**WEEK 4&5**

**EXERCISE 3**

**Online Bookstore - Handling Path Variables and Query Parameters**

**Business Scenario**

The task is to enhance the book management endpoints to handle dynamic URLs and query parameters in the online bookstore application.

**Instructions**

**1. Path Variables:**

**Task:** Implement an endpoint to fetch a book by its ID using a path variable.

**Implementation:**

* Use the @PathVariable annotation to extract the book ID from the URL.
* The endpoint /books/{id} will be used to retrieve the book by its ID.

**Code Example:**

@GetMapping("/{id}")

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

Book book = bookService.getBookById(id);

return book != null ? ResponseEntity.ok(book) : ResponseEntity.notFound().build();

}

**Explanation:**

* The getBookById method is mapped to the GET /books/{id} URL.
* The @PathVariable annotation is used to extract the id from the URL and pass it to the getBookById method in the BookService class.
* If the book is found, it returns a 200 OK status with the book data; otherwise, it returns a 404 Not Found status.

**2. Query Parameters:**

**Task:** Implement an endpoint to filter books based on query parameters like title and author.

**Implementation:**

* Use the @RequestParam annotation to capture query parameters from the URL.
* The endpoint /books/search will be used to filter books based on the provided query parameters.

**Code Example:**

@GetMapping("/search")

public ResponseEntity<List<Book>> searchBooks(

@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

List<Book> books = bookService.searchBooks(title, author);

return ResponseEntity.ok(books);

}

**Explanation:**

* The searchBooks method is mapped to the GET /books/search URL.
* The @RequestParam annotation is used to capture the title and author query parameters from the URL.
* Both parameters are optional (required = false), allowing users to filter books by title, author, or both.
* The method returns a list of books that match the search criteria with a 200 OK status.

**Example Logic for the Book Service:**

To support these endpoints, the BookService class can include the following logic:

public List<Book> searchBooks(String title, String author) {

return bookRepository.values().stream()

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

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

.collect(Collectors.toList());

}

**Explanation:**

* The searchBooks method filters the books based on the title and author parameters.
* The filter method checks if each book matches the given title and author (if provided).
* The collect method gathers all matching books into a list.

**Conclusion:**

By implementing these enhancements, The RESTful service can now handle dynamic URLs using path variables and provide flexible search functionality using query parameters.