

JBoss Portlet Bridge

Reference Guide

Release 2.0.0.CR1

JBoss Portlet Bridge Overview	v
1. Getting started with JBoss Portlet Bridge	1
1.1. What's New in 2.0?	1
1.1.1. Eventing	1
1.1.2. Portlet Served Resources	1
1.1.3. Public Render Parameters	1
1.2. Bridge Frameworks and Extensions	1
1.2.1. Seam Bridgelets	2
1.2.2. RichFaces Bridgelets	2
1.3. Before you start	2
1.4. Maven Archetypes	3
1.4.1. Running the Examples (Optional)	4
1.5. Video Tutorials	5
2. Bridge Configuration	7
2.1. Core Setup and Configuration	7
2.1.1. portlet.xml	7
2.1.2. faces-config.xml	8
2.1.3. Facelets Configuration	8
2.1.4. JSP Only Configuration	9
2.1.5. JSR-329	10
2.2. RichFaces Setup and Configuration Options	10
2.2.1. web.xml	10
2.3. Seam Setup and Configuration Options	12
2.3.1. Configuration	12
2.4. Portlet 2.0 Coordination	12
2.4.1. Sending and Receiving Events	13
2.4.2. Public Render Parameters	14
2.4.3. Serving Your JSF Resources in a Portlet	15
3. Developing Portlets with the Bridge	17
3.1. Excluding Attributes from the Bridge Request Scope	17
3.2. Supporting PortletMode Changes	17
3.3. Navigating to a mode's last viewId	18
3.4. General Error Handling	19
3.5. Custom Ajax Error Handling	19
3.6. Communication Between Your Portlets	20
3.6.1. Storing Components in PortletSession.APPLICATION_SCOPE	21
3.6.2. Using the PortletSession	21
3.7. Direct Linking to Portlet/JSF Pages Using h:outputink	22

JBoss Portlet Bridge Overview

To get an idea of the JBoss Portlet Bridge community, the developers, and for wiki information, checkout [the project page](http://www.jboss.org/portletbridge/) [http://www.jboss.org/portletbridge/].

What is the JBoss Portlet Bridge?

The JBoss Portlet Bridge (or JBPB for short) is a non-final draft implementation of the [JSR-329](http://jcp.org/en/jsr/detail?id=329) [http://jcp.org/en/jsr/detail?id=329] specification which supports the JSF 1.2 runtime within a JSR 286 portlet and with added enhancements to support other web frameworks (such as [Seam](http://www.seamframework.org/) [http://www.seamframework.org/] and [RichFaces](http://www.jboss.org/jbossrichfaces/) [http://www.jboss.org/jbossrichfaces/]). It allows any Java developer to get started quickly with their JSF web application running in a portal environment. The developer no longer needs to worry about the underlying portlet development, portlet concepts, or the API.

Understanding how JSF works with Portal

The portlet bridge isn't a portlet. It's the mediator between the two environments and allows JSF and Portal to be completely unaware of each other. The bridge is used to execute Faces requests on behalf of the portlet. During each request, the Faces environment is setup and handled by the bridge. Part of this implementation acts as a Faces controller much as the FacesServlet does in the direct client request world. The other part of this implementation is provided by implementing a variety of (standard) Faces extensions.



Note

This draft specification for the JSR 329 specification is not final. Any final specification that may be published will likely contain differences, some of which may be substantial. Publication of this draft specification is not intended to provide the basis for implementations of the specification. This draft specification is provided AS IS, with all faults. THERE ARE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF CONDITION OF TITLE OR NONINFRINGEMENT. You may copy and display this draft specification provided that you include this notice and any existing copyright notice. Except for the limited copyright license granted above, there are no other licenses granted to any intellectual property owned or controlled by any of the authors or developers of this material. No other rights are granted by implication, estoppel or otherwise.

Getting started with JBoss Portlet Bridge

JBoss Portlet Bridge not only gives you the ability to run JSF web applications in a portlet, but also gives you the benefit of running supported JBoss frameworks like Seam and RichFaces.

