Implementing JSR 168 inter-portlet communication using Rational Application Developer V6.0 and WebSphere Portal V5.1

Level: Intermediate

Asim Saddal (mailto:asaddal@us.ibm.com)
Senior IT Specialist, IBM

14 September 2005

Table of contents

Abstract

This tutorial shows you how to implement JSR 168 compliant cooperative portlets using IBM® Rational® Application Developer V6.0 and IBM WebSphere Portal V5.1. You see how to pass multiple values from the source portlet to the target portlet, without defining a complex data type inside a Web Services Definition (WSDL) file.

Introduction

One of the most significant advantages of the portlet architecture is the portlets' ability to communicate with each other to create dynamic, interactive applications. You can use portlet messages to share information, notify each other of a user's actions, or to simply help better manage screen real estate. Messages can be sent to a specific portlet, to all portlets on a page, or to all portlets in a single portlet application. In order to make full use of this potential, you need to adequately architect the entire portlet application, anticipating inter-portlet communication.

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 to take repititious actions in other portlets on the page.

Cooperation between source and target portlets is facilitated by an IBM® WebSphere® Portal runtime entity called the *property broker*. Portlets on a page can cooperate in this way even if they were developed independently, without the programmer's awareness of the existence of the other cooperative portlets.

The most important difference between portlet messaging and cooperative portlets is that cooperative portlets are more loosely coupled than portlets using messaging. Cooperative portlets do not have to know the name of the target portlet, even if they do not broadcast data. The matching of source and target portlets is done at runtime based on registered properties and actions.

During runtime, the WebSphere Portal property broker is responsible for enabling cooperative portlets. Therefore, you can rely on the WebSphere Portal property boker to enable the passing of complex data between portlets, without needing to define complex data types inside the WSDL files.

JSR 168 is a specification from the Java™ Community Process for portlet development. WebSphere Portal V5.1 provides support for the JSR 168 API. With an IBM extension, WebSphere Portal V5.1 supports cooperative portlets for JSR 168 portlets, in which one JSR 168 portlet can communicate with another JSR 168 portlet.

About the sample code

A download accompanies this tutorial. The download includes a WAR file which contains the two demo portlets with the source code and WSDL files. You can deploy

the WAR file directly into WebSphere Portal, but you need to create the wires between them. See Deploy the DemoPortlets.war portlet for more details.

Before you begin

To work through this tutorial, you need to basic knowledge of portlet development and deployment.

To develop and deploy the sample application, you use the following IBM products:

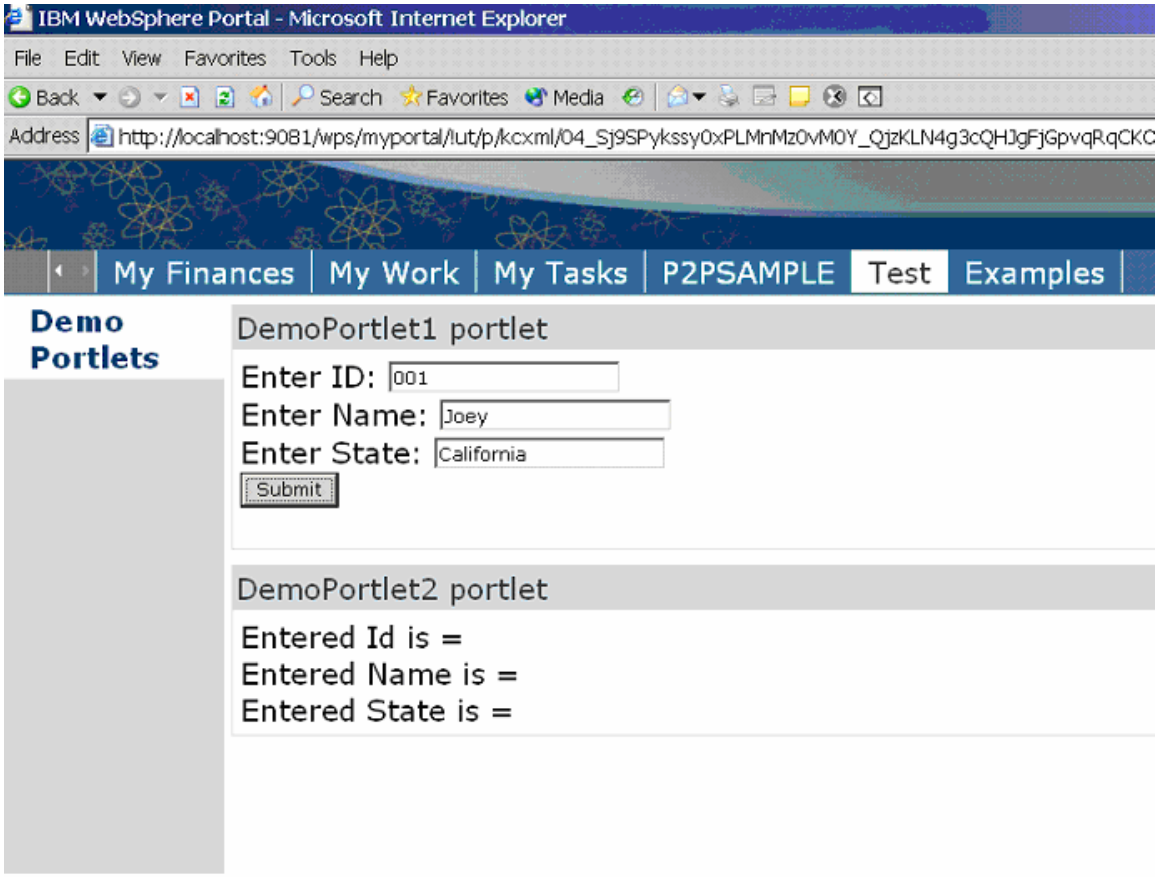
- Rational Application Developer for Rational Software Development Platform V6.0
- WebSphere Portal V5.1.x

Introducing the DemoPortlets scenario

In the DemoPortlets scenario you create two JSR 168 portlets. The DemoPortlet1 passes multiple values to the DemoPortlet2 without defining complex data type inside Web Services Definition (WSDL) file.

