Introduction

This project is a Library Management System built with Spring Boot. It allows you to manage books and patrons, track borrowing records, and provides authentication and caching features.

Prerequisites

- Java 17 or higher.
- Maven 3.6.3 or higher.
- An IDE of your choice (IntelliJ IDEA, Eclipse, VS Code, etc.)
- Postman or any other API testing tool.

Setup and Installation

1. Clone the repository
    a. git clone <https://github.com/kkarimwahba/library>
    cd library
2. Build The project
    a. mvn clean install
3. Setup the database
    a. By default, the application uses an H2 in-memory database. No additional setup is required for H2.
    b. If you prefer to use another database (e.g., MySQL, PostgreSQL), update the application.properties file with the appropriate configurations.

Running the application

1. Start the springboot application
    a. mvn spring-boot:run
2. Access the H2 Console
    a. URL: http://localhost:8080/h2-console
    b. JDBC URL: jdbc:h2:mem:testdb
    c. Username: sa
    d. Password: password

API Endpoints

The following are available API endpoints to manage books, patrons and borrowing. You can test these APIs by postman after running the application.

BOOK

- Add Book

  POST /api/books

  Content-Type: application/json

  {

  "title": "Oliver Twist",

  "authorId": 1,

  "categoryId":1,

  "isbn": 1234567890 ,

  "publicationYear": 2020

  }

- Get Books
  - GET /api/books

- Get Book by ID
  - GET /api/books/{id}
- Delete Book by ID
  - DELETE /api/books/{id}
- Update Book by ID
  - PUT /api/books/{id}
    Content-Type: application/json
    {
    "title": "Updated Title",
    "authorId": 2,
    "categoryId":1,
    "isbn": 0987654321,
    "publicationYear":2020
    }

Patrons

- Add Patron
  - POST /api/patrons
    Content-Type: application/json
    {
    "name": "Patron Name",
    "email": "patron@example.com" ,
    "phone":"010xxxxxx9",
    "address":"NewCairo"
    }
- Get All Patrons
  - GET /api/patrons
- Get Patron by ID
  - GET /api/patrons/{id}
- Delete patron by ID
  - DELETE /api/patrons/{id}
- Update Patron by ID
  - PUT /api/patrons/{id}
    Content-Type: application/json
    {
    "name": "Updated Name",
    "email": updated@example.com,
    "phone":"010xxxxxx7",
    "address":"NewCairo"

    }

Borrowings

- Borrow Book
  - POST /api/borrowing/borrow/{bookId}/patron/{patronId}
- Return a Book
  - PUT /api/borrowing/return/{bookId}/patron/{patronId}
- Get All Borrowings
  - GET /api/borrowings

Testing

Unit tests are written to validate the functionality of the API endpoints. To run the tests:

1. **Run the tests using Maven**:
    - mvn test
2. **View test reports**:
    - Test reports can be found in the target/surefire-reports directory.

Additional Notes

- **Caching**: The project uses Spring's caching mechanism to cache frequently accessed data.
- **Aspect-Oriented Programming (AOP)**: Logging aspects are implemented to log method calls, exceptions, and performance metrics.

Conclusion

This Library Management System project, built using Spring Boot, provides a robust framework for managing books, patrons, and borrowing records. With features such as basic authentication, caching, and logging through Aspect-Oriented Programming (AOP), it offers a secure and efficient solution for library operations.

The clear documentation provided ensures that you can easily set up, run, and interact with the application. Unit tests validate the functionality of the API endpoints, ensuring reliability and correctness. The implementation of these features aims to enhance the overall performance and maintainability of the system.

By following the guidelines and examples given, you should be able to extend and adapt this project to fit the specific needs of any library management scenario. This project serves as a solid foundation for building more complex systems, integrating additional features, and ensuring a seamless library management experience.