JPA 1장

참조: 자바 ORM 표준 JPA 프로그래밍(김영한)

JPA(Java Persistence API)란

JPA: 자바 진영의 ORM 기술 표준으로 **인터페이스**이다.

ORM(Object-Relational Mapping): 우리가 일반적으로 알고 있는 애플리케이션 Class와 RDB(Relational DataBase)의 테이블을 매핑(연결)한다는 뜻이며, 기술적으로는 어플리케이션의 객체를 RDB 테이블에 자동으로 영속화 해주는 것이라고 보면 된다.

ORM->객체와 관계형 데이터베이스를 매핑한다는 뜻이다.
JPA -> 객체와 데이터베이스를 매핑할 수 있도록 제공해주는 인터페이스(구현체X)

https://dbjh.tistory.com/77

SQL 직접 다룰 때 문제

반복 ↑

Find함수 작성
SQL 작성 -> SQL실행 -> SQL로부터 결과 가져와 객체로 맵핑

Save 함수 추가
SQL작성 -> 등록 -> SQL 실행

~함수 추가
다시 SQL 작성~ 과정 반복

.
.
.

SQL의존적 개발

사례) 코드 완성 후 요구 사항 추가

Member 객체에 team 필드가 추가됨
그러나 코드를 실행해보니 member.getTeam()의 값이 항상 null이 나옴.

memberDAO 코드를 열어보니 회원을 출력할 때 사용하는 find() 메소드는 회원만 조회하는 SQL을 그대로 유지함
그래서 **findWithTeam() 메서드를 추가**

findWithTeam()에 **연관된 SQL 추가적 작성** ->

```
SELECT M.MEMBER_ID, M.NAME, M.TEL, T.TEAM_ID,
T.TEAM_NAME FROM MEMBER MJOIN TEAM T ON
M.TEAM_ID = T.TEAM_ID
```

Team 객체를 사용할 수 있을지 없을지는 전적으로 사용하는 SQL에 달려있음

참고: https://ddd4117.github.io/2021/04/jpa-1%EC%9E%A5-sql%EC%9D%84-%EC%A7%81%EC%A0%91-%EB%8B%A4%EB%A3%B0-%EB%95%8C-%EB%B0%9C%EC%83%9D%ED%95%98%EB%8A%94-%EB%AC%B8%EC%A0%9C%EC%A0%90/

SQL 직접 다룰 때 문제

패러다임 불일치

자바 어플리케이션은 관계형 데이터베이스를 주로 사용

데이터베이스는 데이터 중심 구조

(객체 중심 구조X)

->객체를 직접 저장하거나 조회할 수 없다.

자바 객체 지향 어플리케이션과 데이터베이스 중간에서 SQL과 JDBC API를 코드로 작성해야 한다.

-> 객체를 데이터베이스에 CRUD하기 위해서 많은 코드 작성 요구

	데이터	객체
상속	X	O
연관관계	양방향 (외래키 이용해서)	한 방향 (field값으로 참조를 넣은것만 접근 가능)
객체 그래프 탐색	사용하는 객체 그래프가 언제 끊어질지 알 수 없으므로 객체를 함부로 탐색 할 수 없다.	

용어정리>

외래 키: **두 테이블의 데이터 간 연결을 설정하고 강제 적용하여 외래 키 테이블에 저장될 수 있는 데이터를 제어하는 데 사용되는 열**

Cf> 객체 그래프 탐색

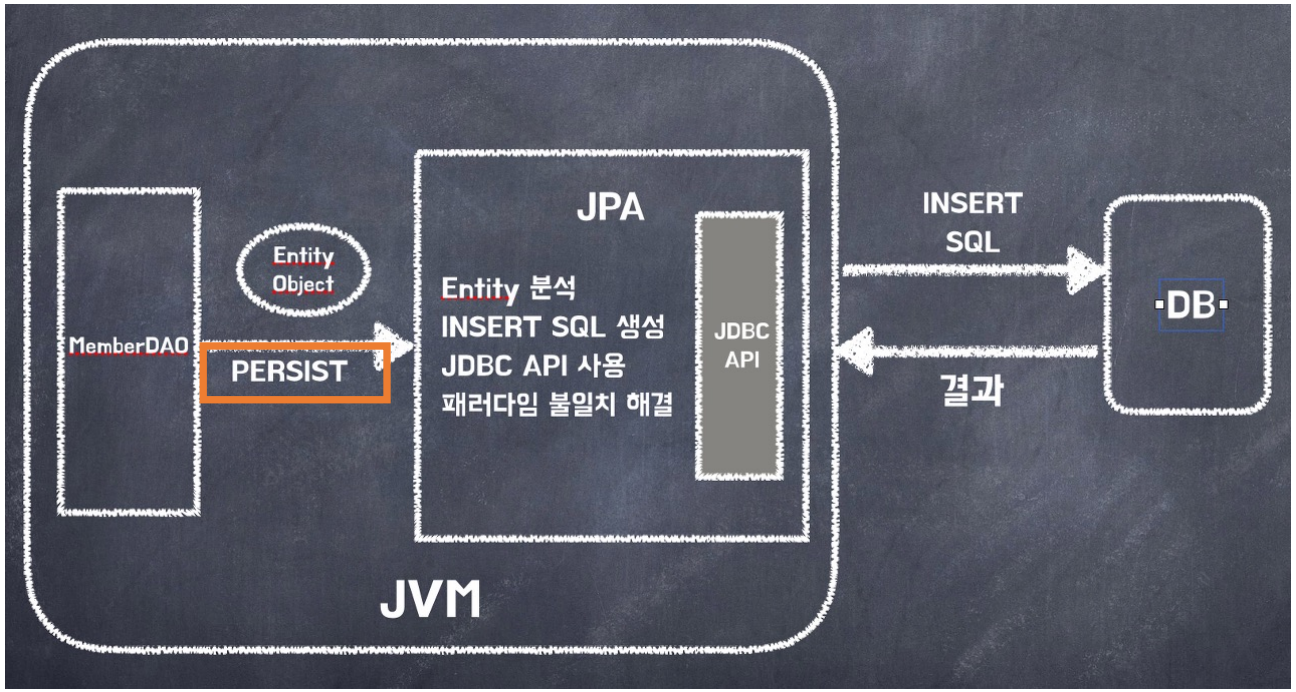
```
class MemberService {
        public void process() {
                Member member = memberDAO.find(memberId);
        member.getTeam();
        //member -> team 객체 그래프 탐색이 가능한가?
                member.getOrder().getDelivery();
        // memberDAO.find(memberId)의 Query를 확인해보지 않고 선
        //코드를 사용 불가능
        }
}
```

