| **Aspect** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification / API (standard) | Implementation of JPA + extra features | Abstraction layer over JPA implementations (like Hibernate) |
| **Who provides it?** | Oracle (part of Java EE / Jakarta EE) | Red Hat (JBoss) | Spring Framework team |
| **Purpose** | Define ORM standard and API | Provide working ORM engine that follows JPA + more | Simplify JPA-based data access in Spring apps |
| **Implementation** | None (it’s just an interface/API) | Yes — provides actual persistence logic | No (delegates to JPA provider like Hibernate) |
| **Boilerplate reduction** | No — you write all persistence code | Somewhat reduced | Greatly reduces boilerplate (repository interfaces) |
| **Query methods support** | You define JPQL or criteria queries manually | JPQL, HQL, Criteria API | Auto-generates queries from method names |
| **Custom queries** | Via JPQL or Criteria API | JPQL, HQL, Criteria API | JPQL, native SQL, method names, or @Query annotation |
| **Vendor independence** | Yes — just API, no dependency on a provider | No — it is a specific vendor implementation | Yes — works with any JPA provider |
| **Features beyond JPA** | No | Yes — caching, filters, interceptors, multi-tenancy | No (delegates those to underlying JPA provider) |
| **Spring integration** | Needs manual config / EntityManager setup | Needs manual or Spring-specific config | Tight Spring integration out of the box |
| **Repository abstraction** | No | No | Yes — extends JpaRepository, CrudRepository etc. |
| **Ease of use** | Medium — you must write more code | Medium — easier than pure JPA | High — minimal code to get working CRUD |
| **Learning curve** | Steeper — you need to understand JPA concepts fully | Moderate — with JPA plus Hibernate extensions | Easier — Spring handles much behind the scenes |
| **Transaction management** | Needs manual setup or EJB | Works with JPA’s or its own | Integrated with Spring transaction management |
| **Example use case** | When using any JPA-compliant provider | When you want advanced ORM features | When you want to minimize code in Spring apps |