乐字节教育高级架构课程

正所谓“授人以鱼不如授人以渔”，你们想要的 Java 学习资料来啦！不管你是学生，还是已经步入职场的同行，希望你们都要珍惜眼前的学习机会，奋斗没有终点，知识永不过时。

扫描下方二维码即可领取

乐字节晓啡

乐字节官方交流群

MyBatis逆向工程generator的使用

generator是什么?

是一个逆向工程，用于自动生成mapper.xml，mapper.java，pojo

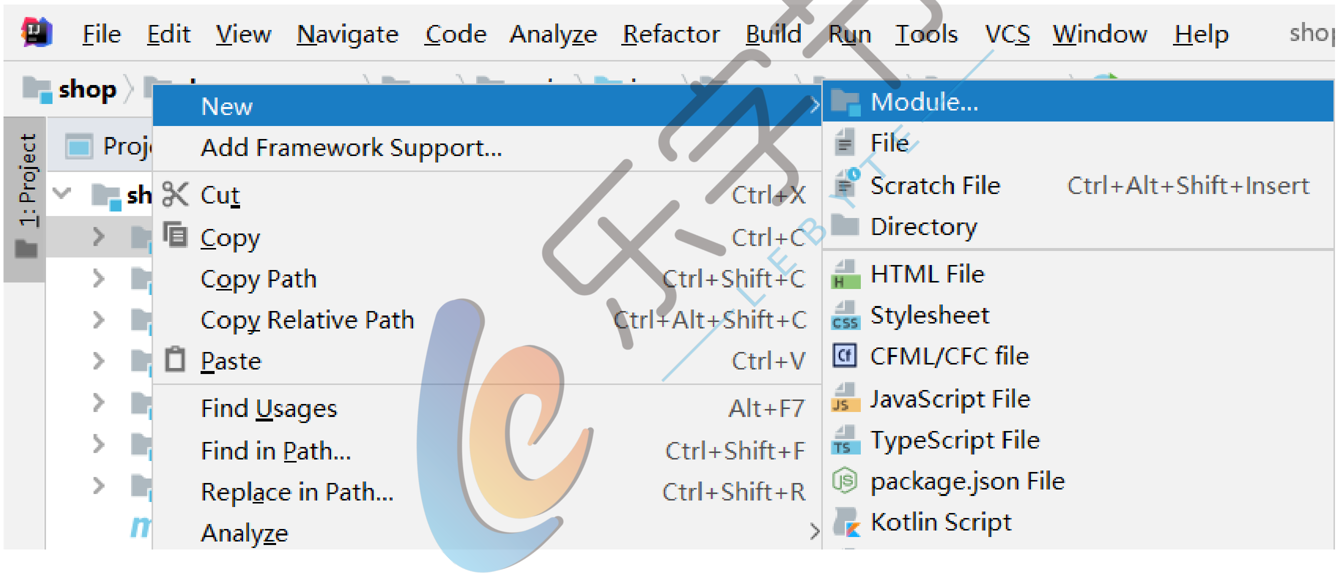
generator能干什么?

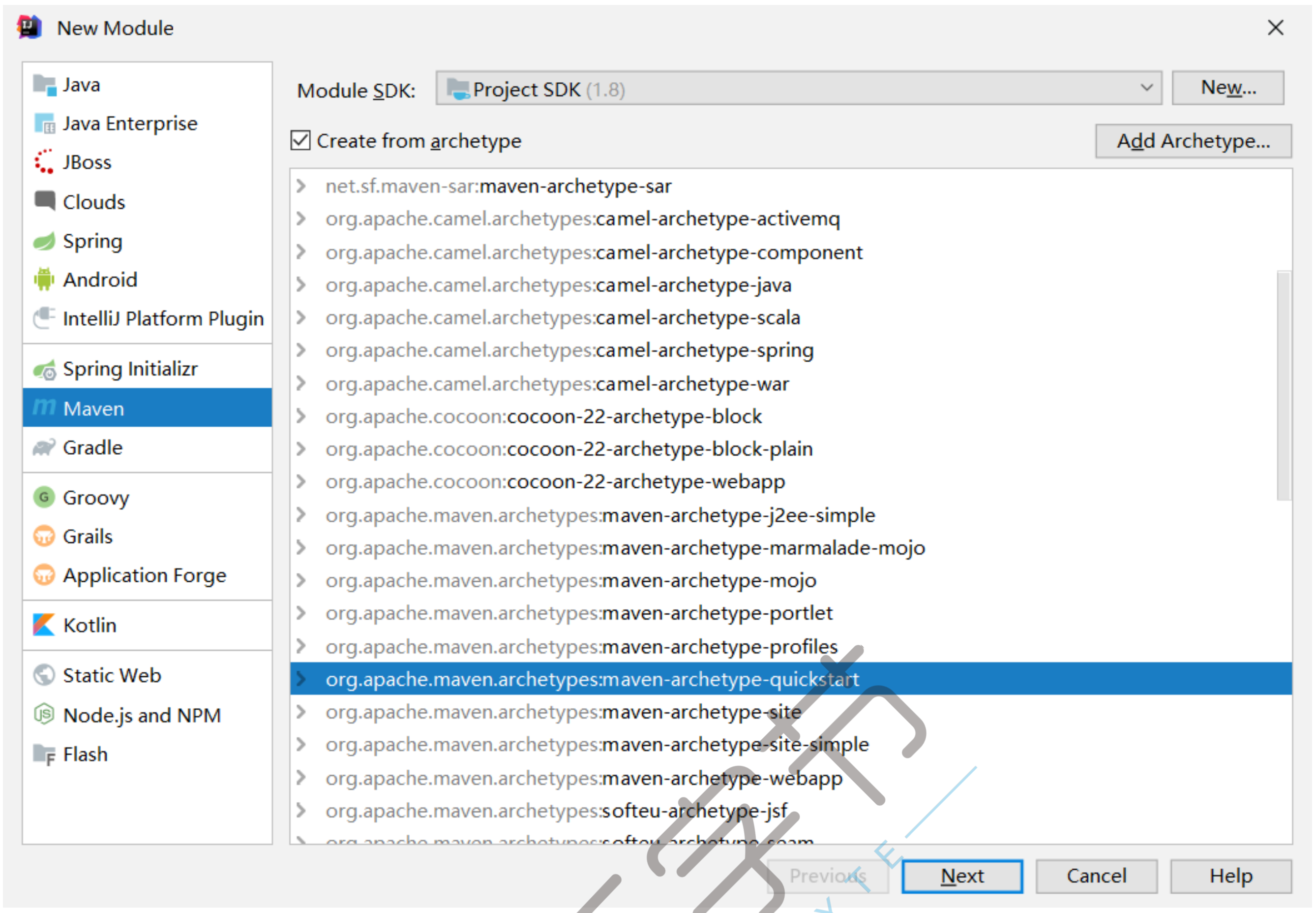
对于单表而言，几乎是一个全能的工具，能很大提高编程效率。更多的关注业务逻辑的实现。

