# Difference between JPA, Hibernate, and Spring Data JPA

|  |  |  |  |
| --- | --- | --- | --- |
| Feature / Aspect | JPA | Hibernate | Spring Data JPA |
| Type | Specification (Interface/API) | Framework / Implementation | Framework built on top of JPA & Spring |
| Defined By | Java EE / Jakarta EE | Red Hat | Pivotal / Spring Team |
| Purpose | Defines a standard for ORM | Provides a working ORM implementation | Simplifies and automates JPA-based data access |
| Implementation | Needs provider (e.g., Hibernate, EclipseLink) | Hibernate is one such provider | Uses JPA provider internally (commonly Hibernate) |
| Example Annotation | `@Entity`, `@Id`, `@OneToMany` | Same as JPA, plus `@Fetch`, `@LazyCollection`, etc. | Same JPA annotations + Repository interfaces |
| EntityManager Usage | Required to manage entities manually | Provides Session API (extends JPA’s EntityManager) | Hides EntityManager behind interfaces like `JpaRepository` |
| Query Language | JPQL (Java Persistence Query Language) | JPQL + HQL (Hibernate Query Language) | Supports JPQL + method name-based query generation |
| Boilerplate Code | Requires boilerplate (create queries, etc.) | Slightly less than JPA alone | Minimizes boilerplate via Spring Repositories |
| Ease of Use | Medium | Easier than JPA | Easiest - declarative and convention-based |
| Advanced Features | Limited to standard ORM features | Lazy loading, caching, batch fetch, etc. | Pagination, Sorting, Custom Queries, Projections, etc. |