1-1-3. 엔티티(Entity)

- JPA의 엔티티는 테이블을 객체로 표현한 자바 클래스이다. 이 엔티티의 인스턴스는 테이블의 행을 나타내고 대체로 다른 엔티티와 관계를 맺고 있으며 이러한 관계(오브젝트/관계) 메타데이터는 클래스에 어노테이션(@ManyToOne, @OneToOne, @ManyToOne, @ManyToMany)을 기술하거나 XML 파일 등에서 설정할 수 있다. 이러한 엔티티는 구현 클래스 또는 추상 클래스가 될 수 있으며 특성 또는 필드를 사용하여 상태를 관리한다.
- @Entity, @Table 어노테이션을 이용하여 엔티티와 테이블을 매핑한다.
- JPQL(Java Persistence Query Language)은 지속적 엔티티를 저장하는 데 사용되는 메커니즘과 독립적으로 지속적 엔티티에서 검색을 정의하는 데 사용되는데 데이터베이스와 관련된 SQL을 사용하지 않고 오브젝트를 검색하는 언어이다.
- 엔티티의 4가지 생명주기
  - ✓ 비영속 객체(new/transient) : new 키워드를 생성한 엔티티 객체를 말하며 Managed 상태가 아닌 것을 말한다.
  - ✓ 영속 객체(managed) : new 이후 perist 메소드를 이용해서 저장한 경우 또는 DB의 데이터를 find나 query등으로 조회한 경우인데 영속성 컨텍스에 저장된 상태이다.
  - ✓ 삭제 객체(removed) : 영속객체 즉 Managed 상태인 객체를 remove 메소드로 삭제한 경우, 작업이 커밋되는 시점에 DB에 동기화 된다.
  - ✓ 준영속 객체(detached) : 트랜잭션이 커밋되었거나 clear, flush 메소드가 실행된 경우 모든 영속객체의 상태는 준영속 상태(Detached) 상태가 된다. 영속성 컨텍스트에서 분리된 상황이며 merge 메소드로 다시 영속객체가 될 수 있다.

- 엔티티는 JPA가 생성할 때 기본 생성자를 이용하므로 반드시 기본 생성자를 가져야 하며 식별자 프로퍼티(@Id)가 반드시 존재 해야 한다.
- final 클래스는 엔티티가 될 수 없다. Proxy를 생성할 때 자식클래스 형태로 만드는데 final 클래스는 상속되지 않는 클래스이므로 불가하다. 만약 final 메소드를 가진 클래스를 엔티티로 사용하려면 명시적으로 lazy="false"로 설정해서 Proxy 생성을 막아야 한다.
- 하이버네이트는 기본적으로 자바빈의 프로퍼티를 테이블의 칼럼에 매핑하며 getter/setter 메소드를 자동 인식한다.
- 하이버네이트에서는 기본적으로 동일한 세션에서만 엔티티 객체에 대해 Persistent identity(데이터베이스 행의 식별)과 Java identity(데이터베이스행에 대응되는 자바객체의 식별)을 보장하므로 여러 세션에서 같은 객체로 인식되게 하기 위해서는 equals(), hashCode()를 오버라이딩 해야 한다. 특히 엔티티(영속 클래스)를 컬렉션 등에 담을 때는 반드시 위해서는 equals(), hashCode()를 구현해야 한다.

[하이버네이트 예]

```
Session sessionA = sessionFactory.openSession();


// 사번(식별자)으로 데이터 읽기
```

```
Emp emp1 = sessionA.get(Emp.class, new Integer(7369) );
Emp emp2 = sessionA.get(Emp.class, new Integer(7369) );

// 이 경우 emp1과 emp2는 동일하다.(emp1 == emp2)
session1.close();
// 이때 emp1과 emp2는 영속성 컨덱스트에서 빠져나와 detached 상태가 된다.

Session sessionB = sessionFactory.openSession();
Emp emp3 = sessionB.get(Emp.class, new Integer(7369) );
.....
.....
// 이 경우 emp1, emp2와 emp3 객체는 세션이 다르기 때문에 동일하지 않다.
// emp1, emp2는 detached 객체이고 emp3은 영속성 객체이다.
// 위 세개의 객체는 PK값 자체로 보면 모두 7369라는 사번을 가지는 객체이므로 사번으로 비교하면 같은 객체로 인식이 가능하다.
```

자바의 equals 메소드로 위 세객체를 비교한다고 했을 때 Emp 클래스가 equals 메소드를 구현하지 않았다면 Object 클래스의 equals로 비교하므로 == 와같은 의미이다.

위 세개의 객체를 자바 컬렉션(Set)에 emp1, emp2, emp3 순서로 추가한다고 가정해 보자. Set은 순서도 없고 중복도 허락하지 않는 객체이므로 추가 될 때 equals 메소드로 비교하여 같은 것이 없는 경우에만 추가한다. Emp 클래스가 equals를 구현하지 않았다면 == 로 비교하여 판단하므로 세 객체중 emp1, emp3 두객체가 저장 될 것이다. 만약 같은 사번의 객체가 하나만 저장되기를 원한다면 euqals 메소드를 구현해야 한다.

1-1-4. 엔티티 매니저(Entity Manager)

➢ **JPA EntityManagerFactory**

- 데이터베이스와의 상호 작용을 위한 EntityManager를 생성하기 위해 사용되는데 엔티티 매니저 팩토리는 애플리케이션 전체에서 딱 한번만 생성하고 공유해서 사용한다.

