Difference Between JPA, Hibernate, and Spring Data JPA

# 1. JPA (Java Persistence API)

• JPA is a Java specification for accessing, persisting, and managing data between Java objects and relational databases.

• It is not a framework; it only defines a set of interfaces and annotations (e.g., @Entity, @Id, @OneToMany).

• Provided by Jakarta EE (formerly Java EE).

• It requires an implementation like Hibernate to perform operations.

• Example:  
 EntityManager em = entityManagerFactory.createEntityManager();  
 em.persist(new Employee());

# 2. Hibernate

• Hibernate is a powerful ORM framework that implements the JPA specification.

• Provided by Red Hat.

• Can be used with JPA or with its own native APIs (e.g., SessionFactory, HQL).

• Offers additional features like caching, batch processing, and native SQL support.

• Example:  
 Session session = sessionFactory.openSession();  
 session.save(new Employee());

# 3. Spring Data JPA

• Spring Data JPA is part of the Spring ecosystem that builds on top of JPA.

• Provided by the Spring team (VMware).

• Simplifies repository layer by reducing boilerplate code.

• Offers auto-generated queries and integrates seamlessly with Spring Boot.

• Example:  
 public interface EmployeeRepository extends JpaRepository<Employee, Long> {  
 List<Employee> findByName(String name);  
 }

# 4. Relationship Among Them

• Spring Data JPA uses JPA under the hood.

• JPA requires an implementation like Hibernate.

• Hibernate performs the actual ORM work.

• Flow: Spring Data JPA → JPA → Hibernate → Database

# 5. Comparison Table

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification (interface) | Framework (implementation) | Framework/Abstraction Layer |
| **Ownership** | Jakarta EE / Oracle | Red Hat | Spring (VMware) |
| **Uses** | Interfaces & annotations | Native APIs + JPA APIs | JPA + Spring magic |
| **Boilerplate Code** | Medium | Medium | Very Low |
| **Ease of Use** | Moderate | Moderate | Easy |
| **Custom Queries** | JPQL | HQL | JPQL, @Query, Query Methods |
| **Transactions** | Yes | Yes | Spring-managed with @Transactional |
| **Caching** | Not specified | Yes (1st and 2nd level) | Inherited from JPA provider |
| **Spring Boot Friendly** | Partial | Yes | Yes (fully integrated) |

# 6. Conclusion

• JPA defines the rules (standard interfaces).

• Hibernate implements those rules (actual logic).

• Spring Data JPA makes working with JPA much easier and faster by reducing code.