**[ iBatis기본 개요]**

**① JDBC 코드의 패턴**   
- Connection -> Statement -> 쿼리전송 -> 연결 close

-벤더별 JDBC 드라이버 로딩 -> Connection 객체 생성 (DBCP 사용가능)

- 모든 JDBC 코드는 위의 패턴을 가진다.

- 이 패턴을 캡슐화 하여 JDBC 코드를 간편하게 사용할 수 있도록 Framework화 가능

**② iBATIS 란**

- SQL 실행 결과를 JavaBeans(객체:DTO or VO) 혹은 Map 객체에 매핑 해주는 Persistence(영속적:DB적) 솔루션으로 SQL을 소스 코드가 아닌 XML로 따로 분리해 관리하도록 지원

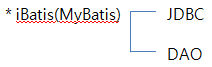
③ 장점

- SQL 문장과 프로그래밍 코드의 분리

- JDBC라이브러리를 통해 매개변수를 전달하고 결과를 추출하는 일을 간단히 처리

- 자주 쓰이는 데이터를 변경되지 않는 동안에 임시 보관(Cache) 가능

- 트랜잭션처리 제공



- iBATIS framework에 대한 설정

\*iBATIS SqlMap Config(sqlmap-config.xml) 작성하기

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMapConfig   
PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"   
"<http://ibatis.apache.org/dtd/sql-map-config-2.dtd>">

<sqlMapConfig>

<transactionManager type="JDBC" commitRequired="false">  
<dataSource type="SIMPLE">  
<property name="JDBC.Driver" value="oracle.jdbc.driver.OracleDriver"/>  
<property name="JDBC.ConnectionURL" value="jdbc:oracle:thin:@127.0.0.1:1521:ORCL"/>  
<property name="JDBC.Username" value="scott"/>  
<property name="JDBC.Password" value="tiger"/>  
</dataSource>  
</transactionManager>

<sqlMap resource="config/member.xml"/>

<sqlMap resource="config/DEPT.xml"/>

<sqlMap resource="config/Board.xml"/>

</sqlMapConfig>

- SQL문을 등록하는 설정파일

**SqlMap (User.xml)**

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMap   
PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"   
"<http://ibatis.apache.org/dtd/sql-map-2.dtd>">  
<sqlMap>  
<!-- insert 태그 작성 -->  
<insert id="insertMember" parameterClass="ibatis.dto.MemberDTO">  
insert into member(id, password, name, register\_Number1, register\_Number2, mileage) values(#id#,#password#,#name#,#registerNumber1#,#registerNumber2#,#mileage#)

values(?,?,?,?,?,\_ ........  
</insert>  
<!-- select 태그 작성 -->  
<select id="selectMemberById" parameterClass="java.lang.String" resultClass="ibatis.dto.MemberDTO">  
select id,  
password  
name,  
register\_Number1 as registerNumber1,  
register\_Number2 as registerNumber2,  
mileage  
from member  
where id=#id# // where id = ?  
</select>  
</sqlMap>

<!-- xxx.insert("insertMember", mdto); -->  
<!-- Object obj = xxx.queryForObject("selectMemberById", "id"); 나중에 MemberDTO로 캐스팅 -->

**3. i**

**BATIS Data Access Object 작성**

**iBATIS Data Access Object 작성**

package ibatis.dao;

import ibatis.dto.MemberDTO;

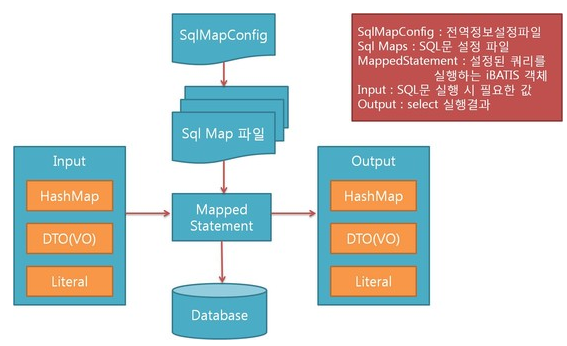
import java.io.Reader;  
import java.sql.SQLException;

