



JSR-314: JavaServer Faces 2.0

A Red Hat Perspective

Dan Allen

Senior Software Engineer, RedHat

JSR-314 Expert Group Member


October 8, 2009

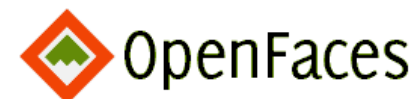
Roadmap

- **Background**
- What's New?
- Red Hat's Contributions
- Community



The JSF Story

- Introduced in 2004 as Java EE standard
- Developing using the Java Community Process (JCP)
 - JSR-127 – JSF 1.0 and 1.1
 - JSR-252 – JSR 1.2, Java EE 5 technology
 - JSR-314 – JSF 2.0, Java EE 6 technology
- Two open source implementations
 - Mojarra – JSF RI
 - Apache MyFaces 
- Strong ecosystem of components



Goals

- **JavaServer Faces** is a standard user interface (UI) framework for developing Java EE web applications
 - APIs for creating user interface (UI) components
 - A default set of UI components
 - Custom tag libraries for adding UI components to a view
 - A server-side event model
 - Managed beans (state management)
 - Unified EL integration



JSR-314 (JSF 2.0) Mission

- Many great ideas
- Ideas extend from the specification
- No standard specification to ensure compatibility
- Harvest ideas and align them with Java EE platform



Roadmap

- Background
- **What's New?**
- Red Hat's Contributions
- Community



What's New in JSF 2?

- View Declaration Language and Facelets 2*
- Composite Components
- System Events
- Enhanced Navigation**
- GET Support**
- Bean Validation**
- Proper Error Handling*
- Project Stages
- First-class JavaScript and Ajax Support*
- Component Behaviors*
- Resource Handling
- Delta State Saving
- Tree Visiting
- Annotation-based Configuration
- Additional Scopes

<http://andyschwartz.wordpress.com/2009/07/31/whats-new-in-jsf-2>



What's New in JSF 2?

- View Declaration Language and Facelets 2*
- Composite Components
- System Events
- Enhanced Navigation**
- GET Support**
- Bean Validation**
- Proper Error Handling*
- Project Stages
- First-class JavaScript and Ajax Support*
- Component Behaviors*
- Resource Handling
- Delta State Saving
- Tree Visiting
- Annotation-based JSR-299**
- Additional Scopes

<http://andyschwartz.wordpress.com/2009/07/31/whats-new-in-jsf-2>



Your kind of component

```
/resources/property/display.xhtml
```

```
<ui:composition>
  <comp:interface>
    <comp:attribute name="label" required="true"/>
    <comp:attribute name="value" required="false"/>
  </comp:interface>

  <comp:implementation>
    <div class="field" id="#{cc.clientId}>
      <span class="label">#{cc.attrs.label}</span>
      <span class="value">#{cc.attrs.value}</span>
    </div>
  </comp:implementation>
</ui:composition>
```



Your component in use

/userDetail.xhtml

Name of resources directory

```
<f:view ... xmlns:p="http://java.sun.com/jsf/composite/components/property"
```

... Base name of component template

```
<p:display id="username" label="Username" value="#{user.username}"/>
```

```
<p:display id="email" label="E-mail" value="#{user.email}"/>
```

```
<p:display id="name" label="Real name" value="#{user.name}"/>
```

...

```
</f:view>
```



Roadmap

- Background
- What's New?
- **Red Hat's Contributions**
- Community



Why contribute?

JSF 1.2



GET Support and Bookmarkability

- Bookmarkable links and buttons (UIOutcomeTarget)
 - `<h:link>`
 - `<h:button>`
- Preemptive navigation
 - Consult navigation rules at render time
- View-oriented functionality
 - View metadata
 - View parameters
 - View processing events



View Parameters

- Evolution of Seam's page parameters
- Bind request parameters to model values
 - Conversion
 - Validation
- Propagate model values through bookmarkable URL



State in the URL (1)

- View parameters mapped in metadata facet

```
<f:view>
  <f:metadata>
    <f:viewParam name="id" value="#{blog.entryId}"/>
    <f:event type="preRenderView" listener="#{blog.load}"/>
  </f:metadata>
</f:view>
```

- View parameters appended to bookmarkable links

```
<h:link value="Link to entry"/>
```



Outcome assumed to be current view ID



State in the URL (2)

- View parameters appended to redirect URLs

```
<navigation-rule>
  <from-view-id>/entry.xhtml</from-view-id>
  <navigation-case>
    <from-action>#{blog.postComment}</from-action>
    <to-view-id>/entry.xhtml</to-view-id>
    <redirect include-view-params="true"/>
  </navigation-case>
</navigation-rule>
```

- Or defined in navigation rule

```
<navigation-case>
  <from-outcome>entry</from-outcome>
  <to-view-id>/entry.xhtml</to-view-id>
  <redirect>
    <view-param>
      <name>id</name><value>#{blog.entryId}</value>
    </view-param>
  </redirect>
</navigation-case>
```

```
<h:link outcome="entry"
  value="Link to entry"/>
```

←



Implicit Navigation

- Return view ID from action method

```
public String book() {  
    return "/confirmation.xhtml";  
}
```



Implicit Navigation

- Return root of view ID from action method

```
public String book() {  
    return "confirmation";  
}
```



Implicit Navigation

- Redirect after action method (POST)

```
public String book() {  
    return "confirmation?faces-redirect=true";  
}
```



Implicit Navigation

- Redirect with view parameters

```
public String book() {  
    return "confirmation?faces-redirect=true&includeViewParams=true";  
}
```



Implicit Navigation in the View

- Declare target view ID in UICommand tag

```
<h:commandButton action="/createEvent.xhtml" value="Create"/>
```

- Simple redirect after POST

```
<h:commandButton action="createEvent?faces-redirect=true"
  value="Create"/>
```

- Declare target view ID in UIOutcomeTarget

```
<h:link outcome="/agenda.xhtml" value="Agenda"/>
```

Current view ID used if outcome is absent



Conditional Navigation

- Navigation logic based on state

```
<navigation-case>  
  <from-action>#{bookingAgent.checkDates}</from-action>  
  <if>#{hotel.roomsAvailable}</if>  
  <to-view-id>/confirmBooking.xhtml</to-view-id>  
</navigation-case>
```



Partial Page Rendering

- Standardization of the Ajax4jsf `<a4j:support>` tag
 - `<f:ajax>`
- Classified as a component behavior
- Formal JavaScript API
 - `jsf.ajax.request()`
- More control over server-side processing
 - `PartialViewContext`
 - `<f:ajax ... render="@form">`



Partial Page Rendering Example

```
<script type="text/javascript">
function controlSpinner(status) {
    if (status.name == 'begin') {
        document.getElementById('spinner').style.display = 'inline';
    } else if (status.name == 'complete') {
        document.getElementById('spinner').style.display = 'none';
    }
}
</script>
```

```
...
<h:form id="search">
    <h:inputText id="query" value="#{searchCriteria.query}">
        <f:ajax event="keyup" listener="#{hotelSearch.find}"
            render="hotels" onevent="controlSpinner"/>
    </h:inputText>
</h:form>
```

```
...
<h:dataTable id="hotels" value="#{hotels}" var="_hotel">
    ...
</h:dataTable>
```



Partial Page Rendering Example

```
<script type="text/javascript">
function controlSpinner(status) {
  if (status.name == 'begin') {
    document.getElementById('spinner').style.display = 'inline';
  } else if (status.name == 'complete') {
    document.getElementById('spinner').style.display = 'none';
  }
}
</script>
```

```
...
<h:form id="search">
  <h:inputText id="query" value="#{searchCriteria.query}">
    <f:ajax event="keyup" listener="#{hotelSearch.find}"
      render="hotels" onevent="controlSpinner"/>
  </h:inputText>
</h:form>
```

```
...
<h:dataTable id="hotels" value="#{hotels}" var="_hotel">
  ...
</h:dataTable>
```



Partial Page Rendering Example

```
<script type="text/javascript">
function controlSpinner(status) {
  if (status.name == 'begin') {
    document.getElementById('spinner').style.display = 'inline';
  } else if (status.name == 'complete') {
    document.getElementById('spinner').style.display = 'none';
  }
}
</script>
```

```
...
<h:form id="search">
  <h:inputText id="query" value="#{searchCriteria.query}">
    <f:ajax event="keyup" listener="#{hotelSearch.find}"
      render="hotels" onevent="controlSpinner"/>
  </h:inputText>
</h:form>
```



```
...
<h:dataTable id="hotels" value="#{hotels}" var="_hotel">
  ...
</h:dataTable>
```



Bean Validation

- JSR-303: Bean Validation
 - **Red Hat** led spec
- Centrally define validation constraints on model
- JSF integration: Extend validation to UI