쿼리가 짜져 있지 않으면 함수를 불러 봤자 소용없음
언제 끊어질 지 모른다.

JPA

무엇을 해주는가?
-> SQL 생성 / 패러다임 불일치 문제 해결 등

SQL 직접 작성 ✕
객체 작성 -> SQL연결 -> SQL 작성

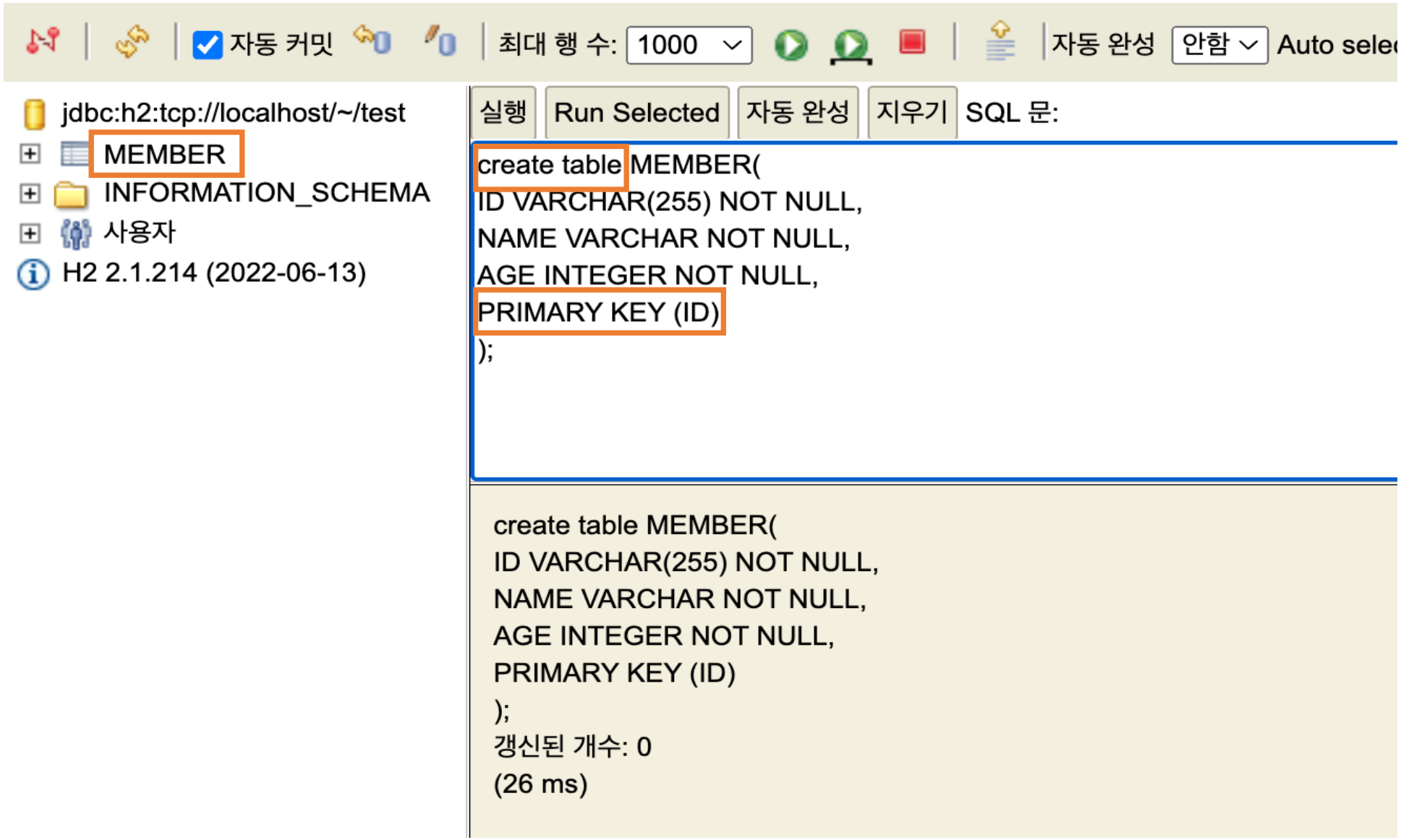


Jvm: 자바 가상 머신
벤더: 생산회사
출처 : https://ultrakain.gitbooks.io/jpa/content/chapter1/chapter1.3.html

JPA 2장

참조: 자바 ORM 표준 JPA 프로그래밍(김영한)

CF> h2 DB 테이블 만들기



Hibernate(하이버네이트)

