Lab 3 - Web Applications, Servlet and JSP

Summary

The purpose of this assignment is to learn the basics of an MVC style approach using JSP/Servlet, and to become familiar with these technologies in order to compare their use to later frameworks.

Requirements

Documentation

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

Database Setup

Use your sample database and itmd4515 user from Lab 2.

Project Setup

Project Setup

Create a **Java with Maven Web Application** project in NetBeans with the following coordinates and configuration:

1. Project Name: **uid-fp** (uid means your myIIT username)
2. Artifact ID: **uid-fp** (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**
6. Under Server, choose Payara Server and Jakarta EE 10 Web

For example, my NetBeans project will be called "**sspyriso-fp**" with a base package and group id of **edu.iit.sat.itmd4515.sspyriso**.

Accept the Lab 3 assignment to create your GitHub repository and configure your projects for source control as documented in Lab 1.

Once your project is open in the NetBeans project window, highlight your project root (sspyriso-fp in my example) and then right-click to get the New → File menu. Create a new file named LAB3README.md. Write some initial content in the file such as "Your Name Lab 3 README" and save the file

We will be working in this repository from now until the end of the semester. Please remember, I will be looking for multiple commits. I would suggest using the lab number in your commit message as a prefix so you can also review the history throughout the semester, for example:

- Lab 3 - Initial Commit
- Lab 3 - POJO implementation and HTML form
- Lab 3 - Servlet MVC Controller
- Lab 3 - Fixed the Servlet MVC Controller RequestDispatcher logic
- Lab 3 - Finishing touches (comments, javadoc, code style)

Project Requirements

Implement the following within your uid-fp project:

Make sure you are following standard Java naming conventions. Please review the following if you are not sure what those naming conventions are:

- Java 8 Pocket Guide by Patricia Liguori, Robert Liguori - Chapter 1. Naming Conventions
- Code Conventions for the Java Programming Language - 9 Naming Conventions (dated, but still correct)

1. Use your POJO from Lab 2 for the following requirements. Enhance it by adding or refining fields as needed.
2. Add an HTML form and a Servlet to your project.
    1. The form should contain fields necessary to create a new instance of your entity.
    2. The form should POST to the Servlet.
    3. The form should be contained within a JSP page
3. Process the form submission in your Servlet by overriding the **doPost** method as demonstrated in class
    1. Get the parameters of the form in your Servlet
    2. Build and validate your entity using a Validator (obtain the Validator through Resource Injection as discussed in class)
    3. If the user input passes validation:
        1. Set the validated entity as a request attribute
        2. **Forward** (using RequestDispatcher) the user to a confirmation view in the WEB-INF/views directory of your project. You may need to create this directory.
        3. As demonstrated in class, the confirmation view should display fields of your entity using Expression Language
        4. Document this with a description and screenshots. The screenshots should display your form (before) and your confirmation view (after).
    4. If the user input does not pass validation
        1. **Forward** (using RequestDispatcher) the user back to the form.
        2. Display appropriate error messages so they can fix their mistake(s).
        3. Re-populate the form with data from their prior submission.

        4. Document this with a description and screenshots. The screenshots should display your form with bad input (before) and your form with error messages (after).
4. Write a brief summary paragraph to document:
    1. Your understanding of the difference between the forward and redirect operations.
    2. How would you be validating user submissions without the Bean Validation API standard?
    3. How do you think this approach would scale to a real application with 100's of entities?
    4. Why didn't we need to include any additional dependencies (i.e. Bean Validation, JDBC) in this project?
5. Submit to Canvas or Beacon
    1. Right your **uid-fp** project and select "Clean"
    2. Go to your NetBeans Projects directory. Create a zip file of the **uid-fp** folder and submit it to the Canvas or Beacon assignment.

Graduate/Undergradaute

Graduate students also need to implement database persistence. Undergraduate students can do this for extra credit (5 points).

1. Using JDBC, connect your webapp to the database.
    1. Make sure you add the MySQL JDBC driver to your pom.xml. Refer back to Lab 2 (or the lecture) if you don't remember how to do this.
    2. Create a Payara JDBC Resource for connectivity to your itmd4515 database. There are different way to do this:
        1. Adding <data-source> configuration to web.xml
        2. Using a @DataSourceDefinition annotation
        3. Configuring a Payara JDBC Connection Pool and JDBC Resource
    3. Ensure that Payara has access to the JDBC driver for MySQL **at runtime**. Which of the above options require us to do "extra steps" as system or application administrators to handle this?
2. In your Servlet, obtain a DataSource using @Resource Injection
3. **Only on successful validation** write the submitted entity to the database using JDBC.