Creating Groundwork Portlets -- A Quick developer guide

0.5

REVISION HISTORY

Date	Version	Description	Author
06th Dec 2010	0.1	Creating Groundwork Portlets -- A Quick developer guide	Tousif, Manish
09th Dec 2010	0.2	Added Dependencies, Packaging & Deployment and Appendix. Updated Interval rendering and JavaBean class sections.	Tousif, Manish
Dec. 10, 2010	0.3	Interval rendering, build	Roger Ruttimann
Jan 13, 2011	0.4	Added class & sequeunce diagrams. Also added notes for webservice authentication and RTMM info.	Arul Shanmugam
May, 2011	0.5	Final review before distribution	Roger Ruttimann

Table of Contents

1 Introduction

Groundwork Open source is one of the best in class industry standard open source network monitoring tool. This tool is a web based application developed on top of JBoss Portal server and uses a bunch of open source tools and frameworks.

This document attempts to layout best practices to be used for developing Portlets using Groundwork platform in conjunction with ICEFaces. This document also provides a sample Portlet as a reference to java developers for helping them to quick start Portlet development.

This document describes all the necessary steps for creating a custom portlet. It is a step by step portlet development guide that explains

- The classes/methods required for implementation
- How to create jspx / facelet
- How to implement Interval rendering
- Create / Edit Preferences in Portlet
- Some general configurations required to create portlet

1.1 Purpose

The purpose of this document is to guide java developers in developing custom Portlet on JBoss portal Server.

1.2 Target Audience

This document is for java developers who want to develop a custom Portlet and want to deploy it on JBoss portal Server. The step-by-step guide helps to learn the basics of Portlet development and deployment on Groundwork Monitor.

This document assumes that the reader is a java developer and has enough knowledge of java, j2ee technology.

2 Developing Custom Portlet

For creating custom Portlet in Groundwork we require two java classes and one jspx file. Following is the list of all those files required to create a custom Portlet.

- Java bean class
- Portlet class
- Jspx file

2.1 JavaBean class

This java class is a plain old java bean containing the properties and setter, getter methods in it. This class is a serialized class, may implement serialization and can have one no argument public constructor, which is optional. JavaBean is having one of the three scopes (request, session, application) depending upon the requirement of the application. These scopes of bean must be defined in faces-config.xml

Now let's develop a simple Portlet, configure and deploy it on JBoss portal server. Here we are going to create a Portlet which shows the Default HostGroup and List of all Host which comes under that HostGroup. This tutorial also demonstrates the use of Edit Preferences and Interval Rendering. You will find the details of Edit Preferences and Interval Rendering in its respective section.

Source Code: DemoBean.java

In DemoBean.java class, we are using Groundwork Web Services for getting the Host list under particular HostGroup by using WSHost API.

We are using apache axis 1.x to implement web service clients. We have generate java client stubs using wsdl2java' tool and use those stubs to implement actual operations.

URL of web service server is stored in the DemoBean.java's *WS_PATH variable.*

WS_PATH = "http://localhost:8080/foundation-webapp/services/ "

Note: Since Groundwork Monitor 6.4, all webservices are protected and require user/password authentication to access. This is configurable and is specified in the /usr/local/groundwork/config/ws_client.properties**.** Authentication request code is in the new collage-api-3.0.xxx.jar.

DemoBean=>hostLocator():-This method is responsible for creating the "host" (wshost) web service locator object and setting end point web service address(http://localhost:8080/foundation-webapp/services/wshost) .

```
public static WSHostServiceLocator hostLocator() {
        if (hostLocator == null) {
            hostLocator = new WSHostServiceLocator();
            try {
                hostLocator.setEndpointAddress(
                        DemoBean.FOUNDATION_END_POINT_HOST,
                        DemoBean.WS_PATH
                                + DemoBean.FOUNDATION_END_POINT_HOST);
```

```
            } catch (Exception exc) {
                // logger.error(exc.getMessage());
            } // end try/catch
        }
        return hostLocator;
    }
```

