JPA 소개

1. sql의 문제점

현 데이터 베이스 세게는 관계형 db이다. 객체를 관계형 데이터베이스에 저장한다.

그러나 sql 중심적인 개발에는 무한 반복, 지루한 코드라는 문제점이 있다.

객체를 자바 컬렉션에 저장하듯이 db에 저장할 수는 없을까? -> sol = jpa

2. JPA 소개

1) JPA : java persistence api, 자바 진영의 ORM 기술 표준

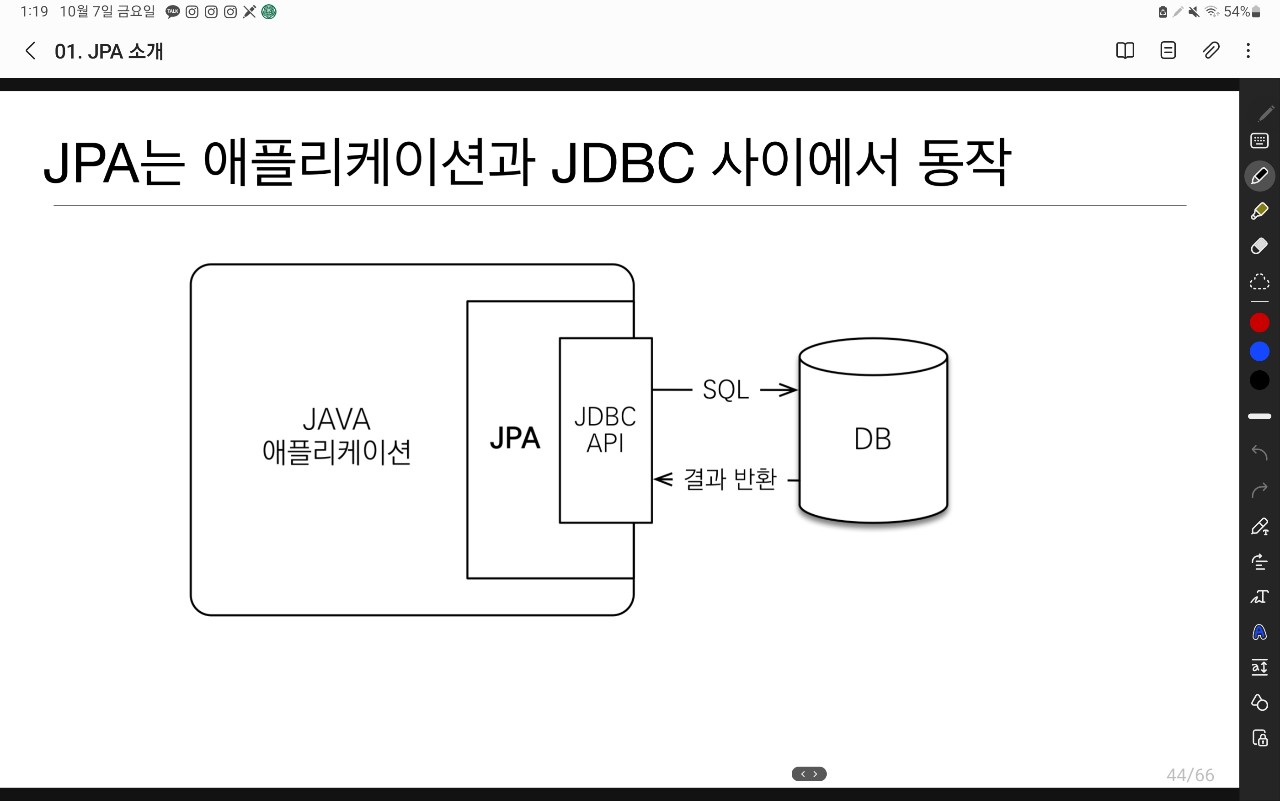
2) ORM : object-relational mapping (객체관계매핑)

