

# 영속 / 비즈니스 계층의 CRUD 구현

테이블의 컬럼 구조를 바느 꺾는 VO(Value Object) 클래스 생성

MyBatis의 Mapper 인터페이스의 작성 처리 및 XML 처리

작성한 Mapper 인터페이스의 테스트

위의 과정 전에

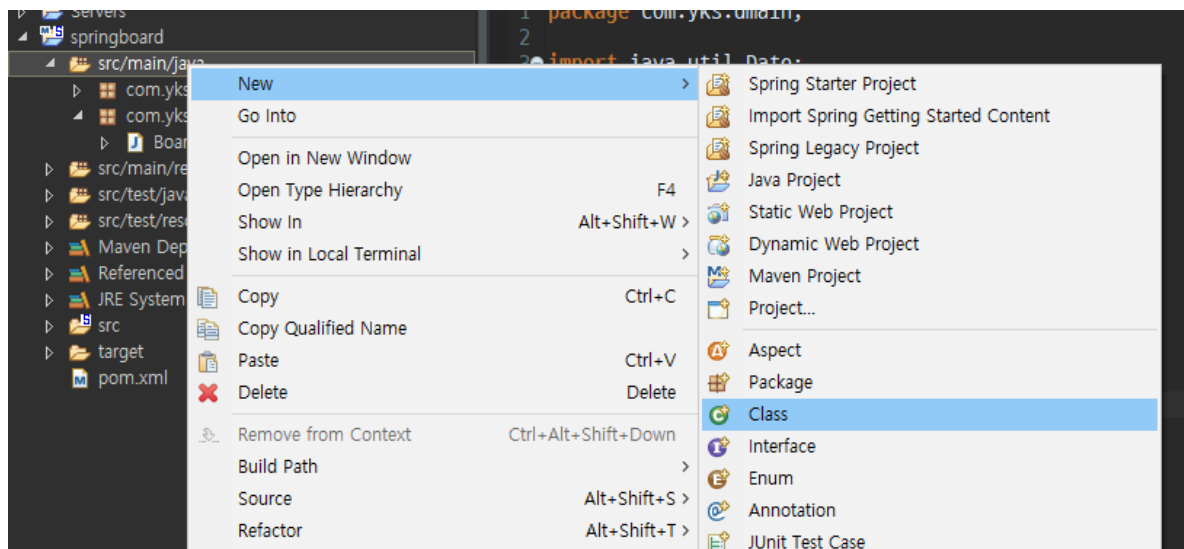
먼저 JDBC 연결을 테스트하는 과정을 반드시 거친다.

