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

**Java Persistence API (JPA)**

* JPA is a **specification** that provides a set of rules and interfaces for ORM (Object-Relational Mapping).
* It does not contain any implementation.
* It requires a **provider** (like Hibernate or EclipseLink) to function.
* With JPA, developers write SQL-like queries (JPQL) and manage entities using the EntityManager.

**Example using JPA:**

@PersistenceContext

private EntityManager entityManager;

@Transactional

public void saveEmployee(Employee employee) {

entityManager.persist(employee);

}

**Hibernate**

* Hibernate is an ORM tool that provides a concrete implementation of JPA.
* It also provides extra features such as lazy loading, caching, and custom query language (HQL).
* You interact directly with the Hibernate Session object to perform CRUD operations.

**Example using Hibernate:**

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;

}

**Spring Data JPA**

* Spring Data JPA is a wrapper around JPA and Hibernate that eliminates most of the boilerplate code.
* It provides JpaRepository and CrudRepository interfaces which include basic CRUD methods.
* It automatically handles transaction management and query generation.
* You only need to define method names and Spring will generate the queries for you.

**Example using Spring Data JPA:**

**Employee.java**

@Entity

public class Employee {

@Id

private int id;

private String name;

}

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

List<Employee> findByName(String name); // Spring will auto-generate query

}

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository repository;

@Transactional

public void addEmployee(Employee employee) {

repository.save(employee);

}

public List<Employee> getByName(String name) {

return repository.findByName(name);

}

}

**Summary:**

* JPA is just a standard – it defines *what* should be done but not *how*.
* Hibernate is a tool – it does the actual ORM work by following JPA rules (and more).
* Spring Data JPA is a helper layer – it simplifies Hibernate and JPA by generating most of the boilerplate code for you.