KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

영속,비즈니스 계층의 CRUD 구현 DB 연동 (DAO)

[KB] IT's Your Life

객체지향설계가 원래 바텀업

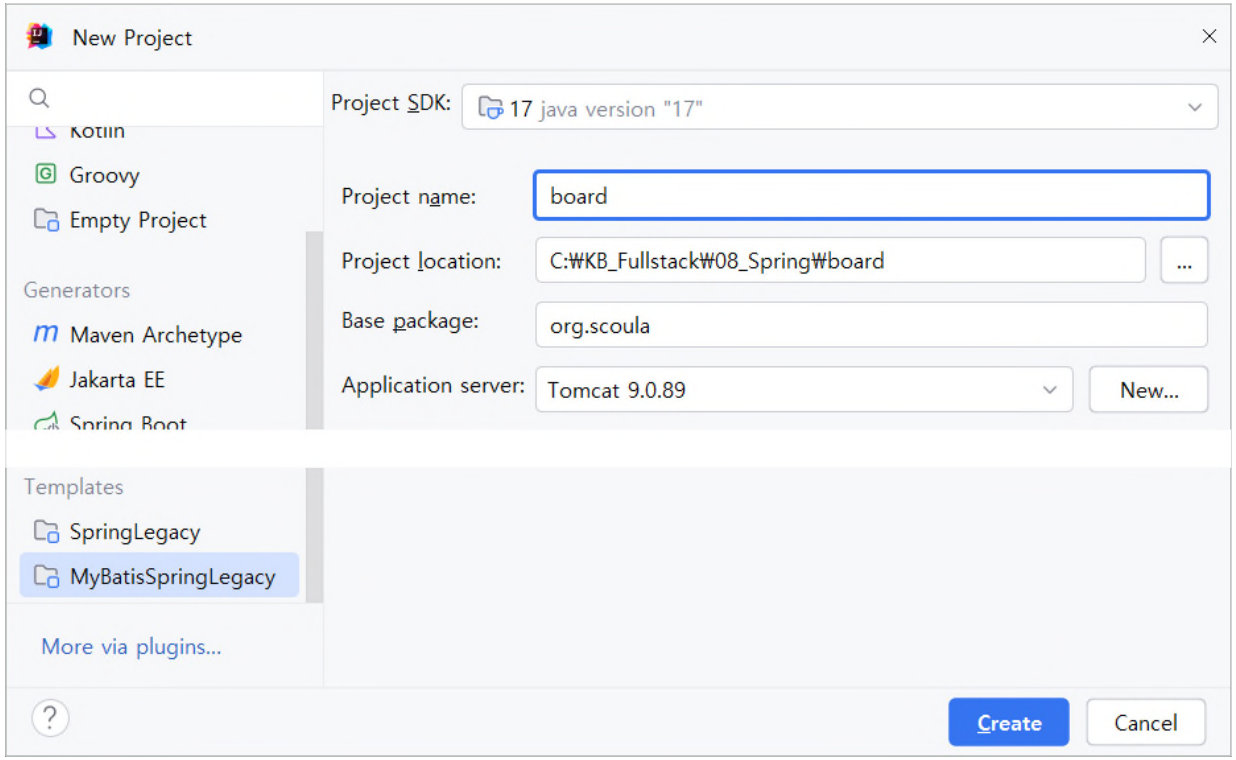
KB 국민은행

프로젝트 만들기

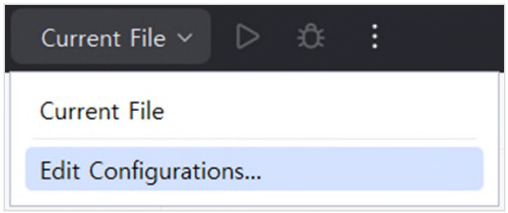
- template: MyBatisSpringLegacy
- project name: board

