Google

Code

Archive

Rapid **rapid-framework - rapid_generator.wiki** Export to GitHub

---

核心理念

为你生成一切,再根据所需手工*copy*回工作区

通常代码生成器存在的问题.

- 二次开发困难,没有源码可以修改模板文件的model对象等
- 过于智能,自动插入我们的项目中,程序员还需考虑旧的代码会不会被覆盖的问题
- 生成文件的时候让你选要生成那些文件,而rapid只负责生成代码，这样生成器核心一分精简
- 没有将存放模板的目录名称及文件名称利用起来,导致还需配置每个模板文件生成的文件名,目录结构
- 具体请查看我写的文章:为何代码生成器都要这么智能呢?

特性

- 基于FreeMarker模板语言,并且模板易于修改
- 基于数据库,内建好数据库的model,并支持多种数据库(mysql,sql server,oracle测试通过)
- 半手工方式,生成的代码放在某个目录,再手工拷贝回来工作区
- 易于做二次开发,整个生成器本身就是java源代码,源代码核心十分精简,并且鼓励你修改代码,也可以作为任何语言的代码生成器
- 配置简单,只有一个配置文件generator.properties

生成器的运行

在**eclipse**中运行

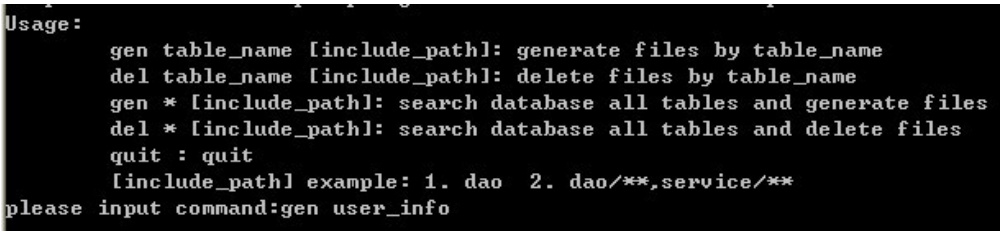
1. 配置classpath,将generator/lib中的rapid-generator.jar及其它数据库驱动加入classpath
2. 修改generator.xml的数据库连接属性及其它属性
3. 以application的方式运行GeneratorMain类,要生成不同的table,直接修改代码即可 ``` public class GeneratoMain { public static void main(String[] args) throws Exception { GeneratorFacade g = new GeneratorFacade();

   g.deleteOutRootDir(); //删除生成器的输出目录 // g.generateByTable("table_name","template"); //通过数据库表生成文件,template为模板的根目录 g.generateByAllTable("template"); //自动搜索数据库中的所有表并生成文件,template为模板的根目录 // g.generateByClass(Blog.class,"template_clazz");

// g.deleteByTable("table_name", "template"); //删除生成的文件 } } ``` 1. 以application方式运行 cn.org.rapid_framework.generator.ext.CommandLine

独立版运行

下载*standandalone-rapid-generator.zip*,解压并运行*rapid-gen.bat*



独立版下载地址

生成器讲解

生成器模板路径可以引用相关变量

示例:*dao/${basepackage_dir}/${className}.java*,根据该变量生成输出文件

如果*basepackage_dir = com/company/rapid, className=UserInfo*

那么完整路径则为:*dao/com/company/rapid/UserInfo.java*

自动搜索某个目录所有模板文件,无需配置

代码生成器模板可以引用的相关变量

``` 1. g.generateByTable("table_name") 方法可以引用的变量 table : cn.org.rapid_framework.generator.provider.db.table.model.Table

2. g.generateByClass("class") 方法可以引用的变量
clazz : cn.org.rapid_framework.generator.provider.java.model.JavaClass

3. g.generateBySql(Sql) 方法可以引用的变量
sql : cn.org.rapid_framework.generator.provider.db.sql.model.Sql

4.公共变量
env : 系统环境变量
System.getProperties() : 直接引用,没有前缀
generator.properties 文件中的所有属性,直接引用,没有前缀
gg : 模板控制变量, cn.org.rapid_framework.generator.GeneratorControl

```

每个模板有**gg**变量可以控制自身的自定义配置 **(**每一个模板都会创建新的**gg**实例**)**

如是否生成,是否覆盖目标文件,甚至是生成其它文件 示例: *${gg.setIgnoreOutput(true)}*, 参考: *rapid_generator_gg*

支持生成**(gen)**及删除操作**(del),**即生成的代码也可以很方便的删除

自动拷贝二进制文件至输出目录

如模板目录下的 *zip,rar,doc*文件将会自动拷贝至输出目录,不会破坏文件格式 *(*通过扩展名自动识别*)*

自动删除模板扩展名**:.ftl,.vm**

举例*:* 如你有一个模板 *SqlMap.xml.ftl* 将变为 *SqlMap.xml* 所以你要生成*ftl*扩展名的文件,应该将文件名从 *list.ftl => list.ftl.ftl*

