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

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

· JPA is a **Java specification** for accessing, persisting, and managing data between Java objects and a relational database.

· It is **only an interface**, meaning it provides a set of rules (APIs), but not the actual code to perform operations.

· It defines concepts like **Entity**, **EntityManager**, and annotations such as @Entity, @Id, etc.

· **JPA is implemented by tools like Hibernate**

**2.** **Hibernate**

· Hibernate is an **ORM (Object-Relational Mapping) framework** and one of the most popular implementations of JPA.

· It provides the **actual code (logic)** to map Java objects to database tables and handle database operations.

· It includes both **JPA-compliant features** and additional **non-standard features** (like caching and performance optimizations).

· Developers must **manually manage sessions and transactions** if not using Spring.

**3.** **Spring Data JPA**

· Spring Data JPA is a **part of the Spring Framework** that builds on top of JPA.

· It **does not implement JPA**, but provides an abstraction that reduces the need for writing boilerplate code.

· With Spring Data JPA, you can **create repository interfaces instead of writing your own queries**.

· It **integrates smoothly with Spring Boot** and **manages transactions automatically**.

**Code Comparison -** **Hibernate Code (Manual Configuration)**

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 (Exception e) {

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

} finally {

session.close();

}

return employeeID;

}

· Requires manual session handling

· Must explicitly manage transactions

· More boilerplate code

**Spring Data JPA Code**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}