**File Created By Sayan Saha (**[**sayansaha00876@gmail.com**](mailto:sayansaha00876@gmail.com)**)**

**Spring Data JPA – 2: Difference between JPA, Hibernate & Spring Data JPA :-**

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

JPA is a Java specification (JSR 338) for managing relational data. It provides a set of annotations and interfaces to map Java objects to database tables but does not contain any implementation.

**Role:** Specification only  
**Key Features:** Annotations, JPQL (Java Persistence Query Language), entity management  
**Common Implementations:** Hibernate, EclipseLink, OpenJPA

**Example: JPA-style Entity**

java

CopyEdit

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private String department;

// Getters and Setters

}

**2. Hibernate**

Hibernate is a popular implementation of the JPA specification. It adds many enhancements over standard JPA like lazy loading, caching, and custom SQL support. Using Hibernate directly requires manual session and transaction handling.

**Role:** JPA Implementation + additional features  
**Common Use Case:** When fine-tuned ORM control is needed

**Example: Hibernate DAO-based Insert**

java

CopyEdit

public int saveEmployee(Employee emp) {

Session session = sessionFactory.openSession();

Transaction tx = null;

Integer id = null;

try {

tx = session.beginTransaction();

id = (Integer) session.save(emp);

tx.commit();

} catch (Exception e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return id;

}

**3. Spring Data JPA**

Spring Data JPA is a Spring project that abstracts JPA operations to reduce boilerplate. It uses repositories to provide automatic implementations for CRUD, and handles transactions under the hood. It still relies on JPA + implementation (like Hibernate) underneath.

**Role:** Abstraction layer over JPA + Integration with Spring Boot  
**Key Feature:** Just define interfaces — Spring provides method implementation!

**Example: Spring Data JPA Insert**

**Entity**

java

CopyEdit

@Entity

public class Employee {

@Id

@GeneratedValue

private Integer id;

private String name;

private String department;

// Getters and Setters

}

**Repository**

java

CopyEdit

import org.springframework.data.jpa.repository.JpaRepository;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// No need to write insert, find, delete — it's auto-implemented

}

**Service**

java

CopyEdit

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void createEmployee(Employee emp) {

employeeRepository.save(emp); // Just one line!

}

}