SPRING BOOT 3 REST API JPA DATA

JAVA ACADEMY - XIDERAL

SEPTEMBER 6, 2024

AUTHOR: REGINA RODRIGUEZ CAMPO GARRIDO

Spring Boot 3 REST API JPA DATA

Introduction

Spring Boot is a tool that simplifies the creation of Java applications by automatically configuring many aspects. JPA is a specification for managing data persistence in databases. To make working with JPA even easier, we use Spring Data JPA, which extends JPA's capabilities by providing a simpler and more efficient way to interact with the database.

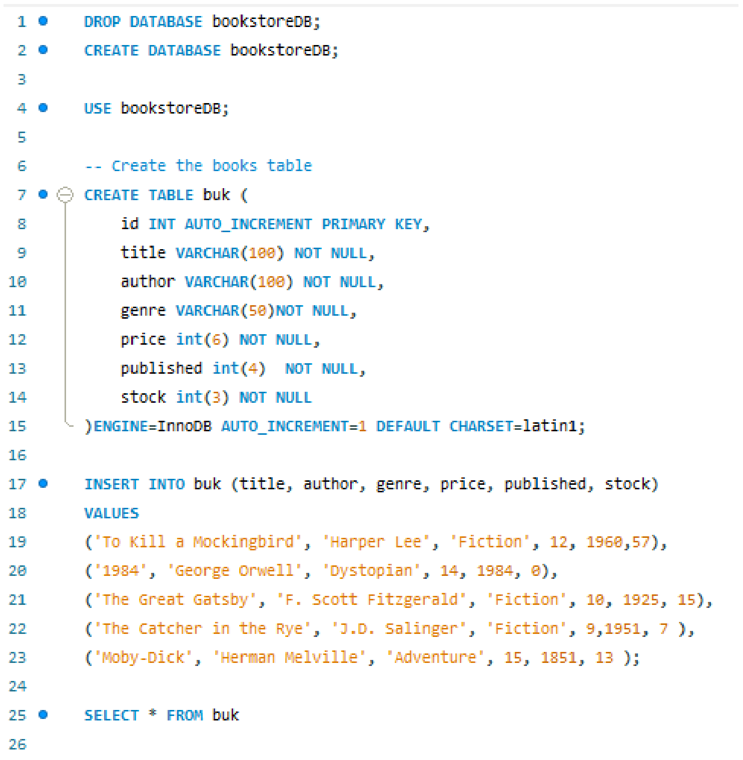
Spring Data JPA simplifies repository implementation by allowing the creation of interfaces for data access without having to write SQL or JPQL queries manually. Additionally, it automatically handles common operations such as storing, retrieving, and deleting entities, which reduces boilerplate code and enhances project maintainability.

In the following project, using Spring Boot and Spring Data JPA, we created a library system that will allow us to:

1. Show all the books we have.
2. Add books.
3. Delete books.
4. Search for books by ID.
5. Count the number of books we have by category.
6. Show whether the book is in stock or not

Database Schema

Create a table with the name "buk".

