**MyBatis**

1. **MyBatis 简介**

MyBatis 本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。2013年11月迁移到Github。

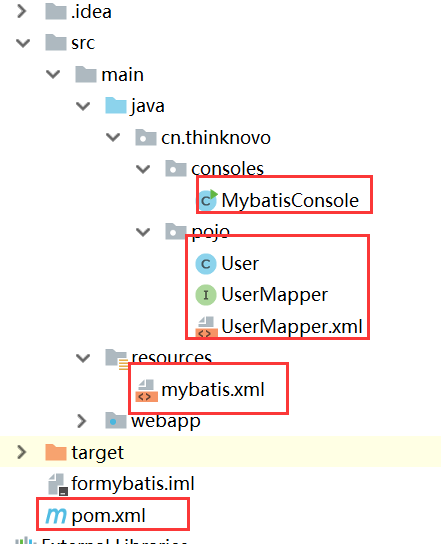
iBATIS一词来源于“internet”和“abatis”的组合，是一个基于Java的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（DAOs）

MyBatis虽然改变了名字，但是实际的内核还是ibatis

MyBatis以高轻量级闻名于全球，提出以注入SQL的方式来管理持久化，让整个项目在数据库操作这块与SQL语言完全解耦，所以深受开发人员的喜爱

1. **Maven MyBatis搭建**

基础脚手架效果如图：



需要配置的几项内容：

(1).pom文件

(2).mybatis.xml文件

(3).pojo类

(4).数据库映射XML

(5).创建数据库操作对应的接口

(6).控制器

第一项，配置pom文件，需要加入两个maven包，一个是数据库连接驱动，一个是mybatis

<**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>5.1.46</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>org.mybatis</**groupId**>  
 <**artifactId**>mybatis</**artifactId**>  
 <**version**>3.4.6</**version**>  
</**dependency**>

第二项，添加mybatis.xml文件

*<?***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**>  
 <**settings**>  
 *<!--当返回行的所有列都是空时，MyBatis默认返回null-->* <**setting name="returnInstanceForEmptyRow" value="true"**/>

*<!--显示sql语句-->*<**setting name="logImpl" value="STDOUT\_LOGGING"**/>

</**settings**>  
  
 <**environments default="development"**>  
  
 <**environment id="development"**>  
 *<!--使用了 JDBC 的提交和回滚设置，它依赖于从数据源得到的连接来管理事务作用域-->* <**transactionManager type="JDBC"**/>  
 *<!-- mybatis提供了3种数据源类型，分别是：POOLED,UNPOOLED,JNDI -->  
 <!-- POOLED 表示支持JDBC数据源连接池 -->  
 <!-- UNPOOLED 表示不支持数据源连接池 -->  
 <!-- JNDI 表示支持外部数据源连接池 -->* <**dataSource type="POOLED"**>  
 <**property name="driver" value="com.mysql.jdbc.Driver"**/>  
 <**property name="url" value="jdbc:mysql://localhost:3306/hiber"**/>  
 <**property name="username" value="root"**/>  
 <**property name="password" value="thinknovo"**/>  
 </**dataSource**>  
 </**environment**>  
 </**environments**>  
  
 <**mappers**>  
 *<!-- 映射文件方式1，一个一个的配置  
 <mapper resource="cn/thinknovo/pojo/UserMapper.xml"/>-->  
 <!-- 映射文件方式2，自动扫描包内的Mapper接口与配置文件 -->* <**package name="cn/thinknovo/pojo"**/>  
 </**mappers**>  
</**configuration**>

第三项，pojo类，创建一个类名与数据库相同，属性名和数据库字段名相同的pojo对象。

第四项，数据库映射XML文件

*<?***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"*>****<!-- namespace一般定义为 包名+文件名 -->*<**mapper namespace="cn.thinknovo.pojo.UserMapper"**>  
 <**select id="selectUser" resultType="cn.thinknovo.pojo.User"**>  
 select *\** from user;  
 </**select**>  
  
 <**insert id="insertUser" parameterType="cn.thinknovo.pojo.User"**>  
 insert into user values (#{id},#{username},#{password});  
 </**insert**>  
</**mapper**>

第五项，定义数据库操作接口，这里需要注意的是，mybatis以此接口为调用入口，通过此接口对应的方法来检查并对应mapper.xml文件中的增删改查

**public interface** UserMapper {  
 *// 方法名、参数类型、返回类型要和mapper.xml文件中的id相同* List<User> selectUser();  
 **void** insertUser(User user);  
}

