MyBatis

MyBatis 란?

- SQL 기반의 Persistence Framework
- JDBC Framework
- 특징
  - SQL문과 프로그램 코드의 분리
  - 공통된 JDBC 코드를 MyBatis 가 처리
  - 동적쿼리, 캐시모드 등 다양한 서비스 제공

MyBatis 설치 - 라이브러리

- MyBatis 메인 사이트
  - http://blog.mybatis.org
- 다운로드
  - http://github.com/mybatis
  - mybatis-3 클릭 -> releases 탭 클릭 후 다운로드
- Maven Repository

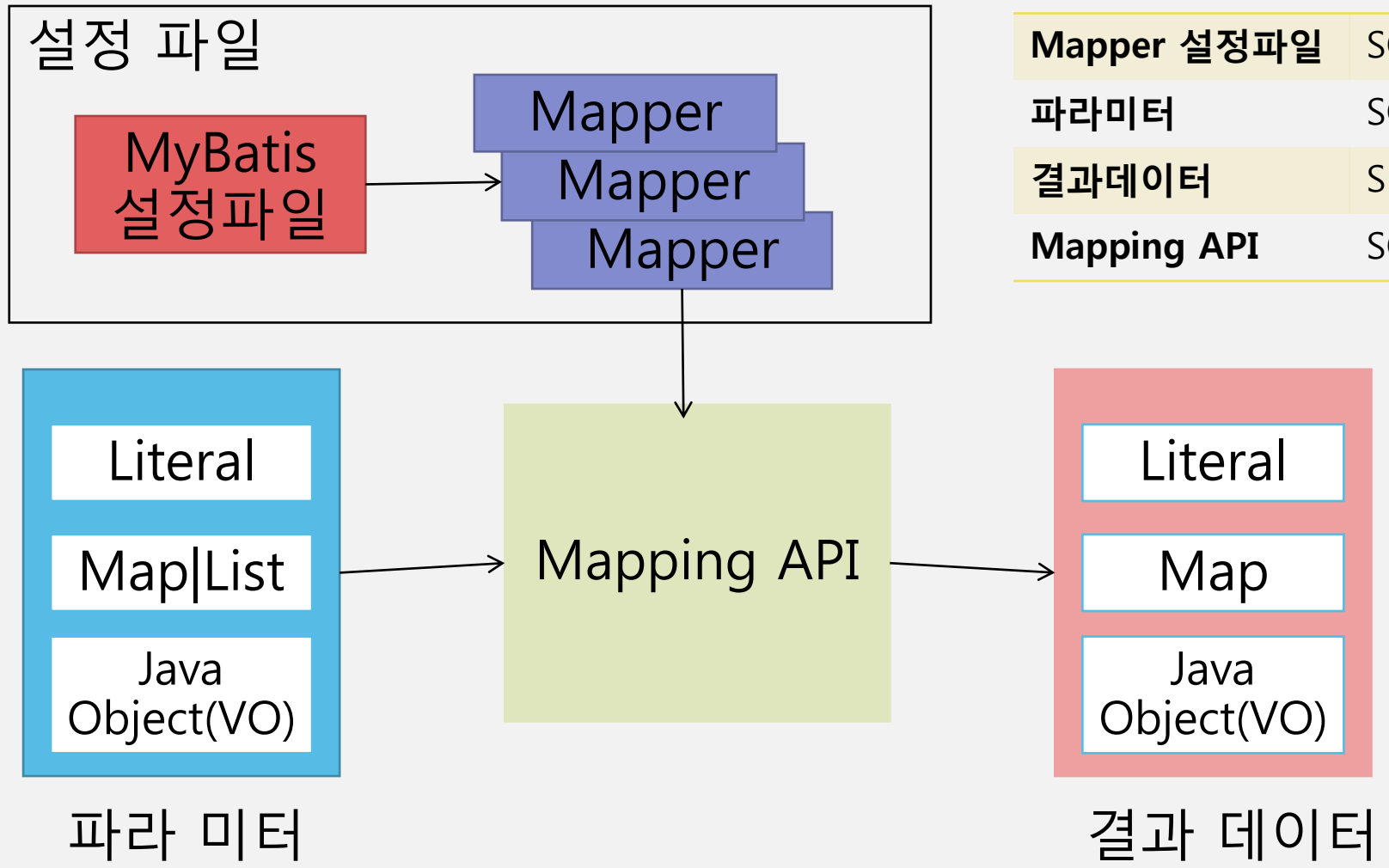
```
<dependency>
   <groupId>org.mybatis</groupId>
   <artifactId>mybatis</artifactId>
   <version>X.X.X</version>
</dependency>
```

