Spring Part 2 - Assessment No: 1

Spring REST

Name: Gaurav Nandgopal Deshmukh

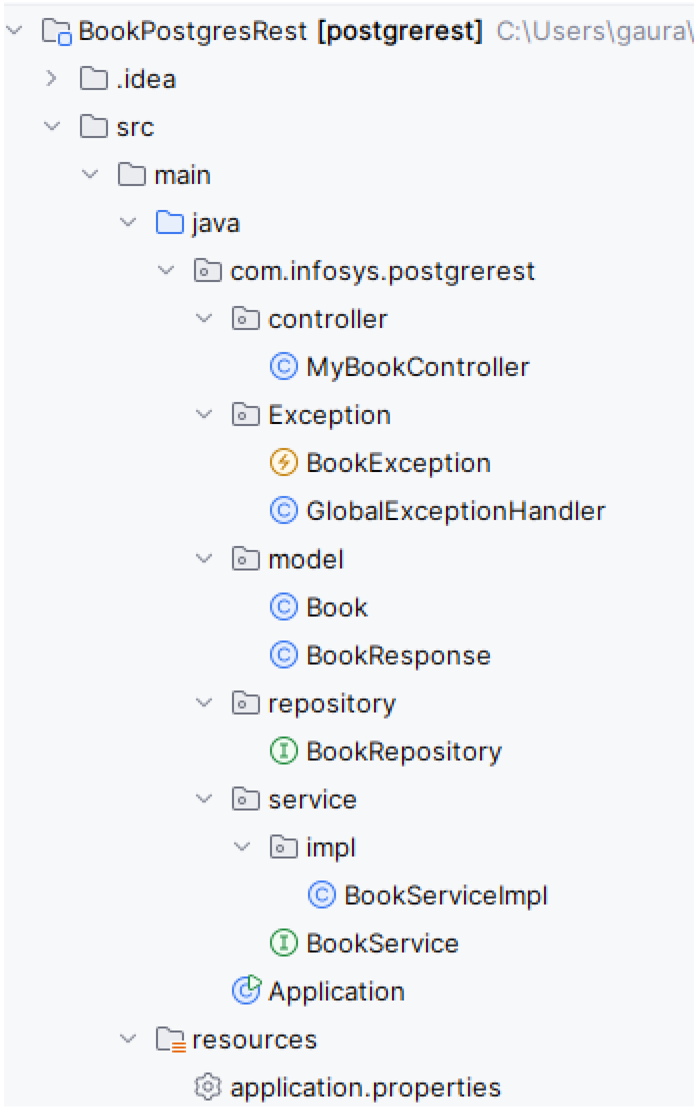
USN: 72232433C

Question 1. Create a RESTful web service using Spring Boot that manages a collection of "Books" with basic CRUD (Create, Read, Update, Delete) operations.

Implement a REST API with the following endpoints:

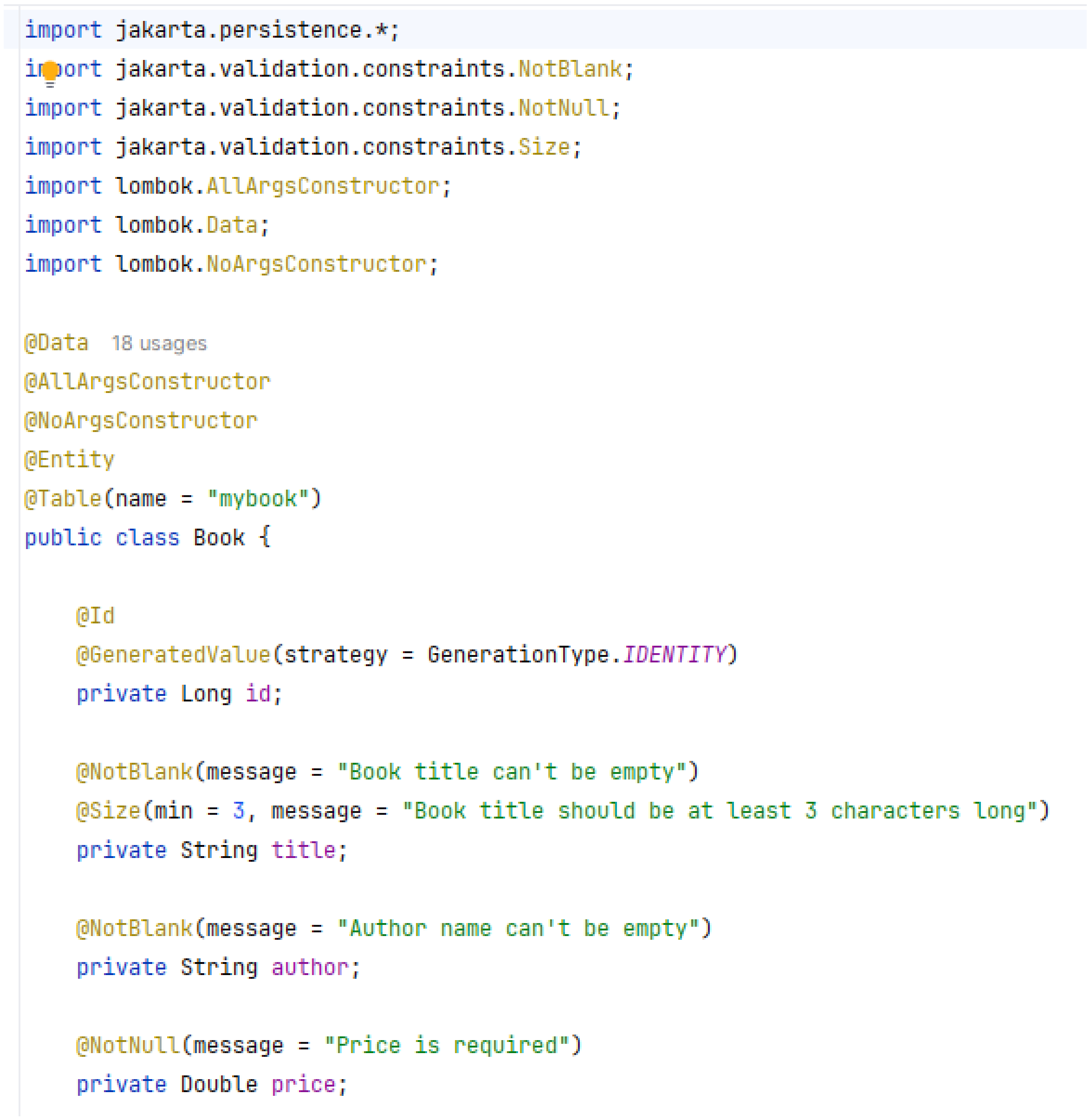
1. **GET /books** → Retrieve all books
2. **GET /books/{id}** → Retrieve a specific book by ID
3. **POST /books** → Add a new book
4. **PUT /books/{id}** → Update an existing book
5. **DELETE /books/{id}** → Delete a book

Folder Structure:



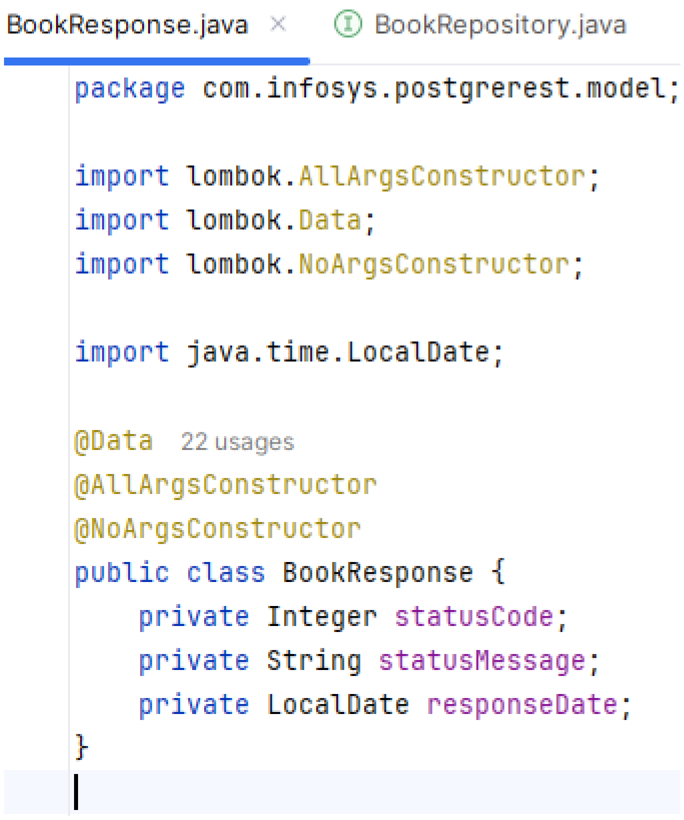
```
BookPostgresRest [postgrerest] C:\Users\gaura\
  .idea
  src
    main
      java
        com.infosys.postgrerest
          controller
            MyBookController
          Exception
            BookException
            GlobalExceptionHandler
          model
            Book
            BookResponse
          repository
            BookRepository
          service
            impl
              BookServiceImpl
            BookService
          Application
      resources
        application.properties
```

Book.java:



```java
import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotNull;
import jakarta.validation.constraints.Size;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data   18 usages
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "mybook")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank(message = "Book title can't be empty")
    @Size(min = 3, message = "Book title should be at least 3 characters long")
    private String title;

    @NotBlank(message = "Author name can't be empty")
    private String author;

    @NotNull(message = "Price is required")
    private Double price;
```

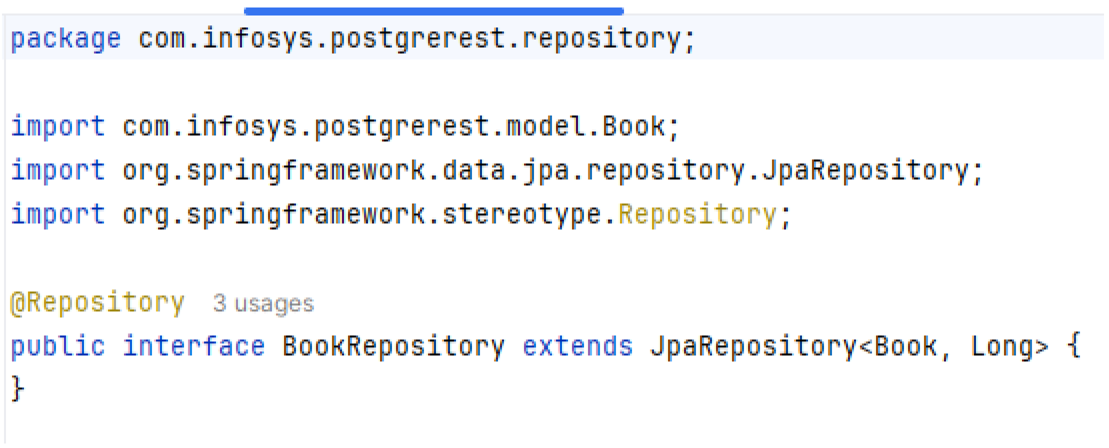
BookResponse.java:



BookResponse.java × BookRepository.java

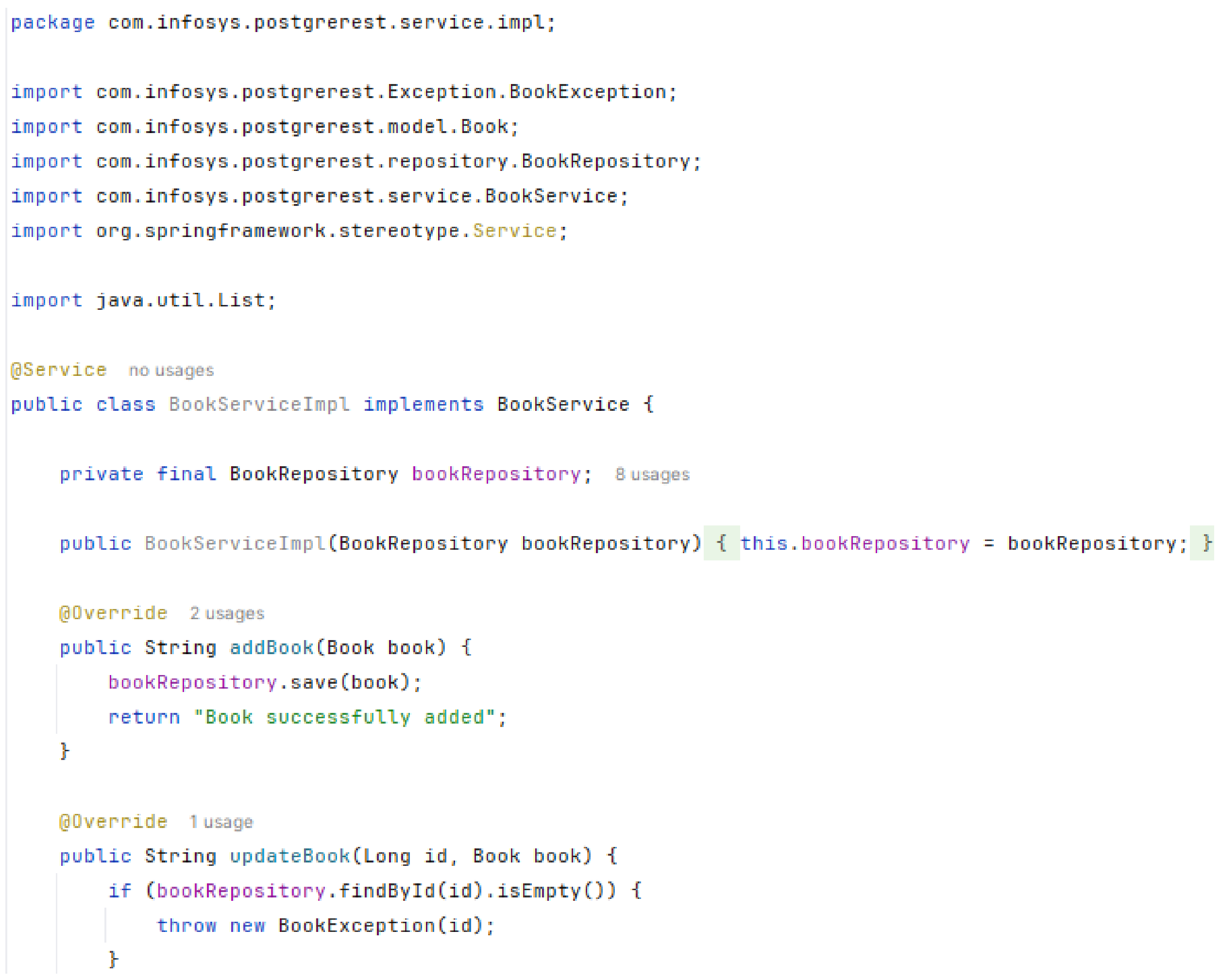
```java
package com.infosys.postgrerest.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDate;

@Data  22 usages
@AllArgsConstructor
@NoArgsConstructor
public class BookResponse {
    private Integer statusCode;
    private String statusMessage;
    private LocalDate responseDate;
}
```

BookRepository:

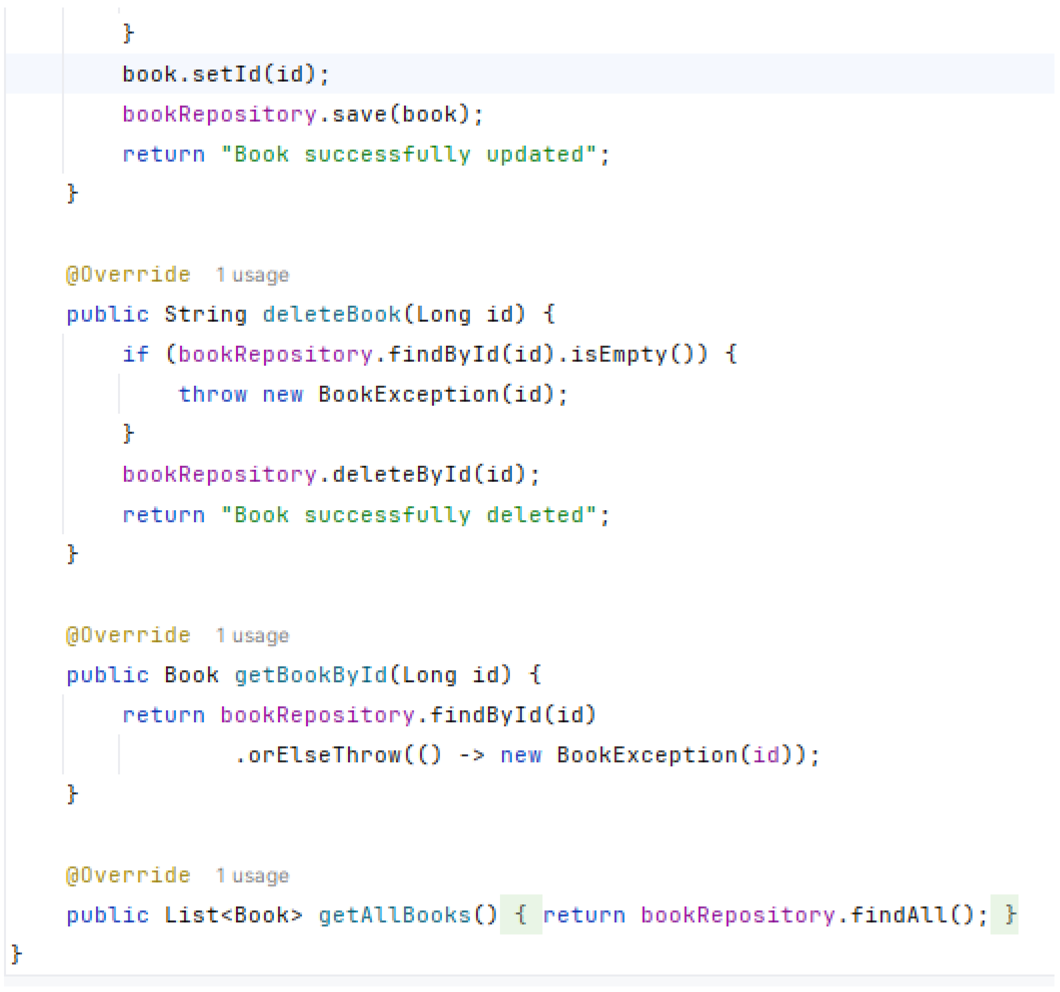


```java
package com.infosys.postgrerest.repository;

import com.infosys.postgrerest.model.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository  3 usages
public interface BookRepository extends JpaRepository<Book, Long> {
}
```

BookServiceImpl:

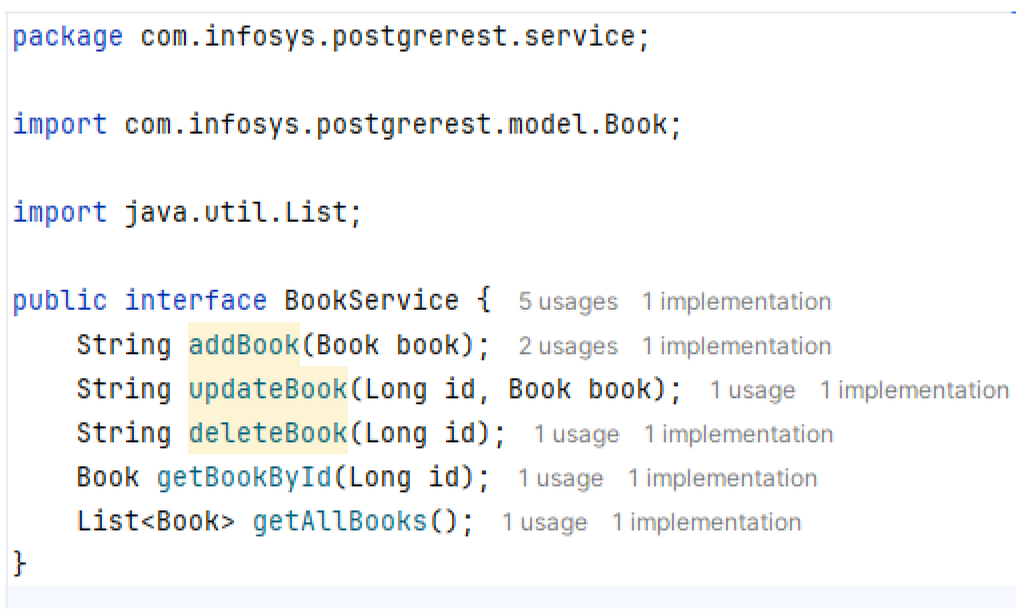


```java
package com.infosys.postgrerest.service.impl;

import com.infosys.postgrerest.Exception.BookException;
import com.infosys.postgrerest.model.Book;
import com.infosys.postgrerest.repository.BookRepository;
import com.infosys.postgrerest.service.BookService;
import org.springframework.stereotype.Service;

import java.util.List;

@Service  no usages
public class BookServiceImpl implements BookService {

    private final BookRepository bookRepository;  8 usages

    public BookServiceImpl(BookRepository bookRepository) { this.bookRepository = bookRepository; }

    @Override  2 usages
    public String addBook(Book book) {
        bookRepository.save(book);
        return "Book successfully added";
    }

    @Override  1 usage
    public String updateBook(Long id, Book book) {
        if (bookRepository.findById(id).isEmpty()) {
            throw new BookException(id);
        }
```



