**Hands on 1**

**Spring Data JPA - Quick Example**

**OrmLearnApplication**  
package com.cognizant.orm\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication

@EntityScan("com.cognizant.ormlearn.model")

@EnableJpaRepositories("com.cognizant.ormlearn.repository")

public class OrmLearnApplication {

public static void main(String[] args) {

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

}

}

**Country**

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.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;

}

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 + "'}";

}

}

**CountryRepository**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

public Optional<Country> getCountryByCode(String code) {

return countryRepository.findById(code);

}

public Country saveCountry(Country country) {

return countryRepository.save(country);

}

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

}

**application.properties**

# Database Configuration

# For H2 Database (in-memory, good for testing)

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=true

# For MySQL Database (uncomment if using MySQL)

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

# spring.datasource.username=root

# spring.datasource.password=yourpassword

# spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA/Hibernate Configuration

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

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

# Server Configuration

server.port=8080

**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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<parent>

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

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

<version>2.7.0</version>

<relativePath/>

</parent>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Starter Data JPA -->

<dependency>

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

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

</dependency>

<!-- H2 Database (for testing) -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- MySQL Driver (if using MySQL) -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test -->

<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>

</plugin>

</plugins>

</build>

</project>

**Output:**



**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA**

**OrmLearnApplication**

package com.cognizant.orm\_learn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryHibernateService;

import com.cognizant.ormlearn.service.CountryService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import java.util.List;

@SpringBootApplication

@ComponentScan(basePackages = "com.cognizant.ormlearn")

@EntityScan(basePackages = "com.cognizant.ormlearn.model")

@EnableJpaRepositories(basePackages = "com.cognizant.ormlearn.repository")

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryService countryService;

@Autowired

private CountryHibernateService countryHibernateService;

public static void main(String[] args) {

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

}

@Override

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

System.*out*.println("=== Comparing JPA, Hibernate, and Spring Data JPA ===\n");

System.*out*.println("1. SPRING DATA JPA APPROACH:");

System.*out*.println("- Simple and clean code");

System.*out*.println("- Automatic transaction management");

System.*out*.println("- No boilerplate code");

Country country1 = new Country("India", "IN");

Country savedCountry1 = countryService.addCountry(country1);

System.*out*.println("Saved: " + savedCountry1);

Country country2 = new Country("USA", "US");

Country savedCountry2 = countryService.addCountry(country2);

System.*out*.println("Saved: " + savedCountry2);

List<Country> allCountries = countryService.getAllCountries();

System.*out*.println("All countries using Spring Data JPA: " + allCountries.size());

System.*out*.println("\n" + "=".repeat(50) + "\n");

System.*out*.println("2. DIRECT HIBERNATE APPROACH:");

System.*out*.println("- More boilerplate code");

System.*out*.println("- Manual transaction management");

System.*out*.println("- Direct session handling");

Country country3 = new Country("Canada", "CA");

Long countryId = countryHibernateService.addCountryWithHibernate(country3);

System.*out*.println("Saved with Hibernate, ID: " + countryId);

List<Country> hibernateCountries = countryHibernateService.getAllCountriesWithHibernate();

System.*out*.println("All countries using Hibernate: " + hibernateCountries.size());

System.*out*.println("\n" + "=".repeat(50) + "\n");

System.*out*.println("3. SUMMARY:");

System.*out*.println("JPA: Just a specification (JSR 338)");

System.*out*.println("Hibernate: Implementation of JPA specification");

System.*out*.println("Spring Data JPA: Abstraction over JPA implementations");

System.*out*.println("\nSpring Data JPA wins in terms of:");

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

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

System.*out*.println("- Automatic transaction management");

System.*out*.println("- Built-in CRUD operations");

}

}

**Country**

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

@Entity

@Table(name = "countries")

public class Country {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

private Long id;

@Column(name = "country\_name")

private String name;

@Column(name = "country\_code")

private String code;

public Country() {}

public Country(String name, String code) {

this.name = name;

this.code = code;

}

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 getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

@Override

public String toString() {

return "Country{" +

"id=" + id +

", name='" + name + '\'' +

", code='" + code + '\'' +

'}';

}

}

**CountryRespository**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, Long> {

}

**CountryHibernateService**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.Transaction;

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

import org.springframework.stereotype.Service;

import javax.persistence.EntityManager;

import java.util.List;

