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

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

JPA is a specification that defines a set of interfaces and annotations to manage relational data in Java applications. It is not a framework or library but rather a standard that must be implemented by a provider like Hibernate. JPA provides annotations like @Entity, @Id, and @OneToMany to map Java classes to database tables. Because it is just a specification, it cannot perform any database operations by itself and needs an implementation to work.

**Example:**

@Entity

public class Employee {

@Id

private Long id;

private String name;

}

// Using JPA

EntityManager em = emf.createEntityManager();

em.getTransaction().begin();

em.persist(new Employee(1L, "Alice"));

em.getTransaction().commit();

**2. Hibernate**

Hibernate is a framework that provides a complete implementation of the JPA specification. It handles the actual database operations and object-relational mapping (ORM) under the hood. In addition to supporting all JPA features, Hibernate offers many advanced functionalities like lazy loading, caching, batch processing, and support for HQL (Hibernate Query Language). You can use Hibernate directly with its own APIs, or indirectly as the JPA provider in a JPA-based application.

**Example:**

@Entity

public class Employee {

@Id

private Long id;

private String name;

}

// Using Hibernate

Session session = factory.openSession();

session.beginTransaction();

session.save(new Employee(1L, "Alice"));

session.getTransaction().commit();

**3. Spring Data JPA**

Spring Data JPA is a Spring framework module that builds on top of JPA to make database access easier and less error-prone. It provides ready-to-use repository interfaces like JpaRepository that reduce the need to write boilerplate code for common CRUD operations. Spring Data JPA also supports query methods based on method naming conventions, automatic query generation, and integration with Spring Boot for seamless configuration. It uses a JPA provider like Hibernate under the hood to perform the actual data access.

**Example:**

@Entity

public class Employee {

@Id

private Long id;

private String name;

}

public interface EmployeeRepo extends JpaRepository<Employee, Long> {}

// In a service/controller

employeeRepo.save(new Employee(1L, "Alice"));