模板自动**include**同级目录**:macro.include**文件

示例*:* 如你的模板为 *java_src/com/project/UserDao.java,* 将自动*include: java_src/com/project/macro.include* 及根目录的*macro.include*

generator.xml (或者**generator.properties)**配置文件

1. 类似ant可以变量引用,引用环境变量使用${env.JAVA_HOME}, 引用System.getProperties()直接引用

2.自动替换*generator.properties*中的句号*(.)*为反斜杠,设置*key*为*key+"dir"*后缀

示例*: pkg=com.company => pkg_dir=com/company*

十**.** 数据库表配置**,**用于自定义生成器模板引用的**table**变量

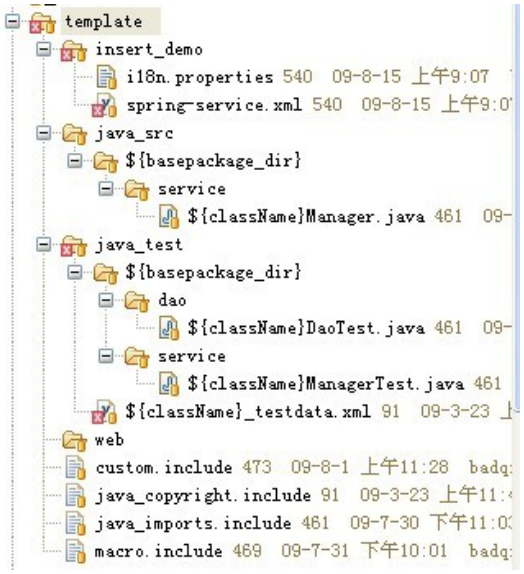
配置文件必须存放在classpath: generator_config/table/table_name.xml (该文件生成器可以生成，自己再自定义修改) <!-- <table sqlName="数据库表名" className="类名称" tableAlias="表的别名"> <column sqlName="数据库列名" columnAlias="列的别名" javaType="自定义javaType" unique="是否唯一性约束" nullable="是否可以为空" pk="是否主键,在表没有主键的情况下,可以指定一个代理主键" updatable="是否可以更新" insertable="是否插入" enumString="枚举值,以分号分隔,示例值:M(1,男);F(0,女) 或者是:M(男);F(女)" enumClassName="如果枚举有值,生成的类名称将是这个,没有枚举值，该配置无用.示例值:Sex" /> </table> --> <table sqlName="user_info" className="UserInfo" tableAlias="UserInfo" > <column sqlName="username" columnAlias="用户名" javaType="String" unique="false" nullable="true" pk="false" updatable="true" insertable="true" enumString="F(1,Female);M(0,Male)" enumClassName="用户枚举" /> <column sqlName="password" columnAlias="password" javaType="String" unique="false" nullable="true" pk="false" updatable="true" insertable="true" enumString="" enumClassName="PasswordEnum" /> </table>

生成的代码插入文档的某个部位

如模板输出生成的地方已经有该 同名的文件 存在,并且该文件中有包含"generator-insert-location"标记,则模板生成的内容会插入在该标记之后.该特性对如生成的spring配置内容插入spring配置文件十分有用

