Project Report

“ RESTful Bookstore API “

Introduction

The rapid advancement of technology and the growing demand for digital solutions in the publishing industry necessitate the implementation of robust and efficient bookstore management systems. This project focuses on creating a RESTful Bookstore API with Java, Spring Boot, and Hibernate (JPA) to meet this need. Using industry-standard tools and frameworks, the project was completed as part of an internship program to give participants practical experience in database administration, API design, and backend development.

Abstract

The primary objective of the Bookstore REST API project is to make managing books and authors in an online bookstore more efficient. The system's design ensures data integrity, validation, and ease of use while carrying out CRUD (Create, Read, Update, Delete) operations on books and authors. By dividing issues among controllers, services, and repository layers, the solution highlights a clear, tiered architecture. Swagger/OpenAPI is used to document the API, making integration and testing simple. The project provides an excellent opportunity to learn about the foundations of enterprise backend development and best practices for software engineering.

Tools Used

- **Programming Language:** Java 17
- **Framework:** Spring Boot
- **ORM:** Hibernate (JPA)
- **Database**: H2 (in memory, for development/testing)
- **API Documentation:** Swagger/OpenAPI
- **Build Tool:** Maven
- **Validation:** Jakarta Bean Validation (JSR 380)
- **IDE:** IntelliJ IDEA / Eclipse (for development)
- **Version Control:** Git

Steps Involved in Building the Project

a) Project Setup

Initialized a Spring Boot project using Maven and added dependencies for Spring Web, Spring Data JPA, H2 Database, Hibernate, Swagger, and Validation. Configured `application.Properties` for database connectivity and enabled H2 console and Swagger UI.

b) Designing the Data Model

Defined two primary entities: Author and Book. Used JPA annotations to establish a many-To-one relationship (each book is linked to one author).

c) Implementing Data Access Layer

Created repository interfaces for `Author` and `Book` using Spring Data JPA, enabling easy CRUD operations without boilerplate SQL code.

d) Developing Service Layer

Implemented service classes for business logic, handling entity validation, and ensuring referential integrity (e.g., book references a valid author).

e) Building REST Controllers

Developed RESTful endpoints for both books and authors to handle create, read, update, and delete operations. Used DTOs (Data Transfer Objects) for request/response mapping, keeping the API decoupled from internal entity structures.

f) Input Validation

Applied Jakarta Bean Validation annotations to DTOs to enforce constraints like nonempty fields and valid pricing. Handled exceptions to return meaningful error messages to API clients.

g) API Documentation & Testing

Integrated Swagger/OpenAPI to autogenerate interactive API documentation, making endpoints easy to test and share. Used the H2 console for direct database inspection during development.

5. Conclusion

The Bookstore REST API project gave me valuable experience with modern backend development principles like RESTful design, ORM mapping, layered architecture, and API documentation. The project not only met its goal of efficiently managing books and authors, but it also helped to develop practical skills such as input validation, exception handling, and the use of industry standard frameworks. This experience has provided a solid foundation for my future work in software development and backend engineering.