

Digital Nurture 3.0 I Deep Skilling (WEEK 4&5 SOLUTIONS)

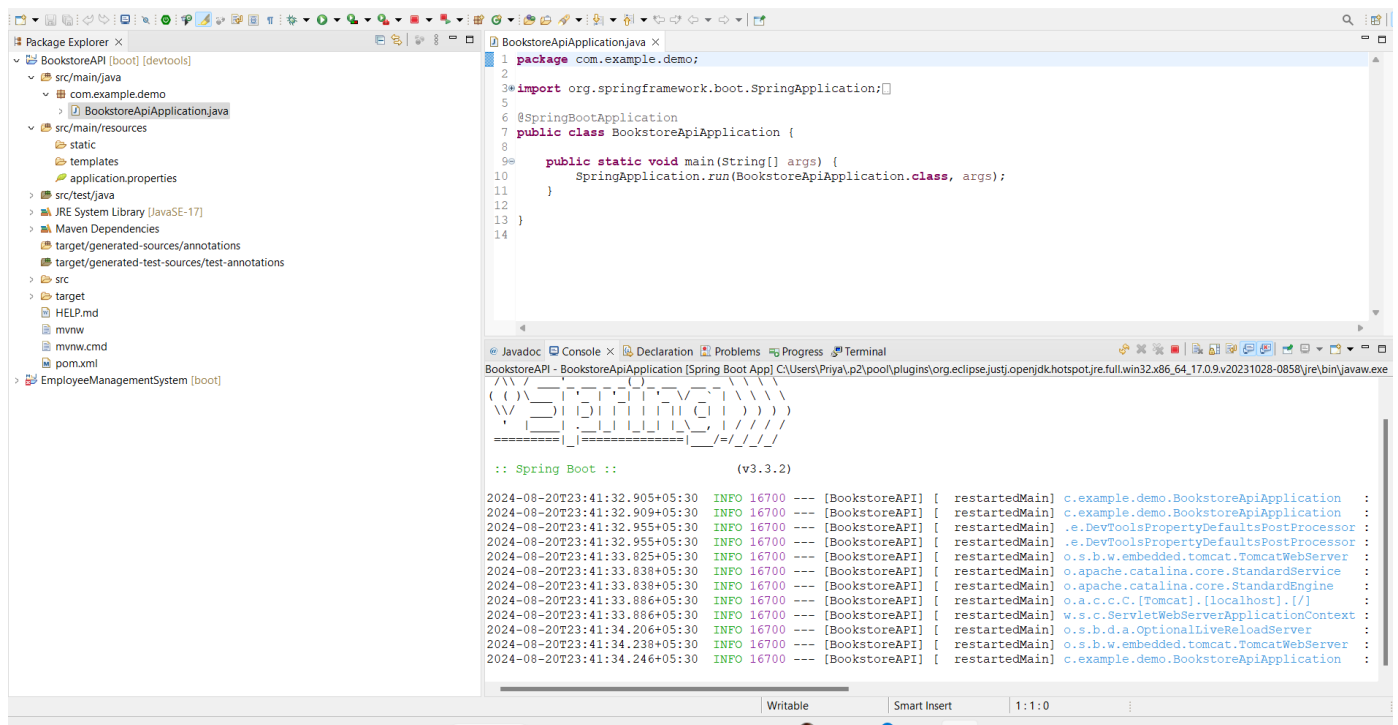
Superset ID:5021661

Name: Priya Hazra

Exercise 1: Online Bookstore - Setting Up RESTful Services Business Scenario:

What's New in Spring Boot 3:

- **Java 17+ Support:** Spring Boot 3 requires at least Java 17, taking advantage of its new features and performance improvements.
- **Observability:** Enhanced support for observability and monitoring through Micrometer, with built-in support for distributed tracing.
- **GraalVM Native Image:** First-class support for building and running native images using GraalVM, reducing memory consumption and startup time.
- **Jakarta EE 9+:** Migration from javax.* to jakarta.* package namespaces in alignment with Jakarta EE 9.
- **Spring AOT Plugin:** Introduces Ahead-of-Time (AOT) compilation for faster startup and lower memory consumption.



The screenshot shows an IDE with the Package Explorer on the left, the BookstoreAPIApplication.java file open in the center, and the Console on the right. The Package Explorer shows the project structure for BookstoreAPI, including src/main/java, src/main/resources, and target. The BookstoreAPIApplication.java file contains the following code:

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class BookstoreAPIApplication {
7     public static void main(String[] args) {
8         SpringApplication.run(BookstoreAPIApplication.class, args);
9     }
10 }
11
12
13
14
```

The Console shows the output of the application, including the Spring Boot logo and the following log messages:

```
2024-08-20T23:41:32.905+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] c.example.demo.BookstoreAPIApplication :
2024-08-20T23:41:32.909+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] c.example.demo.BookstoreAPIApplication :
2024-08-20T23:41:32.955+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor :
2024-08-20T23:41:33.025+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer :
2024-08-20T23:41:33.838+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] o.apache.catalina.core.StandardService :
2024-08-20T23:41:33.886+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] :
2024-08-20T23:41:33.886+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] w.s.c.ServletWebServerApplicationContext :
2024-08-20T23:41:34.206+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer :
2024-08-20T23:41:34.238+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer :
2024-08-20T23:41:34.246+05:30 INFO 16700 --- [BookstoreAPI] [ restartedMain] c.example.demo.BookstoreAPIApplication :
```

Exercise 2: Online Bookstore - Creating Basic REST Controllers Business Scenario:

Book.JAVA

```
package com.example.demo.model;
```

```
public class Book {
```

```

private Long id;
private String title;
private String author;
private Double price;
private String isbn;
public Book() {
}

public Book(Long id, String title, String author, Double price, String isbn) {
    this.id = id;
    this.title = title;
    this.author = author;
    this.price = price;
    this.isbn = isbn;
}
public Long getId() {
    return id;
}
public void setId(Long id) {
    this.id = id;
}
public String getTitle() {
    return title;
}
public void setTitle(String title) {
    this.title = title;
}
public String getAuthor() {
    return author;
}
public void setAuthor(String author) {
    this.author = author;
}
public Double getPrice() {
    return price;
}
public void setPrice(Double price) {
    this.price = price;
}
public String getIsbn() {
    return isbn;
}
public void setIsbn(String isbn) {
    this.isbn = isbn;
}
}

```

BookController.java

```

package com.example.demo.controller;

import com.example.demo.model.Book;

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

```

```
import java.util.List;

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

    @GetMapping
    public List<Book> getAllBooks() {
        return List.of(
            new Book(1L, "Book Title 1", "Author 1", 19.99, "123-4567890123"),
            new Book(2L, "Book Title 2", "Author 2", 29.99, "456-7890123456")
        );
    }

    @PostMapping
    public Book createBook(@RequestBody Book book) {
        book.setId(3L);
        return book;
    }

    @PutMapping("/{id}")
    public Book updateBook(@PathVariable Long id, @RequestBody Book book) {
        book.setId(id);
        return book;
    }

    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable Long id) {
        System.out.println("Book with ID " + id + " has been deleted.");
    }
}
```

The screenshot shows an IDE with two Java files open: `BookstoreAPIApplication.java` and `BookController.java`. The `BookstoreAPIApplication.java` file contains the following code:

```
14= public Book(Long id, String title, String author, Double price, String isbn) {
15     this.id = id;
16     this.title = title;
17     this.author = author;
18     this.price = price;
19     this.isbn = isbn;
20 }
21
22= public Long getId() {
23     return id;
24 }
25
26= public void setId(Long id) {
27     this.id = id;
28 }
29
30= public String getTitle() {
31     return title;
32 }
33
```

The console window shows the following logs:

```
BookstoreAPI - BookstoreAPIApplication [Spring Boot App] C:\Users\Priya\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (21-Aug-2024, 12:12:36 am) [pid
:: Spring Boot :: (v3.3.2)
2024-08-21T00:31:22.620+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] c.example.demo.BookstoreAPIApplication : Starting BookstoreAPI
2024-08-21T00:31:22.620+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] c.example.demo.BookstoreAPIApplication : No active profile set,
2024-08-21T00:31:22.709+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized wit
2024-08-21T00:31:22.710+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomc
2024-08-21T00:31:22.710+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet engin
2024-08-21T00:31:22.721+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] o.a.c.c.C.[Tomcat-1].[localhost].[/] : Initializing Spring em
2024-08-21T00:31:22.721+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationCor
2024-08-21T00:31:22.769+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is r
2024-08-21T00:31:22.775+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port
2024-08-21T00:31:22.778+05:30 INFO 10700 --- [BookstoreAPI] [ restartedMain] c.example.demo.BookstoreAPIApplication : Started BookstoreAPI
```

Exercise 3: Online Bookstore - Handling Path Variables and Query Parameters

BookController.java

```
package com.example.demo.controller;
```

```
import com.example.demo.model.Book;
```

```
import com.example.demo.service.BookService;
```

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

```
import org.springframework.http.ResponseEntity;
```

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

```
import java.util.List;
```

```
@RestController
```

```
@RequestMapping("/books")
```

```
public class BookController {
```

```
    @Autowired
```

```
    private BookService bookService;
```

@GetMapping

```
public List<Book> getAllBooks() {  
    return bookService.findAllBooks();  
}
```

@GetMapping("/{id}")

```
public ResponseEntity<Book> getBookById(@PathVariable("id") Long id) {  
    Book book = bookService.findBookById(id);  
    if (book != null) {  
        return ResponseEntity.ok(book);  
    } else {  
        return ResponseEntity.notFound().build();  
    }  
}
```

@PostMapping

```
public Book createBook(@RequestBody Book book) {  
    return bookService.saveBook(book);  
}
```

@PutMapping("/{id}")

```
public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book book) {  
    Book updatedBook = bookService.updateBook(id, book);  
    if (updatedBook != null) {  
        return ResponseEntity.ok(updatedBook);  
    } else {  
        return ResponseEntity.notFound().build();  
    }  
}
```

@DeleteMapping("/{id}")

```
public ResponseEntity<Void> deleteBook(@PathVariable Long id) {  
    boolean isDeleted = bookService.deleteBook(id);  
    if (isDeleted) {
```

```

        return ResponseEntity.noContent().build();
    } else {
        return ResponseEntity.notFound().build();
    }
}

@GetMapping("/search")
public List<Book> getBooksByFilters(
    @RequestParam(required = false) String title,
    @RequestParam(required = false) String author) {

    return bookService.findBooksByFilters(title, author);
}
}

```

BookService.java

```
package com.example.demo.service;
```

```
import com.example.demo.model.Book;
```

```
import org.springframework.stereotype.Service;
```

```
import java.util.List;
```

```
import java.util.stream.Collectors;
```

```
import java.util.ArrayList;
```

```
@Service
```

```
public class BookService {
```

```
    private List<Book> books = new ArrayList<>();
```

```
    public BookService() {
```

```
        // Initialize with some dummy data
```

```
        books.add(new Book(1L, "Book Title 1", "Author 1", 19.99, "123-4567890123"));
```

```
        books.add(new Book(2L, "Book Title 2", "Author 2", 29.99, "456-7890123456"));
```

```
    }
```

```
public List<Book> findAllBooks() {  
    return books;  
}
```

```
public Book findBookById(Long id) {  
    return books.stream()  
        .filter(book -> book.getId().equals(id))  
        .findFirst()  
        .orElse(null);  
}
```

```
public Book saveBook(Book book) {  
    book.setId((long) (books.size() + 1)); // Simple ID assignment  
    books.add(book);  
    return book;  
}
```

```
public Book updateBook(Long id, Book book) {  
    Book existingBook = findBookById(id);  
    if (existingBook != null) {  
        existingBook.setTitle(book.getTitle());  
        existingBook.setAuthor(book.getAuthor());  
        existingBook.setPrice(book.getPrice());  
        existingBook.setIsbn(book.getIsbn());  
        return existingBook;  
    }  
    return null;  
}
```

```
public boolean deleteBook(Long id) {  
    return books.removeIf(book -> book.getId().equals(id));  
}
```

```

public List<Book> findBooksByFilters(String title, String author) {

    return books.stream()

        .filter(book -> (title == null || book.getTitle().equalsIgnoreCase(title)) &&

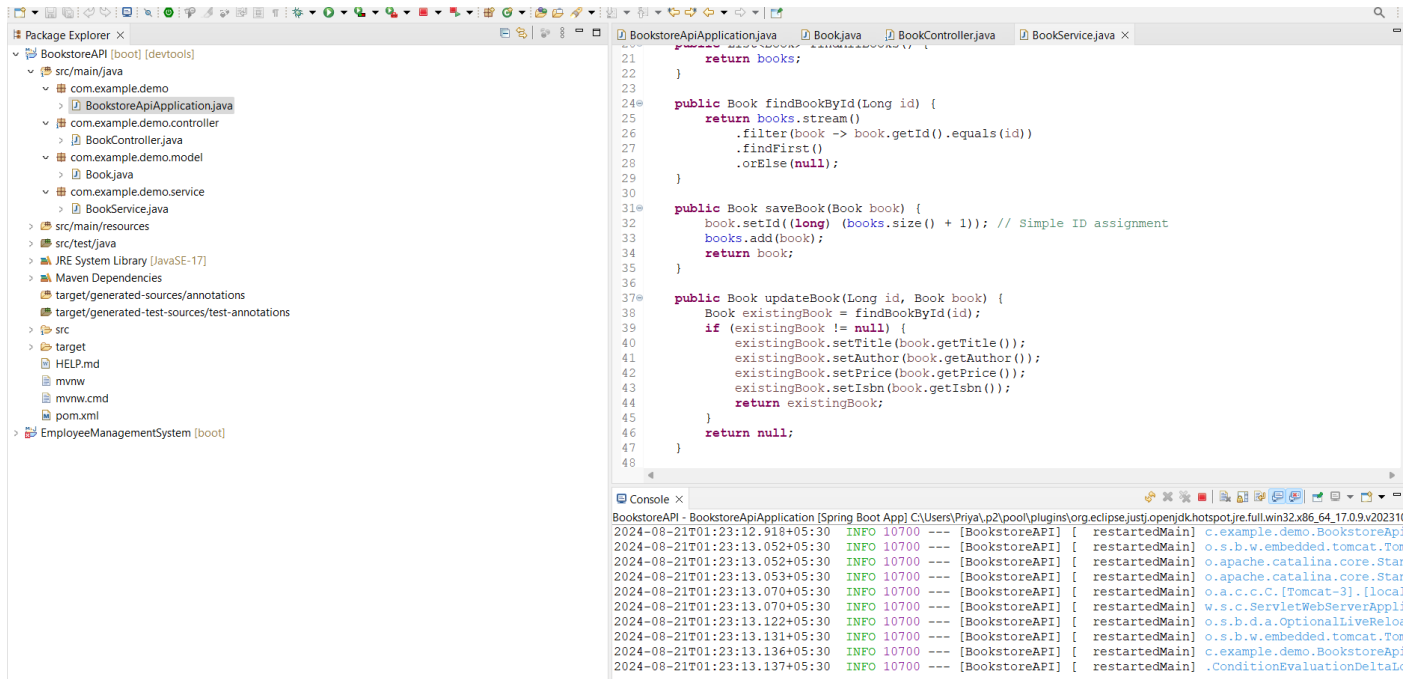
            (author == null || book.getAuthor().equalsIgnoreCase(author)))

        .collect(Collectors.toList());

}

}

```



Exercise 4: Online Bookstore - Processing Request Body and Form Data

Customer.java