怎么使用?

创建一个generator项目

generator本身和我们商城项目没有关联，所以可以单独新建为一个Project，这边也做成Maven聚合项目里的一个子项目

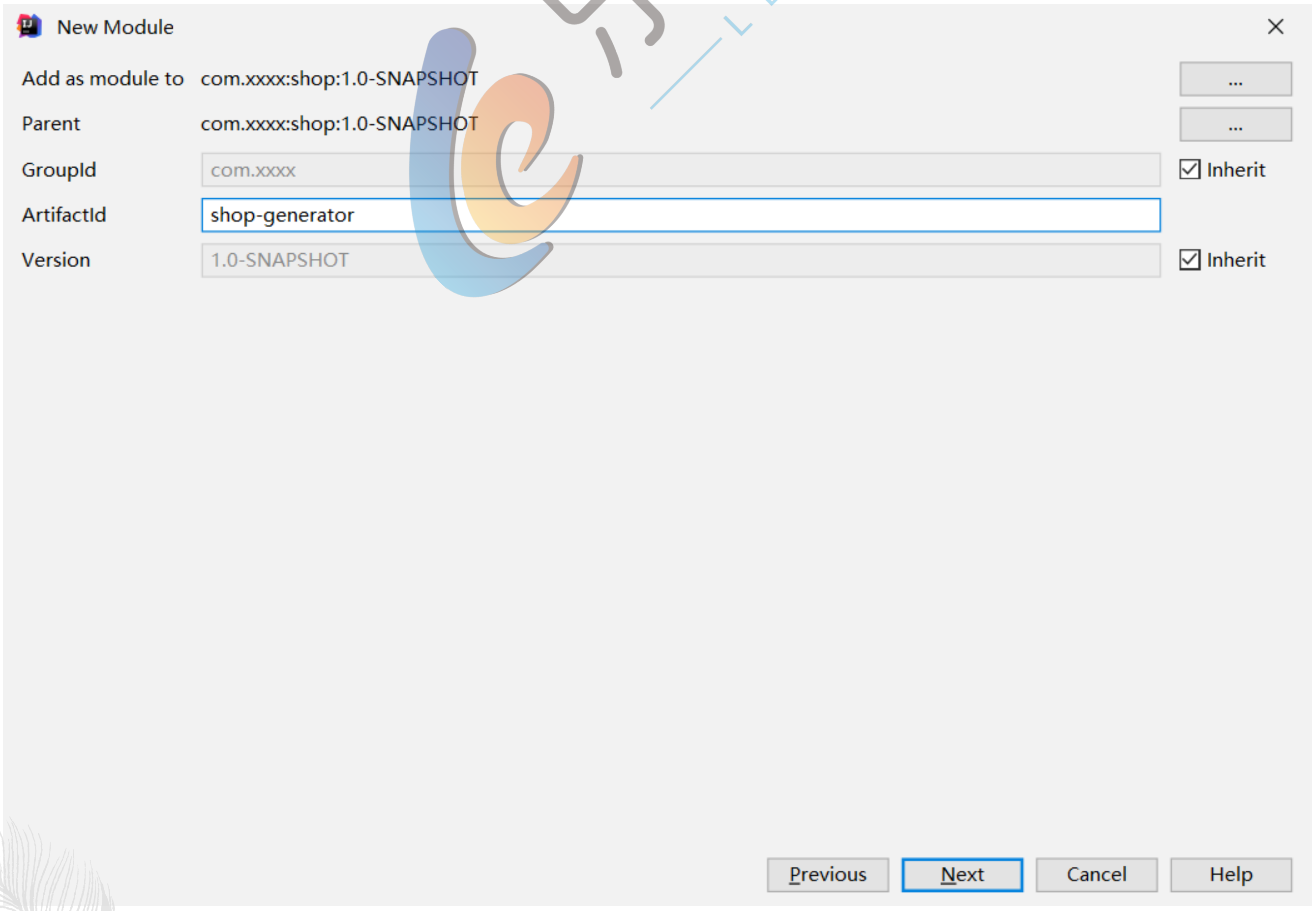




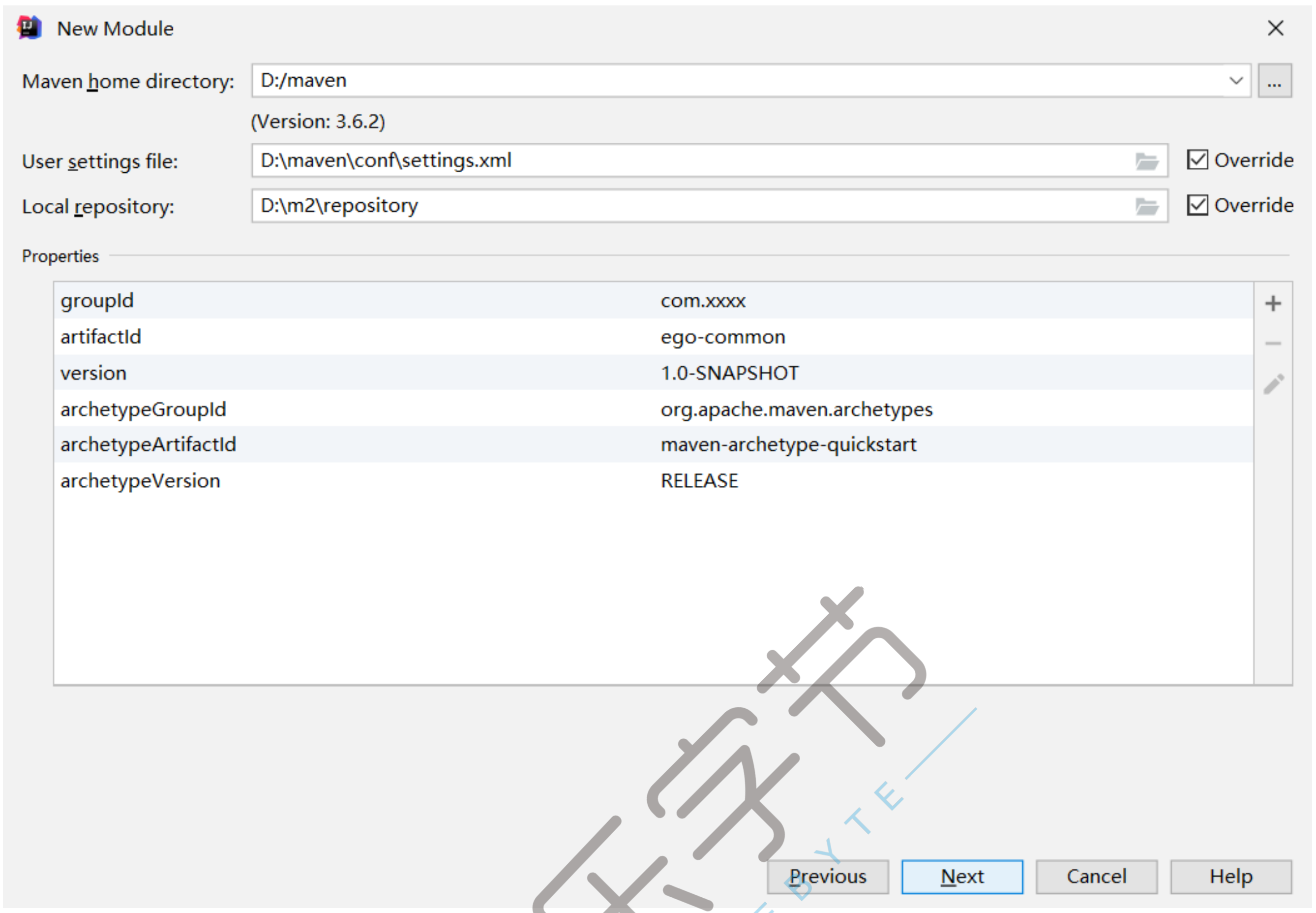
Module SDK: Project SDK (1.8)

☑ Create from archetype

```
> net.sf.maven-sar:maven-archetype-sar
> org.apache.camel.archetypes:camel-archetype-activemq
> org.apache.camel.archetypes:camel-archetype-component
> org.apache.camel.archetypes:camel-archetype-java
> org.apache.camel.archetypes:camel-archetype-scala
> org.apache.camel.archetypes:camel-archetype-spring
> org.apache.camel.archetypes:camel-archetype-war
> org.apache.cocoon:cocoon-22-archetype-block
> org.apache.cocoon:cocoon-22-archetype-block-plain
> org.apache.cocoon:cocoon-22-archetype-webapp
> org.apache.maven.archetypes:maven-archetype-j2ee-simple
> org.apache.maven.archetypes:maven-archetype-marmalade-mojo
> org.apache.maven.archetypes:maven-archetype-mojo
> org.apache.maven.archetypes:maven-archetype-portlet
> org.apache.maven.archetypes:maven-archetype-profiles
> org.apache.maven.archetypes:maven-archetype-quickstart
> org.apache.maven.archetypes:maven-archetype-site
> org.apache.maven.archetypes:maven-archetype-site-simple
> org.apache.maven.archetypes:maven-archetype-webapp
> org.apache.maven.archetypes:softeu-archetype-jsf
> org.apache.maven.archetypes:softeu-archetype-seam
```



