**Hands on 4**

**Difference between JPA, Hibernate and Spring Data JPA** :

### **Java Persistence API (JPA)**

* JPA is a **Java specification (JSR 338)** that defines how data should be persisted, read, and managed from Java objects to relational databases.
* It **does not provide an implementation**—it only provides interfaces and guidelines.
* Frameworks like **Hibernate** provide the actual implementation of JPA.

### **Hibernate**

* Hibernate is a popular **ORM (Object Relational Mapping)** tool that **implements the JPA specification.**
* It provides features such as lazy loading, caching, and custom queries using HQL (Hibernate Query Language).
* Developers can use Hibernate **directly** or via JPA interfaces.

### **Spring Data JPA**

* Spring Data JPA is a **Spring framework module** that provides **another level of abstraction** over JPA and its implementations (like Hibernate).
* It **does not implement JPA itself**, but it **reduces boilerplate code** by automatically generating repository implementations.
* It handles common operations like save, delete, find, and also manages **transactions** behind the scenes.

### **Code Comparison**

### 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**

EmployeeRepository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}