**Hands-On 4: Difference between JPA, Hibernate, and Spring Data JPA**

**Introduction**

In this hands-on, we’ll understand the basic differences between **JPA**, **Hibernate**, and **Spring Data JPA** with real-world-style. These tools are used for database interaction in Java applications but differ in their roles and abstraction levels.

**1. Java Persistence API (JPA)**

* JPA is a **specification** (JSR 338) for managing relational data in Java applications.
* It provides guidelines and interfaces but **no implementation**.
* It requires a provider like Hibernate or EclipseLink to work.

Think of JPA as a set of rules that others must follow. It doesn't do anything by itself but tells others what to do.

**2. Hibernate**

* Hibernate is an **ORM (Object Relational Mapping)** tool and an **implementation of JPA**.
* It maps Java objects to database tables and handles all the low-level JDBC work.
* It offers extra features beyond JPA like **caching**, **lazy loading**, etc.

**🔧 Example (Hibernate Style):**

*public Integer addEmployee(Employee employee) {*

*Session session = factory.openSession();*

*Transaction tx = null;*

*Integer employeeID = null;*

*try {*

*tx = session.beginTransaction();*

*employeeID = (Integer) session.save(employee);*

*tx.commit();*

*} catch (HibernateException e) {*

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

*e.printStackTrace();*

*} finally {*

*session.close();*

*}*

*return employeeID;*

}

➡️ Here, you can see that we manually handle session and transaction management.

**3. Spring Data JPA**

* A **high-level abstraction** over JPA.
* Uses JPA providers (like Hibernate) under the hood.
* Reduces **boilerplate code** like writing DAOs or handling transactions.
* Comes with **ready-made methods** such as .save(), .deleteById(), etc.

**🔧 Example (Spring Data JPA Style):**

*EmployeeRepository.java*

*java*

*CopyEdit*

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

*}*

*EmployeeService.java*

*java*

*CopyEdit*

*@Autowired*

*private EmployeeRepository employeeRepository;*

*@Transactional*

*public void addEmployee(Employee employee) {*

*employeeRepository.save(employee);*

*}*

➡️ Clean and simple. No manual session or transaction handling required.