第六项，创建控制器

**public class** MybatisConsole {  
 **public static void** main(String[] args) {  
 **try** {  
 Reader reader = Resources.*getResourceAsReader*(**"mybatis.xml"**);  
 SqlSessionFactory sqlSessionFactory = **new** SqlSessionFactoryBuilder().build(reader);  
 SqlSession session = sqlSessionFactory.openSession();  
 UserMapper userMapper = session.getMapper(UserMapper.**class**);  
 List<User> userList = userMapper.selectUser();  
 System.***out***.println(userList);  
 session.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

至此，基础脚手架搭建完成

1. **Mapper.xml文件的注册和自动扫描**

在mybatis.xml文件中，可以直接使用package的方式来完成自动扫描，这样以后就不需要一个个的添加mapper.xml文件了

<**mappers**>  
 *<!-- 映射文件方式1，一个一个的配置  
 <mapper resource="cn/thinknovo/pojo/UserMapper.xml"/>-->  
 <!-- 映射文件方式2，自动扫描包内的Mapper接口与配置文件 -->* <**package name="cn/thinknovo/pojo"**/>  
 </**mappers**>

1. **两种处理操作数据库的方式**

(1).通过session得到对应的Mapper接口对象，然后由此对象调用对应的操作方法，例如：

UserMapper userMapper = session.getMapper(UserMapper.**class**);  
List<User> userList = userMapper.selectUser();  
System.***out***.println(userList);

(2).通过session直接调用接口中对应的操作方法

*// 第一个参数对应接口中的方法名*session.selectList(**"selectUser"**);

**(PS：注意在这里接口文件和Mapper.xml文件是必须的，缺一不可，可以通过删除对应的文件来测试执行结果。)**

1. **Xml配置方式的增删改查**

首先配置mapper.xml文件

*<!-- namespace一般定义为 包名+文件名 -->*<**mapper namespace="cn.thinknovo.pojo.UserMapper"**>  
 <**select id="selectUser" resultType="cn.thinknovo.pojo.User"**>  
 select *\** from user;  
 </**select**>  
  
 <**insert id="insertUser" parameterType="cn.thinknovo.pojo.User"**>  
 insert into user values (#{id},#{username},#{password});  
 </**insert**>  
  
 <**update id="updateUser" parameterType="cn.thinknovo.pojo.User"**>  
 update user set **username**=#{username},**password**=#{password} where **id**=#{id};  
 </**update**>  
  
 <**delete id="deleteUser" parameterType="java.lang.String"**>  
 delete from user where **id**=#{id};  
 </**delete**>  
</**mapper**>

然后在修改接口代码

**public interface** UserMapper {  
 *// 方法名、参数类型、返回类型要和mapper.xml文件中的id相同* List<User> selectUser();  
 **void** insertUser(User user);  
 **void** updateUser(User user);  
 **void** deleteUser(String id);  
}

在控制器中调用接口中指定的四个方法，就可以完成基本的增删改查

1. **多参数传递**

在项目中，多参数传递更多的是基于pojo对象，或者map结构传递，但是也可能有需要多个形式参数传递，比如这样：

UserMapper.insertUserByPassword(**"think12345"**,**"54321"**);

这种情况，配置文件需要做对应的处理，改成下面的样子：

<**insert id="insertUserByPassword" parameterType="java.lang.String"**>  
 insert into user values(null,#{param1},#{param2});  
</**insert**>

param1代表获取第一个形式参数，依次类推

也可以这样：

<**insert id="insertUserByPassword" parameterType="java.lang.String"**>  
 insert into user values(null,#{arg0},#{arg1});  
</**insert**>

arg0代表获取第一个形式参数，依次类推

如果有更多的这样的形式参数，就递增数字，比如第三个就是parma3或者arg2，在动态配置sql的时候，根据需要可以任意调换顺序，比如可以换成下面的效果：

<**insert id="insertUserByPassword" parameterType="java.lang.String"**>  
 insert into user values(null,#{arg1},#{arg0});  
</**insert**>

arg0还是代表第一个参数，arg1还是代表第二个参数，不受影响。

1. **Java中的Map结构传递**

如果在项目中需要多参数传递，更建议大家使用java.util.map的结构传递，这也是大部分项目使用的方式，mybatis会根据key来对应获取value值，设置的传递类型还是parameterType，比如这样：

