



操作手册

惠万鹏

2016 年 10 月

修改记录

编号	日期	描述	版本	作者	审核	发布日期
1	2016/10/3	初稿	0.1	惠万鹏		201610/3

Copyright © 2016 Wanpeng.Hui All rights reserved. Always free.

目 录

{提示：请完成文档后更新整个目录域}

1. 概述	4
1.1. 名词定义.....	4
1.2. 软件背景.....	4
1.3. 本手册目的.....	5
2. PP 系统简述.....	6
2.1. PP 的目录结构说明	6
2.2. PP 的全局配置文件	7
2.3. 数据源面板说明.....	8
2.4. 选择目标表面板说明.....	10
2.5. 生成代码面板说明.....	11
2.6. 系统参数面板说明.....	11
3. 如何自定义模板	12
3.1. 模板描述文件.....	13
3.2. 模板文件.....	14
3.3. 数据模型.....	14
3.4. PO 生成器的数据模型	14
3.5. DAO 生成器的数据模型	15
4. 如何扩展数据库	17
4.1. 数据库信息.....	17
4.2. SQL 配置	18
4.3. 是否主键和是否为空的值映射.....	19
4.4. 表字段类型映射.....	19
5. 如何生成代码	21
5.1. 数据源配置.....	21
5.2. 选择目标表.....	22
5.3. 生成代码.....	22
6. 后记	23

1. 概述

1.1. 名词定义

PP 持久层代码生成器(PP Persistence Layer Code Generator)，我们在后面的文档中简称为 PP。

1.2. 软件背景

为实现软件系统的易维护，易扩展，易理解的目标，我们必须遵循“高内聚，低耦合”的基本设计原则。其简单易行的方案就是分层架构，把整个业务应用分表现层（UI），业务逻辑层（BLL）、持久层（PL）。如图 1.2-01。

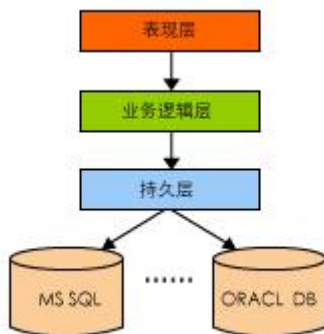


图 1.2-01

持久层对数据库数据的访问是机械的，任何操作，本质上都可以抽象成对数据库 CRUD 操作。为了规范持久层代码开发；为了提高开发效率；为了把我们从沉重、机械、枯燥的持久层代码编写中解放出来，持久层代码生成器存在着积极的意义。

目前的持久层代码生成器，普遍以两种形式出现：一种是独立应用程序，如：MyBatis Generator。另一种是基于 eclipse 插件的，如：hibernate 插件。

它们给我们开发带来了一些便利，但也存在以下一些问题：

- 1) 基于 XML 配置的，没有所见即所得的 GUI 界面。
- 2) 生成的代码死板晦涩、不够自然、可读性差、代码散发着坏的味道。

- 3) 可扩展性差, 更改生成代码的样式, 成本高昂, 需要修改源码。
- 4) 生成器只能为特定的持久层框架生成代码。如 MyBatis Generator 只能为 MyBatis 持久层框架生成代码; Hibernate 插件只能生成 Hibernate 持久层框架生成代码。

PP 是 JAVA 语言编写的一个独立的应用程序, 它正是为了改善上述问题应运而生的。它具有如下特点:

- 1) PP 拥有简洁、友好的所见即所的 GUI 界面, 让生成代码的过程更简单明了。
- 2) PP 是基于 freemarker 模板生成代码的, 您可以修改 PP 内置 freemarker 模板或创建新的模板文件, 生成符合您的编码风格的代码。
- 3) PP 可扩展性极强。它的扩展性体现在几个方面: 支持扩展 PP 没有内置的数据库; 支持 Excel 作为数据源; 支持自定义模板; 安装新模板简单, 只需简单复制文件。
- 4) PP 能为常用的持久层框架生成代码。如 ibatis, MyBatis, Hibernate 等, 包括生成 C#生成持久层代码。

1.3. 本手册目的

本手册介绍 PP 的基本使用方法, 从浅到深解决以下几个问题:

- 1) PP 系统简述。
- 2) 如何扩展新的生成模板。
- 3) 如何扩展新的数据库支持。
- 4) 如何使用 PP 生成代码

2. PP 系统简述

2.1. PP 的目录结构说明

您可以从官网下载最新的 PP，解压到任何您喜欢的目录。解压后的目录结构如图 2.1-01:

pp-generator-0.0.1	此目录为PP的HOME目录
bin	此目录中存放的是启动脚本
run-debug-gui.bat	Windows OS 启动脚本(调试模式)
run-gui.bat	Windows OS 启动脚本
run-gui.sh	Mac OS、Linux OS 启动脚本
config	此目录存放的配置文件
databases	此目录存放的是PP数据库配置文件
db_config_db2_91.xml	DB2数据库配置文件
db_config_mysql_5.xml	My SQL 数据库配置文件
db_config_oracle_9.xml	ORACLE 数据库配置文件
db_config_postgresql_91.xml	PostgreSQL 数据库配置文件
templates	此目录存放Freemarker模板文件
ibatis_ccbjrsc	ibatis建行金融市场DAO模板目录
ibatis_ccbjrsc_dao.ftl	dao模板
ibatis_ccbjrsc_describe.xml	模板自述文件
ibatis_ccbjrsc_mapping.ftl	mapping模板文件
ibatis_comm	ibatis通用DAO模板目录
mybatis_comm	MyBatis通用DAO模板目录
mybatis_rowbounds	MyBatis插件分页DAO模板目录
po_comm	通用PO模板目录
po_jpa_inner1	JPA静态内部类PO模板目录
po_jpa_outer1	JPA普通外部类PO模板目录
sshjpa_comm	SSH JPA通用DAO模板目录
userinterfaces	UI参数目录
ui_prparameters.xml	此文件保存最后一次运行PP的UI参数
config.xml	全局配置文件
libs	系统所需jar包
log	日志目录
error.log	错误日志文件

图 2.1-01

2.2. PP 的全局配置文件

PP 数据库的配置和模板的配置支持两种方式：一种方式是配置在数据库配置文件中中和模板描述文件中，这种方式的好处是安装数据支持或安装模板简单，只需要复制文件。另一种方式是在全局文件 config.xml 里配置，这种配置的方式的好处是一目了然，便于统一管理。缺点是安装新的数据库或插件，需要修改 config.xml，容易出错。

config.xml 中配置的数据库和模板，将被全局解析器解析。在数据配置文中或模板描述文件中的配置，将被局部解析器解析。全局解析器解析结果将覆盖局部解析器解析的结果。

