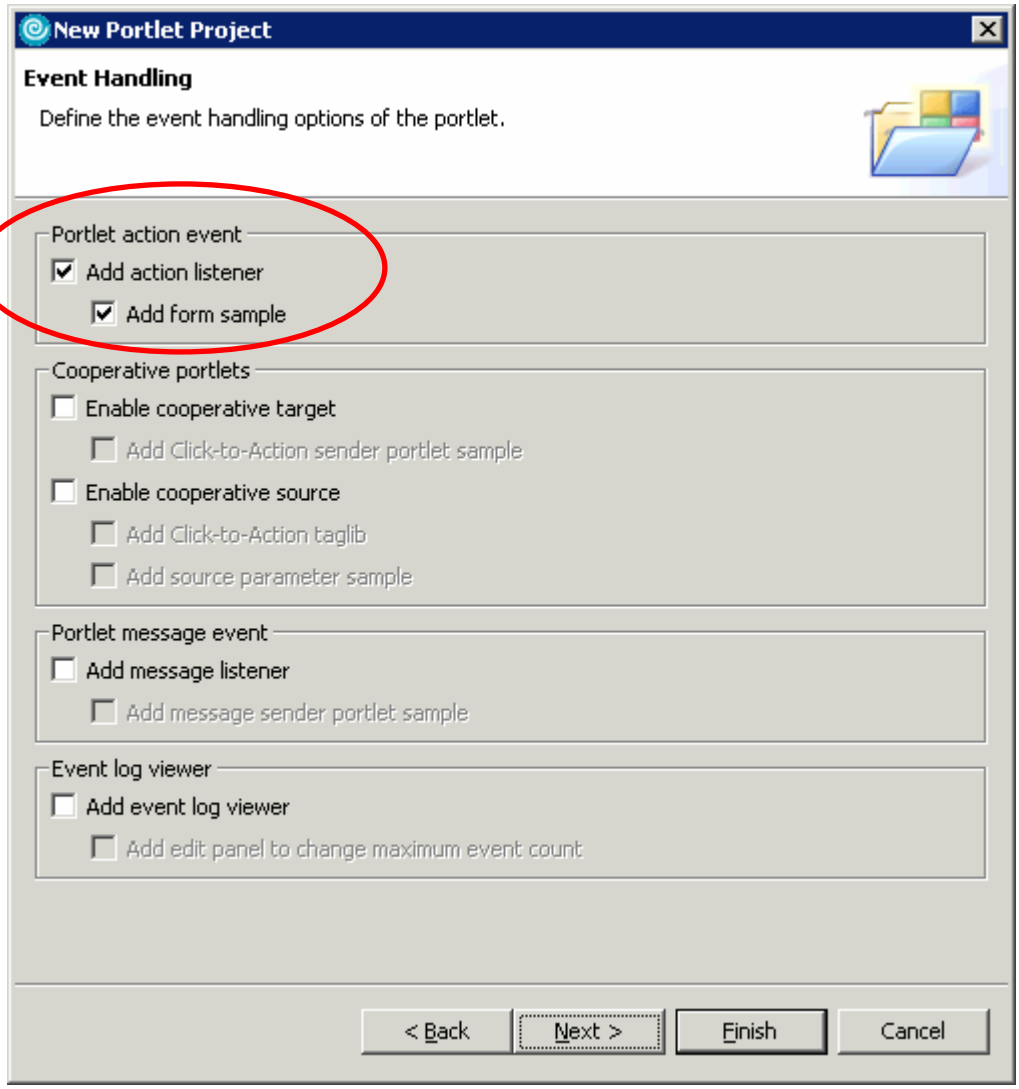
__ e. In the Portlet Type panel, select **Basic portlet** and click **Next >**.



__ f. In the Features panel, observe the defaults and click **Next >**.

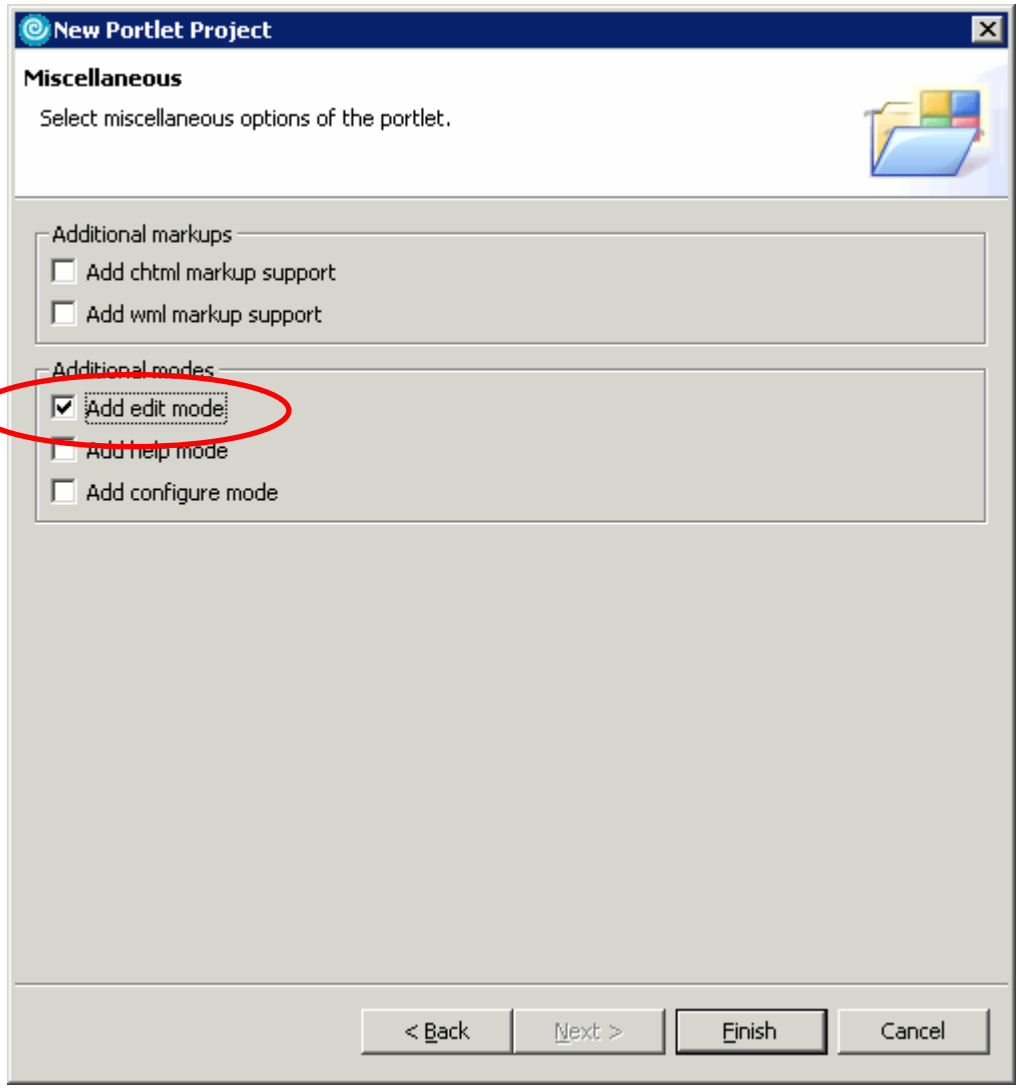
__ g. In the Portlet Settings panel, observe the defaults and click **Next >**.

