**Exercise 7 : Online Bookstore - Introduction to Data Transfer Objects (DTOs)**

**Business Scenario:**

Use DTOs to transfer data between the client and server for books and customers.

**Instructions:**

1. **Create DTOs:**
   * Define BookDTO and CustomerDTO classes.
2. **Mapping Entities to DTOs:**
   * Use a library like **MapStruct** or **ModelMapper** to map entities to DTOs and vice versa.
3. **Custom Serialization/Deserialization:**
   * Customize JSON serialization and deserialization using Jackson annotations.

**BookDTO.java**

package com.example.bookstoreapi.dto;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

public class BookDTO {

private Long id;

private String title;

private String author;

private Double price;

// Constructors, getters, and setters

public BookDTO() {

}

public BookDTO(Long id, String title, String author, Double price) {

this.id = id;

this.title = title;

this.author = author;

this.price = price;

}

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;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

**CustomerDTO.java**

package com.example.bookstoreapi.dto;

import com.fasterxml.jackson.annotation.JsonInclude;

@JsonInclude(JsonInclude.Include.NON\_NULL)

public class CustomerDTO {

private Long id;

private String name;

private String email;

// Constructors, getters, and setters

public CustomerDTO() {

}

public CustomerDTO(Long id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

**pom.xml**

<dependencies>

<!-- Other dependencies -->

<!-- MapStruct -->

<dependency>

<groupId>org.mapstruct</groupId>

<artifactId>mapstruct</artifactId>

<version>1.5.2.Final</version>

</dependency>

<dependency>

<groupId>org.mapstruct</groupId>

<artifactId>mapstruct-processor</artifactId>

<version>1.5.2.Final</version>

<scope>provided</scope>

</dependency>

</dependencies>

**BookMapper.java**

package com.example.bookstoreapi.mapper;

import com.example.bookstoreapi.dto.BookDTO;

import com.example.bookstoreapi.model.Book;

import org.mapstruct.Mapper;

import org.mapstruct.factory.Mappers;

@Mapper

public interface BookMapper {

BookMapper INSTANCE = Mappers.getMapper(BookMapper.class);

BookDTO bookToBookDTO(Book book);

Book bookDTOToBook(BookDTO bookDTO);

}

**CustomerMapper.java**

package com.example.bookstoreapi.mapper;

import com.example.bookstoreapi.dto.CustomerDTO;

import com.example.bookstoreapi.model.Customer;

import org.mapstruct.Mapper;

import org.mapstruct.factory.Mappers;

@Mapper

public interface CustomerMapper {

CustomerMapper INSTANCE = Mappers.getMapper(CustomerMapper.class);

CustomerDTO customerToCustomerDTO(Customer customer);

Customer customerDTOToCustomer(CustomerDTO customerDTO);

}

**BookController.java**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.dto.BookDTO;

import com.example.bookstoreapi.mapper.BookMapper;

import com.example.bookstoreapi.model.Book;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/api/books")

public class BookController {

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

private final BookMapper bookMapper = BookMapper.INSTANCE;

// GET: Retrieve all books

@GetMapping

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

List<BookDTO> bookDTOs = new ArrayList<>();

for (Book book : books) {

bookDTOs.add(bookMapper.bookToBookDTO(book));

}

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "BookListHeader");

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

}

// GET: Retrieve a book by ID

@GetMapping("/{id}")

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

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

BookDTO bookDTO = bookMapper.bookToBookDTO(book);

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "SingleBookHeader");

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

}

// POST: Create a new book

@PostMapping

public ResponseEntity<BookDTO> createBook(@RequestBody BookDTO bookDTO) {

Book book = bookMapper.bookDTOToBook(bookDTO);

books.add(book);

BookDTO newBookDTO = bookMapper.bookToBookDTO(book);

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "CreateBookHeader");

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

}

// PUT: Update an existing book

@PutMapping("/{id}")

public ResponseEntity<BookDTO> updateBook(@PathVariable Long id, @RequestBody BookDTO bookDTO) {

for (int i = 0; i < books.size(); i++) {

Book book = books.get(i);

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

Book updatedBook = bookMapper.bookDTOToBook(bookDTO);

books.set(i, updatedBook);

BookDTO updatedBookDTO = bookMapper.bookToBookDTO(updatedBook);

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "UpdateBookHeader");

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

}

}

return new ResponseEntity<>(null, HttpStatus.NOT\_FOUND);

}

// DELETE: Delete a book by ID

@DeleteMapping("/{id}")

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

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

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "DeleteBookHeader");

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

}

}