Introduction to Portlet Programming with JSR-168

Noel J. Bergman
DevTech®
Session Overview

Portals are a hot growth area, but prior to JSR-168, we lacked a standard API for Portlet Development. This session will introduce the unique characteristics of Portals, the Portlets of which they are built, and the JSR-168 specification that enables portability between Portlet containers.

Topics will include aggregation, request processing, URIs, modes and states, dispatching, preferences, selected vendor extensions and more.

Although this session is offered as an introduction to Portals and JSR-168, prerequisites include understanding of XML, HTML, Servlets and JavaServer Pages.

PLEASE ASK QUESTIONS! 😊
Session Prerequisites

- You will want a good understanding of Servlets and JavaServer Pages.
- You will want an understanding of XML and HTML.
- In order to work with the technology covered during the session, you will need a JSR-168 compliant Portal server, such as one of the ones we’ll use for demonstrations.
Portal Demonstrations

Let’s look at some sample portals.
What Is a Portal?

- Portals are distinguished by a number of characteristics. The two most often observed by users are:
  - Aggregation
  - Personalization

- These characteristics have a major impact on the programming model, and how it differs from a non-portal web site.
Aggregation

- Aggregation refers to taking content from multiple, independent, delivery channels, and providing a unified user experience.
- A Portal page will often contain multiple portlets, each rendering content from a separate source.
- Aggregation is outside of the scope of JSR-168. Many Portal engines manage pages and aggregation differently.
Personalization

- Personalization refers to allowing each user to customize their portal experience.
- Different portal engines will provide for differing levels of personalization.
- JSR-168 includes a notion of Portlet Preferences, which a developer can use to allow a user to customize how the portlet will perform for that specific user.
What Is JSR-168?

- The numeric designation for the JCP’s first local Portlet Specification.
  - Remote portlet technology is defined by the WSRP specification.
- Extends Servlet Specification 2.3 with new Portlet concepts.
- Defines the standard contract between the Portlet container and Portlet components.
  - JSR-168 does not define Portal behaviors, such as how pages are laid out, how aggregation works, how administration is performed, etc.
Why JSR-168?

- Prior to JSR-168, we had many different Portlet APIs. The lack of standards fragmented the solution space, creating issues for customers and ISVs.

This session focuses on Portlets and related semantics as defined by JSR-168. We will not be discussing the original Jetspeed API, the IBM Portlet API (a Jetspeed fork), the uPortal API, or other, non-standard, Portlet APIs, which may offer more or less differing semantics.
Portlet-based Applications vs Servlet-based Applications

- Installing a Portlet Application provides capability, not content.
- Portlets from multiple applications can be aggregated on the same page.
- More administration with Portlets and Portals than with traditional web applications.
Some of the Portal Servers That Support JSR-168

- **Open Source**
  - Pluto (http://portals.apache.org/pluto)
  - Jetspeed-2 (http://portals.apache.org/jetspeed-2)
  - GridSphere (http://www.gridsphere.org)
  - uPortal (http://www.uPortal.org)
  - Liferay (http://www.liferay.com)

- **Commercial**
  - WebSphere Portal Server (http://www.ibm.com/software/genservers/portal)
  - WebLogic Portal Server (www.bea.com/products/weblogic/portal)
  - Plumtree Corporate Portal (http://www.plumtree.com/products/platform)

☆ JSR-168 is not supported by all versions of each portal server
What Is a Portlet?

- A new type of Web Component, cousin to the Servlet.
- Provides the interface to content. In MVC terms:
  - Portlet class – Controller
  - Portlet’s JSP – View
Portlets Differ from Servlets

- Two phase request processing
- Persistent render parameters
- Portlet render as markup fragments
- Special URLs to access a Portlet
- Portlet Modes
- Portlet Window States
- Portlet Preferences
Model/View/Controller Pattern

- **Model**
  - Abstract interface
  - No user interaction
  - No presentation

- **View**
  - Acquires Model data and renders it
  - Can be replaced, independent of the Model

- **Controller**
  - Receives input events
  - Provides the means to effect change on the Model
  - Orchestrates interaction between Model and View
MVC and Portlets

- The Portlet's primary role is as the Controller.
  - Receives events from the container
    - Process actions
    - Render content
- Portlets typically dispatch to the View (usually a JSP page).
- As the Controller, Portlets orchestrate interaction with the Model and moving content to the View.
- The Model can be anything: POJO, EJB, Web Service, etc.
  - A Portal is intended to provide an aggregated UI to models, not to necessarily contain those models.
Portlet URLs

- Portlets do not have a directly exposed endpoint exposed to the browser. Special URLs are encoded that tell the container what to do.

