



# What's new and cool in Portlet 2.0

Julien Viet

JBoss a division of Red Hat

Project Lead & Founder

15<sup>th</sup> of February 2008

# About me

- Julien Viet
  - JBoss Portal founder and project lead
  - Red Hat representative for the JSR 286 Portlet 2.0 Expert Group

# Agenda

- Introduction
- Portlet Container 2.0
- Portlet 2.0 features
- Product roadmap

# Introduction

# Portals

- Portals provide an integration layer between several heterogeneous applications
  - Provide the user with the capability to use several applications across organizational boundaries
- Integration occurs at the presentation layer
  - Single Sign-On
  - Aggregate syndicated content into a single web page
- Other features
  - Personalization
  - Internationalization
  - Entitlement
  - Collaboration

# Portlets

- A portlet is a pluggable user interface component model living in the portal ecosystem
  - Its minimum behavior is to be able to produce a markup fragment
- A fragmented market
  - Historically Java EE has seen a variety of different component models
  - Beyond Java EE, other middleware stacks have defined their own component models (pre JSR 168)

# Portlet Container

- A Portlet Container is a service that manages a portlet component, and interacts with a portal in a various manner
  - Portlet description
  - Portlet state management
  - Portlet runtime interactions

# The Portlet 1.0 specification

- The Portlet 1.0 specification defines the component model for the Java EE stack
  - Finalized in October 2003
  - Wide market adoption
  - Integrated with the Java EE stack but not part of it

# The Portlet 1.0 specification

- The portlet application defines the packaging unit as a standard war file, that contains a portlet.xml file in its **WEB-INF** directory
- Enable the portlet to participate in the global portal request/response cycle
- Provide advanced personalization features
  - Portlet preferences
  - Portlet metadata
    - Description
    - Categorization
    - Capabilities
    - etc...

# The Portlet 1.0 specification

- Portlet 1.0 defines the bare minimum:
  - Portlet integration within an aggregated page
  - An obvious lack of coordination between portlets
  - Problems with servlet based frameworks, the creation of portlet bridges are not easy

# The Portlet 2.0 specification

- Portlet 2.0 specification
  - Started 29th of November 2005
  - Led by IBM
  - Involvement from all major portal vendors
  - More information at <http://jcp.org/en/jsr/detail?id=286>
  - Final draft currently submitted to the JCP

# JBoss Portlet Container 2.0

# JBoss Portlet Container 2.0

- Aka portlet module in SVN
  - Detached from the JBoss Portal codebase in August 2007
  - Server agnostic
    - It is not tied to any app server
    - Deploys in JBoss 4.2 and Tomcat 6
- Portlet 2.0 status
  - 97% spec feature complete
  - 100% useful feature complete
  - Heavily tested (more than 400 unit tests)
  - Certification effort has not yet started (waiting for TCK)

# JBoss Portlet Container 2.0

- Provides a simple out of the box portal to quickly build pages
  - You only get the portlet container
  - Defines a flexible JSP taglib to build pages

```
<html><body><portal:page>
  ...
  <portal:portlet name="MyPortlet" applicationName="in_my_war_file">
    ...
    <portal:portlettitle/>
    ...
    <portal:portletmarkup/>
  ...
</portal:portlet/>
...
</portal:page></body></html>
```

# JBoss Portlet Container 2.0

- Project page
  - <http://labs.jboss.com/portletcontainer>

# The spec

# Portlet 2.0 features

- Portlet coordination
  - Eventing
  - Public render parameters
- Resource serving
- Other features that matter

# Public render parameters

- A very powerful feature
  - Easy to use
  - Flexible
  - Allows the caching of fragments
- A portlet exposes its render parameters
  - The developer declares the public parameters in the deployment descriptor of the portlet
  - The scope of sharing is determined by the portal : page, site, ...

