**HANDS-ON 4: DIFFERENCE BETWEEN JPA, HIBERNATE, AND SPRING DATA JPA**

# Objective

To understand the differences between JPA, Hibernate, and Spring Data JPA, and how they are used in real-world Java applications with examples.

# Key Concepts

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification (JSR 338) | JPA Implementation | Abstraction layer over JPA |
| Responsibility | Defines standard for persistence | Implements persistence with ORM features | Reduces boilerplate using Repository interfaces |
| Dependency | javax.persistence | org.hibernate | org.springframework.data.jpa |
| Transaction Handling | Manual | Manual | Automatic with @Transactional |
| Querying | JPQL | HQL + native SQL | Derived Queries, JPQL, native SQL |
| Code Complexity | Medium | High | Low |

# Hibernate Approach

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 Approach

EmployeeRepository.java  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

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