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

**GlobalExceptionHandler**

package com.bookstore.api.exceptions;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(BookNotFoundException.class)

public ResponseEntity<ErrorResponse> handleBookNotFoundException(BookNotFoundException e) {

ErrorResponse errorResponse = new ErrorResponse("Book not found", e.getMessage());

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

}

@ExceptionHandler(InvalidBookDataException.class)

public ResponseEntity<ErrorResponse> handleInvalidBookDataException(InvalidBookDataException e) {

ErrorResponse errorResponse = new ErrorResponse("Invalid book data", e.getMessage());

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

}

@ExceptionHandler(Exception.class)

public ResponseEntity<ErrorResponse> handleGenericException(Exception e) {

ErrorResponse errorResponse = new ErrorResponse("Internal server error", e.getMessage());

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

}

}

class ErrorResponse {

private String message;

private String details;

public ErrorResponse(String message, String details) {

this.message = message;

this.details = details;

}

public String getMessage() {

return message;

}

public String getDetails() {

return details;

}

}

class BookNotFoundException extends RuntimeException {

public BookNotFoundException(String message) {

super(message);

}

}

class InvalidBookDataException extends RuntimeException {

public InvalidBookDataException(String message) {

super(message);

}

}

In the above code, I have implemented a global exception handling mechanism using @ControllerAdvice.

- I have created a GlobalExceptionHandler class to handle various exceptions.

- I have defined methods to handle specific exceptions and return appropriate HTTP status codes.

- I have created custom exception classes BookNotFoundException and InvalidBookDataException to handle specific error scenarios.

- I have used ResponseEntity to return a custom error response with details.