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

**Business Scenario:**

Implement a global exception handling mechanism for the bookstore RESTful services.

**Instructions:**

1. **Global Exception Handler:**
   * Create a **GlobalExceptionHandler** class using **@ControllerAdvice**.
   * Define methods to handle various exceptions and return appropriate HTTP status codes.

**Answers: -**

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

**1. Global Exception Handler**

**Step-by-Step Guide:**

1. **Create a GlobalExceptionHandler Class Using @ControllerAdvice:**
   * **@ControllerAdvice** is a specialized form of the @Component annotation, which is used to define global exception handling for all controllers in a Spring Boot application.
   * Create a new class named GlobalExceptionHandler in the src/main/java/com/example/bookstoreapi/exception package (you may need to create this package).

package com.example.bookstoreapi.exception;

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 java.time.LocalDateTime;

import java.util.HashMap;

import java.util.Map;

@ControllerAdvice

public class GlobalExceptionHandler {

// Handle specific exception: BookNotFoundException

@ExceptionHandler(BookNotFoundException.class)

public ResponseEntity<Map<String, Object>> handleBookNotFoundException(BookNotFoundException ex) {

Map<String, Object> response = new HashMap<>();

response.put("timestamp", LocalDateTime.now());

response.put("status", HttpStatus.NOT\_FOUND.value());

response.put("error", "Not Found");

response.put("message", ex.getMessage());

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

}

// Handle generic exceptions

@ExceptionHandler(Exception.class)

public ResponseEntity<Map<String, Object>> handleGenericException(Exception ex) {

Map<String, Object> response = new HashMap<>();

response.put("timestamp", LocalDateTime.now());

response.put("status", HttpStatus.INTERNAL\_SERVER\_ERROR.value());

response.put("error", "Internal Server Error");

response.put("message", ex.getMessage());

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

}

}

* + **Explanation:**
    - **@ControllerAdvice:** This annotation makes the GlobalExceptionHandler a global handler for exceptions across all controllers.
    - **@ExceptionHandler(BookNotFoundException.class):** This method specifically handles BookNotFoundException and returns a 404 Not Found status with a custom error message.
    - **handleGenericException:** This method handles any other exceptions that are not explicitly handled elsewhere and returns a 500 Internal Server Error status.

1. **Define Methods to Handle Various Exceptions and Return Appropriate HTTP Status Codes:**
   * **BookNotFoundException:**
     + This exception is thrown when a book is not found in the bookstore. The global exception handler catches this and returns a 404 Not Found status with a detailed error message.
   * **Generic Exception Handling:**
     + Any other exceptions that are not specifically handled are caught by the handleGenericException method, which returns a 500 Internal Server Error status.
   * **Example of BookNotFoundException:**

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class BookNotFoundException extends RuntimeException {

public BookNotFoundException(String message) {

super(message);

}

}

* + **Custom Error Response:**
    - The response includes details such as the timestamp, status code, error type, and the exception message, providing clear information to the client.

By implementing the GlobalExceptionHandler, you create a centralized and consistent approach to handling exceptions in your RESTful services. This not only improves code maintainability but also ensures that clients receive meaningful and uniform error responses.