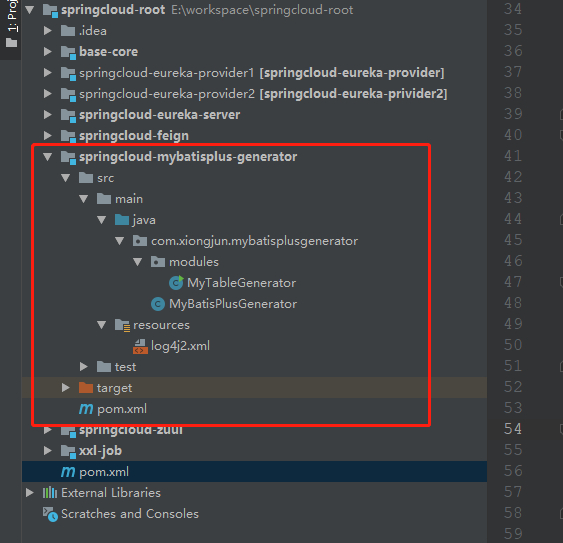
本文是根据batisplus-generator工具实现自动生成代码的功能。

1. 首先得引入jar

<?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">  
 <parent>  
 <artifactId>springcloud-root</artifactId>  
 <groupId>springcloud-root</groupId>  
 <version>1.0-SNAPSHOT</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>springcloud-mybatisplus-generator</artifactId>  
  
 <properties></properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>3.5.1</version>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.48</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.velocity</groupId>  
 <artifactId>velocity</artifactId>  
 <version>1.7</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-generator</artifactId>  
 <version>3.1.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.freemarker</groupId>  
 <artifactId>freemarker</artifactId>  
 <version>2.3.28</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.logging.log4j</groupId>  
 <artifactId>log4j-api</artifactId>  
 <version>2.11.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.logging.log4j</groupId>  
 <artifactId>log4j-core</artifactId>  
 <version>2.11.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.logging.log4j</groupId>  
 <artifactId>log4j-slf4j-impl</artifactId>  
 <version>2.11.2</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>5.1.4.RELEASE</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.5.1</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 <encoding>UTF-8</encoding>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

项目结构：



Log4j2.xml:

<?xml version="1.0" encoding="UTF-8"?>  
<Configuration status="off">  
 <Appenders>  
 <Console name="Console" target="SYSTEM\_OUT">  
 <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss,SSS} {thread-%T} [%F:%L] %p:%m%n" />  
 </Console>  
 </Appenders>  
 <Loggers>  
 <Root level="debug">  
 <AppenderRef ref="Console" />  
 </Root>  
 </Loggers>  
</Configuration>

MyBatisPlusGenerator.java:

import com.baomidou.mybatisplus.annotation.DbType;

import com.baomidou.mybatisplus.generator.AutoGenerator;

import com.baomidou.mybatisplus.generator.InjectionConfig;

import com.baomidou.mybatisplus.generator.config.\*;

import com.baomidou.mybatisplus.generator.config.converts.MySqlTypeConvert;

import com.baomidou.mybatisplus.generator.config.po.TableInfo;

import com.baomidou.mybatisplus.generator.config.rules.NamingStrategy;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class MyBatisPlusGenerator {

private static final String DEFAULT\_OUTPUT\_DIR = "E:/workspace\_code\_test/";

//mabatis的xml文件生成后所存放的包名

private static final String MAPPER\_FILE\_PACKAGE = "mapper";

private static final String CONTROLLER\_PACKAGE = "controller";

//实体类继承的父类，需要提前创建好

private static final String SUPER\_ENTITY\_CLASS = "com.xiongjun.core.base.BasePO";

//service类继承的父类，需要提前创建好

private static final String SUPER\_SERVICE\_CLASS = "com.xiongjun.core.base.BaseService";

//service实现类继承的父类，需要提前创建好

private static final String SUPER\_SERVICE\_IMPL\_CLASS = "com.xiongjun.core.base.BaseServiceImpl";

//数据库驱动名称，跟引入的mysql.jar包版本有关

private static final String DB\_DRIVER\_NAME = "com.mysql.jdbc.Driver";

//数据库地址，跟引入的mysql.jar包版本有关

private static final String DB\_URL = "jdbc:mysql://10.10.1.12:3306/xxl\_crud?useUnicode=true&amp;" +

"characterEncoding=UTF-8";

//数据库登陆用户名

private static final String DB\_USERNAME = "root";

//数据库登陆密码

private static final String DB\_PASS\_WORD = "ubt83474428!@";

protected String outputDir = DEFAULT\_OUTPUT\_DIR;

protected List<String> tableNames;

protected String entityPackage;

protected String mapperPackage;

protected String servicePackage;

protected String serviceImplPackage;

public void generate() {

// 代码生成器

AutoGenerator mpg = new AutoGenerator();

