

9-1. 프록시

멤버 객체를 조회할 때 팀도 함께 조회해야하나??

Member를 조회할 때 Team도 함께 조회해야 할까?

회원과 팀 함께 출력

```
public void printUserAndTeam(String memberId) {
    Member member = em.find(Member.class, memberId);
    Team team = member.getTeam();
    System.out.println("회원 이름: " + member.getUsername());
    System.out.println("소속팀: " + team.getName());
}
```

회원만 출력

```
public void printUser(String memberId) {
    Member member = em.find(Member.class, memberId);
    Team team = member.getTeam();
    System.out.println("회원 이름: " + member.getUsername());
}
```

```
package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class JpaMain {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        try {
```

```
Member member = em.find(Member.class, 1L); // 여기서 한번에 팀과 멤버를 가져오면  
좋다!
```

```
printMemberAndTeam(member);  
printMember(member);
```

```
tx.commit();
```

```
} catch (Exception e) {  
    tx.rollback();  
}  
finally {  
    em.close();  
}
```

```
emf.close();
```

```
}
```

```
private static void printMemberAndTeam(Member member) {  
    String username = member.getName();  
    System.out.println("username = " + username);  
  
    Team team = member.getTeam();  
    System.out.println("team.getName() = " + team.getName());  
}
```

```
private static void printMember(Member member) {  
    String username = member.getName();  
    System.out.println("username = " + username);  
}
```

```
}
```

`printMember(member)`, `printMemberAndTeam(member)` 를 비교해보면, 어떤 메소드는 멤버만, 어떤 메소드는 멤버와 팀 모두를 조회하고 싶어한다.

JPA는 이것을 지연 로딩과 프록시로 기가막히게 해결한다.

프록시

- `em.getReference()` : DB에 쿼리가 안나가는데 객체가 조회가 되는 가짜 엔티티 객체 조회 메소드
- `em.find()` : DB에 쿼리를 내보내면서 실제 엔티티 객체를 조회하는 메소드

```
package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class JpaMain {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        try {

            Member member = new Member();
            member.setName("hello");

            em.persist(member);

            em.flush();
            em.clear(); // 영속성 컨텍스트 초기화. 거의 애플리케이션 처음 띄운것과 마찬가지..

            Member findMember = em.find(Member.class, member.getId());
```

```
        System.out.println("findMember.getId() = " + findMember.getId());
        System.out.println("findMember.getName() = " + findMember.getName());

        tx.commit();

    } catch (Exception e) {
        tx.rollback();
    } finally {
        em.close();
    }

    emf.close();

}

private static void printMemberAndTeam(Member member) {
    String username = member.getName();
    System.out.println("username = " + username);

    Team team = member.getTeam();
    System.out.println("team.getName() = " + team.getName());
}

}
```

```
JpaMain x
Member
(createdBy, createdAt, lastModifiedBy, lastModifiedDate, USERNAME, TEAM_ID, id)
values
(?, ?, ?, ?, ?, ?, ?)
Hibernate:
select
member0_.id as id1_1_0_,
member0_.createdBy as created82_1_0_,
member0_.createdAt as createdD3_1_0_,
member0_.lastModifiedBy as lastModi4_1_0_,
member0_.lastModifiedDate as lastModi5_1_0_,
member0_.USERNAME as USERNAME6_1_0_,
member0_.TEAM_ID as TEAM_ID7_1_0_,
team1_.id as id1_5_1_,
team1_.name as name2_5_1_
from
Member member0_
left outer join
Team team1_
on member0_.TEAM_ID=team1_.id
where
member0_.id=?
findMember.getId() = 1
findMember.getName() = hello
6월 14, 2023 2:25:21 오전 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:h2:tcp://localhost/~test2]

Process finished with exit code 0
```

