# **Spring Data JPA - Quick Example**

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 org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

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

import org.springframework.transaction.annotation.Transactional;

import javax.persistence.\*;

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

@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");

}

}

@Entity

@Table(name = "country")

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 + "]";

}

}

@Repository

interface CountryRepository extends JpaRepository<Country, String> {

}

@Service

class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

## **OUTPUT:**

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

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

06-07-25 16:35:12.123 main INFO com.cognizant.ormlearn.OrmLearnApplication main 21 Inside main

06-07-25 16:35:12.456 main INFO com.cognizant.ormlearn.OrmLearnApplication testGetAllCountries 26 Start

06-07-25 16:35:12.789 main DEBUG com.cognizant.ormlearn.OrmLearnApplication testGetAllCountries 28 countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]

06-07-25 16:35:12.790 main INFO com.cognizant.ormlearn.OrmLearnApplication testGetAllCountries 29 End

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

1. Java Persistence API (JPA)

* **Definition**: A **Java specification** (JSR 338) that standardizes **Object-Relational Mapping (ORM)** between Java objects and relational databases.
* **Key Points**:  
  + Only provides **interfaces and annotations** (like @Entity, @Id, @OneToMany)
  + Requires a **provider/implementation** (like Hibernate) to actually perform DB operations.
* **Does not do anything by itself** — it's just a **contract**.

2. Hibernate

* **Definition**: A **popular ORM tool** that provides a concrete implementation of the JPA specification.
* **Key Points**:  
  + Handles DB connection, SQL generation, caching, lazy loading, etc.
  + Can be used with or without JPA annotations
  + Handles sessions, transactions manually or semi-automatically

3. Spring Data JPA

* **Definition**: A **Spring module** that adds another layer of abstraction **on top of JPA providers** like Hibernate.
* **Key Points**:  
  + No need to write boilerplate DAO code (like openSession, beginTransaction)
  + Uses **interfaces and annotations** to perform CRUD operations
  + Works with any JPA provider (Hibernate is most common)
  + Handles **transactions**, **queries**, and **repository patterns** automatically  
     Comparison Table

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (API) | Implementation (ORM tool) | Spring module using JPA |
| Needs implementation | Yes | No (Hibernate is the implementation) | Uses JPA + Hibernate or any provider |
| Boilerplate code | Medium | High | Low (almost none) |
| Transaction management | Manual or via container | Manual | Automatic via @Transactional |
| Usage complexity | Moderate | Complex | Very simple |