**Spring Data JPA with Spring Boot, Hibernate**

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

**application.properties**

spring.application.name=orm-learn

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

spring.datasource.username=root

spring.datasource.password=root

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

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

# 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

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

**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>3.5.3</version>

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

</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>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<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>

**Schema creation**

CREATE DATABASE ormlearn;

use ormlearn;

**Country table Creation**

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

**Persistence Class**

**Country.java (com.cognizant.orm-learn.model.Country)**

package com.cognizant.orm\_learn.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 = "code")

private String code;

*@Column*(name = "name")

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 (com.cognizant.orm-learn.CountryRepository)**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String>{

}

**CountryService.java (com.cognizant.orm-learn.service.CountryService)**

package com.cognizant.orm\_learn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

import jakarta.transaction.Transactional;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import java.util.List;

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 com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

*@SpringBootApplication*

public class OrmLearnApplication {

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

private static CountryService *countryService*;

public static void main(String[] args) {

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

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

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

*testGetAllCountries*();

}

private static void testGetAllCountries() {

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

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

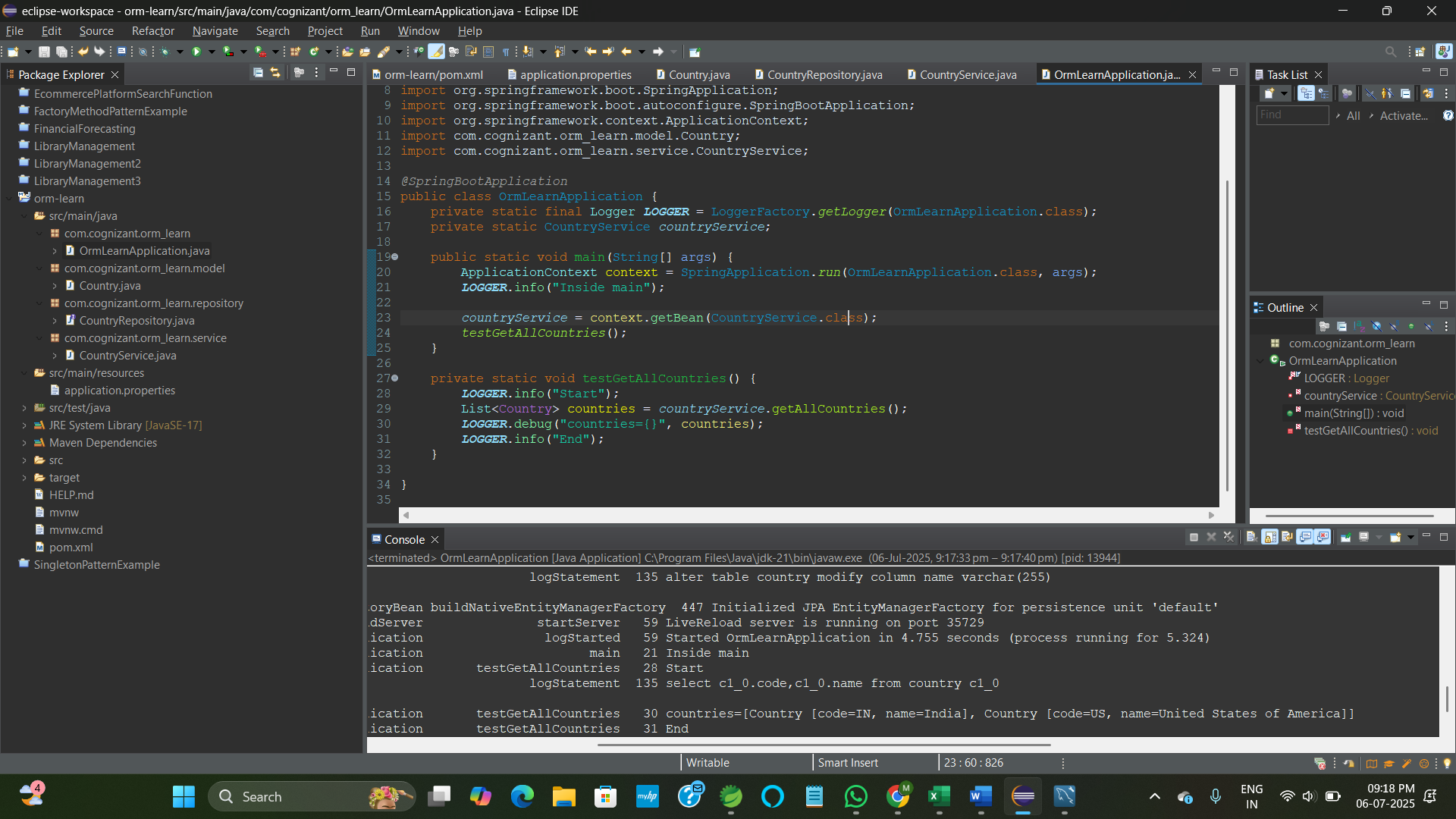
***LOGGER***.debug("countries={}", countries);

