**Mybatis-Generator自动生成器**

**MyBatis-Generator 介绍**

MyBatis Generator 是 MyBatis 官方提供的代码生成器插件，可以用于 MyBatis 和 iBatis 框架的代码生成，支持所有版本的 MyBatis 框架以及 2.2.0 版本及以上的 iBatis 框架。

在前文介绍了如何使用 MyBatis 进行数据库操作，在进行功能开发时，一张表我们需要编写实体类、DAO 接口类以及 Mapper 文件，这些是必不可少的文件，如果表的数量较大，我们就需要重复的去创建实体类、Mapper 文件以及 DAO 类并且需要配置它们之间的依赖关系，这无疑是一个很麻烦的事情，而 MyBatis Generator 插件就可以帮助我们去完成这些开发步骤，使用该插件可以帮助开发者自动去创建实体类、Mapper 文件以及 DAO 类并且配置好它们之间的依赖关系，我们可以直接在项目中使用这些代码，后续只需要关注业务方法的开发即可。

将 MyBatis-Generator 简单的理解为一个 MyBatis 框架的代码生成器，至于推荐使用它的原因，理由整理如下：

* 减少重复工作
* 减少人为操作带来的错误
* 提升开发效率
* 使用灵活

MyBatis 属于半自动 ORM 框架，在使用这个框架中，工作量最大的就是书写 Mapper 及相关映射文件，同时需要配置其依赖关系，由于手动书写很容易出错，可以利用 MyBatis-Generator 来帮我们自动生成文件。

其次，一个项目通常有很多张表，比如接下来要开发的 my\_blog 项目，所有的实体类、SQL 语句都要手动编写，这是一个比较繁琐的过程，如果有这么一个插件能够适当的减少我们的一些工作量，自动将这些代码生成到对应的项目目录中，将是一件十分幸福的事情，当然，该插件生成的代码都是常用的一些增删改查代码，如果有其他功能或方法依然需要自己去编写代码，它只是一个提升效率的工具，给予开发者一定程度的帮助。

**MyBatis-Generator 整合**

**添加依赖**

*<*build*>  
 <*plugins*>  
 <*plugin*>  
 <*groupId*>*org.mybatis.generator*</*groupId*>  
 <*artifactId*>*mybatis-generator-maven-plugin*</*artifactId*>  
 <*version*>*1.3.5*</*version*>  
 <*dependencies*>  
 <*dependency*>  
 <*groupId*>*org.mybatis.generator*</*groupId*>  
 <*artifactId*>*mybatis-generator-core*</*artifactId*>  
 <*version*>*1.3.5*</*version*>  
 </*dependency*>  
 </*dependencies*>  
 <*executions*>  
 <*execution*>  
 <*id*>*Generate MyBatis Artifacts*</*id*>  
 <*phase*>*package*</*phase*>  
 <*goals*>  
 <*goal*>*generate*</*goal*>  
 </*goals*>  
 </*execution*>  
 </*executions*>  
 <*configuration*>  
 <*verbose*>*true*</*verbose*>* <!-- 是否覆盖 -->  
 *<*overwrite*>*true*</*overwrite*>* <!-- MybatisGenerator的配置文件位置 -->  
 *<*configurationFile*>*src/main/resources/mybatisGeneratorConfig.xml*</*configurationFile*>  
 </*configuration*>  
 </*plugin*>  
 </*plugins*>  
</*build*>*

**新增 MyBatis-Generator 的配置文件**

在添加插件依赖到 pom.xml 文件中时，我们定义了 Mybatis Generator 的配置文件位置为 src/main/resources/mybatisGeneratorConfig.xml，该文件即为代码生成器插件最重要的配置文件，后续生成代码的规则、数据库连接信息、代码生成后的存放目录等等配置都需要在该配置文件中定义，配置文件内容及相关注释如下：