Add as module to	com.xxxx:shop:1.0-SNAPSHOT	
Parent	com.xxxx:shop:1.0-SNAPSHOT	
GroupId	com.xxxx	☑ Inherit
ArtifactId	shop-generator	
Version	1.0-SNAPSHOT	☑ Inherit


New Module

Maven home directory: D:/maven

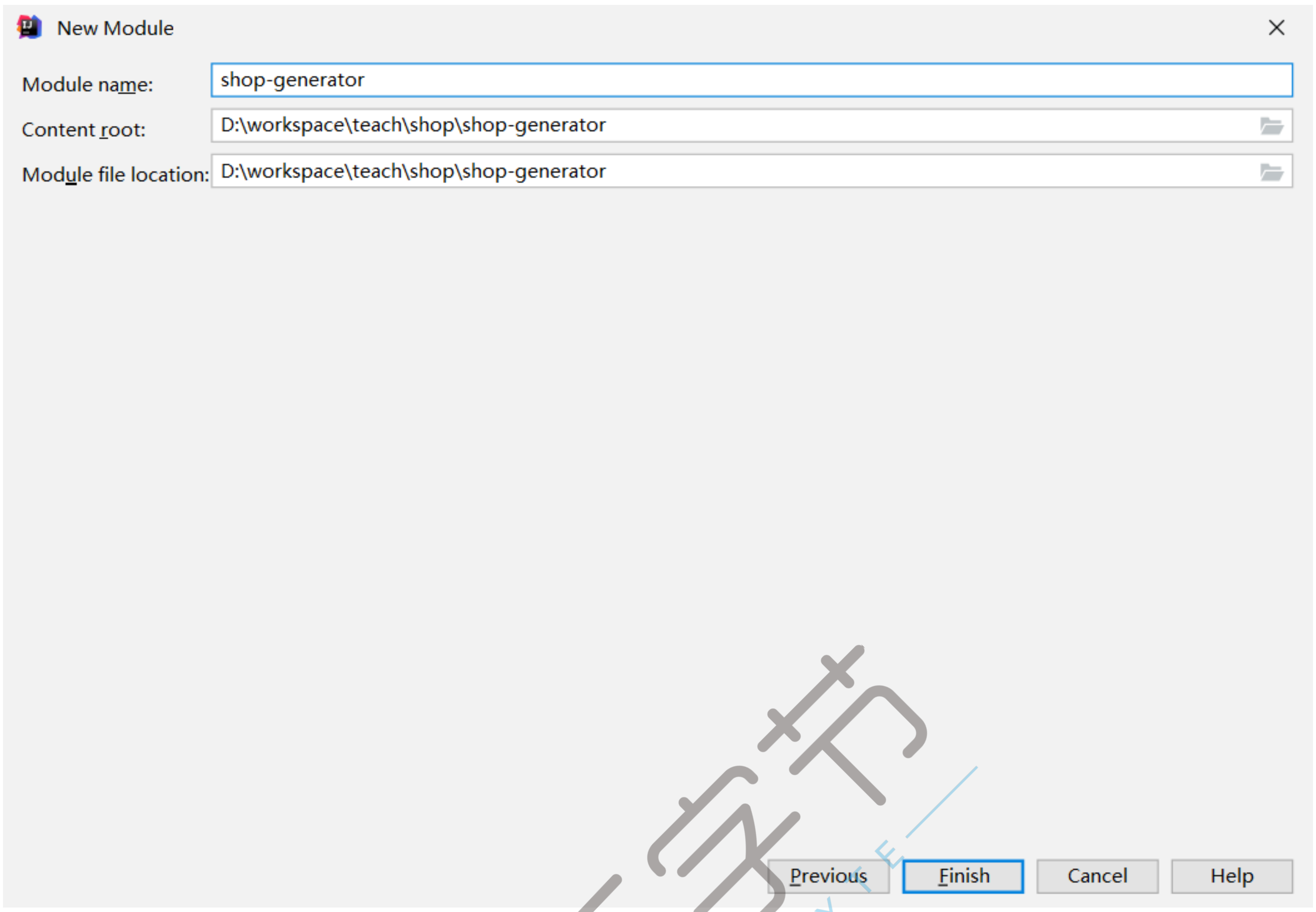
(Version: 3.6.2)

User settings file: D:\maven\conf\settings.xml ☑ Override

Local repository: D:\m2\repository ☑ Override

Properties

groupId	com.xxxx
artifactId	ego-common
version	1.0-SNAPSHOT
archetypeGroupId	org.apache.maven.archetypes
archetypeArtifactId	maven-archetype-quickstart
archetypeVersion	RELEASE



Module name: shop-generator
Content root: D:\workspace\teach\shop\shop-generator
Module file location: D:\workspace\teach\shop\shop-generator

