What's New in the Java™ Portlet Specification 2.0 (JSR 286)?

Stefan Hepper
JSR 286 Spec Lead
IBM Corp.

Wesley Budziwojski
JSR 286 EG Member
Sun Microsystems, Inc.

Session TS-4225

Goal of This Talk

Learn what the Java™ Portlet Specification v.2.0 provides and how you can leverage these capabilities in your portlets

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

Major Themes

Where do we want to go from v.1.0?

- v.1.0 (JSR 168)
  - Provide the programming model for standalone, pluggable UI application components
- v.2.0 (JSR 286)
  - Enable coordination between portlets and allow building composite applications based on portlet components
  - Allow for a better user experience using AJAX patterns
  - Alignment with Web Service for Remote Portlets (WSRP) 2.0

JSR= Java Specification Request

JSR 286

Details on the Expert Group (EG)

- IBM is leading this JSR, all major Java technology portal (commercial and open source) vendors in the EG
- Expert Group members:
  - Apache, BEA, R. Butler, P. Dabke, D. DeWolf, C. Doremus, A. Douma, eXo, S. Frid, IBM, JBoss, Liferay, K. Mann, S. Millidge, Novell, J. Novotny, Oracle, P. Pandey, S&N, SAP, C. Severance, H. Suleiman, Sun, SunGard Higher Education, TIBCO, University Jena, Vignette
- Reference implementation will be provided at Apache
  - As Apache Pluto 2.0
    - http://portals.apache.org/pluto
- TCK will be available for free
  - Will extend the JSR 168 TCK

JSR 286

Details on the schedule

Schedule

Kick-off:	February 2006
First draft with base features:	July 2006
Second draft:	April 2007
Public draft (planned):	June 2007
Final version (planned):	August 2007

More information at

- http://jcp.org/en/jsr/detail?id=286
- http://ipc658.inf-swt.uni-jena.de/spec/
    - Contains the most current version of spec and API

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

Overview

Why is coordination so important?

- The #1 complaint about v.1.0 was the missing capability to send events between portlets
  - V.1.0 only has the portlet application session scope for coordination
  - Only usable within the one portlet application, not across portlet applications
- V.2.0 will add additional coordination capabilities
  - Eventing
  - Public render parameters across portlets
- Coordination allows business users building composite applications out of portlet components
  - Can be done at runtime, without programming

Events

Overview

- JSR 286 introduces a loosely coupled event paradigm
    - A portlet can declare events it wants to receive and events it wants to emit
    - The portal/portlet container will act as broker and distribute the events accordingly
    - Allows wiring of portlets at runtime
    - Dynamic event declaration only for sending events
- Event handling will be an additional step in the overall action phase
    - State changes are allowed
    - Event handling must be finished before rendering starts

