

JSR-286: Portlet Specification 2.0

for Portal and Portlet Developers

Ate Douma

Apache Software Foundation Member
Apache Portals and Apache Wicket Committer & PMC Member

JSR-286 & JSR-301 Expert Group Member

Portal Architect & Project Lead Hippo Portal
Hippo Open Source ECM Software

ate@apache.org / a.douma@hippo.nl

<http://www.hippo-ecm.org>



Preamble

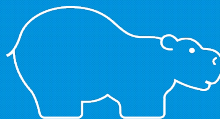
*This presentation only represents the **current** thinking of the JSR-286 Expert Group.*

Although the final specification is almost ready, it still may differ from the content of this presentation.



Agenda

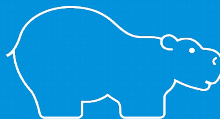
- A short summary of JSR-168: Portlet Specification 1.0
- Limitations of JSR-168
- JSR-286 - Portlet Specification 2.0
- Important new features of JSR-286
- Questions and Answers



JSR-168 - A short summary

PLT.2.1 What is a Portal?

“ A portal is a web based application that -commonly- provides personalization, single sign on, content aggregation from different sources and hosts the presentation layer of Information Systems. ”



JSR-168 - A short summary

PLT.2.2 What is a Portlet?

“A portlet is a Java technology based web component managed by a portlet container that processes requests and generates dynamic content.

Portlets are used by portals as pluggable user interface components that provide a presentation layer to Information Systems.”



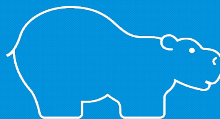
JSR-168 - A short summary

PLT.2.3 What is a Portlet Container

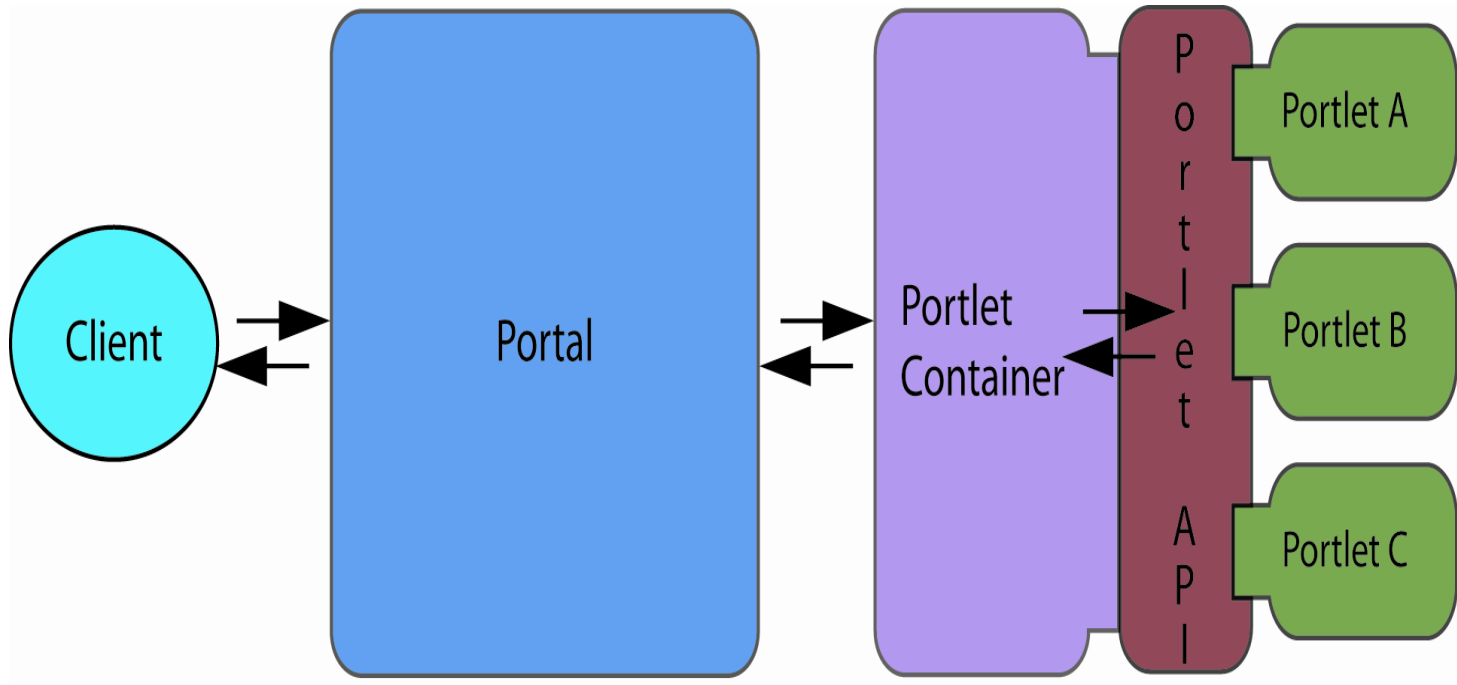
“A portlet container runs portlets and provides them with the required runtime environment. A portlet container contains portlets and manages their lifecycle.”