# Public render parameters

- Use cases
  - Several portlets on the same page share the same zipcode, and are able to display informations related to the geographic location
    - Weather
    - Map
    - Etc...
  - Contextualization of a portlet by the portal
    - Pages could have properties bound to portlets public render parameters

# Public render parameters use case

```
<portlet-app>
  <portlet>
    <portlet-name>GoogleMap</portlet-name>
    ...
    <supported-public-render-parameter>zipcode</supported-public-render-parameter>
    ...
  </portlet>

  <public-render-parameter>
    <identifier>zipcode</identifier>
    <qname xmlns:g='urn:google'>g:zipcode</qname>
  </public-render-parameter>
</portlet-app>
```

## Public render parameters use case

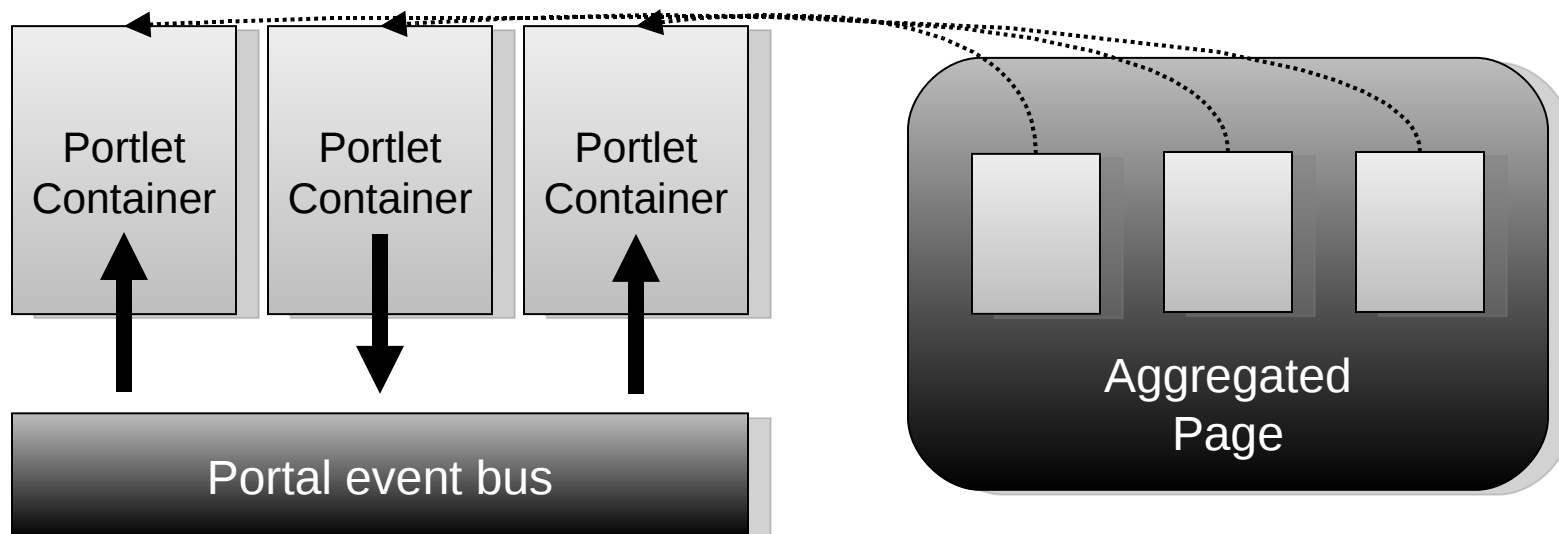
```
public void render(RenderRequest req, RenderResponse
resp) {
    String zipcode = req.getParameter("zipcode");
    render(zipcode, resp);
}

public void processAction(ActionRequest req,
ActionResponse resp) {
    String newZip = req.getParameter("zipcode");
    resp.setRenderParameter("zipcode", newZip);
}
```

