Portlets and Ajax: Building More Dynamic Web Apps

Subbu Allamaraju
Senior Staff Engineer
BEA Systems, Inc.

TS-4003

Goal of the Session

Learn how to build dynamic and heterogeneous portlet apps with Java™ Specification Request (JSR) 286 and Ajax.

Agenda

- JSR 168 and Ajax
  - Rights, Wrongs, and Relevance
- Overview of JSR 286
  - New Features
- Ajax and JSR 286
  - How Far Does the Spec Go?
- Framework Integration
  - Dojo as an Example

Agenda

JSR 168 and Ajax

- Rights, Wrongs, and Relevance

Overview of JSR 286

- New Features

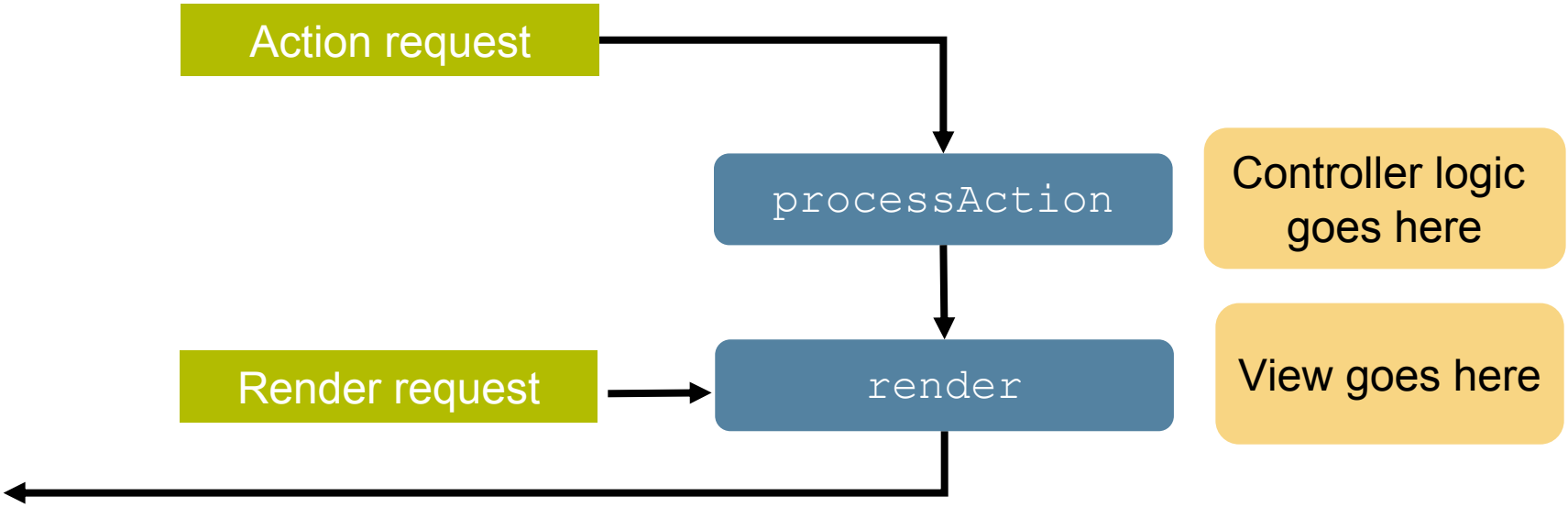
Ajax and JSR 286

- How Far Does the Spec Go?