import com.ibatis.common.resources.Resources;  
import com.ibatis.sqlmap.client.SqlMapClient;  
import com.ibatis.sqlmap.client.SqlMapClientBuilder;

public class MemberDAO {  
private static MemberDAO instance = new MemberDAO();  
private SqlMapClient sqlMap;//sql문 실행 메소드를 가진 객체  
private MemberDAO(){  
try {  
 Reader reader = Resources.getResourceAsReader("config/SqlMapConfig.xml");  
 sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);  
} catch (Exception e) {  
e.printStackTrace();  
}  
}  
public static MemberDAO getInstance(){  
return instance;  
}  
//회원 데이터 insert 메소드  
public void insertMember(MemberDTO mdto) throws SQLException{  
sqlMap.insert("insertMember", mdto);  
}  
//id로 회원 정보 조회  
public MemberDTO selectMemberById(String id) throws SQLException{  
return (MemberDTO) sqlMap.queryForObject("selectMemberById", id); }}

iBATIS 실행 흐름도

**1. iBATIS 실행 흐름도**



**2. iBATIS 설정 파일**

① **설정파일은 XML기반으로 작성**

② **SqlMapConfig**

- 전역 설정 위한 파일 : iBATIS Framework가 실행되는 데 필요한 여러 설정들을 한다

- Transaction 관리정보, DataSource 생성을 위한 설정 정보, SqlMap 파일 위치 등

③ **SqlMap**

- SQL문을 등록

- SQL문을 실행하기 위해 필요한 input Data와 OutputData에 대한 설정을 한다.

④ 설정 파일들은 작성 후 classpath내 저장한다.

① SQLMapConfig 설정 파일(SqlMapConfig.xml)

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE sqlMapConfig PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN" "<http://ibatis.apache.org/dtd/sql-map-config-2.dtd>">

<sqlMapConfig>

<전역 설정 옵션>

<settings useStatementNamespaces=“false” cacheModelsEnabled=“true” />

<트랜 잭션 관리>

<transactionManager type="JDBC" commitRequired="false">  
<DataSource 설정>

<dataSource type="SIMPLE">  
<property name="JDBC.Driver" value=“DriverClassName"/>  
<property name="JDBC.ConnectionURL" value=“Url”/>  
<property name="JDBC.Username" value=“User/>  
<property name="JDBC.Password" value=“Password"/>  
</dataSource>  
</transactionManager>

<sqlMap 파일 참조>

<sqlMap resource="config/member.xml"/>

</sqlMapConfig>

**② <typeAlias> 요소**

- 설정파일에서 사용할 클래스의 별칭(alias) 설정

- SqlMap에서도 설정 가능

- iBATIS는 정의된 alias로 언제든지 원래 type에 접근 가능

- SqlMapConfig에 설정하면 모든 SqlMap에서 사용가능

· <typeAlias alias="mdto" type="member.dto.MemberDTO" />

- Framework에서 미리 정의하여 제공하는 typeAlias

· Transaction manager : JDBC, JTA, EXTERNAL

· Data types : string, int, long, double, boolean, hashmap, arraylist, object 등

· Data source factory : SIMPLE, DBCP, JNDI

**③ <transactionManager> 요소**

**- Transaction Manager 타입 설정**

**· JDBC : 단순 JDBC 기반의 Transaction Manager를 제공함.**

**· JTA : application이 동작하는 컨테이너 기반의 Transaction Manager를 제공함.**

**· EXTERNAL : 트랜잭션 관리를 iBATIS에서 하지 않음.**

- <dataSource> : iBATIS에서 사용할 DataSource를 생성.

**· SIMPLE : iBATIS 자체 제공하는 DataSourceFactory 사용.**

**· DBCP : Jakarta Commons Database Connection Pool 구현함.**

**· JNDI : Naming서버에 등록된 DataSource를 사용함.**

보통 Container가 제공하는 것을 사용.

· 하위태그를 이용하여 필요한 property들을 설정한다. (driver, url 계정정보 등)

**④ <sqlMap> 요소**

- SQL문을 가지고 있는 설정파일인 SQL Map파일의 위치 지정

- resource 속성 : class path 상의 SQL Map 파일

- 여러 파일 지정 가능

① SqlMap - SQL문 설정, Input Data, Output Data를 설정하는 xml 기반 설정 파일

· SqlMap 파일은 SqlMapConfig에 등록한다.

