**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,

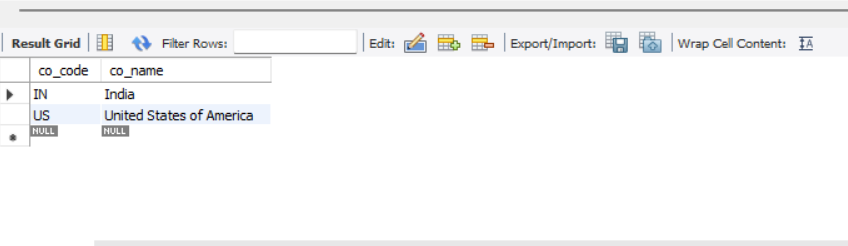
co\_name VARCHAR(50)

);

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

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

select \* from country;



**src/main/resources/application.properties:**

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=rootkp

spring.jpa.hibernate.ddl-auto=validate

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

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

**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 + "]";

}

}

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

**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> {

}

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

**CountryService.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.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);

LOGGER.info("Inside main");

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

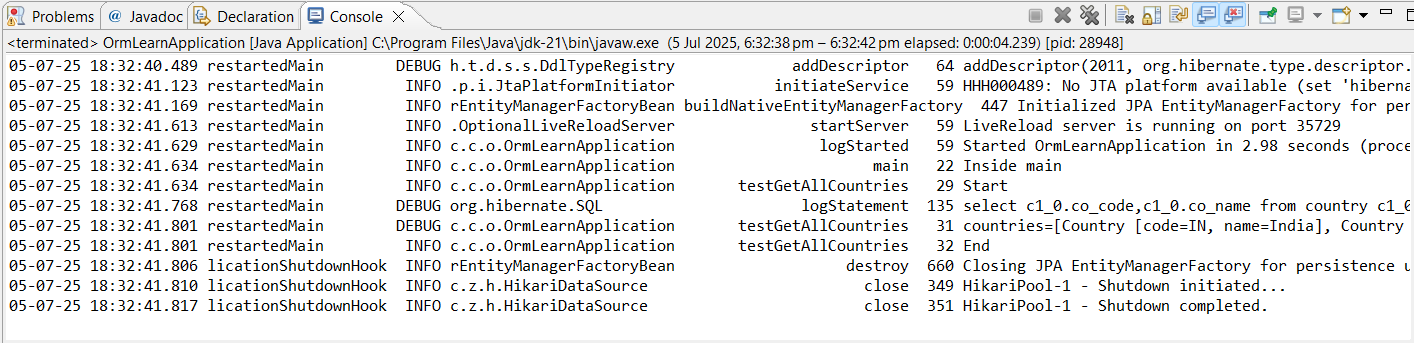
LOGGER.debug("countries={}", countries);

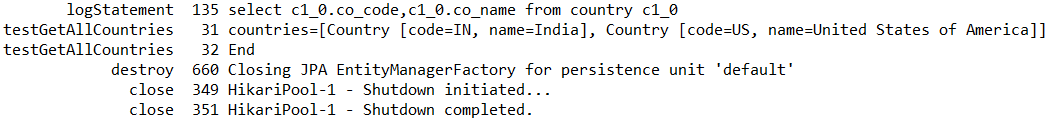
LOGGER.info("End");

}

}

**Output:**

****

****

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

**Java Persistence API (JPA):**

* JSR 338 Specification for persisting, reading and managing data from Java objects.
* Does not contain concrete implementation of the specification.
* Hibernate is one of the implementation of JPA.

**Hibernate:**

* ORM Tool that implements JPA.

**Spring Data JPA:**

* Does not have JPA implementation, but reduces boiler plate code.
* This is another level of abstraction over JPA implementation provider like Hibernate.
* Manages transactions.

**MySql Setup:**

CREATE DATABASE employee\_db;

USE employee\_db;

CREATE TABLE employee (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

department VARCHAR(100),

salary DOUBLE

);

**Spring Data JPA Approach:**

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

2.Fill in:

* Group: com.example
* 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.application.name=demo

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

spring.datasource.username=root

spring.datasource.password=rootkp

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**Create com.example.demo.model package and create Employee.java class**

**Employee.java:**

package com.example.demo.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.persistence.Column;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

@Column(name = "id")

private int id;

@Column(name = "name")

private String name;

@Column(name = "department")

private String department;

@Column(name = "salary")

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;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", department=" + department + ", salary=" + salary + "]";

}

}

**Create com.example.demo.repository package and create EmployeeRepository.java class**

**EmployeeRepository.java:**

package com.example.demo.repository;

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

import com.example.demo.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**Create com.example.demo.service package and create EmployeeService.java class**

**EmployeeService.java:**

package com.example.demo.service;

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

import org.springframework.stereotype.Service;

import com.example.demo.model.Employee;

import com.example.demo.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public Employee getEmployee(int id) {

return employeeRepository.findById(id).orElse(null);

}

public Employee addEmployee(Employee employee) {

return employeeRepository.save(employee);

}

}

**DemoApplication.java:**

package com.example.demo;

import com.example.demo.model.Employee;

import com.example.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("Josh");

emp.setDepartment("IT");

emp.setSalary(40000);