2.2 Portlet class

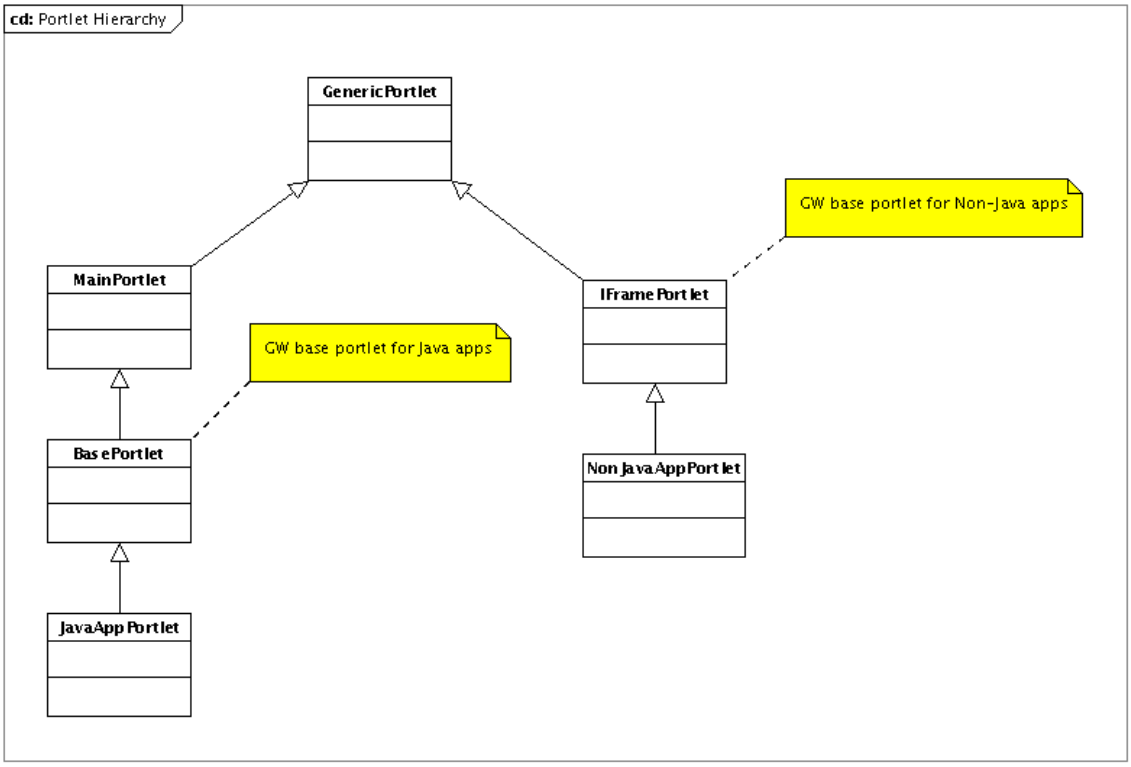
The Portlet interface is used by the portlet container to invoke the portlets. Every portlet has to implement this interface, either by directly implementing it, or by using an existing class implementing the Portlet interface.
A portlet is a Java technology-based web component. It is managed by the portlet container and processes requests and generates dynamic content as response. Portlets are used by portals as pluggable user interface components.

The portlet container instantiates portlets, manages their lifecycle and invokes them to process requests. The lifecycle consists of:

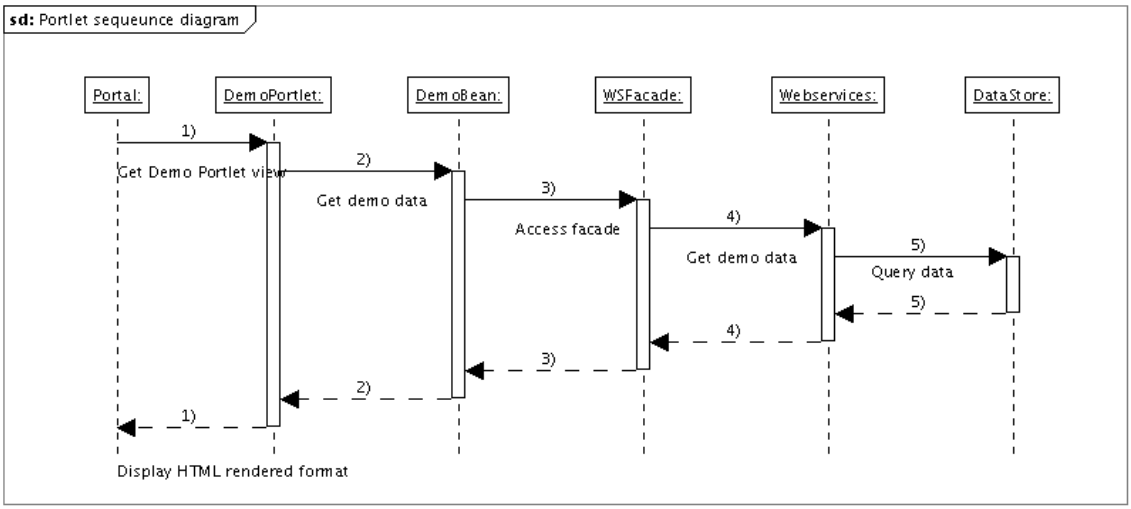
- initializing the portlet using the init method
- request processing
- taking the portlet out of service using the destroy method

This java class is a Portlet class which is a sub class of BasePortlet class. Portlet class diagram and hierarchy is show below. BasePortlet class is a Groundwork specific Portlet class which is a direct sub-class of MainPortlet.
(TheMainPortlet is the entry point for ICEfaces-based portlets. The goal is to set up the environment as required and then dispatch the request to the MainServlet.), which is an icefaces specific Portlet class. For wrapping external non-java applications like NMS cacti, weathermap etc, with the portlet, extend the class from IframePortlet. After extending BasePortlet, we can override the life-cycle methods specified in Portlet Specification. Following is the list of some important methods available to override.

- `doView()`
- `doEdit()`
- `processAction()`



The following sequence diagram illustrates how a data is fetched from the datastore and displayed in the portal.