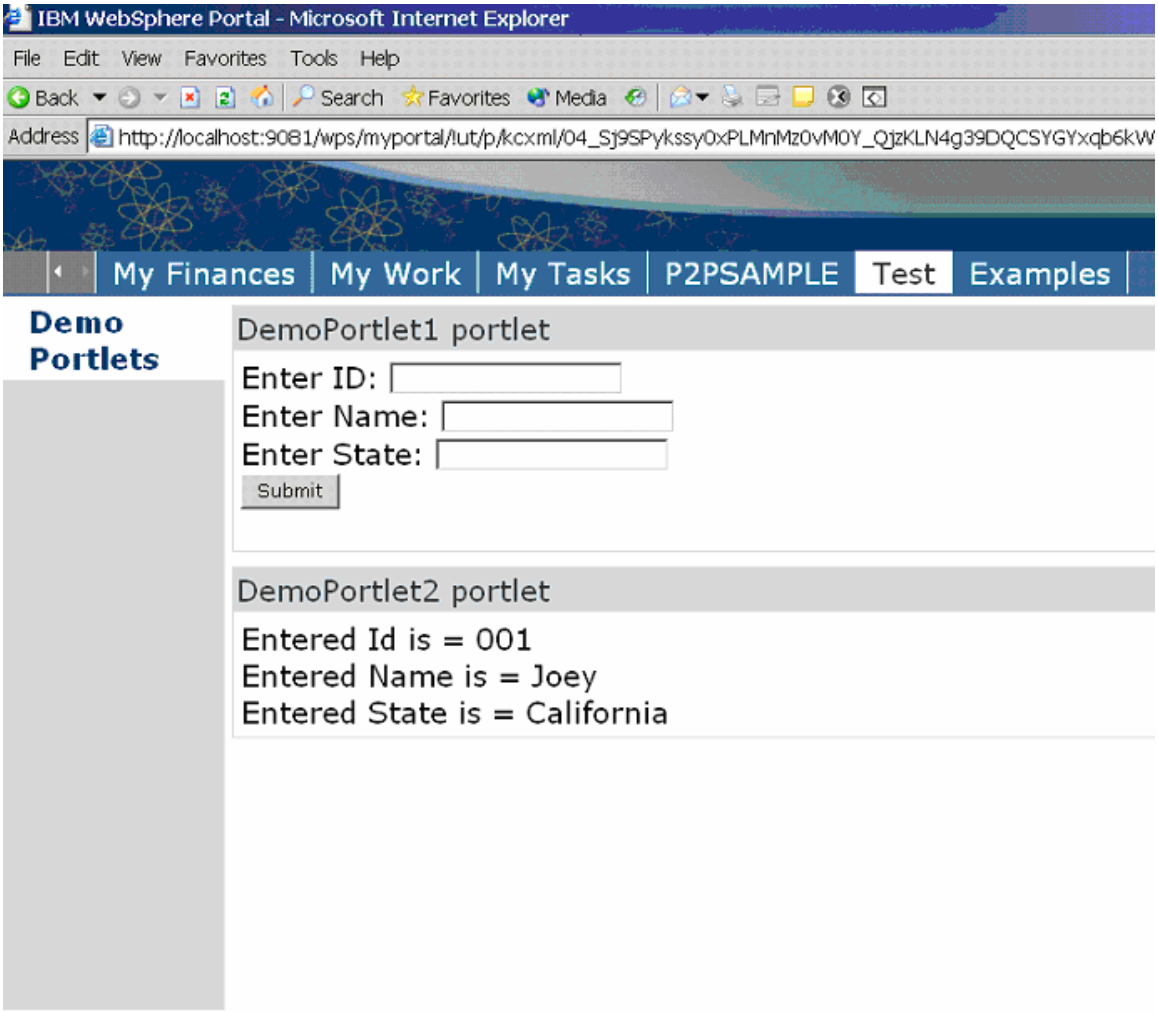
DemoPortlet1, as shown in Figure 1, gets three fields of user input and then passes these values to the target portlet, Demo2Portlet, as shown in Figure 2. DemoPortlet1 is the source portlet in this scenario.

Figure 1. DemoPortlet1 receives user input



DemoPortlet2 is the target portlet. It receives and then displays the three values from DemoPortlet1.

Figure 2. DemoPortlet2 displaying values it received from DemoPortlet1



Creating a JSR 168 portlet project

To develop the application, start the Rational Application Developer (hereafter called Application Developer).

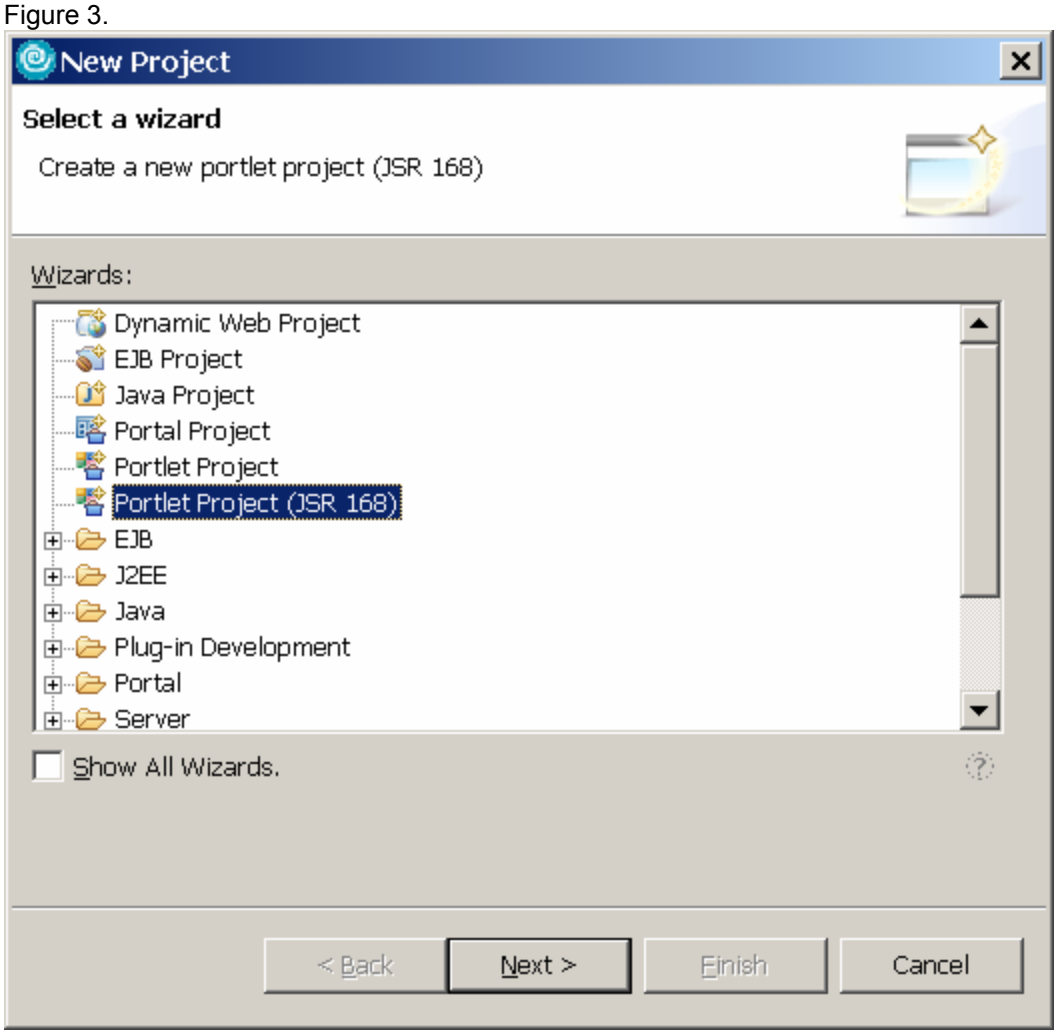
To create the project:

1. Open the Web perspective by clicking **Window => Open Perspective =>Web**.

2. Click **New => Other**.

3. Select **Portlet Project (JSR 168)** from the list. The New Portlet Project (JSR 168) wizard launches, as shown in Figure 4.

Figure 3.

