



链滴

# springboot+mybatis+mysql 通过 generator 配置文件生成代码

作者: [524021835](#)

原文链接: <https://ld246.com/article/1538298559325>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

第一次写文章！有写错的地方请指出！

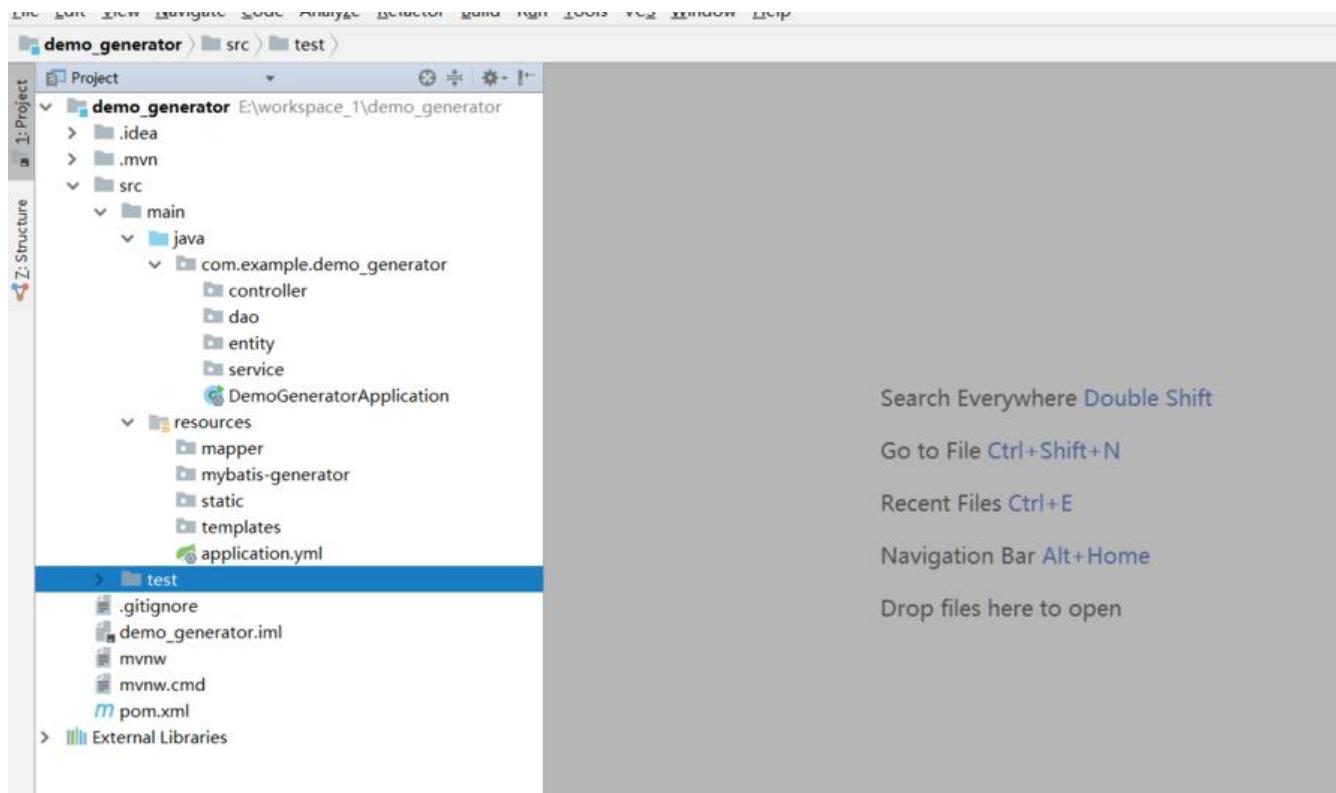
此篇文章主要是这两天自己搭springboot时用到generator生成相应的entity, dao以及xml文件的总。

## 一、通过<https://start.spring.io/>生成相应的demo文件

The screenshot shows the Spring Initializr interface. In the 'Project Metadata' section, 'Group' is set to 'com.example' and 'Artifact' is set to 'demo\_generator'. In the 'Dependencies' section, 'Web', 'MySQL', and 'MyBatis' are selected. A green 'Generate Project' button is at the bottom.