...

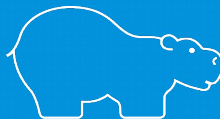
“A portlet container receives requests from the portal to execute requests on the portlets hosted by it.”



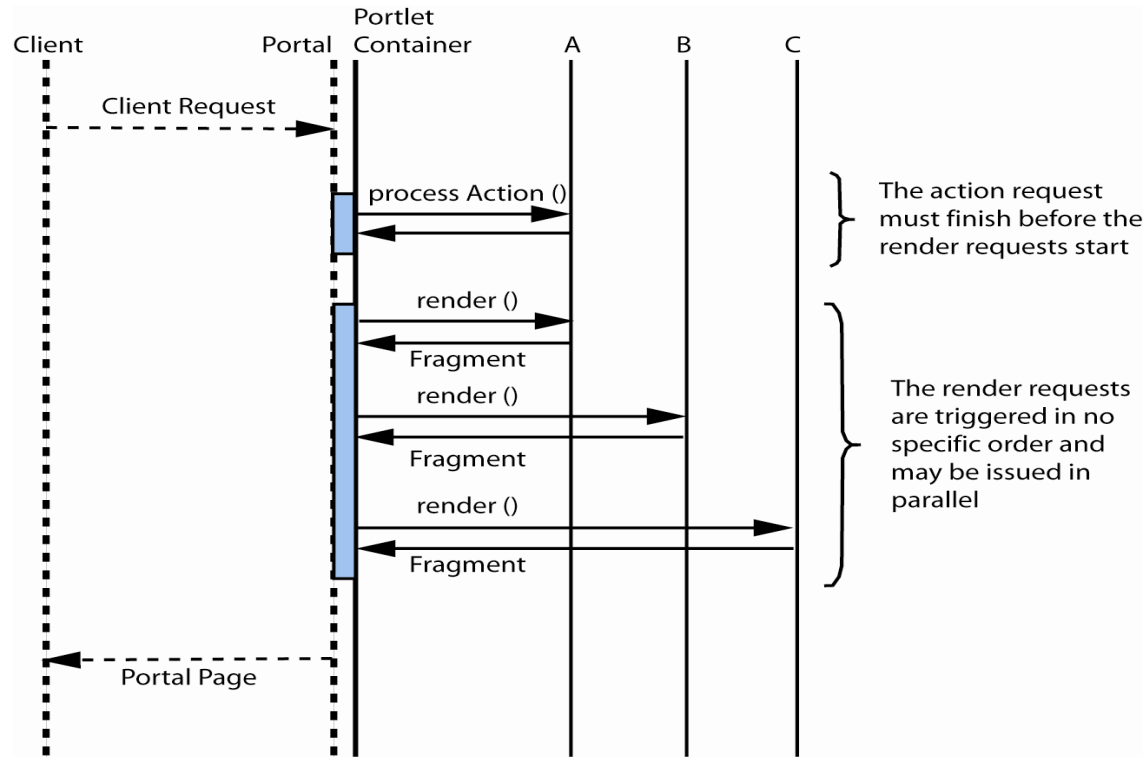
JSR-168 - A short summary



(Portlets and Apache Portals, Manning 2005)



