

# mybatis.generator生成代码插件使用

可以生成简单CRUD操作的XML配置文件、Mapper文件(DAO接口)、实体类。实际开发中能够有效减少程序员的工作量，甚至不用程序员手动写sql。

步骤：

1.引入依赖

*<!--引入mybatis的生成插件-->*<**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.mybatis.generator</**groupId**>  
 <**artifactId**>mybatis-generator-maven-plugin</**artifactId**>  
 <**version**>1.3.5</**version**>  
 </**plugin**>  
 </**plugins**>  
</**build**>

2.配置文件

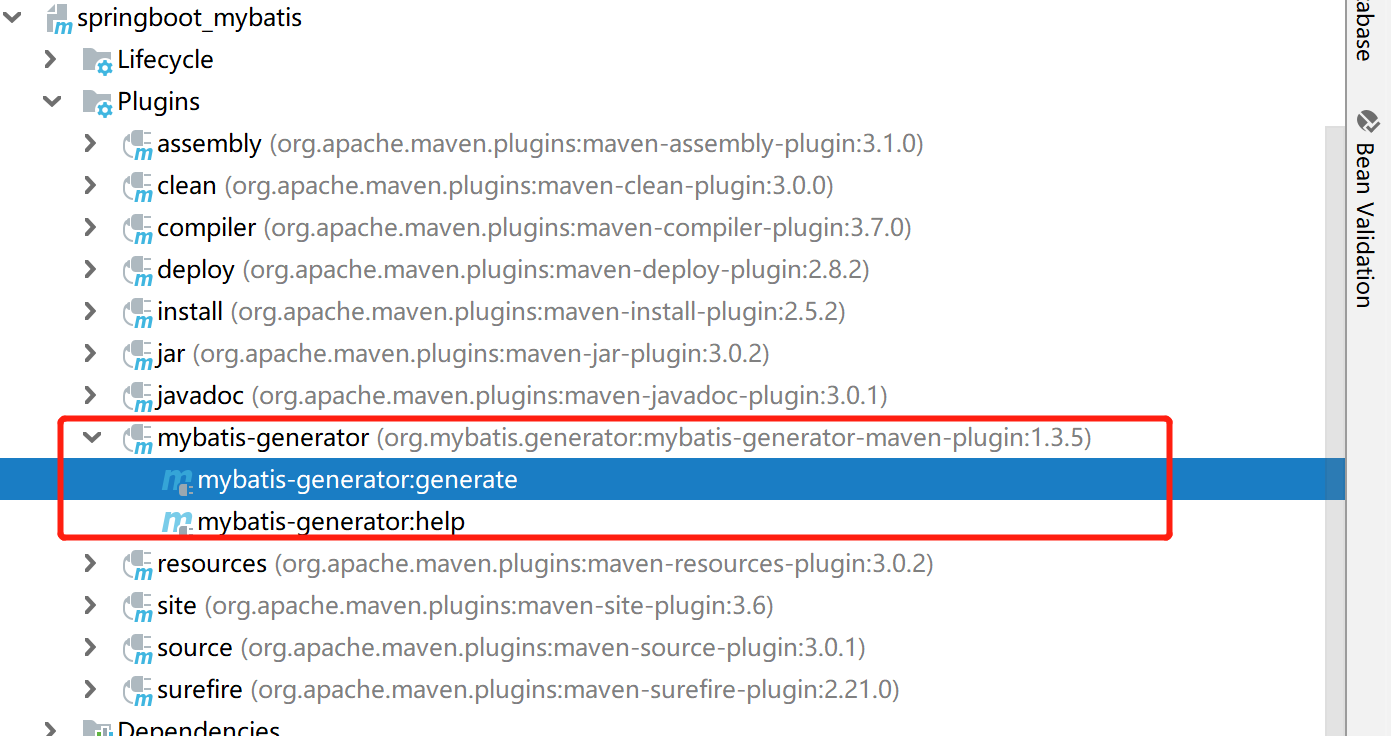
generator.properties文件如下：

|  |
| --- |
| *#generatorConfig Info* **generator.location**=**E:\\soft\\apache-maven-3.6.1\\maven-repository\\mysql\\mysql-connector-java\\5.1.47\\mysql-connector-java-5.1.47.jar** *#generator.targetPackage=com.springboot.mybatis #gererator.schema=oracle-schema* **gererator.tableName**=**user gererator.domainObjectName**=**MyUser generator.targetProject**=**./src/main/java generator.resourceTargetProject**=**./src/main/resources generator.domainTargetPackage**=**com.springboot.mybatis.bean generator.mapperXmlTargetPackage**=**mapper generator.mapperInterfaceTargetPackage**=**com.springboot.mybatis.mapper  jdbc.driver**=**com.mysql.jdbc.Driver jdbc.host**=**jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=utf-8&useSSL=false&allowPublicKeyRetrieval=true jdbc.userName**=**root jdbc.passWord**=**1234 jdbc.initialSize**=**0 jdbc.maxActive**=**20 jdbc.maxIdle**=**20 jdbc.minIdle**=**1 jdbc.maxWait**=**1000** |