1.1. What's New in 2.0?

1.1.1. Eventing

The bridge considers a portlet event a model event. I.e. the event is targeted to the applications data model not its view. As JSF events primarily concern its view, the bridge processes the portlet events manually, however provisions are made to make sure that any model changes that result from processing the event are updated in the view. Since event payloads are arbitrarily complex, the manual processing of the data, though managed by the bridge, is left to the (portlet) application to support.

See [Section 2.4.1, “Sending and Receiving Events”](#) for details and examples.

1.1.2. Portlet Served Resources

The bridge deals with portlet served resources in one of two ways. If the request is for a non-JSF resource, the bridge handles the request by acquiring a request dispatcher and forwarding the request to the named resource. If the request is for a JSF resource, the bridge runs the full JSF lifecycle ensuring that data is processed and the resource (markup) is rendered.

See [Section 2.4.3, “Serving Your JSF Resources in a Portlet”](#) for details and examples.

1.1.3. Public Render Parameters

The bridge automates the processing of public render parameters. A public render parameter can be mapped to an object's accessor (get/set method) designed to handle a String representation of the value via a Faces ValueExpression. When a new public render parameter value is received in a request, the bridge sets the value by calling the ValueExpression's setValue(). At the end of a request, if the current value of any mapped public render parameter doesn't match the current incoming value, the bridge sets the new value in an outgoing public render parameter (if feasible in the given phase).

See [Section 2.4.2, “Public Render Parameters”](#) for details and examples.

1.2. Bridge Frameworks and Extensions

The JBoss Portlet Bridge currently supports JBoss Portal, GateIn, JSF 1.2, JBoss Seam, and JBoss Richfaces. There are configurations that apply to supporting each framework. See [section Chapter 2, Bridge Configuration](#) for instructions.

The JBoss Portlet Bridge project is also actively developing extensions, and to differentiate from just another "project" that has boring ol' "extensions" we coined the term "Bridgelets" - because what would a project with Java and JSF be without having "*let" on the end of it? Not very cool in my opinion ;) With that said, in this current release we decided to bring all of our bridgelets into the impl code base since they are critical in most JSF portlet applications. Now it only takes a single line of configuration to utilize these features.

1.2.1. Seam Bridgelets

For example, the PortalIdentity seam component allows you to instantly have SSO between Seam and GateIn or JBoss Portal. This extension is configured in your Seam application's components.xml file as follows.

```
<security:portal-identity authenticate-method="#{authenticator.authenticate}"/>
```

1.2.2. RichFaces Bridgelets

Richfaces does not account for multiple components on the same portal page by default. This following web.xml renders all RichFaces component javascript to be portal friendly.

```
<context-param>
  <param-name>org.jboss.portletbridge.WRAP_SCRIPTS</param-name>
  <param-value>true</param-value>
</context-param>
```



Note

Don't forget that the bridge is still in Beta and so are any extensions. If you would like to contribute to any part of this project, we encourage you to be active on the [user forum](http://www.jboss.com/index.html?module=bb&op=viewforum&f=273) [http://www.jboss.com/index.html?module=bb&op=viewforum&f=273] and bring issues/enhancements to attention.

1.3. Before you start

Current version and compatibility information can be easily located on the [JBPB wiki](http://www.jboss.org/community/wiki/JBossPortletBridge) [http://www.jboss.org/community/wiki/JBossPortletBridge]. Ensure you are using compatible versions of all integrated frameworks before you begin.

JBoss Portal and GateIn provides it's latest distribution included in JBoss Application Server. All of the guesswork has been eliminated so that you can unzip and run the Portal with a few clicks. [Get the latest here](http://jboss.org/gatein/downloads.html) [http://jboss.org/gatein/downloads.html] (ensure you choose the Portal + JBoss AS link)

Next, all that's left is to download the [JBoss Portlet Bridge distribution](http://www.jboss.org/portletbridge/download/) [http://www.jboss.org/portletbridge/download/] and configure your portlet to use the bridge. Or, you can run a provided archetype [Section 1.4, "Maven Archetypes"](#) and deploy the generated war in a few easy steps. This will also give you an empty project to play around with or start from scratch.

