Data Access Service

Summary

The Data Access Service is to react various kinds of database solution and database approach technology in a consistent way, simplifying the mechanism of performing the function to view, enter, modify and delete the data. In addition, if the database solution or approach technology change, it abstracts the contact with the database so as to minimize changes of the system area that deals with the data and provides the abstracted data approach method in template, resulting in improvement of task efficiency of developers.

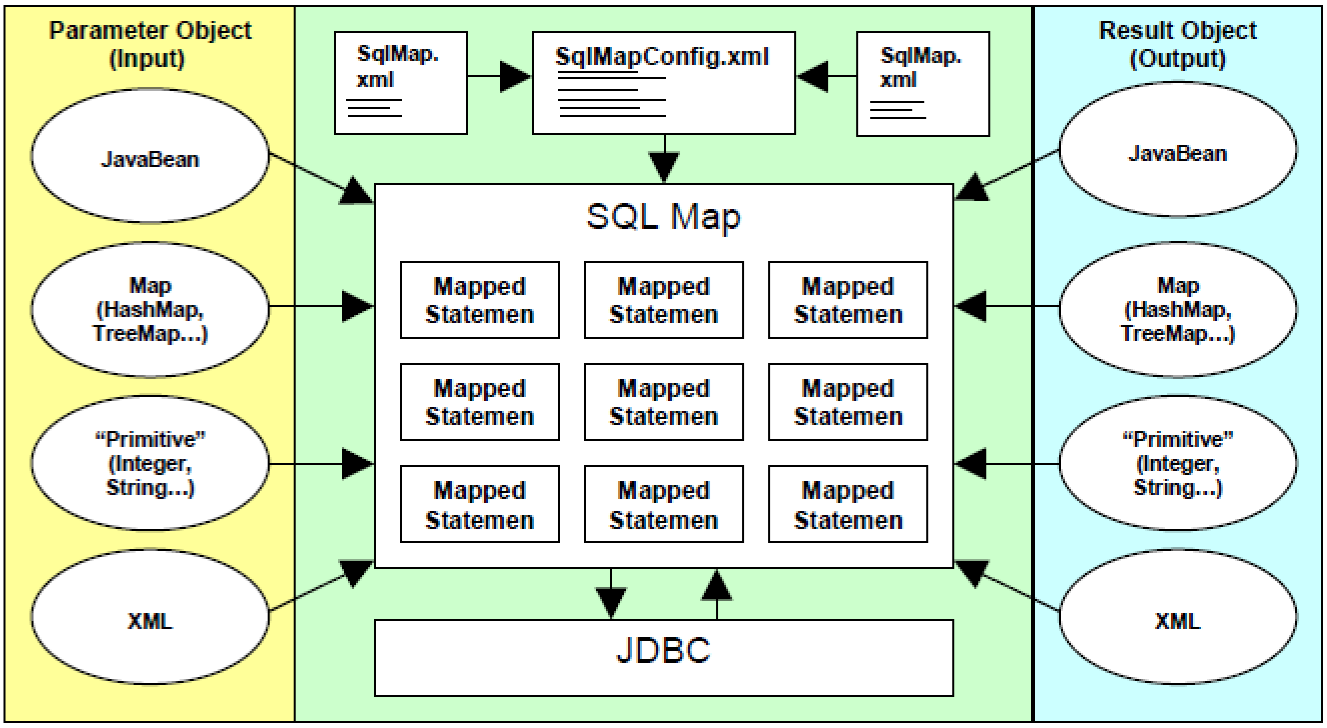
iBATIS Framework

The e-government framework adopts iBATIS, the Data Mapper framework that is convenient and easy to use, as the foundation open source of Data Access function by abstracting the Data Access using JDBC. Using iBATIScan significantly reduce the use of series of Java code required to access the relational database and can simply map the SQL sentence to JavaBeans (or Map) using simple XML technology.