Source Code: DemoPortlet.java

2.3 jspx File

The jspx page is used to create a UI of custom Portlet. This file is XTHML complains file and contains icefaces custom tags. We use icefaces specific tags to render the data on UI which is populated by JavaBean class. All the icefaces tags should contains in between <`ice:portlet>` and `</ice:portlet>` facelet tag.

Source Code: demoPortlet.jspx

1.1 Other configuration settings

Followings are the other configuration settings we require to develop a custom Portlet for Groundwork. These settings are important and without it the Portlet will not deployed.

1) Add the JavaBean class reference in faces-config.xml file with the scope.

```
<managed-bean>
<description>Demo bean</description>
<managed-bean-name>demoBean</managed-bean-name>
<managed-bean-
class>com.groundworkopensource.portal.statusviewer.bean.DemoBean</managed
-bean-class>
<managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

2) Add the entry of Portlet class in portlet.xml file

```
<portlet>
<portlet-name>DemoPortlet</portlet-name>
<display-name>Demo Portlet</display-name>
<description>
Demo Portlet
</description>
<portlet-class>
com.groundworkopensource.portal.statusviewer.portlet.DemoPortlet
</portlet-class>
<supports>
<mime-type>text/html</mime-type>
<portlet-mode>view</portlet-mode>
<portlet-mode>edit</portlet-mode>
</supports>
<supported-locale>de</supported-locale>
<supported-locale>fr</supported-locale>
<supported-locale>es</supported-locale>
<portlet-info>
<title>Demo Portlet</title>
</portlet-info>
</portlet>
```

3) Add the entry of that Portlet in portlet-instance.xml and give the <portlet-ref> as same as given in portlet.xml's <portlet-name> tag.

```
<deployment>
 <instance>
```

```
   <security-constraint>
     <policy-permission>
             <role-name>Authenticated</role-name>
              <action-name>view</action-name>
      </policy-permission>
      </security-constraint>
      <instance-id>Demo-PortletInstance</instance-id>
      <portlet-ref>DemoPortlet</portlet-ref>
   </instance>
</deployment>
```

3 Interval Rendering

Each Portlet in Groundwork Monitor application uses interval rendering mechanism to "re-render" its contents after a certain period (here 30 seconds). This is needed so that Portlet can always display latest available data which might have changed in the back-end. Each Portlet has its own mechanism of fetching this data by calling web services or getting it from cache (RTMM).

Note : RTMM stands for ReferenceTreeMetaModel which is a mega bean that has most of monitoring data. RTMM implements OnDemandServerpush which subscribes to the topic server. The most efficient way to access the monitoring/foundation data is via RTMM.

To implement Interval Rendering our JavaBean class needs to extend ServerPush class. ServerPush makes use of Icefaces Renderable interface to implement server initiated rendering.
By Default the rendering interval is 30 seconds, but we can customize according to our requirement by calling the parameterized constructor. Pass the desired interval in seconds in the constructor it will override the default 30 second time interval.

```
public class DemoBean extends ServerPush {
.
public DemoBean(long renderingInterval){
       super(renderingInterval);
}// constructor
.
}//class
```

Source Code: Server Push
For more information about Interval Rendering please refer to:

- See this link to get all the information about how one can implement On Demand OR Interval Rendering: http://www.icefaces.org/docs/latest/htmlguide/devguide/AdvancedTopics2.html
- Java Docs: http://www.icefaces.org/docs/v1_5_2/javadocs/icefaces/api/com/icesoft/faces/async/render/IntervalRenderer.html

4 Preferences

Editing Preferences is functionality available on many Portlets in "my groundwork" tab. User can edit ‘enter choices’ and ‘select options’ to customize Portlet according to need. When user clicks on edit preference icon, a jsp page is displayed to user. Using this jsp page user can customize the Portlet according to its need. These pages are generally suffixed with "XXXpref.jsp" and are simple JSP pages.

When user clicks on "edit preferences" icon, `doEdit()` method of portlet class gets called. Here in this example, we are getting the preference value from PortletPreferences Object by calling `PortletPreferences.getValue()` method and setting required attributes in render request object so that we further use it on our demoPref.jsp page. After setting the attribute in render request, dispatch the request to demoPref.jsp page by using `PortletRequestDispatcher's getRequestDispatcher()` method.

```
@Override
protected void doEdit(RenderRequest request, RenderResponse response)
    throws PortletException, IOException {
    response.setTitle("Edit HostGroup Demo Portlet Preferences");
    String perfValue = "";
    // get preferences from request
    PortletPreferences pref = request.getPreferences();
    // Check if prefrence is null then set default Host Group as Linux Servers
    if (pref != null) {
              perfValue = pref.getValue("demoHostGroupPreference",
              "Linux Servers");
    }
    //set the value of demoHostGroupPreference
    request.setAttribute("demoHostGroupPreference", perfValue);
    // get the request dispatcher using viewId
    PortletRequestDispatcher disp =
getPortletContext().getRequestDispatcher("/jsp/demoPref.jsp");
    response.setContentType("text/html");
    disp.include(request, response);
}
```

When user saves preferences, `processAction()` method of portlet class gets called. In our example, we are saving the HostGroup name into preference and calling the `PortletPreferences.store()` method to save the preference. After saving this preference just switch the PortalMode to View. Following snippet gives a brief idea of `processAction()` method.