```
package com.example.demo.model;
```

```

public class Customer {

    private Long id;

    private String name;

    private String email;

    private String password;

    public Customer() {}
}

```



```
public Customer(Long id, String name, String email, String password) {  
    this.id = id;  
    this.name = name;  
    this.email = email;  
    this.password = password;  
}
```

```
public Long getId() {  
    return id;  
}
```

```
public void setId(Long id) {  
    this.id = id;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public void setName(String name) {  
    this.name = name;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```
public void setEmail(String email) {
```

```
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

CustomerService.java

```
package com.example.demo.service;

import com.example.demo.model.Customer;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service

public class CustomerService {
    private List<Customer> customers = new ArrayList<>();

    public CustomerService() {

        customers.add(new Customer(1L, "John Doe", "john@example.com", "password123"));
    }
}
```

```
}
```

```
public Customer saveCustomer(Customer customer) {  
    customer.setId((long) (customers.size() + 1)); // Simple ID assignment  
    customers.add(customer);  
    return customer;  
}
```

```
public List<Customer> findAllCustomers() {  
    return customers;  
}  
}
```

CustomerController.java

```
package com.example.demo.controller;  
  
import com.example.demo.model.Customer;  
import com.example.demo.service.CustomerService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.*;  
import org.springframework.web.multipart.MultipartFile;  
  
import java.io.IOException;  
  
@RestController  
@RequestMapping("/customers")  
public class CustomerController {
```

@Autowired

private CustomerService customerService;

@PostMapping

```
public ResponseEntity<Customer> createCustomer(@RequestBody Customer customer) {  
    Customer createdCustomer = customerService.saveCustomer(customer);  
    return ResponseEntity.ok(createdCustomer);  
}
```

@PostMapping("/register")

```
public ResponseEntity<Customer> registerCustomer(  
    @RequestParam("name") String name,  
    @RequestParam("email") String email,  
    @RequestParam("password") String password) {  
  
    Customer customer = new Customer(null, name, email, password);  
    Customer registeredCustomer = customerService.saveCustomer(customer);  
  
    return ResponseEntity.ok(registeredCustomer);  
}
```

@PostMapping("/registerWithFile")

```
public ResponseEntity<String> registerCustomerWithFile(  
    @RequestParam("name") String name,  
    @RequestParam("email") String email,  
    @RequestParam("password") String password,  
    @RequestParam("file") MultipartFile file) throws IOException {
```

```
Customer customer = new Customer(null, name, email, password);
customerService.saveCustomer(customer);

return ResponseEntity.ok("Customer registered with file");
}
}
```

```
curl -X POST http://localhost:8083/customers \
  -H "Content-Type: application/json" \
  -d '{"name": "Priya", "email": "priya222@example.com", "password": "securepassword"}'
```

output:

```
"id": 1,
"name": "Priya",
"email": "priya222@example.com",
"password": "securepassword"
```

```
curl -X POST http://localhost:8083/customers/register \
  -d "name=Priya&email=priya222@example.com&password=securepassword"
```

Output:

```
"id": 2,
"name": "Priya",
"email": "priya222@example.com",
"password": "securepassword"
```

```
curl -X POST http://localhost:8083/customers/registerWithFile \  
-F "name=Priya" \  
-F "email=priya222@example.com" \  
-F "password=securepassword" \  
-F "file=@profile.jpg"
```

Output:

"Customer registered with file"

Exercise 5: Online Bookstore - Customizing Response Status and Headers

```
package com.example.demo.controller;
```

```
import com.example.demo.model.Book;
```

```
import com.example.demo.service.BookService;
```

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

```
import org.springframework.http.HttpHeaders;
```

```
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.ResponseEntity;
```

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

```
import java.util.List;
```

```
import java.util.Optional;
```

```
@RestController
```

```
@RequestMapping("/books")
```

```
public class BookController {
```

```
    @Autowired
```

```
    private BookService bookService;
```

@GetMapping

```
public ResponseEntity<List<Book>> getAllBooks() {  
    List<Book> books = bookService.findAllBooks();  
    HttpHeaders headers = new HttpHeaders();  
    headers.add("Custom-Header", "CustomHeaderValue");  
    return new ResponseEntity<>(books, headers, HttpStatus.OK);  
}
```

@GetMapping("/{id}")

```
public ResponseEntity<Book> getBookById(@PathVariable("id") Long id) {  
    Optional<Book> book = bookService.findBookById(id);  
    if (book.isPresent()) {  
        return new ResponseEntity<>(book.get(), HttpStatus.OK);  
    } else {  
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);  
    }  
}
```

@PostMapping

```
public ResponseEntity<Book> createBook(@RequestBody Book book) {  
    Book createdBook = bookService.saveBook(book);  
    HttpHeaders headers = new HttpHeaders();  
    headers.add("Location", "/books/" + createdBook.getId());  
    return new ResponseEntity<>(createdBook, headers, HttpStatus.CREATED);  
}
```

@PutMapping("/{id}")

```
public ResponseEntity<Book> updateBook(@PathVariable("id") Long id, @RequestBody Book book) {  
    if (bookService.existsById(id)) {  
        book.setId(id);  
        Book updatedBook = bookService.saveBook(book);  
        return new ResponseEntity<>(updatedBook, HttpStatus.OK);  
    }  
}
```

```

    } else {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}

@DeleteMapping("/{id}")
public ResponseEntity<Void> deleteBook(@PathVariable("id") Long id) {
    if (bookService.existsById(id)) {
        bookService.deleteBookById(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    } else {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}
}

```

Exercise 6: Online Bookstore - Exception Handling in REST Controllers

ErrorResponse.java

```

package com.example.demo.model;

import org.springframework.http.HttpStatus;

public class ErrorResponse {
    private String message;
    private HttpStatus status;

    public ErrorResponse(String message, HttpStatus status) {
        this.message = message;
        this.status = status;
    }
}

```



```
// Getters and setters

public String getMessage() {
    return message;
}

public void setMessage(String message) {
    this.message = message;
}

public HttpStatus getStatus() {
    return status;
}

public void setStatus(HttpStatus status) {
    this.status = status;
}
}
```

GlobalExceptionHandler.java

```
package com.example.demo.controller;

import com.example.demo.model.ErrorResponse;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.method.annotation.MethodArgumentTypeMismatchException;

import javax.persistence.EntityNotFoundException;
import javax.validation.ConstraintViolationException;
```

@ControllerAdvice

public class GlobalExceptionHandler {

 @ExceptionHandler(EntityNotFoundException.class)

 @ResponseStatus(HttpStatus.NOT_FOUND)

 public ResponseEntity<ErrorResponse> handleEntityNotFoundException(EntityNotFoundException ex) {

 ErrorResponse errorResponse = new ErrorResponse(ex.getMessage(), HttpStatus.NOT_FOUND);

 return new ResponseEntity<>(errorResponse, HttpStatus.NOT_FOUND);

 }

 @ExceptionHandler(ConstraintViolationException.class)

 @ResponseStatus(HttpStatus.BAD_REQUEST)

 public ResponseEntity<ErrorResponse> handleConstraintViolationException(ConstraintViolationException ex) {

 ErrorResponse errorResponse = new ErrorResponse(ex.getMessage(), HttpStatus.BAD_REQUEST);

 return new ResponseEntity<>(errorResponse, HttpStatus.BAD_REQUEST);

 }

 @ExceptionHandler(MethodArgumentTypeMismatchException.class)

 @ResponseStatus(HttpStatus.BAD_REQUEST)

 public ResponseEntity<ErrorResponse>
 handleMethodArgumentTypeMismatchException(MethodArgumentTypeMismatchException ex) {

 String message = String.format("Invalid value for parameter '%s': %s", ex.getName(), ex.getValue());

 ErrorResponse errorResponse = new ErrorResponse(message, HttpStatus.BAD_REQUEST);

 return new ResponseEntity<>(errorResponse, HttpStatus.BAD_REQUEST);

