JPA ORM Chapter 9

Bangjang

JPA의 데이터 타입

엔티티 타입	값 타입
@Entity	기본 타입 / 객체
식별자를 통한 추적	숫자 / 문자같은 속성

기본값 타입

- String, int
- 식별자 값 X, 생명주기 엔티티 의존
- 공유 불가

임베디드 타입(복합 값 타입)

- 새로운 값 타입을 직접 정의하여 사용
- 임베디드 타입도 값 타입
- 엔티티를 의미 있고 응집도 높게 만듬
- **컴포지션 관계** (엔티티 - 임베디드 관계)

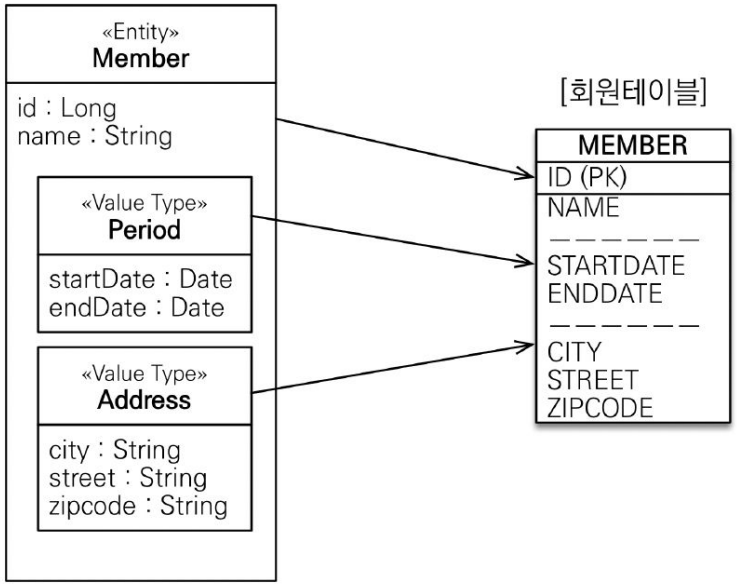
```java
@Entity
@Data
public class Member {
    @Id @GeneratedValue
    private Long id;
    private String name;

    @Embedded
    Period workPeriod;

    @Embedded
    Address homeAddress;
}
```

```java
@Embeddable
@Getter @Setter
public class Address {
    @Column(name = "city")
    private String city;
    private String street;
    private String zipcode;
}
```

```java
@Embeddable
@Getter @Setter
public class Period {
    @Temporal(TemporalType.DATE)
    Date startDate;
    @Temporal(TemporalType.DATE)
    Date endDate;

    public boolean isWork(Date date) {
        return date.after(startDate) && date.before(endDate);
    }
}
```

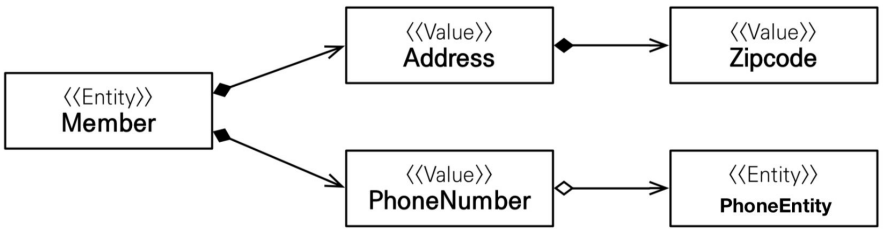
임베디드 타입과 테이블 매핑

- 임베디드 사용 전과 후의 매핑 테이블은 동일
- 객체와 테이블의 세밀한 매핑
- 잘 설계한 ORM 어플리케이션은 테이블 < 클래스
- 객체지향적 모델 설계



임베디드 타입과 연관관계

- 값 타입 포함 / 엔티티 참조 가능
- 값 타입 Address가 값 타입 Zipcode 참조
- 값 타입 PhoneNumber가 엔티티 타입 참조



임베디드 타입 속성 재정의

- @AttributeOverride로 매핑명 재정의
- 중복 시 @AttributeOverrides
- 지저분한 코드, 허나 희귀한 케이스

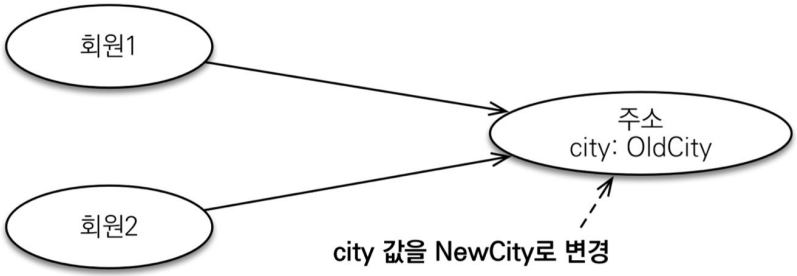
```
@Embedded
Address homeAddress;

@Embedded
@AttributeOverrides({
        @AttributeOverride(name = "city", column = @Column(name = "COMPANY_CITY")),
        @AttributeOverride(name = "street", column = @Column(name = "COMPANY_STREET")),
        @AttributeOverride(name = "zipcode", column = @Column(name = "COMPANY_ZIPCODE"))
})
Address companyAddress;
```