__ h. In the Event Handling panel, ensure that **Add action listener** and **Add form sample** are selected (circled in the screen capture below) and click **Next >**.



__ i. In the Single Sign-On panel, observe the defaults and click **Next >**.

__ j. In the Miscellaneous panel, select the box next to **Add edit mode** (circled in the screen capture below) and click **Finish**.



__ k. In the Confirm Perspective Switch window, read the message and click **Yes**. Portlet projects are associated with the Web Perspective within Application Developer.

__ l. The portlet will be created and the editor for the portlet's HTML view mode will be opened within Application Developer.

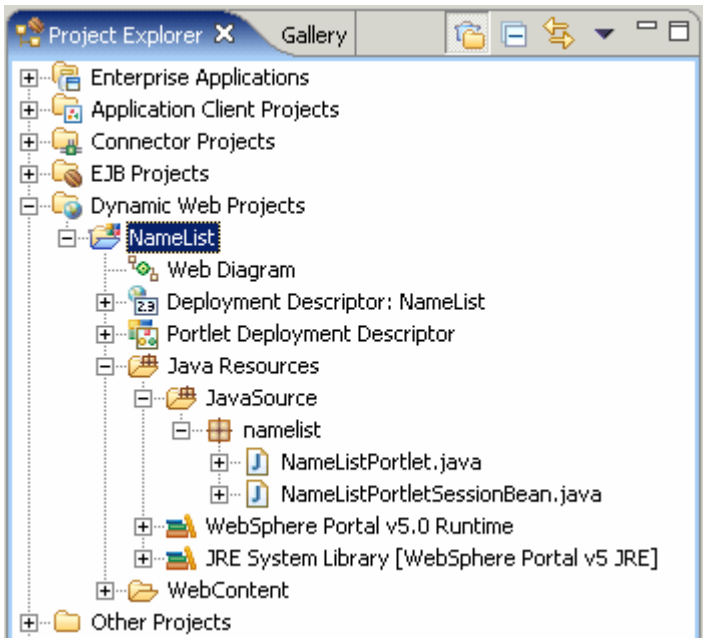
__ m. **Close** the NameListPortletView.jsp editor.

Part 2: Implement the Name List Portlet

In this part, you will add the appropriate methods and code to NameListPortlet.java and NameListPortletViewBean.java. To save time, text snippets have been provided that contain all of the code that you will be adding to the portlet.

____ 1. Open the NameListPortlet.java file.

__ a. In the Project Explorer view, expand **Dynamic Web Projects > NameList > Java Resources > JavaSource > namelist**.



__ b. Double click on **NameListPortlet.java** to open it in the Java editor.

____ 2. Edit the NameListPortlet.java file.

__ a. Add additional private fields to the **NameListPortlet** class just below the class' declaration. The appropriate lines of code are highlighted in the following screen capture. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet1.txt**.

```
/**
 *
 * A sample portlet based on PortletAdapter
 *
 */
public class NameListPortlet extends PortletAdapter implements ActionListener {
    private static final String NAME_PREFIX = "name";
    private static final String COMPANY_PREFIX = "company";
    private static final String COUNT = "count";

    public static final String VIEW_JSP          = "/namelist/jsp/NameListPortletView.";
```

__ b. Modify the **actionPerformed** method to look for an ActionEvent named “Submit” in the HTTP request object, Replace the contents of the actionPerformed method with the lines of code highlighted in the screen capture below. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet2.txt**.

NOTE: The following further explains why the code below has been added:

An ActionEvent occurs when an HTTP request is received that is associated with a PortletAction. To receive events, the portlet class must implement the ActionListener interface and an object with the type PortletAction. Typically, a PortletAction is linked with HTTP references or buttons in HTML forms. In the actionPerformed method, actions can be defined to follow certain paths based on users’ actions. In this lab, when a user clicks the submit button in the NameListPortletEdit.jsp, an ActionEvent called “submit” is added to the HTTP request object.

```
public void actionPerformed(ActionEvent event) throws PortletException {
    if (getPortletLog().isDebugEnabled())
        getPortletLog().debug("ActionListener - actionPerformed called");
    // ActionEvent handler
    String actionString = event.getActionString();
    // Add action string handler here
    if (actionString != null) {
        if (actionString.equals("submit")) {
            PortletRequest request = event.getRequest();
            PortletData portData = request.getData();
            try {
                String count = (String) this.getVariable(COUNT);
                for (int i = 1; i <= Integer.parseInt(count); i++) {
                    portData.setAttribute(
                        NAME_PREFIX + i,
                        request.getParameter(NAME_PREFIX + i));
                    portData.setAttribute(
                        COMPANY_PREFIX + i,
                        request.getParameter(COMPANY_PREFIX + i));
                    portData.store();
                }
            } catch (AccessDeniedException ade) {
                System.out.println("AccessDeniedException " + ade);
            } catch (IOException IO) {
                System.out.println(
                    "Could not write data due to IO error " + IO);
            }
        }
    }
}
```

__ c. Add an **initConcrete method**. The portlet container calls the initConcrete method when the concrete portlet is put into service. It is called before the portlet is accessed for the first time. Add the lines highlighted in the screen capture below to the end of the NameListPortlet class just before the last curly ('}') bracket. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet3.txt**.

```
    private static String getJspExtension(PortletRequest request) {
        String markupName = request.getClient().getMarkupName();
        return "jsp";
    }

    public void initConcrete(PortletSettings settings)
            throws UnavailableException {
        try {
            java.util.Enumeration e = settings.getAttributeNames();
            while (e.hasMoreElements()) {
                Object a = e.nextElement();
                this.setVariable((String) a, settings.getAttribute((String) a));
            }
        } catch (Exception e) {
            System.out.println("Exception in initConcrete method = " + e);
        }
    }
}
```

__ d. Add a **destroyConcrete** method. To indicate that the portlet was taken out of service, the portlet container calls the destroyConcrete method. This only occurs when an administrator deletes a portlet during runtime on the portal server. The goal of this method is to remove the attributes set in the initConcrete method. Add the lines highlighted in the screen capture below to the end of the NameListPortlet class just before the last curly ('}') bracket. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet4.txt**.

```
        } catch (Exception e) {
            System.out.println("Exception in initConcrete method = " + e);
        }
    }

    public void destroyConcrete(PortletSettings settings) {
        try {
            java.util.Enumeration e = settings.getAttributeNames();
            while (e.hasMoreElements()) {
                Object a = e.nextElement();
                this.removeVariable((String) a);
            }
        } catch (Exception e) {
            System.out.println(
                "Exception in the destroyConcrete method = " + e);
        }
    }
}
```

__ e. Add the **createBean** method. The createBean method is being used to save data to the Portlet bean (NameListPortletViewBean). The bean is then used to pass information to the NameListPortletView.jsp and NameListPortletEdit.jsp files. Add the lines highlighted in the screen capture below to the end of the NameListPortlet class just before the last curly ('}') bracket. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet5.txt**.