添加依赖

```
<?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>com.xxxx</groupId>
  <artifactId>shop-generator</artifactId>
  <version>1.0-SNAPSHOT</version>

  <!-- 继承 shop-parent 依赖 -->
  <parent>
    <groupId>com.xxxx</groupId>
    <artifactId>shop</artifactId>
    <version>1.0-SNAPSHOT</version>
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
    <!-- mybatis 依赖(让生成的 pojo 和 mapper 不缺少注解相关类型) -->
    <dependency>
      <groupId>org.mybatis.spring.boot</groupId>
      <artifactId>mybatis-spring-boot-starter</artifactId>
    </dependency>
    <!-- mybatis generator core 依赖 -->
    <dependency>
      <groupId>org.mybatis.generator</groupId>
      <artifactId>mybatis-generator-core</artifactId>
      <version>1.3.7</version>
    </dependency>
  </dependencies>

  <!-- build 标签常用于添加插件及编译配置 -->
  <build>
    <plugins>
      <!-- mybatis generator plugin 依赖 -->
      <plugin>
        <groupId>org.mybatis.generator</groupId>
        <artifactId>mybatis-generator-maven-plugin</artifactId>
        <version>1.3.7</version>
        <configuration>
          <verbose>true</verbose>
          <!-- 是否覆盖 -->
          <overwrite>true</overwrite>
          <!-- 自动生成的配置 -->
          <configurationFile>src/main/resources/mybatis-generator.xml</configurationFile>
        </configuration>
        <dependencies>
          <!-- mysql 数据库依赖 -->
          <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql.version}</version>
          </dependency>
          <!-- mybatis generator core 依赖 -->
          <dependency>
            <groupId>org.mybatis.generator</groupId>
            <artifactId>mybatis-generator-core</artifactId>
            <version>1.3.7</version>
          </dependency>
          <!-- 将项目打包至本地仓库并添加依赖 -->
          <dependency>
            <groupId>com.xxxx</groupId>
            <artifactId>shop-generator</artifactId>
            <version>1.0-SNAPSHOT</version>
          </dependency>
        </dependencies>
      </plugin>
    </plugins>
  </build>
</project>
```

编写mybatis-generator.xml配置文件

```
<?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>
    <context id="MysqlTables" targetRuntime="MyBatis3">
        <!-- 生成的Java文件的编码 -->
        <property name="javaFileEncoding" value="UTF-8"/>
        <!-- 增加Models ToStirng方法 -->
        <plugin type="org.mybatis.generator.plugins.ToStringPlugin"/>
        <!-- 增加Models Serializable实现 -->
        <plugin type="org.mybatis.generator.plugins.SerializablePlugin"/>

        <!-- 分页插件 -->
        <!-- 在example类中增 page 属性，并在mapper.xml的查询中加入page !=null 时的查询 -->
        <!-- <plugin type="org.mybatis.generator.plugins.MySQLPagerPlugin" /> -->
        <!-- 在example类中增 offset和limit属性，并在mapper.xml的查询中加入limit ${offset} ,
${limit} 提供在offset和limit>0时的查询 -->
        <!-- <plugin type="org.mybatis.generator.plugins.MySQLPaginationPlugin"></plugin> -
->


        <!-- 是否去除自动生成的注释 true：是 ： false:否 -->
        <!-- type指定生成注释使用的对象 -->
        <commentGenerator type="com.xxxx.generator.ShopCommentGenerator">
            <property name="suppressAllComments" value="false"/>
        </commentGenerator>

        <!-- mysql数据库连接配置 -->
        <jdbcConnection driverClass="com.mysql.cj.jdbc.Driver"
                        connectionURL="jdbc:mysql://localhost:3306/shop?
useUnicode=true&amp;characterEncoding=UTF-
8&amp;serverTimezone=Asia/Shanghai&amp;tinyInt1isBit=false"
                        userId="root" password="root">
        </jdbcConnection>

        <!--
            是否忽略BigDecimals 非必填项
                自动生成Java对象的时候，会根据number类型的长度不同生成不同的数据类型
                    number长度   Java类型
                    1~4          Short
                    5~9          Integer
                    10~18        Long
                    18+          BigDecimal
         -->
        <javaTypeResolver>
            <property name="forceBigDecimals" value="false"/>
        </javaTypeResolver>

        <!-- 以下内容，需要改动 -->
        <!-- java类生成的位置  -->
        <javaModelGenerator targetPackage="com.xxxx.generator.pojo"
targetProject="src/main/java">
            <!-- 在targetPackage的基础上，根据数据库的schema再生成一层package，最终生成的类放在这个
package下，默认为false -->
            <property name="enableSubPackages" value="true"/>
            <!-- 从数据库返回的值去除前后空格 -->
            <property name="trimStrings" value="true"/>
        </javaModelGenerator>

        <!-- *Mapper.xml配置文件生成的位置 -->
        <sqlMapGenerator targetPackage="com.xxxx.generator.mapper"
targetProject="src/main/java">
            <!-- 是否让schema作为包后缀 -->
            <property name="enableSubPackages" value="true"/>
        </sqlMapGenerator>

        <!-- java mapper接口生成的位置(interface) -->
        <javaClientGenerator type="XMLMAPPER" targetPackage="com.xxxx.generator.mapper"
targetProject="src/main/java">
            <!-- 是否让schema作为包后缀 -->
            <property name="enableSubPackages" value="true"/>
        </javaClientGenerator>

        <!--
            指定数据库表
                tableName数据库表名
                domainObjectName生成的实体类名
                是否需要mapper配置文件加入sql的where条件查询,需要将enableCountByExample等设为true,
会生成一个对应domainObjectName的Example类
         -->
        <table tableName="t_order_goods" domainObjectName="OrderGoods"
               enableCountByExample="true" enableDeleteByExample="true"
               enableSelectByExample="true" enableUpdateByExample="true">
            <!-- 用于insert时，返回主键的编号 -->
            <generatedKey column="id" sqlStatement="MySql" identity="true"/>
        </table>
    </context>
</generatorConfiguration>
```

