**Spring Data JPA - Quick Example** (HandsOn 1):

**Code:**

1. pom.xml – Maven Dependencies

<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>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orm-learn</name>

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

<parent>

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

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

<version>3.2.4</version> <!-- Use the latest stable Spring Boot version -->

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>17</java.version> <!-- Spring Boot 3.x requires Java 17+ -->

</properties>

<dependencies>

<!-- Spring Boot Core -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Data JPA (includes Hibernate + Jakarta Persistence) -->

<dependency>

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

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

</dependency>

<!-- MySQL JDBC Driver -->

<dependency>

<groupId>com.mysql</groupId>

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

<scope>runtime</scope>

</dependency>

<!-- Spring Boot DevTools (optional for development) -->

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Testing -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Java Compiler Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

<!-- Spring Boot Maven Plugin -->

<plugin>

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

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

<version>3.2.4</version>

</plugin>

</plugins>

</build>

</project>

1. Country.java – JPA Entity

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.persistence.Column;

*@Entity*

*@Table*(name = "country")

public class Country {

*@Id*

*@Column*(name = "code")

private String code;

*@Column*(name = "name")

private String 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;

}

// toString

*@Override*

public String toString() {

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

}

}

1. CountryRepository.java – Spring Data Repository

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> {

}

1. CountryService.java – Service Layer

package com.cognizant.ormlearn.service;

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 java.util.List;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

1. OrmLearnApplication.java – Main Class

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.service.CountryService;

import com.cognizant.ormlearn.model.Country;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

*@SpringBootApplication*

public class OrmLearnApplication {

private static CountryService *countryService*;

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

public static void main(String[] args) {

ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);

*countryService* = context.getBean(CountryService.class);

***LOGGER***.info("Inside main");

*testGetAllCountries*();