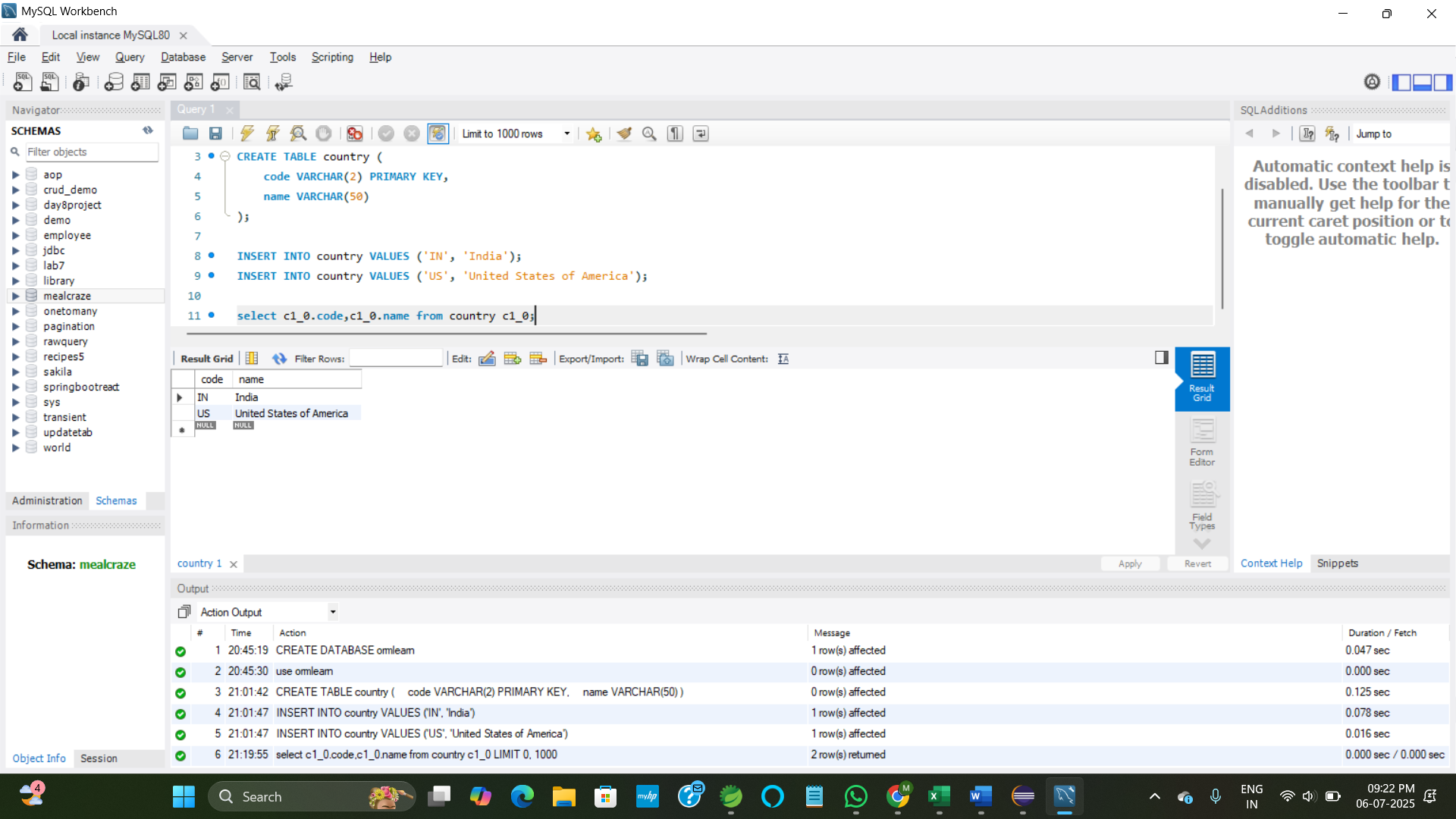
***LOGGER***.info("End");