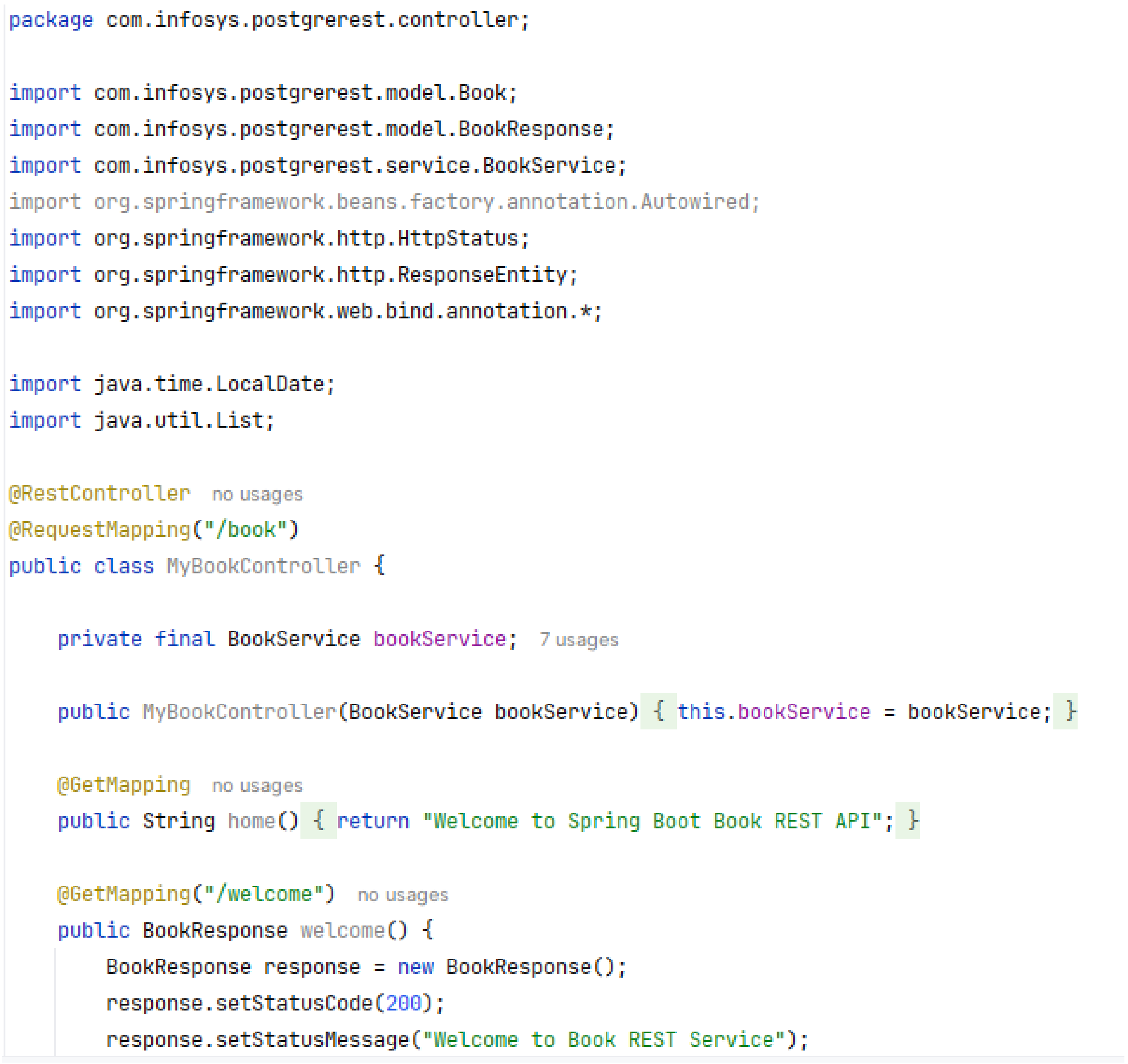
```
        }
        book.setId(id);
        bookRepository.save(book);
        return "Book successfully updated";
    }

    @Override  1 usage
    public String deleteBook(Long id) {
        if (bookRepository.findById(id).isEmpty()) {
            throw new BookException(id);
        }
        bookRepository.deleteById(id);
        return "Book successfully deleted";
    }

    @Override  1 usage
    public Book getBookById(Long id) {
        return bookRepository.findById(id)
                .orElseThrow(() -> new BookException(id));
    }

    @Override  1 usage
    public List<Book> getAllBooks() { return bookRepository.findAll(); }
}
```

BookService:

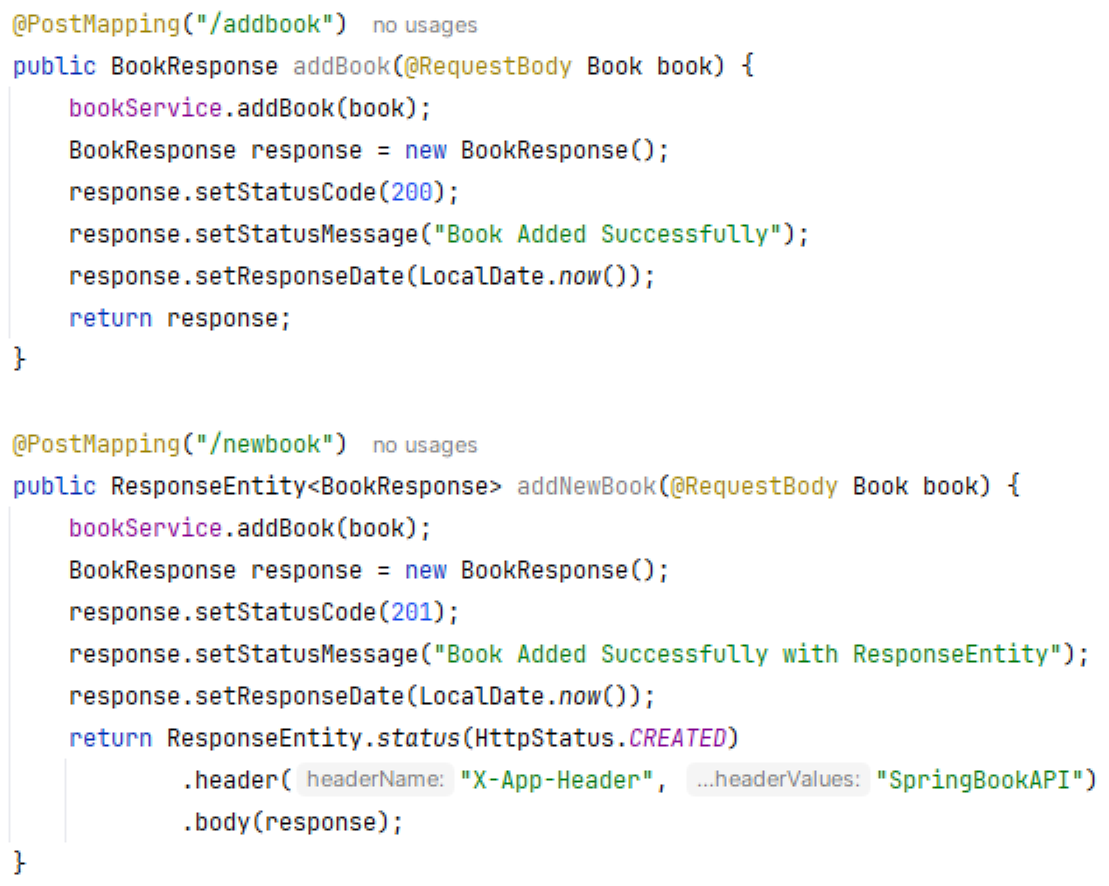


```
package com.infosys.postgrerest.service;

import com.infosys.postgrerest.model.Book;

import java.util.List;

public interface BookService {  5 usages  1 implementation
    String addBook(Book book);  2 usages  1 implementation
    String updateBook(Long id, Book book);  1 usage  1 implementation
    String deleteBook(Long id);  1 usage  1 implementation
    Book getBookById(Long id);  1 usage  1 implementation
    List<Book> getAllBooks();  1 usage  1 implementation
}
```

