# 04. Entity Mapping

...

Blur

# JPA Annotation

```java
@NoArgsConstructor
@Getter
@Setter
@Entity
@Table(name="MEMBER", uniqueConstraints = {@UniqueConstraint(
        name = "NAME_AGE_UNIQUE",
        columnNames = {"NAME", "AGE"})})
public class Member {
    @Id
    @Column(name = "ID")
    private String id;
    @Column(name = "NAME", nullable = false, length = 10)
    private String username;
    //자동 매핑 @Column(name = "AGE") 대소문자 구문 없는 데이터베이스 기준
    private Integer age;
```

@Entity
- JPA를 사용해서 Table과 Mapping
- @NoArgasConstructor

@Table
- Entity와 Mapping할 Table 지정

@Id
- Primary Key

@Column(name = ~~)
- Mapping할 DB의 Column 이름

* 제약 조건
1. DDL으로 사용 가능
2. 개발자가 Entity만 보고도 다양한 제약조건 파악을 할 수 있다는 장점

# JPA Annotation

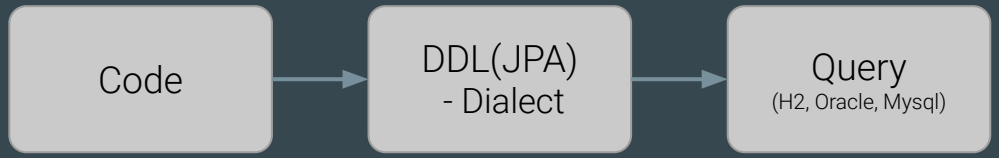
```
    @Enumerated(EnumType.STRING)
    private RoleType roleType;
    @Temporal(TemporalType.TIMESTAMP)
    private Date createdDate;
    @Temporal(TemporalType.TIMESTAMP)
    private Date lastModifiedDate;
    @Lob
    private String description;
}
```

@Enumrated
- EnumType.STRING- Enum to String
- EnumType.ORDINAL- Enum to int

@Temporal
- DATE(2013-10-11)
- TIME(11:11:11)
- TIMESTAMP(2013-10-11 11:11:11)

@Lob
- CLOB(문자)- String, char[], java.sql.CLOB
- BLOB(나머지)- byte[], java.sql.BLOB

# DDL(Data Definition Language)




```
Hibernate:
    drop table member if exists
Hibernate:
    create table member (
        id varchar(255) not null,
        age integer,
        created_date timestamp,
        description clob,
        last_modified_date timestamp,
        role_type varchar(255),
        name varchar(10) not null,
        primary key (id)
    )
Hibernate:
    alter table member
        add constraint NAME_AGE_UNIQUE  unique (name, age)
```

- 사용자 대신 DB Schema를 자동으로 생성하는 기능
- 운영 환경에서는 사용하지 않음

```
<!-- 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" />
```

# @id Mapping

```java
@Table(name = "MEMBER")
@Entity
public class Member {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "MEMBER_ID")
    private Long id;

    private String name;

    private String city;
    private String street;
    private String zipcode;
}
```

Primary Key

1. Not Null
2. Unique
3. Constant

- 자연 키(주민번호, 이메일, 전화번호)
- 대리 키(대체 키)

  - Sequence, Auto_increment, Table

# @id Strategy

```java
@Table(name = "MEMBER")
@Entity
public class Member {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "MEMBER_ID")
    private Long id;

    private String name;

    private String city;
    private String street;
    private String zipcode;
}
```

IDENTITY
- DB에 위임하는 전략(AUTO_INCREMENT)
- Entity가 DB에 저장되어야 Id가 생성
- 쓰기지연이 동작하지 않음.(쓰기지연 활용 최적화 X)

SEQUENCE
- 유일한 값을 순서대로 생성하는 특별한 데이터베이스 오브젝트 이용
- @SequenceGenerator 등록 필요

TABLE
- 키 생성 전용 테이블 이용(Sequence를 흉내내는 전략)
- @TableGenerator 등록 필요

AUTO
- DB Dialect에 따라 자동으로 설정
- Oracle(SEQUENCE), MySQL(IDENTITY)

# Sequence, Table Strategy

```java
@Entity
@SequenceGenerator(
        name = "BOARD_SEQ_GENERATOR",
        sequenceName = "BOARD_SEQ",
        initialValue = 1, allocationSize = 1)
public class Member {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
                    generator = "BOARD_SEQ_GENERATOR")
    @Column(name = "MEMBER_ID")
    private Long id;
```

```
Hibernate:
    create sequence BOARD_SEQ start with 1 increment by 1
Hibernate:
    create sequence hibernate_sequence start with 1 increment by 1
```

```java
@Entity
@TableGenerator(
        name = "BOARD_SEQ_GENERATOR",
        table = "MY_SEQUENCE1",
        pkColumnValue = "BOARD_SEQ", allocationSize = 1)
public class Member {

    @Id
    @GeneratedValue(strategy = GenerationType.TABLE,
                    generator = "BOARD_SEQ_GENERATOR")
    @Column(name = "MEMBER_ID")
    private Long id;
```

```
Hibernate:
    create table MY_SEQUENCE1 (
         sequence_name varchar(255) not null ,
        next_val bigint,
        primary key ( sequence_name )
    )
```

# @Access

```java
@Entity
@Table(name = "MEMBER")
@Access(AccessType.FIELD)
public class Member {
    @Id
    @GeneratedValue
    @Column(name = "MEMBER_ID")
    private Long id;

    private String name;
    @Transient
    private String lastName;

    @Access(AccessType.PROPERTY)
    public String getFullName() {
        return name + lastName;
    }
```

AccessType.FIELD
- 변수 직접 접근

AccessType.PROPERTY
- Getter 이용해서 접근

@Access 생략 시
- @Id의 위치를 기준으로 접근

두가지 방식 혼용 가능

Thank you