JBoss Portal 2.0 Final

Reference Guide

Table of Contents

JBoss Portal - Overview

Many IT organizations look to achieve a competitive advantage for the enterprise by improving business productivity and reducing costs. Today's top enterprises are realizing this goal by deploying enterprise portals within their IT infrastructure. Enterprise portals simplify access to information by providing a single source of interaction with corporate information. Although today?s packaged portal frameworks help enterprises launch portals more quickly, only JBoss Portal can deliver the benefits of a zero-cost open source license combined with a flexible and scalable underlying platform.

JBoss Portal 2.0 provides an open source and standards-based environment for hosting and serving a portal?s Web interface, publishing and managing its content, and customizing its experience. It is entirely standards-based and supports the JSR-168 portlet specification, which allows you to easily plug-in standards-compliant portlets to meet your specific portal needs. JBoss Portal 2.0 is available through the business-friendly LGPL [http://www.jboss.com/company/aboutopensource] open source license and is supported by JBoss Inc. Professional Support and Consulting [http://www.jboss.com/services/index]. JBoss support services are available to assist you in designing, developing, deploying, and ultimately managing your portal environment. JBoss Portal is currently developed by JBoss, Inc. developers, Novell developers, and community contributors.

The JBoss Portal 2.0 framework and architecture includes the portal container and supports a wide range of features including standard portlets, single sign-on, clustering and internationalization. Portal themes and layouts are configurable. Fine-grained security administration down to portlet permissions rounds out the security model. JBoss Portal 2.0 includes a rich content management system and message board support.

JBoss Portal Resources:

1. JBoss Portal Home Page [http://www.jboss.org/products/jbossportal]
2. Forums: User [http://www.jboss.org/index.html?module=bb&op=viewforum&f=215] | Developer [http://www.jboss.org/index.html?module=bb&op=viewforum&f=205]
3. Wiki [http://www.jboss.com/wiki/Wiki.jsp?page=JBossPortal]
4. Our Roadmap [http://jira.jboss.com/jira/browse/JBPORTAL?report=com.atlassian.jira.plugin.system.project:roadmap-panel]

The JBoss Portal team encourages you to use this guide to configure and develop on the JBoss Portal framework. If you encounter any configuration issues or simply want to take part in our community, we would love to hear from you in our forums.

Feature List

The following list details features found in this document's related release. Currently, this is JBoss Portal 2.0 Final. For a technical view of our features, view the Project Roadmap and Task List [http://jira.jboss.com/jira/browse/JBPORTAL].

Technology and Architecture

- **JEMS:** Leverages the power of JBoss Enterprise Middleware Services : JBoss Application Server, JBoss Cache, JGroups, and Hibernate.
- **DB Agnostic:** Will work with any RDBMS supported by Hibernate
- **SSO/LDAP:** Leverages Tomcat and JBoss single sign on (SSO) solutions.
- **JAAS Authentication:** Custom authentication via JAAS login modules.
- **Cacheing:** Utilizes render-view caching for improved performance.
- **Clusterable:** Cluster support allows for portal state to be clustered for all portal instances.
- **Hot-Deployment:** Leverages JBoss dynamic auto deployment features.
- **SAR Installer:** Browser-based installer makes installation and initial configuration a breeze.

Portal and Portal Container

- **Multiple Portal Instances:** Ability to have multiple Portal instances running inside of one Portal container.
- **Internationalization:** Ability to use internationalization resource files for every portlet.
- **Pluggable services:** Authentication performed by the servlet container and JAAS make it possible to swap the authentication scheme.
- **Page-based Architecture:** Allows for the grouping/division of portlets on a per-page basis.
- **Existing Framework support:** Portlets utilizing Struts, Spring MVC, Sun JSF-RI, or MyFaces are supported.

Themes and Layouts

- **Easily swappable themes/layouts:** New themes and layouts containing images can be deployed in WAR archives.
- **Flexible API:** Theme and Layout API are designed to separate the business layer from the presentation layer.
- **Per-page layout strategy:** Different layouts can be assigned to different pages.

