Handling User Authentication WriteUp

Workflow for this Project

1. The project is a user authentication system developed using Spring Boot.

2. The main class is `AuthenticationApplication`, which is annotated with `@SpringBootApplication`. It serves as the entry point of the application.

3. The `AuthenticationApplication` class imports various components using the `@Import` annotation. These components include the `AuthenticationController`, `UserNotFoundException`, `AuthenticationService`, and `User`.

4. The `User` class represents the user entity and is annotated with JPA annotations to define its mapping to the database table.

5. The `AuthenticationController` class is a Spring MVC controller responsible for handling requests related to user authentication.

6. It has methods such as `showGreeting`, `showLogin`, and `authenticateUser`, which are mapped to specific URLs using the `@GetMapping` and `@PostMapping` annotations.

7. The `AuthenticationController` class uses the `AuthenticationService` to perform user authentication and logging operations.

8. The `AuthenticationService` class is a service component that interacts with the database through the `AuthenticationRepository`.

9. It provides methods such as `GetUserByName` to retrieve a user by name and `isValidPassword` to validate a user's password.

10. The `AuthenticationRepository` interface extends the Spring Data `CrudRepository` and defines additional query methods to access the user data in the database.

11. The project includes several test classes (`AuthenticationApplicationTests`, `AuthenticationCodeTests`, `AuthenticationWebTesting`) for testing different aspects of the application.

12. The `authentication.jsp` files are JavaServer Pages (JSP) templates used for rendering the login form, greeting page, success page, and failure page.

13. The `application.properties` file contains configuration settings for the Spring Boot application, including database connection details, view resolution, and logging levels.

Workflow Summary:

1. Start the Spring Boot application by running the `AuthenticationApplication` class.

2. Access the landing page ("/") and see the greeting message.

3. Navigate to the login page ("/Auth") and enter the username and password.

4. Submit the login form, which triggers the `authenticateUser` method in the `AuthenticationController`.

5. The controller uses the `AuthenticationService` to retrieve the user by the provided username.

6. The user's login attempt is logged, and the password is validated against the stored password.

7. Depending on the password validation result, the controller returns either "success" or "failure" as the path.

8. If successful, the user is redirected to the success page, otherwise to the failure page.

9. The user can attempt login again from the failure page or logout and return to the landing page.

10. The project includes various test classes to ensure the functionality of the application.

11. The JSP templates are used to render the appropriate views based on the requested URLs.