

mybatis-Spring

스프링 프레임워크에서 데이터베이스 연동에 필요한 JDBC, iBatis, Hibernate 등의 라이브러리를 제공한다. 특히 스프링 프레임워크 3.0부터는 아이바티스(iBatis)의 후속 버전으로 강력한 마이바티스(MyBatis)와 마이바티스-스프링 연동 모듈을 적용하여 데이터에 접근과 처리를 하고 있다.

(마이바티스는 객체지향 언어인 자바의 관계형 데이터베이스 프로그래밍을 좀더 쉽게 할 수 있도록 도와주는 개발 프레임워크다.)

1. 마이바티스와 마이바티스-스프링

마이바티스(MyBatis)란 XML구문과 어노테이션을 사용한 SQL문이나 저장된 프로시저를 데이터베이스와 자바 등을 연결시켜 주는 역할을 하는 영속성 프레임워크이다. 영속성 프레임워크란 정보에 대한 접근과 저장을 단순화하는 라이브러리를 말한다. JDBC 코드와 수동으로 설정하는 파라미터와 결과 매핑을 없애 주고, 데이터베이스에 원시타입(int, String, Date)과 맵 인터페이스, 자바 객체를 설정하고 매핑하기 위해 XML과 어노테이션을 사용할 수 있다.

구분	아이바티스(iBatis)	마이바티스(MyBatis)
스프링 버전	2.x부터 지원	3.x부터 지원
네임스페이스	선택 사용	필수 사용
매핑 구문	XML	XML과 어노테이션
동적 SQL 요소	16개의 XML 엘리먼트	4개의 XML 엘리먼트
스프링 모듈	자체 모듈	별도 모듈
용어 사용	SqlMapConfig / sqlMap	Configuration / sqlMap

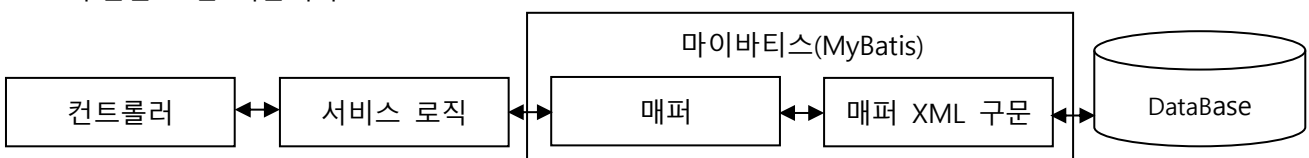
마이바티스(MyBatis)와 스프링 3.x에 통합 지원하는 마이바티스-스프링 라이브러리가 생기게 되었고, 스프링에서 마이바티스-스프링 라이브러리를 사용하면, 마이바티스 객체를 쉽게 생성하고 주입할 수 있으며, 스프링 트랜잭션을 통해 처리할 수 있다. 마이바티스-스프링 연동모듈을 사용하려면 자바 1.5이상의 버전이어야 하고, 마이바티스와 스프링은 각각 버전별로 조금씩 다르다.

마이바티스-스프링	마이바티스(MyBatis)	스프링 프레임워크
1.0.0 그리고 1.0.1	3.0.1 에서 3.0.5까지	3.0.0 또는 그 이상
1.0.2	3.0.6	3.0.0 또는 그 이상
1.1.0 또는 그 이상	3.1.0 또는 그 이상	3.0.0 또는 그 이상

스프링 3.0 이상에서 마이바티스(MyBatis) 3.0 이상의 라이브러리와 마이바티스-스프링 1.0 이상 라이브러리 파일이 필요하다. 마이바티스 설치는 "http://blog.mybatis.org/" 사이트에서 다운로드 받아 "WEB-INF/lib" 폴더에 복사하면 된다. MyBatis 3.2.4.zip과 MyBatis-spring-1.2.2.zip 파일을 다운로드하여 사용한다.

- MyBatis 3.2.4.zip : 마이바티스 라이브러리
- MyBatis-spring-1.2.2.zip : 마이바티스에서 스프링을 사용할 수 있는 라이브러리

스프링과 마이바티스-스프링을 연동한 데이터에 대한 흐름은 컨트롤러 → 서비스 로직 → 매퍼 → 매퍼 XML 구문 → DataBase



- 매퍼 XML은 매퍼 XML 요소와 SQL 문으로 작성되는 매퍼 XML 구문이다.
- 매퍼는 인터페이스로 선언하고, 구현 클래스에서 SQL문을 실행한다.
- 서비스 로직은 자바빈으로 구성하고, 데이터베이스를 검색하거나 관리한다.
- 컨트롤러는 스프링 구성요소로 서비스 로직으로부터 데이터를 전달받는다.
- 스프링 설정 파일에서 데이터베이스 서버에 관한 데이터소스(DataSource), 매퍼위치, 트랜잭션, SqlSession, 매퍼 주입에 관하여 설정한다.

2. 매퍼 XML

매퍼 XML은 SQL문을 매퍼 XML 파일로 작성하는 가장 중요한 부분이다. 데이터베이스를 다루는 SQL문은 마이바티스가 제공하는 기능의 XML이나 어노테이션을 통한 매퍼 기법으로 작성된다. 마이바티스의 매퍼 XML 요소이다.

매퍼 XML 요소	설명
cache	해당 네임스페이스를 위한 캐시 설정
cache-ref	다른 네임스페이스를 캐시 설정에 대한 참조
resultMap	데이터베이스 결과 데이터를 객체에 로드하는 방법 정의.
parameterMap	파라미터를 매핑하기 위해서 사용.
insert	매핑된 INSERT문
update	매핑된 UPDATE문
delete	매핑된 DELETE문
select	매핑된 SELECT문

SQL문에서 SELECT문은 select 요소, INSERT문은 insert 요소, UPDATE문은 update 요소, DELETE문은 delete 요소로 매핑 XML 구문을 만든다. select, insert, update, delete 요소에 사용할 수 있는 공통 속성이다.

속성	설명
id	네임스페이스(namespace)의 유일한 구분자
parameterType	구문에 전달될 파라미터의 전체 클래스명이나 별명
parameterMap	외부 parameterMap을 찾기 위한 접근 방법.
flushCache	구문을 호출할 때마다 캐시를 지울지 여부를 설정한다. 디폴트는 false 이다.
timeout	데이터베이스의 요청 결과를 기다리는 최대시간을 설정
statementType	Statement, Prepared, Callable 중 선택. 디폴트는 Prepared이다.