User and Group Functionality

- **User registration/validation:** Configurable registration parameters allow for user email validation before activation.
- **User login:** Makes use of servlet container authentication.
- **Create/Edit Users:** Ability for administrators to create/edit user profiles.
- **Create/Edit Roles:** Ability for administrators create/edit roles.
- **Role Assignment:** Ability for administrators to assign users to roles.

Permissions Management

- **Extendable permissions API:** Allows custom portlets fine-grained permissions based on role definition or portal context.
- **Administrative interface:** Allows for fine-grained permissions assignments to roles at any time for any deployed portlet.

Content Management System

- **Full WebDAV support:** CMS store implements Jakarta Slide [http://jakarta.apache.org/slide/index.html] WebDAV system. Allows for native OS access to file store.
- **DB or Filesystem store support:** Configurable content store to either a filesystem or RDBMS.
- **Versioning support:** All content edited/created is autoversioned with a history of edits that can be viewed at any time.
- **Content Serving Search-engine-friendly URLS:** http://yourdomain/portal/default/index.html (Does not apply to portlet actions.)
- **No long portal URLS:** Serve binaries with simple urls. (http://domain/files/products.pdf)
- **Bundled HTML Portlet:** Allows for extra instances of static content from the CMS to be served under separate windows.
- **Directory Support:** create, move, delete, and copy entire directory trees.
- **File Functions:** create, move, copy, and delete files.
- **Custom error pages:** Error pages can be editted to suit the needs of your website - 404, invalid login pages.
- **Embedded directory-browser:** When copying, moving, deleting, or creating files, administrators can simply navigate the directory tree to find the collection they want to perform the action on.
- **Ease-of-use architecture:** All actions to be performed on files and folder are one mouse-click away.
- **Full-featured HTML editor:** HTML Editor contains WYWIWYG mode, preview functionality, and HTML source editting mode. HTML commands support tables, fonts, zooming, image and url linking, flash movie support, bulleted and numbered list, and dozens more.

- **Editor style-sheet support:** WYSIWYG editor displays current Portal style-sheet, for easy choosing of classes.

Message Boards

- **Instant reply:** Instant reply feature, makes for one-click replies to posts.
- **Post quoting:** Quote an existing topic and poster within a reply.
- **Flood control:** Prevents abuse of multiple posts withing a set configurable time-frame.
- **Category creation:** Create a category that contains forums within it.
- **Forum creation:** Create a forum and assign it to a specific category.
- **Forum modification:** Edit, move, delete forums.
- **Forum and category reordering:** Reorder categories and forums in the order you would like them to appear on the page.

Target Audience

Portlet developers or those wishing to implement/extend the JBoss Portal framework.

Acknowledgements

We would like to thank **all** the developers that participate in the JBoss Portal project effort.

Specifically,

1. Remy for his help with Tomcat configuration.
2. The Nodesk team that gave us our default theme.
3. Kev "kevs3d" Roast for supplying us with two working portlets that integrate esisting frameworks in to the portal: Sun JSF-RI and Spring MVC Portlet. These formed the base for our sample pack.
4. Swarn "sdhaliwal" Dhaliwal for supplying us with the Struts-Bridge, that will allow for existing struts applications to work with the Portal.

Contributions of any kind are always welcome, you can contribute by providing ideas, filling bug reports, producing some code, designing a theme, writing some documentation, etc... To report a bug please use our Jira system [http://jira.jboss.com].

1

JSR168 portlets

Thomas Heute `<theute@jboss.org>`

1.1. Introduction

The JSR 168 specification aims at defining porlets that can be used by any JSR168 portlet container also called portals. There are different portals out there with commercial and non-commercial licences. In this chapter we will briefly describe such portlets but for more details you should read the specifications available on the web.

As of today, JBoss portal is fully JSR168 1.0 compliant, that means that any JSR168 portlet will behave as it should inside the portal.

1.2. The basics

What is really important to know about such portlets is that when a page is displayed it is divided into two distincts parts, an action part on one portlet followed by rendering parts for every porlets displayed on a page. A portal just aggregates all the chunks of HTML rendered by the different portlets of a page.

