**Spring Data JPA**

**=> Hands on 1 - Quick Example**   
 **=> Project Name – orm-learn**

**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);

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

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

LOGGER.info("Completed Successfully");

}

}

**Country.java**

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 = "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.java**

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**

package com.cognizant.orm\_learn.service;

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

import org.springframework.stereotype.Service;

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

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

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**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{yyyy-MM-dd HH:mm:ss} %-5level [%t] %c{1}#%M:%L - %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=Dharshini@12

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

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

spring.jpa.show-sql=true

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

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

spring.datasource.username=root

spring.datasource.password=Dharshini@12

**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>

<dependency>

<groupId>javax.persistence</groupId>

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

<version>2.2</version>

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

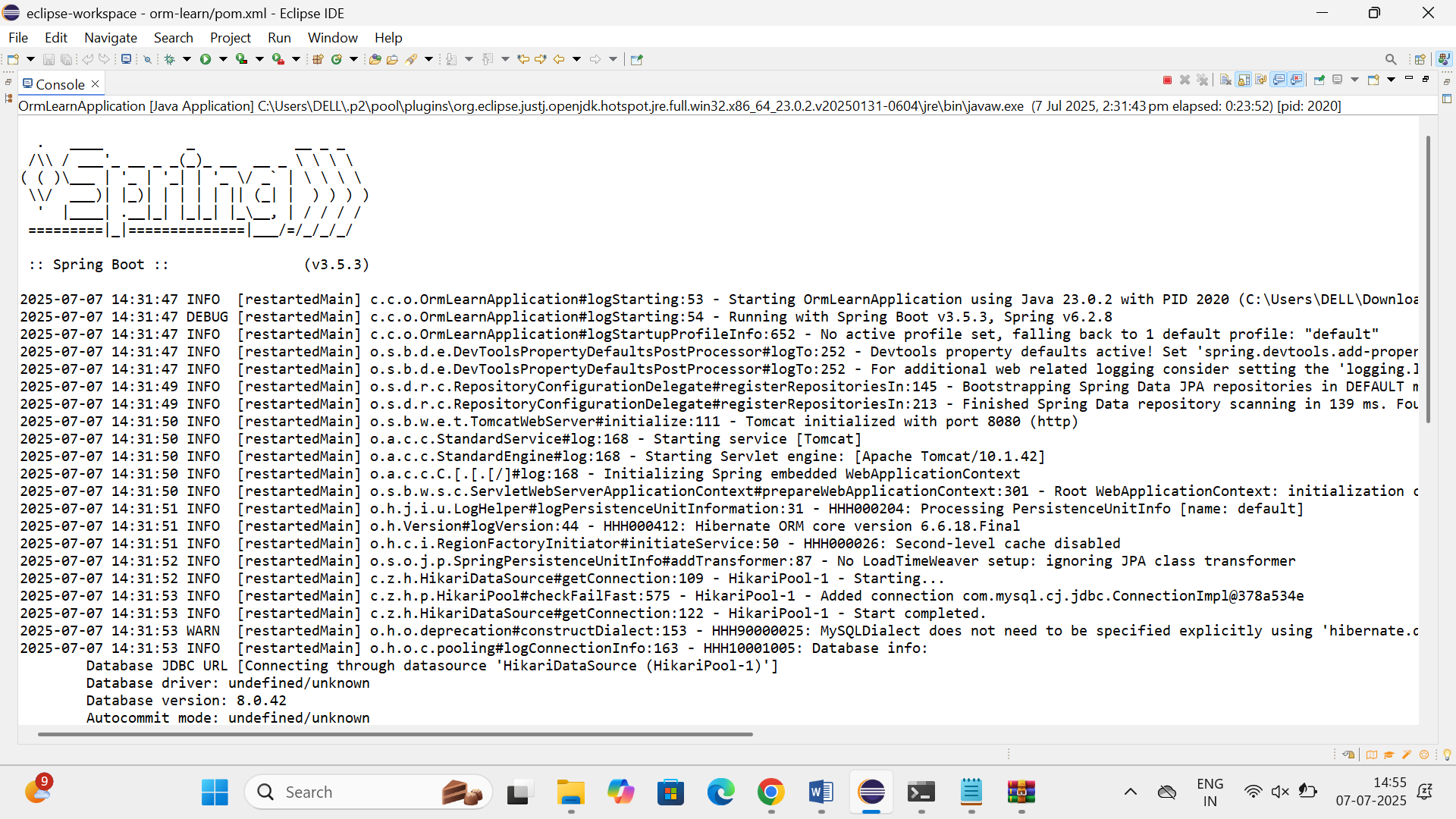
</plugin>

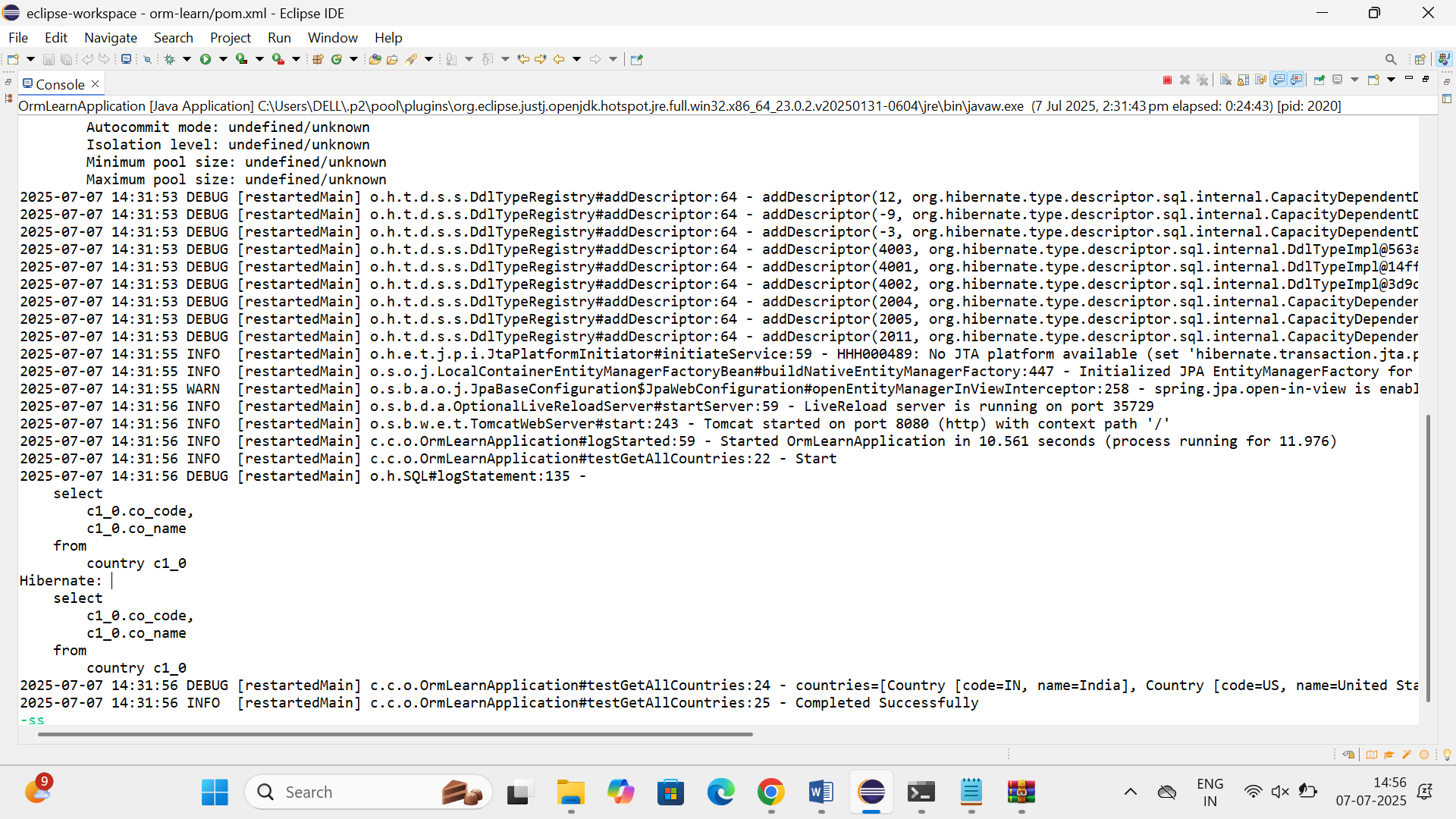
</plugins>

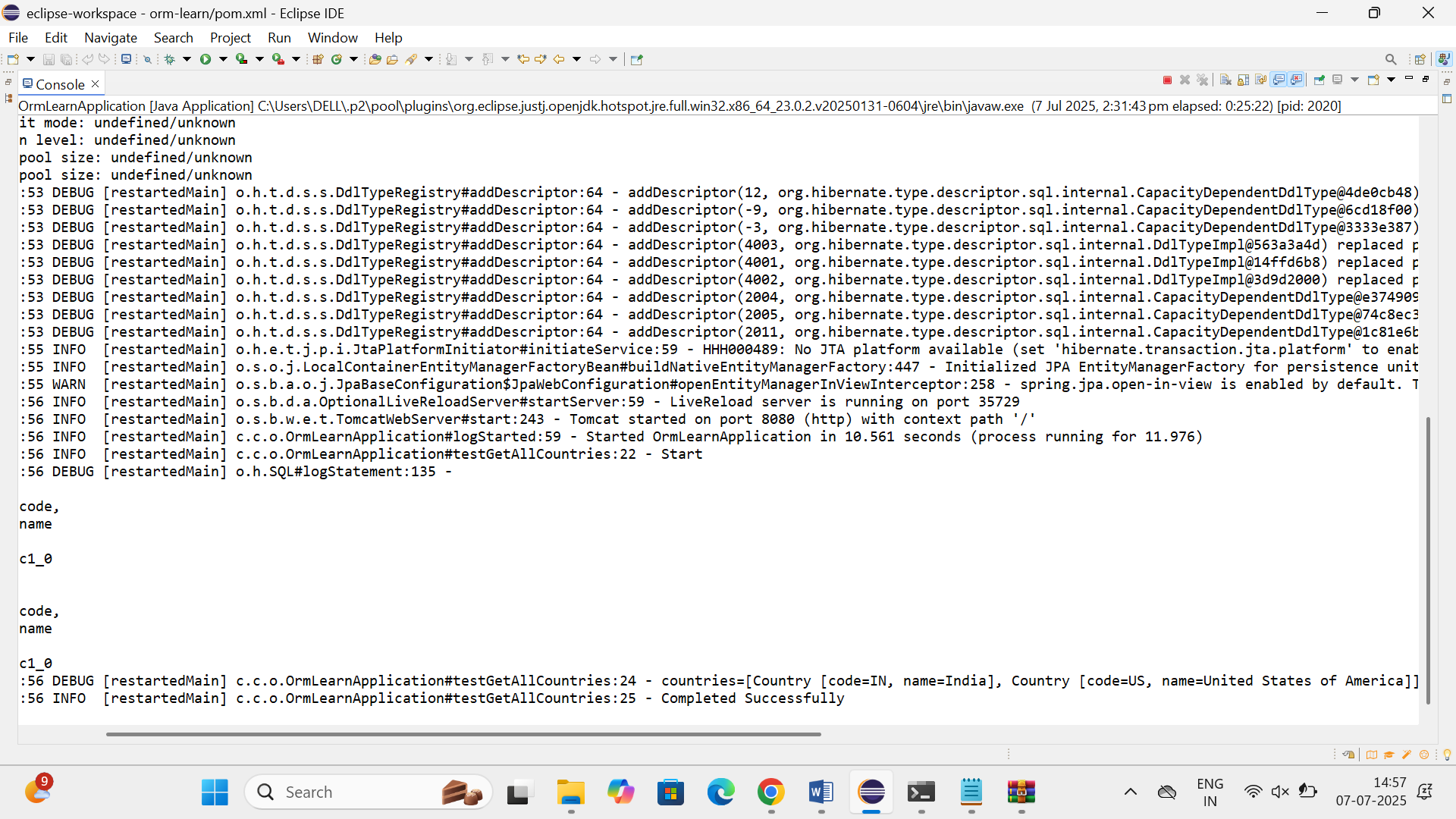
</build>

</project>