(1) select

select 요소는 데이터를 검색하는 SELECT문의 매퍼 구문을 작성한다. 테이블에서 전체 행을 검색하는 SELECT문의 매퍼 구문 형식은 다음과 같다.

```
<select id="구분자" resultType="반환타입">
    select 컬럼명1, 컬럼명2 ... from 테이블명
</select>
```

select 요소는 공통 속성과 속성들로 매퍼 구문을 작성한다.

속성	설명
resultType	반환되는 타입의 전체 클래스명이나 별명.
resultMap	외부 resultMap 참조명

useCache	구문의 결과에 캐시 사용 여부 지정. 디폴트는 true이다.
fetchSize	지정된 수만큼의 결과를 반환하는 값.

```
<select id="listDepartment" resultType="vo.DeptVO">
    Select * from Department
</select>
```

(2) insert, update, delete

INSERT문, UPDATE문, DELETE문은 insert, update, delete 매퍼 요소와 공통 속성과 속성을 사용하여 매퍼 구문을 작성한다.

- insert 요소는 테이블에 행을 추가하는 INSERT문의 매퍼 구문을 작성한다.
- update 요소는 테이블의 컬럼 값을 수정하는 UPDATE문의 매퍼 구문을 작성한다.
- delete 요소는 테이블의 행을 삭제하는 DELETE의 매퍼 구문을 작성한다.

속성	설명
useGeneratedKeys	생성 키 사용 여부를 결정한다. 오라클은 sequence를 생성키로 제공한다. 디폴트 값은 false이다.
keyProperty	getGeneratedKeys 메서드나 insert 구문의 selectKey 하위 요소에 의해 리턴된 키를 셋팅할 프로퍼티를 지정. 디폴트는 셋팅하지 않는 것이다.
keyColumn	생성키를 가진 테이블의 컬럼명을 셋팅. 키컬럼이 테이블에 첫번째 컬럼이 아닌 데이터베이스에서만 필요하다.

생성키는 Oracle의 Sequence, MySql의 auto_increment 처럼 자동 증가 필드를 말한다.

(3) 파라미터의 위치 지정자(?) 표기

JSP 프로그래밍에서 "SELECT * FROM BOARD WHERE NUM=?"문의 위치 지정자(?)가 있는 SELECT문은 PreparedStatement 객체로 쿼리문을 생성하고 위치 지정자(?)의 값은 "setXXX(위치인덱스, 값)"으로 코딩되었다.

```
String sql = "SELECT * FROM BOARD WHERE NUM=?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, num);
...
```

파라미터는 마이바티스에서 매우 중요한 요소로, SQL문의 파라미터 값을 받기 위해 "#{}" 기호로 표기하며, {}속에 "값"이나 자바빈의 "프로퍼티명"을 기술한다.

표기법 : #{값 또는 프로퍼티명}

① SELECT문

특정 행을 검색하는 SELECT문의 select 요소의 구문 형식은 다음과 같다.

```
<select id="구분자" parameterType="파라미터타입" resultType="반환타입">
    select 컬럼명1, 컬럼명2 ... from 테이블명 where 컬럼명 = #{값}
</select>
```

② INSERT문

한 행을 추가하는 INSERT문의 insert 요소의 구문 형식은 다음과 같다.

```
<insert id="구분자" parameterType="파라미터타입">
    insert into 테이블명(컬럼명1, 컬럼명2...) values(#{값1}, #{값2} ...)
</insert>
```

③ UPDATE문

특정 또는 일부 행의 컬럼 값을 수정하는 UPDATE문의 update 요소의 구문 형식은 다음과 같다.

```
<update id="구분자" parameterType="파라미터타입">
    update 테이블명 set 컬럼명1 = #{값1}, 컬럼명2 = #{값2} where 컬럼명 = #{값3}
</update>
```

④ DELETE문

특정 행을 삭제하는 DELETE문의 delete 요소의 구문 형식은 다음과 같다.

```
<delete id="구분자" parameterType="파라미터타입">
    delete from 테이블명 where 컬럼명 = #{값1}
</delete>
```

- Department 테이블의 학과코드(#{dept_id})로 특정 행을 검색하는 매퍼 XML 구문을 작성한다.

```
<select id="selectDepartment" parameterType="vo.DeptVO" resultType="vo.DeptVO">
    select *
    from Department
    where Dept_ID = #{dept_id}
</select>
```

- Department 테이블에서 한 행을 추가하는 매퍼 XML 구문을 작성한다.

```
<insert id="insertDepartment" parameterType="vo.DeptVO" >
    insert into Department
    (Dept_ID, Dept_Name, Dept_tel)
    values (#{dept_id}, #{dept_name}, #{dept_tel})
</select>
```

- Department 테이블에서 학과코드(#{dept_id})로 특정 행의 학과명과 전화번호를 수정하는 매퍼 XML 구문을 작성한다.

```
<update id="updateDepartment" parameterType="vo.DeptVO" >
    update Department
    set Dept_Name = #{dept_name}, Dept_tel = #{dept_tel}
    where Dept_ID = #{dept_id}
</select>
```

- Department 테이블에서 학과코드("#{dept_id}")로 특정 행을 삭제하는 매퍼 XML 구문을 작성한다.

```
<delete id="deleteDepartment" parameterType="vo.DeptVO">
    delete from Department where Dept_ID = #{dept_id}
</delete>
```

⑤ 데이터베이스가 자동 생성키를 생성한 경우

insert문에서 자동 생성키를 사용하는 기능이 있다. insert는 키(key) 생성과 같은 기능을 추가하기 위한 속성과 하위 요소가 있다.

useGeneratedKeys = "true"로 설정하고 자동 생성키를 적용할 키 컬럼을 keyProperty에 지정한다. INSERT 문에는 키 컬럼은 기술하지 않는다.

```
<insert id="구분자" useGeneratedKeys="true" keyProperty="키컬럼">
    insert into 테이블명(컬럼명1, 컬럼명2...) values("#{값1}", "#{값2}" ...)
</insert>
```

⑥ 데이터베이스가 자동 생성키를 생성하지 않은 경우

<insert>요소 내에 <selectKey>요소로 키 생성에 필요한 SELECT문을 기술한다. <selectKey>의 속성이다.