- JPA 구현체
- JPA의 구현체에는 하이버네이트만 있는 것은 아니지만 가장 많이 쓰임

Maven(메이븐)

- 라이브러리 관리 및 빌드를 돕는 도구
- pom.xml에 사용할 라이브러리를 적어주면 라이브러리를 자동으로 내려받아 관리해준다.
- Dependencies에 사용할 라이브러리 지정

Maven(메이븐)_dependencies

groupId:해당 프로젝트를 모든 프로젝트 사이에서 고유하게 식별해줌
Ex)org.apache.maven(하위 그룹 추가 가능)

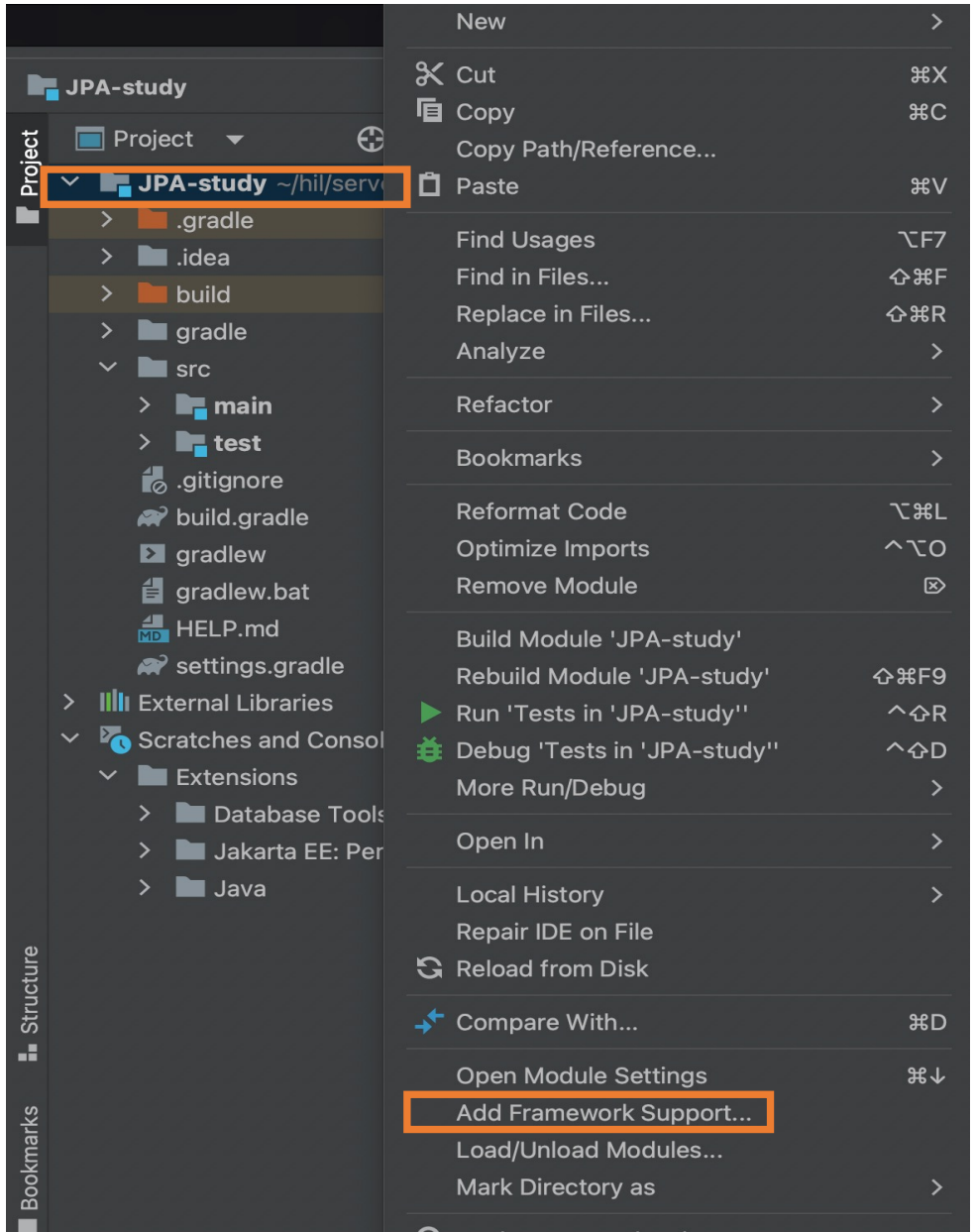
artifactId:버전 정보를 생략한 jar 파일의 이름
Ex)maven

