**Spring Data JPA with Spring Boot, Hibernate**

**1. Hands on 1: Spring Data JPA - Quick Example**

**Solution:**

**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.example</groupId>

<artifactId>ormlearn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ormlearn</name>

<description>Demo project for Spring Boot</description>

<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>

<version>8.0.33</version>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</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>

**application.properties**

spring.application.name=ormlearn

# Logging

logging.level.org.springframework=info

logging.level.com.example.ormlearn=debug

logging.level.org.hibernate.SQL=debug

logging.level.org.hibernate.type.descriptor.sql=trace

# Database

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=Shri19

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

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

**Country.java**

package com.example.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.persistence.Column;

@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.example.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.example.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.example.ormlearn.service;

import java.util.List;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmlearnApllication.java**

package com.example.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 com.example.ormlearn.model.Country;

import com.example.ormlearn.service.CountryService;

@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);

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

List<Country> countries = countryService.getAllCountries();

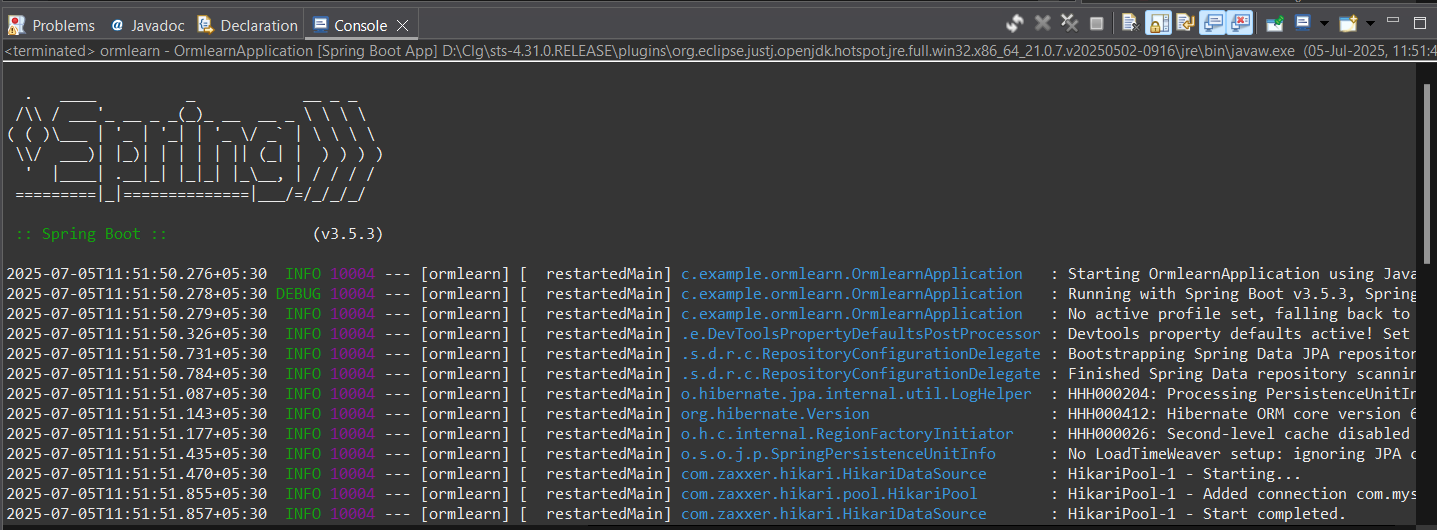
LOGGER.debug("countries={}", countries);