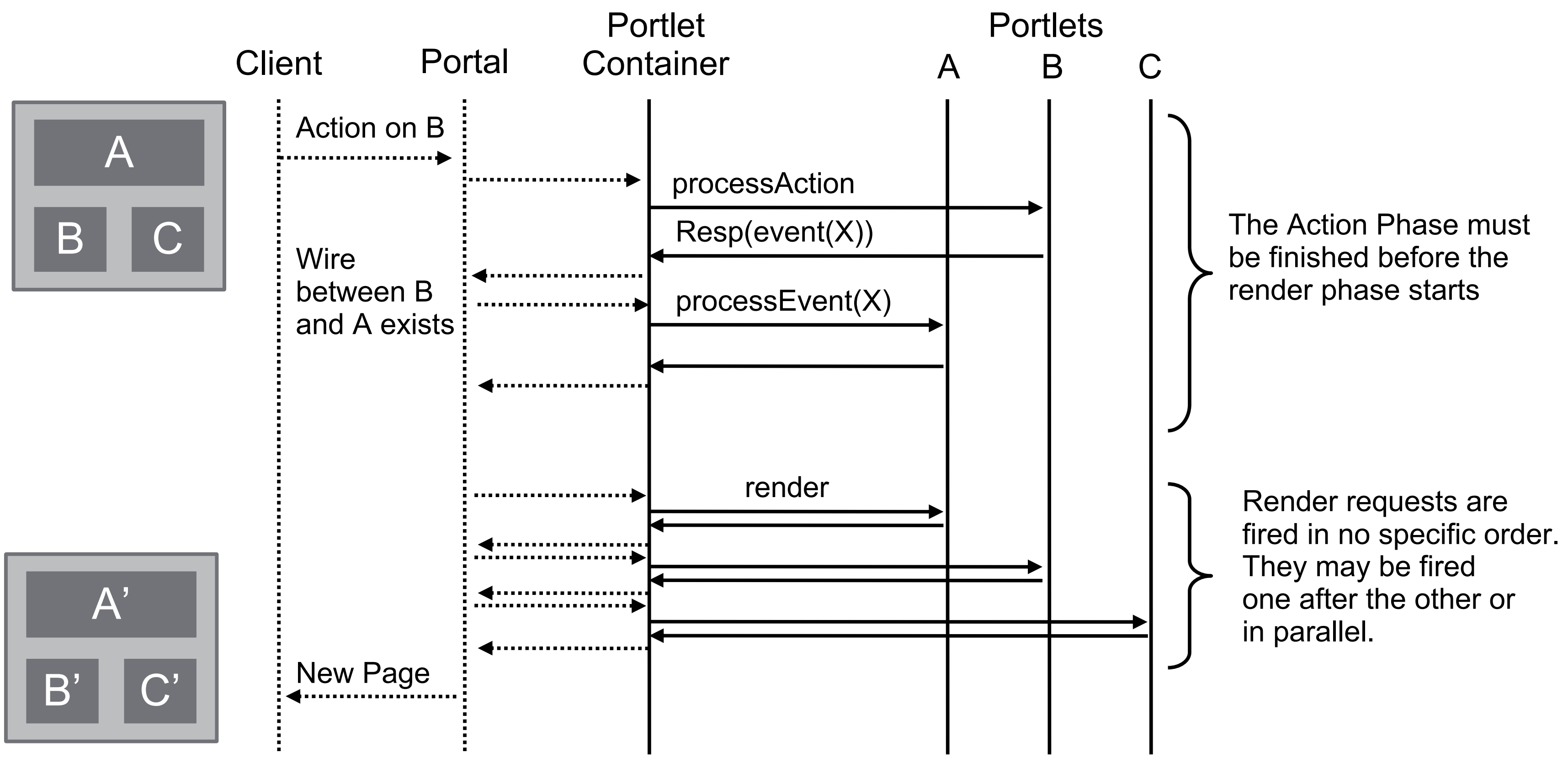
Events

Overview

- Event types
  - Can be complex, but must be Java technology and Java Architecture for XML Binding (JAXB) serializable
  - Note: String or XML simple types as type is strongly recommended in as complex types introduce coupling between portlet components
  - Use complex types only as last resort
- Event names
  - Are defined as QNames in the DD

Events

Request flow



Client
Portal
Portlet Container
Portlets
A
B
C
A
B
C
Action on B
processAction
Resp(event(X))
Wire between B and A exists
processEvent(X)
render
New Page
A'
B'
C'
The Action Phase must be finished before the render phase starts
Render requests are fired in no specific order. They may be fired one after the other or in parallel.

.... Not defined by the Java Portlet Specification

Code Sample for Sending an Event

```
Event defined in the DD:
<event-definition>
    <name xmlns:x="http:acme.com/events">
        x:Address.Created
    </name>
    <java-class>com.acme.Address</java-class>
</event-definition>

<portlet>
    <supported-publishing-event>
        <name xmlns:x="http:acme.com/events">
                x:Address.Created
        </name>
    </supported-publishing-event>
</portlet>
```

Event Processing in the Portlet

```
@XmlRootElement
    public class Address implements Serializable {
       private String street; private String city;
       public void setStreet(String s) {street = s;}
       public String getStreet() { return street;}
       public void setCity(String c) { city = c;}
       public String getCity() { return city;}
    }
void processEvent(EventRequest req, EventResponse resp) {
    ...
    Address sampleAddress = new Address();
    sampleAddress.setStreet("myStreet");
    sampleAddress.setCity("myCity");
    QName name = new QName ("http:acme.com/events",
                               "Address.Created");
    resp.setEvent(name, sampleAddress);
}
```

