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

**CODE:-**

**Software Pre-requisites:**

* MySQL Server 8.0 or higher
* MySQL Workbench 8 or CLI
* Eclipse IDE for Enterprise Java Developers 2019-03 R or higher
* Maven 3.6.2 or higher
* Java 17 or Java 11

**1. Project Setup in Eclipse**

1. Navigate to https://start.spring.io/
2. Set:
   * Group: com.cognizant
   * Artifact: orm-learn
   * Description: Demo project for Spring Data JPA and Hibernate
3. Add Dependencies:
   * Spring Boot DevTools
   * Spring Data JPA
   * MySQL Driver
4. Generate and download the project zip.
5. Extract the zip into your Eclipse workspace.
6. In Eclipse, go to File > Import > Maven > Existing Maven Projects and select the extracted folder.

**2. Database Setup**

Using MySQL CLI or Workbench:

create schema ormlearn;

use ormlearn;

create table country(co\_code varchar(2) primary key, co\_name varchar(50));

insert into country values ('IN', 'India');

insert into country values ('US', 'United States of America');

## 3. application.properties Configuration

Path: src/main/resources/application.properties

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=root

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

## 4. Build the Project

Run the following in the terminal or Eclipse:

mvn clean package

## 5. Java File Structure and Content

### 5.1 OrmLearnApplication.java

Path: src/main/java/com/cognizant/ormlearn/OrmLearnApplication.java

package com.cognizant.ormlearn;

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.ormlearn.service.CountryService;

import java.util.List;

import com.cognizant.ormlearn.model.Country;

@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");

}

}

### 5.2 Country.java

Path: src/main/java/com/cognizant/ormlearn/model/Country.java

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.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 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 + "]";

}

}

### 5.3 CountryRepository.java

Path: src/main/java/com/cognizant/ormlearn/repository/CountryRepository.java

package com.cognizant.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

### 5.4 CountryService.java

Path: src/main/java/com/cognizant/ormlearn/service/CountryService.java

package com.cognizant.ormlearn.service;

import java.util.List;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

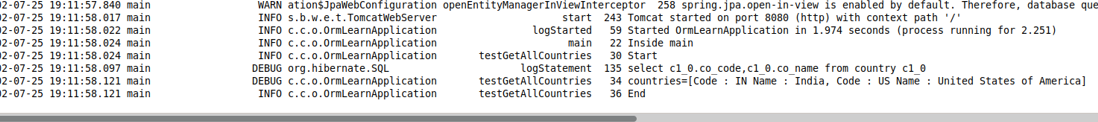
}

}

## 6. Execution

1. Right-click on OrmLearnApplication.java > Run As > Java Application.
2. Check the console for logs:
   * "Inside main"
   * "Start"
   * List of countries fetched from the database
   * "End"

**OUTPUT:-**



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

### Java Persistence API (JPA)

* JSR 338 Specification for persisting, reading, and managing data from Java objects.
* Does not provide a concrete implementation.
* Hibernate is one of the implementations of JPA.

### Hibernate

* An Object Relational Mapping (ORM) tool that implements JPA.

### Spring Data JPA

* Provides another abstraction layer over JPA implementation providers like Hibernate.
* Does not implement JPA but reduces boilerplate code.
* Manages transactions internally for repository operations.

### **Code Comparison**

#### Using Hibernate:

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;

}

#### Using Spring Data JPA:

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

This comparison demonstrates how Spring Data JPA simplifies data persistence by reducing manual session and transaction management code required in plain Hibernate.