CGS4854 Tutorial 5

This tutorial will take longer than other tutorials. Start earlier.

Use the same web application that you used for previous tutorials. If you did not do a previous Tutorial, then create a web application named Tutorials in NetBeans. Review the web page that contains instructions on creating a servlet in a NetBeans project: Creating a Project in NetBeans http://users.cis.fiu.edu/~downeyt/cgs4825/netbeans.html

All previous Tutorials should still be accessible and runnable from the application.

If you had errors in any of the previous tutorials, fix them, so that you do not lose points again for old mistakes.

Implement the Persistent Controller in the Tutorials web application. It must work exactly like the example from Chapter 5 of the book. The example from the book also does required validation and uses POST and GET.

- Download the following Zip files to a directory on your system. Unpack them to a directory on your system. These ZIP files contain all the JAR files that are need to implement all the features in Chapter 5.
  - Hibernate ZIP file http://bytesizebook.com/jar/hibernate.zip
  - Non-Hibernate ZIP file http://bytesizebook.com/jar/non-hibernate.zip

- In NetBeans, right click the Libraries folder and add all of the JAR files, that were unzipped, to the web application. If you did Tutorial 3, then the following do NOT need to be added, since they have already been added.
  - log4j-1.2.11.jar (should already be there from Tutorial 3)
  - commons-collections-2.1.1.jar (should already be there from Tutorial 3)
  - commons-logging-1.0.4.jar (should already be there from Tutorial 3)
  - commons-beanutils.jar (should already be there from Tutorial 3)
  - There should be 27 jar files when you are done.

- Add the following to the shared package in the Source Packages folder.
  - InitLog4j.java (should already be there from Tutorial 3)
  - HelperBaseCh4.java (should already be there from Tutorial 3)
  - ButtonMethod.java (should already be there from Tutorial 3)
  - HelperBaseCh5.java
  - HibernateHelper.java
  - PersistentBase.java
  - WebappListener.java

- Place the Controller.java, and ControllerHelper.java in the ch5.persistentData package in the Source Packages folder in NetBeans.
  ****we need to talk about the following in class*******
  - In the following, replace username with your ocelot user name. Notice that the mysql user name has fall11_ prepended to your ocelot user name (word fall followed by number 11). The name of your database is the same as your mysql user name: fall11_username
  - In the controller helper, set the connection URL to jdbc:mysql://ocelot.aul.fiu.edu:3306/fall11_username
  - In the controller helper, set the username to your mysql username: fall11_username
  - In the controller helper, set the password to your mysql password. This starts as your panther ID, but it is a good idea to change it.
  - The contoller and controller helper should handle both GET and POST requests.
  - GET requests should always be treated as a first request to the application.

- You will need to create the RequestDataPersistent.java file and place it in the ch5.persistentData package.
  - Start with a copy of the RequestDataRequired.java, from the first part of Chapter 5.
  - Add annotations so that the bean can be saved to a database.
    - Annotate the class so that it can be saved to its own table.
    - You may add a key field to the bean in one of two ways, but only do one of these
      - Add the Id property as is outlined in the book,
      - or extend the bean from PersistentBase.
      - Only do one of these.

- Place the .jsp files in the ch5.persistentData package in the Source Packages folder in NetBeans.
  - Use the JSPs from the ch4.enhanced controller, but copy them into the ch5.persistentData package.
  - Place all three JSPs in this directory. Please note that you are NOT placing these in the Web Pages folder.

  - Change each form method to POST. If you don't do this, then you will always be stuck on the edit page.
  - Modify the Edit.jsp file so that it will display any errors generated by the validation of the Hobby and Aversion.
  - Modify the process page so that it will display all of the data that is currently in the database.
    - Display the id, hobby and aversion for each row in the table. Place the data from each row on a separate line.
    - Use a JSTL forEach loop to iterate through the rows in the table. You will need to add the taglib statement that includes the JSTL in the JSP.
  - Add a second form to the process page that uses the GET method. The button in the form should be for the edit page.
    - Notice the difference between the form that uses the POST method and the one that uses the GET method.
    - When you use the POST form, then the last record that was added to the data base will be modified.
    - When you use the GET form, then a new record will be added to the data base.
    - The reason for this is that the GET request does not retrieve the old bean from the session.
  - Add a page named Expired.jsp. This will be called in the event that the session expires before the data is written to the database.

- Modify the web.xml file so that it has a servlet definition and a servlet mapping for the servlet controller.
  - Map the controller to the URL /ch5/persistentData/Controller.
  - Add the create initialization parameter to the servlet definition.
    - Set its value to true, to create the table the first time you run the servlet.
    - After the table has been created, set it to false, so that your data will not be lost the next time Tomcat restarts.
  - Add the listener tag so that the WebappListener will close resources when Tomcat shuts down.