Code Sample for Receiving an Event

```
event defined in the DD:
<event-definition>
    <name xmlns:x=”http:acme.com/events”>
        x:CustomerID.Changed
    </name>
    <java-class>java.lang.String</java-class>
</event-definition>

<portlet>
    <supported-processing-event>
        <name xmlns:x=”http:acme.com/events”>
                x:CustomerID.Changed
        </name>
    </supported-processing-event>
</portlet>
```

Event Processing in the Portlet

```
void processEvent(EventRequest req, EventResponse resp)
{
    ...
    Event event = req.getEvent();
    if ( event.getName().getLocalPart().
              equals("CustomerID.Changed") )
    {
        String payload = event.getValue();
        ...
    }
```

Public Render Parameters

Overview

- Allow render parameters to be shared across portlets
  - Not restricted to the portlet application
  - May be even across pages
  - Lightweight coordination based on HTTP GET (contrary to events which have POST semantics)
- Example
  - The zip code of a selected city allowing different portlets (map, tourist information, weather) to display information for this city

Public Render Parameters

Overview

- Semantic is that these are visible to the portal and allowed to be shared with other components
- Re-use existing render parameter APIs
  - Allows to even enable JSR 168 portlets to use public render params by just giving them an JSR 286 DD
- Define in the portlet.xml which render parameters are public
  - Has an simple string ID that the portlet can use in the code
  - Provides a QName and optional alias names for wiring the parameter
  - Allow getting all public params via the PortletContext at runtime

Public Render Parameters

Public render parameters versus events

- Advantages of using public render parameters
  - Less processing overhead, no action phase required
  - Parallel rendering of portlets possible
- Limitations when using public render parameters
  - Only defines new view state, no server side state changes (HTTP GET semantics)
  - No active notification that something has changed
- As public render parameters can be encoded in the URL this allows for
  - Bookmarkablity
  - Support of browser back/forward button
  - Caching in the browser

Code Sample for Public Render Parameters: Deployment Descriptor

```
<public-render-parameter>
    <identifier>zip</identifier>
    <name xmlns:x="http://acme.com/params">
        x:address.zipcode
    </name>
</public-render-parameter>
<portlet>
    <portlet-name>portletA</portlet-name>
    …
    <supported-public-render-parameter>zip
    </supported-public-render-parameter>
</portlet>
```

DEMO

Coordination

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

Overview

v.1.0 versus v.2.0

- Resource serving in JSR 168: Direct serving via the portal/portlet container
  - Done using encodeURL(resourceURL)
  - No portlet runtime context available
- New in JSR 286: Resource serving via the portlet
  - New ResourceURLs that trigger a new lifecycle method serveResource
  - Portlet context available (render params, portlet mode, window state, preferences...)
  - No state changes on portlet container managed state allowed
  - Protected via the portal access control

Resource Serving via the Portlet

Details

- Different cache levels of resource URLs
  - For supporting caching of the resource at the browser
  - Three types introduced: FULL, PORTLET, PAGE
- Resource Ids
  - You can set a specific resource ID on a resource URL
  - Default behavior of GenericPortlet is to try to forward the resource serving to the resource ID specified

Resource Serving via the Portlet

API

- Resource URLs
  - ResourceURL
    - setResourceID(String id)
    - setCacheability(String cacheLevel)
    - cacheLevels: full, portlet, page
- New lifecycle interface
  - ResourceServingPortlet
    - void serveResource (ResourceRequest req,
      ResourceResponse resp)
  - ResourceRequest
    - Like render request + ability to get uploaded data
    - ResourceResponse
    - Like render response + full control over the output stream

