Spring Boot 03

[도메인 분석 설계]

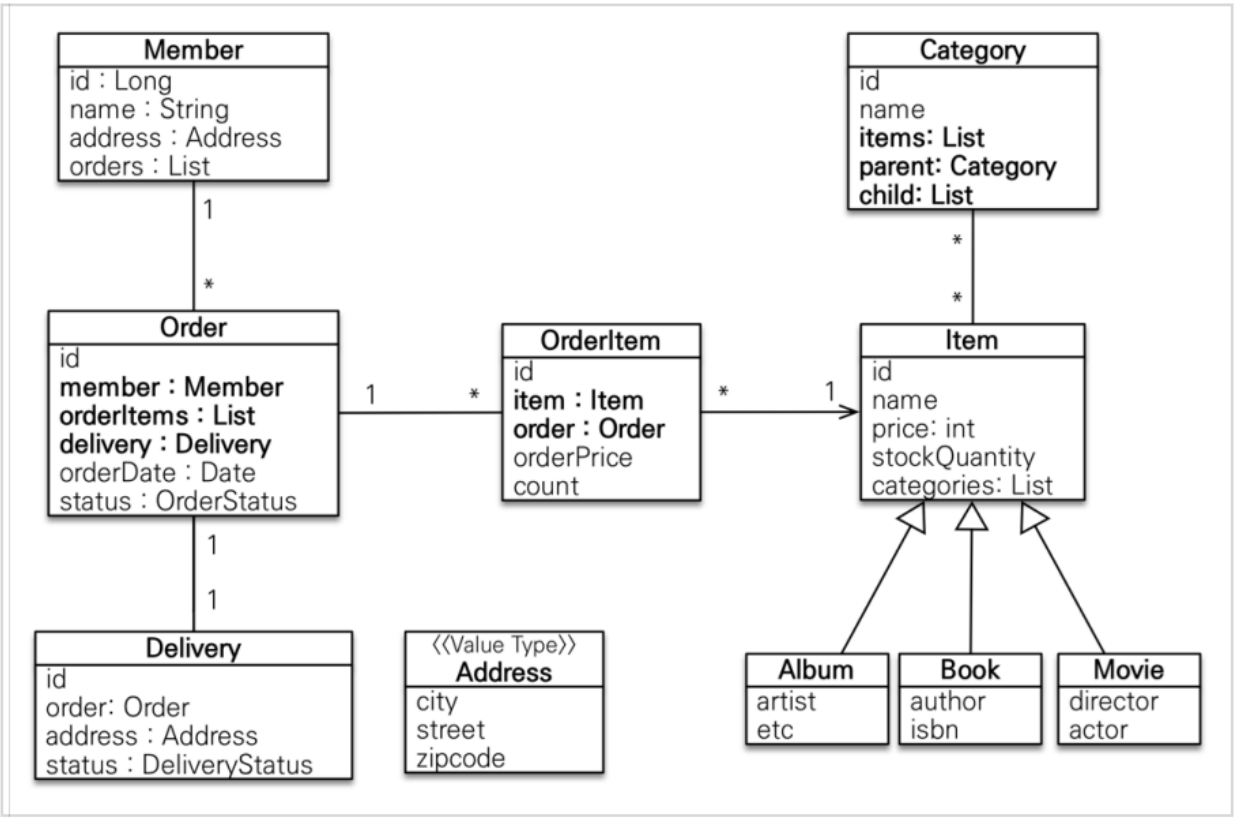
실전! 스프링 부트와 JPA 활용 1 [복습]

김영한

2022.08.10

기능

회원, 상품, 주문 CRUD



도메인 설계

다대다는 orderItem 처럼 일대다, 다대일로 풀어낸다.

테이블 명 같은경우 회사마다 다르다. 언더스코어 스타일을 사용을 많이하긴 한다.

엔티티 설계

연관관계의 주인

Member - Order에서 연관관계 주인을 정해야 하는데 외래키가 있는 다 쪽이 연관관계 주인인것이 좋다. 연관관계 주인의 값을 바꿔야 엔티티 멤버가 바뀌게 되는것.

