Here is the implementation of the BookController class with request mappings for /books and handling of HTTP methods:

BookController.java

package com.example.bookstoreapi.controllers;

import com.example.bookstoreapi.models.Book;

import com.example.bookstoreapi.services.BookService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

private final BookService bookService;

@Autowired

public BookController(BookService bookService) {

this.bookService = bookService;

}

@GetMapping

public ResponseEntity<List<Book>> getAllBooks() {

List<Book> books = bookService.getAllBooks();

return new ResponseEntity<>(books, HttpStatus.OK);

}

@GetMapping("/{id}")

public ResponseEntity<Book> getBookById(@PathVariable Long id) {

Book book = bookService.getBookById(id);

return new ResponseEntity<>(book, HttpStatus.OK);

}

@PostMapping

public ResponseEntity<Book> createBook(@RequestBody Book book) {

Book newBook = bookService.createBook(book);

return new ResponseEntity<>(newBook, HttpStatus.CREATED);

}

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book book) {

Book updatedBook = bookService.updateBook(id, book);

return new ResponseEntity<>(updatedBook, HttpStatus.OK);

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

bookService.deleteBook(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

}

BookService.java

package com.example.bookstoreapi.services;

import com.example.bookstoreapi.models.Book;

import com.example.bookstoreapi.repositories.BookRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class BookService {

private final BookRepository bookRepository;

@Autowired

public BookService(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Book getBookById(Long id) {

return bookRepository.findById(id).orElse(null);

}

public Book createBook(Book book) {

return bookRepository.save(book);

}

public Book updateBook(Long id, Book book) {

Book existingBook = getBookById(id);

if (existingBook != null) {

existingBook.setTitle(book.getTitle());

existingBook.setAuthor(book.getAuthor());

existingBook.setPrice(book.getPrice());

existingBook.setIsbn(book.getIsbn());

return bookRepository.save(existingBook);

}

return null;

}

public void deleteBook(Long id) {

bookRepository.deleteById(id);

}

}

BookRepository.java

package com.example.bookstoreapi.repositories;

import com.example.bookstoreapi.models.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}

Book.java

package com.example.bookstoreapi.models;

import lombok.Data;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Data

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private double price;

private String isbn;

}

This implementation provides the following RESTful endpoints:

GET /books: Returns a list of all books.

GET /books/{id}: Returns a book by ID.

POST /books: Creates a new book.

PUT /books/{id}: Updates a book by ID.

DELETE /books/{id}: Deletes a book by ID.