4장 엔티티 매핑

4.1 @Entity

JPA를 사용해서 테이블과 매핑할 클래스는 @Entity 어노테이션을 필수로 붙여야 한다.

Entity 어노테이션 속성

name : JPA에서 사용할 엔티티 이름을 지정.default : 클래스 이름

주의사항

* 기본 생성자는 필수
* final 클래스, enum, interface, inner 클래스에는 사용할 수 없다.
* 저장할 필드에 final을 사용하면 안 된다.

자바는 생성자가 하나도 없으면 기본 생성자를 자동으로 만드므로 1번은 상관없다? X

생성자가 하나라도 있으면 기본 생성자를 자동으로 만들지 않기 때문에 생성자를 만들게되면 기본 생성자를 직접 만들어줘야 한다.

4.2 @Table

엔티티와 매핑할 테이블을 지정한다. 생략하면 매핑한 엔티티 이름을 테이블 이름으로 사용.

Table 어노테이션 속성

name : 매핑할 테이블 이름. default : 엔티티 이름

catalog : catalog 기능이 있는 데이터베이스에서 catalog를 매핑

catalog : **데이터베이스**의 개체들에 대한 정의를 담고 있는 메타데이터들로 구성된 **데이터베이스** 내의 인스턴스이다.

schema : schema 기능이 있는 데이터베이스에서 schema를 매핑

-> 데이터베이스의 구조

uniqueConstraints : DDL 생성 시에 유니크 제약조건을 만든다. 이 기능은 스키마 자동 생성 기능을 사용해서 DDL을 만들 때만 사용됨.

4.3 다양한 매핑 사용



@Entity : 클래스이름을 엔티티 이름으로 사용하고, JPA를 사용할 때 이 엔티티를 사용하겠다.

@Table : MEMBER 테이블과 매핑하고 NAME과 AGE 컬럼에 대해서는 유니크 제약조건을 사용하겠다.

@Id : 기본키로 사용할 필드

@Column 테이블의 컬럼으로 매핑하고 name속성을 통해 컬럼명 지정, NAME, AGE 컬럼에 대해서는 제약조건을 걸 수 있다.

@Enumerated : 회원의 타입을 일반 회원과 관리자로 나눠서 관리한다고 하면 enum을 사용하는데 enum을 사용해 매핑하려면 이 어노테이션을 사용해야 한다.

@Temporal : 자바의 날짜 타입

@Lob : 회원 설명에 대한 필드는 길이 제한이 없으므로 VARCHAR이 아니라 CLOB 타입으로 저장해야 하는데, Lob 어노테이션을 사용하면 CLOB, BLOB 타입을 매핑할 수 있다.

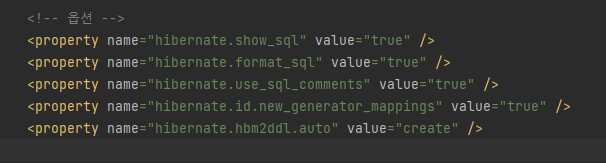
CLOB과 BLOB이란?

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=rlasksd>

@Transient : @Transient는 엔티티 클래스 내의 특정 변수를 영속 필드에서 제외할 때 사용한다.

4.4 데이터베이스 스키마 자동 생성

persistence.xml



show\_sql : sql을 콘솔에 출력해준다. show\_sql만 사용하면 줄 바꿈 없이 출력되므로

format\_sql을 추가하면 좀 더 예쁘게 출력이 된다.

use\_sql\_comments를 추가하면 sql문에 있어서 /\*\*/을 활용하여 주석을 추가할 수 있음.

id.new\_generator\_mappings는 true로 하면 자동 키 생성 전략을 사용할 수 있다고 함.

**@GeneratedValue을 사용해야함.**

hbm2ddl.auto의 value를 create로 설정하면 애플리케이션 실행 시점에 데이터베이스 테이블을 자동으로 생성한다.

value :

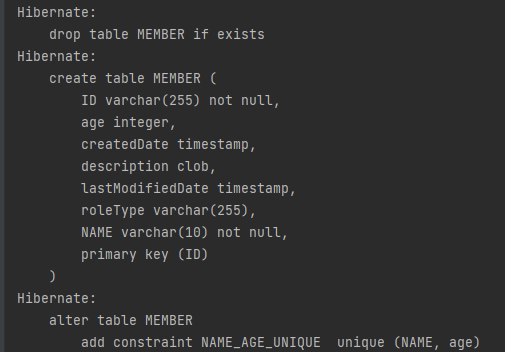
create : DROP + CREATE

create-drop : DROP + CREATE + DELETE 끝나면 삭제까지 함.

update : 데이터베이스 테이블과 엔티티 매핑정보를 비교해서 변경 사항만 수정

validate : DB 테이블과 엔티티 매핑정보를 비교해서 차이가 있으면 경고를 날리고 실행하지 않음. 이 설정은 DDL을 수정하지 않음.

none : 자동 생성 기능을 사용하지 않겠다.

