Lab 2 - Junit, JDBC and Bean Validation

Summary

The purpose of this assignment is to refresh on the basics of JDBC, learn the basics of Bean Validation, and to practice these techniques using JUnit test cases.

Requirements

Documentation

You will use a README file in your GitHub repository as you did in previous labs.

Database Setup

No custom database for this lab - but you must pick one of the sample databases to work with. You must grant access to the sample database for your itmd4515 user.

Some MySQL sample databases were installed for you during Week 1, however you may need to install a sample database as demonstrated in class. Here are some links to other MySQL sample databases:

- https://dev.mysql.com/doc/index-other.html
- https://github.com/lerocha/chinook-database

Project Setup

Next create a **Java with Maven** Java Application project in NetBeans (*not* a Java Enterprise project) with the following coordinates and configuration:

1. Project Name: **uid-lab2** (uid means your myIIT username)
2. Artifact ID: **uid-lab2** (default is fine)
3. Group ID: **edu.iit.sat.itmd4515.uid**
4. Version: **1.0-SNAPSHOT** (default is fine)
5. Use a base package for your Java code of **edu.iit.sat.itmd4515.uid**

Once your project is open in the NetBeans project window, highlight your project root (sspyriso- lab2 in my example) and then right-click to get the New → File menu. Create a new file named **readme.md.** Write some initial content in the file such as "Your Name Lab 2 README" and save the file

Accept the Lab 2 assignment to create your GitHub repository and configure your projects for source control as documented in Lab 1.

Note, this is a **Java SE** application. It is ***not*** a web application.

Project Requirements

1. What database did you select, and which table are you going to represent as a Java POJO? What fields did you select to map from table to Java class? **Discuss** in your README.
2. Add the following dependencies to your `pom.xml` using a scope you feel is appropriate:
    a. junit (latest non-beta version of junit-jupiter-engine 5)
    b. mysql-connector-java (latest non-beta version of 8)
3. What scope(s) did you select and why? **Discuss** in your README.

4. Follow the Bean Validation Getting Started guide to add Hibernate Validator and dependencies to your `pom.xml` using a scope you feel is appropriate.
5. In your Lab 2 project, create the following as demonstrated in class:
    a. POJO to represent the structure of a single database table. You can pick any table in your preferred sample database (except my demo table). You do not have to traverse relationships. Keep it simple for this lab.
        i. Include all **required** (not null) database columns as fields in your POJO.
        ii. Include a minimum of 2 bean validation constraints in your POJO. These bean validation constraints should **relate** to database constraints (column length, not null, etc) or type of data (email, string, date, etc)
    b. Two JUnit test classes. One should be for testing the validation constraints in your POJO, and the other for testing JDBC CRUD operations.
        i. Make appropriate use of test fixtures, as discussed in class.
        ii. Make appropriate use of helper methods to extract and parameterize functionality - DRY (Don't repeat yourself)
        iii. In your validation test class, include at least 2 test methods proving your bean validation constraints work
        iv. In your JDBC test class, include a test method for each of the 4 CRUD operations (Create, Read, Update, Delete).

        ⚠ *I recommend you consider inserting a row for testing purposes in the @BeforeEach test fixture, and removing it in the @After test fixture. By following that pattern, your test cases will always be testing a consistent database row, and will be completely separate from the "actual" data in the tables themselves.*

        I will be running your projects using maven, so make sure not to deviate from database name, username and password conventions outlined in our initial setup.
6. Document Lab 2 execution in your README by taking a screenshot of your IDE Test Results window. This can be the maven test output window. Discuss any other issues or insights you had with Lab 2.
7. Submit to Blackboard or Beacon
    a. Right click your **uid-lab2** project and select "Clean"
    b. Go to your IDE's Projects directory. Create a zip file of the **uid-lab2** folder named (you guessed it) **uid-lab2.zip**. Submit that zip file to the Blackboard or Beacon assignment.

Graduate/Undergraduate

Graduate students - test your Java SE project on the command line using maven. Document your experience (with code block output) on your README, and discuss how command line Java relates to Maven (hint - think classpath and dependencies). What would you need to do (step by step) in order to run your project without maven, using only the Java SE provided java and javac binaries? In what ways does Maven help us?

Undergraduate students - you can do this as well, and receive extra credit of 5 points on this lab. Sorry, no extra credit for graduate students this time.