Groovy Bridge v.1.0.3

Project Documentation

Apache Software Foundation 15 July 2007

Table of Contents

1.1 Jetspeed Groovy Portlet Guide

Jetspeed Groovy Portlet Guide

This guide provides a tutorial for creating a Groovy portlet with full-featured portlet modes.

1. The Portlet Class

Create the file HelloGroovy.groovy in a directory called groovy-simplest/WEB-INF/classes:

```
import javax.portlet.GenericPortlet;
import javax.portlet.PortletContext;
import javax.portlet.PortletRequestDispatcher;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;
import javax.portlet.PortletPreferences;

public class HelloGroovy extends GenericPortlet
{
    public void doView(RenderRequest request, RenderResponse response)
    {
        PortletContext context = getPortletContext();
        PortletRequestDispatcher rd =
            context.getRequestDispatcher("/WEB-INF/view/hello-groovy-view.jsp");
        rd.include(request, response);
    }

    public void doEdit(RenderRequest request, RenderResponse response)
    {
        PortletContext context = getPortletContext();
        PortletRequestDispatcher rd =
            context.getRequestDispatcher("/WEB-INF/view/hello-groovy-edit.jsp");
        rd.include(request, response);
    }

    public void doHelp(RenderRequest request, RenderResponse response)
    {
        PortletContext context = getPortletContext();
        PortletRequestDispatcher rd =
            context.getRequestDispatcher("/WEB-INF/view/hello-groovy-help.html");
        rd.include(request,response);
    }

    public void processAction(ActionRequest request, ActionResponse response)
    {
        String message = request.getParameter("message");

        if (null != message && !"".equals(message)) {
            PortletPreferences prefs = request.getPreferences();
```

```
            prefs.setValue("message", message);
            prefs.store();
        }
    }
}
```

You don't have to compile the source because it's groovy.

2. The portlet.xml

Create the file portlet.xml in the groovy-simplest/WEB-INF directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app id="velocitysimplest" version="1.0">
  <portlet id="HelloGroovy">
    <portlet-name>HelloGroovy</portlet-name>
    <display-name>Hello Groovy Display Name</display-name>
    <portlet-class>org.apache.portals.bridges.groovy.GroovyPortlet</portlet-class>
    <init-param>
      <name>script-source</name>
      <!-- Note: You can set script source in three ways.
                 The first is to use relative path uri,
                 the second is to use file: url,
                 and the last is to classpath: uri -->
      <!--
      <value>/WEB-INF/groovy/HelloGroovy.groovy</value>
      <value>file:/C:/Program Files/Apache Software Foundation/Tomcat
5.5/webapps/demo/WEB-INF/groovy/HelloGroovy.groovy</value>
      -->
      <value>classpath:HelloGroovy.groovy</value>
    </init-param>
    <!-- If auto-refresh is true, then a modification of script source applies
instantly. -->
    <init-param>
      <name>auto-refresh</name>
      <value>true</value>
    </init-param>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>VIEW</portlet-mode>
      <portlet-mode>EDIT</portlet-mode>
      <portlet-mode>HELP</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>Hello Groovy Title</title>
      <short-title>Hello Groovy Short Title</short-title>
    </portlet-info>
  </portlet>
</portlet-app>
```

3. The web.xml

You don't have to add any special tags for this simple example, but you can add some tags for supporting Groovlet or Groovy template as a view page. Please see the groovy documentation for those.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
                         "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Groovy Simplest</display-name>
  <description>The world's simplest Groovy portlet</description>

</web-app>
```

4. The View pages for view, edit and help mode

Create the hello-groovy-view.jsp file in the groovy-simplest/WEB-INF/view directory. Put whatever content you desire in it. Here is an example:

```
<%@ page session="false" %>
<%@ page import="javax.portlet.*"%>
<%@ taglib uri='/WEB-INF/portlet.tld' prefix='portlet'%>

<portlet:defineObjects/>

<%
String message = renderRequest.getPreferences().getValue("message", "Hello,
Groovy!");
%>

<h1><%=message%>!</h1>
```