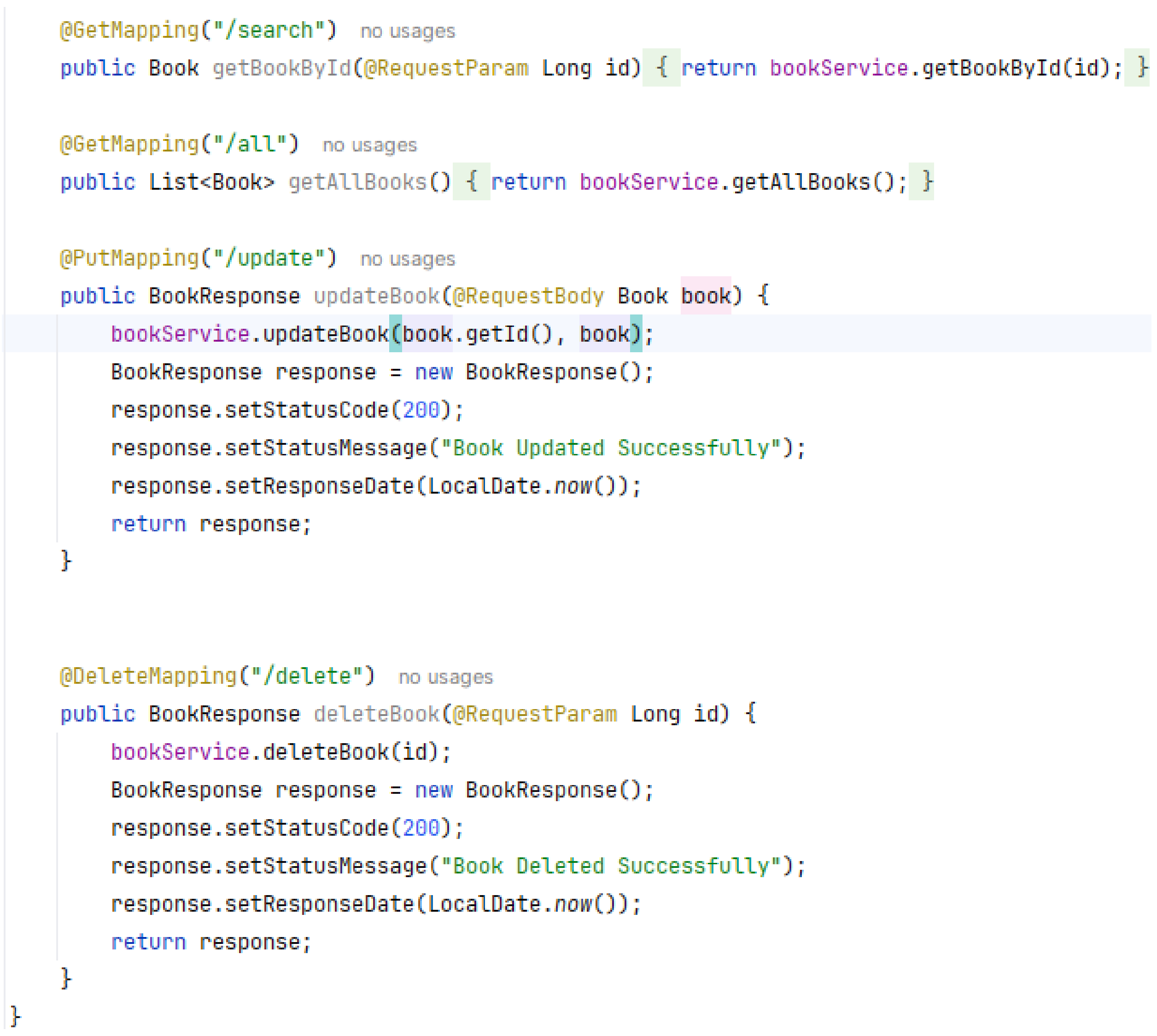
MyBookController:



```java
package com.infosys.postgrerest.controller;

import com.infosys.postgrerest.model.Book;
import com.infosys.postgrerest.model.BookResponse;
import com.infosys.postgrerest.service.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.time.LocalDate;
import java.util.List;

@RestController  no usages
@RequestMapping("/book")
public class MyBookController {

    private final BookService bookService;  7 usages

    public MyBookController(BookService bookService) { this.bookService = bookService; }

    @GetMapping  no usages
    public String home() { return "Welcome to Spring Boot Book REST API"; }

    @GetMapping("/welcome")  no usages
    public BookResponse welcome() {
        BookResponse response = new BookResponse();
        response.setStatusCode(200);
        response.setStatusMessage("Welcome to Book REST Service");
```

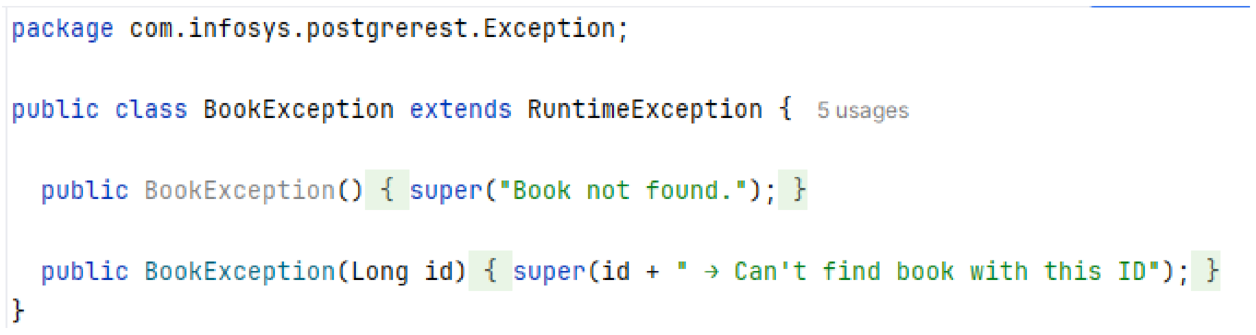


```
@PostMapping("/addbook")  no usages
public BookResponse addBook(@RequestBody Book book) {
    bookService.addBook(book);
    BookResponse response = new BookResponse();
    response.setStatusCode(200);
    response.setStatusMessage("Book Added Successfully");
    response.setResponseDate(LocalDate.now());
    return response;
}

@PostMapping("/newbook")  no usages
public ResponseEntity<BookResponse> addNewBook(@RequestBody Book book) {
    bookService.addBook(book);
    BookResponse response = new BookResponse();
    response.setStatusCode(201);
    response.setStatusMessage("Book Added Successfully with ResponseEntity");
    response.setResponseDate(LocalDate.now());
    return ResponseEntity.status(HttpStatus.CREATED)
            .header( headerName: "X-App-Header",  ...headerValues: "SpringBookAPI")
            .body(response);
}
```



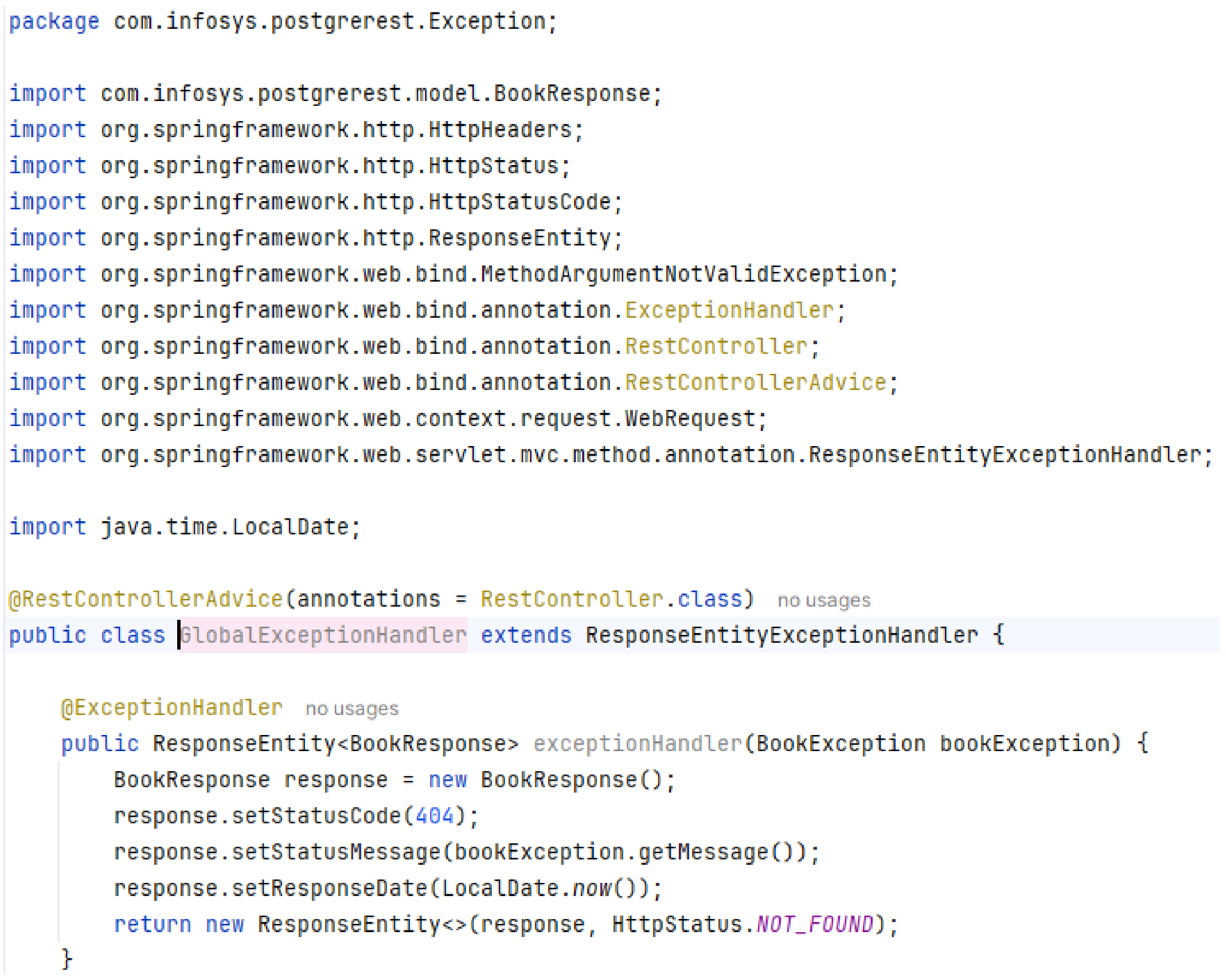
```java
    @GetMapping("/search")  no usages
    public Book getBookById(@RequestParam Long id) { return bookService.getBookById(id); }

    @GetMapping("/all")  no usages
    public List<Book> getAllBooks() { return bookService.getAllBooks(); }

    @PutMapping("/update")  no usages
    public BookResponse updateBook(@RequestBody Book book) {
        bookService.updateBook(book.getId(), book);
        BookResponse response = new BookResponse();
        response.setStatusCode(200);
        response.setStatusMessage("Book Updated Successfully");
        response.setResponseDate(LocalDate.now());
        return response;
    }


    @DeleteMapping("/delete")  no usages
    public BookResponse deleteBook(@RequestParam Long id) {
        bookService.deleteBook(id);
        BookResponse response = new BookResponse();
        response.setStatusCode(200);
        response.setStatusMessage("Book Deleted Successfully");
        response.setResponseDate(LocalDate.now());
        return response;
    }
}
```

BookException:

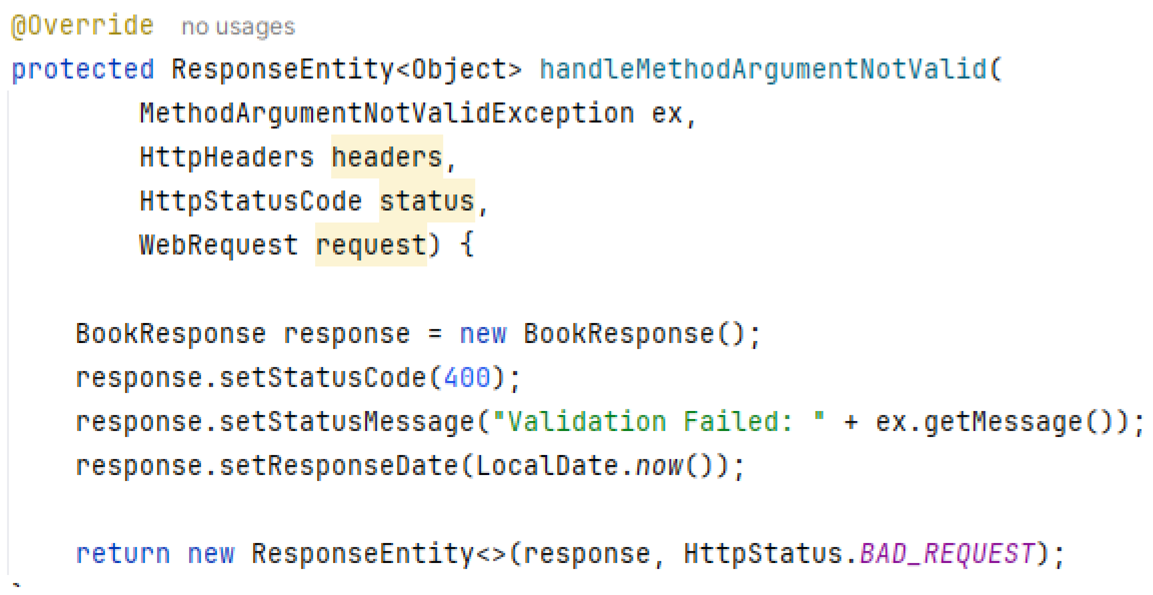


```java
package com.infosys.postgrerest.Exception;

public class BookException extends RuntimeException {  5 usages

  public BookException() { super("Book not found."); }

  public BookException(Long id) { super(id + " → Can't find book with this ID"); }
}
```

GlobalExceptionHandler:

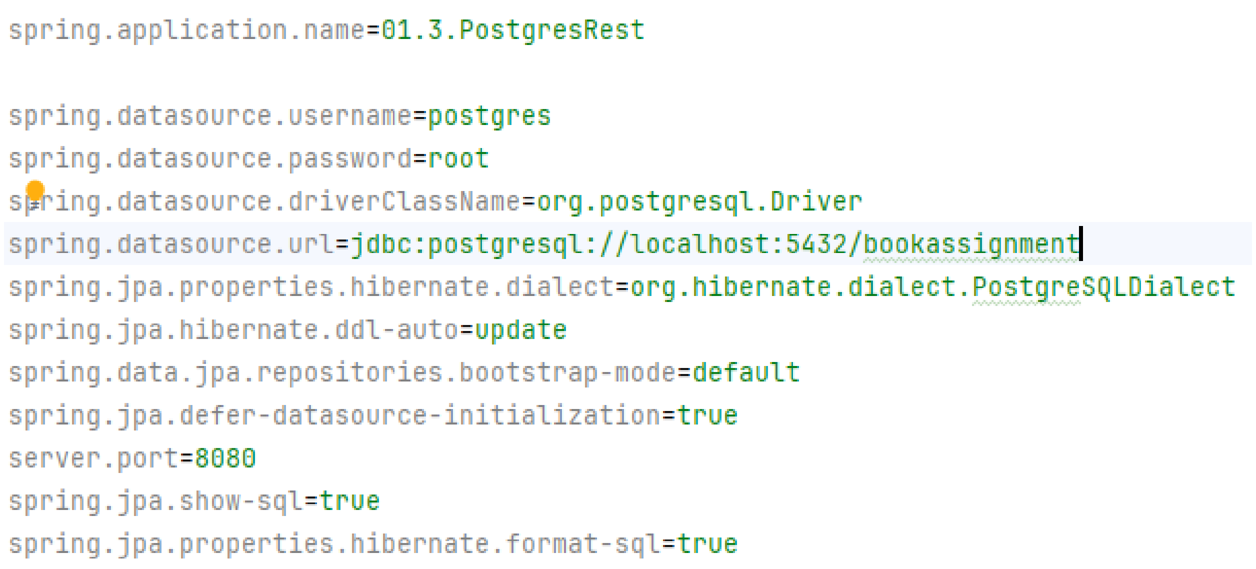


```
package com.infosys.postgrerest.Exception;

import com.infosys.postgrerest.model.BookResponse;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.HttpStatusCode;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.time.LocalDate;

@RestControllerAdvice(annotations = RestController.class)  no usages
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

    @ExceptionHandler  no usages
    public ResponseEntity<BookResponse> exceptionHandler(BookException bookException) {
        BookResponse response = new BookResponse();
        response.setStatusCode(404);
        response.setStatusMessage(bookException.getMessage());
        response.setResponseDate(LocalDate.now());
        return new ResponseEntity<>(response, HttpStatus.NOT_FOUND);
    }
```



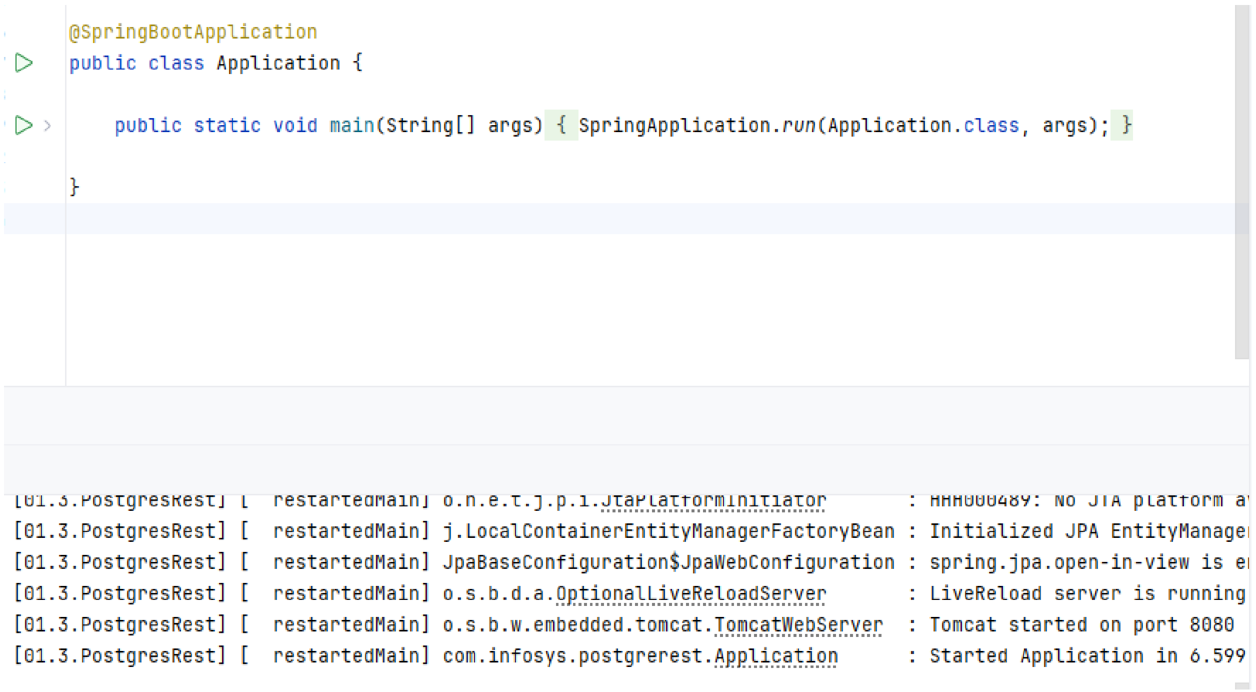
```
    @Override  no usages
    protected ResponseEntity<Object> handleMethodArgumentNotValid(
            MethodArgumentNotValidException ex,
            HttpHeaders headers,
            HttpStatusCode status,
            WebRequest request) {

        BookResponse response = new BookResponse();
        response.setStatusCode(400);
        response.setStatusMessage("Validation Failed: " + ex.getMessage());
        response.setResponseDate(LocalDate.now());

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

Application.properties:



```
spring.application.name=01.3.PostgresRest

spring.datasource.username=postgres
spring.datasource.password=root
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/bookassignment
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.data.jpa.repositories.bootstrap-mode=default
spring.jpa.defer-datasource-initialization=true
server.port=8080
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format-sql=true
```

Application.java:

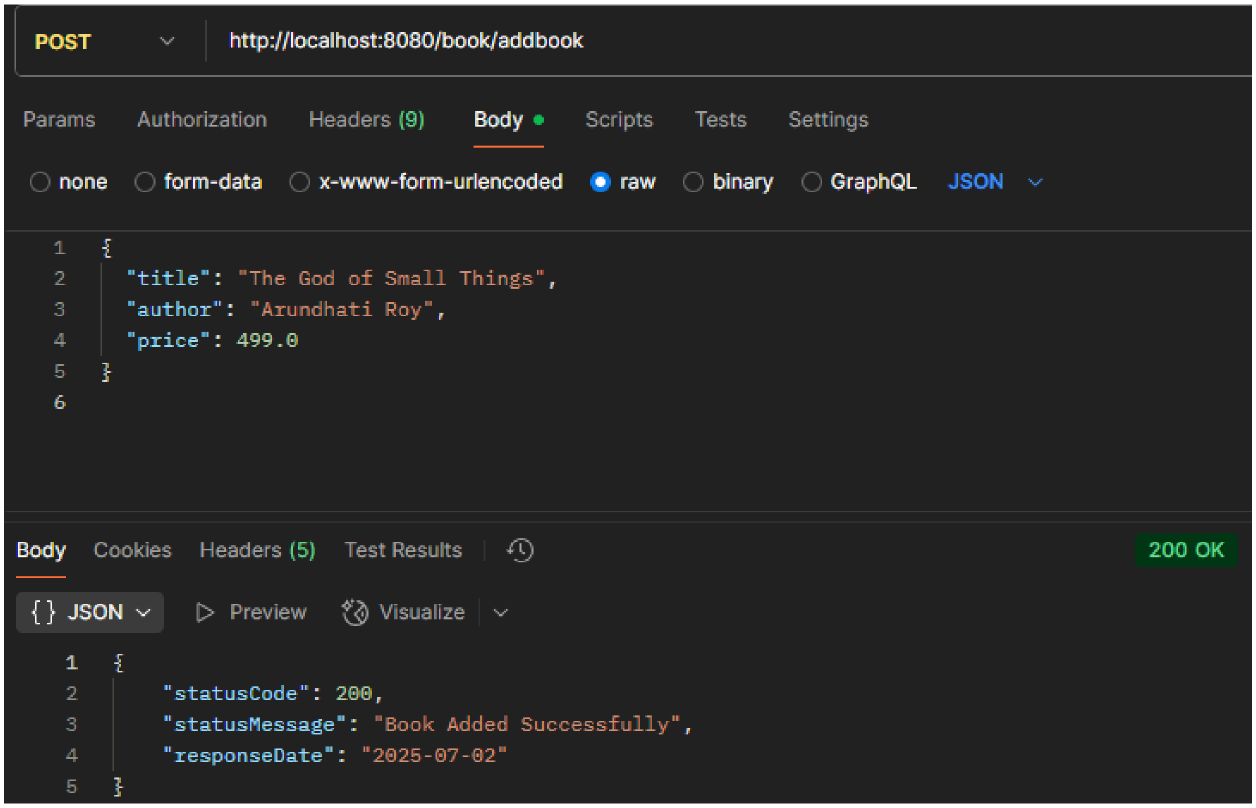