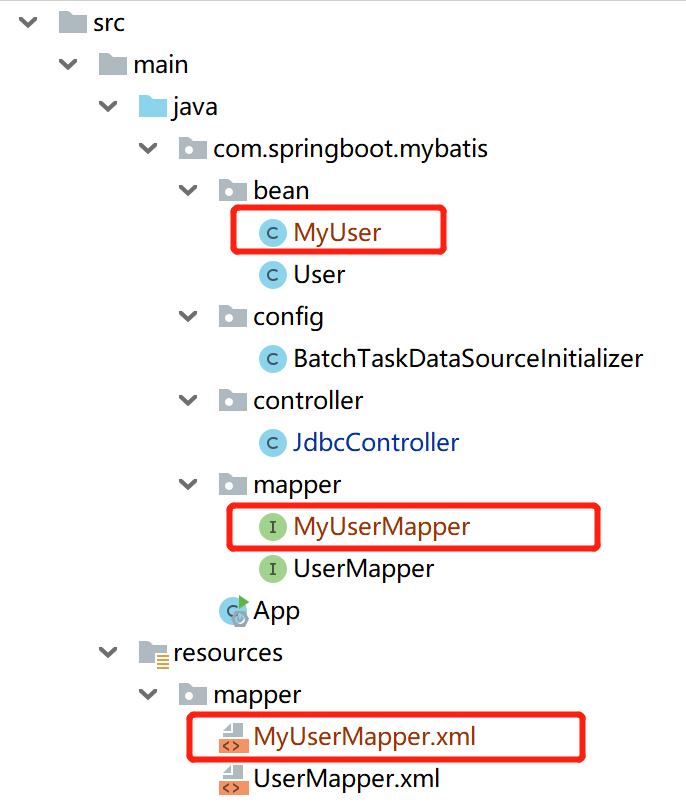
generatorConfig.xml文件如下：

|  |
| --- |
| *<?***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**>  *<!-- 引入配置文件 -->* <**properties resource="generator.properties"**/>  *<!-- 数据库驱动包位置,路径请不要有中文-->  <!-- <classPathEntry location="D:\software\lib\mysql-connector-java-5.1.21.jar" /> -->* <**classPathEntry location="${generator.location}"**/>  *<!-- 一个数据库一个context-->* <**context id="DB2Tables" targetRuntime="MyBatis3"**>  *<!-- 生成的pojo，将implements Serializable -->* <**plugin type="org.mybatis.generator.plugins.SerializablePlugin"**></**plugin**>   *<!-- 注释 -->* <**commentGenerator**>  <**property name="suppressAllComments" value="true"**/>*<!-- 是否取消注释 -->* <**property name="suppressDate" value="true"** /> *<!--是否生成注释代时间戳 -->* </**commentGenerator**>   *<!-- 数据库链接URL、用户名、密码 -->  <!-- <jdbcConnection driverClass="com.mysql.jdbc.Driver" connectionURL="jdbc:mysql://localhost:3306/test" userId="root" password="1234"> -->* <**jdbcConnection driverClass="${jdbc.driver}" connectionURL="${jdbc.host}" userId="${jdbc.userName}"  password="${jdbc.passWord}"**>  </**jdbcConnection**>   *<!-- 类型转换 -->* <**javaTypeResolver**>  *<!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer true，把JDBC DECIMAL  和 NUMERIC 类型解析为java.math.BigDecimal -->* <**property name="forceBigDecimals" value="false"**/>  </**javaTypeResolver**>   *<!-- 生成model模型，设置对应的包名(targetPackage)和存放路径(targetProject)。targetProject可以指定具体的路径,如./src/main/java,也可以使用MAVEN来自动生成,这样生成的代码会在target/generatord-source目录下 -->* <**javaModelGenerator targetPackage="${generator.domainTargetPackage}" targetProject="${generator.targetProject}"**>  *<!