jpa가 쿼리를 내보내서 멤버와 팀을 가지고 조회함. (find를 썼을 때)

```
package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class JpaMain {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        try {
```

```

Member member = new Member();
member.setName("hello");

em.persist(member);

em.flush();
em.clear(); // 영속성 컨텍스트 초기화. 거의 애플리케이션 처음 띄운것과 마찬가지로..

//      Member findMember = em.find(Member.class, member.getId());
Member findMember = em.getReference(Member.class, member.getId());
//      System.out.println("findMember.getId() = " + findMember.getId());
//      System.out.println("findMember.getName() = " + findMember.getName());

tx.commit();

} catch (Exception e) {
    tx.rollback();
} finally {
    em.close();
}

emf.close();

}

private static void printMemberAndTeam(Member member) {
    String username = member.getName();
    System.out.println("username = " + username);

    Team team = member.getTeam();
    System.out.println("team.getName() = " + team.getName());
}
}

```

em.getReference() 만 했을 때는

```

        foreign key (Member_id)
        references Member
Hibernate:

        alter table MemberProduct
        add constraint FKrgt6jorh7iaec1tae84ljye8c
        foreign key (MEMBER_ID)
        references Member
Hibernate:

        alter table MemberProduct
        add constraint FKrgt6jorh7iaec1tae84ljye8c
        foreign key (PRODUCT_ID)
        references Product
6월 14, 2023 2:29:28 오전 org.hibernate.tool.schema.internal.SchemaCreatorImpl applyImportSources
INFO: HHH000476: Executing import script 'org.hibernate.tool.schema.internal.exec.ScriptSourceInputNonExistentImpl@2caa5d7c'
Hibernate:
        call next value for hibernate_sequence
Hibernate:
        /* insert hellojpa.Member
        */ insert
        into
            Member
            (createdBy, createdDate, lastModifiedBy, lastModifiedDate, USERNAME, TEAM_ID, id)
        values
            (?, ?, ?, ?, ?, ?, ?)
6월 14, 2023 2:29:28 오전 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:h2:tcp://localhost/~test2]

Process finished with exit code 0

```

insert 쿼리만 나감...

근데 밑에 있는 sout의 주석을 벗기고 실행하면

```

JpaMain x
        Member
        (createdBy, createdDate, lastModifiedBy, lastModifiedDate, USERNAME, TEAM_ID, id)
        values
            (?, ?, ?, ?, ?, ?, ?)
findMember.getId() = 1
Hibernate:
        select
            member0_.id as id1_1_0_,
            member0_.createdBy as createdB2_1_0_,
            member0_.createdDate as createdD3_1_0_,
            member0_.lastModifiedBy as lastModi4_1_0_,
            member0_.lastModifiedDate as lastModi5_1_0_,
            member0_.USERNAME as USERNAME6_1_0_,
            member0_.TEAM_ID as TEAM_ID7_1_0_,
            team1_.id as id1_5_1_,
            team1_.name as name2_5_1_
        from
            Member member0_
        left outer join
            Team team1_
            on member0_.TEAM_ID=team1_.id
        where
            member0_.id=?
findMember.getName() = hello
6월 14, 2023 2:30:38 오전 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:h2:tcp://localhost/~test2]

Process finished with exit code 0

```

위처럼 쿼리가 나간다...

getReference() 가 호출되는 그 시점에는 쿼리가 나가지 않으나, 그걸로 가져온 것을 사용할 때는 쿼리를 내보낸다.

id를 가져올때는 쿼리가 안나갔다. 왜냐면 findMember를 찾을 때는 쿼리를 내놓지 않아도 되어서 디비까

지 가지 않아도 되었다.

id는 파라미터로 넣은 것이므로 디비까지 안가도됨

근데 getName같은 경우는 디비에 있기 때문에 가짜 레퍼런스를 가져온 것. 그래서 jpa는 가짜 객체를 디비에서 읽어오기 위해 insert쿼리를 날림. 그래서 getName을 실행할 때 jpa가 db에 쿼리를 날린다!!