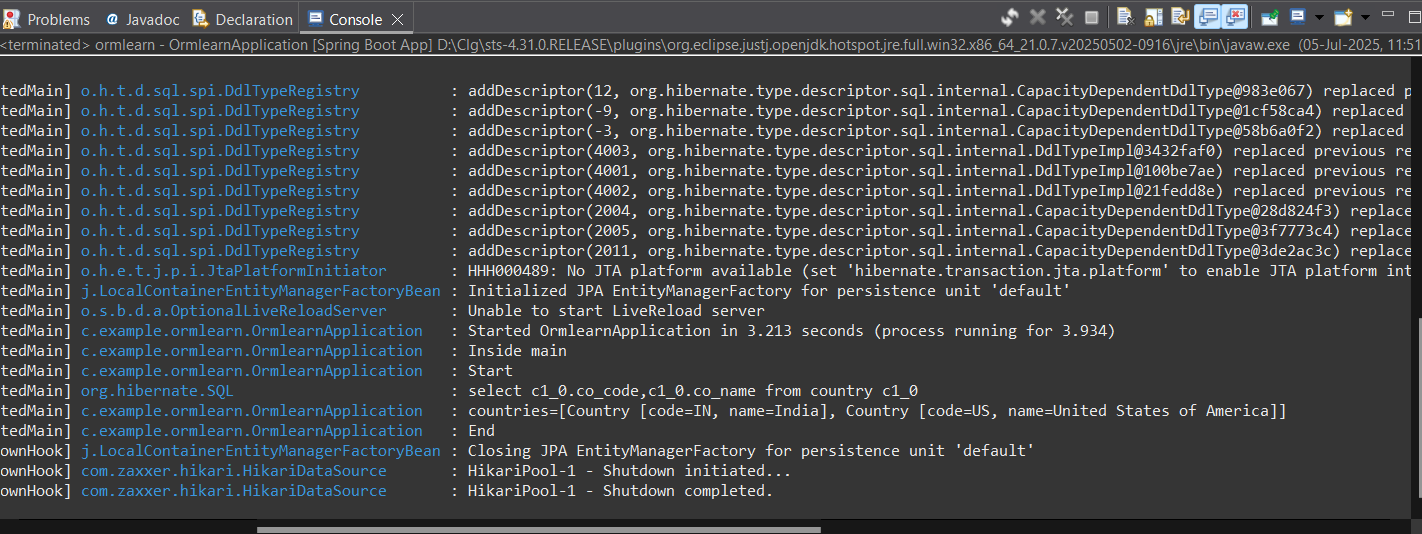
LOGGER.info("End");

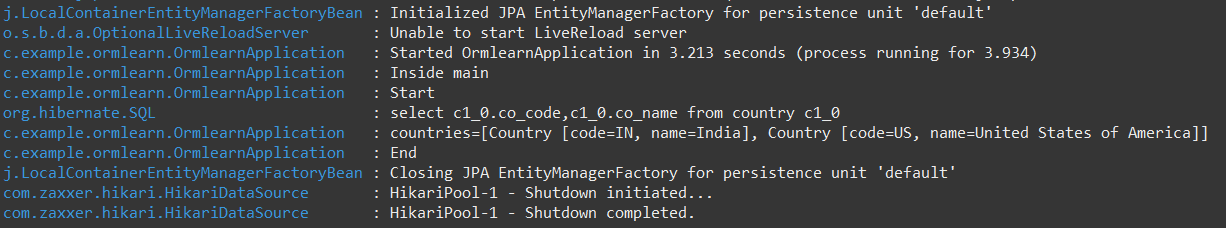
}

}

**Output:**







**Explanation:**

1. Project Setup

Use Spring Initializr to create a Maven project with group com.cognizant and artifact orm-learn.

Select dependencies: Spring Boot DevTools, Spring Data JPA, MySQL Driver.

Import the generated project into Eclipse.

1. MySQL Configuration

Create schema ormlearn in MySQL.

Configure database properties and logging in application.properties.

1. Build the Project

Use mvn clean package with proxy settings if required.

1. Main Application Setup

Add SLF4J logger to OrmLearnApplication.

Run the application and verify logger output from main().

1. Project Structure Overview

* src/main/java: Java application code.
* src/main/resources: Config files (application.properties).
* src/test/java: Unit tests.
* @SpringBootApplication: Enables component scanning, configuration, and auto-configuration.
* pom.xml: Contains all project dependencies and build configuration.

1. Database Table Creation

Create country table with co\_code and co\_name.

Insert sample records: IN - India, US - United States of America.

1. Entity Class: Country

Annotate with @Entity and @Table(name="country").

Use @Id for primary key and @Column for mapping fields.

1. Repository Interface: CountryRepository

Extend JpaRepository<Country, String>.

Annotate with @Repository.

1. Service Class: CountryService

Annotate with @Service.

Autowire CountryRepository.

Add method getAllCountries() with @Transactional.

1. Testing in OrmLearnApplication

Autowire CountryService using ApplicationContext.

Add and call testGetAllCountries() to print country list using logger.

