**Hands-on 4**

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

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (JSR 338) | ORM Framework | Abstraction layer over JPA implementation (like Hibernate) |
| Implementation | Does not have one — defines API contracts | Provides actual implementation of JPA | Uses underlying JPA providers like Hibernate |
| Boilerplate Code | Developer has to write a lot of boilerplate | Less than JDBC but still more than Spring Data JPA | Greatly reduces code with built-in CRUD methods |
| Transaction Management | Not defined (left to implementation) | Manual or declarative via annotations | Fully managed using @Transactional |

**Code Demonstration:**

**A. Hibernate Implementation (Without Spring Data JPA)**

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;

}

**B. Spring Data JPA Implementation (With Abstraction)**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

**Summary:**

* JPA is just a specification, Hibernate implements it.
* Spring Data JPA builds on Hibernate to reduce boilerplate.
* Demonstrated code difference: Hibernate requires explicit session and transaction management.
* Spring Data JPA uses simple repository and service layers with annotations to persist data.