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

**Java Persistence API (JPA)**

* JPA stands for **Java Persistence API**.
* It is a specification (JSR 338) for persisting, reading, and managing data between Java objects and relational databases.
* It provides a standard interface but **does not include a concrete implementation**.
* JPA is only a guideline; actual implementations (like Hibernate) are needed to use it.

**Hibernate**

* Hibernate is an **Object Relational Mapping (ORM) tool**.
* It is one of the most popular implementations of the JPA specification.
* It provides powerful features for mapping Java objects to database tables.
* Requires writing more boilerplate code (session, transaction management, etc.).

**Spring Data JPA**

* Spring Data JPA is **not an implementation of JPA**, but rather a **framework built on top of JPA** (usually using Hibernate underneath).
* It helps reduce boilerplate code such as DAO (Data Access Object) implementation.
* It provides built-in methods for CRUD operations and manages transactions using annotations like @Transactional.
* It works with any JPA provider (Hibernate is the default).

**Comparison Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification | ORM tool + JPA implementation | Framework built over JPA |
| Implementation | No | Yes (e.g., Hibernate) | No (relies on JPA provider like Hibernate) |
| Boilerplate Code | Moderate | More code needed | Very less code, uses Spring annotations |
| Transaction Management | Requires manual setup | Needs manual handling | Managed by Spring using @Transactional |
| Abstraction Level | Base level | Implements JPA | Higher-level abstraction |

**Code Comparison**

**Hibernate**

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**

**EmployeeRepository.java**

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

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}