Session 21

Introduction to Java Server Faces (JSF)

 1

Reading

- Reading
  - IBM Article - www.ibm.com/developerworks/java/library/j-jsf2fu1/index.html

2

JSF Reference

- H Library tags
  - http://www.exadel.com/web/portal/jsftags-guide
- Sun Tutorial (chapters 4-9)
  - download.oracle.com/javaee/6/tutorial/doc/
- Core Java Server Faces Book
  - horstmann.com/corejsf/
- API
  - download.oracle.com/javaee/6/api/index.html

3

History

- 2002 - First JSF standard
  - Some design problems
  - Implemented by framework developers
- 2009 - JSF 2.0
  - Includes facelets, the preferred presentation technology
  - Corrects many design issues & simplifies programming
  - Compatible with major frameworks
  - More efficient
- 2010 - Core JSF textbook published

4

Architectural Issues

- MVC Architecture developed in stages
  - Model 1 architecture - jsp handles requests, and populates itself from a bean
  - Model 2 architecture - request is handled by a controller servlet, which populates a bean and selects a JSP for viewing data

The Model 2 architecture was the basis for JSF

Becoming the dominant programming style for Web applications

5

What is JSF ...

- A standard to simplify Java EE development
- Can be implemented with drag-and-drop visual frameworks
- Removes much html low-level coding
- Allows for high-level GUI components (e.g. trees)
- Includes
  - An API for representing and managing UI components
  - Tag libraries (JSF custom tags are similar to JSP custom tags)

6

... What is JSF ...

- Manages UI components on the server side
  - Event handling
  - Validation
  - Back end data handling

Compatible with the Ajax programming model

- Designed to be used effectively by tools (NetBeans has a visual component)

A major goal of JSF is to allow tool vendors to provide JSF frameworks that developers can use to build Web applications using drag and drop techniques

7

... What is JSF

- Provides a higher level abstraction for Web development
  - Event driven programming compared to request/response
  - MVC for the Web
  - Support for client device independence
  - UI elements are stateful objects on the server
  - Works with multiple presentation technologies (e.g., JSP, facelets)

Requests are converted into server side events and handled by event handlers

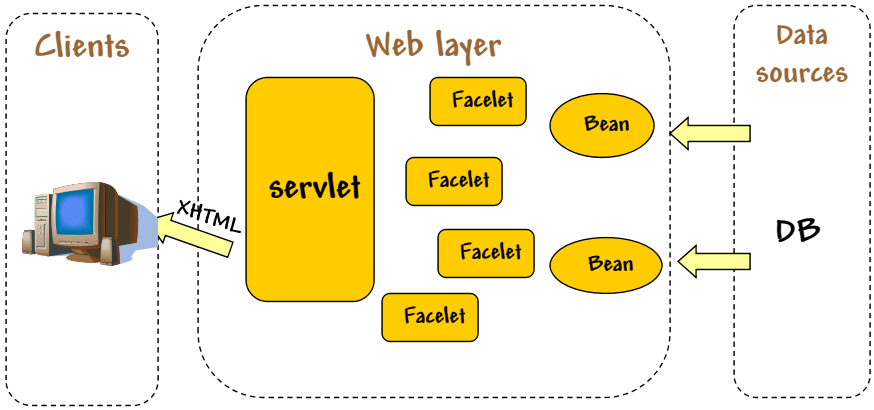
8

JSF Separation

- Application code - contained in beans
- Design - contained in Web pages

9

JSFWeb Architecture



What Does Your Project Include?

- Html form
- Controller servlet (populate bean, forward to JSP)
- Bean
- Code to manage the bean
- JavaScript/Ajax validation

These are all easier to code with JSF 2.0

Backing Beans

- A managed bean
  - Is a Java Bean that can be accessed from a JSF
  - Contains properties bound to components
  - Can include methods to handle validation, navigation, and event handling
- Managed – when a bean name occurs in a JSF page, the implementation locates (or instantiates) the object
- JSF EL is used to bind a component's value to a bean property (similar to JSP EL)

You specify name and score through Java Annotation

```
@ManagedBean(name = "hello")
@RequestScoped
```

Hello World JSF Facelet

Note mixture of JSF tags and html tags

beanhello.xhtml

Facelets behaves like a JSP on steroids

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>Facelet Hello World</title>
  </h:head>
  <h:body>
    #{hello.world}
  </h:body>
</html>
```

Accesses JSF h library

Uses the head element in the JSF library

EL expression that binds to the hello bean

Note the URL

13

JSF Hello World Bean

Looks almost like a standard Java Bean

```
package lectures;

import javax.faces.bean.ManagedBean;