```
@SpringBootApplication
public class Application {

    public static void main(String[] args) { SpringApplication.run(Application.class, args); }

}
```

```
[01.3.PostgresRest] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform a
[01.3.PostgresRest] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManage
[01.3.PostgresRest] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is e
[01.3.PostgresRest] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running
[01.3.PostgresRest] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080
[01.3.PostgresRest] [  restartedMain] com.infosys.postgrerest.Application      : Started Application in 6.599
```

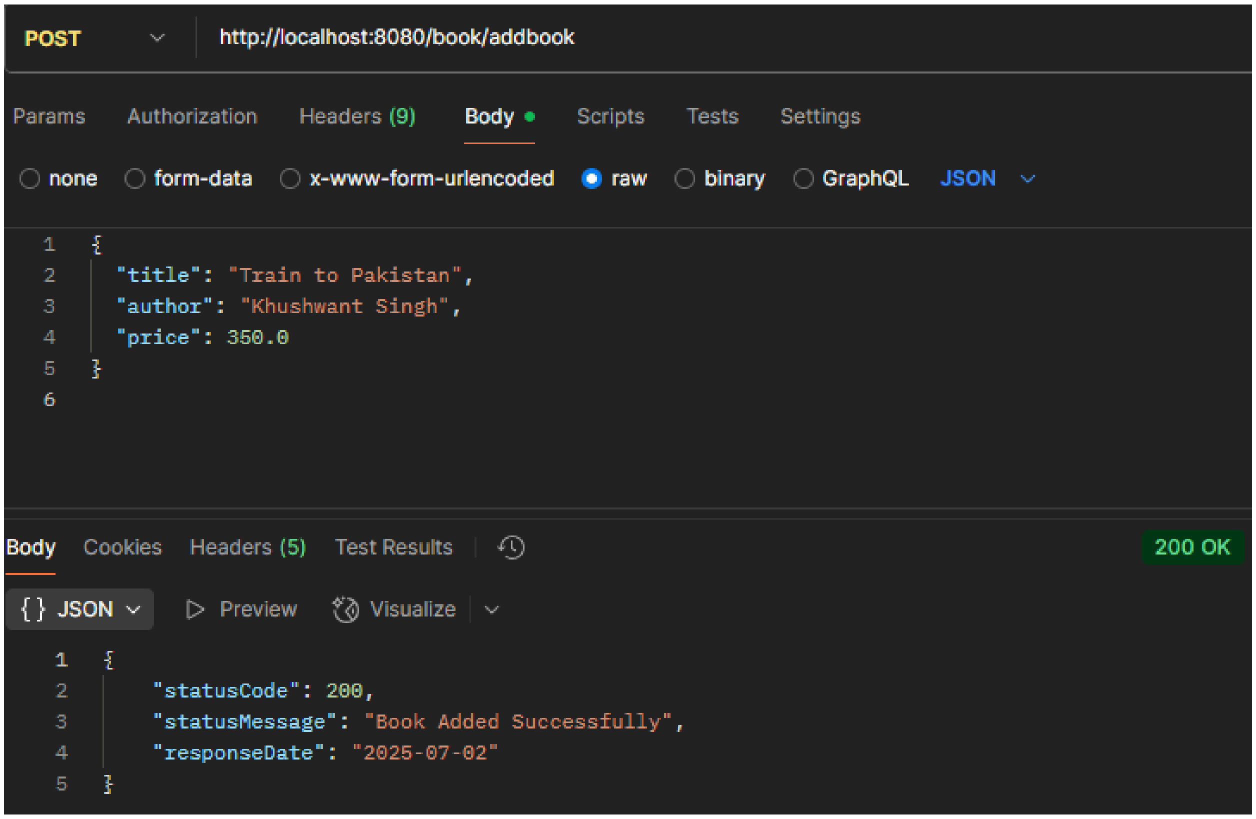
OUTPUT:

3. POST /books → Add a new book



```
{
  "title": "The God of Small Things",
  "author": "Arundhati Roy",
  "price": 499.0
}

```

```
{
    "statusCode": 200,
    "statusMessage": "Book Added Successfully",
    "responseDate": "2025-07-02"
}
```


POST http://localhost:8080/book/addbook

Params Authorization Headers (9) Body Scripts Tests Settings

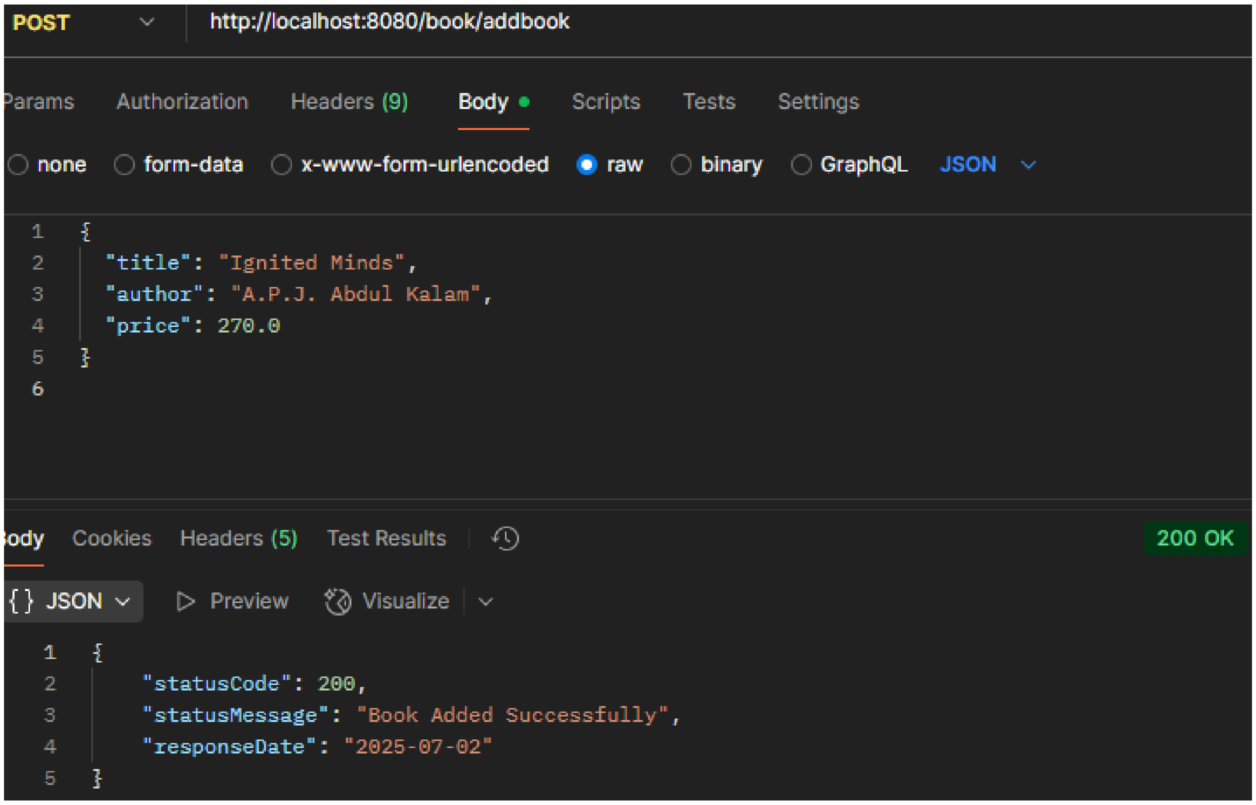
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
  "title": "Train to Pakistan",
  "author": "Khushwant Singh",
  "price": 350.0
}
```

Body Cookies Headers (5) Test Results 200 OK

{ } JSON Preview Visualize

```
{
    "statusCode": 200,
    "statusMessage": "Book Added Successfully",
    "responseDate": "2025-07-02"
}
```