- Modify the index.jsp file in the root of the Tutorials web app.
  - Be sure that the index.jsp uses the strict DOCTYPE.
  - Use a relative link from the current location to the controller. Use the url-pattern from the web.xml, but omit the leading /.
  - All the previous Tutorials should also have links from this page. All of them should still work.

Take a look at your log file, you will see that there are messages in it.

- Open the Files tab in NetBeans and navigate to build -> web -> WEB-INF -> logs
- Open the error.log file.

Common Problems

In the event that you cannot connect when you try to access the controller, then Tomcat will get stuck. You will not be able to stop it or start it. Follow one of these steps to free up Tomcat

- Drastic: Reboot
- Not so drastic: open the task manager; click the processes tab; sort by process name; look for java.exe; stop them. One of the java.exe processes is Tomcat. Kill it and you will be able to start it again. Before you start it again, be sure you have found the database configuration error that caused the original problem.

Connecting to a database for the first time can be frustrating. Most of the time, the problem is with the username and password for mysql.

- Try connecting to mysql directly from ocelot.
  mysql -h ocelot.aul.fiu.edu -u fall11_username -p <Enter Key>
  Enter your password on the next line after hitting enter. After you connect, just type exit; to disconnect.
- Check that you are using the mysql username in the initHibernate method in the controller helper; it looks like fall11_username.
- Check that you are using the mysql password in the initHibernate method in the controller helper. This password starts as your Panther ID. It is a good idea to change it.
- Check that you have the correct connection string for ocelot:
  jdbc:mysql://ocelot.aul.fiu.edu:3306/fall11_username
- Be sure to add the initialization parameter named create to the servlet definition for the controller.
- Check that you have changed the create parameter to true in web.xml, so that the table can be created.
- Check that you have changed the create parameter to false in web.xml after you create the table, so

that you can accumulate records in the database.

Submitting Homework
Please ZIP you entire project and email it to me.
Print the index.jsp page and turn it in on Monday

==
NOTE: (DO NOT DO THE FOLLOWING)

At this time we can not do the following instructions because we have not access to ocelot.
They are JUST for your infomation as to what else can we do with this program.

Test the connection in NetBeans
- Open the Services tab.
- Right-click the Databases folder.
- Select Add New Connection.
- Select the MySQL driver from the Driver drop-down list.
- Click Next.
- Set the Host to ocelot.aul.fiu.edu
- Set the port to 3306
- Set the database to fall11_username
- Set the user name to fall11_username
- Set the password to your password (initial password is pid)
- Click Test Connection
- If it does not succeed, then you have a problem. Make changes and try again.
- Click Cancel after you have connected successfully.

Before you submit your homework, be sure that you have set the create parameter in the web.xml file to false.

Be sure that your NetBeans project is configured so that .java files are added to the WAR file:
Adding .java files to a WAR file http://users.cis.fiu.edu/~downeyt/cgs4854/netbeans.html#WAR

Clean and Build your web application. After doing this, navigate to the WAR file in NetBeans and verify that the .java files are included: View WAR File
http://users.cis.fiu.edu/~downeyt/cgs4854/netbeans.html#WHERE
- Since you are using a log file that is in the web application, you will need to shut down Tomcat before you can do a Clean and Build.
- Open the Services (Runtime in 5.5) folder and open the Servers folder.
- Right-click the Tomcat server and select Stop.
- Now you can do a Clean and Build.

In the operating system (not in NetBeans) navigate to the dist folder in the NetBeans project.

Deploy the WAR file on ocelot: Deploy a WAR File http://users.cis.fiu.edu/~downeyt/cgs4854/deploy.html

- After deploying, open the WEB-INF/logs directory in winscp or on ocelot.
- Change the permission of the error.log file to 606 and restart the web app from your manager.
  This will allow Tomcat to write to the file.
- Access your application and visit all the pages, then verify that some new messages have been added to the log file.

The WAR file is too big for you to submit to me. To hand in the tutorial, zip the Tutorials web app directory and upload it. If your web app is named something other than Tutorials, then use that name in the zip command.

- Remove the old zip file from the last upload.
  ```
  rm ~/username.zip
  ```
- Please use the following command to compress your webapps folder.
  ```
  zip -r ~/username ~/cgs4854/webapps/Tutorials -x \*.jar \*.zip \*.war \*.jpg \*.png \*.gif
  ```
  Replacing username with your user name. This will create a file named username.zip.
- Check the size of the ZIP file, it should be less than 3 MB. If it is not, then you probably have another ZIP file stored in the ZIP archive. Delete it and re-zip.
- You then have a choice for submitting the file.
  - If you are logged onto ocelot, use my uploading program to upload the file. Enter this command at the command prompt: ~downeyt/cs/public/webftp/webftp.pl
  - If you are using winscp, then download the zip file to your computer; then, upload the file to me: Submitting homework on-line via the Web.
  - Upload the file as binary.