② SqlMapClient interface : SQL문 실행 메소드를 정의

· package : com.ibatis.sqlmap.client

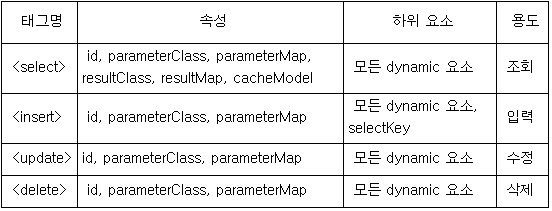
· SqlMapClientBuilder를 통해 얻어 온다.



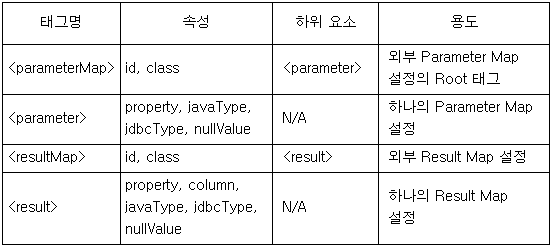
 - Sql Map에 SQL 문을 등록하고 프로그램에서는 SqlMapClient의 메소드를

이용해 등록한 쿼리를 실행 시킨다.

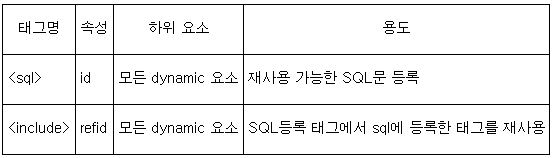
① SQL문 등록 태그



**② 외부 Parameter, Result 매핑 설정**

****

**③ SQL문 생성 태그**

****

**3. Parameter mapping**

**① Sql Map에 쿼리 등록시 세가지 요소 설정 필요**

**- SQL문**

**- Parameter : Input Data**

**- Select문의 경우 Result : Output Data**

**② Parameter Mapping**

**- Parameter : SQL문에 넣을 값**

**- 인 라인 Parameter, 외부 Parameter 매핑 두가지 방식이 있다.**

**· 인라인 - 값과 값을 넣을 위치를 SQL 안에 설정**

**· 외부 parameter mmapping - SQL문에는 ?로 설정하고, 넣을 parameter 연결은**

**SQL문 밖에서 설정**

**- SqlMapClient의 메소드를 통해 받은 값과 SQL 문에 값이 들어갈 곳을**

**Mapping 한다.  
- SQL문 내 mapping 방법**

**참고>**

 #id# 와 $id$의 차이는

JDBC에서는 쿼리를 재사용 할수 있느냐 없느냐의 문제가 될꺼구요. DB에서는 플래닝을 다시할 필요가 있느냐 없느냐의 문제가 될것입니다.

 사용자가 느끼기에는 ##은 변수 타입에 맞춰서 쿼리에 매핑되고 ex>id=#id# 일때 id가 문자열이면 id='xx' , id가 int면 id=xx 로 변경됨

$$는 변수타입과 관계없이 문자열타입 그대로 넣게 됍니다. ex>id=$id$ 일때 id가 문자열이면 id=xx , id가 int면 id=xx 로 변경됨

따라서 sql문으로 데이터를 직접 업데이트 하거나 할때는 $$보다는 ##이 안전합니다.

1) SELECT a.id, a.name FROM 어떤테이블 a WHERE a.id = #id#

와

2) SELECT a.id, a.name FROM 어떤테이블 a WHERE a.id = $id$

를 놓고 설명을 드리자면

1)의 경우 iBatis는 다음과 같은 PreparedStatement를 생성해서 사용하게 됩니다.

SELECT a.id, a.name FROM 어떤테이블 a WHERE a.id = ?

따라서 #id#에 설정되는 값이 바뀌어도 위의 PreparedStatement를 재사용하게 되고,

DB입장에서도 이미 컴파일된 SQL을 사용하게 됩니다.

하지만 2)의 경우 iBatis는 $id$를 마치 상수와 같이 처리해서 PreparedStatement를 생성하게 됩니다.

