Book Project Spring Boot

\*\*\*\*\*\*\*\*\*\*\* Book Controller \*\*\*\*\*\*\*\*\*\*\*\*\*

@RestController  
@RequestMapping(value = "/books")  
@CrossOrigin(origins = "\*")  
public class BookController {  
 @Autowired  
 BookService service;  
  
 @GetMapping("/getall")  
 public List<Book> getAllBooks(){  
 return service.getAllBooks();  
 }  
  
 @PostMapping("/save")  
 public Book addBook(@RequestBody Book book){  
 return service.addBook(book);  
 }  
  
 @GetMapping("/get/{bookId}")  
 public ResponseEntity<Book> getBookById(@PathVariable Long bookId){  
 Book book = service.getBookById(bookId);  
 return ResponseEntity.*ok*(book);  
 }  
  
 @PutMapping("/update/{bookId}")  
 public ResponseEntity<Book> updateBook(@PathVariable Long bookId, @RequestBody Book bookDetails){  
  
 Book updatedBook = service.updateBook(bookId, bookDetails);  
 return ResponseEntity.*ok*(updatedBook);  
 }  
  
 @DeleteMapping("/delete/{bookId}")  
 public ResponseEntity<Map<String, Boolean>> deleteBook(@PathVariable Long bookId){  
 Boolean resp = service.deleteBook(bookId);  
  
 Map<String, Boolean> response = new HashMap<>();  
 response.put("deleted", resp);  
 return ResponseEntity.*ok*(response);  
 }  
  
}

\*\*\*\*\*\*\*\*\*\*\* Book.java \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Entity  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class Book {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long bookId;  
 private String bookName;  
 private String bookAuthor;  
 private String bookPublisher;  
 private String bookGenre;  
 private int bookPrice;  
}

\*\*\*\*\*\*\*\*\*\*\*\*\* ResourceNotFoundException \*\*\*\*\*\*\*\*\*

package com.stg.book.project.exception;  
  
import org.springframework.http.HttpStatus;  
import org.springframework.web.bind.annotation.ResponseStatus;  
  
@ResponseStatus(value = HttpStatus.*NOT\_FOUND*)  
public class ResourceNotFoundException extends RuntimeException{  
  
 public ResourceNotFoundException(String message) {  
 super(message);  
 }  
}

\*\*\*\*\*\*\*\*\*\*\*\*\* BookRepository \*\*\*\*\*\*\*\*\*

import com.stg.book.project.entity.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Long> {  
}

\*\*\*\*\*\*\*\*\*\*\*BookService\*\*\*\*\*\*\*\*\*\*\*8

@Service  
public class BookService {  
  
 @Autowired  
 BookRepository repository;  
  
 public List<Book> getAllBooks() {  
 return repository.findAll();  
 }  
  
 public Book addBook(Book book) {  
 return repository.save(book);  
 }  
  
 public Book getBookById(Long bookId) {  
 Book book = repository.findById(bookId)  
 .orElseThrow(() -> new ResourceNotFoundException("Bood does not exist with id: " + bookId));  
  
 return book;  
 }  
  
 public Book updateBook(Long bookId, Book bookDetails) {  
  
 Book book = repository.findById(bookId)  
 .orElseThrow(() -> new ResourceNotFoundException("Book does not exist with id :" + bookId));  
  
 book.setBookAuthor(bookDetails.getBookAuthor());  
 book.setBookName(bookDetails.getBookName());  
 book.setBookGenre(bookDetails.getBookGenre());  
 book.setBookPublisher(bookDetails.getBookPublisher());  
 book.setBookPrice(bookDetails.getBookPrice());  
  
 Book updatedBook = repository.save(book);  
 return updatedBook;  
 }  
  
 public Boolean deleteBook(Long bookId) {  
 Book book = repository.findById(bookId)  
 .orElseThrow(() -> new ResourceNotFoundException("Book does not exist with id :" + bookId));  
  
 repository.delete(book);  
 return Boolean.*TRUE*;  
 }  
}

\*\*\*\*\*\*\*\*\*applicatipn.yml \*\*\*\*\*\*\*\*\*\*\*

spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/test  
  
 username: root  
 password: root  
  
 jpa:  
 show-sql: true  
 hibernate:  
 ddl-auto: update  
  
logging:  
 level:  
 root: INFO  
  
 file:  
 path: 'D:\SAMEER\Practice\angular\project\bookproject\logs'  
 name: 'D:\SAMEER\Practice\angular\project\bookproject\logs\bookproject.log'