속성	설명
keyProperty	selectKey 구문의 결과가 할당될 대상 프로퍼티.
resultType	결과 타입
order	before는 insert문 전 실행. after는 insert문 후 실행.
statementType	Statement, Prepared, Callable 중 선택. 디폴트는 Prepared이다.

order 속성이 before는 먼저 키 값을 생성한 후 INSERT문이 실행되고, after는 INSERT문이 실행된 후에 <selectKey>가 실행된다.

```
<insert id="구분자">
    <selectKey keyProperty="id" resultType="int" order="BEFORE" statementType="PREPARED">
        SELECT문(키 생성 관련)
    </selectKey>
    insert into 테이블명(컬럼명1, 컬럼명2...) values("#{값1}", "#{값2}" ...)
</insert>
```

예제) board 테이블에 게시물이 추가될 때 게시물번호(b_id)를 현재 게시물 번호의 최대값을 구하여 1을 증가하고, null일 때 1로 지정하여 저장하는 XML 매퍼 구문을 작성하여 보자.

```
<insert id="insertBoard" parameterType="com.vo.BoardVO">
    <selectKey keyProperty="bid" resultType="int" order="BEFORE" >
        SELECT CASE WHEN MAX(b_id) IS NULL THEN 1
            ELSE MAX(b_id)+1 END FROM board
    </selectKey>
    INSERT INTO board(b_id, b_name, ...) values("#{bid}", "#{bname}", ...)
</insert>
```

(4) resultMap

모든 컬럼 값의 결과가 HashMap에서 키 형태로 자동 매핑 된다. resultMap의 id, result, constructor, association, collection, discriminator 요소가 있다.

속성	설명
id	기본 키에 해당하는 값을 설정한다.
result	기본 키가 아닌 나머지 컬럼에 대해 매핑한다.
constructor	생성자를 통해 값을 설정할 때 사용한다.
association	1:1 관계를 처리한다.
collection	1:N 관계를 처리한다.
discriminator	매핑 과정에서 조건을 지정해서 값을 설정할 때 사용한다.

id나 result 요소는 결과 매핑의 가장 기본적인 형태로 한 개의 컬럼을 한 개의 프로퍼티나 간단한 데이터 타입의 필드에 매핑한다. id와 result에 사용할 수 있는 요소들이다. 테이블 컬럼명과 프로퍼티명이 다른 경우 사용할 수 있다.

속성	설명
property	결과 컬럼에 매핑되는 필드 또는 자바빈과 동일한 프로퍼티
column	데이터베이스 컬럼명이나 컬럼의 별명
javaType	클래스명이나 타입 별명.
jdbcType	jdbc 타입을 명시.

(5) 컬럼에 별명을 사용

테이블의 컬럼명과 자바빈의 프로퍼티명이 다를 때 SELECT문의 컬럼명에 별명을 사용하여 일치시킨다.

```
<select id="구분자" resultType="반환타입">
    select 컬럼명1 as 별칭, 컬럼명2 ... from 테이블명 where 컬럼명 = #{값}
</select>
```

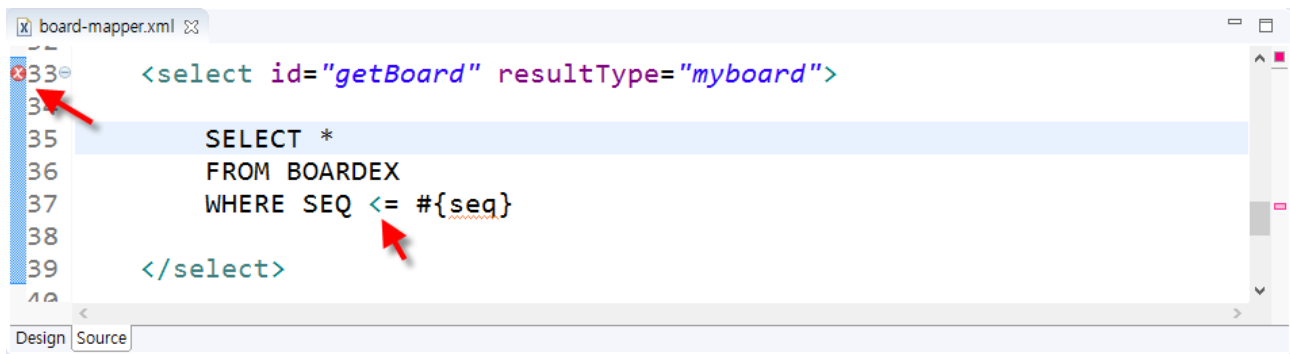
(6) resultMap의 id와 result 요소

id에 "id명"을 기술하고 <select> 요소의 resultMap에 사용한다. result 요소에 property와 column 속성을 기술한다. 테이블의 컬럼명과 자바빈의 프로퍼티명이 다를 경우 column에 테이블의 컬럼명, property에 자바빈의 프로퍼티명을 기술하여 일치시킨다.

```
<resultMap id="id명" type="타입명">
    <result property="프로퍼티명" column="컬럼명" />
</resultMap>
<select id="구분자" resultType="id명">
    select 컬럼명1, 컬럼명2 ... from 테이블명
</select>
```

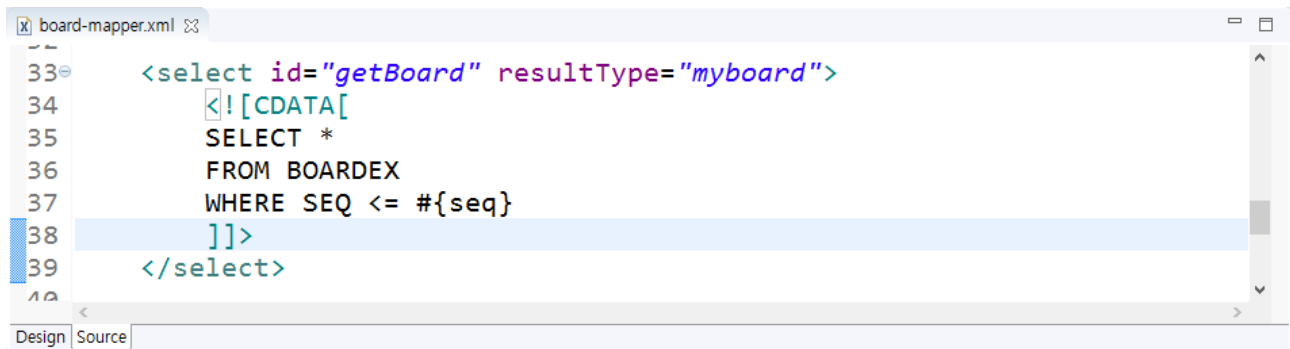
(7) CDATA Section 사용

SQL 구문 내에 '<'기호를 사용하면 다음처럼 에러가 발생한다.



```
board-mapper.xml
33 <select id="getBoard" resultType="myboard">
34
35     SELECT *
36     FROM BOARDX
37     WHERE SEQ <= #{seq}
38
39 </select>
```