1.4. Maven Archetypes

This project utilizes [Maven archetypes](http://maven.apache.org/guides/introduction/introduction-to-archetypes.html) [http://maven.apache.org/guides/introduction/introduction-to-archetypes.html] which allow you get up and running with different flavors of the bridge quickly.

Table 1.1. Available Archetypes

Archetype	Command
JSF 1.2 Basic	<pre>mvn archetype:generate -DarchetypeGroupId=org.jboss.portletbridge.archetypes -DarchetypeArtifactId=1.2-basic -DarchetypeVersion=2.0.0.CR1 -DgroupId=org.whatever.project -DartifactId=myprojectname -DarchetypeRepository=http://repository.jboss.org/maven2/</pre>
RichFaces Basic	<pre>mvn archetype:generate -DarchetypeGroupId=org.jboss.portletbridge.archetypes -DarchetypeArtifactId=richfaces-basic -DarchetypeVersion=2.0.0.CR1 -DgroupId=org.whatever.project -DartifactId=myprojectname -DarchetypeRepository=http://repository.jboss.org/maven2/</pre>
Seam Basic (Modular EAR)	<pre>mvn archetype:generate -DarchetypeGroupId=org.jboss.portletbridge.archetypes -DarchetypeArtifactId=seam-basic -DarchetypeVersion=2.0.0.CR1 -DgroupId=org.whatever.project -DartifactId=seamproject -DarchetypeRepository=http://repository.jboss.org/maven2/</pre>

1.4.1. Running the Examples (Optional)



JSF 1.2 Basic, RichFaces Basic, Seam Basic, and other demos

The Maven-Cargo setup mentioned below is completely optional. It gives users who are new to the bridge or ones who may have problems running the bridge in their current setup, a completely fresh install of both JBoss AS and JBoss Portal to run against - with only a few commands. This setup only requires that the latest versions of Java and Maven be installed on your machine, and the following 3 commands will take care of the rest.

To see a step-by-step tutorial of this setup in action, follow along here:

[Lesson 1: Getting Started With The Bridge](http://www.vimeo.com/3977469) [http://www.vimeo.com/3977469]

Each example application is configured to download the latest versions of JBoss Portal bundled with JBoss Application Server. After running the archetype [Section 1.4, "Maven Archetypes"](#) or [downloading the source code](#) [http://www.jboss.org/portletbridge/download/] for the example application that you're interested in, you can run one of the following Maven profiles to save time and get everything up and running with only 2 commands. You have 2 options for deploying the generated project using Maven. You can let the project download, run, and deploy JBoss AS and Portal automatically with the first option . Or you can use your own local install of JBoss AS and Portal to deploy this project with the second option.

JBoss Portal 2.7.2.GA + JBoss AS 4.2.3 (Bundled download)

1. mvn install -Plocal-portal cargo:start
2. mvn cargo:deploy -Plocal-portal
3. visit <http://localhost:8080/portal>

For more commands, view the README.txt in each project. *If you plan on using the cargo profiles to do active development, you can save a lot of time by not downloading the bundle each time you do a clean install. To use a locally configured server bundled with portal, use the following command line parameters. The variable for JBOSS_HOME_DIR is related to how you zip the server directory. If you zip the files under JBOSS_HOME/* then it will only be the name of your archive. But if you zip the actual folder JBOSS_HOME then JBOSS_HOME_DIR*

must be defined as 'zip file name/JBOSS_HOME folder name'. So basically, just zip up your local install of JBoss AS and portal (or download the bundle from sourceforge) if you want to use this option.

JBoss Portal 2.7.2.GA + JBoss AS 4.2.3 (Downloaded and stored locally)

1. `mvn install cargo:start -Dlocal -Plocal-portal -DJBoss_ZIP_HOME={path to zipped portal + JBoss AS}/jboss-portal-2.7.2-bundled.zip -DJBoss_HOME_DIR=jboss-portal-2.7.2-bundled/jboss-portal-2.7.2`
2. `mvn cargo:deploy -Plocal-portal`
3. visit `http://localhost:8080/portal`

1.5. Video Tutorials

Lesson 1: Getting Started With The Bridge [<http://www.vimeo.com/3977469>]