Defining a Set of Constraints

```
public class Booking {  
    ...  
    @NotNull(message = "Credit card number is required")  
    @Size(min = 16, max = 16,  
        message = "Credit card number must 16 digits long")  
    @Pattern(regexp = "^\\d*$",  
        message = "Credit card number must be numeric")  
    public String getCreditCardNumber() {  
        return creditCardNumber;  
    }  
}
```



Enforcing the Constraints in the UI

- Put Bean Validation implementation on classpath

```
<h:inputText id="creditCard" value="#{booking.creditCardNumber}"/>
```

- Disable for single input

Validation tag not required!

```
<h:inputText id="occupants" value="#{booking.numOccupants}">
  <f:validateBean disabled="true"/>
</h:inputText>
```

- Disable per branch of component tree

```
<f:validateBean disabled="true">
  <h:inputText id="occupants" value="#{booking.numOccupants}"/>
</f:validateBean>
```

- Filter constraints by validation group

```
<f:validateBean validationGroups="org.acme.ReservedBlock">
  <h:inputText id="groupCode" value="#{booking.groupCode}"/>
</f:validateBean>
```



Bean Validation Integration Summary

- JSF automatically enforces field-level constraints
- I18N is inherited
 - Handled by Bean Validation
 - JSF passes constraint message as argument when resolving JSF validator message

```
javax.faces.validator.BeanValidator.MESSAGE={0}
```



Simple Select Items

- Create select items from any collection
- SelectItem should be an implementation detail

```
public class CalendarBean {  
    public List<Month> getMonths() {  
        List<Month> months = new ArrayList<Month>();  
        months.add(new Month(0, "January"));  
        ...  
        return months;  
    }  
}
```

```
<h:selectOneMenu "#{booking.creditCardExpiryMonth}">  
    <f:selectItems value="#{calendarBean.months}"  
        var="_month" itemValue="#{_month.index}"  
        itemLabel="#{_month.name}"  
</h:selectOneMenu>
```



Proper Exception Handling

- Handles *unexpected* exceptions
- No exceptions are swallowed!
- Ties into system events: `ExceptionHandler`
- Otherwise handled in `web.xml`

```
<error-page>
  <exception-type>
    org.acme.AuthorizationException
  </exception-type>
  <location>/denied.jsf</location>
</error-page>
```

- Facelets debug page when project stage = development
 - Custom debug page: `javax.faces.error.xhtml`