만약 $id$가 1인 경우는

SELECT a.id, a.name FROM 어떤테이블 a WHERE a.id =1

을 생성합니다.

여기서 $id$값이 바뀌어서 2가 되는 경우는

SELECT a.id, a.name FROM 어떤테이블 a WHERE a.id =2

를 또 PreparedStatement로 생성하게 됩니다.

따라서 $를 사용해서 변수를 바인딩하는 경우 해당 변수의 값이 많지 않은 경우에는

바인드할 변수가 없는 PreparedStatement처럼 사용할 수 있기 때문에 수행 능력을 높일 수 있지만,

게시판의 식별자처럼 계속 바뀌는 변수에 대해 사용하는 경우는 오히려 수행 능력이 떨어집니다.

대부분 테이블 이름과 같이 SQL자체가 바뀌는 경우 혹은 DDL에서 재활용할 부분이 많을 때에 사용합니다.

**· #parameter name# - 인라인 파라미터**

**· $parameter name$ - 인라인 파라미터**

**- parameter name**

**- DTO(VO) : getter와 matching**

**- Map : key와 matching**

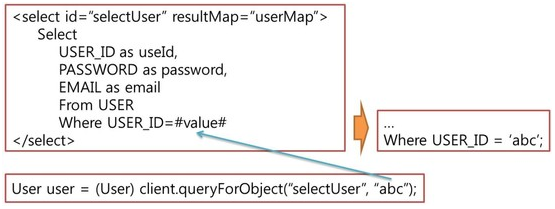
**· ? - 외부 파라미터 매칭**

**③ '#' 지시자로 인라인 parameter 사용**

**- ?로 바꾼 뒤 값을 치환한다. - PreparedStatement 형식**

**- 타입에 맞게 치환 처리 (String의 경우 ''처리)**

**- 값만 치환 가능**

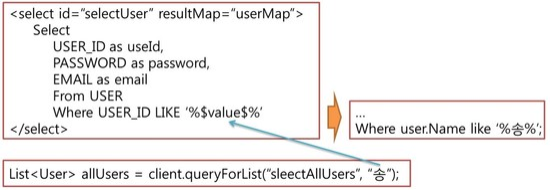


**④ '$' 지시자로 인라인 parameter 사용**

**- 전달받은 값을 바로 치환한다. - copy & paste 개념**

**- String일 경우 ''로 감싸주어야 한다.**

**- 값 뿐만 아니라 키워드 등 쿼리문 무엇이든 치환 가능**



**4. Parameter 매핑 - 외부 매핑**

**① Parameter 매핑의 두가진 방안**

**- 인라인 매핑**

**· SqlMap Mapping문 내부에서 바로 기술함.**

**· 매핑이 복잡한 경우 명시적이지 못함.**

**- 외부 매핑**

**· 외부에서 정의됨**

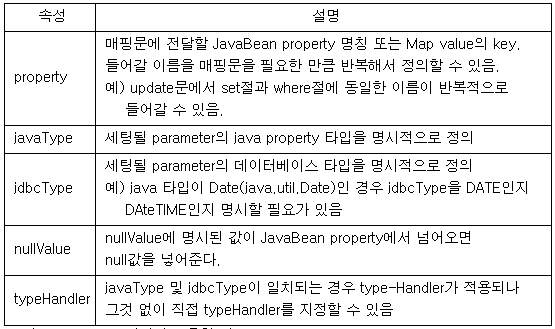
**· 보다 명시적임**

**· 인라인 매핑과 외부 매핑은 같이 사용할 수 없다.**

**- ?와 매칭되어 값이 할당된다.**



② 외부 Parameter 매핑



**- 외부 Parameter 매핑이 유용한 경우**

**· 인라인 Parameter 매핑이 잘 동작하지 않을 경우**

**· 성능을 향상시킬 경우**

**· 명시적 매핑을 할 경우**

**5. Result 매핑**

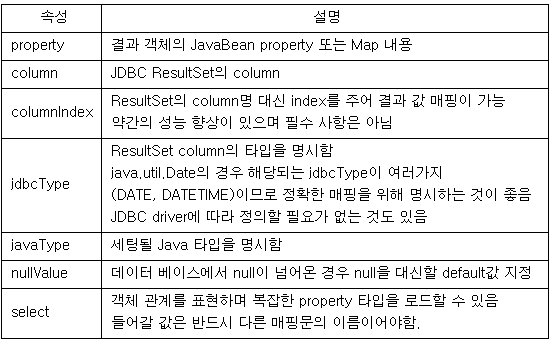
**① 명시적 Result 매핑**

**- <resultMap> : 하나의 resultClass에 대한 설정**

**· 속성 : id, class**

**- <result> : 하나의 property와 resultSet의 Column 매핑**

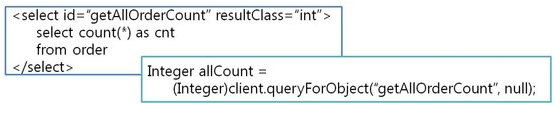
**- result태그**



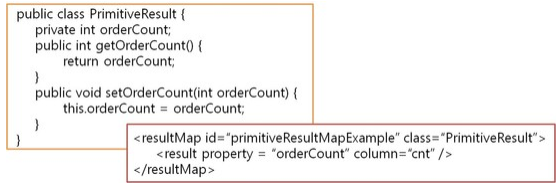
**② Primitive result**

**- iBATIS는 primitive 형태의 결과를 허용하지 않음(int, long, double, ..)**

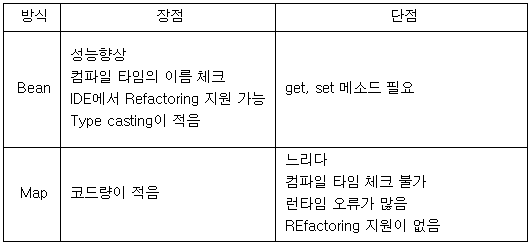
**- Wrapping된 형태의 타입을 리턴(Integer, Long, Double, ..)**



  - JavaBean타입으로 Wrapping할 경우 primitive 형태의 결과를 얻을 수 있음



