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

**1. Creating a Global Exception Handler**

To centralize and handle exceptions globally across all our controllers, we can use the @ControllerAdvice annotation, which allows us to define a global exception handler.

**Steps:**

1. Create a GlobalExceptionHandler class.
2. Define methods in this class to handle different types of exceptions (e.g., ResourceNotFoundException, IllegalArgumentException).
3. Use @ExceptionHandler to catch specific exceptions and customize the response.

**2. Defining Custom Exception Handling**

In this example, we'll handle ResourceNotFoundException and IllegalArgumentException. We'll also return custom error messages with appropriate HTTP status codes using ResponseEntity.

**Example: Global Exception Handler**

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.context.request.WebRequest;

import org.springframework.web.server.ResponseStatusException;

import java.time.LocalDateTime;

import java.util.HashMap;

import java.util.Map;

@ControllerAdvice

public class GlobalExceptionHandler {

// Handle ResourceNotFoundException (e.g., when a book or customer is not found)

@ExceptionHandler(ResponseStatusException.class)

public ResponseEntity<Object> handleResourceNotFoundException(ResponseStatusException ex, WebRequest request) {

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

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

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

body.put("error", "Resource Not Found");

body.put("message", ex.getReason());

body.put("path", request.getDescription(false));

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

}

// Handle IllegalArgumentException (e.g., when invalid arguments are passed)

@ExceptionHandler(IllegalArgumentException.class)

public ResponseEntity<Object> handleIllegalArgumentException(IllegalArgumentException ex, WebRequest request) {

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

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

body.put("status", HttpStatus.BAD\_REQUEST.value());

body.put("error", "Bad Request");

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

body.put("path", request.getDescription(false));

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

}

// Handle generic exceptions (other unexpected exceptions)

@ExceptionHandler(Exception.class)

public ResponseEntity<Object> handleGlobalException(Exception ex, WebRequest request) {

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

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

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

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

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

body.put("path", request.getDescription(false));

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

}

}

**Explanation:**

* **Global Exception Handler:**
  + The @ControllerAdvice annotation makes this class a global exception handler for the entire application.
  + The @ExceptionHandler methods capture specific exceptions and return custom error responses.
* **Handling ResponseStatusException:**
  + Used for exceptions like "Resource Not Found" (e.g., book or customer is not found).
  + Returns a 404 Not Found status with a detailed error message in JSON format.
* **Handling IllegalArgumentException:**
  + Captures bad input cases where arguments are invalid.
  + Returns a 400 Bad Request status.
* **Handling Exception:**
  + A generic method to catch any other unhandled exceptions.
  + Returns a 500 Internal Server Error.

**3. Throwing Custom Exceptions in Controllers**

We can throw exceptions in your controllers, and the global handler will capture them.

**Example: Throwing Exceptions in BookController**

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookList.stream()

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

.findFirst()

.orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT\_FOUND, "Book not found"));

}

* This method throws a ResponseStatusException with a 404 Not Found status if the book is not found. The GlobalExceptionHandler will capture this and return a custom error response.

**4. Example Error Response**

If a book with the specified ID is not found, the global handler will return a response like this:

**Response:**

{

"timestamp": "2024-08-22T14:56:17",

"status": 404,

"error": "Resource Not Found",

"message": "Book not found",

"path": "/books/999"

}