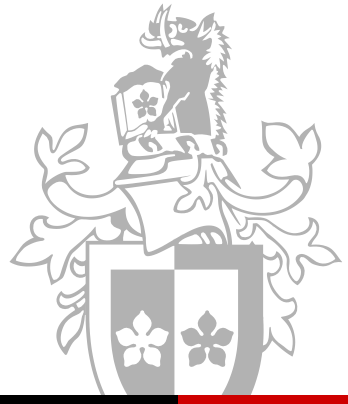
COS30041 Creating Secure and Scalable Software

Lecture 05b – JavaServer Faces and related technologies



SWINBURNE UNIVERSITY OF TECHNOLOGY

Learning Objective

- After studying the lecture material, you will be able to
  - Understand and explain how to use JavaServer Faces Technology to build web application in Java EE 7
  - Understand and explain the roles of using different components of JSF in Java EE 7
  - Program an enterprise web application using JSF technology

Pre-requisite

- Stateless Session Bean
- Entity Class

Outline

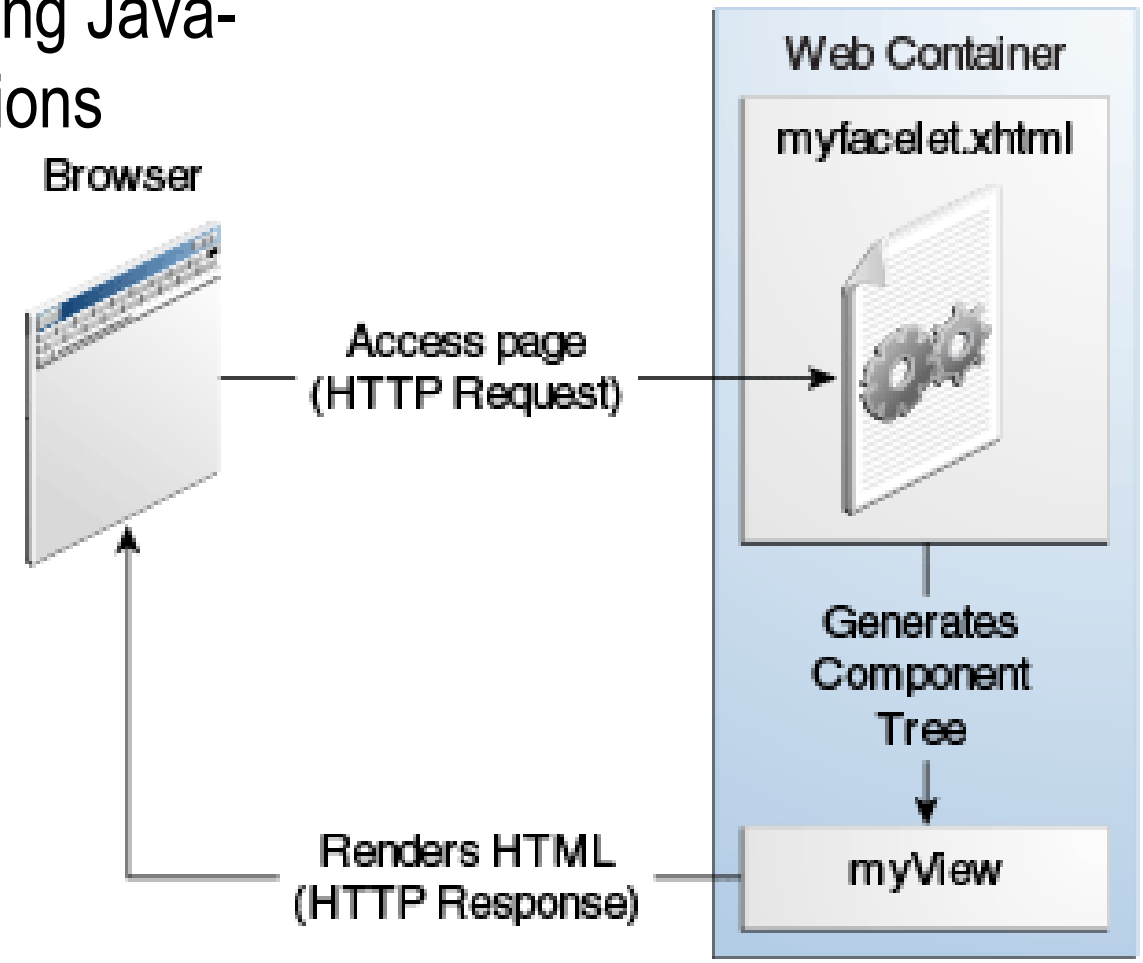
- JavaServer Faces
- Problem for Discussions
- Expression Language