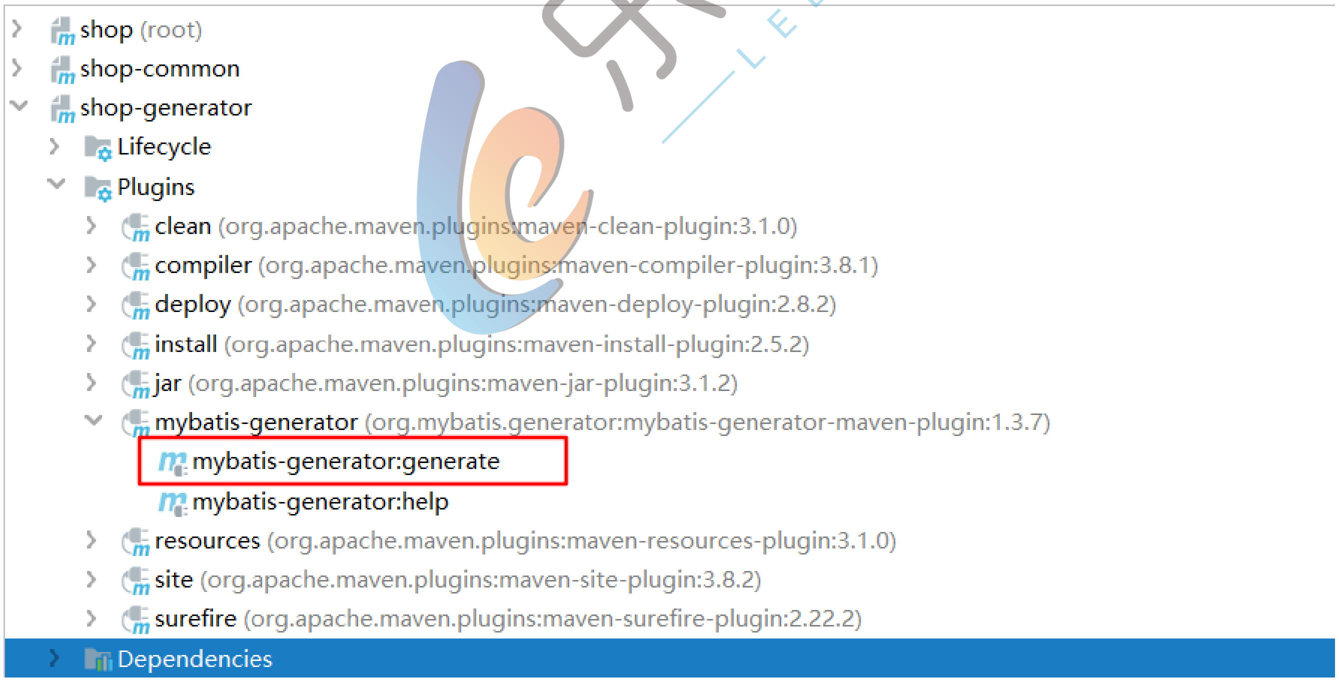
ShopCommentGenerator工具类

```
package com.xxxx.generator;

import org.mybatis.generator.api.CommentGenerator;
import org.mybatis.generator.api.IntrospectedColumn;
import org.mybatis.generator.api.IntrospectedTable;
import org.mybatis.generator.api.dom.java.CompilationUnit;
import org.mybatis.generator.api.dom.java.Field;
import org.mybatis.generator.api.dom.java.FullyQualifiedJavaType;
import org.mybatis.generator.api.dom.java.InnerClass;
import org.mybatis.generator.api.dom.java.InnerEnum;
import org.mybatis.generator.api.dom.java.JavaElement;
```

```java
import org.mybatis.generator.api.dom.java.Method;
import org.mybatis.generator.api.dom.java.TopLevelClass;
import org.mybatis.generator.api.dom.xml.XmlElement;
import org.mybatis.generator.config.PropertyRegistry;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Properties;
import java.util.Set;

import static org.mybatis.generator.internal.util.StringUtility.isTrue;

/**
 * 生成注释配置类
 */
public class ShopCommentGenerator implements CommentGenerator {

    /**
     * properties属性，即配置在 commentGenerator 标签之内的 Property 标签
     */
    private Properties properties;
    /**
     * properties配置文件
     */
    private Properties systemPro;
    /*
     * 是否生成日期
     */
    private boolean suppressDate;
    /**
     * 是否生成注释
     */
    private boolean suppressAllComments;
    /**
     * 日期格式
     */
    private String currentDateStr;

    public ShopCommentGenerator() {
        super();
        properties = new Properties();
        systemPro = System.getProperties();
        suppressDate = false;
        suppressAllComments = false;
        currentDateStr =
DateTimeFormatter.ofPattern("yyyy/MM/dd").format(LocalDateTime.now());
    }

    /**
     * 此方法返回格式化的日期字符串以包含在Javadoc标记中和XML注释。
     *
     * @return
     */
    protected String getDateString() {
        String result = null;
        if (!suppressDate) {
            result = currentDateStr;
        }
        return result;
    }

    /**
     * 从该配置中的任何属性添加此实例的属性CommentGenerator配置。
     *
     * @param properties
     */
    @Override
    public void addConfigurationProperties(Properties properties) {
        this.properties.putAll(properties);
        suppressDate =
isTrue(properties.getProperty(PropertyRegistry.COMMENT_GENERATOR_SUPPRESS_DATE));
        suppressAllComments =
isTrue(properties.getProperty(PropertyRegistry.COMMENT_GENERATOR_SUPPRESS_ALL_COMMENTS));
    }

    /**
     * 为字段添加注释
     *
     * @param field
     * @param introspectedTable
     * @param introspectedColumn
     */
    @Override
    public void addFieldComment(Field field, IntrospectedTable introspectedTable,
IntrospectedColumn introspectedColumn) {
        if (suppressAllComments)
            return;
        StringBuilder sb = new StringBuilder();
        field.addJavaDocLine("/**");
        sb.append(" * ");
        sb.append(introspectedColumn.getRemarks());
        field.addJavaDocLine(sb.toString().replace("\n", " "));
        field.addJavaDocLine(" */");
    }

    /**
     * Java 属性注释
     *
     * @param field
     * @param introspectedTable
     */
    @Override
    public void addFieldComment(Field field, IntrospectedTable introspectedTable) {
        if (suppressAllComments)
            return;
        StringBuilder sb = new StringBuilder();
        field.addJavaDocLine("/**");
        sb.append(" * ");
        sb.append(introspectedTable.getFullyQualifiedTable());
        field.addJavaDocLine(sb.toString().replace("\n", " "));
        field.addJavaDocLine(" */");
    }

    /**
     * 为模型类添加注释
     *
     * @param topLevelClass
     * @param introspectedTable
     */
    @Override
    public void addModelClassComment(TopLevelClass topLevelClass, IntrospectedTable
introspectedTable) {
        if (suppressAllComments) {
            return;
        }
        topLevelClass.addJavaDocLine("/**");
        topLevelClass.addJavaDocLine(" * @author zhoubin ");
        topLevelClass.addJavaDocLine(" * @since 1.0.0");
        topLevelClass.addJavaDocLine(" */");