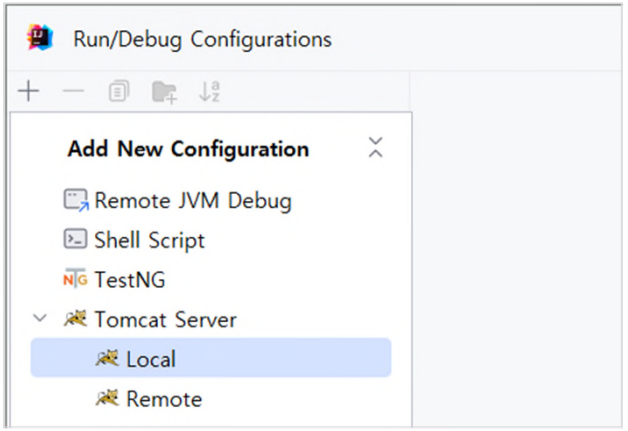
settings.gradle

```
rootProject.name = 'board'
```

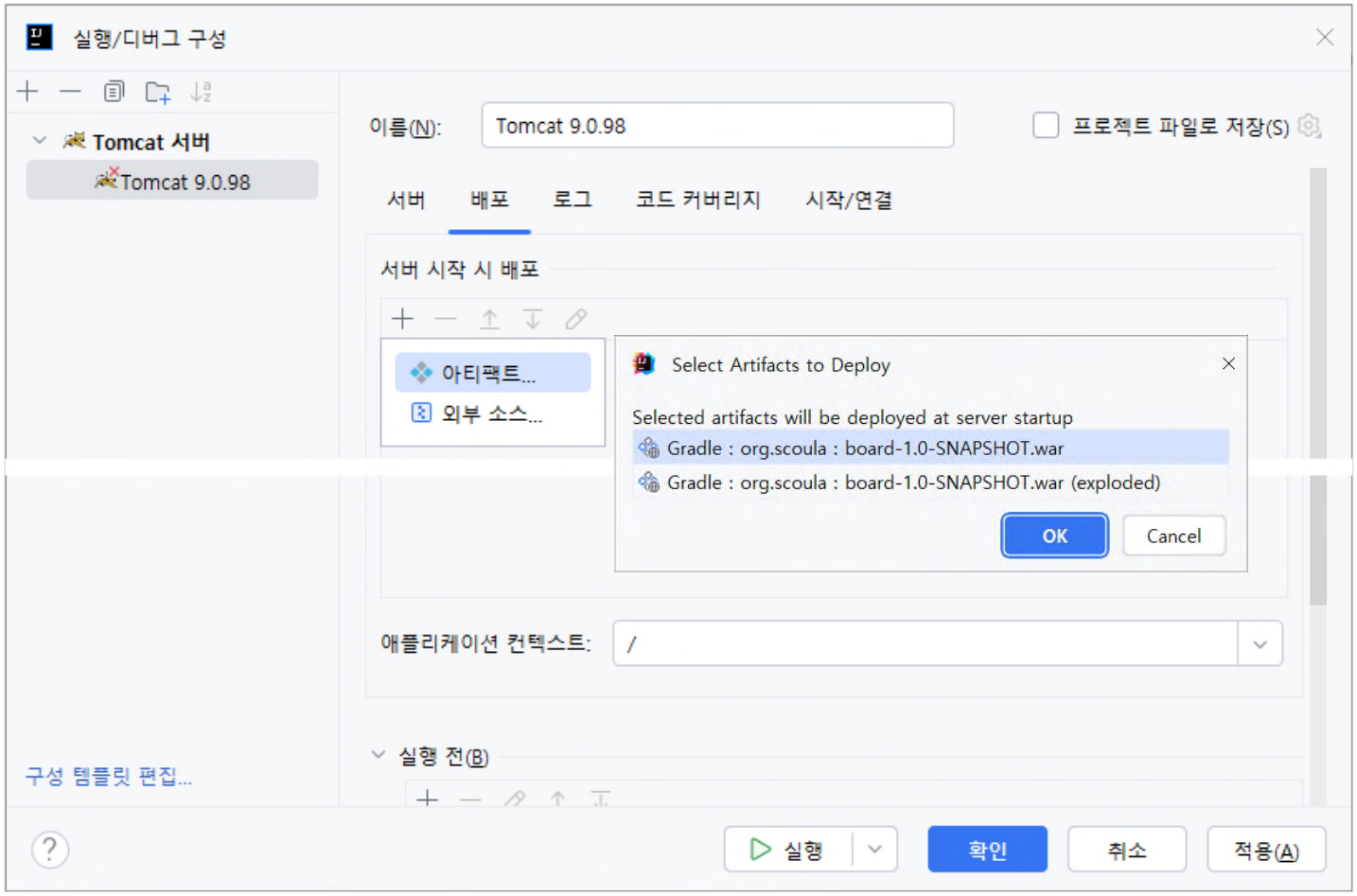


Tomcat 설정


Current File
Current File
Edit Configurations...


