**Exercise 5: Online Bookstore - Customizing Response Status and Headers**

**Business Scenario:**

Customize the HTTP response status and headers for the book management endpoints.

**Instructions:**

1. **Response Status:**
   * Use **@ResponseStatus** to customize HTTP status codes for your endpoints.
2. **Custom Headers:**
   * Add custom headers to the response using **ResponseEntity**.

**Answers: -**

**Exercise 5: Online Bookstore - Customizing Response Status and Headers**

**1. Response Status**

**Step-by-Step Guide:**

1. **Use @ResponseStatus to Customize HTTP Status Codes for Your Endpoints:**
   * The @ResponseStatus annotation can be used on methods to set the HTTP status code for the response.

Here’s how you can customize the status codes in the BookController:

import org.springframework.http.HttpStatus;

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

@RestController

@RequestMapping("/books")

public class BookController {

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

// POST: /books - Add a new book

@PostMapping

@ResponseStatus(HttpStatus.CREATED) // Customizing status to 201 Created

public Book addBook(@RequestBody Book book) {

books.add(book);

return book;

}

// GET: /books/{id} - Retrieve a book by its ID

@GetMapping("/{id}")

@ResponseStatus(HttpStatus.OK) // Customizing status to 200 OK (default, but explicitly stated)

public Book getBookById(@PathVariable int id) {

return books.stream()

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

.findFirst()

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

}

// DELETE: /books/{id} - Remove a book

@DeleteMapping("/{id}")

@ResponseStatus(HttpStatus.NO\_CONTENT) // Customizing status to 204 No Content

public void deleteBook(@PathVariable int id) {

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

}

}

* + **Explanation:**
    - **@ResponseStatus(HttpStatus.CREATED):** Sets the status to 201 Created for the addBook method.
    - **@ResponseStatus(HttpStatus.NO\_CONTENT):** Sets the status to 204 No Content for the deleteBook method.
    - If a book is not found, you might want to throw a custom exception, which can be mapped to a 404 Not Found status (e.g., using @ResponseStatus(HttpStatus.NOT\_FOUND) in the exception class).
  + **Custom Exception for Not Found:**

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class BookNotFoundException extends RuntimeException {

public BookNotFoundException(String message) {

super(message);

}

}

**2. Custom Headers**

1. **Add Custom Headers to the Response Using ResponseEntity:**
   * Use ResponseEntity to control the full HTTP response, including the status code and headers.

Here’s how you can modify your existing endpoints to include custom headers:

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

@RestController

@RequestMapping("/books")

public class BookController {

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

// POST: /books - Add a new book with custom headers

@PostMapping

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

books.add(book);

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "Book Created");

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

}

// GET: /books/{id} - Retrieve a book by its ID with custom headers

@GetMapping("/{id}")

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

Book book = books.stream()

.filter(b -> b.getId() == id)

.findFirst()

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

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "Book Retrieved");

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

}

// DELETE: /books/{id} - Remove a book with custom headers

@DeleteMapping("/{id}")

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

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

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "Book Deleted");

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

}

}

* + **Explanation:**
    - **HttpHeaders headers = new HttpHeaders();**: Creates a HttpHeaders object to store custom headers.
    - **headers.add("Custom-Header", "Value");**: Adds a custom header to the response.
    - **new ResponseEntity<>(body, headers, status)**: Constructs the ResponseEntity with the body, headers, and status.
  + **Example Custom Header:**
    - "Custom-Header": "Book Created" is added to the response when a new book is created.

By using @ResponseStatus and ResponseEntity, we can finely control the HTTP status codes and headers returned by your RESTful service, making it more informative and flexible for client applications.