Hibernate

주차	45주차
스터디 일자	@2024/10/11

순수 JDBC —추상화→ **영속성 프레임 워크**

SQL Mapper

MyBatis

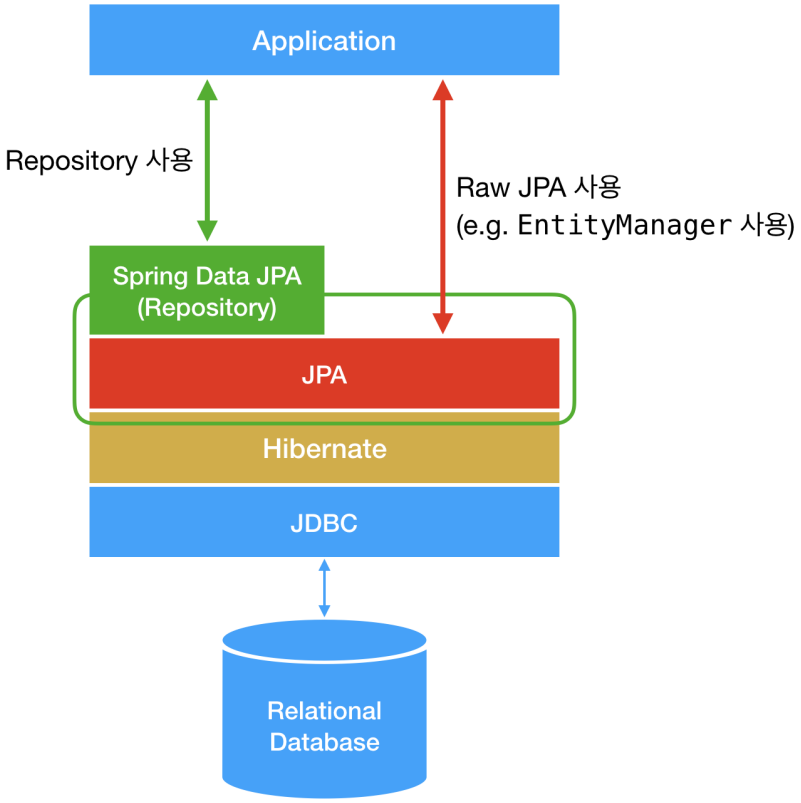
Spring Data JDBC
JdbcTemplate

ORM

JPA

HIBERNATE

\+ Spring Data JPA



Hibernate는 ORM, JDBC, JPA 로 설명할 수 있다.

1. ORM (Object Relational Mapping)

> **ORM**은 테이블을 객체지향적으로 사용하기 위한 기술이다. 객체와 DB 테이블이 매핑을 이루는 것을 의미한다.

→ 즉, 내가 코드 상에서 생성한 객체가 DB 상에 어떤 테이블과 연결이 된다는 것을 의미한다.

이렇게 되면 객체를 조작함으로써 DB를 조작할 수 있게 된다.

이로 인해 DB에 접근을 시도할 때 직접 SQL 쿼리문을 만들지 않고 아래처럼

`SELECT * FROM user` → `user.findAll()`

메서드 호출만으로 쿼리가 수행되니 ORM을 사용하면 생산성이 매우 높아진다.

