JPA 개발하기

메인클래스를 만들고 그 안에 메인메서드 작성

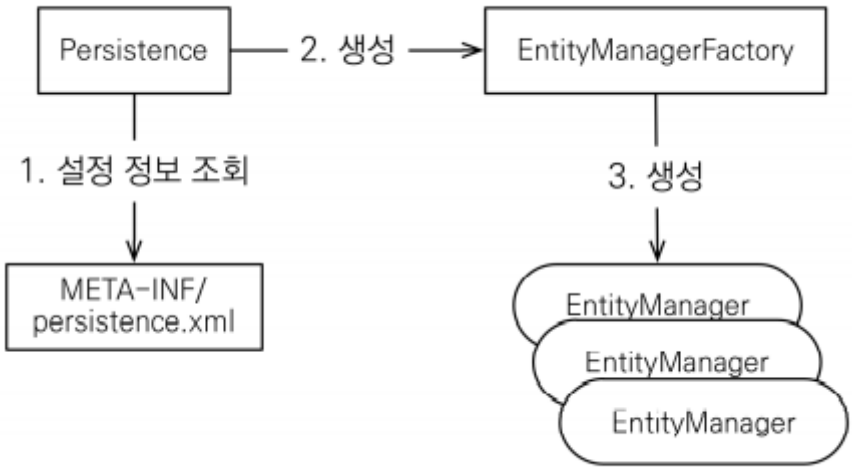
com.megait.Main

```
public class Main {
    public static void main(String[] args) {

        // EntityManager 생성
        EntityManagerFactory myjpa =
                Persistence.createEntityManagerFactory("myunit");


        // Manager를 통해 EntityManager 받아오기
        EntityManager entityManager = myjpa.createEntityManager();



        // 마지막엔 꼭 close() 하기!
        entityManager.close();
        myjpa.close();
    }
}
```

에러 없이 info 로그까지 나오면 된다.

JPA 구동 방식

EntityManager 란



JPA 로 회원 레코드 추가해보기

1. Member 테이블 생성하기 (h2에서)

```
CREATE TABLE member (
  id BIGINT NOT NULL,
  name VARCHAR(255),
  PRIMARY KEY(id)
);
```

2. Member 엔티티 생성하기

Member.java

```
package com.megait;


import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
@Getter @Setter @ToString
public class Member {

    @Id
    private Long id;

    private String name;

}
```

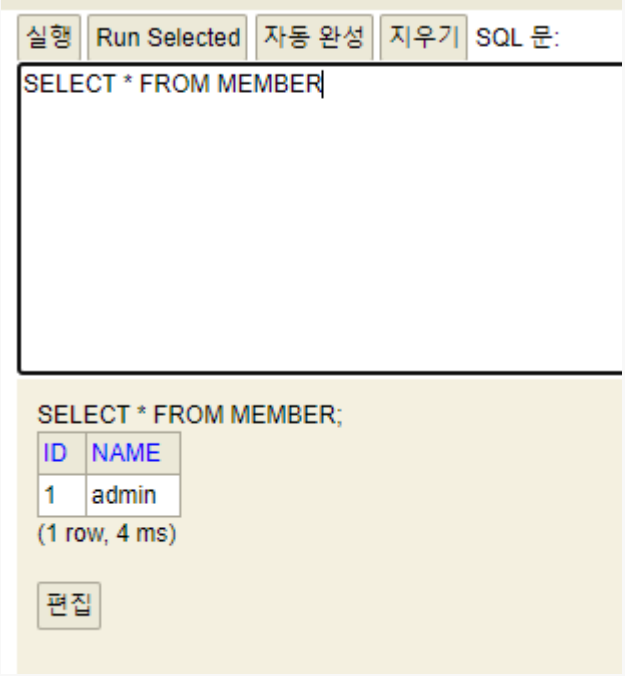
3. 새 멤버 추가하기 (INSERT)

```
package com.megait;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
```

```java
public class Main {
    public static void main(String[] args) {

        // EntityManager 생성
        EntityManagerFactory factory =
Persistence.createEntityManagerFactory("myunit");


        // Manager를 통해 EntityManager 받아오기
        EntityManager entityManager = factory.createEntityManager();

        ////////////////////////// 수정할 부분 ////////////////////////////////
        // 트랜잭션 시작
        EntityTransaction transaction = entityManager.getTransaction();
        transaction.begin();

        // member 객체 생성 및 값 저장
        Member member = new Member();
        member.setId(1L);
        member.setName("admin");

        // persist에 저장
        entityManager.persist(member);


        // 트랜잭션 종료 및 커밋
        transaction.commit();
        ////////////////////////// 수정할 부분 끝 ////////////////////////////////


        // 마지막엔 꼭 close() 하기! (close() 하지 않으면 프로그램이 종료되지 않는다.)
        entityManager.close();
        myjpa.close();
    }
}
```