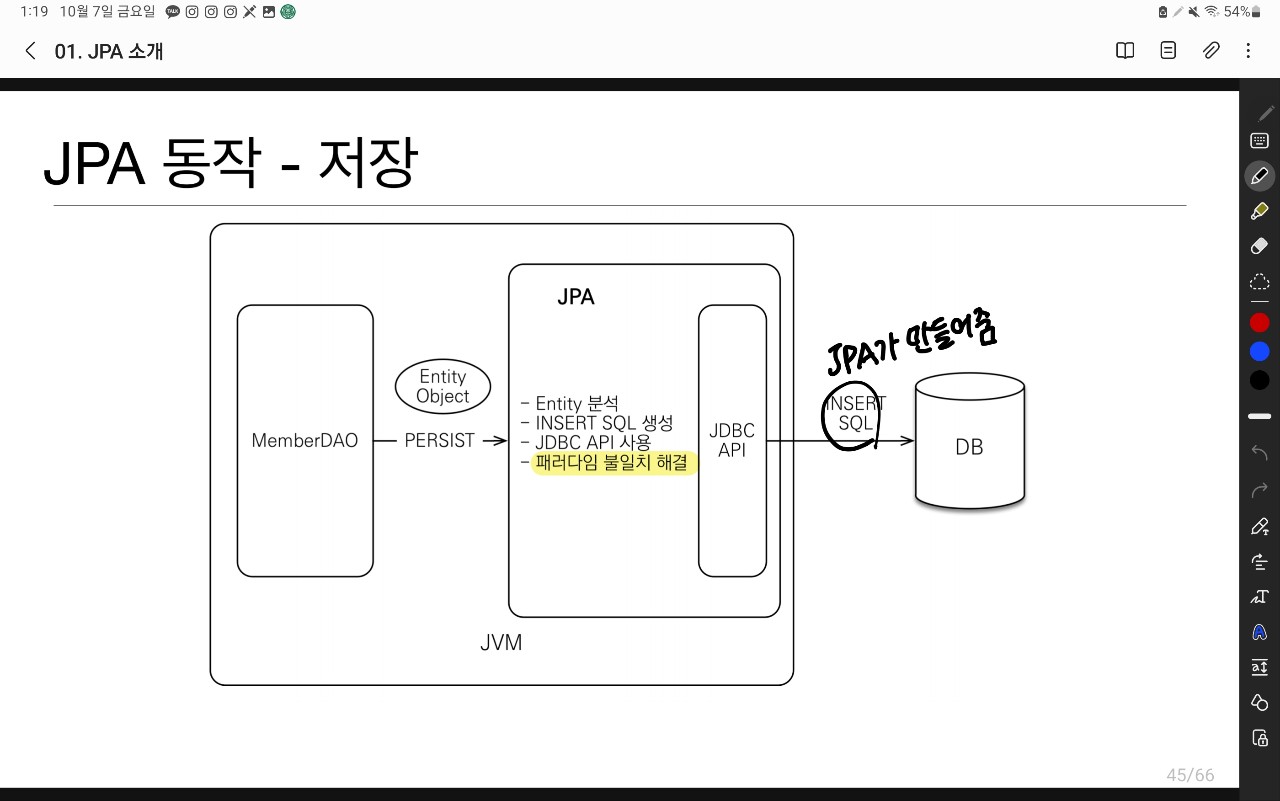
- 객체는 객체대로 설계

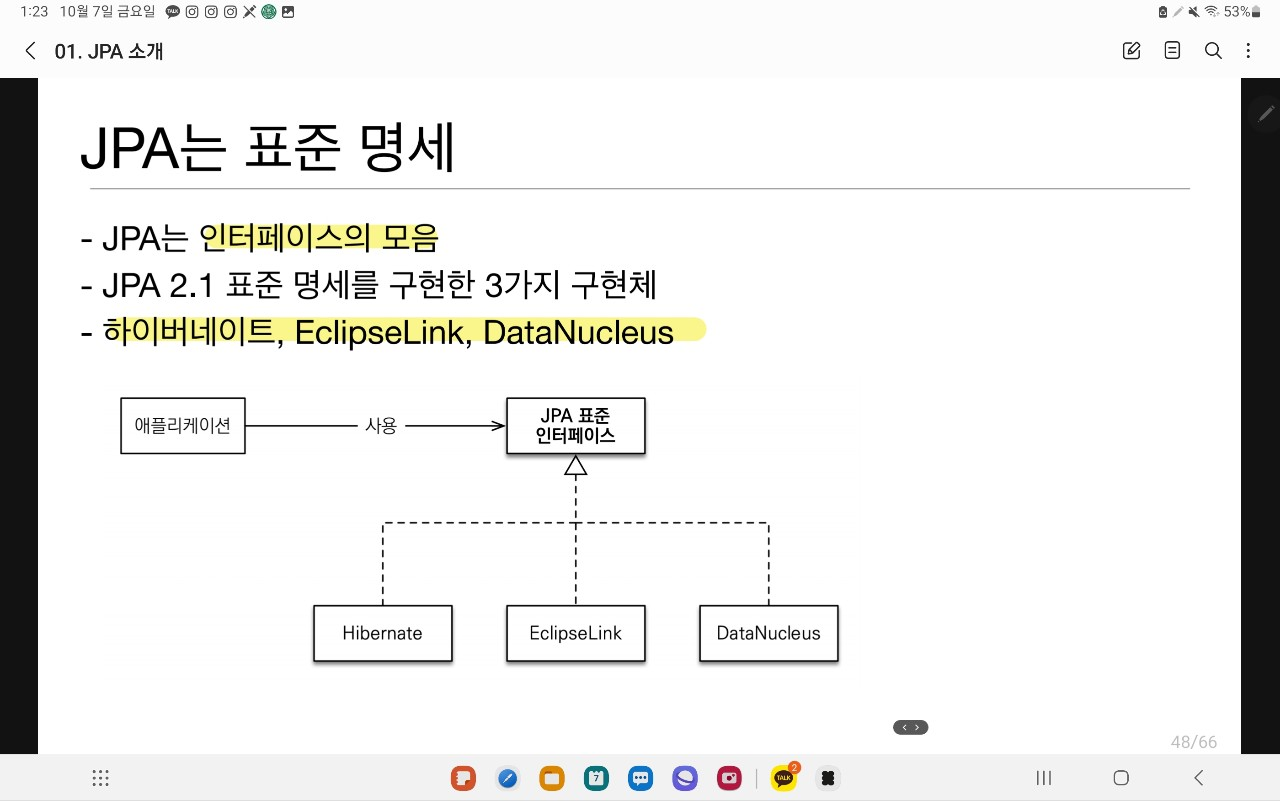
- 관계형 데이터베이스는 관계형 데이터베이스대로 설계

- ORM 프레임워크가 중간에서 매핑

- 대중적인 언어에는 대부분 ORM 기술이 존재