Lesson 2: Portlet 1.0 Advanced Seam and RichFaces [<http://www.vimeo.com/4521877>]

Lesson 3: Seam and Portlet 2.0 Eventing [<http://www.vimeo.com/5847864>]

Lesson 4: Running the 2.0 bridge on Gateln and deploy using JBoss Tools [<http://www.vimeo.com/7255033>]

Check here for the latest videos. [<http://www.vimeo.com/wesleyhales/videos>]

Bridge Configuration

The 329 specification is aimed at making the developers life as easy as possible with JSF+Portlet development. You will see below that there are minimal settings to getting any JSF web application up and running in the Portal environment.

If you are starting from scratch, we highly recommend you use the [Section 1.4, “Maven Archetypes”](#).

2.1. Core Setup and Configuration

2.1.1. portlet.xml

The basic JSR-329 portlet configuration.

```
<portlet>
  <portlet-name>yourPortletName</portlet-name>
  <portlet-class>
    javax.portlet.faces.GenericFacesPortlet
  </portlet-class>

  <init-param>
    <name>javax.portlet.faces.defaultViewId.view</name>
    <value>/welcome.xhtml</value>
  </init-param>

  <init-param>
    <name>javax.portlet.faces.defaultViewId.edit</name>
    <value>/jsf/edit.xhtml</value>
  </init-param>

  <init-param>
    <name>javax.portlet.faces.defaultViewId.help</name>
    <value>/jsf/help.xhtml</value>
  </init-param>
```

When `preserveActionParams` is set to `TRUE`, the bridge must maintain any request parameters assigned during the portlet's action request. The request parameters are maintained in the *"bridge request scope"*. When this attribute isn't present or is `FALSE` the action's request parameters are only maintained for the duration of the *portlet request scope*.

```
<init-param>
  <name>javax.portlet.faces.preserveActionParams</name>
  <value>>true</value>
</init-param>
```

2.1.2. faces-config.xml

The PortletViewHandler ensures that each JSF portlet instance is properly namespaced.

```
<faces-config>
  <application>
    <view-handler>
      org.jboss.portletbridge.application.PortletViewHandler
    </view-handler>
    <state-manager>org.jboss.portletbridge.application.PortletStateManager</state-
manager>
  </application>
  ...
```

2.1.3. Facelets Configuration

The following web.xml setting is only for Facelets based applications

2.1.3.1. web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/
web-app_2_4.xsd"
  version="2.4">
  ...
  <!-- This is optional parameters for a facelets based application -->
  <context-param>
    <param-name>org.ajax4jsf.VIEW_HANDLERS</param-name>
    <param-value>org.jboss.portletbridge.application.FaceletPortletViewHandler</param-
value>
  </context-param>
```

```

<context-param>
  <param-name>javax.portlet.faces.RENDER_POLICY</param-name>
  <param-value>
    ALWAYS_DELEGATE
  </param-value>
</context-param>
...
</web-app>

```



RenderPolicy Options

- **ALWAYS_DELEGATE** Indicates the bridge should not render the view itself but rather always delegate the rendering.
- **NEVER_DELEGATE** Indicates the bridge should always render the view itself and never delegate.
- **DEFAULT** Directs the bridge to first delegate the render and if and only if an Exception is thrown then render the view based on its own logic. If the configuration parameter is not present or has an invalid value the bridge renders using default behavior. I.e. as if DEFAULT is set.

2.1.4. JSP Only Configuration

The following web.xml setting is only for JSP based applications. Download the demo application [here](http://anonsvn.jboss.org/repos/portletbridge/trunk/examples/jsf-ri/1.2-basic/) [http://anonsvn.jboss.org/repos/portletbridge/trunk/examples/jsf-ri/1.2-basic/].

2.1.4.1. web.xml

```

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/
web-app_2_4.xsd"
  version="2.4">

  <context-param>
    <param-name>javax.portlet.faces.renderPolicy</param-name>
    <param-value>

```

```
        NEVER_DELEGATE
    </param-value>
</context-param>

...
</web-app>
```

2.1.5. JSR-329

The Jboss Portlet Bridge can be used with a any compatible implementation (for example, MyFaces implementation). Simply put the following into web.xml :