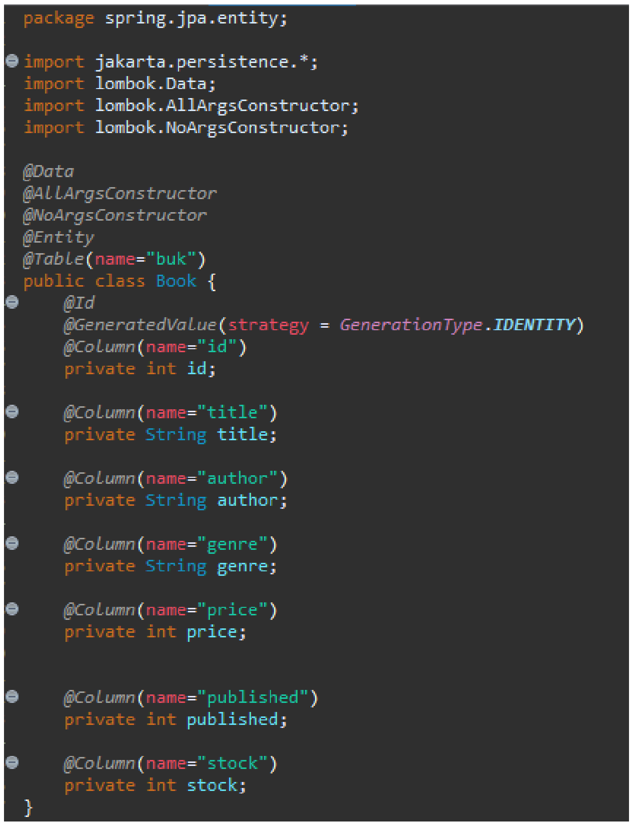


```sql
DROP DATABASE bookstoreDB;
CREATE DATABASE bookstoreDB;

USE bookstoreDB;

-- Create the books table
CREATE TABLE buk (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    author VARCHAR(100) NOT NULL,
    genre VARCHAR(50)NOT NULL,
    price int(6) NOT NULL,
    published int(4)  NOT NULL,
    stock int(3) NOT NULL
)ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;

INSERT INTO buk (title, author, genre, price, published, stock)
VALUES
('To Kill a Mockingbird', 'Harper Lee', 'Fiction', 12, 1960,57),
('1984', 'George Orwell', 'Dystopian', 14, 1984, 0),
('The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 10, 1925, 15),
('The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 9,1951, 7 ),
('Moby-Dick', 'Herman Melville', 'Adventure', 15, 1851, 13 );

SELECT * FROM buk

```

Application

Entity

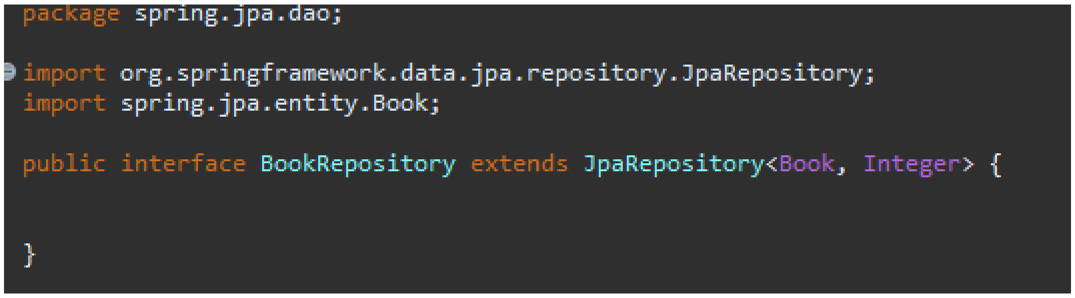
The Book class represents the Book entity. It uses JPA annotations to map the class to a table in the database and Lombok annotations to reduce code.



```java
package spring.jpa.entity;

import jakarta.persistence.*;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name="buk")
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="title")
    private String title;

    @Column(name="author")
    private String author;

    @Column(name="genre")
    private String genre;

    @Column(name="price")
    private int price;


    @Column(name="published")
    private int published;

    @Column(name="stock")
    private int stock;
}
```

Repository

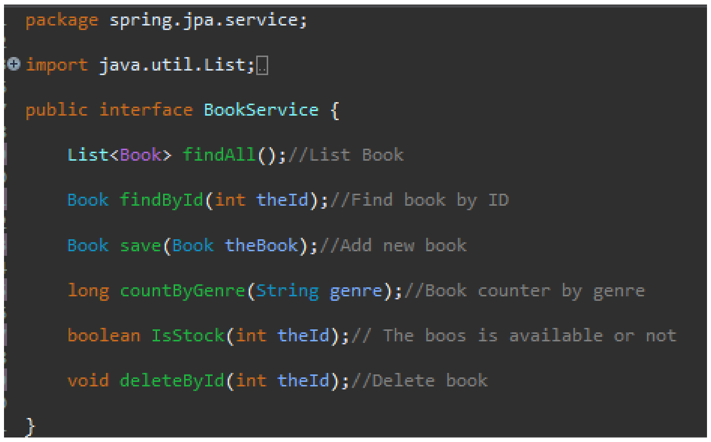
BookRepository extends JpaRepository



```java
package spring.jpa.dao;

import org.springframework.data.jpa.repository.JpaRepository;
import spring.jpa.entity.Book;

public interface BookRepository extends JpaRepository<Book, Integer> {


}
```