创建一个模板

代码**template**目录结构



如上所示，目录及文件名称可以引用相关变量。

创建一个生成器模板文件

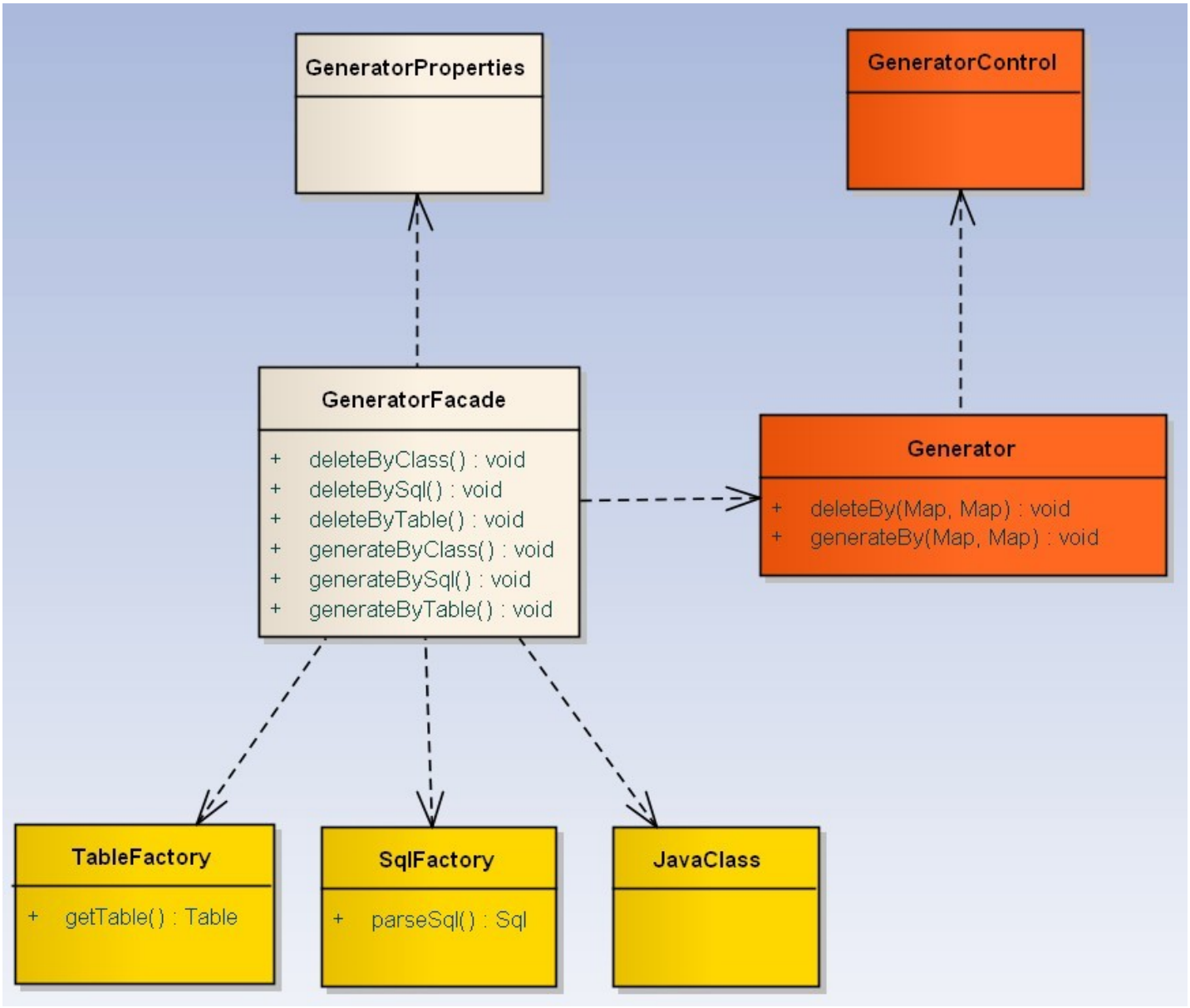
在temppate目录创建一个文件,如${className}SpringControler.java 则代码生成器会自动将该模板文件加载并生成该文件

充分利用各种文件的注释 如在.xml中我们可以使用 在.properties文件中我们可以使用 #generator-insert-location 具体请查看template/insert_demo目录的内容

生成器参数配置

通过设置GeneratorProperties.setProperty(key,value)设置相关参数值. 完整的配置参数请查看:
http://code.google.com/p/rapid-framework/source/browse/trunk/rapid-mvn/rapid-generator/rapid-generator/src/main/java/cn/org/rapid_framework/generator/GeneratorConstants.java

生成器核心类图



* Generator为生成器引擎 * GeneratorFacade为生成器入口调用类 * GeneratorProperties 生成器的相关配置，用于读取generator.properties(或者是generator.xml) * TableFactory用于创建Table.java对象，用于GeneratorFacade.generateByTable()使用 * SqlFactory用于创建Sql.java对象,用于GeneratorFacade.generateBySql()使用 * JavaClass模板变量，用于GeneratorFacade.generateByClass()使用

重复生成代码

现rapid自带的模板都不支持重复生成代码的，如果你需要重复生成代码，可以采用继承机制技巧重复生成。

Java代码的重复生成,善用 **"继承机制"** ,示例如下: `UserInfoBaseDao ： 自动生成的代码, 不能手工修改,用于重复生成 UserInfoDao extends UserInfoBaseDao ： 存放手工的代码，不能重复生成` 页面的重复生成还没有啥好办法。

参考

generateByTable()

通过表(或视图)查询生成代码

generateBySql()

通过sql语句生成代码

generateByClass()

通过java class生成代码

GG控制变量参考

gg控制变量

table变量参考

Table.java 请使用IE查看 table模板变量参考

column变量参考]

Column.java 请使用IE查看

sql变量参考

Sql.java 请使用IE查看

模板引擎freemarker

语法参考: http://freemarker.sourceforge.net/docs/index.html