JBoss Portal

Specifications guide

0.1

Table of Contents

Target Audience

Developers of portlets for JBoss portal should read this document

Acknowledgements

We would like to thank all developers that participate in the JBoss Portal project effort, Remy for his help and the Nodesk team that gave us that nice theme.

1

JSR168 portlets

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

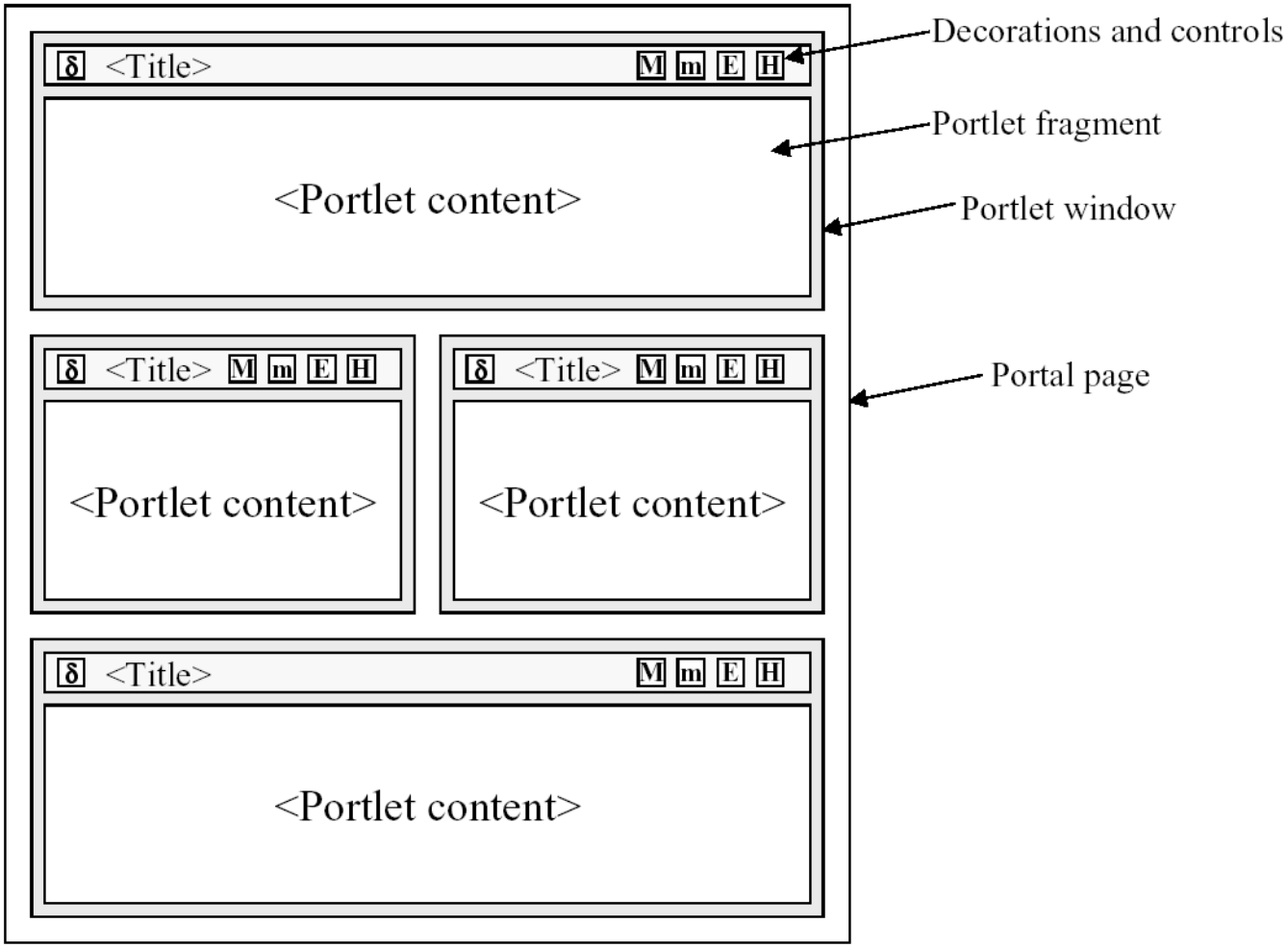
Before we even talk about portlets, let's talk about the container called portal.

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, even though only one at a time can be displayed in its maximized mode (explained later).

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 need to create few XML files in order to declare your portlet, portlet instances, windows, pages and then your portals.