Service Interface

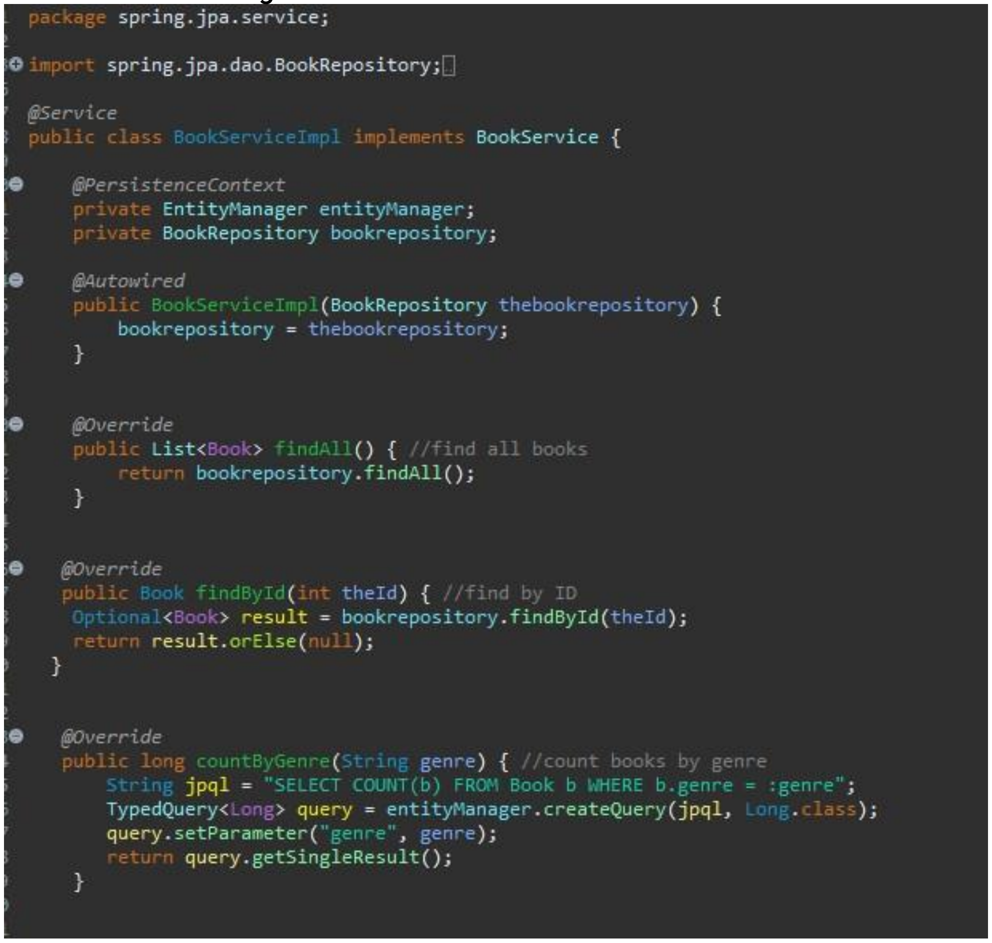
The BookService interface defines the business logic methods for managing books.



