**1.Spring Data JPA - Quick Example**

1.Go to <https://start.spring.io/>

2.Fill in:

* Group: com.cognizant
* Artifact: orm-learn
* Description: Demo project for Spring Data JPA and Hibernate

3.Add dependencies:

* Spring Boot DevTools
* Spring Data JPA
* MySQL Driver

4.Click Generate, then extract the downloaded ZIP to your Eclipse workspace folder.

5.In Eclipse:  
File > Import > Maven > Existing Maven Projects,Select the extracted orm-learn folder.

Click Finish.

6.Open MySQL Workbench or terminal.

7.Create schema:

CREATE DATABASE ormlearn;

8.Create table and insert data:

use ormlearn;

CREATE TABLE country(

co\_code VARCHAR(2) PRIMARY KEY,

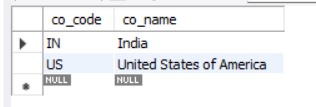
co\_name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

select \* from country;



**src/main/resources/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.MySQLDialect

**Country.java:**

Create a package **com.cognizant.orm\_learn.model** and create a class **Country**

package com.cognizant.orm\_learn.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:**

Create **com.cognizant.orm\_learn.repository** and create **CountryRepository** class

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:**

Create **com.cognizant.ormlearn.service** package and create **CountryService** class

package com.cognizant.orm\_learn.service;

import java.util.List;

import jakarta.transaction.Transactional;

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();

}

}

**OrmLearnApplication.java:**

package com.cognizant.orm\_learn;

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.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.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);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

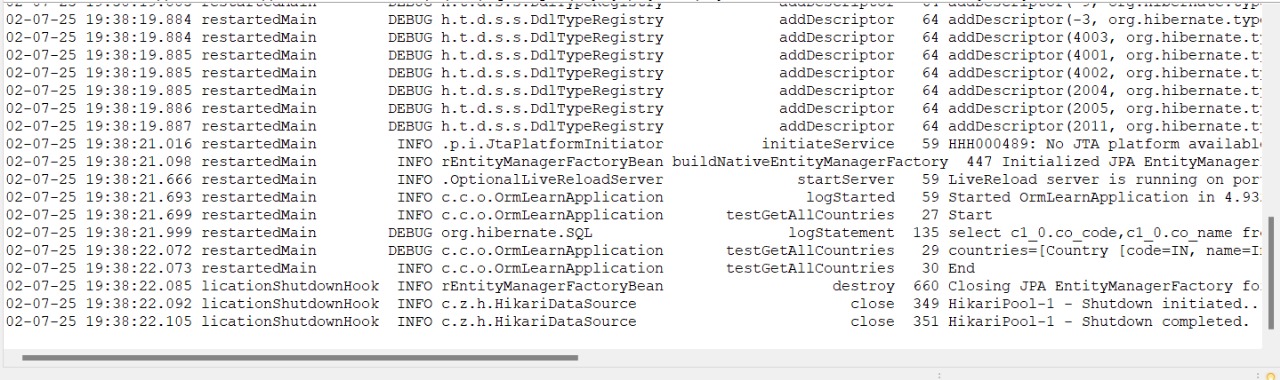
LOGGER.debug("countries={}", countries);

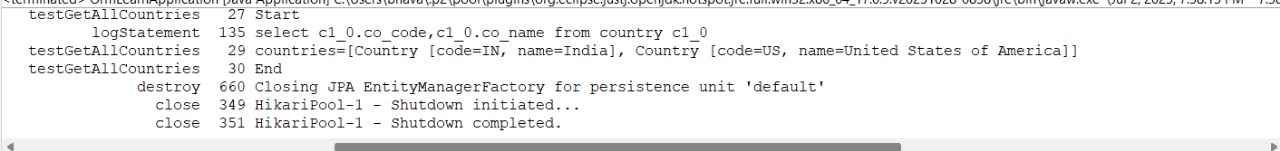
LOGGER.info("End");

}

}

**Output:**

****



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

**Java Persistence API (JPA)**

* JPA is a specification defined under JSR 338 for persisting, reading, and managing data between Java objects and relational databases.
* It only provides guidelines and standards, but does not offer any concrete implementation.
* Popular frameworks like Hibernate act as implementations of the JPA specification.

**Hibernate**

* Hibernate is an Object Relational Mapping (ORM) tool that provides a concrete implementation of JPA.
* It allows developers to map Java objects to database tables and handle database operations easily using JPA standards.

**Spring Data JPA**

* Spring Data JPA is not an implementation of JPA, but rather a framework that simplifies the development of JPA-based data access layers.
* It provides an abstraction layer on top of JPA implementations like Hibernate, reducing boilerplate code for repository creation.
* It also helps in managing transactions, query generation, and simplifies CRUD operations with minimal code.