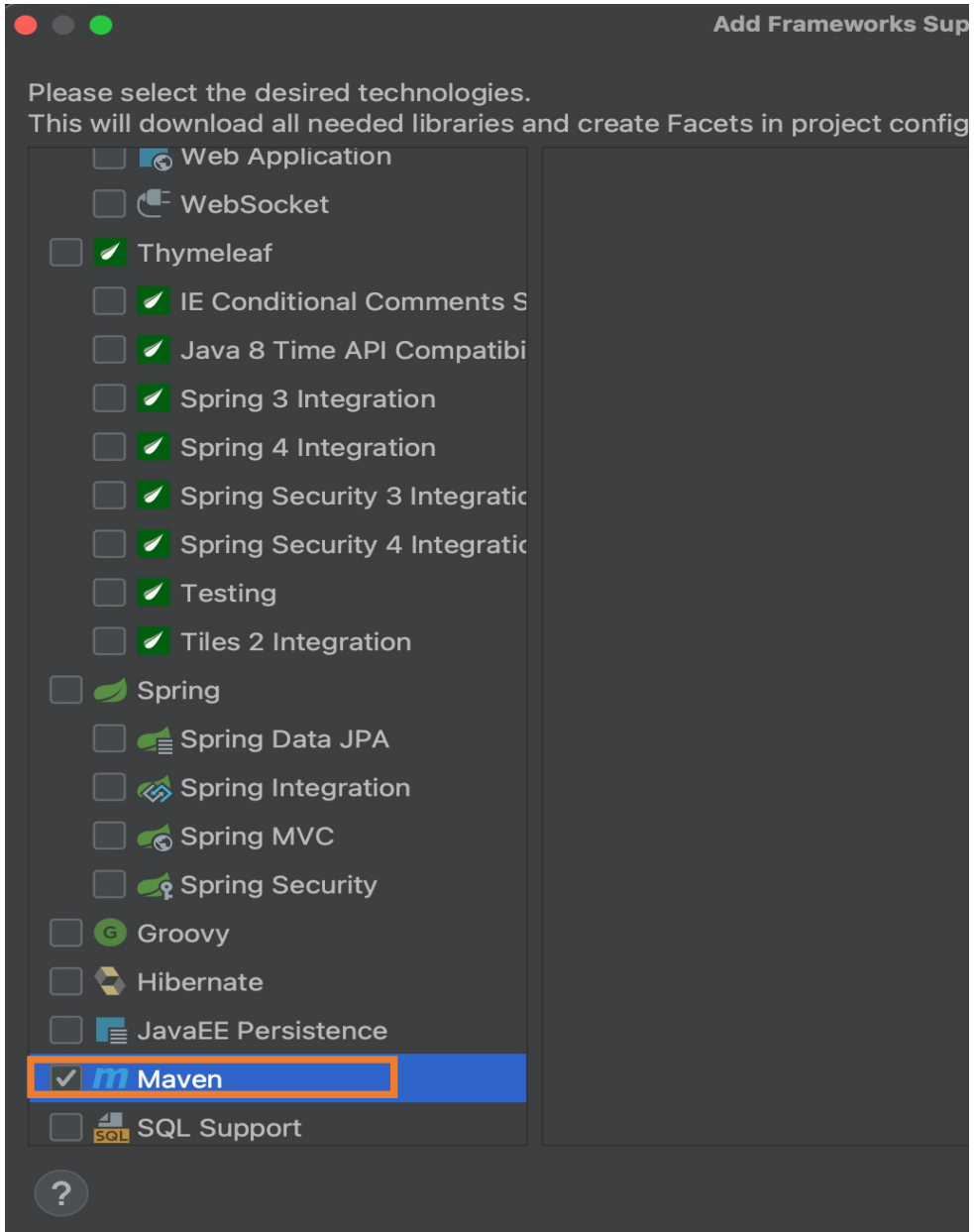
H2, hibernate 사용을 위해 pom.xml을 통해 내려받음

2.3장

