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

-No execution or program

-Only theory about their differences

**The differences between JPA, Hibernate, and Spring Data JPA can be understood by recognizing their roles in the Java persistence ecosystem:**

* **JPA (Java Persistence API):**
* Nature: JPA is a specification for object-relational mapping (ORM) in Java. It defines a standard set of interfaces and annotations for managing relational data with Java objects.
* Purpose: It provides a common API for persistence, allowing developers to write persistence logic that is independent of the underlying ORM implementation.
* Key Components: EntityManager, EntityManagerFactory, @Entity, @Table, @Column, etc.
* **Hibernate:**
* Nature: Hibernate is a popular implementation of the JPA specification. It is a full-fledged ORM framework that provides concrete classes and features to realize the concepts defined in JPA.
* Purpose: It maps Java objects to database tables and offers mechanisms for data persistence, retrieval, and manipulation.
* Key Features: Beyond basic JPA implementation, Hibernate offers its own features like caching, lazy loading, and HQL (Hibernate Query Language).
* **Spring Data JPA:**
* Nature: Spring Data JPA is a module within the Spring Framework that provides a higher-level abstraction for working with JPA. It is not a JPA provider itself, but rather a library that simplifies the development of data access layers using JPA.
* Purpose: It significantly reduces the amount of boilerplate code required for common data access operations by providing repository interfaces and automatically generating queries based on method names.
* Relationship: Spring Data JPA works on top of a JPA provider like Hibernate (or EclipseLink, etc.) to offer a more convenient and opinionated way to interact with the persistence layer. It leverages JPA underneath to perform the actual database operations.

**Code comparison**

**Hibernate Style**

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();

} finally {

session.close();

}

return employeeID;

}

**Spring Data JPA Style**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}