Create the hello-groovy-edit.jsp file in the groovy-simplest/WEB-INF/view directory. Put whatever content you desire in it. Here is an example:

```
<%@ page session="false" %>
<%@ page import="javax.portlet.*"%>
<%@ taglib uri='/WEB-INF/portlet.tld' prefix='portlet'%>

<portlet:defineObjects/>

<%
String message = renderRequest.getPreferences().getValue("message", "Hello,
Groovy!");
%>

<form method="post" action="<portlet:actionURL/>">
    Message: <input type="text" name="message" value="<%=message%>">
    <input type="submit" value="Submit">
</form>
```

Last, create the hello-groovy-help.html file in the groovy-simplest/WEB-INF/view directory. Put whatever content you desire in it. Here is an example:

```
<H1>Hello Groovy Help</H1>
<HR>

<P>Groovy Portlet support rapid portlet application development.</P>
```

5. The Dependency JARs

Copy the portals-bridges-groovy-1.0.jar, groovy-1.0.jar, antlr-2.7.5.jar, and asm-2.2.jar to the groovy-simplest/WEB-INF/lib directory. IMPORTANT: Do NOT put the portlet-api-1.0.jar in the war file. If you have already built Jetspeed some of the jars should be in your Maven repository. If so executing these commands in the lib directory will set up the dependencies for you.

```
ln -s
~/.maven/repository/org.apache.portals.bridges/jars/portals-bridges-groovy-1.0.jar
```

And, if you download from http://groovy.codehaus.org and install groovy, you can find groovy-1.0.jar, antlr-2.7.5.jar, and asm-2.2.jar in the lib directory under groovy home directory. Also, copy the portlet.tld to the groovy-simplest/WEB-INF directory. You can find the portlet.tld file in jetspeed-2/src/webapps/WEB-INF/ source directory. Or you can copy that from the WEB-INF/ directory of the demo portlet.

6. The WAR file

From the directory groovy-simplest combine the files above into a war file using the command,

```
jar cvf ../groovy-simplest.war .
```

7. Deploy the WAR file

Copy the war file to `$CATALINA_HOME/webapps/jetspeed/WEB-INF/deploy`. Jetspeed-2 will deploy the webapp.

8. The PSML

Create the PSML page using the Jetspeed portlet chooser. Login and click on the edit page icon. Your user must have the permission to edit pages. The user `admin` password `admin` has permission to edit all pages.

9. Additional Notes

- You can make the script source simpler than Java. See the groovy documentation.
- GroovyPortlet instantiates a groovy-scripted portlet instance just like any Java portlet, and so you can use any techniques used in Java portlet programming. For example, your groovy script portlet can extend org.apache.portals.bridges.common.GenericServletPortlet to simplify implementation.
- In this example, JSP and JSTL is used for view pages. However, you can use other technologies such as Velocity, Groovlet or Groovy template.
- If you use Groovlet or Groovy template, a solution for getting renderRequest and renderResponse can be like this:

```
def renderRequest = request.getAttribute("javax.portlet.request")
def renderResponse = request.getAttribute("javax.portlet.response")
```

10. See Also

You can write preferences validator with groovy, and you can write a groovy script portlet supporting header phase (pre-286 feature).

- Jetspeed Groovy PreferencesValidator Guide
- Jetspeed Groovy Portlet with Header Phase Support Guide

1.2 Jetspeed Groovy PreferencesValidator Guide

Jetspeed Groovy PreferencesValidator Guide

This guide provides a tutorial for creating a Groovy Preferences Validator.