建议在全局 config.xml 里配置数据库和模板。

下面我们就看一下全局配置文件的内容如图 2.2-01 所示：

```
1 *config.xml
2
3 <uiConfig>
4   <uiParamFilePath>config/userinterfaces/ui_parameters.xml</uiParamFilePath>
5 </uiConfig>
6
7 <!--
8   数据库信息配置
9   /config/databases里的数据库配置文件的name|configFilePath|remark信息也可以在这里配置
10  这里配置后，config/databases里的配置将不会生效
11  在这里配置的好处是方便统一管理
12  1. 统一管理，
13  2. 可预定模板在UI界面下拉框中的排列顺序
14 -->
15 <databasesConfig>
16   <database id="DB_ORACLE_01" enable="true">
17     <name>ORACLE 9+</name>
18     <configFilePath>config/databases/db_config_oracle_9.xml</configFilePath>
19     <remark>ORACLE 数据库 Version 9.0 以上</remark>
20   </database>
21 </databasesConfig>
22 -->
23
24 <!--
25   模板文件配置
26   /config/templates里的描述文件信息也可以在这里配置
27   在这里配置后，对应的描述文件配置将不在生效
28   在这里配置的好处是：
29   1. 统一管理，
30   2. 可预定模板在UI界面下拉框中的排列顺序
31 -->
32 <poTemplates>
33   <poTemplate id="PT_COMM_01" enable="true">
34     <name>通用 PO 模板</name>
35     <poTemplateFilePath>config/templates/po_comm/po_comm.ftl</poTemplateFilePath>
36     <remark>通用PO模板</remark>
37   </poTemplate>
38   <poTemplate id="PT_JPA_02" enable="true">
39     <name>通用 JPA PO 模板</name>
40     <poTemplateFilePath>config/templates/po_jpa/po_jpa.ftl</poTemplateFilePath>
41     <remark>JPA通用模板，适用于遵循JPA规范的持久层框架，如：Hibernate</remark>
42   </poTemplate>
43 </poTemplates>
```

2.3. 数据源面板说明

如果您的操作系统是 Windows OS，可以双击 run-gui.bat 启动 PP。

如果您的操作系统是 Mac OS、Ubuntu OS、Linux OS，您可以通过双击 run-gui.sh 来启动 PP。

运行相应的启动脚本后，将出现数据源配置的界面如图 2.3-01：

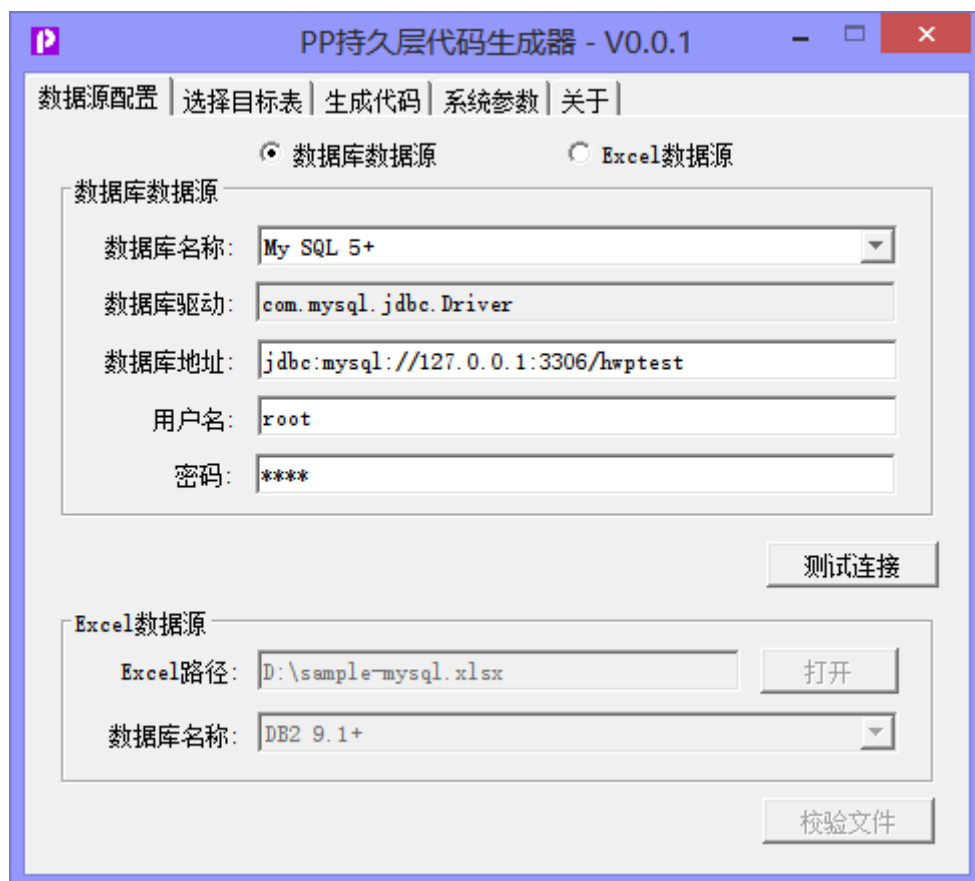


图 2.3-01

数据源界面，主要用于配置从哪里获得需要生成代码的表信息。得到表信息有两个途径：

- 1) 通过 SQL 查询数据库的记录表信息的表或视图。
- 2) 通 Excel 文件记录表的信息。

通过 SQL 查询数据库记录表信息的表或视图，需要相应的数据库权限。而我们往往又没有这些权限。试想一下以下场景：

某电商系统使用的是 ORACLE 数据库。DBA 根据“最小权限原则”创建了三个用户：

- 1) shop_db 拥有数据库所有的权限
- 2) shop_tb 只拥有 shop_db 用户创建的表的查询、添加、修改、删除记录的权限
- 3) shop_qr 只拥有 shop_db 用户创建的表的查询记录的权限。

现在 DBA 只把 shop_tb 用户给了您，您并没有 shop_db 用户下的 user_tab_columns, user_col_comments, user_constraints 表或视图的权限。因您将不能使用数据库数据源这种方式生成代码。此时，Excel 数据的方式派上了用场。

我们先看看一个格式良好的 Excel 数据源的 Excel 文件的内容。如图 2.3-02 所示：

	A	B	C	D	E	F	G	H	I
1	@	ORACLE 9+							
2	*	sys_user	系统表						
3		字段中文名	字段英文名	数据类型	长度	精度	标度	是否主键	可为空
4	#	#用户ID	USER_ID	char	11			P	NO
5	#	登陆名	LOGIN_NAME	varchar	32				
6	#	密码	PASSWORD	varchar	256				
7	#	登记次数	LOGIN_COUNT	int		10			
8	#	最后登陆时间	LAST_LOGIN_TIME	datetime					
9	#	删除标识	DEL_FLAG	int		10			
10	#	最后修改时间	LAST_UPDATE_TIME	datetime					
11	#	创建时间	CREATE_TIME	datetime					
12	#	备注信息	REMARK	varchar	512				

图 2.3-02

为了简化 POI 对 Excel 的解析，我们对 Excel 内容的格式作了以下约定：

- 1) @所在的行的第 2 个单元格是数据库的类型。如图 2.3-02 中第 1 行。
- 2) *所在的行的第 2 个单元格是表名，第 3 个单元格为表的注释。如图 2.3-02 中第 2 行。
- 3) #所在的行的为表的字段信息。如图 2.3-02 中的 4-12 行。
- 4) 行的第 1 个单格无任何内容的，将被解析器忽略。如图 2.3-02 中第 3 行。

5) 每个 sheet 中可以包含多个表,表可以分布在多个 sheet 中。

2.4. 选择目标表面板说明

点击 tabpanel 的“选择目标表”标签,可以切换到“选择目标表”面板如图 2.4-01:

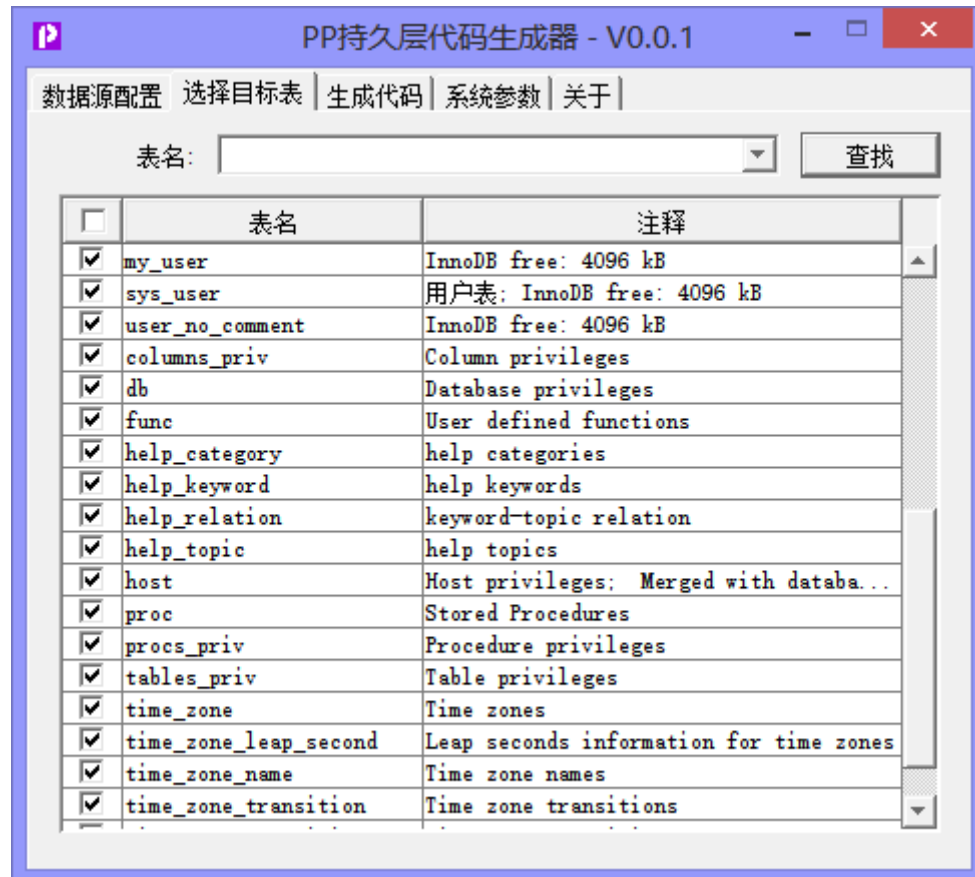


图 2.4-01

此面板用于选择需要生成代码的目标表。如果需要生成特定的表。可以在表名文本框中输入表名,支持模糊匹配。

2.5. 生成代码面板说明

点击 tabpanel 的“生成代码”，可以切换到“生成代码”面板。如图 2.5-01

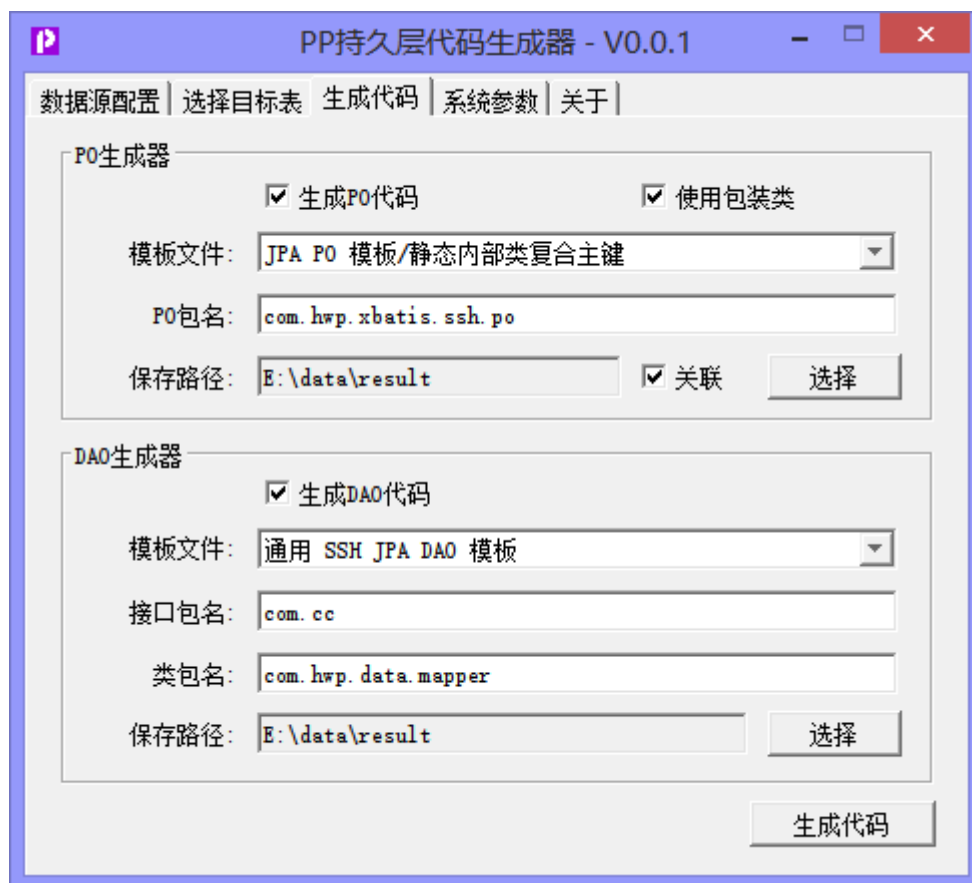


图 2.5-01

这个面板主要用于生成持久层代码。它包含 PO 生成器和 DAO 生成器，把鼠标停留在相应的控件上，会有相应的提示信息。

2.6. 系统参数面板说明

点击 tabpanel 的“系统参数”，可以切换到“系统参数”面板。如图 2.6-01 所示：

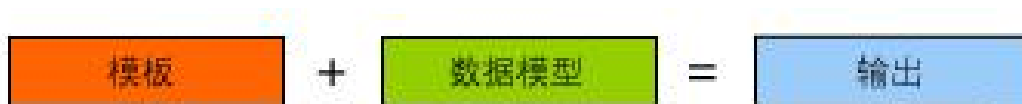


2.6-01

这个面板主要用于配置系统。也还是比较容易理解。唯一需要说明的是“使用驼峰命名法”的复选框。如果你的数据库的表的字段是以下划线作分隔，如 USER_NAME，请勾选此项，Po 的属性将自动命名为 userName；如果你的数据库的表的字段已经遵循了驼峰命名法的规则，如 userName，请不用勾选此项。PP 将不作任何转换。

3. 如何自定义模板

PP 生成代码是基于 Freemarker 模板引擎的。其基本思想如图 1.2-02。



PP 模板按文件类型分为两种：模板描述文件和模板文件。

3.1. 模板描述文件

1. PO 模板描述文件如图 3.1-01:

```
po_jpa_outer1_describe.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <huiWanpengRoot parserName="partConfigParser">
3   <poTemplates>
4     <poTemplate id="PT_JPA_02" enable="true">
5       <name>JPA PO 模板/普通外部类复合主键</name>
6       <pkTemplateFilePath>config/templates/po_jpa_outer1/pk_jpa_outer1.ftl</pkTemplateFilePath>
7       <poTemplateFilePath>config/templates/po_jpa_outer1/po_jpa_outer1.ftl</poTemplateFilePath>
8       <remark>JPA PO 模板, 多个主键为外部类, 适用于遵循JPA规范的持久层框架, 如:Hibernate</remark>
9     </poTemplate>
10  </poTemplates>
11 </huiWanpengRoot>
```

图 3.1-01

PO 模板描述文件主要定义了以下属性:

- 1) parserName, 解析器名称。指定模板描述文件由哪个解析器所解析。
- 2) id, 模板 ID, 多个模板要不能重复。
- 3) enable, 模板是否可用, 为 false 将不会出现在“生成代码”面板的模板下拉列表里。
- 4) name, 模板名称。
- 5) pkTemplateFilePaht, 可选配置, 主键模板文件的路径。如果是要生成的表是复合主键时, 将根据此项生成 PO 的主键。
- 6) poTemplateFilePath, 必须配置, PO 模板文件路径。
- 7) remark, 备注信息, 鼠标停留在“生成代码”面板是的模板下拉列表上, 备注信息将会显示出来。

2. DAO 模板描述文件如图 3.1-02

```
ibatis_comm_describe.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <huiWanpengRoot parserName="partConfigParser">
3   <daoMappingTemplates>
4     <daoMappingTemplate id="IT01" type="ibatis" enable="true">
5       <name>通用 ibatis SQL 分页 DAO 模板</name>
6       <daoInterfaceTemplateFilePath>config/templates/ibatis_comm/ibatis_comm_interface.ftl</daoInterfaceTemplateFilePath>
7       <daoClassTemplateFilePath>config/templates/ibatis_comm/ibatis_comm_dao.ftl</daoClassTemplateFilePath>
8       <daoMappingTemplateFilePath>config/templates/ibatis_comm/ibatis_comm_mapping.ftl</daoMappingTemplateFilePath>
9       <remark>通用ibatis模板, SQL分页</remark>
10    </daoMappingTemplate>
11  </daoMappingTemplates>
12 </huiWanpengRoot>
```

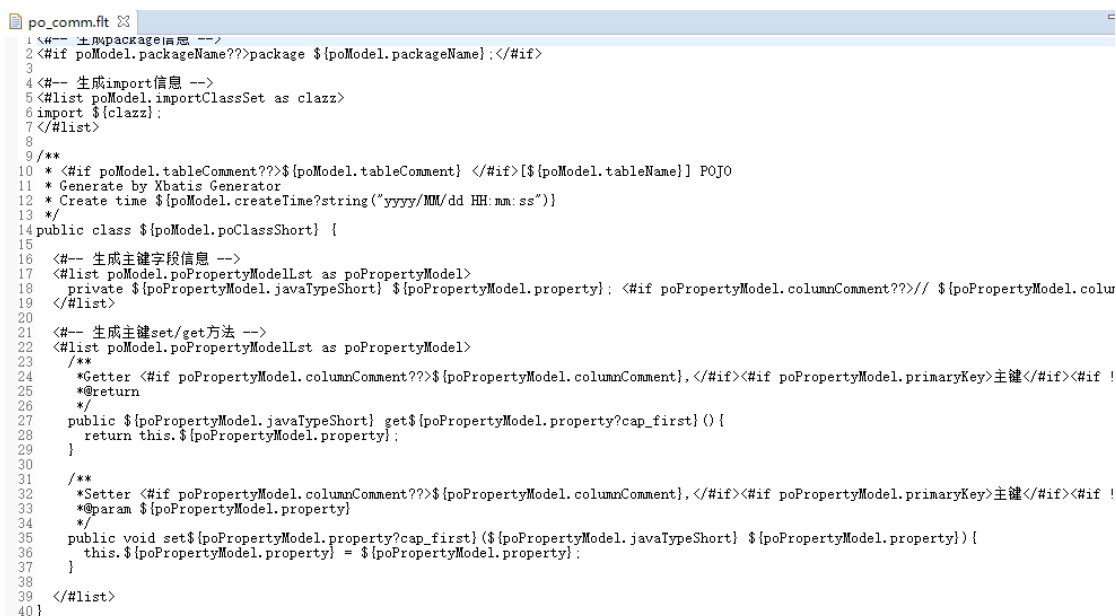
图 3.1-02

DAO 模板描述文件主要定义了以下属性：

- 1) daoInterfaceTemplateFilePath 生成 DAO 接口的模板文件路径
- 2) daoClassTemplateFilePath 生成 DAO 的实现类模板文件路径
- 3) daoMappingTemplateFilePath 生 ibatis、MyBatis 的 Mapping 文件的模板文件路径。

3.2. 模板文件

PP 内置了数据模型，允许用户可以根据数据模板编写模板文件。下面我们看一下内置的模板文件。如图 3.2-01



```
1 <!-- 生成package信息 -->
2 <#if poModel.packageName?>package ${poModel.packageName};</#if>
3
4 <!-- 生成import信息 -->
5 <#list poModel.importClassSet as clazz>
6 import ${clazz};
7 </#list>
8
9 /**
10  * <#if poModel.tableComment?>${poModel.tableComment} </#if>[${poModel.tableName}] POJO
11  * Generate by Xbatis Generator
12  * Create time ${poModel.createTime?string("yyyy/MM/dd HH:mm:ss")}
13  */
14 public class ${poModel.poClassShort} {
15
16     <!-- 生成主键字段信息 -->
17     <#list poModel.poPropertyModelLst as poPropertyModel>
18         private ${poPropertyModel.javaTypeShort} ${poPropertyModel.property}; <#if poPropertyModel.columnComment?>${poPropertyModel.columnComment}</#if>
19     </#list>
20
21     <!-- 生成主键set/get方法 -->
22     <#list poModel.poPropertyModelLst as poPropertyModel>
23         /**
24          * Getter <#if poPropertyModel.columnComment?>${poPropertyModel.columnComment},</#if><#if poPropertyModel.primaryKey>主键</#if><#if !
25          * @return
26          */
27         public ${poPropertyModel.javaTypeShort} get${poPropertyModel.property?cap_first}() {
28             return this.${poPropertyModel.property};
29         }
30
31         /**
32          * Setter <#if poPropertyModel.columnComment?>${poPropertyModel.columnComment},</#if><#if poPropertyModel.primaryKey>主键</#if><#if !
33          * @param ${poPropertyModel.property}
34          */
35         public void set${poPropertyModel.property?cap_first}(${poPropertyModel.javaTypeShort} ${poPropertyModel.property}) {
36             this.${poPropertyModel.property} = ${poPropertyModel.property};
37         }
38     </#list>
39 }
40 }
```

图 3.2-01

3.3. 数据模型

我们要修改 PP 内置的模板文件或自定义 PP 模板文件，关键是要了解 PP 的内置数据模型。我们按生成器分类可以分为 PO 生成器相关的数据模型和 DAO 生成器的数据模型。

3.4. PO 生成器的数据模型

- 1) PO 数据模型， 如图 3.4-01:

2) PO 属性模板， 如图 3.4-02

```
7 /**
8  * PO模型
9  * @version 1.0
10 */
11 public class PoModel
12 {
13     private String packageName; //po包名
14     private HashSet<String> importClassSet; //需要import的class
15     private HashSet<String> importPkClassSet; //Pk需要import的class
16     private HashSet<String> importCmClassSet; //除开Pk需要import的class
17     private String pkClass; // po的PK类
18     private String pkClassShort; //po的PK类不包含包路径
19     private String poClass; //类名包含包路径
20     private String poClassShort; //类名不包含包路径
21     private List<PoPropertyModel> poPropertyModelLst; // 所有属性集合
22     private List<PoPropertyModel> poPkPropertyModelLst; //po主键属性集合
23     private List<PoPropertyModel> poCmPropertyModelLst; //po非主键属性集合
24     private String dbType; // 数据库类型
25     private String dbName; // 数据库名称
26     private String tableName; //表名称
27     private String tableComment; // 表注释
28     private boolean multiPrimaryKey; //是否复合主键
29     private Date createTime = new Date(); //创建时间
30 }
```

图 3.4-01

```
3 /**
4  * 字段模型
5  * @version 1.0
6  */
7 public class PoPropertyModel
8 {
9
10     private String column; //列名
11     private String columnComment; //列注释
12     private String property; //属性名
13     private String dataType; //数据库数据类型
14     private String jdbcType; //jdbc类型
15     private String javaType; //java类型
16     private String javaTypeShort; //java类型不带包路径
17     private int dataLength; //数据长度
18     private int numericPrecision; //小数精度
19     private int numericScale; //小数标度
20     private boolean primaryKey; //是否为主键
21     private boolean nullable; //是否可以NULL
22 }
```

图 3.4-02

3.5. DAO 生成器的数据模型

- 1) DAO 数据模型，DAO 接口， DAO 实现类模板使用,如图 3.5-01
- 2) DAO 主键数据模型 DAO 接口，DAO 实现类模板使用， 如图 3.5-02
- 3) Mapping 数据模型，ibatis,MyBatis 的 Mapping 模板文件使用， 如图 3.5-03

- 4) Mapping 字段数据模型, ibatis, MyBatis 的 Mapping 模板文件使用, 如图 3.5-04

```

/**
 *生成DAO的模型
 */
public class DaoMode
{
    private String daoInterfacePackageName; // dao接口包名
    private String daoClassPackageName; // dao实现类包名
    private HashSet<String> importClassSet; //本类需要引入的类型
    private String daoInterface; // dao接口
    private String daoInterfaceShort; // dao接口不包含包路径
    private String daoClass; //dao类名
    private String daoClassShort; //dao类型不包含包路径
    private String poClass; //po类型
    private String poClassShort; //po类型不包含包路径
    private String dbType; // 数据库类型
    private String dbName; // 数据库名称
    private String tableName; //表名
    private String tableComment; //表注释
    private List<DaoPkModel> daoPkModelList; //主键列表
    private Date createTime = new Date(); //创建时间
}