-- 是否在当前路径下新加一层schema,eg：false路径com.oop.eksp.user.model 而true:com.oop.eksp.user.model.[schemaName] -->* <**property name="enableSubPackages" value="false"**/>  *<!-- 从数据库返回的值被清理前后的空格 -->* <**property name="trimStrings" value="true"**/>  </**javaModelGenerator**>   *<!--对应的mapper.xml文件 -->* <**sqlMapGenerator targetPackage="${generator.mapperXmlTargetPackage}" targetProject="${generator.resourceTargetProject}}"**>  <**property name="enableSubPackages" value="true"**/>  </**sqlMapGenerator**>   *<!-- 对应的Mapper接口类文件 -->* <**javaClientGenerator type="XMLMAPPER" targetPackage="${generator.mapperInterfaceTargetPackage}"  targetProject="${generator.targetProject}"**>  <**property name="enableSubPackages" value="true"**/>  </**javaClientGenerator**>   *<!-- 列出要生成代码的所有表，这里配置的是不生成Example文件 -->  <!-- schema即为数据库名tableName为对应的数据库表 domainObjectName是要生成的实体类 enable\*ByExample是否生成 example类 -->* <**table tableName="${gererator.tableName}" domainObjectName="${gererator.domainObjectName}"  schema="${gererator.schema}"  enableCountByExample="false" enableUpdateByExample="false"  enableDeleteByExample="false" enableSelectByExample="false"  selectByExampleQueryId="false"**>  *<!-- 忽略列，不生成bean 字段  <ignoreColumn column="FRED" />-->  <!-- 指定列的java数据类型  <columnOverride column="LONG\_VARCHAR\_FIELD" jdbcType="VARCHAR" /> -->  <!-- 用于指定生成实体类时是否使用实际的列名作为实体类的属性名。false是 Camel Case风格-->* <**property name="useActualColumnNames" value="false"**/>  </**table**>  </**context**> </**generatorConfiguration**> |

3.生成代码



生成结果如下：



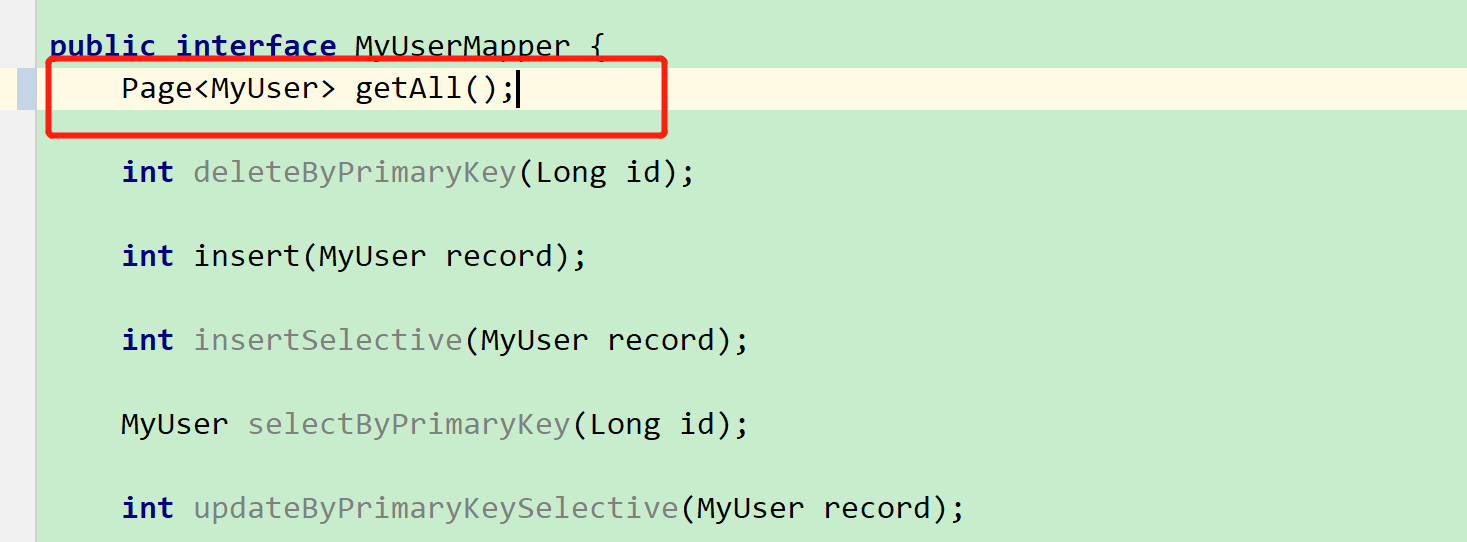
# Mybatis 分页使用

这里使用pagehelper分页插件后端实现分页：

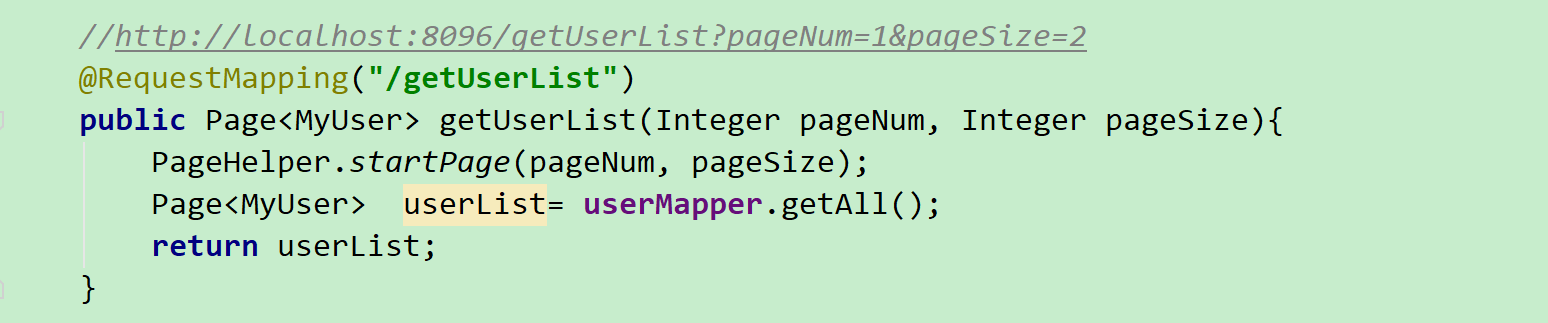
1.引入分页插件依赖

|  |
| --- |
| <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper-spring-boot-starter</**artifactId**>  <**version**>1.2.5</**version**> </**dependency**> |

返回Page对象



Controller里设置获取页数和每页的数量：



|  |
| --- |
| ****首先对于批量数据的插入有两种解决方案（下面内容只讨论和Mysql交互的情况）**** 1）for循环调用Dao中的单条插入方法  2）传一个List<Object>参数，使用Mybatis的批量插入 （foreach）  对于批量插入它的Mapper看起来向这样  *<!--批量插入-->* <**insert id="addUser" parameterType="java.util.List"** >  insert into user(id, name, age, sex) values  <**foreach collection="list" item="item" index="index" separator=","**>  (#{item.id},#{item.name},#{item.age},#{item.sex})  </**foreach**> </**insert**> |