**Spring Data JPA - Quick Example**

**PART 1: Project Setup via Spring Initializer**

**PART 2: Create Schema in MySQL**

**create schema ormlearn;**

**PART 3: Configure 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

**Add logging in main method**

package com.cognizant.ormlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

}

**PART 6: Create country Table in MySQL**

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');

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 + "]";

}

}

A close up of words

AI-generated content may be incorrect.

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

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

**🔹 What is it?**

* **JPA is a Java specification (interface) for managing relational data in Java applications.**
* **It defines standard interfaces and annotations for ORM (Object-Relational Mapping).**
* **It does not provide implementation; just the guidelines.**

**🔹 Key Points:**

* **It's part of Java EE (now Jakarta EE).**
* **Annotations: @Entity, @Table, @Id, @OneToMany, etc.**
* **Needs a provider like Hibernate to work.**

**Example :**

**import javax.persistence.\*;**

@Entity

@Table(name = "employee")

public class Employee {

@Id

private Long id;

private String name;

}

**2. Hibernate**

**🔹 What is it?**

* Hibernate is the **most popular JPA implementation**.
* It provides the actual functionality to persist Java objects to a database.
* Also includes some features **beyond JPA** (like caching, lazy loading tuning, etc.).

**🔹 Key Points:**

* Implements JPA spec.
* Can be used **with or without JPA**.
* Provides **Session**, **Query**, and other powerful APIs.

**3. Spring Data JPA**

**🔹 What is it?**

* **Abstraction over JPA + Hibernate** provided by Spring.
* Simplifies data access by **automatically generating queries** based on method names.
* Reduces boilerplate code to manage entities and repositories.

**🔹 Key Points:**

* Uses Hibernate as default provider.
* Offers JpaRepository, CrudRepository interfaces.

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (interface) | Implementation (framework) | Framework (built on JPA + Spring) |
| Provides implementation? | ❌ No | ✅ Yes | ✅ Yes (via Hibernate) |
| Boilerplate code | Moderate | Low | Very Low (most auto-generated) |
| Configuration complexity | Medium | Medium | Very simple (Spring Boot friendly) |
| Query support | JPQL | HQL/JPQL/Native SQL | Method naming convention + JPQL |
| Example APIs | EntityManager | SessionFactory, Session | JpaRepository, CrudRepository |