```

3.5-01

```

/**
 * DAO PK模型
 *
 * @version 1.0
 */
public class DaoPkModel
{
    private String property; // 属性名
    private String javaType; //java类型
    private String javaTypeShort; //java类型不包括包名
    private String columnComment; //注释说明
}

```

3.5-02

```

60 /**
7  * Mapping 配置文件模型
8  *
9  * @version 1.0
10 */
11 public class MappingModel
12 {
13     private String namespace; // 命名空间名称
14     private String poClass; //po类型
15     private String poClassShort; //po类型不包含包路径
16     private String dbType; // 数据库类型
17     private String dbName; // 数据库名称
18     private String tableName; // 表名
19     private String tableComment; //表注释
20     private String startPageSql; // 分页开始SQL
21     private String endPageSql; // 分页结束SQL
22     private List<MappingColumnModel> mappingColumnModelList; // 数据操作模型对象集合
23     private Date createTime = new Date(); //创建时间

```

3.5-0


```

4  * mapping子表数据模型
5  *
6  * @version 1.0
7  */
8  public class MappingColumnModel
9  {
10     private String column; // 数据库表的列名
11     private String columnComment; // 注释说明
12     private String jdbcType; // jdbc数据类型
13     private String property; // po的属性名
14     private String javaType; // java数据类型
15     private String typeHandler; // 类型处理器
16     private boolean primaryKey; // 是否主键
17     private boolean nullable; // 是否可以空
18 }

```

3.5-04

4. 如何扩展数据库

PP 目前只支持 4 种数据库，它们分别是 ORACLE，My SQL，DB2，PostgreSQL。对于没有的数据库，用户可以自行扩展。理论上 PP 支持任何关系数据库。

新扩展一个数据库，只需要在 PP_HOME\config\databases 里新增一个 xml 配置文件即可。

下面我们对数据库配置文件进行说明。

4.1. 数据库信息

数据库信息包含：数据库名称，配置文件相对路径，说明备注数据类型，数据库驱动，数据连接样例。如图 4.1-01：

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <huiWanpengRoot parserName="dbConfigParser">
3    <database id="DB_MYSQL_01" enable="true">
4
5      <!-- 数据库名称，配置文件相对路径，说明备注 -->
6      <name>My SQL 5+</name>
7      <configFilePath>config/databases/db_config_mysql_5.xml</configFilePath>
8      <remark>My SQL 数据库 Version 5.0 以上</remark>
9
10     <!-- 数据类型，数据库驱动，数据连接样例 -->
11     <dbType>mysql</dbType>
12     <driver>com.mysql.jdbc.Driver</driver>
13     <connectUrlExample>jdbc:mysql://127.0.0.1:3306/YourDBName</connectUrlExample>

```

图 4.1-01

上图中，parserName 是解析器名称，指明该配置文件由哪个解析器解析。Id 是数据库配置的 ID，不能重复。Enable 指本配置是否有效，设置成 false 后，

将不会出现在“数据源配置”面板的数据库下拉列表中。

4.2. SQL 配置

配置文件的 sql 分三类：

- 1) 查询表信息的 sql，此 sql 用于查出表名，表注释，我们拿 mysql 举例如图 4.2-02：

```
16      <!-- 通过表名查找表信息 -->
17      <selectTableSql>
18      <mainSql>
19          SELECT
20              table_name,
21              table_comment
22          FROM information_schema.tables
23          WHERE 1=1
24      </mainSql>
25      <conditionEqual>AND table_name = '?'</conditionEqual>
26      <conditionLike>AND table_name LIKE '%?%'</conditionLike>
27      </selectTableSql>
28
```

图 4.2-02

- 2) 查询列信息 sql，此 sql 用于查询出表这些列信息：字段名，字段说明，数据类型，字符长度，数据精度，数据标度，是否主键，是否允许为空。我们拿 mysql 举例，如图 4.2-03：

```
29      <!-- 通过表名查找指定表的所有列信息 -->
30      <selectColumnSql>
31          SELECT
32              column_name,
33              column_comment,
34              data_type,
35              character_maximum_length,
36              numeric_precision,
37              numeric_scale,
38              column_key,
39              is_nullable
40          FROM information_schema.columns
41          WHERE table_name = '?'
42          ORDER BY ordinal_position ASC
43      </selectColumnSql>
44
```

如图 4.2-03

- 3) 分页 sql，任何关系数据库，它的分页 sql 都可以分成两部份。我们分别拿 mysql 和 oracle 举例说明如图 4.2-04，图 4.2-05，

```
45      <!-- 数据库分页SQL, -->
46      <pageSql>
47          <startPageSql></startPageSql>
48          <endPageSql>LIMIT @offsetIndex@, @pageSize@</endPageSql>
49      </pageSql>
```

图 4.2-04

```
56      <!-- 数据库分页SQL, -->
57      <pageSql>
58          <startPageSql>SELECT * FROM (SELECT t.*, rownum rn FROM (</startPageSql>
59          <endPageSql>) t WHERE rownum <![CDATA[<=]]> @offsetIndex@ WHERE rn <![CDATA[>]]> @offsetIndex@ + @pageSize@</endPageSql>
60      </pageSql>
```

图 4.2-05

4.3. 是否主键和是否为空的值映射

通用上一节查字段信息的信息查出字段是否为主键，是否为空的值，是一个字符串。不同数据库它的值是不一样的。My SQL 是以 PRI 表示主键，YES 表示允许为空。ORACLE 以 P 表示主键，Y 表示允许为空。PP 最后会统一转换成一个 boolean 值。因此需要进行值映射。我们分别以 my sql 和 oracle 举例，如图 4.3-01，图 4.3-02 所示：

```
51      <!-- 字段值映射 -->
52      <columnValueMapping>
53          <!-- column_key的值为PRI时,表示该字段为主键 -->
54          <primaryKeyToBoolean>PRI</primaryKeyToBoolean>
55          <!-- is_nullable的值为YES,表可该字段允许为空 -->
56          <nullableToBoolean>YES</nullableToBoolean>
57      </columnValueMapping>
```

图 4.3-01

```
62      <!-- 字段值映射 -->
63      <columnValueMapping>
64          <!-- column_key的值为P时,表示该字段为主键 -->
65          <primaryKeyToBoolean>P</primaryKeyToBoolean>
66          <!-- is_nullable的值为Y时,表可该字段允许为空 -->
67          <nullableToBoolean>Y</nullableToBoolean>
68      </columnValueMapping>
```

图 4.3-02

4.4. 表字段类型映射

字段的类型映射是指根据 SQL 查出的字段信息中的字段类型，对应哪一种 java 类型，jdbc 类型，ibatis 处理器等。下面我们还是配置文件截图来举例说明，如图 4.4-01 所示：

