# HANDS-ON 4: JPA vs HIBERNATE vs SPRING DATA JPA

In this exercise, we understand the difference between JPA (Java Persistence API), Hibernate (a popular ORM implementation), and Spring Data JPA (a framework built on top of JPA and Hibernate).

## 1. Java Persistence API (JPA)

JPA is a specification that defines how Java objects map to relational databases. It is only an interface – it doesn’t provide any implementation by itself. You need a provider like Hibernate to use it.

## 2. Hibernate

Hibernate is an ORM tool and a popular implementation of JPA. It manages the mapping of Java objects to database tables and handles transactions, caching, and lazy loading. However, you still need to write boilerplate code like session management.

## 3. Spring Data JPA

Spring Data JPA is a part of the Spring ecosystem that simplifies the use of JPA. It removes boilerplate code by providing interfaces like JpaRepository. Query methods are automatically implemented based on method names.

## Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| What it is | API / Specification | JPA Implementation | Abstraction over JPA |
| Boilerplate Code | High | Medium | Very Low |
| Used Alone? | No | Yes | Yes (with Spring Boot) |
| Transaction Handling | Manual | Manual | Spring Manages |
| Save Example | em.persist(obj) | session.save(obj) | repository.save(obj) |

## Code Example: Hibernate

Session session = factory.openSession();  
Transaction tx = session.beginTransaction();  
session.save(new Country("IN", "India"));  
tx.commit();  
session.close();

## Code Example: Spring Data JPA

@Autowired  
private CountryRepository countryRepository;  
  
countryRepository.save(new Country("IN", "India"));