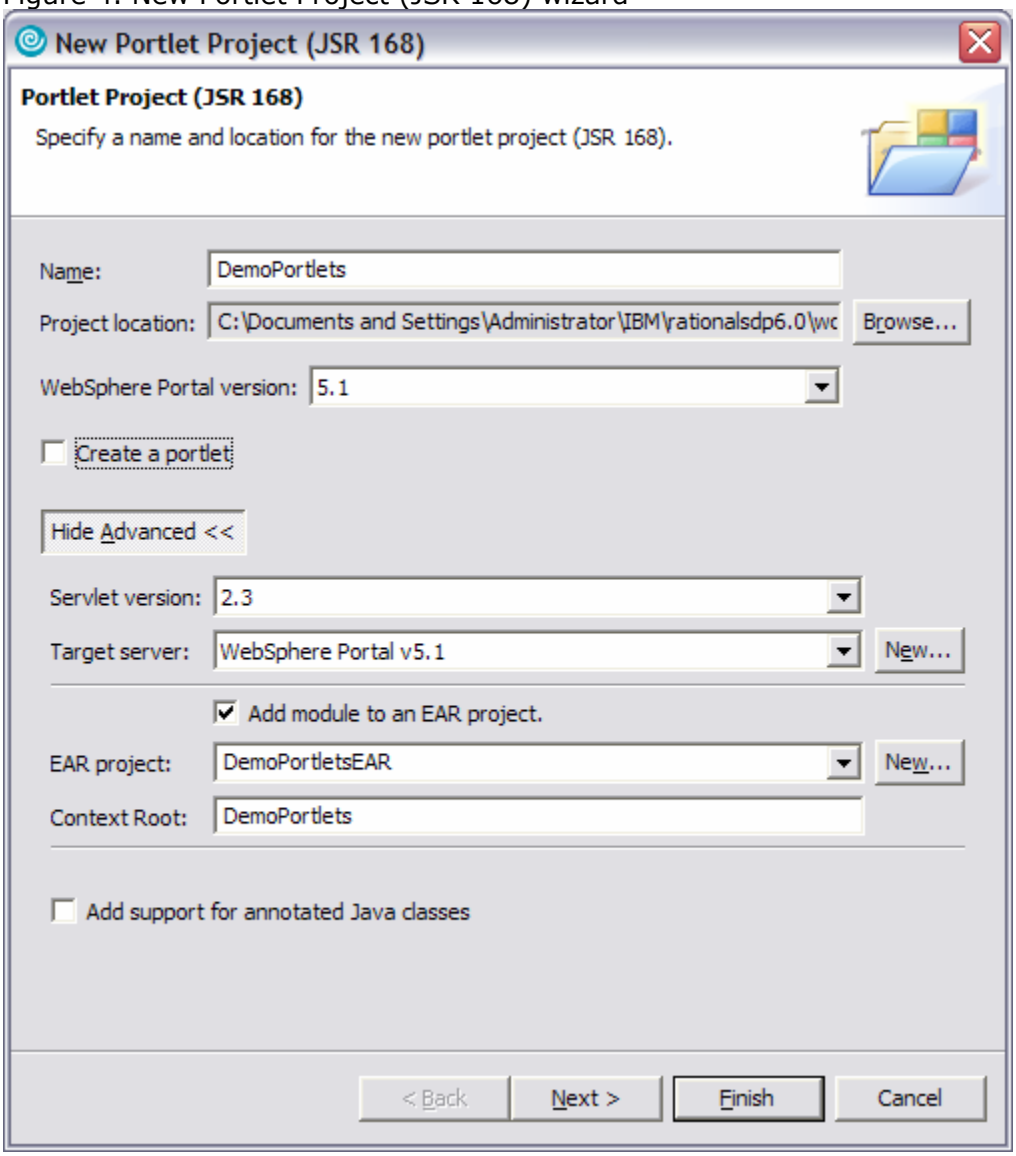


4. Enter `DemoPortlets` as the **Name**.

5. Clear the **Create a portlet** checkbox. You will create your portlets separately in order to have better control over portlet naming conventions.

6. Click the **Show Advanced** button.
7. Select WebSphere Portal V5.1 Unit Test Environment in the **Target Server** list.
8. Accept defaults for the other fields.

Figure 4. New Portlet Project (JSR 168) wizard



9. Click **Finish**.

Creating DemoPortlet1

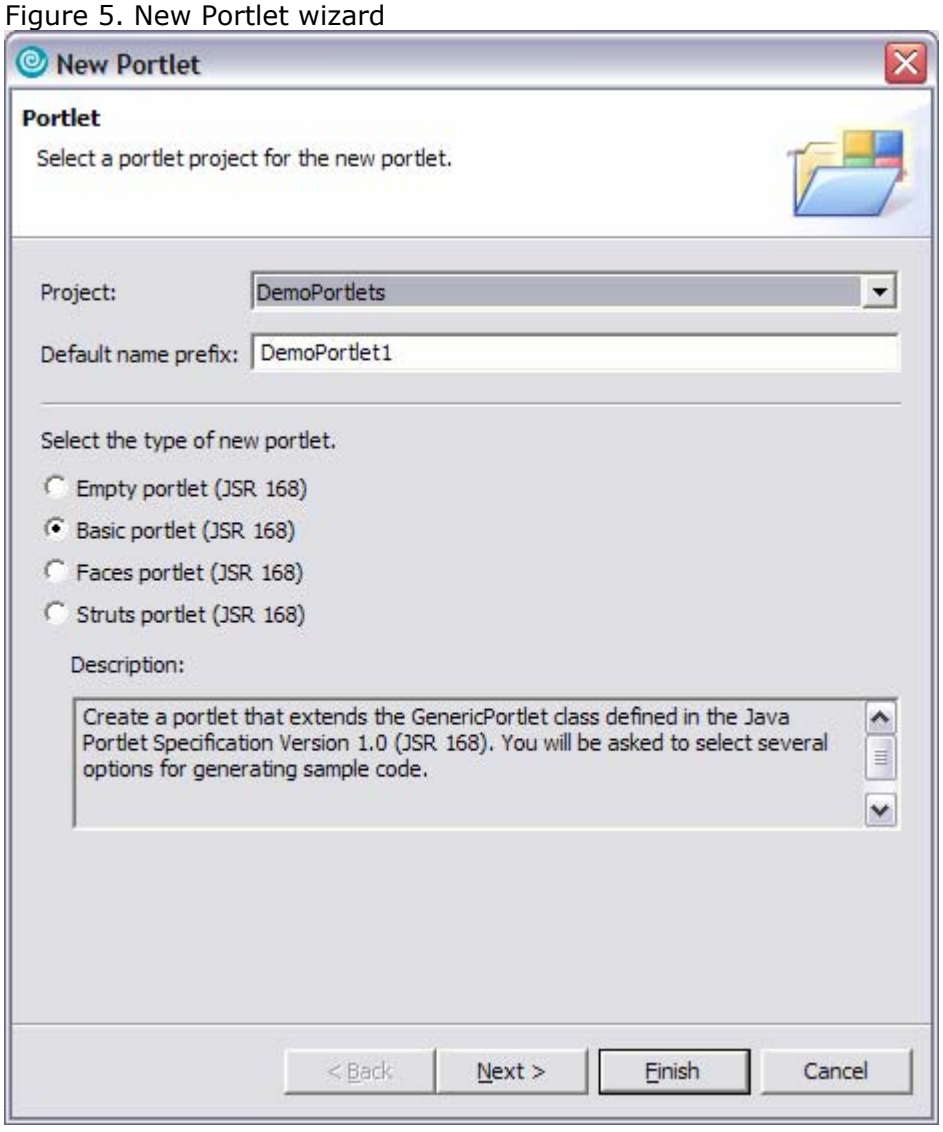
To create the first portlet:

1. In the Project Navigator view, select the **DemoPortlets** project.

2. Right-click to bring up the context menu, and click **New => Portlet**. The New Portlet wizard launches.

3. Enter `DemoPortlet1` as the **Default Name prefix**, and click **Next**.

   Figure 5. New Portlet wizard



4. Accept the default values, and click **Finish**.

Creating DemoPortlet2

Create a second portlet in the same project following same steps you used for the first portlet, but name this one `DemoPortlet2`.