JSR-168 - A short summary



Request processing

(Portlets and Apache Portals, Manning 2005)



Limitations of JSR-168

- inter-portlet communication / portlet coordination
- serving non-markup resources (.pdf, .doc, images etc.)
- portlet filters
- contributing javascript or CSS to <head>; using cookies
- proper support for common web frameworks
- AJAX



Limitations of JSR-168

Inter-portlet communication

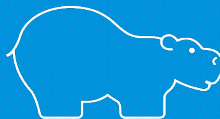
- only supported within the same portlet application using session attributes
- target portlets will only “see” messages during next render request
- portlets cannot (should not) update their state during a render request: “event” handling not really possible



Limitations of JSR-168

Serving non-markup resources

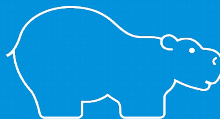
- A portlet can only render markup fragments
- Direct URL access to a portlet is not possible
- Requires direct servlet URLs for serving resource requests outside the markup fragment
- Might require (session) state coordination, like for security, between the portlet and the direct servlet requests



Limitations of JSR-168

Contributing to <head>, setting cookies

- Javascript or CSS can only be embedded within the content markup; no body onLoad handling hooks
- No way to optimize or prevent duplicate contributions
- The API forbids adding cookies: only client side setting of cookies using javascript is possible

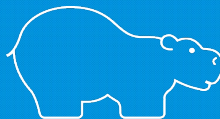


Limitations of JSR-168

Proper support for common web frameworks

- A lot of web frameworks are still primarily (or only) Servlet API oriented
- Servlet dispatching not supported from processAction
- Needs Portals Bridges or similar solutions
- Separate Action and Render request phases are difficult to incorporate or to migrate to
- “Unaware” servlets only “see” the application scope session
- JSP/JSTL support very limited:

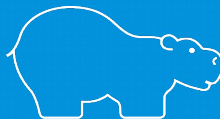
```
<c:out value="<%= ((FooBean)renderRequest.getSession()
                .getAttribute("fooBean",PortletSession.PORTLET_SCOPE))
                .getBeanValue() %>" />
```



Limitations of JSR-168

AJAX

- Portable solutions only using servlets are inefficient and ugly
- Many portals do provide native AJAX support now
- Native AJAX support with access to the portlet environment requires custom / portal specific extensions
- But none of that is endorsed by the spec.



JSR-286 - Portlet Specification 2.0

- Expert Group started January 2006
- 1st Early Public Draft released August 2006
- 1st Public Review released July 2007
- Non-official drafts are provided regularly too
- Latest (last?) non-official draft at revision 28, d.d. 2007-10-31, feature complete now
- Now working on last refinements and corrections
- Final Release still scheduled before end of 2007
- RI done under Apache Pluto umbrella with help from a developers group at University of Jena



JSR-286: Important new features

- Java 5, alignment with J2EE 1.4 and WSRP 2.0
- Portlet coordination (Public Render Parameters, Portlet Events)
- Direct Portlet access using a ResourceURL with full control over the response
- Portlet Filters
- Extended Caching (private/shared, expiration/validation caching)
- Java 5 annotation support in GenericPortlet
- Access to the Portlet Window ID
- PortletURL generation callback listeners
- support for page <head> contributions and setting Cookies
- “private” PortletModes and setting nextPossiblePortletModes
- PORTLET_SCOPED session attributes for dispatched Servlets
- Providing ActionRequest attributes to the RenderRequest
- Much improved support for common web frameworks



JSR-286: Public render Parameters

- Coordination of render state across web applications
- Limited to String values
- Defined in portlet.xml using Qnames:

```
<public-render-parameter>
  <identifier>foo</identifier>
  <qname xmlns:x="http://acme.com/parameters-2.0">x:foo.2</qname>
  <alias xmlns:y="http://acme.com/parameters-1.0">y:foo.2</qname>
</public-render-parameter>
```

```
<portlet>
  <portlet-name>PortletA</portlet-name>
  ...
  <supported-public-render-parameter>foo</supported-public-render-parameter>
</portlet>
```