Framework Integration

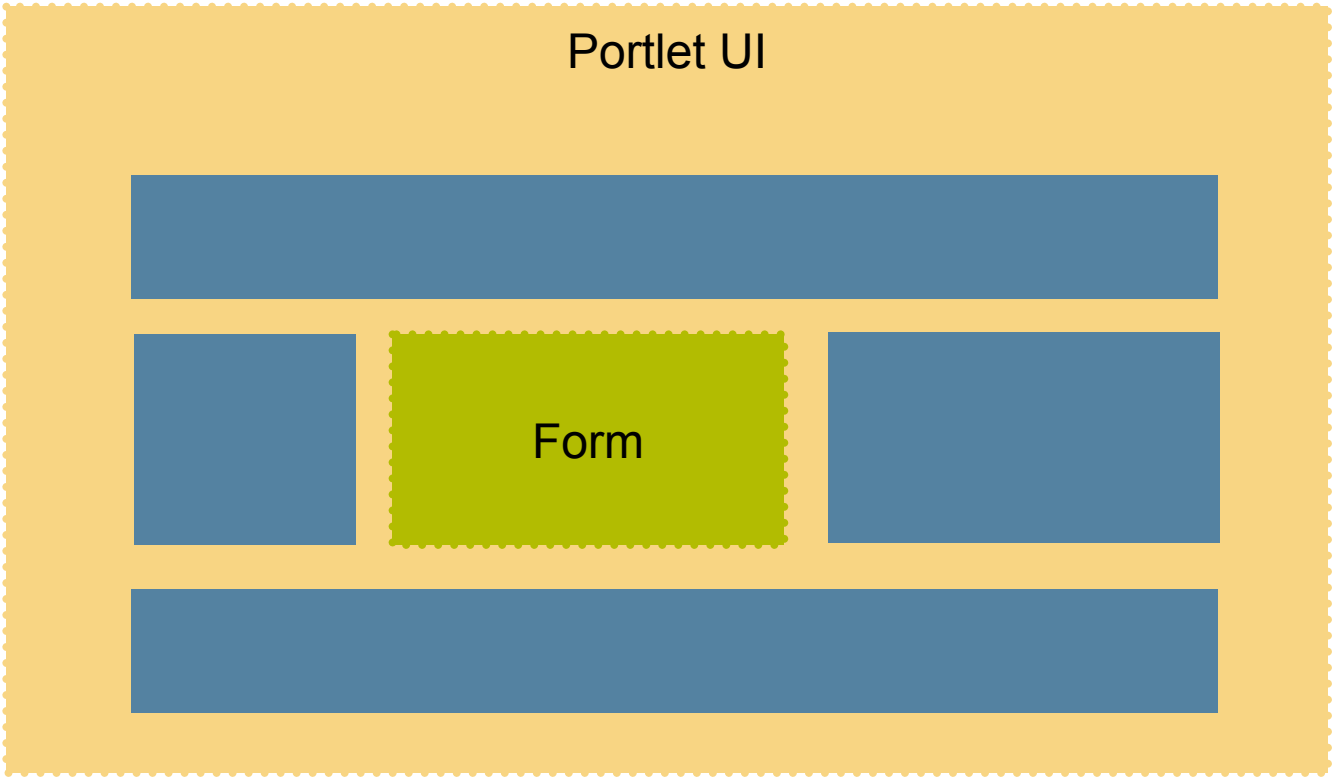
- Dojo as an Example

JSR 168 View of a Web App

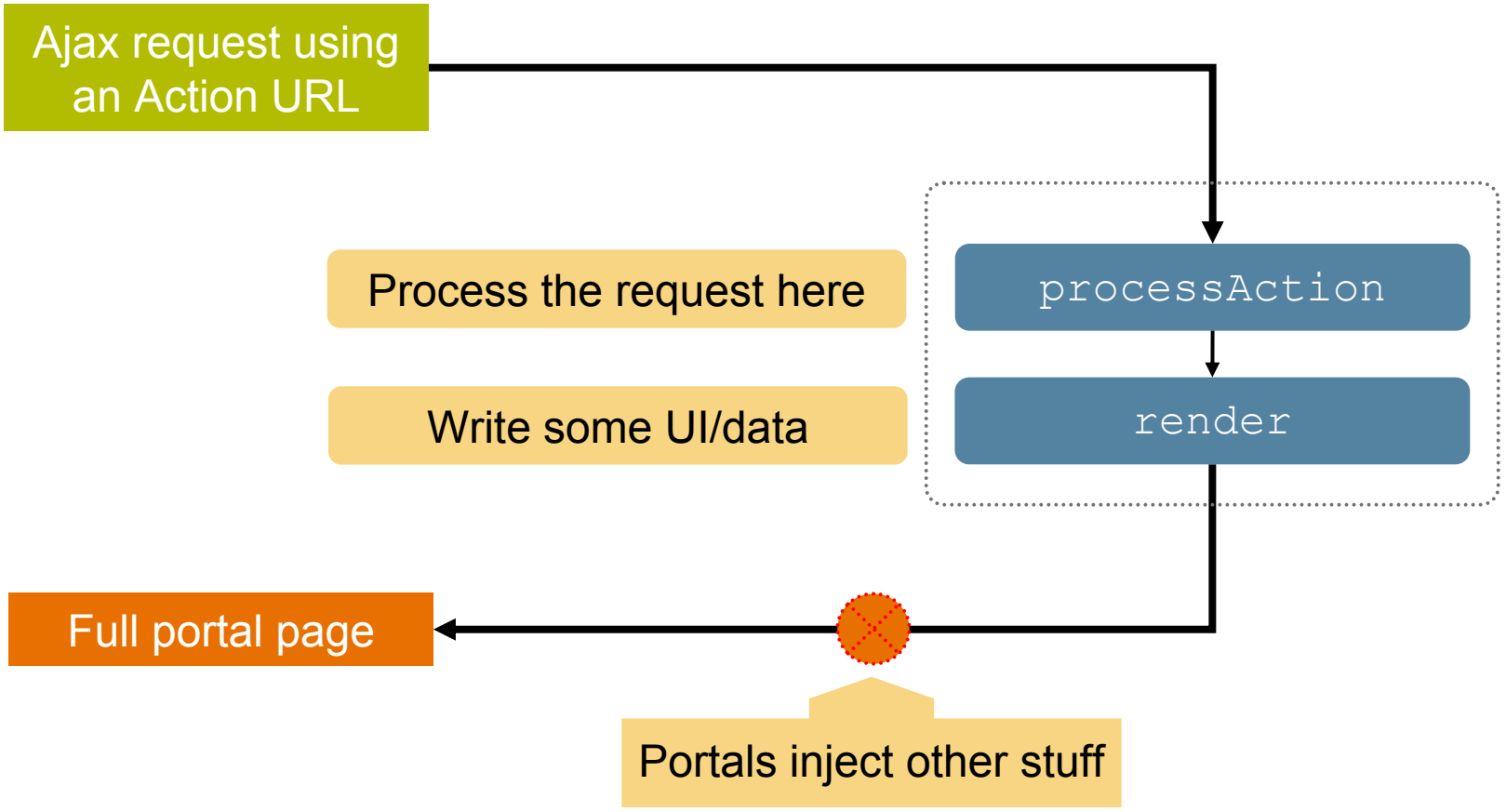


Simple lifecycle with clear separation of concerns

JSR 168 and Ajax: Example



JSR 168 and Ajax: Example

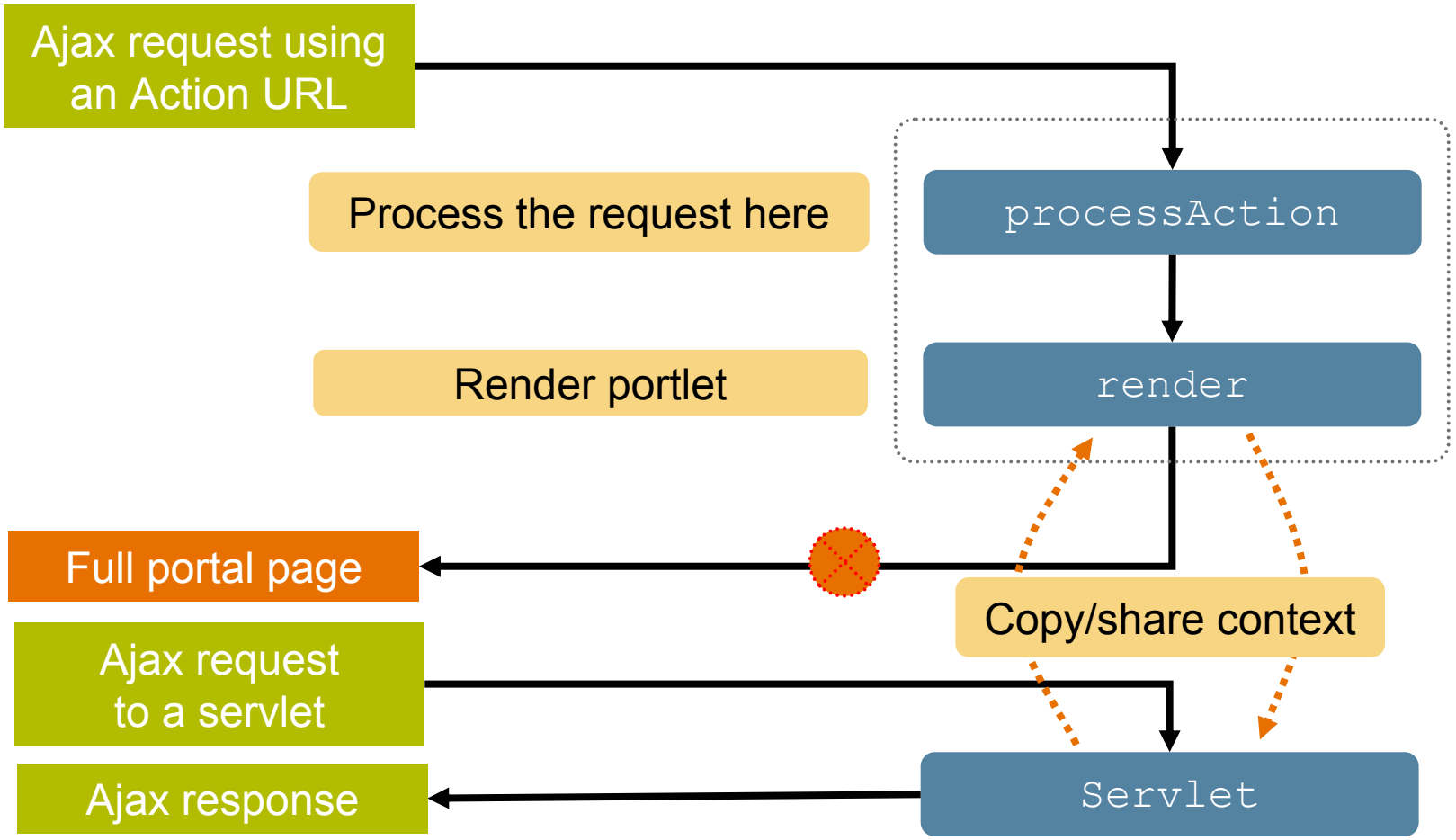


JSR 168 and Ajax

The pitfalls

- Action and render URLs point to the portal
  - Not to the portlet
- Render output is meant for aggregation
  - Portals add more markup to the response
  - i.e., scripts can't process the response
  - Can't scrape the response?
- Contradicts with XMLHttpRequest's view
  - XMLHttpRequest is a client API and wants to consume the response itself

JSR 168 Work-Around



JSR 168 Work-Around

- Things to watch out for
  - Shared context can become stale or invalid
  - Make sure that the servlet end point is reachable from browsers
    - Particularly when portlets are remoted via protocols like WSRP (Web Services for Remote Portlets), or when you are moving portlets around

Agenda

JSR 168 and Ajax

- Rights, Wrongs, and Relevance

Overview of JSR 286

- New Features

Ajax and JSR 286

- How Far Does the Spec Go?

Framework Integration

- Dojo as an Example

JSR 286: Key Changes

A more pragmatic API

- Improved support for bridges