**MySql Setup:**

CREATE DATABASE company\_db;

USE company\_db;

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(100),

department VARCHAR(100),

salary DOUBLE

);

**Spring Data JPA Approach:**

1.Go to <https://start.spring.io/>

2.Fill in:

* Group: com.example1
* Artifact: demo
* Description: Demo project for Spring Data JPA and Hibernate

3.Add dependencies:

* Spring Web
* Spring Data JPA
* MySQL Driver

4.Click Generate, then extract the downloaded ZIP to your Eclipse workspace folder.

5.In Eclipse:  
File > Import > Maven > Existing Maven Projects,Select the extracted orm-learn folder.

Click Finish.

**application.properties:**

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

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**Employee.java:**

package com.example1.demo.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String department;

private double salary;

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; }

public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

}

**EmployeeRepository.java:**

package com.example1.demo.repository;

import com.example1.demo.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java:**

package com.example1.demo.service;

import com.example1.demo.model.Employee;

import com.example1.demo.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee emp) {

employeeRepository.save(emp);

}

}

**DemoApplication.java:**

package com.example1.demo;

import com.example1.demo.model.Employee;

import com.example1.demo.service.EmployeeService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

@Override

public void run(String... args) {

Employee emp = new Employee();

emp.setName("Ram");

emp.setDepartment("IT");

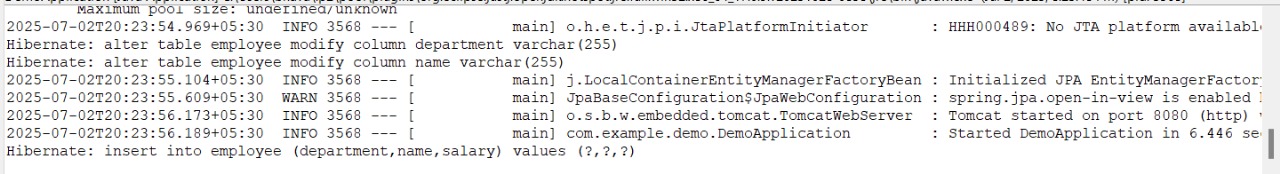
emp.setSalary(50000);

employeeService.addEmployee(emp);

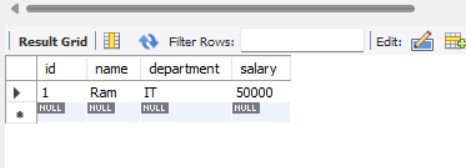
}

}

**Output:**



**MYSQL:**

****

**Hibernate Approach:**

Create a new Maven project:

* File → New Project→ Maven Project → Choose quickstart archetype.

**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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>HibernateDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

<version>7.0.0.Final</version>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.3.0</version>

</dependency>

<dependency>

<groupId>jakarta.platform</groupId>

<artifactId>jakarta.jakartaee-api</artifactId>

<version>10.0.0</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<resources>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

</build>

</project>

**hibernate.cfg.xml:** (in src/main/resources)

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/employee\_db</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<property name="show\_sql">true</property>

<mapping class="com.example.Employee"/>

</session-factory>

</hibernate-configuration>

**Employee.java:**

package com.example1;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String department;

private double salary;

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; }

public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

}

**HibernateExample.java:**

package com.example1;

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

public class HibernateExample {

public static void main(String[] args) {

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee emp = new Employee();

emp.setName("Raju");

emp.setDepartment("HR");

emp.setSalary(45000);

session.persist(emp);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

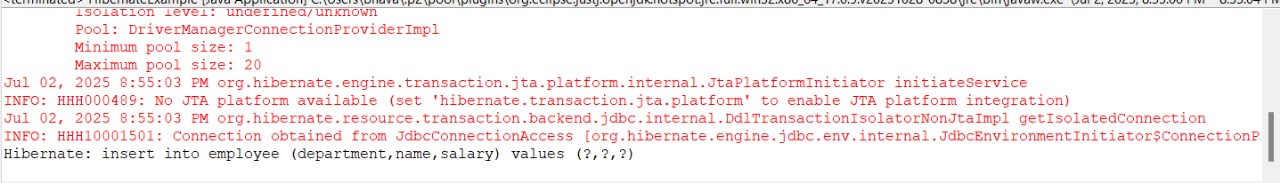
session.close();

}

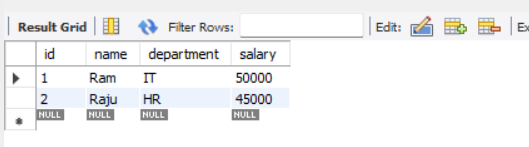
factory.close();

}

}

**Output:**

**MYSQL:**

****