**Spring Data JPA with Spring Boot – Quick Example**

1.Install MySQL server and workbench

2. Go to <https://start.spring.io/>

3.Do the following

Group: com.cognizant

ArtifactId: orm-learn

Description: Demo project for Spring Data JPA and Hibernate

4.Add the following dependencies

Spring Boot DevTools

Spring Data JPA

MySQL Driver

5. Click Generate, then download the zip and Extract the ZIP into your Eclipse Workspace folder.

6. In Eclipse:

**File > Import > Maven > Existing Maven Projects,** Select the extracted folder, then click Finish.

7.In MYSQL:

Create a schema and in that create a table and insert values

create database ormlearn;

use ormlearn;

create table person(

id int primary key,

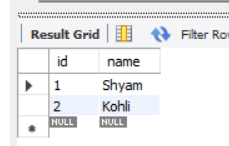
name varchar(20)

);

INSERT INTO person VALUES (1, 'Shyam');

INSERT INTO person VALUES (2, 'Kohli');

select \* from person;



8.In **src/main/resources/application.properties**

spring.application.name=orm-learn

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=9515698138

spring.jpa.hibernate.ddl-auto=validate

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

9.Create a **com.cognizant.orm\_learn.model** and create a class **Person**

**Person.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 = "person")

public class Person {

@Id

@Column(name = "id")

private int id;

@Column(name = "name")

private String name;

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; } @Override

public String toString() {

return "Person [id=" + id + ", name=" + name + "]";

}

}

10.Create a **com.cognizant.orm\_learn.repository** and create a class **PersonRepository**

**PersonRepository.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.Person;

@Repository

public interface PersonRepository extends JpaRepository<Person, Integer> {

}

11. Create a **com.cognizant.orm\_learn.service** and create a class **PersonService**

**PersonService.java:**

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.Person;

import com.cognizant.orm\_learn.repository.PersonRepository;

*@Service*

public class PersonService {

*@Autowired*

private PersonRepository personRepository;

*@Transactional*

public List<Person> getAllPersons() {

return personRepository.findAll();

}

}

12.**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.Person;

import com.cognizant.orm\_learn.service.PersonService;

*@SpringBootApplication*

public class OrmLearnApplication {

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

public static void main(String[] args) {

ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);

PersonService personService = context.getBean(PersonService.class);

*testGetAllPersons*(personService);

}

private static void testGetAllPersons(PersonService personService) {

***LOGGER***.info("Start");

List<Person> persons = personService.getAllPersons();

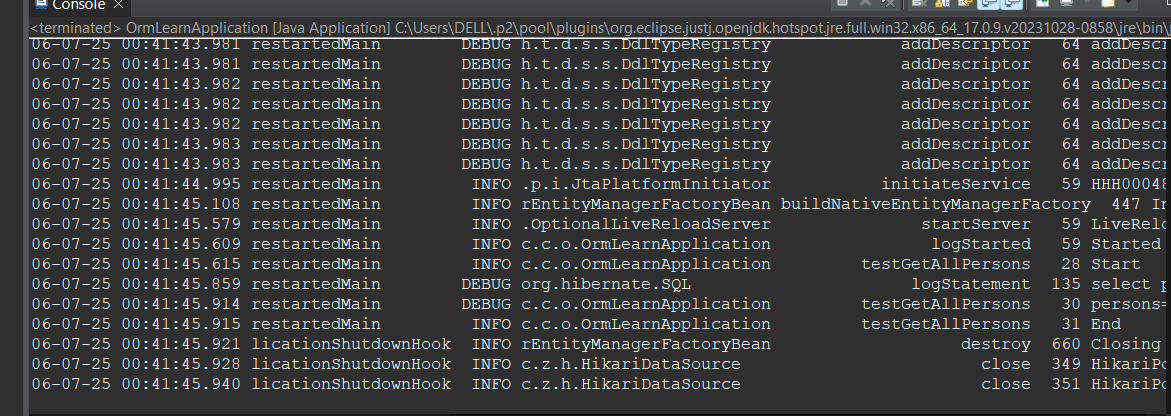
***LOGGER***.debug("persons={}", persons);

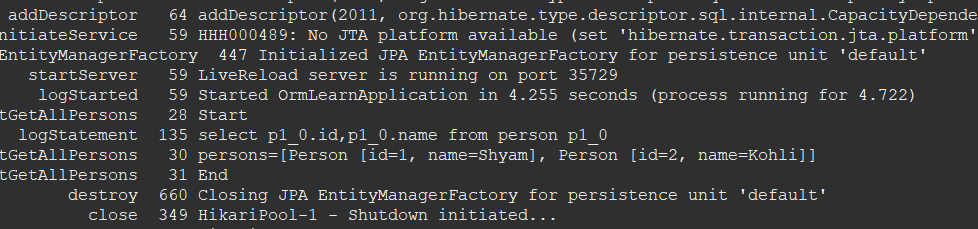
***LOGGER***.info("End");

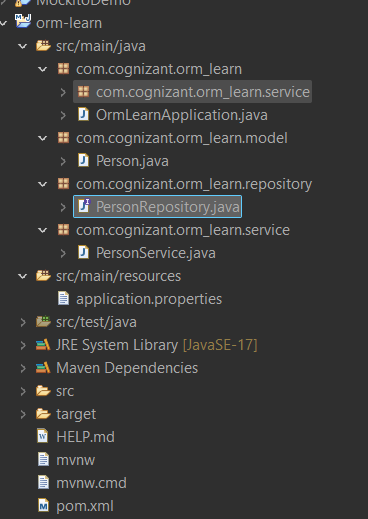
}

}

**Output:**







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

1.Initially create a database in MYSQL and create a table as below

CREATE DATABASE employee\_db;

USE employee\_db;

CREATE TABLE employee (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

department VARCHAR(100)3.