  - Portlet request and response wrappers
  - Portlet filters
- Loosely coupled coordination
  - Server-side events
  - Shared render parameters
- Serving resources
  - Generate arbitrary response

Event-Based Coordination

Step 1: Event source

```
public class EmployeePortlet extends GenericPortlet {
   public void processAction(ActionRequest req,
                             ActionResponse resp) {
      // Process the request
      ...
      EmplBean emplBean = new EmplBean(…);
      // Fire an event
      resp.setEvent("emplCreated", emplBean);
   }
   ...
}
```

Event-Based Coordination

Step 2: Declare interest in events

```
<portlet>
  ...
  <supported-processing-event>emplCreated</supported-
   processing-event>
</portlet>

<event-definition>
  <name>emplCreated</name>
  <java-class>samples.EmplBean</java-class>
</event-definiton>
```

Event-Based Coordination

Step 3: Event target

```
public class NewEmployeePortlet extends GenericPortlet
{
   public void processEvent(EventRequest req,
                            EventResponse resp) {
      Event e = req.getEvent();
      EmplBean emplBean = (EmpBean) e.getValue();
      ...
   }
   public void render(RenderRequesr req,
                      RenderResponse resp) {
      ...
   }
}
```

Shared Render Parameters

Step 1: Set a render parameter

```
public class ZipSelector extends GenericPortlet {
   public void processAction(ActionRequest req,
                             ActionResponse resp)
   {
       resp.setRenderParameter("zip", "12345");
   }
   ...
}

OR

<portlet:renderURL var="url"><param name="zip"
   value="12345"/></portlet:renderURL>
```

Shared Render Parameters

Step 2: Receive a render parameter

```
public class MapsPortlet extends GenericPortlet
{
   public void render(RenderRequest req,
                      RenderResponse resp)
   {
       String zip = req.getParameter("zip);
       ...
   }
   ...
}
```

Resource Serving

```
<portlet:resourceURL var="url">
  <portlet:param name="p1" value="v1"/>
</portlet:resourceURL/>
<img src="<%=url%>" alt="..."/>
```

```
public class ChartPortlet extends GenericPortlet
{
   public void serveResource(ResourceRequest req,
                             ResourceResponse resp) {
      // Write the chart
      resp.setContentType("img/gif");
      resp.getOutputStream().write(...);
   }
   ...
}
```

Resource requests are idempotent

Agenda

JSR 168 and Ajax

- Rights, Wrongs, and Relevance

Overview of JSR 286

- New Features

Ajax and JSR 286

- How Far Does the Spec Go?