Roadmap

- **JavaServer Faces**
- Problem for Discussions
- Expression Language, EL

JavaServer Faces, JSF

- A server-side UI component framework for building Java-based web applications



Srvlt+JSP+JB versus JSF

- Servlet as a controller
  - The logics of which JSP pages to dispatch
  - Coded by developers
- JSP pages
  - HTML tag, JSP tag for UI
  - EL in JSP
  - JSTL
- JavaBeans components
  - Properties, Getters and Setters
  - Information sharing is managed by developers

- Map to a FacesServlet instance
  - The logics becomes the page navigation rules
  - No need to code the servlet
- JSF pages – xhtml scripting*
  - Better UI components in JSF
  - EL in JSF
  - JSF standard tag libraries
- Managed Beans
  - Same as JavaBeans components
  - Information sharing is managed by the Web container

*Note: JSF now uses XHTML as the default scripting language

JSF – Main Components

- An API for
  - Representing UI components and managing their state
  - Handling events, server-side validation and data conversion
  - Defining page navigation
  - Supporting internationalization and accessibility
  - Providing extensibility for all these features
- Two JSF custom tag libraries for
  - expressing UI components within an xhtml page
  - wiring components to (web) server-side objects

Why JSF?

- Offer a finer-grained separation between behaviour (logics) and presentation (view) than JSP because
  - JSP cannot map HTTP requests to component-specific event handling
  - JSP cannot manage UI elements as stateful objects on the web server (container)
- Allow developers to build web application like traditional client-side UI architecture
- An important goal: to make the UI-component independent of any particular scripting technology or markup language
  - Implication: redefine some existing (probably well known) UI components using JSF

JSF Application

- A set of JSF pages
- A set of **Managed Beans**
  - JavaBeans components that define properties and functions for UI components on a page
- An application configuration resource file (usually called `faces-config.xml`)
  - Optional if the standard / default navigation is assumed
  - Define page navigation rules
  - Configure beans and other custom objects* (see next slide)

JSF Application (cont'd)

- A deployment descriptor
  - `web.xml`
- Possibly a set of custom objects
  - Custom objects may include custom components, validators, convertors, or listeners
- A set of custom tags for representing custom objects, if any, on the page

JSF Application – Who is doing What?

Who	What
Application architect	Configure the application including defining the navigation rules, configuring custom objects, and creating deployment descriptors
Application developer or programmer	Develop custom converters, validators, listeners, and backing beans
Component author	Develop custom UI components and renderers
Page author	Develop JSF pages (with JSF tag lib)

JSF Application – Generic Steps

- Map the FacesServlet instance
- Create the pages using the UI component and core tags
- Defining page navigation
  - Put them in the web pages if implicit rules are used
  - Put them in the application configuration resource file
- Develop the managed beans
- Add managed bean declarations to the application configuration resource file

Roadmap

- JavaServer Faces
- **Problem for Discussions**
- Expression Language, EL

Example 1 – A simple login case study

- A simple log in action involves the following
  - login page prompts for username and password pair
  - retryLogin page when username and password do not match
  - mainmenu page when login is successful
  - logout page when user wants to logout
- How do we control the flow of these 4 JSF pages
- The username and password pair is stored in a database
- The checking of username-password pair is done by a stateless session bean

Ex. 1 – A simple login case study (cont'd)

Before we get to the code, we have to fix the following first

- How to get inputs from the web pages?
  - Where do we store these information?
- How to determine whether the login is successful or not after getting the inputs (say, userid and password)?
  - Where should we initiate such request?
- What are the page navigation rules?
- How to activate certain web pages based on dynamic information supplied by the users?