```
@Override
public void processAction(ActionRequest request, ActionResponse response)
    throws PortletException, IOException {

    // get preferences
    PortletPreferences pref = request.getPreferences();
    String hostgroupNameValue = (String)request.getParameter("demoHostGroupPreference");
    if (null == hostgroupNameValue || "".equals(hostgroupNameValue)) {
        pref.setValue("demoHostGroupPreference", "Linux Servers");
    } else {
        pref.setValue("demoHostGroupPreference", hostgroupNameValue);
    }
    // store the preferences
    pref.store();
    // set the portlet mode
    response.setPortletMode(PortletMode.VIEW);
}
```

For more information about Interval Rendering please refer to Edit preferences

Source Code: demoPref.jsp

5 Screenshots:

5.1 Edit Preferences page

5.2 Host Group demo Portlet page

6 Dependencies

Please find below the list of compile and run time dependencies:-

6.1 Compile time dependencies

1) Collage-api-3.0.jar
2) Icefaces-facelet-1.8.2-p01EE.jar
3) Portlet-api-jbp-2.7.2.jar
4) Icefaces-1.8.2-p01-EE.jar
5) Log4j-1.2.8.jar
6) Gwportal-common-3.0.320.jar
7) Jaxrpc.jar

6.2 Run time dependencies

1) commons-collections-3.0.jar
2) commons-digester-1.6.jar
3) commons-lang-2.0.jar
4) gwportal-common-3.0.320.jar
5) icefaces-1.8.2-P01-EE.jar
6) icefaces-comps-1.8.2-P01-EE.jar
7) icefaces-facelets-1.8.2-P01-EE.jar
8) icefaces-portlet-1.8.2-P01-EE.jar
9) foundation-webapp.war (Should be deployed on Jboss portal server-/usr/local/groundwork/foundation/container/webapps)

7 Packaging & Deployment

The build scripts included with the demo sample are based on apache ant and apache maven. The tools are available at the following location:

ANT : ftp://archive/pub/groundwork-core/build/apache-ant-1.7.1-bin.tar.gz
Maven: ftp://archive/pub/groundwork-core/build/maven-1.0.2.tar.gz

Install the tools and make sure that the bin directory are in the path.

The maven script is used to include all dependencies and create the the war file. You can find the maven.xml script in the base directory (DemoPortlet) of demo application. User can compile or create a war file by executing following maven command:-

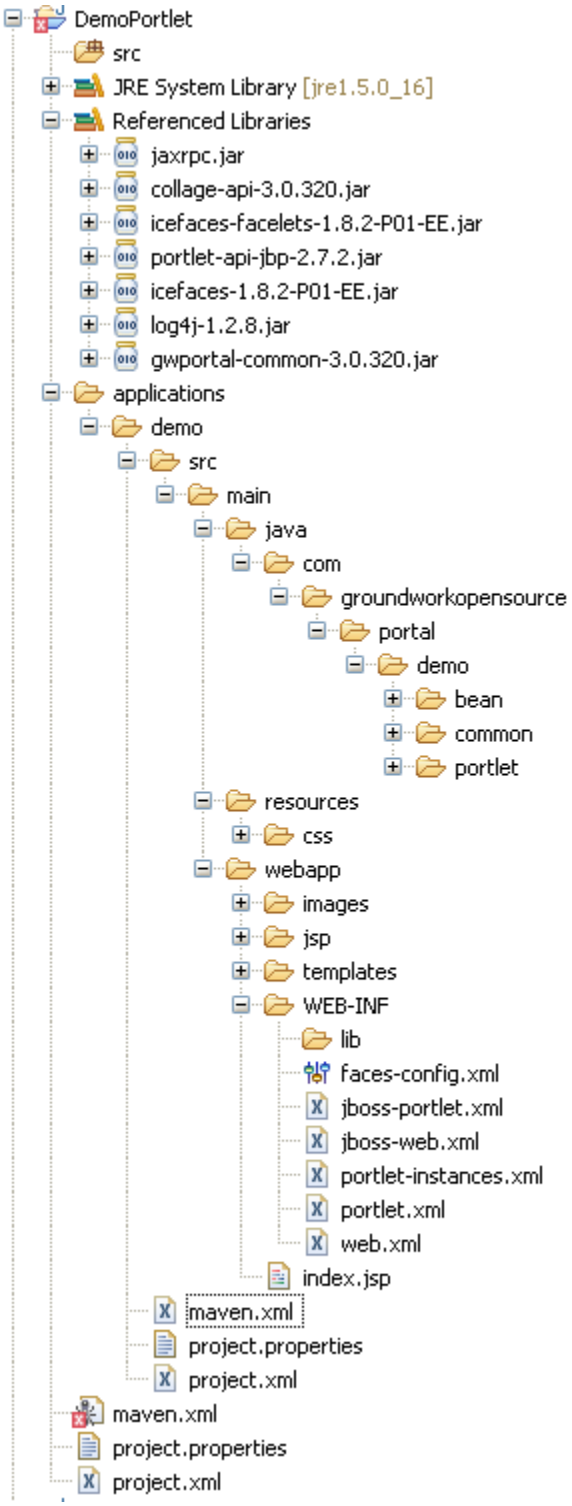
E:\DemoPortlet>maven clean build

Above maven command will create demo-portal.war file in target directory (i.e. DemoPortlet\applications\demo\target).

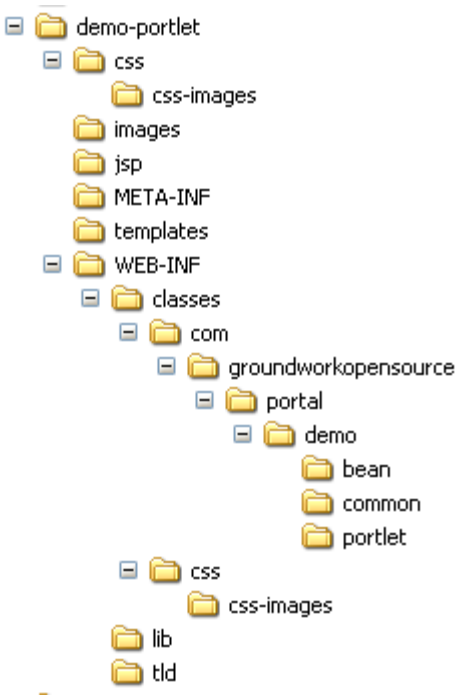
JBoss Deployment Dierctory: Place the “demo-portal.war” file in following directory on JBoss portal Server.

/usr/local/groundwork/foundation/container/webapps

Following image gives an idea about the Demo project directory in Eclipse.


DemoPortlet
src
JRE System Library [jre1.5.0_16]
Referenced Libraries
jaxrpc.jar
collage-api-3.0.320.jar
icefaces-facelets-1.8.2-P01-EE.jar
portlet-api-jbp-2.7.2.jar
icefaces-1.8.2-P01-EE.jar
log4j-1.2.8.jar
gwportal-common-3.0.320.jar
applications
demo
src
main
java
com
groundworkopensource
portal
demo
bean
common
portlet
resources
css
webapp
images
jsp
templates
WEB-INF
lib
faces-config.xml
jboss-portlet.xml
jboss-web.xml
portlet-instances.xml
portlet.xml
web.xml
index.jsp
maven.xml
project.properties
project.xml
maven.xml
project.properties
project.xml

After creating war file, this war can be deployed into JBoss Portal Server. Please refer to following image for war structure.



8 Appendix

8.1 DemoBean.java

