Solution of Hands-on 4: Difference Between JPA, Hibernate, and Spring Data JPA

# 1. Java Persistence API (JPA)

- JPA is a Java specification used for managing relational data in Java applications.

- It defines a set of rules and APIs, but does not provide any implementation.

- Common concepts in JPA include Entity, EntityManager, and annotations like @Entity, @Id, etc.

- JPA needs an implementation like Hibernate to function.

# 2. Hibernate

- Hibernate is a widely used ORM (Object-Relational Mapping) framework that implements JPA.

- It provides the actual code to map Java objects to database tables and manage operations.

- Hibernate includes JPA features and its own extra capabilities, such as caching and query optimizations.

- Developers have to manage sessions and transactions manually when not using Spring.

# 3. Spring Data JPA

- Spring Data JPA is part of the Spring ecosystem, built on top of JPA.

- It does not implement JPA itself, but simplifies JPA usage through abstraction.

- It allows you to define repository interfaces, avoiding boilerplate DAO code.

- Integrates seamlessly with Spring Boot and handles transactions automatically.

# Code Comparison

## Using Hibernate

Manual session handling, transaction management, and explicit save operation:

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 (Exception e) {  
 if (tx != null) tx.rollback();  
 } finally {  
 session.close();  
 }  
  
 return employeeID;  
}

## Using Spring Data JPA

Only interfaces and minimal code are required:

EmployeeRepository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
 // Spring generates implementation automatically  
}

EmployeeService.java

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