다음처럼 CDATA Section으로 SQL 구문을 감싸주면 에러가 사라진다.



```
board-mapper.xml
33 <select id="getBoard" resultType="myboard">
34     <![CDATA[
35     SELECT *
36     FROM BOARDX
37     WHERE SEQ <= #{seq}
38     ]]>
39 </select>
```

추후 문제가 발생하지 않도록 하기 위해서 모든 SQL 구문을 CDATA Section으로 감싼다.

[실습 예제]

EX 1. MyBatis 프레임워크

MyBatis는 원래 Apache에서 ibatis라는 이름의 프레임워크였으나 2010년 ibatis가 Apache에서 탈퇴하여 Google에 흡수되면서 MyBatis로 변경되었다.

EX 1.1 MyBatis 프레임워크의 특징

첫 번째는 한두줄의 자바 코드로 DB연동을 처리한다는 것이며, 두 번째는 SQL 명령어를 자바 코드에서 분리하여 XML 파일에 따로 관리한다는 것이다.

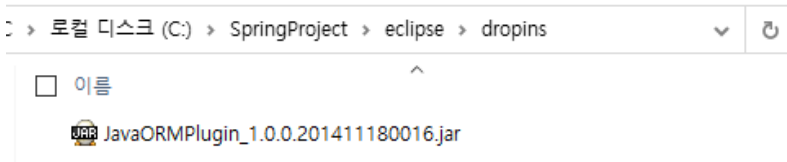
EX 1.2 Java ORM Plugin 수동설치

<https://sourceforge.net/projects/java-orm-plugin/> 에서 다운받는다.

Project Activity

Released [/site/plugins/JavaORMPlugin_1.0.0.201411180016.jar](#) 다운 로드

이클립스가 설치된 폴더로 가서 eclipse -> dropins 경로에 붙여넣기 한다.

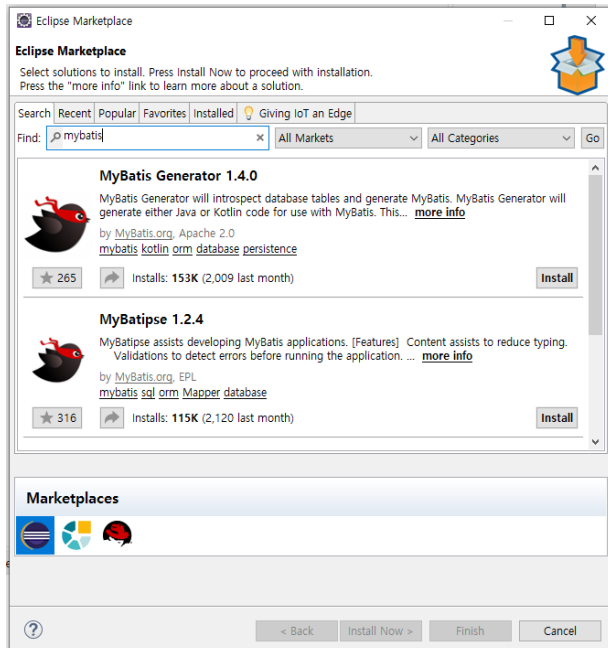


이클립스 재실행 한다.

MyBatis Plugin 설치

MyBatis와 관련된 복잡한 XML 설정 파일들을 자동으로 만들고 관리할 수 있다.

mybatis로 검색해서 MyBatis Generator 1.4.0와 MyBatipse 1.2.4을 설치 한다.



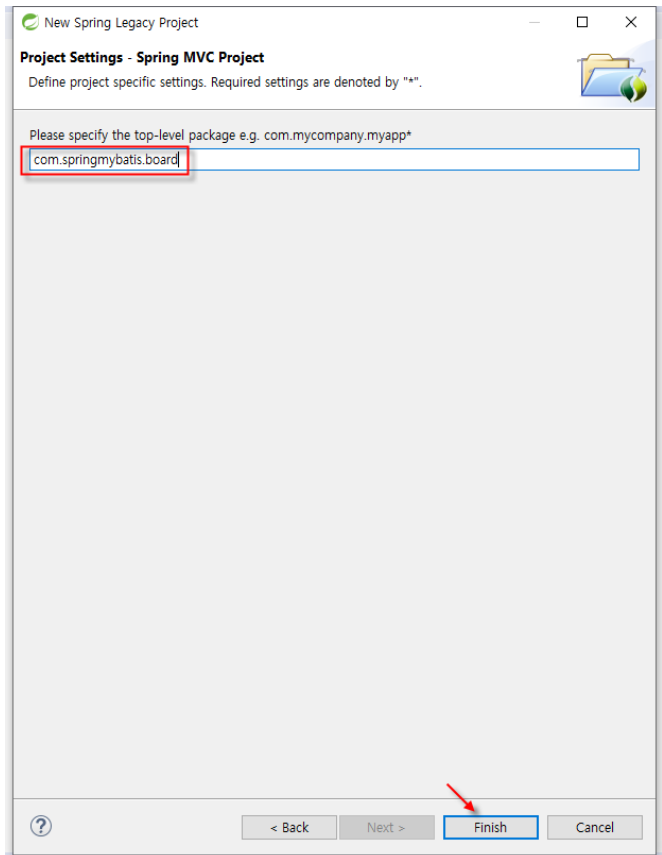
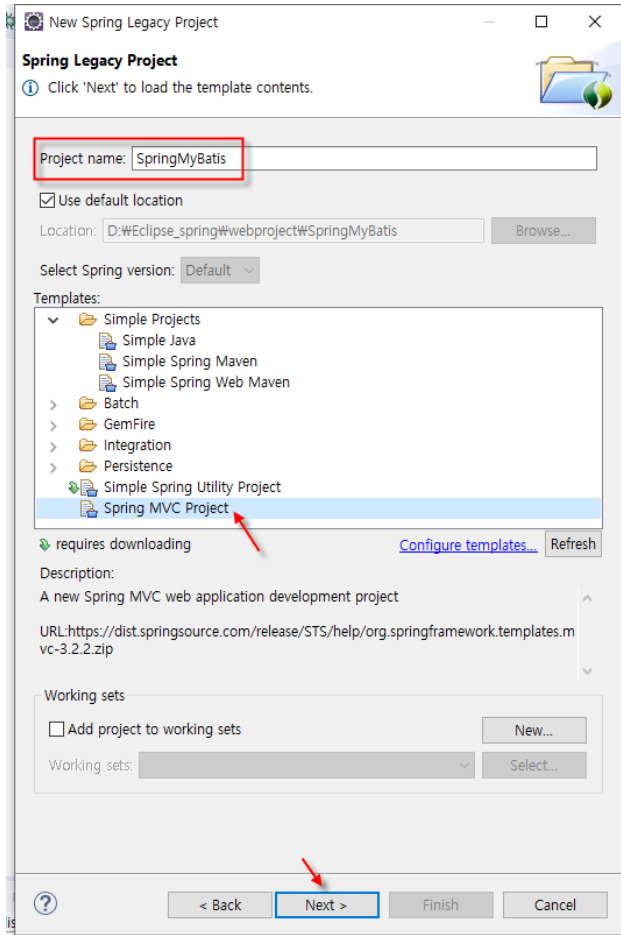
EX 1.3 프로젝트 생성

