云起-YunQi管理系统

目 录

[开始](#Top_of_qianyanjieshao_html)1

[前言介绍](#Top_of_qianyanjieshao_html)1.1

[系统安装](#Top_of_xitonganzhuang_html)1.2

[目录结构](#Top_of_mulujiegou_html)1.3

[后端](#Top_of_qingqiuliucheng_html)2

[请求流程](#Top_of_qingqiuliucheng_html)2.1

[Crud功能](#Top_of_Crudgongneng_html)2.2

[权限控制](#Top_of_quanxiankongzhi_1_html)2.3

[上传文件](#Top_of_shangchuanwenjian_html)2.4

[队列任务](#Top_of_duilierenwu_html)2.5

[消息组件](#Top_of_xiaoxizujian_html)2.6

[记录日志](#Top_of_jilurizhi_html)2.7

[前端](#Top_of_hexinduixiangYunqi_html)3

[核心对象Yunqi](#Top_of_hexinduixiangYunqi_html)3.1

[生命周期](#Top_of_shengmingzhouqi_html)3.1.1

[页面对象Yunqi.app](#Top_of_Yunqi_appduixiang_html)3.1.2

[页面配置Yunqi.config](#Top_of_yemianpeizhiYunqi_config)3.1.3

[页面属性Yunqi.data](#Top_of_yemianshuxingYunqi_data_h)3.1.4

[权限控制Yunqi.auth](#Top_of_quanxiankongzhiYunqi_auth)3.1.5

[页面方法Yunqi.api](#Top_of_yemianfangfaYunqi_api_htm)3.1.6

[页面操作](#Top_of_yemiancaozuo_html)3.1.6.1

[数据操作](#Top_of_shujucaozuo_html)3.1.6.2

[AJAX操作Yunqi.ajax](#Top_of_ajaxcaozuo_html)3.1.7

[消息Yunqi.message等](#Top_of_tanchuxiaoxi_html)3.1.8

[引用JS文件Yunqi.use](#Top_of_yinyongJSwenjianYunqi_use)3.1.9

[内置组件](#Top_of_neizhizujian_html)3.2

[通用表格 - Table](#Top_of_tongyongbiaoge_Table_html)3.2.1

[通用表单 - Form](#Top_of_tongyongbiaodan_html)3.2.2

[JSON控件 - Fieldlist](#Top_of_JSONkongjian_Fieldlist_ht)3.2.3

[分页关联表 - SelectPage](#Top_of_fenyeguanlianbiao_SelectP)3.2.4

[上传头像 - UploadImg](#Top_of_shangchuantouxiang_Upload)3.2.5

[相册 - Attachment](#Top_of_xiangce_Attachment_html)3.2.6

[选择图标 - CheckIcon](#Top_of_xuanzetubiao_CheckIcon_ht)3.2.7

[绑定第三方账号 - Third](#Top_of_bangdingdisanfangzhanghao)3.2.8

[权限控制-Auth](#Top_of_quanxiankongzhi_2_html)3.2.9

[应用扩展](#Top_of_kaifayukaifangkuozhan_htm)4

[创建与开放扩展](#Top_of_kaifayukaifangkuozhan_htm)4.1

[插件](#Top_of_chajian_html)4.2

[开放扩展支持](#Top_of_kaifangkuozhanzhichi_html)4.2.1

[省份城市地区数据](#Top_of_shengfenchengshidiqushuju)4.2.2

[腾讯地图位置选择](#Top_of_tengxundituweizhixuanze_h)4.2.3

[仿站扒站工具](#Top_of_fangzhanbazhangongju_html)4.2.4

[阿里云短信](#Top_of_aliyunduanxin_html)4.2.5

[腾讯云COS](#Top_of_tengxunyunCOS_html)4.2.6

[阿里云OSS](#Top_of_aliyunOSS_html)4.2.7

[应用](#Top_of_yingyong_html)4.3

[uniapp开发脚手架](#Top_of_uniappkaifajiaoshoujia_ht)4.3.1

[商品每日预定与配送系统](#Top_of_yudingyupeisongxitong_htm)4.3.2

[其他说明](#Top_of_duoyuyan_html)5

[多语言](#Top_of_duoyuyan_html)5.1

[常见错误【必坑必读】](#Top_of_changjiancuowu_html)5.2

[1.3版本更新](#Top_of_1_3banbengengxin_html)5.3

# 前言介绍

❤❤❤云起开发平台，用一次就离不开的快速开发框架❤❤❤

诗曰：  
行到水穷处  
坐看云起时  
话说天下大势，分久必合，合久必分...

**云起-Yunqi开发平台** 是基于Thinkphp8.0 + ES6.0 + Vue3.2 + Element Plus2.3等技术开发的，开源免费的快速开发平台。独此一家单页面多应用的设计思路，onLoad，onShow，onUnload，onHide，让管理端也能像uniapp一样开发。系统功能支持模块化打包成扩展，换个项目可以一键安装。

**官方网址**：<https://www.56q7.com/>**开源社区**：<https://bbs.56q7.com/>**演示地址**：<https://demo.56q7.com/backend>**QQ讨论群：237626046**

## 前言

本人在好多年前就自己接项目做项目，那时候没有什么开源的开发框架，全凭自己手撸，修修补补的过程中，我开源了第一套快速开发平台[老成lcfms](https://www.oschina.net/p/lcfms)。当时用到了最新的java8.0，插件即插即用开发的思想在java领域还非常少见，UI甚至支持可视化编辑，在那时候看起来真叫一个小母牛开飞机，牛逼上了天。

但是，web技术迭代速度实在是太快，这套系统的缺点也随之显现。  
1、前后端耦合度太高，使得学习成本大增，必须是全栈工程师才能把这套系统跑起来。  
2、跟那时候的web项目一样的通病，前端文件管理混乱，风格各异。  
3、spring boot流行起来后，java开发首推微服务，基于插件开发思想就完全被抛弃。

后来连我自己也懒得用了，我关掉开源代码与社区，解散了QQ群，开始转战php。我发现php这种解释型语言做web开发效率更高，大概是java的两到三倍左右，不需要编译，不需要上传代码，不需要打断点调试，而且插件的开发思想在这里依然屹立不倒，不管是wordpress、discuz、微擎，都是定义好框架标准，业务代码基于插件，我看来，这才是工业化编程的正确打开方式。

在最近几年，我们公司一直用Fastadmin，我这个人比较喜欢追求新技术，随着前端技术的日新月异，Fastadmin逐渐让我感觉落伍，今年我终于忍无可忍，决定给他来一次大升级。开始，我还保持后端不变，用了一个月的时间，前端全部推倒替换成element plus最新版本。我邀请fastadmin官方合作，可惜人家看不上俺，我就索性一不做二不休，把后端也替换成最新的thinkphp8.0，至此，fastadmin的代码被我去得一行不剩，仅保留了Fastadmin的设计思想。

我个人不喜欢前后端分离，原因1、打包太麻烦，原因2、前端学习成本过高，原因3、响应速度慢。相比于以前的web开发，我认为前端有一点矫枉过正的感觉，在以前前端后端搅在一起的确很乱，现在恨不得前端把所有后端的事情都干完，没必要。所以我设计的一个页面对应一个Vue应用，使用主框架对各个页面应用进行统一管理，单个应用中仍然使用模块化编程，充分发挥ES6的长处。

关于构建插件，我希望系统能够任意拆合，像编程模块一样进行任意组装，任何时候，开发者都可以将项目的一部分功能打包成一个扩展，通过安装与卸载的方式为对工程进行管理。其次，Fastadmin十分谨慎，官方为了规避一些法律风险审核一直很严格，我希望我的框架能更加包容更加开放。为了不给公司惹上官司，我将系统的插件功能设计成去中心化，让任何个人或组织都能在自己服务器上发布插件，如果用户足够多，后期我考虑做一个插件搜索引擎，让开发者更容易发布插件，并且让他人搜索到。

## 系统特色

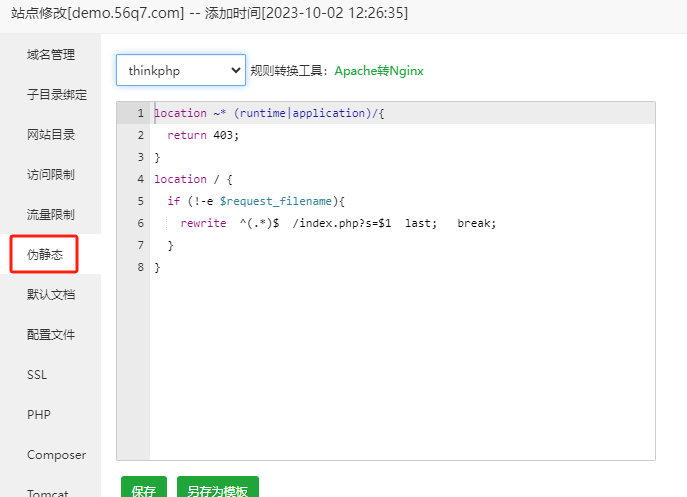
* 🚀一键CRUD，通过网页可视化编辑配置，生成菜单，表格，表单等功能以及所有文件和代码
* 🍜对单表，内置了查看、添加、修改、删除、更新、导入、下载、回收站功能
* 🚩注解方式路由控制
* 🥰 ‍‍前后端不分离，无需编译打包，但仍能采用模块化编程，单页应用支持onLoad，onShow，onUnload，onHide，让前端像uniapp一样写代码。
* 🧊支持打包系统功能成为一个模块，换个项目一键安装，还可以作为扩展直接发布销售。
* 📱手机端，PC端，平板响应式适配，一样的体验效果。
* 🗒️全部使用ES6语法

# 系统安装

环境要求

* PHP >= 8.2
* Thinkphp 8.0+
* Mysql 5.7及以上
* nginx或apache开启静态重写

## 快速安装

1. 两种下载方式：（1）登陆官网，下载稳定版本，已经打包好了composer依赖，解压到apache或nginx配置的网站目录。 （2）使用git clone命令在gitee或github中克隆到你的网站目录。
2. 如果你使用宝塔，请配置好网站的运行目录。  
   
3. 配置好伪静态。  
   
4. 如果你使用git clone的项目，运行 composer install，下载依赖包。
5. 在浏览器中运行：[http://你的域名/install/install.php](http://%E4%BD%A0%E7%9A%84%E5%9F%9F%E5%90%8D/install/install.php)
6. 根据提示完成安装。

**windows环境注意**：  
topthink\think-annotation\src\InteractsWithRoute.php有一个严重的bug导致window环境无法注解带包的控制器，第89行，如果官方修改了可以忽略  
原来是：  
$prefix = str\_replace($this->controllerSuffix . '.php', '', str\_replace('/'', '.', $filename));  
  
改成：  
$prefix = str\_replace($this->controllerSuffix . '.php', '', str\_replace(['/','\\'], '.', $filename));

# 目录结构

├─addons （扩展目录）  
│ ├─ app （应用）  
│ ├─ plugin （插件）  
│ ├─ component （js组件）  
│ └─ cms-temp（ cms模板，cms扩展待开发）  
├─app （thinkphp多应用目录）  
│ ├─ admin （管理端应用，参考thinkphp的目录结构）  
│ │ ├─ command (命令目录)  
│ │ ├─ config (应用配置目录)  
│ │ │ ├─ yunqi.php （管理端基本配置文件）  
│ │ │ └─ session.php （管理端session配置文件）  
│ │ ├─ controller(控制器目录)  
│ │ ├─ lang(语言包目录)  
│ │ ├─ middleware(中间件目录)  
│ │ ├─ route(系统采用注解方式定义路由，这个目录处理特殊路由)  
│ │ ├─ service（服务类目录，管理端权限处理，扩展相关）  
│ │ ├─ traits（管理端实现的crud等应用功能）  
│ │ ├─ view（管理端视图）  
│ │ ├─ common.php （管理端公用函数）  
│ │ └─ middleware.php （中间件映射文件）  
│ ├─ common （应用公用目录）  
│ │ ├─ controller（公用控制器目录）  
│ │ │ ├─ BaseController.php （控制器基类）  
│ │ │ ├─ Backend.php （管理端控制器基类）  
│ │ │ └─ Api.php （Api控制器基类）  
│ │ ├─ library（常用工具类目录，包括日期、http请求、图片处理、二维码等）  
│ │ ├─ model（数据库表模型目录）  
│ │ ├─ service（服务类目录，包括文件上传，多语言控制，发送消息等）  
│ │ ├─ view（公用视图，包括404，消息提示等）  
│ │ └─ 其他目录  
│ └─ 其他应用  
├─config（参考thinkphp配置文件目录）  
├─runtime（参考thinkphp的运行文件目录）  
├─storage（私有文件上传存储目录）  
├─public（管理端前端目录）  
│ ├─ assets（静态文件目录）  
│ │ ├─ css（css目录）  
│ │ ├─ img（公用图片目录）  
│ │ ├─ lib（公用js包目录）  
│ │ ├─ js（js应用目录）  
│ │ │ ├─ components （js组件目录）  
│ │ │ └─ ...其他js文件

# 请求流程

1、定义控制器

使用严格模式定义控制器，需继承\app\common\controller\Backend  
，如果控制器需要初始化，可以重写\_initialize方法，但必须首先执行 parent::\_initialize()

<?php  
declare (strict\_types = 1);  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use think\annotation\route\Get;  
use think\annotation\route\Post;  
use think\annotation\route\Route;  
class Index extends Backend  
{  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 //...需要初始化的内容  
 }  
}

### 2、注解路由

系统采用注解的方式映射路由，可以提高系统的安全性和权限控制的可读性。  
注解路由的名称没有严格规范，理论上，只要不冲突皆可，笔者习惯用包名/控制器名  
定义分组，用方法名  
定义路由名称。

//注解分组  
#[Group("user/index")]  
class Index extends Backend  
{  
 //使用Route注解  
 #[Route('GET,POST,JSON','test1')]  
 public function test1()  
 {  
   
 }  
   
 //使用Get注解  
 #[Get('test2')]  
 public function test2()  
 {  
   
 }  
   
 //使用Post注解  
 #[Post('test3')]  
 public function test3()  
 {  
   
 }  
}

### 3、创建和使用视图

在目录app/admin/view/包名/控制器名/  
中创建视图方法名.html  
  
使用$this->assgin(string $key,mixed $data)  
方法向视图传值  
使用return $this->fetch(string $file='')  
方法使用视图  
注意：js中尽量不要使用php后台渲染标签，诸如{if} {foreach} {$变量}  
，需要使用变量的时候用Yunqi.data.变量  
来实现

视图文件，JS的详细写法[点击这里](https://48rmn452q3.k.topthink.com/@48rmwkl52q/Yunqi.appduixiang.html)查看

<template>  
 <div>视图内容{$test}</div>   
</template>  
<script>  
 export default{  
 data:{   
 },  
 methods: {  
 }  
 }  
</script>  
<style>  
@import url('引用别的css文件')  
div{  
 ...css样式  
}  
</style>

使用视图和传值

#[Group("user/index")]  
class Index extends Backend  
{  
 #[Route('GET','test')]  
 public function test()  
 {  
 $this->assign('abc',123);  
 return $this->fetch();  
 }  
}

### 4、AJAX请求并返回数据

$this->success(string $msg,mixed $data)  
，执行成功时返回数据的方法  
$this->error(string $msg,mixed $data)  
，执行错误时返回数据的方法  
当使用了这两个方法，程序将会在控制器中终止，并继续执行返回数据，记录日志等流程。

### 5、view获取assign的数据

在template中获取assign的数据

{$abc}

在script中获取assign的数据

Yunqi.data.abc

# Crud功能

1、使用Actions内置方法

针对单张表，系统内置了增删改查等功能，只需要在控制器中简单引用app\admin\traits\Actions  
即可实现。

<?php  
declare (strict\_types = 1);  
  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use app\admin\traits\Actions;  
use app\common\model\User;  
  
#[Group("user/index")]  
class Index extends Backend  
{  
 //引入已经写好的curd功能  
 use Actions;  
  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 //定义要操作的表  
 $this->model = new User();  
 }  
}

### 2、内置方法路由

内置路由列表，其中“修改”与“更新”的区别为，“修改”针对单行全部字段，“更新”针对多行单个字段。

|  |  |  |
| --- | --- | --- |
| 功能名称 | 方法名称 | 路由 |
| 查看 | index | 分组路由/index |
| 添加 | add | 分组路由/add |
| 修改 | edit | 分组路由/edit |
| 删除 | del | 分组路由/del |
| 更新 | multi | 分组路由/multi |
| 导入 | import | 分组路由/import |
| 下载 | download | 分组路由/download |
| 回收站 | recyclebin | 分组路由/recyclebin |

### 3、重写方法

可以根据开发场景，选择完全重写内容，或者重写部分内容。

<?php  
declare (strict\_types = 1);  
  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use think\annotation\route\Route;  
use app\admin\traits\Actions;  
use app\common\model\User;  
  
#[Group("user/index")]  
class Index extends Backend  
{  
 use Actions{  
 add as private \_add;  
 }  
  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 $this->model = new User();  
 }  
  
 //重写index方法  
 #[Route('GET,JSON','index')]  
 public function index()  
 {  
 //...具体内容  
 }  
  
 //重写add方法  
 #[Route('GET,POST','add')]  
 public function add()  
 {  
 //...前置内容  
 return $this->\_add();  
 }  
}

### 4、index方法中使用关联表

首先定义模型关联方法

<?php  
declare(strict\_types=1);  
  
namespace app\common\model;  
  
use think\Model;  
  
class User extends Model {  
   
 public function goods()  
 {  
 return $this->hasOne(Goods::class,'id','goods\_id');  
 }  
   
}

然后在控制器中关联属性

<?php  
declare (strict\_types = 1);  
  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use app\admin\traits\Actions;  
use app\common\model\User;  
  
#[Group("user/index")]  
class Index extends Backend  
{  
 use Actions;  
  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 $this->model = new User();  
 //关联goods表  
 $this->relationField=['goods'];  
 }  
}

### 5、重写add、edit方法

在开发中，内置的add与edit方法往往不能满足需求，需要增加业务功能或者增加、修改字段的值。

<?php  
declare (strict\_types = 1);  
  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use app\admin\traits\Actions;  
use app\common\model\User;  
use think\annotation\route\Route;  
  
#[Group("user/index")]  
class Index extends Backend  
{  
 use Actions{  
 add as protected \_add;  
 edit as protected \_edit;  
 }  
  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 $this->model = new User();  
 }  
  
 #[Route('GET,POST','add')]  
 public function add()  
 {  
 if($this->request->isPost()){  
 //通过postParams属性可以增加或者覆盖前端row/a传过来的参数  
 $this->postParams['属性名']='属性的值';  
 }  
 return $this->\_add();  
 }  
  
 #[Route('GET,POST','edit')]  
 public function edit()  
 {  
 $ids = $this->request->get('ids');  
 $row = $this->model->find($ids);  
 //...可以对row进行一些前置的操作  
 if($this->request->isPost()){  
 //通过postParams属性可以增加或者覆盖前端row/a传过来的参数  
 $this->postParams['属性名']='属性的值';  
 }  
 //注意这里传入已经修改过的$row  
 return $this->\_edit($row);  
 }  
}

### 6、定义callback回调函数

在add、edit、del、multi中，开发者都可以通过定义callback回调函数来执行业务完成之后的操作

<?php  
declare (strict\_types = 1);  
  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use app\admin\traits\Actions;  
use app\common\model\User;  
use think\annotation\route\Route;  
  
#[Group("user/index")]  
class Index extends Backend  
{  
 use Actions{  
 add as protected \_add;  
 edit as protected \_edit;  
 del as protected \_del;  
 multi as protected \_multi;  
 }  
  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 $this->model = new User();  
 }  
  