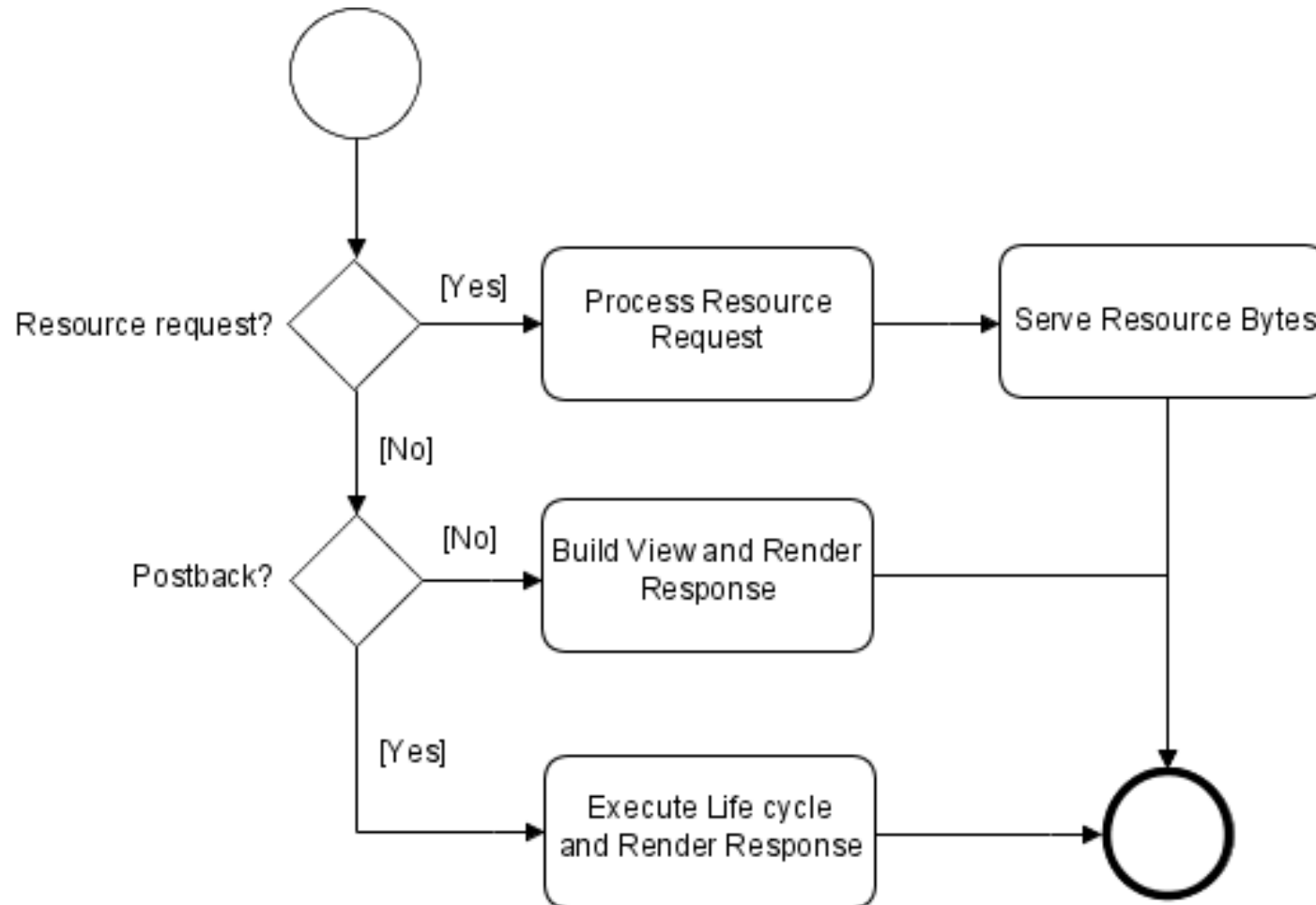


Native Resource Handler

- Serve JavaScript, images, CSS, etc.
 - No bonus servlet required!
- Declarative and dynamic
- Resource structure
 - Name
 - Library
 - Locale
 - Version
- Packaging options
 - /resources directory of web root
 - /META-INF/resources of classpath entry (JAR)

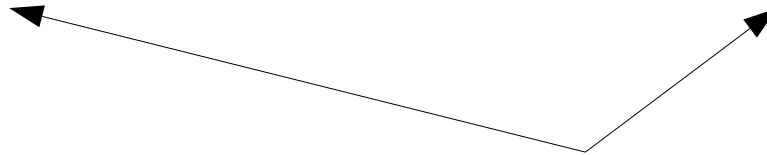


Request processing scenarios



Resolving a Resource

localePrefix/libraryName/libraryVersion/resourceName/resourceVersion



Path segments in gray are optional

- Highest version selected automatically
- Serve from /resources in web root

```
<h:graphicImage name="visa.png"/>
```

- Serve from /META-INF/resources of creditcards.jar

```
<h:graphicImage name="visa.png" library="creditcards"/>
```

```
<h:graphicImage value="#{resources['creditcards:visa.png']}" />
```



Resource Relocation

- Resources can target section of document
- Convenient for templating

```
<h:head>  
  <title>Resource Relocation Example</title>  
</h:head>  
<h:body>  
  <h:outputScript name="script.js" target="head"/>  
</h:body>
```



The book isn't closed

- **Red Hat** is still pushing for change in JSF 2.1
 - <http://seamframework.org/Documentation/JSF21>
- JSR-299 conversations need to be better recognized
- Bean Validation integration could go deeper
- JSF could still be simpler and easier
- Still many great ideas out there to be explored



Roadmap

- Background
- What's New?
- Red Hat's Contributions
- **Community**



Red Hat's Vision: An Open Specification

- Support from Oracle (Andy Schwartz) and Sun
- JSR-314 mailinglist open for reading
 - <http://archives.java.sun.com/jsr-314-open.html>
 - Login with free account required
 - *Only* EG members can post
- JCP.org public forums
 - <http://wiki.jcp.org/boards/index.php?b=958>
 - Relationship to mailinglist unclear
- JCP.org public wiki
 - http://wiki.jcp.org/wiki/index.php?structure_id=6
 - Minimal activity so far



Share your vision



Summary

- JSR-314 is a big step forward for JSF
- Key pain-points resolved
 - JSP dumped for Facelets
 - Simpler component development
 - GET support (bookmarkable URLs)
 - Ajax and JavaScript APIs
 - Resource delivery
- Good integration with other specs
 - JSR-299: Contexts and Dependency Injection
 - JSR-303: Bean Validation
- **Openness**

