SER422 Spring 2019 Lab2: Assigned 2/7, due 2/19 @ 11:59:00pm via Canvas

In this lab you are required to complete 2 tasks. Both ask you to refactor example code from the class GitHub repository into target design pattern(s).

Task 1: Refactor ClueGame to MVC (35%)

The example in the ClueGame_HttpSession directory in the SER422 repo implements a simple game of Clue using the HttpSession object. However the application is basically 2 long servlets and one object (Guess). Please refactor this app so it is functionally equivalent (works the same) but adheres to the MVC pattern.
Constraints:

1. You must again implement your solution entirely on the server-side, no JS or CSS!
2. You must choose a View technology that is not another servlet (option 1).
3. You should create a proper Controller to route requests, so you have to consider the requests possible and refactor the code as such (meaning, the servlets and URLs you have will change and that is fine).
4. You should implement a Model layer that has the business logic and business objects/data, so be sure to refactor this out of your Servlet and View parts as well.
5. Implement appropriate error handling, meaning make sure you validate inputs you receive and HTTP request types and headers as appropriate. If an error occurs make sure the user can recover in an appropriate way.
6. Make sure you preserve conversational state appropriately.
7. Make sure your code uses best practices, including selecting the best way to move model data from the Controller to the View.

Task 2: Refactor the PhoneFull app to MVC and DAO/Repository (65%)

Task 2 asks you to refactor the PhoneFull app in the class GitHub repo to MVC, and then implement a proper DAO or Repository pattern and provide a second data source. The PhoneFull entry is much like the Phone entry we looked at earlier in class, but this version includes Add and Remove features and a simple web form as an interface

1. Refactor to MVC (20 pts)

Like Activity 1, refactor the app to MVC. The same constraints apply, though you will find this code goes much faster.

2. Introduce a 2nd persistence store (a relational database) and implement a DAO or Repository pattern for data access (45 pts)

Requirements/Constraints:

1. The current PhoneFull implementation uses a flat text file for the persistent store. Add a 2nd persistent store that is a relational database. Provide the DDL statement to create your database and populate it with initial data (make 5 entries but use different values than the text file). Use Apache Derby or MySQL for your database. Be sure to tell us what you used and what is the proper way to setup your database.

---

2. Use JDBC to access your relational database. Make sure than connection information is configured outside of the Java source files. We should be able to modify the configuration of your app without a recompile and a redeploy!
3. Applying either the DAO or the Repository patterns, provide a set of interfaces and underlying implementations that decouple the persistent store from the rest of your code. Provide an external configuration file (I suggest a .properties file) to determine the pattern implementation (flat text file or relational database) that is used at runtime. This should be changeable without recompiling and redeploying the app.
4. Best practices are expected, including but not limited to proper handling of SQLExceptions, JDBC credentials and other information outside of code, proper closing of JDBC database resources, and proper use of PreparedStatements.

Constraints for the entire lab:

1. Absolutely no Javascript or CSS for this lab. This includes AJAX. All functionality must be implemented via HTML, view technologies on the server-side, databases, and Java servlets.
2. No 3rd party frameworks, I want to see you code these patterns yourself.

Extra Credit (25 poiints possible):

Note 1: I do not allow extra credit until you score at least 20 points on activity 1 and 40 points on activity 2!

Note 2: We track extra credit separately from the main grade, and add it to the lab component of your grade at the end of the semester!

EC1 (15 pts): Implement a 3rd persistent store in Task 2 as a Mongo database and extend your DAO/Repository implementation accordingly.

EC2 (10 pts): Externalize your SQL queries in Task 2 from Java source code. Provide a way to put SQL external to compiled source code so you do not have to recompile if a query changes.

Submission:

For submission, you should submit to Canvas a single zipfile named lab2_<asurite1>.zip that has within it 2 source trees under subdirectories task1 and task2.

- Source trees are the directory contents of your development directories without the compiled artifacts (no "classes" or "bin" subdirectories) and no WAR files. All of your *.java, properties, and static resources should be in this file. Your ant clean target should take care of this.
- I expect a build.xml to be provided with targets "compile", "build", "dist", "clean" and "deploy" (at a minimum). You should be able to customize mine quiet easily.
- Put a README.txt file at the root of your source tree to tell us anything you think we need to know to run your lab correctly. For example, you might tell us to edit a build.properties file to point to our local tomcat for deployment.
- If you do EC1 or EC2 (or both) please put these source trees in directories named EC1 and EC2, name the web app context the same thing, and include these source tree folders in your zipfile submission.