# Portlet coordination via events

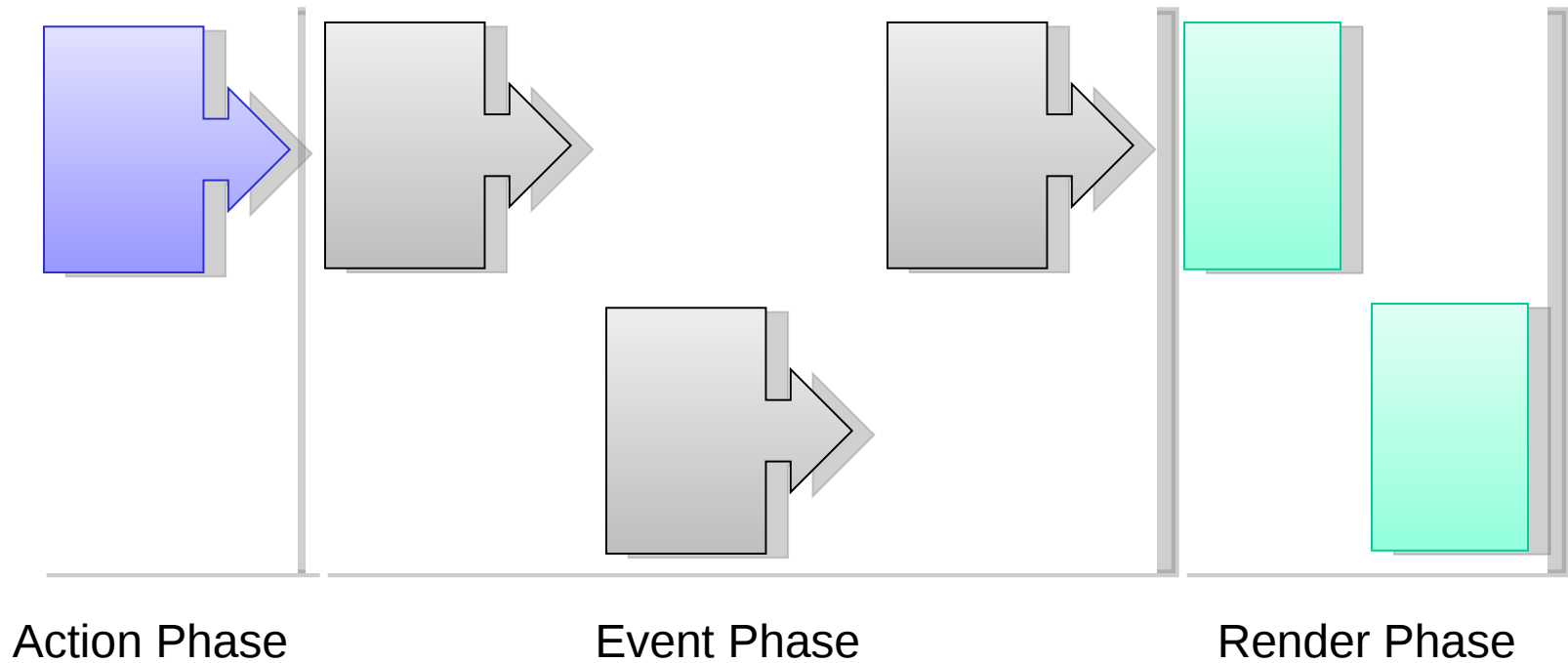
- Really coordination events
  - not business events (use JMS instead)
  - not reliable, e.g no guarantee of delivery
- Declarative
  - What I can process
  - What I can publish
- Event routing is mediated by the portal
- Used in different life-cycle phases
  - Action phase can publish events
  - Event phase can process/publish events

# Portlet event routing is orchestrated by the portal



```
<portlet-app>
  <event-definition>
    <name>product-selected</name>
    <value-type>com.myshopping.Item</value-type>
  </event-definition>
</portlet-app>
```

