**Hands on 5**

**Implement services for managing Country**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.4</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Src/main/resources:**

**application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/orm\_learn

spring.datasource.username=root

spring.datasource.password=Sathwik@123

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**com.cognizant.ormlearn.model:**

**Country.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**com.cognizant.ormlearn.repository.CountryRepository.java**

**package** com.cognizant.ormlearn.repository;

**import** java.util.List;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.cognizant.ormlearn.model.Country;

**public** **interface** CountryRepository **extends** JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String name);

}

**Com.cognizant.ormlearn.service.CountryService.java:**

package com.cognizant.ormlearn.service;

import java.util.List;

import com.cognizant.ormlearn.model.Country;

public interface CountryService {

Country getCountry(String code);

void addCountry(Country country);

void updateCountry(Country country);

void deleteCountry(String code);

List<Country> searchCountries(String keyword);

}

**Com.cognizant.ormlearn.service.CountryServiceImpl.java:**

package com.cognizant.ormlearn.service;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository countryRepository;

@Override

public Country getCountry(String code) {

return countryRepository.findById(code).orElse(null);

}

@Override

public void addCountry(Country country) {

countryRepository.save(country);

}

@Override

public void updateCountry(Country country) {

countryRepository.save(country);

}

@Override

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

@Override

public List<Country> searchCountries(String keyword) {

return countryRepository.findByNameContainingIgnoreCase(keyword);

}

}

**src/main/java/com/cognizant/ormlearn/OrmLearnApplication.java:**

package com.cognizant.ormlearn;

import java.util.List;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.*run*(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

System.*out*.println("---------- Spring Data JPA - Country Operations ----------");

// Add a new country

Country newCountry = new Country();

newCountry.setCode("ZZ");

newCountry.setName("Testland");

countryService.addCountry(newCountry);

System.*out*.println("Added Country: " + newCountry);

// Find by country code

Country country = countryService.getCountry("IN");

if (country != null) {

System.*out*.println("Found Country: " + country);

} else {

System.*out*.println("Country with code IN not found.");

}

// Update existing country

Country updatedCountry = new Country();

updatedCountry.setCode("ZZ");

updatedCountry.setName("Testland Updated");

countryService.updateCountry(updatedCountry);

System.*out*.println("Updated Country ZZ");

// Partial search

List<Country> countries = countryService.searchCountries("land");

System.*out*.println("Search results for 'land':");

countries.forEach(c -> System.*out*.println(c.getCode() + " - " + c.getName()));

// Delete country

countryService.deleteCountry("ZZ");

System.*out*.println("Deleted Country ZZ");

System.*out*.println("----------------------------------------------------------");

}

}

**MySQL:**

CREATE DATABASE IF NOT EXISTS orm\_learn;

USE orm\_learn;

DROP TABLE IF EXISTS country;

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(100) NOT NULL

);