```
<context-param>
  <param-name>javax.portlet.faces.BridgeImplClass</param-name>
  <param-value>org.apache.myfaces.portlet.faces.bridge.BridgeImpl</param-value>
</context-param>
```

2.2. RichFaces Setup and Configuration Options

2.2.1. web.xml

The following configuration is designated for portlets using the RichFaces library. These settings will vary based on your individual needs. See [this section](http://www.jboss.org/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html/ArchitectureOverview.html#ScriptsandStylesLoadStrategy) [http://www.jboss.org/file-access/default/members/jbossrichfaces/freezone/docs/devguide/en/html/ArchitectureOverview.html#ScriptsandStylesLoadStrategy] of the RichFaces documentation for more details.

Sometimes it is better to use the "ALL" load strategy in portlets so you do not need to worry about loading the "framework.pack.js" and "ui.pack.js" files manually in your portlet header.

```
<context-param>
  <param-name>org.richfaces.LoadStyleStrategy</param-name>
  <param-value>ALL</param-value>
</context-param>
<context-param>
  <param-name>org.richfaces.LoadScriptStrategy</param-name>
  <param-value>ALL</param-value>
</context-param>
```




Note

If you use the "NONE" strategy, you must include the following scripts in your portlet or portal page header. If you are using JBoss Portal, you can add this to the jboss-portlet.xml file.

The `org.ajax4jsf.RESOURCE_URI_PREFIX` configuration cross references the path to your scripts below. These settings are required for RichFaces using the "NONE" strategy.

```
<script src="/faces/rfRes/org/ajax4jsf/framework.pack.js" type="text/javascript"></script>
<script src="/faces/rfRes/org/richfaces/ui.pack.js" type="text/javascript"></script>
<link rel="stylesheet" type="text/css" href="/faces/rfRes/org/richfaces/skin.xcss"/>
```

Seam automatically configures your Ajax4JSF Filter, so if you are running a Seam portlet, you do not need the following Filter config. (But you do need the `RESOURCE_URI_PREFIX` no matter what)

```
<context-param>
  <param-name>org.ajax4jsf.RESOURCE_URI_PREFIX</param-name>
  <param-value>rfRes</param-value>
</context-param>

<filter>
  <display-name>Ajax4jsf Filter</display-name>
  <filter-name>ajax4jsf</filter-name>
  <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>

<filter-mapping>
  <filter-name>ajax4jsf</filter-name>
  <servlet-name>FacesServlet</servlet-name>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>

...
</web-app>
```

2.3. Seam Setup and Configuration Options

2.3.1. Configuration

The `ExceptionHandler` is used to clean Seam contexts and transactions after errors.

```
<context-param>
  <param-name>org.jboss.portletbridge.ExceptionHandler</param-name>
  <param-value>
    org.jboss.portletbridge.SeamExceptionHandlerImpl
  </param-value>
</context-param>
```

If you are using this bridge packaged from 2.0.0.BETA onward, you must define the following web.xml parameter to use the JBoss Portlet Bridge provided Seam Phase Listener.

```
<context-param>
  <param-name>javax.faces.LIFECYCLE_ID</param-name>
  <param-value>SEAM_PORTLET</param-value>
</context-param>
```

2.4. Portlet 2.0 Coordination

One very important thing to note before using either of the following mechanisms, is that you must have the proper 2.0 schema and xsd definition at the top of your portlet.xml.

```
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
  version="2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
  http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd">
```

2.4.1. Sending and Receiving Events

2.4.1.1. Configuration

Just like with any portlet 2.0 event consumer and receiver, you must define them in the portlet.xml. To see a working example, checkout the Seam Booking Demo portlet. <http://anonsvn.jboss.org/repos/portletbridge/tags/2.0.0.CR1/examples/seam/booking/>

You must also define the following init params in your portlet.xml.