- 튜토리얼
  - http://mybatis.github.io/mybatis-3/
  - http://mybatis.github.io/mybatis-3/ko/index.html (한글)

MyBatis 설치 - Eclipse Plug-in

- MyBatis 에디터 플러그인
  - http://code.google.com/a/eclipselabs.org/p/mybatiseditor/
- 이클립스 설치 마법사를 이용해 설치
  - 플러그인 경로
  - http://mybatiseditor.eclipselabs.org.codespot.com/git/org.eclipselabs.mybatiseditor.updatesite
- 이클립스 Marketplace에서 인스톨(help – Eclipse Marketplace...)
  - Find 에서 mybatis editor 로 조회
- 제공기능
  - MyBatis 설정 파일 자동생성
  - 설정 파일 구문 자동완성 기능 제공
  - 사용하는 곳에서 SQL문 선언 부로 이동

MyBatis Framework 의 실행 구조



MyBatis 설정파일	전역설정파일
Mapper 설정파일	SQL문 관련 설정
파라미터	SQL문에 넣어줄 값
결과데이터	SELECT 결과
Mapping API	SQL문 실행 처리 MyBatis API

일단 만들어 보기

1. Java Project 생성
2. MyBatis API 및 Dependency API 를 Library에 추가
3. MyBatis 설정 파일 생성
4. Mapper 설정파일 생성 및 MyBatis 설정파일에 등록
5. CRUD 작성

MyBatis 구성요소 역할 – MyBatis 설정파일

- **마이바티스 설정파일**
  - 마이바티스가 JDBC 코드를 실행하는 데 있어 필요로하는 전역적 설정을 한다.
  - TypeAlias 설정
    - <typeAliases>
  - DB연동을 위한 환경 설정
    - <environments>
  - Mapper 설정파일 등록
    - <mappers>
  - Mybatis 실행 관련 설정
    - <settings>

MyBatis 구성요소 역할 – Mapper 설정파일

- **Mapper 설정 파일**
  - SQL문과 관련된 설정을 하는 파일
  - 한 프로젝트에 여러 개의 문서를 작성 할 수 있다.
  - 마이바티스 설정파일에 등록해야 한다.
  - 주요 구성 요소
    - SQL 문 등록 태그
      - SQL문 태그의 구성요소 : Parameter, Result, SQL 문 등록
      - SQL 태그 : <insert>, <delete>, <update>, <select>
      - 공통 SQL 문 설정 태그 : <sql>
    - select 결과 처리 설정
      - <resultMap>

MyBatis 구성요소 역할 – 파라미터/결과 데이터

- Parameter Mapping
  - SQL 문에 실행시점에 넣어줄 값 설정
  - PreparedStatement의 ? 역할
  - 넣어줄 값이 하나일 경우 : Literal
  - 넣어줄 값이 여러 개일 경우 : Map, List 또는 Java Object(VO)
- Result Mapping
  - 조회 결과 데이터 저장 설정
  - SELECT 태그로 조회한 **하나의 ROW**를 담을 데이터 타입 설정
  - 조회 칼럼이 하나일 경우 : Literal
  - 조회 칼럼이 여러 개일 경우 : Map 또는 Java Object(VO)

MyBatis 구성요소 역할 – Mapping API

- Mapping API
  - MyBatis API
  - **SqlSessionFactoryBuilder**
    - 역할 - SqlSessionFactory 생성
  - **SqlSessionFactory**
    - MyBatis의 전역 정보를 가지고 실행을 제어
    - SqlSession 생성
    - Application당 하나만 생성하는 것이 권장됨
  - **SqlSession**
    - 역할 – 쿼리 실행 처리
    - 작업 단위 별로 SqlSessionFactory로 부터 생성

MyBatis 설정파일 – 주요 설정 1

- **ROOT 태그**
  - <configuration>
- 주요 설정 태그 - <typeAliases>, <environments>, <mappers>
- **<typeAliases>**
  - Java Type에 대한 별칭 설정
  - 파라미터, 결과 데이터, 각종 설정 등에서 사용할 Java Type에 대한 짧은 이름 설정

class 단위로 등록

```
<typeAliases>
   <typeAlias alias="customer" type="vo.Customer"/>
   <typeAlias alias="board" type="vo.Board"/>
</typeAliases>
```

package 단위로 등록 – class 이름의 첫 글자를 소문자로 한 이름으로 등록됨

```
<typeAliases>
   <package name="customer.vo"/>
   <package name="board.vo"/>
</typeAliases>
```

MyBatis 설정파일 – 주요 설정 2

- <typeAliases>
  - Framework이 제공하는 주요 기본 TypeAlias

별칭	자바 타입
_primitive	프리미티브 타임(ex : _int – int) cf) _char 은 없음
primitive	Wrapper 타입 (long – java.lang.Long) cf) char은 없음
string	java.lang.String
date	java.util.Date
map	java.util.Map
hashmap	java.util.HashMap
list	java.util.List
arraylist	java.util.ArrayList
object	java.lang.Object

MyBatis 설정파일 – 주요 설정 3

- **<environments>**
  - Transaction 관리자와 DataSource설정 및 Database연결정보를 하는 태그
  - 하위 태그인 environment 를 이용해 여러 개의 환경을 설정 가능
  - 사용할 환경의 id를 default 속성의 값으로 등록

```
<environments  default="development">
  <environment  id="development">
    <transactionManager  type="JDBC"/>
    <dataSource  type="POOLED">
      <property name="driver" value="jdbc 드라이버"/>
      <property name="url" value="DB서버 URL"/>
      <property name="username" value="DB 연결계정"/>
      <property name="password" value="DB 연결 계정 암호"/>
      <property name="..." value="..."/>
    </dataSource>
  </environment>
</environments>
```

MyBatis 설정파일 – 주요 설정 4

- **<mappers>**
  - mappers 설정파일 등록
  - Mapper 설정 파일은 classpath상에 위치시킨다.

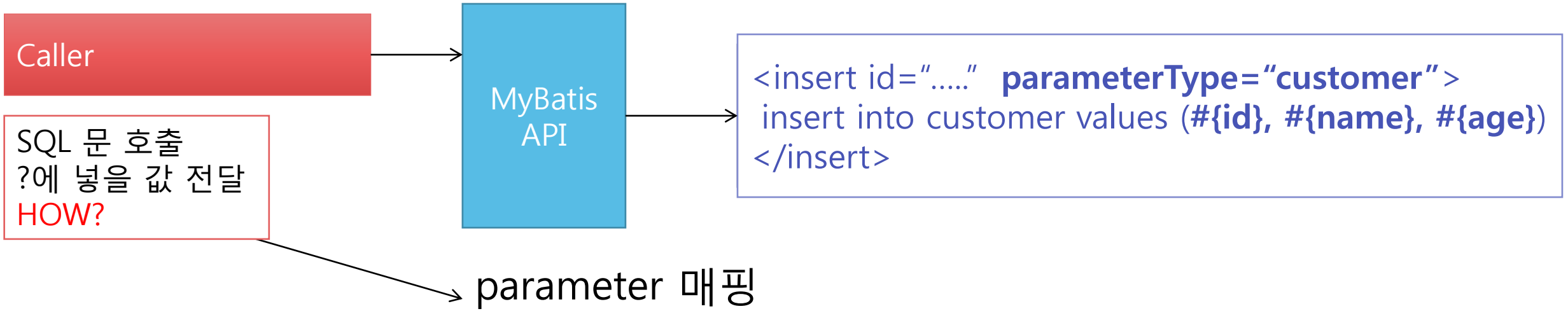
```
<mappers>
   <mapper resource="mybatis/config/customerMapper.xml"/>
   <mapper resource="mybatis/config/boardMapper.xml"/>
</mappers>
```

Mapper 설정 파일 - 개요

- SQL 문 등록 설정 파일
  - SQL 등록 태그
  - 결과 외부 매핑 태그
  - 공통 쿼리 등록을 위한 <sql> 태그
- 작성 후 CLASSPATH내 저장
- MyBatis 설정 파일에 <mappers>를 이용해 등록
- 설정파일은 여러 개 만들어도 상관없다.
- <mapper> : ROOT 태그
  - 속성 : namespace (필수)
    - Mapper 파일 식별값
    - namespace.sqlid 로 SQL문을 호출한다. (단 sql id가 유일하면 namespace는 생략가능)

