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

| **Feature/Aspect** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (part of Java EE) | Implementation of JPA | Spring abstraction over JPA (uses Hibernate by default) |
| Provided By | Oracle / Java EE | Red Hat | Spring Framework |
| Boilerplate Code | Requires writing queries, entity managers | Still verbose, manual session handling | Removes most boilerplate using repositories |
| Configuration | Manual (persistence.xml, entityManager) | Manual (session factory) | Auto-configured (via Spring Boot) |
| Out-of-the-box CRUD | No | No | Yes (save(), findById(), etc.) |
| Integration with Spring | Needs setup | Needs setup | Seamless (Spring Boot ready) |
| Learning Curve | Moderate | High | Low (easy for beginners) |
| Popular Methods | persist(), merge(), remove() | save(), load(), get() | findById(), save(), deleteById() |
| Querying Style | JPQL | JPQL + HQL + Criteria API | JPQL + Derived Queries + @Query annotation |

**JPA (Standard)**

JPA is a specification that defines a standard API for object-relational mapping (ORM) in Java applications. It provides a set of interfaces and annotations for managing relational data with Java objects. JPA itself does not provide an implementation; it's a blueprint for how ORM tools should behave.

EntityManager em = emf.createEntityManager();

em.persist(entity);

**Hibernate (Implementation)**

Hibernate is a popular implementation of the JPA specification. It's a full-featured ORM framework that provides concrete classes and functionalities to perform object-relational mapping, including session management, transaction handling, and query execution. When you use JPA in your application, you typically use a JPA provider like Hibernate to actually carry out the persistence operations.

Session session = sessionFactory.openSession();

session.save(entity);

**Spring Data JPA (Abstraction)**

Spring Data JPA is a higher-level abstraction built on top of JPA (and thus, implicitly, on a JPA provider like Hibernate). It's part of the Spring Framework and significantly simplifies the development of data access layers. Spring Data JPA provides interfaces like **JpaRepository** that automatically generate common database queries based on method names (e.g., findByFirstNameAndLastName). This reduces boilerplate code and accelerates development by providing a convention-over-configuration approach to data access.

userRepository.save(entity);