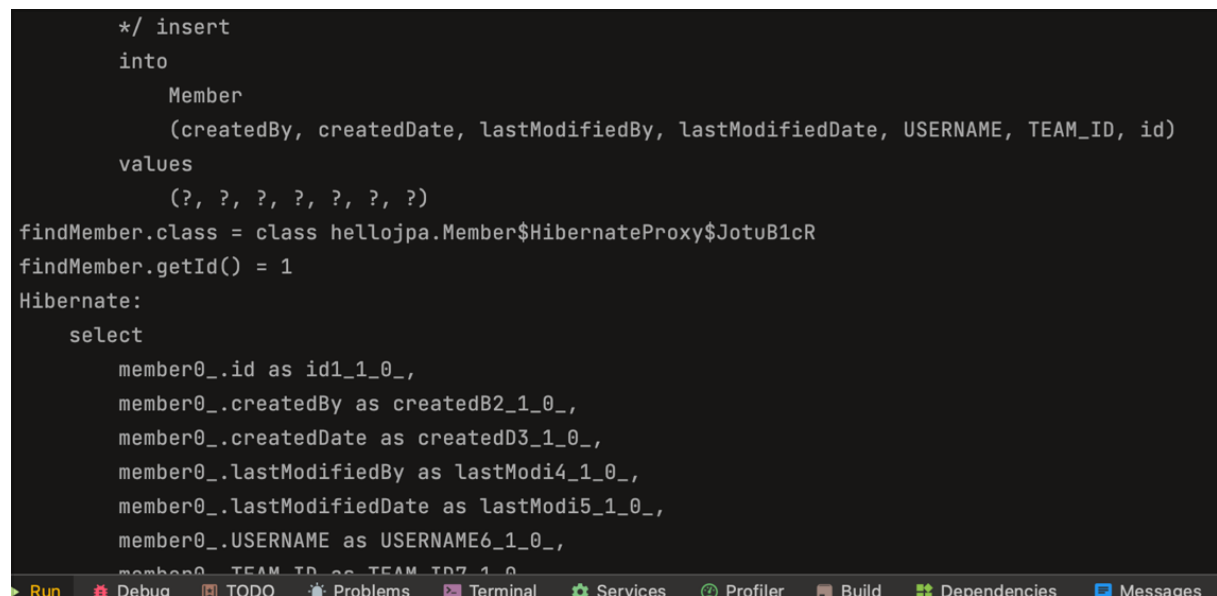
이렇게 되면 db에서 가져온 데이터로 가짜 레퍼런스의 값을 채워서 findMember에 집어넣는다.

근데 값을 채운다는건 너무 간략한 표현임....

더 정확하게 알아보자

그렇다면 과연 `getReference()` 로 불러온 findMember의 정체는 무엇일까

```
System.out.println("findMember.class = " + findMember.getClass());
```



```

    */ insert
    into
        Member
        (createdBy, createdDate, lastModifiedBy, lastModifiedDate, USERNAME, TEAM_ID, id)
    values
        (?, ?, ?, ?, ?, ?, ?)
findMember.class = class hellojpa.Member$HibernateProxy$JotuB1cR
findMember.getId() = 1
Hibernate:
    select
        member0_.id as id1_1_0_,
        member0_.createdBy as createdB2_1_0_,
        member0_.createdDate as createdD3_1_0_,
        member0_.lastModifiedBy as lastModi4_1_0_,
        member0_.lastModifiedDate as lastModi5_1_0_,
        member0_.USERNAME as USERNAME6_1_0_,
        member0_.TEAM_ID as TEAM_ID7_1_0_

```

클래스를 보면 하이버네이트가 강제로 만든 가짜 클래스라는 것!!!

이게 바로 프록시 클래스다....

em.find를 하면 실제 객체를 주고

em.getReference를 하면 가짜 객체를 준다.

proxy : 껍데기는 똑같은데 안에가 텅텅 빈 가짜 객체. 내부에는 target이 있는데 이게 진짜 객체를 가리킴.

hibernate가 내부적으로 라이브러리를 사용해서 실제 클래스를 상속받아서 만들어짐. 그래서 실제 클래스와 겉모양이 같다..

사용하는 입장에서는 진짜인지 아닌지 구분하지 않아도됨.

