**Exercise 10: Online Bookstore - Configuring Content Negotiation**

**Business Scenario:**

Support different media types (JSON, XML) for your bookstore's RESTful services.

**Instructions:**

1. **Content Negotiation:**
   * Configure Spring Boot to support content negotiation.
2. **Accept Header:**
   * Implement logic to produce and consume different media types based on the Accept header.

**pom.xml**

<dependency>

<groupId>com.fasterxml.jackson.dataformat</groupId>

<artifactId>jackson-dataformat-xml</artifactId>

</dependency>

**WebConfig.java**

package com.example.bookstoreapi.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.ContentNegotiationConfigurer;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

public class WebConfig implements WebMvcConfigurer {

@Override

public void configureContentNegotiation(ContentNegotiationConfigurer configurer) {

configurer

.favorParameter(true) // Allows content type to be set using a URL parameter, e.g., /books?format=xml

.parameterName("format")

.ignoreAcceptHeader(false) // Consider the Accept header for content negotiation

.defaultContentType(org.springframework.http.MediaType.APPLICATION\_JSON) // Default to JSON if no type is specified

.mediaType("json", org.springframework.http.MediaType.APPLICATION\_JSON)

.mediaType("xml", org.springframework.http.MediaType.APPLICATION\_XML);

}

}

**BookController.java**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

import com.example.bookstoreapi.service.BookService;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookService bookService;

@GetMapping("/{id}")

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

Book book = bookService.getBookById(id);

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

}

@GetMapping

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

List<Book> books = bookService.getAllBooks();

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

}

@PostMapping

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

Book createdBook = bookService.createBook(book);

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

}

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book book) {

Book updatedBook = bookService.updateBook(id, book);

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

}

@DeleteMapping("/{id}")

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

bookService.deleteBook(id);

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

}

}

**BookService.java**

package com.example.bookstoreapi.service;

import com.example.bookstoreapi.model.Book;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

@Service

public class BookService {

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

public Book getBookById(Long id) {

return books.stream().filter(book -> book.getId().equals(id)).findFirst().orElse(null);

}

public List<Book> getAllBooks() {

return books;

}

public Book createBook(Book book) {

books.add(book);

return book;

}

public Book updateBook(Long id, Book book) {

Book existingBook = getBookById(id);

if (existingBook != null) {

existingBook.setTitle(book.getTitle());

existingBook.setAuthor(book.getAuthor());

}

return existingBook;

}

public void deleteBook(Long id) {

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

}

}

**Book.java**

package com.example.bookstoreapi.model;

import com.fasterxml.jackson.dataformat.xml.annotation.JacksonXmlRootElement;

@JacksonXmlRootElement(localName = "book")

public class Book {

private Long id;

private String title;

private String author;

// Constructors

public Book() {}

public Book(Long id, String title, String author) {

this.id = id;

this.title = title;

this.author = author;

}

// Getters and Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

}

**BookstoreApiApplication.java**

package com.example.bookstoreapi;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class BookstoreApiApplication {

public static void main(String[] args) {

SpringApplication.run(BookstoreApiApplication.class, args);

}

}