DAT152: Oblig #2 – Web Frameworks and Services with JWT Token and OAuth2

In this obligatory task, you will complete the implementation of the RESTful APIs for the eLibrary App and secure the endpoints.

Task #0: Lab #3

- The oblig #2 builds on Lab #3 with advanced features and functionalities. Therefore, you are expected to have a working solution for the Book and Author REST APIs from Lab #3 (see '***dat152-exercise-F17.pdf***' on Canvas)

Task #1: RESTful API Services
Project: *library-spring-ws-rest-oblig.zip*

- Support for users to order (borrow) and return a book. You will complete the "TODOs" in the following classes and methods:
  - UserService
    - updateUser
    - deleteUser
    - createOrdersForUser
  - UserController
    - getUser
    - createUserOrders
  - OrderService
    - deleteOrder
    - findByExpiryDate
    - updateOrder
  - OrderController
    - getAllBorrowOrders
    - returnBookOrder
- Produce a list of all books borrowed by a user.

Task #2: HATEOAS, Pagination and Filtering

- Provide query support to list all 'orders' that expire by the specified date (date format: yyyy-MM-dd).
- Provide support to paginate the list of orders (e.g., Pageable).
- For the above tasks, you will complete the "TODOs" in the following classes and methods:
  - OrderService
    - findByExpiryDate
  - OrderController
    - getAllBorrowOrders
- Provide support with links to other endpoints and actions possible for the resource.
  - Add links to orders in createUserOrders

Task #3: Securing resource endpoints with JWT Access tokens from resource + auth server
Project: *library-spring-ws-rest-security-oblig.zip*

- First, obtain access tokens for a super_admin role (berit) and user role (robert) by running the TestSignupLogin.java Junit test. The access_token can be copied from the console of your IDE. You can also start the application and use Postman to send a POST request with json body {"email":"berit@gmail.com", "password":"berit_pwd"} to http://localhost:8090/elibrary/api/v1/auth/login
    - You will receive a response with the access_token.
    - Copy the access_token and replace the 'super.admin.token' and 'user.token' in the "application.properties" with these new values. To make your testing easier, the tokens are configured to expire in 5 days. When they expire, you need to request for new tokens and replace the old ones.

TODOs:

- Copy your solutions from Task #2 to this project. Do not replace the entire file as there might be new methods in the project. So, only copy method by method.
- Secure the API endpoints you created in Task #0 and Task #1.
    - @PreAuthorize("has Authority('…')")
- Provide support for super admin to assign and revoke roles (SUPER_ADMIN, ADMIN, USER) from specific users. To achieve this task:
    - You will update the two methods to assign and revoke roles for specific users in the AdminUserController and AdminUserService classes.
    - The API support should use query parameter to achieve this task.
        - /users/{id}?role=
    - Update the method for "Assign role": updateUserRole(id, role)
    - Update the method for "Revoke role": deleteUserRole(id, role)
    - Annotate the controller methods with the correct 'SUPER_ADMIN' authority (role)
- Request to your Admin REST APIs will look like:
- **Assign role:**
    - **PUT** http://localhost:8090/elibrary/api/v1/admin/users/1?role=ADMIN
- **Revoke role:**
    - **DELETE** http://localhost:8090/elibrary/api/v1/admin/users/1?role=USER
- Test your solutions.

Task #4: Securing resource endpoints with JWT Access tokens from 3rd party OAuth2 Server (Optional)

Project: *library-spring-ws-rest-security-oauth-oblig.zip*

- Follow the instruction guide to setup and run the Keycloak IdP server. Then, create the app, users, and roles.
- Obtain the access_token for the 'super_admin' and normal 'user' by sending a POST request to the keycloak token endpoint:
    - curl -X POST http://localhost:8080/realms/SpringBootKeycloak/protocol/openid-connect/token --data 'grant_type=password&client_id=elibrary-rest-api&username=super_user&password=berit_pwd'
- You will receive a response with the access_token.
    - Copy the access_token and replace the 'super.admin.token' and 'user.token' in the "application.properties" with these new values. When they expire, you need to request for new tokens and replace the old ones.

    - Update the application.properties with these tokens.
- Copy your solutions from Task #3 and secure the endpoints.
    - @PreAuthorize("has Authority('…')")
    - Note that Spring appends "ROLE_" to the authority, therefore for a role ADMIN in the JWT, you will annotate as @PreAuthorize("has Authority('ROLE_ADMIN')")
- Test your solutions.

Preambles: Download and unzip *each project*. Then, import the maven project into your preferred IDE.
Some ways to run your project:

1. Run the project from a command prompt (e.g., Mac Terminal). Navigate to the root folder of your project and run the maven command: ./mvnw spring-boot:run
2. From your IDE: Right click on 'LibraryApplication.java' in the "no.hvl.dat152.rest.ws.main" package and "Run As" Java Application.

Testing
JUnit tests are provided, and you can run them in your IDE or from a terminal.
Warning!

- Ensure you run each Junit test independently.
- Watch out for the specific HttpStatus codes in the Junit tests and ensure that they correspond to what you are returning in the controller methods.

Use Postman in addition, to support your api testing.