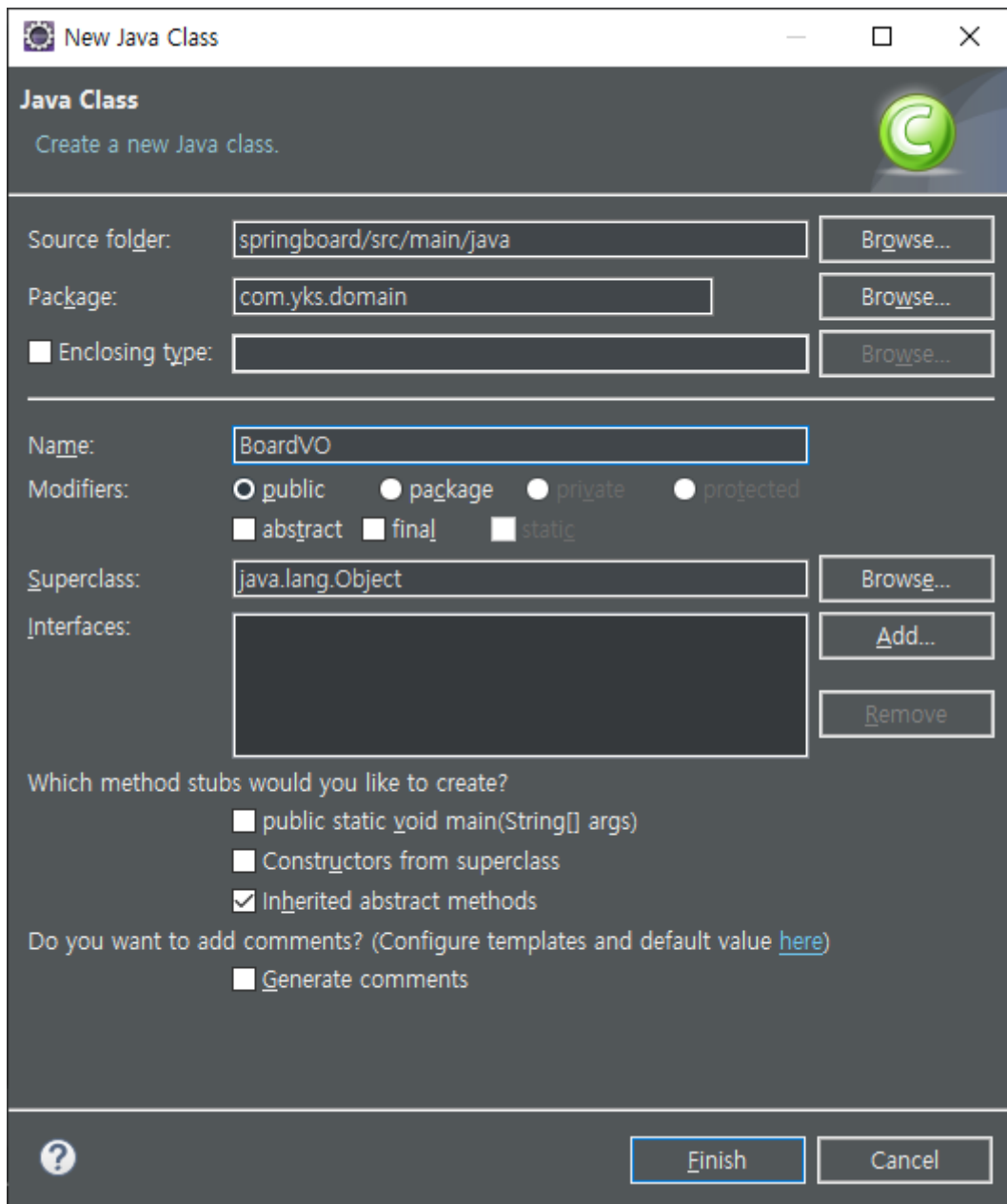
## 1. 영속 계층의 구현 준비

거의 모든 웹 애플리케이션의 최종 목적은

데이터 베이스에 데이터를 기록하거나,

원하는 데이터를 가져오는 것이 목적이기 때문



The image shows a 'New Java Class' dialog box from an IDE. It has a title bar with a gear icon and standard window controls. The main area is dark gray with white text. At the top, it says 'Java Class' and 'Create a new Java class.' with a green 'C' icon. Below this are fields for 'Source folder' (springboard/src/main/java), 'Package' (com.yks.domain), and 'Enclosing type' (empty), each with a 'Browse...' button. The 'Name' field contains 'BoardVO'. Under 'Modifiers', there are radio buttons for 'public' (selected), 'package', 'private', and 'protected', and checkboxes for 'abstract', 'final', and 'static'. The 'Superclass' field contains 'java.lang.Object' with a 'Browse...' button. The 'Interfaces' field is empty with 'Add...' and 'Remove' buttons. A section titled 'Which method stubs would you like to create?' has checkboxes for 'public static void main(String[] args)', 'Constructors from superclass', and 'Inherited abstract methods' (checked). Below this is a question 'Do you want to add comments?' with a 'Generate comments' checkbox. At the bottom are a help icon, 'Finish', and 'Cancel' buttons.

New Java Class

Java Class  
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

```
/springboard/src/main/java/com/yks/domain/BoardVO.java
```

```
package com.yks.domain;
```

```
import java.util.Date;
```

```
import Lombok.Data;
```

```
@Data
```

```
public class BoardVO {
```

```
    private Long bno;
```

```
    private String title;
```

```
    private String content;
```

```
    private String writer;
```

```
    private Date regdate;
```

```
    private Date updateDate;
```

```
}
```

## 2. Mapper 인터페이스의 Mapper XML

MyBatis는 SQL을 처리하는데, 어노테이션이나 XML을 이용할 수 없다.

간단한 SQL일 경우 : 어노테이션을 이용하여 처리

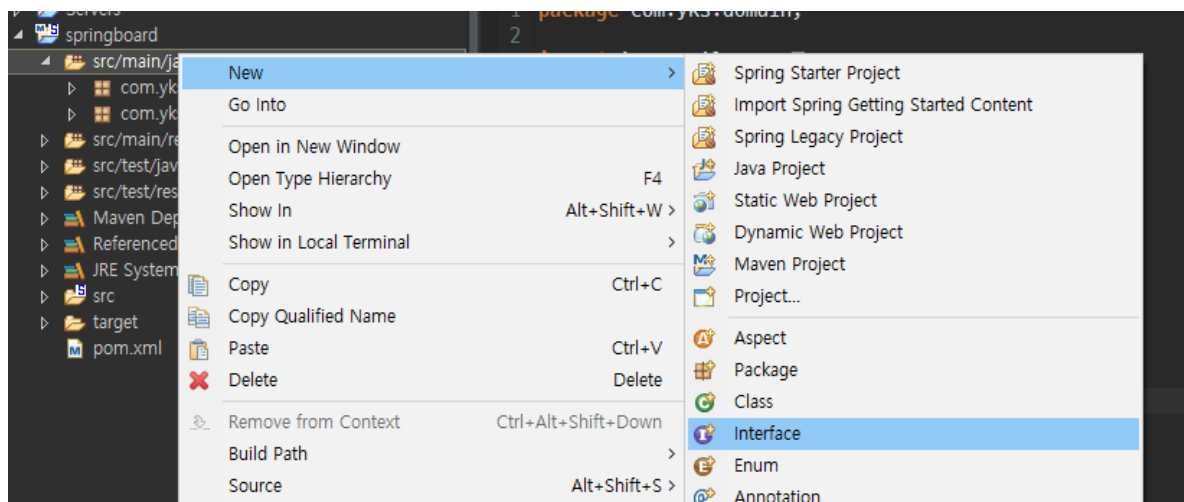
복잡하거나 검색 SQL : XQML 로 처리

XML의 경우

단순 텍스트를 수정하는 과정만으로 처리가 끝나지만,

어노테이션의 경우

코드를 수정하고, 다시 빌드하는 등, 유지 보수성이 떨어진다.



## BoardMapper 인터페이스 작성시

필요한 SQL을 어노테이션으로 속성값으로 처리할 수 있다.

주의사항 : SQL 작성할때 ';' 이 없도록 작성해야 한다.

select 문 뒤에 bno 컬럼 조건을 주어서 Primary Key(PK) 를 이용하도록 한다.