MyBatis를 스프링과 연동하지 않고 단독으로 사용하는 것은 여러가지로 불편한 점이 많다. 하지만 MyBatis의 구조와 기능 이해에 집중하기 위해 간단한 프로젝트를 MyBatis만으로 수행하도록 한다.

File -> New -> Spring Legacy

Project name : springMyBatis

com.springmybatis.board



springMyBatis 프로젝트 자바 버전과 스프링 프레임워크 버전 설정

DB 연동을 위한 Oracle Driver와 MyBatis, iBatis 라이브러리들을 내려 받고자 pom.xml 파일에 추가한다.
pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/maven-
v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.springmybatis</groupId>
    <artifactId>board</artifactId>
    <name>SpringMyBatis</name>
    <packaging>war</packaging>
    <version>1.0.0-BUILD-SNAPSHOT</version>
    <properties>
```

```

<java-version>1.8</java-version>
<org.springframework-version>5.0.7.RELEASE</org.springframework-version>
<org.aspectj-version>1.6.10</org.aspectj-version>
<org.slf4j-version>1.6.6</org.slf4j-version>

</properties>
<dependencies>

    <dependency>

        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>5.0.7.RELEASE</version>

    </dependency>

    <!-- MyBatis -->
    <!-- https://mvnrepository.com/artifact/org.mybatis -->
    <dependency>

        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.6</version>

    </dependency>

    <dependency>

        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>2.0.6</version>

    </dependency>

    <!-- Spring -->
    <dependency>

```

생략

<https://mvnrepository.com/artifact/org.mybatis>

Home » org.mybatis » mybatis-spring » 2.0.2

MyBatis Spring » 2.0.2
An easy-to-use Spring bridge for MyBatis sql mapping framework.

License: [Apache 2.0](#)

HomePage: <http://www.mybatis.org/spring/>

Date: (Jul 15, 2019)

Files: [jar \(66 KB\)](#) [View All](#)

Repositories: [Central](#)

Used By: 334 artifacts

Maven [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.2</version>
</dependency>

```

☒ Include comment with link to declaration

Home » org.mybatis » mybatis » 3.5.2

MyBatis » 3.5.2
The MyBatis SQL mapper framework makes it easier to use a relational database with object relational mapping tools.

License: [Apache 2.0](#)

Categories: [Object/Relational Mapping](#)

HomePage: <http://www.mybatis.org/mybatis-3/>

Date: (Jul 15, 2019)

Files: [jar \(1.6 MB\)](#) [View All](#)

Repositories: [Central](#)

Used By: 656 artifacts

Maven [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

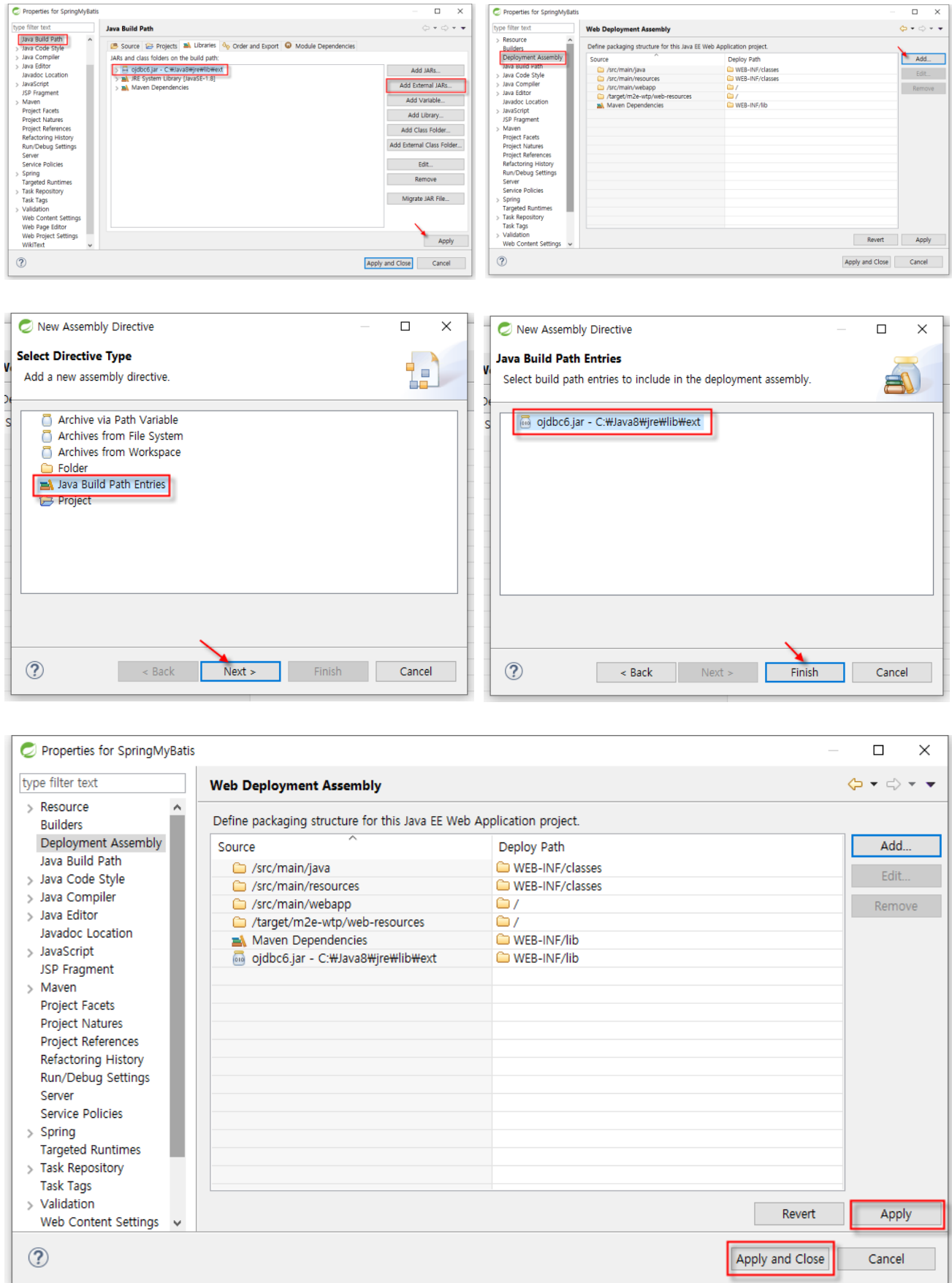
```

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.2</version>
</dependency>

```

☒ Include comment with link to declaration

오라클 드라이버 등록



EX 1.4 VO(Value Object) 클래스

XML 파일에 저장된 SQL 명령어에 사용자가 입력한 값들을 전달하고 실행 결과를 매핑할 VO 클래스를 작성한다.

BoardVO.java

```
package com.springmybatis.board.vo;

import java.util.Date;

public class BoardVO {
    private int seq;
    private String title;
    private String writer;
    private String content;
    private Date regDate;
    private int cnt;
    private String searchCondition;
    private String searchKeyword;
    public int getSeq() {
        return seq;
    }
    public void setSeq(int seq) {
        this.seq = seq;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getWriter() {
        return writer;
    }
    public void setWriter(String writer) {
        this.writer = writer;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Date getRegDate() {
```

```

        return regDate;
    }
    public void setRegDate(Date regDate) {
        this.regDate = regDate;
    }
    public int getCnt() {
        return cnt;
    }
    public void setCnt(int cnt) {
        this.cnt = cnt;
    }
    public String getSearchCondition() {
        return searchCondition;
    }
    public void setSearchCondition(String searchCondition) {
        this.searchCondition = searchCondition;
    }
    public String getSearchKeyword() {
        return searchKeyword;
    }
    public void setSearchKeyword(String searchKeyword) {
        this.searchKeyword = searchKeyword;
    }
    @Override
    public String toString() {
        return "BoardVO [seq=" + seq + ", title=" + title + ", writer=" + writer + ", content="
+ content + ", regDate=" + regDate + ", cnt=" + cnt + "]\n";
    }
}

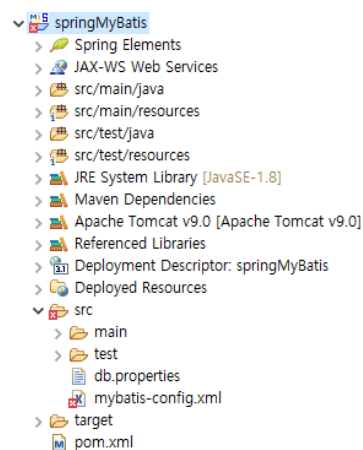
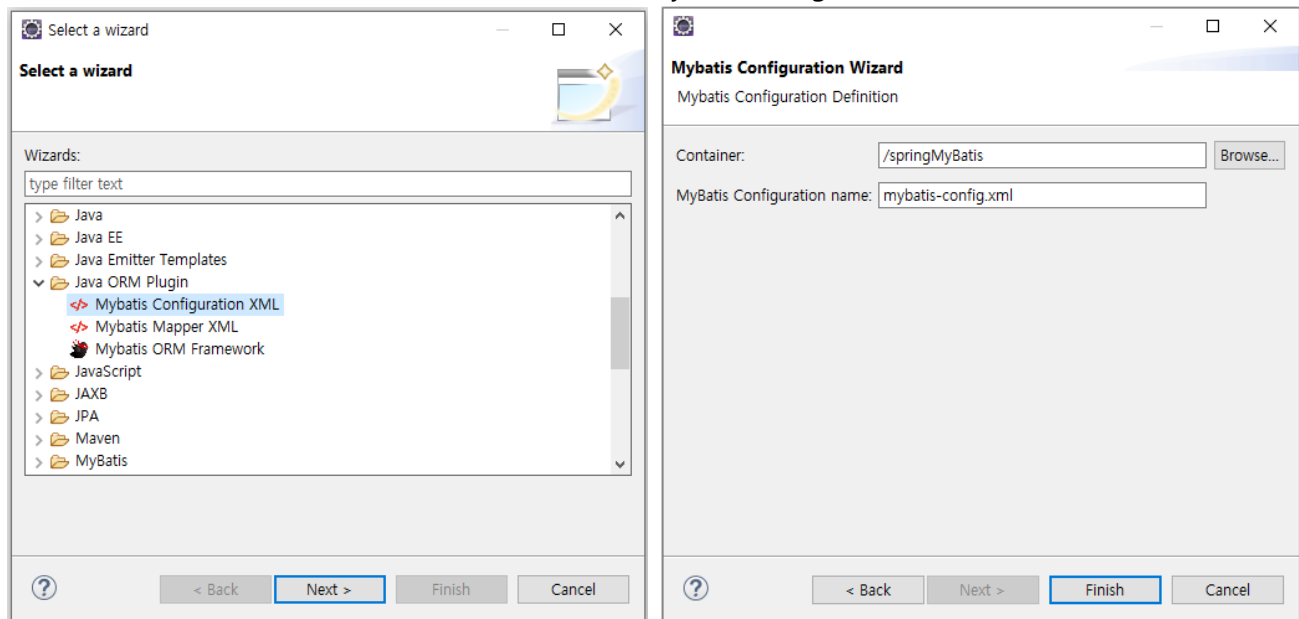
```

EX 1.5 MyBatis 환경 설정

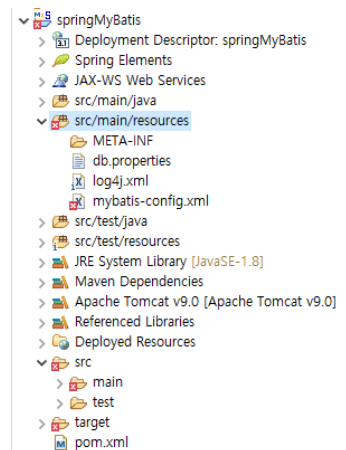
MyBatis 환경 설정 파일도 Java ORM 플러그인을 사용하여 생성한다.

MyBatisExam 프로젝트를 마우스 오른쪽 클릭 New -> Other

mybatis-config.xml



파일을 src/main/resources 폴더로 이동한다.➔



db.properties 기본 설정

```
jdbc.driverClassName=com.mysql.jdbc.Driver  
jdbc.url=jdbc:mysql://localhost:3306/#dbname  
jdbc.username=#user  
jdbc.password=#password
```

db.properties 수정

```
jdbc.driverClassName=oracle.jdbc.OracleDriver  
jdbc.url=jdbc:oracle:thin:@localhost:1521:XE  
jdbc.username=hr  
jdbc.password=hr
```

mybatis-config.xml 기본 설정

```
<?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>
  <properties resource="db.properties" />
  <typeAliases>
    <typeAlias type="#package.#modelName" alias="#modelName"></typeAlias>
  </typeAliases>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC" />
      <dataSource type="POOLED">
        <property name="driver" value="${jdbc.driverClassName}" />
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="#package/#mapper.xml" />
  </mappers>
</configuration>
```

mybatis-config.xml 수정

```
<?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>
  <!-- Properties 설정 -->
  <properties resource="db.properties" />

  <!-- Alias 설정 -->
  <typeAliases>
    <typeAlias type="com.springmybatis.board.vo.BoardVO" alias="board"></typeAlias>
  </typeAliases>

  <!-- DataSource 설정 -->
  <environments default="development">
```

```
<environment id="development">
    <transactionManager type="JDBC" />
    <dataSource type="POOLED">
        <property name="driver" value="${jdbc.driverClassName}" />
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
    </dataSource>
</environment>
</environments>

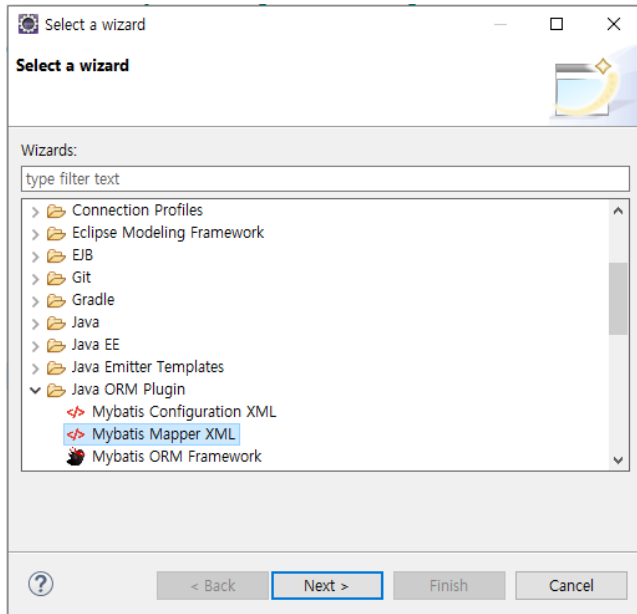
<!-- SQL Mapper 설정 -->
<mappers>
    <mapper resource="mappings/board-mapper.xml" />
</mappers>
</configuration>
```


EX 1.6 SQL Mapper XML 파일 작성

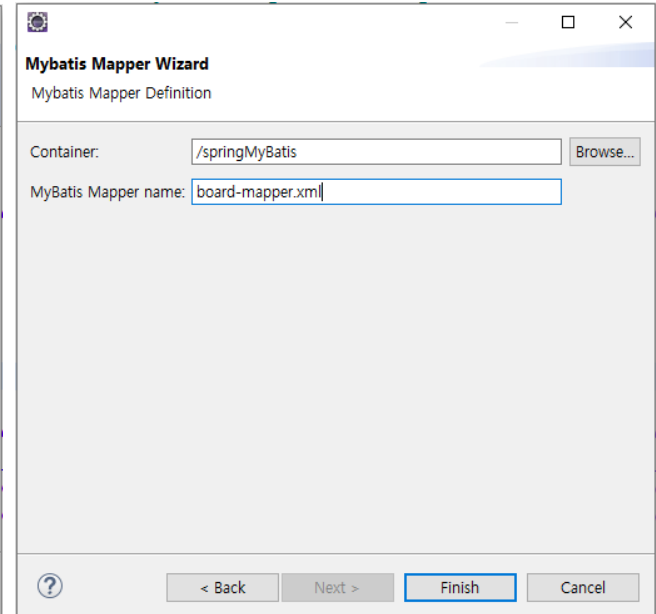
SQL Mapper XML 파일은 MyBatis에서 가장 중요한 파일이다. 이 XML 파일에 DB 연동에 필요한 SQL 명령어들이 저장되기 때문이다. 이 XML 파일을 Java ORM 플러그인을 이용하여 생성한다.

