CO3409 Distributed Enterprise Systems Introduction to Java Server Faces (JSFs)

Summary

JavaServer Faces (JSF) is a user interface (UI) framework for Java web applications. It is designed to significantly ease the burden of writing and maintaining applications that run on a Java application server and render their UIs back to a target client. JSFs is based on the Model, View, Controller (MVC) architecture, which is a well-known way of structuring an application to separate the user interface from the processing. JSF is part of Java Enterprise Edition.

This exercise introduces the JSFs and ideas that will occur later in the module such as bean, framework, annotation, convention over configuration.

Purpose

On completion of this you should be able to

1. Use NetBeans to create a JSF-based application.
2. Explain the meaning of MVC architecture, managed bean, navigation, annotation, configuration, DRY, convention over configuration, framework, facelet, template.
3. Outline the operation of JSF applications.
4. Outline some advantages & disadvantages of JSF.

Introduction to JSF

A JSF page is represented by a tree of UI components, called a view. When a client makes a request for the page, the life cycle starts. The JSF implementation must build the view using state saved from a previous submission of the page. A request is processed through the following stages: (see https://docs.oracle.com/javaee/7/tutorial/jsf-intro006.htm#BNAQQ)

1. Construct a tree representing the components in the view. This will contain the previous state of the components.
2. Apply any parameters provided in the current request – i.e. update the state of any components.
3. Validate the new values.
4. Update the underlying model with the (validated) values from the user

interface
5. Carry out any final processing.
6. Render the display based on the results.

After various stages, any event handlers that have been attached to the components are triggered.

Activities

You are going to do a Netbeans tutorial, which is one way that practising programmers learn relevant concepts and techniques. There are questions at the end to allow you to check if you have understood. To answer them, you may have to search for information for yourself as well as use the links I provide.

HERE BE DRAGONS!
Don't ignore the notes section – read it along while you're working on the tutorial so that you will recognise where in the tutorial you need to refer back to it.

1. First JSF-based Web Application

Carry out the NetBeans tutorial at http://netbeans.org/kb/docs/web/jsf20-intro.html.

You can download `jsfDemo` project from Blackboard or from the tutorial page. Pay close attention to the following notes. They are extremely useful (you have been warned!).

Note : Adding JSF 2.0 Support to a Web Application

1. (Step 3 & 4) The tutorial uses `xhtm`l files and asks you to display them in the browser. Old versions of Internet Explorer won't display `xhtml`, but simply asked if you wanted to download them. Apparently, it refused to recognise the mime type (content type) that the Server attaches to them. If you downloaded them and rename them `.html`, it would display them correctly.
2. (Step 8) JSF is implemented by a servlet cunningly called the Faces Servlet. This takes the client request and any session information and takes it through the JSF request lifecycle. The servlet URL pattern is used by the server (Glassfish) to decide whether to pass the HTTP request from the client to the Faces servlet.
3. (Step 9) The deployment descriptor is used to pass information to the server, when the web application is *deployed*. You can view the `web.xml` deployment descriptor by double-clicking on it. When it is open in the editor pane, you can look at the raw XML or you can look at it in a structured way, by clicking on appropriate tabs and answer the following questions:.

    a) In which section is the servlet pattern stored?

    b) What is the purpose of the welcome-file-list section?

Note : Creating a Managed Bean

1. A POJO conforms to the JavaBeans naming convention – essentially properties are only accessed through getters and setters, where the name of the getter and setter for a property `XXX` is `getXXX()` or `setXXX()`. Getters for Boolean properties can be called `isXXX()`. If a bean can generate an event, `YYY`, it will allow other beans to register to be notified when the event occurs using a method called `addYYYListener`. There should also be a method called `removeYYYListener`.
2. (Step 2) Don't forget to enter both the Name and the Class Name of the managed bean. The Name is really the name of the instance of the class that will be created by the server and used by the application.
3. Annotations are a new feature of Java and C#. Essentially, they are machine-processable extensions to the language. They are objects and are stored with the compiled code. They are used to affect the way an item is processed by the compiler or the run-time system rather than to change the item itself. Information for the server can be stored in annotations instead of in a separate configuration file, which often uses XML. This can reduce the duplication of information (following the **DRY principle** – Don't Repeat Yourself).