- Provide abstracted approach method: as an abstracted approach method for JDBC data access, it supports easy and convenient API, resource connection/termination and common error processing in integrated way.
- Support SQL separation from code: It separates SQL sentence from source code, keeps separate repository(XML of meaningful grammar), implements reference structure for this internally and guarantees the easiness of management/maintenance/tuning.
- Support input/output object binding/mapping of query execution: It supports automation of object (VO, Map and List) level at the time of processing (mapping) process of resultset as a result of execution and binding for input parameter of query sentence.
- Support Dynamic SQL: It supports change of dynamic query sentence depending on input conditions without direct use of API and code creation.
- Provide various DB processing: It supports various DB processing including Batch SQL, Paging, Callable Statement and BLOB/CLOB in addition to basic query.

Description of Details

iBATIS Data Mapper API supports easy and simple description of object mapping for SQL sentence using XML and easily maps Javabeans object, Map implement and various primitive wrapper types (String, Integer..) as the result object for ResultSet or parameter ofPreparedStatement.



1. It provides parameter object. (Similarly, Javabeans, Map or primitive wrapper) The parameter object will be set to input variable of where sentence of query or update sentence.
2. It executes mapped statement. Data Mapper framework creates instances of PreparedStatementand sets the parameter using the parameter object provided above(bind variable processing), executes query sentence and creates result object from ResultSet.
3. In case of update sentence, it returns the number of rows reflecting changes, and returns Collection (object list) for inquiry of several cases or single object for inquiry of single case in case of inquiry. Same as the parameter object, it can be JavaBean of result object, or Map, primitive type wrapper or XML document.

Description

Explain what is required to start Data Access Service simply before explaining Data Access Service in details.

Step1. Preparation

Required Library

Following is the description and library list required to utilize this service.

Library	Description	Associated Library
ibatis-sqlmap-2.3.4.726.jar	iBATIS Library (required)	
commons-dbcp-1.2.2.jar	database connection pooling support library(optional)	
commons-logging-1.1.1.jar	commons logging(optional)	
log4j-1.3alpha-8.jar	log4j(optional)	
oscache-2.4.jar	Central intensive or distributed caching support(optional)	
cglib-nodep-2.1_3.jar	If Runtime Bytecode Enhancing required(optional)	

Only ibatis-sqlmap-2.3.4.726.jar is required library. However, the library for logging process and connection coupling library like commons-dbcpis required. As the improved function supported by iBATIS additionally, the above reference library can be additionally set if you want to write the function related to Runtime Bytecode Enhancement or cache supports. In addition, since we prefer development of application in the form of Spring-iBATISinterlock, Springrelated library and dependency library for this will be included generally. In addition, additional library is required for the suitable jdbc driver depending on DBMS(Oracle, Mysql, Hsqldb, Tibero, etc.), the target of actual Data Access processing.

Step2. sql-map-config.xml Setting and Basic Spring Setting

To use by linking iBATIS in Spring framework-based application, setting is required for SqlMapClientFactoryBean of Spring. Here, actual target sql-map-config setting file and setting for dataSource to be provided to iBATIS will be instructed.

```
	<!--SqlMap setup for iBATIS Database Layer -->
	<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
		<property name="configLocation" value="classpath:/META-INF/sqlmap/sql-map-
config.xml"/>
		<property name="dataSource" ref="dataSource"/>
	</bean>
```

- Using Spring interlock function will help obtain SqlMapClient(a thread safe client for SQL Maps) of iBATISwithout separate iBATIS API.

Following is the SQL Map XML Configuration file (sql-map-config.xml setting file) of main iBATIS.When iBATISis used separately, it is required to include transactionManager, dataSourcesetting additionally, but in the Spring interlink environment, this area automatically uses dataSourcethat the Spring hands over. Transaction management is set declaratively in the business service area so that it is not required to be concerned about in iBATISrelated module.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPEsqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config 2.0//EN"
    "http://www.ibatis.com/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
	<settings useStatementNamespaces="false"
			..
	/>

	<typeHandlerjavaType="java.util.Calendar" jdbcType="TIMESTAMP"
		callback="egovframework.rte.psl.dataaccess.typehandler.CalendarTypeHandler" />

