연관관계매핑

> 본 내용은 김영한님의 인프런 강의와 'JPA 프로그래밍' 책을 정리한 내용입니다.

> **목표: 객체 참조와 외래키 매핑의 차이**

용어 정리

- **방향**: 단방향, 양방향 // **테이블 관계는 항상 양방향**
- **다중성**: 일대일, 일대다, 다대일, 다대다
- **연관관계의 주인**: 외래키를 가진 테이블이 연관관계의 주인이 된다.

> 모든 Entity 테이블은는 양방향 관계를 가지지만, 객체는 단방향 관계를 가진다.
> 객체는 참조하고 싶은 다른 객체를 선언하여 단방향 연관관계를 가지지만, 테이블은 한 테이블을 참조하고자 조인을 통해 관계를 맺을 때, 양방향 연관관계를 가지게 된다.

1. 단방향 연관관계

- **Entity**는 객체이지만 동시에 **JPA를 통해 테이블로서 동작**한다는 점에 유의하자.
- 객체는 다른 객체와 단방향 연관관계를 가지기 때문에 한 객체가 다른 객체를 소속 멤버로 가진다고 해서 다른 객체가 소속 멤버로 지정한 객체를 자신의 소속 멤버로 가지는 것은 아니다.
- **Entity인 테이블은 한 객체가 다른 객체를 소속멤버로 가진다면 이는 조인**한 것이고, 테이블로서 관계를 맺기 때문에 **양방향 관계**를 맺는다고 할 수 있다.
    - 테이블은 외래키로 연관관계를 맺는다.
    - 외래키를 가지는 쪽이 주인이다.

1) 객체관계 매핑

`@JoinColumn`이 가질 수 있는 속성

속성	기능	기본 값
`name`	매핑할 외래 키 이름	필드명_참조하는 테이블의 기본키 (`team_TEAM_ID`)
`referencedColumnName`	외래 키가 참조하는 테이블의 컬럼명	참조하는 테이블의 기본키(`team_id`)
foreignKey(DDL)	외래키 제약조건을 직접 지정할 수 있다.	

`unique`
`nullable`
`insertable`
`updatable`
`columnDefinition`
`table`
	`@Column` 속성과 같다.	

`@ManyToOne`이 가질 수 있는 속성 - 8장에서 이어짐

- `Optional`: `false`로 설정할 경우 연관된 엔티티가 반드시 있어야 한다. (`true`일 경우 일대다의 관계를 가지는 엔티티가 존재할 수도 존재하지 않을 수도 있다.)
- `fetch`
- `cascade`
- `targetEntity`: 거의 사용하지 않는다.

```
package study.datajpa.entity;

import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;

@Entity // 객체임을 명시 (테이블)
@Getter @Setter

public class Member {

    // Column
    @Id @GeneratedValue // Primary
    @Column(name = "member_id")
    private Long id;
    private String username;
    private int age;

    // 연관관계 매핑 - Relationship (Join Column)
    // 관계를 맺는 Entity에도 반대 관계 설정 즉, @OneToMany
    // Foreign Key가 없는 쪽에 mappedBy 설정
    @ManyToOne
    @JoinColumn(name = "team_id")
    private Team team;

    protected Member() {}
}
```

```
package study.datajpa.entity;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
public class Team {

    @Id
    @GeneratedValue
    @Column(name = "team_id")
```

```
    private Long id;
    private String name;

    @OneToMany(mappedBy = "team") //조인을 당하는 테이블
    private List<Member> members = new ArrayList<>();

}
```

---

2. 연관관계의 사용

- 등록(C), 조회(Read), 수정(U), 삭제(D)