```
</init-param>
<init-param>
  <name>javax.portlet.faces.autoDispatchEvents</name>
  <value>true</value>
</init-param>
<init-param>
  <name>javax.portlet.faces.bridgeEventHandler</name>
  <value>org.foo.eventhandler</value>
</init-param>
```

For now, you must dispatch the event in the JSF or Seam backing bean. Future versions on the 2.0 bridge will automate the dispatching and consuming of events.

```
if (response instanceof StateAwareResponse) {
    StateAwareResponse stateResponse = (StateAwareResponse) response;
    stateResponse.setEvent(Foo.QNAME, new Bar());
}
```

Then you must also create the event handler class by implementing the BridgeEventHandler interface to process the event payload.

```
public class BookingEventHandler implements BridgeEventHandler
{
    public EventNavigationResult handleEvent(FacesContext context, Event event)
    {
        //process event payload here
    }
}
```

```
}
```

2.4.2. Public Render Parameters

2.4.2.1. Configuration

Public Render Parameters (or PRPs) are one of the most powerful and simple Portlet 2.0 features. Several portlets (JSF or not) can share the same render parameters. This feature can be used to present a cohesive UI to the user across all portlets on the page (i.e. using an employee ID to display relative data).

The bridge maps a render parameter to a backing bean using settings in your `faces-config.xml` and `portlet.xml`. A clear and working example can be found in the Seam Booking Demo portlet. <http://anonsvn.jboss.org/repos/portletbridge/tags/2.0.0.CR1/examples/seam/booking/>

You must define the following init params in your `portlet.xml`.

```
<init-param>
  <name>javax.portlet.faces.bridgePublicRenderParameterHandler</name>
  <value>org.foo.PRPHandler</value>
</init-param>
...
<supported-public-render-parameter>myCoolPRP</supported-public-render-parameter>
```

Create a managed bean and public-parameter-mappings in your `faces-config.xml`. This should be a basic bean that you can bind the passed parameter to a string with getter and setter.

```
<managed-bean>
  <managed-bean-name>bookingPRP</managed-bean-name>
  <managed-bean-class>your.class.Name</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>

<application>
  <application-extension>
    <bridge:public-parameter-mappings>
      <bridge:public-parameter-mapping>
        <parameter>"the name of your portlet":hotelName</parameter>
```

```
<model-el>#{bookingPRP.hotelName}</model-el>
</bridge:public-parameter-mapping>
</bridge:public-parameter-mappings>
</application-extension>
</application>
```

You must set the parameter in the JSF or Seam backing bean, if you are providing one from your portlet.

```
if (response instanceof StateAwareResponse) {
    StateAwareResponse stateResponse = (StateAwareResponse) response;
    stateResponse.setRenderParameter("hotelName",selectedHotel.getName());
}
```

Then you must also implement the `BridgePublicRenderParameterHandler` interface to process any updates from the received parameter.

```
public void processUpdates(FacesContext context)
{
    ELContext elContext = context.getELContext();
    BookingPRPBean bean = (BookingPRPBean) elContext.getELResolver().getValue(elContext,
    null, "bookingPRP");

    if(null != bean){
        //Do something with bean.getHotelName();
    } else {

    }
}
```

2.4.3. Serving Your JSF Resources in a Portlet

2.4.3.1. Configuration

We have setup a few examples to show you how to use EL and a simple bean that will allow you to use the portlet resource serving mechanism within a JSF portlet.

In [ResourceBean.java](http://anonsvn.jboss.org/repos/portletbridge/tags/2.0.0.CR1/examples/richfaces/richfaces-demo/src/main/java/org/richfaces/demo/common/ResourceBean.java) [http://anonsvn.jboss.org/repos/portletbridge/tags/2.0.0.CR1/examples/richfaces/richfaces-demo/src/main/java/org/richfaces/demo/common/ResourceBean.java], you can see a very simple implementations of a Map object that uses the bridge to get and encode a resource url served from the portlets web application.

So, when you have the normal "/images", "/styles" and other resource folders in your web app, you can use the following EL expression to serve them in your JSF application.

