iBATIS Configuration

This chapter will examine writing the main configuration file of iBATIS, the SQL Map XML Configuration file (sql-map-config.xml configuration file) and detailed option configurations.

sql-map-config.xml

It is the main configuration file that controls details related to SqlMapClient configuration which includes transaction management configuration, various option configuration, and path configuration on SqlMapping.

Sample Configuration

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

<sqlMapConfig>

        <properties resource="META-INF/spring/jdbc.properties" />

        <settings cacheModelsEnabled="true" enhancementEnabled="true"
                lazyLoadingEnabled="true" maxRequests="128" maxSessions="10"
                maxTransactions="5" useStatementNamespaces="false"
                defaultStatementTimeout="1" />

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

        <transactionManager type="JDBC">
                <dataSource type="DBCP">
                        <property name="driverClassName" value="${driver}" />
                        <property name="url" value="${dburl}" />
                        <property name="username" value="${username}" />
                        <property name="password" value="${password}" />
                        <!-- OPTIONAL PROPERTIES BELOW -->
                        <property name="maxActive" value="10" />
                        <property name="maxIdle" value="5" />
                        <property name="maxWait" value="60000" />
                        <!-- validation query -->
                        <!--<property name="validationQuery" value="select * from DUAL" />-->
                        <property name="logAbandoned" value="false" />
                        <property name="removeAbandoned" value="false" />
                        <property name="removeAbandonedTimeout" value="50000" />
                        <property name="Driver.DriverSpecificProperty" value="SomeValue" />
                </dataSource>
        </transactionManager>

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

- properties: support the connection for standard java properties (key=value form) file and within the setting file in the form of ${key}, refer to the actual value externalized in the form of properties (here, DB connection related driver, url, id/pw). Can designate the support in effective URL in resource property and classpath, url property.
- settings: support to optimize through various option settings for SqlMapClient instance crated through this setting file. All properties are optional.

Property	Description	Example, Default
maxRequests	Designate the maximum number of threads that can execute SQL sentence within the same period of time.	maxRequests="256", 512
maxSessions	Designate the number of session(or client) that can be activated within given time.	maxSessions="64", 128
maxTransactions	Designate the maximum number that can fit in SqlMapClient.startTransaction() within same period of time.	maxTransactions="16", 32
cacheModelsEnabled	Designate whether to use SqlMapClient for all cacheModel globally.	cacheModelsEnabled="true", true (enabled)
lazyLoadingEnabled	Designate whether to use all lazy loading for SqlMapClient globally.	lazyLoadingEnabled="true", true (enabled)
enhancementEnabled	Designate whether to use runtime bytecode enhancement technology.	enhancementEnabled="true", false (disabled)
useStatementNamespaces	Designate whether to use namespace combination if referring to mapped statements. If it is true, use same as queryForObject("sqlMapName.statementName");.	useStatementNamespaces="false", false (disabled)
defaultStatementTimeout	Designate the timeout seconds for all JDBC queries. Can override with setting of each statement. Please note that all drivers do not support this setting.	No timeout, if not designated(cf. according to each statement setting)
classInfoCacheEnabled	Set whether to keep cache of introspected(internally referred to by reflection API of java) class	classInfoCacheEnabled="true", true (enabled)
statementCachingEnabled	Setting on whether to keep local cache of prepared statement	statementCachingEnabled="true", true (enabled)

- typeHandler: can register typeHandler implements that processes conversion between javaType ↔ jdbcType (get parameter setting of prepared statement /resultSet value).
- transactionManager: can set the transaction management service. Can designate which transaction manager to use as type property. 3 types of transaction manager of JDBC, JTA, EXTERNAL are included in framework. In above, it was set to JDBC type that manages transaction through general Connection commit()/rollback() method.
- dataSource: setting for DataSource as the certain area of transactionManager setting. Can designate which DataSourceFactory to use as a type property. Up to 3 types such as SIMPLE, DBCP, JNDI can be set. In above, it was set to DBCP type that uses Apache Commons DBCP(Database Connection Pool). iBATIS supports to directly indicate the setting for DBCP property. After iBATIS 2 version, only single dataSource is supported.
- sqlMap: set to include SQL Map XML file explicitly. Load the resources in classpath (designate as resource property) or url(designate as url property) in the form of stream. In above, designate the sql mapping file that exists in classpath.

