INFOSYS 290-17: Java Web Applications

Assignment: Hands-On Model-View-Controller

Table of contents

1. Introduction

In this assignment, you will gain some hand-on experience with the model-view-controller architecture/paradigm using Java, servlets, and JSP. You will create your own versions of the beer advice applications (versions 1, 2, and 3) as presented in [HFSJSP].

2. Instructions

Develop, deploy and test the following excercises (example web applications) as described in chapter 3 of [HFSJSP]:

1. Beer Advisor, V1 (starts on page 75)
2. Beer Advisor, V2
3. Beer Advisor, V3

Lastly, create your own web application using Beer Advisor, V3, as a template. If you are partial to red meat, you could create a Beef Advisor application. Or, perhaps a Tofu Advisor!

1. Your Own Advisor Application

Read the SIMS Tomcat Deployment for guidance on getting your application(s) deployed in the SIMS Tomcat environment. The deployment is somewhat different than what is presented in the book.

As you are developing the various versions of the beer advisor application, observe how the model, view, and controller, are components are coming into play. How might this architecture make for better applications?

3. SIMS Tomcat Deployment

You SIMS account will have a `public_html.jsp` subdirectory (folder). The Un*x path for this will be something like `/home/oski/public_html.jsp`. Note that it is a directory even though the name ends in ".jsp." You will also be able to see this directory when you login using a MS Windows session. Unlike the examples in the book, where each application might have its own deployment directory with its own `WEB-INF` and its own `web.xml`, you get one `WEB-INF` and one `web.xml`. This single `WEB-INF` and `web.xml` will be share amongst all the applications you develop for this class (assuming you continue to use this SIMS Tomcat environment). In general, this is not a desireable situtation for developing independent web applications; however, it is desirable from a multi-user Tomcat management perspective.

Thus, the content of your `web.xml` might need to be a bit different than what is presented in the book. For example, the book has:

```
<servlet-mapping>
    <servlet-name>Ch 3 Beer</servlet-name>
    <url-pattern>/SelectBeer.do</url-pattern>
</servlet-mapping>
```

If you deploy your code into the directory `public_html.jsp/Beer-v1`, the URL pattern will need to be changed as follows:

```
<servlet-mapping>
    <servlet-name>Ch 3 Beer</servlet-name>
    <url-pattern>Beer-v1/SelectBeer.do</url-pattern>
</servlet-mapping>
```

As you might see, this will have some `web.xml` consequences if you are trying to host multiple versions of your application.

After making changes to your `web.xml` file, go to the Tomcat manager and reload your web application environment: look for your userid under the Path column in the Applications table. The Reload link can be found in the Commands column for you userid entry.

3.1. Told-Ya-So: Deployment

Be sure you get the files properly placed in the deployment environment (e.g. `tomcat/webapps/Beer-v1` or `dilbert/public_html.jsp`). For example, the Java class files (e.g. the servlet) belongs in the `WEB-INF/classes` directory (or subdirectories if the classes are in Java packages) and not, say, `WEB-INF` directory.

If you don't get this Java class files deployed properly, Tomcat will report a class not found exception:

```
javax.servlet.ServletException: Wrapper cannot find servlet class ...
```

4. Personal Tomcat Manager

If you are using a Tomcat server on your own computer, you do not need to shut down the server and start it back up ([HFSJSP, p. 85]) every time you deploy or update an application. This is probably covered elsewhere in [HFSJSP]. Until then, you can follow these instructions:

From the Tomcat Web Application Manager web page, enter the directory URL (using using the file: protocol) using the Deploy form: -> Deploy -> Deploy directory or WAR file located on server -> WAR or Directory URL. Click the Deploy button after entering a valid URL. For example: `file:///C:/Platform/apache/tomcat/webapps/beerV2`

5. Eclipse

You can use an Eclipse Java project to develop your web applicaitons. This also works nicely if you have an XML editor application that plugs into Eclipse such as Oxygen.

Be sure to familarize yourself with the development and deployment environment shown in [HFSJSP] page 72.

5.1. Imports

Note that the java build path needs to include Tomcat's common/lib/servlet-api.jar to get at javax.servlet.* classes. Use the project's Properties dialog and adjust the Java Build Path appropriately. In the SIMS environment, Tomcat's common directory is located TBD.

5.2. Output Folder

You will need to make some decisions on the development and deployment process. Eclipse,

by default, puts Java class files in the same directory as the Java source files. If you want to have Eclipse help with the deployment, you can do a variety of differnt things.

The best solution is to use an ANT script/task to handle the build and deployment. However, learning to use ANT is beyond what we want to do for this assignment.

One recommended approach is to configure your Eclipse project to mirror the recommend development environment as shown on page 72 of [HFSJSP]. Change the output directory for the project using Properties dialog -> Java Build Path -> Default output folder. This approach will require you to do copy the classes directory (and subdirectories) into the WEB-INF directory after the Java code has been compiled. This is simple as a drag-n-drop or cut-and-paste of directories in the Eclipse Navigator view. I would recommend this approach as it teaches you the classic Tomcat/JSP development environment (which might be important if you take the SCWCD exam for J2EE).

Another approach is to get Eclipse to deploy the class files into the WEB-INF/classes directory. Change the output directory for the project using Properties dialog -> Java Build Path -> Default output folder. One drawback to this approach, is that Eclipse will copy over non-Java directories and files (e.g. etc/web.xml) and Eclipse may not put them where you want them. In this case, you could simply edit the files in the target deployment location rather than keeping a copy in the development location.

6. NetBeans

NetBeans might be a friendlier IDE to use for developing Java web applications. If you are interested in trying it out, please do and let the rest of the class know about your experience!