注意选择web、mysql、mybatis三个模块，能够在pom中生成相应的配置文件。

二、将文件下载下来后导入到idea中，并在src.main.java目录下增加放entity（实体类）、dao（mapper接口）的package，在resources下增加放mapper.xml的directory，以及增加一个文件夹放generate的配置文件的directory。具体目录结构如下图



## 三、在pom.xml中增加generate 的配置信息

## 1、在dependencies中添加以下内容

```
<!--mybatis 自动生成插件-->
<dependency>
    <groupId>org.mybatis.generator</groupId>
    <artifactId>mybatis-generator-core</artifactId>
    <version>1.3.5</version>
</dependency>
```

## 2、在plugins中添加以下内容

```
<!-- (1) mybatis generator 自动生成代码插件 -->
<plugin>
    <groupId>org.mybatis.generator</groupId>
    <artifactId>mybatis-generator-maven-plugin</artifactId>
    <version>1.3.2</version>
    <executions>
        <execution>
            <id>Generate MyBatis Artifacts</id>
            <phase>deploy</phase>
            <goals>
                <goal>generate</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <!-- generator 工具配置文件的位置 -->
        <configurationFile>src/main/resources/mybatis-generator/generatorConfig.xml</configurationFile>
        <verbose>true</verbose>
        <overwrite>true</overwrite>
    </configuration>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
            <version>${mysql.version}</version>
        </dependency>
    </dependencies>
</plugin>
```