**2. Hands on 4:Difference between JPA, Hibernate and Spring Data JPA**

**Solution:**

**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.example</groupId>

<artifactId>jpa-hibernate-employee</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>jpa-hibernate-employee</name>

<description>Spring Boot app using Spring Data JPA and MySQL</description>

<parent>

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

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

<version>3.2.4</version> <!-- You can update to latest -->

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

</parent>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<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.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</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.apache.maven.plugins</groupId>

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

<version>3.11.0</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**application.properties**

spring.application.name=jpahibernateemployee

# MySQL Configuration

spring.datasource.url=jdbc:mysql://localhost:3306/test\_db

spring.datasource.username=root

spring.datasource.password=Shri19

# JPA Properties

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**Employee.java**

package com.example.employee.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Employee {

@Id

private int id;

private String name;

private String department;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

}

**EmployeeRepository.java**

package com.example.employee.repository;

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

import com.example.employee.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

package com.example.employee.service;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.example.employee.model.Employee;

import com.example.employee.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**JpahibernateemployeeApllication.java**

package com.example.employee;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.example.employee.model.Employee;

import com.example.employee.service.EmployeeService;

@SpringBootApplication

public class JpahibernateemployeeApplication implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(JpahibernateemployeeApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

Employee emp = new Employee();

emp.setId(101);

emp.setName("John");

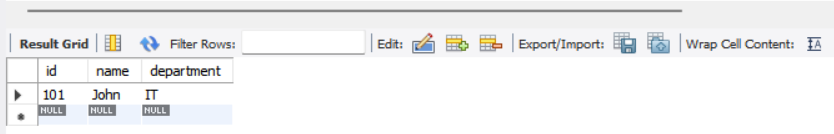
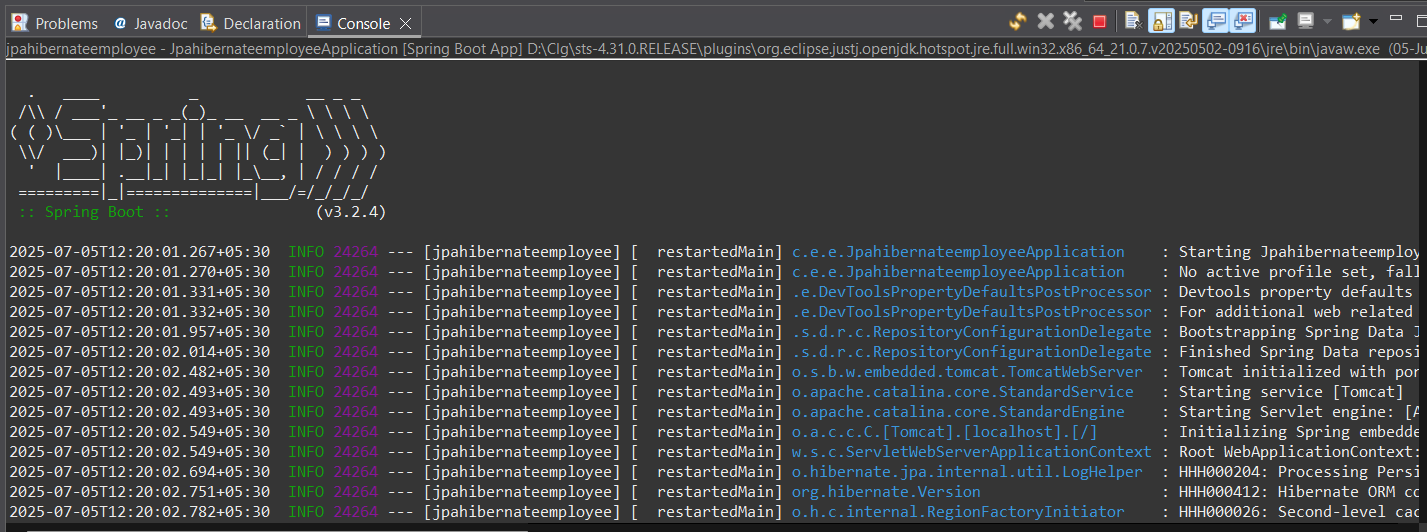
emp.setDepartment("IT");

employeeService.addEmployee(emp);

}

}

**Output:**



**Explanation:**

1. Understand that **JPA** is a specification (no implementation).
2. Recognize that **Hibernate** is an ORM tool that implements JPA.
3. Know that **Spring Data JPA** is an abstraction layer over JPA providers like Hibernate.
4. Compare coding styles:

* Hibernate requires manual session and transaction handling.
* Spring Data JPA uses JpaRepository for CRUD with minimal code.

1. Use annotations like @Autowired and @Transactional in Spring Data JPA for dependency injection and transaction management.
2. Leverage Spring Boot's auto-configuration to manage repositories without XML or manual configuration.