3. JPA를 사용해야하는 이유

1) SQL 중심적인 개발에서 객체중심으로 개발

2) 생산성

3) 유지보수

4) 패러다임의 불일치 해결

5) 성능

6) 데이터 접근 추상화와 벤더 독립성

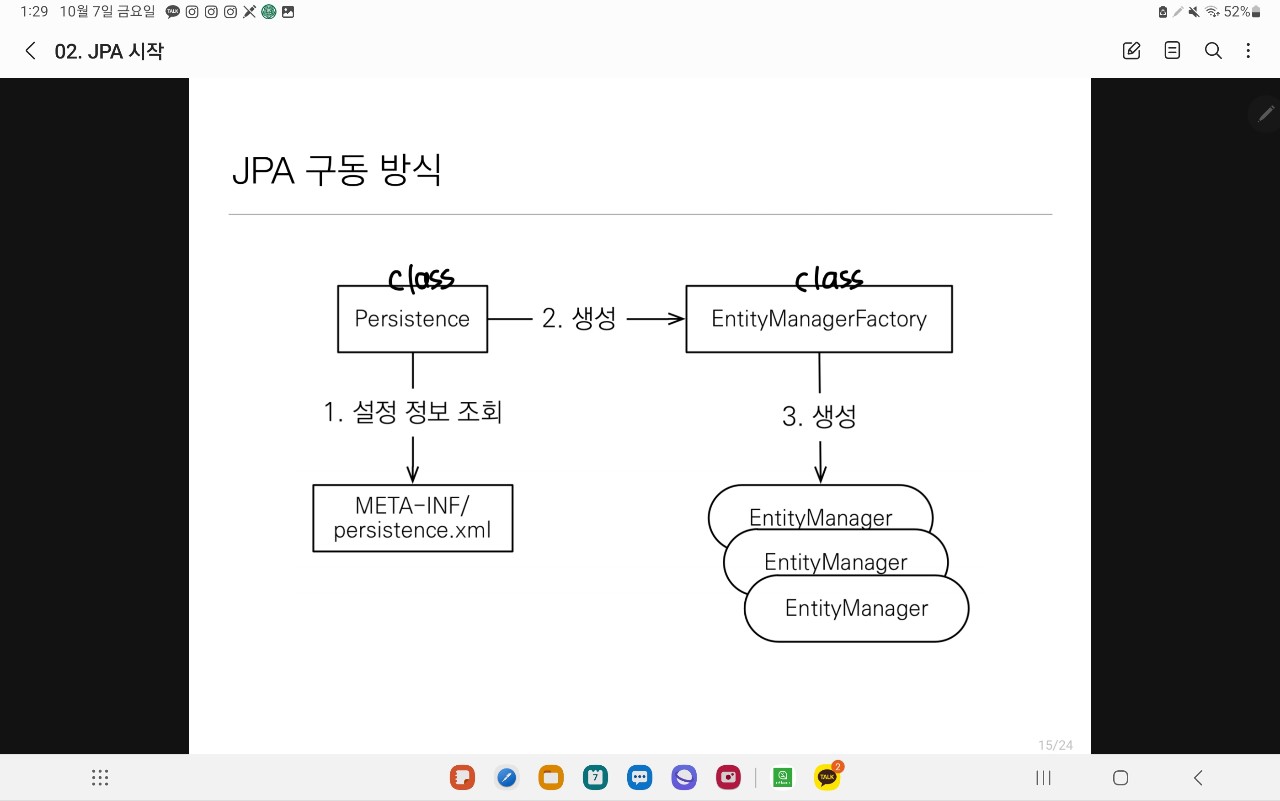
7) 표준

JPA 시작하기

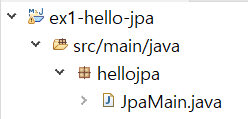
다음과 같은 파일과 코드를 붙여넣어 초기 설정을 완료.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

이후 다음과 같이 파일과 패키지를 생성한다.