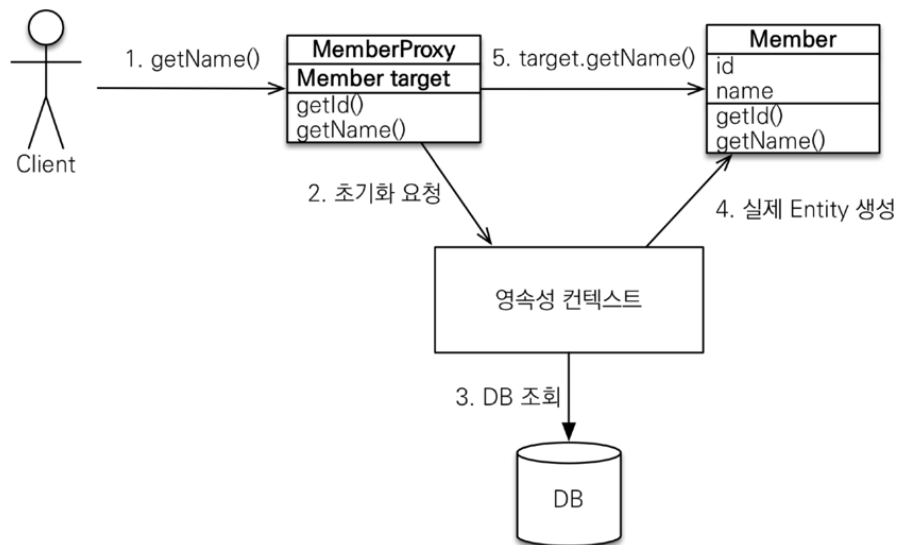
프록시 특징

프록시 객체는 실제 객체의 참조(target)를 보관함.

처음에(Member를 조회하기 전)는 target에 실제 객체의 참조가 없을수밖에없다...

프록시 객체의 초기화

```
Member member = em.getReference(Member.class, "id1");  
member.getName();
```



영속성 컨텍스트를 통해서 프록시 객체의 초기화를 요청한다.

프록시 객체의 초기화 : 프록시에 값이 없을 때 진짜 값을 달라고 하는 것. 디비를 통해 값을 가지고 와서 실제 엔티티를 만들어 내는 것이다.

```

findMember.getId() = 1
Hibernate:
    select
        member0_.id as id1_1_0_,
        member0_.createdBy as createdB2_1_0_,
        member0_.createdDate as createdD3_1_0_,
        member0_.lastModifiedBy as lastModi4_1_0_,
        member0_.lastModifiedDate as lastModi5_1_0_,
        member0_.USERNAME as USERNAME6_1_0_,
        member0_.TEAM_ID as TEAM_ID7_1_0_,
        team1_.id as id1_5_1_,
        team1_.name as name2_5_1_
    from
        Member member0_
    left outer join
        Team team1_
        on member0_.TEAM_ID=team1_.id
    where
        member0_.id=?
findMember.getName() = hello
findMember.getName() = hello
6월 14, 2023 2:45:27 오전 org.hibernate.engine.jdbc.c
INFO: HHH10001008: Cleaning up connection pool [jdb
Process finished with exit code 0

```

만약 getName을 두번 한다면...

역시나 파라미터로 준 Id는 검색하는 쿼리가 나가지 않고,

첫 번째 getName의 경우는 프록시 객체의 초기화를 위해 쿼리를 내보냈지만

두 번째의 경우는 초기화된 프록시 객체로부터 값을 받아왔다고 볼 수 있다.

프록시의 특징

프록시 객체는 처음 사용할 때 **한번만** 초기화된다. 위 사진처럼 두 번 getName을 해도 쿼리가 나가지 않는 것을 생각해보자.

프록시 객체를 초기화할 때, 프록시 객체가 실제 객체로 변하는것이 아니라 프록시 객체를 통해서 실제 엔티티에 접근할 수 있는 것이다.

그냥 자바의 레퍼런스 혹은 C++의 포인터랑 비슷한 느낌으로 보면 되겠다.

target이 있기 때문에 더 이해가 쉽다. target이 어떤 엔티티를 가리킨다!

프록시는 유지가 되고 target에만 값이 채워진다.

프록시 객체는 원본 엔티티를 상속받고 따라서 타입 체크시 주의해야한다.