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

AJAX Usages in a Portal Environment

Portlets run in an aggregated environment—JSR 168

- Portlet level
  - Portlet brings its favorite AJAX library
  - Portlet has to manages end-point on its own
  - JSR 168 is limited
    - Response without portlet context (served via servlet)
    - No state changes for state managed by the portlet container
- Portal level
  - Portal does an aggregation on the client (browser)
  - Portal manages AJAX interaction
  - Transparent to the portlet
  - Possible with JSR 168 portlets

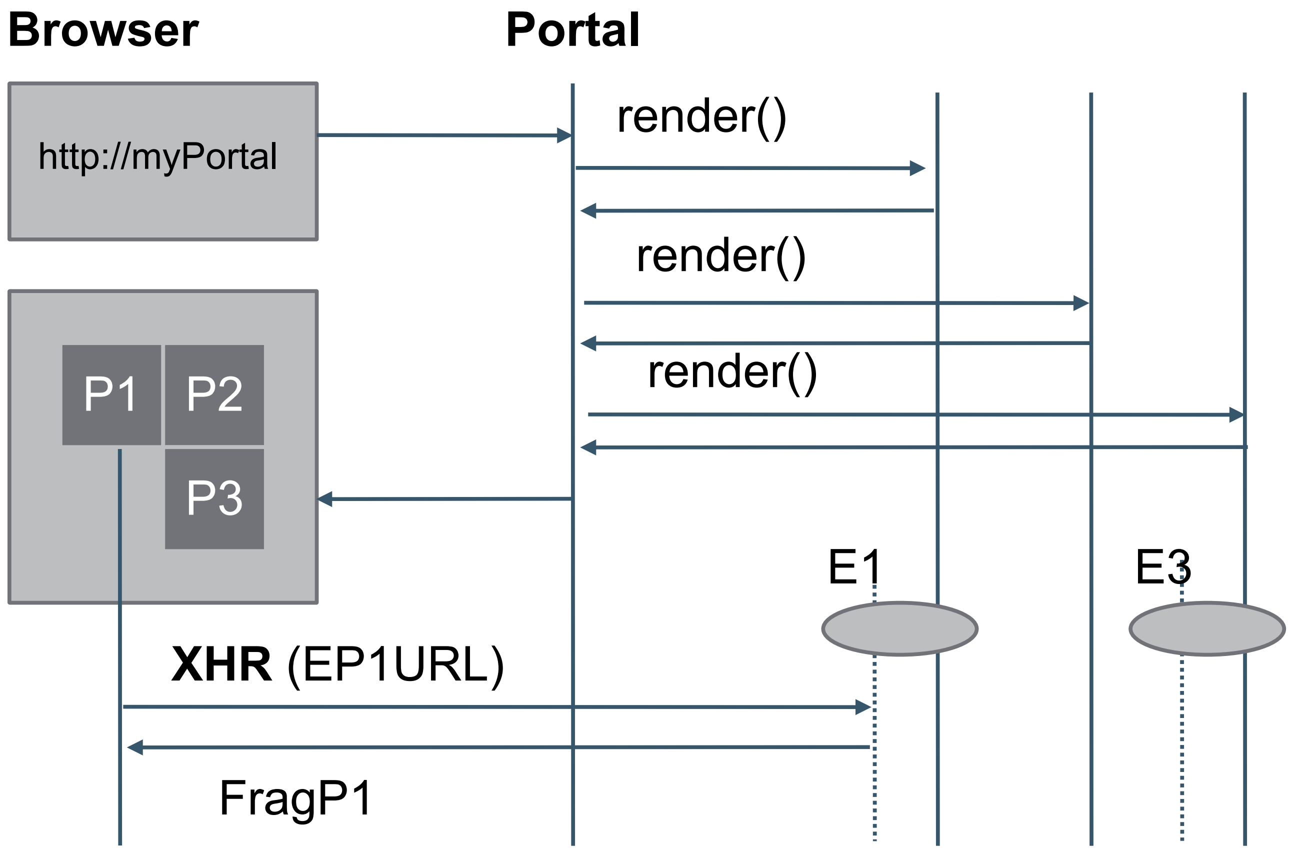
AJAX Usages in a Portal environment

Portlets run in an aggregated environment—JSR 286

- Portlet owned AJAX calls
  - Full access to the portlet state
    - Via XmlHttpRequest and ResourceURLs
  - Functionality restricted
    - No state changes for navigational state
    - No support for events
- Coordinated between portal and portlet
  - Not covered by JSR 286
  - Will be defined as an extension on-top of serveResource
  - Needs to include a client side library that the portlet can leverage