NOTE: There will be errors in the Problems view that say the NameListPortletViewBean cannot be found or instantiated. These errors will be fixed later on in the lab.

```
        } catch (Exception e) {
            System.out.println(
                "Exception in the destroyConcrete method = " + e);
        }
    }

    public NameListPortletViewBean createBean(
            PortletRequest request,
            PortletResponse response)
            throws AccessDeniedException, java.lang.NumberFormatException {

            NameListPortletViewBean bean = new NameListPortletViewBean();
            String count = ((String) this.getVariable(COUNT));
            bean.setCount(Integer.parseInt(count));
            PortletData portletData = request.getData();

            if (count != null) {
                if ((String) portletData.getAttribute(NAME_PREFIX + 1) == null) {
                    for (int i = 1; i <= Integer.parseInt(count); i++) {
                        bean.getNamePrefix().addElement(
                            (String) this.getVariable(NAME_PREFIX + i));
                        bean.getCompanyPrefix().addElement(
                            (String) this.getVariable(COMPANY_PREFIX + i));
                        bean.getAttributeName().addElement(
                            (String) NAME_PREFIX + i);
                        bean.getAttributeName().addElement(
                            (String) COMPANY_PREFIX + i);
                    }
                } else {
                    for (int i = 1; i <= Integer.parseInt(count); i++) {
                        bean.getNamePrefix().addElement(
                            (String) portletData.getAttribute(NAME_PREFIX + i));
                        bean.getCompanyPrefix().addElement(
                            (String) portletData.getAttribute(COMPANY_PREFIX + i));
                        bean.getAttributeName().addElement(
                            (String) NAME_PREFIX + i);
                        bean.getAttributeName().addElement(
                            (String) COMPANY_PREFIX + i);
                    }
                }
            }
            return bean;
    }
}
```

__ f. Replace the contents of the doView method with the lines of code highlighted in the screen capture below. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet6.txt**.

```
    /**
     * @see org.apache.jetspeed.portlet.PortletAdapter#doView(PortletRequest,
     *      PortletResponse)
     */
    public void doView(PortletRequest request, PortletResponse response)
        throws PortletException, IOException {
        // Check if portlet session exists
        NameListPortletSessionBean sessionBean = getSessionBean(request);

        if (sessionBean == null) {
            response.getWriter().println("<b>NO PORTLET SESSION YET</b>");
            return;
        }

        // Make a view mode bean
        NameListPortletViewBean viewBean = createBean(request, response);

        // Save bean in request object
        request.setAttribute("NameListPortletViewBean", viewBean);

        // Invoke the JSP to render
        getPortletConfig().getContext().include(
            VIEW_JSP + getJspExtension(request),
            request,
            response);
    }
```

__ g. Replace the contents of the doEdit method with the lines of code highlighted in the screen capture below. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet7.txt**.

NOTE: The following further explains why the code below has been added:

When a portlet enters edit mode, the portlet data can be changed. To provide the edit functionality, a portlet JSP file will be provided to enter new information. The doEdit () method in the NameList portlet will receive control prior to the display of the NameListPortletEdit.jsp for the portlet. A return URI is created and passed to the JSP for edit mode using the PortletRequest object. The submit action is included in the return URI to result in an invocation of the ActionListener for the portlet. The portal passes control to the ActionListener upon processing the submit action. The ActionListener can preserve the user entered "edit" information in the persistent storage (portletData).

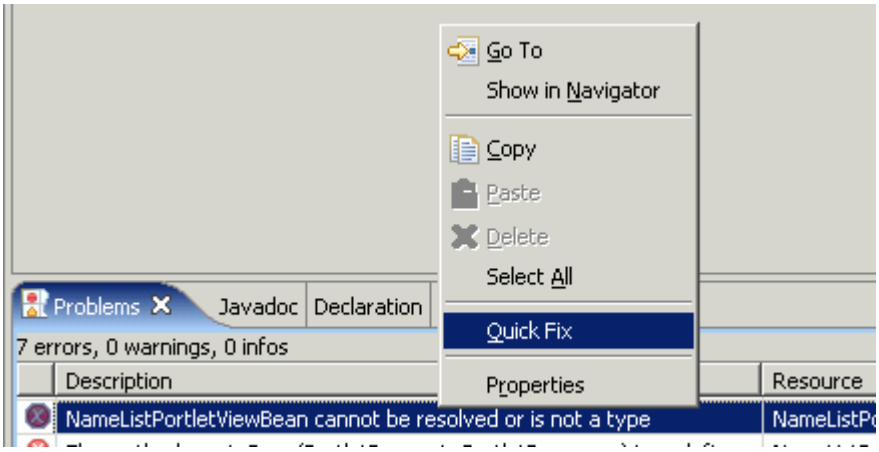
```
/**
 * @see org.apache.jetspeed.portlet.PortletAdapter#doEdit(PortletRequest,
 *      PortletResponse)
 */
public void doEdit(PortletRequest request, PortletResponse response)
        throws PortletException, IOException {
    NameListPortletViewBean bean = createBean(request, response);

    // create the returnURI to the portlet
    PortletURI returnURI = response.createReturnURI();

    // Preserve the cancelURI in the request object to the portlet
    request.setAttribute("cancelURI", returnURI.toString());

    // create a PortletAction to listen for and add action to request object
    returnURI.addAction("submit");

    // Preserve the submit URI action
    request.setAttribute("saveURI", returnURI.toString());

    // Preserve the saveURI in the request object
    request.setAttribute("NameListPortletViewBean", bean);

    // Invoke the JSP to render
    getPortletConfig().getContext().include(
        "/namelist/jsp/NameListPortletEdit." + getJspExtension(request),
        request,
        response);
}
```

__ h. Save NameListPortlet.java. Type **Ctrl-S** to save.

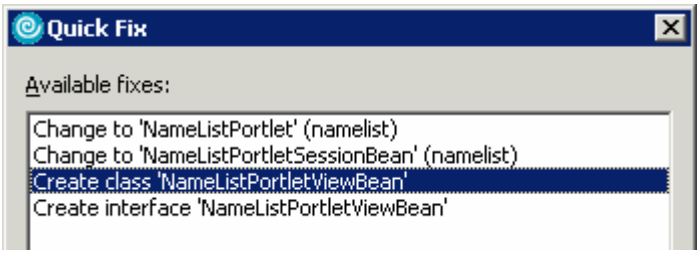
__ i. **Close** the editor for NameListPortlet.java.

____ 3. Use the Quick Fix functionality to create the NameListPortletViewBean Java bean.

__ a. In the Problems view, right-click on the row in the table that says "**NameListPortletViewBean cannot be resolved or is not a type**" and select **Quick Fix**. See the screen capture below for more details.



