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

**1. Java Persistence API (JPA)**

* JPA (Java Persistence API) is a specification (JSR 338) for object-relational mapping (ORM) in Java.
* It allows mapping Java objects to database tables in a standardized way.
* **Key Features:**

1. Defines interfaces and annotations for ORM.
2. No implementation; needs a provider like Hibernate.
3. Helps manage data persistence, transactions, and queries.
4. Standardized approach to database access in Java EE and Spring.

* **Common Annotations:**

@Entity

@Id

@Table

@Column

@GeneratedValue

@OneToMany

@ManyToOne, etc.

**2. Hibernate**

* Hibernate is a JPA provider and an ORM tool that offers the actual implementation for JPA interfaces.
* It is widely used and includes additional features beyond the JPA standard.
* **Key Features:**

1. Implements JPA.
2. Offers extra features like caching, lazy loading, custom SQL support.
3. Provides both JPA and its own native APIs.

* **Example Code using 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();

      } finally {

         session.close();

      }

      return employeeID;

   }

**3. Spring Data JPA**

* Spring Data JPA is a part of the Spring ecosystem that simplifies JPA usage by providing a higher level of abstraction over JPA provider implementations like Hibernate.
* **Key Features:**

1. No implementation of JPA itself.
2. Reduces boilerplate code using repository interfaces.
3. Automatically handles common CRUD operations.
4. Integrates seamlessly with Spring Boot.

* **Example Code using Spring Data JPA:**

***EmployeeRespository.java***

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

***EmployeeService.java***

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

* **Comparison:**

|  |  |  |  |
| --- | --- | --- | --- |
| **FEATURE** | **JPA** | **HIBERNATE** | **SPRING DATA JPA** |
| **Type** | Specification | Implementation | Abstraction over JPA and Hibernate |
| **Implementation** | No | Yes | No |
| **Boilerplate Code** | Medium | High | Very low |
| **Ease of Use** | Medium | Medium | High |
| **Transactions** | Manual setup | Manual setup | Automatic  (@Transactional) |
| **CRUD Operations** | Manual implementation | Manual implementation | Auto-generated via Repository |

* **Summary:**
* **JPA** provides the *what* to persist data but not the *how*.
* **Hibernate** is the *how*—an actual tool that implements JPA.
* **Spring Data JPA** makes it easier and cleaner to use JPA and Hibernate by removing boilerplate code.