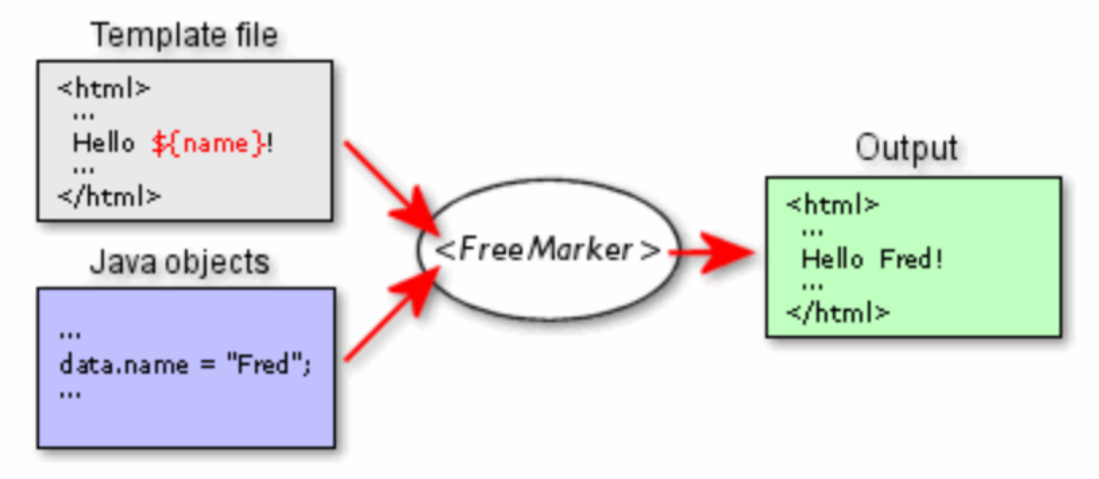
第七篇 代码生成器

一、代码生成器的基础实现原理

我们都使用过或者听说过“模板引擎”，它可以帮我们实现视图与数据的分离，快速开发视图页面，并将模板整合结果用于在浏览器显示。

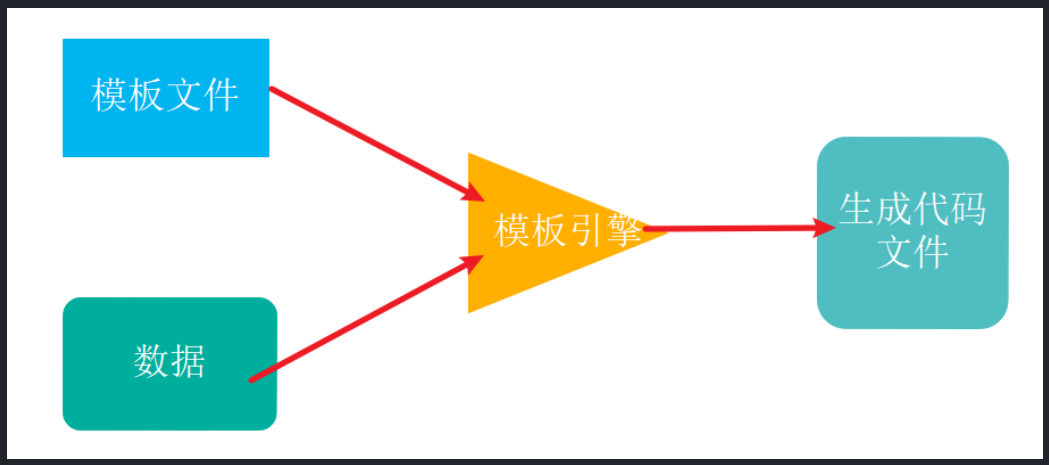
其核心实现原理就是：HTML模板页面 + 页面数据 = 输出结果。

页面视图输出的过程就是通过模板引擎实现的。



代码生成器的实现原理与模板引擎实现页面渲染的逻辑几乎是一致的，除了下面的几个区别：

- 所谓模板：就是某语言的代码 + 模板引擎语法的占位符，该占位符用来数据替换。所以代码生成器的模板文件不再专指HTML页面模板文件，可以是任何类型的代码文件。
- 模板引擎的输出结果在项目中是输出给浏览器进行页面渲染的，但是对于代码生成器而言，模板引擎的输出结果是保存到磁盘文件。



