Developing a portlet application using IBM Portlet Container Tools for WAS Liberty Profile and consuming it using WSRP

Manish Aneja,
Project Lead, RAD Portal Tools & IBM Portlet Container Tools for Liberty Profile

Table of Contents

Overview

IBM Portlet Container Tools for WebSphere Application Server Liberty Profile hosted at IBM Collaboration Solutions Catalog is a light weight set of tools to speed up the development and test process for portlet development targeted for WebSphere Application Server Liberty Profile. It offers the capabilities to create JSR 286 & JSR 168 compliant portlet projects targeted to the IBM WebSphere Application Server Liberty profiles. It offers capabilities to deploy the portlet projects from within the workspace to the Liberty profile to speed up the test and development process.

The article will illustrate how you can create and publish a JSR 286 portlet project on WebSphere Application Server Liberty Profile server using IBM Portlet Container Tools for WebSphere Application Server Liberty Profile and then how the same portlet application can be consumed as a remote service by WebSphere Portal Server.

Usecase

The objective is to depict the process of creating deploying the portlet over WAS Liberty profile and then consume them though the WebSphere Portal Server. To achieve this you will work with a portlet project containing two portlets, the publisher portlet would publish the event and the subscribed portlet would listen to the published event.

- The portlets would communicate with each other via JSR286 based events mechanism.
- Both the portlets would be deployed to the WAS Liberty profile.
- Finally you would configure the WSRP (i.e. web services for remote portlets) producer on a WebSphere Portal server and consume these portlets through WSRP by placing them on a portal page.

We would keep the events usecase simple by passing a string parameter across the publisher and subscriber portlet.

Web Services for Remote Portlets (WSRP)

Web Services for Remote Portlets (WSRP) provides a standard for the communication between Producers who provide WSRP services and Consumers who consume them. WSRP defines a web service communication interface for presentation-oriented Web services. Producers and consumers use this interface for providing and consuming the Web services.

WSRP is based on open standards to enable cross product interoperability. WSRP allows non-portal technologies to integrate with WebSphere Portal. With WSRP portal users can choose from a rich variety of remote content, portlets and applications to integrate them into their portal.

Prerequisites

To achieve the usecase, you need to install the following software:

- IBM Portlet Container Tools for WebSphere Application Server Liberty Profile, it would be installe don top of IBM WebSphere Developer Tools, you can get the detailed instructions at IBM Collaboration Solutions Catalog.

- WebSphere Application Server V8.5.5 Liberty Profile. At the add-ons dialog you need to select the additional features IBM WAS Liberty Profile Extended Content, Portlet Container & Portlet Serving while installing the profile. The Portlet Container serves as the runtime environment for portlet applications and Portlet Serving is required by the toolkit for development scenarios.
- IBM WebSphere Portal v 8.0.
- Download the WSRP Producer for WebSphere Application Server Liberty profile from the IBM Collaboration Solutions Catalog.

You can prepare the development environment by following the instructions provided in the readme file for the IBM Portlet Container Tools for WAS Liberty profile. To install IBM WebSphere Portal you can follow the instructions at the Portal infocenter (link provided in resources section).

Developing the sample portlet application

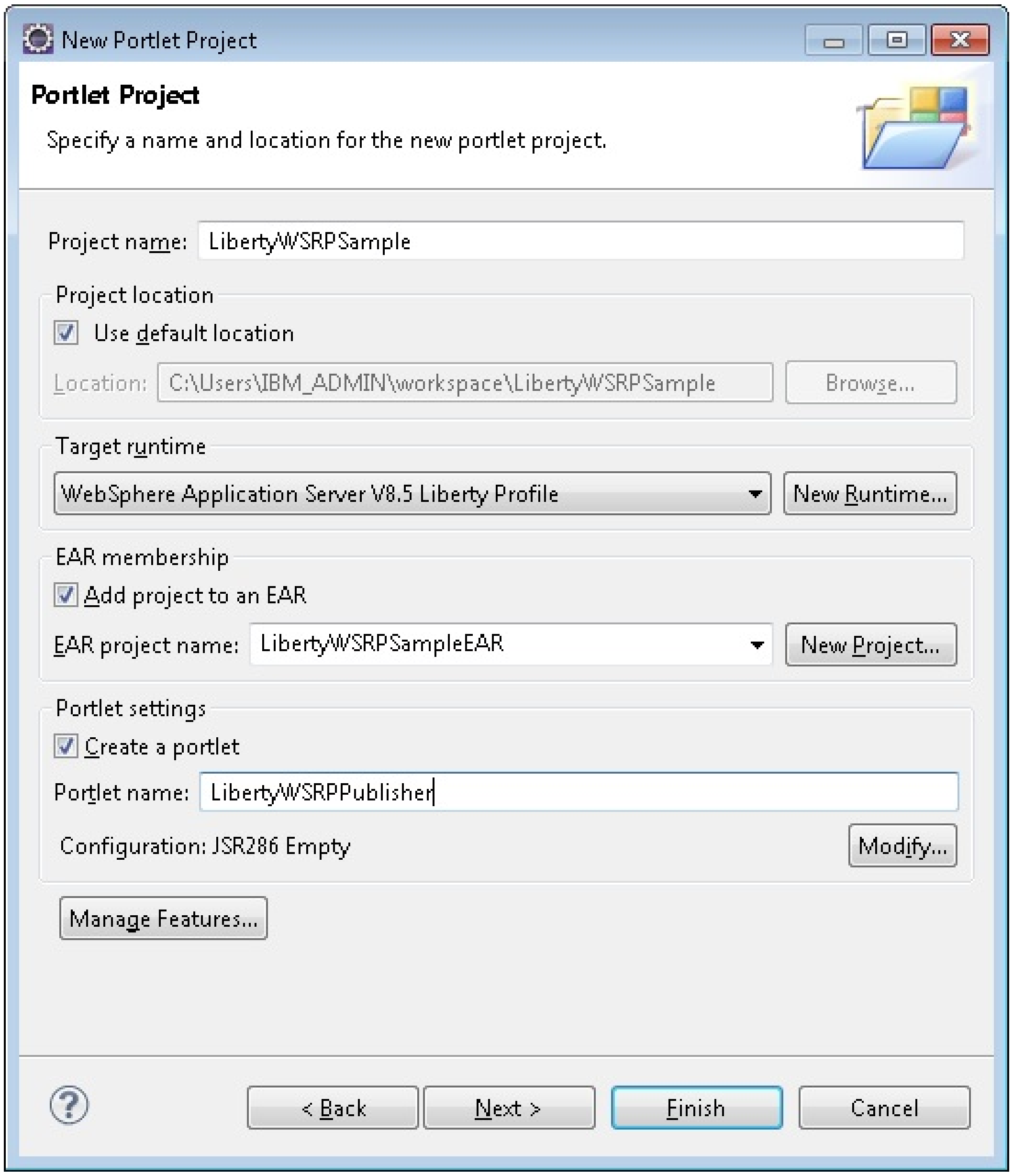
Creating the Portlet project and publisher portlet

You would create a portlet project with two portlet, use LibertyWSRPSample as the portlet project name, LibertyWSRPPublisher as the first portlet name and LibertyWSRPSubscriber as the name for the second portlet. To create the portlet project

Select File->New->Portlet Project. The Project creation dialog as shown in fig.1 would launch

- Enter LibertyWSRPSample as the portlet project name.
- Select WebSphere Application Server v8.5 Liberty Profile as your target runtime.
- Select the Create a portlet checkbox.
- Enter the Portlet name as LibertyWSRPPublisher.
- Click the modify button of the Configuration label, and select JSR286 as portlet API and Empty portlet as Portlet type.
- Click Finish.

Figure 1: New portlet project wizard


New Portlet Project

Portlet Project

Specify a name and location for the new portlet project.

Project name: LibertyWSRPSample

Project location

[✓] Use default location

Location: C:\Users\IBM_ADMIN\workspace\LibertyWSRPSample

Target runtime: WebSphere Application Server V8.5 Liberty Profile

EAR membership

[✓] Add project to an EAR

EAR project name: LibertyWSRPSampleEAR

Portlet settings

[✓] Create a portlet

Portlet name: LibertyWSRPPublisher

Configuration: JSR286 Empty

The portlet project containing the LibertyWSRPPublisher portlet would be created in the workspace. The class LibertyWSRPPublisherPortlet would be opened by default in the editor.

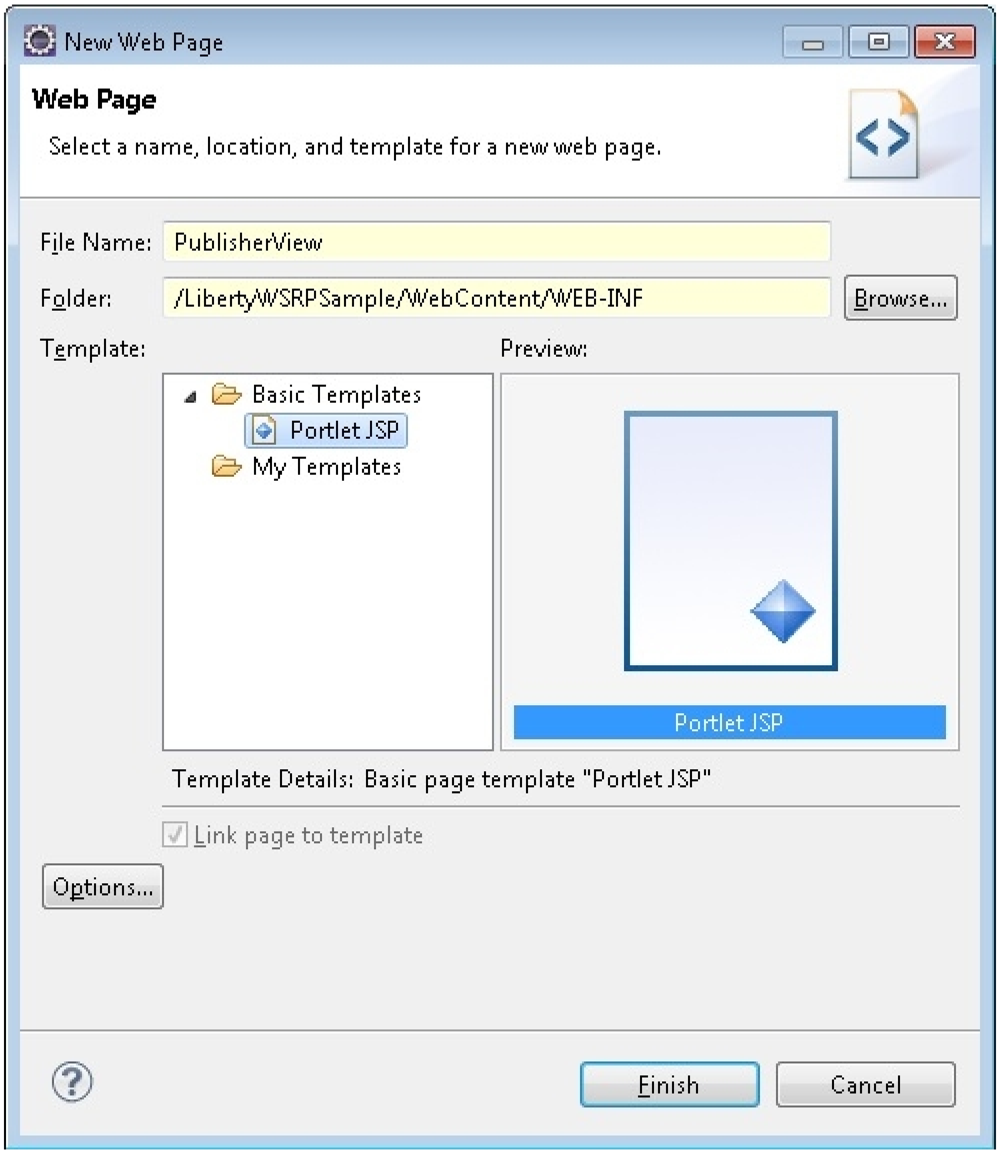
View creation for the publisher portlet

The empty portlet type doesn’t have a portlet JSP created by default so you need to create a JSP view for the portlet created above. To create a JSP file,