__ b. In the Quick Fix window, select **Create class 'NameListViewBean'** and click **OK**.



__ c. In the New window, leave the default values and click **Finish**.

__ d. The NameListPortletViewBean.java file will be opened in a Java Editor.

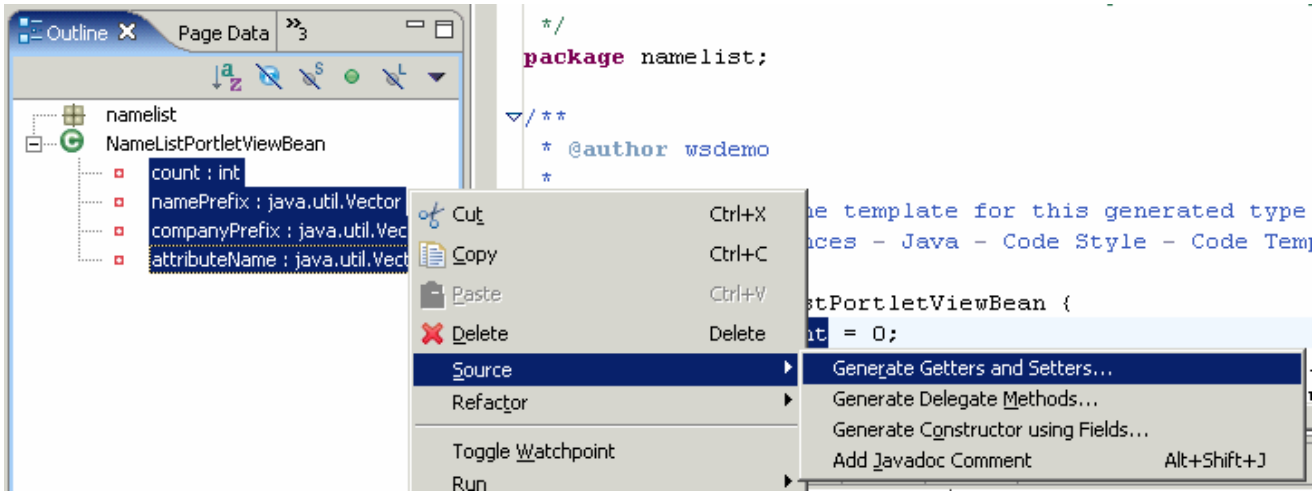
____ 4. Edit the NameListPortletViewBean.java file.

__ a. Add additional private fields to the **NameListPortletViewBean** class just below the class' declaration. The appropriate lines of code are highlighted in the following screen capture. For your convenience, these lines of code can be found in **<LAB_FILES>\<LAB_NAME>\snippets\snippet8.txt**.

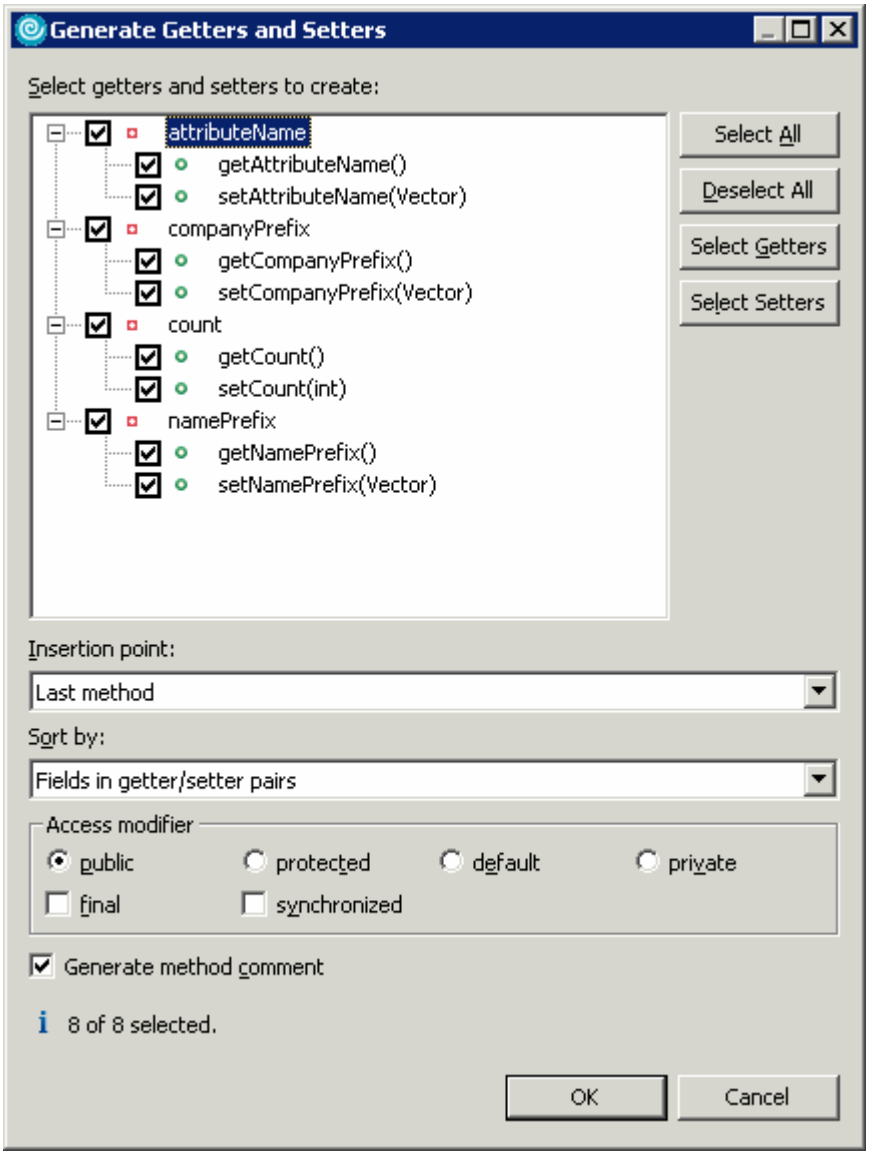
```
/**
 * @author wsdemo
 *
 * TODO To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Style - Code Templates
 */
public class NameListPortletViewBean {
    private int count = 0;
    private java.util.Vector namePrefix = new java.util.Vector();
    private java.util.Vector companyPrefix = new java.util.Vector();
    private java.util.Vector attributeName = new java.util.Vector();

}
```

__ b. Open the Outline view. Select **Window > Show View > Other.** In the Show View window, expand **Basic** and select **Outline**. Click **OK**.

__ c. In the Outline view, select the fields **count**, **namePrefix**, **companyPrefix**, and **attributeName** (by holding Ctrl key). Right-click on the selection and select **Source > Generate Getter and Setter**, as shown in the screen capture below.

