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

MyBatis Application Guideline

Outline

The Section describes how to apply MyBatis to eGovFramework.

ex-dataaccess-mybatis.zip

Step 1. Changes in pom.xml

In eGovFramework, dataaccess artifact version should be changed into 2.7.0, as follows:

```
<!-- Runtime Environment Library -->
<dependency>
        <groupId>egovframework.rte</groupId>
        <artifactId>egovframework.rte.psl.dataaccess</artifactId>
        <version>2.7.0</version>
</dependency>
```

Step 2. Settings

Step 2.1 Setting XML

Add sqlSession bean to the Spring XML configuration (e.g. Context-mapper.xml) as follows:

Ex) context-mapper.xml

```
<!-- SqlSession setup for MyBatis Database Layer -->
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="mapperLocations" value="classpath:/sqlmap/mappers/**/*.xml" />
</bean>
```

Step 2.2 Preparing Mapper XML

You can prepare query XML as per MyBatis guideline, as follows (at mapperLocations in Step 2.1).

Step 3. Preparing DAO

You have a total of three options to prepare DAO:

Method	Description	Remark
Conventional DAO Class	Designate @Repository. Use EgovAbstractMapper extends	Conventional iBatis
Mapper Interfacing	Prepare mapper interface and designate @Mapper annotation	@Mapper being Marker Annotation(Standard Framework provided)
Annotation Method	Methods like @Select and @Insert are used on the Mapper Interface, without Query XML.	Use of Dynamic SQL restricted.

3.1 Conventional DAO Class

As the class with @Repository extends EgovAbstractMapper, the Methods insert, update, delete, selectByPk and list are used.

```
@Repository("deptMapper")
public class DeptMapper extends EgovAbstractMapper {

	public void insertDept(String queryId, DeptVO vo) {
		insert(queryId, vo);
	}

	public int updateDept(String queryId, DeptVO vo) {
		return update(queryId, vo);
	}

	public int deleteDept(String queryId, DeptVO vo) {
		return delete(queryId, vo);
	}

	public DeptVO selectDept(String queryId, DeptVO vo) {
		return (DeptVO)selectByPk(queryId, vo);
	}

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

3.2 Mapper Interfacing

Prepare mapper interface and designate @Mapper annotation as follows:

(This allows Namespace on the mapper XML chosen and the interface method called as a query ID)

```
@Mapper("employerMapper")
public interface EmployerMapper	{

	public List<EmpVO> selectEmployerList(EmpVO vo);

	public EmpVO selectEmployer(BigDecimal empNo);

	public void insertEmployer(EmpVO vo);

	public int updateEmployer(EmpVO vo);

	public int deleteEmployer(BigDecimal empNo);

}
```

You need to configure MapperConfigurer on XML configuration.

Ex: context-mapper.xml

```
<bean class="egovframework.rte.psl.dataaccess.mapper.MapperConfigurer">
		<property name="basePackage" value="egovframework.rte.**.mapper" />
</bean>
```

You can configure @Mapper annotation scanning within the package designated in basePackage.

⇒ Inject the interface designated as @Mapper into @Service.

```
public class EmployerMapperTest {

	@Resource(name = "employerMapper")
	EmployerMapper employerMapper;

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

		// insert
		employerMapper.insertEmployer(vo);

		// select
		EmpVO resultVO = employerMapper.selectEmployer(vo.getEmpNo());

		// check
		checkResult(vo, resultVO);
	}
```

3.3 Annotation Method

With the annotation method, you no longer need to prepare mapper XML as the query is automatically designated in the Mapper Interface by way of annotation @Select, @Insert, @Update and @Delete.

```
@Mapper("departmentMapper")
public interface DepartmentMapper	{

		@Select("select DEPT_NO as deptNo, DEPT_NAME as deptName, LOC as loc from DEPT where
DEPT_NO = #{deptNo}")
	public DeptVO selectDepartment(BigDecimal deptNo);

		@Insert("insert into DEPT(DEPT_NO, DEPT_NAME, LOC) values (#{deptNo}, #{deptName}, #{loc})")
	public void insertDepartment(DeptVO vo);

		@Update("update DEPT set DEPT_NAME = #{deptName}, LOC = #{loc} WHERE DEPT_NO =
#{deptNo}")
	public int updateDepartment(DeptVO vo);

		@Delete("delete from DEPT WHERE DEPT_NO = #{deptNo}")
	public int deleteDepartment(BigDecimal deptNo);

}
```

⇒ Although you do not need a discrete mapper XML, keep in mind dynamic query is not available.