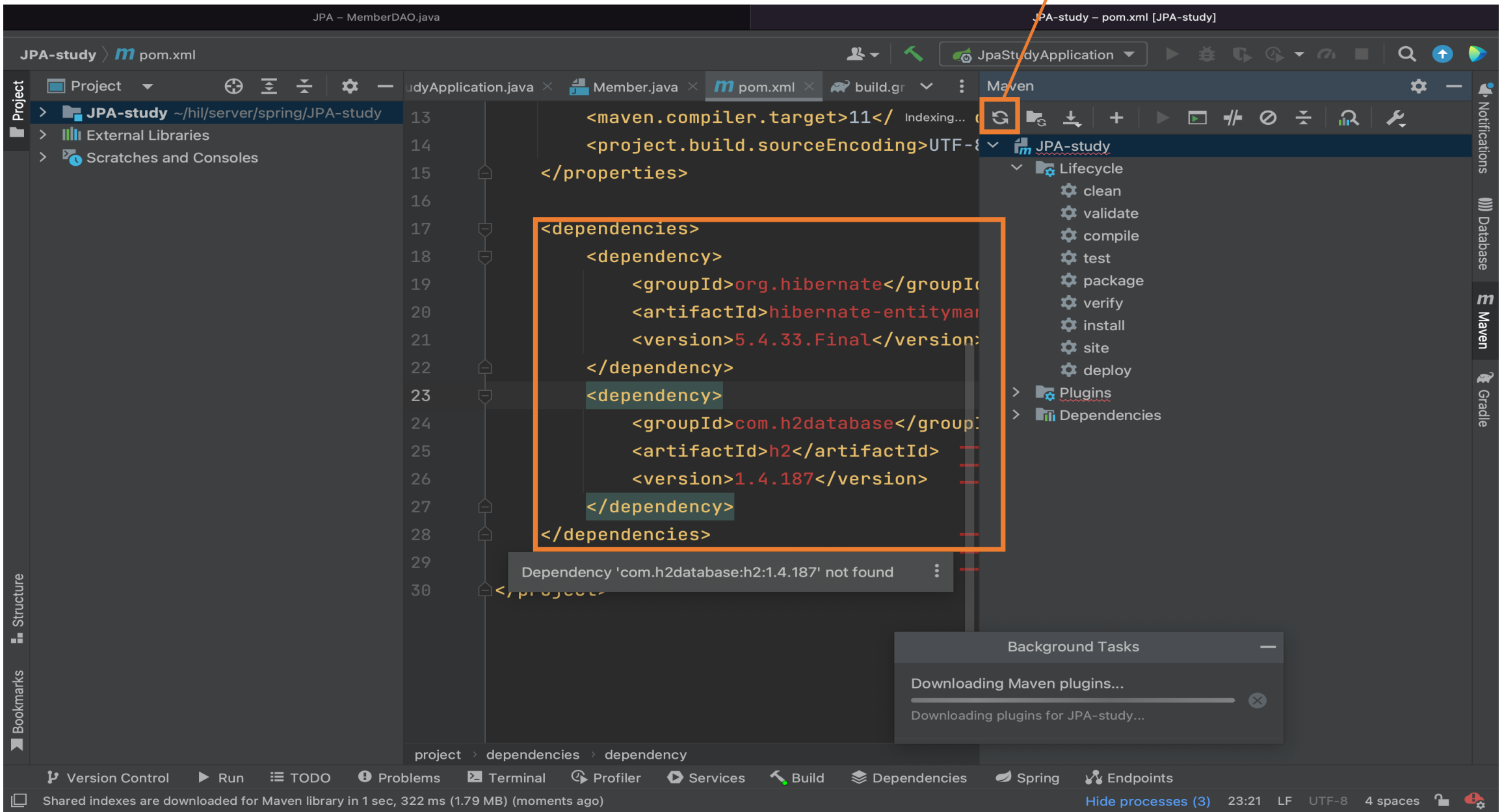
참고: https://johngrib.github.io/wiki/groupId-artifactId/