**OUTPUT**







* **Difference between JPA, Hibernate and Spring Data JPA** – **Hands 4**

**Java Persistence API (JPA)**

**What is JPA?**

JPA (Java Persistence API) is a standard specification in Java that defines how Java objects (called entities) are mapped to database tables. It provides a set of interfaces and annotations, but does not provide any implementation. You need a JPA provider like Hibernate to make it work.

**JPA is used to:**

* Define entities using annotations
* Perform CRUD operations using EntityManager
* Map object fields to database columns
* Write database queries using JPQL (Java Persistence Query Language)

**Key Points**

* JPA is a Java specification (JSR 338) for accessing, persisting, and managing data between Java objects and relational databases.
* It defines a set of interfaces and annotations, but does not provide any implementation.
* It uses an EntityManager to interact with the database.
* Hibernate is the most popular JPA implementation.

**Example**

**Employee.java**  
import javax.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String department;

public Employee() {}

public Employee(String name, String department) {

this.name = name;

this.department = department;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

}

**EmployeeDAO**

import javax.persistence.\*;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

@Repository

public class EmployeeDAO {

@PersistenceContext

private EntityManager entityManager;

@Transactional

public void addEmployee(Employee employee) {

entityManager.persist(employee);

}

public Employee findEmployee(Integer id) {

return entityManager.find(Employee.class, id);

}

}