 #[Route('GET,POST','add')]  
 public function add()  
 {  
 //添加完成后的操作  
 $this->callback=function($row){  
 //自增id  
 $id=$row['id'];  
 };  
 return $this->\_add();  
 }  
  
 #[Route('GET,POST','edit')]  
 public function edit()  
 {  
 //修改完成后的操作  
 $this->callback=function($row){  
  
 };  
 return $this->\_edit();  
 }  
  
 #[Route('GET,POST','del')]  
 public function del()  
 {  
 //删除完成后的操作  
 $this->callback=function($ids){  
  
 };  
 return $this->\_del();  
 }  
  
 #[Route('GET,POST','multi')]  
 public function multi()  
 {  
 //更新完成后的操作  
 $this->callback=function($ids,$field,$value){  
  
 };  
 return $this->\_multi();  
 }  
}

### 6、import导入表格

<?php  
declare (strict\_types = 1);  
  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use app\admin\traits\Actions;  
use app\common\model\User;  
use think\annotation\route\Route;  
  
#[Group("user/index")]  
class Index extends Backend  
{  
 use Actions{  
 import as protected \_import;  
 }  
  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 $this->model = new User();  
 //定义导入字段  
 $this->importFields=[  
 'username'=>'用户名',  
 'mobile'=>'手机号',  
 'nickname'=>'昵称',  
 ];  
 }  
   
 #[Route('GET,POST','import')]  
 public function import()  
 {  
 //获取到表格数据后，开发者可以在这里对数据进行处理  
 $this->callback=function($insert){  
 //返回值为整理后要插入到表格的数据  
 return $insert;  
 };  
 return $this->\_import();  
 }  
}

### 6、recyclebin回收站

回收站功能依赖于模型支持软删除

<?php  
declare (strict\_types = 1);  
  
namespace app\admin\controller\user;  
  
use app\common\controller\Backend;  
use think\annotation\route\Group;  
use app\admin\traits\Actions;  
use app\common\model\User;  
use think\annotation\route\Route;  
  
#[Group("user/index")]  
class Index extends Backend  
{  
 use Actions{  
 recyclebin as private \_recyclebin;  
 }  
  
 protected function \_initialize()  
 {  
 parent::\_initialize();  
 $this->model = new User();  
 }  
  
 //重写回收站  
 #[Route("GET,POST,JSON","recyclebin")]  
 public function recyclebin($action)  
 {  
 //定义回收站展示的字段名称  
 $this->recyclebinColumns=[  
 "id"=>"ID",  
 "nickname"=>"昵称",  
 "mobile"=>"手机号",  
 "avatar"=>"头像"  
 ];  
 //定义回收站展示的字段类型，支持text,image,images,date,datetime,tag,tags  
 $this->recyclebinColumnsType=[  
 "id"=>"text",  
 "nickname"=>"text",  
 "mobile"=>"text",  
 "avatar"=>"img"  
 ];  
 //重写回收站列表  
 if($action=='list' && $this->request->isAjax()){  
 $result = ['total' => 0, 'rows' => []];  
 return json($result);  
 }  
 return $this->\_recyclebin($action);  
 }  
}

# 权限控制

# 上传文件

1、适配器

适配器指的是上传文件后的处理方法，要实现一个适配器，必须继承抽象类，并实现下面方法

class TestUploadService extends \app\common\service\UploadService  
{  
 //保存文件  
 protected function putFile():array{}  
 //生成缩略图  
 protected function thumb(string $url,string $fullurl):string{}  
 //压缩图片  
 protected function compress(string $url,string $fullurl){}  
 //图片加水印  
 protected function watermark(string $url,string $fullurl){}  
 //删除文件  
 public static function deleteFile(Attachment $attachment){}  
}

使用适配器上传文件

$file = $this->request->file('file');  
$savename=TestUploadService::newInstance([  
 //上传配置，可不填，下面是系统默认配置参数  
 'config'=>[  
 'maxsize' => 2,  
 'mimetype' => 'jpg,png,bmp,jpeg,gif',  
 'thumb' => false,  
 'compress' => false,  
 'watermark'=>false  
 ],  
 //分类，可不填  
 'category'=>'fold-1',  
 //用户id，可不填  
 'user\_id'=>$user\_id,  
 //管理员id，可不填  
 'admin\_id'=>$admin\_id,  
 //上传文件，必填  
 'file'=>$file  
])->save();

### 2、系统自带适配器

系统自带两个适配器，公有上传app\common\service\upload\PublicUploadService  
与私有上传app\common\service\upload\PrivateUploadService

共有上传为开放文件，对外可以自由访问。私有上传对外不可访问，只有通过【admin模块别名/ajax/readfile?sha1=文件sha1值】才能访问。

### 3、使用内置表单组件form上传文件

1. 上传单个文件

{"field":"avatar","title":"上传文件","edit":"file"}  
{"field":"avatar","title":"上传文件","edit":{form:"files",multiple:false,limit:1,accept:".txt,.xls,.xlsx"}}

1. 上传多个文件

{"field":"avatar","title":"上传文件","edit":"files"}  
{"field":"avatar","title":"上传文件","edit":{form:"files",multiple:true,limit:5,accept:".txt,.xls,.xlsx"}}

1. 上传私有文件

{"field":"avatar","title":"上传文件","edit":{form:"files",disks:'local\_private',multiple:true,limit:5,accept:".txt,.xls,.xlsx"}}

### 4、使用相册

//选择单张相片  
{"field":"avatar","title":"使用相册","edit":"image"}  
{"field":"avatar","title":"使用相册","edit":{form:"attachment",limit:1}}  
//选择多张相片  
{"field":"avatar","title":"使用相册","edit":"images"}  
{"field":"avatar","title":"使用相册","edit":{form:"attachment",limit:5}}

# 队列任务

# 消息组件

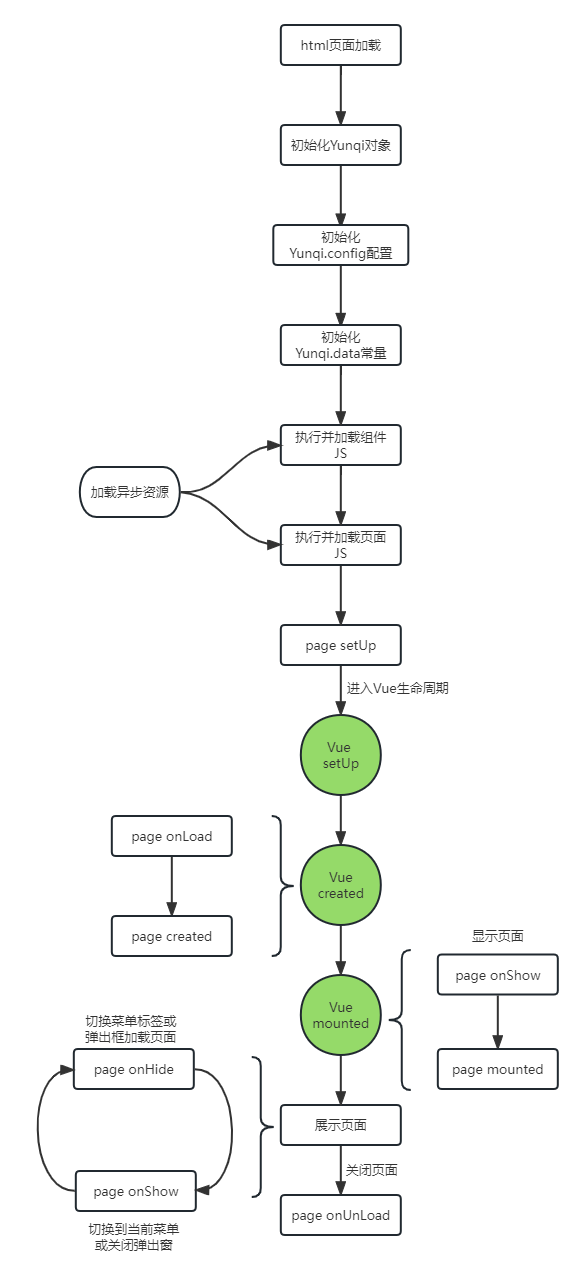
# 记录日志

# 核心对象Yunqi

Yunqi对象封装了管理后台，针对单个页面的全部数据、操作与方法。Yunqi对象核心包括以下内容：

1. Vue构造的页面应用Yunqi.app
2. 页面配置Yunqi.config
3. 页面动态数据Yunqi.data
4. 页面常用的操作，如弹窗，放大缩小Yunqi.api
5. ajax请求操作Yunqi.ajax
6. 在主窗口中使用ElementPlus的消息组件Yunqi.message  
    Yunqi.confirm  
    Yunqi.prompt  
    Yunqi.alert  
    Yunqi.loading
7. 其他方法

# 生命周期



# 页面对象Yunqi.app

Yunqi.app介绍

1、Yunqi.app是通过Vue3构造的一个标准的应用对象，包含了components，data，methods，watch，computed等属性，created，mounted，updated等方法。

2、每个页面都是一个独立的应用，Yunqi.app存储了该应用对象，可以在js的任意地方使用，以方便访问页面属性与方法。  
访问属性：可以直接使用Yunqi.app.属性名  
  
访问方法：可以直接使用Yunqi.app.方法名()

3、跨页面也同样能访问

let app=Yunqi.api.getApp(appid);  
//appid为每个页面唯一识别id，通过该页面Yunqi.config.window.id获取  
app.属性名  
app.方法名()

## 构造app对象示例

1、采用系统内置规范一【推荐】

export default{  
 //引入组件  
 components:{},  
 //页面属性  
 data:{  
  
 },  
 //计算属性  
 computed:{  
  
 },  
 //属性监听  
 watch:{  
  
 },  
 //页面加载完成时执行  
 onLoad:function(query){  
 console.log(query);  
 },  
 //页面初始显示或在框架内显示时执行  
 onShow:function(){  
   
 },  
 //页面在框架内隐藏时执行  
 onHide:function(){  
   
 },  
 //页面在框架内关闭时执行  
 onUnload:function(){  
   
 },  
 //页面方法  
 methods: {  
  
 }  
}

2、采用Vue3规范

export default{  
 //引入组件  
 components:{},  
 //页面属性  
 data:function (){  
 return {  
  
 }  
 },  
 //计算属性  
 computed:{  
  
 },  
 //属性监听  
 watch:{  
  
 },  
 //页面创建时共同调用的钩子  
 created:function (){...},  
 //页面挂载时共同调用的钩子  
 mounted:function (){...},  
 //页面更新时共同调用的钩子  
 updated:function (){...},  
 //页面方法  
 methods: {  
  
 }  
}

3、两种规范嵌套，不同属性之间交叉使用并不会导致冲突

下面的执行顺序依次是，onLoad，created，onShow，mounted

export default{  
 //引入组件  
 components:{},  
 //页面属性  
 data: {},  
 //计算属性  
 computed:{  
  
 },  
 //属性监听  
 watch:{  
  
 },  
 //页面创建时共同调用的钩子  
 created:function (){  
 console.log('created');  
 },  
 //页面加载时调用的钩子  
 onLoad:function(query){  
 console.log(query);  
 },  
 //页面显示时共同调用的钩子  
 onShow:function (){  
 console.log('onShow');  
 },  
 //页面挂载时共同调用的钩子  
 mounted:function (){  
 console.log('mounted');  
 },  
 //页面方法  
 methods: {  
  
 }  
}

# 页面配置Yunqi.config

Fast.config介绍

Yunqi.config存储的是当前页面的基本配置参数，该属性创建于app构造之前，所以可以作为全局常量来使用。

### Fast.config包含的基本属性

1. baseUrl - 访问当前模块的基本路由
2. version - 系统版本号，为了让js不使用本地文件缓存，debug模式下为随机字符串
3. window - 保存了当前页面唯一识别id与页面展示方式
4. elementUi - Ui框架基本设置参数
5. locale - 语言包
6. query - 当前页面的get参数，可以在Yunqi.app的onLoad(query)方法中获取到
7. route - 完整路由
8. jsFile - 当前页面构造Yunqi.app所引用的js路径
9. upload - 上传文件基本配置
10. url - 当前页面完整url

# 页面属性Yunqi.data

Yunqi.data介绍

Yunqi.data存储的是后端通过View::assign()传入到前端的所有参数，该属性创建于app构造之前，所以可以作为全局常量来使用。

### 示例

后端传入数据

$this->assign('groupList', \app\admin\model\UserGroup::column('id,name'));

前端获取数据

Yunqi.data.groupList

# 权限控制Yunqi.auth

Yunqi.auth介绍

Yunqi.auth在前端保存了当前登录管理员的相关信息，以及判断用户是否有权限的方法。

### 1、属性-Yunqi.auth.admin

存储了登录管理员的基本信息，包括管理员id，昵称，用户名，头像等

### 2、属性-Yunqi.auth.rules\_list

存储了管理员的所有操作权限

### 3、方法-Yunqi.auth.check(控制器,方法)

判断管理员是否对某个控制器、方法有操作权限，如下：

Yunqi.auth.check("app\\admin\\controller\\user\\Index","index");

# 页面方法Yunqi.api

# 页面操作

1、从父窗口打开新标签

Yunqi.api.addtabs(Object options)

options参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| id | 为菜单id时可以激活菜单 | 否 | 随机数 |
| url | 标签的url | 是 | 无 |
| icon | 标签icon | 是 | 无 |
| title | 标签title | 是 | 无 |
| close | 关闭标签时的回调方法 | 否 | 无 |

### 2、从父窗口关闭标签

Yunqi.api.closetabs(Number id,Mixed data)

args参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| id | 打开标签时的id | 是 | 无 |
| data | 执行close回调函数时传入的参数 | 否 | 无 |

### 3、从父窗口弹出新窗口

Yunqi.api.open(Object options)

options参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| url | 弹出窗口的url | 是 | 无 |
| icon | 弹出窗口icon | 否 | 无 |
| title | 弹出窗口title | 否 | 无 |
| width | 弹出窗口宽度 | 否 | 800 |
| height | 弹出窗口高度 | 否 | 500 |
| expand | 是否全屏 | 否 | false |
| close | 关闭弹窗时的回调方法 | 否 | 无 |

注：当打开新窗口中使用了<yun-form></yun-form>表单组件，提交表单后窗口将自动关闭，并调用close回调方法function(r){}，正常关闭窗口时参数r为null，提交表单关闭窗口时r为true，这样可以通过r的值来判断是否刷新数据。

### 4、从父窗口关闭弹窗

Yunqi.api.close(Number id,Mixed data)

args参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| id | 打开弹窗时的id | 是 | 无 |
| data | 执行close回调函数时传入的参数 | 否 | 无 |

### 5、预览图片

Yunqi.api.previewImg(Array img 或 String img)

img参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| img | 图片url或图片url数组 | 是 | 无 |

### 6、标签窗口最大化

Yunqi.api.expand()

### 7、标签窗口最小化

Yunqi.api.compress()

# 数据操作

1、删除数据

Yunqi.api.del(String url,String ids 或 Array ids,Function callback)

参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| url | 执行删除数据的url | 是 | 无 |
| ids | id或id数组 | 是 | 无 |
| callback | 删除后的回调函数 | 是 | 无 |

### 2、批量更新

Yunqi.api.multi(String url,String ids 或 Array ids,Object param,Function callback,Function error)

参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| url | 执行批量更新数据的url | 是 | 无 |
| ids | id或id数组 | 是 | 无 |
| param | 更新的{字段名:字段值} | 是 | 无 |
| callback | 更新成功后的回调函数 | 否 | 空 |
| error | 更新失败后的回调函数 | 否 | 空 |

# AJAX操作Yunqi.ajax

1、get请求

Yunqi.ajax.get(String url,Object data,Boolean loading,Boolean msg)

参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| url | get请求的url | 是 | 无 |
| data | get请求的参数 | 否 | 无 |
| loading | 是否显示【加载中...】的遮罩背景 | 否 | false |
| msg | 请求完成后是否弹出后台传来的msg消息 | 否 | false |

### 2、post请求

Yunqi.ajax.post(String url,Object data,Boolean loading,Boolean msg)

参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| url | post请求的url | 是 | 无 |
| data | post请求的参数 | 否 | 无 |
| loading | 是否显示【加载中...】的遮罩背景 | 否 | true |
| msg | 请求完成后是否弹出后台传来的msg消息 | 否 | true |

### 3、content-type为application/json的post请求

Yunqi.ajax.json(String url,Object data,Boolean loading,Boolean msg)

参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 说明 | 必填 | 默认值 |
| url | json请求的url | 是 | 无 |
| data | json请求的参数 | 否 | 无 |
| loading | 是否显示【加载中...】的遮罩背景 | 否 | true |
| msg | 请求完成后是否弹出后台传来的msg消息 | 否 | true |

# 消息Yunqi.message等

Yunqi将ElementPlus常用的消息组件封装在主窗口中使用，这样的好处是让弹出的消息位置不受页面位置，大小的影响，存在遮罩层时，能铺满整个屏幕等。

等同ElementPlus.ElMessage;

等同ElementPlus.ElMessageBox.alert

等同ElementPlus.ElMessageBox.confirm

等同ElementPlus.ElMessageBox.prompt

等同ElementPlus.ElLoading.service

# 引用JS文件Yunqi.use

Fast.use(String file 或 Array file)，采用异步加载文件，返回一个promise对象，当全部文件加载完成后执行then方法。

引入js文件后，新加载到window中的属性会通过then(args)方法的参数args返回来。

引入多个js文件时，他们的顺序并不一定会按照数组的顺序，所以不能相互依赖。

### 1、只加载一个js文件

export default{  
 data:{  
 echarts:'',  
 },  
 onLoad:function (){  
 Yunqi.use('/assets/js/echarts.min.js').then(echarts=>{  
 this.echarts=echarts;  
 });  
 },  
 methods:{  
   
 }  
}

### 2、加载多个js文件

export default{  
 data:{  
 activeName:'first',  
 echarts:'',  
 jQuery:''  
 },  
 onLoad:function (){  
 Yunqi.use(['/assets/js/echarts.min.js','https://cdn.staticfile.org/jquery/1.10.2/jquery.min.js']).then(res=>{  
 let {echarts,jQuery}=res;  
 this.echarts=echarts;  
 this.jQuery=jQuery;  
 });  
 },  
 methods:{  
 }  
}

# 内置组件

# 通用表格 - Table

1、Table属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 可选值 | 默认值/必填 |
| columns | 表格列组成的数组 | Array[Object column] | 【查看2】 | 空/是 |
| extend | 操作链接组成的Object，如下： { index\_url:'查看url', add\_url:'添加url', edit\_url:'修改url', del\_url:'删除url', multi\_url:'更新url', import\_url:'导入url', download\_url:'下载url', recyclebin\_url:回收站url } | Object |  | 空/是 |
| toolbar | 左边操作栏，多个操作用逗号隔开，需要与extend配合使用 | String | refresh【刷新】 add【添加】 edit【修改】 del【删除】 import【导入】 recyclebin【回收站】 more【显示，隐藏】 | refresh,add,edit,del/否 |
| auth | 控制toolbar是否显示的权限 | Object | add【显示添加】 edit【显示修改】 del【显示删除】 import【显示导入】 download【显示下载】 recyclebin【显示回收站】 multi【显示更多】 | { add:true, edit:true, del:true, import:true, download:true, recyclebin:true, multi:true } /否 |
| search | 普通搜索关联字段，多个字段用逗号隔开，不填则不显示普通搜索 | String |  | 空/否 |
| common-search | 是否启用高级筛选 | Boolean |  | true/否 |
| search-form-visible | 默认是否显示高级筛选 | Boolean |  | true/否 |
| pagination | 是否启用分页 | Boolean |  | true/否 |
| pk | 主键 | String |  | id/否 |
| sort-name | 排序字段 | String |  | id/否 |
| order | 排序方式 | String | asc或desc | desc/否 |
| tabs | 增加顶部选项卡，column的field是筛选条件，column的searchList是筛选选项 | String | column的field | 空/否 |
| is-tree | 是否为树形表结构，树形结构，列表数据需要有id与pid字段 | Boolean |  | false/否 |
| tree-expand-all | 树形表结构是否默认全部展开 | Boolean |  | false/否 |
| add-form | 自定义添加表单样式 | Object |  | { icon:'fa fa-plus-square-o', title:'添加'， width:800, height:500, expand:false } /否 |
| edit-form | 自定义修改表单样式 | Object |  | { icon:'fa fa-pencil-square-o', title:'编辑', width:800, height:500, expand:false } /否 |
| style | 自定义表格样式 | Object |  | {width:'100%'}/否 |
| show-summary | 底部显示统计 | Boolean |  | false/否 |
| @render | 表格渲染完成事件 | Function(data) |  | 空/否 |