## SQL Developer 에서 먼저

SQL이 문제 없이 실행 가능한지를 확인

데이터베이스의 commit 여부 확인

```
/springboard/src/main/java/com/yks/mapper/BoardMapper.java

package com.yks.mapper;

import java.util.List;

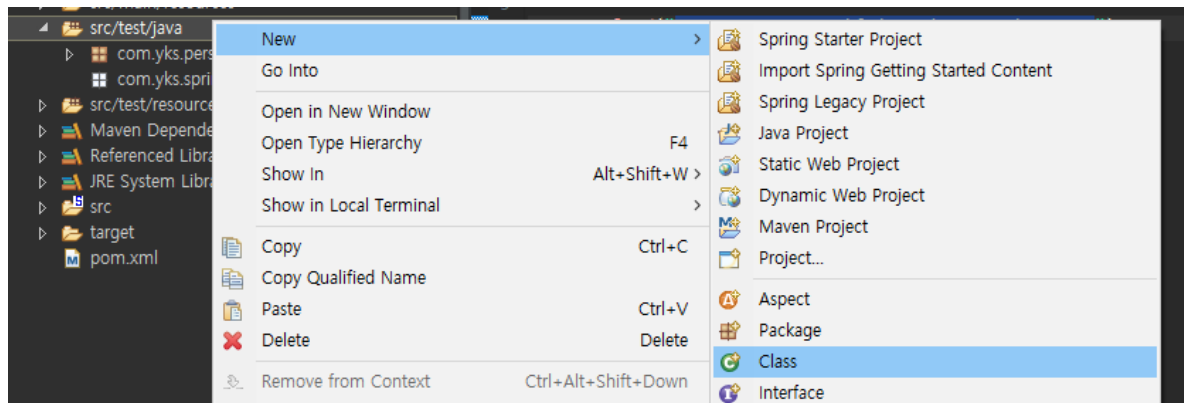
import org.apache.ibatis.annotations.Select;
import com.yks.domain.BoardVO;

public interface BoardMapper {

    @Select("SELECT * FROM tbl_board WHERE bno > 0")
    public List<BoardVO> getList();
}
```

```
}
```

## 테스트 클래스 만들기



**New Java Class**

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☐ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

```
/springboard/src/test/java/com/yks/mapper/BoardMapperTests.java
```

```
package com.yks.mapper;
```

```

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import lombok.Setter;
import lombok.extern.log4j.Log4j;

public class BoardMapperTests {

}

