CGS4854 Program for Chapter 5

If you had any errors in the previous homework, then fix them, so that you don't lose points twice for the same mistake.

*** Please make sure your name and tutorial 5
*** is on the top of the first page

Persistent Controller

- Create a new package for this assignment in your username web app.

  - Do not change the files that were used for the previous assignment.
    - All assignments should still be accessible from the web app.
    - You may copy files from previous assignments into this new package.
    - You may import files from previous assignments, if they are unchanged.
  - Modify the application so that it implements the Persistent Controller from Chapter 5.
    It should also implement the Post Controller and Required Validation.
    Review the steps in Tutorial 5.
    - Add all the necessary files to the shared package.
    - Note that HellperBaseCh5 should be placed in the shared package.
    - Add all the necessary JAR files to the Libraries folder.

  - Bean
    - Place the bean in the same package as the controller.
    - 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.
      - Add a key field to the bean or extend it from PersistentBase.
      - Mark any properties that only have accessors so that they are not saved to the database.
    - The bean should implement required validation.
      - Validate that one of the numeric fields is in a specific range of numbers. Do not include 0 in the range. There are additional annotations that can be used with numeric properties.
        - Remove any default validation for this field.
        - @Min(value=100). Do not use the Hibernate annotation, use mine. Add the following files to the shared package: Min.java, MinValidator.java.
        - @Max(value=200). Do not use the Hibernate annotation, use mine. Add the following files to the shared package: Max.java, MaxValidator.java.
        - @Range(min=100,max=200). This is a built-in annotation for Hibernate: Built-in Annotations for Hibernate.
        - Each of these needs an import statement:
          ```
          import shared.Min;
          import shared.Max;
          import org.hibernate.validator.Range;
          ```
      - Use the Pattern annotation to validate that one of the string properties only contains one of several words.
        - Choose at least three words.
        - The match should not be case sensitive.
      - For each field that is to be validated, display information in the web page that indicates the correct format of the data to be entered.
      - For all the other fields, you must implement either required validation or default validation.

  - Controller Helper
    - Use a unique name, other than "helper" and different than the one for Tutorial 5, to save the controller helper in the session.
    - Change the name of the accessor for the bean to something other than getData.
    - Modify jspLocation so that it returns the correct path for JSPs.
    - Write the current record to the database when the process button is clicked.
      - Do not retrieve records in this method.
      - Write an information message to the log file containing the id of the record that is being written to the database.
    - Add a button method for the view page (described below). Read the records from the database and make them available in the request for next JSP.
      - Do not save to the database in this method.

      - Write a debug message to the log file that contains the number of records that were retrieved from the database.
      - Write a warning message to the log file if no records were retrieved from the database.

  - Use a name other than "database" to store the database records in the request.
  - When you are debugging your application, set the log level to debug. When you submit the assignment, change the log level to error.

- JSPs
  - There should be no hidden fields in any of the JSPs.
  - The JSPs should be in a hidden directory, but not in the same directory as the controller. Move the JSPs to an appropriate subdirectory of WEB-INF (not under classes).
  - Modify all the EL statements so that they refer to the new name that you used to store the helper in the session and the new name you used for the bean accessor.
  - Modify Edit.jsp
    - When appropriate, display error messages next to each input element that implements required validation.
  - Modify Process.jsp.
    - Only display the current record that was saved to the database.
    - Do not display all the records from the database in this page.
    - If you do not have one, add a button that returns to the edit page so the user can change the values that were just saved. When the edit page appears, the current values should appear in the input elements in the page.
    - If you do not have one, add a button that returns to the edit page to start a new over. When the edit page appears, the current values should NOT appear in the input elements.
    - Add a button that sends the user to the view page.
  - Add a new page named View.jsp.
    - Display all the records from the database on this page.
    - Add a button that returns to the edit page so the user can change the values that were just saved. When the edit page appears, the current values should appear in the input elements in the page.
    - Add a button that returns to the edit page to start a new over. When the edit page appears, the current values should NOT appear in the input elements.

- Be sure there is a hypertext link in the index.jsp page to the controller. Use a relative link.
- Be sure to modify the project so that the .java files are placed in the WAR file: Configuring WAR File

Submitting the Assignment

*** Please make sure your name and tutorial 5
*** is on the top of the first page

- 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

- 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.

==

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.

- The web app must use the secure web.xml file and should only allow you and me to access your application:
  Edit tomcat users
- After the app is running in NetBeans, upload it and run it on ocelot: Uploading a WAR file
- Deploy the WAR file on ocelot: Deploy a WAR File
- After deploying, open the WEB-INF/logs directory in winscp or on ocelot.
- Change the permission of the error.log file to 606. This will allow Tomcat to write to the file.
- Use the manager to stop your web application and then start it again. This will force it to reread the web.xml file so that the logger can be initialized.
- Access your application and visit all the pages, then verify that some new messages have been added to the log file.
- To hand in the assignment, zip the following directories into one ZIP file and upload it.

- Please delete your old ZIP file that you used to submit homework.

- rm ~/username.zip
- Replacing username with your user name.
- Please use the following command to compress your cgs4854/webapps/username folder.
- zip -r ~/username ~/cgs4854/webapps/username -x *.jar *.zip *.war
- Replacing username with your user name. This will create a file named username.zip. It contains the username web application.
- Please use the following command to compress your cgs4854/serverRoot folder.
- zip -r ~/username ~/cgs4854/serverRoot -x *.jar *.zip *.war
- Replacing username with your user name. This will add the serverRoot folder to 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.
- Be sure that all your files and subdirectories are included in the zip file.
- After uploading, do not modify your files on ocelot until after your assignment has been graded.