(스터디) 9장 값 타입

엔티티 타입

- @Entity 로 정의하는 객체
- 데이터가 변해도 식별자로 지속해서 추적 가능

ex) 회원 엔티티의 키나 나이값을 변경해도 식별자로 인식 가능

9.1 기본값 타입 p.320

- int, Integer, String 처럼 단순히 값으로 사용하는 자바 기본 타입이나 객체
- 식별자가 없고 값만 있으므로 변경시 추적 불가
- 생명주기를 회원 엔티티에 의존한다. ex) 회원 엔티티 인스턴스를 제거하면 name, age 값도 제거됨.
- 공유하면 안 된다. ⇒ 회원 엔티티의 이름 변경 시 다른 회원 이름도 변경되면 안됨.

- 자바 기본 타입(int, double)
- 래퍼 클래스(Integer, Long)
- String

9.2 임베디드 타입(복합 값 타입) p.321

- 새로운 값 타입을 직접 정의해서 사용할 수 있는데 JPA 에서는 이것을 임베디드 타입(embedded type)이라 한다.
- 임베디드 타입도 int, String 처럼 값 타입이다.
- 주로 기본 값 타입을 모아서 만들어서 복합 값 타입이라고도 한다.

- `@Embeddable` : 값 타입을 정의하는 곳에 표시
- `@Embedded` : 값 타입을 사용하는 곳에 표시
- 기본 생성자 필수

임베디드 타입의 장점

- 재사용할 수 있다.
- 응집도가 높다.
- `Period.isWork()` 처럼 해당 값 타입만 사용하는 의미 있는 메소드를 만들 수 있다.
- 임베디드 타입을 포함한 모든 값 타입은, 값 타입을 소유한 엔티티에 생명주기를 의존한다.

9.2.1 임베디드 타입과 테이블 매핑 p.324

- 임베디드 타입은 엔티티의 값일 뿐이다.
- 임베디드 타입을 사용하기 전과 후에 매핑하는 테이블은 같다.
- 객체와 테이블을 아주 세밀하게(find-grained) 매핑하는 것이 가능하다.
- 잘 설계한 ORM 애플리케이션은 매핑한 테이블의 수보다 클래스의 수가 더 많다.

9.2.3 @AttributeOverride : 속성 재정의 p.327

- 한 엔티티에서 같은 값 타입을 사용하면, 컬럼명이 중복된다.

⇒ **@AttributeOverrides, @AttributeOverride**를 사용해서 컬러 명 속성을 재정의

9.3.2 값 타입 복사 p.330

- 값 타입의 실제 인스턴스인 값을 공유하는 것은 위험하다.
- 대신 값(인스턴스)을 복사해서 사용해야 한다.

객체 타입의 한계

- 항상 값을 복사해서 사용하면 공유 참조로 인해 발생하는 부작용을 피할 수 있다. 하지만 문제는 임베디드 타입처럼 **직접 정의한 값 타입은 자바의 기본 타입이 아니라 객체 타입**이다.
- **객체 타입은 참조 값을 직접 대입하는 것을 막을 방법이 없다.**
- 객체의 공유 참조는 피할 수 없다.

  ⇒ setter 메소드 없애기 : setCity() 같은 수정자 메소드를 모두 제거하기

9.3.3 불변 객체 p.331

- 객체를 불변하게 만들면 값을 수정할 수 없으므로 부작용을 원천 차단할 수 있다.
- 따라서 값 타입은 불변 객체로 설계해야 한다.
- 생성자로만 값을 설정하고 수정자를 만들지 않으면 된다.
- 참고 : Integer, String 은 자바가 제공하는 대표적인 불변 객체다.

9.4 값 타입의 비교 p.333

- 값 타입은 인스턴스가 달라도 그 안에 값이 같으면 같은 것으로 봐야한다.
- 따라서 값 타입을 비교할 때는 `equals()` 를 사용해야 한다.

9.5 값 타입 컬렉션 p.334

- 값 타입을 하나 이상 저장하려면 컬렉션에 보관한다.
- @ElementCollection, @CollectionTable 사용
- 관계형 데이터베이스의 테이블은 컬럼 안에 컬렉션을 포함할 수 없다. 즉 컬렉션을 같은 테이블에 저장할 수 없다.
- 별도의 테이블을 추가하고 @CollectionTable 를 사용해서 추가한 테이블을 매핑해야 한다.

- 컬렉션을 저장하기 위한 별도의 테이블이 필요하다.

9.5.2 값 타입 컬렉션의 제약사항 p.339

- 엔티티는 식별자가 있으므로 엔티티의 값을 변경해도 식별자로 데이터베이스에 저장된 원본 데이터를 쉽게 찾아서 변경할 수 있다.
- 반면에 값 타입은 식별자라는 개념이 없고 단순한 값들의 모임이므로 값을 변경해버리면 데이터베이스에 저장된 원본 데이터를 찾기 어렵다.
- 값 타입 컬렉션에 변경 사항이 발생하면, 주인 엔티티와 연관된 모든 데이터를 삭제하고, 값 타입 컬렉션에 있는 현재 값을 모두 다시 저장한다.
- 값 타입 컬렉션을 매핑하는 테이블은 모든 컬럼을 묶어서 기본 키를 구성해야 한다.
  - 컬럼에 null 을 입력할 수 없고, 같은 값을 중복해서 저장할 수 없다.

값 타입 컬렉션 대안

- 실무에서는 상황에 따라 **값 타입 컬렉션 대신에 일대다 관계를 고려**
- 일대다 관계를 위한 엔티티를 만들고, 여기에서 값 타입을 사용
- 영속성 전이(Cascade) + 고아 객체 제거를 사용해서 값 타입 컬렉션 처럼 사용