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

# Hibernate Employee Project Code

## Java Code

**Employee.java**package com.example.hib;  
  
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 Employee() {}  
  
 public Employee(String name, String department) {  
 this.name = name;  
 this.department = 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; }  
}

**EmployeeDao.java**package com.example.hib;  
  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.Transaction;  
import org.hibernate.cfg.Configuration;  
  
public class EmployeeDao {  
 public EmployeeDao() {  
 Employee emp = new Employee("John", "HR");  
  
 Configuration cfg = new Configuration().configure().addAnnotatedClass(Employee.class);  
 SessionFactory factory = cfg.buildSessionFactory();  
 Session session = factory.openSession();  
 Transaction tx = session.beginTransaction();  
  
 session.save(emp);  
 tx.commit();  
  
 System.out.println("Saved employee with ID: " + emp.getId());  
 session.close();  
 }  
}

package com.example.hib;  
  
public class Main {  
 public static void main(String[] args) {  
 new EmployeeDao();  
 }  
}

**Main.java  
package** com.example.hib;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

EmployeeDao dao = **new** EmployeeDao();

Integer id = dao.add(**new** Employee("Alice", 55000.0));

System.***out***.println("Saved employee with ID: " + id);

}

}

## hibernate.cfg.xml

<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/demo</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.hib.Employee"/>  
 </session-factory>  
</hibernate-configuration>

**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.hib</groupId>

<artifactId>hib-demo</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

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

<artifactId>hibernate-core</artifactId>

<version>6.5.2.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>

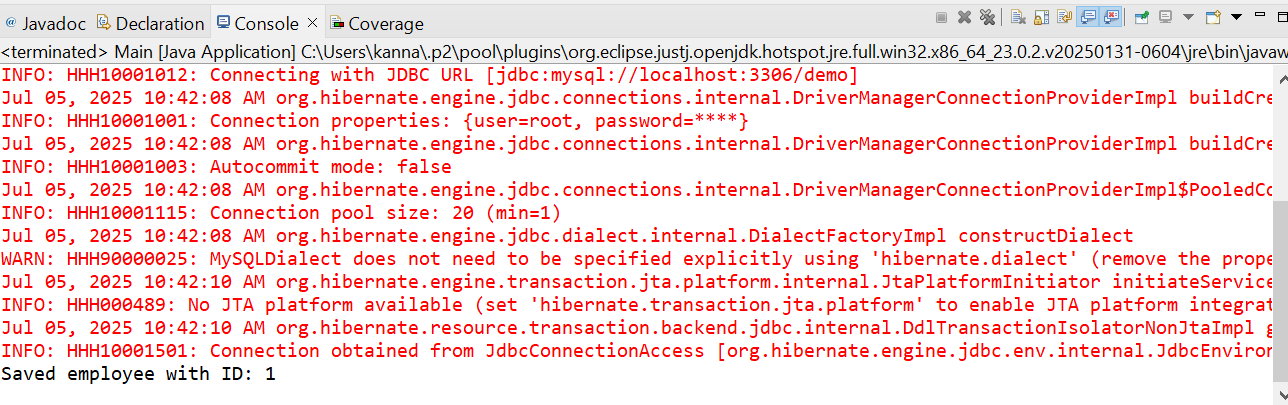
<version>3.1.0</version>

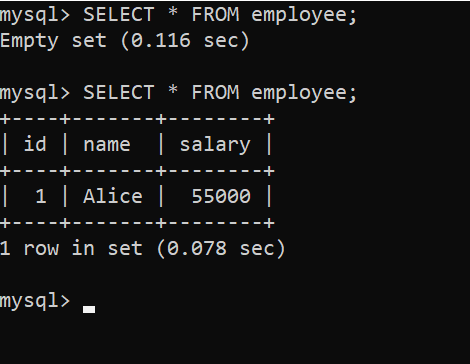
</dependency>

</dependencies>

</project>

**Output:**





## Spring Data JPA Version of the Employee Project

**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>

<parent>

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

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

<version>3.2.4</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>sdata-demo</artifactId>

<version>1.0.0</version>

<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>8.3.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/demo

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**Employee.java**

package com.example.sdata\_demo;

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 Employee() {}

public Employee(String name, String department) {

this.name = name;

this.department = department;

}

}

**EmployeeRepository.java**

package com.example.sdata\_demo;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

**EmployeeService.java**

package com.example.sdata\_demo;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repo;

@Transactional

public void add(Employee e) {

repo.save(e);

}

}

**SdataDemoApplication.java**

package com.example.sdata\_demo;

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 SdataDemoApplication implements CommandLineRunner {

@Autowired

private EmployeeService service;

public static void main(String[] args) {

SpringApplication.run(SdataDemoApplication.class, args);