- In the Project Explorer, right-click on the WebContent folder of your portlet project.
- Select **New > Web Page**.
- The Web Page wizard will open. Enter the JSP name as PublisherView
- Select the Portlet JSP as a Template

- Press Finish.

Figure 2: New web page for the LibertyWSRPPublisher portlet


New Web Page

Web Page

Select a name, location, and template for a new web page.

File Name: PublisherView

Folder: /LibertyWSRPSample/WebContent/WEB-INF

Template: Basic Templates > Portlet JSP; My Templates

Preview: Portlet JSP

Template Details: Basic page template "Portlet JSP"

Link page to template

A new web page PublisherView.jsp would be created in the WebContent folder of your project and would be opened in your workspace.

Updating the Portlet Class and portlet configuration file

You need to add the desired logic to the portlet to make it work as per desired usecase. There are few changes that you need to take care of like referencing the portlet JSP in the portlet class file, event definition in portlet configuration (i.e. portlet.xml) and then add the corresponding login in processAction() method of portlet class file.

Reference of View JSP in the portlet class

Open the portlet class file LibertyWSRPPublisherPortlet under Java Resources, In its doView() method, uncomment the following lines of code and make sure you replace the parameter provided here with the value of the JSP file.

```
PortletRequestDispatcher rd =
getPortletContext().getRequestDispatcher("/PublisherView.jsp");
            rd.include(request,response);
```

The updated method would look like the snippet below.

Code Snippet 1: doView() method in LibertyWSRPPublisherPortlet.java

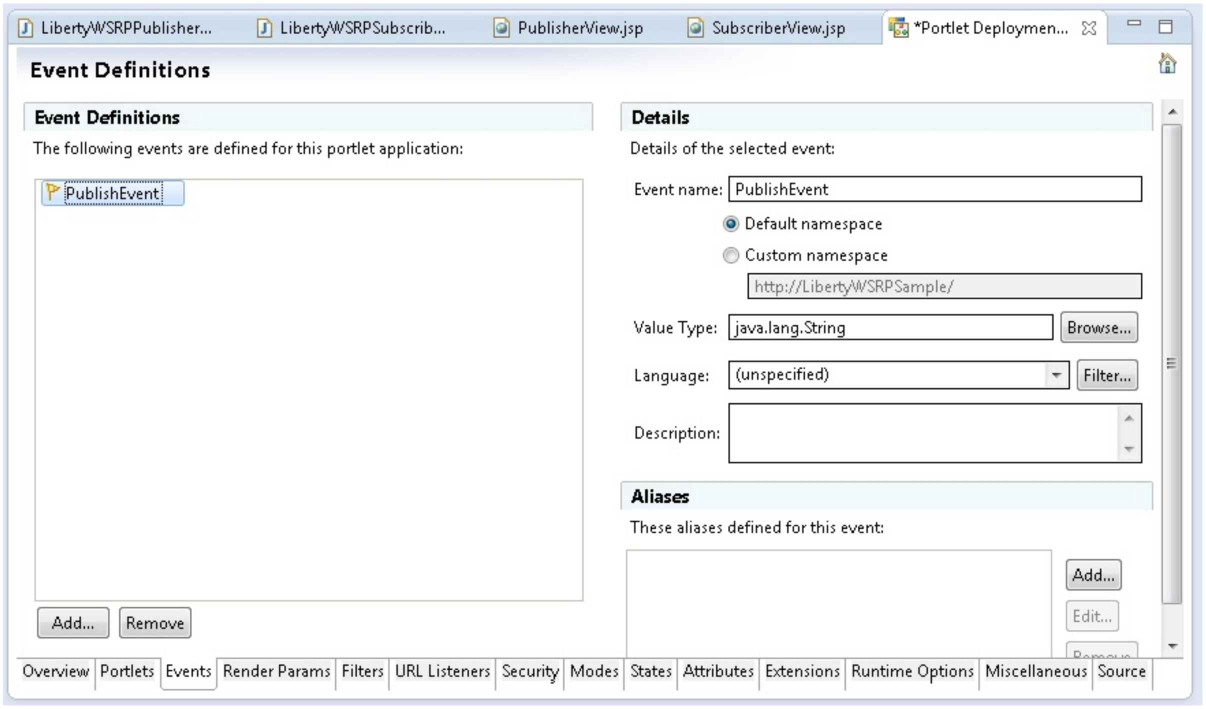
```
public void doView(RenderRequest request, RenderResponse response) throws
PortletException, IOException {
            // Set the MIME type for the render response
            response.setContentType(request.getResponseContentType());

            //
            // TODO: auto-generated method stub for demonstration purposes
            //

            // Invoke the JSP to render, replace with the actual jsp name
            PortletRequestDispatcher rd =
getPortletContext().getRequestDispatcher("/PublisherView.jsp");
            rd.include(request,response);

            // or write to the response directly
            //response.getWriter().println("LibertyWSRPPublisher#doView()");

      }
```

Event definition for publisher portlet

Now you would define the JSR event that you intend to send across the portlet. To achieve that,

- Select the node Portlet deployment descriptor (or portlet.xml) in project explorer.
- Open it.
- It would open the portlet.xml with the visual editor.
- Select the Event tab from the tabs listed at bottom of editor.
- Press the Add button to create a new event. See Figure 3

Figure 3: Events tab in Portlet deployment descriptor.



Event Definitions

Event Definitions

The following events are defined for this portlet application:

PublishEvent

Details

Details of the selected event:

Event name: PublishEvent

Default namespace [✓]

Custom namespace

http://LibertyWSRPSample/

Value Type: java.lang.String

Language: (unspecified)

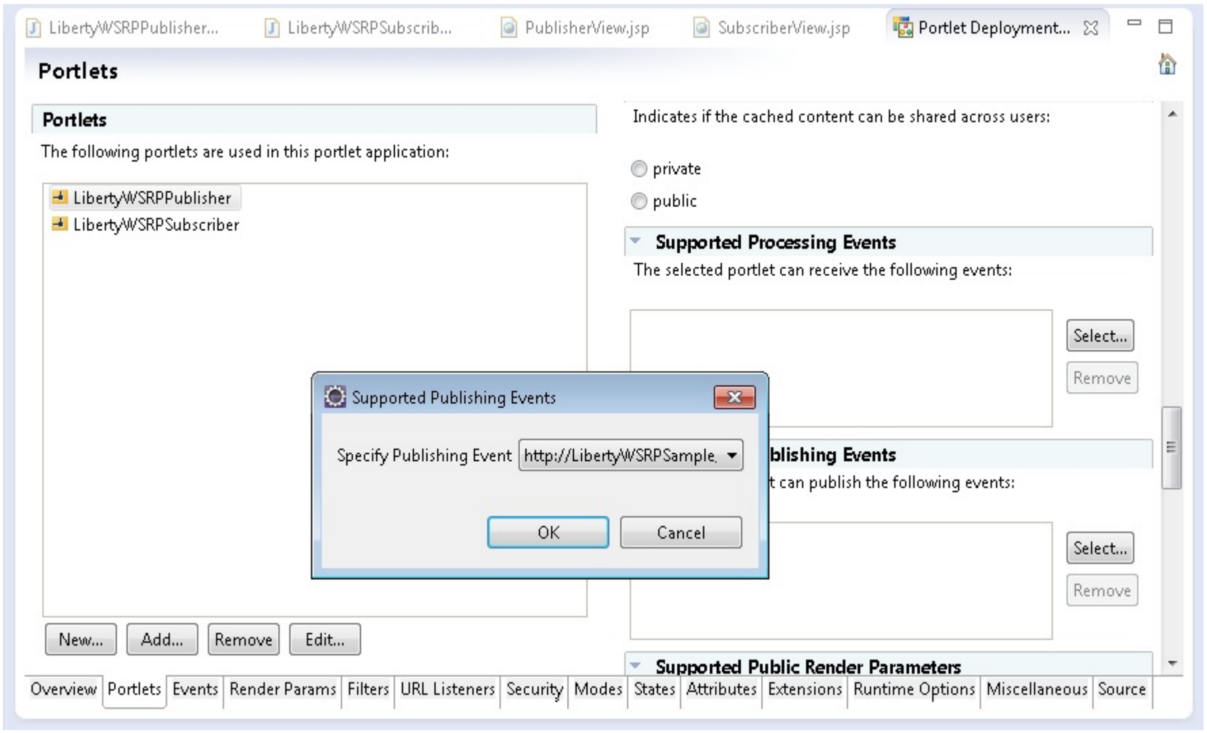
Description:

Aliases

These aliases defined for this event:

- In the Details section enter the name as PublishEvent.
- Enter the Value Type as java.lang.string
- Save the changes.
- Select the Portlet tab to display the portlet(s) available in the project.
- Select the portlet LibertyWSRPPublisher
- Scroll down to the section Supported Publishing Events.
- Press the Select button
- In the dialog with dropdown listing the defined events, select the PublishEvent. See Figure 4

Figure 4: Supported Publishing Events in portlet tab



- Press OK
- Save the changes

In above steps you have first defined an event with name PublishEvent to the portlet project and then you have added the PublishEvent as a publish event for the portlet LibertyWSRPPublisher. Move to the Source tab to ensure the portlet.xml has the desired configuration changes.
Portlet.xml should have event definition tags added to it, outside the portlet tags in the file.

```
<event-definition>
      <name>PublishEvent</name>
      <value-type>java.lang.String</value-type>
</event-definition>
```

The portlet tag should have supported-publishing-event tag added to it with the event name.

```
<supported-publishing-event>
            <name>PublishEvent</name>
</supported-publishing-event>
```

Updating the processAction() for event handling

To publish the event across other applications, you need to update the processAction() method in the portlet class file.