ORM 프레임워크 종류

- JAVA : JPA, Hibernate, EclipseLink, Data Nucleus, Ebean 등등

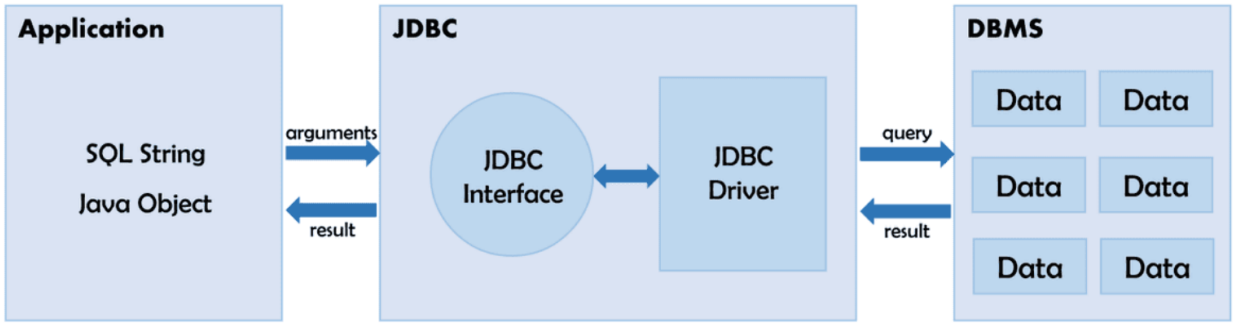
2. JDBC (Java Database Connectivity)

> **JDBC**는 ORM이 아닌, 데이터베이스에 연결 및 작업을 하기 위한 자바 표준 인터페이스이다.

자바는 DBMS의 종류에 상관 없이 하나의 JDBC API를 이용해서 데이터베이스 작업을 처리한다.

JDBC API가 생겨나기 전에는 데이터베이스의 종류마다 (MySQL, Oracle ...) 각각의 SQL을 사용해야 했다.

→ 이러한 불편함을 해결하고자 메소드, 전역변수 등을 하나의 문법으로 통일시켰고 그것이 **JDBC**다!



JDBC 아키텍처

JDBC의 역할은 위와 같은데,

그렇다면 **JPA**와 **Hibernate**는 무엇이고 무슨 역할을 할까?

3. JPA (Java Persistence API)

> **JPA**는 새로운 자바 ORM 기술 표준이자, 자바 어플리케이션에서 관계형 데이터베이스를 사용하는 방식을 정의한 **인터페이스**이다.

⇒ 간단하게 ORM을 사용하기 위해 **자바 인터페이스들**을 모아둔거라 생각하면 된다.

JPA는 영속성 컨텍스트인 `EntityManager`를 통해 Entity를 관리하고

이러한 Entity가 DB와 **매핑되어**

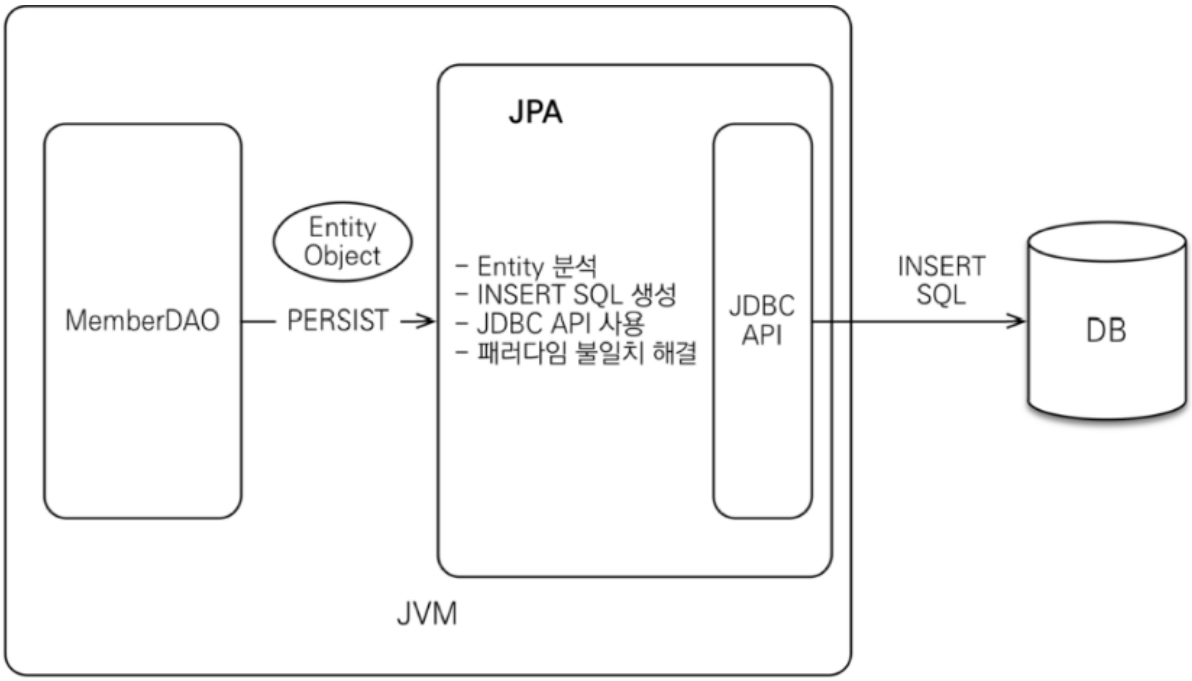
사용자가 Entity에 대한 CRUD를 실행했을 때 Entity와 관련된 테이블에 대한 적절한 SQL 쿼리문을 생성하고