<**insert id="insertUserByUserAndPassword" parameterType="map"**>  
 insert into user values(null,#{username},#{password});  
</**insert**>

1. **Map结构传递**

如果要定义为map结构传递，需要在mybatis.xml中定义map的结构方式，比如这样：

<**parameterMap id="userMap" type="cn.thinknovo.pojos.User"**>  
 <**parameter property="username" javaType="java.lang.String"**/>  
 <**parameter property="password" javaType="java.lang.String"**/>  
</**parameterMap**>

这里的id就是map引入的名字，我们在传递参数类型设定为map以后就可以使用，比如：

<**insert id="insertUserByUserAndPassword2" parameterMap="userMap"**>  
 insert into user values(null,#{username},#{password});  
</**insert**>

1. **注解的使用**

删除mapper.xml文件，然后在接口文件中通过注解的方式添加sql语句

**public interface** UserMapper {  
 *// 方法名、参数类型、返回类型要和mapper.xml文件中的id相同* @Select(**"select \* from user"**)  
 List<User> selectUser();  
 @Insert(**"insert into user values (#{id},#{username},#{password})"**)  
 **void** insertUser(User user);  
 @Update(**"update user set username=#{username},password=#{password} where id=#{id}"**)  
 **void** updateUser(User user);  
 @Delete(**"delete from user where id=#{id}"**)  
 **void** deleteUser(String id);  
}

1. **动态SQL的if使用**

<**select id="selectUser" resultType="cn.thinknovo.pojo.User" parameterType="cn.thinknovo.pojo.User"**>  
 select *\** from user  
 <**if test="username != null"**>  
 where **username**=#{username}  
 </**if**>  
</**select**>

在配置文件中添加动态的if语句，如果传递过来的user对象中，username不等于null，那么追加条件查询

1. **动态SQL的where使用**

如果我们有多个条件需要添加，但是有可能一个条件都不满足，那么这时候使用上面的方式就会出现问题，所以我们还需要在if外部添加一个where标签

<**select id="selectUser" resultType="cn.thinknovo.pojo.User" parameterType="cn.thinknovo.pojo.User"**>  
 select *\** from user  
 <**where**>  
 <**if test="username != null"**>  
 username=#{username}  
 </**if**>  
 <**if test="password != null"**>  
 and password=#{password}  
 </**if**>  
 </**where**>  
</**select**>

可以看到，where包含了两个if，而第一个if中没有and，where标签会自动检查状态，如果标签返回的第一个匹配内容是以 and或or 开头的，则会自动删除and或者or ，然后再追加where

1. **动态SQL的choose when使用**

<**select id="selectUser" resultType="cn.thinknovo.pojo.User" parameterType="cn.thinknovo.pojo.User"**>  
 select *\** from user where **password**='888888'  
 <**choose**>  
 <**when test="username != null"**>  
 AND **username**=#{username}  
 </**when**>  
 <**when test="password != null"**>  
 AND **password**=#{password}  
 </**when**>  
 <**otherwise**>  
 AND **id**=#{id}  
 </**otherwise**>  
 </**choose**>  
</**select**>

Choose when 类似于java中的switch case，任中只会走一个，一般是在有一个条件固定的情况下，需要额外不确定的条件时，使用choose的方式

POJO映射工具的配置及使用：

1. **mybatis-generator-maven-plugin**

**(1)首先修改pom文件为下面的内容，需要注意configurationFile标签是对应mybatis-generator的路径和文件名，修改为自己想要的，其余不做修改：**

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**groupId**>cn.thinknovo</**groupId**>  
 <**artifactId**>mybatis002</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 <**packaging**>war</**packaging**>  
  
 <**name**>mybatis002 Maven Webapp</**name**>  
 *<!--* ***FIXME change it to the project's website*** *-->* <**url**>http://www.example.com</**url**>  
  
 <**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**maven.compiler.source**>1.7</**maven.compiler.source**>  
 <**maven.compiler.target**>1.7</**maven.compiler.target**>  
 </**properties**>  
  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>junit</**groupId**>  
 <**artifactId**>junit</**artifactId**>  
 <**version**>4.11</**version**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>5.1.46</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.mybatis</**groupId**>  
 <**artifactId**>mybatis</**artifactId**>  
 <**version**>3.4.6</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.mybatis.generator</**groupId**>  
 <**artifactId**>mybatis-generator-core</**artifactId**>  
 <**version**>1.3.6</**version**>  
 </**dependency**>  
 </**dependencies**>  
  