### 2、column属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 可选值 | 默认值 |
| field | 列字段名，多级数据可以在字段中间加个”.“，如user.group.name | String | operate | 空 |
| title | 表头标题 | String |  | 空 |
| align | 对齐 | String | left,center,right | center |
| width | 列宽 | Number |  | 空 |
| sortable | 是否支持排序 | Boolean |  | false |
| visible | 是否隐藏，true为显示，false为隐藏，none为不渲染该列 | Boolean 或 String | true，false，none | true |
| formatter | 渲染方式 | Object 或 Function | 【查看3】 | Fast.formatter.text |
| operate | 高级搜索选项 | Boolean 或 Object 或 String | 【查看4】 【查看5】 | = |
| searchList | 选择项 1、渲染表格时，字段的值会被替换成searchList[字段的值] 2、formatter为select或switch时会被作为选项 3、operate为select，radio，switch，checkbox时会被作为选项 4、下载表格时字段的值会被替换 | Object/Function 【注：为Function时必须返回一个Object键值对 |  | undefined |
| checkbox | 选择框 | Boolean |  | undefined |
| selectable | 配合选择框使用，表示是否可选，返回true时，表示可选，返回false表示不可选 | Function(row,index) |  | undefined |
| treeExpand | 树形表格时的展开，收起按钮 | Boolean |  | undefined |
| action | field值为operate时，提供的操作按钮 | Object | 【查看6】 | undefined |
| fixed | 固定列 | String | right,left | undefined |

### 3、formatter渲染方式

|  |  |  |
| --- | --- | --- |
| 渲染方式 | 说明 | 默认值 |
| Fast.formatter.text | 渲染成文本 | { \_name:'text', value:'字段的值' } |
| Fast.formatter.image | 渲染成单张图片 | { \_name:'image', value:'字段的值' } |
| Fast.formatter.images | 渲染成多张图片 | { \_name:'images', value:['字段的值1','字段的值2',...] } |
| Fast.formatter.link | 渲染成超链接 | { \_name:'link', value:'字段的值' } |
| Fast.formatter.datetime | 渲染成日期时间，格式精确到分钟 2023-01-01 23:55 | { \_name:'datetime', value:'字段的值' } |
| Fast.formatter.date | 渲染成日期 | { \_name:'date', value:'字段的值' } |
| Fast.formatter.tag | 渲染成标签 | { \_name:'tag', effect:'dark', type:'primary', color:'', round:false, closable:false, close:'',//关闭事件 click:'',//点击事件 value:'字段的值' } |
| Fast.formatter.tags | 渲染成标签组 | { \_name:'tags', effect:'dark', type:'primary', color:'', round:false, closable:false, close:'',//关闭事件 click:'',//点击事件 value:['字段的值1','字段的值2',...] } |
| Fast.formatter.switch | 渲染成开关，与searchList配合使用 | { \_name:'switch', value:searchList['字段的值'], inactiveValue:0, activeValue:1, disabled:false } |
| Fast.formatter.select | 渲染成下拉框，与searchList配合使用 | { \_name:'select', value:searchList['字段的值'], disabled:false } |
| Fast.formatter.html | 渲染html标签 | { \_name:'html', value:'字段的值' } |
| Fast.formatter.button | 渲染成一个按钮，click属性表示点击事件，可以支持回调函数或者字符串。字符串填写当前的一个method名称 | { \_name:'button', title:'按钮', type:'primary', click:function(){} } |
| Fast.formatter.slot | 自定义插槽 | 【查看7】 |
| Function(data,row) | 通过函数的返回值来渲染，回调参数data为字段的值，row为整行数据。返回值可以是数字，字符串，也可以是formatter对象。 |  |

### 4、operate属性

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 可选项 |
| form | 表单类型 | String | input【文本框】 select【下拉框】 radio【单选框】 checkbox【复选框】 switch【开关】 between【并排输入框】 date-picker【选日期】 time-picker【选时间】 cascader【树形关联表】 selectpage【分页关联表】 |
| type | 1、hidden时隐藏过滤， 2、表单为input,date-picker、time-picker时，对应具体类型 | String | 1、hidden，参与过滤，但不显示 2、form等于input：text,number等input基本类型 3、form等于date-picker：date,year,dates,month,datetime,daterange 4、form等于time-picker：time，timerange |
| value | 默认值 | String 或 Number 或 Array | 当form为多选时，比如checkbox，默认值必须是数组 |
| size | 行宽，默认为default | String | small,default,large |
| placeholder | 输入提示 | String |  |
| multiple | 是否是多选，只在form为select,selectpage下才有效 | Boolean |  |
| filter | 过滤的方式，默认为“=”，为false的时候不参与自动过滤，过滤方式请参考app\common\controller\Backend的getBuildWhere方法 | String 或 Boolean | false【不参与过滤】 = > >= < <= != <> > TIME >= TIME < TIME <= TIME LIKE NOT LIKE FIND\_IN\_SET NOT FIND\_IN\_SET IN NOT IN BETWEEN NOT BETWEEN BETWEEN TIME NOT BETWEEN TIME IS NULL IS NOT NULL |
| prepend | form为input时的前缀 | String |  |
| append | form为input时的后缀 | String |  |
| url | 1、form为selectpage时获取数据的url 2、form为cascader时动态获取数据时的url | String |  |
| labelField | form为selectpage时显示的字段 | String |  |
| keyField | form为selectpage时作为value的字段 | String |  |
| options | form为cascader时，非动态获取数据时的选项 | Object | Object数据源，参考element plus[【查看文档】](https://element-plus.org/zh-CN/component/cascader.html) |
| props | form为cascader时的配置项 | Object | { expandTrigger:'click',//触发方式 multiple:false,//是否为多选 value:'id',//指定数据源值字段 label:'name',//指定数据源显示字段 children:'children',//指定数据源子选项字段 lazy:false,//是否动态加载 ... } 更多属性参考element plus [【参考文档】](https://element-plus.org/zh-CN/component/cascader.html#cascaderprops) |
| level | form为cascader时，props:{laze:true}，动态获取数据的层级 | Number |  |

### 5、operate简写

|  |  |  |
| --- | --- | --- |
| 简写名称 | 说明 | 等价的完整写法 |
| = | 等于 | { form:'input', type:'text', value:'', size:'default', placeholder:'请输入+字段名', filter:'=' } |
| <>或!= | 不等于 | { form:'input', type:'text', value:'', size:'default', placeholder:'请输入+字段名', filter:'=' } |
| like或LIKE | 相似 | { form:'input', type:'text', value:'', size:'default', placeholder:'请输入+字段名', filter:'LIKE' } |
| not like或NOT LIKE | 不相似 | { form:'input', type:'text', value:'', size:'default', placeholder:'请输入+字段名', filter:'NOT LIKE' } |
| select或SELECT | 下拉选择 | { form:'select', value:'', size:'default', placeholder:'请选择+字段名', filter:'=' } |
| checkbox或CHECKBOX | 多项选择 | { form:'checkbox', value:[], size:'default', filter:'IN' } |
| radio或RADIO | 单项选择 | { form:'radio', value:'', size:'default', filter:'=' } |
| switch或SWITCH | 开关 | { form:'switch', value:'', size:'default', filter:'=' } |
| between或BETWEEN | 在两组数之间 | { form:'between', value:[], size:'default', filter:'between' } |
| find\_in\_set或FIND\_IN\_SET | 在字符串str1,str2,str3...中 | { form:'select', value:'', size:'default', filter:'FIND\_IN\_SET' } |
| date或DATE | 等于日期 | { form:'date-picker', type:'date', value:'', size:'default', filter:'=' } |
| datetime或DATETIME | 等于日期时间 | { form:'date-picker', type:'datetime', value:'', size:'default', filter:'=' } |
| daterange或DATERANGE | 在日期期间 | { form:'date-picker', type:'daterange', value:[], size:'default', filter:between time'' } |
| time或TIME | 等于时间 | { form:'time-picker', value:'', type:'time', size:'default', filter:'=' } |
| timerange或TIMERANGE | 在时间区间 | { form:'time-picker', value:[], size:'default', type:'timerange', filter:'between' } |
| area或AREA | 选区域，指定到区县，动态加载 | { form:'cascader', value:'', size:'default', url:'ajax/area', level:3, props:{ expandTrigger:'click', value:'id', label:'name', multiple:false, lazy:true }, filter:'=' } |
| category或CATEGORY | 选分类，动态加载 | { form:'cascader', value:'', size:'default', url:'ajax/category', level:2, props:{ expandTrigger:'click', value:'id', label:'name', children:'childlist', multiple:false, lazy:true }, filter:'=' } |

### 6、操作按钮

内置按钮，edit，del，expand，sort

{  
 field: 'operate',  
 fixed: 'right',  
 title: \_\_('操作'),  
 width:140,  
 action:{  
 //编辑  
 edit:true,  
 //删除  
 del:true,  
 //排序  
 sort:true,  
 }  
}

使用函数，控制按钮是否展示

{  
 field: 'operate',  
 fixed: 'right',  
 title: \_\_('Operate'),  
 width:140,  
 action:{  
 add:function(row){  
 //返回true表示展示，返回false表示隐藏  
 return true;  
 }  
 }  
}

通过Object自定义按钮

{  
 field: 'operate',  
 fixed: 'right',  
 title: \_\_('Operate'),  
 width:140,  
 action:{  
 detail:{  
 tooltip:false,  
 type:'primary',  
 icon:'fa fa-list',  
 text:'详情',  
 visible:function(row){  
 return true;  
 },  
 method:'showDetail'  
 }  
 }  
}

|  |  |  |
| --- | --- | --- |
| 属性名 | 说明 | 类型 |
| tooltip | 是否显示提示 | Boolean |
| text | 提示文本或者不提示时按钮文本 | String |
| type | 按钮类型，参考elemnt plus按钮类型 | String |
| icon | 按钮图标 | String |
| method | 点击事件 1、当值为String，当前页面定义的method名称 2、当值为Function(row)，点击时的回调函数 | String 或 Function |
| visible | Function(row)，根据返回值控制按钮是否展示 | Function |

### 7、插槽slot

|  |  |  |
| --- | --- | --- |
| 插槽名称（name） | 说明 | 属性及说明 |
| toolbar | 增加左上角的操作按钮 | selections【选择框选中的行】 tool【toolbar属性中对应的属性名称】 |
| formatter | formatter属性值为Fast.formatter.slot时，自定义的展示列 | field【列的field属性】 rows【行数据】 searchList【当前列的searchList属性】 |
| header | 自定义表头标题 | field【属性名】 |

### 8、常用属性与方法

//表格写法，通过ref绑定组件指针  
<yun-table  
 :columns="columns"  
 ref="yuntable"  
 toolbar="refresh,edit,del,more">  
</yun-table>  
  
//js写法  
 data:{  
 columns:[...]  
 },  
 methods: {  
 test:function (){  
 this.$refs.yuntable.属性名;  
 this.$refs.yuntable.方法名();  
 }  
 }

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| list | Array[Object data] | 表格数据 |
| selections | Array[Object data] | 多选框被选择的数据 |
| currentPage | Number | 当前页 |
| total | Number | 总行数 |
| reset | Function() | 重新渲染整个组件，当columns修改时，需要重新渲染表格才能生效，可以执行该方法。 |
| render | Function(list) | 重新渲染数据，当表格内的数据变化时，或者传入自定义表格数据，可以通过该方法来实现 |
| reload | Function() | 保持所在的页面，重新获取数据 |
| submit | Function() | 返回第一页，重新获取数据 |
| expandAllTree | Function() | 树形表格展开所有节点 |
| expandTree | Function(topid) | 树形表格展开 指定节点 |

# 通用表单 - Form

1、Form属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 可选值 | 默认值/必填 |
| columns | 表单项 | Array[Object column] | 【查看2】 | 空/是 |
| data | 表单默认值 | Object data |  | {}/否 |
| action | 提交表单的url | String |  | 当前页面url/否 |
| steps | 按步骤提交表单 | Array[String title] |  | 空/否 |
| tabs | 给表单分栏 | Array[String title] |  | 空/否 |
| label-position | 表单项标题所在位置 | String | left,right,top | right/否 |
| label-width | 表单项标题宽度 | Number |  | 120/否 |
| require-asterisk-position | 必填表单项红色“\*”所在位置 | String | left,right | right/否 |
| append-width | 表单项尾部追加一列的宽度，总宽度为24，与append插槽搭配使用 【查看7】 | Number | 1~12 | 0/否 |
| @render | 表单渲染完成后触发的事件 | Function(data) |  | 空/否 |
| @submit | 提交事件，返回true则继续提交，返回false则取消提交 | Function(data) |  | 空/否 |
| @success | 提交成功事件 | Function(response) |  | 空/否 |
| @fail | 提交失败事件 | Function(error) |  | 空/否 |
| @step | 当表单为按步骤提交时，点击上一步、下一步触发的事件 | Function(step) |  | 空/否 |

### 2、column属性

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 可选值 |
| field | 唯一识别标记 | String |  |
| title | 表单项标题 | String |  |
| edit | 构造表单内容 | String 或 Object | 【查看3】 【查看4】 |
| visible | 是否显示，当为Function的时候，返回true为显示，false为隐藏 | Boolean 或 Function(row) |  |
| searchList | 当edit.form为select，radio，switch，checkbox时会被作为选项 |  |  |
| rules | 验证规则 | String 或 Object 或 Array | 【查看5】 【查看6】 |
| step | 所属的步骤，0为第一步 | Int |  |
| tab | 所属的分栏，0为第一栏 | Int |  |

### 3、edit属性

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 可选项 |
| form | 表单类型 | String | input【文本框】 select【下拉框】 radio【单选框】 checkbox【复选框】 editor【富文本】 switch【开关】 files【上传文件】 attachment【相册】 fieldlist【Json输入框】 date-picker【日期选择】 time-picker【时间选择】 cascader【树形关联表】 selectpage【分页关联表】 slot【自定义插槽】 |
| type | 表单为input,date-picker,time-picker时，对应具体类型 | String | 1、form等于input：text,number、textarea等input基本类型 2、form等于date-picker：date,year,dates,month,datetime,daterange 3、form等于time-picker：time、timerange |
| name | 提交表单时的name属性，不填写会默认提交为 row[field的值] |  |  |
| value | 默认值，当form为多项选择，如checkbox时，value须为数组，当value的值为function时，根据返回的值设置value | String 或 Number 或 Array 或 Function |  |
| placeholder | 提示输入内容 | String |  |
| multiple | form为select，selectpage时是否为多项选择，form为files时是否允许上传多个文件 | Boolean |  |
|  |  |  |  |
| checkAll | form为select时，默认全选 | Boolean |  |
|  |  |  |  |
| prepend | form为input时的前缀 | String |  |
| append | form为input时的后缀 | String |  |
| readonly | form为input时只读状态 | String |  |
| width | form为editor时的宽度 | String 或 Number |  |
| height | form为editor时的高度 | String 或 Number |  |
| rows | form为input，type为textarea时的行数 | Number |  |
| activeValue | form为switch时选中的值 | String或Number或Boolean |  |
| inactiveValue | form为switch时未选中的值 | String或Number或Boolean |  |
| accept | form为files时允许上传文件的类型 | String |  |
| limit | 1、form为files，multiple为true时允许上传文件的个数 2、form为attachment允许选中的图片张数 | Number |  |
| label | form为fieldlist时，数组的标题部分，类型 | Array[String,String,...] |  |
| url | 1、form为selectpage时获取数据url 2、form为cascader且动态获取数据时的url | String |  |
| labelField | form为selectpage时显示的字段 | String |  |
| keyField | form为selectpage时作为value的字段 | String |  |
| options | form为cascader且为静态数据时的选项 | String 或 Object | Object数据源，参考element plus[【查看文档】](https://element-plus.org/zh-CN/component/cascader.html) |
| props | form为cascader时的配置项 | Object | { expandTrigger:'click',//触发方式 multiple:false,//是否为多选 value:'id',//指定数据源值字段 label:'name',//指定数据源显示字段 children:'children',//指定数据源子选项字段 lazy:false,//是否动态加载 ... } 更多属性参考element plus [【参考文档】](https://element-plus.org/zh-CN/component/cascader.html#cascaderprops) |
| level | form为cascader时，props:{laze:true}，动态获取数据的层级 | Number |  |
| blur | 当表单项失去焦点时的回调函数 | String 或 Function(data,row) | 1、当blur为String时，回调当前页面同名的方法 2、当blur为函数时直接回调该函数 |
| change | 当表单项值改变时的回调函数 | String 或 Function(data,row) | 1、当change为String时，回调当前页面同名的方法 2、当change为函数时直接回调该函数 |

### 4、edit简写

|  |  |  |
| --- | --- | --- |
| 简写名称 | 说明 | 等价的完整写法 |
| hidden | 隐藏的表单项 | { form:'input', type:'hidden', value:'', placeholder:'请输入+字段名', } |
| text 或 TEXT | 文本输入框 | { form:'input', type:'text', value:'', placeholder:'请输入+字段名', } |
| readonly 或 READONLY | 只读输入框 | { form:'input', type:'text', readonly:true, value:'', placeholder:'请输入+字段名', } |
| password 或 PASSWORD | 密码输入框 | { form:'input', type:'password', value:'', placeholder:'请输入+字段名', } |
| textarea 或 TEXTAREA | 多行文本输入框 | { form:'input', type:'textarea', value:'', rows:4, placeholder:'请输入+字段名', } |
| number 或 NUMBER | 数字输入框 | { form:'input', type:'number', value:'', placeholder:'请输入+字段名', } |
| editor 或 EDITOR | 富文本编辑器 | { form:'editor', width:'100%', height:'350px' } |
| select 或 SELECT | 下拉框 | { form:'select', value:'', placeholder:'请选择+字段名', } |
| radio 或 RADIO | 单项选择 | { form:'radio', value:'', } |
| switch 或 SWITCH | 开关 | { form:'switch', activeValue:'searchList第1个键', inactiveValue:'searchList第2个键', value'searchList第1个键', } |
| date 或 DATE | 日期选择框 | { form:'date-picker', type:'date', value:'', placeholder:'请输入+字段名', } |
| datetime 或 DATETIME | 日期时间选择框 | { form:'date-picker', type:'datetime', value:'', placeholder:'请输入+字段名', } |
| daterange 或 DATERANGE | 日期区间选择框 | { form:'date-picker', type:'daterange', value:[], placeholder:'请输入+字段名', } |
| time 或 TIME | 时间选择框 | { form:'time-picker', type:'time', value:'', } |
| timerange或 TIMERANGE | 时间区间选择框 | { form:'time-picker', type:'timerange', value:[], } |
| area或AREA | 选区域，指定到区县，动态加载 | { form:'cascader', value:'', url:'ajax/area', level:3, props:{ expandTrigger:'click', value:'id', label:'name', multiple:false, lazy:true } } |
| category或CATEGORY | 选分类，静态加载 | { form:'cascader', value:'', url:'ajax/category', props:{ expandTrigger:'hover', value:'id', label:'name', children:'childlist', multiple:false } } |
| image 或 IMAGE | 上传单张图片 | { form:'attachment', limit:1, value:'', } |
| images 或 IMAGES | 上传多张图片 | { form:'attachment', limit:10, value:[], } |
| file 或 FILE | 上传单个文件 | { form:'files', value:'', accept:'后端配置允许上传文件的类型', multiple:false } |
| files 或 FILES | 上传多个文件 | { form:'files', value:[], accept:''后端配置允许上传文件的类型, multiple:true, limit:10 } |
| fieldlist 或 FIELDLIST | JSON输入框 | { form:'fieldlist', value:{}, label:['键名','键值'] } |
| slot 或 SLOT | 自定义插槽【查看7】 |  |

