代码生成器使用

Ver 1.0.0

PigX内部群资料

目录

代码生成

项目支持图形化的代码生成，底层采用velocity模板,可以简单快速地生成针对单表的增删改查。

填写生成配置

目前支持的生成配置如下：

- 包名:支持定制项目生成的包名
- 作者:支持当值生成文件的作者
- 模块:支持定制模块名，最终模块名和上面的项目名组成完整的包名
- 表前缀：写在这里的表前缀最终会从生成的entity,service和controller中移除，举个例子，比如 `sys_user` ,默认生成的实体是将下划线转换成Pascal命名,也就是SysUser,而如果输入了 `sys_` 那么这部分不会参与生成，生成出来的结果就是User。
- 注释:支持定制生成在实体，业务层和控制器中的实体的注释，默认加载表备注。

  其中不进行填写的配置都会默认加载代码生成模块中资源文件夹下的 `generator.properties` 中的配置。

生成代码

点击生成按钮以后就能生成代码，并且自动下载成一个zip压缩包。里面包含了前端和代码，按照前端文件夹的组织方式和后端文件夹的组织方式复制进项目即可。

后端相关：

- controller：后端控制器，放到后端的controller文件夹中
- service：后端接口，放到后端的service文件夹中
- impl：后端接口实现类，放到后端的impl文件夹中
- mapper：后端持久层，放到后端的mapper文件夹中

前端相关:

- api:和后端交互的路径，方法参数配置，放到ui项目的src/api文件夹下
- views: 前端的视图层，放到ui项目的src/views文件夹下
- crud: 前端的crud属性的配置，放到ui项目的src/const/crud文件夹下

  当然也可以定制自己的子文件夹，注意调整VUE中import的指令和数据库中的菜单地址即可。

替换生成的样本sql

菜单里当前的菜单总共有三种类型:

- 左侧菜单导航栏中的顶层菜单。这类菜单的特色，就是它的组件是一个 `Layout` ,而这个 `Layout` 就是一个自定义的组件，它的本质就是一个 `router-view` ,并没有实际的业务含义，并且，这类顶层菜单都是以/作为path的开头
- 左侧菜单导航栏中的业务菜单。这类菜单有自己的组件（通常对应一张VUE页面），并且它有一个顶级父路经（即使是无限路由的最终也会映射到一个第一类中的顶层路径），它和第一类菜单一样，都有一个path属性，最终这个菜单的路由就是由它的父path和它的path一起组成的。
- 按钮。这类菜单只是在菜单管理中才会出现，其他时候，在页面上不会看到这类菜单的表现。但是，其所具有的permission属性是前端和后端进行权限控制的基础。在前端，通过v-if指令判断这个permission属性是否最终会被渲染，在后端，PreAuthorize注解配合SPEL表达式最终实现权限控制。

  而我们得到的菜单基本的样子如下：

```
-- 菜单SQL
insert into `sys_menu` (`parent_id`, `component`, `permission`, `type`, `path`
, `icon`, `menu_id`, `del_flag`, `create_time`, `sort`, `update_time`, `name`
)
    values ('父菜单ID', 'views/daemon/executionlog/index', '', '0', 'executionl
og', 'icon-bangzhushouji', '菜单ID', '0', '2018-01-20 13:17:19', '8', '2018-07-2
9 13:38:19', '管理');

-- 菜单对应按钮SQL
insert into `sys_menu` (`parent_id`, `component`, `permission`, `type`, `path`
, `icon`, `menu_id`, `del_flag`, `create_time`, `sort`, `update_time`, `name`
)
    values ('菜单ID', null, 'daemon_executionlog_add', '1', null, '1', '子按钮
D1', '0', '2018-05-15 21:35:18', '0', '2018-07-29 13:38:59', '新增');
insert into `sys_menu` (`parent_id`, `component`, `permission`, `type`, `path`
, `icon`, `menu_id`, `del_flag`, `create_time`, `sort`, `update_time`, `name`
)
    values ('菜单ID', null, 'daemon_executionlog_edit', '1', null, '1', '子按钮
ID2', '0', '2018-05-15 21:35:18', '1', '2018-07-29 13:38:59', '修改');
insert into `sys_menu` (`parent_id`, `component`, `permission`, `type`, `path`
, `icon`, `menu_id`, `del_flag`, `create_time`, `sort`, `update_time`, `name`
)
    values ('菜单ID', null, 'daemon_executionlog_del', '1', null, '1', '子按钮
D3', '0', '2018-05-15 21:35:18', '2', '2018-07-29 13:38:59', '删除');
```

因此只需要简单的全局替换掉"父菜单ID"， “菜单ID"即可，然后自己手输"子按钮ID1”，“子按钮ID2”，"子按钮ID3"即可。

目前整个系统的菜单表的id都是手动维护的，其基本的规则如下:

- 顶级菜单是一个递增的阿拉伯数字+三个0
- 二级菜单是一位递增的阿拉伯数字（这个假设是基于二级菜单不会超过9项）

- 三级按钮是两位阿拉伯数字，小于10的前面补0（这个假设基于一个菜单相关的按钮状态不会超过99项）

编写前端按钮

项目生成的crud按钮不具备控制权限的能力，所以我们都把它关闭，然后自己编写按钮，左侧的 `添加` 按钮的操作方式就是增加一个 `menuLeft` 插槽，代码如下：

```
<template  slot="menuLeft">
   <el-button  v-if="权限字符串"
               class="filter-item"
               @click="handleCreate"
               size="small"
               type="primary"
               icon="el-icon-edit">添加
   </el-button>
</template>
```

操作栏的更新和删除按钮可以增加一个 `menu` 插槽，代码如下:

```
<template  slot="menu"
           slot-scope="scope">
   <el-button  v-if="权限字符串"
               size="small"
               type="text"
               icon="el-icon-edit"
               @click="handleUpdate(scope.row,scope.index)">编辑
   </el-button>
   <el-button  v-if="权限字符串"
               size="small"
               type="text"
               icon="el-icon-delete"
               @click="deletes(scope.row,scope.index)">删除
   </el-button>
</template>
```

后端权限控制

后端的controller上添加注解:

```
@PreAuthorize("@pms.hasPermission('按钮权限字符串')")
```

即可实现后端的权限控制。