→ 비교 시 ==가 아닌 instance of를 사용해야한다.

→ JPA에서 타입을 비교할 땐 instance of를 사용하자!!

```
package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

public class JpaMain {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        try {

            Member member1 = new Member();
            member1.setName("member1");
```

```

em.persist(member1);

Member member2 = new Member();
member2.setName("member2");
em.persist(member2);

Member member3 = new Member();
member3.setName("member3");
em.persist(member3);

em.flush();
em.clear();

Member m1 = em.find(Member.class, member1.getId());
Member m2 = em.find(Member.class, member2.getId());
Member m3 = em.getReference(Member.class, member3.getId());

System.out.println("m1 == m2 : " + (m1.getClass() == m2.getClass()));
System.out.println("m1 == m3 : " + (m1.getClass() == m3.getClass()));

em.flush();
em.clear(); // 영속성 컨텍스트 초기화. 거의 애플리케이션 처음 띄운것과 마찬가지..

tx.commit();

} catch (Exception e) {
    tx.rollback();
} finally {
    em.close();
}

emf.close();

}

private static void printMemberAndTeam(Member member) {
    String username = member.getName();

```

```

System.out.println("username = " + username);

Team team = member.getTeam();
System.out.println("team.getName() = " + team.getName());
}

}

```

```

        member0_.USERNAME as USERNAME6_1_0_,
        member0_.TEAM_ID as TEAM_ID7_1_0_,
        team1_.id as id1_5_1_,
        team1_.name as name2_5_1_
    from
        Member member0_
    left outer join
        Team team1_
        on member0_.TEAM_ID=team1_.id
    where
        member0_.id=?
m1 == m2 : true
m1 == m3 : false
6월 14, 2023 2:54:34 오전 org.hibernate.engine.jdbc
INFO: HHH10001008: Cleaning up connection pool
Process finished with exit code 0

```

위처럼 프록시 객체와 일반 엔티티 객체를 ==비교하면 같은 것을 가리키더라도 다르다고 나온다.

영속성 컨텍스트에 찾는 엔티티가 이미 있다면 em.getReference() 를 호출해도 실제 엔티티를 가져온다.

jpa에서는 같은 트랜잭션 안의 같은 영속성 컨텍스트 안에서 == 연산을 조회하면 항상 같다고 나와야 한다.

```
member0_.TEAM_ID as TEAM_ID7_1_0_,
team1_.id as id1_5_1_,
team1_.name as name2_5_1_
from
    Member member0_
left outer join
    Team team1_
        on member0_.TEAM_ID=team1_.id
where
    member0_.id=?
m1 = class hellojpa.Member
m3 = class hellojpa.Member$HibernateProxy$SKG1VA
m4 = class hellojpa.Member
6월 14, 2023 3:03:01 오전 org.hibernate.engine.jdbc
INFO: HHH10001008: Cleaning up connection pool
Process finished with exit code 0
```

```
Member m1 = em.find(Member.class, member1.getId());
Member m2 = em.find(Member.class, member2.getId());
Member m3 = em.getReference(Member.class, member3.getId());
Member m4 = em.getReference(Member.class, member1.getId());

System.out.println("m1 = " + m1.getClass());
System.out.println("m3 = " + m3.getClass());
```

```
System.out.println("m4 = " + m4.getClass());
```

getReference로 했는데 왜 m1과 m4의 클래스가 같나??

- find로 찾았던 것의 Reference를 얻는다면 굳이 db까지 갈 이유가 없음.

영속성 컨텍스트에 올려놓았기 때문에 굳이 디비까지 안가도됨. 원본을 반환하는게 오히려 더 나음

- jpa에선 `a == a : (m1 == reference)` 는 항상 true가 나와야 함.

같은 영속성 컨텍스트 혹은 같은 pk를 가지는 놈이라면 한 트랜잭션 안에서는 true가 나와야 하기 때문. 따라서 프록시로 반환했어도 == 비교했을 때 true를 반환하기 위해서 getReference로 가져와도 영속성 컨텍스트에서 가져오는 것을 보장한다. ↓

만약 m1, m4 모두 getReference로 프록시를 가져왔다면 무조건 같은 프록시로 가져와야 한다. 왜냐면 == 연산 true를 보장하기 위해.