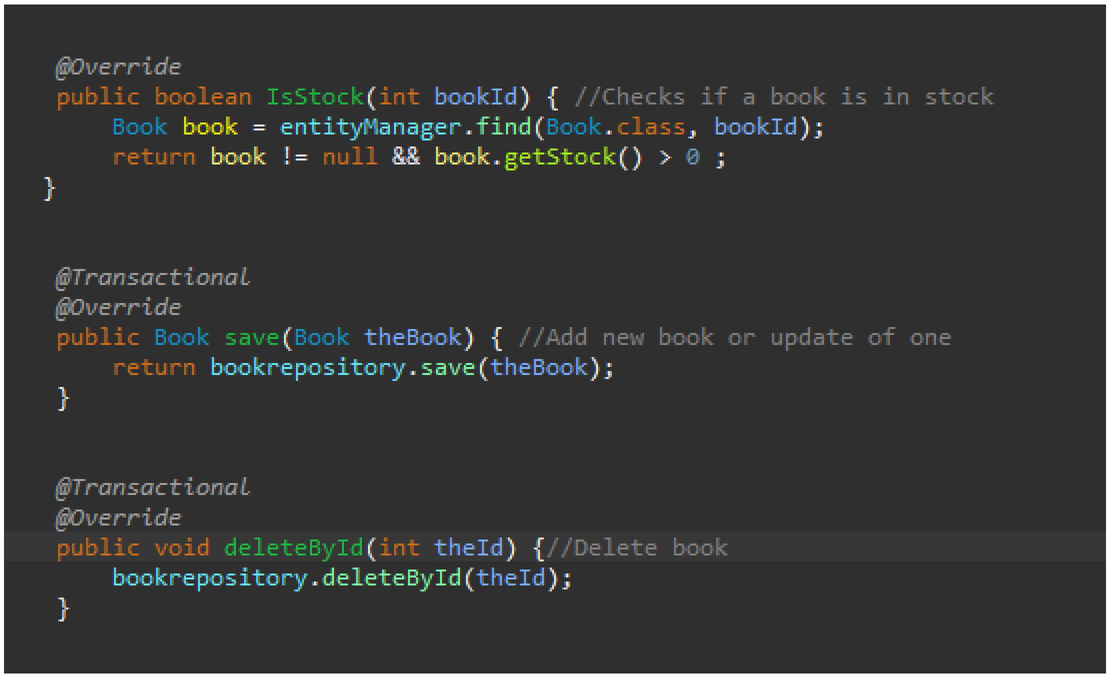
```
package spring.jpa.service;

import java.util.List;

public interface BookService {

    List<Book> findAll();//List Book

    Book findById(int theId);//Find book by ID

    Book save(Book theBook);//Add new book

    long countByGenre(String genre);//Book counter by genre

    boolean IsStock(int theId);// The boos is available or not

    void deleteById(int theId);//Delete book

}
```

Service Implementation

The BookServiceImpl class implements the BookService interface. Now here we add all the logic that we want our methods to have.



