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

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

**Java Persistence API (JPA)**

* JPA is a **Java Specification** (JSR 338) for persisting, reading, and managing data between Java objects and relational databases.
* It is just a **specification**, meaning it does not provide a concrete implementation.
* JPA defines a standard set of annotations and interfaces for object-relational mapping.
* **Hibernate** is one of the most popular implementations of the JPA specification.

**Hibernate**

* Hibernate is an **Object-Relational Mapping (ORM) tool** and a **concrete implementation** of the JPA specification.
* It provides additional features beyond JPA such as:
  + Lazy loading
  + Caching
  + Better performance tuning
* Developers using Hibernate can either use **JPA annotations** or **Hibernate-specific APIs**.

**Spring Data JPA**

* Spring Data JPA is **not a JPA implementation**.
* It is a **Spring framework module** that provides an **abstraction over JPA/Hibernate**, aimed at reducing boilerplate code for repository layers.
* Built on top of JPA providers (like Hibernate), it:
  + Automatically implements basic CRUD operations
  + Generates queries from method names
  + Manages transactions and data access logic

**Comparison via Code Snippets**

**Hibernate Approach**

/\* Method to CREATE an employee in the database \*/

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);

}

**Summary of Differences**

| **Aspect** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (JSR 338) | Implementation of JPA | Abstraction layer over JPA/Hibernate |
| Provides Implementation? | No | Yes | No |
| API Usage | EntityManager | Session, Transaction | JpaRepository, @Repository |
| Boilerplate Code | Medium | High | Very Low (handled by Spring) |
| Vendor | Oracle (spec), Jakarta EE | Red Hat | Spring Framework |
| Custom Query Support | Via JPQL | Via HQL or native SQL | Method names or @Query annotation |
| Integration | Needs manual config | Needs manual config | Easily integrates with Spring Boot |