Run/Debug Configurations
Add New Configuration
Remote JVM Debug
Shell Script
TestNG
Tomcat Server
Local
Remote

Tomcat 설정

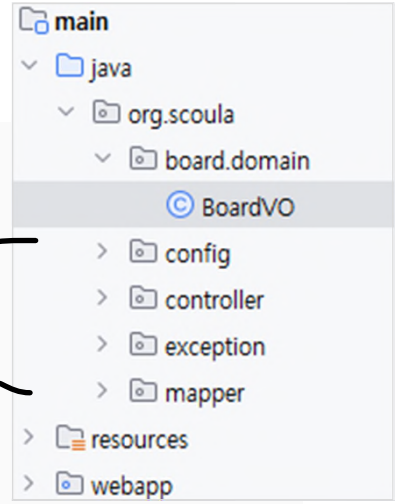

실행/디버그 구성
Tomcat 서버
Tomcat 9.0.98
이름(N): Tomcat 9.0.98
프로젝트 파일로 저장(S)
서버
배포
로그
코드 커버리지
시작/연결
서버 시작 시 배포
아티팩트...
외부 소스...
Select Artifacts to Deploy
Selected artifacts will be deployed at server startup
Gradle : org.scoula : board-1.0-SNAPSHOT.war
Gradle : org.scoula : board-1.0-SNAPSHOT.war (exploded)
OK
Cancel
애플리케이션 컨텍스트: /
실행 전(B)
구성 템플릿 편집...
실행
확인
취소
적용(A)

database.sql – 테이블 만들기

```
DROP TABLE IF EXISTS  tbl_board;

CREATE TABLE tbl_board (
        no                      INTEGER AUTO_INCREMENT PRIMARY KEY,
        title                   VARCHAR(200) NOT NULL,
        content                 TEXT,
        writer                  VARCHAR(50) NOT NULL,
        reg_date                DATETIME DEFAULT CURRENT_TIMESTAMP,
        update_date             DATETIME DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO tbl_board(title, content, writer)
VALUES
        ('테스트 제목1', '테스트 내용1', 'user00'),
        ('테스트 제목2', '테스트 내용2', 'user00'),
        ('테스트 제목3', '테스트 내용3', 'user00'),
        ('테스트 제목4', '테스트 내용4', 'user00'),
        ('테스트 제목5', '테스트 내용5', 'user00');


SELECT * FROM tbl_board;
```

vo정의
=>
class Board{
no
title
regDate//캐멀케이스로 하는것이 좋다
//하지만 이름이 다른경우
//자동화할때 충돌이 발생한다.
//이걸 mybatis에게알려줘야한다
//data에서는 snake표기인데
//java에서는 camelCase
}

BoardVO.java



공통

```
package org.scoula.board.domain;

import java.util.Date;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class BoardVO {
   private Long no;
   private String title;
   private String content;
   private String writer;
   private Date regDate;
   private Date updateDate;
}
```

AllArgConstructor에 준해서 빌더가 형성된단

null을 배정하기 위해서 Long 래퍼 타입으로 long이면 0과 null이 혼용될수있으므로
null은 db에서 아직 안가져온상태를 0이면 값이 0인상태를 .

mybatis에게 알려야하는 이름 규칙

VO 객체의 프로퍼티명과 테이블 컬럼명의 불일치

- updateDate → update_date

- MyBatis 설정 파일에서 설정

resources/mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
      "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>
   <settings>
      <setting name="mapUnderscoreToCamelCase" value="true" />
   </settings>
</configuration>
```

아까말한 이름 규칙 알려줌

MyBatisSpringLegacy 템플릿에 반영

Mapper 인터페이스와 Mapper XML

BoardMapper.java

```
package org.scoula.board.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.scoula.board.domain.BoardVO;

public interface BoardMapper {

  @Select("select * from tbl_board order by no desc")
  public List<BoardVO> getList();

}
```

SQL매퍼 작성

인터페이스 작성!

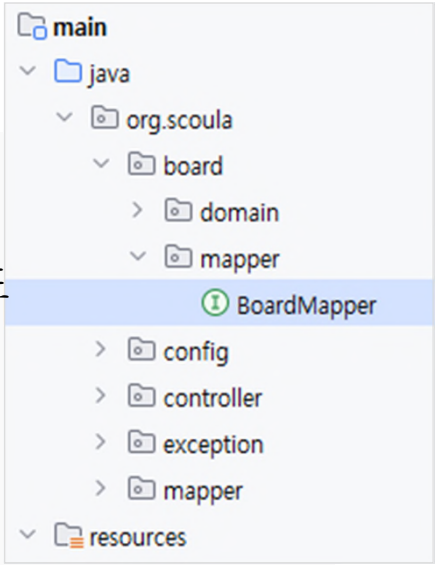
wls원래는 .xml까지 만들어
분리해야함

게시판 운영시 가장 최근에 등록한 글이
먼저 나오게

워래는 orderby reg-date desc여야 하는데
이렇게 하면 해당 쿼리를 실해할때 매번 해당 컬럼에해새ㅓ 정렬하는
오버헤드가 생김

클러스터링 인덱스인 프라이머리 키인 no에다가 하면
위에서 말한 정렬에 의한 오버헤드가 안생긴다.

혹은 다른 인덱스가 있는 칼럼을 정렬하는 것이 좋다



매퍼 역할 코드

SQLSession이 래핑해서
리턴타입을 보고
올바른 메서드를 고름
select머시기

지금은 리스트이므로
selectList메서드를 쓰겠지

내부에서는 setNo setTitle set필드명 세터매서드를 통해서 전달 매핑된다.
즉 vo에는 세터가 있어야 한다!

RootConfig.java

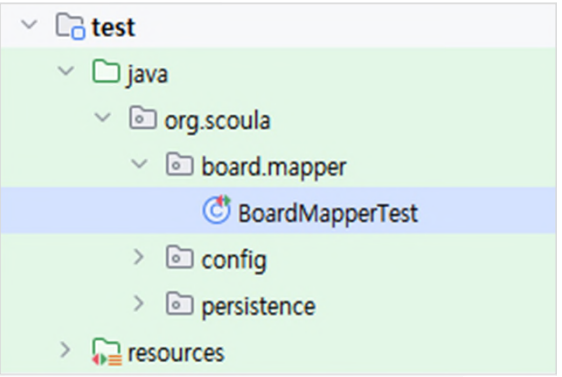
```
…

@Configuration
@MapperScan(basePackages  = {"org.scoula.board.mapper"})
public class RootConfig {
                …
}
```

매퍼를 스캔할 수 있게 rootconfig

rootconfig는 범용이라고 했지 그리고 거기엔 DB관련도 들어있고

test :: BoardMapper.java

cntrl shft t




```
package org.scoula.board.mapper;
...

@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = { RootConfig.class })
@Log4j2
public class BoardMapperTest {

 @Autowired
 private BoardMapper mapper;

 @Test
 @DisplayName("BoardMapper의 목록 불러오기")
 public void getList() {
        for(BoardVO board : mapper.getList()) {
                log.info(board);
        }
 }
}
```

테스트 해보자

없는 경우도 리스트형태로 나가니 물론 비어있음

그래서 비어있는 것을 체크하는 코드 없어도 됨

row행 => board

...

INFO jdbc.sqlonly(sqlOccurred:228) - **select * from tbl_board order by no desc**

INFO org.scoula.board.mapper.BoardMapperTest(getList:28) - BoardVO(no=5, title=테스트 제목5, content=테스트 내용5, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)

INFO org.scoula.board.mapper.BoardMapperTest(getList:28) - BoardVO(no=4, title=테스트 제목4, content=테스트 내용4, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)

INFO org.scoula.board.mapper.BoardMapperTest(getList:28) - BoardVO(no=3, title=테스트 제목3, content=테스트 내용3, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)

INFO org.scoula.board.mapper.BoardMapperTest(getList:28) - BoardVO(no=2, title=테스트 제목2, content=테스트 내용2, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)

INFO org.scoula.board.mapper.BoardMapperTest(getList:28) - BoardVO(no=1, title=테스트 제목1, content=테스트 내용1, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)

컬럼명하고 vo의 필드명이 동일하다. 그 과정에서 이름이 다른 부분은 우리가 알려줬고

마이바티스 채고!

좀 더 복잡한 쿼리 쿼리를 xml로 빼자

2 영속 계층의 구현 준비

Mapper XML

- 복잡한 쿼리인 경우 @Select()로 구성하는 것은 매우 어려움
- resources 영역에 쿼리를 xml로 작성하여 Mapper 인터페이스와 연계
- Mapper 인터페이스의 패키지 경로 동일하게 동일 파일명으로 작성

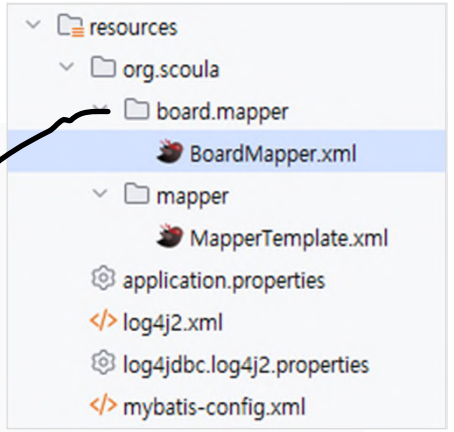
BoardMapper.java