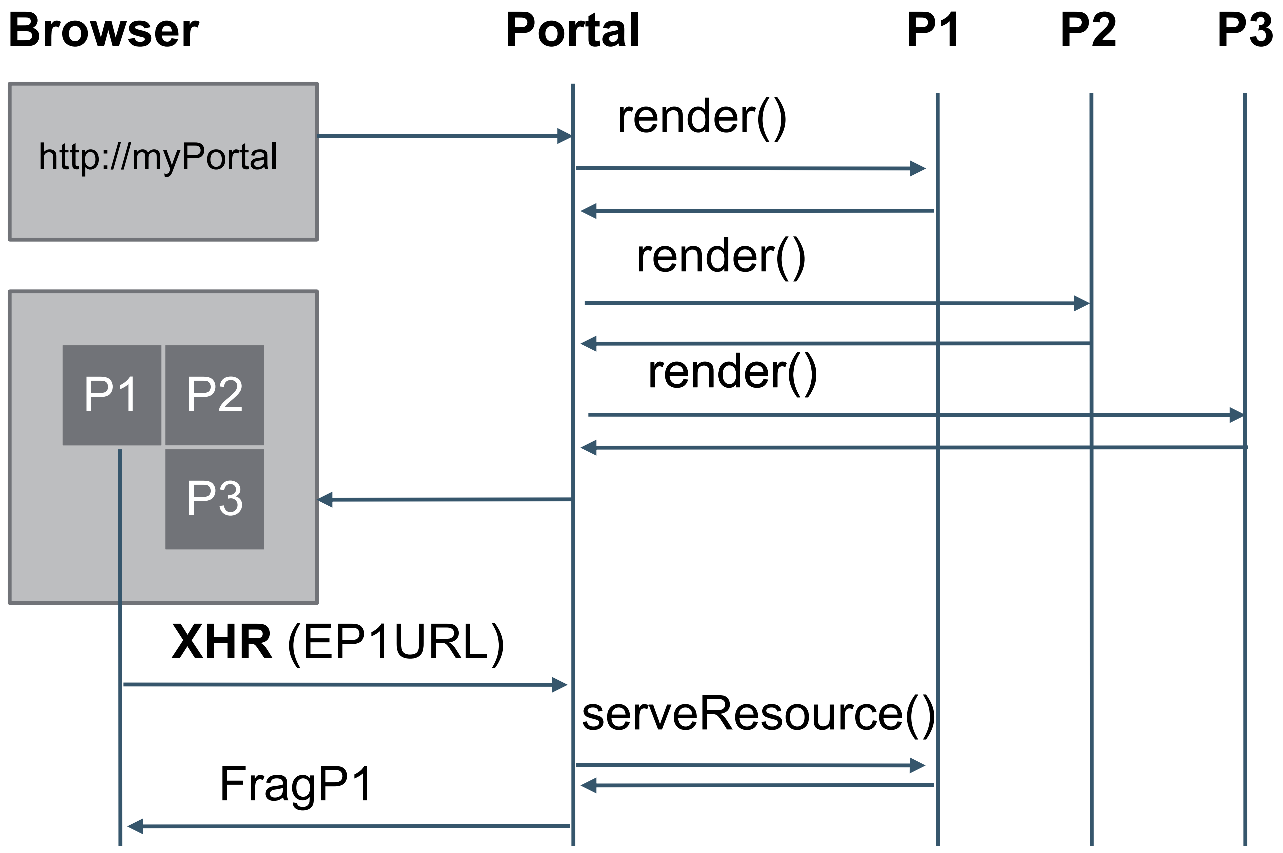
Fragment Serving JSR 168

Request flow with non-portlet endpoints



Fragment Serving JSR 286: Resource URLs

Request flow via serveResource call



Code Sample for AJAX: Client Code

```
<form id="bookFF" method="post"
action="javascript:bookFlightForm('myForm1','bookingResult')
">

 <table>
   <tr>
     <td>Guest name:</td>
     <td><input name="firstName" type="text"/></td>
   </tr>
   <tr>
     <td>Flight number</td>
     <td><input name="flightNumber" type="text"/></td>
     <td></td><td><input name="submit" type="submit"/></td>
    </tr>
 </table>
</form>
```