1.2.1. Portal

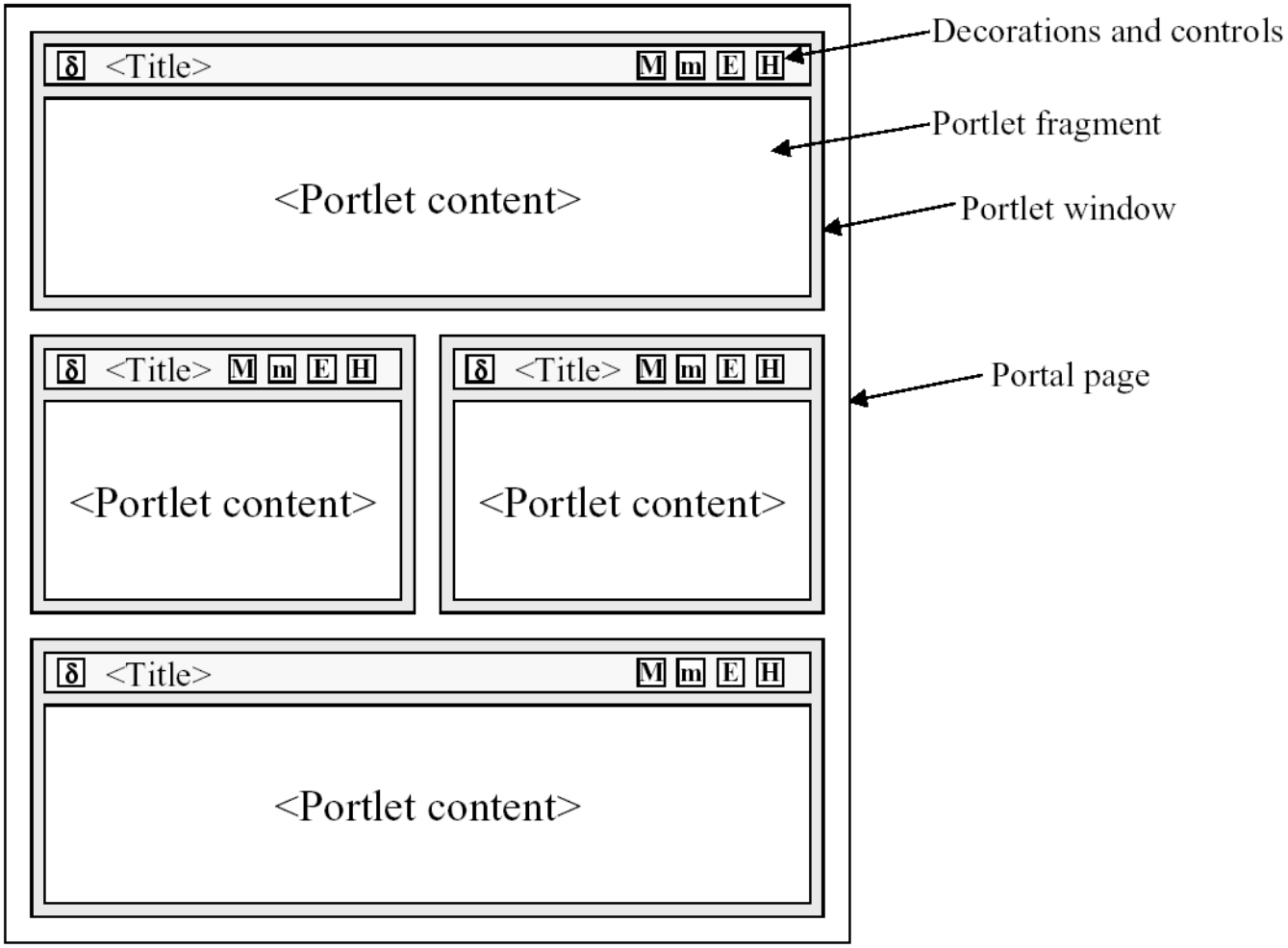
Before we even talk about portlets, let's talk about the container called portal.