INSERT INTO country (co\_code, co\_name) VALUES

("AF", "Afghanistan"),

("AL", "Albania"),

("DZ", "Algeria"),

("AS", "American Samoa"),

("AD", "Andorra"),

("AO", "Angola"),

("AI", "Anguilla"),

("AQ", "Antarctica"),

("AG", "Antigua and Barbuda"),

("AR", "Argentina"),

("AM", "Armenia"),

("AW", "Aruba"),

("AU", "Australia"),

("AT", "Austria"),

("AZ", "Azerbaijan"),

("BS", "Bahamas"),

("BH", "Bahrain"),

("BD", "Bangladesh"),

("BB", "Barbados"),

("BY", "Belarus"),

("BE", "Belgium"),

("BZ", "Belize"),

("BJ", "Benin"),

("BM", "Bermuda"),

("BT", "Bhutan"),

("BO", "Bolivia, Plurinational State of"),

("BQ", "Bonaire, Sint Eustatius and Saba"),

("BA", "Bosnia and Herzegovina"),

("BW", "Botswana"),

("BV", "Bouvet Island"),

("BR", "Brazil"),

("IO", "British Indian Ocean Territory"),

("BN", "Brunei Darussalam"),

("BG", "Bulgaria"),

("BF", "Burkina Faso"),

("BI", "Burundi"),

("KH", "Cambodia"),

("CM", "Cameroon"),

("CA", "Canada"),

("CV", "Cape Verde"),

("KY", "Cayman Islands"),

("CF", "Central African Republic"),

("TD", "Chad"),

("CL", "Chile"),

("CN", "China"),

("CX", "Christmas Island"),

("CC", "Cocos (Keeling) Islands"),

("CO", "Colombia"),

("KM", "Comoros"),

("CG", "Congo"),

("CD", "Congo, the Democratic Republic of the"),

("CK", "Cook Islands"),

("CR", "Costa Rica"),

("HR", "Croatia"),

("CU", "Cuba"),

("CW", "Curaçao"),

("CY", "Cyprus"),

("CZ", "Czech Republic"),

("CI", "Côte d'Ivoire"),

("DK", "Denmark"),

("DJ", "Djibouti"),

("DM", "Dominica"),

("DO", "Dominican Republic"),

("EC", "Ecuador"),

("EG", "Egypt"),

("SV", "El Salvador"),

("GQ", "Equatorial Guinea"),

("ER", "Eritrea"),

("EE", "Estonia"),

("ET", "Ethiopia"),

("FK", "Falkland Islands (Malvinas)"),

("FO", "Faroe Islands"),

("FJ", "Fiji"),

("FI", "Finland"),

("FR", "France"),

("GF", "French Guiana"),

("PF", "French Polynesia"),

("TF", "French Southern Territories"),

("GA", "Gabon"),

("GM", "Gambia"),

("GE", "Georgia"),

("DE", "Germany"),

("GH", "Ghana"),

("GI", "Gibraltar"),

("GR", "Greece"),

("GL", "Greenland"),

("GD", "Grenada"),

("GP", "Guadeloupe"),

("GU", "Guam"),

("GT", "Guatemala"),

("GG", "Guernsey"),

("GN", "Guinea"),

("GW", "Guinea-Bissau"),

("GY", "Guyana"),

("HT", "Haiti"),

("HM", "Heard Island and McDonald Islands"),

("VA", "Holy See (Vatican City State)"),

("HN", "Honduras"),

("HK", "Hong Kong"),

("HU", "Hungary"),

("IS", "Iceland"),

("IN", "India"),

("ID", "Indonesia"),

("IR", "Iran, Islamic Republic of"),

("IQ", "Iraq"),

("IE", "Ireland"),

("IM", "Isle of Man"),

("IL", "Israel"),

("IT", "Italy"),

("JM", "Jamaica"),

("JP", "Japan"),

("JE", "Jersey"),

("JO", "Jordan"),

("KZ", "Kazakhstan"),

("KE", "Kenya"),

("KI", "Kiribati"),

("KP", "Democratic People's Republic of Korea"),

("KR", "Republic of Korea"),

("KW", "Kuwait"),

("KG", "Kyrgyzstan"),

("LA", "Lao People's Democratic Republic"),

("LV", "Latvia"),

("LB", "Lebanon"),

("LS", "Lesotho"),

("LR", "Liberia"),

("LY", "Libya"),

("LI", "Liechtenstein"),

("LT", "Lithuania"),

