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 {

@ExceptionHandler(BookNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

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

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

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

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

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

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

errorResponse.put("path", "/books/{id}");

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

}

@ExceptionHandler(Exception.class)

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)

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

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

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

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

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

errorResponse.put("message", "An unexpected error occurred");

errorResponse.put("path", "N/A");

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

}

// Additional exception handlers can be added here

}

package com.example.bookstoreapi.exception;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class BookNotFoundException extends RuntimeException {

public BookNotFoundException(String message) {

super(message);

}

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookService.getBookById(id)

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

}

{

"timestamp": "2024-08-21T12:34:56.789",

"status": 404,

"error": "Not Found",

"message": "Book not found with id 1",

"path": "/books/1"

}