CF> Maven 추가 방법



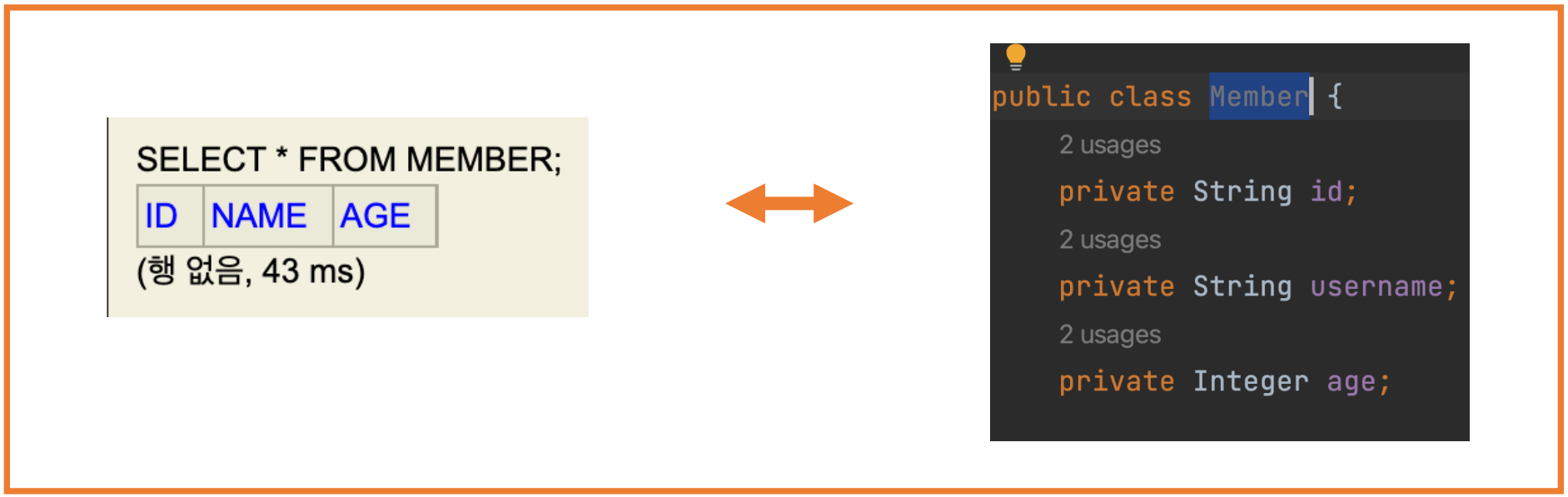


Dependency 작성 후 다운 받을 수 있도록 클릭해야 에러 사라짐



맵핑

매핑 정보	회원 객체	회원 테이블
Class / Table	Member	MEMBER
PK	id	ID
Field1 / column1	username	NAME
Field2 / column2	age	AGE



매핑정보(코드)

매핑 정보	회원 객체	회원 테이블
Class / Table	Member	MEMBER
PK	id	ID
Field1 / column1	username	NAME
Field2 / column2	age	AGE

```java
package JPA.JPAstudy;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table (name = "MEMBER")
public class Member {
    2 usages
    @Id
    @Column(name = "ID")
    private String id;

    2 usages
    @Column(name = "NAME")
    private String username;

    2 usages
    @Column(name = "AGE")
    private Integer age;
```

Import javax.persistence.*(매핑 어노테이션을 위해 사용)

"Class와 table를 매핑한다 " 를 jpa에게 알림

"name속성을 이용해 Member entity를 MEMBER table에 맵핑"
->엔티티 클래스에 매핑할 테이블 정보 알림

"Class의 필드를 table의 PK와 맵핑" -> **식별자 필드**라고 부른다.

Username 필드를 테이블의 NAME에 맵핑
->필드를 컬럼에 매핑

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.1">

    <persistence-unit name="jpastudy">

        <properties>
            <!-- 필수 속성 -->
            <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
            <property name="javax.persistence.jdbc.user" value="sa"/>
            <property name="javax.persistence.jdbc.password" value=""/>
            <property name="javax.persistence.jdbc.url" value="jdbc:h2:tcp://localhost/~/test"/>
            <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect" />

            <!-- 옵션 -->
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="true" />
            <property name="hibernate.use_sql_comments" value="true" />
            <property name="hibernate.id.new_generator_mappings" value="true" />