}

private static void testGetAllCountries() {

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

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

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

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

}

}

1. application.properties – Configuration

# Database config

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

spring.datasource.username=root

spring.datasource.password=1234

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

# JPA & Hibernate

spring.jpa.hibernate.ddl-auto=validate

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

# Logging

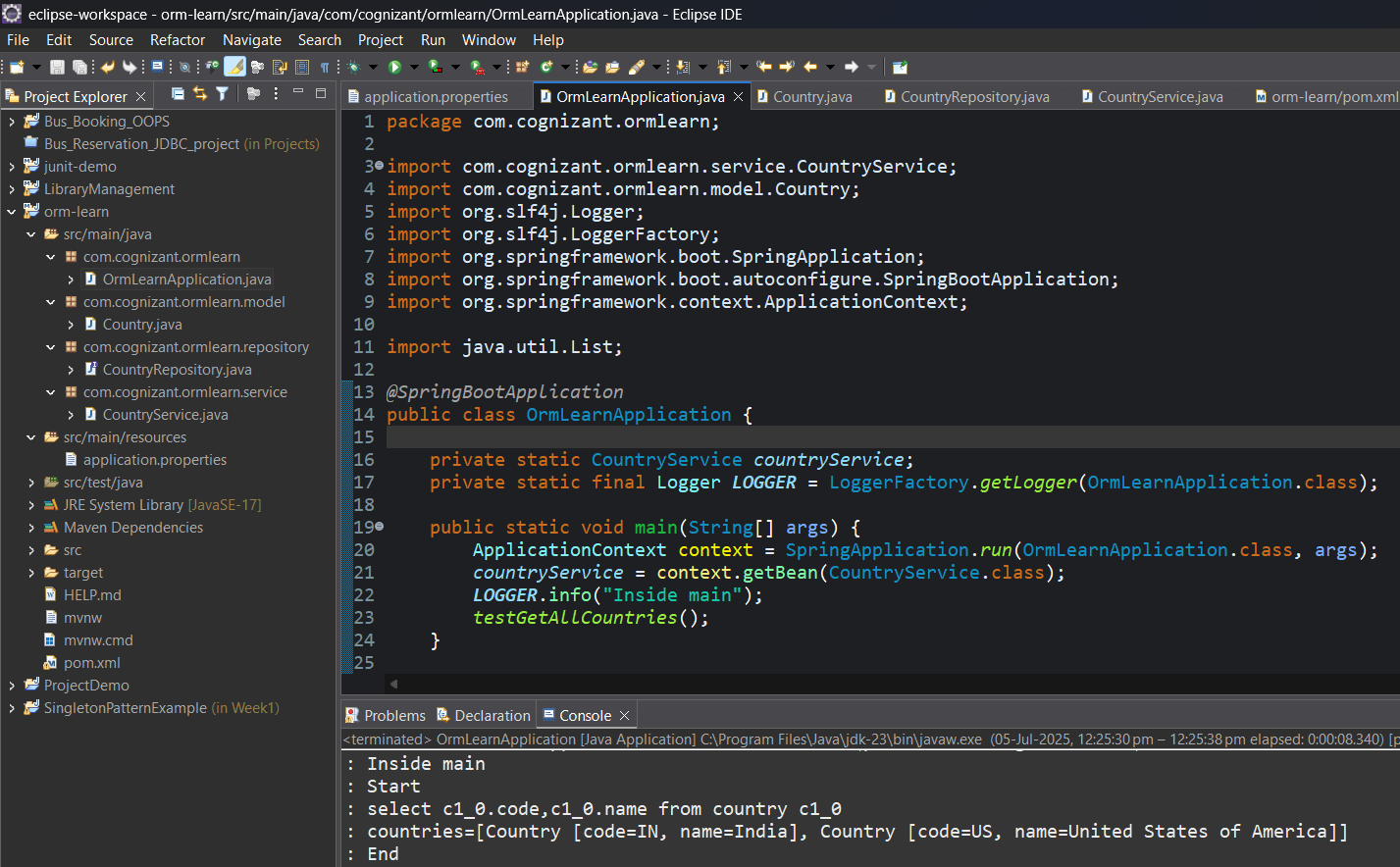
logging.level.org.springframework=INFO

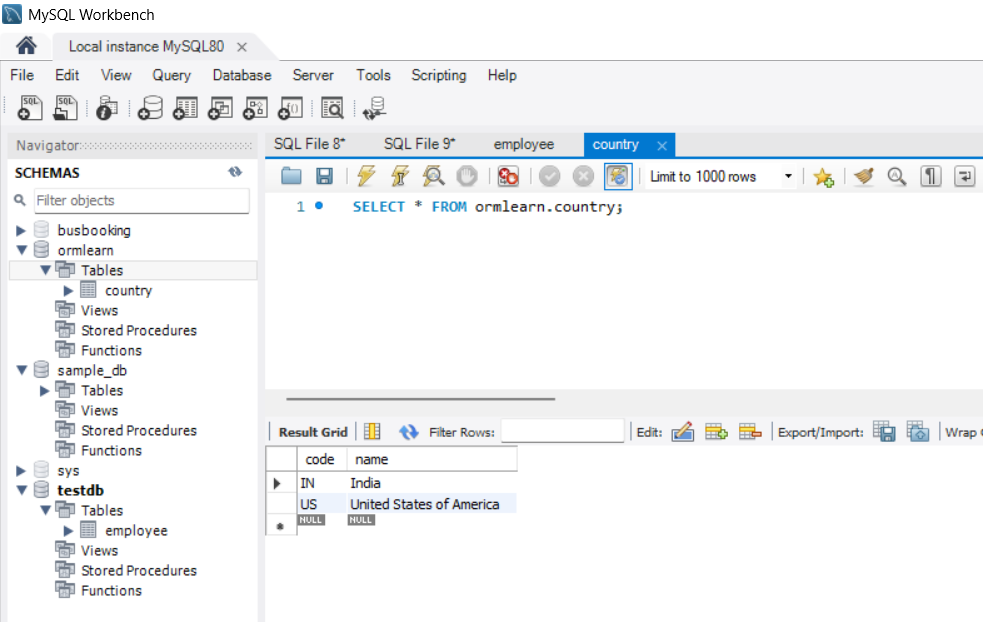
logging.level.com.cognizant=DEBUG

logging.level.org.hibernate.SQL=TRACE

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

**Output:**

****



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

(Handson 2):

**1. JPA (Java Persistence API)**

* **Definition:** It is a **Java specification (JSR 338)** for persisting, reading, and managing data between Java objects and relational databases.
* **Role:** Acts as an abstraction layer, providing a standard API.
* **Note:** It does **not provide any implementation** itself.
* **Examples of Implementations:** Hibernate, EclipseLink, OpenJPA, etc.