**Hibernate**

**What is Hibernate?**

Hibernate is a popular ORM (Object-Relational Mapping) framework that provides a complete implementation of the JPA specification. Hibernate can also work without JPA using its own proprietary APIs.

Hibernate allows:

* Mapping Java objects to relational tables
* Managing sessions and transactions
* Writing queries in HQL (Hibernate Query Language)
* Automatic table creation, caching, batch processing, lazy loading.

**Hibernate Example (Without JPA)**

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

public class HibernateExample {

private static SessionFactory factory;

static {

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

}

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;

}

public Employee getEmployee(Integer id) {

Session session = factory.openSession();

Employee emp = session.get(Employee.class, id);

session.close();

return emp;

}

}

**Spring Data JPA**

**What is Spring Data JPA?**

Spring Data JPA is a part of the Spring Framework that sits on top of JPA and provides a higher-level abstraction for data access. It uses JPA underneath (with Hibernate by default), but hides most of the boilerplate code like writing DAOs and managing transactions.

**Spring Data JPA allows:**

* Creating repositories with interfaces
* Using simple method names to define queries
* Built-in support for pagination and sorting
* Integration with Spring Boot

**Spring Data JPA Example**

**Employee.java**

import javax.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String department;

public Employee() {

}

public Employee(String name, String department) {

this.name = name;

this.department = department;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

}

**EmployeeRepository.java**

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}