Inline Parameters

A simple method of processing same functions as the parameterMap element (mapping elements used in configuring options including input object property name, javaType, jdbcType defined outside the SQL statement) is provided as Inline Parameter method to process bind variables mapping on the prepared statement. Inline Parameter method indicating the bind variable area for input object indicated as parameterClass normally with simple #property# notation is the method generally recommended and much used compared to the method of processing as external parameterMap declaration in line with ? in existing parameterMap and the order. It can be used with Dynamic element and relevant property can be directly used at the position requiring bind variable processing without separate external mapping definition. If necessary, jdbcType or nullValue can be designated like simple addition notation(ex. #empName:VARCHAR:blank#). If detailed option is required, required value = value separated by ,(comma) can be described in details like (ex. #comm,javaType=decimal,jdbcType=NUMERIC,nullValue=-99999#).

Using basic Inline Parameters

Refer to below sample Inline Parameters query.

Sample Inline Parameters

```
..
<typeAlias alias="empVO" type="egovframework.rte.psl.dataaccess.vo.EmpVO" />

<insert id="insertEmptUsingInLineParam">
        <![CDATA[
                insert into EMP
                        (EMP_NO,
                         EMP_NAME,
                         JOB,
                         MGR,
                         HIRE_DATE,
                         SAL,
                         COMM,
                         DEPT_NO)
                values   (#empNo:NUMERIC#,
                         #empName:VARCHAR:blank#,
                         #job:VARCHAR:""#,     /* in inline parameter, the empty String cannot be replaced with nullValue - cf.) in case of oracle, "" is null */
                         #mgr:NUMERIC#,
                         #hireDate:DATE#,
                         #sal:NUMERIC#,
                         #comm,javaType=decimal,jdbcType=NUMERIC,nullValue=-99999#,
                         #deptNo:NUMERIC#)
        ]]>
</insert>
```

In above sql mapping file, without external definition of separate parameterMap, it is directly indicated in the area of direct bind variable in the form of #property:jdbcType#. Even if only #property# is indicated, iBATIS can be processed automatically. Since the input object setting is not indicated for parameterClass="empVO" in relevant query, it is confirmed that iBATIS may automatically recognize the input object given as factor in runtime and perform parameter processing. However, indication of parameterClass is recommended due to performance.

- #property#
- #property:jdbcType#
- #property:jdbcType:nullValue# up to : can be indicated with separator and replacement value of nullValue requires indication of jdbcType.

Besides, option setting of property = value type can be indicated as follows with , as separator.

- #propertyName,javaType=?,jdbcType=?,mode=?,nullValue=?,handler=?,numericScale=?#

In the above, **?** area can be set to suitable value depending on relevant property. Mode is the property that can indicate IN/OUT/INOUT mode of stored procedure and numericScale is the property that should be indicated to keep the Scale information of DBMS if the OUT/INOUT variable of stored procedure is decimal or numeric. For handler property, typeHandler can be indicated.

Sample TestCase

```
..
    public EmpVO makeVO() throws ParseException {
        EmpVO vo = new EmpVO();
        vo.setEmpNo(new BigDecimal(9000));
        vo.setEmpName("test Emp");
        vo.setJob("test Job");
        // 7839,'KING','PRESIDENT'
        vo.setMgr(new BigDecimal(7839));
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd", java.util.Locale.getDefault());
        vo.setHireDate(sdf.parse("2009-02-09"));

        // mysql has digit under decimal point as .00 and is declared to numeric(5) for test
convenience.
        if (isMysql) {
            vo.setSal(new BigDecimal("12345"));
            vo.setComm(new BigDecimal(100));
        } else {
            vo.setSal(new BigDecimal("12345.67"));
            vo.setComm(new BigDecimal(100.00));
        }
        // 10,'ACCOUNTING','NEW YORK'
        vo.setDeptNo(new BigDecimal(10));
        return vo;
    }

    public void checkResult(EmpVO vo, EmpVO resultVO) {
        assertNotNull(resultVO);
        assertEquals(vo.getEmpNo(), resultVO.getEmpNo());
        assertEquals(vo.getEmpName(), resultVO.getEmpName());
        assertEquals(vo.getJob(), resultVO.getJob());
        assertEquals(vo.getMgr(), resultVO.getMgr());
        assertEquals(vo.getHireDate(), resultVO.getHireDate());
        assertEquals(vo.getSal(), resultVO.getSal());
        assertEquals(vo.getComm(), resultVO.getComm());
        assertEquals(vo.getDeptNo(), resultVO.getDeptNo());
    }

    @Test
    public void testInLineParameterInsert() throws Exception {
        EmpVO vo = makeVO();

        // insert
        empDAO.insertEmp("insertEmptUsingInLineParam", vo);

        // select
        EmpVO resultVO = empDAO.selectEmp("selectEmp", vo);

        // check
        checkResult(vo, resultVO);
    }

    @Test
```

```
    public void testInLineParameterInsertWithNullValue() throws Exception {
        EmpVO vo = new EmpVO();
        // key setting
        vo.setEmpNo(new BigDecimal(9000));

        // inline parameter nullValue test
        vo.setEmpName("blank");
        // in inline parameter, empty String cannot be replaced with
        // nullValue
        // ref.)
        // http://www.nabble.com/inline-map-format%3A-empty-String-in-nullValue-td18905940.html
        vo.setJob(""); // cf.) in oracle , "" is null
        //!
        vo.setComm(new BigDecimal("-99999"));

        // insert
        empDAO.insertEmp("insertEmptUsingInLineParam", vo);

        // select
        EmpVO resultVO = empDAO.selectEmp("selectEmp", vo);

        // check
        assertNotNull(resultVO);
        assertEquals(vo.getEmpNo(), resultVO.getEmpNo());
        // in inline parameter configuration,
        // #empName:VARCHAR:blank# ..
        // the value is input as null
        //
        assertNull(resultVO.getEmpName());
        // in inline parameter, it is confirmed that the empty String
        // cannot be replaced with the nullValue!
        // cf.) parameterMap case
        // assertNull(resultVO.getJob()) // cf.) oracle
        // "" is null!
        // assertNotNull(resultVO.getJob());
        assertNull(resultVO.getComm());
    }
..
```

The test case processes input/review after setting the input object for insertEmptUsingInLineParam query sentence processing the parameter object binding using Inline Parameters. Actually, above application area is not different from the case of using parameterMap. Above testInLineParameterInsertWithNullValue test method sets the specific value designated as nullValue property in Inline Parameters and inquire the result of entering null in DB, and check it as assertNull. From above, please note that nullValue setting of empty String is not processed properly due to restriction of inline parameter notation.