결과

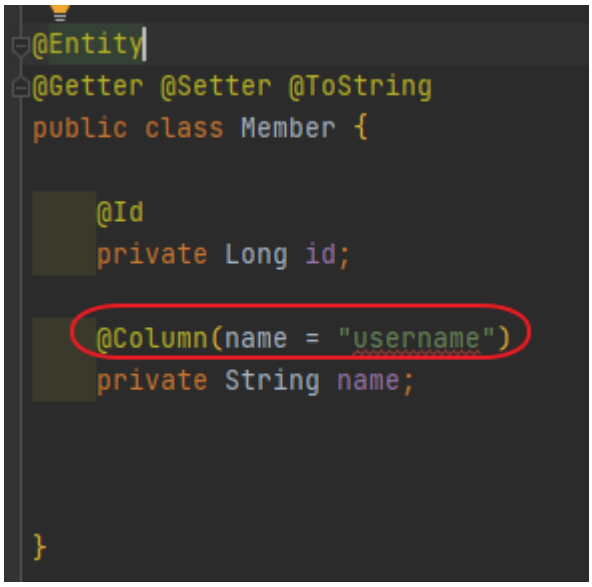
```
INFO: HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
Hibernate:
    /* insert com.megait.Member
        */ insert
        into
            Member
            (name, id)
        values
            (?, ?)
```



Table과 엔티티 이름이 서로 다른 경우

```
@Entity
@Table(name="mem")
@Getter @Setter @ToString
public class Member {

    @Id
    private Long id;

    private String name;


}
```

Column 과 필드 이름이 서로 다른 경우



잊지 말자. 올바른 close()

```
package com.megait;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class Main {
    public static void main(String[] args) {

        EntityManagerFactory factory =
                Persistence.createEntityManagerFactory("myunit");


        EntityManager entityManager = factory.createEntityManager();

        EntityTransaction transaction = entityManager.getTransaction();
        transaction.begin();

        try {
            Member member = new Member();
            member.setId(1L);
            member.setName("admin");

            entityManager.persist(member);


            transaction.commit();

        } catch (Exception e){
            transaction.rollback();  // 수 틀리면 롤백하기
        } finally {
            entityManager.close();
```

```
        }
        factory.close();
    }
}
```

JPA로 회원 검색하기 (SELECT)

Main.java

```
package com.megait;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class Main {
    public static void main(String[] args) {

        EntityManagerFactory factory =
                Persistence.createEntityManagerFactory("myunit");


        EntityManager entityManager = factory.createEntityManager();

        EntityTransaction transaction = entityManager.getTransaction();
        transaction.begin();

        try {

            Member member = entityManager.find(Member.class, 1L);
            System.out.println("result : " + member);

            transaction.commit();

        } catch (Exception e){
            transaction.rollback();
        } finally {
            entityManager.close();
        }
        factory.close();
    }
}
```

결과

```
Hibernate:
    select
        member0_.id as id1_0_0_,
        member0_.name as name2_0_0_
    from
        Member member0_
    where
        member0_.id=?
result : Member(id=1, name=admin)
```

아이디가 1인 회원 삭제하기 (REMOVE)

Main.java

```
 ....... (중략)

try {

    Member member = entityManager.find(Member.class, 1L);
    System.out.println("result : " + member);

    // 이 부분!
    entityManager.remove(member);

    transaction.commit();

} catch (Exception e){
    transaction.rollback();
} finally {
    entityManager.close();
}
 ....... (중략)
```

아이디가 1인 회원의 이름을 수정하기 (UPDATE)

Main.java

```
 ....... (중략)

try {

    Member member = entityManager.find(Member.class, 1L);
    System.out.println("result : " + member);

    // 이 부분!
    member.setName("pikachu");

    transaction.commit();
```

```
} catch (Exception e){
    transaction.rollback();
} finally {
    entityManager.close();
}
 ....... (중략)
```

주의

- 엔티티 매니저 팩토리는 하나만 생성해서 애플리케이션 전체에 서 공유
- 엔티티 매니저는 쓰레드간에 공유X (사용하고 버려야 한다).
- JPA의 모든 데이터 변경은 트랜잭션 안에서 실행

JPQL

- 전체 검색 (SELECT X FROM Y)
- 조건 검색 (SELECT X FROM Y WHERE ..)
- JOIN, HAVING, GROUP ...

```
entityManager.createQuery("쿼리문")
```

실제 DB 쿼리가 아니고 `JPQL`을 날린다.

```
List<Member> result =
    entityManager.createQuery("select m from Member as m", Member.class)
                 .getResultList();
System.out.println(results);
```

결과

```
Hibernate:
    /* select
        m
    from
        Member as m */ select
            member0_.id as id1_0_,
            member0_.name as name2_0_
        from
            Member member0_
[Member(id=1, name=pikachu), Member(id=2, name=user01)]
```

Pagination (페이징)

```
List<Member> results =
        entityManager.createQuery("select m from Member as m", Member.class)
                .setFirstResult(11)  // offset : 11
                .setMaxResults(5)    // limits : 5
                .getResultList();
System.out.println(results);
```

SQL은 테이블을 대상으로 쿼리, JPQL은 객체를 대상으로 쿼리