Difference between JPA, Hibernate and Spring- data-JPA

**Building an app to manage employees:**

we want to store employee details (like name, ID, department) into a database.

Now we have three levels of tools/libraries you can use. Each adds a level of ease or abstraction.

**1. JPA – The Blueprint (Specification)**

JPA is like the *rulebook* or *blueprint*. It tells you how a Java object should be mapped to a database table, but it doesn’t do the actual work.

A car manual. It tells you how to drive a car but doesn't give you the car. For the car, you need a real vehicle (like Toyota or Honda). Here, Hibernate is the car.

**Example:**  
JPA defines interfaces like:

public interface EntityManager {

void persist(Object entity);

<T> T find(Class<T> entityClass, Object primaryKey);

}

But we can’t use them directly unless we plug in a real provider (like Hibernate).

**2. Hibernate – The Actual Car (Implementation of JPA)**

Hibernate is a concretetool that follows JPA rules and actually connects to your database. It knows how to take a Java object and store it as a row in a table.

A Toyota car that follows the car manual (JPA spec). You can drive it, you can use all features, and it’s doing the real job.

**Example using plain Hibernate:**

Session session = sessionFactory.openSession();

Transaction tx = session.beginTransaction();

Employee emp = new Employee("Kunal", "IT");

session.save(emp); // actual DB interaction

tx.commit();

session.close();

we're doing everything manually: open session, begin transaction, commit, etc.

**3. Spring Data JPA – The Driverless Car (Abstraction Over JPA)**

Spring Data JPA is a helper library on top of JPA (and usually Hibernate underneath) that removes the boilerplate. You don’t need to open sessions or write queries – it gives you ready-to-useinterfaces.

A Teslaonautopilot – we just say "Go", and it drives you. Spring Data JPA handles all the logic of saving/fetching data with minimal code.

**Example using Spring Data JPA:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

And using it:

@Autowired

EmployeeRepository employeeRepo;

public void saveEmployee() {

employeeRepo.save(new Employee("Kunal", "IT")); // No need for sessions or transactions

}