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

**1. Java Persistence API (JPA)**

* **JPA** is a Java specification (JSR 338) that defines a standard for **object-relational mapping (ORM)** and data persistence in Java applications.
* It provides a **standard API** to manage relational data using Java objects.
* **JPA itself is not an implementation** — it’s just a specification.
* Popular implementations of JPA include:
  + **Hibernate**
  + EclipseLink
  + OpenJPA

**2. Hibernate**

* **Hibernate** is a **widely used ORM tool** and the most popular **implementation of JPA**.
* It provides additional features **beyond the JPA specification**, such as:
  + Caching
  + Interceptors
  + Lazy loading
  + Enhanced performance tuning
* In traditional Hibernate, developers manually handle:
  + Sessions
  + Transactions
  + Exception handling

**3. Spring Data JPA**

* **Spring Data JPA** is a **Spring-based abstraction** over JPA implementations like Hibernate.
* It **does not implement JPA** directly, but **simplifies interaction with the database** using repositories.
* Benefits:
  + **No need to write boilerplate DAO code**
  + Provides **ready-to-use CRUD operations**
  + Integrates easily with Spring Boot for auto-configuration
  + Automatically handles:
    - Transactions
    - Query generation
    - Exception translation

**4. Code Comparison**

**➤ Hibernate Example: Manually Saving an Employee**

java

CopyEdit

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;

}

**Explanation**:

* Manages session and transaction manually.
* Requires try-catch-finally boilerplate.
* Developer is responsible for transaction safety.

**➤ Spring Data JPA Example: Saving an Employee**

**📁 EmployeeRepository.java**

java

CopyEdit

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**📁 EmployeeService.java**

java

CopyEdit

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**Explanation**:

* No session/transaction handling code.
* JpaRepository provides built-in methods like save(), findAll(), deleteById(), etc.
* Annotated with @Transactional to handle transaction boundaries.
* Much cleaner and maintainable