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

**What is JPA?**

* **JPA (Java Persistence API)** is a **specification** (JSR 338) provided by Oracle as part of Java EE (now Jakarta EE).
* It defines a **standard interface** for ORM (Object-Relational Mapping) in Java — i.e., how to map Java classes to relational database tables.

**Key Points:**

* JPA **does not have any implementation**.
* It just provides interfaces (EntityManager, Query, etc.) and annotations (@Entity, @Id, @OneToMany, etc.).
* You need a **provider** (like Hibernate) to use JPA.

**Example:**

@Entity

public class Book {

@Id

private Long id;

private String title;

}

**2. Hibernate**

**What is Hibernate?**

* **Hibernate** is an **Object Relational Mapping (ORM) tool** and the **most popular implementation of JPA**.
* It also provides **additional features** not included in JPA.

**Key Features:**

* Implements the JPA interfaces.
* Offers **extra functionalities** like:
  + Second-level caching
  + Better fetching strategies
  + Native Hibernate Query Language (HQL)
  + Batch processing
* Can be used in **two ways**:
  + As a **JPA provider** (standard)
  + With **native Hibernate APIs** (more powerful)

**Example (with Hibernate API):**

Session session = sessionFactory.openSession();

Book book = session.get(Book.class, 1L);

**3. Spring Data JPA**

**What is Spring Data JPA?**

* **Spring Data JPA** is a **Spring framework module** that sits **on top of JPA and Hibernate**.
* It provides a **higher-level abstraction** to remove boilerplate code for repositories and queries.

**Key Features:**

* Uses **Hibernate internally** (or any other JPA provider).
* Automatically generates DAO implementations (no need to write them).
* Allows **query method names** like findByTitle().
* Supports JPQL, native queries, and pagination.

**Example:**

public interface BookRepository extends JpaRepository<Book, Long> {

List<Book> findByTitle(String title);

}