**Spring Data JPA with Spring Boot Hibernate**

**Hands On 1 - Spring Data JPA - Quick Example**

**Code :**

**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>3.2.4</version> <!-- use a stable version compatible with your code -->**

**<relativePath/>**

**</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>**

**<properties>**

**<java.version>17</java.version>**

**</properties>**

**<dependencies>**

**<!-- Spring Data JPA -->**

**<dependency>**

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

**<artifactId>spring-boot-starter-data-jpa</artifactId>**

**</dependency>**

**<!-- MySQL JDBC driver -->**

**<dependency>**

**<groupId>com.mysql</groupId>**

**<artifactId>mysql-connector-j</artifactId>**

**<scope>runtime</scope>**

**</dependency>**

**<!-- Spring Boot DevTools -->**

**<dependency>**

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

**<artifactId>spring-boot-devtools</artifactId>**

**<scope>runtime</scope>**

**<optional>true</optional>**

**</dependency>**

**<!-- Logging -->**

**<dependency>**

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

**<artifactId>spring-boot-starter-logging</artifactId>**

**</dependency>**

**<!-- Spring Boot Starter Test -->**

**<dependency>**

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

**<artifactId>spring-boot-starter-test</artifactId>**

**<scope>test</scope>**

**</dependency>**

**<!-- JPA & Transaction API for Spring Boot 3+ (Jakarta) -->**

**<dependency>**

**<groupId>jakarta.persistence</groupId>**

**<artifactId>jakarta.persistence-api</artifactId>**

**<version>3.1.0</version>**

**</dependency>**

**<dependency>**

**<groupId>jakarta.transaction</groupId>**

**<artifactId>jakarta.transaction-api</artifactId>**

**<version>2.0.1</version>**

**</dependency>**

**</dependencies>**

**<build>**

**<plugins>**

**<plugin>**

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

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

**</plugin>**

**</plugins>**

**</build>**

**</project>**

**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");**

**}**

**}**

**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 + "]";**

**}**

**}**

**CountryRepository.java**

**package com.cognizant.orm\_learn.repository;**

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

**import org.springframework.stereotype.Repository;**

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

**@Repository**

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

**}**

**CountryService.java**

**package com.cognizant.orm\_learn.service;**

**import java.util.List;**

**import jakarta.transaction.Transactional;**

**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;**

**@Service**

**public class CountryService {**

**@Autowired**

**private CountryRepository countryRepository;**

**@Transactional**

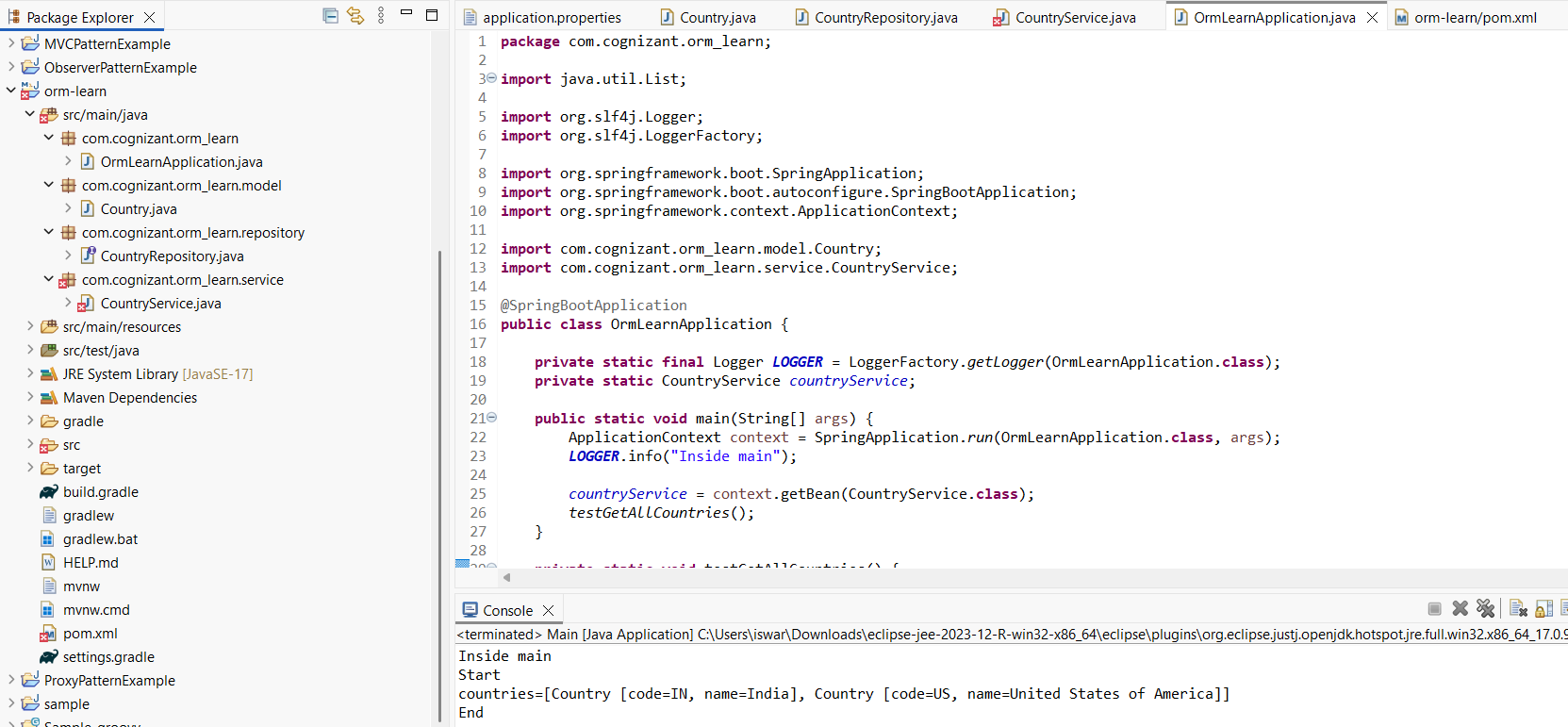
**public List<Country> getAllCountries() {**

**return countryRepository.findAll();**

**}**

**}**

**Output :**

****

**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA Java Persistence API (JPA)**

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

* It is only a specification, not a tool or framework.
* Defines a standard set of annotations and interfaces for ORM (like @Entity, @Id, EntityManager).
* You cannot use JPA alone — you need an implementation like Hibernate, EclipseLink, etc.
* It helps decouple your application from a specific ORM provider.
* Think of it as a contract or blueprint for ORM behavior in Java.

**2. Hibernate**

* **Hibernate is a concrete implementation of JPA (and existed even before** JPA was introduced).
* It provides extra features beyond JPA, like:  
  + Caching (first-level and second-level)
  + Lazy vs. eager fetching strategies
  + Criteria API and native SQL support
* Hibernate allows fine-grained control of database operations, but with more boilerplate code.

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

* A wrapper framework over JPA + Hibernate to reduce developer effort.
* Automatically creates repository implementations from interfaces.
* No need to write SQL or HQL for most queries.
* Provides built-in support for:  
  + CRUD operations
  + Pagination and sorting
  + Query methods based on method names
* It promotes rapid development, especially with Spring Boot.

## **Code Comparison**

### **Hibernate Example (Manual Handling)**

**java**

public void saveEmployee(Employee emp) {

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

session.save(emp);

tx.commit();

session.close();

}

### 

### 

### 

### **Spring Data JPA Example (Minimal Code)**

**java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// No method body needed!

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repo;

public void save(Employee emp) {

repo.save(emp); // Just one line

}

}