Searching for a Specific User and Updating the User Information..

Write –Up

1. Review the project structure and dependencies:

   - The project follows the Maven directory structure convention.

   - It uses Spring Boot, Spring Data JPA, and Spring MVC dependencies.

   - MySQL is used as the database, and the necessary configurations are provided in the `application.properties` file.

   - Additional dependencies include Lombok for reducing boilerplate code and JSTL for JSP rendering.

2. Set up the database:

   - Create a MySQL database named "users."

   - Run the provided MySQL queries to create the "user" table and insert sample user data.

3. Review the main components:

   - `User` class: Represents the User entity with fields such as `id`, `name`, `email`, and `password`.

   - `UserRepository` interface: Extends the Spring Data `CrudRepository` interface to perform CRUD operations on the `User` entity.

   - `UserService` class: Contains methods for retrieving, updating, and saving user data. It utilizes the `UserRepository` for database operations.

   - `UserController` class: Handles user-related HTTP requests, such as displaying all users, searching for a user by ID, and updating user information. It uses the `UserService` for data manipulation.

4. Understand the views:

   - Several JSP files are used for rendering HTML views:

     - `index.jsp`: The main page where users can enter an ID and search for a specific user.

     - `users.jsp`: Displays a table of all users retrieved from the database.

     - `search.jsp`: Displays details of a searched user and allows updating their information.

     - `update.jsp`: Confirms a successful update and displays the updated user's details.

- `error.jsp`: Handles error scenarios and provides a basic error message.

5. Review additional classes:

   - `UserExceptionController`: Handles the exception thrown when a user is not found and returns an appropriate HTTP response.

   - `AppErrorController`: Handles generic error paths and can be customized for logging and error handling.

6. Build and run the application:

   - Make sure you have Java 8 and Maven installed.

   - Navigate to the project's root directory.

   - Run `mvn clean install` to build the project and resolve dependencies.

   - Run `mvn spring-boot:run` to start the application.

   - Access the application in a web browser at `http://localhost:8080`.

7. Test the application:

   - Open the application in a web browser.

   - Enter an ID in the search form on the homepage and submit the form.

   - On the user details page, modify the user's information and click the update button.

   - Verify that the user information is updated and displayed on the update page.

   - Explore other functionalities, such as viewing all users and handling error scenarios.