 <**build**>  
 <**finalName**>mybatis002</**finalName**>  
 <**pluginManagement**>*<!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->* <**plugins**>  
 <**plugin**>  
 <**artifactId**>maven-clean-plugin</**artifactId**>  
 <**version**>3.0.0</**version**>  
 </**plugin**>  
 *<!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_war\_packaging -->* <**plugin**>  
 <**artifactId**>maven-resources-plugin</**artifactId**>  
 <**version**>3.0.2</**version**>  
 </**plugin**>  
 <**plugin**>  
 <**artifactId**>maven-compiler-plugin</**artifactId**>  
 <**version**>3.7.0</**version**>  
 </**plugin**>  
 <**plugin**>  
 <**artifactId**>maven-surefire-plugin</**artifactId**>  
 <**version**>2.20.1</**version**>  
 </**plugin**>  
 <**plugin**>  
 <**artifactId**>maven-war-plugin</**artifactId**>  
 <**version**>3.2.0</**version**>  
 </**plugin**>  
 <**plugin**>  
 <**artifactId**>maven-install-plugin</**artifactId**>  
 <**version**>2.5.2</**version**>  
 </**plugin**>  
 <**plugin**>  
 <**artifactId**>maven-deploy-plugin</**artifactId**>  
 <**version**>2.8.2</**version**>  
 </**plugin**>  
  
 <**plugin**>  
 *<!--Mybatis-generator插件,用于自动生成Mapper和POJO-->* <**groupId**>org.mybatis.generator</**groupId**>  
 <**artifactId**>mybatis-generator-maven-plugin</**artifactId**>  
 <**version**>1.3.6</**version**>  
 <**configuration**>  
 *<!--配置文件的位置-->* <**configurationFile**>src\main\resources\mybatis-generator-config.xml</**configurationFile**>  
 <**verbose**>true</**verbose**>  
 <**overwrite**>true</**overwrite**>  
 </**configuration**>  
 <**executions**>  
 <**execution**>  
 <**id**>Generate MyBatis Artifacts</**id**>  
 <**goals**>  
 <**goal**>generate</**goal**>  
 </**goals**>  
 </**execution**>  
 </**executions**>  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>org.mybatis.generator</**groupId**>  
 <**artifactId**>mybatis-generator-core</**artifactId**>  
 <**version**>1.3.6</**version**>  
 </**dependency**>  
 </**dependencies**>  
 </**plugin**>  
 </**plugins**>  
 </**pluginManagement**>  
 </**build**>  
</**project**>

**(2)接着添加mybatis-generator-config.xml文件，注意修改**

**(1)classPathEntry为自己的数据库连接池jar包物理路径**

**(2)jdbcConnection为自己的数据库连接参数**

**(3)javaModelGenerator为实体类生成的位置 修改targetPackage和targetProject**

**(4)sqlMapGenerator为\*Mapper.xml 文件的位置 修改targetPackage和targetProject**

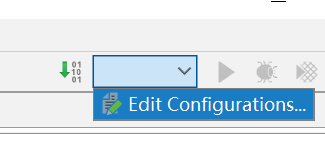
**(5)javaClientGenerator为Mapper 接口文件的位置 修改targetPackage和targetProject**

**(6)table为表对应内容，多表就配置多个，tableName表名 domainObjectName类名**

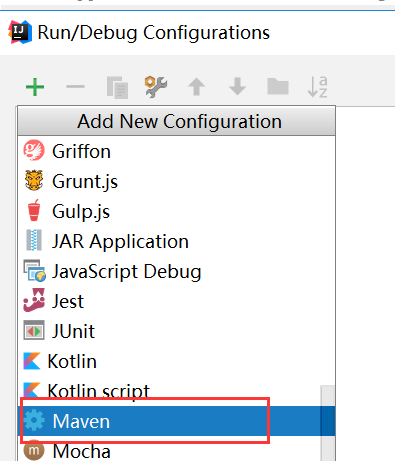
**完整配置如下：**

*<?***xml version="1.0" encoding="UTF-8"** *?>***<!DOCTYPE generatorConfiguration PUBLIC  
 "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  
 "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd" *>***<**generatorConfiguration**>  
  
