**Spring Data JPA - Quick Example**

**Code:**

**Pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<parent>

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

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.0</version> <!-- or use latest stable -->

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

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

</properties>

<dependencies>

<!-- Spring Boot JPA -->

<dependency>

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

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

</dependency>

<!-- MySQL Driver -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.33</version>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot DevTools (Optional) -->

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter -->

<dependency>

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

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

</dependency>

<!-- Optional: Testing support -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Java Compiler Plugin -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.10.1</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

ApplicationProperties.xml

# Logging

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

# MySQL config

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

spring.jpa.hibernate.ddl-auto=validate

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

CountryRepository

**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> {

}

Country

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

**import** jakarta.persistence.\*;

@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 + "]";

}

}

CountryService

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

**import** jakarta.transaction.Transactional;

**import** java.util.List;

**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;

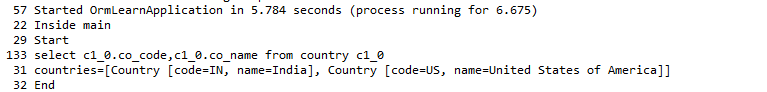
@Transactional

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

**return** countryRepository.findAll();

}

Output:



Difference between JPA, Hibernate and Spring Data JPA

s **JPA (Java Persistence API)**

1. It is a **standard specification** (API) for mapping Java objects to database tables.
2. Part of **Jakarta EE** (previously Java EE).
3. Provides **annotations** like @Entity, @Id, @Table, etc.
4. Does **not contain any implementation** — only defines *how* things should work.
5. Requires a **JPA provider** (like Hibernate) to actually perform the database operations.
6. Can be used with or without Spring.

**Hibernate**

1. It is a **popular JPA implementation** (provider).
2. Follows JPA rules and also adds **extra features** (e.g., HQL, caching, lazy loading).
3. Can be used directly or through JPA.
4. **Handles actual database interactions** like insert, update, delete, select.
5. Used as the **default JPA provider** in Spring Boot.
6. You need to write more code compared to Spring Data JPA (e.g., manage transactions, sessions).

**Spring Data JPA**

1. It is a **Spring framework module** built on top of JPA.
2. Provides **simplified and automated** database access.
3. Reduces boilerplate code — you can define interfaces like JpaRepository and Spring generates queries for you.
4. Allows using method names like findById() or findByName() without writing SQL.
5. Integrates easily with Spring Boot for rapid development.
6. Internally uses **JPA + Hibernate**, but provides a cleaner abstraction.