**SUPERSET ID: 6408269**

**TOPIC : SPRING DATA JPA WITH SPRINGBOOT , HIBERNATE**

**EXERCISE : DIFFERENCE BETWEEN JPA , HIBERNATE AND SPRING DATA JPA**

**Java Persistence API (JPA)**

* It is a specification (JSR 338) for accessing, persisting, and managing data between Java objects and a relational database.
* JPA does not provide an actual implementation.
* It defines interfaces like EntityManager, PersistenceContext, and annotations like @Entity, @Table, @Id, etc.
* Popular JPA implementations: Hibernate, EclipseLink, OpenJPA.

**Hibernate**

* Hibernate is an Object-Relational Mapping (ORM) framework.
* It is a concrete implementation of JPA.
* It includes additional features on top of JPA (e.g., caching, dirty checking, lazy loading).
* With Hibernate, developers need to manage SessionFactory, transactions, and session lifecycles manually unless integrated into frameworks like Spring.

**Spring Data JPA**

* Spring Data JPA is not an implementation of JPA, but an abstraction over it.
* It significantly reduces boilerplate code for common database operations (CRUD).
* It provides Repository interfaces like JpaRepository, CrudRepository, etc.
* Handles transactions, session management, and query creation (even from method names).
* Built on top of JPA and works with Hibernate or any JPA implementation under the hood.

**CODE SNIPPETS FOR COMPARISON :**

**Hibernate Code 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 Code Example:**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**CODE SNIPPET FOR JPA**

import jakarta.persistence.EntityManager;

import jakarta.persistence.EntityManagerFactory;

import jakarta.persistence.Persistence;

import com.example.model.Employee;

public class EmployeeDao {

private static EntityManagerFactory emf =

Persistence.createEntityManagerFactory("my-persistence-unit");

public void addEmployee(Employee employee) {

EntityManager em = emf.createEntityManager();

try {

em.getTransaction().begin();

em.persist(employee);

em.getTransaction().commit();

} catch (Exception e) {

em.getTransaction().rollback();

e.printStackTrace();

} finally {

em.close();

}

}

}