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

**JPA:**

* It is a **specification** (defined in JSR 338) for ORM (Object Relational Mapping) in Java.
* It’s **Purpose** is to provides standard APIs for managing, persisting, and retrieving data between Java objects and relational databases.
* It is **just a set of interfaces and annotations**,and it **doesn't provide implementation**.
* **Examples of Implementations**: Hibernate, EclipseLink,OpenJPA.

---------------------------------------------------------------------------------

**Hibernate**

* **It is a** **concrete ORM framework** that implements the **JPA specification**.
* **It Provides**:
  + Actual implementation of persistence logic.
  + Additional features beyond JPA (e.g., HQL, caching, batch fetching).
* It can be used with or without JPA. When used with JPA, it's called a **JPA provider**.

**Hibernate Code Example**:

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**

* **It is** a **Spring project** that builds **on top of JPA**.
* **It does not provide its own JPA implementation**.
* It adds an **abstraction layer** that simplifies JPA development by:

1:Reducing boilerplate code (like writing DAO classes manually).

2:Providing powerful **query generation** from method names.

* + 3:Managing transactions, repositories, pagination, and auditing automatically.
* **It works with** Hibernate or any other JPA provider.

**Spring Data JPA Code Example**:

**Repository Interface:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**Service Class:**

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

-----------------------------------------------------------------------------------

**Difference between JPA, Hibernate, Spring Data JPA (Summary)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **JPA** | **Hibernate** | **Spring Data**  **JPA** |
| **Type** | A specification  Of ORM in JAVA | Implementation  Of JPA | |  | | --- | | Abstraction  layer over  JPA |  |  | | --- | |  | |
| **Implementation** | It does not  provide | It provides | It does not  provide, it  Relies on JPA provider like Hibernate |
| |  | | --- | | **Boilerplate**  **code** |  |  | | --- | |  | | Moderate | High | Minimal |
| **Transaction Management** | Manual or  container-  managed | Manual | Automatic |
| **Part of Spring Framework** | NO | NO | YES |