```

## 테스트 주입을 위한 어노테이션

```

/springboard/src/test/java/com/yks/mapper/BoardMapperTests.java

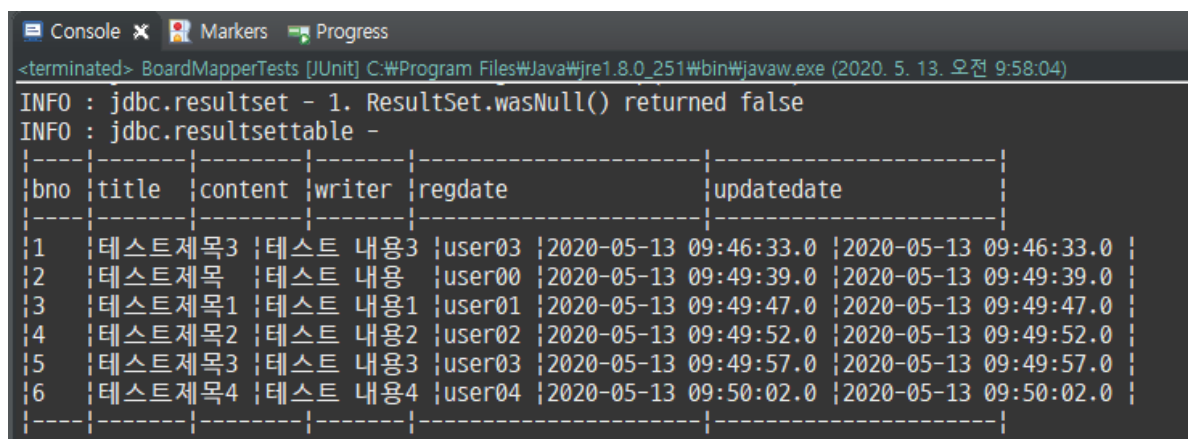
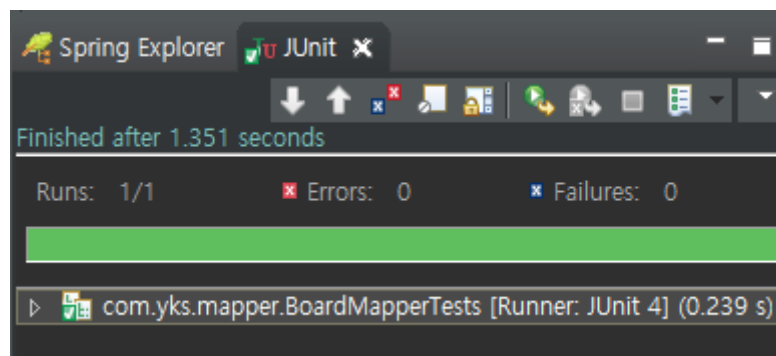
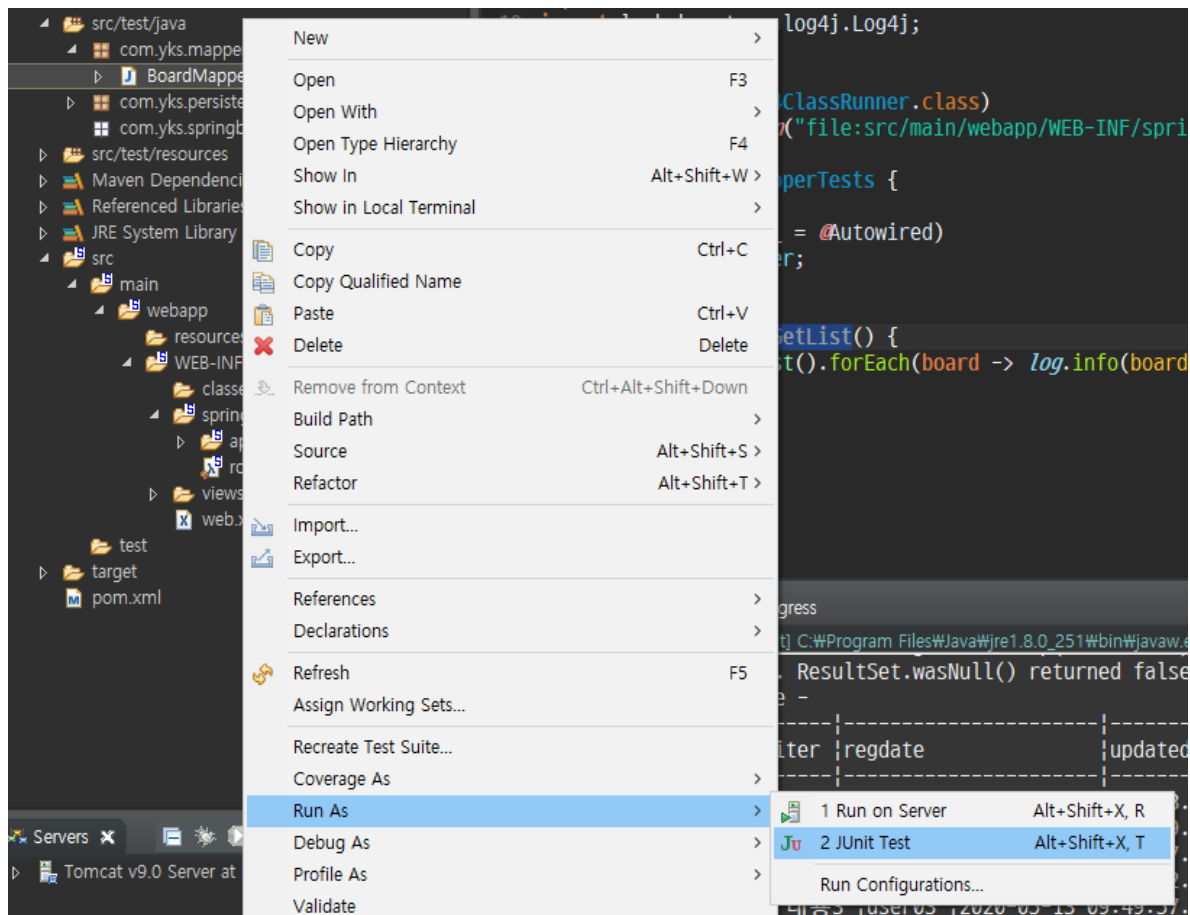