二、如何编写模板文件

要编写模板文件，首先我们要知道正常的代码待如何书写。比如下面的POJO代码：

```java
package top.mqxu.boot.mbp.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.extension.activerecord.Model;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;
import lombok.Builder;


import java.io.Serializable;

/**
 * user表实体类
 * @author mqxu
 * @since 2021-04-01
 */
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode(callSuper = true)
public class User extends Model<User> implements Serializable {
    private static final long serialVersionUID = 6401942840459021558L;
    @TableId(type = IdType.AUTO)
    private Long userId;
    private Long id;
    private String name;
    private Integer age;
    private String email;
}
```

上面的POJO代码写成Freemarker模板文件，就是下面的样子：

```
package ${package.Entity};

<#list table.importPackages as pkg>
import ${pkg};
</#list>

<#if entityLombokModel>
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;
import lombok.Builder;
</#if>

/**
 * <p>
 * ${table.comment!}
 * </p>
 *
 * @author ${author}
 * @since ${date}
 */
<#if entityLombokModel>
@Data
    <#if superEntityClass??>
@EqualsAndHashCode(callSuper = true)
    <#else>
@EqualsAndHashCode(callSuper = false)
    </#if>
</#if>
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class ${entity} extends Model<${entity}> {

<#list table.fields as field>
    private ${field.propertyType} ${field.propertyName};
</#list>
```

Mybatis Plus 代码生成的模板文件：

https://gitee.com/baomidou/mybatis-plus/tree/3.0/mybatis-plus-generator/src/main/resources/templates

三、数据从哪里来?

有了模板文件，我们想通过模板引擎生成代码，下面的一个问题就是数据从哪里来？有了数据我们才能生成代码

- 从配置中来，比如：package路径等一些静态化不经常变化的信息，一个项目生成的代码存放的包路径通常不会经常变化。
- 从数据库中来，比如:实体类名称、实体类字段名称、实体类字段类型等信息。类似于逆向工程，通过数据库表名、字段名、字段类型等信息生成实体信息。

3.1.以MySQL的INFORMATION_SCHEMA信息获取为例

我们的代码自动生成是针对数据库操作，所以首先要了解数据库表的结构

```
SELECT
column_name,data_type,is_nullable,character_maximum_length
,column_comment
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_name='kpi_task'
AND table_schema='home'
ORDER BY ordinal_position
```

如上图SQL查询的是home库，kpi_task表的信息，如下：

COLUMNS (5×18)

column_name	data_type	is_nullable	character_maximum_length	column_comment
id	int	NO	(NULL)	
task_pid	int	NO	(NULL)	父任务id
task_pids	varchar	NO	64	递归父任务id，逗号分隔
is_leaf	int	NO	(NULL)	0：不是叶子结点，1:是
task_name	varchar	NO	128	任务名称（标题）
task_type	varchar	YES	8	任务类别
task_owner	varchar	NO	64	任务归属人，直接存人名
task_creator	varchar	NO	16	任务创建人
task_createtime	datetime	NO	(NULL)	任务创建时间
task_projectid	int	NO	(NULL)	任务归属项目id(对应kpi_project表)
task_orgid	int	NO	(NULL)	任务归属组织id(对应sys_org表)
content	varchar	NO	1,024	任务内容
start_time	datetime	YES	(NULL)	任务开始时间
end_time	datetime	YES	(NULL)	任务结束时间
task_score	int	YES	(NULL)	任务打分
task_evaluate	varchar	YES	128	任务评价
parent_score	int	YES	(NULL)	父任务评分，没有父任务此处填与task_score一样的值
task_status	int	YES	(NULL)	任务状态，0未完成，1已完成等

- column_name作为表的字段可以生成实体类的成员变量参数名称（通常是驼峰标识规则）
- data_type，is_nullable，character_maximum_length可用于生成校验规则。
- 注释可以用于生成column_comment

四、Mybatis Plus代码生成器的使用

如果上面的代码生成器实现原理你都看懂了，下面的这些配置你也就不难理解了。

4.1.添加依赖

- 添加 代码生成器 依赖

```
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-generator</artifactId>
    <version>3.4.1</version>
</dependency>
```

- 添加 模板引擎 依赖，Mybatis-Plus 支持 **Velocity(默认)** 、**Freemarker**、**Beetl**，你可以选择自己熟悉的模板引擎。

添加一个即可，以Velocity为例

```
  <dependency>
      <groupId>org.apache.velocity</groupId>
      <artifactId>velocity-engine-core</artifactId>
      <version>2.3</version>
</dependency>
```

4.2.代码生成配置

AutoGenerator 是 MyBatis-Plus 的代码生成器，通过 AutoGenerator 可以快速生成 Entity、Mapper、Mapper XML、Service、Controller 等各个模块的代码，极大的提升了开发效率。