JSR-286: Portlet Events

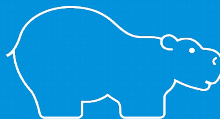
- Declared in portlet.xml using Qnames or a default namespace
- A portlet specifies which events it wants to receive or send

```

<default-namespace>http://acme.com/events-2.0</default-namespace>
<event-definition>
  <name>foo</name>
  <alias xmlns:y="http://acme.com/events-1.0/">y:foo</alias>
  <java-class>java.lang.String</java-class>
</event-definition>

<portlet>
  <portlet-name>Portlet-A</portlet-name>
  ...
  <supported-processing-event>
    <name>foo</name>
  </supported-processing-event>
  <supported-publishing-event>
    <name>foo</name>
  </supported-publishing-event>
</portlet>

```



JSR-286: Portlet Events (cont.)

- Wildcard support for matching to declared events:
 - “foo.event.” matches both “foo.event.one” and “foo.event.two”
- The portal or portlet container can also send events
- New 3rd lifecycle phase: processed before rendering
- Can be send from both `processAction` and `processEvent`
- State changes are allowed during `processEvent`
- The portal / portlet container will act as broker
- Formally not 100% reliable, i.e. no guarantee of delivery (mainly for WSRP, “local” portals are less limited)
- API:

```
EventPortlet.processEvent(EventRequest req, EventResponse res)
StateAwareResponse.setEvent(QName name, Serializable value)
StateAwareResponse.setEvent(String name, Serializable value)
```

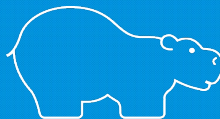


JSR-286: Resource Serving

- Invoking a Portlet directly (but still through the Portal)
- API:

```
MimeResponse.createResourceURL()  
ResourceServingPortlet.serveResource(ResourceRequest, ResourceResponse)
```

- Extends the render phase, NOT a new life-cycle phase
- POST, PUT, DELETE supported
- Should not change shared portlet state
- Additional URL parameters are specific for the request
- Full control over request and response headers
- Can be used for binary data or “readonly” AJAX but cannot change navigational state



JSR-286: Portlet Filters

- Very similar to Servlet Filters both in definition and API
- Filters can be applied to multiple lifecycles
- Filters can be chained like Servlet Filters
- Wrapper classes for request and response objects
- Wildcards can be used for mapping to multiple portlets

```

<filter>
  <filter-name>Log Filter</filter-name>
  <filter-class>com.acme.LogFilter</filter-class>
  <lifecycle>ACTION_PHASE</lifecycle>
  <lifecycle>EVENT_PHASE</lifecycle>
</filter>

<filter-mapping>
  <filter-name>Log Filter</filter-name>
  <portlet-name>*</portlet-name>
</filter-mapping>

```



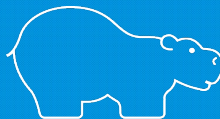
JSR-286: Extended Caching

- Allow public cached content for multiple users

```
<portlet>
  ...
  <expiration-cache>300</expiration-cache>
  <cache-scope>public</cache-scope>
</portlet>
```

- Support validation based caching using ETag
- API:

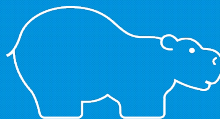
```
response.getCacheControl()
  .getExpirationTime(), .setExpirationTime()
  .isPublicScope(), .setPublicScope(boolean)
  .getETag(), .setETag(String)
  .useCachedContent(), .setUseCachedContent(boolean)
```



JSR-286: Container runtime options

- Defineable in portlet.xml on both application and portlet level
- Queryable from PortletContext
- Four predefined options:
 - javax.portlet.escapeXml (for backwards compatibility)
 - javax.portlet.renderHeaders
 - javax.portlet.servletDefaultSessionScope
 - javax.portlet.actionScopedRequestAttributes

```
<container-runtime-option>  
  <name>javax.portlet.servletDefaultSessionScope</name>  
  <value>PORTLET_SCOPE</value>  
</container-runtime-option>
```