Ex. 1 – A simple login case study (cont'd)

- How to get inputs from the web pages in JSF?
  - Use **deferred** expression language (EL) to set values in the Managed Beans
  - *Example:

    ```
    <h:inputText value="#{userBean.userid}"/>
    ```

    assuming we have the `userBean` object (a Managed Bean) and it has the property `userid`

*Note the use of "#" not "$"

Ex. 1 – A simple login case study (cont'd)

- How to determine whether the login is "successful or not"?
  - Once the values have been set in the Managed Beans component, the Managed Beans then requests the Login stateless session bean to check the credentials
  - Use deferred EL to initiate a method call in the Managed Beans
- *Example:

```
<h:commandButton value="Login"
    action="#{userBean.loginResult}" />
```

assuming we have the `userBean` object [the Managed Beans] and it has the `loginResult()` method

*Note the use of "#" not "$"

Ex. 1 – A simple login case study (cont'd)

- What are the page navigation rules?
  - login.xhtml : if "success" → mainmenu.xhtml
  - login.xhtml : if "failure" → retryLogin.xhtml
  - mainmenu.xhtml : if "logout" → logout.xhtml
    - Option 1: define navigation rule: if "logout" → logout.xhtml
    - Option 2: specify the link in the mainmenu.xhtml

Ex. 1 – A simple login case study (cont'd)

- Since the action in "login.xhtml" depends on the actual login result, how can we put variable actions in a JSF page?
  - Code the Managed Bean so that the method call returns a String
    - In fact, it can be an Object. However, String is the easiest to handle
  - Example: `public String loginResult()` returns
    - "success" if login is successful
    - "failure" otherwise
  - Use deferred EL to defer the decision of the action
  - *Example:

```
<h:commandButton action="#{userBean.loginResult}"
value="Login" />
```

    assuming the `userBean` object has the `loginResult()` method

Ex. 1 – A simple login case study (cont'd)

JSF application – Generic Steps

- Map the FacesServlet instance – settings in `web.xml`
- *Create UI … – create the web pages using JSF's tag libraries
- Define page naviagation rule – settings in `faces-config.xml`
- *Develop the managed bean – code the bean
- Add managed bean declarations – settings in `faces-config.xml`

See sample code – EX-Login-JSF.zip

Ex. 1 – A simple login case study (cont'd)

JSF application – Generic Steps

- Map the FacesServlet instance – settings in `web.xml`

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

Ex. 1 – A simple login case study (cont'd)

JSF application – Generic Steps

- Create the JSF pages with JSF tag libraries
  - Create UI components etc.

Ex. 1 – A simple login case study (cont'd)

JSF application
– Generic Steps

- Define the page navigation rules (`faces-config.xml`)
  - ☐ login.xhtml: if "success" → mainmenu.xhtml
  - ☐ login.xhtml: if "failure" → retryLogin.xhtml
  - ☐ [option 1]mainmenu.xhtml: if "logout" → logout.xhtml

```
<navigation-rule>
  <from-view-id>/login.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/mainmenu.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>failure</from-outcome>
    <to-view-id>/retryLogin.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
<!-- Option 1 | omit if using Option 2 -->
<navigation-rule>
  <from-view-id>/mainmenu.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>logout</from-outcome>
    <to-view-id>/logout.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

Ex. 1 – A simple login case study (cont'd)

JSF application – Generic Steps

- Develop the Managed Beans
  - Code the required properties, their getters and setters
  - Code the required methods for making the "business" requests and return the appropriate "String" for the FacesServlet instance to invoke the appropriate JSF pages

Ex. 1 – A simple login case study (cont'd)

JSF application – Generic Steps

- Add the managed bean declaration in `faces-config.xml`
  - The Managed Beans are managed by the Web container

```
<managed-bean>
  <managed-bean-name>userBean</managed-bean-name>
  <managed-bean-class>
    web.managed.UserBean
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Roadmap

- JavaServer Faces
- Problem for Discussions
- **Expression Language, EL**

Expression Language in JSF, EL

- *[Recall] EL in JSP 2.0 – allow page authors to use simple expressions to dynamically read data from JavaBeans*
- EL in JSF 1.0 – enrich the features in JSP for JSF purposes

  Assume the "userBean" object has the property "userid" and the method "loginResult()"

  - **Deferred** evaluation of expressions
    e.g. #{userBean.userid}
  - Ability to **set** data as well as get data
    e.g. <h:inputText value="#{userBean.userid}"/>
  - Ability to **invoke method**
    e.g. <h:commandButton action="#{userBean.loginResult}" … />

Read [JEE7T] Ch. 9 for details

Reference

- [JEE7T] E. Jendrock et al. (2014) *The Java EE 7 Tutorial*, Oracle, August 2014
  - Chapters 7 – 16