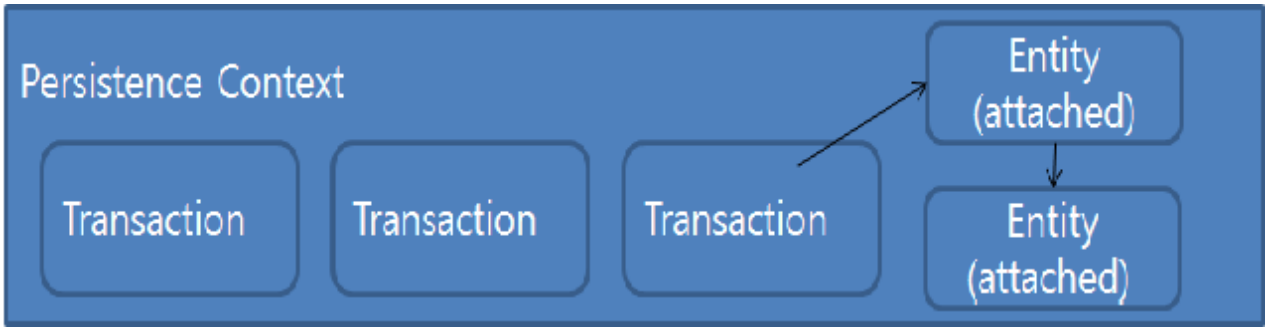
➢ **JPA EntityManager**

- 도메인 객체를 테이블로 변환하고 엔티티와 관련된 저장, 조회, 수정, 삭제와 같은 일을 한다. DB에서 엔티티를 가져오거나 생성, 삭제, 수정하는 일은 모두 EntityManager를 통해 이루어진다.

- 엔티티 저장 요청시 엔티티 객체를 생성하여 DB에 저장하며 SQL과 같은 CRUD 오퍼레이션을 제공한다.
- EntityManager의 인스턴스는 영속성 컨텍스트(Persistence Context)를 나타내며 EntityManagerFactory를 통해서 얻는다.

- 트랜잭션 범위 EntityManager : 트랜잭션 기간동안 엔티티는 attach 상태이고 종료 후 자동으로 detach 상태가 된다.

- 확장 범위 EntityManager : 여러 트랜잭션을 걸쳐 라이프 사이클이 지속되며 유상태 세션빈과 함께 사용되며 빈 인스턴스가 살아있는 동안 계속된다. 빈 자체가 제거되거나 EntityManager가 종료되어야 끝난다.



➢ JPA EntityManager Interface

- **persist(Object entity)** : 엔티티를 DB로 저장
- **merge(T entity)** : EntityManager의 영속성 컨텍스트로 엔티티를 병합
- **remove(Object entity)** : DB에서 엔티티를 삭제
- **find(Class<T> entityClass, Object primaryKey)** : 주키로 엔티티 인스턴스를 찾는다.

- **flush()** : EntityManager의 영속성 컨텍스트안의 엔티티를 DB와 동기화 한다. 트랜잭션을 커밋하거나 JPQL 쿼리를 실행해면 자동으로 flush 된다.
- **setFlushMode(FlushModeType flushMode)** : EntityManager의 지속성 컨텍스트의 flush 모드를 설정한다.(AUTO, COMMIT)
- **getFlushMode()** : 현재의 flush 모드를 리턴한다.
- **refresh(Object entity)** : DB의 엔티티를 리셋
- **createQuery(String jpsqString)** : JPQL문을 이용하여 동적 질의문 생성
- **createNamedQuery(String name)** : 이름있는 질의 인스턴스 생성
- **createNativeQuery(String sqlString)** : 원시 SQL문을 이용하여 동적 질의 생성
- **close()** : EntityManager를 종료한다.
- **isOpen()** : EntityManager의 오픈여부 확인
- **getTransaction()** : 트랜잭션 객체를 검색한다.
- **joinTransaction()** : 기존 JTA 트랜잭션 조인을 요청한다.

1-1-5. 영속성 컨텍스트(Persistence Context)

- EntityManager가 엔티티를 관리할 때 영속성 컨텍스에 보관하고 관리한다. 즉 현재 관리되는 모든 엔티티 인스턴스의 집합을 Persistence Context라고 한다. EntityManager를 통해 접근 가능하며 EntityManager에 의해 관리되는 엔티티 컨테이너의 집합이다.
- JPA에서 영속성 컨텍스트에 엔티티를 보관할때 최초 상태를 복사해서 저장하는데 이것을 스냅샷이라 한다. 플러시 시점에 스냅샷과 엔티티를 비교해서 변경된 엔티티를 찾는다. (플러시란 영속성 컨텍스트의 변경 내용을 데이터베이스에 반영하는 것이다.)
- 자동변경감지 : 영속상태의 객체가 영속성 컨텍스트에서 관리되는데 이 영속상태의 객체의 어떤 속성이 변경됨을 자동으로 감지한다. 이 영속객체는 작업단위가 종료되는 시점에 DB와 동기화 된다.
- 캐시 : 영속성 컨텍스트에서 관리하는 영속객체는 영속성 컨텍스트에서 모두 기억하는데 어떤 key를 기준으로 Entity 객체를 조회한 경우 DB의 실제 데이터를 조회하기 전에 영속성 컨텍스트 내부의 영속객체를 먼저 조회한다. 없는 경우에만 DB에서 조회하고 이를 반복 가능한 읽기라고 하며 성능상 이점이 있다.
- 객체동일성 : 자바쪽과 DB쪽 모두 일치하는 경우 동일하게 간주한다. 즉 자바에서 "==" 연산자로 비교했을 때도 같아야 하며 DB에서 key로 비교했을 때도 일치해야 동일하다는 것이다.

JPA Class Level Architecture