# Portlet event routing is orchestrated by the portal



# Portlet publishing event

```
public void processAction(ActionRequest req,  
ActionResponse resp) {  
  
    Item item = getClickedItem(req);  
    resp.setEvent("product_selected", item);  
  
}
```

```
<portlet>  
    <supported-publishing-event>  
        <name>product_selected</name>  
    </supported-publishing-event>  
</portlet>
```

# Portlet processing event

```
public void processEvent(EventRequest req,  
EventResponse resp) {  
  
    Event event = req.getEvent();  
    Item item = (Item)event.getPayload();  
    ShoppingCart cart = getShoppingCart(req);  
    cart.addItem(item);  
  
}
```

```
<portlet>  
    <supported-processing-event>  
        <name>product_selected</name>  
    </supported-processing-event>  
</portlet>
```

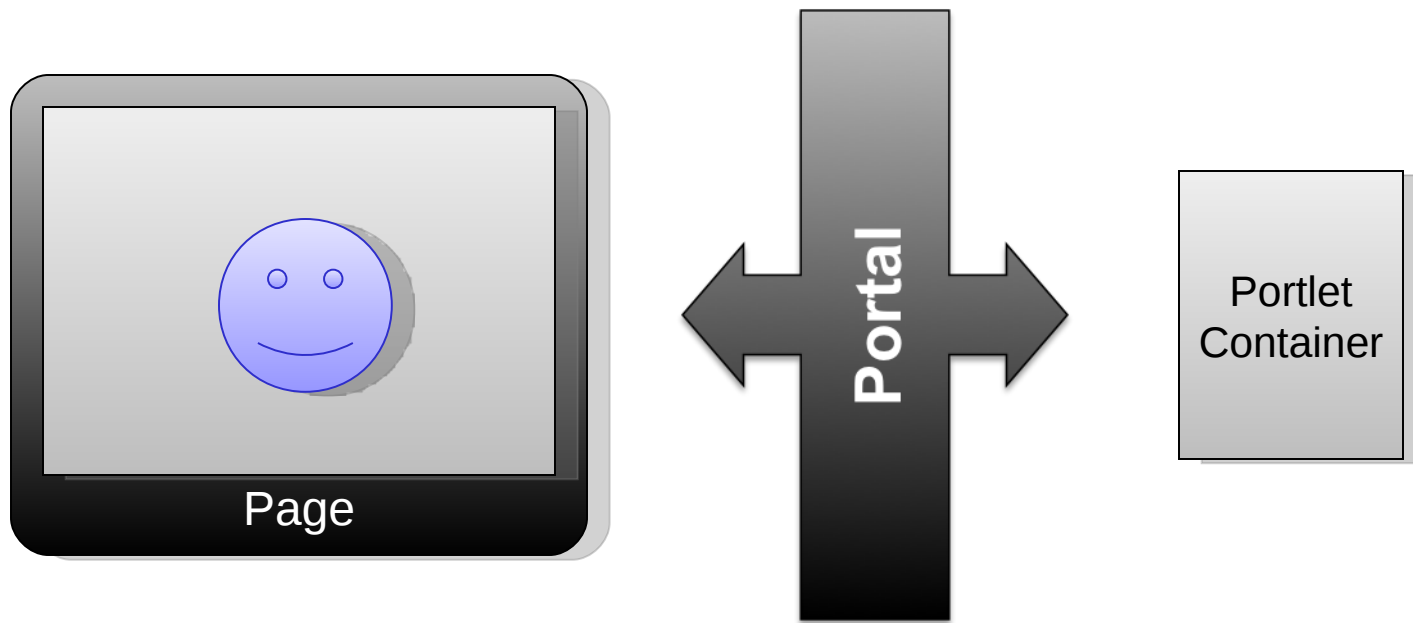
# Portlet events

- Use cases
  - Decoupling of different applications that need to communicate
  - You are already using an application that can generate events, and you want to integrate it with that application
  - Communication between the portal and the portlet
    - Login/Logout
    - Activities
- Antipatterns
  - Event usage for updating render parameters
    - Use public render parameters instead
  - Do not abuse
    - A carefully designed application can avoid to use events
  - Event cascading cycling forever
  - Not a messaging system