- JSR-168 specifies two types of URLs:
  - ActionURL – invokes a portlet’s action handling.
  - RenderURL – invokes a portlet’s render behavior.

- We will cover these concepts in more detail later.
Portlet Modes

- Portlets have *modes*, which are a view-related convenience.
- JSR-168 specifies standard modes, which are typically used as:
  - View – application interaction, but not – as the name might imply – read only.
  - Edit – present a screen for editing user preferences
  - Help – provide help
- JSR-168 also supports the idea of custom modes.
- Unlike, for example, modes within the IBM Portlet API, modes in JSR-168 are advisory.
Window State

- Portlets have a Window State, which tells the container how much space to provide. The standard Window States are:
  - Normal – shares space on the page with other portlets.
  - Minimized – minimal space, *e.g.*, only the titlebar might be shown.
  - Maximized – receives the entire space on the page that is available to a portlet.
- JSR-168 also provides for custom Window States, *e.g.*, a “half” window state, requesting half of the available screen space.
Deployment Descriptor

- portlet.xml

- Defines:
  - Portlets
    - Name
    - Display Name
    - Portlet Class
    - Initialization Parameters
    - Resource Bundle
    - Preferences Validator
    - Caching
    - Markups and Modes supported
    - Locales supported
    - Preferences
    - Security Role References
  - Custom Modes
  - Custom Window States
  - Security Constraints
## The `javax.portlet` Package

- `ActionRequest` — `javax.portlet.ActionRequest`
- `ActionResponse` — `javax.portlet.ActionResponse`
- `GenericPortlet` — `javax.portlet.GenericPortlet`
- `PortalContext` — `javax.portlet.PortalContext`
- `Portlet` — `javax.portlet.Portlet`
- `PortletConfig` — `javax.portlet.PortletConfig`
- `PortalContext` — `javax.portlet.PortalContext`
- `PortletException` — `javax.portlet.PortletException`
- `PortletMode` — `javax.portlet.PortletMode`
- `PortletModeException` — `javax.portlet.PortletModeException`
- `PortletPreferences` — `javax.portlet.PortletPreferences`
- `PortletRequest` — `javax.portlet.PortletRequest`
- `PortletRequestDispatcher` — `javax.portlet.PortletRequestDispatcher`
- `PortletResponse` — `javax.portlet.PortletResponse`
- `PortletSecurityException` — `javax.portlet.PortletSecurityException`
- `PortletSession` — `javax.portlet.PortletSession`
- `PortletSessionUtil` — `javax.portlet.PortletSessionUtil`
- `PortletURL` — `javax.portlet.PortletURL`
- `PreferencesValidator` — `javax.portlet.PreferencesValidator`
- `ReadOnlyException` — `javax.portlet.ReadOnlyException`
- `RenderRequest` — `javax.portlet.RenderRequest`
- `RenderResponse` — `javax.portlet.RenderResponse`
- `UnavailableException` — `javax.portlet.UnavailableException`
- `ValidatorException` — `javax.portlet.ValidatorException`
- `WindowState` — `javax.portlet.WindowState`
- `WindowStateException` — `javax.portlet.WindowStateException`

