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

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification (API) | Implementation (ORM Framework) | Abstraction layer/framework over JPA |
| **Developed By** | Jakarta EE (formerly Java EE) | Red Hat | Spring (Pivotal/VMware) |
| **Purpose** | Standardizes ORM for Java | Provides a working ORM with JPA + extra features | Simplifies data access with repositories |
| **Implementation** | Interfaces only (e.g., EntityManager) | Concrete implementation of JPA | Uses JPA implementations like Hibernate |
| **Boilerplate Code** | Moderate (DAO, queries must be written manually) | Slightly reduced (HQL, Criteria API) | Minimal (auto-generates queries from method names) |
| **Vendor Neutral** | Yes | No (specific ORM provider) | No (built on JPA, usually with Hibernate) |
| **Requires Hibernate** | No (can use any provider: EclipseLink, OpenJPA) | No (Hibernate is itself an implementation) | Yes/No (requires JPA impl, commonly Hibernate) |
| **Ease of Use** | Standard-level | Easier than raw JDBC | Very high (especially with Spring Boot) |
| **Query Language** | JPQL (Java Persistence Query Language) | JPQL + HQL + Criteria API | JPQL + Derived Queries + @Query annotation |
| **Spring Boot Integration** | Possible but manual setup | Supported | Fully integrated |
| **Transaction Management** | Manual via EntityManager | Built-in | Declarative and simplified with Spring |
| **Popular Annotations** | @Entity, @Id, @OneToMany, @Table, etc. | Same as JPA + some extras (@GenericGenerator, etc.) | Reuses JPA annotations + Spring-specific (@Repository) |
| **Example Interface** | N/A (define DAO manually) | SessionFactory, Criteria, etc. | JpaRepository, CrudRepository, PagingAndSortingRepository |
| **Target Users** | Developers needing standard ORM APIs | Developers needing powerful ORM features | Spring developers needing fast and clean data access |