Mastering JavaServer Faces

Bryan Basham
Software Alchemist
basham47@gmail.com
http://www.linkedin.com/in/SoftwareAlchemist
Web UI Architecture Concerns
Client-side Technologies

- Structure
  - XHTML (or HTML v4.01)

- Presentation
  - Cascading Style Sheets (CSS)

- Behavior
  - JavaScript (formally: ECMAScript 262 v3)
  - DOM API specification
  - JS objects (and JSON) for data models
  - Open source frameworks
Server-side Technologies

- **Structure**
  - JSF for components (UI widgets)
  - JSP (or other) for layout of component tree

- **Behavior**
  - JSF component event model
  - OO programming for data model
  - Open source implementations, frameworks, and component libraries
Separation of Concerns

- **Client**
  - Presentation
  - Structure
  - Behavior
  - Data

- **Server**
  - Structure
  - Behavior
  - Data

- **UI Code**
  - Conversion
  - Generate
  - Select next view
  - Invoke
  - Enhance
  - Style / layout
Why JavaServer Faces?

- Third generation Java-Web technology.
- Closer to traditional UI development:
  - component-based
  - event-driven
- Standards-based architecture
- Open source community
The JSF Stack

- Servlet / JavaServer Pages
- JavaServer Faces
- Component Frameworks
- My Webapp

- EL
- tags
- request/response
- Ajax and other control
- ...and other facilities
- validation
- conversion
- components
- actions
- new UI widgets
JSF and Web Standards

- JSF supports other Web standards
  - XHTML, for structure
  - CSS, for presentation
  - JavaScript, for client-side behavior
- Use JSF libraries that facilitate client-side Web standards
- The gotcha... JSF embeds JavaScript into the page
Topics Mind Map

- Introduction to JSF
- Roll Your Own
- JSF Fundamentals
- MyFaces & Tomahawk
- Other tools
- RichFaces & AJAX4JSF

Mastering JSF
JSF Life Cycle

Web Container

JSF

ViewRoot

Form

InputText

CmdLink

request a view

creates comp. tree

extracts data

binds data

perform action(s)

POST form data

select next view

creates comp. tree
JSF Life Cycle

Lifecycle:
- RESTORE VIEW
- APPLY REQ VALUES
- PROCESS VALIDATIONS
- UPDATE MODEL
- PROCESS APPLICATION
- RENDER RESPONSE

Faces Servlet:
- execute
- render

Restoration:
- restoreState

Processing:
- processDecodes
- processValidations
- processUpdates
- processApplication

Rendering:
- render

UI Component Tag:
- dispatch
- doStart
- doEnd
- encodeBegin
- encodeChildren
- encodeEnd
- saveState

New Component Tree:
- existing component tree

Form:
- ViewRoot
- InputText
- CmdLink

existing component tree
Data Conversion

- All HTTP request data are strings.
- Rich apps require rich data
  - simple non-string data: numbers, Boolean values, IP addresses, money, credit card #
  - complex data: structures, objects, entities with relationships

- The JSF Converter interface:
  - The getAsObject method
  - The getAsString method
Data Conversion

The HTML form is rendered with the current value of the ipAddress="172.20.30.42"

The user modifies the value and submits the form: ipAddress="172.20.30.47"

JSF

IPv4Address Converter

getAsString()

getAsObject()

172.20.30.42

current value

new value

172.20.30.47
Field Validation

- Allows field validation checks after data conversion

- Option #1: call a method on a backing bean

  ```xml
  <c:textField id="ipAddress" value="#{editIntf.interface.ipAddress}" required="true"
              converterId="converter.Ipv4Address" validator="#{editIntf.validateIpAddress}" />
  ```

- Option #2: invoke a validator object

  ```xml
  <c:textField id="number" value="#{bean.myNumber}" required="true">
    <f:validateLongRange minimum="10" maximum="100" />
  </c:textField>
  ```
Field Validation

JSF

Ipv4Address Converter

gets stored in the UI component for further processing by the life cycle

validateIpAddress(...)

UIInput

validate(...)

new value

172.20.30.47

a validation error sends the form back to the user with an error message.