### Legend:

<table>
<thead>
<tr>
<th>Class</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>javax.portlet</code></td>
<td></td>
</tr>
<tr>
<td><code>javax.portlet.ActionRequest</code></td>
<td><code>javax.portlet.ActionRequest</code></td>
</tr>
<tr>
<td><code>javax.portlet.ActionResponse</code></td>
<td><code>javax.portlet.ActionResponse</code></td>
</tr>
<tr>
<td><code>javax.portlet.GenericPortlet</code></td>
<td><code>javax.portlet.GenericPortlet</code></td>
</tr>
<tr>
<td><code>javax.portlet.Portlet</code></td>
<td><code>javax.portlet.Portlet</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletConfig</code></td>
<td><code>javax.portlet.PortletConfig</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletException</code></td>
<td><code>javax.portlet.PortletException</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletMode</code></td>
<td><code>javax.portlet.PortletMode</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletModeException</code></td>
<td><code>javax.portlet.PortletModeException</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletPreferences</code></td>
<td><code>javax.portlet.PortletPreferences</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletRequest</code></td>
<td><code>javax.portlet.PortletRequest</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletRequestDispatcher</code></td>
<td><code>javax.portlet.PortletRequestDispatcher</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletResponse</code></td>
<td><code>javax.portlet.PortletResponse</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletSecurityException</code></td>
<td><code>javax.portlet.PortletSecurityException</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletSession</code></td>
<td><code>javax.portlet.PortletSession</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletSessionUtil</code></td>
<td><code>javax.portlet.PortletSessionUtil</code></td>
</tr>
<tr>
<td><code>javax.portlet.PortletURL</code></td>
<td><code>javax.portlet.PortletURL</code></td>
</tr>
<tr>
<td><code>javax.portlet.PreferencesValidator</code></td>
<td><code>javax.portlet.PreferencesValidator</code></td>
</tr>
<tr>
<td><code>javax.portlet.ReadOnlyException</code></td>
<td><code>javax.portlet.ReadOnlyException</code></td>
</tr>
<tr>
<td><code>javax.portlet.RenderRequest</code></td>
<td><code>javax.portlet.RenderRequest</code></td>
</tr>
<tr>
<td><code>javax.portlet.RenderResponse</code></td>
<td><code>javax.portlet.RenderResponse</code></td>
</tr>
<tr>
<td><code>javax.portlet.UnavailableException</code></td>
<td><code>javax.portlet.UnavailableException</code></td>
</tr>
<tr>
<td><code>javax.portlet.ValidatorException</code></td>
<td><code>javax.portlet.ValidatorException</code></td>
</tr>
<tr>
<td><code>javax.portlet.WindowState</code></td>
<td><code>javax.portlet.WindowState</code></td>
</tr>
<tr>
<td><code>javax.portlet.WindowStateException</code></td>
<td><code>javax.portlet.WindowStateException</code></td>
</tr>
</tbody>
</table>
The Portlet Interface

- Minimalist interface

```java
public interface Portlet {
    void init(PortletConfig);
    void processAction(ActionRequest, ActionResponse);
    void render(RenderRequest, RenderResponse);
    void destroy();
}
```

- That’s it!
public abstract class GenericPortlet implements Portlet, PortletConfig
{
    public void init() throws PortletException
    public PortletConfig getPortletConfig()
    protected java.lang.String getTitle(RenderRequest request)
    protected void doDispatch(RenderRequest request, RenderResponse response)
    protected void doView(RenderRequest request, RenderResponse response)
    protected void doEdit(RenderRequest request, RenderResponse response)
    protected void doHelp(RenderRequest request, RenderResponse response)
    ... inherited methods ...
}
The GenericPortlet Class