```
package org.scoula.board.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.scoula.board.domain.BoardVO;

public interface BoardMapper {

 // @Select("select * from tbl_board")
 public List<BoardVO> getList();

}
```

디렉토리 구조를
프로젝트->
프로젝트 파일
왔다갔다하며

KB국민은행

resources :: BoardMapper.xml (MapperTemplate.xml을 복사해서 생성)




```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="org.scoula.board.mapper.BoardMapper">

        <select id="getList" resultType="org.scoula.board.domain.BoardVO">
                <![CDATA[
                        select * from tbl_board
                        order by no desc
                ]]>
        </select>

</mapper>
```

namespace지정해야죠?

경로를 .으로 해야함 / 말고

CDATA == Compiled DATA. 이미 컴파일됐다 그냥 그대로 사용해라 라는 뜻. xml파서가 부등호 같은 경우를 <나 >이거를 태그의 시작 끝점으로 인식함! 그래서 사용하는 거야.

- <![CDATA[SQL 문자열]]>
  - Compiled Data: 이미 컴파일한 데이터임을 나타냄
    - 문자열 속 <, > 태그 글자가 무시됨
  - 태그 문자가 없다면 생략 가능

ControllerAdvice에서 도 mapperscan경로 바꾸고

Type Alias

<select id="getList" resultType="org.scoula.board.domain.BoardVO">

→ mybatis-config.xml에 단축표현을 위한 설정

<select id="getList" resultType="BoardVO">

resources :: /mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
      "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>
   <settings>
      <setting name="mapUnderscoreToCamelCase" value="true" />
   </settings>

   <typeAliases>
      <package name="org.scoula.board.domain" />
   </typeAliases>

</configuration>
```

resources :: BoardMapper.xml

분리한 sql쿼리 적는 곳!

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="org.scoula.board.mapper.BoardMapper">

                <select id="getList" resultType="BoardVO">
                                <![CDATA[
                                select * from tbl_board
                                ]]>
                </select>

</mapper>
```

BoardMapper.java – get(select) 처리

```
package org.scoula.board.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.scoula.board.domain.BoardVO;

public interface BoardMapper {

  public List<BoardVO> getList();

  public BoardVO get(Long no);
}
```

새로운 매퍼

해당 정보는 where no = ? 에서 ?에 들어가야하는

resources :: BoardMapper.xml - get(select) 처리

```
...
<mapper namespace="org.scoula.board.mapper.BoardMapper">
  ...

        <select id="get" resultType="org.scoula.board.domain.BoardVO">
                select * from tbl_board where no = #{no}
        </select>

</mapper>
```

? 대신에 #{ 매개변수명칭 }

내부에서
sql연산이름|갯수이름(쿼리,obj)시그니처를 가진다.

select???(query,obj)
insert???(query,obj)
...

~~test~~ :: BoardMapper.java – get(select) 처리

```
...
public class BoardMapperTests {
  ...