Besides, typeAlias(global type alias- simply compared to full package name), resultObjectFactory (support to process the creation of result object through factory class implementing ResultObjectFactory interface of Ibatis by SQL query execution). sqlMap setting in DTD is required over 1 and other setting is optional.

SQL Map XML File (sql mapping file)

The sql mapping file is the file that externalize various option configuration, mapping definition, sql documents according to SQL Map document structure to process as the mapped statement format of iBATIS.

Sample SQL Map XML

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

```
<!DOCTYPE sqlMap 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>

</sqlMap>
```

- typeAlias: designate simple alias name for objects in current mapping file. (cf. in case of class frequently used, it is better to register in sql-map-config.xml globally)
- resultMap: define additional option and mapping for attribute of result object and DB column name (select query column alias).
- insert, select: Example of mapped statement definition element according to each statement type. Can use insert/update/delete/select/procedure/statement element according to type.

Besides, detailed definition for parameterMap, resultMap, cacheModel setting, sql element setting for reuse of sql sentence. See related guide for details.

Mapped Statement

Helper class that simplifies data access via the MappedStatement API of iBATIS SQL Maps, and converts checked SQLExceptions into unchecked DataAccessExceptions, following the org.springframework.dao exception hierarchy. Uses the same SQLExceptionTranslator mechanism as JdbcTemplate. Simply, class for parameter or result can be directly set (it is not recommended, but can process auto mapping processing provided by framework), in/out mapping and keeping cache of results.

- statement sentence

```
<statement id="statementName"
		[parameterClass="some.class.Name"]
		[resultClass="some.class.Name"]
		[parameterMap="nameOfParameterMap"]
		[resultMap="nameOfResultMap"]
```

```
        [cacheModel="nameOfCache"]
        [timeout="5"]>
    select * from PRODUCT where PRD_ID = [?|#propertyName#]
    order by [$simpleDynamic$]
</statement>
```

- insert, update, delete, select, procedure may appear according to type of sql sentence in above statement element position. In above, the property in square bracket[] is optional.
- As parameterClass or resultClass property, can directly designate the object for In/Out. It can be standard JavaBeans object or Map (indicate map implements in case of result) and designate the single variable as primitive rapper class.
- If separately designating the mapping definition for bind variable mapping for prepared statement, designate the relevant id with above parameterMap property. If designating parameterMap, please note that the bind variable area of sql sentence is created with? and the order and number of parameterMap mapping setting should be matched.
- It is preferred to use Inline Parameter over parameterMap and can be simply described in the form of #property name#.
- If designating as separate resultMap tag for result object mapping for resultSet, designate the relevant id with above resultMap property. Use of resultMap is recommended in terms of performance and application functionality of detail option.
- If performing related setting as separate cacheModel tag for caching the results once inquired, indicate the relevant cacheModel id in target statement like above cacheModel property. Note flush related setting of cacheModel setting to update the cache at the time of DB data change.
- If use DBMS and JDBC driver are supported, timeout can be indicated. It is preferred over defaultStatementTimeout of sql-map-config.xml.
- $Variable name$ written in above order by sentence replaces the string delivered to the value for relevant variable in input variable with replaced Text processing by dynamic change elements of SQL. It can be used for dynamic change of overall sql sentence or form section or order by section that can be processed as a bind variable, but there is a security risk of SQL Injection and if the table or column is changed, it may not agree with cache contents of resultSet metadata for the result object automatically mapped internally, resulting in error.