            <!--<property name="hibernate.hbm2ddl.auto" value="create" />-->
        </properties>
    </persistence-unit>
</persistence>
```

영속성 유닛_일반적으로 연결할 DB당 하나 등록, 고유 이름 부여(jpastudy)

JPA표준 속성

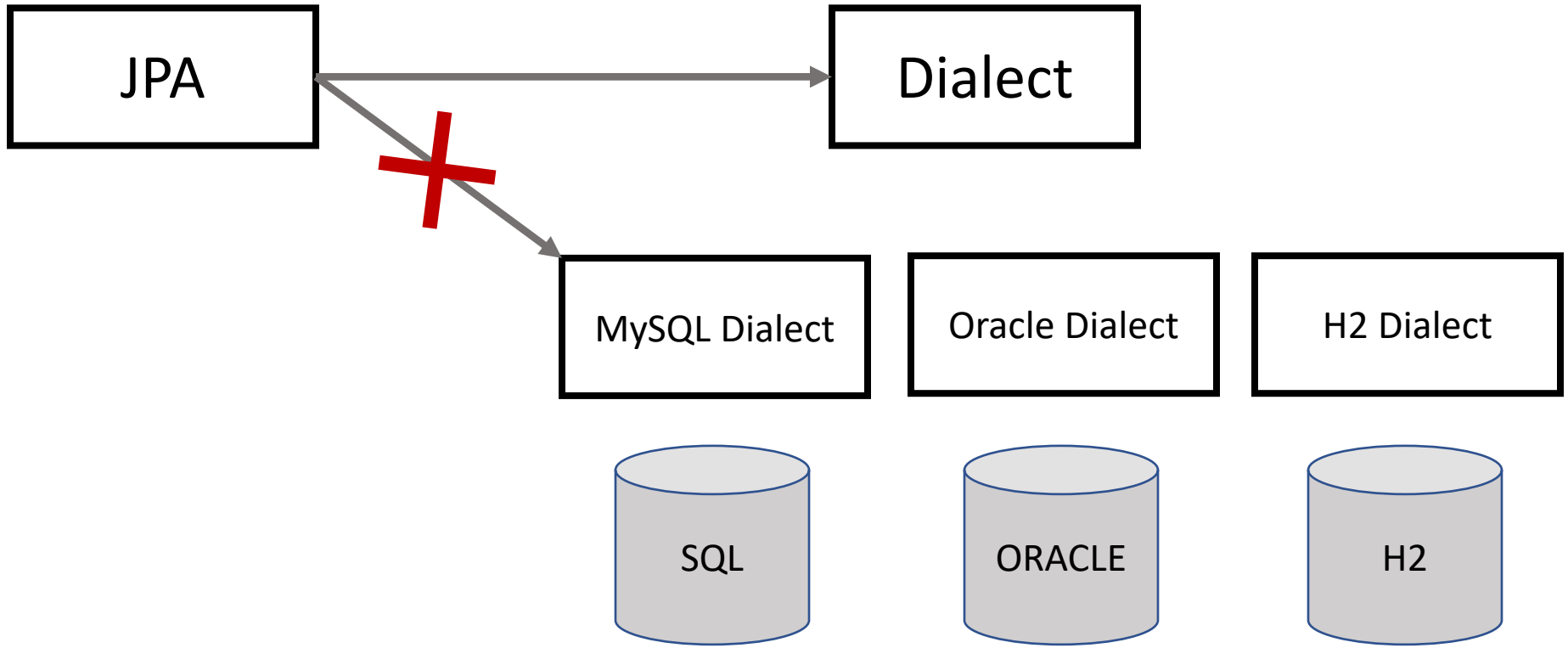
JDBC드라이버
DB접속 아이디
DB접속 비밀번호
DB접속 URL

DB방언 설정
하이버네이트 속성으로
하이버네이트에서만 사용 가능

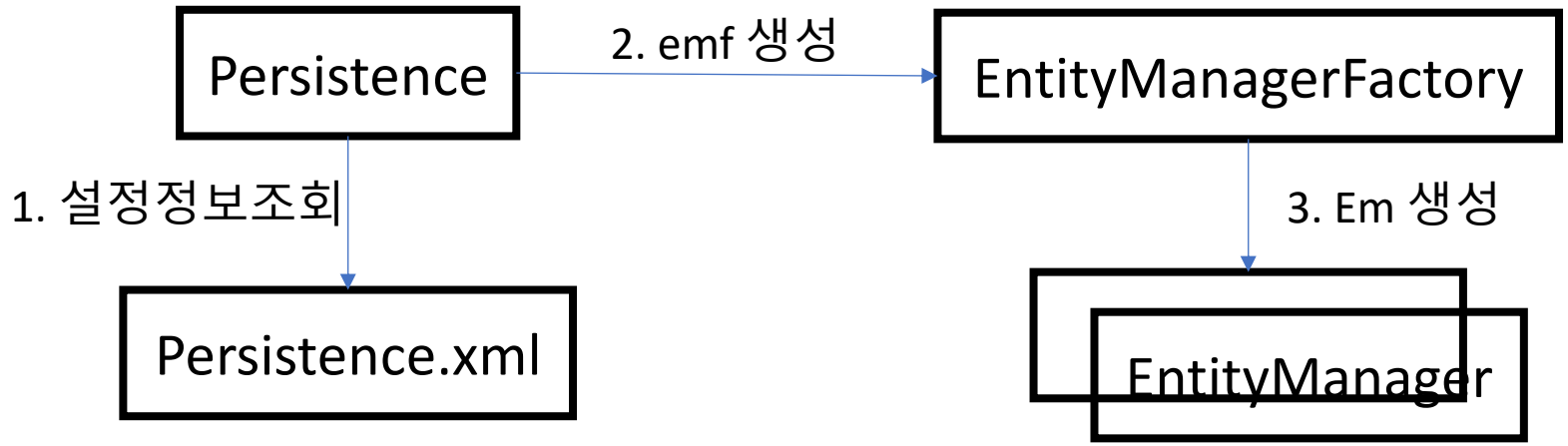
4.6절에서 다룸

DB방언: 특정DB만의 고유한 기능

- 특정 DB에 종속되는 기능을 많이 넣을 경우 DB교체가 쉽지 않음으로 다양한 DB방언 클래스가 제공됨
- 방언 설정은 jpa 표준이 아니다(즉, 구현체 중 하이버네이트를 사용하면 그에 맞게 사용해야 함)



엔티티 매니저 설정



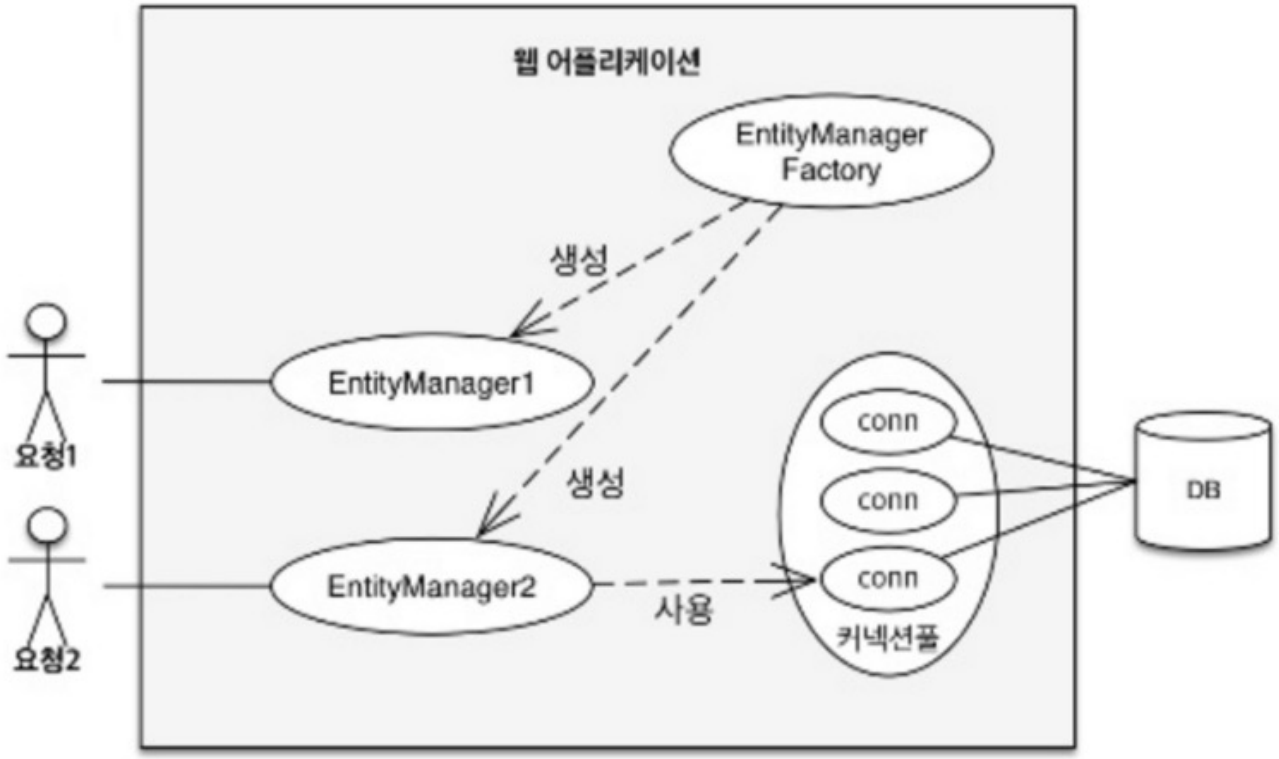
EntityManagerFactory emf = Persistence.*createEntityManagerFactory*("jpastudy");

- JPA 시작을 위해 persistence.xml의 설정 정보를 사용해 emf생성
- <persistence-unit name="jpastudy">을 찾아 emf생성하며 emf는 전체에서 딱 한 번만 생성

EntityManager em = emf.createEntityManager();

- em을 통해 DB에 엔티티를 등록.수정.삭제.조회 가능(CRUD)
- 개발자는 em을 가상의 DB로 생각할 수 있다
- 공유, 재사용 X

em 공유가 안되는 이유



출처: https://debaeloper.tistory.com/38

em은 **데이터베이스 커넥션과 밀접하게** 관계가 있다. 따라서 여러 쓰레드가 동시 접근하면 **동시성 문제**가 발생하기에 **쓰레드간 공유를 하면 안된다.**

동시성 문제: 여러 스레드가 동시에 같은 인스턴스의 필드 값을 변경 하면서 발생하는 문제를 의미

트랜젝션

- EntityTransaction tx = em.getTransaction(); // 트랜젝션 생성

- 항상 트랜젝션 안에서 data변경

- tx.begin();//트랜젝션 시작
- tx.commit();//트랜젝션 정상 동작 시 커밋