Declare the following parameter to the portlet class

```
public static final String PUBLISHEVENT_PARAM = "PUBLISHEVENT_PARAM";
```

Add the following snippet to processAction() method.

Code Snippet 2: Set event in processAction() method of publisher portlet

```
if (request.getParameter(PUBLISHEVENT_PARAM) != null) {
                  response.setEvent("PublishEvent",
                          request.getParameter(PUBLISHEVENT_PARAM));
              }
```

The updated method should look like the following snippet
Code Snippet 3: processAction() method in LibertyWSRPPublisherPortlet.java

```
public void processAction(ActionRequest request, ActionResponse response) throws
PortletException, java.io.IOException {
            // TODO: auto-generated method stub
             if (request.getParameter(PUBLISHEVENT_PARAM) != null) {
                  response.setEvent("PublishEvent",
                          request.getParameter(PUBLISHEVENT_PARAM));
              }
      }
```

Designing the portlet JSP for publisher portlet

Finally you would design the portlet JSP as the use case. For the usecase you would have an input text and a submit button to receive the user input and send it across. Replace the JSP content by the snippet shared below.

Code Snippet 4: code snippet for PublisherView.jsp

```
<%@page language="java"
      contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"
session="false"%>
<%@taglib uri="http://java.sun.com/portlet_2_0"
prefix="portlet"%><portlet:defineObjects />
<form method="POST" action="<portlet:actionURL/>">
      <table border="1">
            <tr bgcolor="silver">
                  <th align="center" valign="middle" colspan ="2" >Publisher
Portlet for the
                        WSRP use case <br> <br></th>
            </tr>
            <tr>
                  <td>Enter your text message here.</td>
                  <td><input name="PUBLISHEVENT_PARAM" value="" type="text"
size="50"></td>
            <tr>
                  <td colspan ="2" align ="center"><input name="submit"
value="Pass It On" size ="30" type="submit"></td>
            </tr>
      </table>

</form>
```

At this stage, your first portlet LibertyWSRPPublisher is ready for deployment. You can test it by publishing the portlet to the server. To test the Portlet, you need to

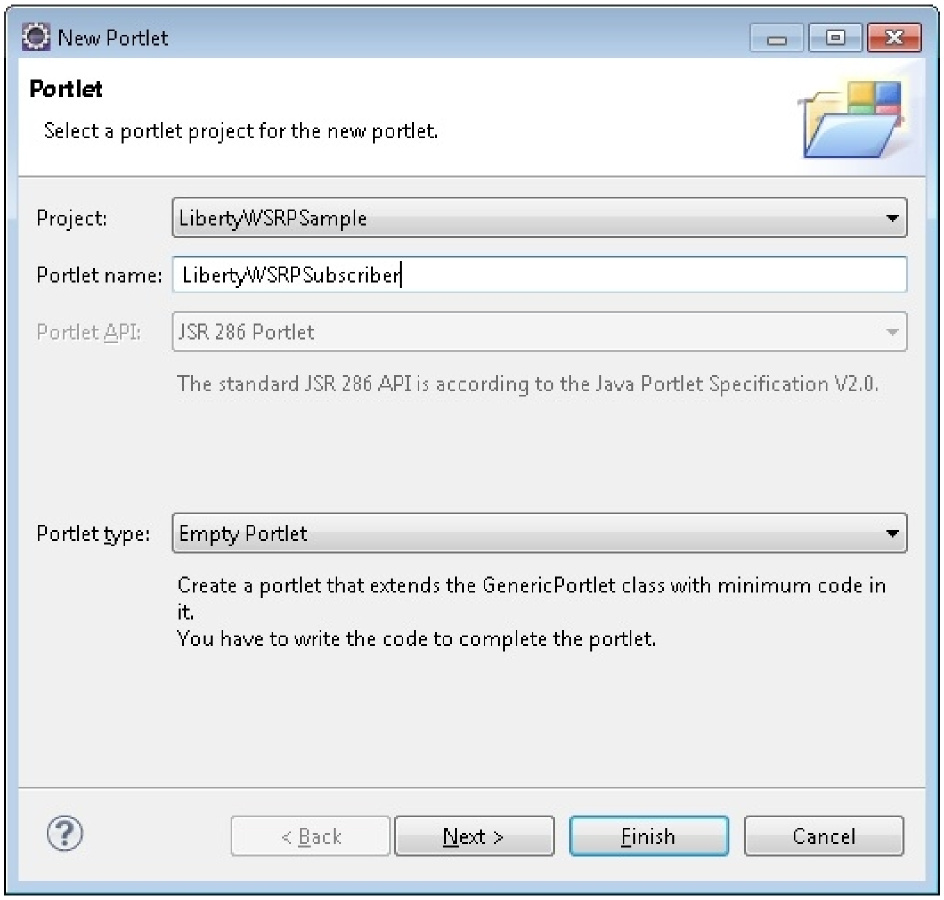
- Switch to the Server view
- Create a server instance for WebSphere Application Server Liberty profile
- Use Run On Server option to publish the portlet project (described in section Publishing the portlet to Server).

Creating the subscriber portlet

To create the second portlet i.e. LibertyWSRPSubscriber porlet, Select the portlet project and then select File->New->Portlet. The Portlet creation dialog would launch

- Enter the Portlet name as LibertyWSRPSubscriber.
- Select Finish

Figure 5: New portlet wizard for the LibertyWSRPSubscriber



Portlet

Select a portlet project for the new portlet.

Project: LibertyWSRPSample

Portlet name: LibertyWSRPSubscriber

Portlet API: JSR 286 Portlet

The standard JSR 286 API is according to the Java Portlet Specification V2.0.

Portlet type: Empty Portlet

Create a portlet that extends the GenericPortlet class with minimum code in it.
You have to write the code to complete the portlet.

The portlet project LibertyWSRPSample would now contain two portlets, LibertyWSRPPublisher portlet and LibertyWSRPSubscriber portlet.

View creation for the subscriber portlet

Use the steps above to create the JSP for the LibertyWSRPSubscriber portlet with the name as SubscriberView.

Updating the Portlet Class and portlet configuration file

Similar to the above portlet, you need to add the desired logic to the portlet to enable it to process the received event. There are few changes that you need to take care of like referencing the portlet JSP in the portlet class file, event definition in portlet configuration (i.e. portlet.xml) and then add the corresponding login in processEvent() method of portlet class file.

Reference of View JSP in the portlet class

To render and invoke the correct portlet view JSP

- Open the portlet class LibertyWSRPsubscriberPortlet under Java Resources.
- In its doView() method, uncomment the following lines of code

```
PortletRequestDispatcher rd =
getPortletContext().getRequestDispatcher("/subscriberView.jsp");
rd.include(request,response);
```

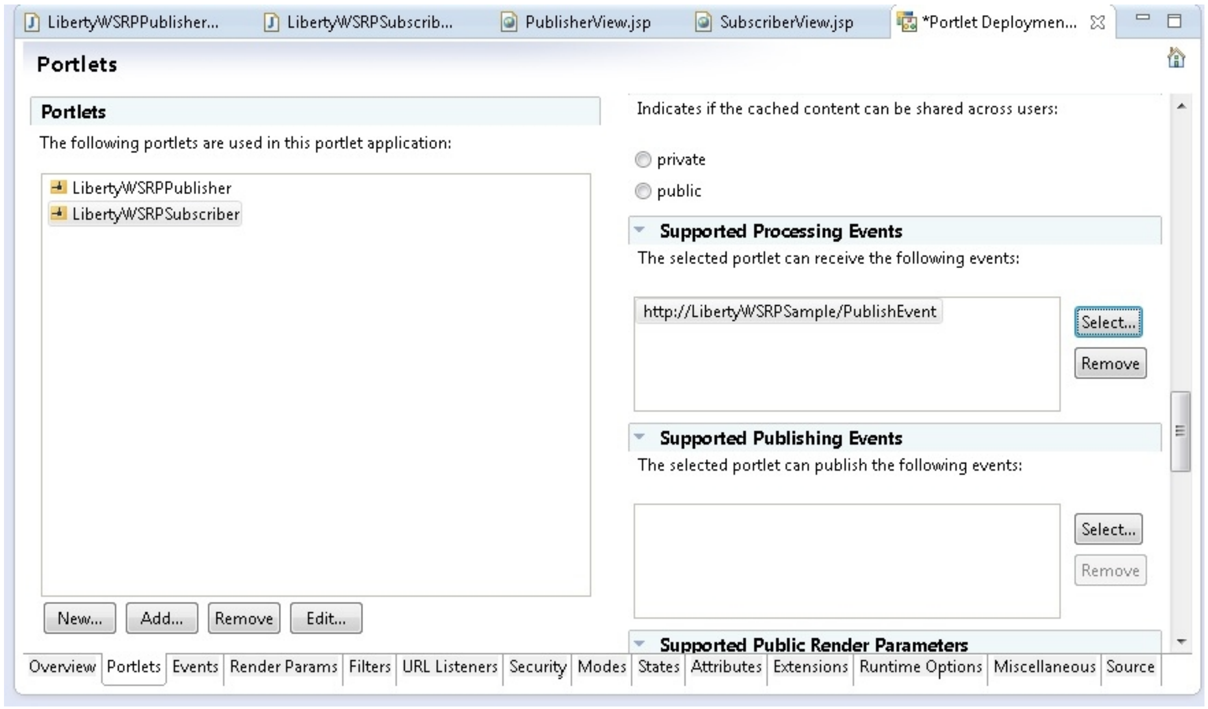
- Replace the parameter provided in auto generated code with the correct name of the JSP file.