Backing Bean
Application Actions

- Action components
  - UICommand (abstract), HtmlCommandLink, and HtmlCommandButton

- Example:
  ```html
  <h:commandButton value="Submit" actionListener="#{myBean.doThis}" />
  ```

- Invokes the `doThis` method of your backing bean when the user clicks the button
Application Actions

JavaScript

- click
  - Button

Java Server Faces

- broadcast(...)
- invoke: doThis()

Back Bean
Topics Mind Map

Mastering JSF

- Introduction to JSF
- MyFaces & Tomahawk
- JSF Fundamentals
- Roll Your Own
- Other tools
- RichFaces & AJAX4JSF
MyFaces

- is an implementation of the JSF spec.
- is an open source tool from Apache
- provides the standard set of UI widgets
- also includes several additional component libraries:
  - Tomahawk
  - Trinidad
  - Tobago
  - Orchestra
Standard Widget Set

- **Output:**  `f:view`, `h:outputText`, `h:graphicImage`,
- **Table:**  `h:dataTable`, `h:column`
- **Forms:**
  - `h:form`, `h:outputLabel`, `h:inputText/Secret/Textarea/…`
  - `h:selectBooleanCheckbox`
  - `h:selectOneListbox/Menu/Radio`
  - `h:selectManyListbox/Menu/Checkbox`
- **Layout:**  `h:panelGrid`, `h:panelGroup`
Select One Menu: Screen Shot

**Select an Exam**

- Welcome, demo
  To edit exam, first select the type and then select an exam from the list provided.

**Exam category:**

- Select an exam category...

**Exam title:**

- First select a category above.

---

Copyright 2008 Bryan Basham

---

Done
SelectOneMenu: JSP

```html
<div id="exam_select">
    <div>
        <h:outputLabel for="exam_type_select">Exam category:</h:outputLabel>
        <h:selectOneMenu value="#{examEditor.categoryOID}" id="exam_type_select">
            <f:selectItems value="#{examEditor.categorySelectItems}" />
        </h:selectOneMenu>
    </div>
    <div>
        <h:outputLabel for="exam_name_select">Exam title:</h:outputLabel>
        <h:selectOneMenu value="#{examEditor.examOID}" id="exam_name_select">
            <f:selectItems value="#{examEditor.examSelectItems}" />
        </h:selectOneMenu>
    </div>
    <div class="button">
        <h:commandLink action="#{examEditor.viewExam}"
            view exam" /></h:commandLink>
    </div>
    <div class="button">
        <h:commandLink action="#{examEditor.editExam}" immediate="false">
            edit exam" /></h:commandLink>
    </div>
</div>
```
public Long getCategoryOID()
{
    return categoryOID;
}
public void setCategoryOID(Long oid)
{
    this.categoryOID = oid;
}
public List<SelectItem> getCategorySelectItems()
{
    List<Category> categories = this.catSvc.getAllCategories(this.em);
    List<SelectItem> result;
    result = new ArrayList<SelectItem>(categories.size() + 1);
    result.add(new SelectItem("0", "Select an exam category..."));
    for ( Category cat : categories )
    {
        SelectItem item = new SelectItem(cat.getId(), cat.getTitle());
        result.add(item);
    }
    return result;
}
Tomahawk Library

- Compatible with MyFaces and Mojarra (RI)
- Includes the `forceId` attribute
- Provides a Tiles-like facility
- Provides additional validators: email, credit card, regexp matching
- Additional UI widgets
Tomahawk Components

- List: t:dataList
- Table: t:dataTable, t:dataScroller, t:newspaperTable, t:schedule
- Input: t:inputDate, t:inputFileUpload, t:inputHtml, ...
- Layout: t:panelLayout, t:panelTabbedPane, t:panelStack, t:collapsiblePanel
- Document: t:document/Body/Head, t:stylesheet
Trinidad Library

- Partial page refresh (PPR)
- Client-side converters and validators
- Advanced client-side dialog boxes
- Advanced navigation tools/components
- File upload
- Additional UI widgets
Trinidad Components

- Table:
  - similar to UIData
  - supports paging, sorting, multiple item selection

- Tree:
  - simple tree and tree-table

- Charts:
  - many types: pie, bar, scatter plot, and so on
  - generates SVG graphics
Tobago Library

- Styling by use of “themes”
- Built-in I18N / L10N support
- Controls:
  - table
  - tree
  - sheet
  - tab panel
RichFaces Library