	<sqlMap resource="META-INF/sqlmap/mappings/testcase-basic.xml" />
	<sqlMap ../>
	..
</sqlMapConfig>
```

- sqlMapConfig: root tag of iBATISsetting file
- Settings: tag that can direct various option settings (ex. Whether to use cacheModel, whether to use Runtime Bytecode Enhance, option setting including whether to use Namespace for query sentence. cf. setting related to transaction/dataSourceconnection is not required in Springinterlock environment.)
- typeHandler: if type conversion processing between javaType↔jdbcTypeis separately required, implement typeHandlerand register this in sql-map-config.
- typeAlias: can designate typeAliasto use globally (simple alias for class full package name).
- sqlMap: register each SQL Mapping XML file. Load the relevant resources from classpathorurl in the stream form. Resource property looks at classpathpath by default.

In case of Spring 2.5.5 or over andiBATIS 2.3.2 or over, it can be designated by batch as a pattern expression formula for the Sql mapping file as the mappingLocations property when defining SqlMapClientFactoryBean for iBATISinterlock. An example of using mappingLocationsproperty is as follows:

```
	<!--SqlMap setup for iBATIS Database Layer -->
	<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
		<property name="configLocation" value="classpath:/META-INF/sqlmap/sql-map-
config.xml"/>
		<!-- Java 1.5 or higher and iBATIS 2.3.2 or higher REQUIRED -->
		<property name="mappingLocations" value="classpath:/META-
INF/sqlmap/mappings/**/*.xml" />
		<property name="dataSource" ref="dataSource"/>
	</bean>
```

In this case, "configLocation" property is not required, but if there is no relevant property at present, SqlMapClientFactoryBean is not initialized so that "configLocation" property should be maintained. At this time, relevant sql-map-config.xml should be processed to have dummy.xml query as shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPEsqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config 2.0//EN"
    "http://www.ibatis.com/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
	<sqlMap resource="sqlmap/sql/common/dummy.xml"/>
</sqlMapConfig>
```

dummy.xml is processed as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPEsqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN" "http://www.ibatis.com/dtd/sql-
map-2.dtd">

<sqlMap namespace="Dummy">
</sqlMap>
```

Step3. sql mapping xml Setting

Various option setting and Mapped statement definition are created in SQL Map document structure defined in iBATIS. Following is the simple mapping file including queries related to CRUD for department information and In/Out object mapping for this.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPEsqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN" "http://www.ibatis.com/dtd/sql-
map-2.dtd">

<sqlMap namespace="Dept">

	<typeAlias alias="deptVO" type="egovframework.DeptVO" />

	<resultMap id="deptResult" class="deptVO">
		<result property="deptNo" column="DEPT_NO" />
		<result property="deptName" column="DEPT_NAME" />
		<result property="loc" column="LOC" />
	</resultMap>

	<insert id="insertDept" parameterClass="deptVO">
		insert into DEPT
			(DEPT_NO,
			 DEPT_NAME,
			 LOC)
		values	(#deptNo#,
			 #deptName#,
			 #loc#)
	</insert>

	<select id="selectDept" parameterClass="deptVO" resultMap="deptResult">
		<![CDATA[
                    select DEPT_NO,
                           DEPT_NAME,
                           LOC
                    from    DEPT
                    where   DEPT_NO = #deptNo#
          ]]>
    </select>

    <update id="updateDept" parameterClass="deptVO">
          update DEPT
          set      DEPT_NAME = #deptName#,
                   LOC = #loc#
          where   DEPT_NO = #deptNo#
    </update>

    <delete id="deleteDept" parameterClass="deptVO">
          delete from DEPT
          where          DEPT_NO = #deptNo#
    </delete>

    <select id="selectDeptList" parameterClass="deptVO" resultMap="deptResult">
          <![CDATA[
                    select DEPT_NO,
                           DEPT_NAME,
                           LOC
                    from    DEPT
                    where   1 = 1
          ]]>
          <isNotNull prepend="and" property="deptNo">
                    DEPT_NO = #deptNo#
          </isNotNull>
          <isNotNull prepend="and" property="deptName">
                    DEPT_NAME LIKE '%' || #deptName# || '%'
          </isNotNull>
    </select>

</sqlMap>
```