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml")
@Log4j
public class BoardMapperTests {

    @Setter(onMethod_ = @Autowired)
    BoardMapper mapper;

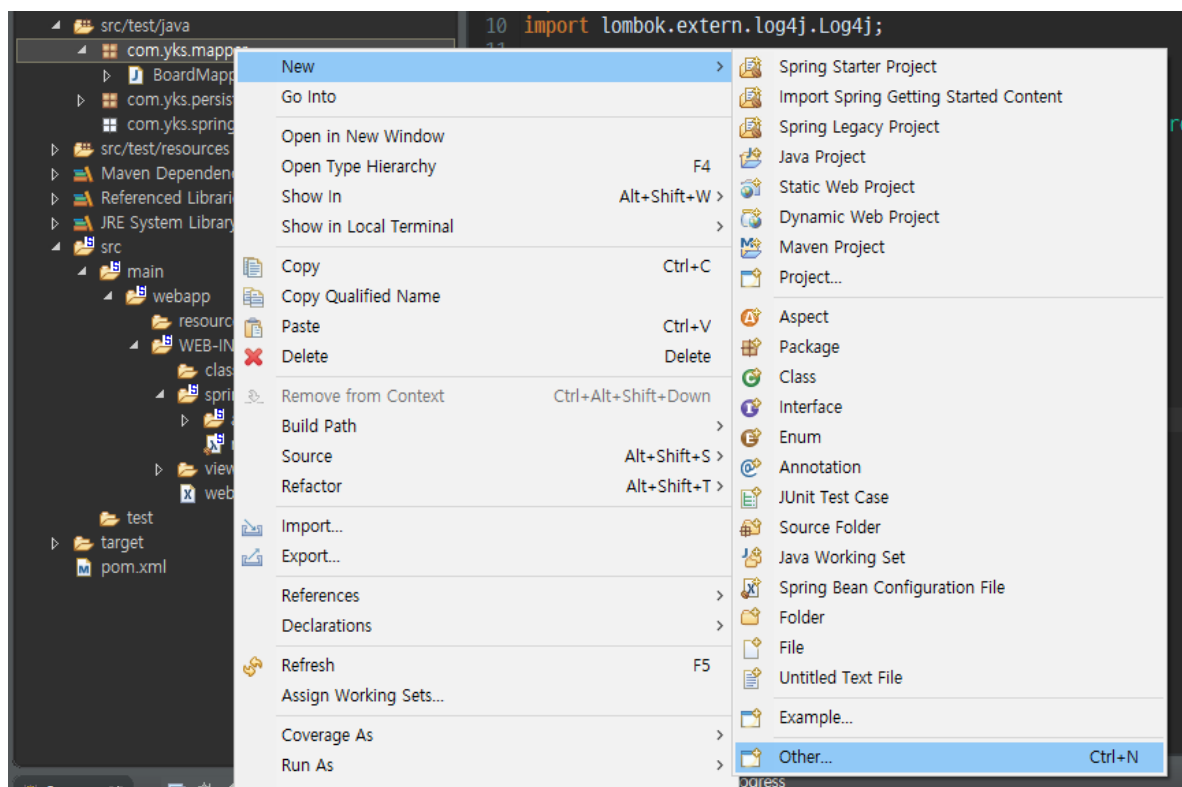
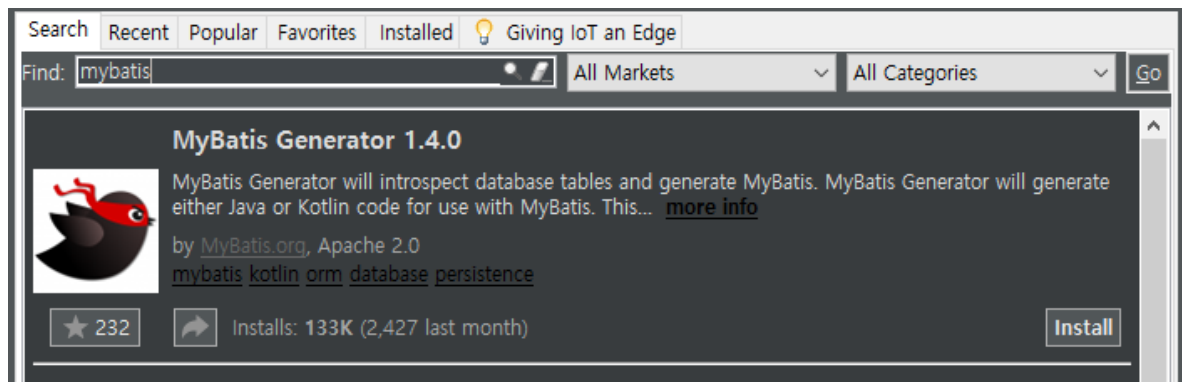
