Difference between JPA, Hibernate and Spring Data JPA

Aspect	JPA	Hibernate	Spring Data JPA
Definition	Java specification for data persistence	ORM framework implementing JPA	Spring module that simplifies JPA usage
Type	Specification (Interface)	Implementation (Library)	Framework abstraction over JPA
Provided By	Oracle (Java EE)	Red Hat	Spring (Pivotal/VMware)
Boilerplate Code	Requires manual implementation	Still needs some code	Minimizes code with interfaces
Configuration	Annotation-based	Annotation & XML	Spring-based, auto-config
Query Support	JPQL	JPQL + HQL + Native SQL	JPQL + Derived + Native Queries
Learning Curve	Moderate	High (more APIs)	Low (declarative style)
Repository	Not available	SessionFactory used	CrudRepository, JpaRepository, etc.

JPA

- **JPA (Java Persistence API)** is just a **specification**, like a blueprint.
- It defines **how** Java objects should be mapped to database tables.
- It does **not do anything by itself** — it needs a tool to actually "implement" it.

Hibernate

Hibernate is the **most popular implementation of JPA**. It takes care of all the actual database interactions. It provides many extra features that JPA doesn't, like:

- Caching
- Lazy/eager loading
- Dirty checking
- HQL (Hibernate Query Language)

 Hibernate can be used **with or without JPA**. But in modern applications, it's usually used **as the engine behind JPA**.

Spring Data JPA

Spring Data JPA is built on top of JPA and Hibernate. It makes using JPA + Hibernate much easier inside Spring applications. We don't have to write boilerplate code — it gives:

- Auto-generated queries (like `findByName()`)
- Built-in CRUD operations
- Custom query support using annotations

It lets us build a working database-backed app with **just an interface** — no implementation required.