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

1. **JPA**

* **What it is jpa:**
* **A Java specification (part of Java EE / Jakarta EE).**
* **Defines a standard way to map Java objects (entities) to database tables.**
* **Provides a common API for ORM (Object Relational Mapping).**
* **Key features:**
* **Uses annotations like @Entity, @Id, @Table, etc.**
* **Allows you to write object-based queries using JPQL (Java Persistence Query Language).**
* **Handles CRUD operations via the EntityManager.**
* **Importance of jpa:**
* **JPA is just a specification—it doesn't do anything by itself.**
* **You need a provider/implementation (like Hibernate) to actually perform database operations.**

**EXAMPLE CODE:**

import jakarta.persistence.\*;

public class Main {

public static void main(String[] args) {

EntityManagerFactory emf = Persistence.createEntityManagerFactory("myPU");

EntityManager em = emf.createEntityManager();

Student student = new Student();

student.setName("Ishika");

student.setEmail("ishika@example.com");

em.getTransaction().begin();

em.persist(student);

em.getTransaction().commit();

System.out.println("Student Saved with ID: " + student.getId());

em.close();

emf.close();

}

}

1. **HIBERNATE**

* **What is hibernate :**
* **A popular ORM framework in Java.**
* **It is a JPA implementation that follows the JPA rules and also adds extra features.**
* **You can use Hibernate with or without JPA.**
* **Extra features Hibernate offers beyond JPA:**
* **First- and second-level caching**
* **Hibernate-specific query language (HQL)**
* **Automatic dirty checking**
* **Enhanced performance features**
* **How it works :**
* **When you use JPA, Hibernate is often the engine under the hood actually doing the work.**

**EXAMPLE CODE:**

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class Main {

public static void main(String[] args) {

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.addAnnotatedClass(Student.class)

.buildSessionFactory();

Session session = factory.openSession();

Student student = new Student();

student.setName("Ishika Hibernate");

student.setEmail("ishika@hibernate.com");

session.beginTransaction();

session.save(student);

session.getTransaction().commit();

System.out.println("Student Saved with ID: " + student.getId());

session.close();

factory.close();

}

}

1. **SPRING DATA JPA:**

* **What it is:**
* **A part of the Spring Framework (a module in Spring Data).**
* **It builds on top of JPA to make data access even easier.**
* **Works with JPA providers like Hibernate.**
* **What it simplifies:**
* **You don’t need to write boilerplate code for repositories or DAO classes.**
* **You can define repository interfaces, and Spring will automatically generate implementations.**
* **Supports method-based query generation, pagination, sorting, and custom queries.**
* **Why better:**
* **Instead of writing a full class to fetch users by email, in Spring Data JPA you just define a method findByEmail() and it automatically works.**

**EXAMPLE CODE:**

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 MyApp implements CommandLineRunner {

@Autowired

private StudentRepository studentRepository;

public static void main(String[] args) {

SpringApplication.run(MyApp.class, args);

}

@Override

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

Student student = new Student();

student.setName("Ishika Spring");

student.setEmail("ishika@spring.com");

studentRepository.save(student);

System.out.println("Student saved using Spring Data JPA with ID: " + student.getId());

}

**}**