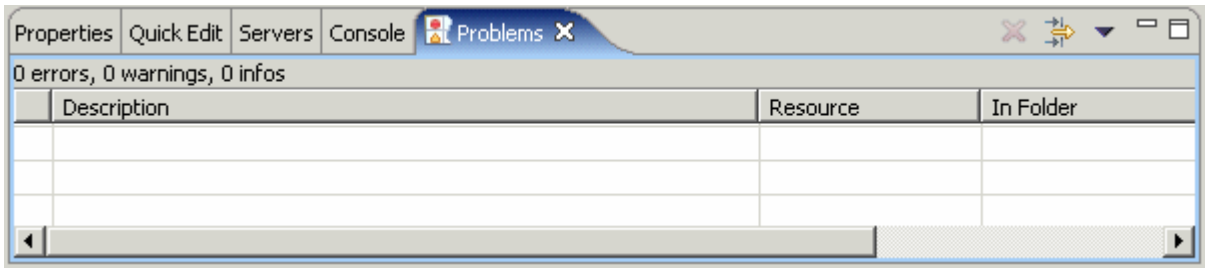


__ d. In the Generate Getters and Setters window, click **Select All**. Leave the rest of the fields at their default values and click **OK**.



__ e. Save NameListPortletViewBean.java. Type **Ctrl-S** to save.

__ f. Verify that the errors in the Problems view have gone away.



__ g. **Close** the editor for NameListPortletViewBean.java.

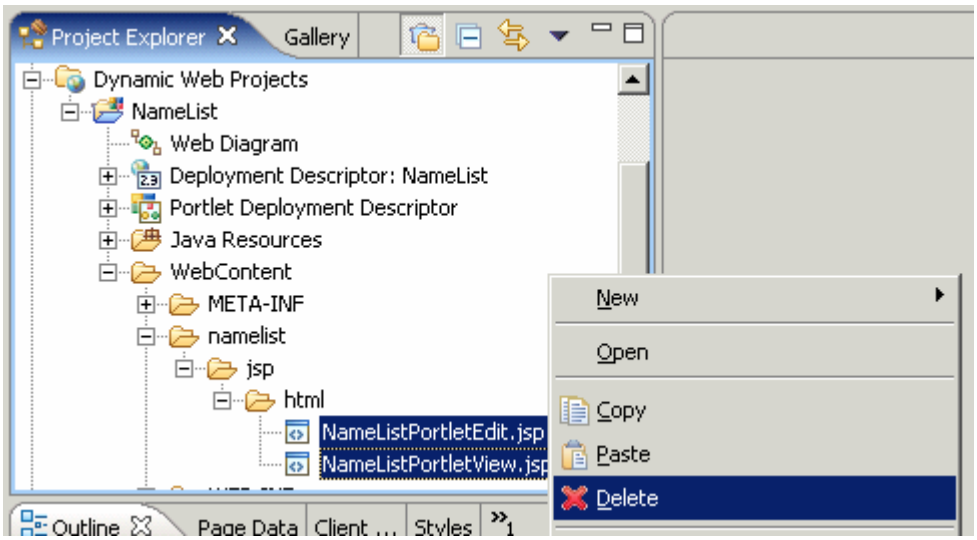
Part 3: Complete the Implementation of the Name List Portlet

____ 1. Import new versions of the view mode (NameListPortletView.jsp) and edit mode's (NameListPortletEdit.jsp) JSP pages.

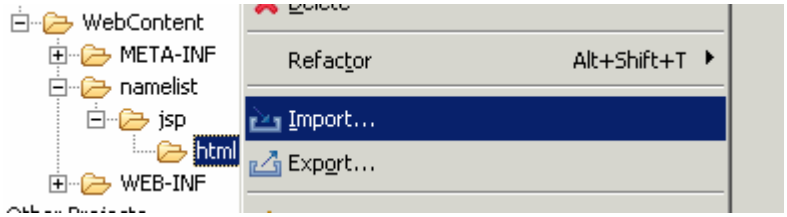
__ a. Navigate to the View Mode's JSP pages. In the Project Explorer view, expand **Dynamic Web Projects > NameList > WebContent > namelist > jsp > html**.

__ b. Under the html folder, use the Ctrl key to select both the **NameListPortletView.jsp** and the **NameListPortletEdit.jsp** files.

__ c. Right-click the selection and select **Delete** (click Yes to confirm deletion).

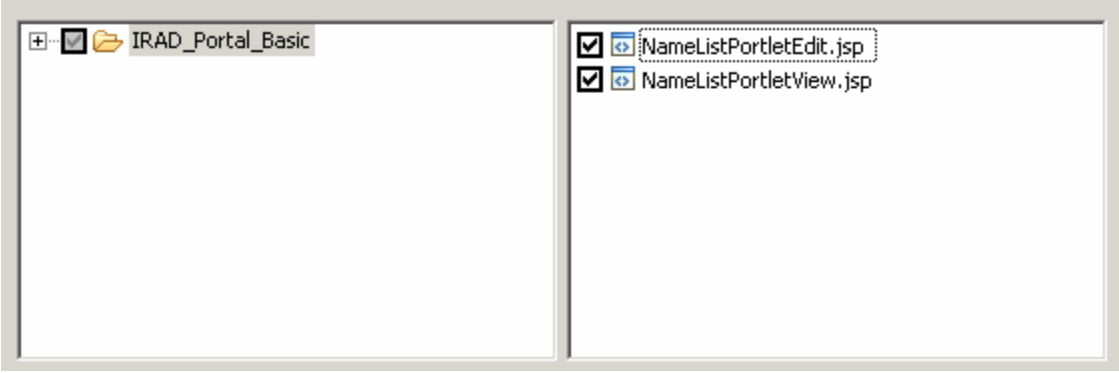


__ d. Right-click on the **NameList > Web Content > namelist > jsp > html** folder and select **Import…**



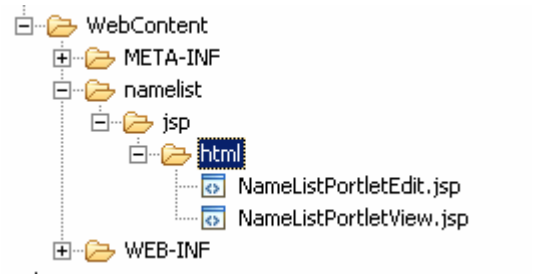
__ e. In the Import window, select **File system** and click **Next >**.

__ f. In the File system panel, for the From directory field enter **<LAB_FILES>\<LAB_NAME>**. Select the files **NameListPortletEdit.jsp** and **NameListPortletView.jsp** (see the screen capture below). For the Into folder field, make sure it states **NameList/WebContent/namelist/jsp/html**. Select **Overwrite existing resources without warning**.



__ g. Click **Finish**.

__ h. Verify that the new JSP files were imported correctly to the NameList > WebContent > namelist > jsp > html directory.

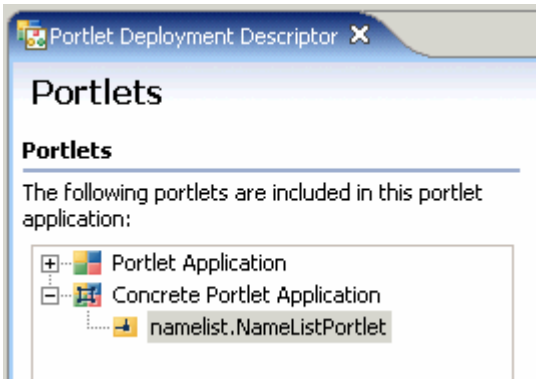


____ 2. Edit the Portlet Deployment Descriptor.

__ a. Navigate to the Portlet Deployment Descriptor. In the Project Explorer view, expand **Dynamic Web Projects > NameList**.