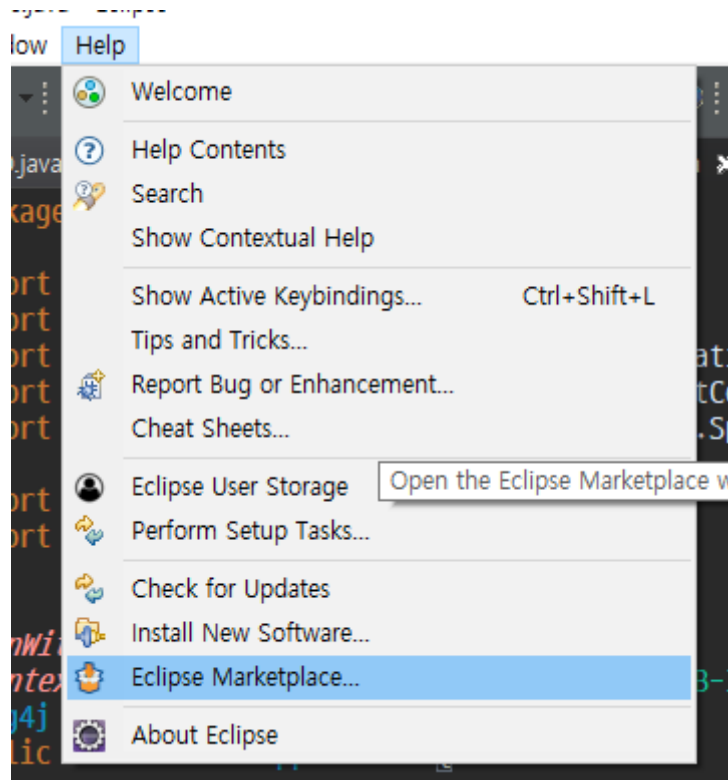
    @Test
    public void testGetList() {
        mapper.getList().forEach(board -> log.info(board));
    }

}

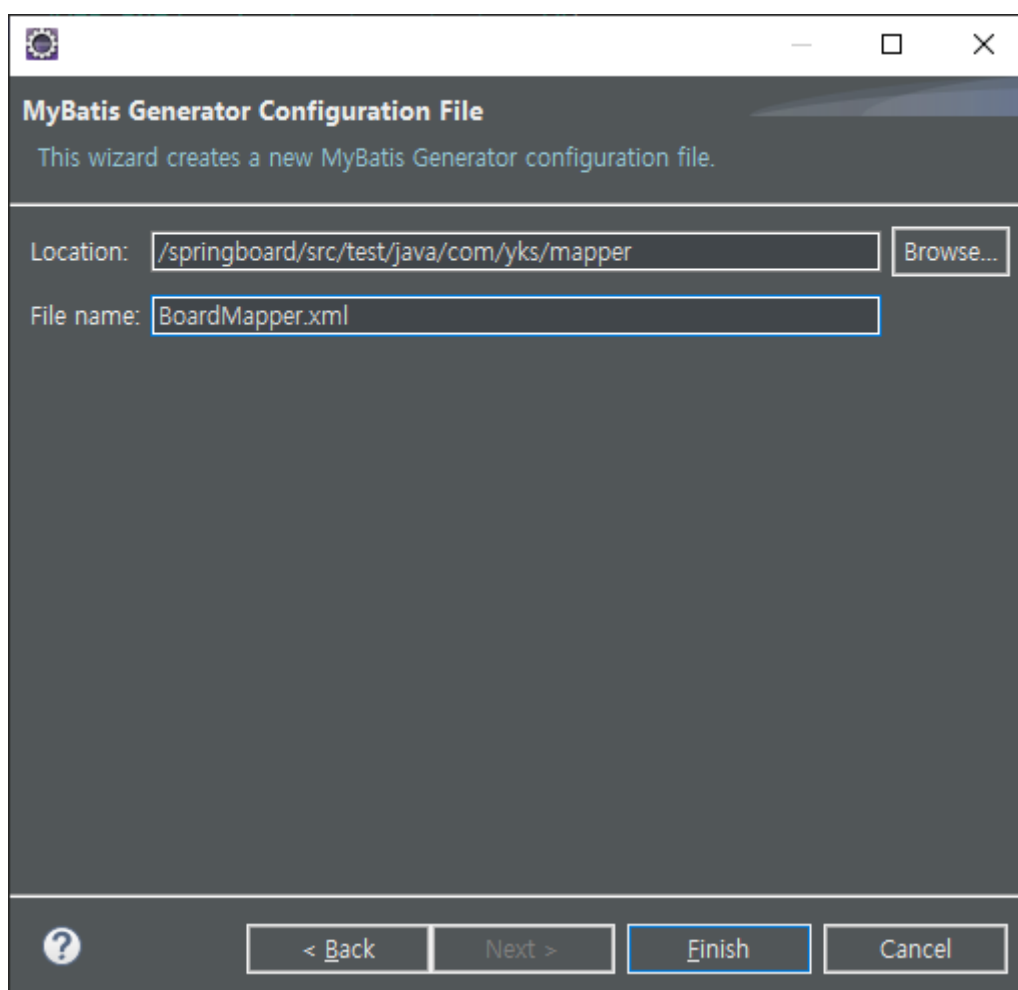
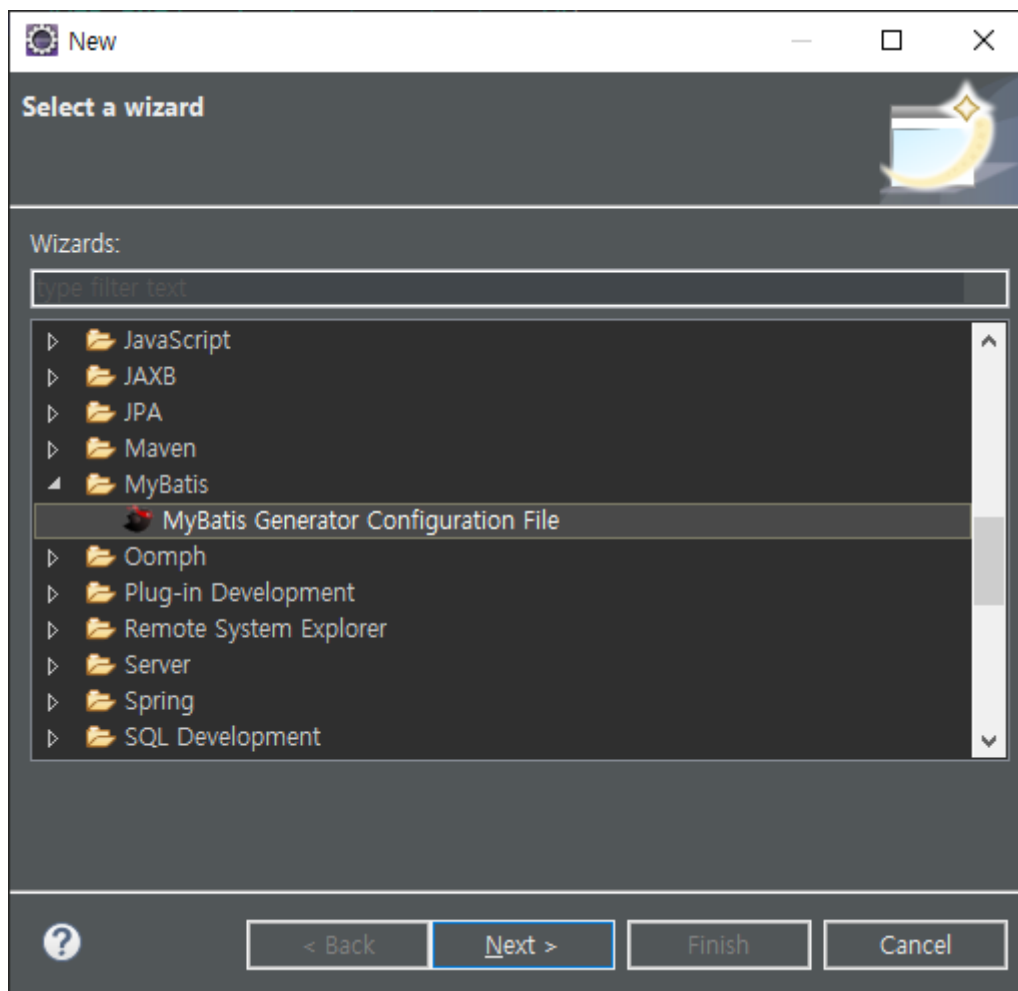
```



