Difference Between JPA, Hibernate, and Spring Data JPA

1. Java Persistence API (JPA)

- JPA is a **standard specification** (JSR 338) provided by Java for ORM (Object-Relational Mapping).
- It defines a **set of rules and annotations** to persist Java objects into relational databases.
- JPA itself **does not provide an implementation**. It only specifies *how* the interaction between Java and the database should occur.
- Common implementations of JPA include **Hibernate**, **EclipseLink**, and **OpenJPA**

Example:

```
@Entity
public class Student {
    @Id
    private int id;
    private String name;
}
```

2. Hibernate

- Hibernate is a **concrete implementation** of the JPA specification.
- It’s also an advanced **ORM framework** that offers features **beyond JPA**, such as:
- Caching
- Lazy/eager loading
- Criteria API
- Hibernate **manages database connections**, **SQL generation**, and **object lifecycle**.

Traditional Hibernate Code Example:

```
Configuration cfg = new Configuration().configure();
SessionFactory factory = cfg.buildSessionFactory();
Session session = factory.openSession();
Transaction tx = session.beginTransaction();

Student student = new Student(1, "Praveen");
session.save(student);

tx.commit();
session.close();
```

3. Spring Data JPA

- Spring Data JPA is part of the **Spring Framework**.
- It provides an abstraction over JPA by **removing boilerplate code** like writing DAOs or CRUD operations.
- It internally uses a **JPA provider** like Hibernate but makes it easier to use via **repositories** and **annotations**.
- It supports features like **pagination, sorting, and derived queries** out of the box.

Code Comparison

◇ Hibernate Example

```
public Integer addEmployee(Employee employee) {

  Session session = factory.openSession();

  Transaction tx = null;
```

```
    Integer empId = null;


    try {
        tx = session.beginTransaction();
        empId = (Integer) session.save(employee);
        tx.commit();
    } catch (Exception e) {
        if (tx != null) tx.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }


    return empId;
}
```

Spring Data JPA Example

Repository Interface

```
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
}
Service Class
@Service
public class EmployeeService {

    @Autowired
    private EmployeeRepository employeeRepository;


    public void addEmployee(Employee employee) {
        employeeRepository.save(employee);
    }
}
```