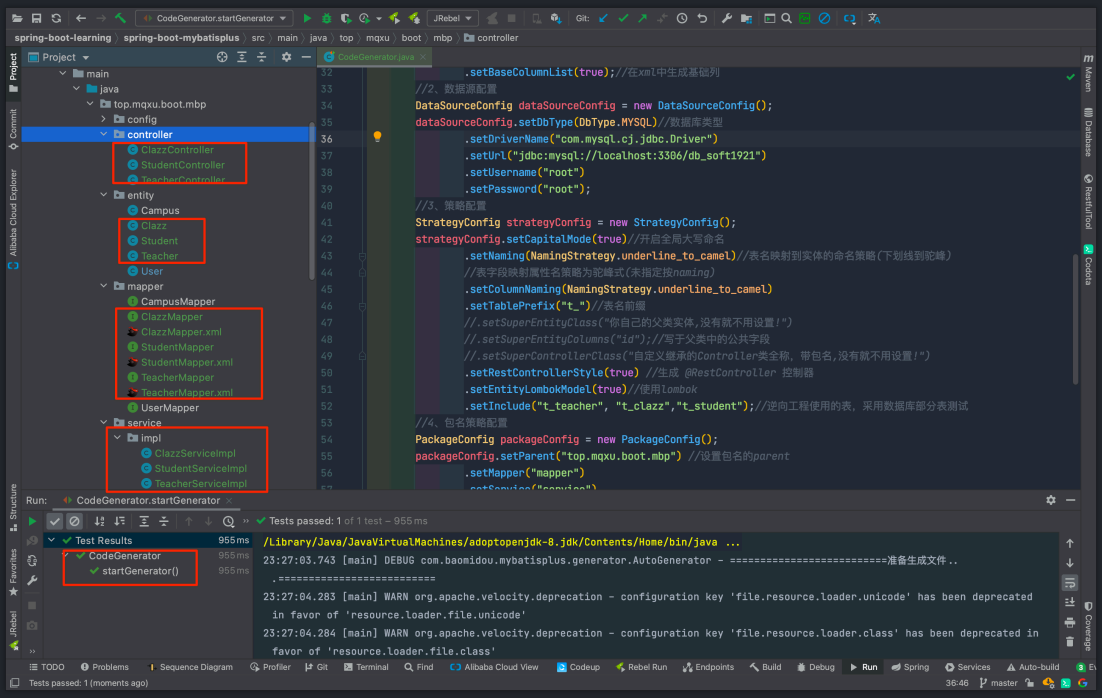
执行下面的这个测试用例，Mybatis Plus就可以帮助我们实现以上各层的代码生成到对应的package路径下。

```
package top.mqxu.boot.mbp.tools;

import com.baomidou.mybatisplus.annotation.DbType;
import com.baomidou.mybatisplus.generator.AutoGenerator;
import
com.baomidou.mybatisplus.generator.config.DataSourceConfig
;
import
com.baomidou.mybatisplus.generator.config.GlobalConfig;
import
com.baomidou.mybatisplus.generator.config.PackageConfig;
import
com.baomidou.mybatisplus.generator.config.StrategyConfig;
import
com.baomidou.mybatisplus.generator.config.rules.NamingStra
tegy;
import org.junit.jupiter.api.Test;

/**
 * @description: 代码生成器
 * @author: mqxu
 * @since: 2021-04-01
 **/
```

```
public class CodeGenerator {
    @Test
    public void startGenerator() {
        //1、全局配置
        GlobalConfig config = new GlobalConfig();
        String projectPath =
System.getProperty("user.dir");
        config.setActiveRecord(true)//开启AR模式
                .setAuthor("mqxu")//设置作者
                .setOutputDir(projectPath +
"/src/main/java")//生成路径(一般在此项目的src/main/java下)
                .setFileOverride(true)//第二次生成会把第一次生
成的覆盖掉
                .setOpen(true)//生成完毕后是否自动打开输出目录
                //.setSwagger2(true)//实体属性 Swagger2 注解
                //.setIdType(IdType.AUTO)//主键策略
                .setServiceName("%sService")//生成的service
接口名字首字母是否为I，这样设置就没有I
                .setBaseResultMap(true)//生成resultMap
                .setBaseColumnList(true);//在xml中生成基础列
        //2、数据源配置
        DataSourceConfig dataSourceConfig = new
DataSourceConfig();
        dataSourceConfig.setDbType(DbType.MYSQL)//数据库类型
                .setDriverName("com.mysql.jdbc.Driver")

.setUrl("jdbc:mysql://localhost:3306/db_soft1921")
                .setUsername("root")
                .setPassword("root");
        //3、策略配置
        StrategyConfig strategyConfig = new
StrategyConfig();
        strategyConfig.setCapitalMode(true)//开启全局大写命名

.setNaming(NamingStrategy.underline_to_camel)//表名映射到实体
的命名策略(下划线到驼峰)
                //表字段映射属性名策略为驼峰式(未指定按naming)

.setColumnNaming(NamingStrategy.underline_to_camel)
                .setTablePrefix("t_")//表名前缀
                //.setSuperEntityClass("你自己的父类实体,没有
就不用设置!")
```

```
                //.setSuperEntityColumns("id");//写于父类中
的公共字段
                //.setSuperControllerClass("自定义继承的
Controller类全称，带包名,没有就不用设置!")
                .setRestControllerStyle(true) //生成
@RestController 控制器
                .setEntityLombokModel(true)//使用lombok
                .setInclude("t_teacher",
"t_clazz","t_student");//逆向工程使用的表，采用数据库部分表测试
        //4、包名策略配置
        PackageConfig packageConfig = new PackageConfig();
        packageConfig.setParent("top.mqxu.boot.mbp") //设置
包名的parent
                .setMapper("mapper")
                .setService("service")
                .setController("controller")
                .setEntity("entity")
                .setXml("mapper");//设置xml文件的目录
        //5、整合配置
        AutoGenerator autoGenerator = new AutoGenerator();
        autoGenerator.setGlobalConfig(config)
                .setDataSource(dataSourceConfig)
                .setStrategy(strategyConfig)
                .setPackageInfo(packageConfig);
        //6、执行
        autoGenerator.execute();
    }
}
```

