8 프록시

1. 프록시와 즉시로딩, 지연로딩 : 객체는 객체 그래프로 연관된 객체들을 탐색한다. 그런데 객체가 데이터베이스에 저장되어 있으므로 연관된 객체를 마음걱 탐색하기는 어렵다. JPA 구현체들은 이 문제를 해결하려고 프록시라는 기술을 사용한다. 프록시를 사용하면 연관된 객체를 처음부터 데이터베이스에서 조회하는 것이 아니라 실제 사용하는 시점에 데이터베이스에서 조회할 수 있다. 하지만 자주함께 사용하는 객체들은 조인을 사용해서 함께 조회하는 것이 효과적이다 JPA는 즉시 로딩과 지연 로딩이라는 방법으로 둘을 모두 지원한다.
2. 영속성 전이와 고아 객체 : JPA는 연관된 갹체를 함께 저장하거나 함께 삭제할 수 있는 영속성 전이와 고아 객체 제거라는 편리한 기능을 제공한다.

8.1.1 프록시 기초

JPA에서 식별자로 엔티티 하나를 조회할 때는 EntityManager.find()를 사용한다 이 메소드는 영속성 컨텍스트에 엔티티가 없으면 데이터베이스를 조회한다.

이렇게 엔티티를 직접 조회하면 엔티티를 실제 사용하든 사용하지 않든 데이터베이스를 조회하게 된다.엔티티를 실제 사용하는 시점까지 데이터베이스 조회를 미루고 싶드면 EntityManager.getReference() 메소드를 사용하면 된다.

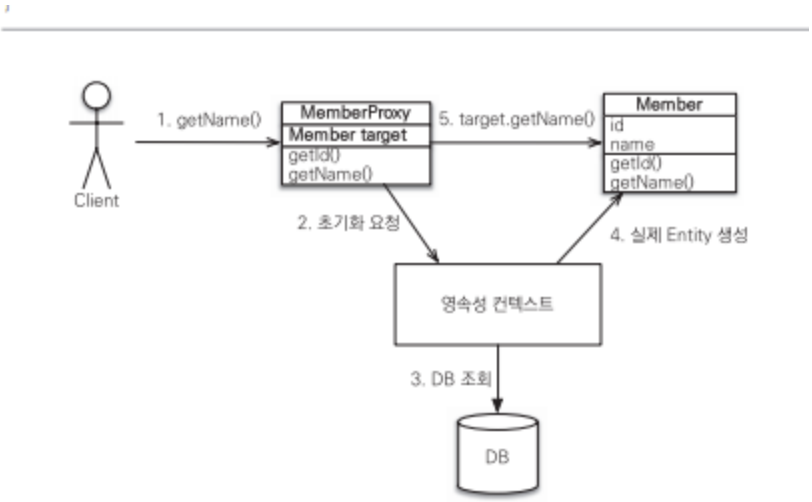
이 메소드를 호출할 때 JPA는 데이터베이스를 조회하지 않고 실제 엔티티 객체도 생성하지 않는다 . 대신에 데이터베이스 접근을 위임한 프록시 객체를 반환한다.

프록시 특징

프록시 클래스는 실제 클래스를 상속 받아서 만들어지므로 실제 클래스와 겉 모양이 같다. 따라서 사용하는 입장에서 이것이 진짜 객체인지 프록시 객체인지 구분하지 않고 사용하면 된다.

▼ 프록시 객체의 초기와

프록시 객체는 실제 사용될 때 데이터베이스를 조회해서 실제 엔티티 객체를 생성하는데 이것을 프록시 객체의 초기화라 한다.



프록시 특징

1. 프록시 객체는 처음사용할 때 한 번만 초기화된다.
2. 프록시 객체를 초기화한다고 프록시 객체가 실제 엔티티로 바뀌는 것은 아니다. 프록시 객체가 초기화되면 프록시 객체를 통해서 실제 엔티티에 접슨할 수 있다
3. 프록시 객체는 원본 엔티티를 상속받은 객체이므로 타입 체크 시에 주의해서 아용해야 한다.
4. 영속성 컨텍스트에 찾는 엔티티가 이미 있으면 데이터베이스를 조회할 필요가 없으므로 em.getReference()를 호출해도 프록시가 아닌 실제 엔티티를 반환한다.
5. 초기화는 영속성 컨텍스트의 도움을 받아야 가능하다. 따라서 영속성 컨텍스트의 도움을 받을 수 없는 준 영속 상태의 프록시를 초기화하면 문제가 발생한다.

8.1.2 프록시와 식별자

엔티티를 프록시로 조회할 때 식별자 (pk) 값을 파라미터로 전달하는데 프록시 객체는 이 식별자 값을 보관한다.

프록시 객체는 식별자 값을 가지고 있으므로 식별자 값을 조회하는 team.getId()를 호출해도 프록시를 초기화하지 않는다. 단 엔티티 접근 방식을 프로퍼티로 설정한 경우에만 초기화하지 않는다.

