**Exercise 4: Online Bookstore - Processing Request Body and Form Data**

**1. Processing Request Body (JSON) for Customer Registration**

We will first define a Customer entity and then create an endpoint that accepts a JSON request body to register a new customer.

**Define Customer Entity**

Create a Customer class inside the model package:

package com.example.bookstoreapi.model;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

public class Customer {

private Long id;

private String name;

private String email;

private String phone;

}

**Implement POST Endpoint for JSON Request Body**

Create a new CustomerController to handle customer registrations. This controller will include a POST endpoint to create a new customer.

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Customer;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/customers")

public class CustomerController {

private List<Customer> customerList = new ArrayList<>();

// POST endpoint to create a new customer with JSON request body

@PostMapping

public Customer createCustomer(@RequestBody Customer newCustomer) {

customerList.add(newCustomer);

return newCustomer;

}

// GET all customers

@GetMapping

public List<Customer> getAllCustomers() {

return customerList;

}

}

**Example Request (JSON):**

* **POST** /customers

**Request Body:**

{

"id": 1,

"name": "John Doe",

"email": "john.doe@example.com",

"phone": "123-456-7890"

}

The createCustomer method accepts the customer details in the form of a JSON request body and registers the new customer in the list.

**2. Processing Form Data for Customer Registration**

To handle form data, we'll need to create a separate endpoint that accepts the data using @RequestParam.

**POST Endpoint for Form Data**

Extend the CustomerController to include another endpoint that accepts form data.

// POST endpoint to process form data for customer registration

@PostMapping("/register")

public Customer registerCustomer(

@RequestParam String name,

@RequestParam String email,

@RequestParam String phone) {

Customer newCustomer = new Customer();

newCustomer.setId((long) (customerList.size() + 1)); // Auto-generate ID

newCustomer.setName(name);

newCustomer.setEmail(email);

newCustomer.setPhone(phone);

customerList.add(newCustomer);

return newCustomer;

}

**Example Request (Form Data):**

* **POST** /customers/register

**Form Data:**

* + name: John Doe
  + email: john.doe@example.com
  + phone: 123-456-7890

We can send this form data using an HTML form or Postman by selecting the "form-data" option.

Updated CustomerController

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Customer;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/customers")

public class CustomerController {

private List<Customer> customerList = new ArrayList<>();

// 1. POST endpoint to create a new customer with JSON request body

@PostMapping

public Customer createCustomer(@RequestBody Customer newCustomer) {

customerList.add(newCustomer);

return newCustomer;

}

// 2. POST endpoint to process form data for customer registration

@PostMapping("/register")

public Customer registerCustomer(

@RequestParam String name,

@RequestParam String email,

@RequestParam String phone) {

Customer newCustomer = new Customer();

newCustomer.setId((long) (customerList.size() + 1)); // Auto-generate ID

newCustomer.setName(name);

newCustomer.setEmail(email);

newCustomer.setPhone(phone);

customerList.add(newCustomer);

return newCustomer;

}

// GET all customers

@GetMapping

public List<Customer> getAllCustomers() {

return customerList;

}

}