RESTful Bookstore API Report

Introduction

The objective of this project was to create a backend API for a bookstore to manage its inventory of books and authors. The API is built following RESTful principles, providing a clear and standardized way for a client application to interact with the server. This report details the tools used and the steps involved in building this robust and efficient backend system.

Abstract

This project successfully developed a RESTful API using Java and Spring Boot. The core of the application provides CRUD (Create, Read, Update, Delete) functionality for Book and Author entities. The data is persisted in an in-memory H2 Database for development ease, and API endpoints were thoroughly tested using Postman. The project also includes features like filtering, pagination, and sorting to enhance data retrieval capabilities, with API documentation provided by Swagger. The final deliverables include the source code, a Postman collection for testing, and the Swagger UI link for documentation.

Tools Used

- **Java:** The primary programming language used to build the entire backend application.
- **Spring Boot:** A powerful framework that provides a rapid application development environment with minimal configuration. It managed dependencies, simplified server setup, and enabled the creation of RESTful web services.
- **H2 Database:** A lightweight, in-memory relational database used during development. It's a convenient choice for testing as it doesn't require a separate installation and clears the data upon application restart.
- **Postman:** An indispensable API client used for testing all API endpoints. It allowed for sending various HTTP requests (GET, POST, PUT, DELETE) with custom headers and JSON payloads, ensuring the API behaved as expected.
- **Swagger (Springdoc OpenAPI):** A tool for automatically generating interactive API documentation from the code. It provided a user-friendly interface to explore and test API endpoints directly in a web browser, making the API easier for developers to understand and use.

Steps Involved in Building the Project

- **Project Initialization:** The project was initiated as a Spring Boot application. The necessary dependencies for Web (spring-boot-starter-web), JPA (spring-boot-starter-data-jpa), and the H2 Database (com.h2database:h2) were added to the project's build file.
- **Entity Modeling:** The Book and Author entities were defined as Java classes. Appropriate annotations (@Entity, @Id, @GeneratedValue) were used to map these

classes to database tables and establish the one-to-many relationship between an Author and their Books.

- **CRUD Implementation:** REST controllers were implemented using the @RestController annotation. Methods were created for each CRUD operation, such as @GetMapping for retrieving data, @PostMapping for creating new resources, @PutMapping for updating, and @DeleteMapping for deleting.
- **Testing with Postman:** Once the controllers were set up, a Postman collection was created to test each endpoint. This included testing successful requests as well as different error scenarios to ensure the API's robustness.
- **Advanced Features:** To improve the API's functionality, features for filtering, pagination, and sorting were added to the retrieval endpoints. This allowed clients to request subsets of data, improving performance and usability for large datasets.
- **API Documentation:** Swagger was integrated by adding the springdoc-openapi-starter-webmvc-ui dependency. This automatically generated the API documentation at the /swagger-ui.html endpoint, which provided a comprehensive guide to the API's structure and usage.

Conclusion

The successful completion of the RESTful Bookstore API project demonstrates proficiency in modern backend development practices. By utilizing a powerful framework like Spring Boot and essential tools such as Postman and Swagger, the project was able to efficiently deliver a functional and well-documented API. The project's deliverables—including the source code, a Postman collection, and the Swagger UI link—provide a complete package for understanding, testing, and interacting with the API.