**Exercise 3: Online Bookstore - Handling Path Variables and Query Parameters**

1. Handling Path Variables:

**Fetch Book by ID Using Path Variable**

In the BookController, we can use @PathVariable to extract the book ID from the URL:

// Fetch a specific book by ID using a path variable

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookList.stream()

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

.findFirst()

.orElse(null); // return null if book is not found

}

1. Handling Query Parameters:

#### Filter Books by Query Parameters

In this example, the endpoint can accept optional query parameters like title and author. You can call this endpoint with different combinations of these parameters.

// Filter books by title and/or author using query parameters

@GetMapping("/search")

public List<Book> filterBooks(

@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

return bookList.stream()

.filter(book -> (title == null || book.getTitle().equalsIgnoreCase(title)) &&

(author == null || book.getAuthor().equalsIgnoreCase(author)))

.toList(); // Return filtered list of books

}

Updated BookController:

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

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

import java.util.ArrayList;

import java.util.List;

import java.util.stream.Collectors;

@RestController

@RequestMapping("/books")

public class BookController {

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

// Get all books

@GetMapping

public List<Book> getAllBooks() {

return bookList;

}

// Get a book by ID (Path Variable)

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookList.stream()

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

.findFirst()

.orElse(null);

}

// Create a new book

@PostMapping

public Book createBook(@RequestBody Book newBook) {

bookList.add(newBook);

return newBook;

}

// Update a book

@PutMapping("/{id}")

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

for (Book book : bookList) {

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

book.setTitle(updatedBook.getTitle());

book.setAuthor(updatedBook.getAuthor());

book.setPrice(updatedBook.getPrice());

book.setIsbn(updatedBook.getIsbn());

return book;

}

}

return null;

}

// Delete a book

@DeleteMapping("/{id}")

public String deleteBook(@PathVariable Long id) {

bookList.removeIf(book -> book.getId().equals(id));

return "Book deleted";

}

// Filter books by query parameters (title and author)

@GetMapping("/search")

public List<Book> filterBooks(

@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

return bookList.stream()

.filter(book -> (title == null || book.getTitle().equalsIgnoreCase(title)) &&

(author == null || book.getAuthor().equalsIgnoreCase(author)))

.collect(Collectors.toList());

}

}

**Example Requests:**

1. **Fetching a Book by ID (Path Variable)**
   * **GET** /books/1
   * Fetches the book with ID 1.
2. **Filtering Books by Title and/or Author (Query Parameters)**
   * **GET** /books/search?title=The Great Gatsby&author=F. Scott Fitzgerald
   * Filters the list of books by the given title and author.
   * **GET** /books/search?title=The Great Gatsby
   * Filters books only by title.
   * **GET** /books/search?author=F. Scott Fitzgerald
   * Filters books only by author.

These dynamic URLs and query parameters give users the flexibility to retrieve specific books based on their preferences or requirements.