- Portlets will normally extend this class, rather than implement Portlet directly.
- Will typically override:
  - `void processAction(ActionRequest, ActionResponse);`
  - `void doMode(RenderRequest, RenderResponse);`
- Might override:
  - `protected java.lang.String getTitle(RenderRequest request)`
- To implement a custom mode, override:
  - `void doDispatch(RenderRequest, RenderResponse);`
  - `void doMode(RenderRequest, RenderResponse);`
Hello World, Portlet-Style

```java
package com.devtech.com.portlets;
import javax.portlet.*
public class HelloPortlet extends GenericPortlet
{
    protected void doView(RenderRequest req, RenderResponse res)
    {
        res.setContentType("text/html");
        res.getWriter().write("<P>Hello World</P>" starring Mark Hamill);
    }
}

• We really should not be using the Writer; should be using a dispatcher with a JSP page.
```
Hello World v2, Portlet-Style

```java
package com.devtech.com.portlets;
import javax.portlet.*
public class HelloPortlet extends GenericPortlet {

    public void processAction(ActionRequest req, ActionResponse res) {
        String who = req.getParameter("greet");
        if (who == null) who = "World";
        res.setRenderParameter("msg", "Hello " + who);
    }

    protected void doView(RenderRequest req, RenderResponse res) {
        res.setContentType("text/html");
        res.getWriter().writeln("<P>" + req.getParameter("msg") + "</P>");
    }

    • This portlet would NOT work as-is. We haven’t provided a means to access the action.
```
Two-phase Request Processing

- **Action Phase**
  - Initiated by an ActionURL.
  - Allows a redirect relative to the Portlet Application.
  - The only time you can access the request body.
  - Provides Render Parameters for the Render Phase.

- **Render Phase**
  - Idempotent
  - Replayable Render Parameters
    - Set by the portlet during action processing, or attached to a RenderURL.
Action Phase

- void processAction(ActionRequest, ActionResponse);

- `ActionRequest` provides access to the HTTP request.

- `ActionResponse` does not generate output. It is used to set the parameters that will be available during the render phase.

- Preferences can be stored only during this phase.

- See also: `ActionRequest/ActionResponse`
Render Phase

- `void render(RenderRequest, RenderResponse);`
- Generally implemented by `doMode(...)` methods.
- `RenderRequest` does not provide access to parameters provided by an ActionURL or form, only to parameters explicitly set by a RenderURL or by methods on ActionResponse.
- See also: `RenderRequest/RenderResponse`
The PortletConfig Interface

```java
public interface PortletConfig
{
    public String getPortletName();
    public PortletContext getPortletContext();
    public ResourceBundle getResourceBundle(Locale locale);
    public String getInitParameter(String name);
    public Enumeration getInitParameterNames();
}
```
The **PortletConfig Interface**

- Allows a Portlet to access its `PortletContext`, resource bundle associated with the Portlet, and initialization parameters.

- Initialization parameters are defined by `<init-param>` tags in `portlet.xml`, similar to those for servlets that are in `web.xml`.
public interface PortletContext
{
    public String getServerInfo();
    public PortletRequestDispatcher getRequestDispatcher(String path);
    public PortletRequestDispatcher getNamedDispatcher(String name);
    public InputStream getResourceAsStream(String path);
    public int getMajorVersion();
    public int getMinorVersion();
    public String getMimeType(String file);
    public String getRealPath(String path);
    public Set getResourcePaths(String path);
    public URL getResource(String path) throws MalformedURLException;
    public Object getAttribute(String name);
    public Enumeration getAttributeNames();
    public String getPortletContextName();
    public void log(String msg);
    public void log(String message, Throwable throwable);
    public void removeAttribute(String name);
    public void setAttribute(String name, Object object);
}
The PortletContext Interface

- The Portlet’s view of the container.
- Provides access to context parameters, defined in `web.xml`, and context scoped attributes.
- Most methods provide the same functionality as the similarly named methods of the ServletContext.
- The `get[Request|Named]Dispatcher` method is used to get a dispatcher for other dynamic resources within this application.
public interface PortletRequest
{
    public static final String USER_INFO
    public static final String BASIC_AUTH
    public static final String FORM_AUTH
    public static final String CLIENT_CERT_AUTH
    public static final String DIGEST_AUTH
    public boolean isWindowStateAllowed(WindowState state);
    public boolean isPortletModeAllowed(PortletMode mode);
    public PortletMode getPortletMode();
    public WindowState getWindowState();
    public PortletPreferences getPreferences();
    public PortletSession getPortletSession();
    public PortletSession getPortletSession(boolean create);
    public String getProperty(String name);
    public Enumeration getProperties(String name);
    public Enumeration getPropertyNames();
    public PortalContext getPortalContext();
    public String getAuthType();
    public String getContextPath();
    public String getRemoteUser();
    public Principal getUserPrincipal();
    public boolean isUserInRole(String role);
    public Object getAttribute(String name);
    public Enumeration getAttributeNames();
    public String getParameter(String name);
    public String getParameterValues(String name);
    public Map getParameterMap();
    public boolean isSecure();
    public void setAttribute(String name, Object o);
    public void removeAttribute(String name);
    public String getRequestedSessionId();
    public boolean isRequestedSessionIdValid();
    public String getResponseContentType();
    public Enumeration getResponseContentTypes();
    public Locale getLocale();
    public Enumeration getLocales();
    public String getScheme();
    public String getServerName();
    public int getServerPort();
}
PortletRequest and Views

