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

**JPA:**

* A standard Java specification for ORM (Object-Relational Mapping).
* Lets you map Java classes (entities) to database tables.
* Common implementation: Hibernate.

Spring Data JPA:

* It simplifies working with databases using JPA (Java Persistence API).
* Instead of writing a lot of boilerplate code (like queries and entity management), Spring Data JPA lets you focus on your domain model and business logic.

| **Technology** | **What is it?** | **Role** |
| --- | --- | --- |
| **JPA** | A Java standard specification for ORM. | Defines *how* to map Java objects to database tables. |
| **Hibernate** | A popular implementation of JPA (and also offers extras). | Does the actual ORM work following JPA rules. |
| **Spring Data JPA** | A Spring library that builds on top of JPA. | Makes JPA easier by auto-generating repositories, queries. |

**Hibernate Example Code**

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;

}

🡺 This is **manual transaction management**

**Spring Data JPA Equivalent Code**

**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);

}

}

🡺No manual transaction/Session—Spring handles it.