```
package com.groundworkopensource.portal.demo.bean;


import java.io.Serializable;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.List;

import javax.xml.rpc.ServiceException;

import org.groundwork.foundation.ws.api.WSFoundationException;
import org.groundwork.foundation.ws.api.WSHost;
import org.groundwork.foundation.ws.impl.WSHostServiceLocator;
import org.groundwork.foundation.ws.model.impl.SimpleHost;
import org.groundwork.foundation.ws.model.impl.WSFoundationCollection;

import com.groundworkopensource.portal.common.FacesUtils;
import
com.groundworkopensource.portal.common.exception.PreferencesException;
import com.groundworkopensource.portal.demo.common.ServerPush;


public class DemoBean extends ServerPush implements Serializable {

    private static final long serialVersionUID = 1L;

    /*
     * HostGroupName
     * */
    private String hostGroupName;

    /*
     * List of Hosts in HostGroup
     * */
    private List<String> hostNamesInGroup = new ArrayList<String>();

    /*
     * Number of Host in HostGroup
     * */
    private int hostCount;

    /*
     * Foundation End-Point wahost
     * */
    private static String FOUNDATION_END_POINT_HOST = "wshost";

    /*
     * WSHostServiceLocator
     * */
    private static WSHostServiceLocator hostLocator;

    /*
```

```
     * WebService Path
     * */
    private static String WS_PATH = "http://localhost:8080/foundation-
webapp/services/";

    /*
     * HiddenField
     * */
    private String hiddenField="";

    /*
     * WSHost
     * */
    private WSHost wshost;

    /*
     * PrefrenceName
     * */
    private String prefName;

    /*
     * Constructer is used to get the Prefrence Name from FacedUtils
     * If no Prefrence fount it use "Linux Servers" as Default Host
Group
     * */
    public DemoBean() {

        try {
            wshost = hostLocator().gethost();
            prefName =
FacesUtils.getPreference("demoHostGroupPreference");
            if (prefName == null || "".equals(prefName)) {
                prefName = "Linux Servers";
                setHostGroupName(prefName);
            }
        } catch (ServiceException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (PreferencesException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public String getHostGroupName() {
        return hostGroupName;
    }

    public void setHostGroupName(String hostGroupName) {
        this.hostGroupName = hostGroupName;
    }

    public List<String> getHostNamesInGroup() {
        return hostNamesInGroup;
    }
```