```
71      <!-- 字段类型映射 -->
72      <columnTypeMapping>
73
74          <!-- 字符型 -->
75          <columnType name="CHAR/NCHAR">
76              <jdbcType>CHAR</jdbcType>
77              <javaType>java.lang.String</javaType>
78          </columnType>
79          <columnType name="VARCHAR2/NVARCHAR2" globalDefault="true">
80              <jdbcType>VARCHAR</jdbcType>
81              <javaType>java.lang.String</javaType>
82          </columnType>
83
84          <!-- 数值型-整型 -->
85          <columnType name="INTEGER">
86              <jdbcType>INTEGER</jdbcType>
87              <javaType>int</javaType>
88              <javaTypeWrapped>java.lang.Integer</javaTypeWrapped>
89          </columnType>
90          <columnType name="NUMBER/DECIMAL" numericPrecisionMin="1" numericPrecisionMax="9">
91              <jdbcType>INTEGER</jdbcType>
92              <javaType>int</javaType>
93              <javaTypeWrapped>java.lang.Integer</javaTypeWrapped>
94          </columnType>
95          <columnType name="NUMBER/DECIMAL" numericPrecisionMin="10" numericPrecisionMax="18">
96              <jdbcType>BIGINT</jdbcType>
97              <javaType>long</javaType>
98              <javaTypeWrapped>java.lang.Long</javaTypeWrapped>
99          </columnType>
100      </columnTypeMapping>
101  
```

图 4.4-01

根据上图作以下几点说明：

- 1) columnType 的 name 可以单独配置，也可以使用“|”分隔多个 name 简化配置。
