SQL Mapping Framework Easy & Simple

MyBatis

자바코드로 부터 SQL문을 분리해서 관리
매개변수 설정과 쿼리결과를 읽어오는 코드를 제거
작성할 코드가 줄어서 생산성 향상 & 유지 보수 편리

```java
public int insertUser(User user) {
    int rowCnt = FAIL;

    Connection conn = null;
    PreparedStatement pstmt = null;

    String sql = "insert into user_info values (?, ?, ?, ?,?,?, now()) ";

    try {
        conn = ds.getConnection();
        pstmt = conn.prepareStatement(sql); // SQL Injection공격, 성능향상
        pstmt.setString( parameterIndex: 1, user.getId());
        pstmt.setString( parameterIndex: 2, user.getPwd());
        pstmt.setString( parameterIndex: 3, user.getName());
        pstmt.setString( parameterIndex: 4, user.getEmail());
        pstmt.setDate( parameterIndex: 5, new java.sql.Date(user.getBirth().getTime()));
        pstmt.setString( parameterIndex: 6, user.getSns());

        return pstmt.executeUpdate(); //  insert, delete, update;
    } catch (SQLException e) {
        e.printStackTrace();
        return FAIL;
    } finally {
        close(pstmt, conn);  //     private void close(AutoCloseable... acs) {
```

```xml
?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.acorn.woo.MemberMapper">
    <select id="count" resultType="int">
        SELECT count(*) FROM test112
    </select>

    <select id="selectAll" resultType="Member">
        SELECT *
        FROM test112
    </select>


    <insert id="insert" parameterType="Member">
      INSERT INTO test112
          (id, pwd, name)
      VALUES
          (#{id}, #{pwd}, #{name})
    </insert>
</mapper>
```

mapper 설정파일
<typeAliases>
지정해야함

```java
@Repository
public class MemberDaoMybatis {
    @Autowired
    private SqlSession session;
    private static String namespace =
"com.acorn.woo.MemberMapper.";

    public int count() throws Exception {
        return session.selectOne(namespace+"count");
    }


    public List<Member> selectAll() throws Exception {
        return session.selectList(namespace+"selectAll");
    }


     public int insert(Member dto) throws Exception {
        return session.insert(namespace+"insert", dto);
    }
```

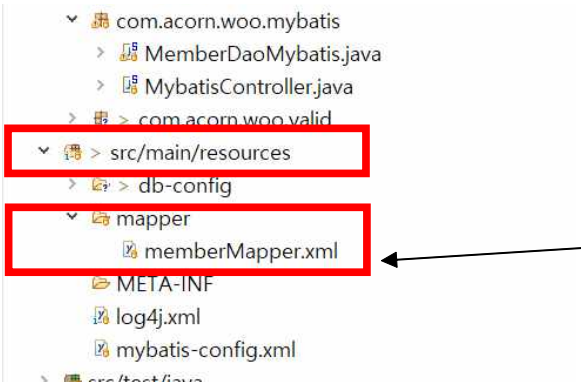
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
                PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
                "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
        <typeAliases>
                <typeAlias alias="Member" type="com.acorn.woo.model.Member"/>
        </typeAliases>
</configuration>
```

typeAlias 지정하면
패키지명 전체를 쓰지 않아도 됨

root-context.xml

```xml
        <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
                <property name="dataSource" ref="dataSource"/>
                <property name="configLocation" value="classpath:mybatis-config.xml"/>
                <property name="mapperLocations" value="classpath:mapper/*Mapper.xml"/>
        </bean>

        <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
                <constructor-arg ref="sqlSessionFactory"/>
        </bean>
```



Mapper 위치
(src/main/resources)

mapper폴더아래 (폴더만들어야함) 에
xxxMapper.xml 끝나는 문서의
이름으로 작성

#{}
${}

값에만 쓰일 수 있다. (제한적 사용)

```
<insert id="insert" parameterType="BoardDto">
    INSERT INTO board
        (title, content, writer)
    VALUES
        (#{title}, #{content}, #{writer})
</insert>
```

```
String sql = "INSERT INTO board "
           + " (title, content, writer) "
           + "VALUES "
           + " (?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(sql);
int result = pstmt.executeUpdate();
```

```
<insert id="insert" parameterType="BoardDto">
    INSERT INTO board
        (title, content, writer)
    VALUES
        ('${title}', '${content}', '${writer}')
</insert>
```

```
String sql = "INSERT INTO board "
           + " (title, content, writer) "
           + "VALUES "
           + " ('"+title+"','"+content+"','"+writer+"')";

Statement stmt = conn.createStatement();
int result = stmt.executeUpdate(sql);
```

동적으로 sql 작성할 때 사용한다.
(컬럼명같은 경우)

XML내의 특수문자 (< , > , &, ...)는 < >로 변환필요
또는 특수문자가 포함된 쿼리를 **<![CDATA[쿼리문작성]]>**

```
<update id="update" parameterType="BoardDto">
    UPDATE board
    SET   title   = #{title}
        , content = #{content}
        , up_date = now()
    WHERE bno = #{bno} and bno &lt;&gt; 0
</update>
```

```
<update id="update" parameterType="BoardDto">
    <![CDATA[
    UPDATE board
    SET   title   = #{title}
        , content = #{content}
        , up_date = now()
    WHERE bno = #{bno} and bno <> 0
    ]]>
</update>
```

특수문자 < >
<> 태그로 오인됨
xml태그가 없다
character data임을 표시함
문자데이타 임을 표시임

dependency 추가

```
<dependency>
		<groupId>org.mybatis</groupId>
		<artifactId>mybatis</artifactId>
		<version>3.5.9</version>
	</dependency>
	<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
	<dependency>
		<groupId>org.mybatis</groupId>
		<artifactId>mybatis-spring</artifactId>
		<version>2.0.7</version>
	</dependency>
```

```
<dependency>
	<groupId>org.springframework</groupId>
	<artifactId>spring-jdbc</artifactId>
	<version>${org.springframework-version}</version>
</dependency>
		<dependency>
	<groupId>com.oracle.database.jdbc</groupId>
	<artifactId>ojdbc8</artifactId>
	<version>21.7.0.0</version>
</dependency>
```