1. The Groovy PreferencesValidator Class

Create the file HelloGroovyValidator.groovy in a directory called groovy-simplest/WEB-INF/classes:

```
import javax.portlet.PortletPreferences;
import javax.portlet.PreferencesValidator;
import javax.portlet.ValidatorException;

public class HelloGroovyValidator implements PreferencesValidator
{
    public void validate(PortletPreferences preferences)
    {
        String message = preferences.getValue("message", null);

        if (message == null || "".equals(message.trim()))
        {
            def failedKeys = [ "message" ];
            throw new ValidatorException("message must be set!", failedKeys);
        }
    }
}
```

2. The portlet.xml

Edit the file portlet.xml in the groovy-simplest/WEB-INF directory to add Groovy PreferencesValidator. In this case, <portlet-preferences> element is added to the previous example.

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app id="velocitysimplest" version="1.0">
  <portlet id="HelloGroovy">
    <portlet-name>HelloGroovy</portlet-name>
    <display-name>Hello Groovy Display Name</display-name>
    <portlet-class>org.apache.portals.bridges.groovy.GroovyPortlet</portlet-class>
    <init-param>
      <name>script-source</name>
      <!-- Note: You can set script source in three ways.
                 The first is to use relative path uri,
                 the second is to use file: url,
                 and the last is to classpath: uri -->
      <!--
```

```
      <value>/WEB-INF/groovy/HelloGroovy.groovy</value>
      <value>file:/C:/Program Files/Apache Software Foundation/Tomcat
5.5/webapps/demo/WEB-INF/groovy/HelloGroovy.groovy</value>
      -->
      <value>classpath:HelloGroovy.groovy</value>
    </init-param>
    <!-- If auto-refresh is true, then a modification of script source applies
instantly. -->
    <init-param>
      <name>auto-refresh</name>
      <value>true</value>
    </init-param>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>VIEW</portlet-mode>
      <portlet-mode>EDIT</portlet-mode>
      <portlet-mode>HELP</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>Hello Groovy Title</title>
      <short-title>Hello Groovy Short Title</short-title>
    </portlet-info>
    <portlet-preferences>
      <preference>
        <name>validator-script-source</name>
        <!-- Note: You can set script source in two ways.
                   The first is to use classpath: path uri,
                   and the other is to file: url.
                   Note that relative path uri is not supported for preferences
validator. -->
        <value>classpath:HelloGroovyValidator.groovy</value>
        <!--
        <value>file:/C:/Program Files/Apache Software Foundation/Tomcat
5.5/webapps/demo/WEB-INF/groovy/HelloGroovyValidator.groovy</value>
        -->
        <read-only>true</read-only>
      </preference>
      <preference>
        <name>validator-auto-refresh</name>
        <!-- If validator-auto-refresh is true,
             then a modification of script source applies instantly. -->
        <value>true</value>
        <read-only>true</read-only>
      </preference>
<preferences-validator>org.apache.portals.bridges.groovy.GroovyPreferencesValidator</preferences-validato
    </portlet-preferences>
  </portlet>
</portlet-app>
```

3. How to handle validation exception

If you set a preferences validator, you should handle a validation exception in your portlet code like the following example. The HelloGroovy.groovy in the previous example now handles validation exception during PortletPreferences.store() call.

```
    public void processAction(ActionRequest request, ActionResponse response)
    {
        String message = request.getParameter("message");

        PortletPreferences prefs = request.getPreferences();
        prefs.setValue("message", message);

        try {
            prefs.store();
        } catch (ValidatorException e) {
            // send this error information to the rendering phase.
            response.setRenderParameter("errorMessage", e.getMessage());
        }
    }
```

4. Additional Notes

In groovy, you can use powerful regular expressions like the following example:

```
    public void validate(PortletPreferences preferences)
    {
        // Let's assume that we have a preference for an email address.
        String email = preferences.getValue("email", "");

        if (!(email ==~
/^[a-zA-Z][\w\.-]*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.]*[a-zA-Z]$/))
        {
            def failedKeys = [ "email" ];
            throw new ValidatorException("message must be set!", failedKeys);
        }
    }
```

1.3 Jetspeed Groovy Portlet with Header Phase Support Guide

Jetspeed Groovy Portlet with Header Phase Support Guide

Jetspeed-2 provides an interface for a portlet to be able to support the pre-286 header phase. If you want your portlet to support this, just use GroovyPortletHeaderPhaseSupport class instead of GroovyPortlet, and implement SupportsHeaderPhase interface.

1. The Portlet Class supporting header phase

Create the file HelloGroovyHeaderSupport.groovy in a directory called groovy-simplest/WEB-INF/classes:

```
import javax.portlet.GenericPortlet;
import javax.portlet.PortletContext;
import javax.portlet.PortletRequestDispatcher;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;
import javax.portlet.PortletPreferences;
import javax.portlet.ValidatorException;

import org.apache.jetspeed.headerresource.HeaderResource;
import org.apache.jetspeed.portlet.PortletHeaderRequest;
import org.apache.jetspeed.portlet.PortletHeaderResponse;
import org.apache.jetspeed.portlet.SupportsHeaderPhase;

public class HelloGroovyHeaderSupport extends GenericPortlet implements
SupportsHeaderPhase
{
    public void doHeader(PortletHeaderRequest request, PortletHeaderResponse
response)
    {
        // use header resource component to ensure header logic is included only
once
        HeaderResource headerResource = response.getHeaderResource();
        headerResource.dojoEnable();

        headerResource.dojoAddCoreLibraryRequire( "dojo.lang.*" );
        headerResource.dojoAddCoreLibraryRequire( "dojo.event.*" );
        headerResource.dojoAddCoreLibraryRequire( "dojo.io.*" );
        headerResource.dojoAddCoreLibraryRequire( "dojo.widget.*" );
        headerResource.dojoAddCoreLibraryRequire( "dojo.widget.Button" );
     }

    public void doView(RenderRequest request, RenderResponse response)
    {
        response.setContentType("text/html");
```

```
        // Let's put a dojo widget button for simplicity here.
        response.getWriter().println """
            <button widgetId="helloGroovyButton" dojoType="Button"
onclick="alert('Hello, Groovy');">
            Say Hello
            </button>
        """
    }

}
```

For simplicity, in the above example, the doView() method just writes a simple HTML fragment to show DOJO button widget.

2. The portlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app id="velocitysimplest" version="1.0">
  <portlet id="HelloGroovyHeaderSupport">
    <portlet-name>HelloGroovyHeaderSupport</portlet-name>
    <display-name>Hello Groovy with Header Support Display Name</display-name>
<portlet-class>org.apache.portals.bridges.groovy.GroovyPortletHeaderPhaseSupport</portlet-class>
    <init-param>
      <name>script-source</name>
      <!-- Note: You can set script source in three ways.
                 The first is to use relative path uri,
                 the second is to use file: url,
                 and the last is to classpath: uri -->
      <!--
      <value>/WEB-INF/groovy/HelloGroovy.groovy</value>
      <value>file:/C:/Program Files/Apache Software Foundation/Tomcat
5.5/webapps/demo/WEB-INF/groovy/HelloGroovy.groovy</value>
      -->
      <value>classpath:HelloGroovyHeaderSupport.groovy</value>
    </init-param>
    <!-- If auto-refresh is true, then a modification of script source applies
instantly. -->
    <init-param>
      <name>auto-refresh</name>
      <value>true</value>
    </init-param>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>VIEW</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
      <title>Hello Groovy with Header Support Title</title>
      <short-title>Hello Groovy with Header Support Short Title</short-title>
    </portlet-info>
  </portlet>
</portlet-app>
```

The <portlet-class> was replaced with 'org.apache.portals.bridges.groovy.GroovyPortletHeaderPhaseSupport' instead of 'org.apache.portals.bridges.groovy.GroovyPortlet' to support header phase.