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

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

* **What it is**: A *specification* (defined by JSR 338) — it provides **interfaces and annotations**, but **no actual logic**.
* **Example**: Think of it like an interface in Java. It says “you should have @Entity, @Id, EntityManager, etc.” — but doesn’t say **how** they should work.
* **You can’t use JPA alone**. You need an **implementation** like Hibernate.

**2. Hibernate**

* **What it is**: A **concrete implementation** of the JPA spec.
* **Adds more features** on top of the JPA spec (like caching, lazy loading, HQL, etc.).
* You have to manage:
  + SessionFactory
  + Session
  + Transactions manually (unless integrated with Spring)

**3. Spring Data JPA**

* **What it is**: A **Spring abstraction** on top of JPA and Hibernate.
* **Does not implement JPA itself**, but uses Hibernate (or other JPA providers) under the hood.
* **Eliminates boilerplate code** — just create a Repository interface and Spring handles all the heavy lifting.
* Comes with:
  + Automatic @Transactional support
  + Method-based queries (findByName, etc.)
  + Paging, sorting, and query derivation

In **Hibernate (**It’s all manual**):-**

public Integer addEmployee(Employee employee){  
 Session session = factory.openSession(); // manually get session  
 Transaction tx = null;  
 Integer employeeID = null;  
  
 try {  
 tx = session.beginTransaction(); // manually start transaction  
 employeeID = (Integer) session.save(employee);  
 tx.commit(); // manually commit  
 } catch (HibernateException e) {  
 if (tx != null) tx.rollback(); // manually rollback  
 e.printStackTrace();  
 } finally {  
 session.close(); // manually close session  
 }  
 return employeeID;  
}

### On the other hand if you are using Spring Data JPA (Auto-managed):-

### You just have to create :-

### 1.Repository Inteface

import org.springframework.stereotype.Repository;  
  
@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

2.Service Class

import jakarta.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Transactional // transaction automatically handled  
 public void addEmployee(Employee employee) {  
 employeeRepository.save(employee); // only one line!  
 }  
}

And your query automatically runs and every basic crud functionality and query is accessible from JPA Repository which is being extended by your Interface\_name\_Repository.java Interface