```
    public void setHostNamesInGroup(List<String> hostNamesInGroup) {
        this.hostNamesInGroup = hostNamesInGroup;
    }

    public int getHostCount() {
        return hostCount;
    }

    public void setHostCount(int hostCount) {
        this.hostCount = hostCount;
    }

    public static WSHostServiceLocator hostLocator() {
        if (hostLocator == null) {
            hostLocator = new WSHostServiceLocator();
            try {
                hostLocator.setEndpointAddress(
                        DemoBean.FOUNDATION_END_POINT_HOST,
                        DemoBean.WS_PATH
                                + DemoBean.FOUNDATION_END_POINT_HOST);
            } catch (Exception exc) {
                // logger.error(exc.getMessage());
            } // end try/catch
        }
        return hostLocator;
    }

    @Override
    public void refresh(String xmlMessage) {
        // TODO Method not implemented yet: ServerPush.refresh(...)
    }

    public void setHiddenField(String hiddenField) {
        this.hiddenField = hiddenField;
    }

   /*
    * For More information about HiddenField
    * Please Refer to http://gwiki/index.php/Use_of_hidden_field
    *
    * */
    public String getHiddenField() {

        if(isIntervalRender()){
            try {
                hostNamesInGroup.clear();
                WSFoundationCollection wsCollection =
wshost.getSimpleHostsByHostGroupName(prefName,
                        false);
                setHostCount(wsCollection.getTotalCount());
                setHostGroupName(prefName);
                SimpleHost[] simpleHosts =
wsCollection.getSimpleHosts();
                if(simpleHosts != null){
                    for(SimpleHost host : simpleHosts){
                        hostNamesInGroup.add(host.getName());
                    }
```

```
                }
            } catch (WSFoundationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (RemoteException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        return hiddenField;
    }

}//class
```

8.2 DemoPortal.java

```
package com.groundworkopensource.portal.demo.portlet;

import java.io.IOException;

import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;
import javax.portlet.PortletException;
import javax.portlet.PortletMode;
import javax.portlet.PortletPreferences;
import javax.portlet.PortletRequestDispatcher;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;

import com.groundworkopensource.portal.common.BasePortlet;

public class DemoPortlet extends BasePortlet {

    /**
     * (non-Javadoc).
     *
     * @see
javax.portlet.GenericPortlet#processAction(javax.portlet.ActionRequest,
     *      javax.portlet.ActionResponse)
     */
    @Override
    public void processAction(ActionRequest request, ActionResponse
response)
            throws PortletException, IOException {
        /*
         * Method is called in response to a user action such as
clicking a
         * hyperlink or submitting a form. In this method a portlet may
modify
         * its own state as well as persistent information it.
         */

        // get preferences
        PortletPreferences pref = request.getPreferences();
        String hostgroupNameValue = (String) request
                .getParameter("demoHostGroupPreference");
        // Check if hostgroupNameValue is null or Empty then set the
        // demoHostGroupPreference as "Linux Servers"
        if (null == hostgroupNameValue ||
"".equals(hostgroupNameValue)) {
            pref.setValue("demoHostGroupPreference", "Linux Servers");
        } else {
            pref.setValue("demoHostGroupPreference",
hostgroupNameValue);
        }
        // store the preferences
        pref.store();
        // set the portlet mode
        response.setPortletMode(PortletMode.VIEW);
    }
```

```
    /**
     * (non-Javadoc).
     *
     * @see javax.portlet.GenericPortlet#doView(RenderRequest
     *      request,RenderResponse response)
     */
    @Override
    protected void doView(RenderRequest request, RenderResponse
response)
            throws PortletException, IOException {
        super.setViewPath("/jsp/demoPortlet.jspx");
        // Set the portlet title.
        response.setTitle("HostGroup Demo Portlet");
        super.doView(request, response);
    }

    /**
     * This method is Responsible for editing preferences of host
statistics
     * portlet
     *
     * @param request
     * @param response
     * @throws PortletException
     * @throws IOException
     */
    @Override
    protected void doEdit(RenderRequest request, RenderResponse
response)
            throws PortletException, IOException {
        response.setTitle("Edit HostGroup Demo Portlet Preferences");
        String perfValue = "";
        // get preferences from request
        PortletPreferences pref = request.getPreferences();
        // Check if prefrence is null then set default Host Group as
Linux
        // Servers
        if (pref != null) {
            perfValue = pref.getValue("demoHostGroupPreference",
                    "Linux Servers");
        }
        // set the value of demoHostGroupPreference
        request.setAttribute("demoHostGroupPreference", perfValue);
        // get the request dispatcher using viewId
        PortletRequestDispatcher disp = getPortletContext()
                .getRequestDispatcher("/jsp/demoPref.jsp");
        response.setContentType("text/html");
        disp.include(request, response);

