**WEEK 4&5**

**EXERCISE 7**

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

**Business Scenario**

The task is to use Data Transfer Objects (DTOs) to transfer data between the client and server for books and customers in the online bookstore application.

**Instructions**

**1. Create DTOs:**

**Task:** Define BookDTO and CustomerDTO classes.

**Implementation:**

* DTOs are simple objects used to transfer data between layers in an application. They help decouple the internal representation of an entity from the representation exposed to clients.

**Code Example:**

public class BookDTO {

private Long id;

private String title;

private String author;

private double price;

// 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;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

}

public class CustomerDTO {

private Long id;

private String name;

private String email;

// Getters and Setters

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;

}

}

**Explanation:**

* The BookDTO class includes fields id, title, author, and price to represent the essential information of a book.
* The CustomerDTO class includes fields id, name, and email to represent the essential information of a customer.
* These DTOs are used to transfer data between the client and server, without exposing internal entity structures.

**2. Mapping Entities to DTOs:**

**Task:** Use a library like MapStruct or ModelMapper to map entities to DTOs and vice versa.

**Implementation:**

* Mapping libraries like MapStruct or ModelMapper simplify the conversion between entity objects and DTOs.

**Code Example using MapStruct:**

@Mapper(componentModel = "spring")

public interface BookMapper {

BookDTO toBookDTO(Book book);

Book toBook(BookDTO bookDTO);

}

@Mapper(componentModel = "spring")

public interface CustomerMapper {

CustomerDTO toCustomerDTO(Customer customer);

Customer toCustomer(CustomerDTO customerDTO);

}

**Explanation:**

* The BookMapper and CustomerMapper interfaces define methods to map between Book entities and BookDTO, and between Customer entities and CustomerDTO, respectively.
* The @Mapper annotation indicates that these interfaces will be implemented by MapStruct to generate the mapping logic.

**3. Custom Serialization/Deserialization:**

**Task:** Customize JSON serialization and deserialization using Jackson annotations.

**Implementation:**

* Jackson annotations can be used to control how fields are serialized and deserialized in JSON.

**Code Example:**

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonProperty;

public class CustomerDTO {

private Long id;

@JsonProperty("full\_name")

private String name;

private String email;

@JsonIgnore

private String password; // This field will not be serialized

// Getters and Setters

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;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**Explanation:**

* The @JsonProperty("full\_name") annotation renames the name field to full\_name in the JSON output.
* The @JsonIgnore annotation prevents the password field from being included in the serialized JSON, ensuring sensitive information is not exposed.

**Conclusion:** By using DTOs, mapping libraries, and custom JSON serialization/deserialization, we can create a clean and secure data transfer layer in your RESTful service.