# Resource serving

- Resource serving offers the portlet the opportunity to act as a servlet
  - The portlet implements the **javax.portlet.ResourceServingPortlet** interface
  - **ResourceURL** triggers resource serving
    - created during the render phase or the resource phase
    - Uses a resource ID and a set resource parameters
  - The portal proxies the request/response
  - Full access to the current context
    - Preferences
    - Render parameters, the window state, and the mode that created the resource URL
    - Security
  - Full control over the response

# Resource serving



# Resource serving

```
public void render(RenderRequest req, RenderResponse
resp) {
    ...
    ResourceURL url = resp.createResourceURL();
    url.setResourceID("foo");
    url.setParameter("juu", "daa");
    writer.write(url.toString());
    ...
}
```

```
public void serveResource(ResourceRequest req,
ResourceResponse resp) {
    String id = url.getResourceID();
    ...
}
```

# Resource serving

- Use cases
  - AJAX
    - Server JSON or markup fragment
    - Read only use case (!)
    - Coordination and state update (session, render parameters, preferences) is not adressed
  - Window popup that has control over the full page markup like a configuration wizard
  - Serve a binary file, a stylesheet, or a script
- Antipatterns
  - Don't serve static files
  - Don't stream files as they can be buffered by the portal and the portlet container, resulting in memory issues. Use a servlet instead

# Sticky action request attributes

- The portlet sets request attributes during the process action
  - Creates a new scope with an action ID injected in the render parameters
  - Objects are really stored in the portlet session
- The portlet should receive the request attributes in the subsequent phases
  - Based on the action ID read from the render parameters
  - Provide identical rendering conditions when page refresh occurs
  - Several scopes of objects may exist in parallel
  - The portlet container keep only a limited set of scopes in session

# Sticky action request attributes

- Use cases
  - Create objects during the action and use them in other phases
- Antipatterns
  - It is an antipattern by itself :-)
    - Portlet should rather rely on render parameters
    - Designed for bridges like Struts that heavily relies on this broken model
    - On a best effort basis: no guarantees that objects will be present later

# GenericPortlet improvements

- Dispatch to annotated methods
  - Action phase
    - Action url needs to carry a parameter named **ActionRequest.ACTION\_NAME**
    - **@ProcessAction("foo")**
  - Event phase
    - Dispatch to the event qname **"{" + nsURI + "}" + localPart** or to event local name
    - **@ProcessEvent(qname="{mysns}foo")**
    - **@ProcessEvent(qname="{mysns}foo.")**
    - **@ProcessEvent(name="foo")**
    - **@ProcessEvent(name="foo.")**
  - Render phase
    - precedence over the old doView / doEdit / doHelp methods
    - **@RenderMode("edit")**

# GenericPortlet improvements

- Default resource serving that attempts to perform a request dispatching using the provided resource ID

## Other features that matter

- Full request/response header access
  - Page headers allow to inject headers in the page
  - Read/write HTTP headers
  - Read/write cookies
  - No guarantees, the portal may filter
- Portlet filters
  - Just like servlet filters
- Caching improvements
  - Resource caching
  - Cache re-validation

## Other features that matter

- Window ID available in the portlet request
  - Can be used to perform per portlet window caching
- More powerful request dispatching to servlets
  - Targeted for bridge development
  - Extended to all phases
  - Allows scoping of session to application or portlet
- Next possible modes
  - The portlet can specify the next possible modes available

# Product roadmap

# Product roadmap

- Portlet Container 2.0 Beta released
- Portlet Container 2.0 GA during Q2/2008
- Support for JBoss Portal 2.7 during Q3/2008

# Shameless plug

- Peter Johson (also known as PeterJ in our forums) has a session at 11am this morning

# Conclusion

- Portlet 2.0 Specification
- JBoss Portlet Container 2.0

**Q/A**

Q&A

Q&A