연관관계를 설정할 때는 식별자 값만 사용하므로 프록시를 사용하면 데이터베이스 접근 횟수를 줄일 수 있다 참고로 연관관계를 설정할 때는 엔티티 접근 방식을 필드로 설정해도 프록시를 초

기화하지 않는다.

8.1.3 프록시 확인

JPA가 제공하는 PersistenceUnitUtil.isLoaded 메소드를 사용하면 프록시 인스턴스 초기화 여부를 확인할 수 있다. 아직 초기화되지 않은 프록시 인스턴스는 false를 반환한다. 이미 초기화되었거나 프록시 인스턴스가 아니면 true를 반환한다. 조회한 엔티티가 진짜 엔티티인지 프록시로 조회한 것인지 확인 하려면 클래스명을 직접 출력해보면 된다.

프록시 강제 초기화

하이버네이트의 initialize() 메소드를 사용하면 프록시를 강제로 초기화할 수 있다.

JPA 표준에는 프록시 강제 초기화 메소드가 없다.따라서 강제로 초기화하려면 member.getName()처럼 프록시의메소드를 직접 호출하면 된다.

JPA표준은 단지 초기화 여부만 확인할 수 있다.

8.2 즉시 로딩과 지연 로딩

프록시 객체는 주로 연관된 엔티티를 지연 로딩할 때 사용한다.

JPA는 개발자가 연관된 엔티티의 조회 시점을 선택할 수 있돍 다음 두가지 방법을 제공한다

- 즉시 로딩 : 엔티티를 조회할 때 연관된 엔티티도 함께 조회한다.

  ▼ 예 : find를 호출할대 회원 엔티티와 연관된 팀 엔티티도 함게 조회한다

  ▼ 설정 방법 : @ManyToOne(fetch = FetchType.EAGER)

- 지연 로딩 : 연관된 엔티티를 실제 사용할 때 조회한다.

  ▼ 조회한 팀 엔티티를 실제 사용하는 시점에 JPA가 SQL을 호출해서 팀 엔티티를 조회한다

  ▼ 설정 방법 : @ManyToOne(fetch = FetchType.LAZY)

8.2.1 즉시로딩

즉시 로딩을 사용하려면 @ManyToOne의 fetch속성을 FetchType.EAGE 지정한다.대부분의 JPA 구현체는 즉시 로딩을 최적화하기 위해 가능하면 조인쿼리를 사용한다.

8.2.2 지연로딩

지연 로딩을 사용하려면 @ManyToOne(fetch = FetchType.LAZY) 지정한다.

예제 8.10 지연 로딩 설정

```
@Entity
public class Member {
    //...
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "TEAM_ID")
    private Team team;
    //...
}
```

예제 8.11 지연 로딩 실행 코드

```
Member member = em.find(Member.class, "member1");
Team team = member.getTeam(); //객체 그래프 탐색
team.getName(); //팀 객체 실제 사용
```

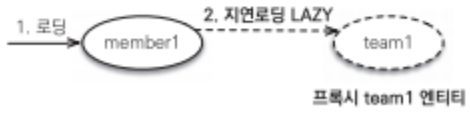


그림 8.6 지연 로딩, 회원을 조회할 때 팀 지연 로딩

em.find(Member.class,"member1")를 호출하면 회원만 조회하고 팀은 조회하지 않는다 대신 즉시 조회한 회원의 team 멤버 변수에 프록시 객체를 넣어둔다

반환된 팀 객체는 프록시 객체다 이 프록시 객체는 실제 사용될 때가지 데이터 로딩을 미룬다 . 그래서 지연 로딩이라 한다.

참고

조회 대상이 영속성 컨텍스트에 이미 있으면 프록시 객체를 사용할 이유가 없다 . 따라서 프록시가 아닌 실제 객체를 사용한다.

예를들어 team1 엔티티가 영속성 컨텍스트에 이미 로딩되어 있으면 프록시가 아닌 실제 엔티티를 사용한다.

8.2.3 즉시 로딩 , 지연 로딩 정리

처음부터 연관된 엔티티를 모두 영속성 컨텍스트에 올려두는 것은 현실적이지 않고 필요할때 sql을 실행해서 연관된 엔티티를 지연 로하는 것도 최적화 관점에서 보면 꼭 좋은것은 아니다

지연로딩 : 연관된 엔티티를 프록시로 조회한다. 프록시를 실제 사용할 때 초기화하면서 데이터베이스를 조회한다.

즉시로딩 : 연관된 엔티티를 즉시 조회한다 . 하이버네이트는 가능하면 sql조인을 사용해서 한번에 조회한다.

8.3.1 프록시와 컬렉션 래퍼

지연로딩으로 설정하면 실제 엔티티 대신에 프록시 객체를 사용한다.