POST http://localhost:8080/book/addbook

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

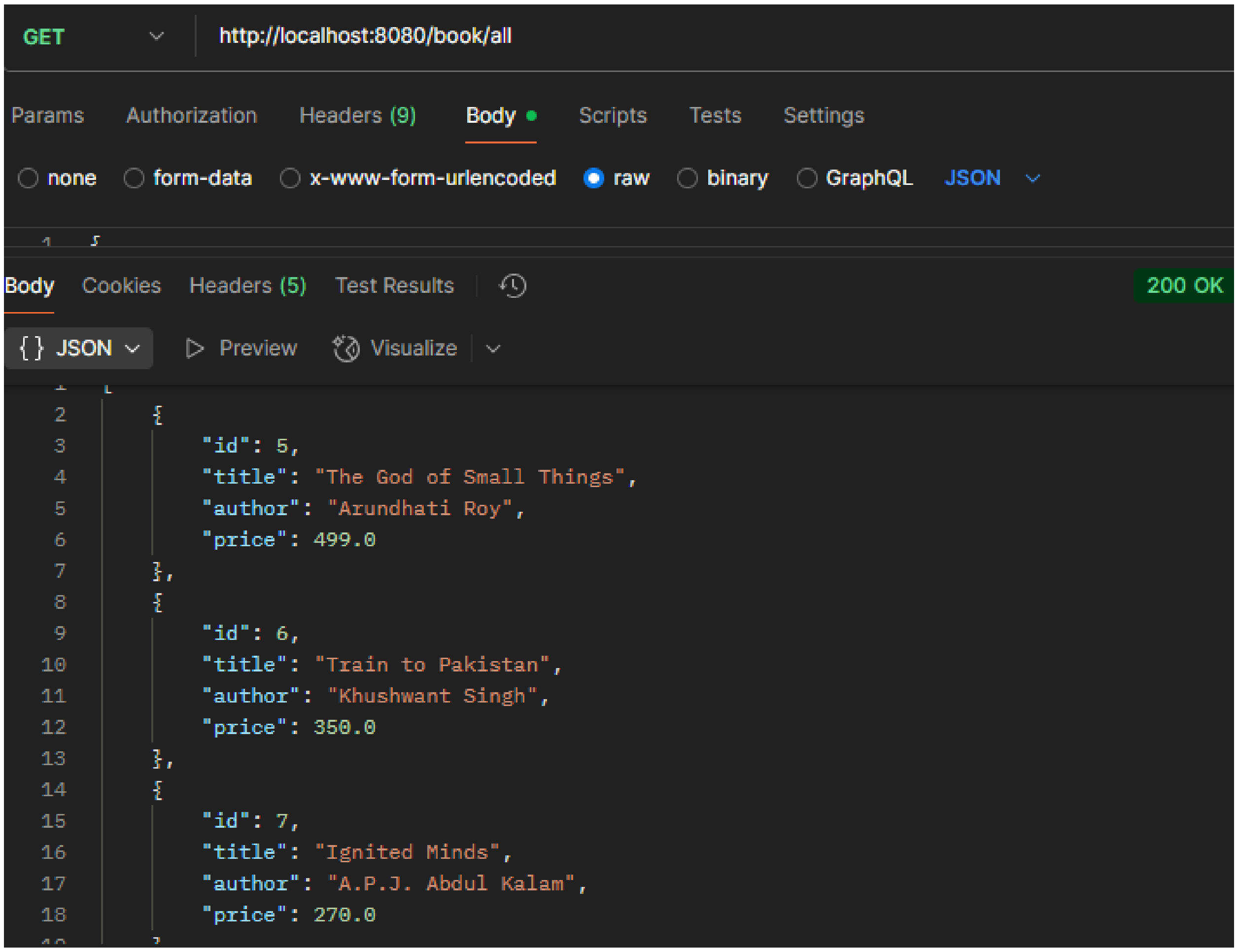
```
{
  "title": "Ignited Minds",
  "author": "A.P.J. Abdul Kalam",
  "price": 270.0
}
```

Body Cookies Headers (5) Test Results 200 OK

{ } JSON Preview Visualize

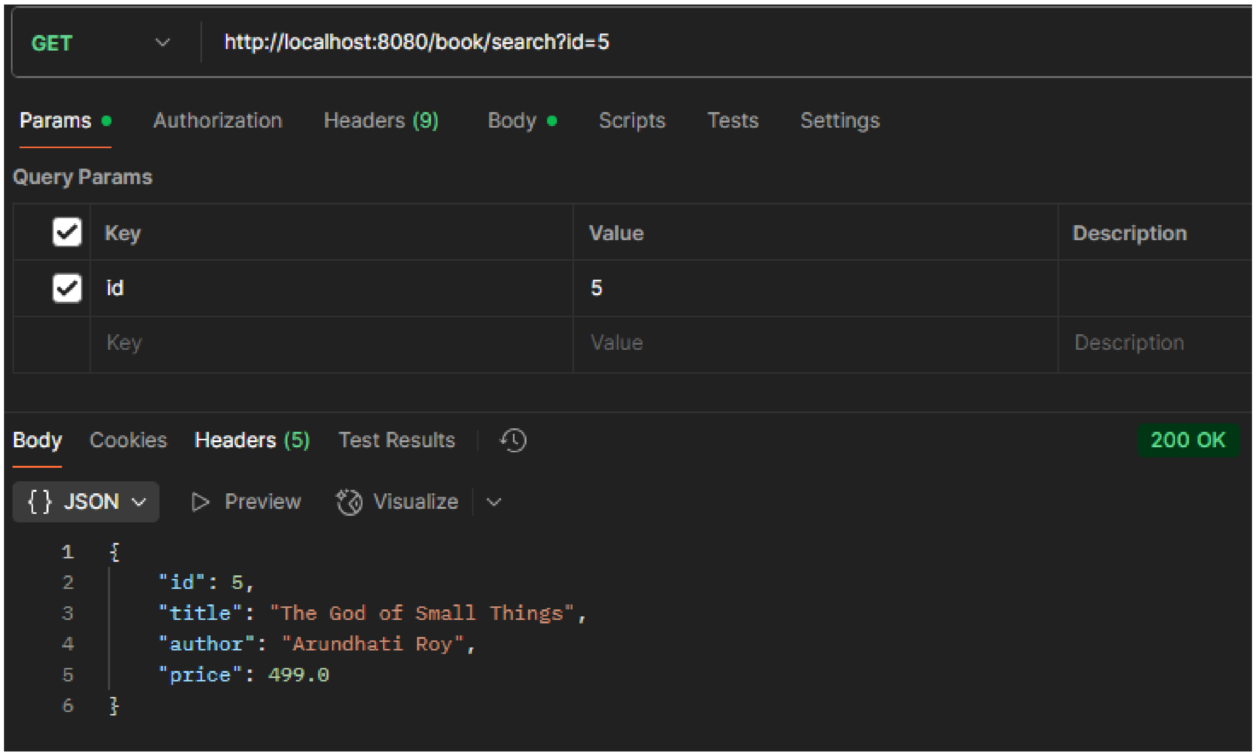
```
{
    "statusCode": 200,
    "statusMessage": "Book Added Successfully",
    "responseDate": "2025-07-02"
}
```

1. GET /books → Retrieve all books



```
    {
        "id": 5,
        "title": "The God of Small Things",
        "author": "Arundhati Roy",
        "price": 499.0
    },
    {
        "id": 6,
        "title": "Train to Pakistan",
        "author": "Khushwant Singh",
        "price": 350.0
    },
    {
        "id": 7,
        "title": "Ignited Minds",
        "author": "A.P.J. Abdul Kalam",
        "price": 270.0
```

2.**GET /books/{id}** → Retrieve a specific book by ID



```
http://localhost:8080/book/search?id=5
```

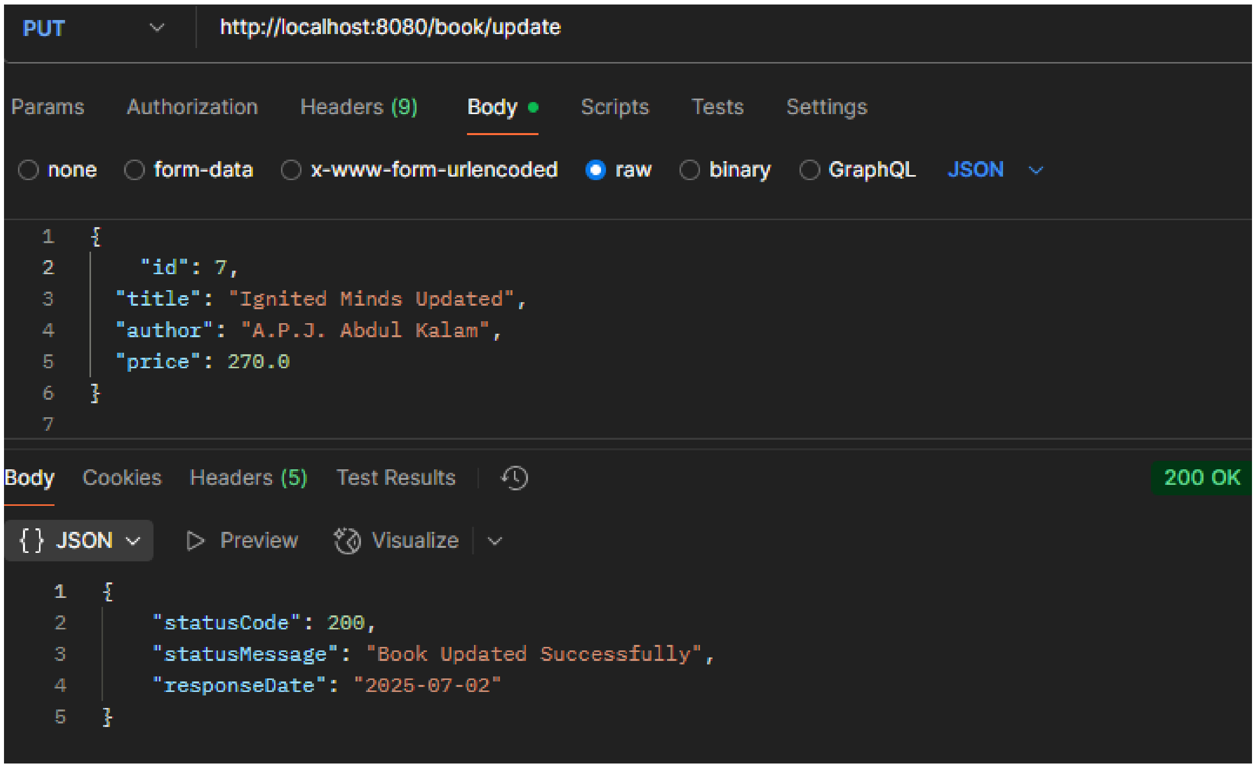
Query Params

	Key	Value	Description
✓	id	5	
	Key	Value	Description

200 OK

```
{
    "id": 5,
    "title": "The God of Small Things",
    "author": "Arundhati Roy",
    "price": 499.0
}
```