Mapper 설정 파일 설정 – 파라미터 매핑 1

- 파라미터 매핑 (Parameter 매핑)
  - Parameter 값과 SQL문 ?를 연결하는 구문.



parameterType	설명
리터럴 타입	?가 하나인 경우
Map||List	
Java Object(VO)	?가 하나 이상인 경우

Mapper 설정 파일 설정 – 파라미터 매핑 2

- 구문 : PreparedStatement 의 ? 처리
  - #{property}
  - #{property, javaType=value, jdbcType=value}
  - #{} 부분을 ? 로 변환한 뒤 값을 PreparedStatement의 setXXX메소드를 이용해 치환
  - property : 필수
    - parameterType이 Map인 경우 – key 값
    - parameterType이 List 인 경우 - **${list[idx]}** 로 설정
    - parameterType이 Java 객체(VO)인 경우 – property 명
      - 실행 시점에 getter 호출
    - parameterType이 리터럴인 경우 – 관계 없다.
  - javaType, jdbcType : 선택
    - java와 jdbc 타입을 명시할 경우 사용
    - 일반 Java 기본 타입은 상관 없다.
    - 주의 : **null입력하는 경우 오라클은 jdbcType을 반드시 명시해야한다.**

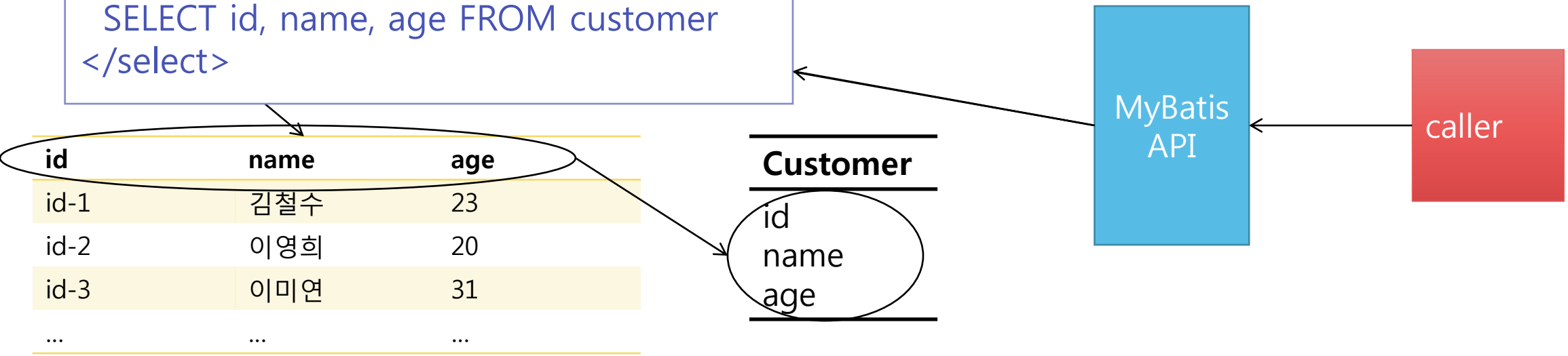
Mapper 설정 파일 설정 – 파라미터 매핑 3

- 구문 : Copy & Paste 방식
  - ${property}
  - 값을 ${} 자리에 그대로 카피해서 붙이는 방식
    - 값(?) 에 넣을 경우 비추천
    - 쿼리 구문을 파라미터를 이용해 만들 경우 사용
  - property : 필수
    - parameterType이 Map인 경우 – key 값
    - parameterType이 List 인 경우 - **${list[idx]}** 로 설정
    - parameterType이 Java 객체(VO)인 경우 – property 명
      - 실행 시점에 getter 호출
    - parameterType이 Literal인 경우 – ${value} 로 설정

Mapper 설정 파일 설정 – 결과 데이터 매핑 1

- 결과 데이터 매핑 (ResultMapping)
  - SELECT한 조회결과를 결과데이터 객체에 연결하는 것.

```
<select id="....."  resultType="customer">
  SELECT id, name, age FROM customer
</select>
```



id	name	age
id-1	김철수	23
id-2	이영희	20
id-3	이미연	31
...	...	...

ResultSet
ResultSet의 조회결과 전달
HOW ? → ResultMapping
-내부 매핑
-외부 매핑

ResultType	설명
리터럴	조회컬럼이 하나인 경우
Map	
Java Object(VO)	조회 컬럼이 하나 이상인 경우

Mapper 설정 파일 설정 – 결과 데이터 매핑 2

- 내부 매핑
  - ResultSet의 컬럼명과 ResultType에 설정한 객체의 property를 매핑
  - resultType이 Java 객체(VO)인 경우 : property 명과 매핑 (setter 호출)
  - resultType이 Map 인 경우 : 컬럼명 – key, 컬럼값 – value 로 들어감
- 외부 매핑
  - <resultMap> 태그로 ResultSet과 resultType의 매핑을 외부에 설정
  - 장점
    - 재사용성
    - 가독성
    - 내부 매핑 보다 다양한 설정 가능
    - JOIN 연산 관련 설정

Mapper 설정 파일 설정 – 결과데이터 외부매핑 3

- <resultMap>
  - 하나의 외부매핑 설정
  - 속성 : id , type, extends
  - 하위 태그
    - <id>,<result>
    - <constructor>
    - <association>, <collection>
- <result> / <id>
  - 조회결과를 setter 메소드를 통해 대입한다.
  - 속성 : 필수 - column, property 선택 - javaType, jdbcType

Mapper 설정 파일 설정 – 결과데이터 외부매핑 3

- <constructor>
  - <resultMap>의 하위태그
  - 조회결과를 생성자(Constructor)를 통해 대입한다.
  - 하위 태그
    - <idArg>, <arg>
  - <idArg>, <arg>
    - 생성자 매개변수와 조회 컬럼을 연결하는 태그
    - 선언 순서가 생성자 매개변수 순서가 된다.
    - 속성 : 필수 – column, 선택 – javaType, jdbcType
    - javaType을 설정하지 않으면 조회결과의 타입을 Object 타입으로 넣으려고 한다.
- <result> <constructor> 같이 사용할 수 있다.

Mapper 설정 파일 설정 – 결과데이터 외부매핑 4

- < association >
  - 1 : 1 참조관계 처리
  - 속성 : property, javaType, resultMap, column(참조키 설정)
  - 하위 태그 : <id>, <result>, <constructor>, <association>, <collection>
- <collection>
  - 1 : N 참조관계 처리
  - 속성
    - property : Java Object(VO)의 프라퍼티. 프라퍼티 타입은 Collection이어야 한다.
    - ofType : 하나의 row값과 매칭되는 Java 객체(VO)타입
  - 하위 태그 : <id>, <result>, <constructor>, <association>, <collection>

Mapper 설정 파일 설정 – 결과데이터 외부매핑 5

- 기존 설정한 결과 외부매핑 재사용
- <resultMap> - extends 속성 사용
  - 기존 설정의 내용을 물려 받은 뒤 추가 설정만 한면된다.
- <association> <collection> - resultMap 사용
  - 각 태그 내에서 해야 하는 매핑설정을 다른 resultMap 설정을 가져다 쓴다.

MyBatis를 이용한 CRUD 처리 - 공통

- mapper 설정 파일에 sql문 등록
  - <insert><delete><update><select> 태그 이용
  - <sql> 태그
    - 여러 SQL 문에서 공통으로 사용할 sql문을 등록하는 태그
    - 속성 : id
  - <include> 태그
    - <sql> 태그로 등록한 공통태그를 참조하는 곳에서 사용하는 태그
    - 속성 : refid
- SqlSession 의 메소드를 이용해 처리
  - SqlSessionFactory객체.openSession(), openSession(true) 로 생성
  - SQL문 처리 메소드 호출 후 close() 를 통해 session 닫기.

MyBatis를 이용한 CRUD 처리 - INSERT 처리

- mapper 태그
  - <insert>
  - 속성 : id, parameterType
  - 하위 태그 : <selectKey>, <include>, Dynamic 태그
- <selectKey>
  - insert 쿼리에 넣을 값을 실행 시점에 조회해야 하는 경우 사용
  - 속성 : keyProperty, resultType, order(값 : AFTER-기본, BEFORE)
- 메소드
  - insert(tagId : String) : int
  - insert(tagId : String, parameter : Object) : int

MyBatis를 이용한 CRUD 처리 - DELETE 처리

- mapper 태그
  - <delete>
  - 속성 : id, parameterType
  - 하위 태그 : <include>, Dynamic 태그
- 메소드
  - delete(tagId : String) : int
  - delete(tagId : String, parameter : Object) : int

MyBatis를 이용한 CRUD 처리 - UPDATE 처리

- mapper 태그
  - <update>
  - 속성 : id, parameterType
  - 하위 태그 : <include>, Dynamic 태그
- 메소드
  - update(tagId : String) : int
  - update(tagId : String, parameter : Object) : int

MyBatis를 이용한 CRUD 처리 - SELECT 처리

- mapper 태그
  - <select>
  - 속성 : id, parameterType, resultType || resultMap
  - 하위 태그 : <include>, Dynamic 태그
- 메소드
  - 단일 ROW 조회 메소드
    - selectOne(tagId : String, parameter : Object) : T
    - selectOne(tagId : String) : T
  - 다중 ROW 조회 메소드
    - 조회결과들(ResultType객체)을 List로 묶어 리턴
      - selectList(tagId : String) : List<E>
      - selectList(tagId : String, parameter : Object) : List<E>
    - 조회결과들(ResultType객체)을 Map으로 묶어 리턴. key값으로 어떤 컬럼을 사용할지 지정 해야함.
      - selectMap(tagId : String, param : Object, key : String) : Map<K,V>
      - selectMap(tagId : String, key : String : Map<K,V>

동적(Dynamic) SQL - 개요

- 제어문 역할을 하는 태그들을 이용해 SQL문을 동적으로 만드는 태그들.
- 조건처리, 반복처리
- 하나의 SQL 태그로 여러 SQL문을 실행 시킬 수 있다.
- 가독성이 떨어져 유지 보수에 어려움을 줄 수 있다.
- **동적 쿼리를 쓸 경우 파라미터는 값이 하나 일 때 파라미터의 이름을 value 로 매핑해야 literal type으로 설정이 가능하다.**
- **//TODO 테스트**

동적(Dynamic) SQL - 조건처리

- <if>
  - 단일 조건을 처리하는 태그
  - 속성
    - test – 조건식 지정
  - content : 조건 만족시 실행될 SQL문
- <choose>(when, otherwise)
  - 다중 조건을 처리하는 태그
  - 하위 태그
    - when, otherwise
  - when : 조건 처리 태그
    - 속성 : test – 조건식 지정
    - content : 조건 만족 시 실행될 SQL문
  - otherwise : when에 지정된 모든 조건이 false일 경우 실행될 쿼리 지정
    - content : 조건 만족 시 실행될 SQL문

동적(Dynamic) SQL - 반복처리

- <foreach>
  - 파라미터의 값이 배열이나 Collection(Set/List) 로 넘어온 경우 그것을 반복 처리할 때 사용
    - 주로 IN 연산자와 연결 되 많이 사용
  - 속성
    - collection : 반복할 값을 가진 parameter 지정 (배열, Set, List).
    - item : 반복 시 조회될 값을 저장할 변수 명 지정. Content에서 사용.
    - index : 몇 번째 반복인지를 저장할 변수 지정. 0 부터 시작
    - open : 반복 시작 전에 가장 앞에 붙일 문자 지정
    - close : 반복 종료 후에 가장 뒤에 붙일 문자 지정
    - separator : 반복 되는 값들 사이에 붙일 구분자 지정.

동적(Dynamic) SQL – 쿼리 정돈 태그 (trim)

- Trimming 태그란?
  - 동적쿼리 태그를 이용해 SQL문이 만들어 질 경우 조건등에 따라 구문상 오류가 생길 수 있다. 그것을 정리하기 위한 태그들을 말함
  - 내용(Content)에 동적 SQL문이 옴.
- <where>
  - where 절을 만드는 태그
- <set>
  - update 구문의 set 절을 만드는 태그
  - set 절의 마지막에 , 를 제거하는 역할

동적(Dynamic) SQL – 쿼리 정돈 태그 (trim)

- <trim>
  - 개발자에게 구문정리에 대한 더 많은 자유도를 주기위한 태그
  - 속성
    - prefix : 처리 후 내용(content)가 있으면 앞에 붙일 것 지정
    - prefixOverrides : 처리 후 내용의 처음에 왔을 경우 삭제할 내용 지정
    - suffix : 처리 후 내용(content)가 있으면 가장 뒤에 붙일 것 지정
    - suffixOverrides : 처리 후 내용의 마지막에 왔을 경우 삭제할 내용 지정