**WEEK3 HANDSON**

**Exercise 2: Difference between JPA, Hibernate, and Spring Data JPA**

**What is JPA?**

* **JPA (Java Persistence API)** is a **specification** (interface/blueprint).
* It provides a **standard way** to map Java objects to relational databases.
* Defined by **JSR 338**.
* Does **not provide an implementation**.

**Example:**

@Entity

public class Country {

@Id

private String code;

}

You use JPA annotations like @Entity, @Id, @Column, but they don't do anything unless an implementation is provided (like Hibernate).

**What is Hibernate?**

* Hibernate is the **most popular implementation of JPA**.
* It's an **ORM (Object Relational Mapping) tool**.
* Offers **advanced features** like caching, batch processing, lazy loading, etc.
* You can use Hibernate directly or via JPA.

**Hibernate adds**:

* SessionFactory, Session, Transaction
* Hibernate-specific annotations and methods

**What is Spring Data JPA?**

* Spring Data JPA is a **wrapper/abstraction over JPA + Hibernate**.
* Helps reduce **boilerplate code** for common operations like findAll(), save(), delete().
* Automatically creates **repository implementations** from interfaces.
* Built on top of Spring framework + JPA.

**⚖️ Comparison Table:**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification | Implementation (ORM) | Abstraction (Framework) |
| Implements JPA? | No | Yes | Uses JPA + Hibernate internally |
| Needs Boilerplate Code? | Yes | Yes | No – auto-generated methods |
| Provided by | Java EE / Jakarta EE | Red Hat | Spring Framework |
| Example Usage | @Entity, EntityManager | Session, Transaction | JpaRepository, @Repository |

**Code Comparison**

**Using Hibernate (manual code):**

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

session.save(employee);

tx.commit();

session.close();

**Using Spring Data JPA:**

@Autowired

private EmployeeRepository repo;

repo.save(employee); // That’s it!