**Definition:**

Java Persistence API (JPA), Hibernate, and Spring Data JPA are three key technologies used in Java applications to handle persistence (i.e., storing and retrieving data from a database). Although they are related, each has a distinct role in the Java persistence ecosystem.

JPA is a **standard specification** provided by Java (JSR 338) that defines how Java objects should be mapped to database tables. However, JPA itself is just an API and **does not contain any implementation**. For actual functioning, JPA requires an implementation provider such as Hibernate.

Hibernate is a **popular implementation of the JPA specification**. It is an Object-Relational Mapping (ORM) tool that provides the actual mechanism to convert Java objects into database records and vice versa. It goes beyond the JPA specification by offering advanced features like caching, lazy loading, and custom SQL.

Spring Data JPA is a **Spring framework module** that builds on top of JPA. It provides an abstraction layer that significantly **reduces boilerplate code** required to implement data access logic. It makes use of repository interfaces to handle standard CRUD operations without the need to write any SQL or JPQL manually.

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (API) | Implementation (ORM Tool) | Framework (Abstraction over JPA + Hibernate) |
| Author/Provider | Oracle / Java EE | Red Hat | Spring Framework |
| Implementation Required? | Yes (e.g., Hibernate) | No, it's a full implementation | Uses JPA provider (like Hibernate) |
| Boilerplate Code | Medium | Medium | Very Low – uses method naming conventions |
| Query Language | JPQL | HQL (similar to JPQL) | Derived Queries / JPQL |
| Ease of Use | Moderate | Moderate | Very Easy |
| Configuration Complexity | Medium | High | Low |
| Integration with Spring | Needs configuration | Needs configuration | Seamless integration |