    }

}
```

8.3 demoPortlet.jspx

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!--
    Coopyright (C) 2009 GroundWork Open Source, Inc. (GroundWork)
    All rights reserved. This program is free software; you can
redistribute
    it and/or modify it under the terms of the GNU General Public
License
    version 2 as published by the Free Software Foundation.

    This program is distributed in the hope that it will be useful, but
    WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
along
    with this program; if not, write to the Free Software Foundation,
Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
-->
<jsp:root version="1.2" xmlns:jsp="http://java.sun.com/JSP/Page"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ice="http://www.icesoft.com/icefaces/component"
      xmlns:ui="http://java.sun.com/jsf/facelets">
      <jsp:directive.page contentType="text/html;charset=ISO-8859-1"
            pageEncoding="ISO-8859-1" />
      <f:view>
            <ice:outputDeclaration doctypeRoot="HTML"
                  doctypePublic="-//W3C//DTD HTML 4.01
Transitional//EN"
                  doctypeSystem="http://www.w3.org/TR/html4/loose.dtd"
/>
            <html>
            <body>
            <ice:portlet>
                  <ui:composition template="/templates/template.xhtml">
                        <ui:define name="content">
                              <ice:form>
                                    <ice:inputHidden
value="#{demoBean.hiddenField}"></ice:inputHidden>

                                    <b><ice:outputText
                                          value="Host Under HostGroup :
#{demoBean.hostGroupName}"></ice:outputText></b>
                                    <br />
                                    <ice:outputText value="Total
Hosts : #{demoBean.hostCount}"></ice:outputText>
                                    <br />
                                    <ice:dataTable
value="#{demoBean.hostNamesInGroup}" var="hostList"
                                          border="1">
                                          <ice:column id="HLcolumnName">
                                                <f:facet name="header">
                                                      <ice:outputText
id="HLtxtColumnName" value="Host Name"></ice:outputText>
```

```
                                                    </f:facet>
                                                    <ice:outputText
value="#{hostList}"></ice:outputText>
                                                </ice:column>

                                            </ice:dataTable>
                                        </ice:form>
                                    </ui:define>
                                </ui:composition>
                        </ice:portlet>
                        </body>
                        </html>
                </f:view>
</jsp:root>
```

8.4 demoPref.jsp

```
<!--
    Coopyright (C) 2009 GroundWork Open Source, Inc. (GroundWork)
    All rights reserved. This program is free software; you can
redistribute
    it and/or modify it under the terms of the GNU General Public
License
    version 2 as published by the Free Software Foundation.

    This program is distributed in the hope that it will be useful, but
    WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    General Public License for more details.

    You should have received a copy of the GNU General Public License
along
    with this program; if not, write to the Free Software Foundation,
Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
-->
<%@ page language="java"%>
<%@ page import="javax.portlet.RenderRequest" %>
<%@ page import="javax.portlet.RenderResponse" %>
<%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>
<portlet:defineObjects/>
<form method="post" action="<portlet:actionURL/>">
<table border="0">
      <tr>
            <td>Enter HostGroup Name :</td>
            <td><input type="text" name='demoHostGroupPreference'
                        id='demoHostGroupPreference'
                        class="demoHostGroupPreference"
                        value='<
%=renderRequest.getAttribute("demoHostGroupPreference")%>' />
            </td>
      </tr>

      <tr>
            <td colspan="2"><input type="submit" value="Save
Preferences"></td>
      </tr>
</table>

</form>
```

