**Ques 1 : Spring Data JPA - Quick Example**

**Code :**

**--Database Setup**

-- Open MySQL client

mysql -u root -p

-- Create Schema

create schema ormlearn;

-- Create and insert into 'country' table

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');

**--application.properties**# Logging config  
logging.level.org.springframework=info  
logging.level.com.cognizant=debug  
logging.level.org.hibernate.SQL=trace  
logging.level.org.hibernate.type.descriptor.sql=trace  
  
# Log output format  
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  
  
# Database connection  
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn  
spring.datasource.username=root  
spring.datasource.password=Kratika@123  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
  
# Hibernate config  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect  
  
  
# Show SQL  
spring.jpa.show-sql=true

**--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;  
  
 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> {  
  
}

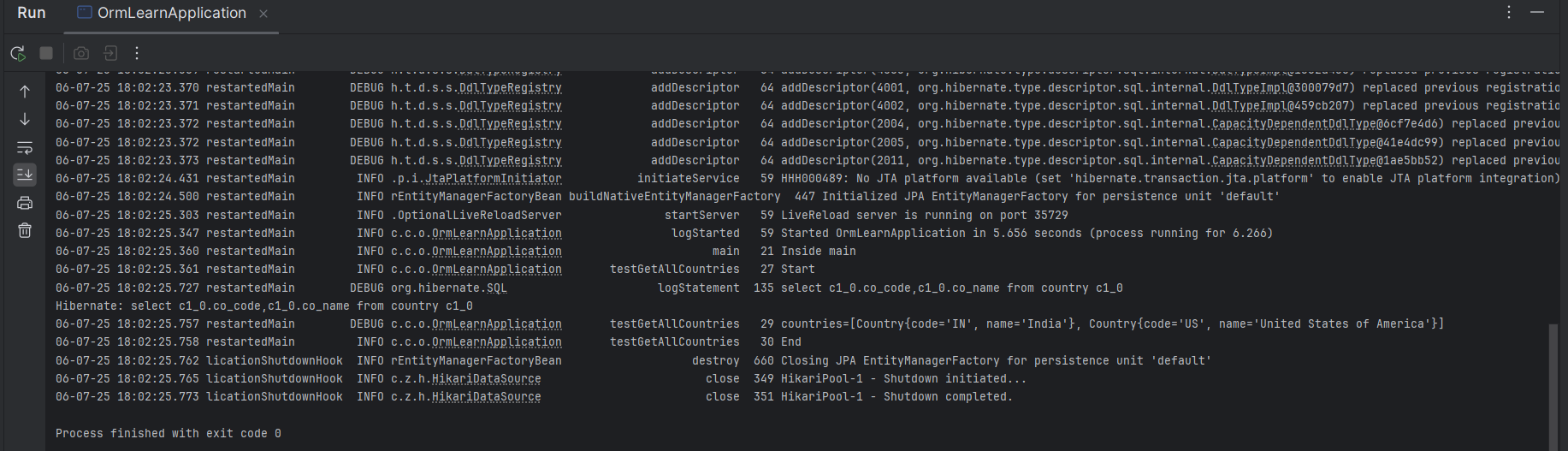
**--CountryService.java**

package com.cognizant.orm\_learn.service;  
  
import java.util.List;  
  
import 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.repository.CountryRepository;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

**--OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
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 java.util.List;  
  
@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 :**

****

**Ques 2 : Difference between JPA, Hibernate and Spring Data JPA**

**1. What is JPA (Java Persistence API)?**

* Definition: JPA is a *specification* provided by Oracle for persisting Java objects into a relational database.
* Key Point: It does not provide implementation.
* Common Implementations:
  + Hibernate
  + EclipseLink
  + OpenJPA

Analogy: JPA is like a *contract*. It says *how things should be done*, not *how to do them*. That’s the job of implementers like Hibernate.

**2. What is Hibernate?**

* Definition: Hibernate is an ORM (Object-Relational Mapping) framework and the most popular implementation of JPA.
* Role: It provides the actual code that handles entity persistence, transaction management, lazy loading, caching, etc.
* It can work with or without JPA.

**3. What is Spring Data JPA?**

* Definition: Spring Data JPA is a wrapper framework provided by Spring that simplifies data access using JPA.
* Key Feature: Reduces boilerplate code through:
  + JpaRepository and CrudRepository interfaces
  + Auto-implemented queries
* Important: It does not implement JPA; it relies on a JPA provider like Hibernate underneath.

**--Comparison Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| **Type** | **Specification (API)** | **Implementation** | **Framework abstraction over JPA** |
| **Boilerplate Code** | **Moderate** | **Less (but still there)** | **Very minimal** |
| **Query Language** | **JPQL** | **HQL (superset of JPQL)** | **Derived Queries, JPQL, Native** |
| **Transactions** | **Not handled** | **Handled** | **Handled (via Spring)** |
| **Provided By** | **Oracle (Java EE)** | **Red Hat** | **Pivotal (Spring)** |

**--Spring Data JPA Example**

**-- EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// No need to define save(), it's inherited

}

**-- EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}