```
{resource['/img/the-path-to-my-image.png']}
```

Just copy the ResourceBean.java code above, and add an entry to you faces-config.xml for the bean:

```
<managed-bean>
  <managed-bean-name>resource</managed-bean-name>
  <managed-bean-class>org.richfaces.demo.common.ResourceBean</managed-bean-class>
  <managed-bean-scope>application</managed-bean-scope>
</managed-bean>
```

Developing Portlets with the Bridge

This chapter demonstrates common development tasks described by the 329 specification.

3.1. Excluding Attributes from the Bridge Request Scope

When your application uses request attributes on a per request basis and you do not want that particular attribute to be managed in the extended bridge request scope, you must use the following configuration in your faces-config.xml. Below you will see that any attribute namespaced as foo.bar or any attribute beginning with foo.baz(wildcard) will be excluded from the bridge request scope and only be used per that application's request.

```
<application>
  <application-extension>
    <bridge:excluded-attributes>
      <bridge:excluded-attribute>foo.bar</bridge:excluded-attribute>
      <bridge:excluded-attribute>foo.baz.*</bridge:excluded-attribute>
    </bridge:excluded-attributes>
  </application-extension>
</application>
```

3.2. Supporting PortletMode Changes

A PortletMode represents a distinct render path within an application. There are three standard modes: view, edit, and help. The bridge's ExternalContext.encodeActionURL recognizes the query string parameter javax.portlet.faces.PortletMode and uses this parameter's value to set the portlet mode on the underlying portlet actionURL or response. Once processed it then removes this parameter from the query string. This means the following navigation rule causes one to render the \edit.jspx viewId in the portlet edit mode:

```
<navigation-rule>
  <from-view-id>/register.jspx</from-view-id>
  <navigation-case>
    <from-outcome>edit</from-outcome>
    <to-view-id>/edit.jspx?javax.portlet.faces.PortletMode=edit</to-view-id>
  </navigation-case>
</navigation-rule>
```

3.3. Navigating to a mode's last viewId

By default a mode change will start in the mode's default view without any (prior) existing state. One common portlet pattern when returning to the mode one left after entering another mode (e.g.. view -> edit -> view) is to return to the last view (and state) of this origin mode. The bridge will explicitly encode the necessary information so that when returning to a prior mode it can target the appropriate view and restore the appropriate state. The session attributes maintained by the bridge are intended to be used by developers to navigate back from a mode to the last location and state of a prior mode. As such a developer needs to describe a dynamic navigation: "from view X return to the last view of mode y". This is most easily expressed via an EL expression. E.g.

```
<navigation-rule>
  <from-view-id>/edit.jspx* </from-view-id>
  <navigation-case>
    <from-outcome>view </from-outcome>
    <to-view-id>#{sessionScope['javax.portlet.faces.viewIdHistory.view']} </to-view-id>
  </navigation-case>
</navigation-rule>
```



Note to Portlet Developers

Depending on the bridge implementation, when using values from these session scoped attributes or any viewIds which may contain query string parameters it may be necessary to use the wildcard syntax when identifying the rule target. For example, the above

```
<to-view-id>
```

expression returns a viewId of the form

```
/viewId?javax.portlet.faces.PortletMode=view&....
```

Without wildcarding, when a subsequent navigation occurs from this new view, the navigation rules wouldn't resolve because there wouldn't be an exact match. Likewise, the above edit.jspx


```
<from-view-id>
```

is wildcarded because there are navigation rules that target it that use a query string (

```
<to-view-id> /edit.jspx?javax.portlet.faces.PortletMode=edit </to-view-id>
```

). Developers are encouraged to use such wildcarding to ensure they execute properly in the broadest set of bridge implementations.

3.4. General Error Handling



Note

If you're developing a Seam portlet you can now use pages.xml for all error handling.

The following configuration may be used to handle exceptions. This is also useful for handling session timeout and ViewExpiredExceptions.

Pay attention to the location element. It must contain the /faces/ mapping to work properly.