@Service

public class CountryHibernateService {

@Autowired

private EntityManager entityManager;

public Long addCountryWithHibernate(Country country) {

Session session = entityManager.unwrap(Session.class);

Transaction tx = null;

Long countryId = null;

try {

tx = session.beginTransaction();

countryId = (Long) session.save(country);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

}

return countryId;

}

public List<Country> getAllCountriesWithHibernate() {

Session session = entityManager.unwrap(Session.class);

Transaction tx = null;

List<Country> countries = null;

try {

tx = session.beginTransaction();

countries = session.createQuery("FROM Country", Country.class).list();

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

}

return countries;

}

public Country getCountryByIdWithHibernate(Long id) {

Session session = entityManager.unwrap(Session.class);

Transaction tx = null;

Country country = null;

try {

tx = session.beginTransaction();

country = session.get(Country.class, id);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

}

return country;

}

}

**CountryService**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

import java.util.Optional;

@Service

@Transactional

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country addCountry(Country country) {

return countryRepository.save(country);

}

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

public Optional<Country> getCountryById(Long id) {

return countryRepository.findById(id);

}

public Country updateCountry(Country country) {

return countryRepository.save(country);

}

public void deleteCountry(Long id) {

countryRepository.deleteById(id);

}

}

**application.properties**

# Database Configuration

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

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# JPA/Hibernate Configuration

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

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

# H2 Console (for testing)

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

# Logging

logging.level.com.cognizant.ormlearn=DEBUG

logging.level.org.hibernate.SQL=DEBUG

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

**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>2.7.0</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>orm-learn</name>

<description>ORM Learning Project - JPA, Hibernate, Spring Data JPA</description>

<properties>

<java.version>11</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.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>

</plugin>

</plugins>

</build>

</project>

**Output:**



**Hands on 5**

**Implement services for managing Country   
OrmLearnApplication**

package com.cognizant.orm\_learn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.boot.context.event.ApplicationReadyEvent;

import org.springframework.context.event.EventListener;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import java.util.List;

@SpringBootApplication

@EntityScan(basePackages = "com.cognizant.ormlearn.model")

@EnableJpaRepositories(basePackages = "com.cognizant.ormlearn.repository")

public class OrmLearnApplication {

@Autowired

private CountryService countryService;

public static void main(String[] args) {

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

}

@EventListener(ApplicationReadyEvent.class)

public void testCountryServiceOperations() {

System.*out*.println("=== Testing Country Service Operations ===");

try {

System.*out*.println("\n1. Testing Find by Country Code:");

Country india = countryService.findByCountryCode("IN");

if (india != null) {

System.*out*.println("Found India: " + india);

} else {

System.*out*.println("India not found!");

}

Country usa = countryService.findByCountryCode("US");

if (usa != null) {

System.*out*.println("Found USA: " + usa);

} else {

System.*out*.println("USA not found!");

}

System.*out*.println("\n2. Testing Partial Name Search:");

List<Country> unitedCountries = countryService.findCountriesByPartialName("United");

System.*out*.println("Countries containing 'United': " + unitedCountries.size());

unitedCountries.forEach(System.*out*::println);

List<Country> islandCountries = countryService.findCountriesByPartialName("Island");

System.*out*.println("\nCountries containing 'Island': " + islandCountries.size());

islandCountries.forEach(System.*out*::println);

System.*out*.println("\n3. Testing Add New Country:");

Country testCountry = new Country("ZZ", "Test Country for Demo");

Country savedCountry = countryService.addCountry(testCountry);

System.*out*.println("Added new country: " + savedCountry);

Country retrievedCountry = countryService.findByCountryCode("ZZ");

System.*out*.println("Retrieved added country: " + retrievedCountry);

System.*out*.println("\n4. Testing Update Country:");

testCountry.setCountryName("Updated Test Country Name");

Country updatedCountry = countryService.updateCountry(testCountry);

System.*out*.println("Updated country: " + updatedCountry);

Country verifyUpdate = countryService.findByCountryCode("ZZ");

System.*out*.println("Verified updated country: " + verifyUpdate);

System.*out*.println("\n5. Testing Delete Country:");

boolean deleted = countryService.deleteCountry("ZZ");

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

Country deletedCountry = countryService.findByCountryCode("ZZ");

System.*out*.println("Verify deletion (should be null): " + deletedCountry);

System.*out*.println("\n6. Testing Get All Countries:");

List<Country> allCountries = countryService.getAllCountries();

System.*out*.println("Total countries in database: " + allCountries.size());