("LU", "Luxembourg"),

("MO", "Macao"),

("MK", "Macedonia, the Former Yugoslav Republic of"),

("MG", "Madagascar"),

("MW", "Malawi"),

("MY", "Malaysia"),

("MV", "Maldives"),

("ML", "Mali"),

("MT", "Malta"),

("MH", "Marshall Islands"),

("MQ", "Martinique"),

("MR", "Mauritania"),

("MU", "Mauritius"),

("YT", "Mayotte"),

("MX", "Mexico"),

("FM", "Micronesia, Federated States of"),

("MD", "Moldova, Republic of"),

("MC", "Monaco"),

("MN", "Mongolia"),

("ME", "Montenegro"),

("MS", "Montserrat"),

("MA", "Morocco"),

("MZ", "Mozambique"),

("MM", "Myanmar"),

("NA", "Namibia"),

("NR", "Nauru"),

("NP", "Nepal"),

("NL", "Netherlands"),

("NC", "New Caledonia"),

("NZ", "New Zealand"),

("NI", "Nicaragua"),

("NE", "Niger"),

("NG", "Nigeria"),

("NU", "Niue"),

("NF", "Norfolk Island"),

("MP", "Northern Mariana Islands"),

("NO", "Norway"),

("OM", "Oman"),

("PK", "Pakistan"),

("PW", "Palau"),

("PS", "Palestine, State of"),

("PA", "Panama"),

("PG", "Papua New Guinea"),

("PY", "Paraguay"),

("PE", "Peru"),

("PH", "Philippines"),

("PN", "Pitcairn"),

("PL", "Poland"),

("PT", "Portugal"),

("PR", "Puerto Rico"),

("QA", "Qatar"),

("RO", "Romania"),

("RU", "Russian Federation"),

("RW", "Rwanda"),

("RE", "Réunion"),

("BL", "Saint Barthélemy"),

("SH", "Saint Helena, Ascension and Tristan da Cunha"),

("KN", "Saint Kitts and Nevis"),

("LC", "Saint Lucia"),

("MF", "Saint Martin (French part)"),

("PM", "Saint Pierre and Miquelon"),

("VC", "Saint Vincent and the Grenadines"),

("WS", "Samoa"),

("SM", "San Marino"),

("ST", "Sao Tome and Principe"),

("SA", "Saudi Arabia"),

("SN", "Senegal"),

("RS", "Serbia"),

("SC", "Seychelles"),

("SL", "Sierra Leone"),

("SG", "Singapore"),

("SX", "Sint Maarten (Dutch part)"),

("SK", "Slovakia"),

("SI", "Slovenia"),

("SB", "Solomon Islands"),

("SO", "Somalia"),

("ZA", "South Africa"),

("GS", "South Georgia and the South Sandwich Islands"),

("SS", "South Sudan"),

("ES", "Spain"),

("LK", "Sri Lanka"),

("SD", "Sudan"),

("SR", "Suriname"),

("SJ", "Svalbard and Jan Mayen"),

("SZ", "Swaziland"),

("SE", "Sweden"),

("CH", "Switzerland"),

("SY", "Syrian Arab Republic"),

("TW", "Taiwan, Province of China"),

("TJ", "Tajikistan"),

("TZ", "Tanzania, United Republic of"),

("TH", "Thailand"),

("TL", "Timor-Leste"),

("TG", "Togo"),

("TK", "Tokelau"),

("TO", "Tonga"),

("TT", "Trinidad and Tobago"),

("TN", "Tunisia"),

("TR", "Turkey"),

("TM", "Turkmenistan"),

("TC", "Turks and Caicos Islands"),

("TV", "Tuvalu"),

("UG", "Uganda"),

("UA", "Ukraine"),

("AE", "United Arab Emirates"),

("GB", "United Kingdom"),

("US", "United States"),

("UM", "United States Minor Outlying Islands"),

("UY", "Uruguay"),

("UZ", "Uzbekistan"),

("VU", "Vanuatu"),

("VE", "Venezuela, Bolivarian Republic of"),

("VN", "Viet Nam"),

("VG", "Virgin Islands, British"),

("VI", "Virgin Islands, U.S."),

("WF", "Wallis and Futuna"),

("EH", "Western Sahara"),

("YE", "Yemen"),

("ZM", "Zambia"),

("ZW", "Zimbabwe"),

("AX", "Åland Islands");

**Output:**

