# Hands-on 4

## Difference between JPA, Hibernate, and Spring Data JPA

### 1. Java Persistence API (JPA)

- JPA is a Java specification (JSR 338) for managing relational data in Java applications.  
- It defines standard annotations and interfaces such as @Entity, @Id, and EntityManager, but does not provide an implementation.  
- It is vendor-neutral and requires a concrete implementation like Hibernate to be functional.

### 2. Hibernate

- Hibernate is a popular ORM (Object Relational Mapping) tool and the most widely used implementation of JPA.  
- It provides additional features beyond the JPA specification, including caching and lazy loading.  
- Using Hibernate directly involves manually managing sessions and transactions using classes like SessionFactory, Session, and Transaction.

### 3. Spring Data JPA

- Spring Data JPA is part of the Spring ecosystem and acts as an abstraction layer over JPA.  
- It simplifies data access by providing repository interfaces and using method naming conventions to auto-generate queries.  
- It internally uses a JPA implementation like Hibernate and manages transactions and entity managers automatically.

### 4. Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification | ORM Framework | Abstraction over JPA/Hibernate |
| Implementation Provided | No | Yes | No (uses JPA implementation) |
| Boilerplate Code | Medium | High | Low |
| Requires Configuration | Yes | Yes | Yes (minimal) |
| Repository Abstraction | No | No | Yes (JpaRepository, CrudRepository) |
| Usage in Project | Used annotations like @Entity, @Id | Used internally as JPA provider | Created CountryRepository extending JpaRepository |

### **Code Comparison**

### Using Hibernate:

Session session = factory.openSession();  
Transaction tx = null;  
try {  
 tx = session.beginTransaction();  
 session.save(employee);  
 tx.commit();  
} catch (Exception e) {  
 if (tx != null) tx.rollback();  
} finally {  
 session.close();  
}

**Using Spring Data JPA:**

@Autowired  
private EmployeeRepository employeeRepository;  
  
@Transactional  
public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
}

### 6. Summary of Project Implementation

- In the project, we used:  
- JPA annotations in the Country entity class to map it to the database table.  
- Spring Data JPA to define CountryRepository, which extends JpaRepository.  
- Hibernate as the default JPA provider to manage ORM internally.  
- Spring Boot to handle configuration and dependency injection, reducing boilerplate code significantly.