1. Spring Data JPA

* Spring Data JPA is a part of the larger Spring Data project that simplifies the use of the Java Persistence API(JPA) for database access. It allows developers to work with databases using plain Java objects and interfaces, removing the need to write complex SQL or boilerplate DAO code. Instead of writing implementation logic for CRUD operation, Spring Data JPA automaticall generates them at runtime based on the method names defined in the repository interfaces.

1. Difference between JPA, Hibernate and Spring Data JPA

|  |  |  |
| --- | --- | --- |
| **JPA(Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| Defines the working of the ORM | Implements JPA and adds etra features | Simplifies JPA by reducing boilerplate code |
| It is only a specification, not an implementation | It is a complete ORM framework and JPA provider | It is an abstraction layer over JPA and a part of the Spring Data project |
| It needs and ORM provider like Hibernate | It is itself a complete implementation | It needs a JPA provider(usually Hibernate underneath) |
| Requires developers to write full repository | Reduces some boilerplate but still requires manual coding for queries | Minimizes boilerplate by generating code for common CRUD query methods |
| Less convenient due to the need for manual setup and coding | Easier than JPA alone, but still required understanding of ORM concepts | Easiest to use with SpringBoot integrates seamlessly with Spring’s ecosystem |