 }

 @ExceptionHandler(Exception.class)

 @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)

 public ResponseEntity<ErrorResponse> handleGenericException(Exception ex) {

 ErrorResponse errorResponse = new ErrorResponse("An unexpected error occurred",
 HttpStatus.INTERNAL_SERVER_ERROR);

 return new ResponseEntity<>(errorResponse, HttpStatus.INTERNAL_SERVER_ERROR);

```
}
```

```
}
```

Dependencies:

```
<dependency>
```

```
    <groupId>javax.validation</groupId>
```

```
    <artifactId>validation-api</artifactId>
```

```
    <version>2.0.1.Final</version> <!-- Make sure this version matches your project setup -->
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>org.hibernate</groupId>
```

```
    <artifactId>hibernate-core</artifactId>
```

```
    <version>5.6.0.Final</version> <!-- Adjust the version according to your setup -->
```

```
</dependency>
```

Exercise7: Online Bookstore - Introduction to Data Transfer Objects (DTOs)

BookDTO.java

```
public class BookDTO {
```

```
    private Long id;
```

```
    private String title;
```

```
    private String author;
```

```
    private Double price;
```

```
    // Getters and Setters
```

```
    public Long getId() {
```

```
        return id;
```

```
}

public void setId(Long id) {
    this.id = id;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

public Double getPrice() {
    return price;
}

public void setPrice(Double price) {
    this.price = price;
}
}
```

CustomerDTO.java

```
public class CustomerDTO {
```

```
private Long id;

private String name;

private String email;


// Getters and Setters

public Long getId() {

    return id;

}


public void setId(Long id) {

    this.id = id;

}


public String getName() {

    return name;

}


public void setName(String name) {

    this.name = name;

}


public String getEmail() {

    return email;

}


public void setEmail(String email) {

    this.email = email;

}

}
```

Dependencies:

<dependency>

<groupId>org.mapstruct</groupId>

```
<artifactId>mapstruct</artifactId>
<version>1.5.5.Final</version>
</dependency>
<dependency>
  <groupId>org.mapstruct</groupId>
  <artifactId>mapstruct-processor</artifactId>
  <version>1.5.5.Final</version>
  <scope>provided</scope>
</dependency>
```

MapperInterface:

```
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.factory.Mappers;
```

@Mapper

```
public interface BookMapper {
    BookMapper INSTANCE = Mappers.getMapper(BookMapper.class);

    BookDTO bookToBookDTO(Book book);
    Book bookDTOToBook(BookDTO bookDTO);
}
```

@Mapper

```
public interface CustomerMapper {
    CustomerMapper INSTANCE = Mappers.getMapper(CustomerMapper.class);

    CustomerDTO customerToCustomerDTO(Customer customer);
    Customer customerDTOToCustomer(CustomerDTO customerDTO);
}
```

Customizing Date Serialization:

```
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonSerialize;

public class BookDTO {

    private Long id;

    private String title;

    private String author;

    private Double price;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-dd")
    private LocalDate publishedDate;

    // Getters and Setters

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public String getAuthor() {
```

```
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public LocalDate getPublishedDate() {
        return publishedDate;
    }

    public void setPublishedDate(LocalDate publishedDate) {
        this.publishedDate = publishedDate;
    }
}
```

Exercise 8: Online Bookstore - Implementing CRUD Operations

BookController.java

```
package com.example.bookstore.controller;

import com.example.bookstore.dto.BookDTO;
import com.example.bookstore.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
```



```
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

@RestController
@RequestMapping("/api/books")
@Validated
public class BookController {

    @Autowired
    private BookService bookService;

    @PostMapping
    public ResponseEntity<BookDTO> createBook(@Valid @RequestBody BookDTO bookDTO) {
        BookDTO createdBook = bookService.createBook(bookDTO);
        return new ResponseEntity<>(createdBook, HttpStatus.CREATED);
    }

    @GetMapping("/{id}")
    public ResponseEntity<BookDTO> getBookById(@PathVariable Long id) {
        BookDTO bookDTO = bookService.getBookById(id);
        return new ResponseEntity<>(bookDTO, HttpStatus.OK);
    }

    @GetMapping
    public ResponseEntity<List<BookDTO>> getAllBooks() {
        List<BookDTO> books = bookService.getAllBooks();
        return new ResponseEntity<>(books, HttpStatus.OK);
    }
}
```

```
@PutMapping("/{id}")
```

```
public ResponseEntity<BookDTO> updateBook(@PathVariable Long id, @Valid @RequestBody BookDTO bookDTO) {
```

```
    BookDTO updatedBook = bookService.updateBook(id, bookDTO);
```

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

```
}
```

```
@DeleteMapping("/{id}")
```

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

```
    bookService.deleteBook(id);
```

```
    return new ResponseEntity<>(HttpStatus.NO_CONTENT);
```

```
}
```

```
}
```

CustomerController.java

```
package com.example.bookstore.controller;
```

```
import com.example.bookstore.dto.CustomerDTO;
```

```
import com.example.bookstore.service.CustomerService;
```

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

```
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.validation.annotation.Validated;
```

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

```
import javax.validation.Valid;
```

```
import java.util.List;
```

```
@RestController
```

```
@RequestMapping("/api/customers")
```

```
@Validated
```

```
public class CustomerController {
```

```
    @Autowired
```

```
private CustomerService customerService;
```

```
@PostMapping
```

```
public ResponseEntity<CustomerDTO> createCustomer(@Valid @RequestBody CustomerDTO customerDTO) {
```

```
    CustomerDTO createdCustomer = customerService.createCustomer(customerDTO);
```

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

```
}
```

```
@GetMapping("/{id}")
```

```
public ResponseEntity<CustomerDTO> getCustomerById(@PathVariable Long id) {
```

```
    CustomerDTO customerDTO = customerService.getCustomerById(id);
```

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

```
}
```

```
@GetMapping
```

```
public ResponseEntity<List<CustomerDTO>> getAllCustomers() {
```

```
    List<CustomerDTO> customers = customerService.getAllCustomers();
```

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

```
}
```

```
@PutMapping("/{id}")
```

```
public ResponseEntity<CustomerDTO> updateCustomer(@PathVariable Long id, @Valid @RequestBody CustomerDTO customerDTO) {
```

```
    CustomerDTO updatedCustomer = customerService.updateCustomer(id, customerDTO);
```

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

```
}
```

```
@DeleteMapping("/{id}")
```

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

```
    customerService.deleteCustomer(id);
```

```
    return new ResponseEntity<>(HttpStatus.NO_CONTENT);
```

```
}
```

```
}
```

BookService.java

```
package com.example.bookstore.service;

import com.example.bookstore.dto.BookDTO;
import com.example.bookstore.entity.Book;
import com.example.bookstore.mapper.BookMapper;
import com.example.bookstore.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;

@Service
@Transactional
public class BookService {

    @Autowired
    private BookRepository bookRepository;

    private final BookMapper bookMapper = BookMapper.INSTANCE;

    public BookDTO createBook(BookDTO bookDTO) {
        Book book = bookMapper.bookDTOToBook(bookDTO);
        Book savedBook = bookRepository.save(book);
        return bookMapper.bookToBookDTO(savedBook);
    }

    public BookDTO getBookById(Long id) {
        Book book = bookRepository.findById(id)
```

```

        .orElseThrow(() -> new RuntimeException("Book not found"));

    return bookMapper.bookToBookDTO(book);
}

public List<BookDTO> getAllBooks() {
    List<Book> books = bookRepository.findAll();
    return books.stream()
        .map(bookMapper::bookToBookDTO)
        .toList();
}

public BookDTO updateBook(Long id, BookDTO bookDTO) {
    if (!bookRepository.existsById(id)) {
        throw new RuntimeException("Book not found");
    }

    Book book = bookMapper.bookDTOToBook(bookDTO);
    book.setId(id);
    Book updatedBook = bookRepository.save(book);
    return bookMapper.bookToBookDTO(updatedBook);
}

public void deleteBook(Long id) {
    if (!bookRepository.existsById(id)) {
        throw new RuntimeException("Book not found");
    }

    bookRepository.deleteById(id);
}
}

