**JPA (Java Persistence API)**

**Definition:**

JPA is a **specification (interface)** provided by Java for object-relational mapping (ORM). It defines how Java objects (entities) should be mapped to relational database tables.

**Key Points:**

* JPA is part of **Jakarta EE (formerly Java EE)**.
* It **does not provide implementation**, only a set of interfaces and annotations.
* You **need a provider** like Hibernate to use JPA.

**Main Features:**

* Mapping Java classes to database tables using annotations (@Entity, @Id, etc.).
* Managing relationships like OneToMany, ManyToOne, etc.
* Writing queries using **JPQL (Java Persistence Query Language)**.

**HIBERNATE**

**Definition:**

**Hibernate is a JPA provider – a concrete implementation of the JPA specification. It is also an ORM framework that existed even before JPA.**

**Key Points:**

* **Implements all features defined in JPA and adds many extra features.**
* **Can be used with or without JPA.**
* **Offers advanced capabilities like caching, lazy loading, interceptors, custom dialects, etc.**

**Main Features:**

* **Hibernate Query Language (HQL)**
* **Automatic table generation**
* **First-level and second-level caching**
* **Native SQL support**

**Example:**

**/\* 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;**

**}**

**SPRING DATA JPA**

**Definition:**

**Spring Data JPA is a Spring-based abstraction that builds on top of JPA and integrates it with Spring Framework. It simplifies JPA-based data access using repository interfaces and custom query generation.**

**Key Points:**

* **Internally uses JPA (and typically Hibernate as the provider).**
* **Helps in reducing boilerplate code.**
* **Enables writing interfaces only, no need for implementation classes.**
* **Uses method name conventions to generate queries automatically.**

**Main Features:**

* **CrudRepository, JpaRepository, PagingAndSortingRepository**
* **Custom query methods (findByTitle, findByPriceGreaterThan)**
* **@Query annotation for custom JPQL/SQL queries**
* **Integration with Spring Boot**

**Example:**

**EmployeeRespository.java**

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

**}**

**EmployeeService.java**

**@Autowire**

**private EmployeeRepository employeeRepository;**

**@Transactional**

**public void addEmployee(Employee employee) {**

**employeeRepository.save(employee);**

**}**