임베디드 타입과 Null

- 임베디드 타입이 null이면 매핑한 컬럼 값은 모두 null이 됨

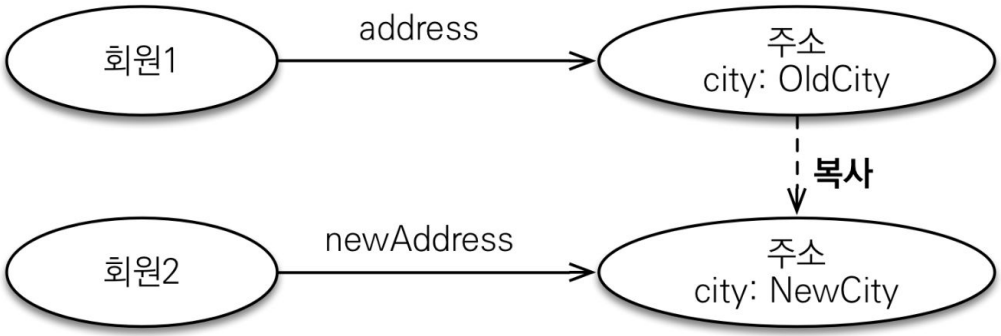
값 타입 공유 참조

- 값 타입 공유는 위험
- 같은 인스턴스 참조 - 변경 위험 부작용



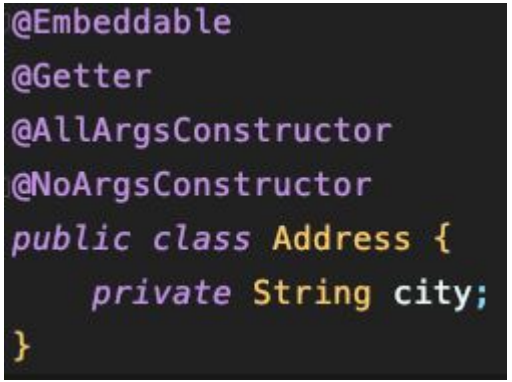
값 타입 복사

- 실제 인스턴스 공유 대신 값 복사
- clone 메소드를 사용해 복사
- 객체 타입 원본 참조 값은 방어 불가
- 객체 공유 참조 피할 수 없음



불변 객체

- 불변값(조회 가능, 수정 불가)
- 부작용 원천 차단
- 될 수 있으면 불변 객체로 설계

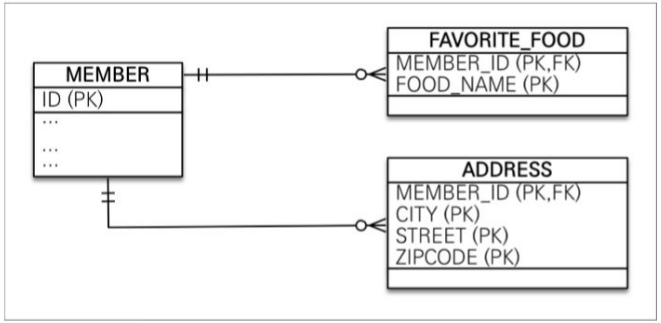


값 타입의 비교

- 동일성 비교 : 인스턴스 참조 값 비교 (==)
- 동등성 비교 : 인스턴스 값 비교 (equals())
- 값 타입 비교에는 동등성 비교 사용

값 타입 컬렉션

- 하나 이상의 값 타입 지정 시 Collection
- @ElementCollection, @CollectionTable
- RDBMS는 컬럼에 Collection 포함 불가
- 추가 테이블 매핑 / 컬럼명 지정



값 타입 컬렉션 사용

- 한 번의 영속화로 값 타입 저장
- 한 번의 Persist 호출로 다수의 SQL 실행
- 조회 시 Fetch 전략 선택 가능

값 타입 컬렉션의 제약사항

- 식별자가 없어 값 변경 시 DB 원본 데이터 찾기 어려움
- 값 타입 컬렉션 변경 시 연관 테이블 데이터 전부 삭제 후 다시 저장
- 실무에서는 값 타입 컬렉션 대신 일대다 관계 고려 필요