JSR-286: renderHeaders

- Configured with Container runtime option `javax.portlet.renderHeaders` (default false)
- Override the `GenericPortlet.doHeaders` method for:
 - providing page `<head>` section element contributions, like CSS and Javascript `<script>` tags
 - other HTTP/Portal specific response header properties
 - setting Cookies
 - setting `nextPossiblePortletMode`
- API

```
org.w3c.dom.Element PortletResponse.createElement(String tagName)
PortletResponse.addProperty(String key, org.w3c.dom.Element element)
PortletResponse.addProperty(String key, String value)
MimeResponse.addProperty(javax.servlet.http.Cookie cookie)
RenderResponse.setNextPossiblePortletModes(Collection<PortletMode> modes)
```

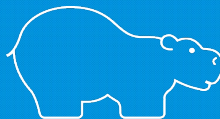


JSR-286: renderHeaders (cont.)

Contributing to <head> example:

```
protected void doHeaders(RenderRequest request, RenderResponse response) {
    org.w3c.dom.Element loadAjaxJS = response.createElement("script");
    loadAjaxJS.setAttribute("type", "text/javascript");
    loadAjaxJS.setAttribute("src", "wicket-ajax.js");
    response.addProperty(MimeResponse.MARKUP_HEAD_ELEMENT, loadAjaxJS);
}
```

(this is actually a bad example as Wicket will provide header contribution support natively and out of the box)



JSR-286: Better web framework support

- Allow servlet dispatching, including using forwards, from:
 - `processAction`
 - `processEvent`
 - `render`
 - `serveResource`
- Optionally providing a `PORTLET_SCOPED` session to Servlets
- Extended JSP tag library using new taglib uri
`"http://java.sun.com/portlet_2_0"`
- Bridging and native integration underway for JSF (JSR-301) and Apache Wicket (1.3+)



JSR-286: actionScopedRequestAttributes

- Action request attributes cached by the container and provided again during the render request
- Support transporting complex objects instead of Strings for which renderParameters should be used
- A required container feature!
- “Back” button supported through a numberOfCachedScopes cache size parameter.

```
<container-runtime-option>
  <name>javax.portlet.actionScopedRequestAttributes</name>
  <value>>true</value>
  <name>numberOfCachedScopes</name>
  <value>10</value>
</container-runtime-option>
```



JSR-286: ~~Native AJAX~~ support

Was intended and discussed at large

- But: will not be part of this spec
- Difficulties:
 - AJAX request identification
 - synchronizing state changes to the client for other portlets on the page
 - XMLPortletRequest or plain XMLHttpRequest?
 - integration with popular AJAX toolkits
 - Too many not yet unknown or fully thought through use-cases
- Instead: possible creation of a new Apache Portals subproject for discussing and defining guidelines and possible solutions in concerto with the WSRP 2.0 team (still TBD)

Let the community determine a feasible API first



JSR-286: ... and more ...

- Annotations: @ProcessAction, @RenderMode, @ProcessEvent

```
@ProcessEvent(qname="{http://acme.com/events}foo")
void processFoo(EventRequest request, EventResponse response)
    throws PortletException, java.io.IOException;
```

- PortletRequest.getWindowID()

- PortletURLGenerationListener

```
void .filterActionURL(PortletURL actionURL)
void .filterRenderURL(PortletURL renderURL)
void .filterResourceURL(PortletURL renderURL)
```

- “unmanaged” (private) custom Portlet modes

```
<custom-portlet-mode>
  <description>Creates content for Cut and Paste</description>
  <portlet-mode>clipboard</portlet-mode>
  <portal-managed>false</portal-managed>
  <decoration-name>ClipboardMode</decoration-name>
</custom-portlet-mode>
```



More information

- This presentation and related information
<http://www.hippo-ecm.org/display/PORTAL/JSR-286>
- JSR-286: Portlet Specification 2.0
<http://jcp.org/en/jsr/detail?id=286>
- Online current early draft (non-official)
<http://ipc658.inf-swt.uni-jena.de/spec/>
- Apache Pluto
<http://portals.apache.org/pluto/>