A portal is basically a web application in which modules can be easily added or removed. We call those modules 'portlets'. A module can be as complex as a forum, a news management system or as simple as a text or text with images with no possible interaction.

On a single web page different portlets can appear at the same time.

1.2.2. Page composition

A portal can be seen as pages with different areas and inside areas, different windows and each window having one portlet.



1.2.3. Rendering modes

A porlet can have different view modes, three modes are defined by the specification but a portal can extends those modes.

2

XML descriptors

Thomas Heute `<theute@jboss.org>`

2.1. Introduction

To define your portals, you will need to create a few XML files in order to declare your portlet, portlet instances, windows, pages and then your portals.

2.2. /WEB-INF/jboss-portlet.xml

Here is a sample taken from our HelloWorldPortlet (packaged in helloworld.war that you can download here [http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossPortalSamples].):

```
<portlet-app>
   <portlet>
      <portlet-name>HelloWorldPortlet</portlet-name>
      <security></security>
   </portlet>
</portlet-app>
```

`portlet-name` must be the same as defined in `portlet.xml`.

(The security tag is explained in the security chapter of this documentation).

2.3. /WEB-INF/jboss-app.xml

This file is JBoss specific and identifies your application WAR file.

```
<jboss-app>
    <app-name>helloworld</app-name>
</jboss-app>
```

2.4. /WEB-INF/portlet.xml

This file is used to declare the portlets of your application:

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd /opt/SUNWps/dtd/portlet.xsd" v
```

```
   <portlet>
      <portlet-name>HelloWorldPortlet</portlet-name>
      <portlet-class>org.jboss.portlet.helloworld.HelloWorld</portlet-class>
      <supported-locale>en</supported-locale>
      <resource-bundle>Resource</resource-bundle>
      <supports>
         <mime-type>text/html</mime-type>
         <portlet-mode>VIEW</portlet-mode>
      </supports>
      <portlet-info>
         <title>My HelloWorld Portlet</title>
      </portlet-info>
   </portlet>
</portlet-app>
```

As you can see for any portlet you need to add a `portlet` tag, then you need to give extra information as follow:

- `portlet-name` a mandatory entry to give a name to this portlet
- `portlet-class` a mandatory entry to specify the class of the portlet
- `supported-locale` optional entries to specify the supported languages (two letter country code)
- `resource-bundle` an optional entry to specify the name of the resource bundle file (.properties)
- `init-param` optional entries to give parameters to the portlet
- `supports/mime-type` optional entries to specify all the supported mime-types
- `supports/portlet-mode` optional entries to specify all the supported modes, VIEW, HELP, EDIT and/or custom modes
- `portlet-info/title` optional entry to define a title to the portlet

2.5. portlet-instances.xml

After you defined all the portlets you need, you will need to define the instances that you will use.

```
<?xml version="1.0" standalone="yes"?>
<instances>
   <instance>
      <instance-name>HelloWorldPortletInstance</instance-name>
      <component-ref>HelloWorldPortlet</component-ref>
   </instance>
</instances>
```

Again the file is pretty simple, the `instances`tag includes `instance` elements with:

- `instance-name` required to specify the name of the instance
- `component-ref` required and has to be defined in the `portlet.xml`
- `preferences` optional to define preferences relevant to this instance of a portlet

2.6. *-pages.xml

You defined your portlets then your instances of portlet, now let's put them together on a page, to do so you can create files with `helloworld-pages.xml` and put them either in the `WEB-INF` of a war file or in the `deploy` directory of JBoss.

Here is the page for the forums module as example:

```
<pages>
    <portal-name>default</portal-name>
    <page>
        <page-name>samples</page-name>
        <window>
            <window-name>HelloWorldPortletWindow</window-name>
            <instance-ref>helloworld.HelloWorldPortlet.HelloWorldPortletInstance</instance-ref>
            <default>true</default>
            <region>center</region>
            <height>0</height>
        </window>
    </page>
</pages>
```

Again this is pretty straightforward. At first you need to define in which portal this page should belongs to (we will see later how to define a portal). Then you define the pages, you can define as many pages as you want, this is how a page is define:

- `page-name` required element to define the name of a page
- `window` required element to define a window
    - `window-name` required element to define the name of a window
    - `instance-ref` required element, it must refers to an existing portlet instance. Notice how the name is decomposed, first you have the application name as defined in `jboss-portlet.xml`, the name of the portlet (as defined in `portlet.xml`) another dot and finally the name of the portlet instance (as defined in portlet-instances.xml.
    - `default` optional element, it defines if this window should be the default window, and will occupy most of the space on the screen.
    - `region` required element, it defines where on the theme, the window should be displayed.
    - `height` required element, this integer defines where the window should be displayed compare to the others in a same region. The lower the number is, the higher in the theme it will be displayed. If two windows have the same height value then the windows will be randomly placed.

2.7. *-portal.xml

Used to define different portal instances within the container. The files `*-portal.xml` can be include into a war file (WEB-INF directory) or directly in the `deploy` directory of JBoss.

```
<?xml version="1.0" encoding="UTF-8"?>
<portal>
   <portal-name>default</portal-name>
   <supported-modes>
      <mode>VIEW</mode>
      <mode>EDIT</mode>
      <mode>HELP</mode>
   </supported-modes>
   <supported-window-states>
      <window-state>NORMAL</window-state>
      <window-state>MINIMIZED</window-state>
      <window-state>MAXIMIZED</window-state>
   </supported-window-states>
   <pages>
      <default-page>default</default-page>
      <page>
         <page-name>default</page-name>
         <window>
            <window-name>CMSPortletWindow</window-name>
            <instance-ref>portal.CMSPortlet.CMSPortletInstance</instance-ref>
            <default>true</default>
            <region>left</region>
            <height>0</height>
         </window>
         <window>
            <window-name>UserPortletWindow1</window-name>
            <instance-ref>portal.UserPortlet.UserPortletInstance</instance-ref>
            <region>left</region>
            <height>0</height>
         </window>
      </page>

      <page>
         <page-name>admin</page-name>
         <window>
            <window-name>UserPortletWindow2</window-name>
            <instance-ref>portal.UserPortlet.UserPortletInstance</instance-ref>
            <default>true</default>
            <region>left</region>
            <height>0</height>
         </window>
         <window>
            <window-name>GroupPortletWindow</window-name>
            <instance-ref>portal.GroupPortlet.GroupPortletInstance</instance-ref>
            <region>right</region>
            <height>1</height>
         </window>
         <window>
            <window-name>AdminCMSPortletWindow</window-name>
            <instance-ref>portal.AdminCMSPortlet.AdminCMSPortletInstance</instance-ref>
            <region>left</region>
            <height>4</height>
         </window>
      </page>
   </pages>
</portal>
```

This file is the descriptor of a portal, of course you can define several portals containing several pages. Here is how you define a portal:

- `portal-name` this is a required element to define the name of a portal.
- `supported-modes` all the supported modes at the portal level, you can then specify for each instance the suppor-

ted modes for a specific portlet.

- `supported-window-states` all the supported window states at the portal level. You can add your own window states here.
- `pages` to define the pages of your portal, the syntax is the same as in the `-pages.xml` files.

3

Portal urls

Julien Viet `<julien@jboss.org>`

Thomas Heute `<theute@jboss.org>`

3.1. Introduction

Most of the time portals use very complicated urls, however it is possible to setup entry points in the portal that follow simple patterns.

Each portal container can contain multiple portals and within a givin portal, windows are organized in pages, a page simply being a collection of windows associated to a name.

Before reading this chapter you must know how to define a page and a portal, you can refer to the chapter about XML descriptors to have a better understanding of those notions.

3.2. Acessing a portal

Each portal container can contains multiple portals, also there is one special portal which is the default portal, i.e the one used when no portal is specified in particular.

- "/", "/index.html" or "/portal/index.html" with no parameter, will point to the default page of the default portal.
- "/portal/portalname/" or "/portal/portalname/index.html" with no parameter will point to the default page of the portal `portalname`
- To access a file from the CMS, each portal should have a top directory with the name of the portal, then the path in the URL is the same as in the CMS. "/portal/portalname/foo.jpg" as URL will display the content of "/portalname/foo.jpg" in the CMS.

3.3. Accessing a page

It is possible to have multiple page per portal. As for portal there is a default page for a given portal. Once the portal has been selected, then a page must be used and all the windows present in that page will be rendered. The page selection mechanism is the following.

- If there is a target window in the URL, the page where this window resides is chosen. Usually these URLs are rendered by portlets to target themselves.
- If there is a "page" parameter then a page with that name is looked in the current portal, for intance the URL "/index.html?_id=page.portalname.pagename" will chose the page called `pagename` in the portal `portalname`.
- the default page for the current portal is used, for instance the URL "/index.html" will use the default page in the default portal.

4

Securing JBoss portlets

Thomas Heute `<theute@jboss.org>`

4.1. Introduction

JSR 168 specifications does not define any particular security implementation even though you can get the authenticated user and its role.

In JBoss portal, each portlet defines its own security model, and the portal gives an easy way to check if a user has a permission.

4.2. Defining the security model for your portlet

To define your security model, you need to edit the file `WEB-INF/jboss-portlet.xml`. First you need to define all kind of permissions you want, and how those permissions are related one to the other.

Let's take a small example, in your portlet you want to define two different permissions, `read` and `write`. Here is an example of security model:

```
<security>
   <model>
      <permission-description>
         <permission-name>write</permission-name>
         <description>Writing permission</description>
      </permission-description>
      <permission-description>
         <permission-name>read</permission-name>
         <description>Reading permission</description>
      </permission-description>
   </model>
</security>
```

This would be sufficient but if a user can write usually he can read as well. With the current model, we would need to add the role to both permission. There is another way, we could just specify that the `write` permission implies the `read` permission. To do so we just need to write:

```
<security>
   <model>
      <permission-description>
         <permission-name>write</permission-name>
         <description>Writing permission</description>
         <implies>read</implies>
      </permission-description>
      <permission-description>
         <permission-name>read</permission-name>
```

```
         <description>Reading permission</description>
      </permission-description>
   </model>
</security>
```

A permission can imply any number of permissions, just make sure you are not doing cycles (when a permission implies another that implies the first)

4.3. Giving permissions to roles

Once you defined what kind of permissions you want, you need to attribute roles to them, to do so you need to add a scheme to the model:

```
<security>
   <model>
      <permission-description>
         <permission-name>write</permission-name>
         <description>Writing permission</description>
         <implies>read</implies>
      </permission-description>
      <permission-description>
         <permission-name>read</permission-name>
         <description>Reading permission</description>
      </permission-description>
   </model>
   <scheme>
      <domain></domain>
      <item>
         <path>/</path>
         <permission>
            <permission-name>read</permission-name>
            <role-name>Users</role-name>
         </permission>
         <!-- For non logged users -->
         <permission>
            <permission-name>read</permission-name>
            <role-name></role-name>
         </permission>
         <permission>
            <permission-name>write</permission-name>
            <role-name>Admins</role-name>
         </permission>
      </item>
   </scheme>
</security>
```

Here we add the `read` permission to the `Users` role and anonymous users then the `write` permission to the `Admins` role. The `path` defines a scope on which the permissions will be defined. This will have different meanings for different portlets. For example the forums portlet uses a path to specify on which category or forum you want to apply the permissions (/mycategory/myforum for example)

4.4. Checking a permission inside your portlet

You can check a permission on a `JBossRenderRequest` or a `JBossActionRequest` using any of the `hasPermission` methods (see the API).

In our simple example, `req.hasPermission("write")` will check if the user accessing the website has the `write`

privilege.

5

Deploying Custom Themes & Layouts

Roy Russo `<roy at jboss dot org>`

5.1. Introduction

With the addition of the new Theme and Layout API it now simple to roll in your own custom layout and theme to change the look and feel of JBoss Portal. We took extra care to make it simple for you to implement your own custom layouts/themes and at the same time, separate the presentation from the business layer. This chapter will walk you through setting up and deploying your own custom layout and theme, making use of the this API.

Note

For the remainder of this chapter, a **Theme** is considered to be the CSS stylesheet that affects the "skin" of the layout. A **Layout** is the HTML markup used to control the placement of portlets on a page.

Themes and layouts are bundled together inside a war file with some additional XML descriptors and then dropped in the deploy directory of the portal. The example for this chapter can be downloaded from here [http://download.jboss.com/jbossportal/themes/myLayout.war.zip]. Once deployed, it will look like this:

5.2. Creating a layout

The first step is to create a layout WAR file with all the necessary xml descriptors, images, and stylesheets.

