Adam Štrobl, Matyáš Urban

Student Information System (SIS)

Project Highlights

- **Spring Boot Framework**: Utilizes Spring Boot for rapid, convention-over-configuration centric framework, reducing development time and increasing efficiency.
- **Security Implementation**: Employs Spring Security in config.SecurityConfig.java for robust authentication and authorization, ensuring secure access to the system's resources.
- **Domain Models**: Defines complex relationships and functionalities in model package, with classes like Student, Teacher, Course, reflecting the real-world educational structures.
- **Data Access Layer**: Uses Spring Data JPA repositories in repository package for optimized data handling, providing an abstraction layer over database interactions.
- **RESTful Controllers**: Implements RESTful controllers in rest package, such as AdminController, StudentController, for handling API requests and responses.
- **Business Logic**: Encapsulates core business logic within services in service package, like TeacherService, StudentService, separating concerns and enhancing maintainability.
- **Utilities and Constants**: Includes helper classes and constants in utils package for code reusability and maintenance, aiding in standardized operations across the application.

Course Feedback

We found setting up Spring Security challenging but educational. The graded tasks during practicals were engaging, and we're impressed with how much Spring does behind the scenes, greatly simplifying our development process.

Running the Application

1. Open Project in IntelliJ IDEA:
   Open IntelliJ IDEA.
   Select "Open" and navigate to your project directory.
   Choose the pom.xml file of your Maven project and open it. IntelliJ IDEA will automatically set up the project based on Maven configuration.
2. Run the SIS Application:
   After Maven dependencies are resolved, locate SisApplication.java in your project directory.
   Right-click on SisApplication.java and select "Run".
   The application should start and be listening on a defined port (usually 8080). Check the console log to ensure it's running without errors.
3. Trying the Application with Postman:
   Open Postman.

Import postmanHttpQuerries1.postman_collection.json and postmanHttpQuerries2.postman_collection.json from the root repository of your project.

Run the imported queries sequentially. This is important to maintain the flow and dependencies of your data.

4. Resolving Database Conflicts:

   If you encounter database conflicts, open the console of your data source in IntelliJ IDEA.

   Run the following SQL queries to clear the data:

```
sql
DELETE FROM ENROLLMENT WHERE id > 0;
DELETE FROM PARALLEL_STUDENT WHERE PARALLEL_STUDENT.PARALLEL_ID > 0;
DELETE FROM PARALLEL WHERE id > 0;
DELETE FROM CLASSROOM WHERE id > 0;
DELETE FROM SEMESTER WHERE id > 0;
DELETE FROM COURSE WHERE id > 0;
DELETE FROM PERSON WHERE id > 0;
ALTER SEQUENCE CLASSROOM_SEQ RESTART WITH 1;
ALTER SEQUENCE COURSE_SEQ RESTART WITH 1;
ALTER SEQUENCE ENROLLMENT_SEQ RESTART WITH 1;
ALTER SEQUENCE PARALLEL_SEQ RESTART WITH 1;
ALTER SEQUENCE PERSON_SEQ RESTART WITH 1;
ALTER SEQUENCE SEMESTER_SEQ RESTART WITH 1;
```

After clearing the data, you can start running the Postman queries again.

5. Database Credentials:

   Ensure the correct database credentials are set in /src/main/resources/application.properties.

   Double-check the database URL, username, and password for correctness.