```
<error-page>
  <exception-type>javax.servlet.ServletException</exception-type>
  <location>/faces/error.xhtml</location>
</error-page>
<error-page>
  <exception-type>javax.faces.application.ViewExpiredException</exception-type>
  <location>/faces/error.xhtml</location>
</error-page>
```

3.5. Custom Ajax Error Handling

By default, error handling is sent to a standard servlet page for Ajax requests. To handle the error inside the portlet, use the following javascript:

```
<script type="text/javascript">
  A4J.AJAX.onError = function(req,status,message){
    window.alert("Custom onError handler "+message);
  }

  A4J.AJAX.onExpired = function(loc,expiredMsg){
    if(window.confirm("Custom onExpired handler "+expiredMsg+" for a location: "+loc)){
      return loc;
    } else {
      return false;
    }
  }
</script>
```

Also, add the following to web.xml. Read more about these settings here [Request Errors and Session Expiration Handling](http://docs.jboss.org/richfaces/3.3.3.BETA1/en/devguide/html/ArchitectureOverview.html#RequestErrorsAndSessionExpirationHandling) [http://docs.jboss.org/richfaces/3.3.3.BETA1/en/devguide/html/ArchitectureOverview.html#RequestErrorsAndSessionExpirationHandling]

```
<context-param>
  <param-name>org.ajax4jsf.handleViewExpiredOnClient</param-name>
  <param-value>true</param-value>
</context-param>
```

3.6. Communication Between Your Portlets

There are roughly 4 different ways to send messages, events, and parameters between portlets which are contained in different ears/wars or contained in the same war. The Portlet Container does not care if you have 2 portlets in the same war or if they are separated, because each portlet has a different HttpSession.

Of course, with the Portlet 2.0 spec, the recommended way to share a parameter or event payload between 2 or more portlets are the [Section 2.4.2, “Public Render Parameters”](#) and [Section 2.4.1, “Sending and Receiving Events”](#) mechanisms. This allows you to decouple your application from surgically managing objects in the PortletSession.APPLICATION_SCOPE.

But, if these do not meet your usecase or you have a different strategy, you can use one of the following methods.

3.6.1. Storing Components in PortletSession.APPLICATION_SCOPE

Sometimes it makes sense to store your Seam components in the portlet APPLICATION_SCOPE. By default, these objects are stored in the PORTLET_SCOPE but with the annotation below, you can fish this class out of the PortletSession and use its values in other portlets across different Seam applications.

```
@PortletScope(PortletScope.ScopeType.APPLICATION_SCOPE)
```

Then you would pull the statefull object from the session:

```
                                YourSessionClass    yourSessionClass  
= (YourSessionClass)getRequest().getAttribute("javax.portlet.p./default/seamproject/  
seamprojectPortletWindow?textHolder");
```

This method is demonstrated in this video: [Lesson 2: Portlet 1.0 Advanced Seam and RichFaces](http://www.vimeo.com/4521877)
[<http://www.vimeo.com/4521877>]

3.6.2. Using the PortletSession

If you need to access the PortletSession to simply share a parameter/value across multiple portlets, you can use the following to do so.

```
                                Object            objSession            =  
FacesContext.getCurrentInstance().getExternalContext().getSession(false);  
try  
{  
    if (objSession instanceof PortletSession)  
    {  
        PortletSession portalSession = (PortletSession)objSession;  
        portalSession.setAttribute("your parameter name","parameter  
value",PortletSession.APPLICATION_SCOPE);  
        ...  
    }  
}
```

Then, in your JSP or Facelets page, you can use:

```
{httpSessionScope['your parameter name']}
```



Note to Portlet Developers

`{httpSessionScope}` was implemented after 2.0.0.BETA. If you are using the 1.0 bridge or pre 2.0.0.BETA, you must use the el variable `{sessionApplicationScope}`

For more information about which EL variables are provided by the bridge, read [section 6.5.1 of the JSR-329 specification](http://jcp.org/aboutJava/communityprocess/edr/jsr329/index2.html) [http://jcp.org/aboutJava/communityprocess/edr/jsr329/index2.html].

3.7. Direct Linking to Portlet/JSF Pages Using `h:outputink`

For linking to any JSF/Facelets page within your portlet web application, you may use the following.

```
<h:outputLink value="{facesContext.externalContext.requestContextPath}/home.xhtml">
  <f:param name="javax.portlet.faces.ViewLink" value="true"/>
  navigate to the test page
</h:outputLink>
```