만약 m1이 getReference, m4가 find를 호출한다면

```
Member refMember = em.getReference(Member.class, member1.getId());
System.out.println("refMember.getClass() = " + refMember.getClass());

Member findMember = em.find(Member.class, member1.getId());
System.out.println("findMember.getClass() = " + findMember.getClass());
```

```

refMember.getClass() = class hellojpa.Member$HibernateProxy$smXM0bq
Hibernate:
    select
        member0_.id as id1_1_0_,
        member0_.createdBy as createdB2_1_0_,
        member0_.createdDate as createdD3_1_0_,
        member0_.lastModifiedBy as lastModi4_1_0_,
        member0_.lastModifiedDate as lastModi5_1_0_,
        member0_.USERNAME as USERNAME6_1_0_,
        member0_.TEAM_ID as TEAM_ID7_1_0_,
        team1_.id as id1_5_1_,
        team1_.name as name2_5_1_
    from
        Member member0_
    left outer join
        Team team1_
            on member0_.TEAM_ID=team1_.id
    where
        member0_.id=?
findMember.getClass() = class hellojpa.Member$HibernateProxy$smXM0bq

```

find 때문에 쿼리는 나가지만 둘 다 똑같은 class를 반환함....!

그냥 무조건 ==비교를 true로 하기 위해!

실무에선 복잡할 일이 거의 없긴함...

영속성 컨텍스트에 찾는 엔티티가 이미 있으면 em.**getReference()**를 호출해도 실제 엔티티 반환 → 반대도 가능.

영속성 컨텍스트의 도움을 받을 수 없는 준영속 상태일때

=> 영속성 컨텍스트로부터 값을 받아와서 프록시 객체를 초기화해야하는데 영속성 컨텍스트에서 빠져버렸으니 당연히 초기화할 수 없는 에러가 뜸...

```

package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import java.time.LocalDateTime;

```



```
import java.util.ArrayList;
import java.util.List;

public class JpaMain {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

        EntityManager em = emf.createEntityManager();

        EntityTransaction tx = em.getTransaction();
        tx.begin();
        try {

            Member member1 = new Member();
            member1.setName("member1");
            em.persist(member1);

            em.flush();
            em.clear();

            Member refMember = em.getReference(Member.class, member1.getId());
            System.out.println("refMember.getClass() = " + refMember.getClass());

            em.detach(refMember);
            //em.close();

            System.out.println(refMember.getName());

            tx.commit();

        } catch (Exception e) {
            tx.rollback();
            e.printStackTrace();
        } finally {
            em.close();
        }

        emf.close();
    }
}
```

```

    }

    private static void printMemberAndTeam(Member member) {
        String username = member.getName();
        System.out.println("username = " + username);

        Team team = member.getTeam();
        System.out.println("team.getName() = " + team.getName());
    }
}

```

```

INFO: HHH000476: Executing import script 'org.hibernate.tool.schema.internal.exec.ScriptSourceInputNonExistentImpl@2caa5d7c'
Hibernate:
    call next value for hibernate_sequence
Hibernate:
    /* insert hellojpa.Member
       */ insert
       into
       Member
       (createdBy, createdDate, lastModifiedBy, lastModifiedDate, USERNAME, TEAM_ID, id)
       values
       (?, ?, ?, ?, ?, ?, ?)
refMember.getClass() = class hellojpa.Member$HibernateProxy$ioVwVww0
org.hibernate.LazyInitializationException: Create breakpoint : could not initialize proxy [hellojpa.Member#1] - no Session <4 internal lines>
    at hellojpa.Member$HibernateProxy$ioVwVww0.getName(Unknown Source)
    at hellojpa.JpaMain.main(JpaMain.java:34)
6월 14, 2023 3:20:03 오전 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:h2:tcp://localhost/~/test2]

Process finished with exit code 0

```

=> LazyInitializationException을 기억하자!