**Exercise 6: Online Bookstore - Exception Handling in REST Controllers**

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.bind.annotation.RestController;

import org.springframework.web.context.request.WebRequest;

@ControllerAdvice

@RestController

public class GlobalExceptionHandler {

// Handle ResourceNotFoundException

@ExceptionHandler(ResourceNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

public ResponseEntity<Object> handleResourceNotFoundException(

ResourceNotFoundException ex, WebRequest request) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

// Handle IllegalArgumentException

@ExceptionHandler(IllegalArgumentException.class)

@ResponseStatus(HttpStatus.BAD\_REQUEST)

public ResponseEntity<Object> handleIllegalArgumentException(

IllegalArgumentException ex, WebRequest request) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.BAD\_REQUEST);

}

// Handle all other exceptions

@ExceptionHandler(Exception.class)

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)

public ResponseEntity<Object> handleGlobalException(

Exception ex, WebRequest request) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

// Custom exception for resource not found

class ResourceNotFoundException extends RuntimeException {

public ResourceNotFoundException(String message) {

super(message);

}

}

// Example Controller using the GlobalExceptionHandler

@RestController

@RequestMapping("/books")

class BookController {

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

// Initialize with some books

public BookController() {

books.add(new Book(1, "Java Programming", "John Smith", 29.99, "1234567890"));

books.add(new Book(2, "Python Essentials", "Jane Doe", 39.99, "0987654321"));

}

// GET a book by ID with exception handling

@GetMapping("/{id}")

public Book getBookById(@PathVariable int id) {

return books.stream()

.filter(book -> book.getId() == id)

.findFirst()

.orElseThrow(() -> new ResourceNotFoundException("Book with id " + id + " not found."));

}

}

// Book class

class Book {

private int id;

private String title;

private String author;

private double price;

private String isbn;

// Constructors, getters, and setters

public Book() {}

public Book(int 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 int getId() {

return id;

}

public void setId(int 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;

}

}