CSC309S Programming on the Web
Assignment 3

Due: 11:59 PM, April 9th, 2009 (worth 10%)

Note: There will be no extensions for this assignment.

Overview

In the first two assignments, you have developed the front-end skeleton and the backend layer of the media website. In this assignment, you are going to use Java Servlets and JSPs to build the middle tier layer which connects both ends and you are also going to convert the front-end into a functioning website.

Description

- Following is a list of screens you are required to develop:
  1. Homepage (you are only required to support simple search - Advanced link is excluded from this assignment).
  2. Register
  3. Login (you can develop this either using a window or a separate page).
  4. My Content (licensed content is excluded)
  5. Subscribe to Channel
  6. My Subscriptions
  7. Accounting (the filter and start/end date are excluded - you can assume that the search will always be from the start date of the website to the date of today).
- Note that each of the above screens does not necessarily map to one xml request/response. For example, the homepage could be used to trigger search request or subscribe request.
- Also, note that some of these screens require the user to be logged-in first.
- Use model-2 architecture with one central servlet and the command design pattern. The list of commands that your web application supports must be loaded from a configuration file where the name of the command is mapped to a class name. The name of the command should be part of the request which the servlet receives. Use reflection to load the command-implementing class dynamically (see slides 65, 66 and 67 in lecture 10).
- Support concurrency in your webapp by synchronizing on shared resources (see slides 30→38 in lecture 10). You are not allowed to use the single-thread model.
- Convert each relevant .html file you've developed in assignment 1 to a .jsp file (JSP tags will be used to initialize the DHTMLX components).
- You are required to support paging of DHTMLX grid-control/accordion-control data; load a max of 20 rows at a time.
- You can use JSTL tags.

Software

- You will use Tomcat as both the Servlet container and also the web server.

- Tomcat is available from http://tomcat.apache.org/

- The software is already installed on CDF with user privileges at:
  /u/csc309h/lib/apache-tomcat-5.5.26

- To assist you in getting your server up and running there are startup files provided in the assignment section in blackboard (a3-files.rar). Copy the **tomcat.tar.gz** file to a directory somewhere under your home directory from which you will be running your tomcat server. In the rest of this document, we will refer to this directory as **$CATALINA_HOME**.

- **Starting and Stopping Tomcat:**

  - cd to **$CATALINA_HOME/bin**
  - Run the script **start.sh** (it is important that you run this file from within the bin directory)
  - Enter the port number you were assigned for running Tomcat. Make sure you only use your assigned port number (ports file is in assignment folder on blackboard). Tomcat will use 3 ports, your assigned number and the next two (e.g., if you are assigned port 32000, tomcat will use 32000, 32001, 32002).
  - The **start.sh** script will create the configuration file **conf/server.xml**
  - To check if your web server is running, issue the command (on CDF): **/bin/ps -ef | grep your_user_id**
  - You can now point your browser at **http://localhost:your_port_number** and you will see the default tomcat page displayed.
  - To stop your server, run the **bin/stop.sh** script. Always remember to stop your server before logging off CDF.
  - You can look through the example servlet pages provided by tomcat.

- **Installing and Compiling Servlets**

  1. Download the sample application from the assignments folder
  2. Decompress and untar the file. Then copy the csc309 directory to **$CATALINA_HOME/webapps** where $CATALINA_HOME is an environment variable that points to the directory where you have installed Tomcat (the directory that contains bin, conf, and webapps).
  3. The sample application consists of a single Servlet implemented by the HelloWorld class
  4. To run the servlet follow these steps:
     a. Compiling the Java class
        1. Include the following jar file in your CLASSPATH
           /u/csc309h/lib/tomcat-5.5.26/common/lib/servlet-api.jar

This library contains class definitions commonly used by servlets, such as the HTTPServlet class. Alternatively, add the commands in **a3-evn.txt** file to your **.cshrc** file in your home directory. For the commands to take effect you will need to either login again or force your shell to reread the configuration file by typing **source .cshrc**

2. cd to **$CATALINA_HOME/webapps/csc309/WEB-INF/classes**
3. Compile the servlet by typing: **javac HelloWorld.java**

b. Start Tomcat

c. Browse to the following URL **http://127.0.0.1:yourPortNumber/csc309/servlet/HelloWorld**