Cf>commit: 변경 사항을 영구적으로 반영하는 것, 이전 데이터를 잃게 됨

- tx.rollback();//롤백

Cf> Rollback: 변경사항 취소, 이전으로 돌아감

참고: https://hoon93.tistory.com/24

비즈니스 로직

```
42
43          //enroll
44          em.persist(member);
45
46          //update
47          member.setAge(20);
48
49          //find one
50          Member findMember = em.find(Member.class, id);
51          System.out.println("Name: " + findMember.getUsername() + " AGE: " + findMember.getAge());
52
53          //find list
54          List<Member> members = em.createQuery( qlString: "select m from Member m ", Member.class)
55                  //MEMBER, member etc. -> X, only Member -> O
56                  .getResultList();
57          System.out.println("size: " + members.size());
58
59          //delete
60          em.remove(member);
61      }
62  }
```

@Id_"id1"

output

```
Name: hil AGE: 20
```

Find(조회할 엔티티 타입, PK)로 조회

JPQL
앤티티 중심 개발 -> 검색도 엔티티 중심이어야 함. 이때 SQL이 필요하게 됨.
그래서 JPA가 사용하는 쿼리언어가 JPQL

JPQL은 데이터베이스 테이블을 전혀 알지 못한다.
em.createQuery(JPQL, 반환타입)_10장

실행 error 중 Unknown entity error

```
...
    <persistence-unit name="jpastudy">
      <class>JPA.JPAstudy.Member</class>
      <properties>
        ...
```

추가

참고: https://gongbuzzangzoa.tistory.com/344