## 三、在数据库中新建表sys\_user，

### 1、在resources下mybatis\_generator下新建generator.xml文件，并在里面加入以下内容

```
<?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">
<!-- 配置生成器 -->
<generatorConfiguration>
    <!--执行generator插件生成文件的命令： call mvn mybatis-generator:generate -e -->
    <!-- 引入配置文件 --> <properties resource="mybatis-generator/mybatisGeneratorInit.prop
```

```

rties"/>
    <!-- 一个数据库一个context -->
    <!--defaultModelType="flat" 大数据字段, 不分表 --> <context id\="MysqlTables" targetRuntime\="MyBatis3Simple" defaultModelType\="flat"\>
        <!-- 自动识别数据库关键字, 默认false, 如果设置为true, 根据SqlReservedWords中定义的键字列表; -->
        一般保留默认值, 遇到数据库关键字 (Java关键字), 使用columnOverride覆盖 --> <property name\="autoDelimitKeywords" value\="true" />
        <!-- 生成的Java文件的编码 -->
        <property name\="javaFileEncoding" value\="utf-8" />
        <!-- beginningDelimiter和endingDelimiter: 指明数据库的用于标记数据库对象名的符号, 如ORACLE就是双引号, MYSQL默认是\反引号; -->
        <property name\="beginningDelimiter" value\="`" />
        <property name\="endingDelimiter" value\="`" />

        <!-- 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.SerializablePlugin" />
        <plugin type\="org.mybatis.generator.plugins.ToStringPlugin" />

        <!-- 注释 -->
<commentGenerator\>
    <property name\="suppressAllComments" value\="false"/><!-- 是否取消注释 -->
    <property name\="suppressDate" value\="true" /> <!-- 是否生成注释代时间戳-->
</commentGenerator\>

        <!-- jdbc连接 -->
<jdbcConnection driverClass\="${jdbc\_driver}" connectionURL\="${jdbc\_url}" userId\="${jdbc\_user}" password\="${jdbc\_password}" />
        <!-- 类型转换 -->
<javaTypeResolver\>
    <!-- 是否使用BigDecimal, false可自动转化以下类型 (Long, Integer, Short, etc.) -->
<property name\="forceBigDecimals" value\="false"/>
</javaTypeResolver\>

        <!-- 生成实体类地址 -->
<javaModelGenerator targetPackage\="com.zhyq.admin.entity" targetProject\="${project}" >
    <property name\="enableSubPackages" value\="false"/>
    <property name\="trimStrings" value\="true"/>
</javaModelGenerator\>
        <!-- 生成mapxml文件 -->
<sqlMapGenerator targetPackage\="mapper" targetProject\="${resources}" >
    <property name\="enableSubPackages" value\="false" />
</sqlMapGenerator\>
        <!-- 生成mapxml对应client, 也就是接口dao -->
<javaClientGenerator targetPackage\="com.zhyq.admin.dao" targetProject\="${project}" type\="XMLMAPPER" >
    <property name\="enableSubPackages" value\="false" />

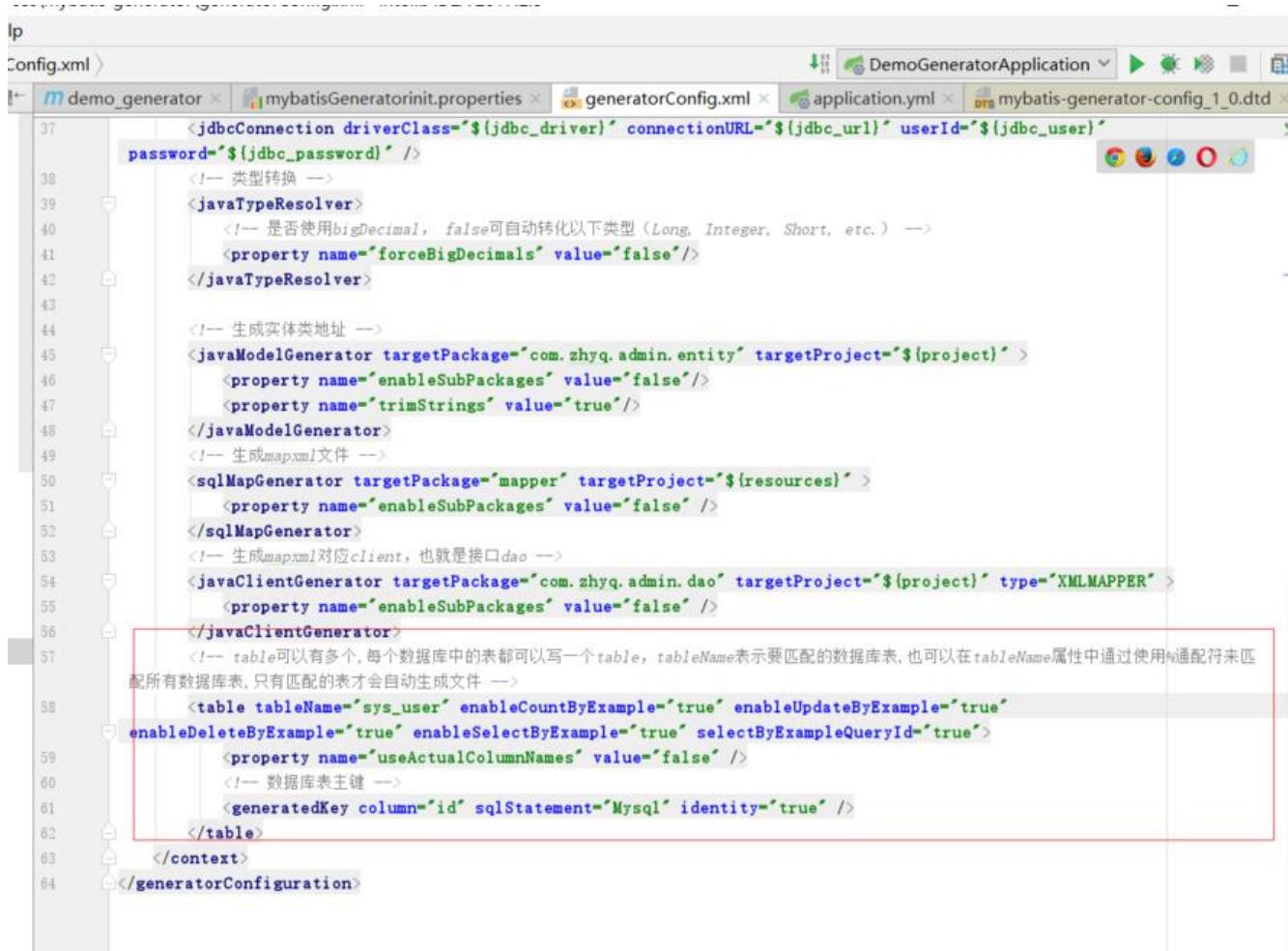
```

```

</javaClientGenerator>
<!-- table可以有多个,每个数据库中的表都可以写一个table, tableName表示要匹配的数据库表
也可以在tableName属性中通过使用%通配符来匹配所有数据库表,只有匹配的表才会自动生成文件 --
<table tableName="sys_user" enableCountByExample="true" enableUpdateByExample="true"
enableDeleteByExample="true" enableSelectByExample="true" selectByExampleQueryId="true">
    <property name="useActualColumnNames" value="false" />
    <!-- 数据库表主键 -->
    <generatedKey column="id" sqlStatement="Mysql" identity="true" />
</table>
</context>
</generatorConfiguration>

