Difference between JPA, Hibernate and Spring data JPA:

Hibernate, JPA, and Spring Data JPA are all frameworks used for interacting with relational databases in Java applications.

1. **JPA (Java Persistence API)**:
   - JPA is a specification for object-relational mapping (ORM) in Java. It defines a set of interfaces and annotations that standardize how Java objects are mapped to relational databases and how persistence operations are performed.
   - JPA itself does not provide an implementation; it's a blueprint that ORM frameworks can follow.
2. **Hibernate**:
   - Hibernate is a popular implementation of the JPA specification. It's a full-featured ORM framework that provides the actual code and mechanisms to manage the mapping between Java objects and database tables, handle data persistence, and execute queries.
   - While Hibernate fully implements JPA, it also offers its own set of proprietary features and APIs that extend beyond the JPA specification.
3. **Spring Data JPA**:
   - Spring Data JPA is a higher-level abstraction built on top of JPA (and thus, typically, a JPA implementation like Hibernate) within the Spring Framework.
   - It aims to reduce the amount of boilerplate code required for implementing data access layers by providing convenient interfaces like JpaRepository and enabling query generation from method names (e.g., findByFirstNameAndLastName).
   - Spring Data JPA simplifies common CRUD operations and query creation, allowing developers to focus more on business logic rather than repetitive data access code. It leverages a JPA provider (like Hibernate) behind the scenes to perform the actual persistence operations.

What is difference between Hibernate, JPA, and Spring Data JPA?

Now that we have fair idea of what is Hibernate, JPA, and Spring Data JPA and their abilities, its time to see the real difference between them. Here’s the comparison of Hibernate, JPA, and Spring Data JPA in a feature-wise list format:

1. Definition

Hibernate is Robust ORM framework providing object-relational mapping capabilities. While JPA is a Specification for ORM in Java, defining a set of standard APIs for persistence and Spring Data JPA provide simplified abstraction over JPA, providing additional features and repository abstractions.

2. Implementation

Hibernate provides its own implementation of the JPA specification while JPA requires an underlying implementation to be used, such as Hibernate, EclipseLink, or OpenJPA. and Spring Data JPA builds on top of JPA and requires a JPA-compliant implementation, such as Hibernate or EclipseLink.

3. Persistence API

Again, JPA defines a set of standard APIs for ORM in Java while Hibernate provides its own API for persistence operations and Spring Data JPA builds on JPA APIs and adds additional functionality, such as repository abstractions and query support.

4. Database Support

In case of JPA database support depends on the JPA implementation used, which may have specific database support. Hibernate does supports various databases through its dialects while Spring Data JPA also depends on the JPA implementation used, offering compatibility with different databases.

5. Transaction Management

Hibernate comes with its own transaction management capabilities, while JPA and Spring Data JPA both depends on the JPA implementation used for transaction management.

6. Query Language

Hibernate offers Hibernate Query Language (HQL) for writing object-oriented queries. While JPA defines Java Persistence Query Language (JPQL) for database queries and Spring Data JPA uses JPQL as the query language, similar to JPA.

7. Caching

Hibernate provides first-level and second-level caching mechanisms. While JPA again depends on the JPA implementation used, which may offer caching capabilities and Spring Data JPA depends on the JPA implementation used for caching support.

8. Configuration

Hibernate supports configuration using XML, annotations, or Java-based approaches. While JPA also supports configuration using XML, annotations, or Java-based approaches and Spring Data JPA supports configuration using XML, annotations, or Java-based approaches.

9. Integration

Hibernate can be used independently or integrated with Spring. JPA works with any JPA-compliant implementation, including integration with Spring and Spring Data JPA is part of the Spring Data family, designed to integrate with Spring applications.

Code Usage Examples:

1. Hibernate:

```
/* Method to CREATE an employee in the database */
  public Integer addEmployee(Employee employee){
    Session session = factory.openSession();
    Transaction tx = null;
    Integer employeeID = null;

    try {
      tx = session.beginTransaction();
      employeeID = (Integer) session.save(employee);
```

```
        tx.commit();
    } catch (HibernateException e) {
        if (tx != null) tx.rollback();
        e.printStackTrace();
```

2. Spring data JPA

EmployeeRespository.java

```
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}
```

EmployeeService.java

```
    @Autowire
    private EmployeeRepository employeeRepository;

    @Transactional
    public void addEmployee(Employee employee) {
        employeeRepository.save(employee);
    }
```