이를 관리하였다가 필요시 JDBC API를 통해 DB에 날리게 된다.

이러한 JPA는 애플리케이션과 JDBC 사이에서 동작한다.

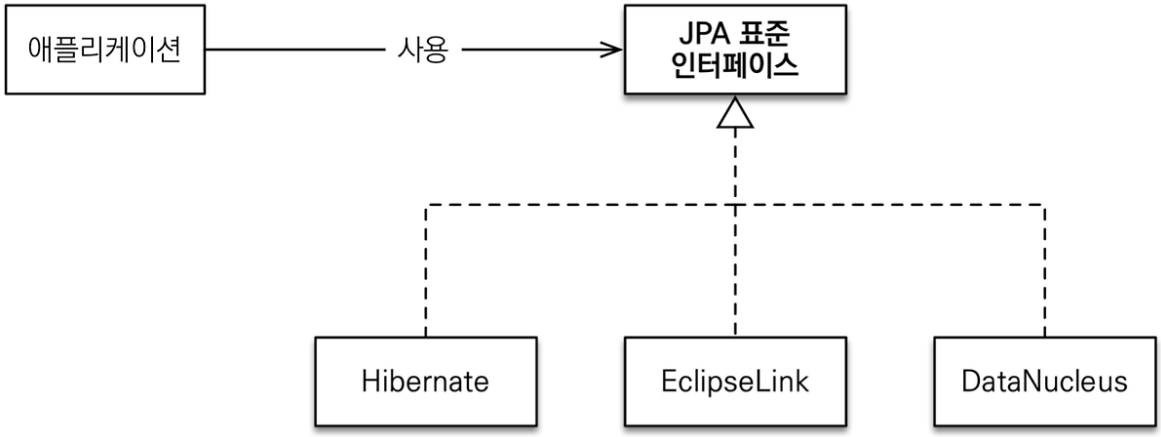
JPA는 데이터베이스와 객체를 매핑하는 기술일 뿐, **내부적으로는 데이터베이스와의 통신을 위해 JDBC를 사용한다.**

→ 개발자가 JPA를 사용하면, JPA 내부에서 JDBC API를 사용하여 SQL을 호출하여 DB와 통신한다.



이러한 JPA도 JDBC와 마찬가지로 **인터페이스**이고,

그렇기 때문에 구현체가 필요하고, 그 구현체 중 하나가 **Hibernate** 이다.