```

注意：配置文件中需要注意需要生成哪个表的相应代码需要进行配置，最好生成一次后进行注释，如未注释，会将上次生成的代码覆盖！（具体配置详情都在配置文件中有注释）



```

<jdbcConnection driverClass="${jdbc_driver}" connectionURL="${jdbc_url}" userId="${jdbc_user}"
password="${jdbc_password}" />
<!-- 类型转换 -->
<javaTypeResolver>
    <!-- 是否使用bigDecimal, false可自动转化以下类型 (Long, Integer, Short, etc.) -->
    <property name="forceBigDecimals" value="false"/>
</javaTypeResolver>

<!-- 生成实体类地址 -->
<javaModelGenerator targetPackage="com.zhyq.admin.entity" targetProject="${project}" >
    <property name="enableSubPackages" value="false"/>
    <property name="trimStrings" value="true"/>
</javaModelGenerator>
<!-- 生成mapxml文件 -->
<sqlMapGenerator targetPackage="mapper" targetProject="${resources}" >
    <property name="enableSubPackages" value="false" />
</sqlMapGenerator>
<!-- 生成mapxml对应client, 也就是接口dao -->
<javaClientGenerator targetPackage="com.zhyq.admin.dao" targetProject="${project}" type="XMLMAPPER" >
    <property name="enableSubPackages" value="false" />
</javaClientGenerator>
<!-- table可以有多个,每个数据库中的表都可以写一个table, tableName表示要匹配的数据库表,也可以在tableName属性中通过使用%通配符来匹配所有数据库表,只有匹配的表才会自动生成文件 -->
<table tableName="sys_user" enableCountByExample="true" enableUpdateByExample="true"
enableDeleteByExample="true" enableSelectByExample="true" selectByExampleQueryId="true">
    <property name="useActualColumnNames" value="false" />
    <!-- 数据库表主键 -->
    <generatedKey column="id" sqlStatement="Mysql" identity="true" />
</table>
</context>
</generatorConfiguration>

```

2、在resources下mybatis\_generator下新建mybatisGeneratorinit.properties文件，并在里面加入下内容，该内容主要是为generator.xml中提供相应的配置信息

```

#Mybatis Generator configuration
#dao类和实体类的位置
project \=src/main/java
#mapper文件的位置
resources\=src/main/resources
#根据数据库中的表生成对应的pojo类、dao、 mapper

```

```
jdbc_driver=com.mysql.jdbc.Driver
jdbc_url=jdbc:mysql://localhost:3306/generator
jdbc_user=root
jdbc_password=123
```