OrderItem - Order 관계도 마찬가지이다. 다만 OrderItem → Item 은 다대일 단방향 관계이다.

테이블 설계는 동일하나 엔티티 설계에서 Item내부에 OrderItem List는 존재하지 않는다.

테이블에는 외래키가 한 곳에 존재하지만 엔티티 객체입장에서 변경 포인트가 **2**곳이다**.** 그러기 때문에 **JPA** 에서 한 곳을 정해서 주인으로 정해준다**.**

엔티티 상속관계

```
@Entity
@Getter @Setter
@Inheritance(strategy = InheritanceType.SINGLE_TABLE) → 테이블 생성전략
public abstract class Item {

   @Id
   @GeneratedValue
   @Column(name = "item_id")
   private Long id;

   private String name;
   private int price;
   private int stockQuantity;
}
```

카테고리 계층 관계

```java
public class Category {

   @Id
   @GeneratedValue
   @Column(name = "category_id")
   private Long id;

   private String name;

   @ManyToMany
   @JoinTable(name = "category_item",
           joinColumns = @JoinColumn(name = "category_id"),
           inverseJoinColumns = @JoinColumn(name = "item_id"))
   private List<Item> items = new ArrayList<>();


   @ManyToOne(fetch = FetchType.LAZY)
   @JoinColumn(name= "parent_id")
   private Category parent;

   @OneToMany(mappedBy = "parent")
   private List<Category> child = new ArrayList<>();
```

→ 자기 자신을 매핑하는데 마치 다른 테이블을 매핑하듯이 부모 자식 관계를 매핑하면 된다.

엔티티 설계시 주의점

Setter사용 지양

엔티티에는 가급적 Setter를 사용하지 말자. ⇒ 변경 포인트가 너무 많다

지연로딩 사용

모든 연관관계는 지연 로딩을 사용해서 N+1쿼리 발생 문제를 예방한다.

- 필요한 부분에서 fetch join 이나 엔티티 그래프 기능을 사용한다.
- @XToOne 관계는 기본이 즉시로딩이니 지연 로딩으로 설정해야 한다.

컬렉션의 초기화

컬렉션은 필드에서 초기화 하는것이 좋다.

하이버네이트가 영속화 할 때, 컬렉션을 감싸서 하이버네이트가 제공하는 내장 컬렉션으로 변경한다. 만약 임의의 메서드에서 컬렉션을 잘못 생성하면 내부 메커니즘에 문제가 생길수 있다.

즉, 하이버네이트가 감싼 상태에서 다른 매서드가 다시 컬렉션을 초기화한다? : 문제 발생

컬렉션은 컬렉션 자체로 사용하는 것을 권장한다**.**

테이블**,** 컬럼명 생성 전략

스프링 부트에서 하이버네이트 기본 매핑 전략을 변경해서 테이블과 컬럼명을 매핑해서 바꿔준다. 카멜케이스 → 언더스코어 + 소문자 로 변경해준다.

논리명 생성**:** 명시적으로 컬럼, 테이블명을 직접 적지 않으면 ImplicitNamingStrategy 사용
spring.jpa.hibernate.naming.implicit-strategy : 테이블이나, 컬럼명을 명시하지 않을 때 논리명

물리명 적용**:** spring.jpa.hibernate.naming.physical-strategy : 모든 논리명에 적용됨, 실제 테이블에 적용 (username usernm 등으로 회사 룰로 바꿀 수 있음

Cascade 전략

List나 엔티티에 add 후 persist를 해줘야하는데 함께 persist가 이뤄질수 있게 한다.

연관 관계 편의 메소드

양방향 연관관계에서 있으면 유용하다.

```
//연관관계 편의 메서드//
public void setMebmer(Member member){
   this.member = member;
   member.getOrders().add(this);
}

public void addOrderItem(OrderItem orderItem){
   orderItems.add(orderItem);
   orderItem.setOrder(this);
}

public void setDelivery(Delivery delivery){
   this.delivery = delivery;
   delivery.setOrder(this);
}
```