  @Test
  @DisplayName("BoardMapper의 게시글 읽기")
  public void get() {
                // 존재하는 게시물 번호로 테스트
                BoardVO board = mapper.get(1L);

                log.info(board);

  }
}
```

```
...
 INFO  jdbc.sqlonly(sqlOccurred:228) - select * from tbl_board where no = 1

 INFO  org.scoula.board.mapper.BoardMapperTest(get:38) - BoardVO(no=1, title=테스트 제목1, content=테스트 내용1, writer=user00, regDate=Wed Jan 15 11:53:10 KST 2025, updateDate=Wed Jan 15 11:53:10 KST 2025)
...
```

create(insert) 처리

- 시퀀스로 PK가 자동으로 정해지는 경우
- PK 처리 방식
  - insert만 처리되고 생성된 PK 값을 알 필요가 없는 경우
  - insert 문이 실행되고 생성된 PK 값을 알아야 하는 경우
    → insert후 그 키로 다른 테이블에 FK로 후속 작업을 해야 하는 경우
    예) 첨부파일 저장

BoardMapper.java

```
package org.scoula.board.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.scoula.board.domain.BoardVO;

public interface BoardMapper {

  public List<BoardVO> getList();

  public BoardVO get(Long no);

  public void create(BoardVO board);

}
```

인터페이스에 관련 매서드 선언

매개변수 유형

단일 값, vo객체, map

resources :: BoardMapper.xml

```
...
<mapper namespace="org.scoula.board.mapper.BoardMapper">
  ...

        <insert id="create">
                insert into tbl_board (title, content, writer)
                values (#{title}, #{content}, #{writer})
        </insert>

</mapper>
```

어떻게 해석될까

VO객체가 매개변수로 넘어갔다. 그게 나뉘어 사용되고있다

#{ vo객체의 프로퍼티명 }으로 봐야한다.
getVO객체프로퍼티명 같은 세터메서드가 호출.

vo객체가 넘어왔을때.

단일값이나 맵 타입인 경우는 또 다르다

맵타입인 경우
#{ 맵의 키 } 이고 get(키)호출된다

test :: BoardMapper.java

```
...
public class BoardMapperTests {
  ...
  @Test
  @DisplayName("BoardMapper의 새글 작성")
  public void create() {

    BoardVO board = new BoardVO();
    board.setTitle("새로 작성하는 글");
    board.setContent("새로 작성하는 내용");
    board.setWriter("user0");

    mapper.create(board);

    log.info(board);
  }
}
```

create(insert)하기전 데이터 설정

매퍼를 통해 쉽게 db에 넣어주기!

#{ title } #{ content } #{user}

```
INFO  jdbc.sqlonly(sqlOccurred:228) - insert into tbl_board (title, content, writer) values ('새로 작성하는 글', '새로 작성하는 내용', 'user0')

INFO  org.scoula.board.mapper.BoardMapperTest(create:53) - BoardVO(no=null, title=새로 작성하는 글, content=새로 작성하는 내용, writer=user0, regDate=null, updateDate=null)
```

아 넣었을때 자동 배정된 no를 알아야

fk로 하는 테이블에 no정보를 넣을수 잇는데

어떻게no를 알지 selectKey사용

<selectKey>

- SQL이 실행되기 전에 별도의 PK값 등을 얻기 위해서 사용
- order='before' 를 이용해서 insert구문이 실행되기 전에 호출
- keyProperty를 통해 BoardVO의 no값으로 세팅

속성	설명
keyProperty	PK 값을 읽어 설정할 vo객체의 프로퍼티명
keyColumn	테이블의 PK 컬럼명
resultType	결과의 타입. vo 객체의 프로퍼티 타입 지정
order	selectKey 구문을 언제 실행할 지 지정 - BEFORE: insert 문 실행 전에 selectKey 구문 실행 - AFTER: insert 문 실행 후에 selectKey 구문 실행
statementType	STATEMENT, PREPARED 또는 CALLABLE중 하나를 선택. 디폴트는 PREPARED

resources :: BoardMapper.xml

```
...
<mapper namespace="org.scoula.board.mapper.BoardMapper">
  ...