- Built-in support for Ajax
- Skinnable styles
- Rich component development kit
- Resource packaging
- Dynamic resources (e.g. building button icons on the fly)
RichFaces Components

- **Layout:** a4j:page, a4j:loadStyle, a4j:loadScript, a4j:include
- **UI widgets:** rich:calendar, rich:comboBox,
- **Tables:** rich:dataTable, rich:datascroller, rich:dataFilterSlider, sorting, filtering
- **Lists:** ordered, unordered, definitions
- **Other:** drag-n-drop support, effects, panels, trees, and a whole lot more
Ajax Support

- **Forms:** form submittal via Ajax; rerender portion of screen

- **Triggers:**
  - Periodic: a4j:poll, a4j:repeat
  - Events: a4j:support
  - JSF: rich:ajaxValidator

- **Most components have built-in Ajax support**
<div id="exam_select">
  <div>
    <h:outputLabel for="exam_type_select">Exam category:</h:outputLabel>
    <h:selectOneMenu value="#{examEditor.categoryOID}" id="exam_type_select">
      <f:selectItems value="#{examEditor.categorySelectItems}" />
      <a4j:support event="onchange" actionListener="#{examEditor.findExams}" reRender="exam_name_select" />
    </h:selectOneMenu>
  </div>
  <div>
    <h:outputLabel for="exam_name_select">Exam title:</h:outputLabel>
    <h:selectOneMenu value="#{examEditor.examOID}" id="exam_name_select">
      <f:selectItems value="#{examEditor.examSelectItems}" />
    </h:selectOneMenu>
  </div>
  ...
</div>
OnChange: Backing Bean

```java
public void findExams() {
    this.category = catSvc.getCategoryById(this.categoryOID, this.em);
    if (this.category != null) {
        this.exams = this.category.getExams();
    }
}

public List<SelectItem> getExamSelectItems() {
    List<SelectItem> result;
    if (this.exams == null || this.exams.size() == 0) {
        result = new ArrayList<SelectItem>(1);
        result.add(new SelectItem("0", "First select a category above."));
    } else {
        result = new ArrayList<SelectItem>(this.exams.size() + 1);
        result.add(new SelectItem("0", "Select an exam..."));
        for (CertificatonExam e : this.exams) {
            SelectItem item = new SelectItem(e.getId(), e.getTitle());
            result.add(item);
        }
    }
    return result;
}
```
Topics Mind Map

- **Introduction to JSF**
- **Roll Your Own**
- **JSF Fundamentals**
- **Other tools**
- **MyFaces & Tomahawk**
- **RichFaces & AJAX4JSF**
- **Mastering JSF**
Other component libraries

- JSF has broad support and rich community
  - See JSFCentral.org
  - See JSF Forum on java.net
- There are many open source and COTS libraries
Facelets

- Built as a replacement for JSP rendering, due to limitations in JSF v1.1
  - eliminates the need for f:verbatim blocks
  - not really needed in JSF v1.2
- Support for tiling, ie. “view composition”
- Support for template-based component creation
Seam

- Integration between UI and Business tiers: JSF <=> EJB3
- Use of annotations simplifies complex deployment descriptors
- Works well with Ajax and Ajax-enabled JSF libraries such as RichFaces
- Conversational scope
Topics Mind Map

- Introduction to JSF
- JSF Fundamentals
- MyFaces & Tomahawk
- RichFaces & AJAX4JSF
- Roll Your Own
- Other tools
Building Components

- Several open source libraries provide rich APIs for building components
  - MyFaces
  - RichFaces
- Consider the need to inject component-specific resources
  - styles
  - JavaScript
  - media
Q & A

Mastering JSF

- Introduction to JSF
- JSF Fundamentals
- MyFaces & Tomahawk
- RichFaces & AJAX4JSF
- Other tools
- Roll Your Own
Resources

- JavaServer Faces in Action (Manning)
- JavaServer Faces:
  - java.sun.com
  - java.net
  - JSF Spec (JSR #252)
  - JSFCentral.org
  - JSF tutorials
- Component Libraries:
  - MyFaces (for JSF v1.2)
  - RichFaces
- Other tools:
  - Facelets
  - Seam Framework