__ b. Double click on **Portlet Deployment Descriptor** to open it in the Portlet Deployment Descriptor editor.

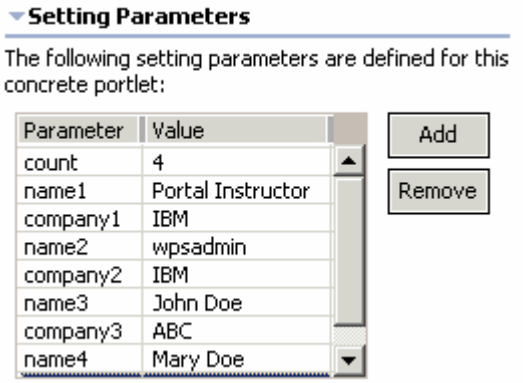
__ c. In the Portlet Deployment Descriptor editor, expand **Concrete Portlet Application** and select **namelist.NameListPortlet**.



__ d. Under **Setting Parameters** (you may have to scroll down to see these values), click **Add** and enter the following values:

Parameter	Values
count	4
name1	Portal Instructor
company1	IBM
name2	wpsadmin
company2	IBM
name3	John Doe
company3	ABC
name4	Mary Doe
company4	XYZ

__ e. After all of the values have been added it should look similar to the screen capture below.



__ f. Save the Portlet Deployment Descriptor. Type **Ctrl-S** to save.

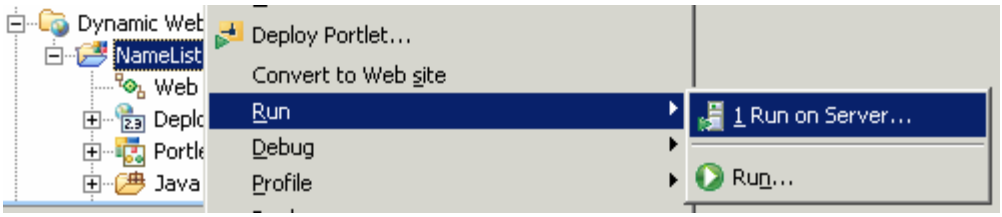
__ g. **Close** the Portlet Deployment Descriptor editor.

Part 4: Test the Name List Portlet in the Portal Test Environment

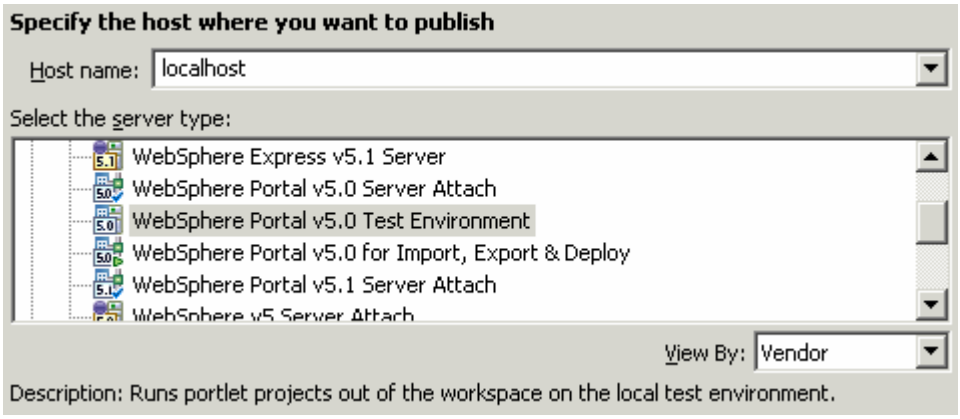
In this part, you will use the WebSphere Portal 5.0 Test Environment to test your portlet application.

____ 1. Create the Test Environment.

__ a. In the Project Explorer view, expand **Dynamic Web Projects**. Right-click on **NameList** and select **Run > Run on Server…**.



__ b. In the Server Selection window, select '**Manually define a server**'. In the Select the server type box, select **WebSphere Portal v5.0 Test Environment**. Click **Next >**.



__ c. In the WebSphere Server Configuration Settings panel, leave the default values and click **Finish**.

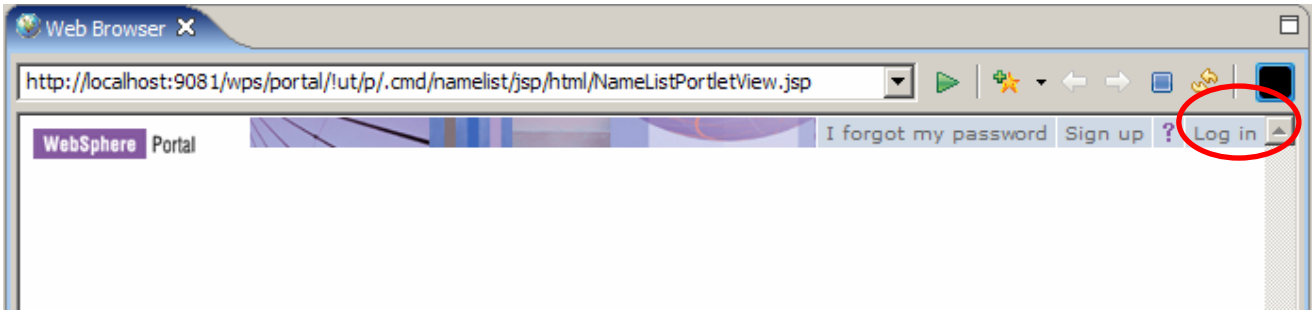
__ d. A new server and server configuration will be automatically generated, and the Portlet Project will be added to this server. The server will then start, and a web browser will open.

NOTE: Depending on the size of the system on which you are working, the starting of the Portal server may require approximately 5 minutes.