        <insert id="create">
                insert into tbl_board (title, content, writer)
                values (#{title}, #{content}, #{writer})

                <selectKey resultType="Long" keyProperty="no" keyColumn="no" order="AFTER">
                        SELECT LAST_INSERT_ID()
                </selectKey>

        </insert>
</mapper>
```

setNo함수를 써라!
vo상에서 no가 아이디다
어떤 값으로set해
테이블의 no칼럼인 값을

안에서다가 작성

MYSQL함수를 호출
마지막 insert할때 사용했던 ID값이 뭐냐

mysql은 insert하고 난후 auto_increment가 발동하여 배정된다.

oracle은 insert하기 직전 발동되어 배정됨

거의 대부분 필요하다
insert할때

test :: BoardMapper.java

```
...
public class BoardMapperTests {
  ...
  @Test
  @DisplayName("BoardMapper의 새글 작성")
  public void create() {

    BoardVO board = new BoardVO();
    board.setTitle("새로 작성하는 글");
    board.setContent("새로 작성하는 내용");
    board.setWriter("user0");

    mapper.create(board);

    log.info(board);
  }
}
```

다시 테스트해보면 no의 값이 nul대신에 발급받은 넘버가 있음

이 정보로 후속 작업을 할 숭 있다.

```
INFO  jdbc.sqlonly(sqlOccurred:228) - insert into tbl_board (title, content, writer) values ('새로 작성하는 글', '새로 작성하는 내용', 'user0')

INFO  jdbc.sqlonly(sqlOccurred:228) - SELECT LAST_INSERT_ID()

INFO  org.scoula.board.mapper.BoardMapperTest(create:53) - BoardVO(no=7, title=새로 작성하는 글, content=새로 작성하는 내용, writer=user
0, regDate=null, updateDate=null)
```

update 처리

인터페이스에 메서드 추가!

그리고

맞는 xml 구성

BoardMapper.java - update 처리

```
package org.scoula.board.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.scoula.board.domain.BoardVO;

public interface BoardMapper {

  public List<BoardVO> getList();

  public BoardVO get(Long no);

  public void create(BoardVO board);

  public int update(BoardVO board);
}
```

mybatisx플러그인으로 xml자동 생성하고잉

몇개 업데이트 됐는지

resources :: BoardMapper.xml – update 처리

```
...
<mapper namespace="org.scoula.board.mapper.BoardMapper">
  ...

        <update id="update">
                update tbl_board set
                        title = #{title},
                        content = #{content},
                        writer = #{writer},
                        update_date = now()
                where no = #{no}
        </update>

</mapper>
```

dml쿼리의 결과는 항상 integer이므로 따로 반환 관련 코드 필요X
인터페이스의 메서드의 반환값이 void여도 상관없음 사용할 필요 없다면

똑같애
get프로퍼티명 메소드 호출.

내부에서는
preparestatement에서는 저게다 ? 로 바뀌고 동작됨

mysql함수를 사용했네

test :: BoardMapper.java - update 처리

```
...
public class BoardMapperTests {
  ...
  @Test
  @DisplayName("BoardMapper의 글 수정")
  public void update() {

    BoardVO board = new BoardVO();
    board.setNo(5L);
    board.setTitle("수정된 제목");
    board.setContent("수정된 내용");
    board.setWriter("user00");

    int count = mapper.update(board);

    log.info("UPDATE COUNT: " + count);
  }
}
```

```
 INFO  jdbc.sqlonly(sqlOccurred:228) - update tbl_board set title = '수정된 제목', content = '수정된 내용', writer = 'user00', update_date = now()
where no = 5

 INFO  org.scoula.board.mapper.BoardMapperTest(update:68) - UPDATE COUNT: 1
```

- delete 처리

BoardMapper.java – delete 처리

```
package org.scoula.board.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Select;
import org.scoula.board.domain.BoardVO;

public interface BoardMapper {
  public List<BoardVO> getList();

  public BoardVO get(Long no);

  public void create(BoardVO board);

  public int update(BoardVO board);

  public int delete(Long no);
}
```

마찬가지

resources :: BoardMapper.xml – delete 처리

```
...
<mapper namespace="org.scoula.board.mapper.BoardMapper">
  ...

        <delete id="delete">
                delete from tbl_board where no = #{no}
        </delete>

</mapper>
```

test :: BoardMapper.java – delete 처리

```
...
public class BoardMapperTests {
  ...
  @Test
  @DisplayName("BoardMapper의 글 삭제")
  public void delete() {

                log.info("DELETE COUNT: " + mapper.delete(3L));

  }
}
```

```
INFO  jdbc.sqlonly(sqlOccurred:228) - delete from tbl_board where no = 3

INFO  org.scoula.board.mapper.BoardMapperTest(delete:74) - DELETE COUNT: 1
```