http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte2:psl:dataaccess:mybatis_dynamic_sql

# Dynamic SQL

In JDBC API, you'll need to execute a variety of queries. MyBatis meets the demand of users for queries as it supports dynamic SQL language.

Flexibly responding against SQL statement compared to iBatis, MyBatis frees users from demand of learning in making use of dynamic elements.

In MyBatis, the dynamic SQL elements are similar to the text processor based upon either JSTL or XML. Also offered by MyBatis is OGNL-based expression by which you can use dynamic elements much easier than the conventional iBatis.

## How to use basic Dynamic elements

### Sample Dynamic SQL mapper xml

Refer to the following Sql mapper for MyBatis dynamic elements one-o-one. Note that the following Sql mapper involves dynamic enable/disable of where EMP_NO = #{empNo} depending on empNo:

```
        <select id="selectJobHistListUsingDynamicElement"
parameterType="egovframework.rte.psl.dataaccess.vo.JobHistVO"
resultMap="egovframework.rte.psl.dataaccess.vo.JobHistVO">
                <![CDATA[
                        select EMP_NO        as empNo,
                               START_DATE as startDate,
                               END_DATE      as endDate,
                               JOB           as job,
                               SAL           as sal,
                               COMM          as comm,
                               DEPT_NO       as deptNo
                        from   JOBHIST
                ]]>
                <where>
                        <if test="empNo != null">
                                EMP_NO = #{empNo}
                        </if>
                </where>
        </select>
```

When a content is returned by way of tag, you can find the element "WHERE" appearing on the syntax. The element does not appear unless the entirety of sub-elements are satisfied.
Note that "AND" and "OR" at the beginning of the contents are ignored.

## if

"IF" is the most commonly used dynamic element for selective retrieval of string. Used in a WHERE paragraph, an IF statement should comprise value(s) in it to prevent return of the results.

```
        ..
        <select id="selectEmployerList" parameterType="egovframework.rte.psl.dataaccess.vo.EmpVO"
resultType="egovframework.rte.psl.dataaccess.vo.EmpVO">
                <![CDATA[
                        select
                                EMP_NO as empNo,
                                EMP_NAME as empName,
                                        JOB as job,
                                        MGR as mgr,
                                        HIRE_DATE as hireDate,
                                        SAL as sal,
                                        COMM as comm,
                                        DEPT_NO as deptNo
                              from EMP
                    ]]>
                    <where>
                              <if test="empNo != null">
                                        EMP_NO = #{empNo}
                              </if>
                              <if test="empName != null">
                                        EMP_NAME LIKE '%' || #{empName} || '%'
                              </if>
                    </where>
          </select>
```

“IF” is most commonly used to draw comparison among the specific properties in a transferred factor.

## choose (when, otherwise)

Working quite similar to SWITCH statement in Java, The “CHOOSE” element is used when a single condition, among others, should be applied.

Refer to the following coding example to retrieve MGR information and, if any, EMP information. Employee information is returned when no other information is retrievable.

```
          <select id="selectEmployeeList" parameterType="egovframework.rte.psl.dataaccess.vo.EmpVO"
resultType="egovframework.rte.psl.dataaccess.vo.EmpVO">
                    SELECT * FROM EMP WHERE JOB = ‘Engineer’
                    <choose>
                              <when test=”mgr ! null”>
                                        AND MGR like #{mgr}
                              </when>
                              <when test=”empNo ! null and empName ! =null”>
                                        AND EMP_NAME like #{empName}
                              </when>
                              <otherwise>
                                        AND HIRE_STATUS = ‘Y’
                              </otherwise>
                    </choose>
          </select>
```

## trim (where, set)

Meanwhile, refer to the following example for <trim prefix=“WHERE” prefixOverrides=“AND|OR”>, working the same with <where> for trimming:

```
          ..
          <select id="selectEmployerList" parameterType="egovframework.rte.psl.dataaccess.vo.EmpVO"
                    resultType="egovframework.rte.psl.dataaccess.vo.EmpVO">
                    <![CDATA[
                              select
                                        EMP_NO as empNo,
                                        EMP_NAME as empName,
                                        JOB as job,
                                        MGR as mgr,
                                        HIRE_DATE as hireDate,
                                SAL as sal,
                                COMM as comm,
                                DEPT_NO as deptNo
                        from EMP
                ]]>
                <trim prefix="WHERE" prefixOverrides="AND|OR ">
                        <if test="empNo != null">
                                EMP_NO = #{empNo}
                        </if>
                        <if test="empName != null">
                                EMP_NAME LIKE '%' || #{empName} || '%'
                        </if>
                </trim>
        </select>
```

## foreach

Refer to the following example for Sql mapper XML that contains IN statement, the most commonly used method of iterate tagging.

Among other functions that foreach offers, the collection is explicitly notified. While declaring variables Item and Index, the foreach element can contain the strings intended to open and close and delimiters:

```
        <select id="selectJobHistListUsingDynamicNestedIterate"
parameterType="egovframework.rte.psl.dataaccess.util.EgovMap" resultMap="jobHistVO">
                <![CDATA[
                        select EMP_NO       as empNo,
                               START_DATE as startDate,
                               END_DATE    as endDate,
                               JOB           as job,
                               SAL           as sal,
                               COMM          as comm,
                               DEPT_NO     as deptNo
                        from    JOBHIST
                ]]>
                        where
                        <foreach collection="condition" item="item" open="(" separator="and" close=")">
                                ${item.columnName} ${item.columnOperation}
                                <if test="item.nested == 'true'">
                                        <foreach item="item" index="index"
collection="item.columnValue" open="(" separator="," close=")">
                                                '${item}'
                                        </foreach>
                                </if>
                                <if test="item.nested != 'true'">
                                        #{item.columnValue}
                                </if>
                        </foreach>
                        order by EMP_NO, START_DATE
        </select>
```