);

2. Go to <https://start.spring.io/>

3.Do the following

Group: com.example

ArtifactId: springDataJpaDemo

Description: Demo project for Spring Data JPA and Hibernate

4. Add the following dependencies

Spring Boot DevTools

Spring Data JPA

MySQL Driver

5. Click Generate, then download the zip and Extract the ZIP into your Eclipse Workspace folder.

6. In Eclipse:

**File > Import > Maven > Existing Maven Projects,** Select the extracted folder, then click Finish.

7.In **src/main/resources/application.properties**

spring.application.name=springDataJpaDemo

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

spring.datasource.username=root

spring.datasource.password=9515698138

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

8. Create a com.example.springDataJpaDemo.model and create a class Employee

**Employee.java**

**package** com.example.springDataJpaDemo.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;

**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; }

}

9. Create a com.example.springDataJpaDemo.repository and create a class EmployeeRepository

**EmployeeRepository.java**

package com.example.springDataJpaDemo.repository;

import com.example.springDataJpaDemo.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

10. Create a com.example.springDataJpaDemo.service and create a class EmployeeService

**EmployeeService.java**

package com.example.springDataJpaDemo.service;

import com.example.springDataJpaDemo.model.Employee;

import com.example.springDataJpaDemo.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);

}

}

11. **SpringDataJpaDemoApplication.java**

package com.example.springDataJpaDemo;

import com.example.springDataJpaDemo.model.Employee;

import com.example.springDataJpaDemo.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 SpringDataJpaDemoApplication implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(SpringDataJpaDemoApplication.class, args);

}

@Override

public void run(String... args) {

Employee emp = new Employee();

emp.setName("Hardik");

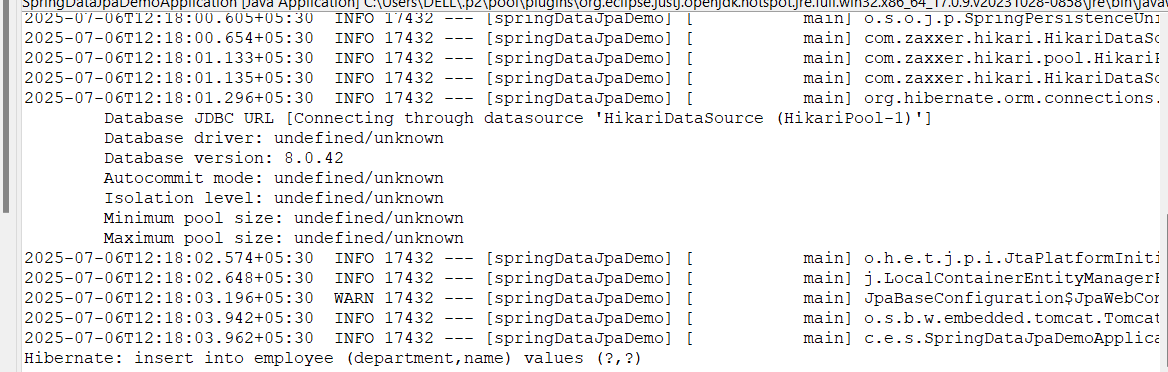
emp.setDepartment("ECE");

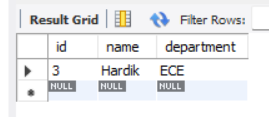
employeeService.addEmployee(emp);

}

}

Output:





**Hibernate Method:**

1.Initially create Maven project

File->new->Maven Project

2.Update pom.xml file

**pom.xml file:**

<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>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

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

<artifactId>hibernate-core</artifactId>

<version>6.4.4.Final</version>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

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

<version>8.3.0</version>

</dependency>

<dependency>

<groupId>jakarta.persistence</groupId>

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

<version>3.1.0</version>

</dependency>

</dependencies>

</project>

3.In src/main/java repository create a ackage com.example and create two classes Employee and HibernateExample

**Employee.java**

package com.example;

import jakarta.persistence.\*

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

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; }

}

**HibernateRxample.java**

package com.example;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

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("Dhoni");

emp.setDepartment("CSE");

session.persist(emp)

tx.commit();

System.out.println("Employee inserted successfully!");

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}

4.In src/main/resources repository create a hibernate.cfg.xml file

**hibernate.cfg.xml file:**

<!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">9515698138</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>

5.Run the application as Java Run

Output:

