Spring mybatis

강사 강태광

I. spring ORM

1. OR-Mapping의개요

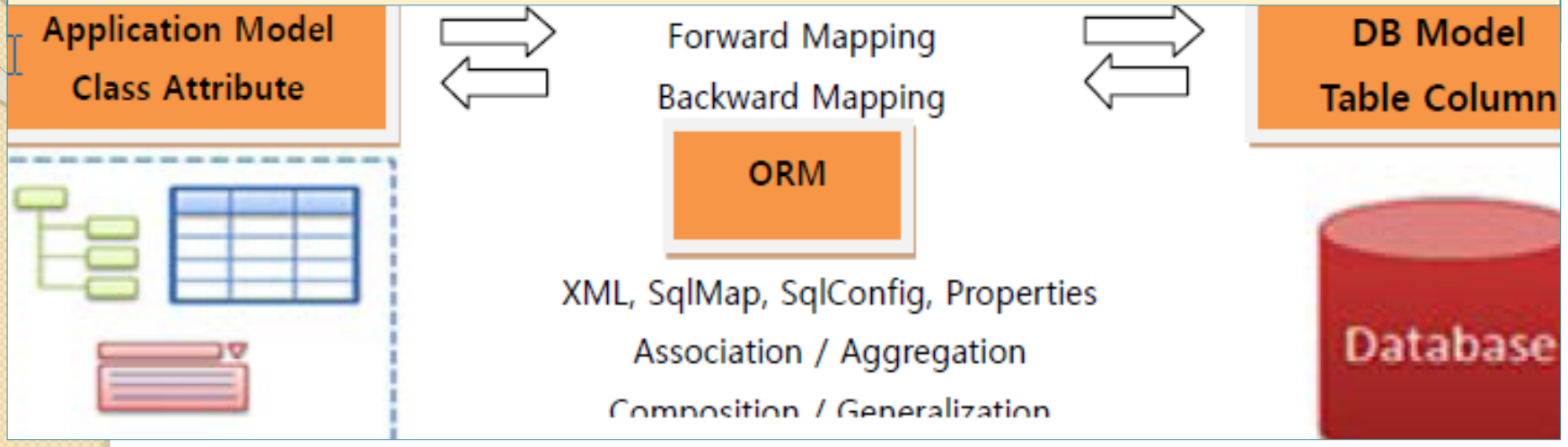
가. OR-Mapping의 정의
- **OOP 프로그래밍시 설계할 클래스들과 데이터저장소로 이용될 RDBMS의 Table 간의 Mapping**
- 데이터베이스 연계처리를 위하여 기존 SQL에 의존하는 것이 아니라 직접 테이블의 컬럼을 Java Class에 매핑하거나 XML형태의 SQL을 실행하여 처리를 수행하는 Persistence Layer를 담당하는 Framework 개발 모델

나. OR-Mapping의 등장배경
- OOP에 있어서 관계형 데이터베이스와의 연계성을 분명히 하자는 의도

다. OR-Mapping의필요성
- 초기 OO전문가들은 Object와 관계형 DB간의 심각한 구조적 불일치를 깨고, OODB를 창안
- 그러나 대부분의 프로젝트에서 OODB가 RDB만큼 안정성이 보장되지않아, Risk가 존재하는 OODB가 확산 되지 못함

2. spring ORM

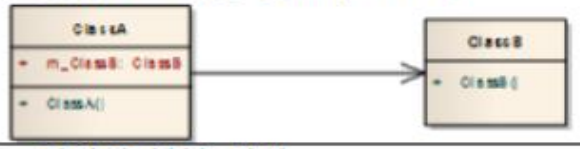
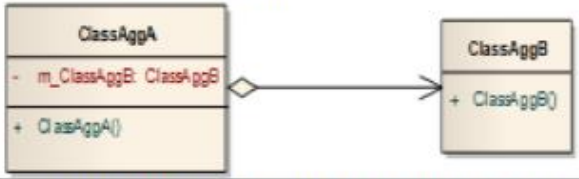
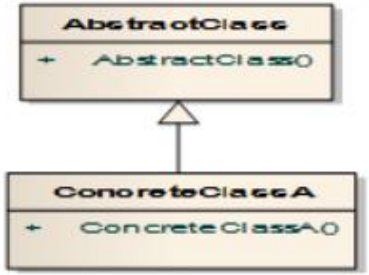
2. 객체지향 클래스와 데이터베이스 테이블간 매핑 관계
가. 객체지향 설계와 그에 맞는 DB 설계를 요구하는 OR Mapping



- 객체지향방법론으로 설계된 클래스를 데이터 저장소로 이용할 수 있는 Database Table로 매핑하는 절차나 방법론
- Database 연계 위해 기존 SQL에 의존하지 않고, 직접 테이블 컬럼을 자바 클래스에 매핑하거나 XML 선언을 통한 SQL 처리를 지원하는 방법론

2. spring ORM

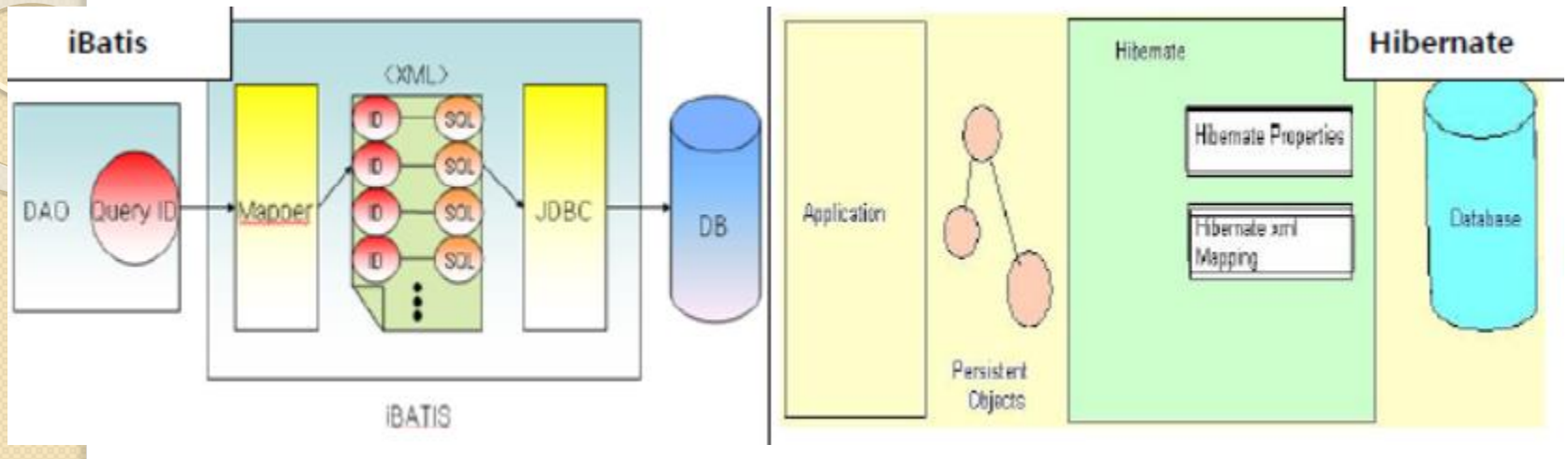
3. 클래스 다이어그램과 테이블간 매핑 관계

관계	설명	매핑 방안
Association (연관)	– 클래스간 가장 일반적인 관계 	– 1:1 관계(접근 빈도수가 많은 쪽으로 상대방 PK가 FK로 등록) – 1:N 관계(N쪽으로 1쪽의 PK를 FK로 매핑) – N:M 관계(새로운 관계 테이블을 추가)
Aggregation (집합)	– 전체와 부분 관계 	– 참고하는 테이블에서 단순 FK로 참조 – 비식별관계(약한 의존성)
Composition (복합연관)	– 동일 생명주기를 가진 전체와 부분 	– Delete Cascade 제약 추가 – 식별관계(강한 의존성)
Generalization (상속)	– 상속을 나타내는 일반화 관계 	– 1안 : 슈퍼 클래스와 서브 클래스를 별도의 테이블로 매핑 – 2안 : 슈퍼 클래스가 서브 클래스의 모든 속성을 포함한 단일 테이블로 매핑 – 3안 : 서브 클래스들이 슈퍼 클래스 속성을 상속받아 서브 클래스들을 테이블로 매핑

- OR Mapper에서 클래스는 테이블에, 클래스의 Attribute는 테이블의 Column에, Class Relationship은 테이블간 Relation에 매핑

3. spring ORM

4. XML에 임베디드된 myBatis와 Full-ORM 프레임워크 Hibernate의 비교



1) myBatis : - 소스 코드 외부에 정의된 SqlMapConfig.xml, SqlMap.xml 파일 정보를 기반으로 생성된 Mapped Statement를 이용하여 SQL과 객체간의 매핑 기능을 제공
- 개발자가 지정한 SQL, 저장프로시저 그리고 몇가지 고급 매핑을 지원하는 퍼시스턴스 프레임워크

2) Hibernate : 자바 클래스 객체와 데이터베이스 테이블 객체간 매핑을 통한 자동화된 질의 수행

3. spring ORM

5. myBatis와 Hibernate의 세부특징 비교

구분	iBatis	Hibernate
공통점	– XML 템플릿 등과 같은 표준 패턴을 이용하여 매핑 – JDBC와 같은 코드 사용시 보다 훨씬 간단한 구현 및 변경사항 적용이 가능 – 사용자에 의한 리소스 관리 및 코드의 중복 개발 감소 – 오픈 소스 기반의 ORM 프레임워크로 자유로운 이용과 배포 허용	
기반 사상	– SQL Mapping(Partial ORM)	– OR Mapping(Full ORM)
매핑 특징	– SQL 문을 활용한 객체 Mapping – 개발자가 직접 객체지향 관점에서 매핑 – 객체 모델과 데이터 모델 사이 매핑에 아무런 제약 사항이 없음	– 자바 클래스 객체와 테이블을 Mapping – 매핑 정보의 수정만으로 변경사항 적용 – SQL Mapper 보다 다양한 작업 수행
활용 특징	– 응답 지연 시간이 짧음 – 사용자 학습 곡선이 작음 – SQL 지식이 높아야 함 – 유연성이 우수	– 응답 지연 시간이 김(쿼리 자동생성) – 사용자 학습 곡선이 큼 – SQL 지식이 별로 필요 없음 – 유연성이 부족함
적용 방법	– 자바 객체를 SQL 문장에 매핑 – 자바 코드에서 SQL 부분을 제거하고 XML에 임베디드된 SQL 활용 – SQL 문장은 개발자에 의해 작성됨	– 자바 객체를 테이블 Row에 동기화 – 모든 SQL 문은 프레임워크에서 생성하고 실행하는 방식 – SQL 작업 필요시 HSQL 통해 이뤄짐
적용/조건	– SQL 문을 통한 튜닝이나 최적화 필요시 – 부적절한 DB 설계 상황 – 3rd Party 데이터베이스에 접근하는 경우 – 여러 개의 테이블과 하나의 자바 클래스 매핑되는 경우	– SQL Mapper 보다 효율적인 매핑 – 새로운 프로젝트가 시작된 상태 – 객체 모델과 데이터베이스 디자인이 미완성인 상태 – 하나의 테이블과 하나의 자바 클래스가 매핑되는 경우
장점/단점	– 복잡한 데이터 전송 환경에 효과적 – SQL의 장점 활용에 비교 우위	– OR Mapper에 의한 자동화 지원 – 간단한 CRUD 어플리케이션 테이블–클래스 매핑 사용시 단순성과 성능 비교 우위
유사 제품	– Oracle SQLJ, Pro*C embedded SQL – 대부분의 임베디드 SQL 시스템	– TopLink, JDO, ADO.NET
환경 설정	– SqlMapConfig.xml : DataSource, Data Mapper 및 Thread 관리 등의 설정 정보 – SqlMap.xml : 많은 캐시 모델, 파라미터 맵, Results Maps, Statements 정보 포함	– hibernate.properties : 전체 구성 지정 – hibernate.cfg.xml : hibernate.properties 파일에 대한 대용 혹은 중복정의에 활용

4. MyBatis setting

1. Root.context

```
<!-- //Oracle 접속부분 -->
<context:property-placeholder
location="classpath:mybatis/jdbc.properties"/>
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
destroy-method="close">
<property name="driverClass" value="${jdbc.driverClassName}" />
<property name="jdbcUrl" value="${jdbc.url}" />
<property name="user" value="${jdbc.username}" />
<property name="password" value="${jdbc.password}" />
<property name="maxPoolSize" value="${jdbc.maxPoolSize}" />
</bean>
<!-- 스프링 jdbc 즉 스프링으로 oracle 디비 연결 -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
<property name="dataSource" ref="dataSource" />
<property name="configLocation" value="classpath:mybatis/configuration.xml" />
</bean>
<bean id="session" class="org.mybatis.spring.SqlSessionTemplate">
<constructor-arg index="0" ref="sqlSessionFactory" />
</bean>
<!-- transactionmanager 선언 -->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<property name="dataSource" ref="dataSource" />
</bean>
```

4. MyBatis setting

2. configuration.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>
<typeAliases>
    <typeAlias alias="Emp" type="oracle.java.myBatis3.model.Emp" />
    <typeAlias alias="Dept" type="oracle.java.myBatis3.model.Dept" />
    <typeAlias alias="EmpDept" type="oracle.java.myBatis3.model.EmpDept" />
</typeAliases>
<mappers>
    <mapper resource="mybatis/Emp.xml" />
    <mapper resource="mybatis/Dept.xml" />
    <mapper resource="mybatis/EmpDept.xml" />
</mappers>
</configuration>
```

4. MyBatis setting

3. EmpDept.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">
<mapper namespace="EmpDept">
  <!-- Use type aliases to avoid typing the full classname every time. -->
  <resultMap id="EmpDeptResult" type="EmpDept">
    <result property="empno" column="empno"/>
    <result property="ename" column="ename"/>
    <result property="job" column="job"/>
    <result property="mgr" column="mgr"/>
    <result property="hiredate" column="hiredate"/>
    <result property="sal" column="sal"/>
    <result property="comm" column="comm"/>
    <result property="deptno" column="deptno"/>
    <result property="dname" column="dname"/>
    <result property="loc" column="loc"/>
  </resultMap>
   <select id="listEmp" parameterType="EmpDept"
   resultMap="EmpDeptResult">
        select e.empno, e.ename, e.job, d.dname, d.loc
    from emp e, dept d where e.deptno=d.deptno order by empno
  </select>
</mapper>
```

4. MyBatis setting

4. jdbc.properties

```
jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
jdbc.url=jdbc:oracle:thin:@127.0.0.1:1521:xe
jdbc.username=scott
jdbc.password=tiger
jdbc.maxPoolSize=20
```

4. MyBatis setting

6. SQL Map XML 파일
   1) MyBatis 의 가장 큰 장점은 매핑된 구문
   2) SQL Map XML 파일은 상대적으로 간단하다.
   3) JDBC코드와 비교하면 아마도 95% 이상 코드수가 감소
   4) MyBatis 는 SQL 을 작성하는데 집중하도록 만들어 짐

cache – 해당 명명공간을 위한 캐시 설정
cache-ref – 다른 명명공간의 캐시 설정에 대한 참조
resultMap – 데이터베이스 결과데이터를 객체에 로드하는 방법을 정의하는 요소
parameterMap – 비권장됨! 예전에 파라미터를 매핑하기 위해 사용되었으나 현재는 사용하지 않음
sql – 다른 구문에서 재사용하기 위한 SQL 조각
insert – 매핑된 INSERT 구문
update – 매핑된 UPDATE 구문.
delete – 매핑된 DELEETE 구문.
select – 매핑된 SELECT 구문.

4. MyBatis setting

7. **Select**

select 구문은 MyBatis 에서 가장 흔히 사용할 요소이다. 데이터베이스에서 데이터를 가져온다. 아마도 대부분의 애플리케이션은 데이터를 수정하기보다는 조회하는 기능을 많이 가진다. 그래서 MyBatis는 데이터를 조회하고 그 결과를 매핑하는데 집중하고 있다. Select 는 다음 예처럼 단순한 경우에는 단순하게 설정된다.

```
<select id="selectPerson" parameterType="int" resultType="hashmap">
SELECT * FROM PERSON WHERE ID = #{id}
</select>
```

- 이 구문의 이름은 selectPerson 이고 int 타입의 파라미터를 가진다.
  그리고 결과 데이터는 HashMap에 저장된다. 파라미터 표기법을 보자.

#{id}

이 표기법은 MyBatis 에게 PreparedStatement 파라미터를 만들도록 지시

4. MyBatis setting

7. **Select**

2] select 요소는 각각의 구문이 처리하는 방식에 대해 좀더 세부적으로 설정하도록 많은 속성을 설정

```
<select
id="selectPerson"
parameterType="int"
parameterMap="deprecated"
resultType="hashmap"
resultMap="personResultMap"
flushCache="false"
useCache="true"
timeout="10000"
fetchSize="256"
statementType="PREPARED"
resultSetType="FORWARD_ONLY"
>
```

속성	설명
id	구문을 찾기 위해 사용될 수 있는 명명공간내 유일한 구분자
parameterType	구문에 전달될 파라미터의 패키지 경로를 포함한 전체 클래스명이나 별칭
parameterMap	외부 parameterMap 을 찾기 위한 비권장된 접근방법. 인라인 파라미터 매핑과 parameterType 을 대신 사용하라.

4. MyBatis setting

7. Select(계속)

속성	설명
resultType	이 구문에 의해 리턴되는 기대타입의 패키지 경로를 포함한 전체 클래스명이나 별칭. collection 이 경우, collection 타입 자체가 아닌 collection 이 포함된 타입이 될 수 있다. resultType 이나 resultMap 을 사용하라.
resultMap	외부 resultMap 의 참조명. 결과맵은 MyBatis 의 가장 강력한 기능이다. resultType 이나 resultMap 을 사용하라.
flushCache	이 값을 true 로 셋팅하면, 구문이 호출될때마다 캐시가 지원질것이다(flush). 디폴트는 false 이다.
useCache	이 값을 true 로 셋팅하면, 구문의 결과가 캐시될 것이다. 디폴트는 true 이다.
timeout	예외가 던져지기 전에 데이터베이스의 요청 결과를 기다리는 최대시간을 설정한다. 디폴트는 셋팅하지 않는 것이고 드라이버에 따라 다소 지원되지 않을 수 있다.
fetchSize	지정된 수만큼의 결과를 리턴하도록 하는 드라이버 힌트 형태의 값이다. 디폴트는 셋팅하지 않는 것이고 드라이버에 따라 다소 지원되지 않을 수 있다.
statementType	STATEMENT, PREPARED 또는 CALLABLE 중 하나를 선택할 수 있다. MyBatis 에게 Statement, PreparedStatement 또는 CallableStatement 를 사용하게 한다. 디폴트는 PREPARED 이다.
resultSetType	FORWARD_ONLY|SCROLL_SENSITIVE|SCROLL_INSENSITIVE 중 하나를 선택할 수 있다. 디폴트는 셋팅하지 않는 것이고 드라이버에 다라 다소 지원되지 않을 수 있다.

4. MyBatis setting

8. **Insert, Update, Delete**

2] select 요소는 각각의 구문이 처리하는 방식에 대해 좀더 세부적으로 설정하도록 많은 속성을 설정

```
<insert
id="insertAuthor"
parameterType="domain.blog.Author"
flushCache="true"
statementType="PREPARED"
keyProperty=""
keyColumn=""
useGeneratedKeys=""
timeout="20000">

<update
id="insertAuthor"
parameterType="domain.blog.Author"
flushCache="true"
statementType="PREPARED"
timeout="20000">

<delete
id="insertAuthor"
parameterType="domain.blog.Author"
flushCache="true"
statementType="PREPARED"
timeout="20000">
```

4. MyBatis setting

8. Insert, Update, Delete (계속)

속성	설명
Id	구문을 찾기 위해 사용될 수 있는 명명공간내 유일한 구분자
parameterType	구문에 전달될 파라미터의 패키지 경로를 포함한 전체 클래스명이나 별칭
parameterMap	외부 parameterMap 을 찾기 위한 비권장된 접근방법. 인라인 파라미터 매핑과 parameterType 을 대신 사용하라.
flushCache	이 값을 true 로 셋팅하면, 구문이 호출될때마다 캐시가 지원질것이다(flush). 디폴트는 false 이다.
Timeout	예외가 던져지기 전에 데이터베이스의 요청 결과를 기다리는 최대시간을 설정한다. 디폴트는 셋팅하지 않는 것이고 드라이버에 따라 다소 지원되지 않을 수 있다.
statementType	STATEMENT, PREPARED 또는 CALLABLE 중 하나를 선택할 수 있다. MyBatis 에게 Statement, PreparedStatement 또는 CallableStatement 를 사용하게 한다. 디폴트는 PREPARED 이다.
useGeneratedKeys	(입력(insert)에만 적용) 데이터베이스에서 내부적으로 생성한 키(예를 들어, MySQL 또는 SQL Server 와 같은 RDBMS 의 자동 증가 필드)를 받는 JDBC getGeneratedKeys 메서드를 사용하도록 설정하다. 디폴트는 false 이다.
keyProperty	(입력(insert)에만 적용) getGeneratedKeys 메서드나 insert 구문의 selectKey 하위 요소에 의해 리턴된 키를 셋팅할 프로퍼티를 지정. 디폴트는 셋팅하지 않는 것이다.
keyColumn	(입력(insert)에만 적용) 생성키를 가진 테이블의 칼럼명을 셋팅. 키 칼럼이 테이블이 첫번째 칼럼이 아닌 데이터베이스(PostgreSQL 처럼)에서만 필요하다.

4. MyBatis setting

8. Insert, Update, Delete 예시문 (계속)

```
<insert id="insertAuthor" parameterType="domain.blog.Author">
    insert into Author (id,username,password,email,bio)
    values (#{id},#{username},#{password},#{email},#{bio})
</insert>
<update id="updateAuthor" parameterType="domain.blog.Author">
    update Author set
    username = #{username},
    password = #{password},
    email = #{email},
    bio = #{bio}
    where id = #{id}
</update>
<delete id="deleteAuthor" parameterType="int">
        delete from Author where id = #{id}
</delete>
```

9 . Mybatis SQL Return 값

1) Insert - 1 (여러개인 경우 1)
2) Update - 업데이트 된 행의 개수 (없으면 0)
3) Delete - 삭제 된 행의 개수 (없으면 0)

5-1. MyBatis 입력 setting

10. 입력 전 Setting 예시
- 아래처럼 만약 Board라는 게시판에서 입력을 수행시 boardID값을 기존에 최대값에 +1 한 다음에 그 값을 입력하고 싶다면 아래의 같이 처리

```
<insert id="insertBoard" parameterType="Board">
    <selectKey resultType="Long" keyProperty="boardID" order="BEFORE">
        SELECT MAX(boardID)+1 FROM board
    </selectKey>
    INSERT INTO board(boardID, title, content)
    VALUES(#{boardID}, #{title}, #{content})
</insert>
```

5-2. MyBatis 입력 setting

11. INSERT 됨과 동시에 생성된 키를 가져오게 하는 **useGeneratedKeys** 속성

```
<!-- 게시글 INSERT -->
<insert id="insertPost" parameterType="java.util.HashMap" useGeneratedKeys="true" keyColumn="POST_NO"
keyProperty="postNo" >
        INSERT INTO POST   (
                                    BBS_NO                  --게시판번호
                                  , POST_NO                 --게시글번호
                                  , POST_TITLE              --게시글제목
                                  , CONTENTS                --게시글내용
                                  , WRT_ID                  --작성자ID
                                )
        VALUES              (
                                  #{bbsNo,jdbcType=VARCHAR}
                                  , TO_CHAR(POST_SEQ.NEXTVAL)          --시퀀스 사용
                                  , #{postTitle,jdbcType=VARCHAR}
                                  , #{contents,jdbcType=VARCHAR}
                            )
</insert>
```

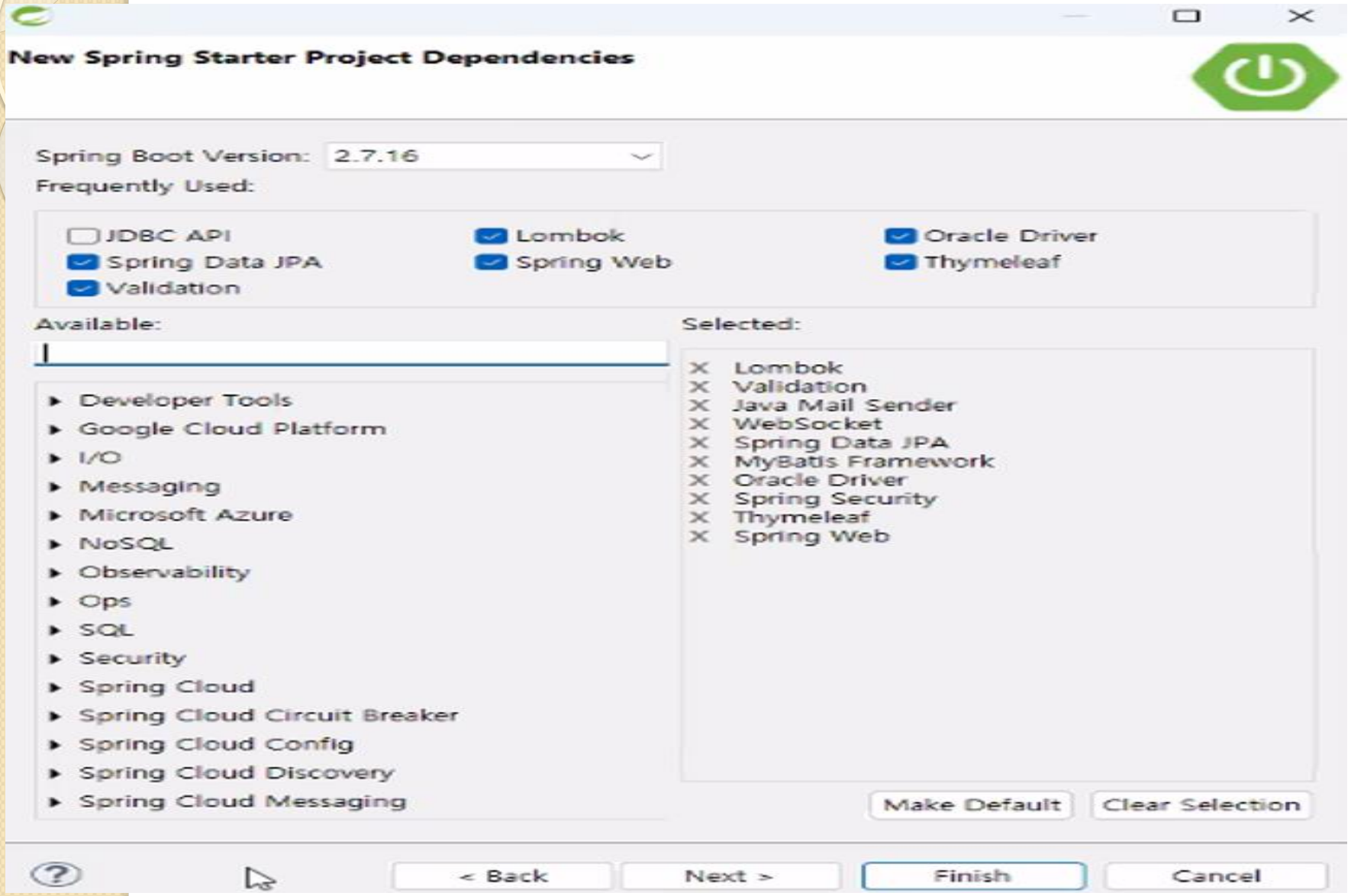
useGeneratedKeys 속성을 사용할 때 keyColumn, keyProperty 속성이 같이 쓰입니다.
useGeneratedKeys = "true"(기본값 false), keyColumn = [PK 컬럼명], keyProperty = [매핑할 변수명]
이렇게 INSERT한 POST_NO 값을 "postNo"로 바로 가져올 수 있음

2. 모델 폼(vo 객체)을 만들어서 넘겨줬다고 하면

```
public Integer insertPost(BoardForm boardForm) {
        this.sqlSession.insert("board.insertPost", boardForm);
        return boardForm.getPostNo();
}
```

이런식으로 받아올 수 있음

5-3. MyBatis 환경 setting



Security 추가

5-4. MyBatis JSP setting

1. **build.gradle 추가**

```
dependencies {
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
    implementation 'javax.servlet:jstl'
}
```

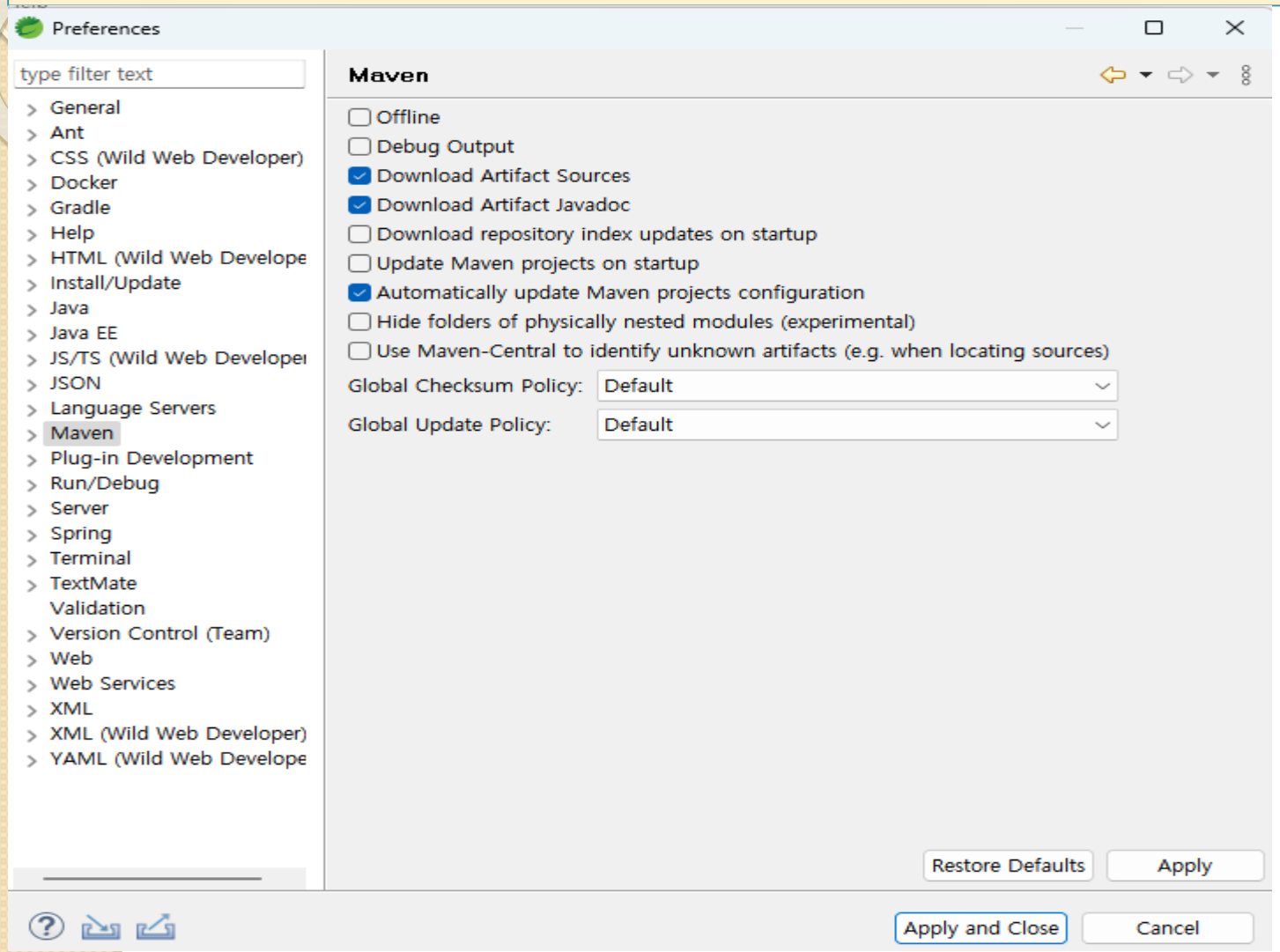
2. application.properties 추가

```
spring.mvc.view.prefix: /WEB-INF/views/
spring.mvc.view.suffix: .jsp

#JSP수정시 서버 재시작없이 바로 적용될 수 있게 설정
server.servlet.jsp.init-parameters.development=true
```

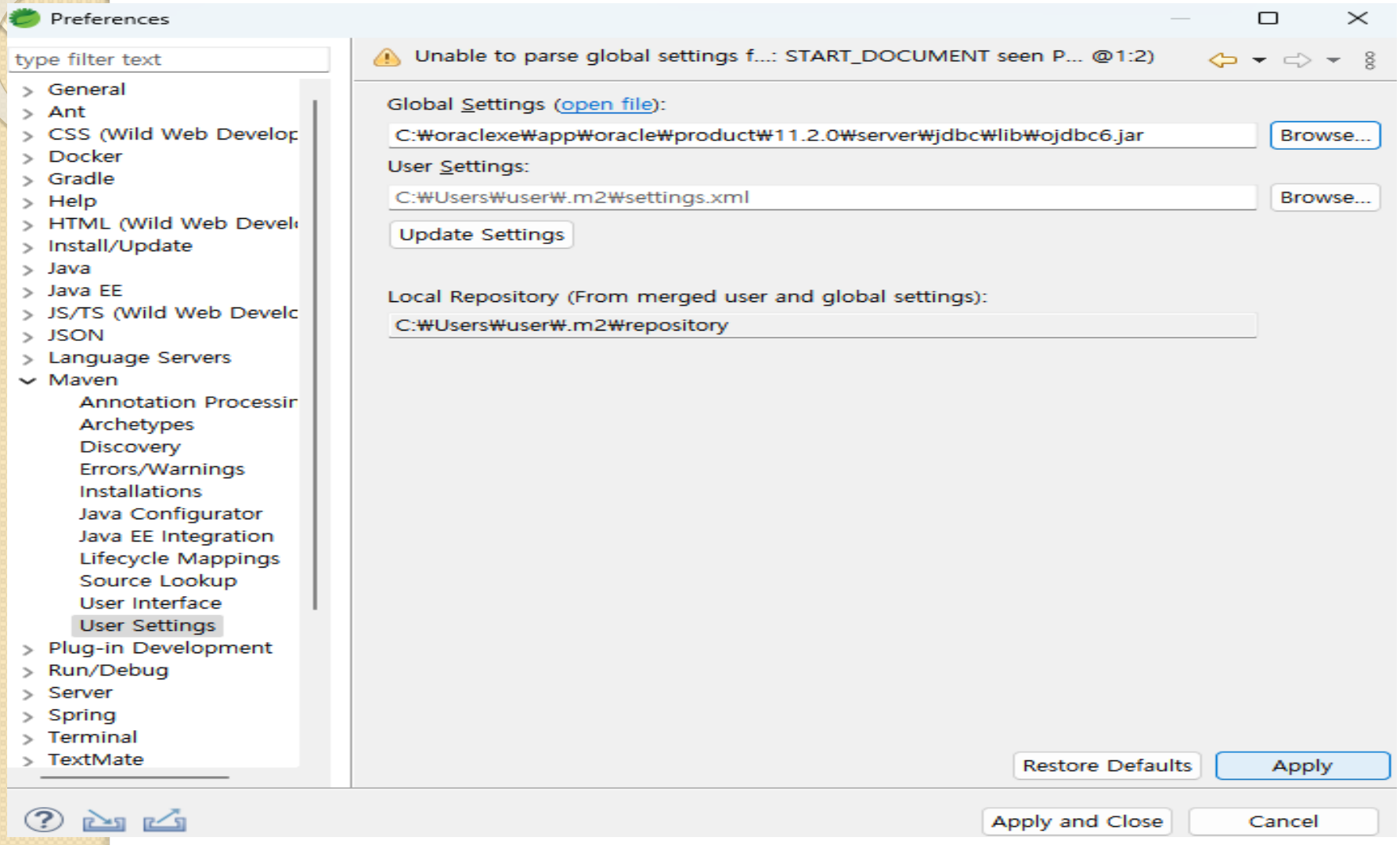
5-5-1. MyBatis XML D/L 오류시

1. D/L Artifact JavaDoc -> Apply
- Window -> Preference -> Maven -> 옵션('아티팩트 javadoc 다운로드')을 선택



5-5-2. MyBatis XML D/L 오류시

1. UserSetting은 건드리지 않고 Apply And Close



5-5-3. MyBatis XML D/L 오류시

1. 아래 사이트 접근후 진행