@ManagedBean(name = "hello")
@RequestScoped
public class HelloWorldBean {

    final String world = "Hello World";

    public String getWorld() {
        return world;
    }
}
```

Uses Java Annotation

In a way, replaces useBean

Bean is managed by the container and a handle to the bean is placed in the appropriate scope

Alternate, preferred syntax is @Named("hello")

14

Tag Libraries Available in Facelets

- ui - tags for templating
- h - JSF tags for all UI components
- f - tags for JSF custom actions (e.g., actionListener)
- c - JSTL c library
- fn - JSTL functions tags

15

Selected JSF Tags

- h:head
- h:body
- h:form
- h:inputText
- h:inputSecret
- h:commandButton (action attribute specifies url)
- h:outputText
- h:graphicImage

A majority of tags share the following attributes: id, binding, rendered, value, required, etc.)

Your facelet must be valid xhtml

16

EL

- EL was extended to support JSF
- Notation is slightly different, but syntax remains very similar

  `${mybean.property}` ⟺ `#{mybean.property}`
- You can also use EL to:
  - Defer evaluation of expressions
  - Set in addition to get
  - Invoke methods

17

JSF Hello World Bean

Looks almost like a standard Java Bean

```
package lectures;

import javax.faces.bean.ManagedBean;

@ManagedBean(name = "hello")
public class HelloWorldBean {

    final String world = "Hello World";

    public String getWorld() {
        return world;
    }
}
```

Uses Java Annotation

In a way, replaces useBean

Bean is managed by the container and a handle to the bean is placed in the appropriate scope

18

Annotation Recap ...

- Part of Java language, starting with Java 5
- Annotations are tags that you insert into source code so that some tool can process it (not part of the normal execution of the program)
- Proper style places the annotation on a line preceding the statement it relates to

```
@ManagedBean(name = "hello")
public class Hello {
```

An EL reference to hello will refer to this bean

Think of it as a modifier for the declaration

You can annotate classes, methods, fields, and local variables

19

... Annotation recap

- Annotations can be defined to have elements

```
@ManagedBean(name = "hello")
public class HelloWorldBean {
```

- Examples
  - Unit testing (JUnit)
  - JPA
  - Mapping classes to XML
  - Defining and using Web services
  - Specifying deployment information

20

Annotation Sample for Bean

- ManagedBean (properties: name and eager)
- RequestScoped
- SessionScoped

Eager element refers to the timing of the instantiation of the bean (application start-up or first use)

21

web.xml for JSF Hello World

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

In JSF all requests are automatically handled by a single container-generated servlet

22

JSF Stages

- Application server (e.g., Glassfish 3.0) started
- Client request facelet (hello facelet is compiled)
- A new component tree is constructed for facelet
- Component tree is populated with bean contents
- A new view is built and rendered as a response
- Component tree is destroyed
- On postbacks, the component tree is rebuilt and saved state is applied

23

Are We on Track Hints

- Write a managed bean (with getters and setters) with properties of name and password
- Be sure to include the correct annotation in your managed bean to name the bean user and store in the request.
- Welcome.xhtml will be your facelet that displays the response to the form request
- Associate the values in your facelet with bean properties
- Be sure to include the submit button in the form
- Your welcome.xhtml facelet will refer to the name property of the bean

26

Intro to Ajax JSF 2.0

- Use additional JSF tags
- Include id attributes for components that are accessed by client code
- Provide text fields for Ajax message, which refers to a new bean property
- Bean property contains logic of server validation

30

Example - Login with Ajax

31

Example ...

loginajax.xhtml

```
 <?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">>
    <h:head>
        <title>Welcome</title>
    </h:head>
```

32

...Example ...

```
<h:body>
  <h:form prependId="false">
  <h3>Please enter your name and password.</h3>
  <table> <tr> <td>Name:</td>
      <td><h:inputText value="#{user.name}" id="name" />
      </td> </tr>
    <tr> <td>Password:</td>
      <td><h:inputSecret
        value="#{user.password}" id="password" /> </td></tr>
  </table>
  <p><h:commandButton value="Login" >
  <f:ajax execute="name password" render="out" />
  </h:commandButton></p>
  <h3>
    <h:outputText id="out" value="#{user.greeting}"/>
  </h3> </h:form> </h:body> </html>
```

Does not prepend form name to ids (simplifies the names)

Separate Ajax calls

Ajax execute contained within login button - when the login button is clicked, Ajax request is send to server

33

... Example

- Modified bean

```
Private String name = “”;
private String greeting = "Welcome to JSF + Ajax,";
...
public String getGreeting() {
        if (name.length() == 0) return "";
        else return greeting + name + "!";
    }
```

34

Assignment 8

- Extra credit (1 point)
  - Convert your project to a JSF application
  - No validation
  - Generate the form populated with values from your bean
- Extra, extra credit (1 point)
  - Add validation

36