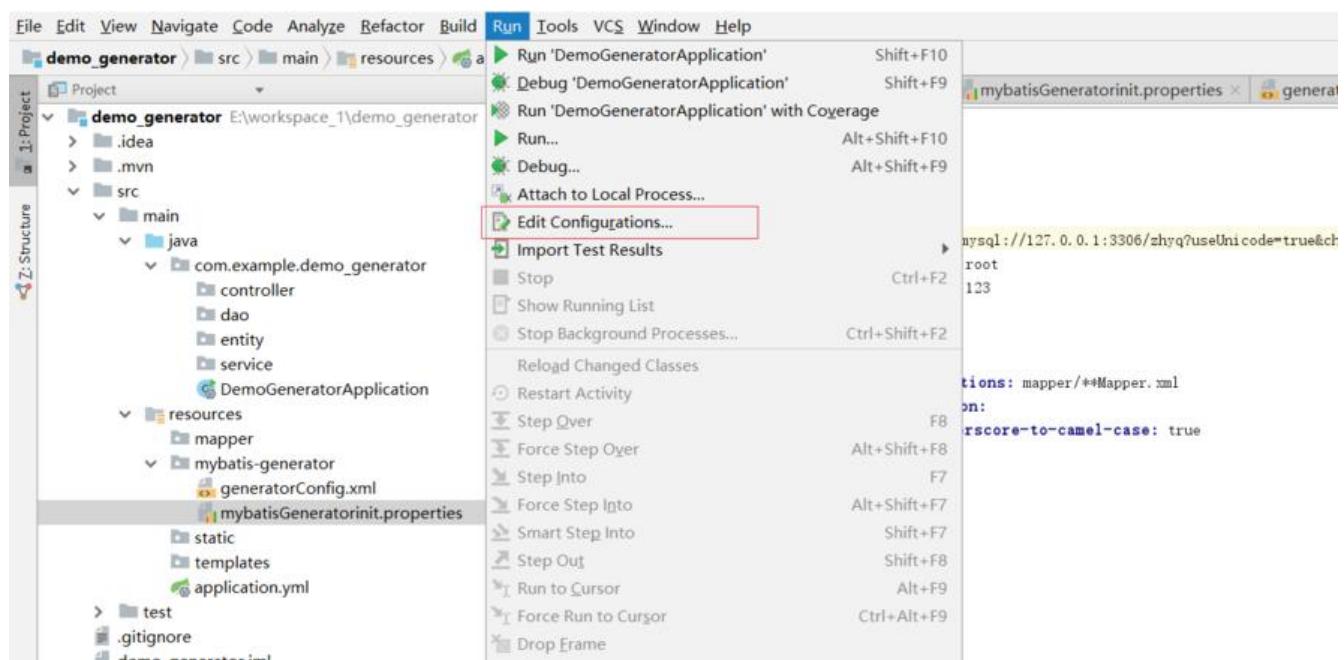
#### 四、在application.yml中增加mybatis的配置信息

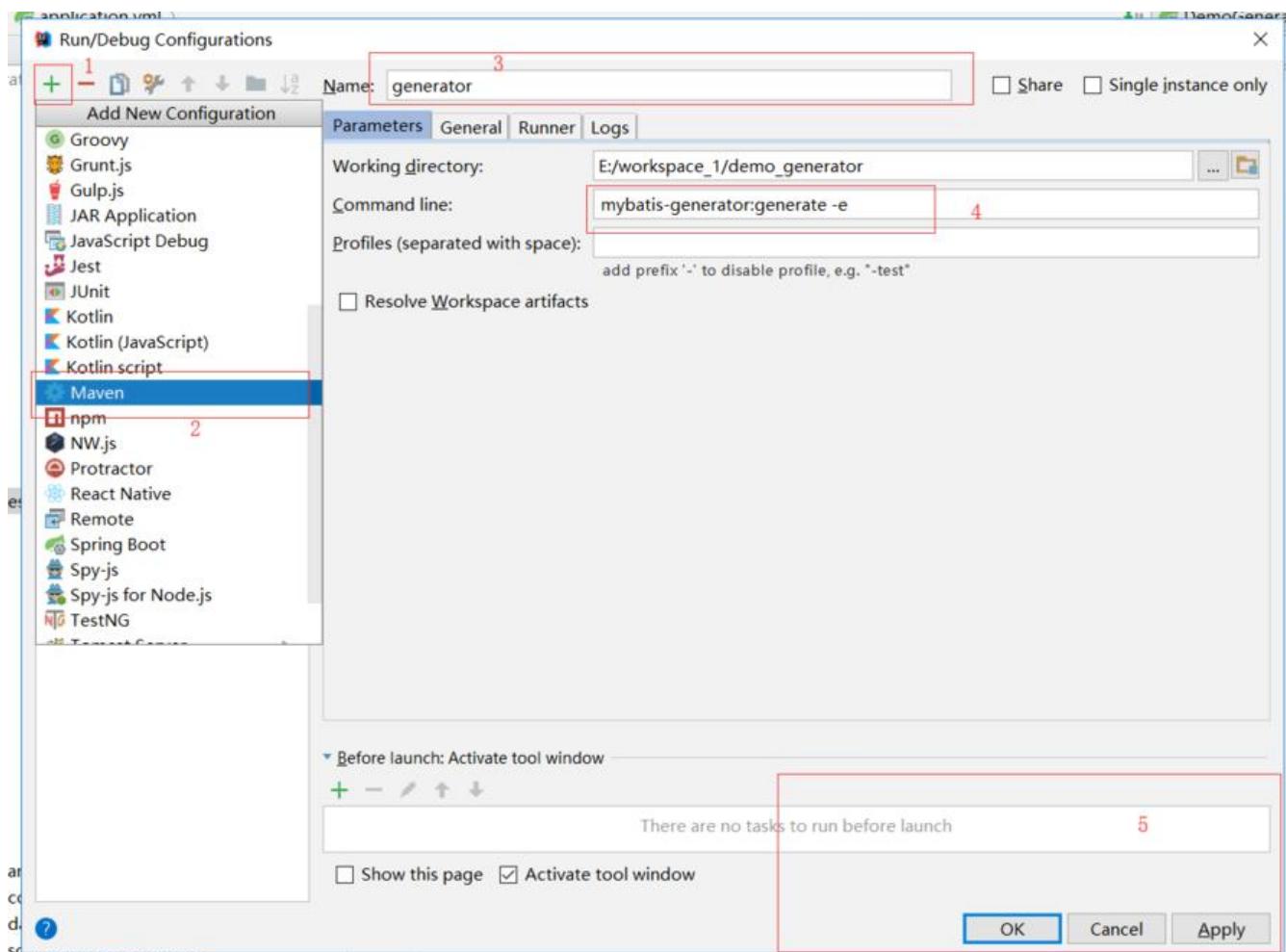
mybatis:

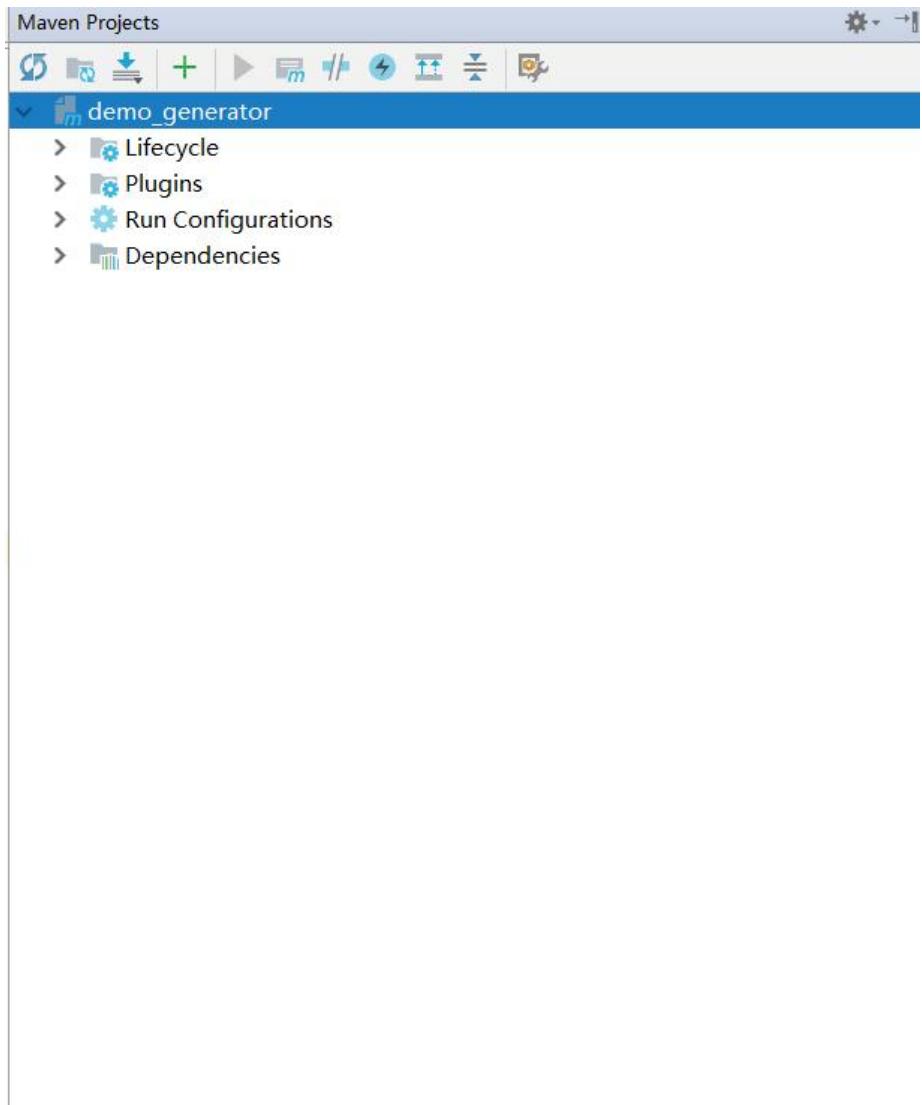
```
  mapper-locations: mapper/*\*Mapper.xml
  configuration:
    map-underscore-to-camel-case: true
```