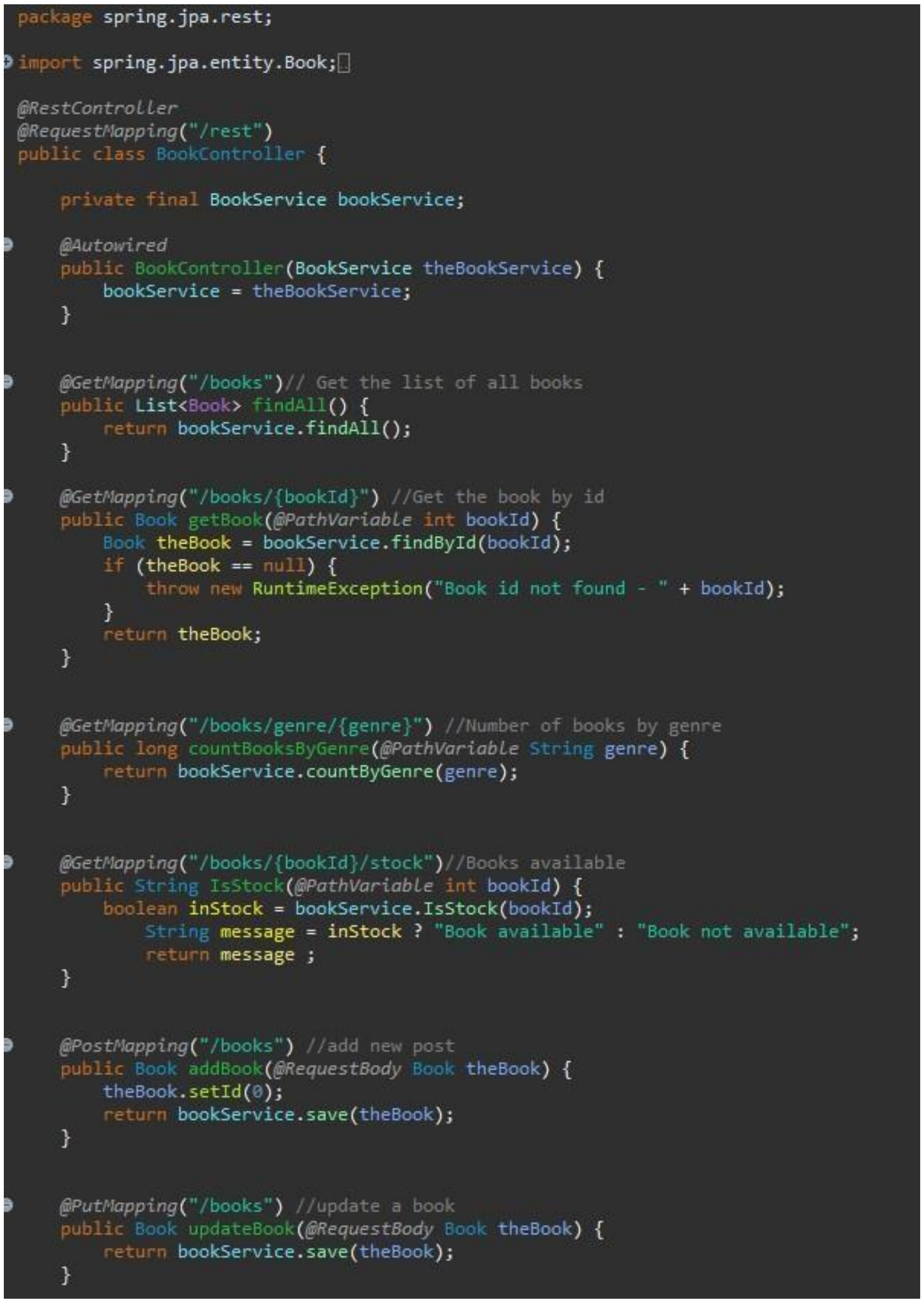
```
package spring.jpa.service;

import spring.jpa.dao.BookRepository;

@Service
public class BookServiceImpl implements BookService {

    @PersistenceContext
    private EntityManager entityManager;
    private BookRepository bookrepository;

    @Autowired
    public BookServiceImpl(BookRepository thebookrepository) {
        bookrepository = thebookrepository;
    }


    @Override
    public List<Book> findAll() { //find all books
        return bookrepository.findAll();
    }


   @Override
   public Book findById(int theId) { //find by ID
    Optional<Book> result = bookrepository.findById(theId);
    return result.orElse(null);
  }


   @Override
   public long countByGenre(String genre) { //count books by genre
       String jpql = "SELECT COUNT(b) FROM Book b WHERE b.genre = :genre";
       TypedQuery<Long> query = entityManager.createQuery(jpql, Long.class);
       query.setParameter("genre", genre);
       return query.getSingleResult();
    }
```



