# Difference between JPA, Hibernate, and Spring Data JPA

This document highlights the key differences between JPA, Hibernate, and Spring Data JPA, which are commonly used for object-relational mapping and database access in Java applications.

|  |  |  |  |
| --- | --- | --- | --- |
| Aspect | JPA | Hibernate | Spring Data JPA |
| Type | Specification (Interface) | Implementation of JPA | Framework built on top of JPA and Hibernate |
| What it is | A set of guidelines for ORM (Object-Relational Mapping) | A popular ORM framework that implements JPA | A Spring module that simplifies JPA-based data access |
| Maintained by | Oracle (Java EE / Jakarta EE) | Red Hat | Spring Team (VMware) |
| Requires boilerplate | Yes – developers write EntityManager, queries, etc. | Less than JPA but still requires session/config code | No – auto-implements repository and query methods |
| Configuration | Manual setup of persistence units and mappings | Requires Hibernate-specific settings | Minimal setup with Spring Boot or Spring Configuration |
| Usage | Used to define entities, relationships, and lifecycle | Used to persist Java objects using SQL | Used for creating repositories and query methods quickly |
| Query Language | JPQL (Java Persistence Query Language) | HQL (Hibernate Query Language), supports JPQL too | Method-based queries, JPQL, native SQL |
| Example | @Entity, EntityManager | SessionFactory, @Entity | JpaRepository, findByName(), @Query |
| Main Goal | Standardize ORM in Java | Provide ORM functionality | Simplify JPA + Hibernate usage in Spring projects |

## Summary:

- JPA is a specification that defines how Java objects are mapped to relational databases.  
- Hibernate is an implementation of JPA that provides the actual ORM engine.  
- Spring Data JPA is built on top of JPA and Hibernate to further simplify data access in Spring-based applications.