**🔹 Difference between JPA, Hibernate, and Spring Data JPA**

**JPA (Java Persistence API):**  
JPA is a **Java specification** (JSR 338) that defines a standard for **object-relational mapping (ORM)** in Java. It provides a set of annotations and interfaces to map Java objects to relational database tables. However, JPA is just a specification — it does not provide a working implementation by itself.

**Hibernate:**  
Hibernate is a **popular implementation of the JPA specification**. It is an ORM tool that handles the actual interaction with the database. Hibernate implements all the interfaces and annotations defined by JPA, and also adds extra features beyond the specification. You can use Hibernate directly (with its own API) or via JPA.

**Spring Data JPA:**  
Spring Data JPA is a part of the larger Spring ecosystem. It builds on top of JPA and Hibernate to **simplify database access and reduce boilerplate code**. With Spring Data JPA, you don't need to write implementation code for common operations like saving, updating, deleting, or finding entities — it provides **prebuilt repository interfaces** and powerful method name conventions to generate queries automatically.

**🔸 Summary:**

| **Concept** | **Type** | **Role** |
| --- | --- | --- |
| **JPA** | Specification | Defines how Java objects are mapped to DB tables |
| **Hibernate** | Implementation | A concrete ORM tool that implements JPA |
| **Spring Data JPA** | Abstraction Layer | Builds on top of JPA/Hibernate to simplify data access in Spring |

**🔗 References:**

* **Difference between Spring Data JPA and Hibernate:**  
  <https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1>
* **Introduction to JPA:**  
  <https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>