    }

    /**
     * Java类的类注释
     *
     * @param innerClass
     * @param introspectedTable
     */
    @Override
    public void addClassComment(InnerClass innerClass, IntrospectedTable introspectedTable)
{
        if (suppressAllComments) {
            return;
        }
        // 获取表注释
        String remarks = introspectedTable.getRemarks();
        innerClass.addJavaDocLine("/**");
        innerClass.addJavaDocLine("/* "+remarks);
        innerClass.addJavaDocLine(" * @author zhoubin ");
        innerClass.addJavaDocLine(" * @since 1.0.0");
        innerClass.addJavaDocLine(" */");
    }

    /**
     * 为类添加注释
     *
     * @param innerClass
     * @param introspectedTable
     * @param b
     */
    @Override
    public void addClassComment(InnerClass innerClass, IntrospectedTable introspectedTable, 
boolean b) {
        // 获取表注释
        String remarks = introspectedTable.getRemarks();
        innerClass.addJavaDocLine("/**");
        innerClass.addJavaDocLine("/* "+remarks);
        innerClass.addJavaDocLine(" * @author zhoubin ");
        innerClass.addJavaDocLine(" * @since 1.0.0");
        innerClass.addJavaDocLine(" */");
    }

    /**
     * 为枚举添加注释
     *
     * @param innerEnum
     * @param introspectedTable
     */
    @Override
    public void addEnumComment(InnerEnum innerEnum, IntrospectedTable introspectedTable) {
        if (suppressAllComments)
            return;
        StringBuilder sb = new StringBuilder();
        innerEnum.addJavaDocLine("/**");
        sb.append(" * ");
        sb.append(introspectedTable.getFullyQualifiedTable());
        innerEnum.addJavaDocLine(sb.toString().replace("\n", " "));
        innerEnum.addJavaDocLine(" */");
    }

    /**
     * 给getter方法加注释
     *
     * @param method
     * @param introspectedTable
     * @param introspectedColumn
     */
    @Override
    public void addGetterComment(Method method, IntrospectedTable introspectedTable, 
IntrospectedColumn introspectedColumn) {
    }

    /**
     * 给setter方法加注释
     *
     * @param method
     * @param introspectedTable
     * @param introspectedColumn
     */
    @Override
    public void addSetterComment(Method method, IntrospectedTable introspectedTable, 
IntrospectedColumn introspectedColumn) {
    }
    /**
     * 普通方法的注释，这里主要是XXXMapper.java里面的接口方法的注释
     *
     * @param method
     * @param introspectedTable
     */
    @Override
    public void addGeneralMethodComment(Method method, IntrospectedTable introspectedTable)
{
    }

    /**
     * 给Java文件加注释，这个注释是在文件的顶部，也就是package上面。
     *
     * @param compilationUnit
     */
    @Override
    public void addJavaFileComment(CompilationUnit compilationUnit) {
    }

    /**
     * Mybatis的Mapper.xml文件里面的注释
     *
     * @param xmlElement
     */
    @Override
    public void addComment(XmlElement xmlElement) {
    }

    /**
     * 此方法为其添加了自定义javadoc标签。
     *
     * @param javaElement
     * @param markAsDoNotDelete
     */
    protected void addJavadocTag(JavaElement javaElement, boolean markAsDoNotDelete) {
    }

    /**
     * 为调用此方法作为根元素的第一个子节点添加注释。
     *
     * @param xmlElement
     */
    @Override
    public void addRootComment(XmlElement xmlElement) {
    }

    @Override
    public void addGeneralMethodAnnotation(Method method, IntrospectedTable
introspectedTable, Set<FullyQualifiedJavaType> set) {
    }
    @Override
    public void addGeneralMethodAnnotation(Method method, IntrospectedTable
introspectedTable, IntrospectedColumn introspectedColumn, Set<FullyQualifiedJavaType> set)
{
    }

    @Override
    public void addFieldAnnotation(Field field, IntrospectedTable introspectedTable,
Set<FullyQualifiedJavaType> set) {
    }

    @Override
    public void addFieldAnnotation(Field field, IntrospectedTable introspectedTable,
IntrospectedColumn introspectedColumn, Set<FullyQualifiedJavaType> set) {
    }

    @Override
    public void addClassAnnotation(InnerClass innerClass, IntrospectedTable
introspectedTable, Set<FullyQualifiedJavaType> set) {
    }

}
```

