parameterMap

Generally useful base class for creating MultiValueMaps that store HTTP query parameters. May be subclassed to add specific getter/setter methods for known parameters used in a specific context. Also makes it easy to adapt a Map<String, List<String>> to a MultiValueMap<String, String>. While it is not used much compared to parameterClass or Inline Parameter that processes similar function, it can be good approach if more descriptive parameterMap(for example, for stored procedure) is required or to keep the consistent use or purity of XML. But, it is not recommended generally since it is inconvenient that the order and number of bind variables should be matched correctly, and it cannot be used together with Dynamic element.

How to use basic parameterMap

Refer to the sample below for parameterMap.

Sample parameterMap

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

<parameterMap id="empParam" class="empVO">
        <parameter property="empNo" javaType="decimal" jdbcType="NUMERIC" />
        <parameter property="empName" javaType="string" jdbcType="VARCHAR"
nullValue="blank" />
        <parameter property="job" javaType="string" jdbcType="VARCHAR" nullValue=""
/>
        <parameter property="mgr" javaType="decimal" jdbcType="NUMERIC" />
        <parameter property="hireDate" javaType="date" jdbcType="DATE" />
        <parameter property="sal" javaType="decimal" jdbcType="NUMERIC" />
        <parameter property="comm" javaType="decimal" jdbcType="NUMERIC"
nullValue="-99999" />
        <parameter property="deptNo" javaType="decimal" jdbcType="NUMERIC" />
</parameterMap>

<insert id="insertEmpUsingParameterMap" parameterMap="empParam">
        <![CDATA[
                insert into EMP
                        (EMP_NO,
                         EMP_NAME,
                         JOB,
                         MGR,
                         HIRE_DATE,
                         SAL,
                         COMM,
                         DEPT_NO)
                values      (?,
                             ?,
                             ?,
                             ?,
                             ?,
                             ?,
                             ?,
                             ?)
        ]]>
</insert>
```

From above sql mapping file, as parameterMap element, the id of empParam was given and target input object designates EmpVO. For detailed attributes for EmpVO, mapping definition is performed with lower element of parameter. At this time, it was indicated above for javaType, jdbcType additionally. (it may be advantageous in terms of performance since instruction for direct type is indicated as setting rather than getting the type for individual property of target class using reflection technology of target.) While there is no case that same property is used in duplicate in above,

parameterMap is mapped in the order for ? as shown in the following insertEmpUsingParameterMap mapped statement example, duplicate definition of same property is required in line with order from the parameterMap definition if it should be used in duplicate. In addition, if the relevant value for property designating nullValue is delivered to the value designated in nullValue, property definition for typeName, resultMap, mode, typeHandler, numericScale is possible in the database in addition to null.

Sample TestCase

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"classpath*:META-INF/spring/context-*.xml" })
@TransactionConfiguration(transactionManager = "txManager", defaultRollback = false)
@Transactional
public class ParameterMapTest extends TestBase {

    @Resource(name = "empDAO")
    EmpDAO empDAO;

    @Before
    public void onSetUp() throws Exception {
        // Initialize DB
    }

    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"));
        // in mysql 5.0.X the decimal digits were deleted for 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 testParameterMapInsert() throws Exception {
        EmpVO vo = makeVO();

        // insert
```

```
        empDAO.insertEmp("insertEmpUsingParameterMap", vo);

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

        // check
        checkResult(vo, resultVO);
    }

    @Test
    public void testParameterMapInsertWithNullValue() throws Exception {
        EmpVO vo = new EmpVO();
        // key setting
        vo.setEmpNo(new BigDecimal(9000));

        // parameterMap nullValue test
        vo.setEmpName("blank");
        vo.setJob("");
        // cf.) -99999.99 is found to be
        // NumberFormatException!
        vo.setComm(new BigDecimal("-99999"));

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

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

        // check
        assertNotNull(resultVO);
        assertEquals(vo.getEmpNo(), resultVO.getEmpNo());
        // in parameterMap configuration nullValue="blank" ..
        // the value was input as null
        assertNull(resultVO.getEmpName());
        assertNull(resultVO.getJob());
        assertNull(resultVO.getComm());
    }

}
```

It is the test case processing input/inquiry after setting the input object for insertEmpUsingParameterMap query sentence processing the parameter object binding using parameterMap as above. Actually, in case of using parameterMap in actual mapping file and the case of using Inline parameter or parameterClass as it is, the application area calls iBATIS API with input object of same type as a factor. Above testParameterMapInsertWithNullValue test method inquires the results entered as null in DB by setting the specific value designated as nullValue property in parameterMap.