- typeAlias: designates simple alias name for the object in current mapping file.
- resultMap: creates the mapping for column name of DB and Attribute name of object. Various additional option can be designated including javaType, jdbcType, columnIndex andtypeHandler.
- insert: Mapped Statement definition tag for input query (insert sentence)
- select: Mapped Statement definition tag for inquiry query
- update: Mapped Statement definition tag for modification query (update sentence)
- delete: Mapped Statement definition tag for deletion query (delete sentence)

The above CRUD related mapping file uses JavaBeans object, "DeptVO" as Parameter/Result object by default. It is simply designated and used as typeAlias. In this case, it is mainly designated as parameterClassto indicate the declarative use for the parameter object. During actual bind variable processing, an Inline Parameter type was used as shown in #attribute name#. In addition, define mapping per field of result object (DeptVO) for result column information according to ResultSet through resultMapdefinition. After that, declare this in resultMapproperty of a select sentence, and process the results of select through resultMap. In addition to this type, Mapped Statement processing can be defined, but as shown above, it is recommended to use JavaBeans object and to define resultMapfor result object processing, and to use as the style of processing bind variable in Inline Parameter method.

Step4. Create DAO Class

Create simple type of DAO class. EgovAbstractDAO, inheriting the following DAO, extends SqlMapClientDaoSupport. Injection processing for SqlMapClient, the basic class for iBATIS SQL Map interaction(in above, provided by iBATIS linkage FactoryBean defined as "sqlMapClient") is applied internally and simple method wrapping is provided for iBATIS execution on basic CRUD. To use detailed API, (ex. getSqlMapClientTemplate().queryWithRowHandler(“selectEmpListToOutFileUsingRowHandler”, paramObject, rowHandler);) can be used through getSqlMapClientTemplate()

DAO Class

```
..

@Repository("deptDAO")
public class DeptDAO extends EgovAbstractDAO {

public void insertDept(DeptVOvo) {
insert("insertDept", vo);
    }

publicintupdateDept(DeptVOvo) {
return update("updateDept", vo);
    }

publicintdeleteDept(DeptVOvo) {
return delete("deleteDept", vo);
    }

publicDeptVOselectDept(DeptVOvo) {
return (DeptVO)selectByPk("selectDept", vo);
    }

@SuppressWarnings("unchecked")
public List<DeptVO>selectDeptList(DeptVOsearchVO) {
return list("selectDeptList", searchVO);
    }
}
```

- @Repository: Definition of Spring Bean using @Repository Stereo Type Annotation for DAO

Each CRUD related method executes Mapped Statement of iBATIS with queryIdand parameter object (DeptVO here) as a factor. For inquiry, it returns DeptVO object as a single case inquiry, and list for DeptVO in case of list inquiry. It is not processed as Generics (processing of collection with defined type) of java 1.5 or over internally iBATIS, but since actual data may be processed to List<DeptVO> depending on sql mapping file, @SuppressWarnings(“unchecked”) is indicated to minimize unnecessary Type Casting in the module before calling. If setting useStatementNamespaces=“true” with setting option of sql-map-config.xml, note that the above queryIdshould contain Namespace prefix designated at sql mapping file as shown in “Dept.insertDept”