-----到此配置信息结束-----

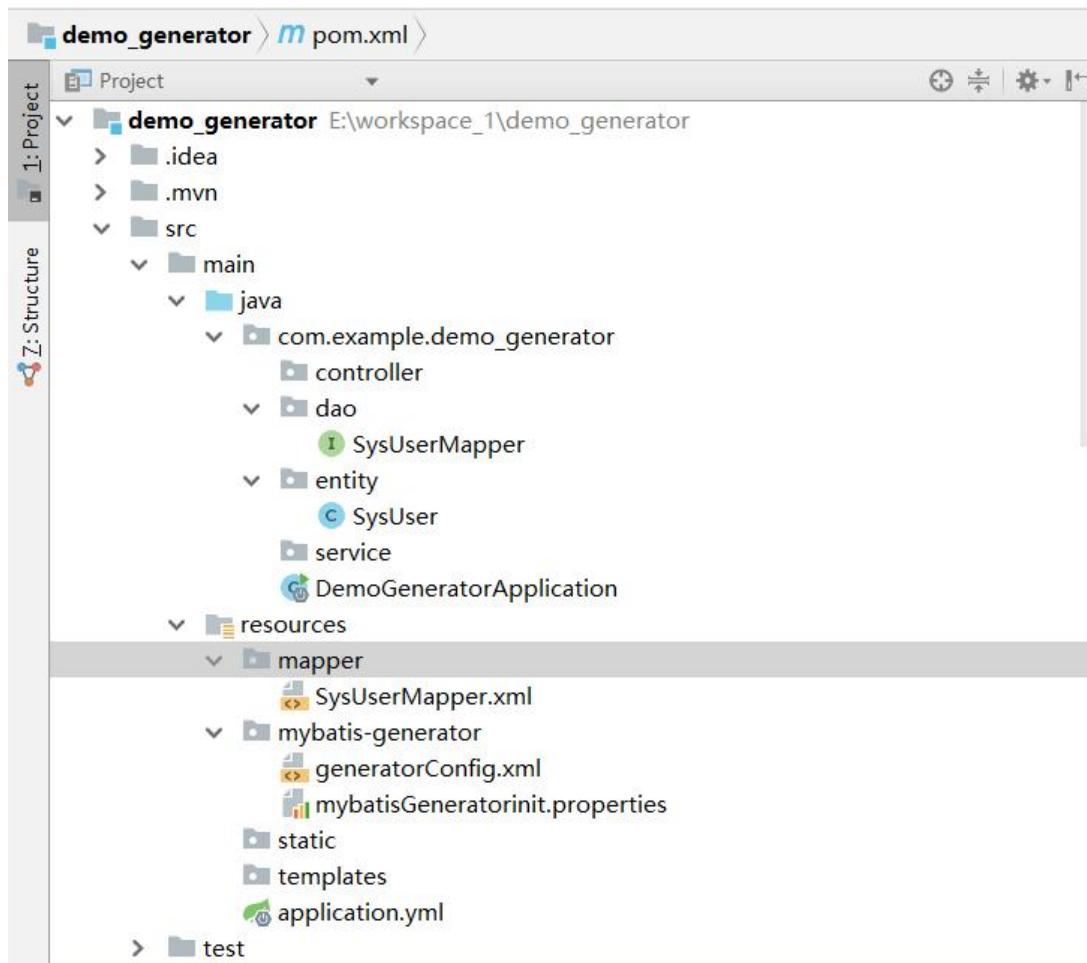
#### 五、通过idea生成相应的代码，配置generator.xml作为maven启动项，按照以下图片步骤可以看到右侧maven多了一个启动项。完成后点击启动项则生成了相应的代码。







注: `mybatis-generator:generate -e -e`为了在生成过程中输出日志  
生成结果如下



简单写一个get请求通过postman进行调试返回结果信息，成功将数据库表数据查出来！

The screenshot shows a POSTMAN request configuration and its response.

Request Details:

- Method: GET
- URL: localhost:8080/getList
- Headers: Authorization, Headers, Body, Pre-request Script, Tests
- Type: No Auth

Response Details:

- Status: 200 OK
- Time: 301 ms
- Body (Pretty):

```
1 * [ {  
2 *   "id": 1,  
3 *   "name": "admin",  
4 *   "password": "123456",  
5 *   "phone": "13569658989",  
6 *   "address": "aaa"  
7 * } ]
```