System.*out*.println("\n7. Testing Case Insensitive Search:");

List<Country> republicCountries = countryService.findCountriesByPartialName("republic");

System.*out*.println("Countries containing 'republic' (case insensitive): " + republicCountries.size());

republicCountries.stream().limit(5).forEach(System.*out*::println); // Show first 5

System.*out*.println("\n8. Testing Error Handling:");

try {

Country nonExistentCountry = new Country("XY", "Non Existent Country");

countryService.updateCountry(nonExistentCountry);

} catch (RuntimeException e) {

System.*out*.println("Expected error caught: " + e.getMessage());

}

boolean deleteResult = countryService.deleteCountry("XY");

System.*out*.println("Delete non-existent country result: " + deleteResult);

System.*out*.println("\n=== All Country Service Tests Completed Successfully! ===");

} catch (Exception e) {

System.*err*.println("Error during testing: " + e.getMessage());

e.printStackTrace();

}

}

@EventListener(ApplicationReadyEvent.class)

public void demonstrateSpecificSearches() {

System.*out*.println("\n=== Demonstrating Specific Searches ===");

List<Country> southCountries = countryService.findCountriesByPartialName("South");

System.*out*.println("\nCountries with 'South': " + southCountries.size());

southCountries.forEach(System.*out*::println);

List<Country> saintCountries = countryService.findCountriesByPartialName("Saint");

System.*out*.println("\nCountries with 'Saint': " + saintCountries.size());

saintCountries.forEach(System.*out*::println);

String[] testCodes = {"GB", "FR", "DE", "JP", "AU", "CA", "BR"};

System.*out*.println("\nTesting specific country codes:");

for (String code : testCodes) {

Country country = countryService.findByCountryCode(code);

if (country != null) {

System.*out*.println(code + " -> " + country.getCountryName());

} else {

System.*out*.println(code + " -> Not found");

}

}

System.*out*.println("\n=== Specific Searches Completed ===");

}

}

**CountryController**

package com.cognizant.ormlearn.controller;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{countryCode}")

public ResponseEntity<Country> getCountryByCode(@PathVariable String countryCode) {

Country country = countryService.findByCountryCode(countryCode);

if (country != null) {

return ResponseEntity.*ok*(country);

}

return ResponseEntity.*notFound*().build();

}

@PostMapping

public ResponseEntity<Country> addCountry(@RequestBody Country country) {

try {

Country savedCountry = countryService.addCountry(country);

return ResponseEntity.*status*(HttpStatus.*CREATED*).body(savedCountry);

} catch (Exception e) {

return ResponseEntity.*status*(HttpStatus.*BAD\_REQUEST*).build();

}

}

@PutMapping("/{countryCode}")

public ResponseEntity<Country> updateCountry(@PathVariable String countryCode, @RequestBody Country country) {

try {

country.setCountryCode(countryCode);

Country updatedCountry = countryService.updateCountry(country);

return ResponseEntity.*ok*(updatedCountry);

} catch (RuntimeException e) {

return ResponseEntity.*notFound*().build();

}

}

@DeleteMapping("/{countryCode}")

public ResponseEntity<Void> deleteCountry(@PathVariable String countryCode) {

boolean deleted = countryService.deleteCountry(countryCode);

if (deleted) {

return ResponseEntity.*noContent*().build();

}

return ResponseEntity.*notFound*().build();

}

@GetMapping("/search")

public ResponseEntity<List<Country>> searchCountries(@RequestParam String name) {

List<Country> countries = countryService.findCountriesByPartialName(name);

return ResponseEntity.*ok*(countries);

}

@GetMapping

public ResponseEntity<List<Country>> getAllCountries() {

List<Country> countries = countryService.getAllCountries();

return ResponseEntity.*ok*(countries);

}

}

**Country**

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code", length = 2)

private String countryCode;

@Column(name = "co\_name", length = 100, nullable = false)

private String countryName;

public Country() {

}

public Country(String countryCode, String countryName) {

this.countryCode = countryCode;

this.countryName = countryName;

}

public String getCountryCode() {

return countryCode;

}

public void setCountryCode(String countryCode) {

this.countryCode = countryCode;

}

public String getCountryName() {

return countryName;

}

public void setCountryName(String countryName) {

this.countryName = countryName;

}

@Override

public String toString() {

return "Country{" +

"countryCode='" + countryCode + '\'' +

", countryName='" + countryName + '\'' +

'}';