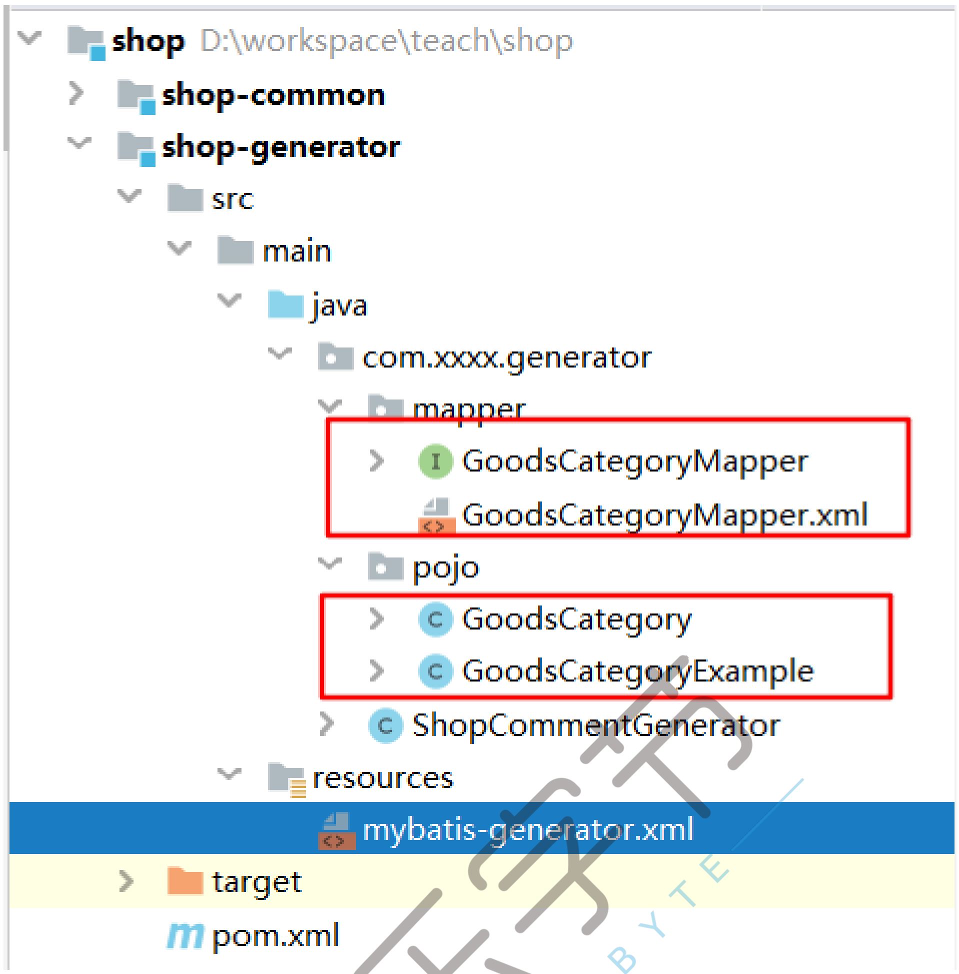
执行

执行前一定要先install，否则有可能报错



```
> shop (root)
> shop-common
v shop-generator
    > Lifecycle
    v Plugins
        > clean (org.apache.maven.plugins:maven-clean-plugin:3.1.0)
        > compiler (org.apache.maven.plugins:maven-compiler-plugin:3.8.1)
        > deploy (org.apache.maven.plugins:maven-deploy-plugin:2.8.2)
        > install (org.apache.maven.plugins:maven-install-plugin:2.5.2)
        > jar (org.apache.maven.plugins:maven-jar-plugin:3.1.2)
        v mybatis-generator (org.mybatis.generator:mybatis-generator-maven-plugin:1.3.7)
              mybatis-generator:generate
              mybatis-generator:help
        > resources (org.apache.maven.plugins:maven-resources-plugin:3.1.0)
        > site (org.apache.maven.plugins:maven-site-plugin:3.8.2)
        > surefire (org.apache.maven.plugins:maven-surefire-plugin:2.22.2)
    > Dependencies
```

结果



```
shop  D:\workspace\teach\shop
  shop-common
  shop-generator
    src
      main
        java
          com.xxxx.generator
            mapper
              GoodsCategoryMapper
              GoodsCategoryMapper.xml
            pojo
              GoodsCategory
              GoodsCategoryExample
            ShopCommentGenerator
        resources
          mybatis-generator.xml
    target
    pom.xml
```