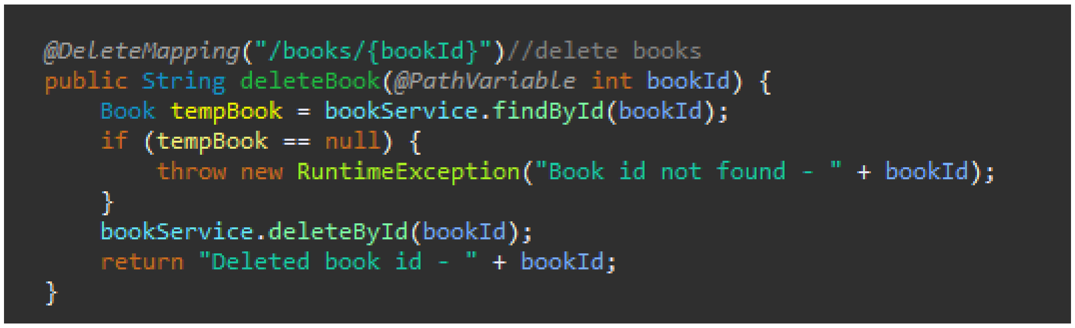
```
  @Override
  public boolean IsStock(int bookId) { //Checks if a book is in stock
      Book book = entityManager.find(Book.class, bookId);
      return book != null && book.getStock() > 0 ;
 }


  @Transactional
  @Override
  public Book save(Book theBook) { //Add new book or update of one
      return bookrepository.save(theBook);
  }


  @Transactional
  @Override
  public void deleteById(int theId) {//Delete book
      bookrepository.deleteById(theId);
  }
```

Rest Controller

The BookController serves as the REST controller, handling HTTP requests related to book management.

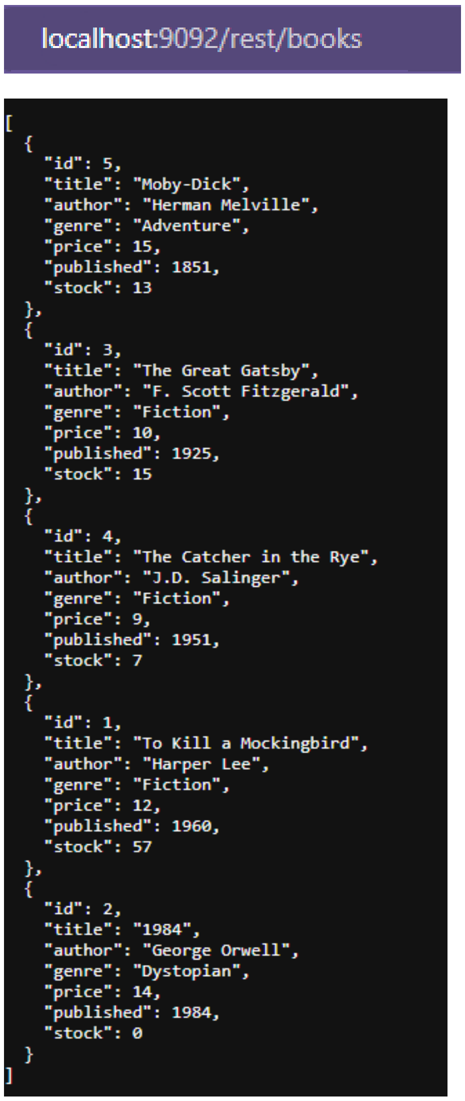


```java
package spring.jpa.rest;

import spring.jpa.entity.Book;

@RestController
@RequestMapping("/rest")
public class BookController {

    private final BookService bookService;

    @Autowired
    public BookController(BookService theBookService) {
        bookService = theBookService;
    }


    @GetMapping("/books")// Get the list of all books
    public List<Book> findAll() {
        return bookService.findAll();
    }

    @GetMapping("/books/{bookId}") //Get the book by id
    public Book getBook(@PathVariable int bookId) {
        Book theBook = bookService.findById(bookId);
        if (theBook == null) {
            throw new RuntimeException("Book id not found - " + bookId);
        }
        return theBook;
    }


    @GetMapping("/books/genre/{genre}") //Number of books by genre
    public long countBooksByGenre(@PathVariable String genre) {
        return bookService.countByGenre(genre);
    }


    @GetMapping("/books/{bookId}/stock")//Books available
    public String IsStock(@PathVariable int bookId) {
        boolean inStock = bookService.IsStock(bookId);
            String message = inStock ? "Book available" : "Book not available";
            return message ;
    }


    @PostMapping("/books") //add new post
    public Book addBook(@RequestBody Book theBook) {
        theBook.setId(0);
        return bookService.save(theBook);
    }


    @PutMapping("/books") //update a book
    public Book updateBook(@RequestBody Book theBook) {
        return bookService.save(theBook);
    }
```



