**Hands on 1**

**Write queries on country table using Query Methods**

**CodeApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CodeApplication {

public static void main(String[] args) {

SpringApplication.run(CodeApplication.class, args);

}

}

**Column.java**

package com.example.ormlearn.model;

public @interface Column {

}

**Country.java**

package com.example.ormlearn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import lombok.Getter;

import lombok.Setter;

import lombok.ToString;

@Entity

@Table(name = "country")

@Getter

@Setter

@ToString

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;

}

}

**Entity.java**

package com.example.ormlearn.model;

public @interface Entity {

}

**Table.java**

package com.example.ormlearn.model;

public @interface Table {

}

**CountryRepository.java**

package com.example.ormlearn.repository; // Adjust package name as needed

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.example.ormlearn.model.Country; // Adjust package name as needed

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

// Query Method 1: Search by containing text

List<Country> findByNameContaining(String text);

// Query Method 2: Search by containing text and sort by name in ascending order

List<Country> findByNameContainingOrderByNameAsc(String text);

// Query Method 3: Search by name starting with a given alphabet

List<Country> findByNameStartingWith(String startingCharacter);

}

**OrmLearnApplication.java**

package com.example.ormlearn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.Bean;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.repository.CountryRepository;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static CountryRepository countryRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

// countryRepository is now injected via CommandLineRunner for better practice

LOGGER.info("Inside main");

}

@Bean

public CommandLineRunner run(CountryRepository repository) {

return args -> {

countryRepository = repository; // Assign the injected repository

// Add dummy data

repository.save(new Country("BV", "Bouvet Island"));

repository.save(new Country("DJ", "Djibouti"));

repository.save(new Country("GP", "Guadeloupe"));

repository.save(new Country("GS", "South Georgia and the South Sandwich Islands"));

repository.save(new Country("LU", "Luxembourg"));

repository.save(new Country("SS", "South Sudan"));

repository.save(new Country("TF", "French Southern Territories"));

repository.save(new Country("UM", "United States Minor Outlying Islands"));

repository.save(new Country("ZA", "South Africa"));

repository.save(new Country("ZM", "Zambia"));

repository.save(new Country("ZW", "Zimbabwe"));

// Call the methods to test the queries

testFindByNameContaining();

testFindByNameContainingOrderByNameAsc();

testFindByNameStartingWith();

};

}

private static void testFindByNameContaining() {

LOGGER.info("Start testFindByNameContaining");

List<Country> countries = countryRepository.findByNameContaining("ou");

LOGGER.debug("Countries with 'ou': {}", countries);

LOGGER.info("End testFindByNameContaining");

}

private static void testFindByNameContainingOrderByNameAsc() {

LOGGER.info("Start testFindByNameContainingOrderByNameAsc");

List<Country> countries = countryRepository.findByNameContainingOrderByNameAsc("ou");

LOGGER.debug("Countries with 'ou' (sorted): {}", countries);

LOGGER.info("End testFindByNameContainingOrderByNameAsc");

}

private static void testFindByNameStartingWith() {

LOGGER.info("Start testFindByNameStartingWith");

List<Country> countries = countryRepository.findByNameStartingWith("Z");

LOGGER.debug("Countries (starting with Z): {}", countries);

LOGGER.info("End testFindByNameStartingWith");

}

}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.jpa.hibernate.ddl-auto=update

logging.level.org.hibernate.SQL=DEBUG

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

server.port=8081

**CodeApplicationTests.java**

package com.example.demo;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class CodeApplicationTests {

@Test

void contextLoads() {

}

}

**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>3.2.5</version> <relativePath/> </parent>

<groupId>com.example</groupId>

<artifactId>ormlearn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ormlearn</name>

<description>Demo project for Spring Boot ORM Learn</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>org.springframework.boot</groupId>

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

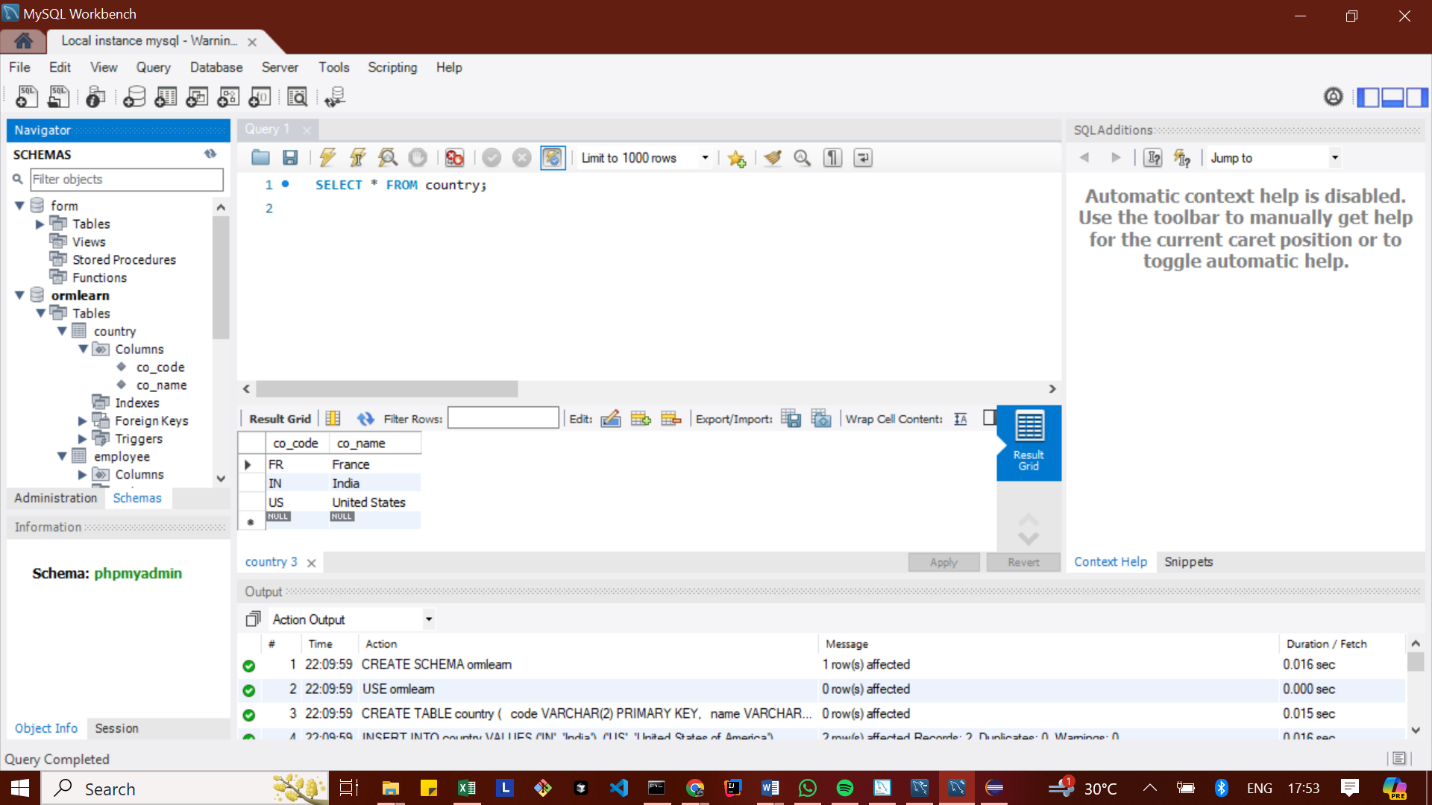
</plugin>

</plugins>

</build>

</project>

**OUTPUT**

****