 *<!-- 本地数据库驱动程序jar包的全路径 -->* <**classPathEntry location="C:\Users\Administrator\.m2\repository\mysql\mysql-connector-java\5.1.46\mysql-connector-java-5.1.46.jar"**/>  
  
 *<!-- 指定mybatis版本 -->* <**context id="context" targetRuntime="MyBatis3"**>  
 <**commentGenerator**>  
 <**property name="suppressAllComments" value="false"**/>  
 <**property name="suppressDate" value="true"**/>  
 </**commentGenerator**>  
  
 *<!-- 数据库的相关配置 -->* <**jdbcConnection driverClass="com.mysql.jdbc.Driver"  
 connectionURL="jdbc:mysql://localhost:3306/hiber"  
 userId="root"  
 password="thinknovo"**/>  
  
 *<!--指定生成的类型为java类型，避免数据库中number等类型字段 -->* <**javaTypeResolver**>  
 <**property name="forceBigDecimals" value="false"**/>  
 </**javaTypeResolver**>  
  
 *<!-- 实体类生成的位置 修改targetPackage和targetProject-->* <**javaModelGenerator targetPackage="cn.thinknovo.test" targetProject="src/main/java"**>  
 <**property name="enableSubPackages" value="false"**/>  
 <**property name="trimStrings" value="true"**/>  
 </**javaModelGenerator**>  
  
 *<!-- \*Mapper.xml 文件的位置 修改targetPackage和targetProject-->* <**sqlMapGenerator targetPackage="cn.thinknovo.test" targetProject="src/main/java"**>  
 <**property name="enableSubPackages" value="false"**/>  
 </**sqlMapGenerator**>  
  
 *<!-- Mapper 接口文件的位置 修改targetPackage和targetProject-->* <**javaClientGenerator targetPackage="cn.thinknovo.test" targetProject="src/main/java"  
 type="XMLMAPPER"**>  
 <**property name="enableSubPackages" value="false"**/>  
 </**javaClientGenerator**>  
  
 *<!-- 相关表的配置 多表就配置多个 tableName表名 domainObjectName类名 -->* <**table tableName="user" domainObjectName="User" enableCountByExample="false"  
 enableDeleteByExample="false" enableSelectByExample="false"  
 enableUpdateByExample="false"**/>  
 </**context**>  
</**generatorConfiguration**>

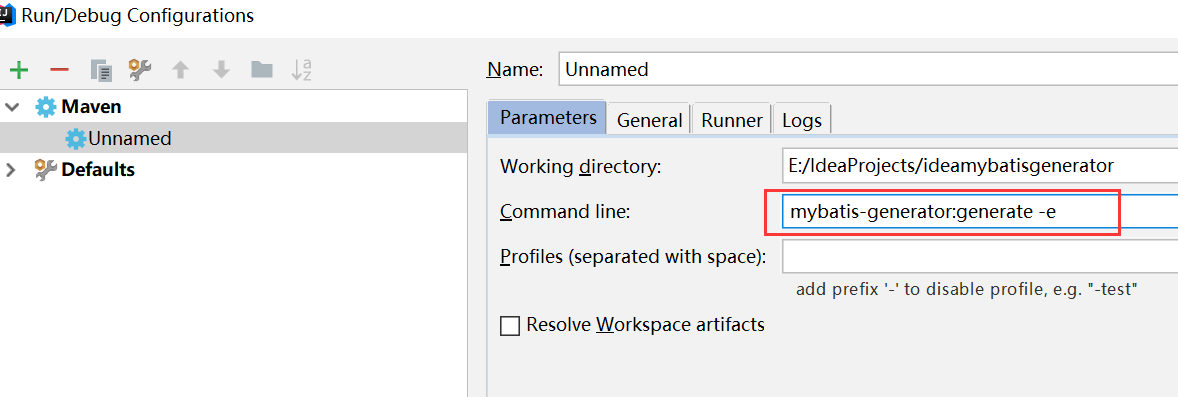
**(3)接着在idea右上角，点击edit**



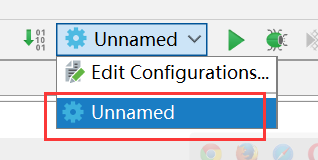
**(4)开始配置mybatis-generator-maven-plugin运行**



**(5)输入运行命令mybatis-generator:generate -e**



**(6)点击保存，现在可以看到一个运行方式了，点击运行就可以自动得到内容了**



**(7)运行生成对应的文件**

