**WEEK-3 Spring Data JPA HandsOn**

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

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

* **Definition**: JPA is a **specification** provided by Java to define standard ways to manage relational data in Java applications.
* **Goal**: Decouple database operations from actual implementation (ORM tool can vary).
* **It only provides interfaces**, not implementations.

Think of JPA as a **contract or interface** that ORM tools like Hibernate implement.

**Example (JPA Annotations):**

import jakarta.persistence.\*;

@Entity

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

}

**2. Hibernate**

* **Definition**: Hibernate is an **ORM (Object Relational Mapping) implementation** of the JPA specification.
* **It is a library** that provides the actual functionality like writing SQL, managing sessions, caching, etc.
* Supports both JPA annotations and its own custom features (@CreationTimestamp, @LazyCollection, etc.).

**Example (Hibernate specific):**

@CreationTimestamp

@Column(updatable = false)

private LocalDateTime createdAt;

* **JPA can work with other providers**, but Hibernate is the most popular and commonly used.

**3. Spring Data JPA**

* **Definition**: A part of the Spring ecosystem that builds on JPA, offering **abstractions and automation**.
* It removes the need to write boilerplate code (like EntityManager, JPQL).
* Provides built-in methods like findAll(), findById(), save().

**Example (Spring Data JPA Repository):**

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String name);

}

* Behind the scenes, Spring Data JPA uses Hibernate as the JPA provider.

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (interface) | Implementation (library) | Framework (built on JPA & Hibernate) |
| Provided by | Java EE / Jakarta EE | Red Hat | Spring Framework |
| Contains | Only interfaces & annotations | Complete ORM engine | Abstractions over JPA + Hibernate |
| Needs implementation | Yes | Implements JPA | Uses Hibernate under the hood |
| Example usage | @Entity, @Id | @CreationTimestamp, Session | JpaRepository, @Repository, save() |
| Boilerplate | High | Medium | Low |

**Practical Example**

**Step 1: JPA Entity**

import jakarta.persistence.\*;

@Entity

public class Country {

@Id

private String code;

private String name;}

**Step 2: Spring Data JPA Repository**

import org.springframework.data.jpa.repository.JpaRepository;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String name);

}

**Step 3: Service using Repository**

@Service

public class CountryService {

@Autowired

private CountryRepository repository;

public List<Country> getAllCountries() {

return repository.findAll();

}

}