Framework Integration

- Dojo as an Example

JSR 286 + Ajax: How Far?

Future or Extensions

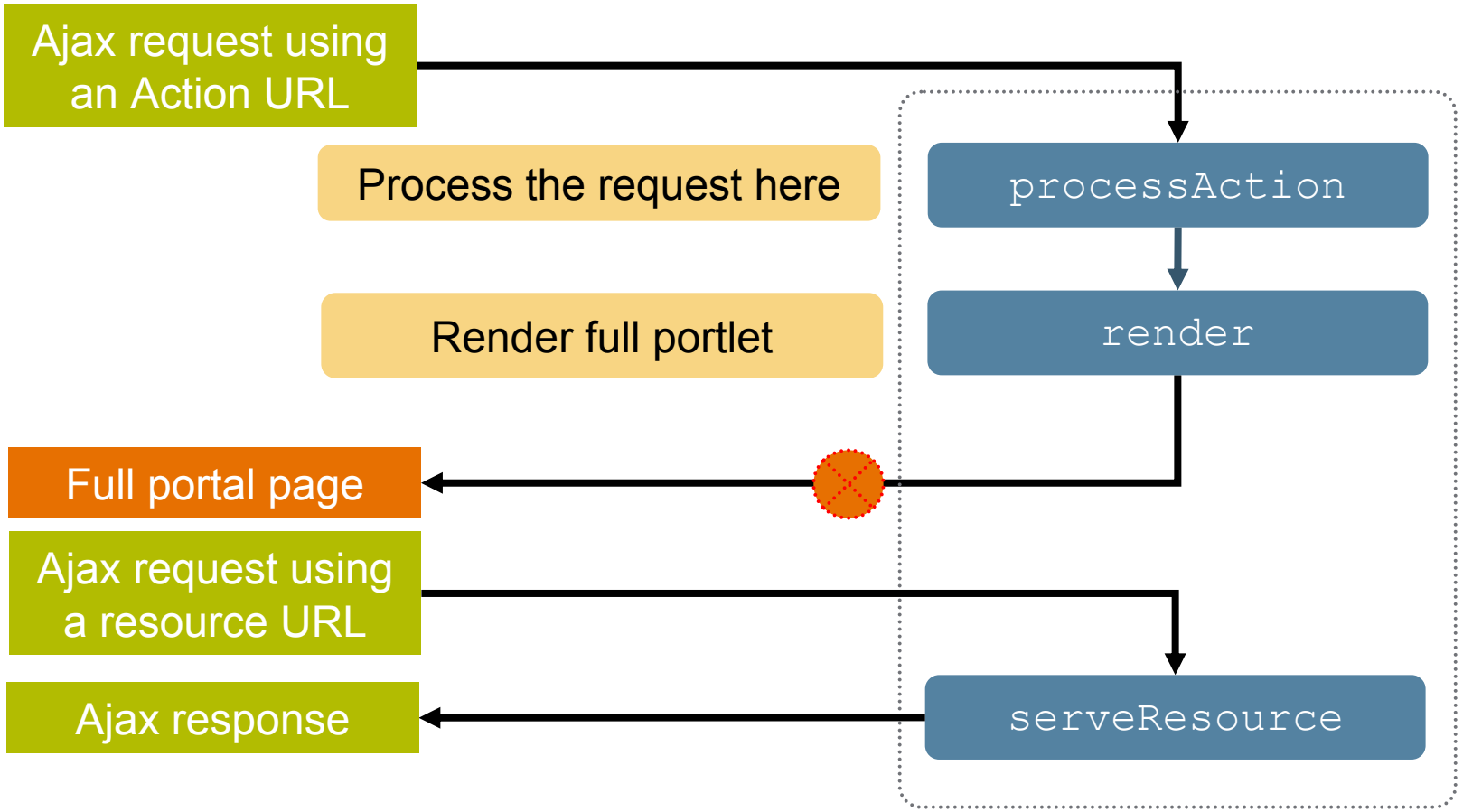
Non-idempotent Ajax Requests
- Events
- Shared render parameter changes
- Container/portal managed state

JSR 286

Idempotent Ajax Requests
- Partial updates to the portlet's UI
- Form submissions or state changes possible
- Cannot change container/portal managed state

serveResource() for Ajax

Look familiar?



