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

Spring-iBATIS Integration

Comprising iBATIS SQL Map as its part, Spring Framework offers template-style programming like JDBC/Hibernate, making exception class processing feasible in Spring-iBATIS environment. Contrary to the iBATIS-only environment demanding transaction managements and DataSource configuration and management, Spring-iBATIS environment takes the essence of Spring's flexible transaction processing and dataSource.

How to configure Spring's SqlMapClientFactoryBean

Intended to generate SqlMapClient for iBATIS, being an implementor for FactoryBean, SqlMapClientFactoryBean sets up SqlMapClient of iBATIS in the context of Spring. When obtained, SqlMapClient is transferred to the iBATIS-based DAO by way of dependency injection.

Sample Configuration

```
<!-- dataSource Configuration -->
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="${driver}"/>
        <property name="url" value="${dburl}"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
        <property name="defaultAutoCommit" value="false"/>
        <property name="poolPreparedStatements" value="true"/>
</bean>

<!-- 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>
```

- dataSource : Turns the database connection abstract. Note the foregoing example has assumed use of Used Apache Commons DBCP. Configuration related to accessing DB has been established by property-placeholder being the external source.
- sqlMapClient : A SqlMapClientFactoryBean configuration intended for Spring-iBATIS. This generates SqlMapClient instance for sql-map-config.xml, being the main configuration file for iBATIS, for use in Spring. SqlMapClient also instructs the injection for transfer of dataSource of Spring to iBATIS, by which iBATIS take the essence of Spring, being flexible data sourcing and transaction with no further configuration (in Spring, iBATIS-based DAO call follows the declarative transaction).

For the current version of Spring, configLocations property is made available to support the pattern expression and multiple interworkings (a runtime is merged into a single integrated configuration). You can wisely configure the attribute "useTransactionAwareDataSource" to make transaction-aware DataSource available (a default configuration) to have SqlMapClient apply transaction timeout governed by Spring and to configure lobHandler for Spring via the corresponding property.

mappingLocations Supported

Another notable improvement in iBATIS is that it takes advantage of the abstract resources of Spring, in the configuration file for SqlMapClientFactoryBean, contrary to the conventional configuration where the attribute "mappingLocations" should be mapped to the sql mapping file by working the tag "sqlMap". The sql mapping files are then merged into the configuration file "sql-map-config" and runtime. Note that mappingLocations is supported in **Spring 2.5.5 and iBATIS 2.3.2 or newer**.

```
<!-- 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="mappingLocations"
                        value="classpath:/META-INF/sqlmap/mappings/testcase-*.xml" />
                <property name="dataSource" ref="dataSource" />
        </bean>
```

Keep in mind that you are advised to prepare sql-map-config.xml designating the dummy sql mapping file, as follows, as DTD(http://www.ibatis.com/dtd/sql-map-config-2.dtd) of sql-map-config.xml should contain at least one sqlMap tag, regardless of the integrated sql mapping file.

How to Change iBATIS Configurations

- sql-map-config.xml devoid of sqlMap Configuration (at least one dummy needs configuration)

```
<!DOCTYPE sqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config 2.0//EN"
    "http://www.ibatis.com/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
        <settings useStatementNamespaces="false"
                                defaultStatementTimeout="10"
        />

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

        <!-- For Spring 2.5.5 and iBATIS 2.3.2 or newer, you can have the Sql mapping file integrated
                , working the attribute "mappingLocations", when intending to define  SqlMapClientFactoryBean
for Spring-iBATIS interwork.
            (When attempting to configure "sqlMapClient" bean, keep in mind
                    mappingLocations="classpath:/META-INF/sqlmap/mappings/testcase-*.xml")
            Provided, however, that a dummy mapping file is configured to satisfy the sqlMap factor requirement
              as follows:
        -->
        <sqlMap resource="META-INF/sqlmap/mappings/testcase-dummy.xml" />

</sqlMapConfig>
```

- testcase-dummy.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="Dummy"/>
```

iBATIS-based DAO

Being a superclass for SqlMapClient data access object, the class "SqlMapClientDaoSupport" provides the extensive sub-classes with SqlMapClientTemplate, a helper class intended to simplify the data accessing via iBATIS that effectively converts SQLException into the unchecked DataAccessException to fit the exception Hierarchy of Spring DAO and makes SQLExceptionTranslator, featuring the same processing structure with JdbcTemplate of Spring, available. SqlMapClientDaoSupport also provides the execution methods for SqlMapExecutor of iBATIS with the mirror methods for use in querying, insertion, updating and/or delete processing. Keep in mind, however, that you need to explicitly implement SqlMapClientCallback of Spring for such complicated execution as batch update (most commonly prepared in the form of anonymous inner class).

```
public class SqlMapAccountDao extends SqlMapClientDaoSupport implements AccountDao {

    public Account getAccount(String email) throws DataAccessException {
        return (Account) getSqlMapClientTemplate().queryForObject("getAccountByEmail", email);
```

```
    }

    public void insertAccount(Account account) throws DataAccessException {
        getSqlMapClientTemplate().update("insertAccount", account);
    }
}
```

Extending SqlMapClientDaoSupport, iBATIS-based DAO obtains SqlMapClientTemplate via getSqlMapClientTemplate() to wrap the data access processing for iBATIS prior to the implementation thereof.

```
<bean id="accountDao" class="example.SqlMapAccountDao">
  <property name="sqlMapClient" ref="sqlMapClient"/>
</bean>
```

You are also need to inject the bean "sqlMapClient" into iBATIS-based DAO.

While you may have a hard time configuring DAO injection into sqlMapClient when attempting to generate beans and proceed with dependency using annotation, the eGovFramework offers extensive EgovAbstractDAO to help you out.