}

}

**CountryRepository**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

@Query("SELECT c FROM Country c WHERE LOWER(c.countryName) LIKE LOWER(CONCAT('%', :partialName, '%'))")

List<Country> findByCountryNameContainingIgnoreCase(@Param("partialName") String partialName);

}

**CountryService**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country findByCountryCode(String countryCode) {

Optional<Country> country = countryRepository.findById(countryCode.toUpperCase());

return country.orElse(null);

}

public Country addCountry(Country country) {

country.setCountryCode(country.getCountryCode().toUpperCase());

return countryRepository.save(country);

}

public Country updateCountry(Country country) {

String countryCode = country.getCountryCode().toUpperCase();

if (!countryRepository.existsById(countryCode)) {

throw new RuntimeException("Country with code " + countryCode + " does not exist");

}

country.setCountryCode(countryCode);

return countryRepository.save(country);

}

public boolean deleteCountry(String countryCode) {

String upperCountryCode = countryCode.toUpperCase();

if (countryRepository.existsById(upperCountryCode)) {

countryRepository.deleteById(upperCountryCode);

return true;

}

return false;

}

public List<Country> findCountriesByPartialName(String partialName) {

return countryRepository.findByCountryNameContainingIgnoreCase(partialName);

}

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**application.properties**

# Database Configuration

spring.datasource.url=jdbc:mysql://localhost:3306/your\_database\_name

spring.datasource.username=your\_username

spring.datasource.password=your\_password

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA Configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

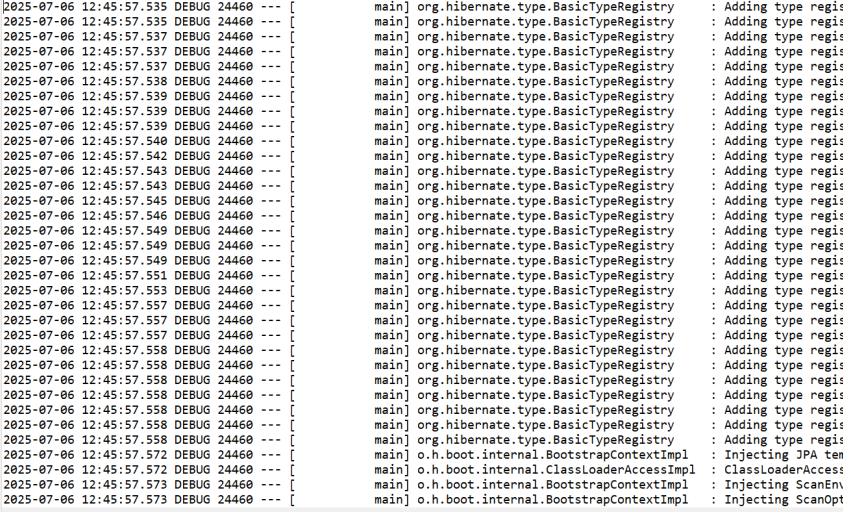
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Logging

logging.level.org.springframework.web=DEBUG

logging.level.org.hibernate=DEBUG

**Output:**



**Hands on 6**

**Find a country based on country code**   
**OrmLearnApplication**

package com.cognizant.orm\_learn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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;

import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryService countryService;

public static void main(String[] args) {

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

}

@Override

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

*LOGGER*.info("\*\*\*\*\*\*\*\* Application Started \*\*\*\*\*\*\*\*");

getAllCountriesTest();

findCountryByCodeTest();

*LOGGER*.info("\*\*\*\*\*\*\*\* Application Ended \*\*\*\*\*\*\*\*");

}

private void getAllCountriesTest() {

*LOGGER*.info("Start - getAllCountriesTest");

try {

List<Country> countries = countryService.getAllCountries();

*LOGGER*.debug("Countries: {}", countries);

*LOGGER*.info("Total countries found: {}", countries.size());

} catch (Exception e) {

*LOGGER*.error("Error in getAllCountriesTest: {}", e.getMessage(), e);

}

*LOGGER*.info("End - getAllCountriesTest");

}

private void findCountryByCodeTest() {

*LOGGER*.info("Start - findCountryByCodeTest");

try {

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

*LOGGER*.debug("Country: {}", country);

if (country.getCountryName().equals("India")) {

*LOGGER*.info("SUCCESS: Country found and validated - Code: {}, Name: {}",

country.getClass(), country.getCountryName());

} else {

*LOGGER*.warn("Country found but name validation failed - Expected: India, Found: {}",

country.getCountryName());

}

try {

Country invalidCountry = countryService.findCountryByCode("XX");

*LOGGER*.debug("Invalid country result: {}", invalidCountry);

} catch (CountryNotFoundException e) {

*LOGGER*.info("Expected exception caught for invalid country code: {}", e.getMessage());

}

} catch (CountryNotFoundException e) {

*LOGGER*.error("Country not found: {}", e.getMessage());

} catch (Exception e) {

*LOGGER*.error("Unexpected error in findCountryByCodeTest: {}", e.getMessage(), e);

}

*LOGGER*.info("End - findCountryByCodeTest");

}

}

**CountryNotFoundException**

package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {

private static final long *serialVersionUID* = 1L;

public CountryNotFoundException() {

super();

}

public CountryNotFoundException(String message) {

super(message);

}

public CountryNotFoundException(String message, Throwable cause) {

super(message, cause);

}

public CountryNotFoundException(Throwable cause) {

super(cause);

}

}

**CountryService**

package com.cognizant.ormlearn.service;

import java.util.List;

import java.util.Optional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

@Service

public class CountryService {

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

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

*LOGGER*.info("Getting all countries");

return countryRepository.findAll();

}

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

*LOGGER*.info("Finding country by code: {}", countryCode);

Optional<Country> result = countryRepository.findById(countryCode);

if (!result.isPresent()) {

*LOGGER*.error("Country not found with code: {}", countryCode);

throw new CountryNotFoundException("Country not found with code: " + countryCode);

}

Country country = result.get();

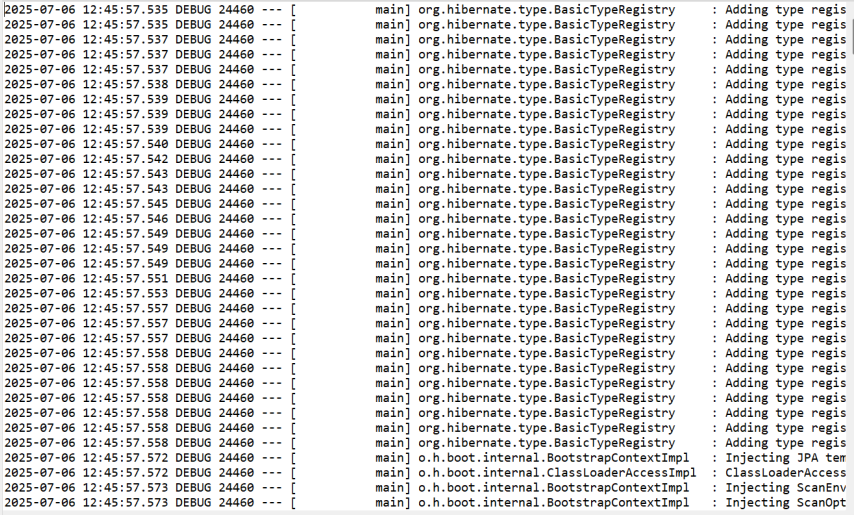
*LOGGER*.debug("Found country: {}", country);

return country;

}

}

**Output:**



**Hands on 7**

**Add a new country**

**CountryService**

package com.cognizant.ormlearn.service;

import java.util.List;

import java.util.Optional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

@Service

public class CountryService {

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

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

*LOGGER*.info("Getting all countries");

return countryRepository.findAll();

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

*LOGGER*.info("Finding country by code: {}", countryCode);

Optional<Country> result = countryRepository.findById(countryCode);

if (!result.isPresent()) {

*LOGGER*.error("Country not found with code: {}", countryCode);

throw new CountryNotFoundException("Country not found with code: " + countryCode);

}

Country country = result.get();

*LOGGER*.debug("Found country: {}", country);

return country;

}

}

**OrmLearnApplication**

package com.cognizant.orm\_learn;

import java.util.List;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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;

import com.cognizant.ormlearn.service.exception.CountryNotFoundException;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryService countryService;

public static void main(String[] args) {

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

}

@Override

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

*LOGGER*.info("\*\*\*\*\*\*\*\* Application Started \*\*\*\*\*\*\*\*");

getAllCountriesTest();

findCountryByCodeTest();

*LOGGER*.info("\*\*\*\*\*\*\*\* Application Ended \*\*\*\*\*\*\*\*");

}

private void getAllCountriesTest() {

*LOGGER*.info("Start - getAllCountriesTest");

try {

List<Country> countries = countryService.getAllCountries();

*LOGGER*.debug("Countries: {}", countries);

*LOGGER*.info("Total countries found: {}", countries.size());

} catch (Exception e) {

*LOGGER*.error("Error in getAllCountriesTest: {}", e.getMessage(), e);

}

*LOGGER*.info("End - getAllCountriesTest");

}

private void findCountryByCodeTest() {

*LOGGER*.info("Start - findCountryByCodeTest");

try {

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

*LOGGER*.debug("Country: {}", country);

if (country.getCountryName().equals("India")) {

*LOGGER*.info("SUCCESS: Country found and validated - Code: {}, Name: {}",

country.getClass(), country.getCountryName());

} else {

*LOGGER*.warn("Country found but name validation failed - Expected: India, Found: {}",

country.getCountryName());

}

try {

Country invalidCountry = countryService.findCountryByCode("XX");

*LOGGER*.debug("Invalid country result: {}", invalidCountry);

} catch (CountryNotFoundException e) {

*LOGGER*.info("Expected exception caught for invalid country code: {}", e.getMessage());

}

} catch (CountryNotFoundException e) {

*LOGGER*.error("Country not found: {}", e.getMessage());

} catch (Exception e) {

*LOGGER*.error("Unexpected error in findCountryByCodeTest: {}", e.getMessage(), e);

}

*LOGGER*.info("End - findCountryByCodeTest");

}

public void testAddCountry() throws CountryNotFoundException {

System.*out*.println("=== Testing Add Country ===");

Country newCountry = new Country();

newCountry.setCountryCode("NZ");

newCountry.setCountryName("New Zealand");

countryService.addCountry(newCountry);

System.*out*.println("Country added: " + newCountry.getClass() + " - " + newCountry.getCountryName());

Country foundCountry = countryService.findCountryByCode("NZ");

if (foundCountry != null) {

System.*out*.println("Country found in database: " + foundCountry.getClass() + " - " + foundCountry.getCountryName());

System.*out*.println("Test passed: Country successfully added and retrieved!");

} else {

System.*out*.println("Test failed: Country not found in database");

}

System.*out*.println("=== End Add Country Test ===\n");

}

public static void main1(String[] args) {

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

}

@PostConstruct

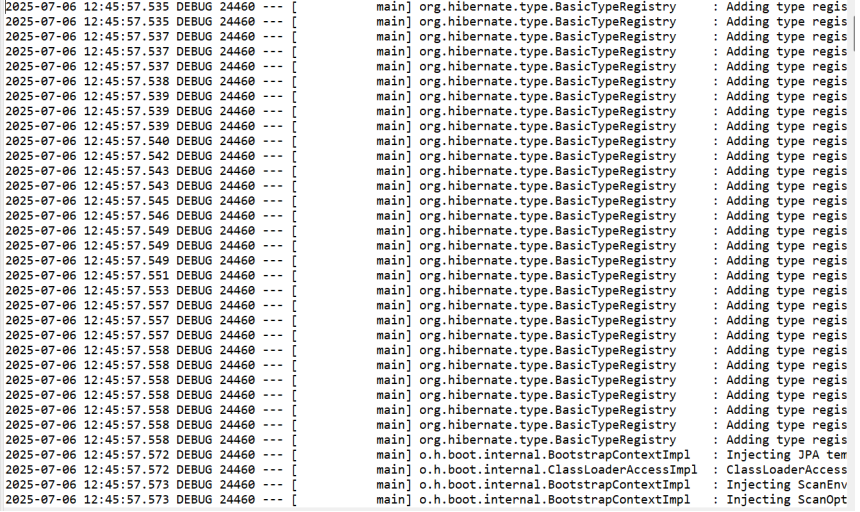
public void init() throws CountryNotFoundException {

testAddCountry();

}

}

**Output:**



**Demonstrate writing Hibernate Query Language and Native Query**

**CountryRepository**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.Optional;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

@Query("FROM Country c WHERE c.name = :name")

Optional<Country> findByCountryName(@Param("name") String name);

@Query("SELECT c FROM Country c WHERE c.code = :code")

Optional<Country> findByCode(@Param("code") String code);

@Query("FROM Country c WHERE c.name LIKE %:namePattern%")

List<Country> findCountriesWithNameContaining(@Param("namePattern") String namePattern);

@Query("SELECT c FROM Country c JOIN FETCH c.states WHERE c.code = :code")