### 5、rules写法

|  |  |
| --- | --- |
| 类型 | 说明 |
| String | rules简写【查看6】 |
| Function(rule, value, callback) | Element-plus自定义验证函数[【查看文档】](https://element-plus.org/zh-CN/component/form.html#%E8%87%AA%E5%AE%9A%E4%B9%89%E6%A0%A1%E9%AA%8C%E8%A7%84%E5%88%99) |
| Object{required,validator,message,trigger} | Element-plus表单验证async-validator[【查看文档】](https://element-plus.org/zh-CN/component/form.html#%E8%87%AA%E5%AE%9A%E4%B9%89%E6%A0%A1%E9%AA%8C%E8%A7%84%E5%88%99) |
| Array[Object{required,validator,message,trigger}] | Element-plus表单验证async-validator[【查看文档】](https://element-plus.org/zh-CN/component/form.html#%E8%A1%A8%E5%8D%95%E6%A0%A1%E9%AA%8C) |

### 6、rules简写

多个规则用逗号隔开

|  |  |  |
| --- | --- | --- |
| 写法名称 | 说明 | 扩展 |
| required | 必填 |  |
| checked | 必选 | checked 必选 checked(n) 必选n项 checked(n~) 至少选择n项 checked(~n) 至多选择n项 checked(n1~n2) 选择n1到n2项 |
| integer | 整数 | integer 填写整数 integer(+) 填写正整数 integer(+0) 填写正整数和0 integer(-) 填写负整数 integer(-0) 填写负整数和0 |
| range | 数值范围 | range(n~) 填写不小于n的数 range(~n) 填写不大于n的数 range(n1~n2) 填写n1到n2的数 |
| length | 字符长度 | length(n) 填写n个字符 length(n~) 至少填写n个字符 length(~n) 最多填写n个字符 length(n1~n2) 填写n1到n2个字符 |
| chinaChar | 中文 |  |
| unChinaChar | 非中文 |  |
| mobile | 手机号 |  |
| email | 邮箱 |  |
| idcard | 身份证号 |  |

### 7、插槽slot

|  |  |  |
| --- | --- | --- |
| 插槽名称（name） | 说明 | 属性及说明 |
| default | 默认插槽，可以在表单开始的地方追加内容 |  |
| append | 表单项结尾追加内容，与表单属性append-width配合使用 | column【表单项全部参数】 |
| 表单项的field | edit为slot时，自定义表单项 | rows【所有表单项的值】 |
| footer | 自定义表单底部按钮 | step【按步骤提交表单的step】 |

### 8、常用方法

//表单写法，通过ref绑定组件指针  
<yun-form  
 :columns="columns"  
 ref="yunform"  
>  
</yun-form>  
  
  
//js写法  
 data:{  
 columns:[...]  
 },  
 methods: {  
 test:function (){  
 this.$refs.yunform.属性名;  
 this.$refs.yunform.方法名();  
 }  
 }

|  |  |
| --- | --- |
| 方法名称 | 说明 |
| submit | 提交表单 |
| reset | 重置表单 |
| showField(field) | 显示表单项 |
| hideField(field) | 隐藏表单项 |
| setField(field,key,value) | 设置表单项某个属性的值 比如表单项为 {field: 'name', title: '名称',edit:'text',rules:'required'} setField('name','title','修改后的名称'); setField('name','edit','select'); 这样就实现了修改表单项，修改完成后记得reset() |
| getValue(field) | 获取表单项的值 |
| setValue(field,value) | 设置表单项的值 |
| setError(field,message) | 让某个表单项显示错误内容 |

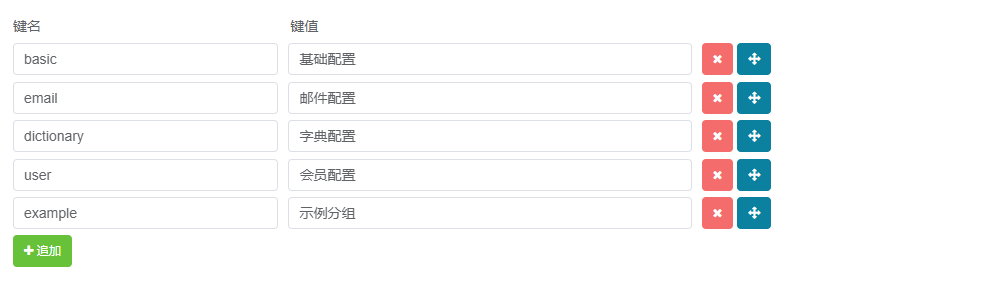
# JSON控件 - Fieldlist

1、Fieldlist属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 默认值 | 必填 |
| label | 数组标题 | Array[String,String,...] | ['键名','键值'] | 否 |
| value | 默认值 | Array 或 Object 【查看2】 | {} | 否 |
| @change | 当值改变时的事件 | Function(data) | 空 | 否 |

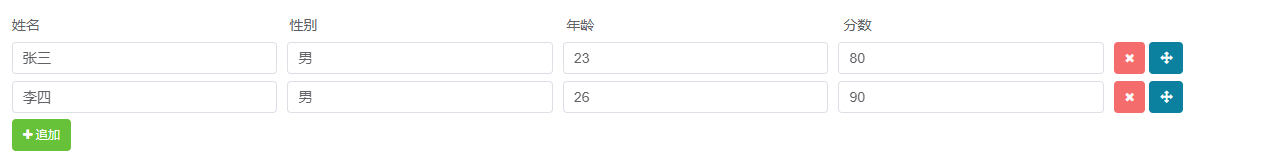
### 2、value为Object时例子与效果

<Fieldlist :value='{  
 "basic":"基础配置",  
 "email":"邮件配置",  
 "dictionary":"字典配置",  
 "user":"会员配置",  
 "example":"示例分组"  
}'></Fieldlist>



### 3、value为Array时例子与效果

<Fieldlist :label='["姓名","性别","年龄","分数"]' :value='[  
 {"name":"张三","gender":"男","age":"23","score":"80"},  
 {"name":"李四","gender":"男","age":"26","score":"90"}  
 ]'>  
</Fieldlist>



### 4、自定义列插槽

当value值为Object对象时

|  |  |  |
| --- | --- | --- |
| 插槽名称-name | 说明 | 参数及说明 |
| 0 | 自定义第1列 | list【所有行组成的数组】 |
| 1 | 自定义第2列 | row【行数】 |

export default{  
 components:{  
 'Fieldlist':fieldlist,  
 },  
 data:{  
 json:{  
 "name":"张三",  
 "sex":"男",  
 "age":"18",  
 "grade":"高级",  
 }  
 }  
}  
  
<Fieldlist :value="json">  
<template #0="{list,row}">  
 <el-select v-model="list[row][0]" placeholder="请选择">  
 <el-option label="姓名" value="name"></el-option>  
 <el-option label="性别" value="sex"></el-option>  
 <el-option label="年龄" value="age"></el-option>  
 <el-option label="班级" value="grade"></el-option>  
 </el-select>  
</template>  
</Fieldlist>

当value值为Array数组时

|  |  |  |
| --- | --- | --- |
| 插槽名称-name | 说明 | 参数及说明 |
| Array[Object{key:value}]的key | 自定义key所在的列 | list【所有行组成的数组】 row【行数】 |

import fieldlist from "../../components/Fieldlist.js";  
export default{  
 components:{  
 'Fieldlist':fieldlist,  
 },  
 data:{  
 json:[  
 {"user\_id":1,"gender":"男","age":"23","score":"80"},  
 {"user\_id":2,"gender":"女","age":"26","score":"90"}  
 ],  
 students:[  
 {id:1,name:'张三'},  
 {id:2,name:'李四'},  
 {id:3,name:'隔壁老王'},  
 ]  
 }  
}  
  
<Fieldlist :label='["姓名","性别","年龄","分数"]' :value="json">  
 <template #user\_id="{list,row}">  
 <el-select filterable placeholder="请选择学生" v-model="list[row].user\_id">  
 <el-option  
 v-for="item in students"  
 :key="item.id"  
 :label="item.name"  
 :value="item.id">  
 </el-option>  
 </el-select>  
 </template>  
 <template #gender="{list,row}">  
 <el-select filterable placeholder="请选择性别" v-model="list[row].gender">  
 <el-option label="男" value="boy"></el-option>  
 <el-option label="女" value="girl"></el-option>  
 <el-option label="不男不女" value="太监"></el-option>  
 </el-select>  
 </template>  
</Fieldlist>

# 分页关联表 - SelectPage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 默认值 | 必填 |
| url | 关联表url | String | 空 | 是 |
| key-field | 作为value的字段 | String | id | 否 |
| label-field | 显示的字段 | String | name | 否 |
| value | 默认值，多选时默认值为数组 | Number 或 Array | 空 | 否 |
| page-size | 每页显示几行数据 | Number | 7 | 否 |
| multiple | 是否多选 | Boolean | false | 否 |
| @change | 当值改变时的事件 | Function(data) | 空 | 否 |

# 上传头像 - UploadImg

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 默认值 | 必填 |
| image-url | 默认图片链接地址 | String | 空 | 否 |
| width | 预览图片宽度 | Number | 150 | 否 |
| height | 预览图片高度 | Number | 150 | 否 |
| is-circle | 是否为圆角 | Boolean | false | 否 |
| disabled | 禁止编辑 | Boolean | false | 否 |
| 插槽名称-name | 说明 | 参数及说明 |
| title | 没有头像时显示在中间的标题 |  |

# 相册 - Attachment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 默认值 | 必填 |
| limit | 允许选择的照片数量 | Number | 5 | 否 |
| value | 默认选中的照片 | Array | 空 | 否 |
| @change | 当选中照片后的回调事件 | Function(Array data) |  |  |

# 选择图标 - CheckIcon

1、在JS中引入组件

import checkicon from "../../components/CheckIcon.js";  
export default{  
 components:{'CheckIcon':checkicon},  
 data:{  
 icon:''  
 },  
 methods: {  
 //打开图标选择菜单  
 openIconPanel:function (){  
 this.$refs.checkicon.open();  
 },  
 //处理选择后的事件  
 selectIcon:function (i){  
 this.icon=i;  
 }  
 }  
}

### 2、在View中使用组件

<el-card shadow="never" style="border: 0;">  
 <el-input v-model="icon"></el-input>  
 <el-button size="small" @click="openIconPanel">选择图标</el-button>  
</el-card>  
<check-icon ref="checkicon" @selected="selectIcon"></check-icon>

# 绑定第三方账号 - Third

# 权限控制-Auth

判断管理员是否有权限，如果有，则展示标签内的内容，其中：  
1、管理员有权访问controller与action。  
2、管理员ID包含于admin\_id。  
3、管理员的角色ID包含于group\_id。  
4、1到3满足任意一项，都成立。

<Auth controller="app\admin\controller\user\Index" action="index" :admin\_id="[1,2,3]" :groupp\_id="[1,2,3]"></Auth>

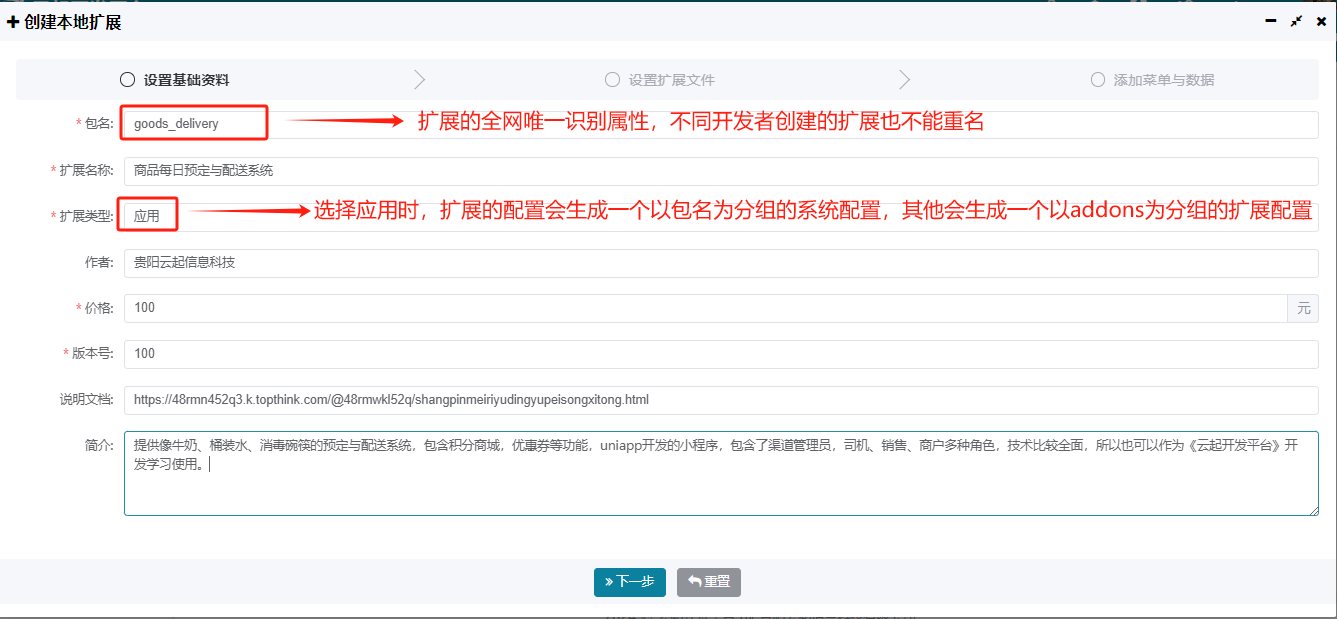
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 说明 | 类型 | 默认值 | 必填 |
| controller | 控制器名称 | String | 空 | 否 |
| action | 方法名称 | String | 空 | 否 |
| admin\_id | 管理员id | Array | 空 | 否 |
| group\_id | 角色id | Array | 空 | 否 |

# 创建与开放扩展

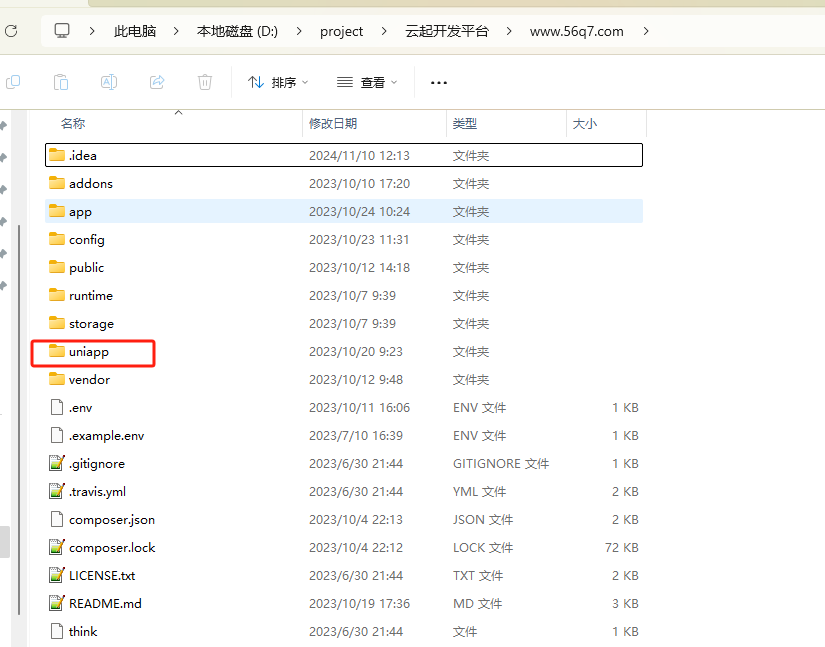
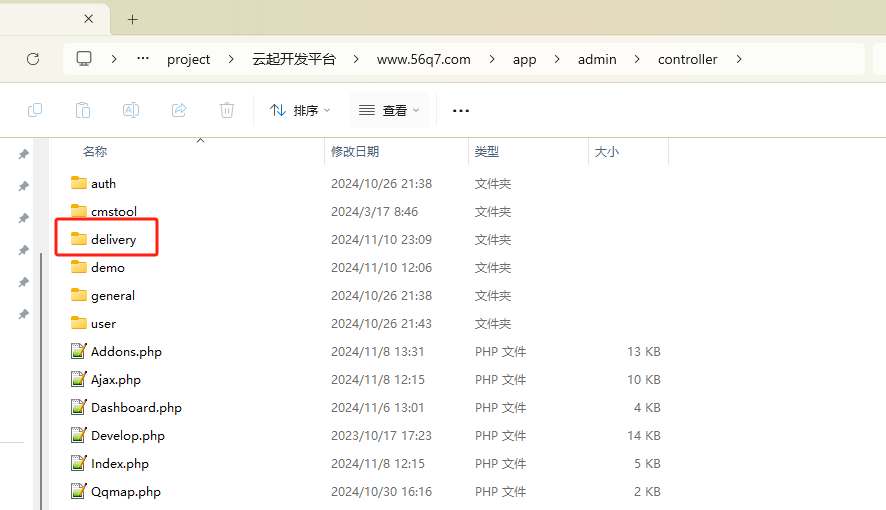
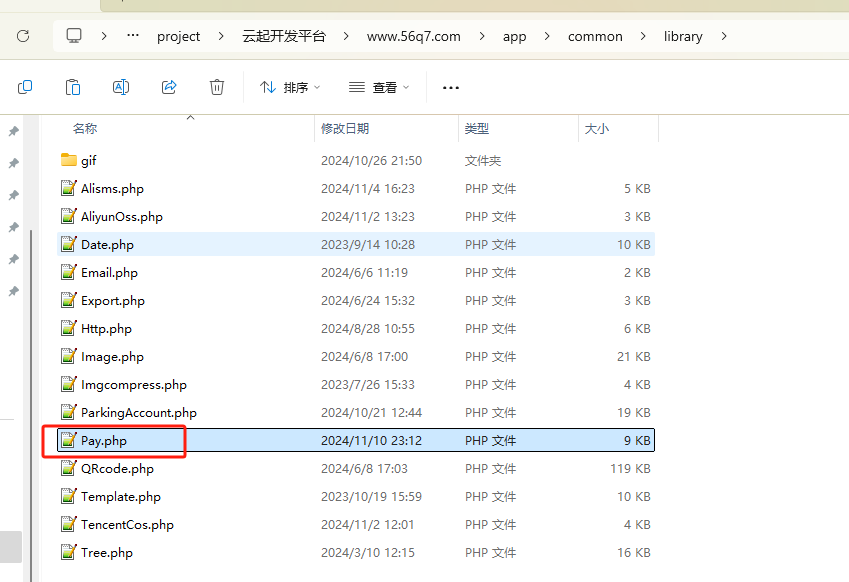
1、扩展开发的设计思想。

扩展并不存在严格的嵌入接口，扩展开发，实际上是将开发者设计好的程序，打包共享给其他开发者。所以扩展无需遵循任何规则，开发者可以将扩展开发理解为：  
扩展分享者：程序开发=》文件、配置、菜单整理=》程序打包。  
扩展安装者：扩展包下载=》扩展包安装。  