Using serveResource() for Ajax

Step 1: Client-side

```
<portlet:resourceURL var="url"><param name="p1"
   value="v1"/>
</portlet:resourceURL>

<script type="text/javascript">
  var req = new XMLHttpRequest();
  req.onreadystatechange = function() {
    // Update UI
  }
  req.open("GET", <%=url%>);
  req.send();
<script>
```

Using serveResource() for Ajax

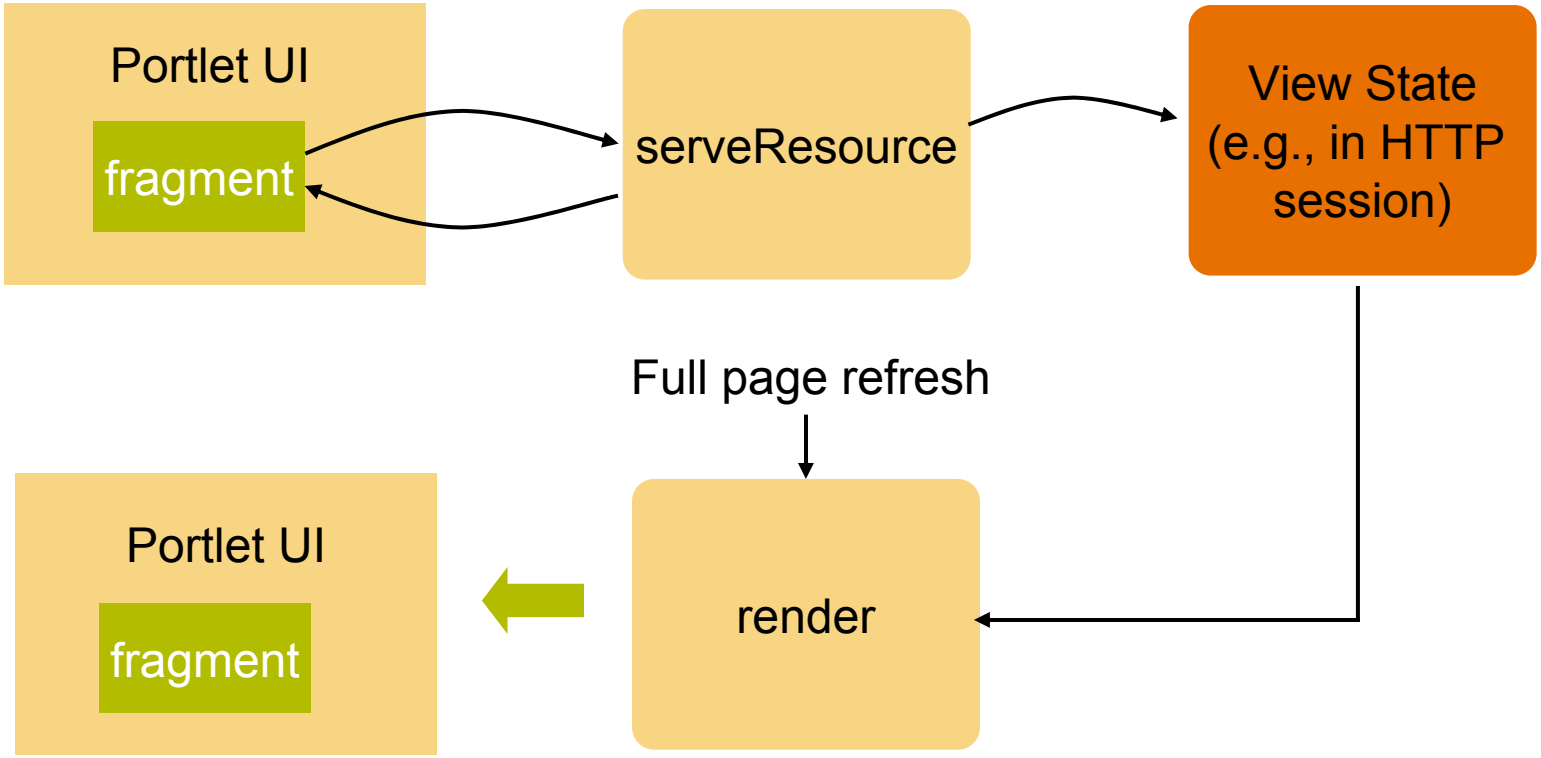
Step 2: Server-side

```
public class MyPortlet extends GenericPortlet
{
   public void serveResource(ResourceRequest req,
                             ResourceResponse resp)
   {
       // Process request and write some JSO data
       resp.setContentType("application/json");
       resp.getWriter().write("...");
   }
}
```

serveResource Programming

- Model
  - For “traditional” requests, continue to use action/render URLs
  - For Ajax requests, use resource URLs
- Advantages
  - Easy to migrate existing JSR 168 applications
  - Works with existing client-side libraries
  - Addresses all idempotent use cases