```

CustomerService.java

```
package com.example.bookstore.service;
```

```
import com.example.bookstore.dto.CustomerDTO;
import com.example.bookstore.entity.Customer;
import com.example.bookstore.mapper.CustomerMapper;
import com.example.bookstore.repository.CustomerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.Optional;

@Service
@Transactional

public class CustomerService {

    @Autowired
    private CustomerRepository customerRepository;

    private final CustomerMapper customerMapper = CustomerMapper.INSTANCE;

    public CustomerDTO createCustomer(CustomerDTO customerDTO) {
        Customer customer = customerMapper.customerDTOToCustomer(customerDTO);
        Customer savedCustomer = customerRepository.save(customer);
        return customerMapper.customerToCustomerDTO(savedCustomer);
    }

    public CustomerDTO getCustomerById(Long id) {
        Customer customer = customerRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("Customer not found"));
        return customerMapper.customerToCustomerDTO(customer);
    }
}
```

```

public List<CustomerDTO> getAllCustomers() {
    List<Customer> customers = customerRepository.findAll();
    return customers.stream()
        .map(customerMapper::customerToCustomerDTO)
        .toList();
}

public CustomerDTO updateCustomer(Long id, CustomerDTO customerDTO) {
    if (!customerRepository.existsById(id)) {
        throw new RuntimeException("Customer not found");
    }
    Customer customer = customerMapper.customerDTOToCustomer(customerDTO);
    customer.setId(id);
    Customer updatedCustomer = customerRepository.save(customer);
    return customerMapper.customerToCustomerDTO(updatedCustomer);
}

public void deleteCustomer(Long id) {
    if (!customerRepository.existsById(id)) {
        throw new RuntimeException("Customer not found");
    }
    customerRepository.deleteById(id);
}
}

```

BookDTO.java

```

package com.example.bookstore.dto;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

public class BookDTO {

```

```
private Long id;
```

```
@NotNull(message = "Title cannot be null")
```

```
@Size(min = 2, max = 100, message = "Title must be between 2 and 100 characters")
```

```
private String title;
```

```
@NotNull(message = "Author cannot be null")
```

```
private String author;
```

```
@Min(value = 0, message = "Price must be greater than or equal to 0")
```

```
private Double price;
```

```
}
```

CustomerDTO.java

```
package com.example.bookstore.dto;
```

```
import javax.validation.constraints.Email;
```

```
import javax.validation.constraints.NotNull;
```

```
import javax.validation.constraints.Size;
```

```
public class CustomerDTO {
```

```
    private Long id;
```

```
    @NotNull(message = "Name cannot be null")
```

```
    @Size(min = 2, max = 100, message = "Name must be between 2 and 100 characters")
```

```
    private String name;
```

```
    @NotNull(message = "Email cannot be null")
```

```
    @Email(message = "Email should be valid")
```

```
    private String email;
```

```
}
```


Exercise 9: Online Bookstore - Understanding HATEOAS

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-hateoas</artifactId>
</dependency>
```

Updating BookDTO.java

```
package com.example.bookstore.dto;

import org.springframework.hateoas.RepresentationModel;

public class BookDTO extends RepresentationModel<BookDTO> {
    private Long id;
    private String title;
    private String author;
    private Double price;
}
```

CustomerDTO.java

```
package com.example.bookstore.dto;

import org.springframework.hateoas.RepresentationModel;

public class CustomerDTO extends RepresentationModel<CustomerDTO> {
    private Long id;
    private String name;
    private String email;
}
```

BookResourceAssembler.java

```
package com.example.bookstore.assembler;

import com.example.bookstore.controller.BookController;
import com.example.bookstore.dto.BookDTO;
import com.example.bookstore.entity.Book;
import org.springframework.hateoas.EntityModel;
import org.springframework.hateoas.Link;
import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;
import org.springframework.stereotype.Component;

@Component
public class BookResourceAssembler {

    public EntityModel<BookDTO> toModel(BookDTO bookDTO) {

        return EntityModel.of(bookDTO,

WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getBookById(bookDTO.getId())).withSelfRel(),

WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getAllBooks()).withRel("books")

        );
    }
}
```

CustomerResourceAssembler.java

```
package com.example.bookstore.assembler;

import com.example.bookstore.controller.CustomerController;
import com.example.bookstore.dto.CustomerDTO;
import org.springframework.hateoas.EntityModel;
```

```

import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;

import org.springframework.stereotype.Component;

@Component

public class CustomerResourceAssembler {

    public EntityModel<CustomerDTO> toModel(CustomerDTO customerDTO) {

        return EntityModel.of(customerDTO,

WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getCustomerById(customerDTO.getId())).withSelfRel(),

WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getAllCustomers()).withRel("customers")

        );

    }

}

```

Updating BookController.java

```

package com.example.bookstore.controller;

import com.example.bookstore.dto.BookDTO;
import com.example.bookstore.service.BookService;
import com.example.bookstore.assembler.BookResourceAssembler;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.hateoas.EntityModel;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;

```

```

@RestController

```

```
@RequestMapping("/api/books")
```

```
@Validated
```

```
public class BookController {
```

```
    @Autowired
```

```
    private BookService bookService;
```

```
    @Autowired
```

```
    private BookResourceAssembler bookResourceAssembler;
```

```
    @PostMapping
```

```
    public ResponseEntity<EntityModel<BookDTO>> createBook(@Valid @RequestBody BookDTO bookDTO)
    {
        BookDTO createdBook = bookService.createBook(bookDTO);
        EntityModel<BookDTO> resource = bookResourceAssembler.toModel(createdBook);
        return new ResponseEntity<>(resource, HttpStatus.CREATED);
    }
```

```
    @GetMapping("/{id}")
```

```
    public ResponseEntity<EntityModel<BookDTO>> getBookById(@PathVariable Long id) {
        BookDTO bookDTO = bookService.getBookById(id);
        EntityModel<BookDTO> resource = bookResourceAssembler.toModel(bookDTO);
        return new ResponseEntity<>(resource, HttpStatus.OK);
    }
```

```
    @GetMapping
```

```
    public ResponseEntity<List<EntityModel<BookDTO>>> getAllBooks() {
        List<BookDTO> books = bookService.getAllBooks();
        List<EntityModel<BookDTO>> resources = books.stream()
            .map(bookResourceAssembler::toModel)
            .toList();
        return new ResponseEntity<>(resources, HttpStatus.OK);
    }
```

```
@PutMapping("/{id}")
```

```
public ResponseEntity<EntityModel<BookDTO>> updateBook(@PathVariable Long id, @Valid  
@RequestBody BookDTO bookDTO) {
```

```
    BookDTO updatedBook = bookService.updateBook(id, bookDTO);
```

```
    EntityModel<BookDTO> resource = bookResourceAssembler.toModel(updatedBook);
```

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

```
}
```

```
@DeleteMapping("/{id}")
```

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

```
    bookService.deleteBook(id);
```

```
    return new ResponseEntity<>(HttpStatus.NO_CONTENT);
```

```
}
```

```
}
```

CustomerController.java

```
package com.example.bookstore.controller;
```

```
import com.example.bookstore.dto.CustomerDTO;
```

```
import com.example.bookstore.service.CustomerService;
```

```
import com.example.bookstore.assembler.CustomerResourceAssembler;
```

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

```
import org.springframework.hateoas.EntityModel;
```

```
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.validation.annotation.Validated;
```

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