## Mapper XML 문서







아래와 같이 수정한다.

```

/springboard/src/test/java/com/yks/mapper/BoardMapper.xml

<?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">    # 이부분 수정 나머지는 주석처리
<!--
<generatorConfiguration>
  <context id="context1">
    <jdbcConnection connectionURL="???" driverClass="???" password="???"
    userId="???" />
    <javaModelGenerator targetPackage="???" targetProject="???" />
    <javaClientGenerator targetPackage="???" targetProject="???"
    type="XMLMAPPER" />
    <table schema="???" tableName="???">
      <columnOverride column="???" property="???" />
    </table>
  </context>
</generatorConfiguration>
-->

```

XML을 작성할때 주의사항

`<mapper>`의 namespace 속성값 : Mapper 인터페이스와 동일한 이름

`<select>` 태그의 id 속성값 : 메서드 이름과 일치

resultType 속성값 : select 쿼리의 결과를 특정 클래스의 객체로 자동 생성하기 위함

아래를 주석 다음에 추가

```

<mapper namespace="com.yks.mapper.BoardMapper">
  <select id="getList" resultType="com.yks.domain.BoardVO">
    <![CDATA[    # SELECT문 안에 부등호가 태그의 닫는부분으로 보일까봐 CDATA사용
    SELECT * FROM tbl_board WHERE bno > 0
    ]]>
  </select>
</mapper>

```

그리고 mapper 파일을 수정(주석처리)한다.

어노테이션과 중복될수 있기 때문

```

/springboard/src/main/java/com/yks/mapper/BoardMapper.java

package com.yks.mapper;

import java.util.List;

// import org.apache.ibatis.annotations.Select;
import com.yks.domain.BoardVO;

public interface BoardMapper {

```

```
// @Select("SELECT * FROM tbl_board WHERE bno > 0")
public List<BoardVO> getList();

}
```

bno	title	content	writer	regdate	updatedate
1	테스트제목3	테스트 내용3	user03	2020-05-13 09:46:33.0	2020-05-13 09:46:33.0
2	테스트제목	테스트 내용	user00	2020-05-13 09:49:39.0	2020-05-13 09:49:39.0
3	테스트제목1	테스트 내용1	user01	2020-05-13 09:49:47.0	2020-05-13 09:49:47.0
4	테스트제목2	테스트 내용2	user02	2020-05-13 09:49:52.0	2020-05-13 09:49:52.0
5	테스트제목3	테스트 내용3	user03	2020-05-13 09:49:57.0	2020-05-13 09:49:57.0
6	테스트제목4	테스트 내용4	user04	2020-05-13 09:50:02.0	2020-05-13 09:50:02.0

```
INFO : jdbc.resultset - 1. ResultSet.next() returned false
INFO : jdbc.resultset - 1. ResultSet.close() returned void
INFO : jdbc.audit - 1. Connection.getMetaData() returned oracle.jdbc.driver.OracleDatabaseMetaData@a5b0b86
INFO : jdbc.audit - 1. PreparedStatement.isClosed() returned false
INFO : jdbc.audit - 1. PreparedStatement.close() returned
INFO : jdbc.audit - 1. Connection.clearWarnings() returned
INFO : com.yks.mapper.BoardMapperTests - BoardVO(bno=1, title=테스트제목3, content=테스트 내용3, writer=user03, regdate=Wed May 13 09:46:33 KST 2020)
INFO : com.yks.mapper.BoardMapperTests - BoardVO(bno=2, title=테스트제목, content=테스트 내용, writer=user00, regdate=Wed May 13 09:49:39 KST 2020)
INFO : com.yks.mapper.BoardMapperTests - BoardVO(bno=3, title=테스트제목1, content=테스트 내용1, writer=user01, regdate=Wed May 13 09:49:47 KST 2020)
INFO : com.yks.mapper.BoardMapperTests - BoardVO(bno=4, title=테스트제목2, content=테스트 내용2, writer=user02, regdate=Wed May 13 09:49:52 KST 2020)
INFO : com.yks.mapper.BoardMapperTests - BoardVO(bno=5, title=테스트제목3, content=테스트 내용3, writer=user03, regdate=Wed May 13 09:49:57 KST 2020)
INFO : com.yks.mapper.BoardMapperTests - BoardVO(bno=6, title=테스트제목4, content=테스트 내용4, writer=user04, regdate=Wed May 13 09:50:02 KST 2020)
```