Employee savedEmp = employeeService.addEmployee(emp);

System.*out*.println("Added Employee: " + savedEmp);

int id = savedEmp.getId();

Employee fetchedEmp = employeeService.getEmployee(id);

if (fetchedEmp != null) {

System.*out*.println("Fetched Employee: " + fetchedEmp);

}

else {

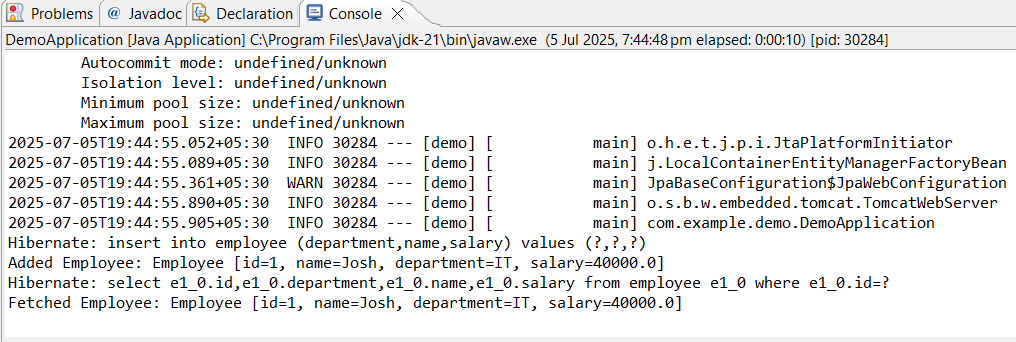
System.*out*.println("Employee not found with ID: " + id);

}

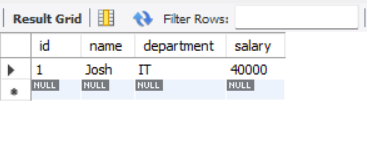
}

}

**Output:**



**Employee table in mysql:**



**Hibernate Approach:**

Create a new Maven project:

* File → New → 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>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.17.0</version>

</dependency>

</dependencies>

<build>

<resources>

<resource>

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

</resource>

</resources>

</build>

</project>

**In src/main/resources, create a new XML file named hibernate.cfg.xml.**

**hibernate.cfg.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

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

"https://hibernate.org/dtd/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">rootkp</property>

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

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

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

<property name="connection.pool\_size">5</property>

<property name="current\_session\_context\_class">thread</property>

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

</session-factory>

</hibernate-configuration>

**Create com.example package and create Employee.java class**

**Employee.java:**

package com.example;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.persistence.Column;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.*IDENTITY*)

@Column(name = "id")

private int id;

@Column(name = "name")

private String name;

@Column(name = "department")

private String department;

@Column(name = "salary")

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;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", department=" + department + ", salary=" + salary + "]";

}

}

**In com.example package create HibernateExample.java class**

**HibernateExample.java:**

package com.example;

import org.hibernate.HibernateException;

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("hibernate.cfg.xml")

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

Transaction tx = null;

Transaction fetchTx = null;

try {

Session session = factory.getCurrentSession();

tx = session.beginTransaction();

Employee emp = new Employee();

emp.setName("Gary");

emp.setDepartment("HR");

emp.setSalary(44000);

session.persist(emp);

System.*out*.println("Inserted: " + emp);

tx.commit();

Session newSession = factory.getCurrentSession();

fetchTx = newSession.beginTransaction();

Employee fetched = newSession.byId(Employee.class).load(emp.getId());

System.*out*.println("Fetched: " + fetched);

fetchTx.commit();

}

catch (HibernateException e) {

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

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

e.printStackTrace();

}

finally {

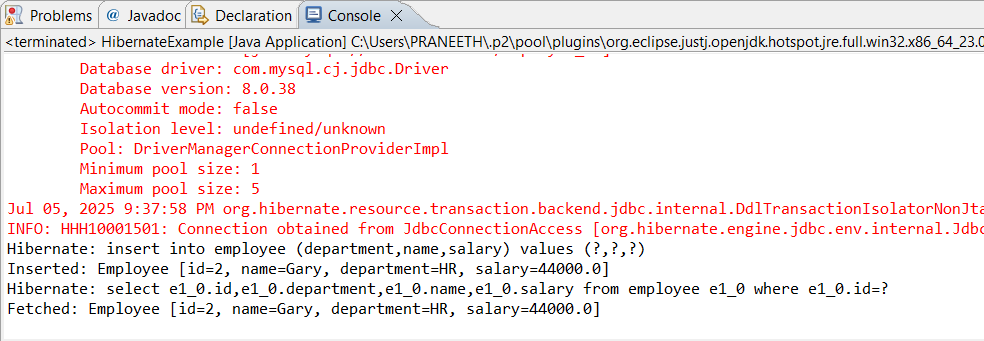
factory.close();

}

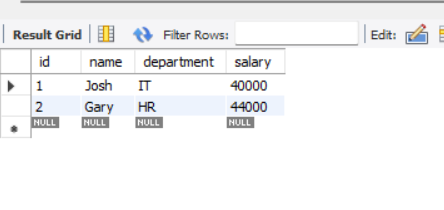
}

}

**Output:**

****

**employee table in MySQL:**

****