**Exercise-1 Spring Data JPA - Quick Example**

// pom.xml

<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>3.5.3</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>org.springframework.boot</groupId>

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

<scope>runtime</scope>

<optional>true</optional>

</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>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

// application.properties

spring.application.name=orm-learn

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

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

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=@456

spring.jpa.hibernate.ddl-auto=update

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

spring.jpa.show-sql=true

// Country.java

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

private String 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.java

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

// 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.repository.CountryRepository;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryRepository countryRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

System.out.println("Inside main");

}

@Override

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

System.out.println("Start");

List<Country> countries = countryRepository.findAll();

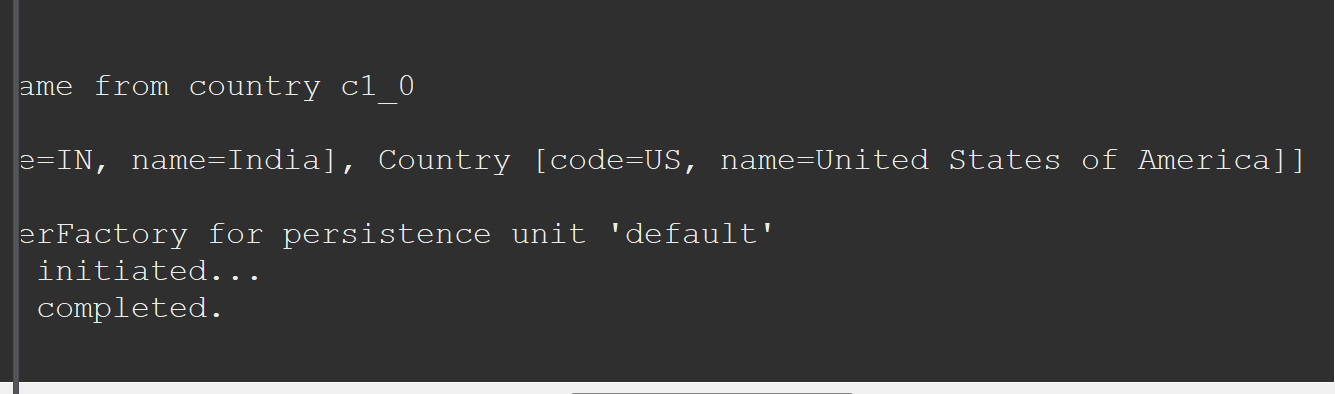
System.out.println("countries=" + countries);

System.out.println("End");

}

}

**OUTPUT:**

****

**Exercise-4 Difference between JPA, Hibernate and Spring Data JPA**

### **1. Introduction**

This hands-on exercise demonstrates the differences between Java Persistence API (JPA), Hibernate ORM framework, and Spring Data JPA. It includes conceptual explanations and example code snippets showing how each approach is used to persist an Employee entity into a database.

### **2. JPA Overview**

Java Persistence API (JPA) is a specification for object-relational mapping in Java. It provides annotations and interfaces for persisting Java objects but requires an implementation like Hibernate or EclipseLink to perform the actual database operations.

### **3. Hibernate Example (Code & Explanation)**

// Hibernate Example: Saving an Employee

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

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

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**Explanation**:

* Manually manages sessions and transactions using Hibernate's API.
* More control but results in more boilerplate code.

### **4. Spring Data JPA Example (Code & Explanation)**

// EmployeeRepository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

// EmployeeService.java

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Explanation**:

* Reduces boilerplate by providing default CRUD operations.
* Automatically manages transactions and integrates well with Spring's ecosystem.

**Output Demonstration**

**Hibernate Output (Console):**

Hibernate: insert into employee (name, age, department) values (?, ?, ?)

Employee added with ID: 1

**Spring Data JPA Output (Console):**

Saving Employee using Spring Data JPA...

Hibernate: insert into employee (name, age, department) values (?, ?, ?)

Employee saved successfully

**Conclusion**

Hibernate and Spring Data JPA both work on top of JPA, but Spring Data JPA significantly reduces development effort by abstracting the repetitive CRUD operations. Hibernate requires more boilerplate and manual transaction/session management.