Event definition for subscriber portlet

Now you would define the JSR event that you intend to send across the portlet. To achieve that,

- Select the node Portlet deployment descriptor (or portlet.xml) in project explorer.
- Open it.
- It would open the portlet.xml with the visual editor.
- Select the portlet tab to display the portlet(s) available in the project.
- Select the portlet LibertyWSRPSubscriber
- Scroll down to the section Supported Processing Events.
- Press the Select button
- Select the PublishEvent from the list of defined events. See Figure 6

Figure 6: Supported Processing Events in portlet tab



- Press OK
- Save the changes

The above steps would add the the PublishEvent as a processed event for the portlet LibertyWSRPSubscriber. Move to the Source tab to ensure the portlet.xml has the desired configuration changes.
Portlet.xml should have event definition tags added to it, outside the portlet tags in the file.

```
<event-definition>
      <name>PublishEvent</name>
      <value-type>java.lang.String</value-type>
</event-definition>
```

The portlet tag for LibertyWSRPSubscriber should have supported-processing-event tag added to it with the event name.

```
<supported-processing-event>
      <name>PublishEvent</name>
</supported-processing-event>
```

Event handling for subscriber portlet

To process the event, you need to add a method the processEvent()in the portlet class file for the subscriber portlet.

- Declare the following parameter to the portlet class

```
public static final String PUBLISHEVENT_PARAM = "PUBLISHEVENT_PARAM";
private String eventValue = "";
```

- Add the following processEvent() method to the portlet class

Code Snippet 5: processEvent() method in LibertyWSRPSubscriberPortlet.java

```
public void processEvent(EventRequest request, EventResponse response)
                  throws PortletException, java.io.IOException {
              Event sampleEvent = request.getEvent();
              if (sampleEvent.getName().toString().equals("PublishEvent")) {
                  Object sampleProcessObject = sampleEvent.getValue();
                  eventValue = sampleProcessObject.toString();
              }
        }
```

- Set the parameter PUBLISHEVENT_PARAM to the event value by adding the following statement to the doView() method.

```
request.setAttribute(PUBLISHEVENT_PARAM, eventValue);
```

The updated doView() method would look like the snippet below:

Code Snippet 6: doView() method in LibertyWSRPSubscriberPortlet.java

```
public void doView(RenderRequest request, RenderResponse response) throws
PortletException, IOException {
            // Set the MIME type for the render response
            response.setContentType(request.getResponseContentType());
            //
            // TODO: auto-generated method stub for demonstration purposes
            //
             request.setAttribute(PUBLISHEVENT_PARAM, eventValue);
            // Invoke the JSP to render, replace with the actual jsp name
            PortletRequestDispatcher rd =
getPortletContext().getRequestDispatcher("/SubscriberView.jsp");
            rd.include(request,response);
            // or write to the response directly
      //response.getWriter().println("LibertyWSRPSubscriber#doView()");
      }
```

Designing the portlet JSP for subscriber portlet

Finally you would design the portlet JSP for the subscriber JSP to receive and process the event. Replace the JSP content by the snippet shared below.
Code Snippet 7: code snippet for SubscriberView.jsp

```
<%@page language="java"
      contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"
session="false"%>
<%@taglib uri="http://java.sun.com/portlet_2_0"
prefix="portlet"%><portlet:defineObjects />

<table border="1">
            <tr bgcolor="silver">
                  <th align="center" valign="middle">Message Received: <br></th>
            </tr>
            <tr>
      <td><%=request.getAttribute("PUBLISHEVENT_PARAM")%><br></td>
```

```
            </tr>
</table>
```

Deploying the portlet application

Configuring WSRP producer for WAS liberty profile and enabling the add-on features

Next you need to configure the WSRP producer for WAS Liberty profile to enable All WSRP compliant Consumer portals to consume and integrate portlets that are provided by the WSRP Producer. To achieve the same

- Copy the "was.wsrp.producer.ear" in dropins folder, the sample path is <Liberty_Home>\usr\servers\defaultServer\dropins, where <Liberty_Home> denotes the root installation folder.
- Add the following entry in the server.xml (located at <Liberty_Home>\usr\servers\defaultServer)
  <webContainer deferServletLoad="false"/>
- To enable the add-on features recommended in pre-requisite section, add the following entries to server.xml of your WAS liberty profile.

```
<featureManager>
      <feature>jaxws-2.2</feature>
      <feature>usr:portlet-2.0</feature>
      <feature>usr:portletserving-2.0</feature>
</featureManager>
```

Refer the following code snippet of the updated server.xml file for the profile.

Code Snippet 8: Server.xml for the WAS Liberty profile

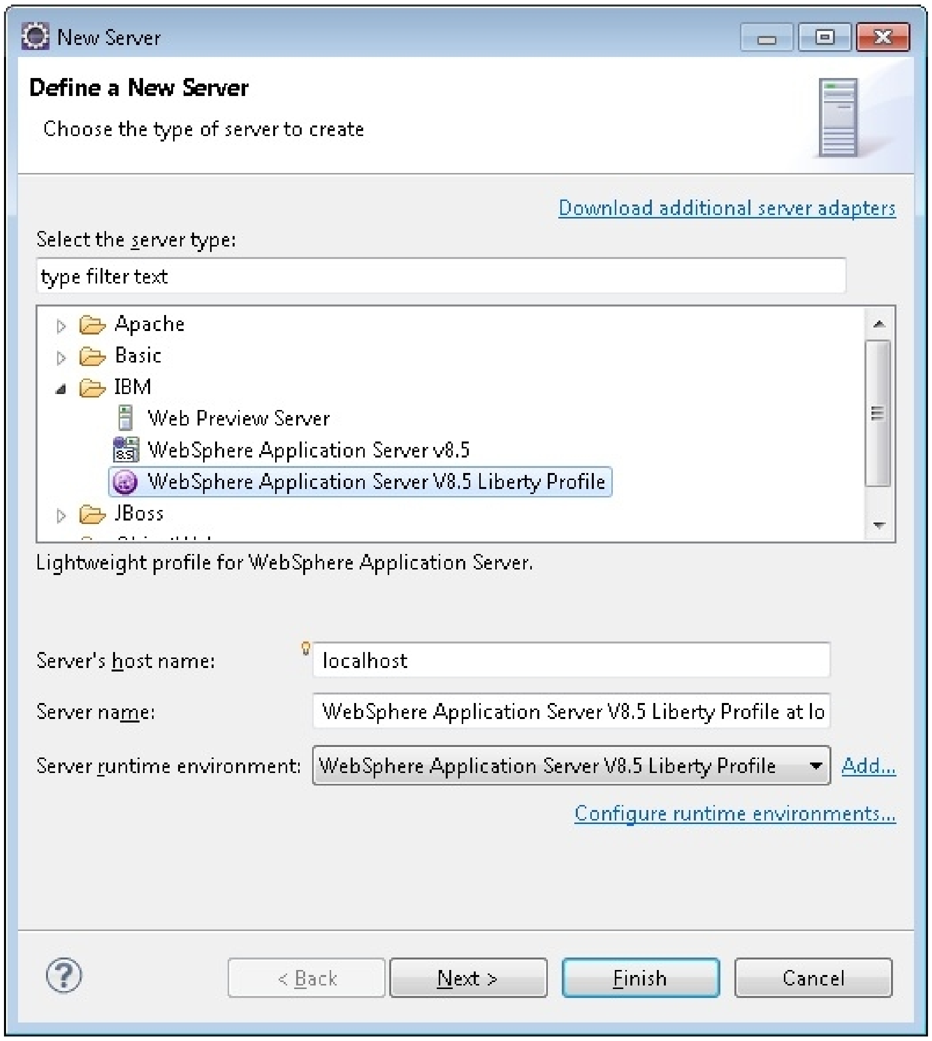
```
<server description="new server">
    <!-- Enable features -->
   <featureManager>
        <feature>jsp-2.2</feature>
        <feature>jaxws-2.2</feature>
        <feature>usr:portlet-2.0</feature>
        <feature>usr:portletserving-2.0</feature>
        <feature>localConnector-1.0</feature>
    </featureManager>
      <webContainer deferServletLoad="false"/>
    <httpEndpoint host="localhost" httpPort="9080" httpsPort="9443"
id="defaultHttpEndpoint"/>
    <applicationMonitor updateTrigger="mbean"/>
</server>
```

Publishing the portlet to Server

To publish a project to the server, first you need to define a desired runtime for it, to define a runtime,

- Select **Windows > Show View > Servers.**
- Right-click in the **Servers** view and select **New Server.**
- Choose Server Type **IBM > WebSphere Application Server v8.5 Liberty Profile** from the list. See **Figure 7**.

Figure 7: New Server Wizard



New Server

Define a New Server

Choose the type of server to create

Download additional server adapters

Select the server type:

type filter text

- Apache
- Basic
- IBM
  - Web Preview Server
  - WebSphere Application Server v8.5
  - WebSphere Application Server V8.5 Liberty Profile
- JBoss

Lightweight profile for WebSphere Application Server.

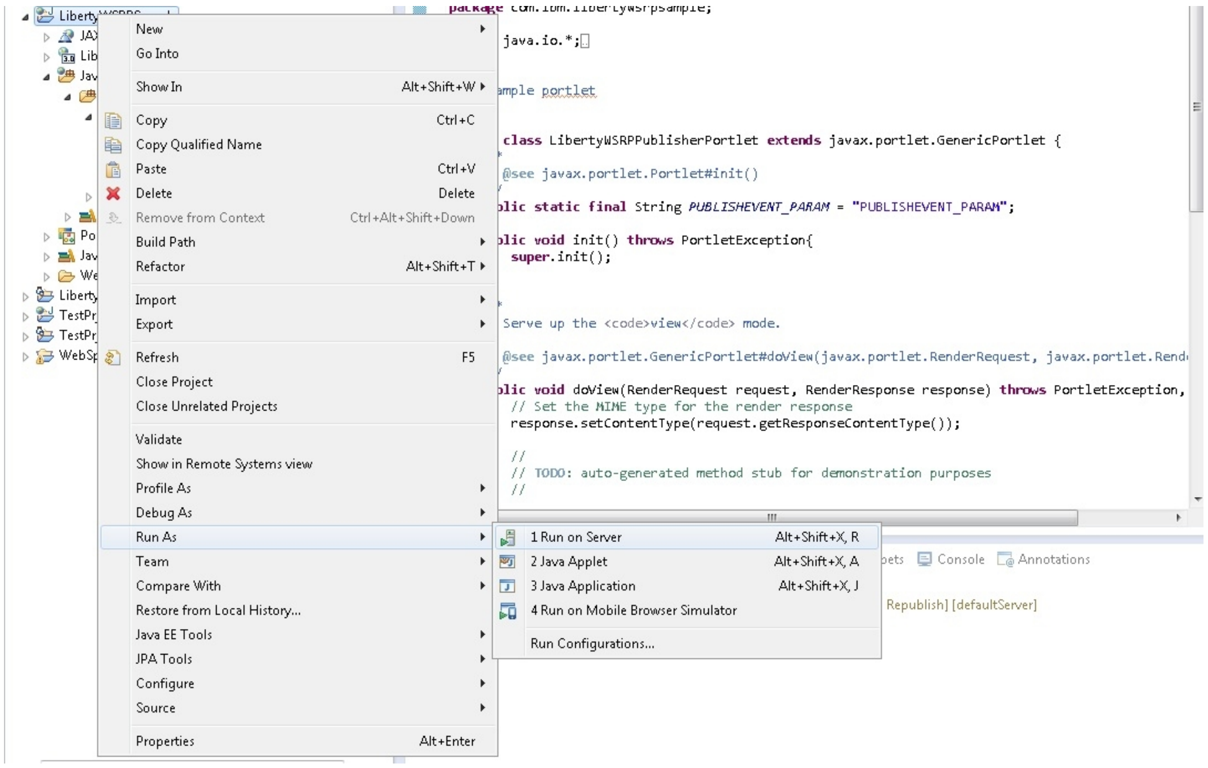
Server's host name: localhost

Server name: WebSphere Application Server V8.5 Liberty Profile at lo

Server runtime environment: WebSphere Application Server V8.5 Liberty Profile

- Select **Next**.
- Provide a path to an installed Liberty Profile runtime environment.
- Select **Finish.**
- Right-click on project and select **Run as >Run on Server.**
- Select **WebSphere Application Server v8.5 Liberty Profile** and click **Finish**. See Figure 8.