4.**PUT /books/{id}** → Update an existing book



PUT http://localhost:8080/book/update

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

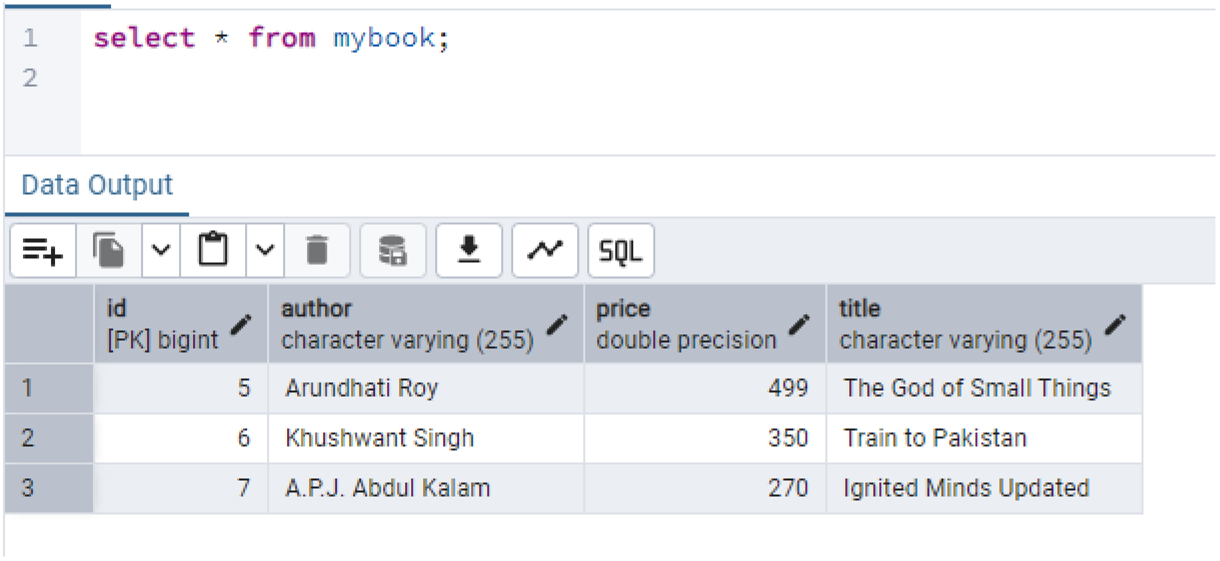
```
{
    "id": 7,
  "title": "Ignited Minds Updated",
  "author": "A.P.J. Abdul Kalam",
  "price": 270.0
}

```

Body Cookies Headers (5) Test Results 200 OK

{} JSON Preview Visualize

```
{
    "statusCode": 200,
    "statusMessage": "Book Updated Successfully",
    "responseDate": "2025-07-02"
}
```

Verifed on pgadmin:

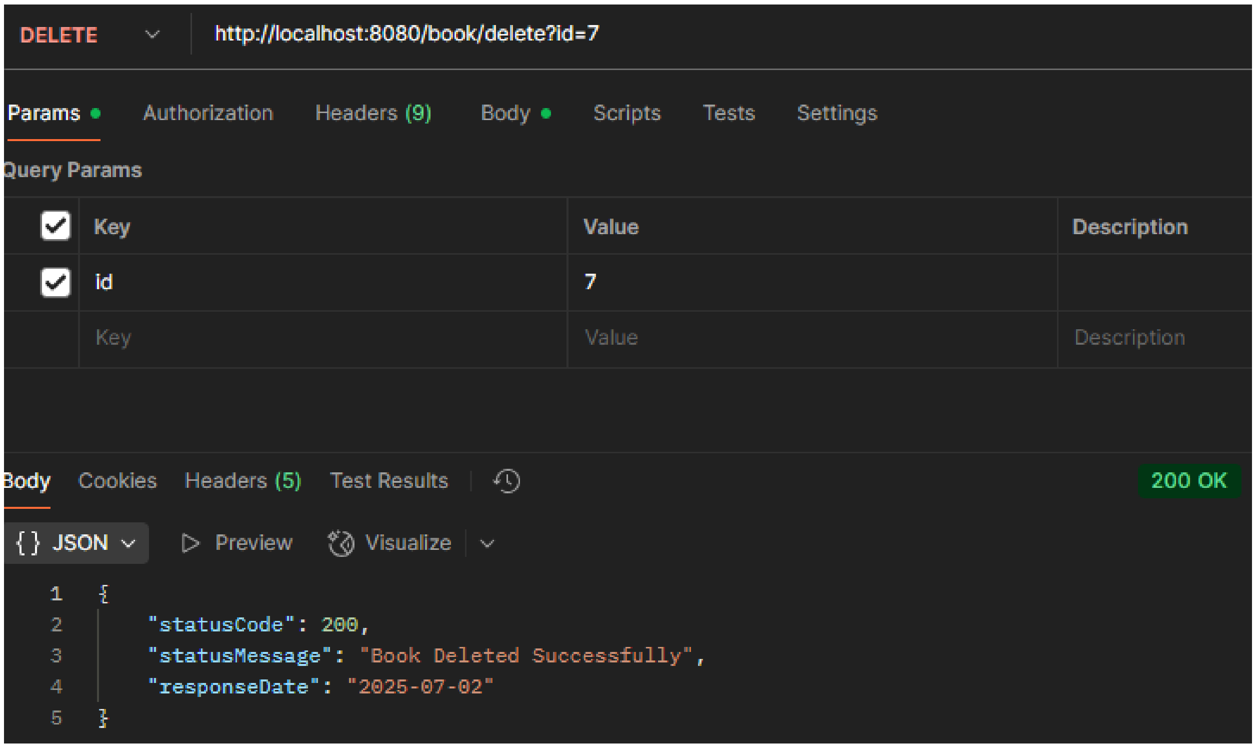


```
select * from mybook;
```

Data Output

	id [PK] bigint	author character varying (255)	price double precision	title character varying (255)
1	5	Arundhati Roy	499	The God of Small Things
2	6	Khushwant Singh	350	Train to Pakistan
3	7	A.P.J. Abdul Kalam	270	Ignited Minds Updated

5. **DELETE /books/{id}** → Delete a book

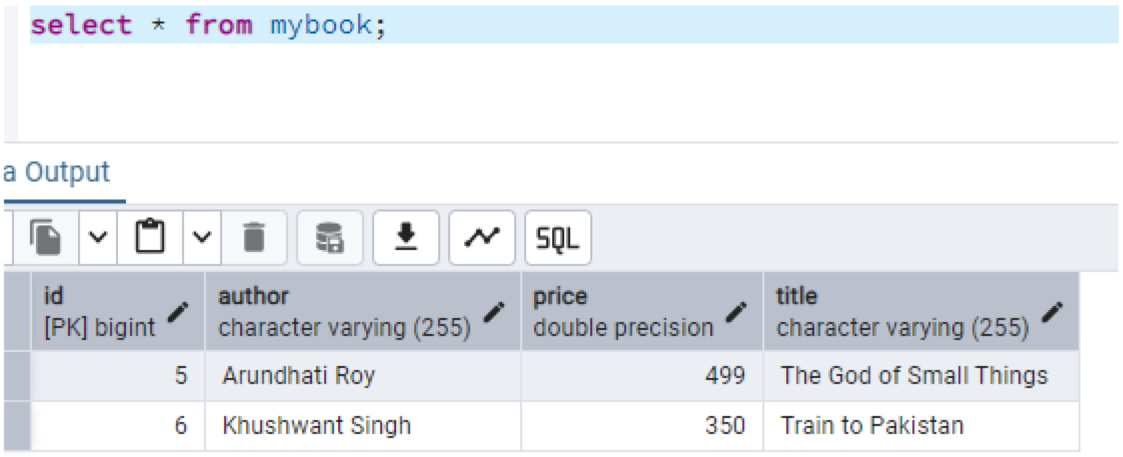


DELETE http://localhost:8080/book/delete?id=7

	Key	Value	Description
✓	id	7	

```
{
    "statusCode": 200,
    "statusMessage": "Book Deleted Successfully",
    "responseDate": "2025-07-02"
}
```

Verifed on pgadmin:

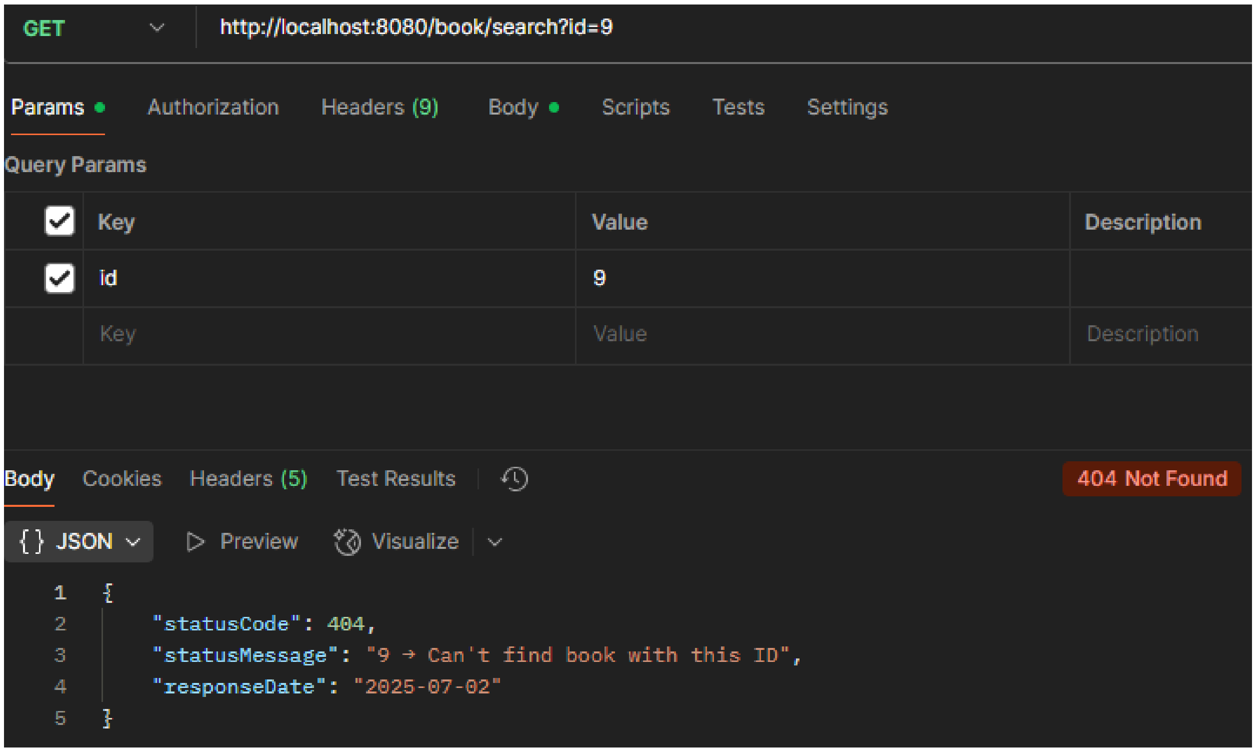


```sql
select * from mybook;
```

a Output

id [PK] bigint	author character varying (255)	price double precision	title character varying (255)
5	Arundhati Roy	499	The God of Small Things
6	Khushwant Singh	350	Train to Pakistan

- **Exception Handling output:**



GET http://localhost:8080/book/search?id=9

Query Params

	Key	Value	Description
✓	id	9	

```json
{
    "statusCode": 404,
    "statusMessage": "9 → Can't find book with this ID",
    "responseDate": "2025-07-02"
}
```