- **Adding a Servlet to a Web Application**

To add the PrintEnv servlet to the csc309 Web Application follow these steps:

1. Copy PrintEnv.java to **$CATALINA_HOME/webapps/csc309/WEB-INF/classes**
2. Compile PrintEnv.java
3. Add the following entries to the application descriptor located at

$CATALINA_HOME/webapps/csc309/WEB-INF/web.xml

```
<servlet>
        <servlet-name>PrintEnv</servlet-name>
 <servlet-class>PrintEnv</servlet-class>
 </servlet>
 <servlet-mapping>
     <servlet-name>PrintEnv</servlet-name>
     <url-pattern>/servlet/PrintEnv</url-pattern>
  </servlet-mapping>
```

4. Restart Tomcat
5. Point your browser to **http://127.0.0.1:yourPortNumber/csc309/servlet/PrintEnv**

- **Debugging Servlets**

Debugging servlets is a little more complex than one would hope. In a nutshell, what you need to do is start Tomcat in debugging mode, and then attach a debugger to the running instance of Tomcat.

This section describes how to debug servlets using the Java Platform Debugger Architecture (JPDA) and the NetBeans IDE.

To debug the HelloWorld servlet described above follow these steps:

1. Compile the servlet with the **-g** flag so that the javac compiler generates all debugging information.
2. Set the environment variable **JPDA_ADDRESS** to your assigned port number + 1000. For example, if your assigned port number is 30901, you can set the environment variable as follows:
   sh-2.05b$ **JPDA_ADDRESS=31901**
   sh-2.05b$ **export JPDA_ADDRESS**

   Tomcat listens on the JPDA_ADDRESS port for debuggers that want to attach the running servlet container.
3. Start Tomcat in debugging mode using the **catalina.sh** script. Type:
   sh-2.05b$ **catalina.sh jpda start**
4. Start the NerBeans IDE by executing the following script:
   /u/csc309h/lib/netbeans-6.0.1/bin/netbeans
5. Under the Debug menu, click on "Start Session" and then on "Attach..."
6. In the Attach dialog box, type the name of the "Host" where Tomcat is running (typically localhost), and the "Port" number to which you set the environment variable **JPDA_ADDRESS** (31901 in our example).
7. If, everything went well you should see the following messages on the "Debugger Console" tab in the lower left corner of the window:
   Connecting to localhost:31901
   Connection established
   - Otherwise, you will get a message box saying something like "Cannot attach. Check the parameters and try again." This is an indication that the debugger was not able to connect to Tomcat. Make sure that the environment variable $JPDA_ADDRESS is set to the right value and that you are starting Tomcat in debugging mode (see Step 3).
8. Select the "Runtime" tab on the top left corner of the window. Click on "Debugger" and then again on "Classes." You should see a list of all the classes that are currently loaded into Tomcat.
9. Point your browser to
   http://127.0.0.1:yourPortNumber/csc309/servlet/HelloWorld
10. Go back to NetBeans, the class HelloWorld should have been added to the list of available classes.
11. Double click on HelloWorld.

Deliverable Directory Structure

<a3-team-name>

1. README.TXT (file containing your team members, name, ids, cdf login. In addition to any missing features you didn't have time to implement)

2. runDBApp (shell script to run the DBApp java application. Should include in the classpath the jars you are using, e.g. /u/csc309h/lib/activemq-core-5.2.0.jar)
3. <db> (directory containing all database related scripts: ddl, stored procedures, inserts, etc…)

4. <config> (directory containing any configuration file for your java programs or the programs/libraries you use)
5. <lib> (directory containing any additional library you are using)
6. <bin> (java class files, you could have your own packages under this directory)
7. <src> (java source files for DBApp, you could have your own packages under this directory)
   - DBApp.java (under <src>)
   - Makefile (under <src> Compiles your application and place executables under <bin>)
8. <web-app> (directory containing all your sources and classes related to your tomcat web application)
   - HOW-TO.TXT (file containing your team members, name, ids, cdf login. In addition to step-by-step guide on how to install and run your web-app)

Submissions

- Your submission must work on CDF.
- Using one of the team members cdf account, submit a single zipped file using the following command: **submit -N assign3 csc309h a3-teamname.zip**
- The zip file should contain all your work including the parent directory (named after your team with prefix a3-).
- Do a **man submit** if you want more information about how to the submit command works.