**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.

**GlobalExceptionHandler.java**

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;

// Exception handler class

@ControllerAdvice

public class GlobalExceptionHandler {

// Handle specific exceptions

@ExceptionHandler(BookNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

public ResponseEntity<String> handleBookNotFoundException(BookNotFoundException ex) {

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

}

// Handle generic exceptions

@ExceptionHandler(Exception.class)

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)

public ResponseEntity<String> handleGeneralException(Exception ex) {

return new ResponseEntity<>("An unexpected error occurred: " + ex.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

**BookController.java**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.exception.BookNotFoundException;

import com.example.bookstoreapi.model.Book;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/api/books")

public class BookController {

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

// GET: Retrieve all books with custom headers

@GetMapping

public ResponseEntity<List<Book>> getAllBooks() {

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "BookListHeader");

return new ResponseEntity<>(books, headers, HttpStatus.OK);

}

// GET: Retrieve a book by ID with custom headers

@GetMapping("/{id}")

public ResponseEntity<Book> getBookById(@PathVariable Long id) {

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "SingleBookHeader");

Book book = books.stream().filter(b -> b.getId().equals(id)).findFirst()

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

return new ResponseEntity<>(book, headers, HttpStatus.OK);

}

// POST: Create a new book with custom headers

@PostMapping

public ResponseEntity<Book> createBook(@RequestBody Book book) {

books.add(book);

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "CreateBookHeader");

return new ResponseEntity<>(book, headers, HttpStatus.CREATED);

}

// PUT: Update an existing book with custom headers

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book updatedBook) {

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "UpdateBookHeader");

for (int i = 0; i < books.size(); i++) {

Book book = books.get(i);

if (book.getId().equals(id)) {

books.set(i, updatedBook);

return new ResponseEntity<>(updatedBook, headers, HttpStatus.OK);

}

}

throw new BookNotFoundException("Book with ID " + id + " not found");

}

// DELETE: Delete a book by ID with custom headers

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "DeleteBookHeader");

boolean removed = books.removeIf(book -> book.getId().equals(id));

if (!removed) {

throw new BookNotFoundException("Book with ID " + id + " not found");

}

return new ResponseEntity<>(headers, HttpStatus.NO\_CONTENT);

}

}

**BookNotFoundException.java**

package com.example.bookstoreapi.exception;

public class BookNotFoundException extends RuntimeException {

public BookNotFoundException(String message) {

super(message);

}

}