**Java Persistence API (JPA)**

* JSR 338 specification for persisting, reading, and managing data from Java objects.
* Acts as an interface/specification; does not provide actual implementation.
* Requires a provider like Hibernate, EclipseLink, etc.
* Hibernate is one of the most common implementations of JPA.

**Hibernate**

* A popular ORM (Object Relational Mapping) tool that implements the JPA specification.
* Provides additional features such as caching, lazy loading, and custom mappings.
* Offers its own APIs (Session, Criteria, etc.) along with JPA support.

**Code:**

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**

* A Spring project that adds a layer of abstraction over JPA.
* Does not implement JPA itself — instead, it works with providers like Hibernate.
* Reduces boilerplate code using repository interfaces (JpaRepository).
* Manages transactions and supports query derivation and custom queries.

**Code:**

EmployeeRepository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java

@Autowire

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}