실습 – 회원 저장

~ 회원 등록

~ 회원 수정

~ 회원 삭제

~ 회원 단 건 조회

jpa에서 가장 중요한 2가지

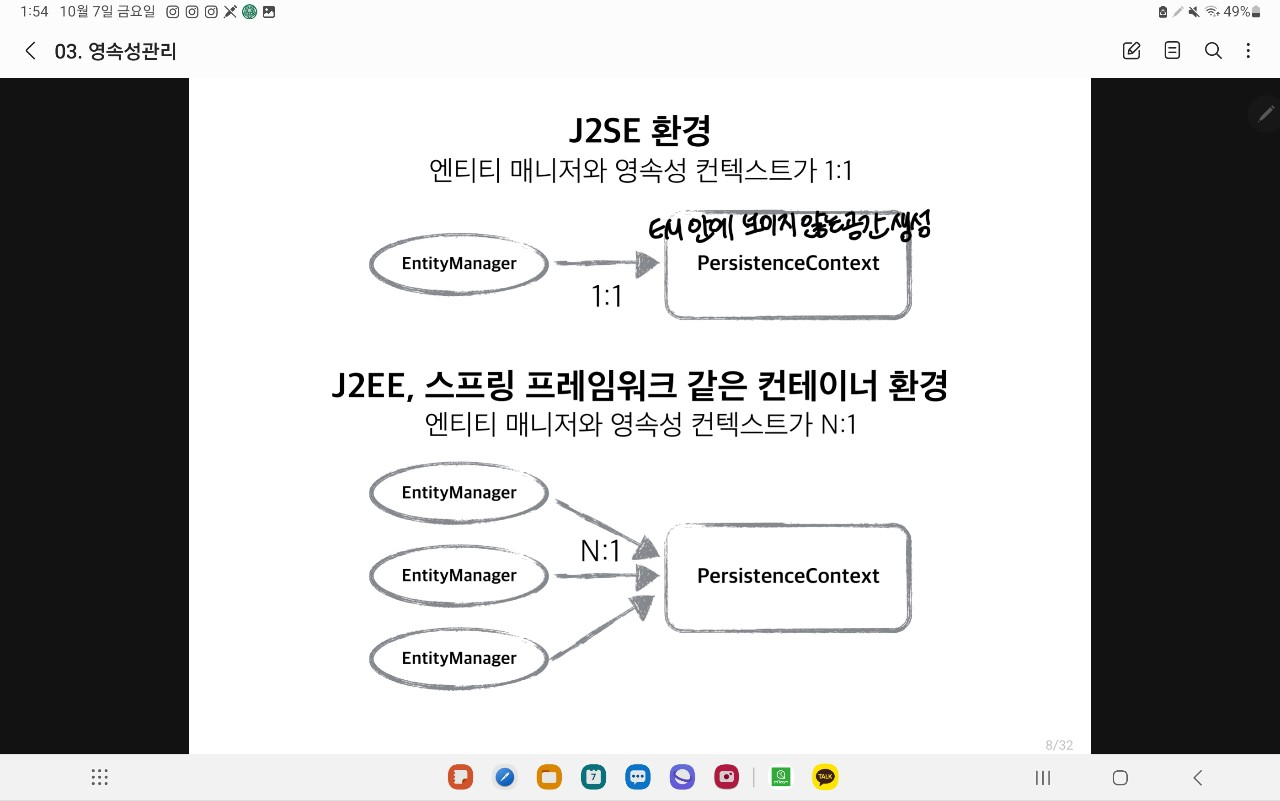
- 객체와 관계형 데이터베이스 매핑하기 : 즉 정적 설계

- 영속성 컨텍스트 : jpa가 실제 동작하는 방식

영속성 컨텍스트 : 엔티티를 영구 저장하는 환경

EntityManager.persist(entity); entity를 영속성 컨텍스트에 저장

엔티티 매니저를 통해서 영속성 컨텍스트에 접근



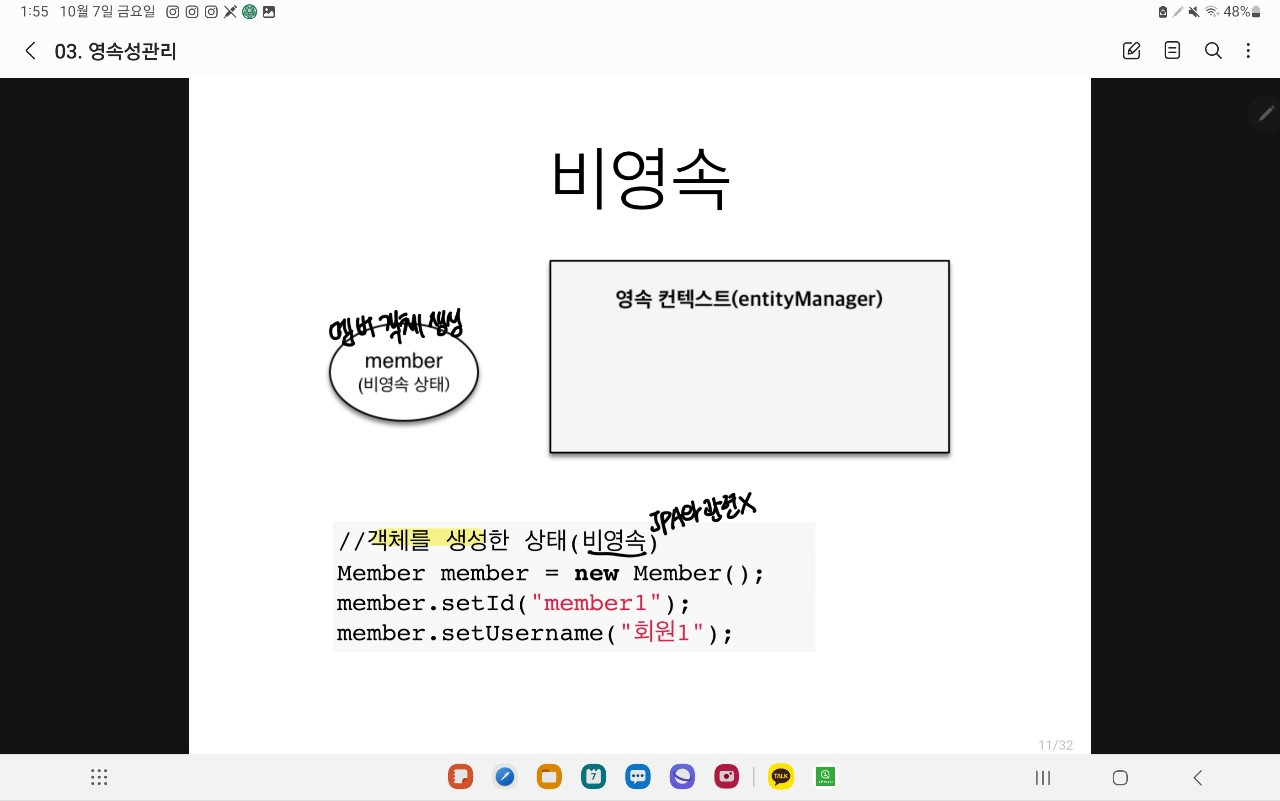
엔티티의 생명 주기

- 비영속성 : 영속성 컨텍스트와 전혀 관계없는 새로운 상태

- 영속 : 영속성 컨텍스트에 의해 관리되는 상태

- 준영속 : 영속성 컨텍스트에 저장되었다가 분리된 상태

- 삭제 : 삭제된 상태



영속은 위의 그림과 같은 코드에 em.persist(member); 코드를 추가하면 됨

플러시

: 영속성 컨텍스트의 변경 내용을 데이터 베이스에 반영

영속성 컨텍스트를 비우지 않음

영속성 컨텍스트의 변경 내용을 데이터 베이스에 동기화

트랜잭션이라는 작업 단위가 중요 -> 커밋 직전에만 동기화하면 된다.

플러시 발생 시

: 변경 감지, 수정된 엔티티 쓰기 지연 sql 저장소에 등록, 쓰기 지연 sql 저장소의 쿼리를 데이터베이스에 전송

영속성 컨텍스트를 플러시 하는 방법

: em.flush() – 직접 호출, 트랜잭션 커밋 – 플러시 자동 호출, jpql 쿼리 실행 – 플러시 자동 호출