Figure 8: Run On Server option



```
package com.ibm.libertywsrpsample;

 java.io.*;

ample portlet

 class LibertyWSRPPublisherPortlet extends javax.portlet.GenericPortlet {

 @see javax.portlet.Portlet#init()

blic static final String PUBLISHEVENT_PARAM = "PUBLISHEVENT_PARAM";

blic void init() throws PortletException{
  super.init();

 Serve up the <code>view</code> mode.

 @see javax.portlet.GenericPortlet#doView(javax.portlet.RenderRequest, javax.portlet.Rend

blic void doView(RenderRequest request, RenderResponse response) throws PortletException,
  // Set the MIME type for the render response
  response.setContentType(request.getResponseContentType());

  //
  // TODO: auto-generated method stub for demonstration purposes
  //
```

Configurations on WebSphere portal

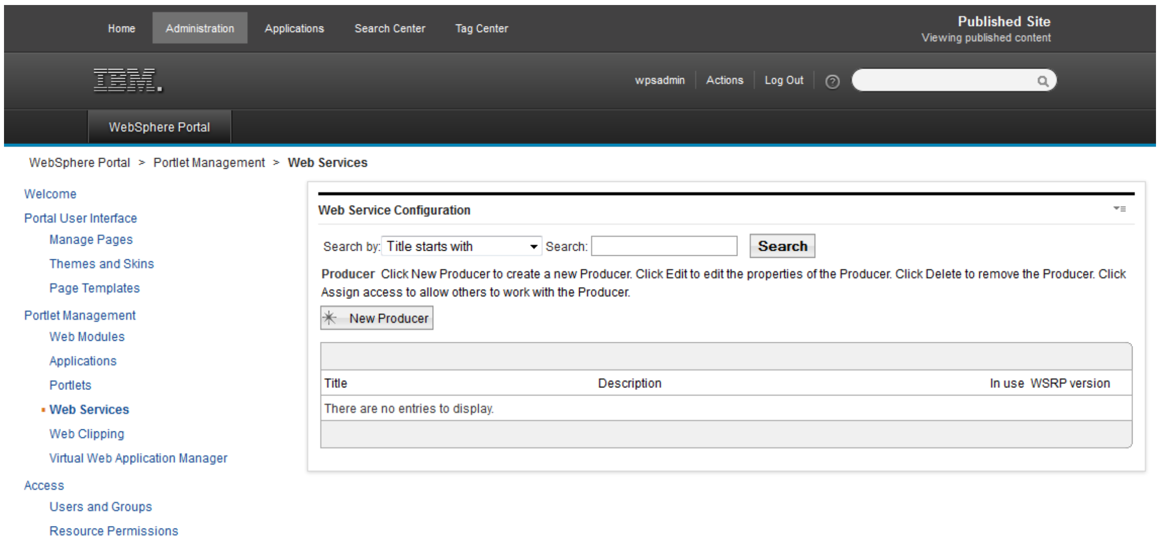
Adding IBM WebSphere Application Server Liberty Profile as Web Service Producer

As mentioned earlier, you have a WSRP Producer already installed on your IBM WebSphere Application Server Liberty Profile, so when you publish your portlets to it, they are available to be consumed remotely. In this section we will see how it can be achieved.

To add a WSRP producer:

- Login to the portal home page, and browse to the administration console. The standard login url is https://<hostname>:<port>/wps/portal
- Navigate to **Administration** > **Web Services**.
- Select the **New Producer.** See Figure 9.

Figure 9: Web Services Configuration

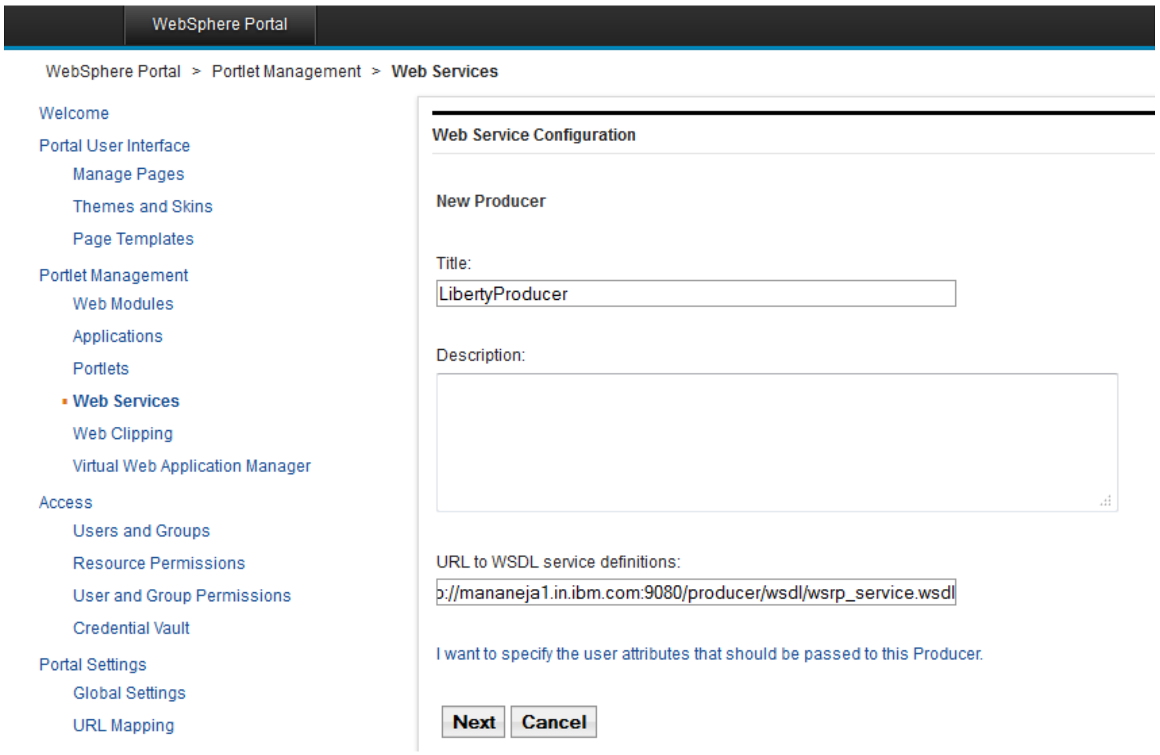


Producer Click New Producer to create a new Producer. Click Edit to edit the properties of the Producer. Click Delete to remove the Producer. Click Assign access to allow others to work with the Producer.

Title	Description	In use	WSRP version
There are no entries to display.			

- Enter the title and location of wsdl on IBM WebSphere Application Server Liberty Profile to add the server as a web service producer and click Next. See Figure 10.

Figure 10: Web Service Producer Configuration



Title: LibertyProducer

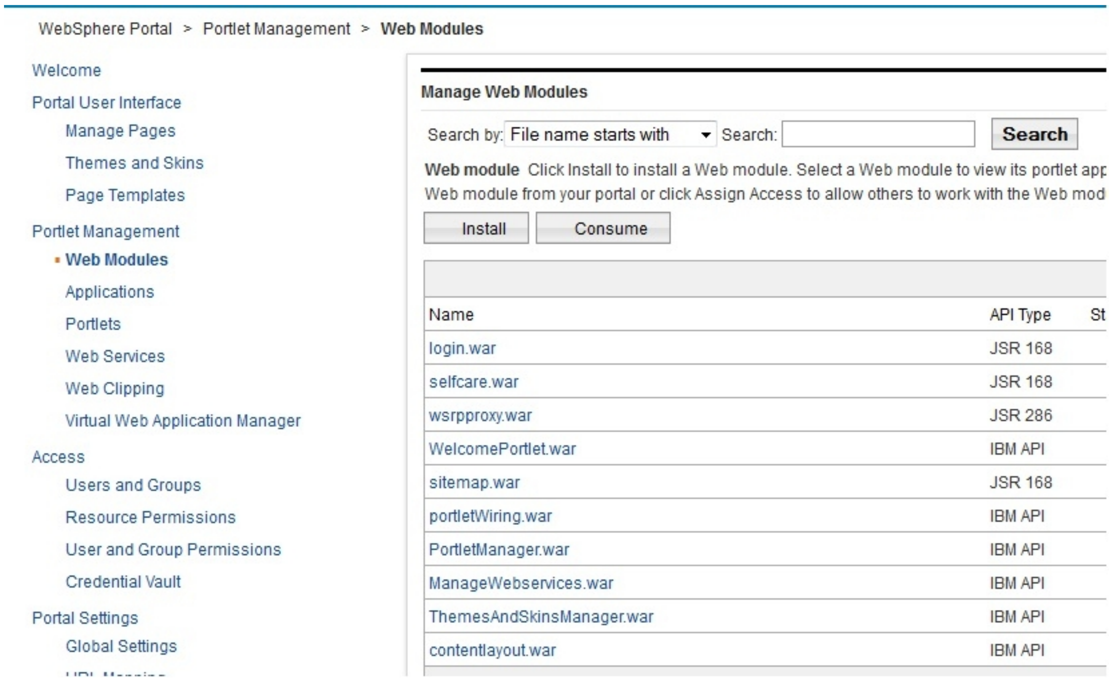
Description:

URL to WSDL service definitions: ɔ://mananeja1.in.ibm.com:9080/producer/wsdl/wsrp_service.wsdl

Consuming the remote portlet application

- The remote portlet application is to be consumed into IBM WebSphere Portal v8.0.
- Navigate to the **Web Modules**, and select **Consume**. See Figure 11.