____ 2. Test the NameList portlet application.

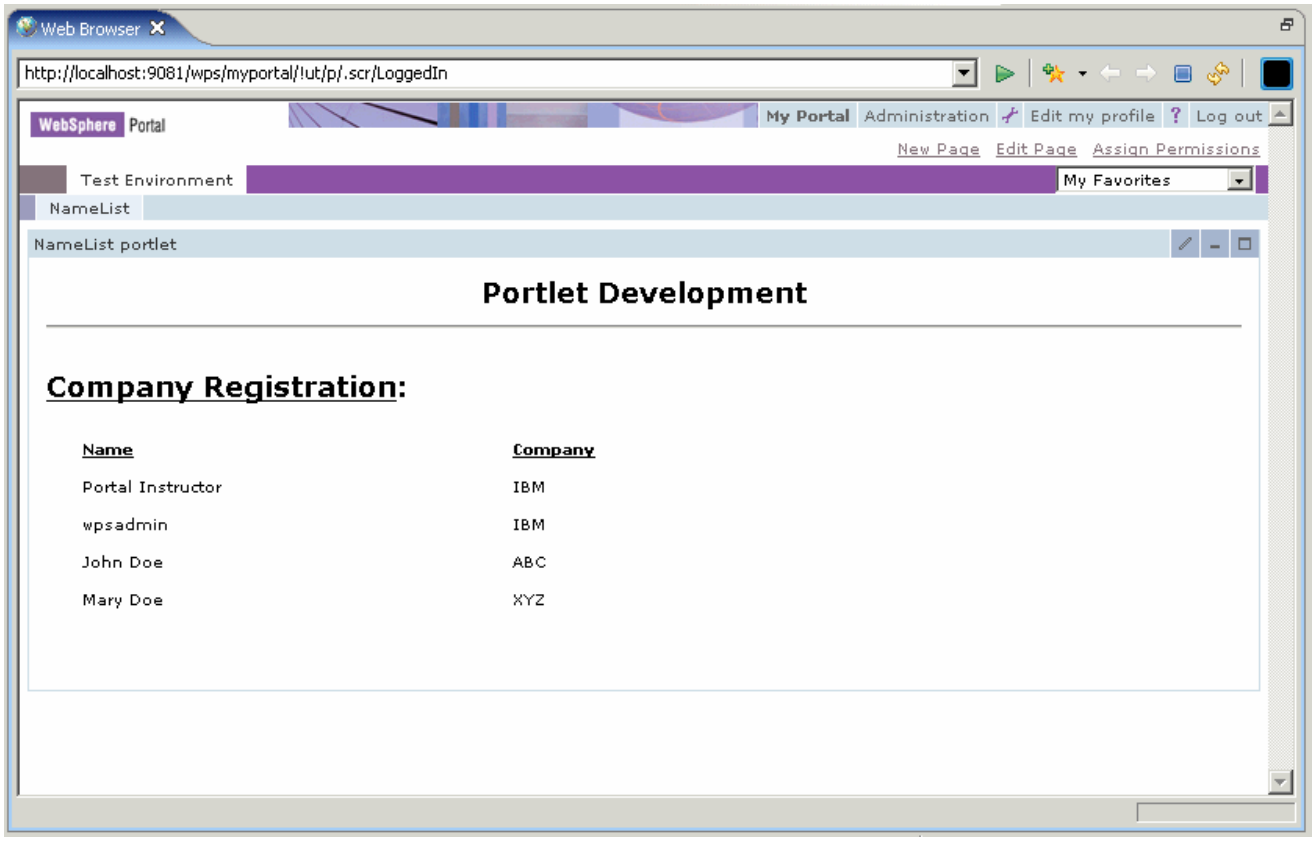
Note: If you are not automatically logged into the portal follow steps a and b, otherwise skip to step c.

__ a. You must log into the portal before the request to the NameList portlet application is carried out. Click on the Login in link (the icon is circled in the screen capture below).

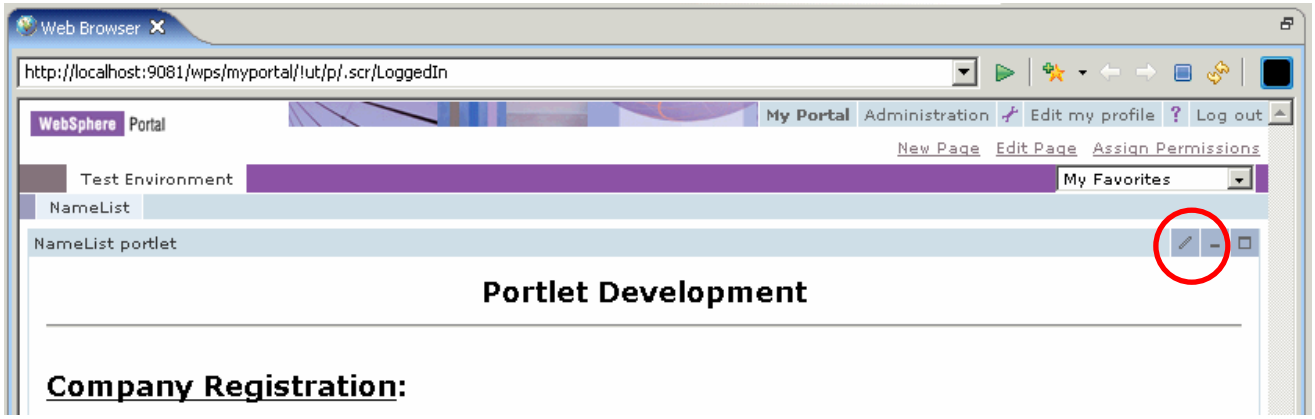


__ b. Enter **wpsadmin** for the User ID and Password and click **Log in**.

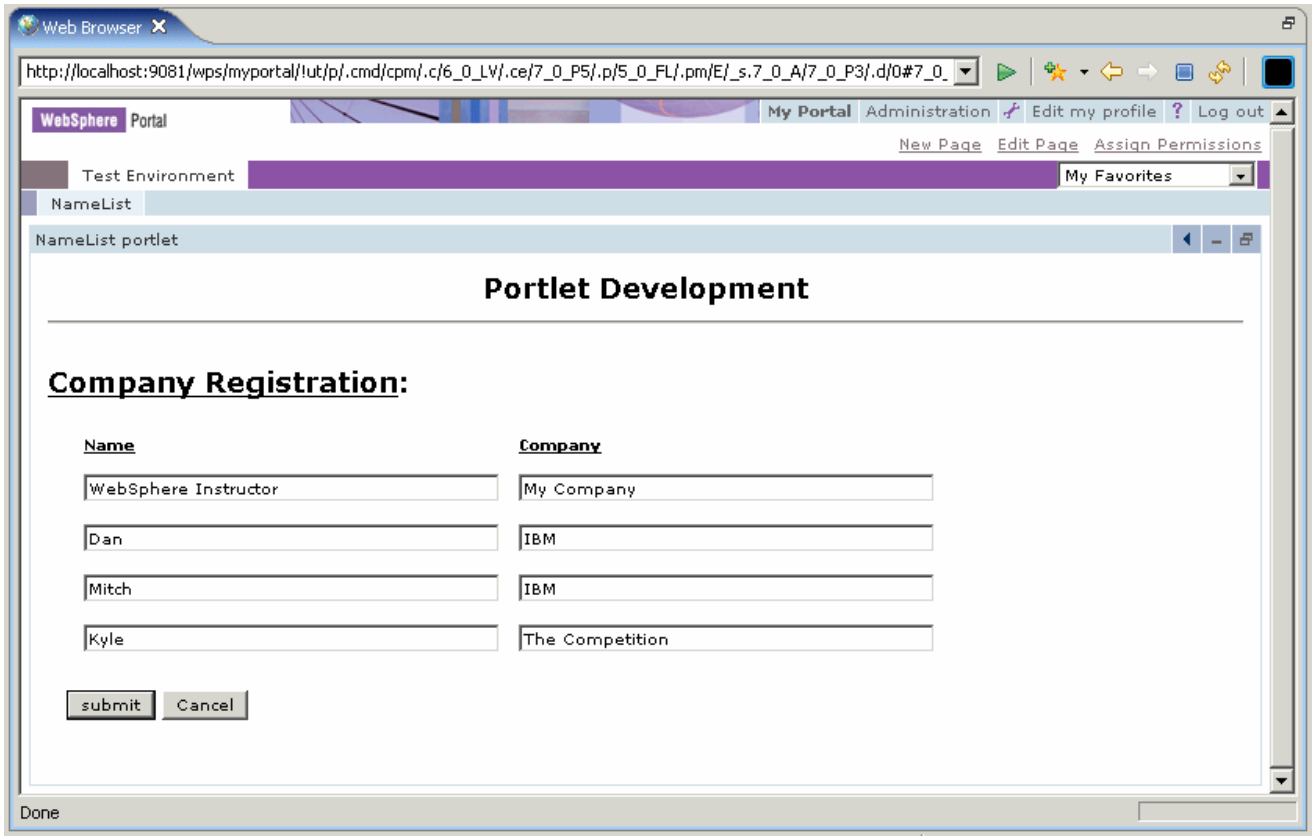
__ c. Your portlet application's view mode will be displayed. This shows the values you originally entered in the Portlet settings defined in the Portlet Deployment Descriptor.