- `getResponseContentType()`
  - preferred content type
- `getResponseContentTypes()`
  - enumeration of acceptable content types in descending order of preference
- `getLocale()`
  - What locale is associated with this request?
  - We can use this and the content type to determine which JSP page to dispatch.
public interface PortletResponse
{
    public void addProperty(String key, String value);
    public void setProperty(String key, String value);
    public String encodeURL(String path);
}
The `ActionRequest` Interface

```java
public interface ActionRequest extends PortletRequest {
    public InputStream getPortletInputStream() throws IOException;
    public void setCharacterEncoding(String enc) throws UnsupportedEncodingException;
    public BufferedReader getReader() throws UnsupportedEncodingException, IOException;
    public String getCharacterEncoding();
    public String getContentType();
    public int getContentLength();
}
```
**ActionRequest**

- Provides access to the request body
  - Only parameters are available from a *RenderRequest*
The `ActionResponse` Interface

```java
public interface ActionResponse extends PortletResponse {
    public void setWindowState(WindowState windowState)
        throws WindowStateException;
    public void setPortletMode(PortletMode portletMode)
        throws PortletModeException;
    public void sendRedirect(String location)
        throws IOException;
    public void setRenderParameters(Map parameters);
    public void setRenderParameter(String key, String value);
    public void setRenderParameter(String key, String[] values);
}
```
**ActionResponse**

- send redirect
- set portlet mode for the render call
- set window state for the render call
- set render parameters
Render Parameters

- Set via RenderURL or ActionResponse.
- Preserved by the Portlet container, and replayed when a Portlet is rendered.
- A portlet’s render parameters are reset when a Portlet URL is submitted to the Portlet Container for that portlet.
The `RenderRequest` Interface

- Just an empty extension of `PortletRequest`

```java
public interface RenderRequest extends PortletRequest {

}
```
The **RenderResponse** Interface

```java
public interface RenderResponse extends PortletResponse {
    public static final String EXPIRATION_CACHE
    public String getContentType();
    public PortletURL createRenderURL();
    public PortletURL createActionURL();
    public String getNamespace();
    public void setTitle(String title);
    public void setContentType(String type);
    public String getCharacterEncoding();
    public java.io.PrintWriter getWriter() throws IOException;
    public java.util.Locale getLocale();
    public void setBufferSize(int size);
    public int getBufferSize();
    public void flushBuffer() throws IOException;
    public void setBufferSize();
    public void flushBuffer() throws IOException;
    public void resetBuffer();
    public boolean isCommitted();
    public void reset();
    public void reset();
    public OutputStream getPortletOutputStream() throws IOException;
}
```
SetTitle(String)
- sets the portlet's title
- normally called by GenericPortlet.render(...)
- must be called before committing portlet content

SetContentType(String)
- must be called before calling getWriter() or getOutputStream()
  - IllegalStateException thrown otherwise
- must be a valid type for this response
  - see PortletRequest.getResponseContentTypes()
  - IllegalArgumentException thrown if invalid type

GetWriter() / GetOutputStream()
- mutually exclusive
- IllegalStateException if called before setContentType()
public interface PortletURL
{
    public void setWindowState(WindowState windowState)
            throws WindowStateException;
    public void setPortletMode(PortletMode portletMode)
            throws PortletModeException;
    public void setParameter(String name, String value);
    public void setParameter(String name, String[] values);
    public void setParameters(Map parameters);
    public void setSecure(boolean secure)
            throws PortletSecurityException;
    public String toString();
}
Portlet URLs