运行该测试方法，可以看到如下代码生成成功：


spring-boot-learning
spring-boot-mybatisplus
Project
main
java
top.mqxu.boot.mbp
config
controller
ClazzController
StudentController
TeacherController
entity
Campus
Clazz
Student
Teacher
User
mapper
CampusMapper
ClazzMapper
ClazzMapper.xml
StudentMapper
StudentMapper.xml
TeacherMapper
TeacherMapper.xml
UserMapper
service
impl
ClazzServiceImpl
StudentServiceImpl
TeacherServiceImpl
CodeGenerator.java
.setBaseColumnList(true);//在xml中生成基础列
//2、数据源配置
DataSourceConfig dataSourceConfig = new DataSourceConfig();
dataSourceConfig.setDbType(DbType.MYSQL)//数据库类型
.setDriverName("com.mysql.cj.jdbc.Driver")
.setUrl("jdbc:mysql://localhost:3306/db_soft1921")
.setUsername("root")
.setPassword("root");
//3、策略配置
StrategyConfig strategyConfig = new StrategyConfig();
strategyConfig.setCapitalMode(true)//开启全局大写命名
.setNaming(NamingStrategy.underline_to_camel)//表名映射到实体的命名策略(下划线到驼峰)
//表字段映射属性名策略为驼峰式(未指定按naming)
.setColumnNaming(NamingStrategy.underline_to_camel)
.setTablePrefix("t_")//表名前缀
//.setSuperEntityClass("你自己的父类实体,没有就不用设置!")
//.setSuperEntityColumns("id");//写于父类中的公共字段
//.setSuperControllerClass("自定义继承的Controller类全称，带包名,没有就不用设置!")
.setRestControllerStyle(true) //生成 @RestController 控制器
.setEntityLombokModel(true)//使用lombok
.setInclude("t_teacher", "t_clazz","t_student");//逆向工程使用的表，采用数据库部分表测试
//4、包名策略配置
PackageConfig packageConfig = new PackageConfig();
packageConfig.setParent("top.mqxu.boot.mbp") //设置包名的parent
.setMapper("mapper")
Run: CodeGenerator.startGenerator
Tests passed: 1 of 1 test – 955 ms
Test Results 955 ms
CodeGenerator 955 ms
startGenerator() 955 ms
/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...
23:27:03.743 [main] DEBUG com.baomidou.mybatisplus.generator.AutoGenerator - ==========准备生成文件...
==========
23:27:04.283 [main] WARN org.apache.velocity.deprecation - configuration key 'file.resource.loader.unicode' has been deprecated in favor of 'resource.loader.file.unicode'
23:27:04.284 [main] WARN org.apache.velocity.deprecation - configuration key 'file.resource.loader.class' has been deprecated in favor of 'resource.loader.file.class'
Tests passed: 1 (moments ago)