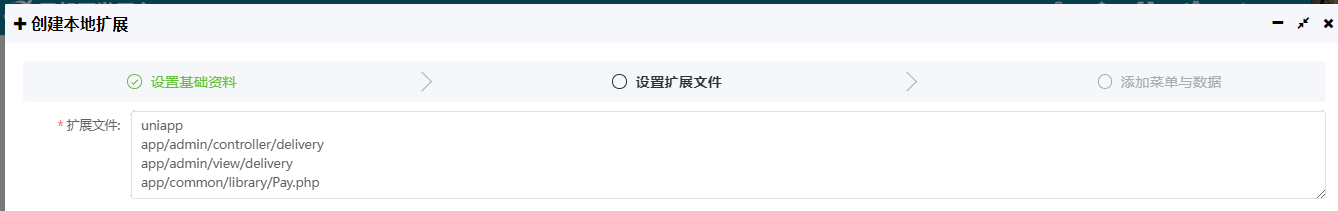

#### 2、设置包名等基础信息。



#### 3、添加扩展文件。

扩展文件可以是文件，也可以是目录，使用相对于系统根目录的相对路径，每一行输入一个文件或者目录路径。  
  
  
  


以上四个张图片标识的目录或文件就可以如下图填写：



#### 4、添加过滤文件。

过滤文件，就是不参与打包的文件，比如开发工具生成的.idea，.git，.gitignore文件或目录，文件或目录写法跟扩展文件一样。

#### 5、添加依赖文件。

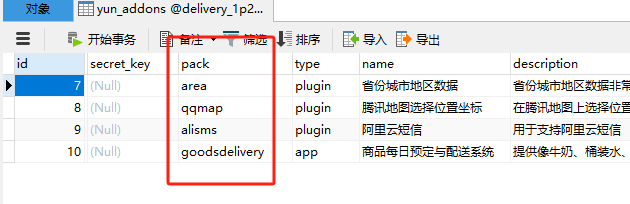
依赖文件，主要指依赖composer添加的库，系统通过判断依赖的类是否存在来判断依赖库是否添加，所以这里可以随便挑选几个依赖的类来填写。  
比如：  
腾讯云COS依赖库是，qcloud/cos-sdk-v5，这里只用填写

\Qcloud\Cos\Client

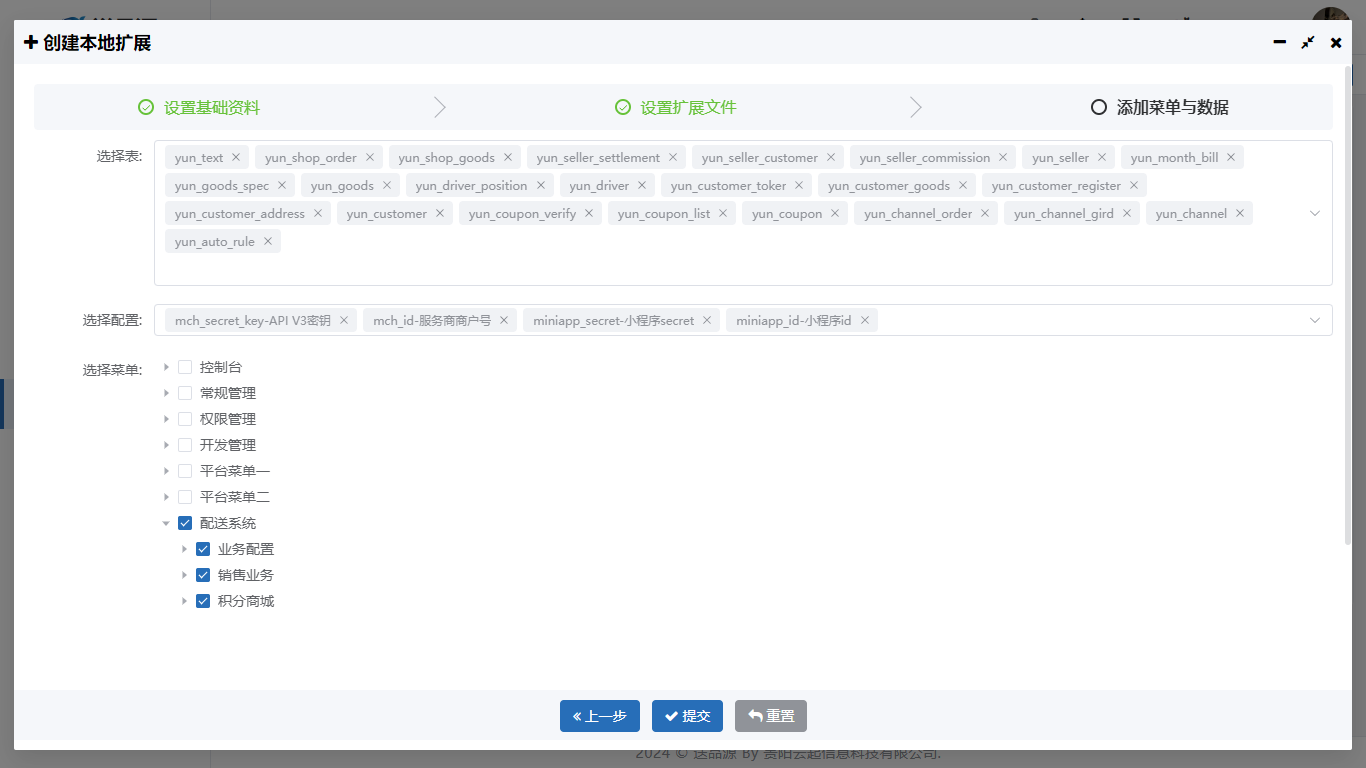
阿里云OSS依赖库是，aliyuncs/oss-sdk-php，这里只用填写

\OSS\OssClient

#### 6、添加依赖扩展。

依赖扩展指的是，云起开发平台上存在的扩展，这里填写扩展的包名，包名可以在数据库的【表前缀\_addons】表中查找到。

#### 7、添加数据库、系统配置项，菜单。



#### 9、扩展创建完成与Install.php勾子编写。

上一步提交后，在addons/【扩展类型】会增加一个名为【扩展包名】的目录，并生成一个Install.php的安装文件。



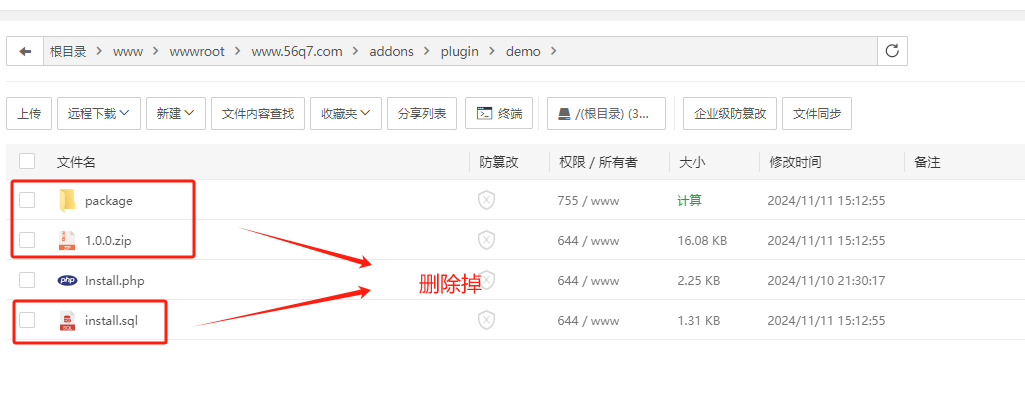
打开编辑Install.php文件，里面有两个静态方法，可以编写这两个方法，解决安装与卸载时需要额外处理的事务，比如给表增加字段，删除表的字段等。

//安装扩展时的回调方法  
public static function install()  
{  
  
}  
  
//卸载扩展时的回调方法  
public static function uninstall()  
{  
  
}

#### 10、扩展打包。

处理完上一步，就可以点击按钮给扩展打包，打包完成后会在“addons/【扩展类型】/【扩展包名】”中生成一个package目录，并将所有的依赖文件拷贝到该目录，并创建一个压缩包。  
  

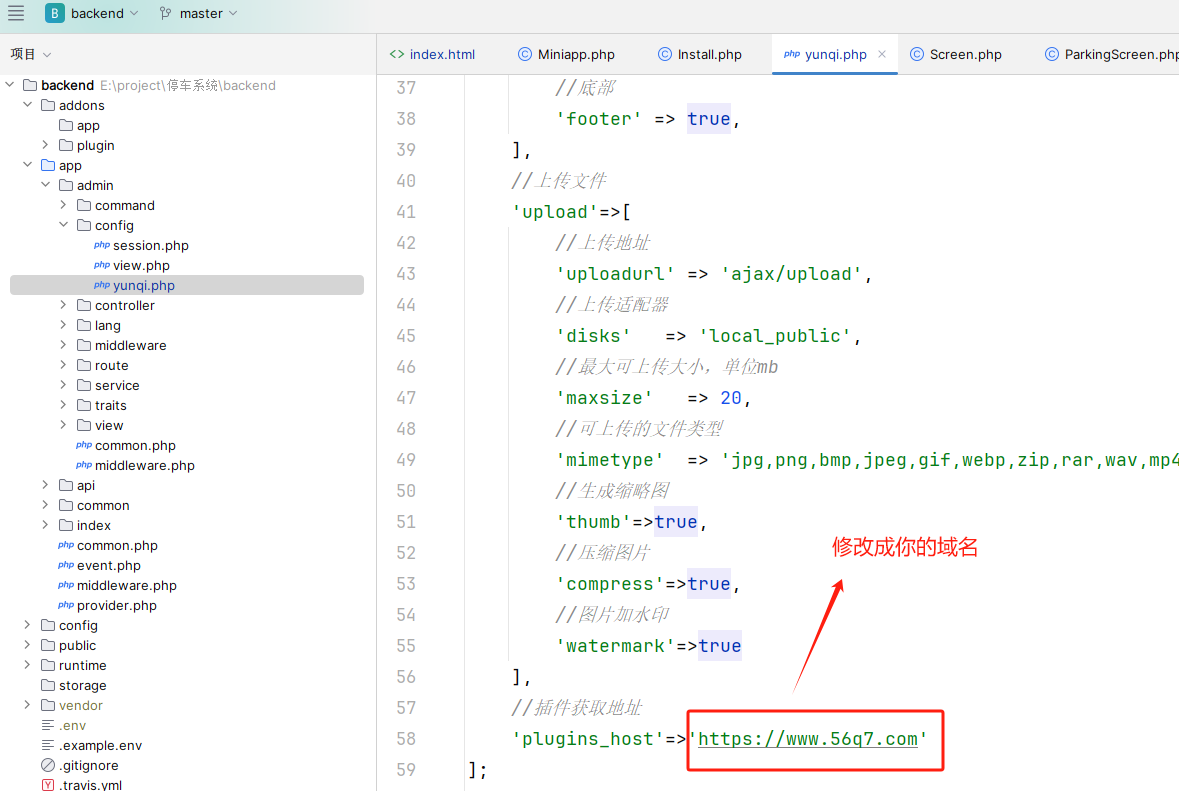

#### 11、重新编辑扩展，或扩展升级。

打包好后的扩展将不能再编辑，如果要重新编辑扩展，**千万别点卸载，千万别点卸载，千万别点卸载**，需要将目录中除了Install.php外都删除掉，刷新列表。  


#### 12、分享扩展给其他开发者。

1、在扩展列表中安装“开放扩展支持”  


2、然后填写开放扩展的相关配置  


3、开发者配置好你扩展的下载域名，这样他就可以从你的服务器下载扩展了。  


# 插件

# 开放扩展支持

1、功能介绍

### 2、扩展依赖

### 3、扩展配置

4、更新日志

# 省份城市地区数据

1、功能介绍

### 2、扩展依赖

### 3、扩展配置

4、更新日志

# 腾讯地图位置选择

1、功能介绍

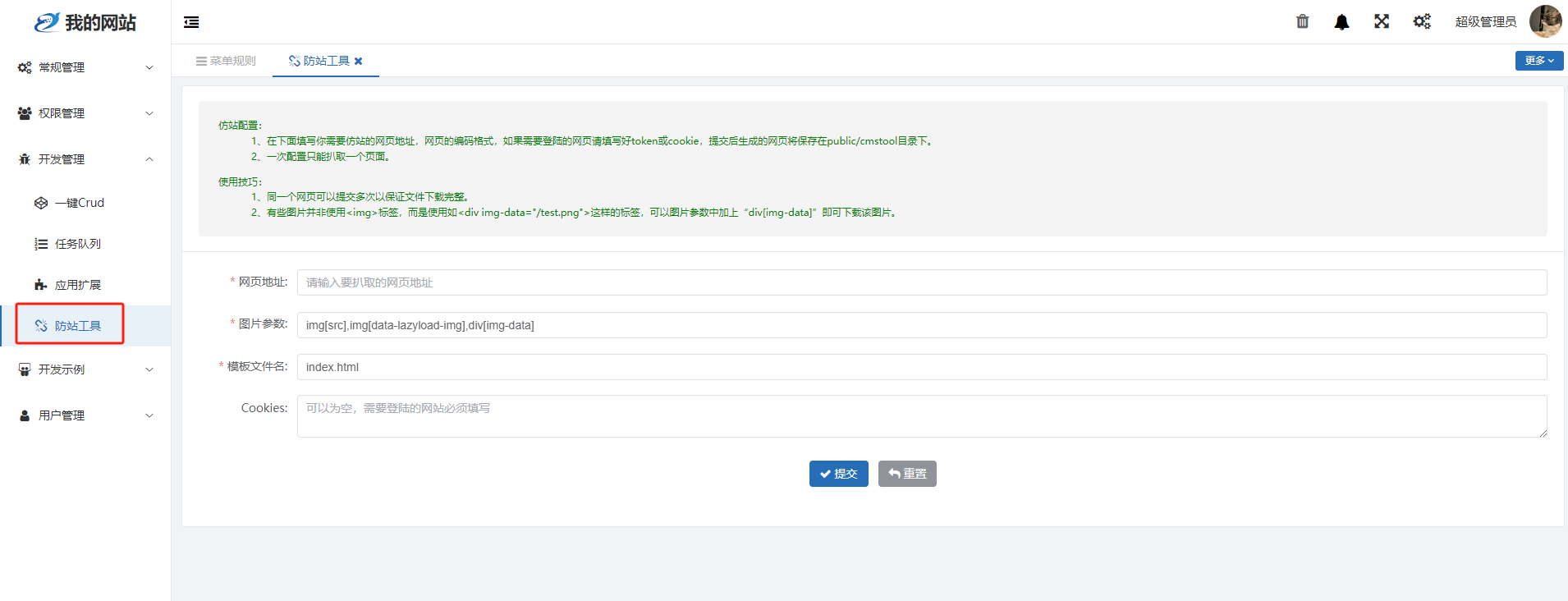
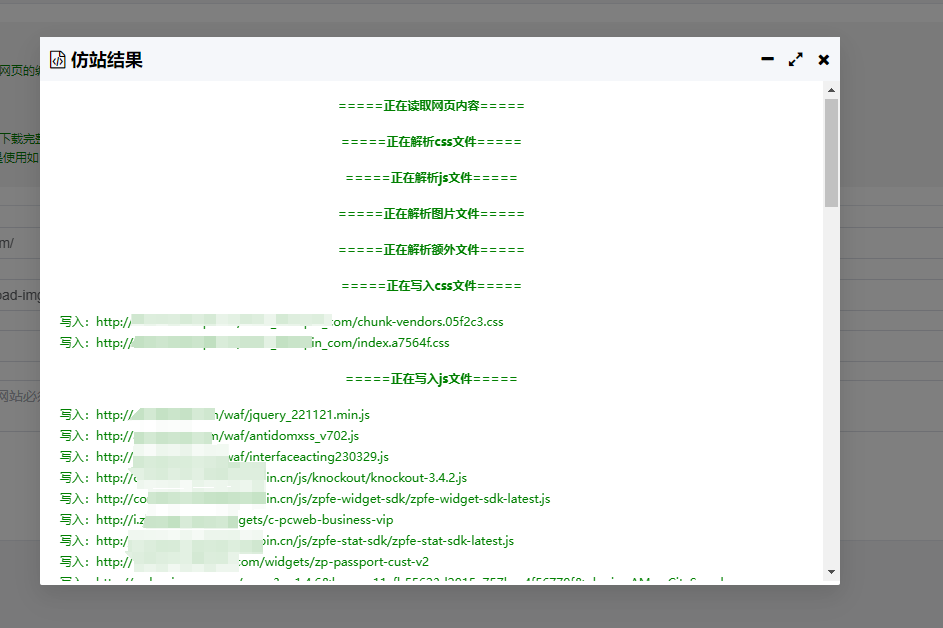
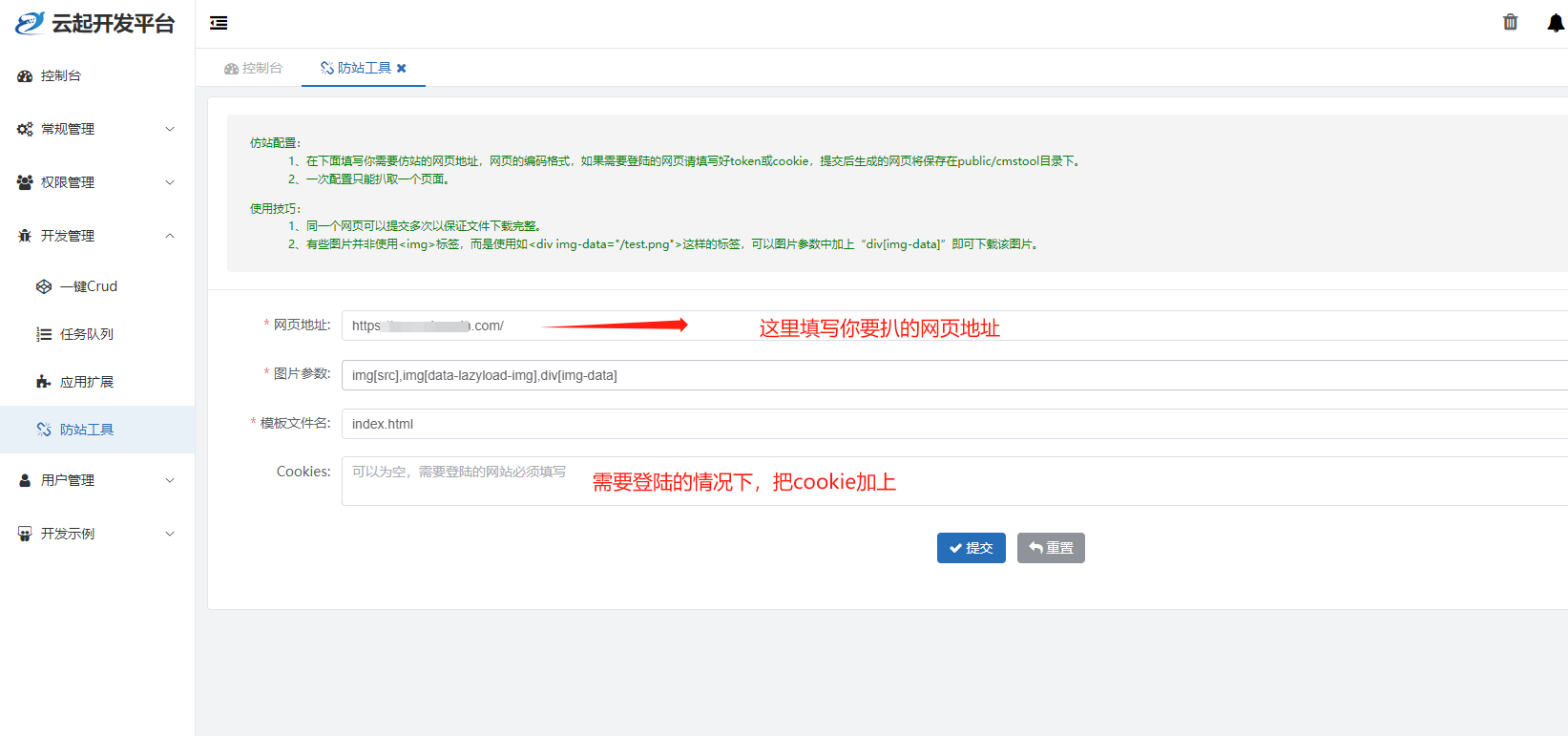
### 2、扩展依赖