}

}

**OUTPUT:**





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

**1. Object-Relational Mapping (ORM) using XML**

* In Hibernate, XML mapping helps convert Java objects into database tables.

**Employee.hbm.xml**

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="Employee" table="EMPLOYEE">

<id name="id" type="int" column="ID">

<generator class="native"/>

</id>

<property name="firstName" column="FIRST\_NAME" type="string"/>

<property name="lastName" column="LAST\_NAME" type="string"/>

<property name="salary" column="SALARY" type="float"/>

</class>

</hibernate-mapping>

| **XML Element** | **Purpose** |
| --- | --- |

|  |  |
| --- | --- |
| <class> | Maps a Java class to a DB table |

|  |  |
| --- | --- |
| <id> | Maps primary key |

|  |  |
| --- | --- |
| <property> | Maps class fields to table columns |

|  |  |
| --- | --- |
| name="" | Java class field name |

|  |  |
| --- | --- |
| column="" | DB column name |

|  |  |
| --- | --- |
| type="" | Data type used for conversion (int, string, float, etc.) |

**2. Hibernate Core Components**

**SessionFactory**

* Heavyweight object created once during application startup
* Provides Session objects
* Configured using hibernate.cfg.xml

**Session**

* Lightweight, short-lived object to interact with the DB
* Used to perform CRUD operations
* Obtained from SessionFactory

**Transaction**

* Interface used to manage atomic operations
* Ensures consistency via commit/rollback

**3. End-to-End Operations in Hibernate**

**hibernate.cfg.xml**

<hibernate-configuration>

<session-factory>

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

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

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

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

<mapping resource="Employee.hbm.xml"/>

</session-factory>

</hibernate-configuration>

| **Operation** | **Explanation** |
| --- | --- |
| sessionFactory | Reads config (hibernate.cfg.xml), builds SessionFactory |
| session | Created from SessionFactory; performs DB operations |
| beginTransaction() | Starts a DB transaction |
| commit() | Saves changes permanently to DB |
| rollback() | Cancels transaction in case of exception |
| session.save(object) | Persists a new object (INSERT) |
| session.createQuery().list() | Executes HQL query and returns result list |
| session.get(Class, id) | Fetches one object by primary key (SELECT \* WHERE ID = ?) |
| session.delete(object) | Deletes a record from DB (DELETE) |

**Employee.java**

public class Employee {

private int id;

private String firstName;

private String lastName;

private float salary;

// Getters, Setters, toString()

}

**MainApp.java**

public class MainApp {

public static void main(String[] args) {

SessionFactory factory = new Configuration() .configure("hibernate.cfg.xml") .buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

// Save

Employee emp = new Employee("John", "Doe", 50000);

session.save(emp);

// Read

List<Employee> employees = session.createQuery("FROM Employee").list();

for (Employee e : employees) {

System.out.println(e);

}

// Get by ID

Employee emp1 = session.get(Employee.class, 1);

// Delete

session.delete(emp1);

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}