- 2) globalDefault 是指如果没有对应的 columnType，将以此类型作为映射类型。如上图 4.4-01，如果我们不配置 NCHAR，NCHAR 将被映射成 VARCHAR2 类型。
- 3) 可以通过 numericPrecisionMin, numericPrecisionMax, numericScaleMin, numericScaleMax 四个属性来映射指定的类型。
我们知道，oracle 的所有数值型类型都是 number(p,s) 衍生出来的。一个健壮的、易于迁移的表设计只应该包含：CHAR，VARCHAR2，NUMBER，DATE 四种数据类型。如果是 number 定义的整型，长整型，浮点型数据，我也可以通过以上的四个属性去映射成需要的类型。
- 4) typeDefault 它和 globalDefault 有点类似，只是不针对全局，而是针对本类型。
- 5) typeHandler 是指 ibatis 或 MyBatis 的类型处理器。

5. 如何生成代码

这一节我们将以 Excel 数据源来生成代码。样本 Excel 文件在 PP_HOME 的 sample 目录里。

5.1. 数据源配置

- 1) 双击 PP_HOME/bin/run-gui.bat 文件，启动 PP。
- 2) 然后选择 Excel 数据源，此时 Excel 数据源变为可用状态，点击“打开”按钮，选择 PP_HOME/sample/sample-oracle.xlsx 文件。
- 3) 点击“校验文件”按钮，如果 Excel 文件格式正确，将提示“the excel file formate is correct”。如图 5.1-01 所示：

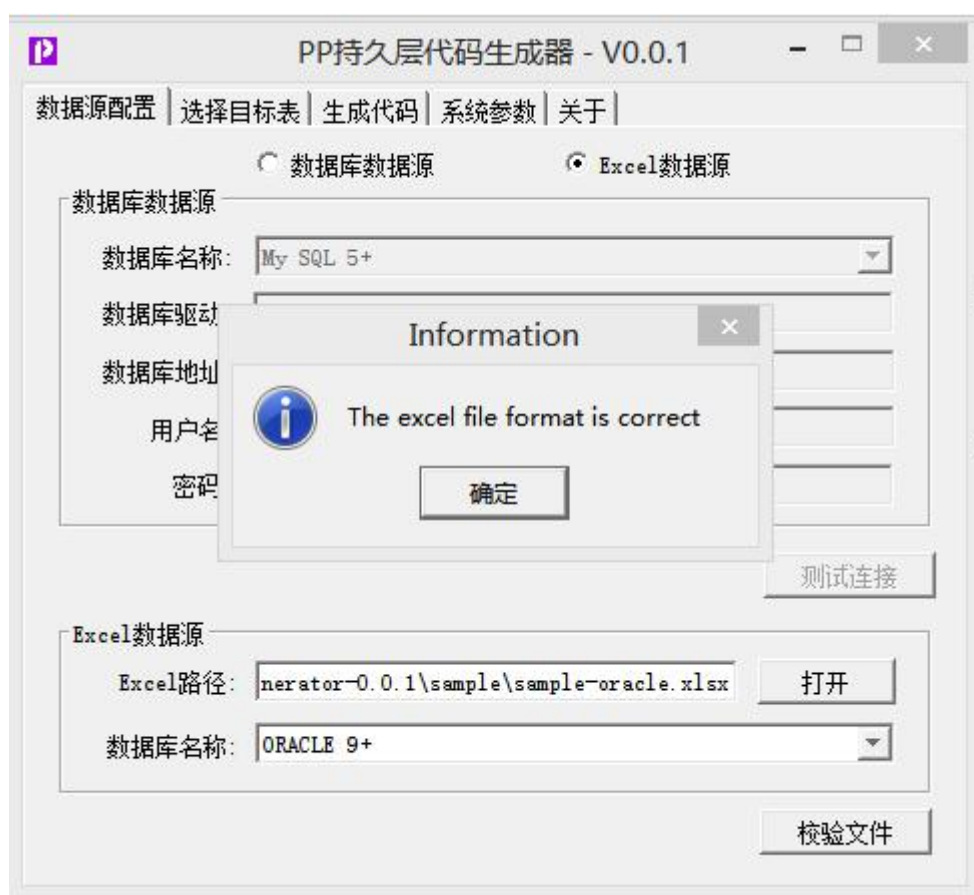


图 5.1-01

5.2. 选择目标表

接下来我们切换到“选择目标表”面板，选择要生成的表。如图 5.2-01 所示：

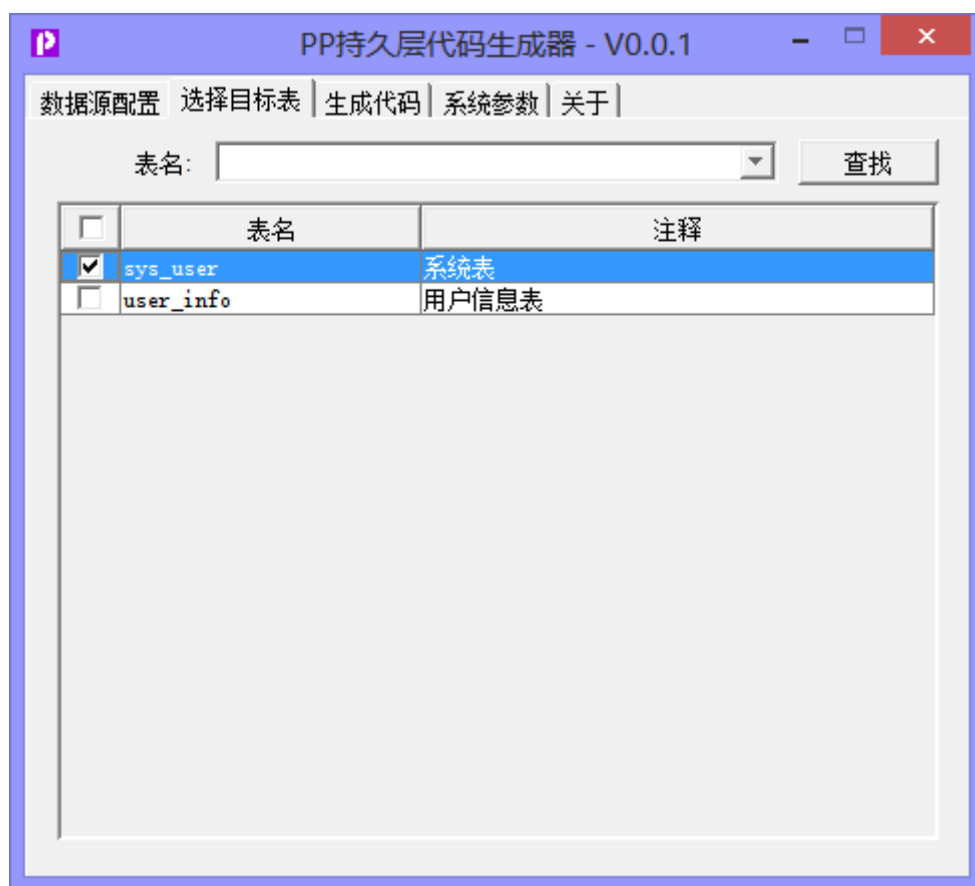


图 5.2-01

5.3. 生成代码

接着我们点击“生成代码”标签切换到“生成代码面板”，如 5.3-01 所示填好参数。

再点击“生成代码”按钮，你会发现 PP 已经为我们生成了代码。如图 5.3-02 所示。

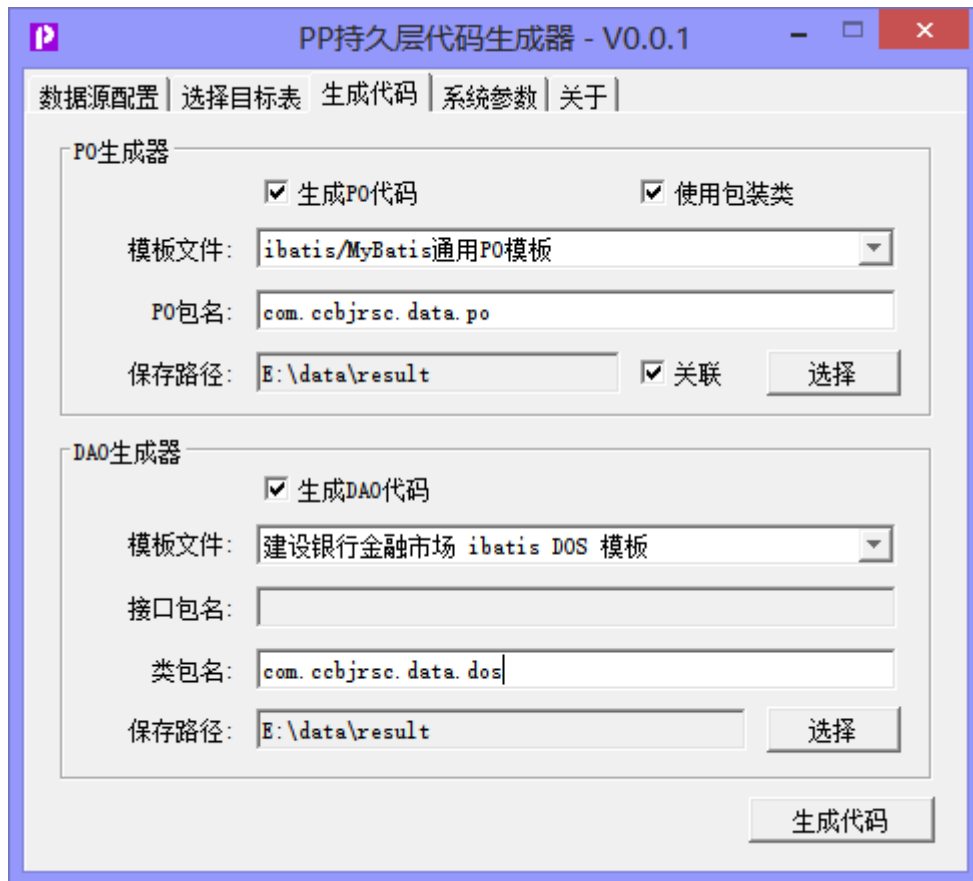


图 5.3-01

名称	修改日期	类型
 SysUserDao.java	2016/10/5 14:56	JAVA 文件
 SysUserDao.xml	2016/10/5 14:56	XML 文档
 SysUserPo.java	2016/10/5 14:56	JAVA 文件

图 5.3-02

6. 后记

PP 是作者利用周末业余时间开发的，未进行大规模测试，如有问题，请发邮件至 hwpok@163.com，我将尽快修复。

作者正在筹备 PP 官网。您将能在官网上预览各种模板的代码样式和风格。在下一个版本的 PP 上输入模板代号，将可以自动在线安装模板。请在 PP 的“关于”面板上扫一扫二维码，1 元，2 元都是您莫大的支持，谢谢！希望 PP 能为您的编码生活带来便利和快乐。