# Comparison: JPA vs Hibernate vs Spring Data JPA

## 1. Java Persistence API (JPA)

JPA is a specification or standard. It acts like a blueprint that defines how Java applications should interact with databases. However, it doesn't implement these interactions itself. Compared to Hibernate or Spring Data JPA, JPA is more abstract and requires an implementation to be usable.

## 2. Hibernate

Hibernate is an implementation of the JPA specification and provides additional features beyond JPA. Unlike JPA, Hibernate handles actual database interactions such as creating sessions, transactions, and querying. It is more powerful than JPA alone but requires more configuration than Spring Data JPA.

## 3. Spring Data JPA

Spring Data JPA builds on top of JPA and Hibernate to simplify development. In contrast to Hibernate, it significantly reduces boilerplate code by allowing you to define interfaces, and Spring automatically provides implementations for common methods. It is the easiest to use among the three, especially when working with Spring Boot.

## Hibernate Example:

public Integer addEmployee(Employee employee) {  
 Session session = factory.openSession();  
 Transaction tx = null;  
 Integer employeeID = null;  
 try {  
 tx = session.beginTransaction();  
 employeeID = (Integer) session.save(employee);  
 tx.commit();  
 } catch (HibernateException e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 return employeeID;  
}

## Spring Data JPA Example:

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}  
  
@Autowired  
private EmployeeRepository employeeRepository;  
  
@Transactional  
public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
}