**2. Hibernate**

* **Definition:** Hibernate is an **Object-Relational Mapping (ORM) framework** and a **JPA implementation**.
* **Role:** It provides the actual working code that interacts with the database based on the JPA spec.
* **Features:**
  + Implements all JPA features.
  + Adds its own powerful extensions.
  + Requires manual transaction and session management if JPA is not used.

**3. Spring Data JPA**

* **Definition:** It is a **Spring module** that simplifies the use of JPA.
* **Role:** It **abstracts and automates** the boilerplate code (like writing DAOs or repositories).
* **Key Features:**
  + Works **on top of JPA** (typically with Hibernate underneath).
  + Automatically handles common database operations.
  + Integrates with Spring for dependency injection, transactions, and more.

**Task:**

Create a Java console application using **Hibernate only** (no Spring) that:

* Connects to a MySQL database
* Saves an Employee object to the database

**Code:**

1. Add Dependencies to 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>

<groupId>com.example</groupId>

<artifactId>HibernateStandalone</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<!-- Hibernate Core -->

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.4.4.Final</version>

</dependency>

<!-- MySQL JDBC Driver -->

<dependency>

<groupId>com.mysql</groupId>

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

<version>8.3.0</version>

</dependency>

<!-- JPA API -->

<dependency>

<groupId>jakarta.persistence</groupId>

<artifactId>jakarta.persistence-api</artifactId>

<version>3.1.0</version>

</dependency>

</dependencies>

</project

1. Create Hibernate Configuration File

<?**xml** version=*'1.0'* encoding=*'utf-8'*?>

<!**DOCTYPE** hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"https://hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<**hibernate-configuration**>

<**session-factory**>

<!-- Database connection settings -->

<**property** name=*"hibernate.connection.driver\_class"*>com.mysql.cj.jdbc.Driver</**property**>

<**property** name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/testdb</**property**>

<**property** name=*"hibernate.connection.username"*>root</**property**>

<**property** name=*"hibernate.connection.password"*>1234</**property**>

<!-- JDBC connection pool (default is 20) -->

<**property** name=*"hibernate.connection.pool\_size"*>1</**property**>

<!-- SQL dialect -->

<**property** name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</**property**>

<!-- Enable Hibernate's automatic session context management -->

<**property** name=*"hibernate.current\_session\_context\_class"*>thread</**property**>

<!-- Echo all executed SQL to stdout -->

<**property** name=*"show\_sql"*>true</**property**>

<!-- Drop and re-create the database schema on startup -->

<**property** name=*"hibernate.hbm2ddl.auto"*>update</**property**>

<!-- Annotated class -->

<**mapping** class=*"com.example.Employee"*/>

</**session-factory**>

</**hibernate-configuration**>

1. Employee.java – Entity Class

package com.example;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "employee")

public class Employee {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private int id;

private String name;

private String email;

// Constructors

public Employee() {}

public Employee(String name, String email) {

this.name = name;

this.email = email;

}

// Getters and Setters

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

}

4. App.java – Main Application

package com.example;

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

public class App {

public static void main(String[] args) {

// Create SessionFactory from hibernate.cfg.xml

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

// Create and save employee

Employee emp = new Employee("Akash", "akash@example.com");

session.~~save~~(emp);

session.~~save~~(new Employee("Agilan", "agilan@example.com"));

session.~~save~~(new Employee("Abilash", "abi@example.com"));

tx.commit();

System.***out***.println("Employee saved with ID: " + emp.getId());

} catch (HibernateException e) {

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

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}

**Output:**

