JSF – JavaServer Faces

By Sosuke Tokunaga
CS491b Fall 2004

Introduction

- New Java Technology for Java Server Application
- Bundled with J2EE 1.4 SDK

First Look

- JSF = Struts + Swing
  - Java server application framework based on MVC architecture
  - UI component model for Java server application

Develop server applications in a similar manner with desktop applications

Basic Components

- Faces Servlet
- Faces Context
- UI Components
- Events and Listeners
- Faces Core/HTML Tag Libraries

HTML UI Components

- HtmCommandButton
- HtmCommandLink
- HtmForm
- HtmInputText
- HtmInputSecret
- HtmInputHidden
- HtmInputTextArea
- HtmSelectBooleanCheckbox
- HtmSelectManyCheckbox
- HtmSelectOneMenu
- HtmSelectManyMenu
- HtmSelectOneList
- HtmSelectManyList
- HtmSelectOneRadio

- HtmOutputLabel
- HtmOutputText
- HtmOutputLink
- HtmOutputFormat
- HtmDataTable
- HtmGraphicImage
- HtmMessage
- HtmMessages
- HtmPanelGrid
- HtmPanelGroup
- UVViewRoot
- UIColumn

Faces Request Life Cycle
Phase 1: Restore View
- **View**
  - A UI component tree which represents structured UI components associated with a page
- **Initial Request**
  - Create a empty view and skip to the Render Response Phase
- **Postback**
  - Restore the view by using the state information

Phase 1: Restore View – example(1)
- JSP page with JSF tags
  ```jsp
  <html>
  <head><title>Add 2 Numbers</title></head>
  <body>
    <form method="post" action="/calculator">
      <input type="text" name="num1" />
      <input type="text" name="num2" />
      <input type="submit" value="Add" />
    </form>
  </body>
  </html>
  ```

Phase 1: Restore View – example(2)
- **View – Component Tree**

Phase 2: Apply Request Values
- **For Each Component** in the View
  - Retrieve parameters associated with the component
  - Convert the parameters into appropriate type
  - Set the converted value into the component as submitted values
- **Redisplay**
  - If an event associated with this phase, the handler can skip to the Render Response Phase

Phase 3: Process Validation
- **Each Component in the View**
  - Validates submitted values
  - Sets the submitted values to local values
  - If values have been changed, ValueChangeEvent will be broadcast.
  - Clears the submitted values
- **If Errors Occurred**
  - Stores error messages into the Context and skips to the Render Response Phase

Phase 4: Update Model Values
- **Each Component in the View**
  - Sets the local values into the referenced properties of Backing Beans
  - Clears the local values
- **If Errors Occurred**
  - Stores error messages into the Context and skips to the Render Response Phase
Phase 5: Invoke Application

- Broadcast Application-level Events
- Submitting
- Linking to another page
- Tasks of Event Handlers
- Invoke services provided by Business Logic
- Determine the forwarding page displayed next by using NavigationHandler
- Faces Servlet sets the response view of the forwarding page.

Phase 6: Render Response

- If Initial Request
- Add Components on the page into the View
- Rendering
  - Components render themselves onto the page displayed
  - Store the state information

Navigation Model

- A Set of Navigation Rules

```
<?xml version="1.0" encoding="UTF-8"?>
<navigation-case>
  <from-view-id>source.jsp</from-view-id>
  <to-view-id>dest1.jsp</to-view-id>
  <action>action1</action>
</navigation-case>
<navigation-case>
  <from-view-id>source.jsp</from-view-id>
  <to-view-id>dest2.jsp</to-view-id>
  <action>action2</action>
</navigation-case>
```

Backing Bean Management

- Backing Beans
  - Java Beans whose properties or methods are associated with UIComponents

```
<outputText id="operand1" value="#{Addition.operand1}"/>
<outputText id="operand2" value="#{Addition.operand2}"/>
<h:commandButton id="add" value="Calculate" action="success" actionListener="#{Addition.addListener}"/>
```

Backing Bean

```
Addition.java
package cs491.bean;

class Addition {
    private int operand1;
    private int operand2;
    private int result;
    
    public void Add() { operand1 = operand2 = result = 0; }
    public int getOperand1() { return operand1; }
    public int getOperand2() { return operand2; }
    public int getResult() { return result; }
    
    public void setOperand1(int operand1) { this.operand1 = operand1; }
    public void setOperand2(int operand2) { this.operand2 = operand2; }
    public void add() { result = operand1 + operand2; }
}
```

Managed Bean Definition

```
<managed-bean>
    <managed-bean-name>Addition</managed-bean-name>
    <managed-bean-class>cs491.bean.Addition</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```
Installation

- Download Faces package
- JavaServer Faces v1.1.01 Reference Implementation
  http://java.sun.com/j2ee/javaserverfaces/download.html
- Unzip the downloaded file
- Copy Faces API and Required API
  - Copy all jar files in $<download>/lib directory into your web application library directory
    $<WebAppRoot>/WEB-INF/lib

Configuration

- web.xml
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

Development Steps

- Create JSP page with JSF tags
- Define Page Navigation in faces-config.xml
- Develop the Backing Beans
- Add Managed Bean Declaration in faces-config.xml