Figure 11: Consuming web modules


WebSphere Portal > Portlet Management > **Web Modules**

Manage Web Modules

Search by: File name starts with Search: [Search]

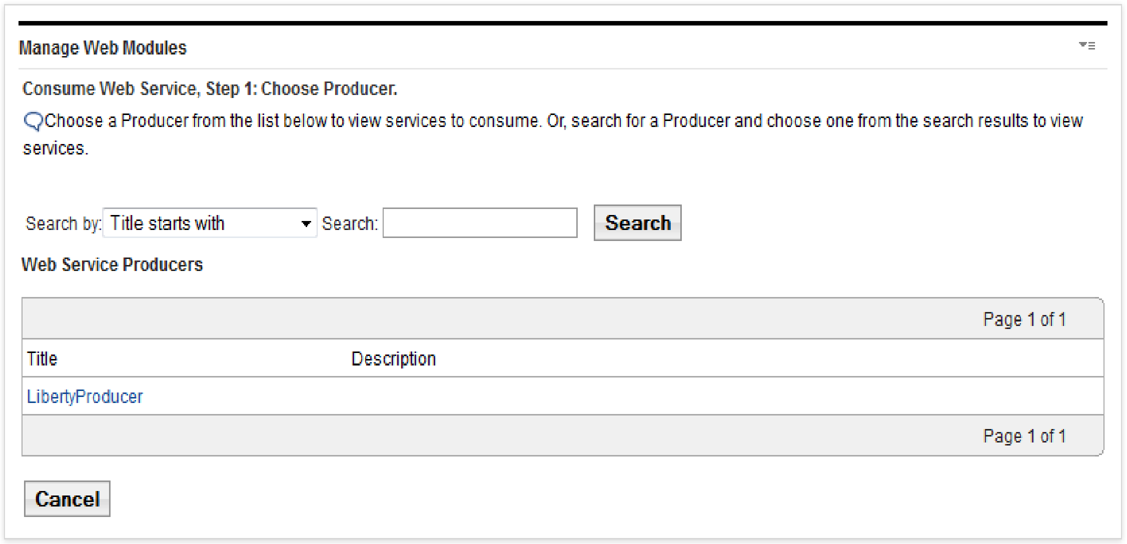
Web module Click Install to install a Web module. Select a Web module to view its portlet app
Web module from your portal or click Assign Access to allow others to work with the Web mod

[Install] [Consume]

Name	API Type	St
login.war	JSR 168	
selfcare.war	JSR 168	
wsrpproxy.war	JSR 286	
WelcomePortlet.war	IBM API	
sitemap.war	JSR 168	
portletWiring.war	IBM API	
PortletManager.war	IBM API	
ManageWebservices.war	IBM API	
ThemesAndSkinsManager.war	IBM API	
contentlayout.war	IBM API	

- Select the web services producer that you have created earlier. See Figure 12.

Figure 12: Manage web module


Manage Web Modules

Consume Web Service, Step 1: Choose Producer.

Choose a Producer from the list below to view services to consume. Or, search for a Producer and choose one from the search results to view services.

Search by: Title starts with Search: [Search]

Web Service Producers

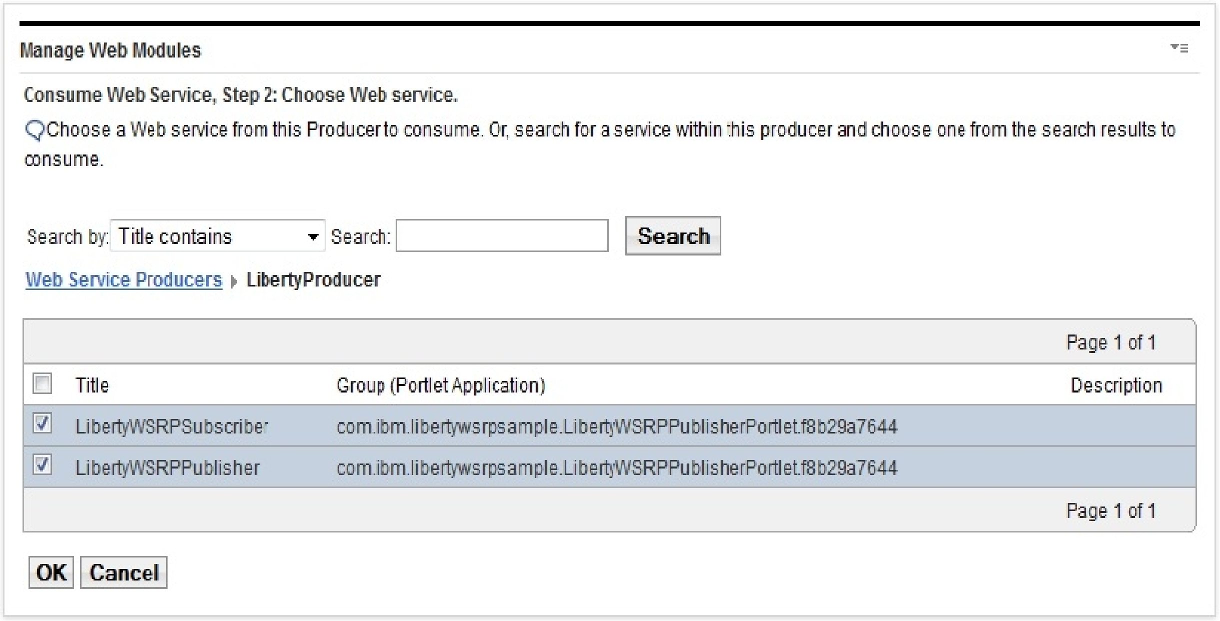
Title	Description
LibertyProducer	

Page 1 of 1

[Cancel]

- All remote portlets that are available for consumption are listed on the page. Select the **LibertyWSRPPublisher and LibertyWSRPsubscriber** portlet and click **Ok**. The remote source portlets are now available for addition to any IBM WebSphere Portal v8.0 page. See Figure 13.

Figure 13: Selecting remote web module


Manage Web Modules

Consume Web Service, Step 2: Choose Web service.

Choose a Web service from this Producer to consume. Or, search for a service within this producer and choose one from the search results to consume.

Search by: Title contains Search:

Web Service Producers > LibertyProducer

Page 1 of 1

	Title	Group (Portlet Application)	Description
✓	LibertyWSRPSubscriber	com.ibm.libertywsrpsample.LibertyWSRPPublisherPortlet.f8b29a7644	
✓	LibertyWSRPPublisher	com.ibm.libertywsrpsample.LibertyWSRPPublisherPortlet.f8b29a7644	

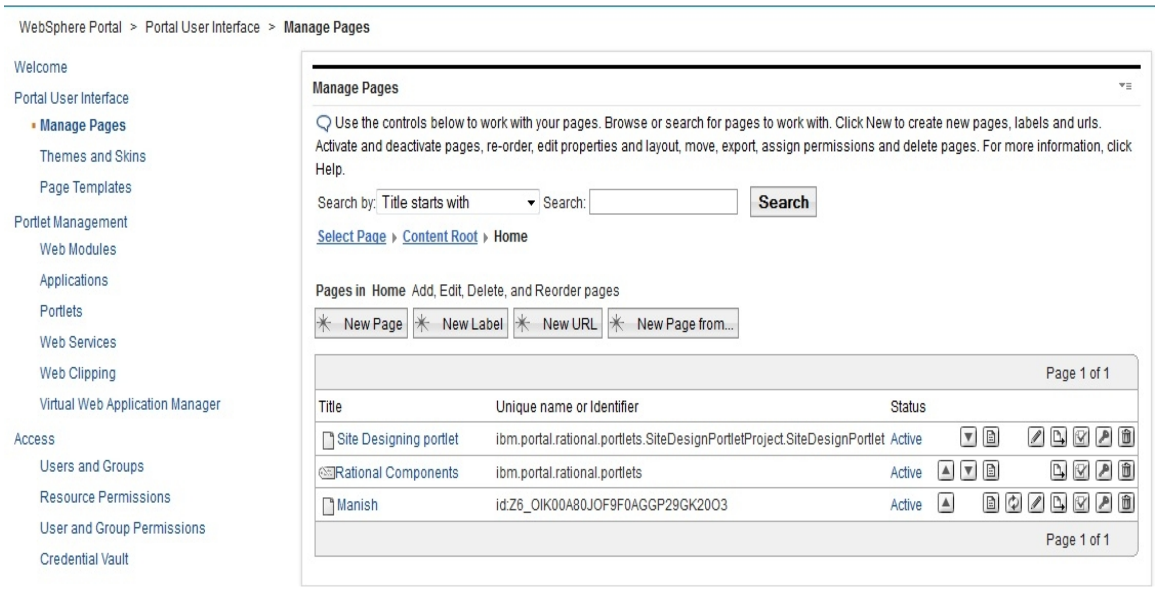
Page 1 of 1

Creating portal page

To create a portal page for hosting the remote portlet:

- Open the IBM WebSphere Portal v8 administration console and select **Administration** > **Manage Pages**.
- Select **Content Root** > **Home** page.
- Select **New Page**. See Figure 14.

Figure 14: New portal page creation


WebSphere Portal > Portal User Interface > **Manage Pages**

Welcome
Portal User Interface
Manage Pages
Themes and Skins
Page Templates
Portlet Management
Web Modules
Applications
Portlets
Web Services
Web Clipping
Virtual Web Application Manager
Access
Users and Groups
Resource Permissions
User and Group Permissions
Credential Vault

Manage Pages

Use the controls below to work with your pages. Browse or search for pages to work with. Click New to create new pages, labels and urls. Activate and deactivate pages, re-order, edit properties and layout, move, export, assign permissions and delete pages. For more information, click Help.

Search by: Title starts with Search: Search

Select Page ▸ Content Root ▸ Home

Pages in Home Add, Edit, Delete, and Reorder pages

New Page | New Label | New URL | New Page from...

