# Difference between JPA, Hibernate and Spring Data JPA

**JPA (Java Persistence API)**

JPA is **just a specification**—a set of rules or a standard for persisting Java objects to relational databases. It doesn't do anything by itself but defines how ORM (Object Relational Mapping) should work.

#### Real-life analogy:

Imagine JPA as a recipe. You need a cook to actually follow it and make the food. That cook is Hibernate.

**Hibernate**

Hibernate is an **implementation of the JPA specification**. It provides the actual code and features that follow JPA rules and adds some extra capabilities on top.

#### Real-life analogy:

Hibernate is like the chef that uses JPA’s recipe and adds some of their own flair to make the dish even better.

**Spring Data JPA**

Spring Data JPA is a **framework built on top of JPA**. It adds an **abstraction layer** that simplifies writing database access code (repositories) by **generating queries automatically** based on method names

#### Real-life analogy:

If JPA is the recipe and Hibernate is the chef, then Spring Data JPA is a smart kitchen assistant who prepares the ingredients, cleans up, and sometimes even finishes the dish for you.

**Difference:**

| **Aspect** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **What it is** | Just a set of rules for ORM | A tool that follows JPA rules and adds features | A helper on top of JPA that reduces boilerplate code |
| **Does it work alone?** | No, it's only a specification | Yes, it can be used directly | No, it needs a JPA provider like Hibernate |
| **Code Writing** | You write a lot of code | You still write queries and configs | Very little code – queries can be auto-generated |
| **Who uses it?** | Developers who want to follow standards | Devs who want more power and flexibility | Devs who want things to “just work” with minimum effort |
| **Best Use Case** | For setting a standard across tools | When you want full control over ORM | When you want quick and easy data access with less configuration |
| **Example in use** | Annotating classes with @Entity | Managing sessions, fetching/storing data | Writing just an interface like findByEmail()—no query needed |

.