Ex1 : Spring Data JPA – Quick Example

**1. application.properties (src/main/resources)**

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

logging.level.org.hibernate.SQL=trace

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

# Log pattern

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

# Database configuration

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

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

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

**2. Database Commands (MySQL)**

> mysql -u root -p

mysql> create schema ormlearn;

-- Create country table

mysql> create table country(

co\_code varchar(2) primary key,

co\_name varchar(50)

);

-- Insert records

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

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

**3. Country.java – Entity Class**

**Path:** com.cognizant.ormlearn.model.Country

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;

// 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 + "]";

}

}

**4. CountryRepository.java – Repository Interface**

**Path:** com.cognizant.ormlearn.repository.CountryRepository

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. CountryService.java – Service Class**

**Path:** com.cognizant.ormlearn.service.CountryService

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. OrmLearnApplication.java – Main Class**

**Path:** com.cognizant.ormlearn.OrmLearnApplication

package com.cognizant.ormlearn;

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.ormlearn.model.Country;

import com.cognizant.ormlearn.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");

}

}

**✅ Expected Output (in Console Log)**

INFO Inside main

INFO Start

DEBUG countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]

INFO End

## **Ex2 : Difference Between JPA, Hibernate, and Spring Data JPA**

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

* **What it is:** A **standard specification** (JSR 338) for accessing, persisting, and managing data between Java objects and relational databases.
* **Key Points:**
  + JPA is just a **specification**, not an implementation.
  + Requires a **provider** (like Hibernate, EclipseLink) to work.
  + Defines standard annotations and interfaces (@Entity, EntityManager, etc.).

### ****2. Hibernate****

* **What it is:** A **popular ORM tool** and **implementation** of the JPA specification.
* **Key Points:**
  + Provides advanced features beyond JPA (like caching, better lazy loading).
  + Requires manual session and transaction management (unless integrated with frameworks).

### ****3. Spring Data JPA****

* **What it is:** A part of the **Spring ecosystem** that sits on top of JPA.
* **Key Points:**
  + Simplifies JPA-based data access by **removing boilerplate code**.
  + Automatically provides repository implementations (like save(), findById(), etc.).
  + Works with JPA providers (usually Hibernate under the hood).
  + Manages transactions and integrates well with Spring.

### 🟠 ****Using Hibernate Directly****

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

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}