Authentication Exercise (Java)

In this exercise, you'll continue to work on to the auctions server and client applications. Now that the API and client have been built, you'll add authentication and authorization to them.

Feel free to refer to the student book or tutorial for guidance.

Step One: Open client and server applications

Before you begin, import both the client and server starter code into Eclipse using the "Import Existing Maven Projects" feature. Review both projects. The code should look familiar to you as it's a continuation of previous exercises.

Security

Just like the tutorial and the lecture for this day, the server application contains a package called `com.techelevator.auctions.security` that contains all of the security-related code. You won't need to modify any code in this package.

Tests

There are test classes in both the client and server application.

Client application

In the client application, there are two test classes located in `/src/test/java/com/techelevator/services/`:

- `AuthenticationServiceTests.java`
- `AuctionServiceTests.java`

Each class has a single test. The two tests fail before you implement any changes. To complete this exercise, all tests must pass.

Server application

In the server application, there is one test class located in `/src/test/java/com/techelevator/auctions/controller` with a total of six tests:

- `AuctionControllerTests.java`

These tests fail before you implement any changes. To complete this exercise, all tests must pass.

Step Two: Complete the login method

In the client project, open `AuthenticationService.java`, and locate the `login()` method.

Most of the functionality has been created for you, but you have to create the request. You need to send a `POST` request to the login endpoint `/login` with an object that has `username` and `password` fields and the

appropriate header.

After you complete this step, the `step2_loginMethod` test in `AuthenticationServiceTests` passes.

> Note: If you're having trouble with this, you can go back to the tutorial and see what the `login()` method should look like.

At any point during the exercise, you can test the client application by logging in with the following credentials as the username and password:

- `user/password`: Role: USER
- `creator/password`: Role: CREATOR
- `admin/admin`: Role: ADMIN

Step Three: List all auctions

In the client project, open `AuctionService.java`, and locate the `getAll()` method. Again, most of the functionality has been created for you, but you have to create the request.

First, send a `GET` request to the server endpoint `/auctions`. This request must contain the authorization header needed to verify the identify of the request.

After you complete this step, the `step3_getAllAuctions` test in `AuctionServiceTests` passes.

Step Four: Add authentication to controller methods

In the server project, open `AuctionController.java`. All methods must require authentication except `list()`, the method that responds to `/auctions`. See if you can accomplish this by only adding two lines to the class.

After you complete this step, the `step4_AllMethods_ExpectUnauthorized` and `step4_list_ExpectOk` tests in `AuctionControllerTests` pass.

Step Five: Add authorization roles

In `AuctionController.java`, add the following authorization rules:

- `create()`: allow `CREATOR` and `ADMIN` roles
- `update()`: allow `CREATOR` and `ADMIN` roles
- `delete()`: allow `ADMIN` role

After you complete this step, the `step5_CreateMethod`, `step5_UpdateMethod`, and `step5_DeleteMethod` tests in `AuctionControllerTests` pass.

> Note: If you need to review checking for multiple roles, go back to the student book and re-read the Require roles section.

Step Five: Return user identity

In `AuctionController.java`, locate the `whoAmI()` method. Instead of returning an empty string, return the logged in user's name.

After you complete this step, the `step5_WhoAmI` test in `AuctionControllerTests` passes.

If you followed the instructions correctly, all tests now pass.