```
myLayout.war
|- simple-sample.css
|-/images
|-/layouts
  |- twoColumns.jsp
|-/WEB-INF
  |- jboss-web.xml
  |- portal-layouts.xml
  |- portal-themes.xml
```

The XML descriptors explained:

- **portal-layouts.xml** - Describes the layout name and files that should be accessed by the renderer. Here, our example is assigning a name to our layout and a reference to the file that should be used. In our example, we are only using one layout as such:

```
<layouts>
    <layout>
        <name>2ColumnLayout</name>
        <uri>/layouts/twoColumns.jsp</uri>
    </layout>
</layouts>
```

- **portal-themes.xml** - Describes the stylesheet that should be used for our layout.

```
<themes>
    <theme>
        <name>simple-sample</name>
        <link href="/simple-sample.css" title="" rel="stylesheet" type="text/css" media="screen" />
    </theme>
</themes>
```

- **jboss-web.xml** - Describes the context-root of our war file containing the theme and layout.

```
<jboss-web>
    <context-root>/myLayout</context-root>
</jboss-web>
```

The layout JSP explained:

Below we have our jsp that handles the positioning of the portlet windows in their assigned regions and also references the stylesheet we declared in portal-themes.xml above.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.d
<%@ taglib uri="/WEB-INF/theme/portal-layout.tld" prefix="p" %>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"><head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1"/>

<p:theme themeName='simple-sample' />

<body>

<!-- header table -->
<table class="headerTable">
<tr>
<td align="center">
<img src="/myLayout/images/logo.gif" />
</td>
</tr>
</table>

<!-- center table with columns -->
<table width="100%" bgcolor="#FFFFFF">
<tr>
<td class="leftColumn"><p:region regionName='left'/></td>
<td class="centerColumn"><p:region regionName='center'/></td>
</tr>
</table>

<!-- footer table - HTML ommited for brevity. -->

</body>
</html>
```

There are two tags here that are important to note:

- **<p:theme themeName='simple-sample' />** - Maps to the theme declared in portal-themes.xml
- **<p:region regionName='<regionName>'/>** - Defines regions where the assigned portlet windows will be places; left, center, right are acceptible values.

The actual *simple-sample.css* file can include any styles you have defined for your theme. It is advisable that you follow the JSR-168 specification [http://docs.jboss.org/jbportal/spec/portlet-1_0-fr-spec.pdf] in defining your

stylesheets.

5.3. Configuring JBoss Portal

All that is left to do now is to let the Portal know you will be using this new theme and layout combination. To do this, you will have to edit *$PORTAL_HOME/portal-core.war/WEB-INF/default-portal.xml*.

- **Setting the Layout:** You will have to modify the xml descriptor to point to the new layout you declared in *portal-layouts.xml*, using the layout name above as the value:

```
<property>
   <name>org.jboss.portal.property.layout</name>
   <value>2ColumnLayout</value>
</property>
```

- **Setting the Render Class:** Our example layout uses the *org.jboss.portal.property.renderSet.divRender* class. Describing the different render classes bundled in the distribution, is beyond the scope of this document. For now, edit the following as it is the class we are using:

```
<property>
   <name>org.jboss.portal.property.renderSet</name>
   <value>divRenderer</value>
</property>
```

Now all that is left to do, is to drop in this archive in to the /deploy directory where JBoss Portal is deployed and navigate to the homepage to see it.