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

Understanding the differences between JPA, Hibernate, and Spring Data JPA is essential for mastering modern Java persistence techniques. Although they work closely together, each plays a distinct role in the persistence layer of a Java application.

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

* JPA is a **specification** (not an implementation) defined by **JSR 338**.
* It provides a standard for **object-relational mapping (ORM)** in Java.
* JPA allows Java objects to be mapped to relational database tables.
* It defines interfaces like EntityManager, Query, PersistenceContext, etc.
* **Does not do anything on its own** – it needs an implementation like Hibernate.

**Example:**

@PersistenceContext

private EntityManager entityManager;

public void saveEmployee(Employee emp) {

entityManager.persist(emp);

}

**🔹 2. Hibernate**

* Hibernate is an **ORM framework** that provides a **concrete implementation of JPA**.
* It also provides additional powerful features that go beyond the JPA specification (like caching, criteria queries, etc.).
* Prior to JPA, Hibernate had its own API; after JPA, it supports both its native API and JPA.

**Example (Hibernate-native approach):**

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;

}

**3. Spring Data JPA**

* Spring Data JPA is a **wrapper/abstraction layer** built on top of JPA (and its implementations like Hibernate).
* It reduces boilerplate code by providing **built-in repositories** like JpaRepository.
* Automatically handles **CRUD operations**, pagination, sorting, and query generation from method names.
* Manages transactions and integrates smoothly with Spring Boot.

**Example (Spring Data JPA approach):**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}