3. 양방향 연관관계

- 테이블은 외래 키 하나로 두 테이블의 연관관계를 관리한다.
- 외래키를 가지고 있는 쪽(관리하는 쪽)이 연관관계의 주인이다.
- 위 예제에서 멤버가 팀의 외래키를 가지고 있기 때문에 연관관계의 주인이 된다.
- 다대일 관계에서 다에 속하는 쪽이 연관관게의 주인이다. (ManyToOne에서 Many에 속하는 쪽)
    - 이때, 일에 속하는 테이블은 `mapped by`라는 속성을 통해 어떤 컬럼이 자신을 관리하고 있는지를 명시해주어야 한다.
    - 연관관계의 주인만이 데이터베이스 연관관계와 매핑된다.
    - 외래키를 관리(등록, 수정, 삭제) 할 수 있다.
    - 주인이 아닌 쪽은 읽기만 할 수 있다.

1) 주의해야 하는 것

- 양방향 연관관계의 주인에 값을 입력해아한다.
- JPA는 객체를 테이블로 여겨지게끔 해주는 인터페이스이지만, 객체의 특성을 여전히 가지기 때문에 주인에 값을 입력했다면 주인이 아닌 객체에도 값을 넣어주는 것이 좋다.
- 즉, **양쪽 방향 모두에 값을 입력해주는 것이 안전한 방법이다**
  Q. 양방향 연관관계의 주인에 값을 입력하면, 주인이 아닌 곳에 값이 자동으로 입력될까?

Test에서 팀을 넣어줄 때

```
public void testORM_bothSide() {
    // 팀1 저장
    Team team1 = new Team("team1");
    em.persist(team1);

    Member member1 = new Member("member1", "20"); //name, age

    // 양방향 관계 설정
    member1.setTeam(team1);
    team1.getMembers().add(member1);
    em.persist(member1);
}
```

Entity 내에서 팀을 넣어줄 때 (권장)

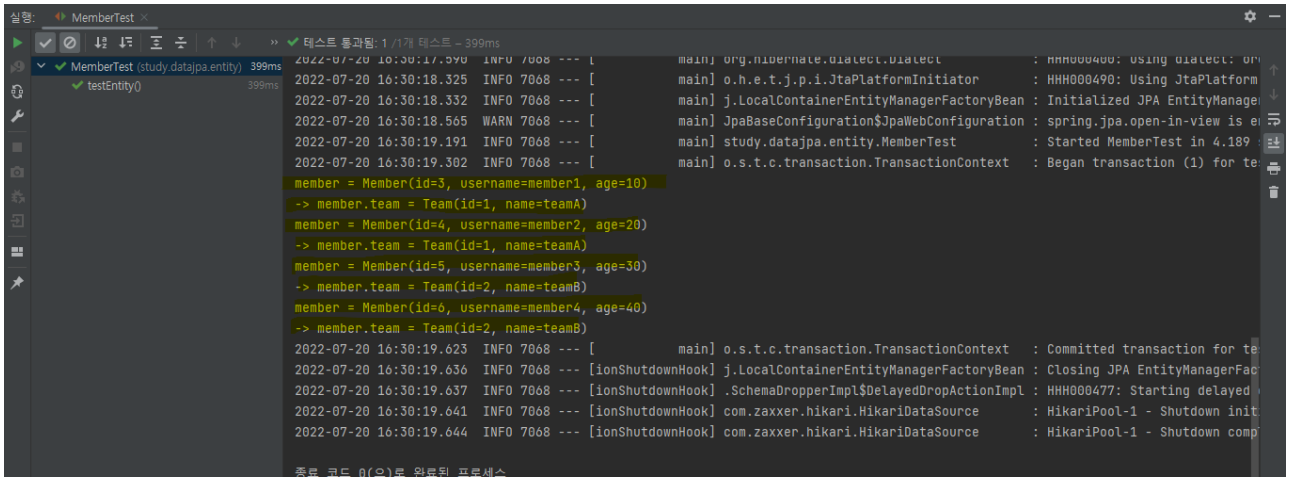
```
package study.datajpa.entity;

import lombok.*;

import javax.persistence.*;

@Entity // 객체임을 명시 (테이블)
@Getter @Setter
@NoArgsConstructor (access = AccessLevel.PROTECTED)
@ToString(of = {"id", "username", "age"})
// 연관관계를 가진 대상은 ToString으로 나타낼 경우 무한루프에 빠질 위험이 있어 ToString
의 대상이 아니다.
public class Member {

    @Id @GeneratedValue // Primary key로서 id를 생성하겠다.
    @Column(name = "member_id")
    private Long id;
    private String username;
    private int age;

    @ManyToOne
    @JoinColumn(name = "team_id")
    private Team team;

    public Member(String username, int age, Team team) {
        this.username = username;
        this.age = age;
        if(team != null)
            changeTeam(team);
    }

    // 양방향 관계 설정 - 팀 설정을 해주는 메소드
    // (setter에서는 Member의 팀만 바꿔주기 때문에 따로 메소드로 생성)
    public void changeTeam(Team team) {
        this.team = team;
        team.getMembers().add(this);
    }
}
```