// 全局配置

GlobalConfig gc = new GlobalConfig();

gc.setOutputDir(outputDir);//代码生成路径

gc.setFileOverride(true);

gc.setActiveRecord(true);// 不需要ActiveRecord特性的请改为false

gc.setEnableCache(false);// XML 二级缓存

gc.setBaseResultMap(true);// XML ResultMap

gc.setBaseColumnList(true);// XML columList

// .setKotlin(true) 是否生成 kotlin 代码

gc.setServiceName("%sService");

gc.setServiceImplName("%sServiceImpl");

gc.setAuthor("ubtech");

mpg.setGlobalConfig(gc);

// 数据源配置

DataSourceConfig dsc = new DataSourceConfig();

dsc.setDbType(DbType.MYSQL);

dsc.setTypeConvert(new MySqlTypeConvert());

dsc.setDriverName(DB\_DRIVER\_NAME);

dsc.setUsername(DB\_USERNAME);

dsc.setPassword(DB\_PASS\_WORD);

dsc.setUrl(DB\_URL);

mpg.setDataSource(dsc);

// 包配置

PackageConfig pc = new PackageConfig();

pc.setParent("");

pc.setEntity(entityPackage);

pc.setService(servicePackage);

pc.setServiceImpl(serviceImplPackage);

pc.setMapper(mapperPackage);

pc.setController(CONTROLLER\_PACKAGE);

pc.setXml(MAPPER\_FILE\_PACKAGE);

mpg.setPackageInfo(pc);

// 策略配置

StrategyConfig strategy = new StrategyConfig();

// strategy.setCapitalMode(true);// 全局大写命名 ORACLE 注意

strategy.setTablePrefix(new String[]{"ucs\_"});// 此处可以修改为您的表前缀

strategy.setNaming(NamingStrategy.underline\_to\_camel);// 表名生成策略

if (tableNames != null && !tableNames.isEmpty()) {

strategy.setInclude(tableNames.toArray(new String[tableNames.size()])); // 需要生成的表

}

// strategy.setExclude(new String[] { "example" }); // 排除生成的表

// strategy.setExclude(new String[]{"test"}); // 排除生成的表

// 自定义实体父类

strategy.setSuperEntityClass(SUPER\_ENTITY\_CLASS);

// 自定义实体，公共字段

// strategy.setSuperEntityColumns(new String[] { "test\_id", "age" });

// 自定义 service 父类

strategy.setSuperServiceClass(SUPER\_SERVICE\_CLASS);

// 自定义 service 实现类父类

strategy.setSuperServiceImplClass(SUPER\_SERVICE\_IMPL\_CLASS);

// 自定义 controller 父类

// strategy.setSuperControllerClass("com.baomidou.demo.TestController");

// 【实体】是否生成字段常量（默认 false）

// public static final String ID = "test\_id";

// strategy.setEntityColumnConstant(true);

// 【实体】是否为构建者模型（默认 false）

// public User setName(String name) {this.name = name; return this;}

strategy.setEntityBuilderModel(true);

mpg.setStrategy(strategy);

// 注入自定义配置，可以在 VM 中使用 cfg.abc 【可无】

InjectionConfig cfg = new InjectionConfig() {

@Override

public void initMap() {

Map<String, Object> map = new HashMap<String, Object>();

map.put("abc", this.getConfig().getGlobalConfig().getAuthor() + "-mp");

this.setMap(map);

}

};

// 自定义 xxList.jsp 生成

List<FileOutConfig> focList = new ArrayList<FileOutConfig>();

mpg.setCfg(cfg);

// 调整 xml 生成目录演示

focList.add(new FileOutConfig("/templates/mapper.xml.vm") {

@Override

public String outputFile(TableInfo tableInfo) {

return "/develop/code/xml/" + tableInfo.getEntityName() + ".xml";

}

});

cfg.setFileOutConfigList(focList);

mpg.setCfg(cfg);

// 执行生成

mpg.execute();

}

}

MyTableGenerator.java:

import com.xiongjun.mybatisplusgenerator.MyBatisPlusGenerator;

import java.util.ArrayList;

public class MyTableGenerator extends MyBatisPlusGenerator {

// 写全路径

{

// PO对象路径

entityPackage = "com.xiongjun.springcloud.po";

// Mapper路径

mapperPackage = "com.xiongjun.springcloud.mapper";

// Service接口路径

servicePackage = "com.ubtech.ucs.springcloud.service";

// ServiceImpl类路径

serviceImplPackage = "com.xiongjun.springcloud.service.impl";

//设置

tableNames = new ArrayList<>();

tableNames.add("order\_tbl");

}

public static void main(String[] args) {

new MyTableGenerator().generate();

}

}

运行MyTableGenerator.java类即可生成代码