Step5. Create Test Class

Organize in the JUnitTestCase format(JUnit 4 style) for simple entering and viewing process, using the setting file and DAO defined above.

```
..

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

@Resource(name = "dataSource")
DataSourcedataSource;

@Resource(name = "deptDAO")
DeptDAOdeptDAO;
```

```
    @Before
public void onSetUp() throws Exception {
        // Initialize DB from sql file outside(delete/create existing table)
SimpleJdbcTestUtils.executeSqlScript(
newSimpleJdbcTemplate(dataSource), new ClassPathResource(
                "META-INF/testdata/sample_schema_ddl_hsql.sql"), true);
    }

publicDeptVOmakeVO() {
DeptVOvo = new DeptVO();
vo.setDeptNo(new BigDecimal(90));
vo.setDeptName("test department");
vo.setLoc("test location");
returnvo;
    }

public void checkResult(DeptVOvo, DeptVOresultVO) {
assertNotNull(resultVO);
assertEquals(vo.getDeptNo(), resultVO.getDeptNo());
assertEquals(vo.getDeptName(), resultVO.getDeptName());
assertEquals(vo.getLoc(), resultVO.getLoc());
    }

    @Test
public void testBasicInsert() throws Exception {
DeptVOvo = makeVO();

        // insert
deptDAO.insertDept(vo);

        // select
DeptVOresultVO = deptDAO.selectDept(vo);

        // check
checkResult(vo, resultVO);
    }

    @Test
public void testBasicUpdate() throws Exception {
DeptVOvo = makeVO();

        // insert
deptDAO.insertDept(vo);

        // data change
vo.setDeptName("updDept");
vo.setLoc("updloc");

        // update
inteffectedRows = deptDAO.updateDept(vo);
assertEquals(1, effectedRows);

        // select
DeptVOresultVO = deptDAO.selectDept(vo);

        // check
checkResult(vo, resultVO);
    }

    @Test
public void testBasicDelete() throws Exception {
DeptVOvo = makeVO();
```

```
          // insert
deptDAO.insertDept(vo);

          // delete
inteffectedRows = deptDAO.deleteDept(vo);
assertEquals(1, effectedRows);

          // select
DeptVOresultVO = deptDAO.selectDept(vo);

          // it should be null
assertNull(resultVO);
     }

     @Test
public void testBasicSelectList() throws Exception {
DeptVOvo = makeVO();

          // insert
deptDAO.insertDept(vo);

          // key setting with search condition
DeptVOsearchVO = new DeptVO();
searchVO.setDeptNo(new BigDecimal(90));

          // selectList
          List<DeptVO>resultList = deptDAO.selectDeptList(searchVO);

          // results for key condition will be one case
assertNotNull(resultList);
assertTrue(resultList.size() > 0);
assertEquals(1, resultList.size());
checkResult(vo, resultList.get(0));

          // set name with search condition - '%' || #deptName# || '%'
DeptVO searchVO2 = new DeptVO();
searchVO2.setDeptName(""); // '%' || '' || '%' --> '%%'

          // selectList
          List<DeptVO> resultList2 = deptDAO.selectDeptList(searchVO2);

          // result for like condition is over 1 case
assertNotNull(resultList2);
assertTrue(resultList2.size() > 0);

     }
}
```

By default, it consists of Spring-based applications applied with the Dependency Injection and creation of Annotation form Bean. Spring Bean such asdataSource, transactionManageris used together. Note that test case is JUnit 4 format and designed to obtain transactionManager anddataSource(see the case of using SimpleJdbcTemplatefor DB initialization), as well as Spring setting file loading. For test convenience, before each test method, method defined as @Before deletes and recreates the existing table; separate method of makeVO separate the VO preparation area for test; and the method called checkResult separates and reuses asset comparison logic for inquiry result resultVO and original VO. Each method instructed with @Test is the test method, and form the basic CRUD test logic for DeptDAO as verification for flow of input-inquiry-result check, input-change-inquiry-result check, input-delete-inquiry-null check, search condition setting-inquiry-result check.

Step6. Execute

1. Download the file and unzip.

2. Select the folder unzipped at eclipse and import the project.
3. Check whether there is DeptVO.java, DeptDAO.java, BasicDataAccessTest.java, Spring, iBATIS setting file and Sql mapping file, initialization script file for test data and log4j.xml normally in src folder of the project.
4. Check whether there is a library file in lib.
5. Select BasicDataAccessTest.java to right-click to Run As >JUnit Test.
6. Check whether it is normally executed in JUnit result window.

※Relevant project uses Hsqldb (currently memory operation type), but if you want to use other DBMS, add related connection information injdbc.properties, add and test the JDBC driver jar file to the library.

Reference

- http://ibatis.apache.org
- iBATIS-SqlMaps-2 Developer Guide
- iBATIS-SqlMaps-2 Developer Guide(Translation of Lee Dong Guk)
- Spring Framework - Reference Documentation