Enabling the DemoPortlet1 as a source

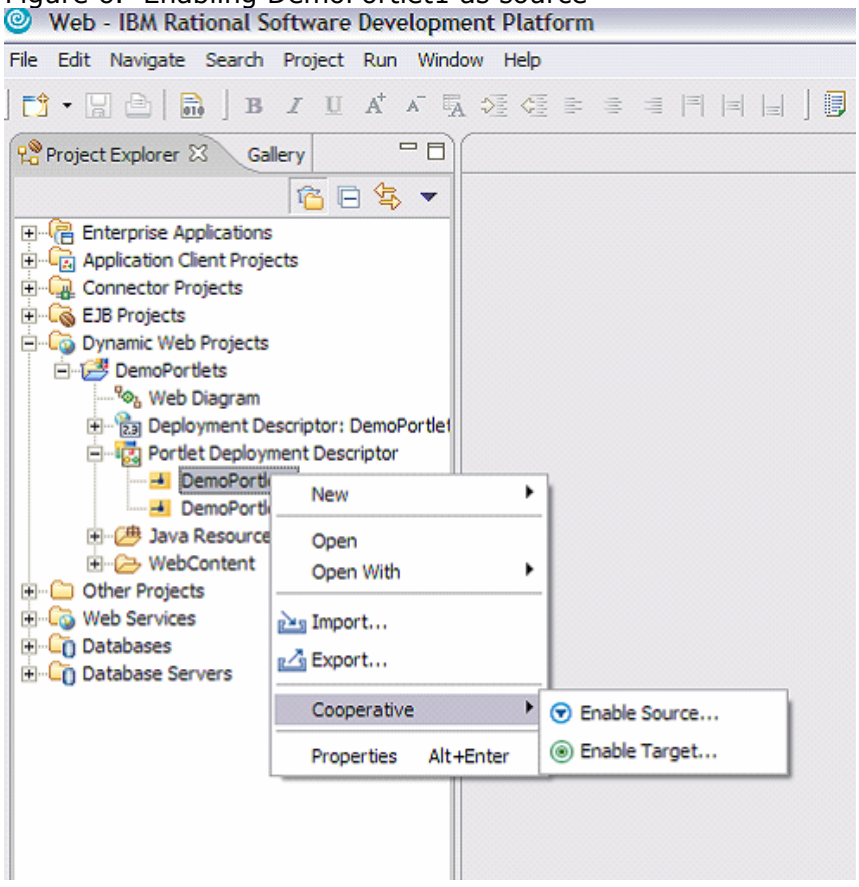
JSR 168 portlets can cooperate with each other by exchanging properties using the property broker. A WSDL file describes the publish (or send) attributes to the property broker.

Describing the source with WSDL

To enable the DemoPortlet1 portlet as a property source:

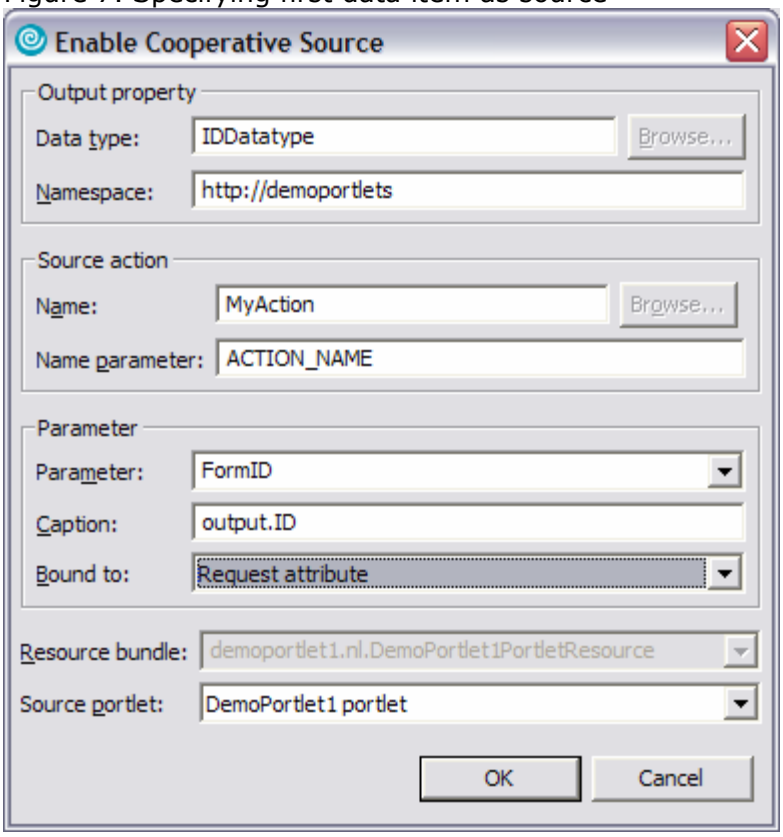
1. Right-click the portlet in the Project Explorer view.
2. Select **Cooperative => Enable Source**.

Figure 6. Enabling DemoPortlet1 as source



In the Enable Cooperative Source wizard, enter these values, as illustrated in Figure 7:
A name for the **Data type**, such as `IDDatatype`
The **Namespace** for the new data type, such as `http://demoportlets`
What you want your parameter **Bound to**, such as `Request attribute`

Figure 7. Specifying first data item as source



The term *parameter* refers to how the value will be transferred from the source portlet to the target portlet. The choices are:

None: This setting implies that you will not specify the way the value is passed, so the default behavior for portlet

Render parameter: Supports only strings. The string value is bound to the RenderRequest object. The render parameter can be set during the action phase and retrieved during the render phase of the portlet lifecycle. It cannot be retrieved during the action phase.

Request parameter: Supports only strings. The string value is bound to the ActionRequest object, and can be retrieved during the action processing stage of the portlet lifecycle. The parameter value is meaningless at the conclusion of request processing (that is, it will not persist beyond a single invocation).

Request attribute: Supports any JavaBean™ type. The bean is bound to the ActionRequest object. Lifecycle Request Parameter.

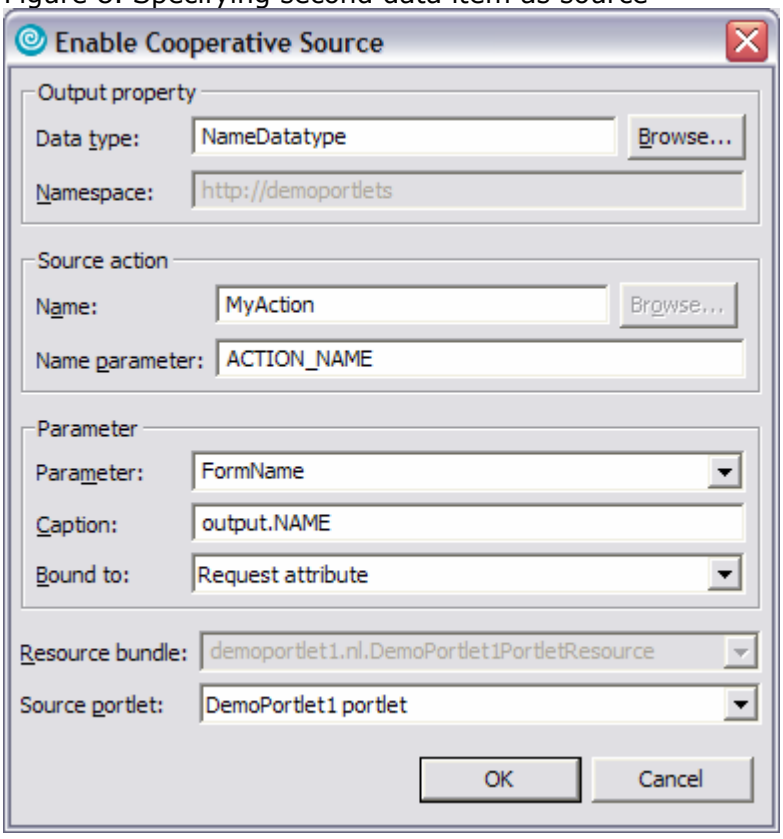
Session: Supports any JavaBean type. The bean is bound to the session object and persists for the duration and the portal server.

The Enable Cooperative Source wizard generates a WSDL file that describes the portlet to the property broker, tagged with a distinctive icon to indicate that it is a property source. The WSDL file contains the following sections:

Types: Describes data types (using XML schema) that can be emitted by the source portlet.
Messages: Describes messages that can be produced or consumed by the portlet.
Port Type: Describes the abstract interface of the portlet, as seen by the property broker.
Binding: Describes how the abstract interface (port type) is implemented.