- 본 케이스에서의 Test Case

```
package study.datajpa.entity;

import lombok.*;

import javax.persistence.*;
```

```java
@Entity // 객체임을 명시 (테이블)
@Getter @Setter
@NoArgsConstructor (access = AccessLevel.PROTECTED)
@ToString(of = {"id", "username", "age"})
// 연관관계를 가진 대상은 ToString으로 나타낼 경우 무한루프에 빠질 위험이 있어 ToString
의 대상이 아니다.
public class Member {

    @Id @GeneratedValue // Primary key로서 id를 생성하겠다.
    @Column(name = "member_id")
    private Long id;
    private String username;
    private int age;

    // 일대 다 관계에서는 ManyToOne의 fetch Type을 LAZY로 설정하는 것이 관례이다.
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "team_id")
    private Team team;

    public Member(String username, int age, Team team) {
        this.username = username;
        this.age = age;
        if(team != null)
            changeTeam(team);
    }

    // 양방향 관계 설정 - 팀 설정을 해주는 메소드
    // (setter에서는 Member의 팀만 바꿔주기 때문에 따로 메소드로 생성)
    public void changeTeam(Team team) {
        // 기존에 팀이 존재했는지를 먼저 확인하는 과정이 필요하다.
        if(this.team != null)
            this.team.getMembers().remove(this);

        this.team = team;
        team.getMembers().add(this);
    }
}
```

- 결과


실행: MemberTest
테스트 통과됨: 1 /1개 테스트 – 399ms
MemberTest (study.datajpa.entity) 399ms
testEntity() 399ms
2022-07-20 16:30:18.325 INFO 7068 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform
2022-07-20 16:30:18.332 INFO 7068 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManage
2022-07-20 16:30:18.565 WARN 7068 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is e
2022-07-20 16:30:19.191 INFO 7068 --- [main] study.datajpa.entity.MemberTest : Started MemberTest in 4.189
2022-07-20 16:30:19.302 INFO 7068 --- [main] o.s.t.c.transaction.TransactionContext : Began transaction (1) for te
member = Member(id=3, username=member1, age=10)
-> member.team = Team(id=1, name=teamA)
member = Member(id=4, username=member2, age=20)
-> member.team = Team(id=1, name=teamA)
member = Member(id=5, username=member3, age=30)
-> member.team = Team(id=2, name=teamB)
member = Member(id=6, username=member4, age=40)
-> member.team = Team(id=2, name=teamB)
2022-07-20 16:30:19.623 INFO 7068 --- [main] o.s.t.c.transaction.TransactionContext : Committed transaction for te
2022-07-20 16:30:19.636 INFO 7068 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFac
2022-07-20 16:30:19.637 INFO 7068 --- [ionShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Starting delayed
2022-07-20 16:30:19.641 INFO 7068 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown init
2022-07-20 16:30:19.644 INFO 7068 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown comp
종료 코드 0(으)로 완료된 프로세스