 ③ JavaBean과 Map Result의 장단점



package ibatis.main;

import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.Map;  
import java.util.Set;

import ibatis.dao.MemberDAO;  
import ibatis.dto.MemberDTO;

public class TestMember {  
 public static void main(String[] args) throws SQLException {  
  MemberDTO mdto1 = new MemberDTO("id-66", "1111", "이상운", "111111", "1111111", 10000);  
  MemberDTO mdto2 = new MemberDTO("id-55", "2222", "장동건", "222222", "2222222", 20000);  
  MemberDTO mdto3 = new MemberDTO("id-77", "3333", "원빈", "333333", "3333333", 30000);  
    
  MemberDAO dao = MemberDAO.getInstance();  
    
 // dao.insertMember(mdto1);  
 // dao.insertMember(mdto2);  
 // dao.insertMember(mdto3);  
    
 // MemberDTO mdto4 = dao.selectMemberById("id-22");  
 // System.out.println(mdto4);  
    
  MemberDTO mdto5 = dao.selectMemberByRegisterNumber("545454", "5454545");  
  System.out.println("주민번호로 조회 결과 : "+mdto5);  
  ArrayList<MemberDTO> mdto6 = dao.selectAllMember();  
  System.out.println("--------------전체 회원 목록 조회------------------");  
  for(int i=0;i<mdto6.size();i++){  
   System.out.println(mdto6.get(i));  
  }  
  System.out.println("---------------전체회원목록조회- map----------");  
  Map map = dao.selectAllMemberMap();  
  System.out.println(map);  
  System.out.println("---------------전체회원목록조회- map(id가 id-77인 회원 찾기)----------");  
  Object obj = map.get("id-77");  
  System.out.println(obj);  
//  Set set = map.keySet();  
//  Iterator itr = set.iterator();  
//  if(itr.next()){  
//     
//  }  
  System.out.println("---------------전체회원목록조회2- map----------");  
  Map map2 = dao.selectAllMemberMap2();  
  System.out.println(map2);  
  System.out.println("---------------전체회원목록조회2- map(id가 id-77인 회원 이름찾기)----------");  
  Object obj2 = map2.get("id-77");  
  System.out.println(obj2);  
 }  
}