Project: RESTful Bookstore API

Introduction:

The RESTful Bookstore API is a backend application built using Spring Boot that provides CRUD operations for managing authors and books. It is designed to follow REST principles and provides endpoints for creating, retrieving, updating, deleting, and filtering data with pagination and sorting support. This project aims to server as a foundational backend for a digital bookstore or any library management system.

Abstract:

This projects implements a REST API that manages authors and their associated books using Spring Boot, JPA, and an in-memory H2 database. Data Transfer Objects (DTOs) are used to separate internal data from exposed endpoints. The application includes validation, exception handling, and API documentation using Swagger UI. Data is preloaded at startup for ease of testing. This project reinforces understanding of REST architecture, service layers, data mapping, and Java backend development best practices.

Tools & Technologies Used:

Tools/Tech	Purpose
Java 21	Primary Programming Language
Spring Boot	Framework for building RESTful APIs
Spring Data JPA	Object Relational Mapping (ORM) and database interactions
H2 Database	In-memory DB for rapid development/testing
Lombok	Boilerplate code reduction
Swagger UI	Interactive API documentation
Postman/ cURL	Testing endpoints
Maven	Build and Dependency Management

Steps Involved in Building the Project:

- **Project Initialization:** Created Spring Boot project with JPA, Lombok, Validation, and H2 dependencies.
- **Entity Design:** Created Author and Book entities with a one-to-many relationship.
- **DTO Creation:** Separated DTOs into AuthorDto, BookDto, and their corresponding detail and create/update variants.
- **Service Layer Implementation:** Developed service interfaces and implementations to handle business logic.
- **Controller Layer:** Created REST controllers for authors and books with standard CRUD endpoints.
- **Validation:** Used @Valid, @NotBlank, @Min, etc., for input validation on DTOs.

- **Exception Handling:** Added custom ResourceNotFoundException,Validation Hndler and a global exception handler.
- **Filtering, Pagination & Sorting:** Implemented dynamic filtering on books with Spring Data JPA Specifications and pageable support.
- **Preloaded Sample Data:** Used CommandLineRunner to insert authors and books at application startup.
- **Testing via Swagger UI & Postman:** Verified all endpoints via Swagger and Postman with multiple test scenarios.

Features

- CRUD operations for **Authors** and **Books**
- Data validation using ***jakarta.validation***
- Filtering, sorting, and pagination support for books
- DTO-based request/response handling
- Global exception handling
- Preloaded data set on startup
- Swagger UI documentation
- H2 console for in-memory database visualization

REST API Endpoints

Books

- **GET** ***/api/books*** – List all books (supports pagination, filtering, sorting)
- **GET** ***/api/books/{id***} – Get book details by ID
- **POST** ***/api/books*** – Create a new book
- **PUT** ***/api/books/{id}*** – Update an existing book
- **DELETE** ***/api/books/{id}*** – Delete a book
- **GET** ***/api/books/filter*** **-** Filter , Pagenate, Sort books based on Title, Genre, PublicationYear, Author etc.

Authors

- **GET** ***/api/authors*** – List all authors
- **GET** ***/api/authors/{id}*** – Get author details with books
- **POST** ***/api/authors*** – Create a new author
- **PUT** ***/api/authors/{id}*** – Update an existing author
- **DELETE** ***/api/authors/{id}*** – Delete an author

Conclusion:

This project provided a hands-on understanding of how to build a robust RESTful API using Spring Boot. The layered architecture, validation, and DTO usage demonstrate clean coding principles. While basic in scope, it can be extended into a full-stack application or evolved into microservices. It lays the foundation for real-world applications like library systems, online bookstores, or knowledge repositories.