```
import javax.validation.Valid;
```

```
import java.util.List;
```

```
@RestController
```

```
@RequestMapping("/api/customers")
```

```
@Validated
```

```
public class CustomerController {
```

```
    @Autowired
```

```
    private CustomerService customerService;
```

```
    @Autowired
```

```
    private CustomerResourceAssembler customerResourceAssembler;
```

```
    @PostMapping
```

```
    public ResponseEntity<EntityModel<CustomerDTO>> createCustomer(@Valid @RequestBody  
CustomerDTO customerDTO) {
```

```
        CustomerDTO createdCustomer = customerService.createCustomer(customerDTO);
```

```
        EntityModel<CustomerDTO> resource = customerResourceAssembler.toModel(createdCustomer);
```

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

```
    }
```

```
    @GetMapping("/{id}")
```

```
    public ResponseEntity<EntityModel<CustomerDTO>> getCustomerById(@PathVariable Long id) {
```

```
        CustomerDTO customerDTO = customerService.getCustomerById(id);
```

```
        EntityModel<CustomerDTO> resource = customerResourceAssembler.toModel(customerDTO);
```

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

```
    }
```

```
    @GetMapping
```

```
    public ResponseEntity<List<EntityModel<CustomerDTO>>> getAllCustomers() {
```

```
        List<CustomerDTO> customers = customerService.getAllCustomers();
```

```
        List<EntityModel<CustomerDTO>> resources = customers.stream()
```

```
            .map(customerResourceAssembler::toModel)
```

```
            .toList();
```

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

```
    }
```

```
@PutMapping("/{id}")
```

```
public ResponseEntity<EntityModel<CustomerDTO>> updateCustomer(@PathVariable Long id, @Valid  
@RequestBody CustomerDTO customerDTO) {
```

```
    CustomerDTO updatedCustomer = customerService.updateCustomer(id, customerDTO);
```

```
    EntityModel<CustomerDTO> resource = customerResourceAssembler.toModel(updatedCustomer);
```

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

```
}
```

```
@DeleteMapping("/{id}")
```

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

```
    customerService.deleteCustomer(id);
```

```
    return new ResponseEntity<>(HttpStatus.NO_CONTENT);
```

```
}
```

```
}
```

RESPONSE FOR A BOOK:

```
"id": 1, "title": "Effective Java", "author": "Joshua Bloch", "price": 45.0, "links": { "self":  
"http://localhost:8080/api/books/1", "books": "http://localhost:8083/api/books"
```

Exercise 10: Online Bookstore - Configuring Content Negotiation

Adding dependencies:

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-web</artifactId>
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>com.fasterxml.jackson.core</groupId>
```

```
    <artifactId>jackson-databind</artifactId>
```

```
</dependency>
```

```
<dependency>
  <groupId>com.fasterxml.jackson.dataformat</groupId>
  <artifactId>jackson-dataformat-xml</artifactId>
</dependency>
```

WebConfig.java

```
package com.example.bookstore.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ContentNegotiationConfigurer;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void configureContentNegotiation(ContentNegotiationConfigurer configurer) {
        configurer.favorPathExtension(true)
            .ignoreAcceptHeader(false)
            .defaultContentType(MediaType.APPLICATION_JSON)
            .mediaType("json", MediaType.APPLICATION_JSON)
            .mediaType("xml", MediaType.APPLICATION_XML);
    }
}
```

XmlConfig.java

```
package com.example.bookstore.config;

import com.fasterxml.jackson.dataformat.xml.XmlMapper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.converter.xml.Jackson2ObjectMapperBuilder;
```


@Configuration

```
public class XmlConfig {
```

```
    @Bean
```

```
    public XmlMapper xmlMapper() {
```

```
        return Jackson2ObjectMapperBuilder.xml().build();
```

```
    }
```

```
}
```

BookController.java

```
package com.example.bookstore.controller;
```

```
import com.example.bookstore.dto.BookDTO;
```

```
import com.example.bookstore.service.BookService;
```

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

```
import org.springframework.hateoas.EntityModel;
```

```
import org.springframework.http.HttpStatus;
```

```
import org.springframework.http.MediaType;
```

```
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.validation.annotation.Validated;
```

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

```
import javax.validation.Valid;
```

```
import java.util.List;
```

```
@RestController
```

```
@RequestMapping("/api/books")
```

```
@Validated
```

```
public class BookController {
```

```
    @Autowired
```

```
    private BookService bookService;
```

```
@PostMapping(consumes = {MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE}, produces = {MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE})
```

```
public ResponseEntity<EntityModel<BookDTO>> createBook(@Valid @RequestBody BookDTO bookDTO)
{
    BookDTO createdBook = bookService.createBook(bookDTO);
    EntityModel<BookDTO> resource = bookResourceAssembler.toModel(createdBook);
    return new ResponseEntity<>(resource, HttpStatus.CREATED);
}
```

```
@GetMapping(value =("/{id}", produces = {MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE})
```

```
public ResponseEntity<EntityModel<BookDTO>> getBookById(@PathVariable Long id) {
    BookDTO bookDTO = bookService.getBookById(id);
    EntityModel<BookDTO> resource = bookResourceAssembler.toModel(bookDTO);
    return new ResponseEntity<>(resource, HttpStatus.OK);
}
```

```
@GetMapping(produces = {MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE})
```

```
public ResponseEntity<List<EntityModel<BookDTO>>> getAllBooks() {
    List<BookDTO> books = bookService.getAllBooks();
    List<EntityModel<BookDTO>> resources = books.stream()
        .map(bookResourceAssembler::toModel)
        .toList();
    return new ResponseEntity<>(resources, HttpStatus.OK);
}
```

```
@PutMapping(value =("/{id}", consumes = {MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE}, produces = {MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE})
```

```
public ResponseEntity<EntityModel<BookDTO>> updateBook(@PathVariable Long id, @Valid
@RequestBody BookDTO bookDTO) {
    BookDTO updatedBook = bookService.updateBook(id, bookDTO);
    EntityModel<BookDTO> resource = bookResourceAssembler.toModel(updatedBook);
}
```

```

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

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBook(@PathVariable Long id) {
        bookService.deleteBook(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }
}

```

To get a book in json format:

GET /api/books/1 HTTP/1.1

Accept: application/json

To get a book in xml format:

GET /api/books/1 HTTP/1.1

Accept: application/xml

To post a new book:

POST /api/books HTTP/1.1

Content-Type: application/xml

Accept: application/xml

<bookDTO>

<title>Effective Java</title>

<author>Joshua Bloch</author>

<price>45.0</price>

</bookDTO>

Exercise 11: Online Bookstore - Integrating Spring Boot Actuator

Dependency:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

Application.properties

```
management.endpoints.web.exposure.include=*  
management.endpoint.health.show-details=always  
management.endpoint.info.enabled=true  
management.endpoints.web.base-path=/actuator
```

CustomHealthIndicator.java

```
package com.example.bookstore.actuator;  
  
import org.springframework.boot.actuate.health.Health;  
import org.springframework.boot.actuate.health.HealthIndicator;  
import org.springframework.stereotype.Component;  
  
@Component  
public class CustomHealthIndicator implements HealthIndicator {  
  
    @Override  
    public Health health() {  
        boolean isHealthy = checkHealth();  
        if (isHealthy) {  
            return Health.up().withDetail("custom", "All systems operational").build();  
        } else {  
            return Health.down().withDetail("custom", "Service is down").build();  
        }  
    }  
  
    private boolean checkHealth() {  
  
        return true;  
    }  
}
```

CustomMetrics.java

```
package com.example.bookstore.metrics;

import io.micrometer.core.instrument.MeterRegistry;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component

public class CustomMetrics {

    @Autowired

    public CustomMetrics(MeterRegistry meterRegistry) {

        meterRegistry.counter("custom.metric", "type", "example");
    }
}
```

Viewing metrics:

GET /actuator/metrics/custom.metric

Exercise 12: Online Bookstore - Securing RESTful Endpoints with Spring Security

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt</artifactId>
  <version>0.9.1</version>
```

</dependency>

SecurityConfig.java

```
package com.example.bookstore.config;
```

```
import com.example.bookstore.security.JwtAuthenticationFilter;  
import com.example.bookstore.security.JwtAuthenticationEntryPoint;  
import com.example.bookstore.service.UserDetailsServiceImpl;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.builders.WebSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.config.http.SessionCreationPolicy;  
import org.springframework.security.core.userdetails.UserDetailsService;  
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
```

@Configuration

@EnableWebSecurity

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

 @Autowired

```
    private JwtAuthenticationEntryPoint unauthorizedHandler;
```

 @Bean

```
    public JwtAuthenticationFilter jwtAuthenticationFilter() {  
        return new JwtAuthenticationFilter();  
    }
```

 @Override

```
public void configure(WebSecurity web) {  
    web.ignoring().antMatchers("/actuator/**");  
}
```

@Override

```
protected void configure(HttpSecurity http) throws Exception {  
    http.csrf().disable()  
        .authorizeRequests()  
        .antMatchers("/api/auth/**").permitAll()  
        .antMatchers("/api/**").authenticated()  
        .and()  
        .exceptionHandling().authenticationEntryPoint(unauthorizedHandler)  
        .and()  
        .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);  
  
    http.addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.class);  
}
```

@Bean

@Override

```
public UserDetailsService userDetailsServiceBean() throws Exception {  
    return new UserDetailsServiceImpl();  
}  
}
```

JwtTokenUtil.java

```
package com.example.bookstore.security;  
  
import io.jsonwebtoken.Claims;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.stereotype.Component;
```

```
import java.util.Date;
```

```
@Component
```

```
public class JwtTokenUtil {
```

```
    @Value("${jwt.secret}")
```

```
    private String secret;
```

```
    @Value("${jwt.expiration}")
```

```
    private long expiration;
```

```
    public String generateToken(String username) {
```

```
        return Jwts.builder()
```

```
            .setSubject(username)
```

```
            .setIssuedAt(new Date())
```

```
            .setExpiration(new Date(System.currentTimeMillis() + expiration))
```

```
            .signWith(SignatureAlgorithm.HS512, secret)
```

```
            .compact();
```

```
    }
```

```
    public Claims getClaimsFromToken(String token) {
```

```
        return Jwts.parser()
```

```
            .setSigningKey(secret)
```

```
            .parseClaimsJws(token)
```

```
            .getBody();
```

```
    }
```

```
    public String getUsernameFromToken(String token) {
```

```
        return getClaimsFromToken(token).getSubject();
```

```
    }
```



```

public boolean isTokenExpired(String token) {
    return getClaimsFromToken(token).getExpiration().before(new Date());
}

public boolean validateToken(String token, String username) {
    return (username.equals(getUsernameFromToken(token)) && !isTokenExpired(token));
}
}

```

JwtAuthenticationFilter.java

```

package com.example.bookstore.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class JwtAuthenticationFilter extends UsernamePasswordAuthenticationFilter {

    @Autowired
    private JwtTokenUtil jwtTokenUtil;

    @Override
    public void doFilter(HttpServletRequest request, HttpServletResponse response, FilterChain chain)
        throws IOException, ServletException {

        final String requestHeader = request.getHeader("Authorization");

        String username = null;

        String jwtToken = null;

```

```

    if (requestHeader != null && requestHeader.startsWith("Bearer ")) {
        jwtToken = requestHeader.substring(7);
        try {
            username = jwtTokenUtil.getUsernameFromToken(jwtToken);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    if (username != null && SecurityContextHolder.getContext().getAuthentication() == null) {
        if (jwtTokenUtil.validateToken(jwtToken, username)) {

        }
    }

    chain.doFilter(request, response);
}
}

```

JwtAuthenticationEntryPoint.java

```

package com.example.bookstore.security;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component

```

```
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {

    @Override

    public void commence(HttpServletRequest request, HttpServletResponse response,
AuthenticationException authException)

        throws IOException {

        response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");

    }

}
```

AuthController.java

```
package com.example.bookstore.controller;

import com.example.bookstore.security.JwtTokenUtil;
import com.example.bookstore.service.UserDetailsServiceImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/auth")
public class AuthController {

    @Autowired

    private AuthenticationManager authenticationManager;

    @Autowired

    private JwtTokenUtil jwtTokenUtil;

    @Autowired

    private UserDetailsServiceImpl userDetailsService;
```

```

@PostMapping("/login")

public ResponseEntity<?> authenticateUser(@RequestParam String username, @RequestParam String
password) {

    Authentication authentication = authenticationManager.authenticate(

        new UsernamePasswordAuthenticationToken(username, password)

    );

    String token = jwtTokenUtil.generateToken(username);

    return ResponseEntity.ok().body("Bearer " + token);

}
}

```

CorsConfig.java

```

package com.example.bookstore.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

```

```

@Configuration

```

```

public class CorsConfig implements WebMvcConfigurer {

```

```

    @Bean

```

```

    public WebMvcConfigurer corsConfigurer() {

```

```

        return new WebMvcConfigurer() {

```

```

            @Override

```

```

            public void addCorsMappings(CorsRegistry registry) {

```

```

                registry.addMapping("/**")

```

```

                    .allowedOrigins("http://localhost:3000") // Adjust as needed

```

```

                    .allowedMethods("GET", "POST", "PUT", "DELETE")

```

```

                    .allowedHeaders("*");

```

```

            }

```

```

        };

```

```
}  
}
```

Exercise 13: Online Bookstore - Unit Testing REST Controllers

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-test</artifactId>  
  <scope>test</scope>  
</dependency>  
  
<dependency>  
  <groupId>org.mockito</groupId>  
  <artifactId>mockito-core</artifactId>  
  <scope>test</scope>  
</dependency>  
  
<dependency>  
  <groupId>org.mockito</groupId>  
  <artifactId>mockito-junit-jupiter</artifactId>  
  <scope>test</scope>  
</dependency>
```

BookControllerTest.java

```
package com.example.bookstore.controller;  
  
import com.example.bookstore.dto.BookDTO;  
import com.example.bookstore.service.BookService;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.MockitoAnnotations;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;  
import org.springframework.http.MediaType;
```

```
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

import static org.mockito.Mockito.*;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;

@WebMvcTest(BookController.class)
public class BookControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Mock
    private BookService bookService;

    @InjectMocks
    private BookController bookController;

    @BeforeEach
    void setUp() {
        MockitoAnnotations.openMocks(this);
    }

    @Test
    void testCreateBook() throws Exception {
        BookDTO bookDTO = new BookDTO("Effective Java", "Joshua Bloch", 45.0);
        when(bookService.createBook(any(BookDTO.class))).thenReturn(bookDTO);

        mockMvc.perform(MockMvcRequestBuilders.post("/api/books")
            .contentType(MediaType.APPLICATION_JSON)
```

```

        .content("{\"title\":\"Effective Java\",\"author\":\"Joshua Bloch\",\"price\":45.0}"))
        .andExpect(status().isCreated())
        .andExpect(jsonPath("$.title").value("Effective Java"))
        .andExpect(jsonPath("$.author").value("Joshua Bloch"))
        .andExpect(jsonPath("$.price").value(45.0));

    verify(bookService, times(1)).createBook(any(BookDTO.class));
}

```

@Test

void testGetBookById() throws Exception {

```

    BookDTO bookDTO = new BookDTO("Effective Java", "Joshua Bloch", 45.0);
    when(bookService.getBookById(1L)).thenReturn(bookDTO);

```

```

    mockMvc.perform(MockMvcRequestBuilders.get("/api/books/1")

```

```

        .accept(MediaType.APPLICATION_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.title").value("Effective Java"))
        .andExpect(jsonPath("$.author").value("Joshua Bloch"))
        .andExpect(jsonPath("$.price").value(45.0));

