# Difference between JPA, Hibernate, and Spring Data JPA

## 1. JPA (Java Persistence API)

• JPA is a Java specification (interface-based) for object-relational mapping (ORM).  
• It defines standard APIs to manage relational data in Java applications.  
• JPA does not provide any implementation – it just defines interfaces.  
• Popular implementations include Hibernate, EclipseLink, and OpenJPA.  
• Part of the Jakarta EE (formerly Java EE) specification.

## 2. Hibernate

• Hibernate is a popular ORM framework and the most widely used JPA implementation.  
• It provides advanced features beyond the JPA specification, like caching, auditing, and batch processing.  
• Hibernate can be used with or without JPA.  
• It offers both native APIs (Session, Criteria, etc.) and JPA-compliant APIs.

## 3. Spring Data JPA

• Spring Data JPA is a part of the Spring Data project.  
• It builds on top of JPA and uses an underlying JPA provider (like Hibernate).  
• Provides repository interfaces that reduce boilerplate code.  
• Supports query methods, custom queries, pagination, sorting, etc.  
• Integrates deeply with Spring Boot and Spring ecosystem.

## 4. Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspect** | **JPA** | **Hibernate** | **Spring Data JPA** |
| Type | Specification | Implementation | Framework built on JPA |
| Goal | Standardize ORM in Java | Provide ORM capabilities | Simplify JPA usage with Spring |
| Boilerplate | Manual | Less (but still some) | Very little (auto repo creation) |
| Integration | Works with any JPA provider | Can work standalone or via JPA | Requires Spring + JPA provider |
| Provider | None (just API) | Itself | Uses Hibernate or other JPA providers |