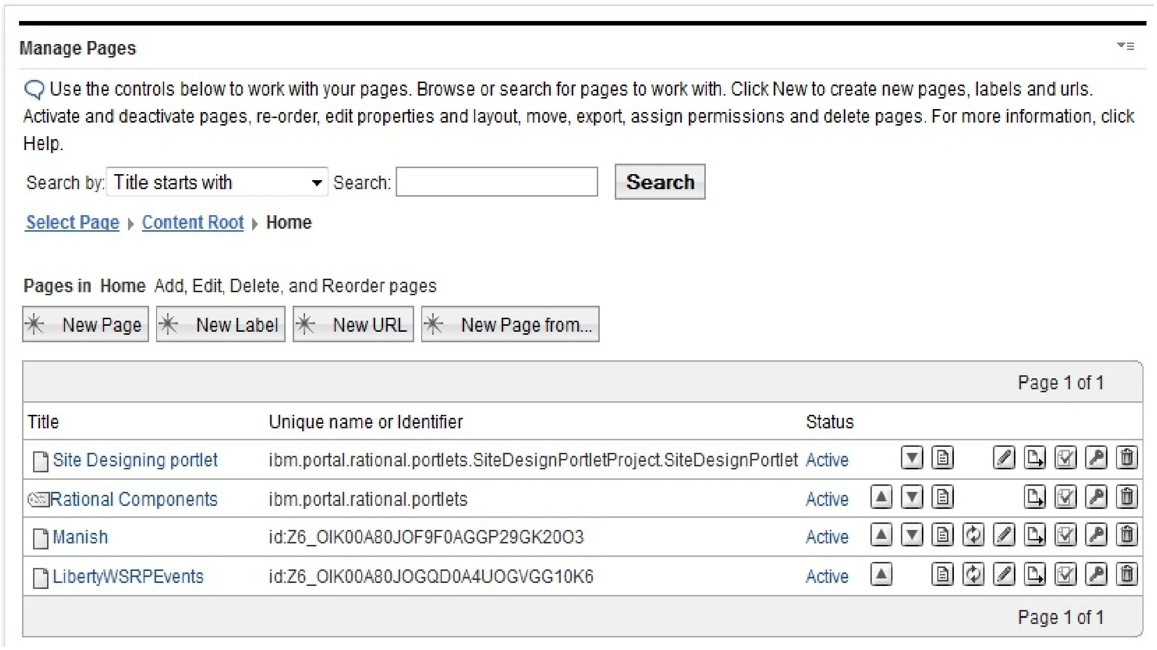
Page 1 of 1

Title	Unique name or Identifier	Status
Site Designing portlet	ibm.portal.rational.portlets.SiteDesignPortletProject.SiteDesignPortlet	Active
Rational Components	ibm.portal.rational.portlets	Active
Manish	id:Z6_OIK00A80JOF9F0AGGP29GK20O3	Active

Page 1 of 1

- Enter LibertyWSRPEvents as page title and click **Ok** to create the portal page.
- The newly created portal page will be added to the existing pages list. See Figure 15.

Figure 15: Manage Pages


Manage Pages

Use the controls below to work with your pages. Browse or search for pages to work with. Click New to create new pages, labels and urls. Activate and deactivate pages, re-order, edit properties and layout, move, export, assign permissions and delete pages. For more information, click Help.

Search by: Title starts with Search: Search

Select Page ▸ Content Root ▸ Home

Pages in Home Add, Edit, Delete, and Reorder pages

New Page | New Label | New URL | New Page from...

Page 1 of 1

Title	Unique name or Identifier	Status
Site Designing portlet	ibm.portal.rational.portlets.SiteDesignPortletProject.SiteDesignPortlet	Active
Rational Components	ibm.portal.rational.portlets	Active
Manish	id:Z6_OIK00A80JOF9F0AGGP29GK20O3	Active
LibertyWSRPEvents	id:Z6_OIK00A80JOGQD0A4UOGVGG10K6	Active

Page 1 of 1

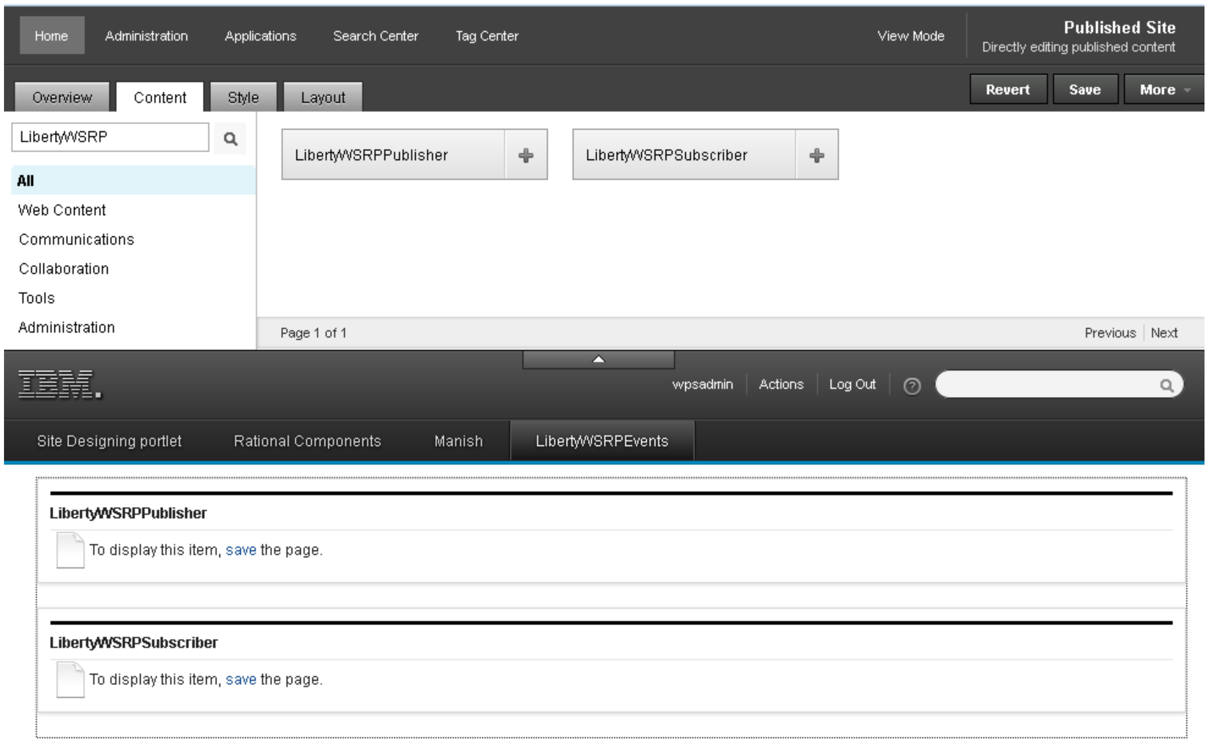
Adding portlets to portal page

To add the portlets to the newly created portal page:

- Open the page created **LibertyWSRPEvents** and switch to **Edit Mode.**

- In the Edit Mode, select the **Content** tab to add the content to the page.
- Search for the subscriber and publisher portlets by using **LibertyWSRP** as the search parameter.
- Add both the portlets to the page. See Figure 16.
- Click **Save** to save the changes made to the page.

Figure 16: Add Content to the Page

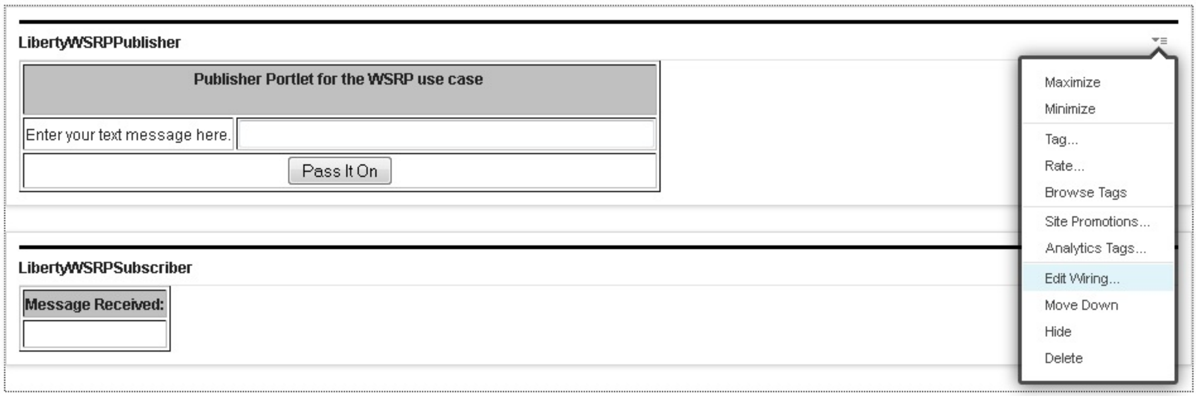


Wiring the portlets

After adding the portlets to the page, wire them to start events brokered by the portal. To wire portlets:

- Select the **Edit Wiring** option in the portlet options menu. See Figure 17.

Figure 17: Creating wires for the portal page


LibertyWSRPPublisher

Publisher Portlet for the WSRP use case

Enter your text message here.