- Action URLs cause the Portlet Container to clear the target portlet’s render parameters, and invoke the target portlet’s `processAction` method.
- Render URLs cause the Portlet Container to invoke the target portlet’s `render` method, with the specified set of render parameters, if any.
- Render URLs should not be associated with forms. Containers are not required to add the form fields to the set of render parameters.
- Portal controls (e.g., skin buttons) must be treated as render URLs that preserve the existing set of render parameters.
- `PortletURL` uses `x-www-form-urlencoded` encoding for parameter names and values.
public interface PortletPreferences
{
    public boolean isReadOnly(String key);
    public String getValue(String key, String def);
    public String[] getValues(String key, String[] def);
    public void setValue(String key, String value)
        throws ReadOnlyException;
    public void setValues(String key, String[] values)
        throws ReadOnlyException;
    public Enumeration getNames();
    public Map getMap();
    public void reset(String key) throws ReadOnlyException;
    public void store() throws IOException, ValidatorException;
}
public interface PreferencesValidator
{
    public void validate(PortletPreferences preferences)
    throws ValidatorException;
}

The PortletRequestDispatcher Interface

```java
public interface PortletRequestDispatcher {
    public void include(RenderRequest request,
                         RenderResponse response)
        throws PortletException, IOException;
}
```
PortletRequestDispatcher

- Can encode parameters in a query string
  - aggregated with existing render parameters
  - override any existing parameter of that name
Portlet JSP Tags

- Assuming:
  
  ```
  <%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
  
  <portlet:defineObjects/>
  
  <portlet:actionURL/>
  
  <portlet:renderURL/>
  
  <portlet:param/>
  
  <portlet:namespace/>
  ```
<portlet:defineObjects/>

- Defines implicit objects and enables the environment for the other tags.
  - RenderRequest renderRequest
  - RenderResponse renderResponse
  - PortletConfig portletConfig
<portlet:actionURL/>

- Creates an Action URL.
- Attributes:
  - portletMode
  - windowState
  - secure
  - var

- The body may include param tags:

```xml
<portlet:actionURL windowState = "maximized"
    portletMode = "edit">
    <portlet:param name="action" value="editStocks"/>
</portlet:actionURL>
```
<portlet:renderURL/>

- Creates an Render URL.
- Attributes:
  - portletMode
  - windowState
  - secure
  - var
- The body may include param tags:
  
  ```xml
  <portlet:renderURL portletMode="view"
                   windowState="normal">
    <portlet:param name="showQuote" value="myCompany"/>
    <portlet:param name="showQuote" value="someOtherCo"/>
  </portlet:renderURL>
  ```
The **PortletSession** Interface

```java
public interface PortletSession {
    public static final int APPLICATION_SCOPE
    public static final int PORTLET_SCOPE
    public Object getAttribute(String name);
    public Object getAttribute(String name, int scope);
    public Enumeration getAttributeNames();
    public Enumeration getAttributeNames(int scope);
    public long getCreationTime();
    public String getId();
    public long getLastAccessedTime();
    public int getMaxInactiveInterval();
    public void invalidate();
    public boolean isNew();
    public void removeAttribute(String name);
    public void removeAttribute(String name, int scope);
    public void setAttribute(String name, Object value);
    public void setAttribute(String name, Object value, int scope);
    public void setMaxInactiveInterval(int interval);
    public PortletContext getPortletContext();
}
```
Portlet Logging

- `PortletContext.log(String[, Throwable])`
- Understand for whom the Portlet Log is intended.
- Provide content accordingly.
- Remember the `NullPointerException` example!
Portlet Services

- IBM (WebSphere) specific, but being discussed for wider deployment.
- JNDI Lookup
- Similar to local EJB, but with simple “POJO” interfaces.
Cooperative Portlets

- From a user’s perspective, the portal is the application.
- We would like to be able to construct new business applications “at the glass” by integrating portlets as components.
- IBM’s “Cooperative Portlet” technology uses portlet filters, a portlet service, and WSDL to support such “at the glass” integration.
WSRP

