**1.Spring Data JPA - Quick Example** 

A screenshot of a computer program

AI-generated content may be incorrect.

**4. Difference between JPA, Hibernate and Spring Data JPA**

**Java Persistence API (JPA)**

* JPA is basically a Java specification (JSR 338) for persisting, reading, and managing data from Java objects.
* It defines *what* should happen in ORM (Object Relational Mapping), but does not contain its own implementation.
* You can think of JPA as a *contract* which frameworks like Hibernate must follow.
* The actual implementation is done by frameworks such as Hibernate or EclipseLink.

**Hibernate**

* Hibernate is a popular ORM tool that implements the JPA specification.
* It helps map Java classes to relational database tables.
* Hibernate provides additional features such as caching, fetching strategies, lazy/eager loading, etc.
* Before Spring Data JPA, developers often wrote a lot of boilerplate code to handle sessions and transactions.

**Spring Data JPA**

* Spring Data JPA is not a JPA implementation, but a layer on top of it (generally on top of Hibernate).
* Its main purpose is to reduce boilerplate code by handling common CRUD operations automatically.
* It makes working with JPA simpler by providing ready-to-use repository interfaces.
* Spring Data JPA also manages transactions behind the scenes.

**Comparison Table**

| **Aspect** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| What is it? | Specification (JSR 338) | Implementation of JPA | Abstraction on top of JPA/Hibernate |
| Purpose | Defines contract for ORM | Provides ORM features | Reduces boilerplate, simplifies data access |
| Provides | Interfaces / Annotations | Concrete classes implementing JPA | Repository support, auto query generation |
| Example Framework | — | Hibernate | Uses Hibernate under the hood |
| Code Amount | Developer writes more code | Less than JDBC, but still manual code | Minimal — only method signatures required |