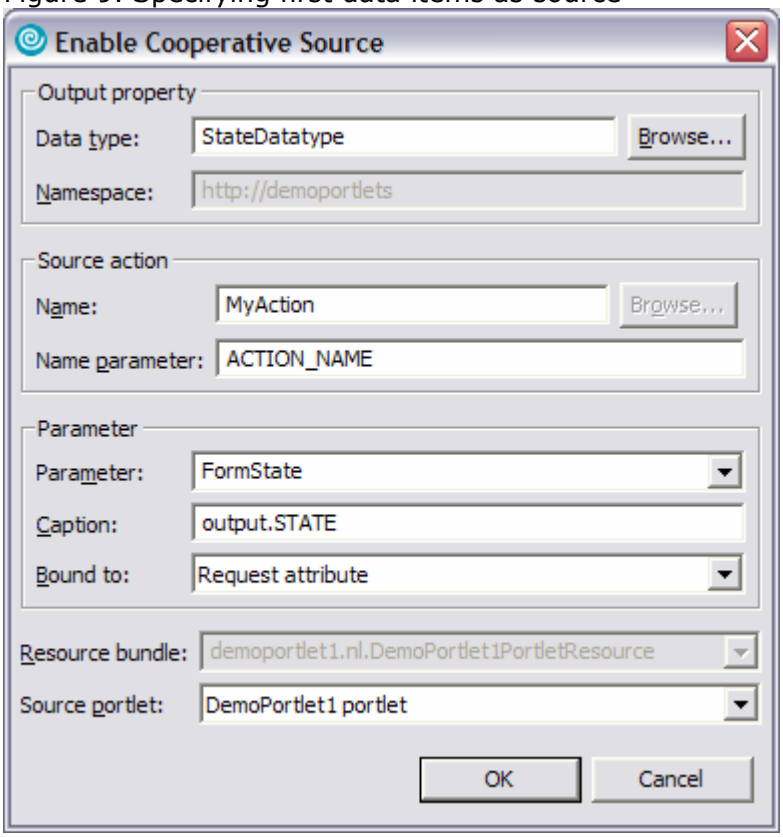
To define the second value you want to pass from the first portlet to the target:
1. Right-click the portlet again and select **Cooperative->Enable Source**
2. Define the values indicated in Figure 8.

Figure 8. Specifying second data item as source



For the third attribute, specify the values shown in Figure 9:

Figure 9. Specifying first data items as source



Listing 1. WSDL for the source portlet

The wizard produces a WSDL file similar to this:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="DemoPortlet1portlet_Service"
targetNamespace="http://demoportlets"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:portlet="http://www.ibm.com/wps/c2a"
xmlns:tns="http://demoportlets"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<types>
<xsd:schema targetNamespace="http://demoportlets">
  <xsd:simpleType name="IDDatatype">
    <xsd:restriction base="xsd:string"></xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="NameDatatype">
    <xsd:restriction base="xsd:string"></xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="StateDatatype">
    <xsd:restriction base="xsd:string"></xsd:restriction>
 </xsd:simpleType>
</xsd:schema>
</types>
<message name="IDDatatype_Response">
```

```
  <part name="IDDatatype_Output" type="tns:IDDatatype" />
  <part name="NameDatatype_Output" type="tns:NameDatatype" />
  <part name="StateDatatype_Output" type="tns:StateDatatype" />
</message>
<portType name="DemoPortlet1portlet_Service">
  <operation name="DemoPortlet1portlet">
    <output message="tns:IDDatatype_Response" />
  </operation>
</portType>
<binding name="DemoPortlet1portlet_Binding"
type="tns:DemoPortlet1portlet_Service">
  <portlet:binding />
  <operation name="DemoPortlet1portlet">
  <portlet:action name="MyAction" actionNameParameter="ACTION_NAME"
type="standard" caption="output.data" description="Output.Data" />
   <output>
    <portlet:param name="FormID" partname="IDDatatype_Output"
boundTo="request-attribute" caption="output.ID" />
    <portlet:param name="FormName" partname="NameDatatype_Output"
boundTo="request-attribute" caption="output.NAME" />
    <portlet:param name="FormState" partname="StateDatatype_Output"
boundTo="request-attribute" caption="output.STATE" />
   </output>
  </operation>
 </binding>