- WSRP (Web Services, Remote Portlets) is an OASIS Specification for allowing a local Portlet to proxy for a portlet running on a remote Portal.
Related Sessions

- “JSP 2.0 Tag Files: Custom Tags Made Easy” – Noel J. Bergman
- “WSRP: Web Services for Remote Portlets” – Dave Landers
- “Comparing Java Web Frameworks: JSF, Struts, Spring, Tapestry and WebWork” – Matt Raible
Supplemental Information

The following slides provide additional reference information, but are not intended to be presented during this session. Please feel free to find me around the conference during the week.

Thank you for coming 😊
public class PortletSessionUtil
{
    public static String decodeAttributeName(String name)
    public static int decodeScope(String name)
}
The `PortletException` Class

```java
public class PortletException extends Exception {
    public PortletException()
    public PortletException(String text)
    public PortletException(String text, Throwable cause)
    public PortletException(Throwable cause)
    public void printStackTrace()
    public void printStackTrace(PrintStream out)
    public void printStackTrace(PrintWriter out)
    public Throwable getCause()
}
```
The `ReadOnlyException` Class

```java
public class ReadOnlyException extends PortletException {
    private ReadOnlyException()
    public ReadOnlyException(String text)
    public ReadOnlyException(String text, Throwable cause)
    public ReadOnlyException(Throwable cause)
}
```
The PortletModeException Class

```java
public class PortletModeException extends PortletException {
    public PortletModeException(String text, PortletMode mode)
    public PortletModeException(String text, Throwable cause, PortletMode mode)
    public PortletModeException(Throwable cause, PortletMode mode)
    public PortletModeException()

    public PortletMode getMode()
}
```
public class UnavailableException extends PortletException
    public UnavailableException(String text)
    public UnavailableException(String text, int seconds)
    public boolean isPermanent()
    public int getUnavailableSeconds()
}
The ValidatorException Class

```java
public class ValidatorException extends PortletException {
    private ValidatorException()
    public ValidatorException(String text, Collection failedKeys)
    public ValidatorException(String text, Throwable cause,
                               Collection failedKeys)
    public ValidatorException(Throwable cause,
                              Collection failedKeys)
    public Enumeration getFailedKeys()
}
```
The `WindowStateException` Class

```java
public class WindowStateException extends PortletException {
    public WindowStateException(String text, WindowState state)
    public WindowStateException(String text, Throwable cause,
                               WindowState state)
    public WindowStateException(Throwable cause, WindowState state)
    public WindowStateException() {
    }
    public WindowState getState()
    }
```
The `WindowState` Class

```java
public class WindowState {
    public final static WindowState NORMAL
    public final static WindowState MAXIMIZED
    public final static WindowState MINIMIZED
    public WindowState(String name)
    public String toString()
    public int hashCode()
    public boolean equals(Object object)
}
```
The PortletMode Class

```java
public class PortletMode {
    public final static PortletMode VIEW
    public final static PortletMode EDIT
    public final static PortletMode HELP
    public PortletMode(String name)
    public String toString()
    public int hashCode()
    public boolean equals(Object object)
}
```
public interface PortalContext
{
    public String getProperty(java.lang.String name);
    public Enumeration getPropertyNames();
    public Enumeration getSupportedPortletModes();
    public Enumeration getSupportedWindowStates();
    public String getPortalInfo();
}
Installing Pluto

- Download binary distribution
- Extract
- Ensure that JAVA_HOME=<JAVA 5>
- Start tomcat
- Browse [http://localhost:8080/pluto/portal](http://localhost:8080/pluto/portal)
Installing Jetspeed-2

- Download binary distribution
- Extract
- Start database
- Start tomcat
Installing Gridsphere

- Install Ant binary distribution
- Download binary distribution
- Extract
- Follow the Gridsphere QuickStart guide
  - The easy way:
    - http://www.gridsphere.org
    - Select the download tab
    - Select “Gridsphere QuickStart Portal”