### 3、扩展配置

4、更新日志

# 仿站扒站工具

使用方法

1. 在应用扩展中下载并安装插件“仿站、扒站工具”
2. 刷新页面，可以看到在开发管理中多了一个“仿站工具”
3. 开扒

# 阿里云短信

发送短信示例

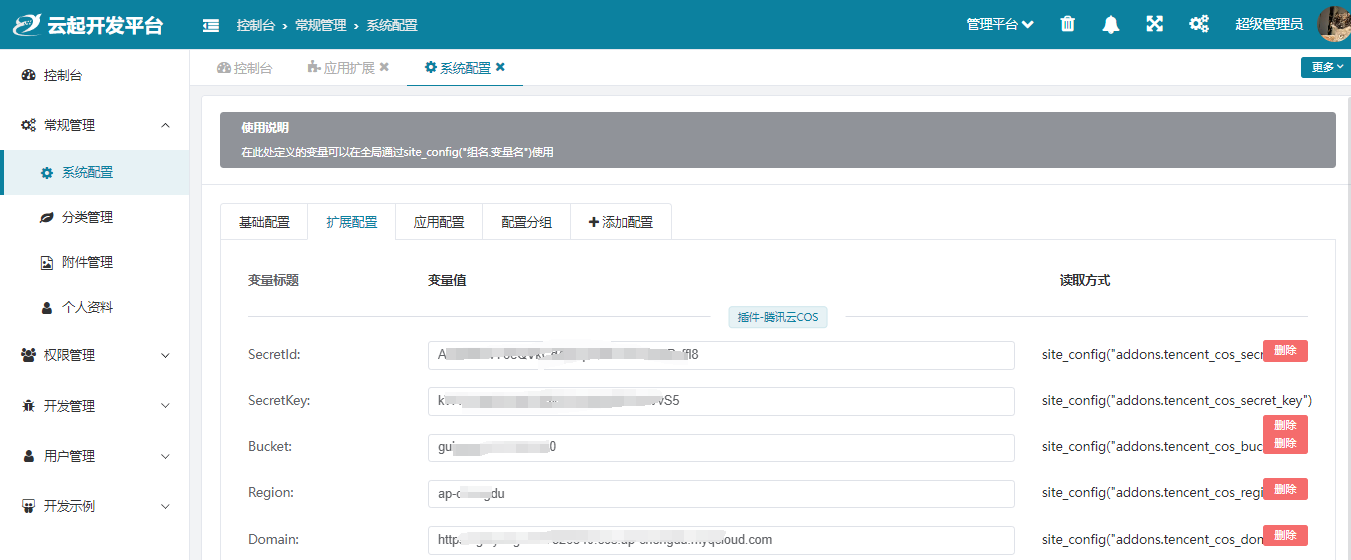
#[Post('sendmsg')]  
 public function sendmsg()  
 {  
 $mobile = $this->request->post('mobile');  
 $template = 'SMS\_461170083';  
 if (!$mobile) {  
 $this->error('手机号不能为空');  
 }  
 $code=rand(1000,9999);  
 Cache::set('sms\_code\_'.$mobile,$code,60\*10);  
 $param = ['code'=>$code];  
 if(!class\_exists('\\app\\common\\library\\Alisms')){  
 $this->error("请安装阿里云短信插件");  
 }  
 $alisms = new \app\common\library\Alisms();  
 $ret = $alisms->mobile($mobile)  
 ->template($template)  
 ->param($param)  
 ->send();  
 if ($ret) {  
 $this->success("发送成功");  
 } else {  
 $this->error("发送失败！失败原因：" . $alisms->getError());  
 }  
 }

# 腾讯云COS

1、功能介绍

支持使用腾讯云COS管理文件上传与删除文件

### 2、安装配置

1. 安装依赖包：composer require qcloud/cos-sdk-v5。
2. 在云起开发平台框架内安装插件。
3. 控制台=》常规管理=》系统配置=》扩展配置中配置插件。

### 3、后端设置为默认上传方式

修改文件：app/admin/config/yunqi.php

//上传文件  
'upload'=>[  
 //上传地址  
 'uploadurl' => 'ajax/upload',  
 //上传适配器  
 'disks' => 'tencent\_cos',  
 //最大可上传大小，单位mb  
 'maxsize' => 20,  
 //可上传的文件类型  
 'mimetype' => 'jpg,png,bmp,jpeg,gif,webp,zip,rar,wav,mp4,mp3,webm,doc,docx,xls,xlsx,pdf',  
 //生成缩略图  
 'thumb'=>true,  
 //压缩图片  
 'compress'=>true,  
 //图片加水印  
 'watermark'=>true  
],

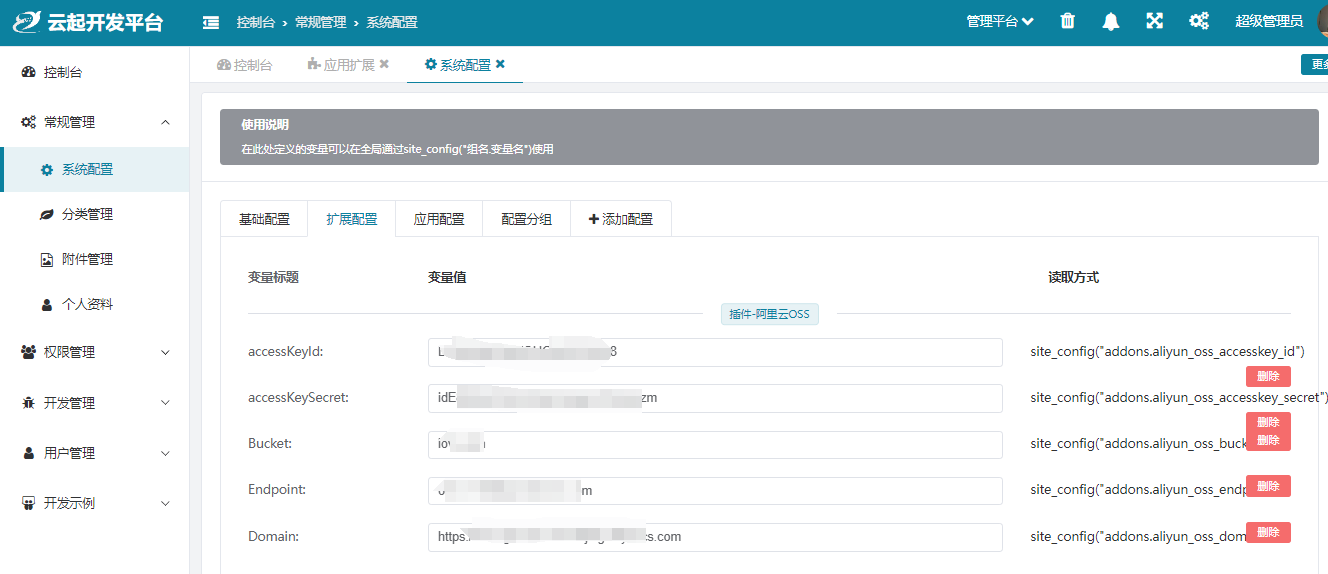
\*\*\*\*

# 阿里云OSS

1、功能介绍

支持使用阿里云OSS管理文件上传与删除文件

### 2、安装配置

1. 安装依赖包：composer require aliyuncs/oss-sdk-php。
2. 在云起开发平台框架内安装插件。
3. 控制台=》常规管理=》系统配置=》扩展配置中配置插件。

### 3、后端设置为默认上传方式

修改文件：app/admin/config/yunqi.php

//上传文件  
'upload'=>[  
 //上传地址  
 'uploadurl' => 'ajax/upload',  
 //上传适配器  
 'disks' => 'aliyun\_oss',  
 //最大可上传大小，单位mb  
 'maxsize' => 20,  
 //可上传的文件类型  
 'mimetype' => 'jpg,png,bmp,jpeg,gif,webp,zip,rar,wav,mp4,mp3,webm,doc,docx,xls,xlsx,pdf',  
 //生成缩略图  
 'thumb'=>true,  
 //压缩图片  
 'compress'=>true,  
 //图片加水印  
 'watermark'=>true  
],

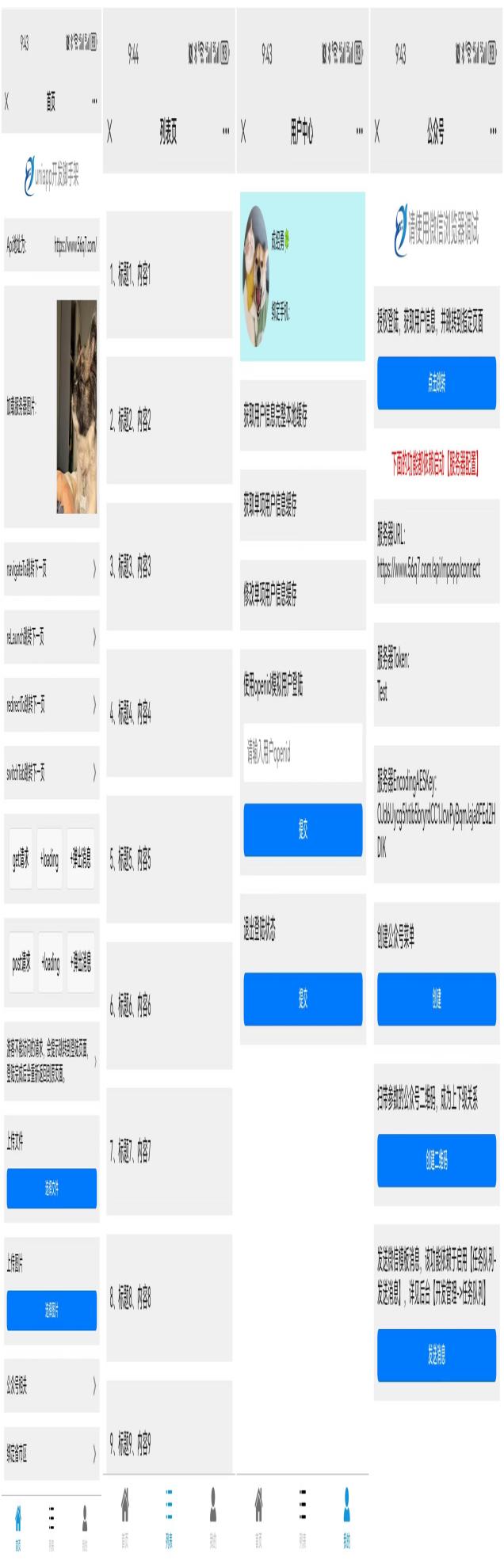
\*\*\*\*

# 应用

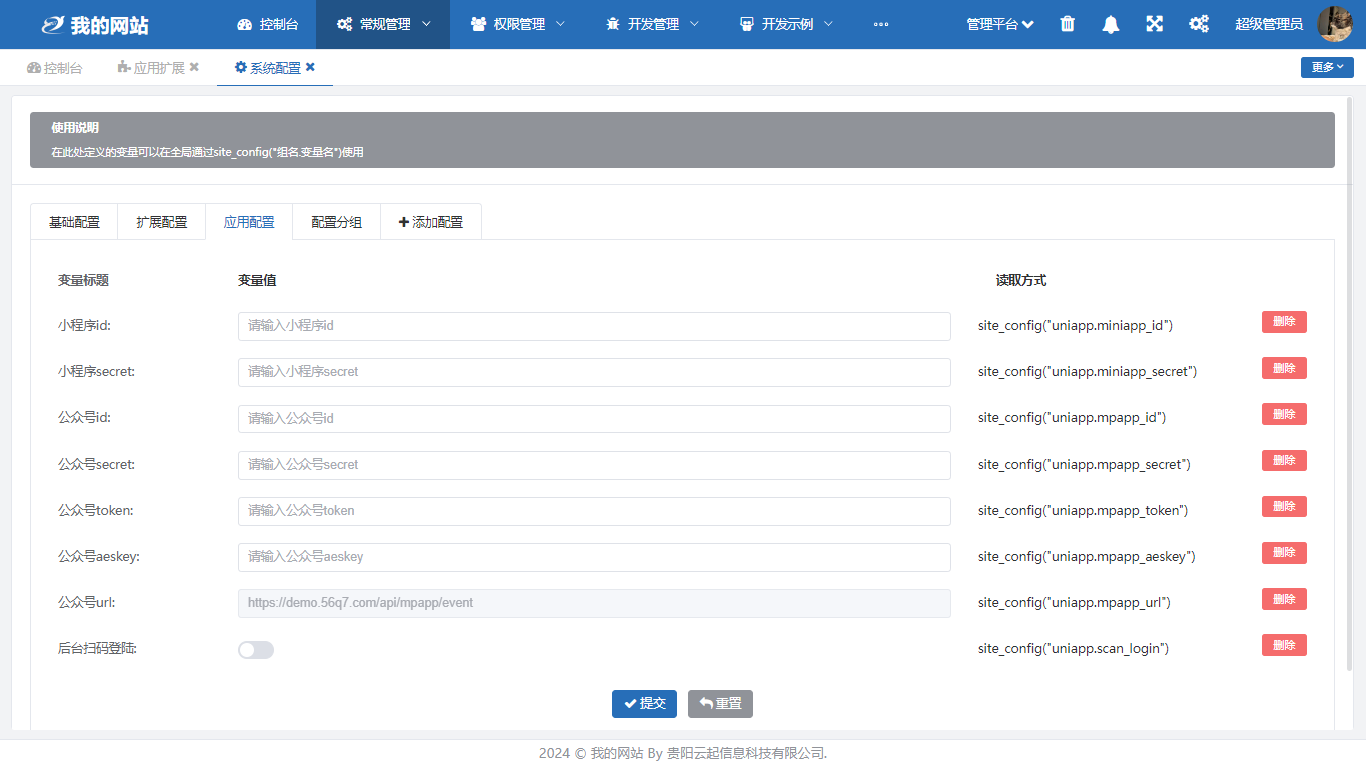
# uniapp开发脚手架

1、功能介绍

**h5端体验地址**：<https://demo.56q7.com/h5>



### 2、安装配置

1. 先安装依赖包，composer require zoujingli/wechat-developer，该包封装了微信公众号的相关操作，[参考文档](https://www.kancloud.cn/zoujingli/wechat-developer/518672)
2. 在框架内下载安装插件
3. 配置应用
4. 应用根目录有一个uniapp的文件夹，导入到hbuilderX
5. 修改uniapp/App.vue

globalData:{  
 //替换成你自己的域名  
 baseUrl:'https://demo.56q7.com/'  
},

### 3、开发相关

1. 小程序与公众号账号互通，需要绑定开放平台，否则相同微信会生成小程序与公众号两个账号。
2. 发送微信模板消息，需要开启消息队列服务。

4、更新日志

# 商品每日预定与配送系统

一、系统介绍

该系统适用于生产厂商，或经销商，针对一些单一，或少量单品，对稳定的签约客户，长期预约与配送。比如，桶装水、美发店的消毒毛巾、消毒碗筷、火锅店的新鲜菜蔬、早餐鲜奶等。  
该系统有如下特点：

1. 签约客户可以是家庭、也可以是企业、单一商品对每个客户可以设置不同的签约价格。
2. 客户下单有多种方式，支持自主下单，定时自动下单，司机代下单。
3. 客户下单有多种结算方式，包括在线付款，货到付款，每月付款。
4. 配送单元网格化管理，提升配送效率，降低配送成本。
5. 针对销售拓客的需求支持销售员模块，客户与销售人员绑定，支持多种提成方案。
6. 支持多贸易商入驻，送水、送气、送牛奶互不影响，提高系统的利用率。
7. 贸易商通过小程序管理，配送，销售，订单，客户手机尽掌握。

### 二、系统安装

1. 在“应用扩展”中下载源码。
2. 安装依赖

支付扩展 - [参考文档](https://pay.yansongda.cn/docs/v3/quick-start/install.html)

composer require yansongda/pay:~3.7.0 -vvv   
composer require guzzlehttp/guzzle:^7.0  
composer require hyperf/pimple:~2.2.0