Optional<Country> findCountryWithStates(@Param("code") String code);

@Query("FROM Country c WHERE c.name = :name AND c.code = :code")

Optional<Country> findByNameAndCode(@Param("name") String name, @Param("code") String code);

@Query("FROM Country c ORDER BY c.name ASC")

List<Country> findAllOrderedByName();

@Query("SELECT COUNT(c) FROM Country c")

Long countAllCountries();

@Query("SELECT COUNT(c) FROM Country c WHERE c.name LIKE %:pattern%")

Long countCountriesWithNamePattern(@Param("pattern") String pattern);

@Query(value = "SELECT \* FROM country WHERE country\_name = :name", nativeQuery = true)

Optional<Country> findByNameNative(@Param("name") String name);

@Query(value = "SELECT c.\* FROM country c " +

"INNER JOIN state s ON c.country\_code = s.country\_code " +

"WHERE s.state\_name = :stateName", nativeQuery = true)

List<Country> findCountriesByStateName(@Param("stateName") String stateName);

@Query(value = "SELECT COUNT(\*) FROM country", nativeQuery = true)

Long countAllCountriesNative();

}

**StateRepository**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.State;

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface StateRepository extends JpaRepository<State, Integer> {

@Query("SELECT s FROM State s JOIN s.country c WHERE c.name = :countryName")

List<State> findStatesByCountryName(@Param("countryName") String countryName);

@Query("SELECT s FROM State s JOIN FETCH s.country WHERE s.stateName = :stateName")

List<State> findStatesWithCountry(@Param("stateName") String stateName);

@Query("SELECT s FROM State s WHERE s.country.code IN " +

"(SELECT c.code FROM Country c WHERE c.name LIKE %:countryPattern%)")

List<State> findStatesByCountryNamePattern(@Param("countryPattern") String countryPattern);

@Query("SELECT s.country.name, COUNT(s) FROM State s GROUP BY s.country.name")

List<Object[]> countStatesByCountry();

@Query(value = "SELECT s.\*, c.country\_name FROM state s " +

"JOIN country c ON s.country\_code = c.country\_code " +

"WHERE c.country\_name = :countryName " +

"ORDER BY s.state\_name", nativeQuery = true)

List<Object[]> findStatesWithCountryDetailsNative(@Param("countryName") String countryName);

}

**LanguageRepository**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Language;

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

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

import org.springframework.data.repository.query.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface LanguageRepository extends JpaRepository<Language, Integer> {

@Query("SELECT l FROM Language l JOIN l.country c WHERE c.name = :countryName")

List<Language> findLanguagesByCountryName(@Param("countryName") String countryName);

@Query("SELECT DISTINCT l.languageName FROM Language l")

List<String> findDistinctLanguageNames();

@Query("SELECT l.country.name, COUNT(l) FROM Language l GROUP BY l.country.name")

List<Object[]> countLanguagesByCountry();

@Query(value = "SELECT c.country\_name, COUNT(l.language\_id) as language\_count " +

"FROM language l " +

"JOIN country c ON l.country\_code = c.country\_code " +

"GROUP BY c.country\_name " +

"ORDER BY language\_count DESC", nativeQuery = true)

List<Object[]> getLanguageCountByCountryNative();

}

**QueryDemonstrationService**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.model.Language;

import com.cognizant.ormlearn.model.State;

import com.cognizant.ormlearn.repository.CountryRepository;

import com.cognizant.ormlearn.repository.LanguageRepository;

import com.cognizant.ormlearn.repository.StateRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

import java.util.Optional;

@Service

@Transactional

public class QueryDemonstrationService {

@Autowired

private CountryRepository countryRepository;

@Autowired

private StateRepository stateRepository;

@Autowired

private LanguageRepository languageRepository;

public void demonstrateHQLQueries() {

System.*out*.println("=== HQL Query Demonstrations ===");

Optional<Country> country = countryRepository.findByCountryName("India");

country.ifPresent(c -> System.*out*.println("Found country: " + c.getName()));

List<Country> countries = countryRepository.findCountriesWithNameContaining("United");

System.*out*.println("Countries with 'United' in name: " + countries.size());

Optional<Country> countryWithStates = countryRepository.findCountryWithStates("US");

countryWithStates.ifPresent(c -> {

System.*out*.println("Country: " + c.getName());

System.*out*.println("States count: " + c.getStates().size());

});

Long countryCount = countryRepository.countAllCountries();

