JPA (Java Persistence API)

- **What it is**: A **Java specification** (JSR 338) for accessing, persisting, and managing data between Java objects and relational databases.
- **Key point**: It is **just an interface**, not an implementation.
- **Purpose**: Standardizes how Java applications interact with databases using ORM (Object Relational Mapping).

Think of JPA like a set of rules or guidelines for ORM.

---

2. Hibernate

- **What it is**: A **concrete implementation** of the JPA specification.
- **Type**: ORM (Object-Relational Mapping) framework.
- **Features**:
  - Lazy/eager loading
  - HQL (Hibernate Query Language)
  - Caching
  - Schema generation

If JPA is a rulebook, Hibernate is one of the tools that follows it.

---

3. Spring Data JPA

- **What it is**: A **Spring project** that builds on top of JPA and Hibernate.
- **Purpose**: Reduces boilerplate code required to use JPA (e.g., @Repository + interfaces instead of custom DAO classes).
- **Key Benefit**:
  - You can just extend JpaRepository or CrudRepository, and Spring generates the implementations automatically.

Think of Spring Data JPA as a smart helper that wraps Hibernate and simplifies JPA usage.

Summary Table

Feature	**JPA**	**Hibernate**	**Spring Data JPA**
Type	Specification (interface)	Implementation (framework)	Abstraction over JPA + Hibernate
Implementation	None	Yes	Uses JPA provider (e.g., Hibernate)
Purpose	Standard API for ORM	Full-featured ORM tool	Reduce boilerplate with repositories
Requires boilerplate	Yes	Yes	No (Spring handles it)