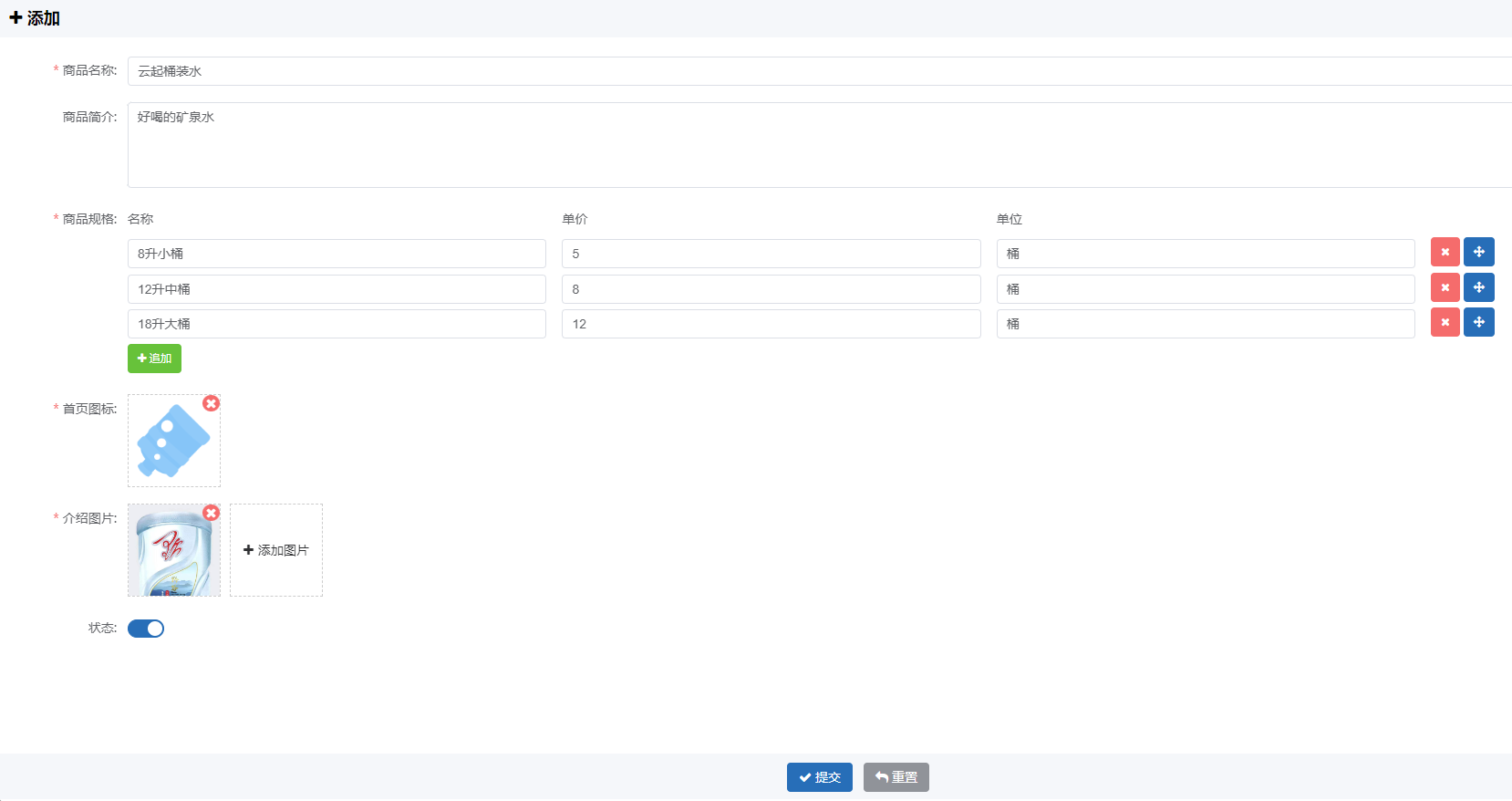
微信小程序扩展 - [参考文档](https://www.kancloud.cn/zoujingli/wechat-developer/519201)

composer require zoujingli/wechat-developer

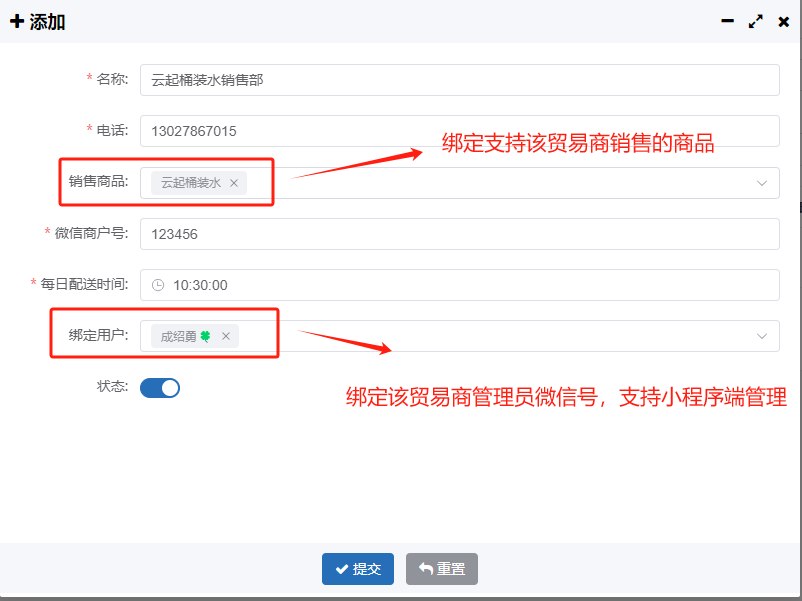
1. 点击安装扩展

### 三、功能设置

##### 1、添加销售商品

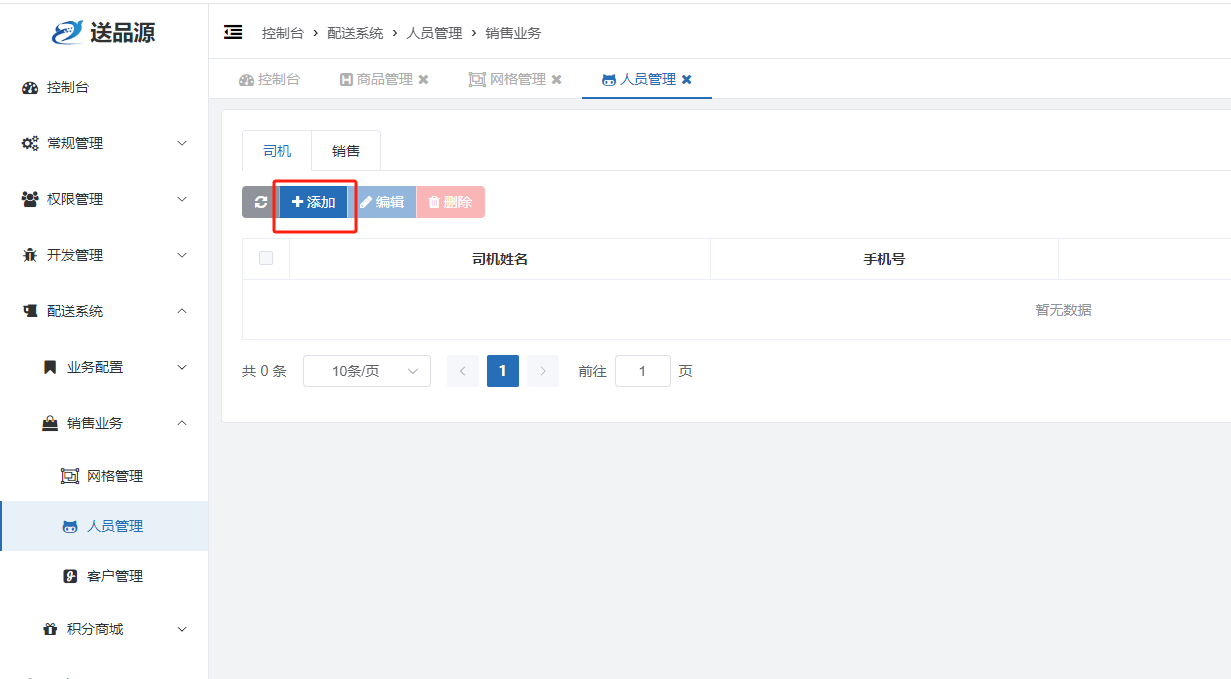
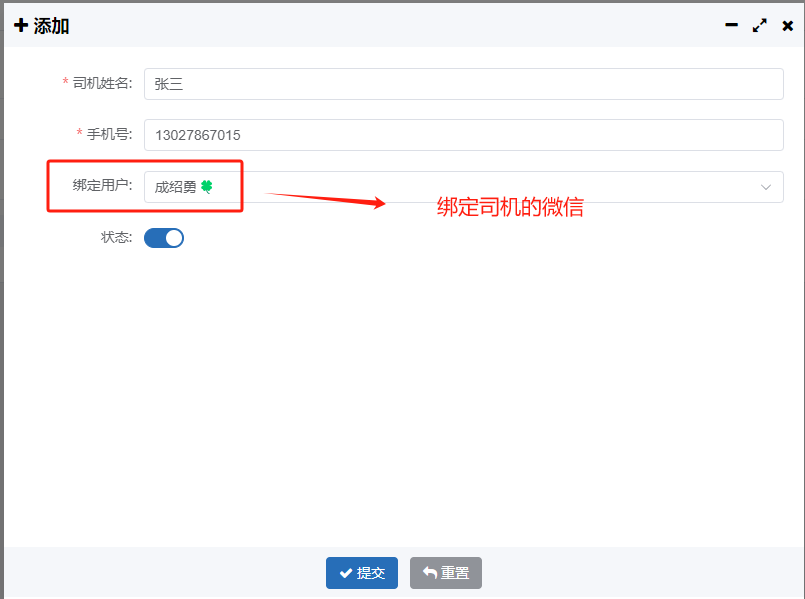
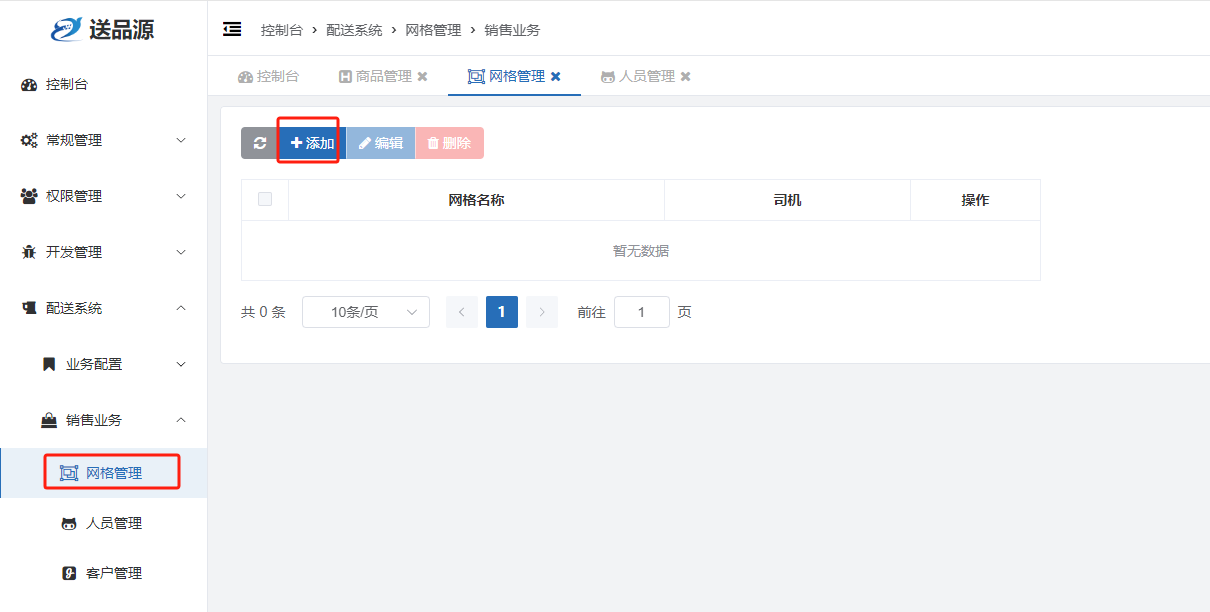
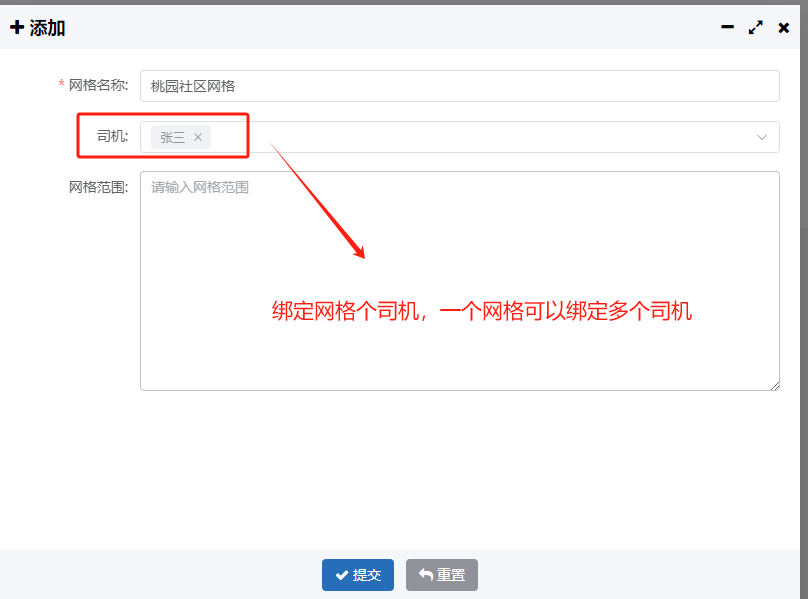
##### 2、添加贸易商

将添加好的贸易商设置为“自营”



##### 3、设置配送网格与配送人员

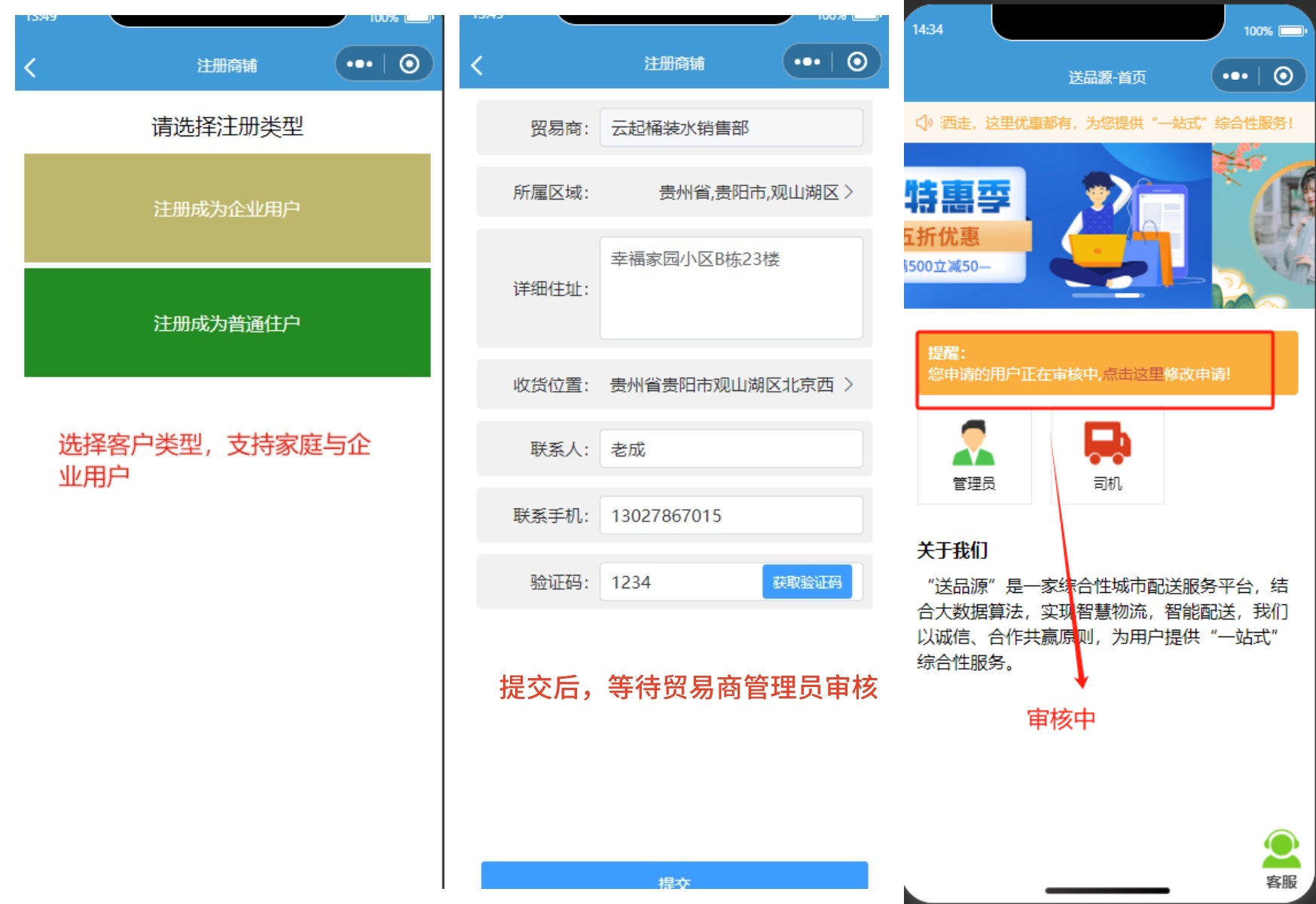
1. 添加配送司机  
     
   
2. 添加网格  
     
   