MyBatisExam 프로젝트명을 마우스 오른쪽 클릭 New -> Other

Mybatis Mapper XML 선택



board-mapper.xml



board-mapper.xml 기본 설정

```
<?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="#package.#mapperinterfacename">
    <select id="getSomething" parameterType="int" resultType="#package.#modelName">
        SELECT
        columnname1,
        columnname2,
        columnname3
        FROM tablename1
```

```

        WHERE columnname1 = #{value}
    </select>
    <resultMap type="#modelName" id="YourResultSet">
        <id property="param1" column="columnname1" />
        <result property="param2" column="columnname2" />
        <result property="param3" column="columnname3" />
    </resultMap>
    <select id="getAll" resultMap="YourResultSet">
        SELECT * FROM tablename1
    </select>
    <insert id="insertSomething" parameterType="#modelName">
        INSERT INTO tablename1 (columnname1, columnname2, columnname3)
        VALUES(#{value1},#{value2},#{value3})
    </insert>
    <update id="updateSomething" parameterType="#modelName">
        UPDATE tablename1
        SET
        columnname1=#{param1}
        WHERE
        columnname2 = #{param2}
    </update>
    <delete id="deleteSomething" parameterType="int">
        DELETE FROM tablename1 WHERE
        columnname1 = #{param1}
    </delete>
</mapper>

```

board-mapper.xml 수정

```

<?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="BoardDAO">
    <insert id="insertBoard">
    <![CDATA[
        INSERT INTO boardex (seq, title, writer, content)
        VALUES((select nvl(max(seq), 0)+1,from boardex), #{title}, #{writer}, #{content})
    ]]>
    </insert>

    <update id="updateBoard">
    <![CDATA[

```

```

        UPDATE boardex
        SET
            title =#{title},
            content =#{content}
        WHERE
            seq = #{seq}
    ]]>
</update>

<delete id="deleteBoard">
    <![CDATA[
        DELETE FROM boardex
        WHERE
            seq = #{seq}
    ]]>
</delete>

<select id="getBoard" resultType="board">
    <![CDATA[
        SELECT *
        FROM
            boardex
    WHERE
        seq = #{seq}
    ]]>
</select>

<select id="getBoardList" resultType="board">
    <![CDATA[
        SELECT *
        FROM
            boardex
    WHERE
        title like '%||#{searchKeyword}||'%'
    ORDER BY
        seq desc
    ]]>
</select>
</mapper>

```

프로젝트명 오른쪽 버튼 클릭 -> Maven -> Update Projects... 한다.

EX 1.7 SqlSession 객체 생성

MyBatis를 이용하여 DAO를 구하려면 SqlSession 객체가 필요하다. SqlSession 객체를 얻으려면 SqlSessionFactory 객체가 필요하다. 따라서 DAO 클래스를 구현하기에 앞서 SqlSessionFactory 객체를 생성하는 유틸리티 클래스를 작성한다.

SqlSessionFactoryBean.java

```
package com.springmybatis.util;

import java.io.Reader;
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class SqlSessionFactoryBean {
    private static SqlSessionFactory sessionFactory = null;
    static {
        try {
            if (sessionFactory == null) {
                Reader reader = Resources.getResourceAsReader("mybatis-config.xml");
                sessionFactory = new SqlSessionFactoryBuilder().build(reader);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static SqlSession getSqlSessionInstance() {
        return sessionFactory.openSession();
    }
}
```

'mybatis-config.xml' 파일로부터 설정 정보를 읽어 들이기 위한 입력 스트림을 생성한다. 그리고 나서 입력 스트림을 통해 'mybatis-config.xml' 파일을 읽어 SqlSessionFactory 객체를 생성한다.

getSqlSessionInstance() 메소드는 SqlSessionFactory 객체로부터 SqlSession 객체를 얻어내어 리턴한다. 이 메소드를 이용하면 SqlSession 객체가 필요한 DAO 클래스를 구현하면 된다.

EX 1.8 DAO 클래스

데이터베이스와 연동을 처리한다.

BoardDAO.java

```
package com.springmybatis.board.dao;

import java.util.List;
import org.apache.ibatis.session.SqlSession;
import com.springmybatis.board.vo.BoardVO;
import com.springmybatis.util.SqlSessionFactoryBean;

public class BoardDAO {
    private SqlSession mybatis;

    public BoardDAO() {
        mybatis = SqlSessionFactoryBean.getSqlSessionInstance();
    }

    public void insertBoard(BoardVO vo) {
        // board-mapper.xml 파일의 namespace명.id명
        mybatis.insert("BoardDAO.insertBoard", vo);
        mybatis.commit();
    }

    public void updateBoard(BoardVO vo) {
        mybatis.update("BoardDAO.updateBoard", vo);
        mybatis.commit();
    }

    public void deleteBoard(BoardVO vo) {
        mybatis.delete("BoardDAO.deleteBoard", vo);
        mybatis.commit();
    }

    public BoardVO getBoard(BoardVO vo) {
        return (BoardVO) mybatis.selectOne("BoardDAO.getBoard", vo);
    }

    public List<BoardVO> getBoardList(BoardVO vo) {
        // board-mapper.xml 파일의 namespace명.id명
        return mybatis.selectList("BoardDAO.getBoardList", vo);
    }
}
```

BoardDAO 클래스 생성자에서 SqlSessionFactroyBean을 이용하여 SqlSession 객체를 얻는다. 그리고 이 SqlSession 객체의 메소드를 이용해서 CRUD 기능의 메소드를 구현한다.

각 메소드는 첫 번째 인자가 실행될 SQL의 id 정보이다. 이 때 SQL Mapper에 선언된 네임스페이스와 아이디를 조합하여 아이디를 지정해야 한다. 두 번째 인자는 parameterType 속성으로 지정된 파라미터 객체이다.

EX 1.9 테스트 클라이언트 작성 및 실행

scr/test/java 폴더에 패키지 com.springmybatis.board에 생성한다.

BoardServiceClient.java

```
package com.springmybatis.board;

import java.util.List;
import org.junit.Test;
import com.springmybatis.board.dao.BoardDAO;
import com.springmybatis.board.vo.BoardVO;

public class BoardServiceClient {

    @Test
    public void boardTest() {

        BoardDAO boardDAO = new BoardDAO();
        BoardVO vo = new BoardVO();
        vo.setSearchCondition("TITLE");
        vo.setSearchKeyword("");
        List<BoardVO> boardList = boardDAO.getBoardList(vo);
        for (BoardVO board : boardList) {
            System.out.println("---> " + board.toString());
        }
    }
}
```

실행 결과