프록시 객체는 실제 자신이 사용될 때까지 데이터베이스를 조회하지 않는다

컬렉션 래퍼란 하이버데이트는 엔티티를 영속 상태로 만들 때 엔티티에 컬렉션이 있으면 컬렉션을 추적하고 관리할 목적으로 원본 컬렉션을 하이버네이트가 제공하는 내장 컬렉션으로 변경하는데 이것을 컬렉션 래퍼라 한다.

엔티티를 지연 로딩하면 프록시 객체를 사용해서 지연 로딩을 수행하지만 주문 내역 같은 컬렉션은 컬렉션 래퍼가 지연 로딩을 처리해준다.

참고로 member.getOrders()를 호출해도 컬렉션은 초기화되지 않는다 컬렉션은 member.getOrders().get(0)처럼 컬렉션에서 실제 데이터를 조회할 때 데이터베이스를 조회해서 초기화한다.

8.3.2 JPA 기본 페치 전략

@ManyToOne, @OneToOne : 즉시로딩 (FetchType.EAGER)

@OneToMany, @ManyToMany : 지연로딩 (FetchType.LAZY)

JPA의 기본 페치 전략은 연관된 엔티티가 하나면 즉시 로딩을, 컬렉션이면 지연 로딩을 사용한다. 추천하는 방법은 모든 연관관계에 지연 로딩을 사용하는 것이다. 실제 사용하는 상황을 보고 꼭 필요한 곳에만 즉시 로딩을 사용하도록 최적화하면 된다.

8.3.3 컬렉션에 FetchType.EAGER 사용 시 주의점

1. 컬렉션을 하나 이상 즉시 로딩하는 것은 권장하지 않는다.

로 다른 컬렉션을 2개 이상 조인할때 n x m 이 되면서 너무 많은 데이터를 반환할 수 있고 결과적으로 어플리케이션 성능이 저하될 수 있다 따라서

2개 이상의 컬렉션을 즉시 로딩으로 설정하는 것은 권장하지 않는다

1. 컬렉션 즉시 로딩은 항상 외부 조인을 사용한다.

8.4 영속성 전이 : cascade

특정 엔티티를 영속 상태로 만들 때 연관된 엔티티도 함께 영속 상태로 만들고 싶으면 영속성 전이 기능을 사용하면 된다. jpa는 cascade옵션으로 영속성 전이를 제공한다. 영속성 전이를 사용하면 부모 엔티티를 저장할 대 자식 엔티티도 함께 저장할 수 있다.

JPA에서 엔티티를 저장할 때 연관된 모든 엔티티는 영속 상태여야 한다.

8.4.1 영속성 전이 : 저장

영속성 전이를 활성화하는 cascade 옵션을 적용해보자

부모를 영속화할 때 연관된 자식들도 함께 영속화하라고 cascade = CascadeType.PERSIST 옵션을 설정했다.

8.4.2 영속성 전이 : 삭제

영속성 전이는 엔티티를 삭제할 때도 사용할 수 있다 CascadeType.REMOVE로 설정하고 부모 엔티티만 삭제하면 연관된 자식 엔티티도 함께 삭제된다.

삭제 순서는 외래 키 제약조건을 고려해서 자식을 먼저 삭제하고 부모를 삭제한다. CascadeType.REMOVE로 설정하지 않고 실행시 부모 엔티티만 삭제된다.

8.5 고아 객체

JPA는 부모 엔티티와 연관관계가 끊어진 자식 엔티티를 자동으로 삭제하는 기능을 제공하는데 이것을 고아 객체 제거라 한다. 이 기능을 사용해서 부모 엔티티의 컬렉션에서 자식 엔티티의 참조만 제거하면 자식 엔티티가 자동으로 삭제되도록 해보자

고아객체 정리

고아 객체 제거는 참조가 제거된 엔티티는 다른 곳에서 참조하지 않는 고아 객체로 보고 삭제하는 기능이다 따라서 이 기능은 참조하는 곳이 하나일 때만 사용해야 한다 쉽게 이야기해서 특정 엔티티가 개인 소유하는 엔티티에만 이 기능을 적용해야 한다. 만약 삭제한 엔티티를 다른곳에서도 참조한다면 문ㄴ제가 발생할 수 있다 이런 이유로 orphanRemovel은 @OneToOne , @OneToMany에만 사용할 수 있다

정리

1.JPA 구현체들은 객체 그래프를 마음껏 탐색할 수 있도록 지원하는데 이때 프록시 기술을 사용한다.

1. 객체를 조회할 때 연관된 객체를 즉시 로딩하는 방법을 즉시 로딩이라 하고 연관된 객체를 지연해서 로딩하는 방법을 지연 로딩이라 한다.

2. 객체를 저장하거나 삭제할 때 연관된 객체도 함께 저장하거나 삭제할 수 있는데 이것을 영속성 전이라 한다.
3. 부모 엔티티와 연관관계가 끊어진 자식 엔티티를 자동으로 삭제하려면 고아 객체 제거 기능을 사용하면 된다