MyBatis 는 내부적으로

**JDBC의 PreparedStatement** 를 활용하고 필요한 파라미터를 처리하는 '?' 에대한 치환은 '#{속성}'을 이용하여 처리.

tbl\_board 테이블은 PK 컬럼으로 bno를 이용하고

시퀀스를 이용하여 데이터가 추가될때 자동으로 번호가 만들어지는 방식을 사용

이처럼 자동으로 PK값이 정해지는 경우 처리 방법

insert 만 처리되고 생성된 PK 값을 알 필요가 없는 경우

insert 문이 실행되고 생성된 PK 값을 알아야 할 경우

```
/springboard/src/main/java/com/yks/mapper/BoardMapper.java

package com.yks.mapper;

import java.util.List;

// import org.apache.ibatis.annotations.Select;
import com.yks.domain.BoardVO;

public interface BoardMapper {

// @Select("SELECT * FROM tbl_board WHERE bno > 0")
public List<BoardVO> getList();

public void insert(BoardVO board); // 테이블에 입력할 메서드

public void insertSelectKey(BoardVO board); // insert했을때 bno값을 꺼낼수
있는 메서드

}
```

```

/springboard/src/test/java/com/yks/mapper/BoardMapper.xml

<mapper namespace="com.yks.mapper.BoardMapper">

    <select id="getList" resultType="com.yks.domain.BoardVO">
        <![CDATA[
            SELECT * FROM tbl_board WHERE bno > 0
        ]]>
    </select>

    <insert id="insert">
        INSERT INTO tbl_board (bno, title, content, writer)
        VALUES (seq_board.nextval, #{title}, #{content}, #{writer})
    </insert>

    <insert id="insertSelectKey">

        <selectKey keyProperty="bno" order="BEFORE" resultType="long">
            SELECT seq_board.nextval FROM dual
        </selectKey>
    </insert>

</mapper>

```

## BoardMapper

`<insert id="insert">` : 단순히 시퀀스의 다음 값을 구해서 insert 할 때 사용

insert into SQL 문 :

몇건의 데이터가 변경되었는지 만을 알려주기 때문에

추가된 데이터의 PK 값을 알 수는 없지만,

한번의 SQL 처리만으로 작업이 완료된다는 장점이 있다.

@SelectKey:

주로 PK 값을 미리(before) SQL 문을 통하여 처리해두고,

특정한 이름으로 결과를 보관하는 방식.

@ Insert 할때 SQL 문을 보면 \${bno} 와 같이 이미 처리된 결과를 이용.

아래 내용 추가

```

import com.yks.domain.BoardVO;

@Test
public void testInsert() {
    BoardVO board = new BoardVO();
    board.setTitle("새로 작성하는 글");
}

```

```

        board.setContent("새로 작성하는 내용");
        board.setWriter("newbie");

        mapper.insert(board);

        log.info(board);

        @Test
        public void testInsertSelectKey() {
            BoardVO board = new BoardVO();
            board.setTitle("새로 작성하는 글");
            board.setContent("새로 작성하는 내용");
            board.setWriter("newbie");

            mapper.insert(board);

            log.info(board);
        }
    }
}

```

테스트 코드의

각 메서드에서 마지막에 log.info(board)를 작성해야 하는 이유:

Lombok이 만들어주는 toString()을 이용하여 bno 멤버변수(인스턴스 변수)의 값을 알아보기 위함.

테스트 코드의 마지막 부분을 보면

BoardVO객체의 bno값이 지정된 것을 확인할 수 있다.

- 참고 : 시퀀스의 값이므로 테스트 할 때마다 다른 값이 나온다.

시퀀스 값 : 중복 없는 값을 위한 것일 뿐 다른 의미는 없다.

@SelectKey를 이용하는 방식은

SQL을 한 번 더 실행하는 부담이 있기는 하지만,

자동으로 추가되는 PK 값을 확인해야 하는 상황에서는 유용하게 사용될 수 있다.

MyBatis 는

Mapper 인터페이스의 리턴 타입에 맞게 select 의 결과를 처리하기 때문에

tbl\_board 테이블 내 모든 컬럼은 BoardVO의

'bno, title, content, writer, regdate, updateDate' 속성값으로 처리된다.

MyBatis 는 bno라는 컬럼이 존재하면

BoardVO 인스턴스의 setBno()를 호출하여 해당 데이터를 세팅한다.

MyBatis의 모든 파라미터와 리턴 타입의 처리는

get 파라미터명(), set컬럼명() 의 규칙으로 호출한다.

## 삭제

```
/springboard/src/test/java/com/yks/mapper/BoardMapper.xml

<select id="read" resultType="com.yks.domain.BoardVO">
    SELECT * FROM tbl_board WHERE bno = #{bno}
</select>

<delete id="delete">
    DELETE FROM tbl_board WHERE bno = #{bno}
</delete>
```

```
/springboard/src/main/java/com/yks/mapper/BoardMapper.java

public BoardVO read(Long bno);

public int delete(Long bno);
```

## 테스트코드(BoardMapperTests.java)

아래 내용 추가

```
/springboard/src/test/java/com/yks/mapper/BoardMapperTests.java

@Test
public void testRead() {
    // 존재하는 게시물 번호로 테스트
    BoardVO board = mapper.read(5L);

    log.info(board);
}

@Test
public void testDelete() {
    log.info("DELETE COUNT : " + mapper.delete(3L));
}
```

## 업데이트

게시물의 업데이트는 제목, 내용, 작성자를 수정한다고 가정.

업데이트 할 때에는

최종 수정시간을 데이터베이스 내의 현재 시간으로 수정한다.

update는 delete와 마찬가지로

'몇 개의 데이터가 수정되었는가'를 반환받을 수 있도록

int 타입으로 메서드의 리턴 값을 설계

```
/springboard/src/main/java/com/yks/mapper/BoardMapper.java
```

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

```
/springboard/src/test/java/com/yks/mapper/BoardMapper.xml
```

```
<update id="update">  
    UPDATE tbl_board  
    SET title = #{title}, content = #{content}, writer = #{writer}, updateDate =  
sysdate  
    WHERE bno = #{bno}  
</update>
```