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>3.5.3</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>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>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

CountryService.java

package com.cognizant.orm\_learn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

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

import com.cognizant.orm\_learn.CountryRepository;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

CountryRepository.java

package com.cognizant.orm\_learn;

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> {

}

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;

// getters and setters

// Getter for code

public String getCode() {

return code;

}

// Setter for code

public void setCode(String code) {

this.code = code;

}

// Getter for name

public String getName() {

return name;

}

// Setter for name

public void setName(String name) {

this.name = name;

}

// toString()

*@Override*

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

OrmLearnApplication.java

package com.cognizant.orm\_learn;

import org.springframework.context.ApplicationContext;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@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();

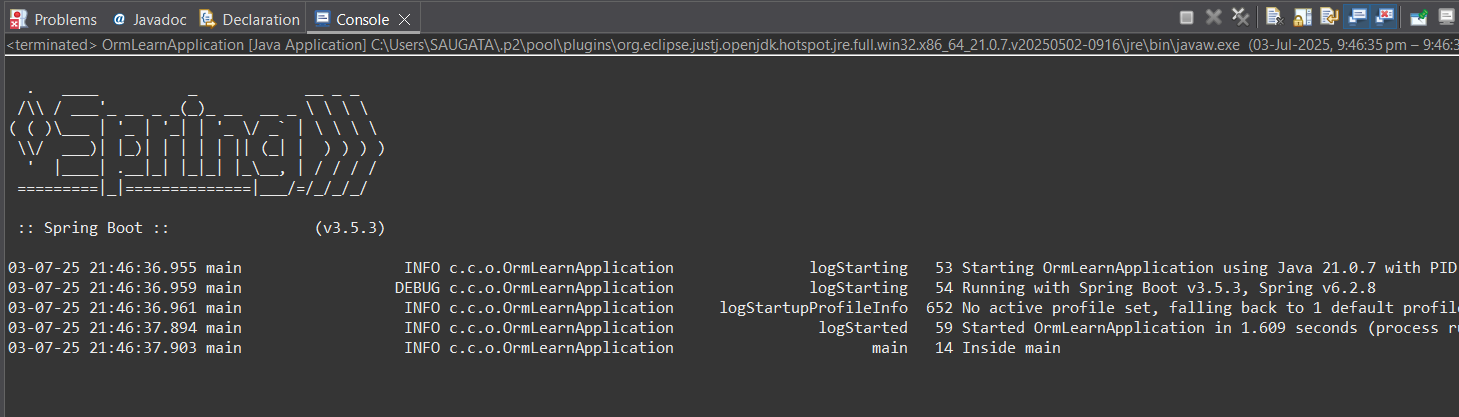
***LOGGER***.debug("countries={}", countries);

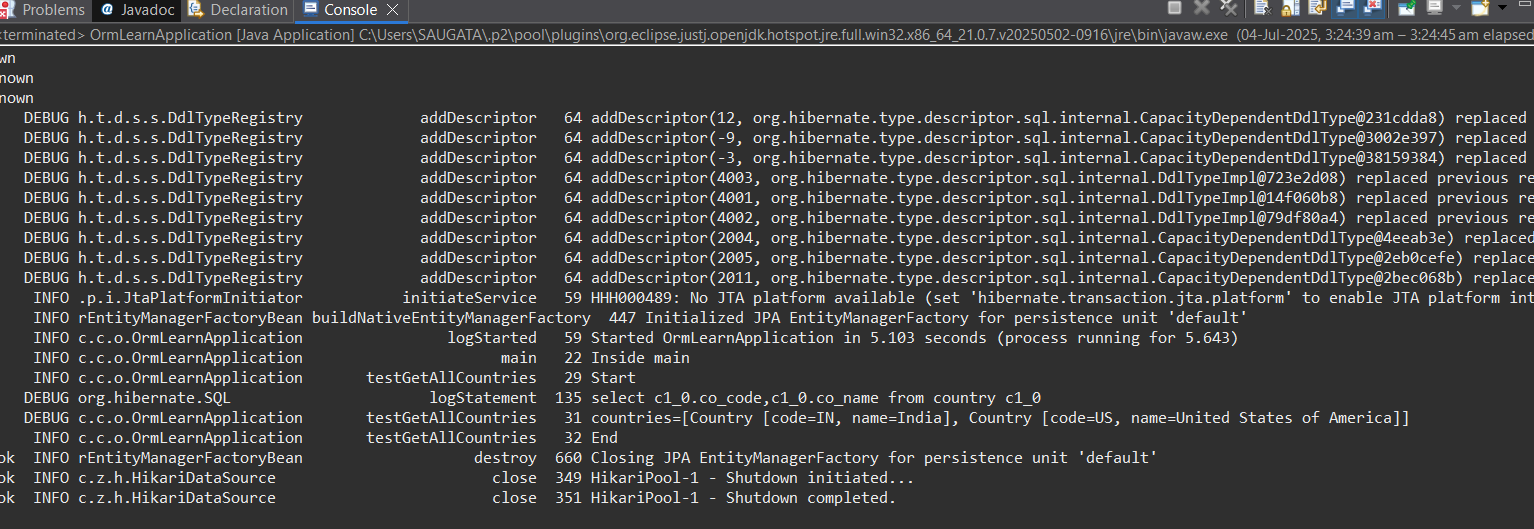
***LOGGER***.info("End");

}

}

**Output:**





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

* **What it is**: A **Java specification (JSR 338)** for managing relational data in Java applications.
* **Key traits**:
  + Provides annotations and interfaces like @Entity, @Id, EntityManager, etc.
  + **Does not provide an implementation**—it’s just a standard API.
  + Requires an implementation like Hibernate, EclipseLink, etc., to function.

**🔹 Hibernate**

* **What it is**: A **concrete implementation of JPA** (though it existed before JPA and has extra features beyond the JPA spec).
* **Key traits**:
  + It's an **ORM (Object-Relational Mapping)** tool.
  + Handles database operations via session management.
  + You manually handle transactions, sessions, and exceptions.

**Hibernate Example:**

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;

}

**🔹 Spring Data JPA**

* **What it is**: A **Spring project** that builds on top of JPA and provides another layer of abstraction.
* **Key traits**:
  + Works with **any JPA implementation** (typically Hibernate).
  + Uses **Spring’s dependency injection and transaction management**.
  + **Minimizes boilerplate code**—no need to write DAOs manually.
  + Provides built-in repository interfaces like JpaRepository, CrudRepository.

**Spring Data JPA Example:**

**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);

}

}

**🔄 Summary Comparison Table**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (API) | Implementation (ORM) | Abstraction over JPA (Spring Module) |
| Provides persistence | ✅ (via spec) | ✅ (implements JPA & more) | ✅ (uses JPA impl like Hibernate) |
| Implementation | ❌ | ✅ | ❌ |
| Boilerplate Code | Medium | High | Very Low |
| Transaction Mgmt | Needs manual config | Manual or programmatic | Handled by Spring (@Transactional) |
| Integration | Requires setup | Requires session mgmt | Easy with Spring Boot / Spring Context |