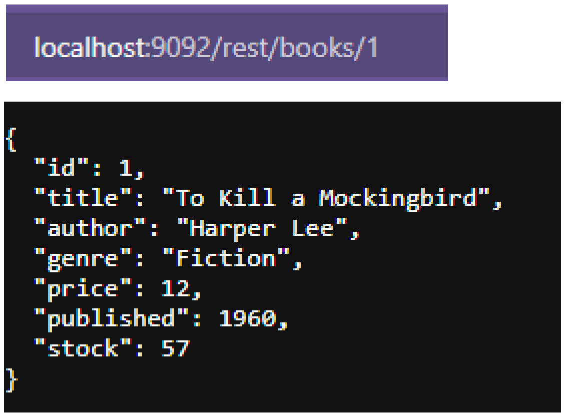
```java
@DeleteMapping("/books/{bookId}")//delete books
public String deleteBook(@PathVariable int bookId) {
    Book tempBook = bookService.findById(bookId);
    if (tempBook == null) {
        throw new RuntimeException("Book id not found - " + bookId);
    }
    bookService.deleteById(bookId);
    return "Deleted book id - " + bookId;
}
```

Outputs



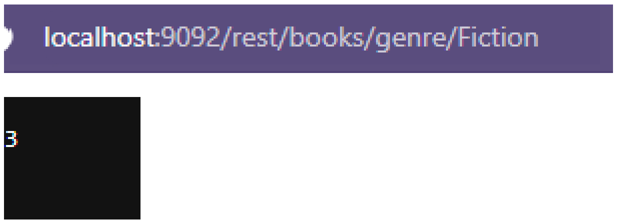
localhost:9092/rest/books

```
[
  {
    "id": 5,
    "title": "Moby-Dick",
    "author": "Herman Melville",
    "genre": "Adventure",
    "price": 15,
    "published": 1851,
    "stock": 13
  },
  {
    "id": 3,
    "title": "The Great Gatsby",
    "author": "F. Scott Fitzgerald",
    "genre": "Fiction",
    "price": 10,
    "published": 1925,
    "stock": 15
  },
  {
    "id": 4,
    "title": "The Catcher in the Rye",
    "author": "J.D. Salinger",
    "genre": "Fiction",
    "price": 9,
    "published": 1951,
    "stock": 7
  },
  {
    "id": 1,
    "title": "To Kill a Mockingbird",
    "author": "Harper Lee",
    "genre": "Fiction",
    "price": 12,
    "published": 1960,
    "stock": 57
  },
  {
    "id": 2,
    "title": "1984",
    "author": "George Orwell",
    "genre": "Dystopian",
    "price": 14,
    "published": 1984,
    "stock": 0
  }
]
```



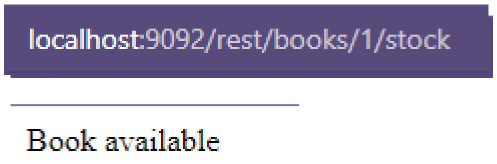
localhost:9092/rest/books/1

```
{
  "id": 1,
  "title": "To Kill a Mockingbird",
  "author": "Harper Lee",
  "genre": "Fiction",
  "price": 12,
  "published": 1960,
  "stock": 57
}
```



localhost:9092/rest/books/genre/Fiction

```
3
```



localhost:9092/rest/books/1/stock

```
Book available
```