### 四、客户入驻与管理

##### 1、客户申请入口



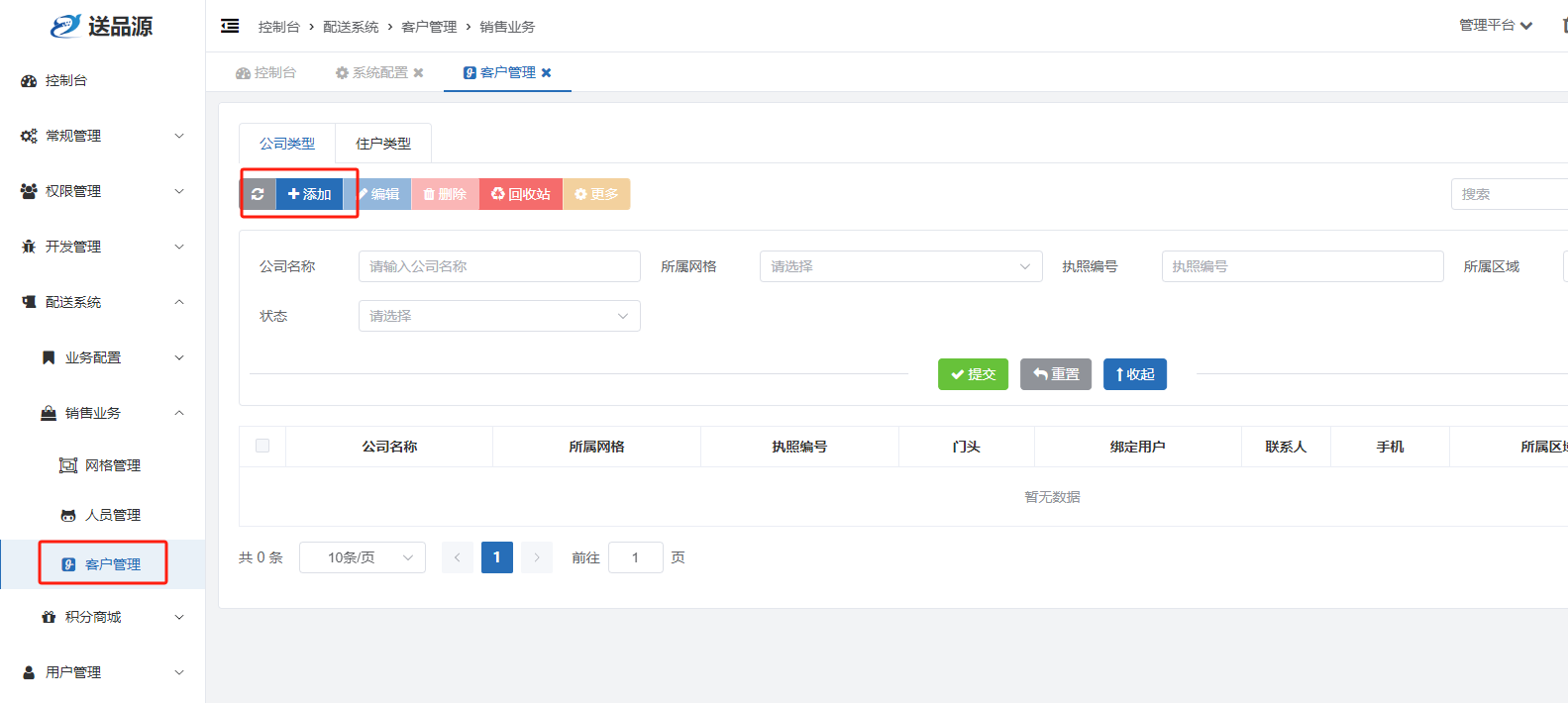
##### 2、客户填写申请资料



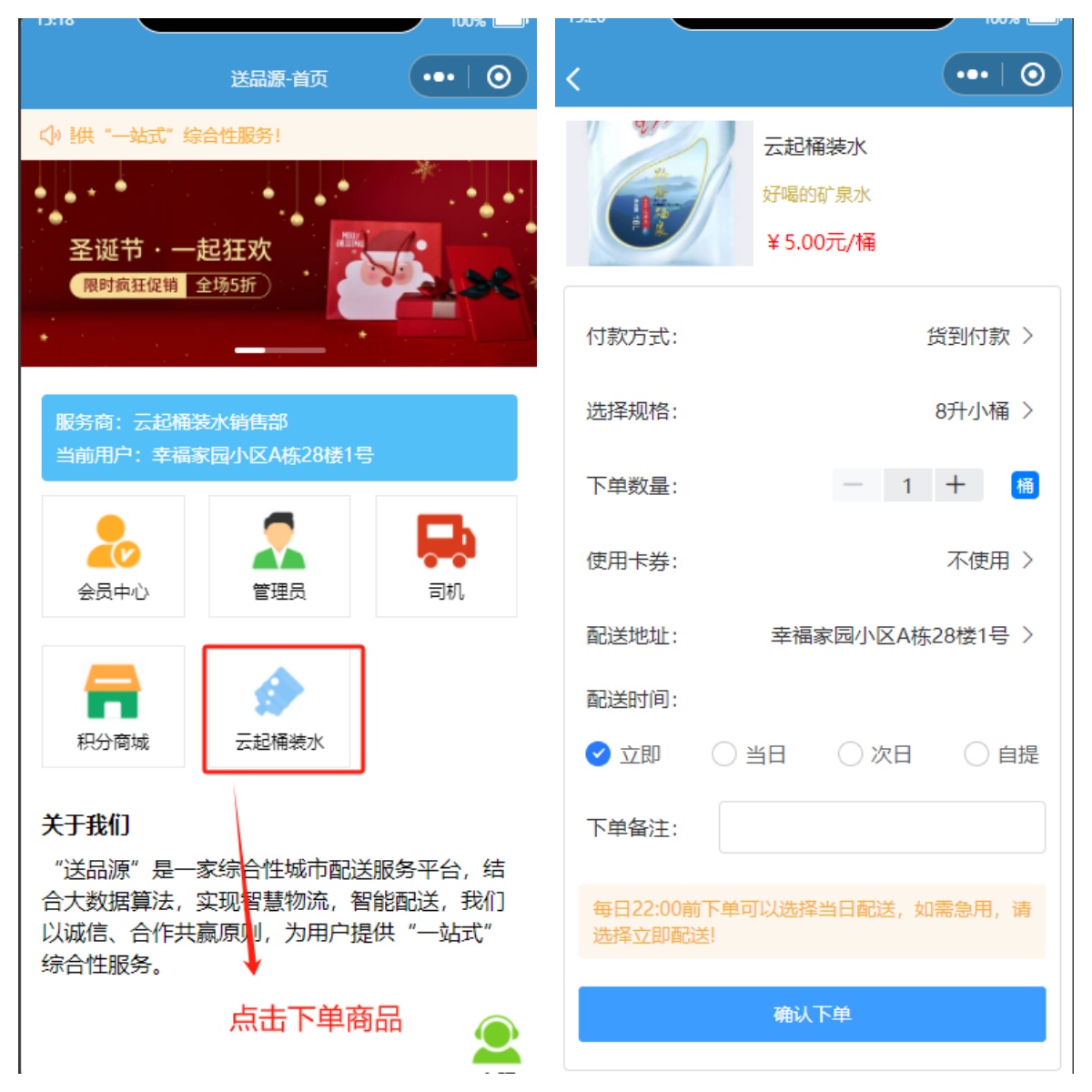
##### 3、客户审核



##### 4、PC端添加客户，添加后无需审核



### 五、客户下单

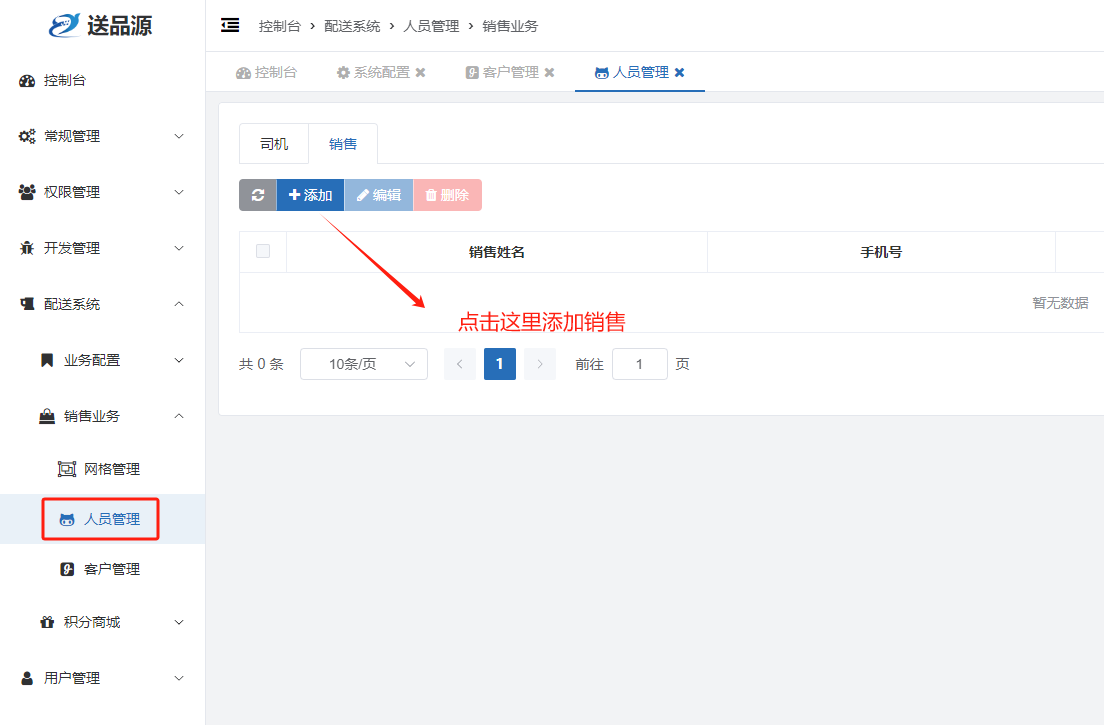
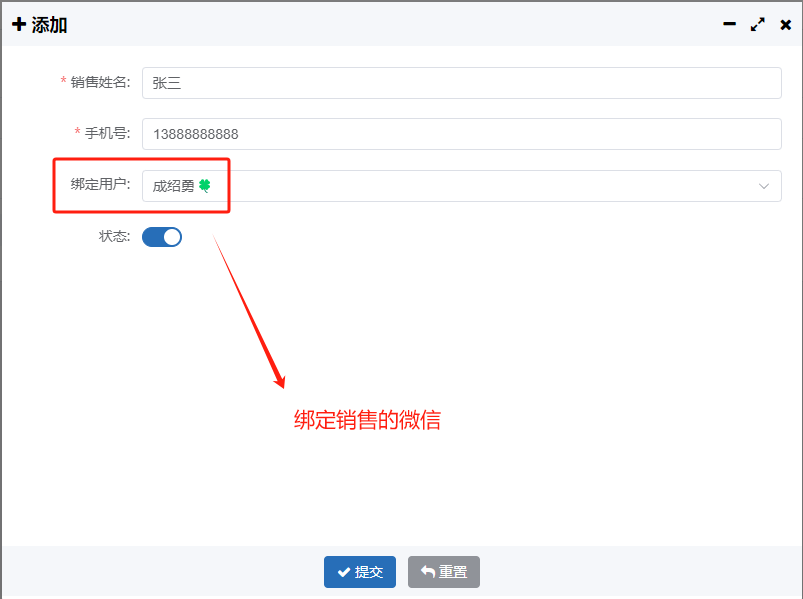


### 六、司机配送



### 七、销售与拓客

##### 1、添加销售员

##### 2、销售拓客



##### 3、销售注册客户

销售可以通过推广二维码来注册新客户。

# 多语言

本系统没有使用到Thinkphp自带的多语言处理方式，之所以这样决定，是为了解决下面两个方面的问题：

1、使用本系统不希望代码中涉及文本的地方都使用英文，如lang('English is balabala')这样的语法，特别是在模板中，英文不好看起来脑壳大。

2、使用本系统希望使用一个语言文件就能管理一个应用，不必要创建许多语言文件，不方便管理。

## 一、语言包定义方式

1、创建语言包文件。

创建app/应用名/lang/语言名.php  
，使用“中文”作为多语言的“键”，要替换的语言作为多语言的“值”，下面我们创建一个en-us.php为例

<?php  
use app\admin\controller\Ajax;  
use app\admin\controller\Index;  
  
return [  
 //全局语言包  
 'default'=>[  
 '添加'=>'Add',  
 '编辑'=>'Edit',  
 '删除'=>'Delete',  
 '更多'=>'More',  
 '正常'=>'Normal',  
 '隐藏'=>'Hidden',  
 '是'=>'Yes',  
 '否'=>'No',  
 ],  
 //控制器语言包  
 'controller'=>[  
 Index::class=>[  
 '控制台'=>'Dashboard',  
 '常规管理'=>'General',  
 '系统配置'=>'System',  
 '分类管理'=>'Category',  
 '附件管理'=>'Attachment',  
 '个人资料'=>'Profile',  
 '菜单规则'=>'Menu',  
 '管理员管理'=>'Admin',  
 '角色组'=>'Role',  
 '管理员日志'=>'Admin Log',  
 '一键Crud'=>'Crud',  
 '任务队列'=>'Task Queue',  
 '权限管理'=>'Permission',  
 '开发管理'=>'Development',  
 ],  
 Ajax::class=>[  
  
 ]  
 //...更多控制器  
 ]  
];

## 二、语言包的使用

1、php脚本中实现语言包替换。

\_\_('控制台')

2、html模板中实现语言包替换。

{:\_\_('控制台')}

3、页面js中来实现语言包替换。

\_\_('控制台')

4、components模板中来实现语言包替换。

${\_\_('控制台')}

5、php语言包带参数。

\_\_('一段文字:a=%a,b=%b',['a'=>1,'b'=>2])

5、js语言包带参数。

\_\_('一段文字:a=%a,b=%b',{a:1,b:2})

# 常见错误【必坑必读】

1、非html标签必须成对出现，不能省略为<标签名/>

正确：<el-input v-model="data"></el-input>  
错误：<el-input v-model="data"/>

2、自定义组件严格按照VUE3驼峰命名规范。

export default{  
 components:{'YunqiTable':table},  
 data:{}  
 ...  
}  
正确：<yunqi-table></yunqi-table>  
错误：<YunqiTable></YunqiTable>

3、即便在debug模式下，如果修改组件js务必清空浏览器缓存，否则不会生效

4、内置Form组件中，searchList的key为数字时会自动排序，所以如果表单下拉菜单不希望被打乱顺序的情况，推荐使用slot，然后用php的loop标签来循环

<template v-slot:pid="{rows}">  
 <el-form-item label="父级:" prop="pid">  
 <el-select placeholder="请选择父级" v-model="rows.pid" :clearable="true" style="width: 100%">  
 {foreach $ruledata as $key=>$value}  
 <el-option key="{$key}" label="{$value}" value="{$key}"></el-option>  
 {/foreach}  
 </el-select>  
 </el-form-item>  
</template>

5、在使用element-plus表单标签el-select，el-radio，el-checkbox中，要特别注意选中值是数字还是字符串，给默认值类型必须保持一致

//下面这种情况实际上是字符串  
<el-checkbox-group v-model="checkList">  
 <el-checkbox label="1" />  
 <el-checkbox label="2" />  
 <el-checkbox label="3" />  
</el-checkbox-group>  
//默认选中1和2  
checkList:['1','2']  
  
//下面这种情况实际上是数字  
<el-checkbox-group v-model="checkList">  
 <el-checkbox :label="1" />  
 <el-checkbox :label="2" />  
 <el-checkbox :label="3" />  
</el-checkbox-group>  
//默认选中1和2  
checkList:[1,2]

6、view开发中，js是一次性生成然后缓存在后端，所以脚本切勿使用php动态编译的语法。

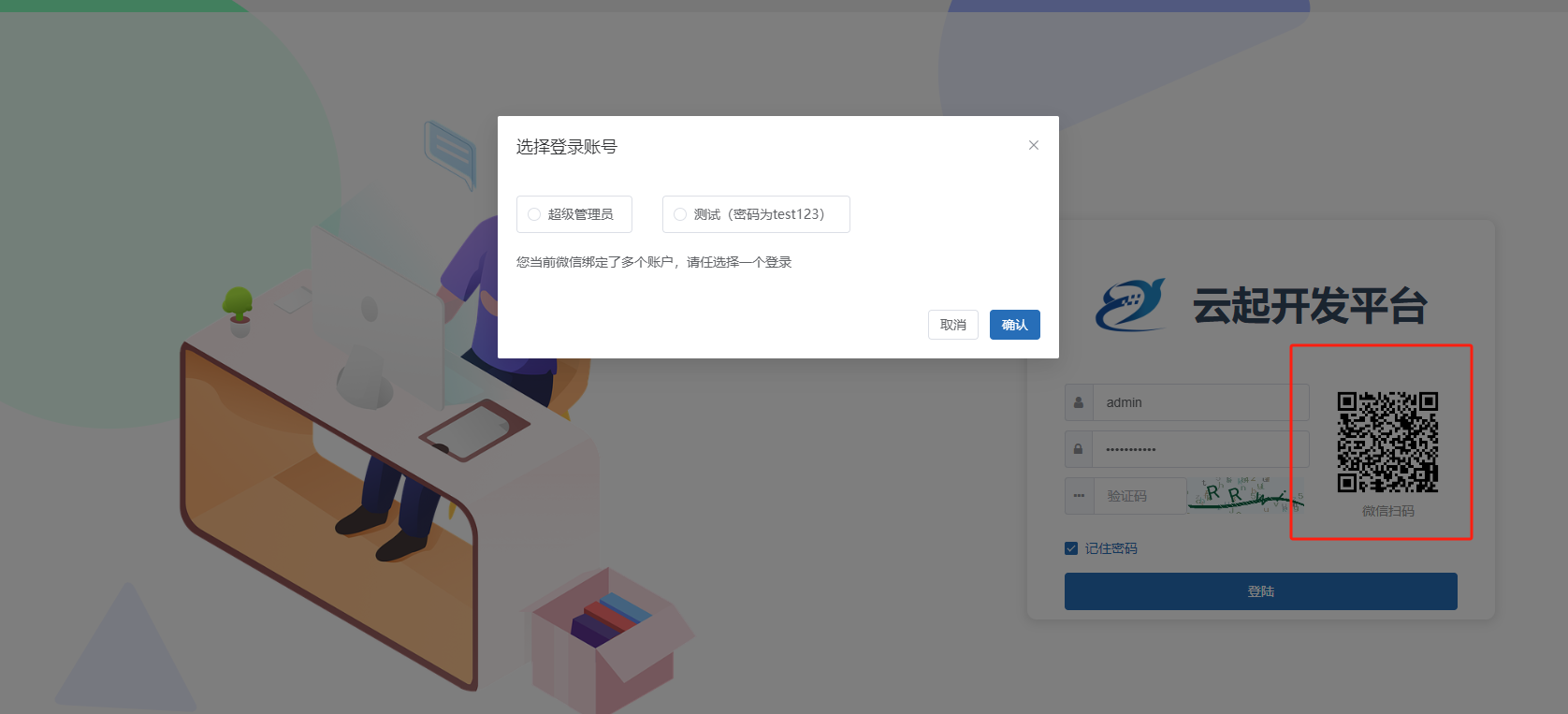
<script>  
export default{  
 data:{  
 t1:Yunqi.data.t1,//这种写法是正确的。  
 t1:{$t1}//这种写法是错误的！  
 },  
 methods: {  
  
 }  
}  
</script>

# 1.3版本更新

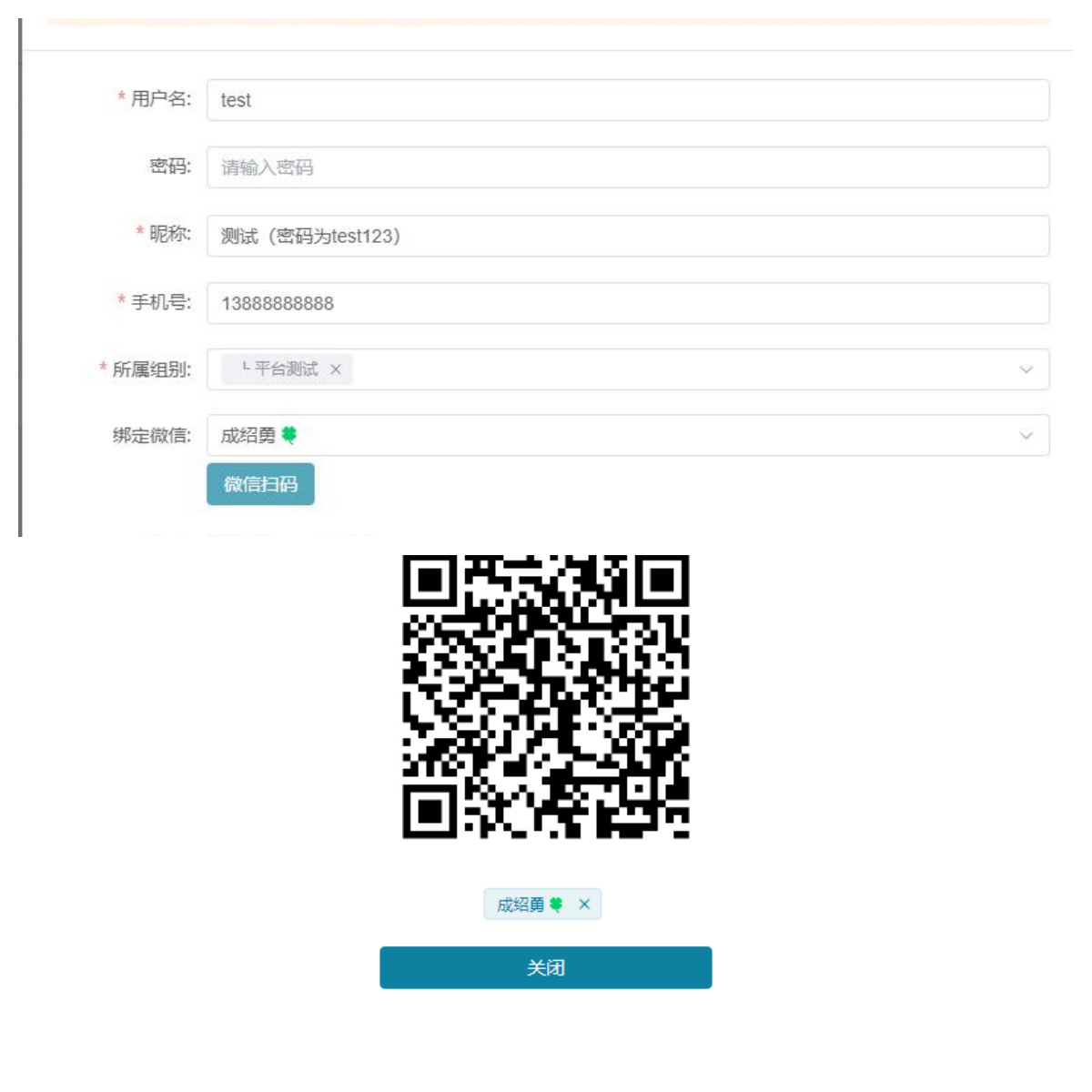
一、管理端UI优化。

1. 处理已知的ui相关的bug若干。
2. 优化自定义ui存储方式，将之前的cookies存储改成数据库存储，解决cookies过期导致界面的错乱问题。
3. 处理修改主题与暗黑模式中，文字、背景、即时反应，框架内外层协调统一，切换更丝滑，效果更契合。
4. 增加亮色logo，让暗黑模式，深色主题时，logo更美观。

二、扩展开发优化。

1. 增加扩展的编辑功能，便于扩展的修改与升级。
2. 优化生成扩展配置，app应用类型扩展，安装后直接生成应用配置，其他类型扩展继续原来的方案，生成扩展配置。
3. 增加扩展开发的相关文档。
4. 增加2个免费插件扩展，腾讯云上传文件、百度云上传文件。
5. 增加1个免费应用扩展，商品配送系统，便于开发者掌握与学习云起开发平台。
6. uniapp脚手架插件支持绑定管理员使用微信登录，一个微信可以绑定多个管理员。  
   

三、系统优化。

1. 增加平台类型菜单，方便针对多角色的平台开发。  
   
2. 增加绑定第三方账号的组件components/Third.js的组件  
   
3. 优化表格组件，让表格的宽度自适应，当表格内容过多时，宽度会自动适应变宽，以前会压缩换行，让界面很难看。