}

@Override

public void run(String... args) {

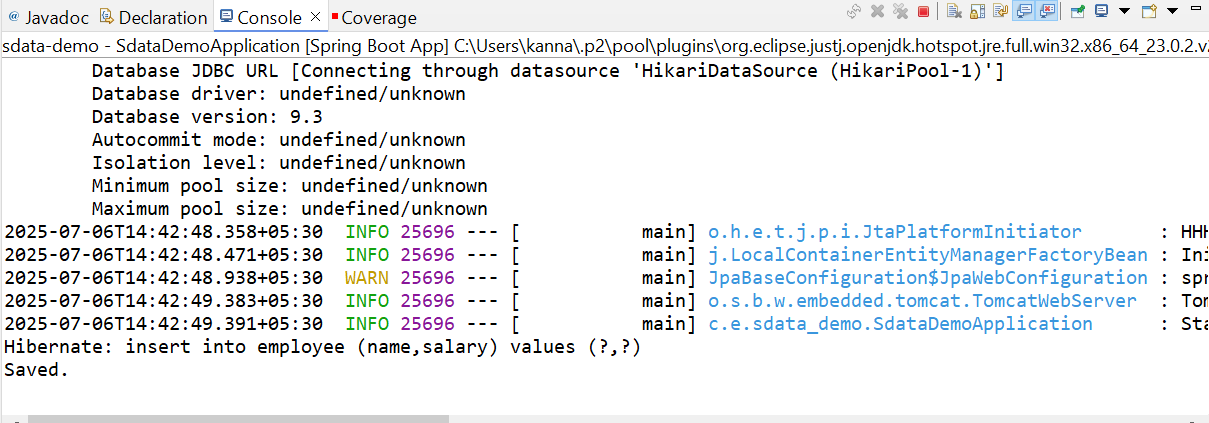
service.add(new Employee("John", "HR"));

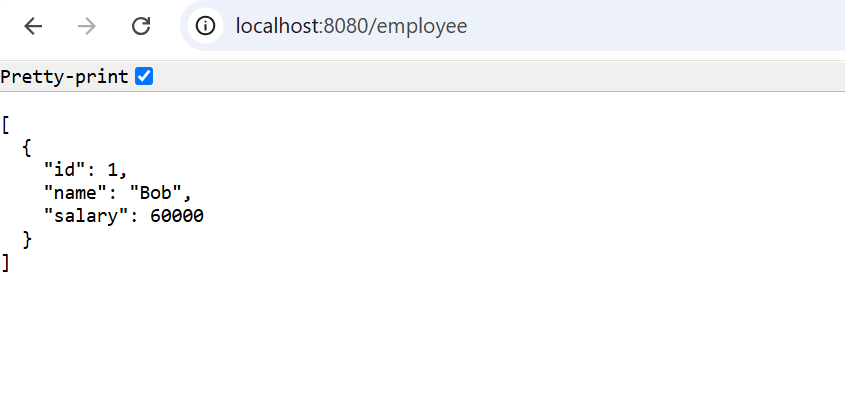
System.out.println("Saved.");

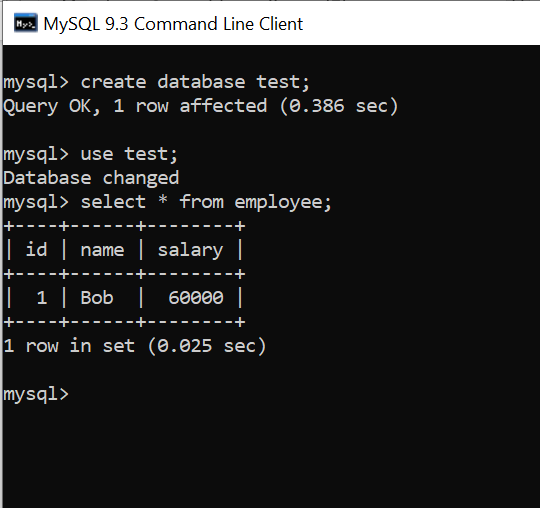
}

}

**Output:**







**Differences:**

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Definition** | Specification (JSR 338) for object-relational mapping | A concrete implementation of JPA | Abstraction layer on top of JPA (e.g., Hibernate) |
| **Type** | Interface/standard | Framework (ORM tool) | Spring module (extension of Spring Data) |
| **Boilerplate Code** | Needs to be written manually | Less than JPA but still manual | Eliminates most boilerplate with auto-generated methods |
| **Implementation** | Doesn't include one (must use Hibernate, EclipseLink, etc.) | Implements JPA + adds extra features | Relies on JPA provider (like Hibernate) |
| **Query Language** | JPQL (Java Persistence Query Language) | JPQL + native SQL + HQL | Method name queries + JPQL + custom queries |
| **Configuration** | persistence.xml | hibernate.cfg.xml or annotations | application.properties or application.yml |
| **Transaction Handling** | Manual | Manual or declarative | Declarative via @Transactional |
| **Repository Layer** | Needs to be manually coded | Needs to be manually coded | Auto-implemented using JpaRepository, CrudRepository |
| **Ease of Use** | Low (more manual work) | Medium | High (minimal code, powerful abstraction) |
| **Use Case** | When you want a standard abstraction over ORM | When you need full control over ORM features | When you want rapid development with minimal code |