2.2. /WEB-INF/portlet.xml

This file is used to declare the portlets of your application, following is an example of such a file

```
<?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" version="1.0">
   <portlet>
      <portlet-name>UserPortlet</portlet-name>
      <portlet-class>org.jboss.nukes.core.portlet.user.UserPortlet</portlet-class>
      <supported-locale>en</supported-locale>
      <supported-locale>fr</supported-locale>
      <resource-bundle>Resource</resource-bundle>
      <supports>
         <mime-type>text/html</mime-type>
         <portlet-mode>VIEW</portlet-mode>
      </supports>
      <portlet-info>
         <title>User portlet</title>
      </portlet-info>
   </portlet>
   <portlet>
      <portlet-name>CMSPortlet</portlet-name>
      <portlet-class>org.jboss.nukes.core.portlet.cms.CMSPortlet</portlet-class>
      <init-param>
         <description>Content Repository Type: (webdav|http)</description>
         <name>repository</name>
         <value>webdav</value>
      </init-param>
      <init-param>
         <description>WebDAV server username</description>
         <name>username</name>
         <value>root</value>
      </init-param>
      <init-param>
         <description>WebDAV server password</description>
         <name>password</name>
```

```
         <value>root</value>
      </init-param>
      <supported-locale>en</supported-locale>
      <resource-bundle>Resource</resource-bundle>
      <supports>
         <mime-type>text/html</mime-type>
         <portlet-mode>VIEW</portlet-mode>
         <portlet-mode>HELP</portlet-mode>
      </supports>
      <portlet-info>
         <title>CMS 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

2.3. 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>UserPortletInstance</instance-name>
      <component-name>UserPortlet</component-name>
      <security>
         <rules>
            <rule group="Admins" patterns=".*:.*:.*" level="admin"/>
            <rule group="Users" patterns=".*:.*:.*" level="read"/>
         </rules>
      </security>
   </instance>
   <instance>
      <instance-name>CMSPortletInstance</instance-name>
      <component-name>CMSPortlet</component-name>
   </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-name` required and has to be defined in the `portlet.xml`
- `security` that's where you define the security rules for this instance of portlet, please refer to the security chapter for more information

2.4. *-page.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 `-page.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>forums</page-name>
      <window>
         <window-name>ForumsPortletWindow</window-name>
         <instance-ref>/portal-forums.ForumsPortlet.ForumsPortletInstance</instance-ref>
         <default>true</default>
         <region>left</region>
         <height>0</height>
      </window>
   </page>
</pages>
```

Again this is pretty straightforward. At first you need to defin 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 page 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 name of the web application context followed by a dot, 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, only the documentation of the theme can define what region exists for this specific theme.

- `height` required element, this integer defines where the window should be displayed compare to the others in a same region. The higher 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.

2.5. *-portal.xml

Those files are used to define different portals. 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>/nukes.CMSPortlet.CMSPortletInstance</instance-ref>
            <default>true</default>
            <region>left</region>
            <height>0</height>
         </window>
         <window>
            <window-name>UserPortletWindow1</window-name>
            <instance-ref>/nukes.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>/nukes.UserPortlet.UserPortletInstance</instance-ref>
            <default>true</default>
            <region>left</region>
            <height>0</height>
         </window>
         <window>
            <window-name>GroupPortletWindow</window-name>
            <instance-ref>/nukes.GroupPortlet.GroupPortletInstance</instance-ref>
            <region>right</region>
            <height>1</height>
         </window>
         <window>
            <window-name>AdminCMSPortletWindow</window-name>
            <instance-ref>/nukes.AdminCMSPortlet.AdminCMSPortletInstance</instance-ref>
            <region>left</region>
```

```
            <height>4</height>
         </window>
      </page>

      <page>
         <page-name>test</page-name>
         <window>
            <window-name>DefaultPortletWindow</window-name>
            <instance-ref>/nukes.DefaultPortlet.DefaultPorltetInstance</instance-ref>
            <default>true</default>
            <region>left</region>
            <height>0</height>
         </window>
         <window>
            <window-name>TestPortletWindow</window-name>
            <instance-ref>/nukes.TestPortlet.TestPortletInstance</instance-ref>
            <region>left</region>
            <height>1</height>
         </window>
         <window>
            <window-name>PreferencesPortletWindow</window-name>
            <instance-ref>/nukes.PreferencesPortlet.PreferencesPortletInstance</instance-ref>
            <region>left</region>
            <height>2</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 supported 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 `-page.xml` files.

3

Portal urls

Julien Viet <julien@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.

The following examples show you how the selection is done.

- ```
  ""
  "/"
  "/index.html" with no parameters
  "/portal"
  "/portal/"
  => default / "/index.html"
  ```

- ```
  "/portal/blah.html"
  => default / "/blah.html"
  ```

- ```
  "/portal/another/"
  => another / "/index.html"
  ```

- ```
  "/portal/another/blah.html"
  => another / "/blah.html"
  ```

- ```
  "files"
  "/files/"
  => default / "/index.html"
  ```

- ```
  "/files/blah.html"
  => default / "/blah.html"
  ```

- any other URL, like "/blah.html" is served from the another servlet (war files or security for instance)

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 th URL "/index.html?page=admin" will chose the admin page in the default portal.

- 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

Bring security to JBoss portlets

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 rules but the portal gives an easy way to implement it using the AuthorizationRealm obtained by NukesRenderRequest or NukesActionRequest.

4.2. Defining rules

The security mechanism works with rules, a rule consist in three elements:

- Role
  The role that you want to apply the rule to
- Context
  The context in which the rule will be applied
- Level of privilege
  The credential given to the role in the context specified

4.2.1. Setting up a context

A context is defined by the portlet developer, it is represented by an array of Strings. For example, the forum portlet is divided into sections and each sections are divided into forums. Typically a webmaster will want to grant premissions on whole sections and/or just some forums. `{"CategoryName", "forumName"}` could be a context for the forum called `forumName` of the category `CAtegoryName`. To define a context for a whole section, a user can use regular expression like `{"CategoryName", ".*"}` and it would define context on the whole category. Again this is just a matter of portlet developer choice. The meaning of a context for a particular portlet should be documented in the portlet guide.

4.2.2. Privileges

The rights to give are the same for any portlet, we often want to add write or read access, JBoss portal defines 9 levels of privileges that are linked together. In the default configuration, a role has also the rights of roles with a lower number. For example, the level 8 has the rights of level 0 to 7.

All those levels have a name for their common usage

- NONE
  No restrict access to a role in a context
- OVERVIEW
  Can be used to give access to a summary of a story but not to the whole story
- READ
  This gives reading access to a role
- COMMENT
  This gives the ability to add comments to a role if the portlet has such a feature
- MODERATE
  This should be used to add the possibility to add or remove some data entered by others
- EDIT
  This should be used to allow a role to edit data on the portal
- ADD
  This should be used to allow a role to add content to the portal
- DELETE
  Gives the right to delete content from the portal
- ADMIN
  This is the ultimate level, this gives rights to anything

Advanced configuration

As specified earlier, by default someone who has `COMMENT` access level also have `READ`, `OVERVIEW` and `NONE` acces levels. For even more flexibility, the portal allows you to change how the levels are dependant one to the other. This is done by modifying the authorization matrix in `jboss.nukes.core.security.Level`. You can then define which access level implies which other.

4.2.3. Define rules

To define rules on a portlet, you need to adapt the nukes-instance.xml file. Inside a `instance` tag, you should add a `security` tag. This tag contains all the rules you want. For example, the following tags would give reading access on everything for the `Users` role and admin access to the `Admins` role. Of course you can define a finer grain by specifying the correct pattern.

```
<security>
  <rules>
    <rule role="Admins" patterns="" level="admin"/>
    <rule role="Users" patterns="" level="read"/>
  </rules>
</security>
```

4.3. Testing security access

A portlet developer will need to make sure all the operations are correctly checked for credentials. To programatically do so, he will need to get a `AuthorizationRealm` from the `NukesRequest` then use this object to check against the user's roles.

```
AuthorizationRealm authorizationRealm = nukesRequest.getAuthorizationRealm();
boolean readAuthorized = authorizationRealm.isAuthorized("Users",
                          new String[] {"My Category", "My Forum"},
                          org.jboss.nukes.core.security.Level.ACCESS_READ);
```

Note

If someone didn't logged in, he is part of the `RuleAuthorizationRealm.ANONYMOUS_ROLE_SET` role.

Getting the list of roles names is not always convenient, to simplify this task, the class `org.jboss.nukes.core.portlet.PortletHelper` has the following static method `public static boolean isAuthorized(AuthorizationRealm authRealm, User user, String[] test, Level level)`, this method is even more convenient as you get the user from the `NukesRequest`. A call to this method will look like:

```
PortletHelper.isAuthorized(nukesRequest.getAuthorizationRealm(),
                          nukesRequest.getUser(),
                          new String[] {"My category", "My forum"},
                          Level.ACCESS_READ)
```

5

Deploying Custom Portlets

Roy Russo `<roy at jboss dot org>`

5.1. Introduction

This section described the steps to incorporate your own portlets in to JBoss Portal.

5.2. Assumptions

Assumptions:

1. You already have a portlet war named helloworld.war
2. The WAR file has a standard directory structure:

```
/helloworld.war
   /WEB-INF
      /classes
      /lib
      /portlet.xml
      /web.xml
   /META-INF
```

3. Your `porlet.xml` looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
version="1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
<portlet>
    <portlet-name>HelloWorldPortlet</portlet-name>
    <portlet-class>com.myapp.portlet.HelloWorldPortlet</portlet-class>
    <expiration-cache>0</expiration-cache>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>help</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>Hello World Portlet</title>
```

```
    </portlet-info>
  </portlet>
</portlet-app>
```

5.3. Adding the XML Descriptors

There are two files you will need to add under `/WEB-INF` for your portlet to work in Jboss Portal. The first is a `helloworld-pages.xml`. The portal will scan this file to find out which portal instance to target and what page name it will be in. In this case, we can access our portlet by going to `http://localhost:8080/portal/index.html?page=hello`

```
<pages>
   <portal-name>default</portal-name>
   <page>
      <page-name>hello</page-name>
      <window>
         <window-name>HelloWorldPortletWindow</window-name>
         <instance-ref>/portal-hello.HelloWorldPortlet.HelloWorldPortletInstance</instance-ref>
         <default>true</default>
         <region>user1</region>
         <height>0</height>
      </window>
   </page>
</pages>
```

The second file needed under `/WEB-INF` is a `portlet-instances.xml`. This file maps the portlet name in your `portlet.xml` file to the portlet instance in your `helloworld-pages.xml`.

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

Now add these two files to your war and you should be able to access the helloworld portlet by going to `http://localhost:8080/portal/index.html?page=hello`.