</definitions>
```

Define captions and description for `<portlet:action>` and <portlet:param> attributes. These fields are optional but are very useful when creating wires.

Enabling DemoPortlet2 as target

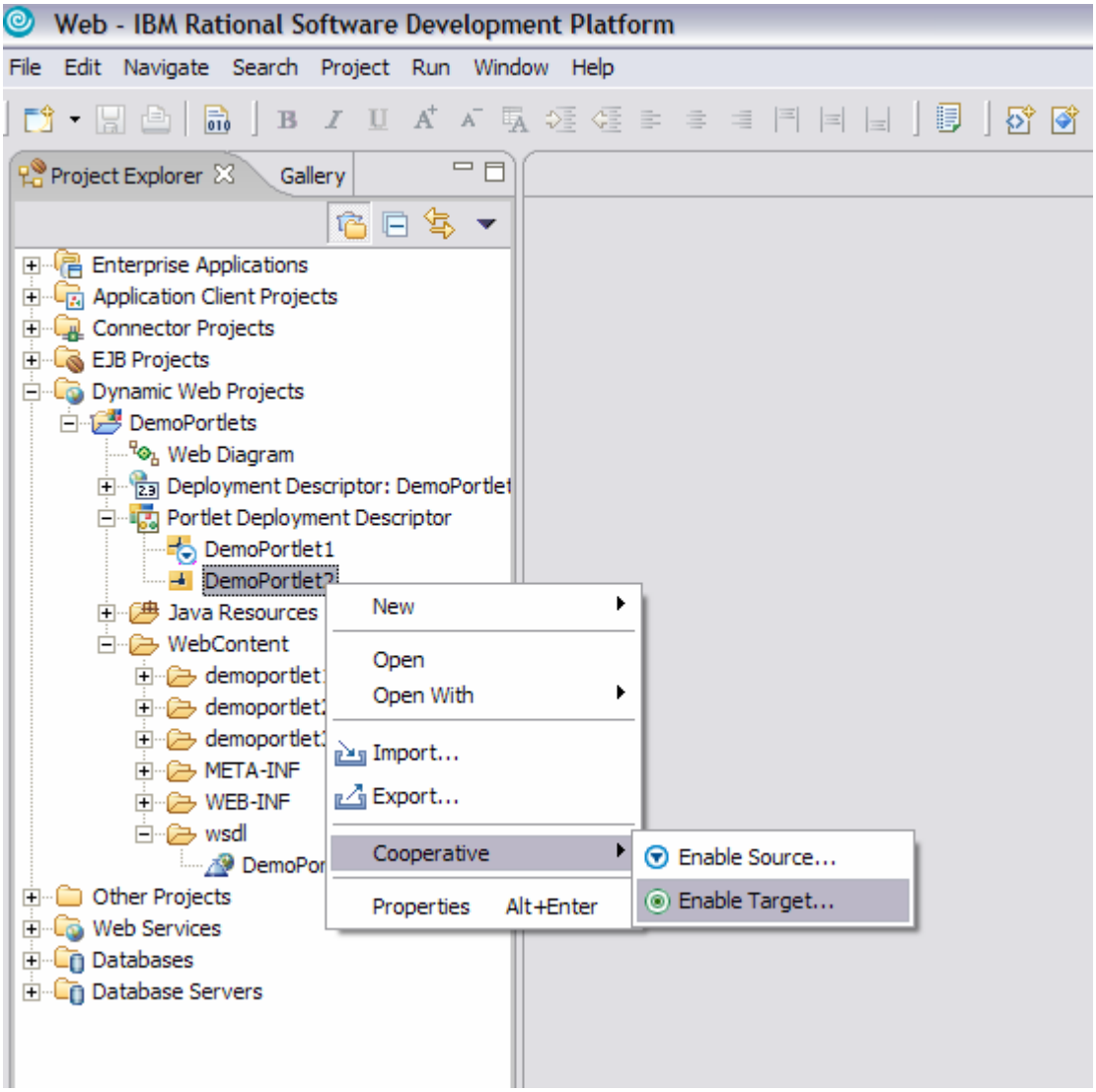
Just as you enabled the first portlet to act as a source, you need to enable the second portlet to act as a target.

Generating the WSDL

You use the Enable Cooperative Target wizard to generate the WSDL.

To launch the wizard, select the portlet and click **Cooperative => Enable Target** (this step is nearly identical to enabling cooperative source).

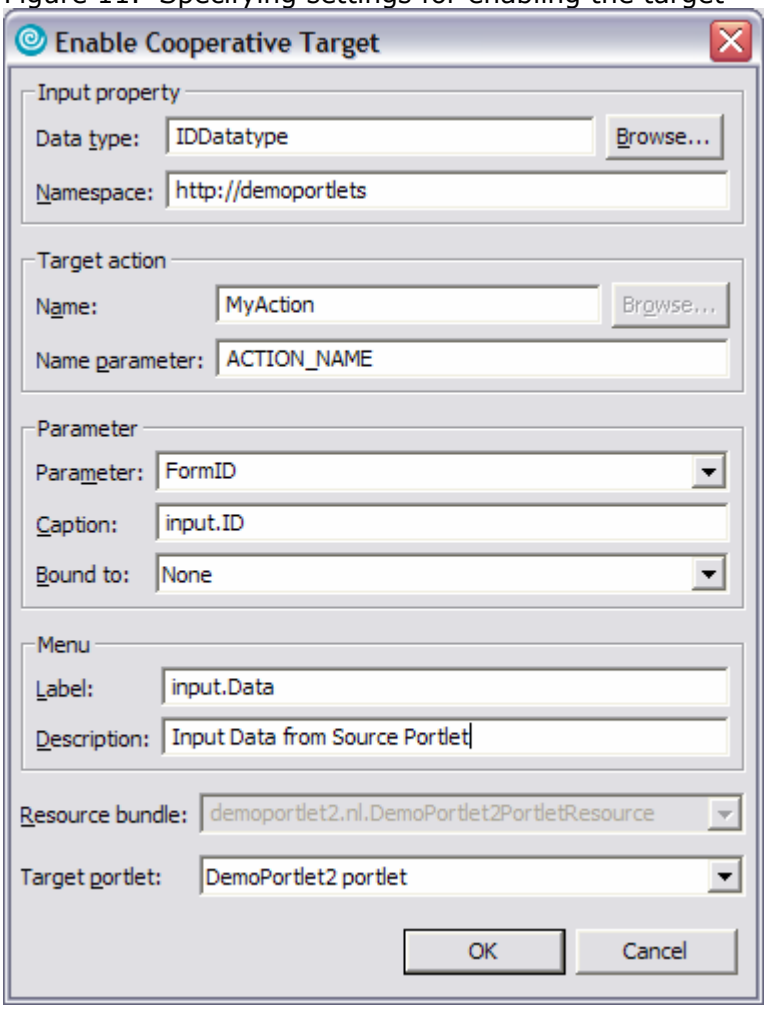
Figure 10. Enabling the second portlet to act as target


Web - IBM Rational Software Development Platform
File Edit Navigate Search Project Run Window Help
Project Explorer
Gallery
Enterprise Applications
Application Client Projects
Connector Projects
EJB Projects
Dynamic Web Projects
DemoPortlets
Web Diagram
Deployment Descriptor: DemoPortlet
Portlet Deployment Descriptor
DemoPortlet1
DemoPortlet2
Java Resources
WebContent
META-INF
WEB-INF
wsdl
Other Projects
Web Services
Databases
Database Servers
New
Open
Open With
Import...
Export...
Cooperative
Properties Alt+Enter
Enable Source...
Enable Target...

In the Enable Cooperative Target wizard, enter these values (shown in Figure 11):

Data type: Use the exact same name that you used when you enabled the source portlet.
Namespace: Enter the same one you used for the source.
Action: Use the exact same name that you used when you enabled the source portlet.
Parameter: Use the exact same name that you used when you enabled the source portlet.
Bound to: Choose **None** from the list.
Label and Description: These fields are optional. However, it's a good idea to fill them because because you can use them if you decide to create wires.

Figure 11. Specifying settings for enabling the target



In order to define the remaining two attributes for the same target portlet, DO NOT use the wizard. You need to define them inside the WSDL file directly because wizard only supports simple data types for a single entry. If you try to use the wizard to define multiple entries, the previous values would be overwritten.

Open the DemoPortlet2portlet.wsdl file and make the modifications shown in the WSDL code listing below.

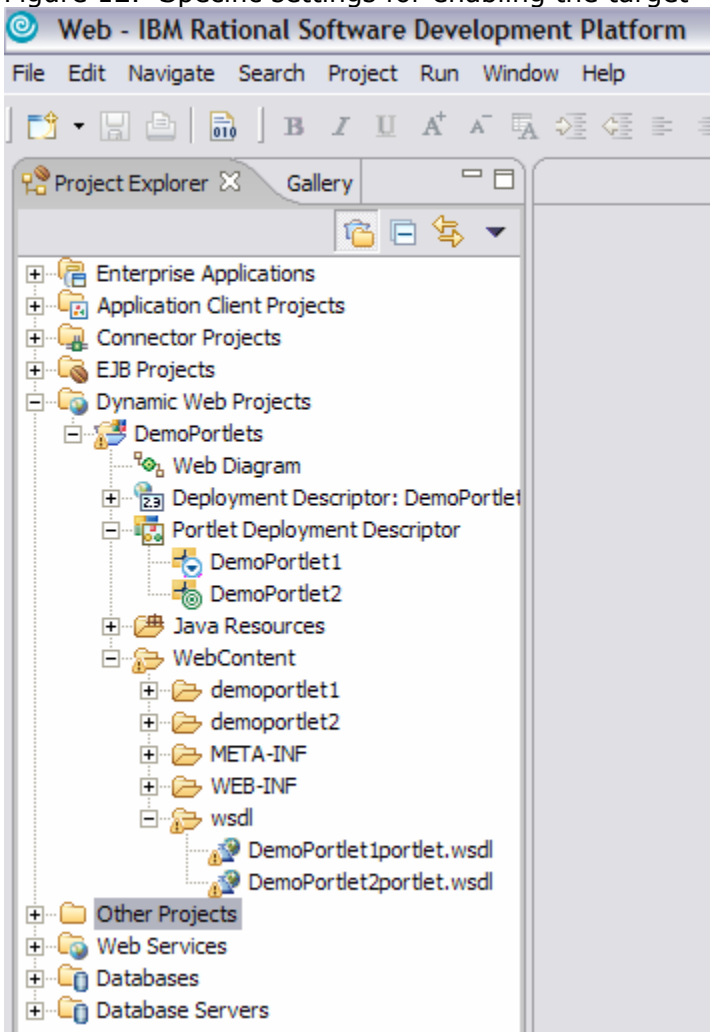
Listing 2: WSDL for target portlet

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="DemoPortlet2portlet_Service"
targetNamespace="http://demoportlets"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:portlet="http://www.ibm.com/wps/c2a"
xmlns:tns="http://demoportlets"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<types>
<xsd:schema targetNamespace="http://demoportlets">
  <xsd:simpleType name="IDDatatype">
   <xsd:restriction base="xsd:string"></xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
<xsd:schema targetNamespace="http://demoportlets">
  <xsd:simpleType name="NameDatatype">
   <xsd:restriction base="xsd:string"></xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
<xsd:schema targetNamespace="http://demoportlets">
  <xsd:simpleType name="StateDatatype">
   <xsd:restriction base="xsd:string"></xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
</types>
<message name="IDDatatype_Request">
  <part name="IDDatatype_Input" type="tns:IDDatatype" />
  <part name="NameDatatype_Input" type="tns:NameDatatype" />
  <part name="StateDatatype_Input" type="tns:StateDatatype" />
</message>
<portType name="DemoPortlet2portlet_Service">
  <operation name="DemoPortlet2portlet">
    <input message="tns:IDDatatype_Request" />
  </operation>
</portType>
 <binding name="DemoPortlet2portlet_Binding"
type="tns:DemoPortlet2portlet_Service">
 <portlet:binding />
  <operation name="DemoPortlet2portlet">
   <portlet:action name="MyAction" actionNameParameter="ACTION_NAME"
type="standard" caption="input.Data" description="Input Data from
Source Portlet" />
    <input>
      <portlet:param name="FormID" partname="IDDatatype_Input"
caption="input.ID" />
      <portlet:param name="FormName" partname="NameDatatype_Input"
caption="input.NAME" />
      <portlet:param name="FormState" partname="StateDatatype_Input"
caption="input.STATE" />
    </input>
  </operation>
 </binding>
</definitions>
```

Again, define `captions` and `description` for `<portlet:action>` and `<portlet:param>` attributes.