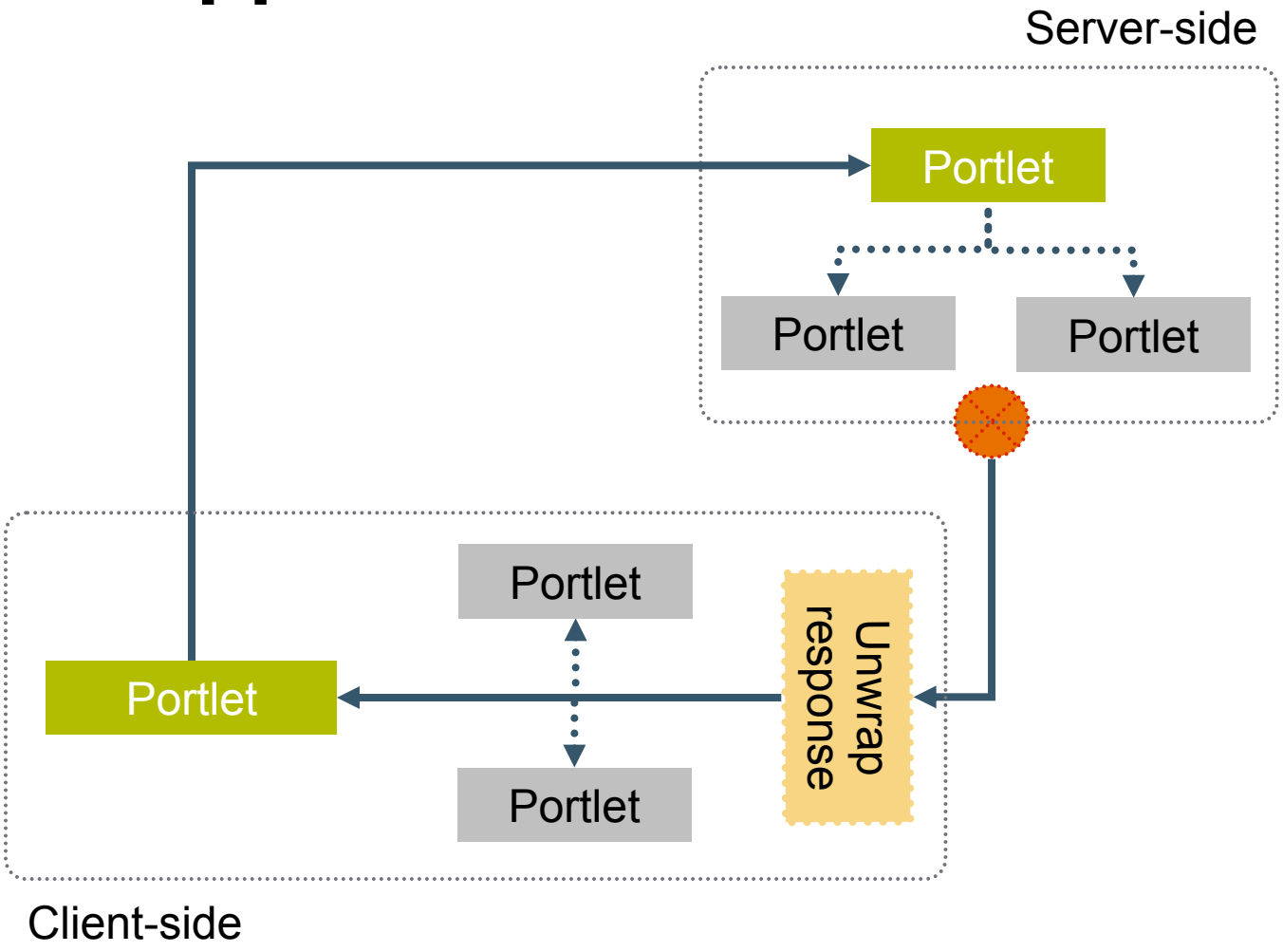
serveResource and State Management



Non-Idempotent Ajax Requests

- Why special?
  - Client requests have ripple effects
    - Portlet's state and view
    - Other portlets' state and view
    - Portal's state and view
  - Portal and the container need to collaborate on the server-side as well as the client-side
- Not addressed, but leaves the door open