Code Sample for AJAX: Client Code

```
function bookFlightForm(formId, resultId)
{
   var url = <%=renderRequest.createResourceURL()%>;
   var form = document.getElementById(formId);
   var request = new XMLHttpRequest();
   request.onreadystatechange = function()
   {
     if (request.readyState == 4 && request.status == 200) {
        // update the form with flight confirmation num
     }
   };
   request.open('POST', url, true);
   request.setRequestHeader("Content-Type", "application/x-
www-form-urlencoded");
   var query = encodeForm(form); // encode form using
application/x-www-form-urlencoded
   request.send(query);
}
```

Code Sample for AJAX: Portlet Code

```
void serveResource (ResourceRequest req, ResourceResponse
resp) {
    if ((req.getParameter(“firstName”) != null) &&
         (req.getParameter(“flightNumber”) != null))
    {
        // process the fragment request
        // e.g. Store data in portlet preferences
        ...
        // return markup for fragment request
    ...
}
}
```

DEMO

AJAX

Agenda

Setting Cookies and HTTP Headers

Servlet developers dreams come true...

- Portlets in v.1.0 could not set cookies and HTTP headers
  - Portal has the control over the output stream to the client and body content may already be written
- Portlets in v.2.0 can set cookies and HTTP headers
  - In all lifecycle methods
  - Also available for render response
    - But may be overridden by the portal or other portlets
  - Restrictions on cookies
    - Cookies may be stored on the portal or get re-written and thus not accessible on the client

Setting Cookies and HTTP Headers

API

- HTTP headers
  - Setting: via the set/add property methods on the response
  - Retrieving: via the getProperty methods on the request
- Cookies
  - Setting: addProperty(javax.servlet.http.Cookie)
  - Retrieving: javax.servlet.http.Cookie[] getCookies()

Supporting Setting Headers/Cookies in Render

Splitting render in two parts

- Headers/Cookies needs to be set before the document body starts
  - Buffer all output and at the end create the response to the client
  - Split render into two parts: Headers and markup
- JSR 286 allows portals to set a render request attribute RENDER_PART with values
  - RENDER_HEADERS for setting headers, cookies, title
  - RENDER_MARKUP for rendering the markup
- GenericPortlet takes care of this request attribute
  - Calls doHeaders and setTitle for RENDER_HEADERS
  - Calls dispatch to doXYZ for RENDER_MARKUP

Code Sample for Headers

```
Setting cookies
public class MyPortlet extends GenericPortlet {
...
protected doHeaders(RenderRequest req, RenderResponse resp) {
    Cookie cookie = new Cookie("myCookie", "42");
    resp.setProperty(cookie);
}
Retrieving cookies
public class MyPortlet extends GenericPortlet {
...
protected doView(RenderRequest req, RenderResponse resp) {
    Cookie[] cookies = req.getCookies();
    if ( cookies != null ) {
           // find my cookie in the array and retrieve
           // value with cookie.getValue()
        }
```

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

Portlet Filters

Overview

- Portlet filters and request/response wrappers available
  - Similar in large parts to the servlet filter model
  - Filters are declared in the DD via the filter and filter-mapping element
  - Filters are restricted to one of the portlet lifecycle methods in the DD via the filter-mapping element
  - One filter interface per portlet lifecycle
  - Filter chain that gets called by the portlet container
- Available via new javax.portlet.filter package