Application Developer generates `DemoPortlet1portlet.wsdl` and `DemoPortlet2portlet.wsdl`, as shown in the project view (Figure 12).

Figure 12. Specific settings for enabling the target



Modifying the Java Classes

After defining the attributes and the portlet action in the WSDL file, you need to modify the Java code for the source and target portlets. In order for a portlet to be a source of data, you need to use a WSDL file to flag sharable data on the output pages. In order for a portlet to be a target, you must describe a subset of the portlet actions, including type information for the action parameters. You use the WSDL file, with some custom extensions, as the supported format for declaring actions and their associated parameters for JSR 168 compliant portlets.

1- Open `DemoPortlet1Portlet.java` file.

2- Declare the following attributes:

```
public static final String FORM_ID              = "FormID";
public static final String FORM_NAME            = "FormName";
public static final String FORM_STATE           = "FormState";
public static final String ACTION_TEXT          = "MyAction";
```

Make sure the values of `FORM_ID`, `FORM_NAME` and `FORM_STATE` are the same name defined for `<portlet:param>` in the WSDL file.

3- Modify `processAction()` method

```
public void processAction(ActionRequest request,
ActionResponse response) throws PortletException,
java.io.IOException {

      String id   = request.getParameter(FORM_ID);
      String name = request.getParameter(FORM_NAME);
      String state      = request.getParameter(FORM_STATE);

      request.setAttribute(FORM_ID, id);
      request.setAttribute(FORM_NAME, name);
      request.setAttribute(FORM_STATE, state);
}
```

3- Save the changes.

4- Open `DemoPortlet2PortletSessionBean.java` and define following attributes:

```
private String id = "";
private String name = "";
private String state = "";
```

5- Generate getter and setter methods for the above attributes and save the changes.

6- Open `DemoPortlet2Portlet.java` file.

7- Declare the following attributes:

```
public static final String FORM_ID        = "FormID";
public static final String FORM_NAME      = "FormName";
public static final String FORM_STATE     = "FormState";
public static final String ACTION_TEXT    = "MyAction";
```

Important: Make sure that the values for `FORM_ID`, `FORM_NAME`, and `FORM_STATE` are the same as the `name` values defined for `<portlet:param>` in the WSDL file.

8- Modify `processAction()` method:

```
public void processAction(ActionRequest request,
ActionResponse response) throws PortletException,
java.io.IOException {
```

```
   DemoPortlet2PortletSessionBean sessionBean =
getSessionBean(request);

   String id = request.getParameter(FORM_ID);
   String name = request.getParameter(FORM_NAME);
   String state = request.getParameter(FORM_STATE);

   sessionBean.setId(id);
   sessionBean.setName(name);
   sessionBean.setState(state);
}
```

9- Save the changes.

Modifying the JSP

You need to make the following modifications in the JSP code, so that a user can provide input values to the source portlet. The target portlet will display the input values on the page.

1- Open `DemoPortlet1PortletView.jsp`.

2- Modify the JSP code as follows:

```
<%@ page session="false" contentType="text/html"
import="java.util.*,javax.portlet.*,demoportlet1.*" %>
<%@taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
<portlet:defineObjects/>

<%
PortletURL actionUrl = renderResponse.createActionURL();
actionUrl.setParameter("ACTION_NAME",
DemoPortlet1Portlet.ACTION_TEXT);
%>

<FORM method="POST" action="<%= actionUrl.toString() %>">
Enter ID: <INPUT name="<%=DemoPortlet1Portlet.FORM_ID%>"
type="text"/> <br/>
Enter Name: <INPUT name="<%=DemoPortlet1Portlet.FORM_NAME%>"
type="text"/> <br/>
Enter State: <INPUT name="<%=DemoPortlet1Portlet.FORM_STATE%>"
type="text"/> <br/>
<INPUT name="<%=DemoPortlet1Portlet.FORM_SUBMIT%>" type="submit"
value="Submit"/>
</FORM>
```

3- Save the changes.

4- Open `DemoPortlet2PortletView.jsp`.

5- Modify the JSP code as follows:

```
<%@ page session="false" contentType="text/html"
import="java.util.*,javax.portlet.*,demoportlet2.*" %>
<%@taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
<portlet:defineObjects/>

<%
DemoPortlet2PortletSessionBean sessionBean =
(DemoPortlet2PortletSessionBean)renderRequest.getPortletSession().ge
tAttribute(DemoPortlet2Portlet.SESSION_BEAN);
%>

Entered Id is = <%= sessionBean.getId()%> <br/>
Entered Name is = <%= sessionBean.getName()%> <br/>
Entered State is = <%= sessionBean.getState()%>
```

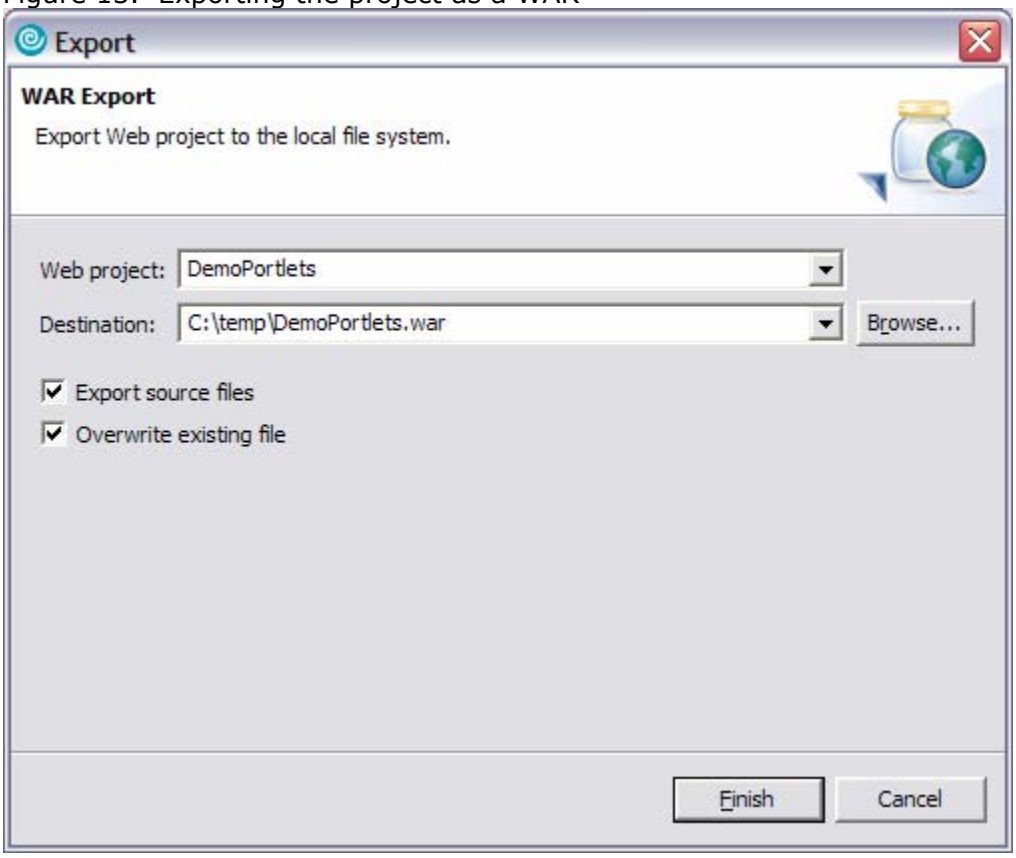
6- Save the changes.