```

```

    verify(bookService, times(1)).getBookById(1L);

```

```

}

```

@Test

void testUpdateBook() throws Exception {

```

    BookDTO bookDTO = new BookDTO("Effective Java", "Joshua Bloch", 45.0);
    when(bookService.updateBook(eq(1L), any(BookDTO.class))).thenReturn(bookDTO);

```

```

    mockMvc.perform(MockMvcRequestBuilders.put("/api/books/1")

```

```

        .contentType(MediaType.APPLICATION_JSON)
        .content("{\"title\":\"Effective Java\",\"author\":\"Joshua Bloch\",\"price\":45.0}"))

```

```
.andExpect(status().isOk())  
.andExpect(jsonPath("$.title").value("Effective Java"))  
.andExpect(jsonPath("$.author").value("Joshua Bloch"))  
.andExpect(jsonPath("$.price").value(45.0));
```

```
verify(bookService, times(1)).updateBook(eq(1L), any(BookDTO.class));  
}
```

@Test

```
void testDeleteBook() throws Exception {  
    mockMvc.perform(MockMvcRequestBuilders.delete("/api/books/1"))  
        .andExpect(status().isNoContent());  
}
```

```
verify(bookService, times(1)).deleteBook(1L);  
}  
}
```

Test POST Request:

MockMvcRequestBuilders.post() to test POST requests, including setting the content type and request body.

Test GET Request:

MockMvcRequestBuilders.get() to test GET requests and validate the response.

Test PUT Request:

MockMvcRequestBuilders.put() to test PUT requests and ensure the updated data is correct.

Test DELETE Request:

MockMvcRequestBuilders.delete() to test DELETE requests and ensure the resource is removed.

Exercise 14: Online Bookstore - Integration Testing for REST Services

<dependency>

<groupId>org.springframework.boot</groupId>


```
<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

    <groupId>com.h2database</groupId>

    <artifactId>h2</artifactId>

    <scope>test</scope>

</dependency>
```

BookControllerIntegrationTest.java

```
package com.example.bookstore.controller;

import com.example.bookstore.dto.BookDTO;
import com.example.bookstore.service.BookService;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;

@SpringBootTest
@ActiveProfiles("test") // Use application-test.properties for configurations

public class BookControllerIntegrationTest {
```

@Autowired

private MockMvc mockMvc;

@Autowired

private BookService bookService;

@BeforeEach

void setUp() {

 // Set up any preconditions or data needed before tests

}

@Test

void testCreateBook() throws Exception {

 BookDTO bookDTO = new BookDTO("Effective Java", "Joshua Bloch", 45.0);

 mockMvc.perform(MockMvcRequestBuilders.post("/api/books")

 .contentType(MediaType.APPLICATION_JSON)

 .content("{\"title\":\"Effective Java\",\"author\":\"Joshua Bloch\",\"price\":45.0}"))

 .andExpect(status().isCreated())

 .andExpect(jsonPath("\$.title").value("Effective Java"))

 .andExpect(jsonPath("\$.author").value("Joshua Bloch"))

 .andExpect(jsonPath("\$.price").value(45.0));

}

@Test

void testGetBookById() throws Exception {

 mockMvc.perform(MockMvcRequestBuilders.get("/api/books/1")

 .accept(MediaType.APPLICATION_JSON))

 .andExpect(status().isOk())

 .andExpect(jsonPath("\$.title").value("Effective Java"))

 .andExpect(jsonPath("\$.author").value("Joshua Bloch"))

 .andExpect(jsonPath("\$.price").value(45.0));

```

}

@Test
void testUpdateBook() throws Exception {
    mockMvc.perform(MockMvcRequestBuilders.put("/api/books/1")
        .contentType(MediaType.APPLICATION_JSON)
        .content("{\"title\":\"Effective Java\",\"author\":\"Joshua Bloch\",\"price\":45.0}"))
        .andExpect(status().isOk())
        .andExpect(jsonPath("$.title").value("Effective Java"))
        .andExpect(jsonPath("$.author").value("Joshua Bloch"))
        .andExpect(jsonPath("$.price").value(45.0));
}

```

```

@Test
void testDeleteBook() throws Exception {
    mockMvc.perform(MockMvcRequestBuilders.delete("/api/books/1"))
        .andExpect(status().isNoContent());
}
}

```

application-test.properties

```

spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=create-drop

```

Scenario 15: Online Bookstore - API Documentation with Swagger

```

<dependency>
    <groupId>org.springdoc</groupId>

```

```
<artifactId>springdoc-openapi-ui</artifactId>
<version>2.2.0</version> <!-- Check for the latest version -->
</dependency>
```

BookController.java

```
package com.example.bookstore.controller;

import com.example.bookstore.dto.BookDTO;
import com.example.bookstore.service.BookService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.responses.ApiResponses;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/books")
public class BookController {

    @Autowired
    private BookService bookService;

    @Operation(summary = "Create a new book", description = "Add a new book to the bookstore")
    @ApiResponses(value = {
        @ApiResponse(responseCode = "201", description = "Book created successfully"),
        @ApiResponse(responseCode = "400", description = "Invalid input")
    })
    @PostMapping
    public ResponseEntity<BookDTO> createBook(
```

```
    @io.swagger.v3.oas.annotations.parameters.RequestBody(description = "Book details", required = true) @RequestBody BookDTO bookDTO) {  
        BookDTO createdBook = bookService.createBook(bookDTO);  
        return new ResponseEntity<>(createdBook, HttpStatus.CREATED);  
    }
```

```
@Operation(summary = "Get a book by ID", description = "Retrieve details of a book using its ID")
```

```
@ApiResponses(value = {
```

```
    @ApiResponse(responseCode = "200", description = "Book found"),
```

```
    @ApiResponse(responseCode = "404", description = "Book not found")
```

```
})
```

```
@GetMapping("/{id}")
```

```
public ResponseEntity<BookDTO> getBookById(  
    @Parameter(description = "ID of the book to be retrieved", required = true) @PathVariable Long id)
```

```
{  
    BookDTO bookDTO = bookService.getBookById(id);  
    return ResponseEntity.ok(bookDTO);  
}
```

```
@Operation(summary = "Update a book", description = "Update details of an existing book")
```

```
@ApiResponses(value = {
```

```
    @ApiResponse(responseCode = "200", description = "Book updated successfully"),
```

```
    @ApiResponse(responseCode = "400", description = "Invalid input"),
```

```
    @ApiResponse(responseCode = "404", description = "Book not found")
```

```
})
```

```
@PutMapping("/{id}")
```

```
public ResponseEntity<BookDTO> updateBook(  
    @Parameter(description = "ID of the book to be updated", required = true) @PathVariable Long id,
```

```
    @RequestBody BookDTO bookDTO) {
```

```
    BookDTO updatedBook = bookService.updateBook(id, bookDTO);  
    return ResponseEntity.ok(updatedBook);  
}
```

```

@Operation(summary = "Delete a book", description = "Remove a book from the bookstore")
@ApiResponses(value = {
    @ApiResponse(responseCode = "204", description = "Book deleted successfully"),
    @ApiResponse(responseCode = "404", description = "Book not found")
})
>DeleteMapping("/{id}")
public ResponseEntity<Void> deleteBook(
    @Parameter(description = "ID of the book to be deleted", required = true) @PathVariable Long id) {
    bookService.deleteBook(id);
    return ResponseEntity.noContent().build();
}
}

```

<http://localhost:8083/swagger-ui.html>

SwaggerConfig.java

```

package com.example.bookstore.config;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SwaggerConfig {

    @Bean
    public OpenAPI customOpenAPI() {
        return new OpenAPI()
            .info(new Info()
                .title("Online Bookstore API")
                .version("1.0.0")
            )
    }
}

```

```
.description("API documentation for the Online Bookstore application"));
```

```
}
```

```
}
```