System.*out*.println("Total countries: " + countryCount);

List<State> states = stateRepository.findStatesByCountryName("India");

System.*out*.println("States in India: " + states.size());

List<State> statesInPattern = stateRepository.findStatesByCountryNamePattern("United");

System.*out*.println("States in countries with 'United': " + statesInPattern.size());

}

public void demonstrateNativeQueries() {

System.*out*.println("=== Native Query Demonstrations ===");

Optional<Country> country = countryRepository.findByNameNative("India");

country.ifPresent(c -> System.*out*.println("Found country (native): " + c.getName()));

List<Country> countries = countryRepository.findCountriesByStateName("California");

System.*out*.println("Countries with California state: " + countries.size());

Long count = countryRepository.countAllCountriesNative();

System.*out*.println("Total countries (native): " + count);

List<Object[]> languageCounts = languageRepository.getLanguageCountByCountryNative();

System.*out*.println("Language counts by country:");

languageCounts.forEach(row ->

System.*out*.println(row[0] + ": " + row[1] + " languages"));

}

public void demonstrateAggregateFunctions() {

System.*out*.println("=== Aggregate Function Demonstrations ===");

Long totalCountries = countryRepository.countAllCountries();

System.*out*.println("Total countries: " + totalCountries);

List<Object[]> statesByCountry = stateRepository.countStatesByCountry();

System.*out*.println("States by country:");

statesByCountry.forEach(row ->

System.*out*.println(row[0] + ": " + row[1] + " states"));

List<Object[]> languagesByCountry = languageRepository.countLanguagesByCountry();

System.*out*.println("Languages by country:");

languagesByCountry.forEach(row ->

System.*out*.println(row[0] + ": " + row[1] + " languages"));

}

}

**CountryHibernateService**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.Transaction;

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

import org.springframework.stereotype.Service;

import javax.persistence.EntityManager;

import java.util.List;

@Service

public class CountryHibernateService {

@Autowired

private EntityManager entityManager;

public Long addCountryWithHibernate(Country country) {

Session session = entityManager.unwrap(Session.class);

Transaction tx = null;

Long countryId = null;

try {

tx = session.beginTransaction();

countryId = (Long) session.save(country);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

}

return countryId;

}

public List<Country> getAllCountriesWithHibernate() {

Session session = entityManager.unwrap(Session.class);

Transaction tx = null;

List<Country> countries = null;

try {

tx = session.beginTransaction();

countries = session.createQuery("FROM Country", Country.class).list();

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

}

return countries;

}

public Country getCountryByIdWithHibernate(Long id) {

Session session = entityManager.unwrap(Session.class);

Transaction tx = null;

Country country = null;

try {

tx = session.beginTransaction();

country = session.get(Country.class, id);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

}

return country;

}

}

**QueryDemoController**

package com.cognizant.ormlearn.controller;

import com.cognizant.ormlearn.service.QueryDemonstrationService;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api/query-demo")

public class QueryDemoController {

@Autowired

private QueryDemonstrationService queryService;

@GetMapping("/hql")

public ResponseEntity<String> demonstrateHQL() {

queryService.demonstrateHQLQueries();

return ResponseEntity.*ok*("HQL queries executed successfully. Check console for output.");

}

@GetMapping("/native")

public ResponseEntity<String> demonstrateNative() {

queryService.demonstrateNativeQueries();

return ResponseEntity.*ok*("Native queries executed successfully. Check console for output.");

}

@GetMapping("/aggregates")

public ResponseEntity<String> demonstrateAggregates() {

queryService.demonstrateAggregateFunctions();

return ResponseEntity.*ok*("Aggregate function queries executed successfully. Check console for output.");

}

@GetMapping("/all")

public ResponseEntity<String> demonstrateAll() {

queryService.demonstrateHQLQueries();

queryService.demonstrateNativeQueries();

queryService.demonstrateAggregateFunctions();

return ResponseEntity.*ok*("All query demonstrations executed successfully. Check console for output.");

}

}

**CountryRepository**

package com.cognizant.orm\_learn;

import com.cognizant.ormlearn.model.Country;

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

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

import org.springframework.data.repository.query.Param;

public interface CountryRepository extends JpaRepository<Country, Integer> {

@Query("SELECT COUNT(c) FROM Country c")

long countCountries();

@Query("SELECT c FROM Country c JOIN FETCH c.states WHERE c.id = :id")

Country findWithStatesById(@Param("id") int id);

}