1. JPA (Java Persistence API) – Specification

- **Description**: A standard interface/specification defined by Java EE (now Jakarta EE).
- **Why we use?**: Defines how Java objects map to relational databases and how to manage them.
- **It Contains**: Interfaces like EntityManager, annotations like @Entity, @Id, @OneToMany, etc.
- **implementation**: JPA doesn't provide any implementation, only guidelines.

2. Hibernate – JPA Implementation + More

- **Description**: A popular JPA provider (i.e., implements the JPA specification).
- **Additional**: Provides features beyond JPA, like:
- **It Contains**: @Filter, @BatchSize, native SQL support, better caching
- Enhanced lazy loading
- **Implementation**: You can use Hibernate-specific APIs (e.g., Session) for more control.

Hibernate implements JPA, so you can use it with pure JPA or access Hibernate-specific features if needed.

3. Spring Data JPA – Abstraction Layer on Top of JPA

- **Description**: A Spring module that simplifies JPA-based data access.
- **Why we use?**: Eliminates boilerplate code — you don't have to write your own DAO classes.
- **Features**:
    - Auto-implementing repositories (e.g., CrudRepository, JpaRepository)
    - Query derivation from method names (findByUsername, deleteByEmail)
    - Integration with Spring Boot, Transactions, AOP, etc.

Spring Data JPA is not a JPA implementation, but a tool that makes using JPA easier and more declarative.

Hibernate vs Spring Data JPA?:

Hibernate, a widely-used Java framework, implements the Java Persistence API (JPA). It serves as an Object-Relational Mapping (ORM) tool, enabling you to map Java classes to database tables and manage database operations using Java objects. Hibernate offers robust features such as caching, lazy loading, and HQL (Hibernate Query Language). When utilizing Hibernate, you typically create Data Access Objects (DAOs) manually and manage queries using JPQL, HQL, or native SQL. While Hibernate provides complete control over the persistence layer, it also necessitates writing a significant amount of boilerplate code.

Spring Data JPA, unlike an ORM tool or JPA provider, is a Spring Framework module that sits atop JPA. It simplifies the process of building the data access layer in Spring applications by reducing boilerplate code. By defining repository interfaces, Spring Data JPA automatically generates the implementation. For instance, instead of manually writing queries or DAO classes, you can simply declare a method like findByUsername, and Spring Data JPA will handle the query generation behind the scenes.

A major advantage of Spring Data JPA is that it works with any JPA provider, not just Hibernate. So while Hibernate is often used underneath, you can also plug in EclipseLink or other providers. Another benefit is that it integrates naturally with other Spring features like dependency injection and declarative transaction management using the @Transactional annotation.

Hibernate: An engine that implements JPA and handles object-relational mapping (ORM).
Spring Data JPA: A helper framework that sits on top of JPA

Finding a Country by Code Using Spring Data JPA

Entity: Country.java

```
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
@Entity
public class Country {
    @Id
    private String code; // For example: "IN", "US", "FR"
    private String name;
    // Getters and setters
}
```

Repository: CountryRepository.java

```
import org.springframework.data.jpa.repository.JpaRepository;
import java.util.Optional;

public interface CountryRepository extends JpaRepository<Country, String> {
    Optional<Country> findByCode(String code); // Auto-generated query!
}
```

Usage:

```
@Service
public class CountryService {
    @Autowired
    private CountryRepository countryRepository;

    public Country getCountryByCode(String code) {
        return countryRepository.findByCode(code)
                .orElseThrow(() -> new RuntimeException("Country not found"));
```

}

JpaRepository<Country, String>: Because code is the primary key (of type String)

findByCode(String code): Spring will generate the query:

SELECT * FROM country WHERE code = ?;