# DIFFERENCE BETWEEN JPA, HIBERNATE AND SPRING DATA JPA

## 1. JPA (Java Persistence API)

* Definition

JPA is a Java specification that provides a standard for ORM (Object-Relational Mapping). It allows developers to map Java objects to database tables using annotations.

* Key Point

JPA is only a specification and does not perform any actual persistence. It needs a provider (like Hibernate, EclipseLink) to implement the underlying functionality.

* Usage

Used in enterprise Java applications to define how objects should be stored, updated, deleted, and retrieved from relational databases in a standardized way.

* Example

@Entity  
public class User {  
 @Id @GeneratedValue  
 private Long id;  
 private String name;  
}  
  
EntityManager em = emf.createEntityManager();  
em.getTransaction().begin();  
em.persist(new User("abc"));  
em.getTransaction().commit();  
em.close();

## 2. Hibernate

* Definition

Hibernate is an open-source ORM framework that implements the JPA specification and adds additional features such as caching, lazy loading, and HQL (Hibernate Query Language).

* Key Point

Hibernate can be used both as a JPA implementation and as a standalone ORM with its own configuration and features.

* Usage

Used when you want more control over SQL generation, caching, and performance optimizations. It gives flexibility for advanced ORM requirements.

* Example

@Entity  
public class User {  
 @Id @GeneratedValue  
 private Long id;  
 private String name;  
}  
  
Session session = factory.openSession();  
session.beginTransaction();  
session.save(new User("abc"));  
session.getTransaction().commit();  
session.close();

## 3. Spring Data JPA

* Definition

Spring Data JPA is a part of the Spring ecosystem that simplifies the use of JPA by eliminating boilerplate code and providing a repository abstraction layer.

* Key Point

It builds on top of JPA and provides ready-to-use CRUD operations and query methods using method naming conventions.

* Usage

Commonly used in Spring Boot applications to handle data access layers with minimal code and configuration. It works out of the box with Hibernate or any other JPA provider.

* Example

@Entity  
public class User {  
 @Id @GeneratedValue  
 private Long id;  
 private String name;  
}  
  
public interface UserRepo extends JpaRepository<User, Long> {}  
  
@Autowired  
UserRepo repo;  
  
repo.save(new User("abc"));

## CONCLUSION

* JPA provides the standard API for ORM in Java.
* Hibernate is a powerful implementation of JPA (and can also work independently).
* Spring Data JPA builds on JPA (usually with Hibernate) to eliminate boilerplate and speed up development.