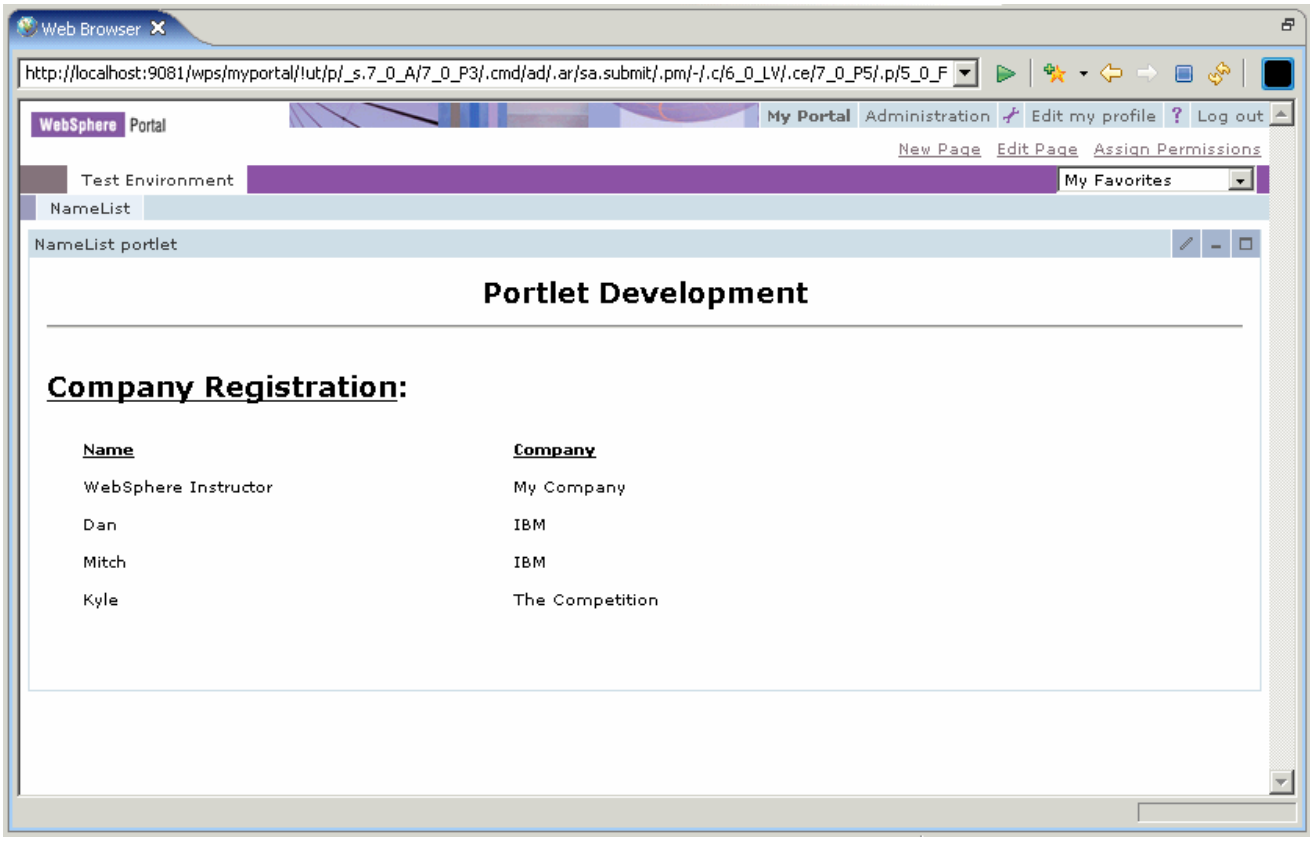
__ d. **Edit** the information within the portlet. **Click** on the edit icon in the portlet's toolbar to go into edit mode (the icon is circled in the screen capture below).



__ e. Change the values to anything you want and then click the **submit** button.



__ f. You are then taken to the view mode again and can see your changes (see the screen capture below).



____ 3. Stop the Portal Test Environment.

__ a. Switch to the Servers view by selecting **Window > Show View > Servers**.

__ b. Right-click on **WebSphere Portal v5.0 Test Environment @ localhost** and select **Stop**.

__ c. Watch the Console view and wait for the server to come to a complete stop.

____ 4. Exit Application Developer. Select **File > Exit**.

What you did in this exercise

Using the Portal Tools within Application Developer, you learned how to create a simple portlet application that supports both the view and edit modes. You then used the Portal Test Environment to test your portal application.

Solution Instructions

____ 1. Start Rational Application Developer.

__ a. Select **Start > Programs > IBM Rational > IBM Rational Application Developer V6.0 > Rational Application Developer**.

__ b. A dialog box will be displayed allowing you to select the location where you would like the workspace directory to be stored. Enter **<LAB_FILES>\<LAB_NAME>\solution\workspace** for the location and select **OK**. Application Developer will start with an empty workspace. An empty workspace will leave your existing workspace untouched and help avoid name conflicts between what you may already have in your workspace and what you will be creating in this lab.

____ 2. Import the completed portlet application.

__ a. Select **File > Import…**. The Import wizard will open up.

__ b. In the Select panel, select **EAR file** and click **Next >**.

__ c. In the Enterprise Application Import panel, for the EAR file field enter **<LAB_FILES>\<LAB_NAME>\solution\NameListEAR.ear**. Change the Target server field to **WebSphere Portal v5.0**. Click **Finish**.

____ 3. Test the portlet application.

__ a. Switch to the Web Perspective. Select **Window > Open Perspective > Web**.

__ b. Follow the steps in Part 4 to test the completed portlet application.

Trademarks and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	iSeries	OS/400	Informix	WebSphere
IBM(logo)	pSeries	AIX	Cloudscape	MQSeries
e(logo)business	xSeries	CICS	DB2 Universal Database	DB2
Tivoli	zSeries	OS/390	IMS	Lotus

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Microsoft, Windows, Windows NT, and

the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both. Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are

trademarks of Intel Corporation in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries. Linux is a registered trademark of Linus Torvalds. Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

This page is left intentionally blank.