 실행결과

4.6 기본키 매핑

데이터베이스 벤더마다 기본 키를 생성하는 방식은 다르다. 예를 들어 MySQLl의 AUTO\_INCREMENT 같은 기능을 사용해서 생성된 값을 기본 키로 사용한다면?.. 역시 데이터베이스마다 다른데 이러한 점을 JPA에서는 어떻게 해결했는지 알아본다.

직접할당 : 기본 키를 애플리케이션에서 직접 할당한다.

자동 생성 : 대리 키(대체 키) 사용 방식 + @GeneratedValue 어노테이션 사용

* IDENTITY : 기본 키 생성을 데이터베이스에 위임한다.
* SEQUENCE : 데이터베이스 시퀀스를 사용해서 기본 키를 할당한다.
* TABLE : 키 생성 테이블을 사용한다.

4.6.1 기본 키 직접 할당 전략

@Id로 매핑을 한다.

직접 할당 전략은 em.persist()로 엔티티를 저장하기 전에 애플리케이션에서 기본 키를 직접 할당하는 방법.

이 방법을 사용할 때 식별자의 값을 설정하지 않고 persist하게 되면 예외가 발생한다.

4.6.2 IDENTITY 전략

IDENTITY는 기본 키 생성을 데이터베이스에 위임하는 전략.

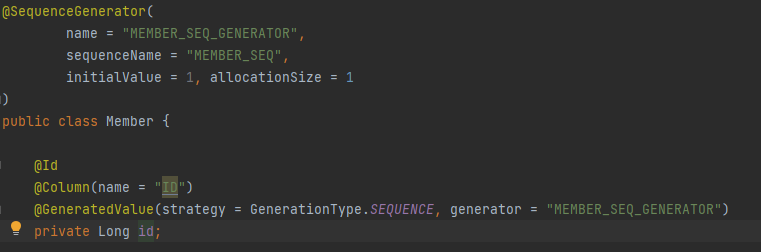
사용 : @GeneratedValue(strategy = GenerationType.IDENTITY)어노테이션을 추가한다.

IDENTITY 전략은 em.persist를 통해 엔티티를 저장할 때 데이터베이스에 먼저 생성을 하고(쓰기 지연 X) 영속 상태로 등록이 된다.

4.6.3 SEQUENCE 전략

데이터베이스 시퀀스는 유일한 값을 순서대로 생성하는 특별한 데이터베이스 오브젝트다.

SequenceGenerator라는 어노테이션을 사용해 원하는 값을 시퀀셜하게 넣어주는 역할을 해준다. 사용법은 다음과 같다.



4.6.4 TABLE 전략

TABLE 전략은 키 생성 전용 테이블을 하나 만들고 여기에 이름과 값으로 사용할 컬럼을 만들어 데이터베이스 시퀀스를 흉내내는 전략이다.

직접할당이나 IDENTITY, SEQUENCE 전략은 사용가능한 자바 타입, 데이터베이스 종류가 있는데 TABLE 전략은 테이블을 사용하므로 모든 데이터베이스에 적용 가능하다.

키 생성 전용 테이블 DDL

create table MY\_SEQUENCES (

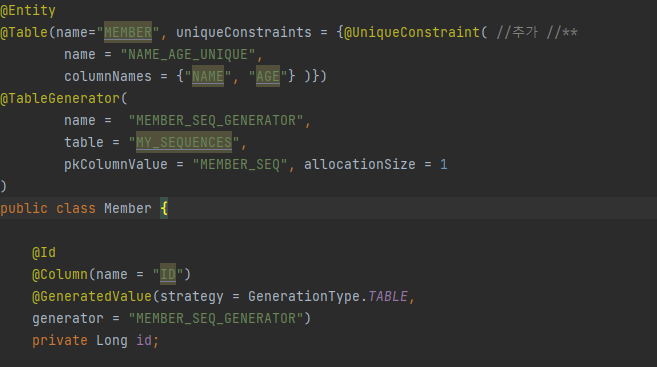
sequence\_name varchar(255) not null,

next\_val bigint,

primary key ( sequence\_name)

)

이렇게 생성을 한 후, 사용하면 다음과 같다.



4.6.5 AUTO 전략

GeneratedValue(strategy = GenerationType.AUTO) 어노테이션을 사용하게 되면

선택한 데이터베이스 방언에 따라 IDENTITY, SEQUENCE, TABLE 전략 중 하나를 자동으로 선택하며, GeneratedValue어노테이션의 기본값은 AUTO이다.

따라서 그냥 사용하면 된다. 그런데, TABLE이나 SEQUENCE 전략의 경우는 테이블이나 시퀀스를 미리 만들어둬야하는데 어떡하나?..

JPA의 스키마 자동 생성 기능을 사용한다면 하이버네이트가 기본값을 사용해서 적절한 시퀀스나 키 생성용 테이블을 만들어 준다. 그냥 사용해라..

4.7 필드와 컬럼 매핑 : 레퍼런스

JPA가 제공하는 필드와 컬럼 매핑용 어노테이션들을 레퍼런스 형태로 정리해놓은 장이다.

위에서 코드를 보면서 사용한 어노테이션 의미가 무엇인지 찾아봤었는데 이렇게 정리를 해놓았다. 역시 어노테이션은 사용하면서 찾아보는게 맞는 것 같아 정리는 생략하겠다.