8.5 ServerPush.java

```
package com.groundworkopensource.portal.demo.common;


import java.io.Serializable;
import java.util.Collection;
import java.util.concurrent.ConcurrentHashMap;

import org.apache.log4j.Logger;

import com.icesoft.faces.async.render.IntervalRenderer;
import com.icesoft.faces.async.render.RenderManager;
import com.icesoft.faces.async.render.Renderable;
import com.icesoft.faces.async.render.SessionRenderer;
import com.icesoft.faces.webapp.xmlhttp.FatalRenderingException;
import com.icesoft.faces.webapp.xmlhttp.PersistentFacesState;
import com.icesoft.faces.webapp.xmlhttp.RenderingException;


/**
 * ServerPush - a class that should be extended by all want to have JMS
Push
 * functionality. Threading is required in this class for parallel
processing
 * for the push.
 */
public abstract class ServerPush implements Serializable, Renderable {

    /**
     * HALF_MINUTE
     */
    private static final int HALF_MINUTE = 30000;

    /** Logger. */
    private static final Logger LOGGER =
Logger.getLogger(ServerPush.class
            .getName());

    /** serialVersionUID. */
    private static final long serialVersionUID = 1L;

    /** Active server push beans. */
    private static final ConcurrentHashMap<String, ServerPush> BEANS =
new ConcurrentHashMap<String, ServerPush>();

    // /** requestId. */
    // protected String requestId = null;

    /** Group Render Name. */
    protected String groupRenderName = "entity";

    /** ICEFaces rendering state. */
    protected PersistentFacesState renderState = PersistentFacesState
            .getInstance();

    /** intervalRender. */
```

```
    private boolean intervalRender = true;

    /** intervalRenderer. */
    private IntervalRenderer intervalRenderer;

    /**
     * Default Constructor.
     */
    public ServerPush() {
        initialize();
    }

    /**
     * Constructor.
     *
     * @param renderingInterval
     *            the rendering interval
     */
    public ServerPush(long renderingInterval) {
        initialize(renderingInterval);
    }

    /**
     * One argument constructor taking listenToTopic as parameter. This
bean
     * will listen to 'topic'..
     *
     * @param listenToTopic
     *            the listen to topic
     */
    public ServerPush(String listenToTopic) {
        // this.listenToTopic = listenToTopic;
        initialize();
    }

    /**
     * Get an iterator over all of the currently active server push
beans.
     *
     * @return Collection of the currently active server push beans
     */
    public static final Collection<ServerPush> getBeans() {
        return BEANS.values();
    }

    /**
     * Initializes the Server Push.Adds the bean to the resource
manager.
     */
    private void initialize() {
        this.groupRenderName = Integer.toString(this.hashCode());
        // beans.put(groupRenderName, this);
        //
RenderManager.getInstance().getOnDemandRenderer(groupRenderName).add(
        // this);
        LOGGER.debug("initialize Inverval render for class "
                + this.getClass().getName());
```

```
        intervalRenderer =
RenderManager.getInstance().getIntervalRenderer(
                groupRenderName);
        intervalRenderer.setInterval(HALF_MINUTE);
        intervalRenderer.setName(groupRenderName);
        intervalRenderer.add(this);
        intervalRenderer.requestRender();
        // Remove any session attributes for tree expansion

        LOGGER.debug("Creation of  [" + this.getClass().getName()
                + Integer.toString(this.hashCode()) + "]");
    }

    /**
     * Initializes the Server Push.Adds the bean to the resourcemanager.
     *
     * @param renderingInterval
     *            the rendering interval
     */
    private void initialize(long renderingInterval) {
        this.groupRenderName = Integer.toString(this.hashCode());
        // beans.put(groupRenderName, this);
        //
RenderManager.getInstance().getOnDemandRenderer(groupRenderName).add(
        // this);
        LOGGER.debug("initialize Inverval render for class "
                + this.getClass().getName());
        intervalRenderer =
RenderManager.getInstance().getIntervalRenderer(
                groupRenderName);
        intervalRenderer.setInterval(renderingInterval);
        intervalRenderer.setName(groupRenderName);
        intervalRenderer.add(this);
        intervalRenderer.requestRender();
        LOGGER.debug("Creation of  [" + this.getClass().getName()
                + Integer.toString(this.hashCode()) + "]");
    }

    /**
     * Abstract method refresh() - that should be implemented by all
classes
     * want to have JMS Push functionality.
     *
     * @param xmlMessage
     *            the xml message
     */
    public abstract void refresh(String xmlMessage);

    /**
     * finalize method.
     */
    @Override
    protected void finalize() {
        LOGGER.debug("Finalize called on [" + this.getClass().getName()
                + Integer.toString(this.hashCode()) + "]");

    }
```

```
    /**
     * Free.
     */
    public void free() {
        if (groupRenderName != null) {
            LOGGER.debug("free() called on [" +
this.getClass().getName()
                    + Integer.toString(this.hashCode()) + "]");
            // beans.remove(groupRenderName);
            SessionRenderer.removeCurrentSession(groupRenderName);
        }
    }

    /**
     * (non-Javadoc).
     *
     * @return the state
     *
     * @see com.icesoft.faces.async.render.Renderable#getState()
     */
    public PersistentFacesState getState() {
        setIntervalRender(true);
        return renderState;
    }

    /**
     * (non-Javadoc).
     *
     * @param exception
     *            the exception
     *
     * @see
com.icesoft.faces.async.render.Renderable#renderingException(com.icesoft
     *      .faces.webapp.xmlhttp.RenderingException)
     */
    public void renderingException(RenderingException exception) {
        if (LOGGER.isDebugEnabled()) {
            LOGGER.debug("Received exception while rendering view for
bean "
                    + getClass().getName() +
Integer.toString(hashCode())
                    + ": " + exception.getMessage());
        }
        if (exception instanceof FatalRenderingException) {
            if (LOGGER.isDebugEnabled()) {
                LOGGER.debug("Cleaning up bean " + getClass().getName()
                        + Integer.toString(hashCode()));
            }
            performCleanup();
            free();
        }
    }

    /**
     * Used to properly shut-off the ticking clock.
     *
```

```
     * @return true if properly shut-off, false if not.
     */
    protected boolean performCleanup() {
        try {
            if (intervalRenderer != null) {
                intervalRenderer.remove(this);
                // whether or not this is necessary depends on how
'shutdown'
                // you want an empty renderer. If it's emptied often,
the cost
                // of shutdown+startup is too great
                if (intervalRenderer.isEmpty()) {
                    intervalRenderer.dispose();
                }
                intervalRenderer = null;
                if (LOGGER.isDebugEnabled()) {
                    LOGGER.debug("performCleanup() is called for "
                            + getClass().getName()
                            + Integer.toString(hashCode()));
                }
            }

            return true;

        } catch (Exception failedCleanup) {
            LOGGER.error("Failed to cleanup a clock inside render bean
",
                    failedCleanup);
        }
        return false;
    }

    /**
     * Sets the intervalRender.
     *
     * @param intervalRender
     *            the intervalRender to set
     */
    public void setIntervalRender(boolean intervalRender) {
        this.intervalRender = intervalRender;
    }

    /**
     * Returns the intervalRender.
     *
     * @return the intervalRender
     */
    public boolean isIntervalRender() {
        return intervalRender;
    }
}
```