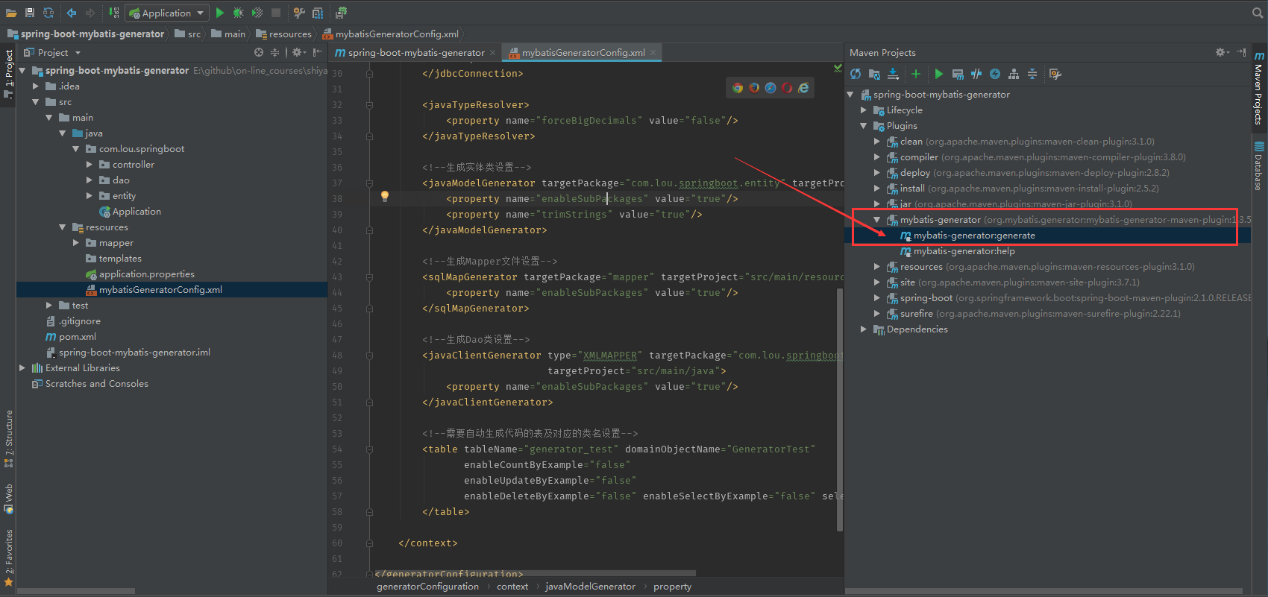
*<?*xml version="1.0" encoding="UTF-8"*?>  
<!DOCTYPE* generatorConfiguration PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"  
 "http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd"*>  
<*generatorConfiguration*>  
 <*context id="my-blog-generator-config" targetRuntime="MyBatis3"*>* <!-- 生成的Java文件的编码 -->  
 *<*property name="javaFileEncoding" value="utf-8"*/>* <!-- 格式化java代码 -->  
 *<*property name="javaFormatter" value="org.mybatis.generator.api.dom.DefaultJavaFormatter"*/>* <!-- 格式化XML代码 -->  
 *<*property name="xmlFormatter" value="org.mybatis.generator.api.dom.DefaultXmlFormatter"*/>  
 <*plugin type="org.mybatis.generator.plugins.ToStringPlugin"*/>* <!--创建Java类时对注释进行控制-->  
 *<*commentGenerator*>  
 <*property name="suppressDate" value="true"*/>* <!-- 是否去除自动生成的注释 true：是 ： false:否 -->  
 *<*property name="suppressAllComments" value="true"*/>  
 </*commentGenerator*>* <!--数据库地址及登陆账号密码 改成你自己的配置-->  
 *<*jdbcConnection  
 driverClass="com.mysql.jdbc.Driver"  
 connectionURL="jdbc:mysql://localhost:3306/my\_blog"  
 userId="root"  
 password="1002"*>  
 </*jdbcConnection*>  
 <*javaTypeResolver*>  
 <*property name="forceBigDecimals" value="false"*/>  
 </*javaTypeResolver*>* <!--生成实体类设置-->  
 *<*javaModelGenerator targetPackage="com.mybatisgenerator.entity" targetProject="src/main/java"*>* <!-- enableSubPackages:是否让schema作为包的后缀 -->  
 *<*property name="enableSubPackages" value="true"*/>* <!-- 从数据库返回的值被清理前后的空格 -->  
 *<*property name="trimStrings" value="true"*/>  
 </*javaModelGenerator*>* <!--生成Mapper文件设置-->  
 *<*sqlMapGenerator targetPackage="mapper" targetProject="src/main/resources"*>  
 <*property name="enableSubPackages" value="true"*/>  
 </*sqlMapGenerator*>* <!--生成Dao类设置-->  
 *<*javaClientGenerator type="XMLMAPPER" targetPackage="com.mybatisgenerator.dao"  
 targetProject="src/main/java"*>  
 <*property name="enableSubPackages" value="true"*/>  
 </*javaClientGenerator*>* <!--需要自动生成代码的表及对应的类名设置-->  
 *<*table tableName="bg\_users" domainObjectName="UsersTest"  
 enableCountByExample="false"  
 enableUpdateByExample="false"  
 enableDeleteByExample="false" enableSelectByExample="false" selectByExampleQueryId="false"*>  
 </*table*>  
 </*context*>  
</*generatorConfiguration*>*

需要自动生成代码的表及对应类名的配置是写在 table 标签中的，如上所示，generator\_test 表对应的实体类可以配置为 GeneratorTest，这些是开发者自定义的，也可以改成其他合适的类名。 如果有多张表同时生成，增加多个 table 标签配置即可。

**生成代码**

配置文件和表结构都处理完成后就可以进行代码生成步骤了，两种方式来进行代码生成：

* 方式一：IDEA 工具中的 Maven 插件中含有 mybatis-generator 的选项，点击 generate 即可，如下图所示：



* 方式二：执行 mvn 如下命令来进行代码生成，这是不通过开发工具提供的图形界面来进行的步骤。

mvn mybatis-generator:generate

执行这些步骤之后就可以看到实体类、DAO 接口类、Mapper 文件已经生成在对应的目录中了

**将 DAO 接口注册至 IOC 容器**

在代码生成后，我们需要在 DAO 接口类上增加 @Mapper 注解，将其注册到 Spring 的 IOC 容器中以供其他类调用，在生成的文件中默认是没有这个注解的，或者是在主类中添加 @MapperScan 注解将相应包下的所有 Mapper 接口扫描到容器中。