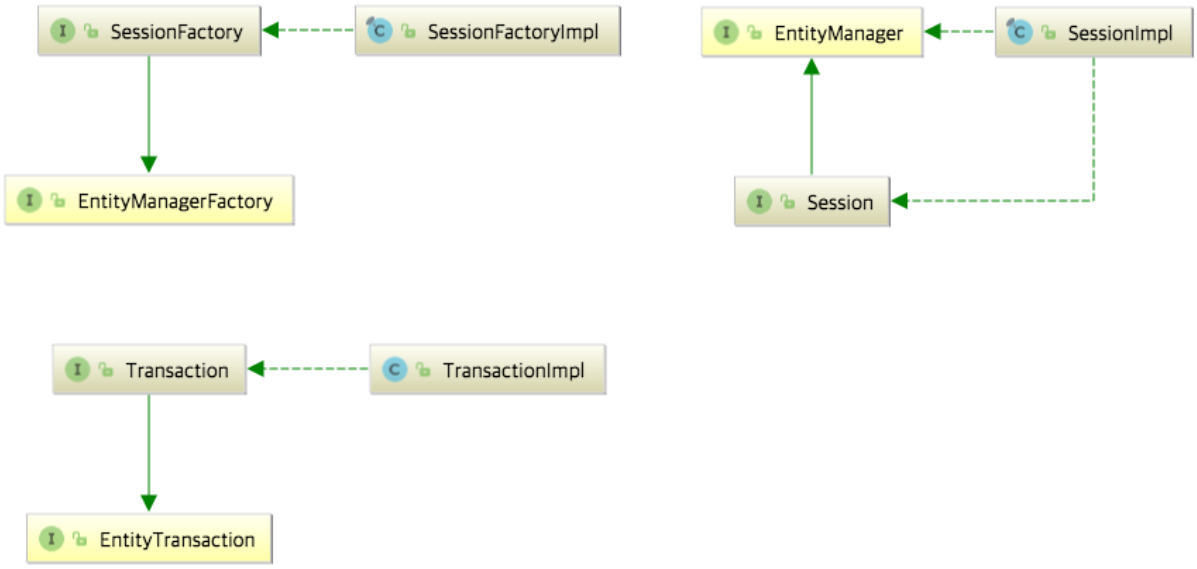


4. Hibernate

> **Hibernate**는 자바 언어를 위한 ORM 프레임워크다. **JPA의 구현체**로, JPA 인터페이스를 구현하며, **내부적으로는 JDBC API**를 사용한다.

`JPA ↔ Hibernate` 는 마치

`자바의 interface ↔ 해당 interface를 구현한 class` 와 같은 관계이다.



JPA와 Hibernate의 상속 및 구현 관계

JPA의 핵심인 `EntityManagerFactory`, `EntityManager`, `EntityTransaction` 을 Hibernate 에서는 각각 `SessionFactory`, `Session`, `Transaction` 으로 상속받고 각각 Impl 로 구현하고 있음을 확인할 수 있다.

이러한 Hibernate는 **JPA의 구현체**이다.

즉, JPA를 사용하기 위해서 **반드시 Hibernate만을 사용해야 할 필요는 없다는 것**

→ Hibernate의 작동 방식이 마음에 들지 않는다면 Data Nucleus, EclipseLink 등 다른 JPA 구현체를 사용해도 되고, 본인이 직접 JPA를 구현해서 사용할 수도 있다.

그치만 Hibernate가 제일 잘되있고 제일 적합해서 딴 건 안 쓴다고 한다

Hibernate 장단점

👍 장점

- **생산성**

  복잡한 SQL 쿼리를 자동으로 생성하여 데이터베이스 접근 코드를 단순화

- **유지보수성**

  객체 모델을 기반으로 데이터베이스 스키마를 자동으로 생성 및 업데이트할 수 있어 유지보수 용이

- **이식성**

  다양한 데이터베이스 시스템을 지원하여 데이터베이스 변경 시 코드 수정 최소화

👎 단점

- **학습 곡선**

  Hibernate의 다양한 기능과 설정을 이해하는 데 시간이 필요

- **성능 오버헤드**

  자동화된 작업으로 인해 일부 성능 오버헤드 발생 가능

https://velog.io/@murphytklee/Spring-ORM-JPA-Hibernate-JDBC-총정리#5-hibernate

https://m.blog.naver.com/rorean/221411555979

https://livenow14.tistory.com/70

https://nbcamp.spartacodingclub.kr/blog/개념-콕-웹-개발-지식-편-jpa란-orm-hibernate-jpa--21323