**Scenario 15: Online Bookstore - API Documentation with Swagger**

**Business Scenario:**

Document your bookstore's REST APIs using Swagger and Springdoc.

**Instructions:**

1. **Add Swagger Dependency:**
   * Include Swagger or Springdoc dependencies in your project.
2. **Document Endpoints:**
   * Annotate your REST controllers and methods to generate API documentation.
3. **API Documentation:**
   * Generate and review the API documentation using **Swagger UI** or **Springdoc UI**.

**pom.xml**

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-ui</artifactId>

<version>2.2.0</version> <!-- Check for the latest version -->

</dependency>

**SwaggerConfig.java**

package com.example.bookstoreapi.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springdoc.core.GroupedOpenApi;

import org.springdoc.webmvc.ui.SwaggerUiConfigParameters;

import org.springdoc.webmvc.ui.SwaggerUiResourceProvider;

@Configuration

public class SwaggerConfig {

@Bean

public GroupedOpenApi publicApi() {

return GroupedOpenApi.builder()

.group("bookstore-public")

.packagesToScan("com.example.bookstoreapi.controller")

.build();

}

}

**BookController.java**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

import com.example.bookstoreapi.repository.BookRepository;

import io.swagger.v3.oas.annotations.Operation;

import io.swagger.v3.oas.annotations.Parameter;

import io.swagger.v3.oas.annotations.responses.ApiResponse;

import io.swagger.v3.oas.annotations.tags.Tag;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/books")

@Tag(name = "Book Controller", description = "Operations related to books")

public class BookController {

@Autowired

private BookRepository bookRepository;

@Operation(summary = "Get all books", description = "Retrieve a list of all books")

@GetMapping

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

List<Book> books = bookRepository.findAll();

return ResponseEntity.ok(books);

}

@Operation(summary = "Get book by ID", description = "Retrieve a book by its ID")

@ApiResponse(responseCode = "200", description = "Successfully retrieved book")

@ApiResponse(responseCode = "404", description = "Book not found")

@GetMapping("/{id}")

public ResponseEntity<Book> getBookById(

@Parameter(description = "ID of the book to retrieve", required = true)

@PathVariable Long id) {

return bookRepository.findById(id)

.map(book -> ResponseEntity.ok().body(book))

.orElse(ResponseEntity.notFound().build());

}

@Operation(summary = "Create a new book", description = "Add a new book to the store")

@PostMapping

public ResponseEntity<Book> createBook(

@Parameter(description = "Book object to be created", required = true)

@RequestBody Book book) {

Book createdBook = bookRepository.save(book);

return ResponseEntity.ok(createdBook);

}

@Operation(summary = "Update a book", description = "Update details of an existing book")

@ApiResponse(responseCode = "200", description = "Successfully updated book")

@ApiResponse(responseCode = "404", description = "Book not found")

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(

@Parameter(description = "ID of the book to update", required = true)

@PathVariable Long id,

@Parameter(description = "Updated book object", required = true)

@RequestBody Book book) {

return bookRepository.findById(id)

.map(existingBook -> {

existingBook.setTitle(book.getTitle());

existingBook.setAuthor(book.getAuthor());

existingBook.setPrice(book.getPrice());

Book updatedBook = bookRepository.save(existingBook);

return ResponseEntity.ok(updatedBook);

})

.orElse(ResponseEntity.notFound().build());

}

@Operation(summary = "Delete a book", description = "Remove a book from the store")

@ApiResponse(responseCode = "204", description = "Successfully deleted book")

@ApiResponse(responseCode = "404", description = "Book not found")

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(

@Parameter(description = "ID of the book to delete", required = true)

@PathVariable Long id) {

return bookRepository.findById(id)

.map(existingBook -> {

bookRepository.delete(existingBook);

return ResponseEntity.noContent().build();

})

.orElse(ResponseEntity.notFound().build());

}

}