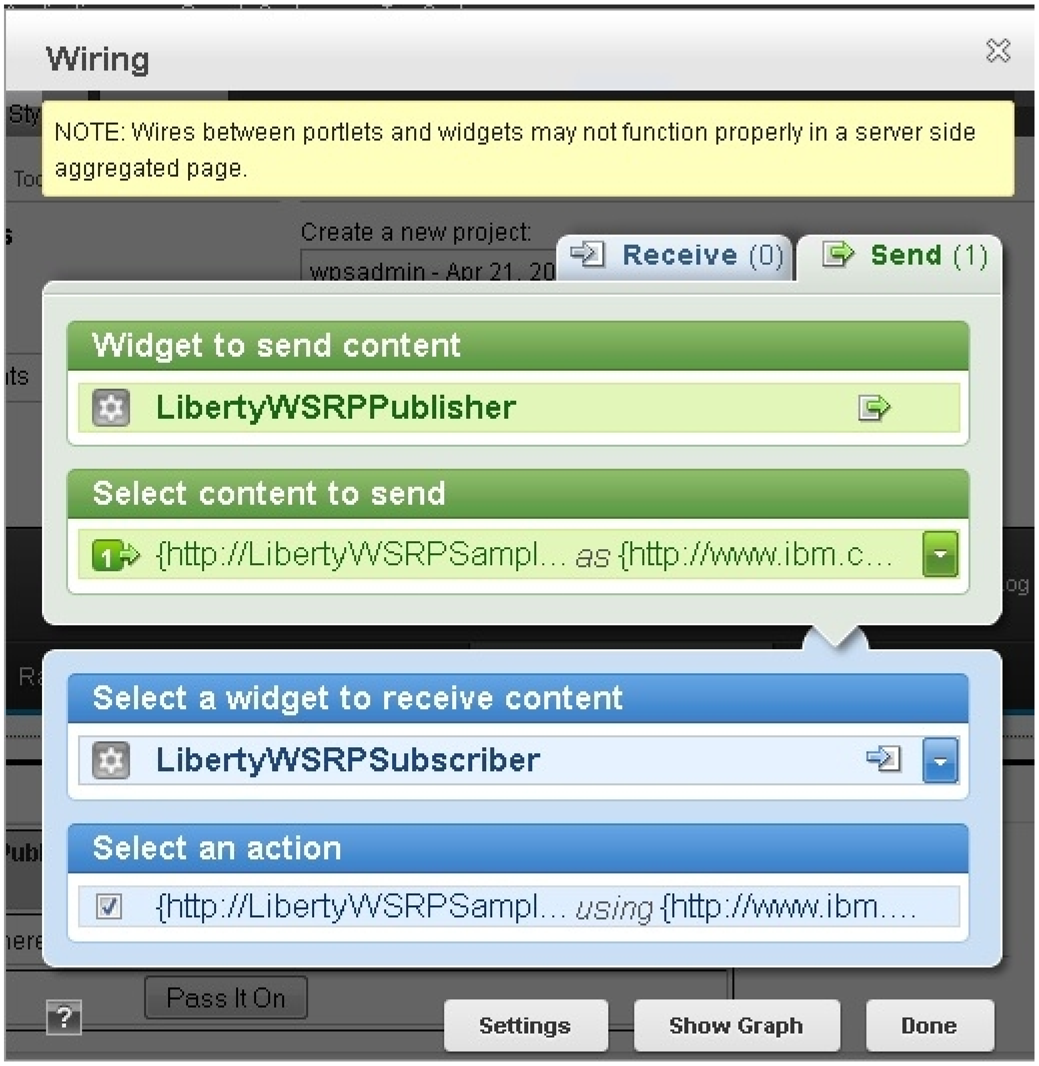
Pass It On

LibertyWSRPSubscriber

Message Received:

- **Wiring** dialog would display (see Figure 18), select the portlet to send and receive the events in the dialog.
- Use the Done button to Save the wiring changes.

Figure 18: Wires created for the portal page


Wiring

NOTE: Wires between portlets and widgets may not function properly in a server side aggregated page.

Receive (0) Send (1)

Widget to send content

LibertyWSRPPublisher

Select content to send

{http://LibertyWSRPSampl… as {http://www.ibm.c…

Select a widget to receive content

LibertyWSRPSubscriber

Select an action

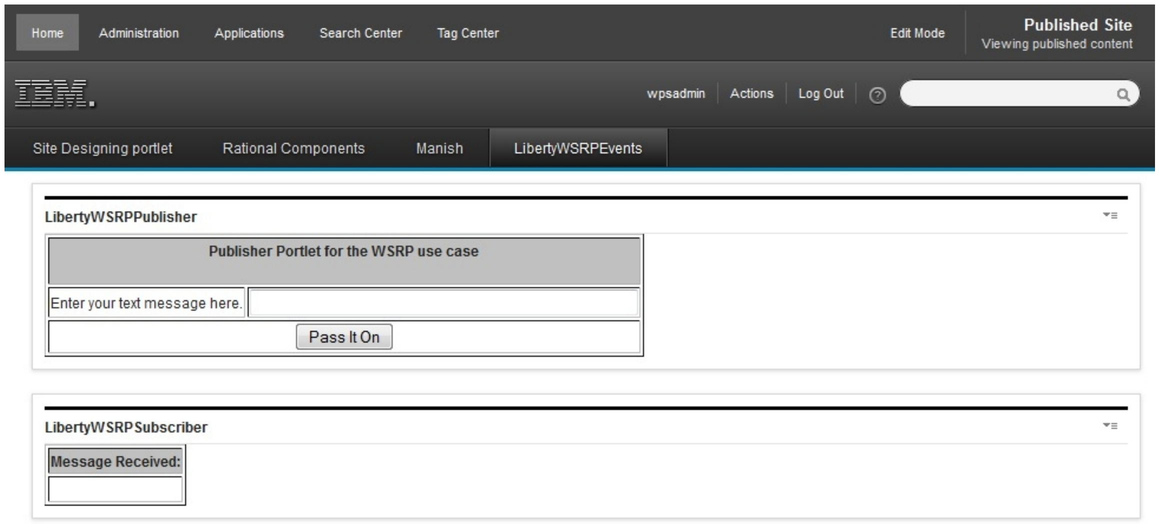
{http://LibertyWSRPSampl… using {http://www.ibm.…

- Save the changes made to the Portal Page.
- Switch to View Mode.

Result

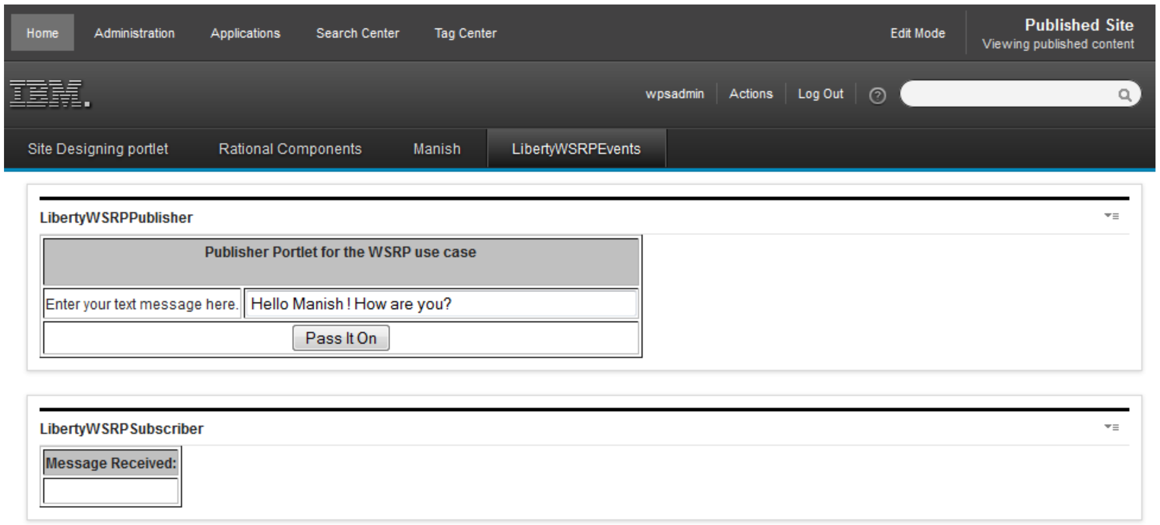
- To see the portlets in action, navigate to LibertyWSRPEvents page under Home menu. See Figure 19.

Figure 19: Portlets displayed on Portal page



- Enter **a** string in the message box for LibertyWSRPPublisher portlet. See Figure 20.
- Click the button.

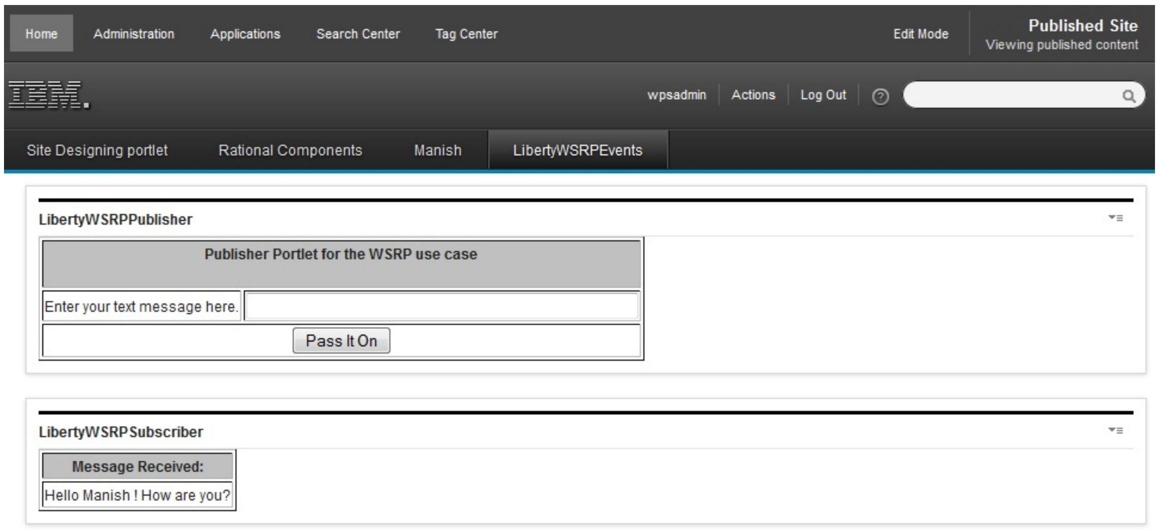
Figure 20: String entered to the first portlet



Hello Manish ! How are you?

- The page refreshes to show the same string appear in LibertyWSRPSubscriber portlet. See Figure 21.

Figure 21: Message displays in the subscriber portlet



LibertyWSRPPublisher

Publisher Portlet for the WSRP use case

Enter your text message here.

Pass It On

LibertyWSRPSubscriber

Message Received:

Hello Manish ! How are you?

Summary

In this article, you first learnt how easily you can create and publish a portlet project targeted against WebSphere Application Server Liberty profile using IBM Portlet Container Tools for WebSphere Application Server Liberty Profile. Next you configured the two portlets to communicate with each other using JSR events mechanism. Finally you observed how a portlet hosted on IBM WebSphere Application Server Liberty profile can be consumed through IBM WebSphere Portal, using WSRP.

Resources

- IBM Portlet Container Tools for WebSphere Application Server Liberty Profile at IBM Collaboration Solutions Catalog,
  http://ibmurl.hursley.ibm.com/47QE

- To install WebSphere Developer Tools, you can visit
  https://developer.ibm.com/wasdev/downloads/#wasdev/view/535905568ee62f5d660003ea

- To install WebSphere Application Server V8.5.5 Liberty Profile, you can refer
  https://developer.ibm.com/wasdev/downloads/liberty-profile-using-eclipse/

- To install Portlet Container feature for WAS Liberty Profile, see the following link,
  https://developer.ibm.com/wasdev/downloads/#wasdev/view/535abc8c528289eb2200006b

- To install Portlet Serving feature for WAS Liberty Profile, check the following link, https://developer.ibm.com/wasdev/downloads/#wasdev/view/5358f5b18ee62f5d660003dd

- The WSRP Producer for WAS Liberty Profile is available at the IBM Collaboration Solutions Catalog,
http://ibmurl.hursley.ibm.com/4A47

- IBM WSRP 2.0 Producer for WebSphere Application Server documentation, http://ibmurl.hursley.ibm.com/MHGH

- IBM WebSphere Portal 8 product documentation, http://ibmurl.hursley.ibm.com/MHGG