An Approach


Server-side
Portlet
Portlet
Portlet
Portlet
Portlet
Portlet
Unwrap response
Client-side

An Approach

1. Use the existing JSR 168/JSR 286 APIs
   - i.e., take full advantage of the container
2. Use a wrapped XMLHttpRequest
   - For all browser-server I/O
   - Portal provides an implementation of the wrapped XMLHttpRequest

XMLPortletRequest

A Wrapper over XMLHttpRequest (*)

```
interface XMLPortletRequest {
    attribute EventListener onreadystatechange;
    readonly attribute unsigned short readyState;
    void open(method, url, async, user, password);
    void setRequestHeader(header, value);
    void send(data);
    void abort();
    DOMString getAllResponseHeaders();
    DOMString getResponseHeader(header);
    readonly attribute DOMString responseText;
    readonly attribute Document responseXML;
    readonly attribute unsigned short status;
    readonly attribute DOMString statusText;
}
```

* http://www.w3.org/TR/XMLHttpRequest

XMLPortletRequest

Example

```
<script type="text/javascript">
  var request = null;
  try {
    request = new XMLPortletRequest();
  } catch(e) {
    request = new XMLHttpRequest();
  }
  request.onreadystatechange = function() {
    if(request.readyState == 4 && request.status == 200) {
      // Process responseXML or responseText
    }
    request.open('GET', <%=url%>, false);
    request.send(null);
  }
</script>
```

XMLPortletRequest

Key properties

- Least invasive
  - **Semantically and syntactically equivalent to the familiar XMLHttpRequest**
    - i.e., no need to learn/write portal/portlet-specific JavaScript™ technology
  - Ajax frameworks can be adopted for use in portals
  - Designed as an XMLHttpRequest-equivalent for portlets
- Ajax requests are portal-managed
  - Portals implement XMLPortletRequest
  - **Portals can update other portlets on the page**
- Ajax requests get full benefits of the portlet API

Server-Side Portlet API

- Use existing JSR 168/JSR 286 APIs
- A well-defined extension
  - Request attributes
    - Standard behavior: unspecified
    - Ajax: Request attributes are preserved between processAction and the first render
  - Render URL parameters
    - Standard behavior: Replace current render parameters
    - Ajax: Do not replace current render parameters
  - JSR 286 allows such extensions to be declared and queried at runtime

DEMO

Events and Ajax

Agenda

JSR 168 and Ajax

- Rights, Wrongs, and Relevance

Overview of JSR 286

- New Features

Ajax and JSR 286

- How Far Does the Spec Go?

Framework Integration

- Dojo as an Example

Ajax Frameworks and Portlets

- Frameworks make Ajax development easy
- Client-side frameworks
  - Dojo, prototype, etc.
- Server-side frameworks
  - JavaServer™ Faces technology-oriented frameworks like ICEFaces, Backbase, etc.
  - GWT, DWR, etc.
- Can these toolkits be used for portlet development?
  - Depends
  - Work in progress

Adopting an Ajax Framework

Example: Dojo

```
dojo.hostenv.getXmlhttpObject = function() {
   try{
      http = new XMLPortletRequest();
   } catch(e) {
      ... // Rest of the code to create XMLHttpRequest
      try { http = new XMLHttpRequest; }catch(e) {}
      ...
  }
  return http;
}
```

Adopting an Ajax Framework

Example: Dojo in a portlet

```
dojo.io.bind({
  url: ”<%=url%>",
  method : "GET",
  mimetype: "application/json",
  load: function(type, data) {
    ...
  },
  error: function(type, error) {
    ...
  }
});
```

Framework Selection for Portlet Development

- If writing portlets natively
  - Rely on serveResouce to the extent possible
    - Portability, by limiting the use cases
  - Use extensions like XMLPortletRequest for more complex use cases
  - Vendors are likely to offer other kinds of extensions
- If writing portlets using Ajax frameworks
  - Portability may not be guaranteed yet
  - More standardization efforts are on the horizon

Summary

- JSR 168 and Ajax
  - Limited support
- JSR 286 and Ajax
  - Provides a basic solution for idempotent Ajax requests
    - Meets most common use cases
  - Leaves the door open for vendor extensions

Q&A

subbu@bea.com

Portlets and Ajax: Building More Dynamic Web Apps

Subbu Allamaraju
Senior Staff Engineer
BEA Systems, Inc.

TS-4003