**WEEK 3 – SPRING DATA JPA WITH SPRING BOOT, HIBERNATE**

**EXERCISE 1 – SPRING DATA JPA – QUICK EXAMPLE**

**STEP 1:** Create a Maven project in Spring Initializer.

**STEP 2:** Update pom.xml.

**STEP 3:** Create a schema and table in MySQL. Insert the data to it.

create schema ormlearn;

use ormlearn;

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');

**STEP 4:** 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 = "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 + "]";

}

}

OrmLearnApplication.java:

package com.cognizant.orm\_learn;

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

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

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 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");

}

}

**CountryRepository.java:**

package com.cognizant.orm\_learn.repository;

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

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

import org.springframework.stereotype.Repository;

*@Repository*

public interface CountryRepository extends JpaRepository<Country, String> {}

**CountryService.java:**

package com.cognizant.orm\_learn.service;

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

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

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

import org.springframework.stereotype.Service;

import java.util.List;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@*jakarta.transaction.*Transactional*

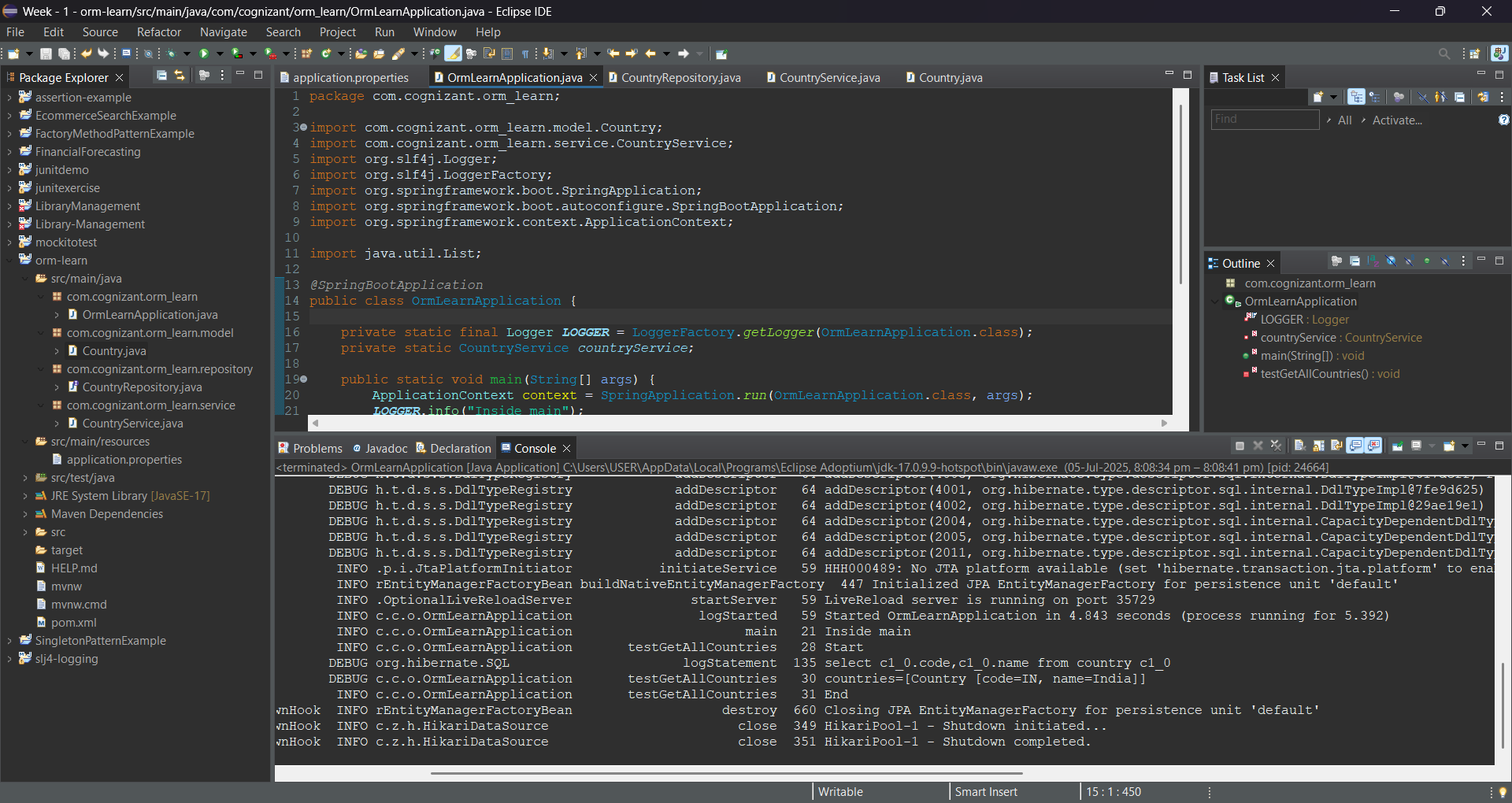
public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OUTPUT:**



**EXERCISE 4: DIFFERENCE BETWEEN JPA, HIBERNATE AND SPRING DATA JPA**

**1. What is JPA?**

* **JPA** is just a **specification** (a set of interfaces and annotations).
* It does **not contain actual implementation**.
* You can think of it as a **"contract"** that defines how ORM (Object Relational Mapping) should work in Java.

@Entity

public class Employee {

@Id

private int id;

private String name;

}

### 2. ****What is Hibernate?****

* Hibernate is a **concrete implementation** of the JPA specification.
* It takes care of the **underlying database interaction**, translating Java objects to SQL and vice versa.
* It uses **Session**, **Transaction**, and **Query** objects.

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

session.save(employee);

tx.commit();

### 3. ****What is Spring Data JPA?****

* Spring Data JPA is a **wrapper/abstraction** over JPA implementations like Hibernate.
* It removes the **boilerplate code** for queries, transactions, sessions.
* Just define a **Repository interface**, and Spring Boot does the rest.

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

**service.java:**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee emp) {

employeeRepository.save(emp); // No need for session, transaction

}