Deploy the DemoPortlets.war portlet

Now, you export the WAR so that you can deploy the portlet application into WebSphere Portal 5.0 to test the portlets.

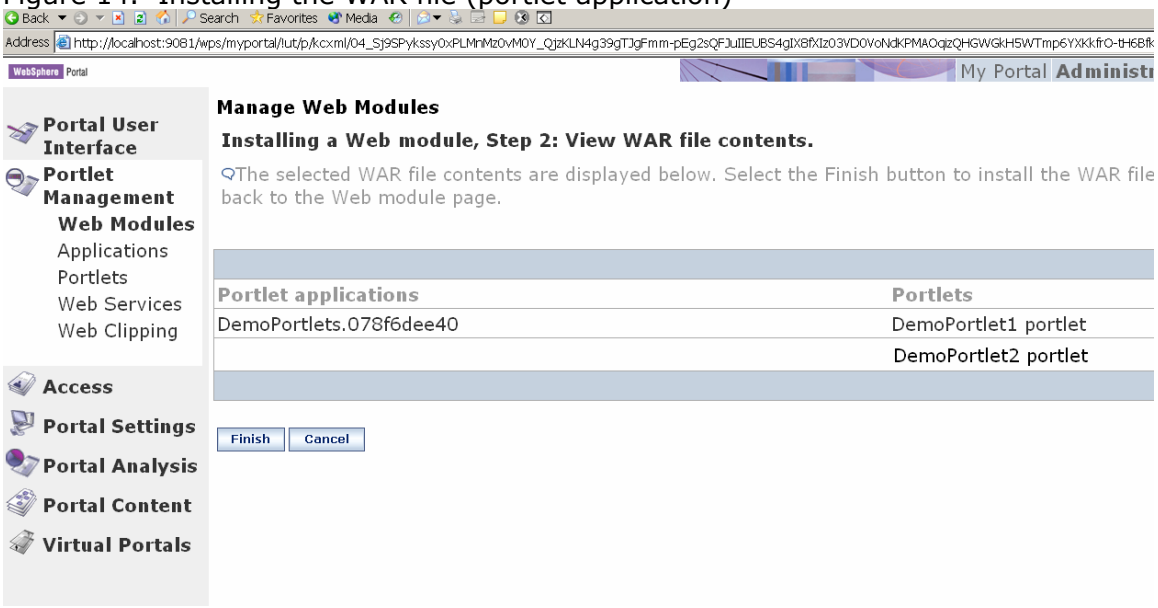
1- Export the `DemoPortlets` Project as a WAR file.

Figure 13. Exporting the project as a WAR



2- Install the `DemoPortlets.war` file in WebSphere Portal Server.

Figure 14. Installing the WAR file (portlet application)



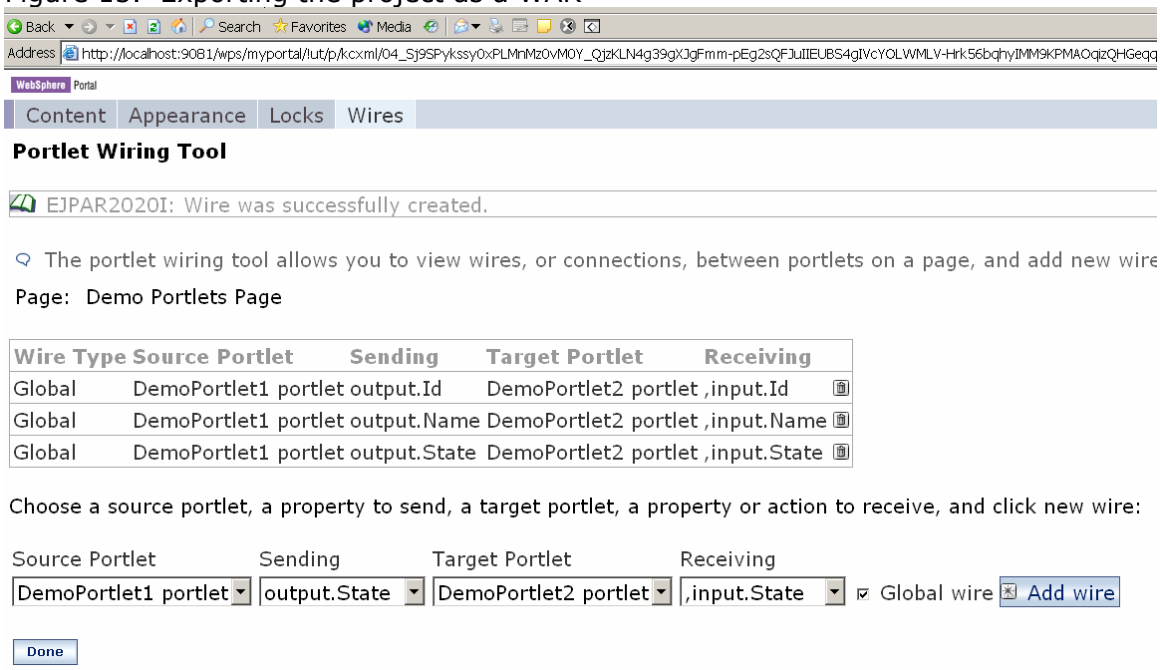
3- After the installation of the portlet is completed, create a page.

4- Place both the portlets on the same page.

5- Create the wires between both portlets by selecting the **Wire** link on top of the page.

6- Define a wire for each attribute defined and click **Add wire**, as shown in the figure.

Figure 15. Exporting the project as a WAR



7- After you have created all the wires, click **Done**.

8- Go to the page where both portlets are placed.

9- Test the portlets by entering dummy data and pressing **Submit**. You should see the values on the second portlet.

Conclusion

Even though the JSR 168 specification does not include a definition for inter-portlet communication, WebSphere Portal V5.1 provides an extension for cooperative portlets which enables one JSR 168 portlet to communicate with another JSR 168 portlet. Enterprise portals are filled with opportunities for applying cooperative portlet capabilities. You can use the simple example you just completed as an example for applying this capability in your own portal.

Download

To get the download file, see the cover page for this tutorial, at:
https://www6.software.ibm.com/developerworks/education/websphere/0509_saddal/0509_saddal.html

Resources

WebSphere Portal Server V5.1 InfoCenter
http://publib.boulder.ibm.com/pvc/wp/510/ent/en/InfoCenter/index.html

WebSphere Portal product documentation
http://www.ibm.com/developerworks/websphere/zones/portal/proddoc.html

WSDL specifications
http://www.w3.org/TR/wsdl

IBM Redbook: IBM Rational Application Developer V6 Portlet Application Development and Portal Tools
http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246681.html

Developing JSF Portlets with Rational Application Developer 6.0 and WebSphere Portal Server 5.1, Part 1: Developing JSR168 JSF Portlets
http://www.ibm.com/developerworks/rational/library/05/genkin/index.html

Other articles about portlet cooperation:
http://www.ibm.com/developerworks/views/websphere/libraryview.jsp?search_by=cooperative+portlets&product_by=WebSphere+Portal

Other JSR 168 resources:
http://www.ibm.com/developerworks/views/websphere/libraryview.jsp?search_by=JSR+168&product_by=WebSphere+Portal

WebSphere Portal zone
http://www.ibm.com/developerworks/websphere/zones/portal/

Rational Application Developer for WebSphere Software
http://www.ibm.com/developerworks/rational/products/rad/

About the author

Asim S Saddal is a Senior IT Specialist with the WebSphere Application and Portal server practice in the IBM Software Services for WebSphere (ISSW) in Chicago, Illinois.