Code Sample for Portlet Filter

```
// filter declaration

   <filter>
      <filter-name>PortletFilter</filter-name>
      <filter-class>com.acme.PortletFilter</filter-class>
      <lifecycle>RENDER</lifecycle>
   </filter>

// filter mapping

<filter-mapping>
      <filter-name>PortletFilter</filter-name>
      <portlet-name>MyPortlet</portlet-name>
</filter-mapping>
```

Code Sample for Portlet Filter

```
public class PortletFilter implements RenderFilter
    ...
public void init(FilterConfig fc) throws .. {;}
...
public void doFilter(RenderRequest req, RenderResponse resp,
FilterChain chain) throws ..
    PrintWriter pw = resp.getWriter();
    pw.write("Pre-processing");

    RenderResponseWrapper resWrapper =
              new RenderResponseWrapper(res);
    chain.doFilter(req, resWraper);

    pw.write("Post-processing");
...
public void destroy() {;}
```

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

Extended Request Dispatcher Capabilities

Better support servlet-based frameworks on top of portlets

- Request dispatch includes now allowed for all lifecycle methods
  - For action/event no markup can be returned
  - Allows servlet-based bridges to handle controller logic via servlets
- Request dispatcher forward allowed for serveResource calls
  - Delegate complete resource serving to a page created with the JavaServer Page™ technology (JSP™ page)
  - Leveraged by GenericPortlet for forwarding to the specified resource ID if that reflects the path of the resource

Caching

Address additional caching use cases

- Shared cache entries
  - Response can be cached across users
- Validation based caching
  - In addition to the expiry time the portlet can provide a token for the currently returned markup
  - When the content is expired the portal can call the portlet with the provided token of the expired content
  - The portlet can now either re-validate the content and set a new expiry time for the token or create new content with a new token and a new expiry time
  - Based on the HTTP ETag validation caching scheme

Tag lib Additions

Improved useability

- New tag for creating resource URLs
- New variables available via defineObjects
  - portletSession
  - portletPreferences
- New attributes for the URL tags
  - escapeXml for turning of XML escaping, like in JSTL
    - Per default URLs are XML escaped
    - Default can be change in the portlet.xml via a new container-runtime-option element and setting javax.portlet.escapeXml to false
  - copyCurrentRenderParameters for copying the current render parameters

Portlet Managed Modes

Note: This feature is still under discussion in the EG

- Allow portlets to specify their own portlet modes
    - From the portal point of view they are treated like the view mode
    - Portlet can specify a text and description for that mode
    - Portal may include this additional mode in the navigation area
- Portlet can register for portlet mode change events
    - Preset render params or data in prefs/backend systems
- Portlets can return a list of meaningful new portlet modes from action/event
    - Hint for the portal to render the appropriate controls

Misc

Lots of small but important stuff...

- Extended runtime Ids
  - Namespace is now valid for the lifetime of the portlet window
  - Portlet can access the portlet window ID at the request
    - Use this ID if a per portlet window cache key is needed
- PortletURL now accepts a writer
  - Much more efficient than creating Strings
  - Move default of PortletURL.toString to PortletURL.write
- CC/PP support (JSR 188)
  - Available as attribute on the request

Misc

Lots of small but important stuff...

- Restricting the custom window states that a portlet supports for a given markup
- Have a resource bundle allowing text applying to the portlet application level to be localized in a resource bundle
  - In 1.0 it needed to be inline in the portlet.xml
- Lots of small clarifications and clean-ups

Agenda

JSR 286 Overview

Coordination

Resource Serving

AJAX

Cookies and Headers

Filters

Other Additions

Summary

Summary

- Portlet moves from stand-alone component model to a coordinated model enabling composite applications
- Support for AJAX use cases
- Better integration of portlets with servlet-based frameworks
  - Setting HTTP headers/cookies, filters, request dispatching
- Better scalability
  - Different cache levels for resource serving, shared cache entries, validation-based caching, PortletURL.write

Q&A

Stefan Hepper

Wesley Budziwojski

What's New in the Java™ Portlet Specification 2.0 (JSR 286)?

Stefan Hepper

JSR 286 Spec Lead

IBM Corp.

Wesley Budziwojski

JSR 286 EG Member

Sun Microsystems, Inc.

Session TS-4225