**DPOP组基础技术框架-代码生成**

|  |  |
| --- | --- |
| **组内审核时间** |  |
| **重要性** |  |
| **紧迫性** |  |
| **技术评审拟制人** | 胡海亮 |
| **技术评审提交日期** | 2014/7/15 |
| **需求变更控制时间点** |  |

百度联盟产品组

（版权所有翻版必究）

**修改记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **更新时间** | **内容** | **修改人** | **版本** |
| 2014/7/15 | DPOP组基础技术框架-代码生成 | 胡海亮 | V1.0.0 |
|  |  |  |  |

# 生成工具原理

代码生成工具主流程如下：



如上图流程：生产工具需配置一个生产代码的规则配置generator.xml，配置文件中会配置代码生产的详细规则，具体到包名结构和类名规则，下节会详细介绍配置详细。

运行开始，程序会获取到配置的的路径，加载配置并解析xml，之后会连接DB加载表的基本信息，其中包括字段的基本信息。后会选择代码生产的插件，插件初始化之后，会负责封装成本插件依赖的MateData的对象。后直接交互给Velocity模板生成代码

再看看DPOP一个全局的继承图：



红色标注的是代码生产工具生产的代码，生产的代码（除了\*Mapper.xml）基本都是继承基类的方法。

# 代码生成配置规则

运行generator.bat即可生产代码.

但运行之前需配置你需要生产的代码的数据库模型，也就是生产的代码依据哪个数据库表。

配置在conf/generator.xml,详细解释：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <generators batisVersion = *"MyBatis3"* >  <project:base>  <!-- 生成代码存放路径 -->  <property name=*"targetPath"* value=*"E:\temp"*/>  <!-- 工程的顶层包名 -->  <property name=*"rootPackage"* value=*"com.baidu.dpop.rmp"*/>  <property name=*"boPackage"* value=*"${rootPackage}.${modulename}.bo"*/>  <property name=*"mapperPackage"* value=*"${rootPackage}.${modulename}.dao.mapper"*/>  <property name=*"daoPackage"* value=*"${rootPackage}.${modulename}.dao"*/>  <property name=*"daoImplPackage"* value=*"${rootPackage}.${modulename}.dao.impl"*/>  <property name=*"servicePackage"* value=*"${rootPackage}.${modulename}.service"*/>  <property name=*"serviceImplPackage"* value=*"${rootPackage}.${modulename}.service.impl"*/>  <property name=*"controllerPackage"* value=*"${rootPackage}.${modulename}.web.controller"*/>  <property name=*"validatorPackage"* value=*"${rootPackage}.${modulename}.web.validator"*/>    <!-- 修改表格字段时候：<property name="generators" value="bo-base,mapperxml-base"></property> -->  <!-- 新建表格的时候： <property name="generators" value="bo,bo-base,mapper,mapperxml,mapperxml-base,dao,dao-ipml,service,service-impl,controller,form,validator"></property> -->  <!-- <property name="generators" value="mapperxml-base"></property> -->  </project:base>    <project:driver>  <property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"*></property>  <property name=*"connectionURL"* value=*"jdbc:mysql://10.46.133.67:8866/os\_test"*></property>  <property name=*"username"* value=*"crm"*></property>  <property name=*"password"* value=*"123456"*></property>  </project:driver>    <project:table>  <!-- 表名 -->  <property name=*"tablename"* value=*"tb\_user"*></property>  <!-- 模块名称，生成java代码包路径的时候需要替换，不能大写 -->  <property name=*"modulename"* value=*"user"*></property>  <!-- 模型的BO名称，生成java代码BO名称，不填写的时候是表名驼峰规则转化而来 -->  <property name=*"boname"* value=*"User"*></property>  </project:table>    <project:table >  <property name=*"tablename"* value=*"tb\_cust"*></property>  <property name=*"modulename"* value=*"cust"*></property>  <property name=*"boname"* value=*"Cust"*></property>  </project:table>    <project:table >  <property name=*"tablename"* value=*"tb\_os\_status"*></property>  <property name=*"modulename"* value=*"osstatus"*></property>  <property name=*"boname"* value=*"OsStatus"*></property>  </project:table>      <project:table >  <property name=*"tablename"* value=*"tb\_opp\_turn\_out\_info"*></property>  <property name=*"modulename"* value=*"turnout"*></property>  <property name=*"boname"* value=*"OppTurnOutInfo"*></property>  </project:table>    <project:table >  <property name=*"tablename"* value=*"tb\_os\_status\_old"*></property>  <property name=*"modulename"* value=*"oldosstatus"*></property>  <property name=*"boname"* value=*"OldOSStatus"*></property>  </project:table>      </generator> |

1、<project:driver>是数据库驱动和用户密码等配置

2、<project:base>配置的是工程的基础信息，

包括存放生产代码的目录、工程顶级包、工程各层包名规范，如上：

表示生产的代码存放在E:\temp下面，顶级包名叫“com.baidu.dpop.rmp”；

BO层包名生产规范是${rootPackage}.${modulename}.bo，modulename后面会介绍；

同理服务层包名是${rootPackage}.${modulename}.service"。

3、<project:table>表示是生产的模型，模型的基本属性从<project:base>继承而来，如果本处自己设定了包名规则，则会覆盖<project:base>的设置。

详细设置DEMO如下：

<project:table>

<!-- 表名 -->

<property name="tablename" value="tb\_material"></property>

<!-- 模块名称，生成java代码包路径的时候需要替换，不能大写 -->

<property name="modulename" value="material"></property>

<!-- 模型的BO名称，生成java代码BO名称，不填写的时候是表名驼峰规则转化而来 -->

<property name="boname" value="Material"></property>

</project:table>

上述设置表示依据表名tb\_material生产代码，模块名（modulename注：上面的规则就是用此处去替换）称是material ,BO名称是Material 其他的包名规则重base继承而来。

<project:table>可以设置多个，表明可以生产多个模块的代码。