   In this tutorial, the usage of the annotaion `@ManagedBean` is being phased out. Therefore, you'll notice in your created bean in NetBeans that the annotation `@Named(value = "UserNumberBean")` is used instead. The main difference is that `@Named` beans are visible to the whole JEE container, while `@ManagedBean` are visible only to the JSF container.

Note : Wiring Managed Beans to Pages : `index.xhtml`

1. Expression language (EL) is part of JSF. Expressions can access Java Beans (e.g. `#{UserNumberBean.userNumber}` is evaluated by the server.)
2. Replace the opening and closing head tags in both `index.xhtml` and `response.xhtml` with `<h:head> … </h:head>` otherwise you will get an error message displayed on the pages: One or more resources have the target of `head`, but no `head` component has been defined within the view.

2. An Alternative Implementation using JSP

Right-click on the Web Pages folder in the Project pane and create a new JSP file called `simple.jsp` (don't add .jsp – **why?**).

Replace the contents of `simple.jsp` with:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
```

```
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>Number Guessing Game in JSP</title>
    </head>
    <body>
        <p><jsp:useBean id="userNumberBean" scope="session"
class="guessNumber.UserNumberBean" /></p>
        <h1>Guessing Game</h1>
        Debugging — theGuess parameter was:
<%=request.getParameter("theGuess")%>
        <%
          int theVal;
          try {
              theVal =
Integer.parseInt(request.getParameter("theGuess"));
          }
          catch (Exception ex){
              theVal = 0;
              }
          userNumberBean.setUserNumber(theVal);

        %>
        <p>Your number was <%=userNumberBean.getUserNumber()%>
</p>
        <p><%=userNumberBean.getResponse()%></p>
        <form name="jspform" action="simple.jsp">
            <input type="text" name="theGuess" value="
<%=userNumberBean.getUserNumber()%>" /><br/>
            <input type="submit"/>
        </form>
    </body>
</html>
```

Try it out **and explain how it works**. Notice that it is using the same bean, which writes its "hidden" random number to the server output page.

Make sure that you explore what happens after the number has been guessed correctly and fix the program. **Evaluate your fix** – does it raise any long-term issues – is the bean too tightly coupled to the Faces approach to be useful with JSP? Would it make the application better if you added a `bool check()` method and a `void reset()` method to the bean?

3. Answer the following questions

a) What is the purpose of JSF?

b) JSF is a framework. What is a framework?

c) What is a namespace?

d) What namespace includes the JSF tags?

e) How are the JSF tags identified? How could you change this?

f) What is a POJO?

g) One of the benefits of using POJOs is that the beans can be reused in other contexts, can the UserNumberBean be easily reused in a different application.

h) What is the MVC architecture?

i) In the JSF implementation of the MVC architecture, the controller is the FacesServer. What are the view and the model in the application you've developed?

j) What is deployment?

k) What is configuration?

l) What language is used for configuration files in JEE?

m) State two ways in which configuration information can be provided in a JSF application.

n) What is EL?

o) What property does the JavaBean, UserNumberBean have?

p) What is a template used for in JSF?

q) What presentation technology is used for the view in JSF 1? What about JSF 2? What language is used for the latest presentation technology (also known as a view declaration language)?

r) What is meant by "wiring a bean to a page"?

s) What is meant by navigation in JSF? How was it done in JSF 1? Outline two ways in which navigation can be specified in JSF 2.

t) What happens in the validation stage of the JSF request processing lifecycle?

u) Both the JSP and JSF applications make use of a bean with session scope, what does this mean?

v) Which of the two implementations is simpler, the JSP or the JSF implementation?

w) What are the advantages and disadvantages of JSF?

Additional Reading

- **The Java EE 7 Tutorial** – Chapter 12: Developing with JavaServer Faces Technology (https://docs.oracle.com/javaee/7/tutorial/jsf-develop.htm (visited 06/10/2017))