thinkAdmin 开发手册

更新时间：2018-10-31 09:15:16

目录

一、Thinkadmin 原理 1
1.thinkadmin 后台 js 插件的加载机制 1
2.后台框架布局 2
3.后台路由重定向问题 3
4.权限验证 4
二、thinkadmin 封装函数 4
1.控制器 4
2．Service 层(关于 http 封装的函数): 5
三、php 插件 10
四、前端插件 11
1.$.form 通用表单数据插件 13
2.$.msg 通用消息弹出提示插件 14
3.$.vali 通用表单验证插件 14
五 html 标签 15
1.data-load 标准异常网络请求（其中菜单不会发生改变） 15
2.data-open 内容区打开新页（其中菜单会发生改变） 16
3.data-modal Modal 弹出层 16
4.data-reload 强制刷新内容区 16
5.data-update 更新数据 17
6.data-href 打开新的网页 17
7.data-file 文件上传插件 以 HTML 属性的方式实现文件上传 17
8.data-iframe Iframe 打开网页 18
9.data-icon 打开图标选择器 18
10.data-tips-image 点击图片展示图片 19
11.data-tips-text 鼠标悬停文字提示 19
12.data-phone-view 以手机模式显示 19

Ps：thinkadmin 框架截止现在一共三个版本，v1 与 v2 是基于 thinkphp5.0 基础上开发的 V3 版本是基于 thinkphp5.1 基础上开发的，v3 版本用到了 thinkphp5.1 的新特性，例如：中间件。Thinkadmin 不是响应式后台框架。

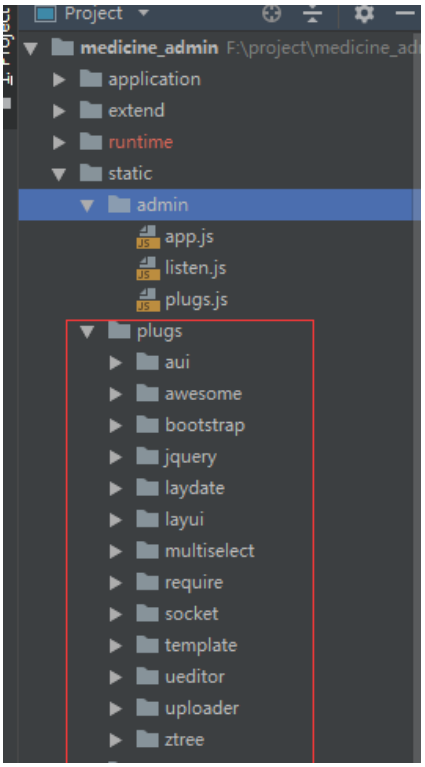
本片文档是基于 thinkadmin——v2 后台源码的基础上写的，文档也含有 v3 的特性。

更新框架使用 composer 命令。

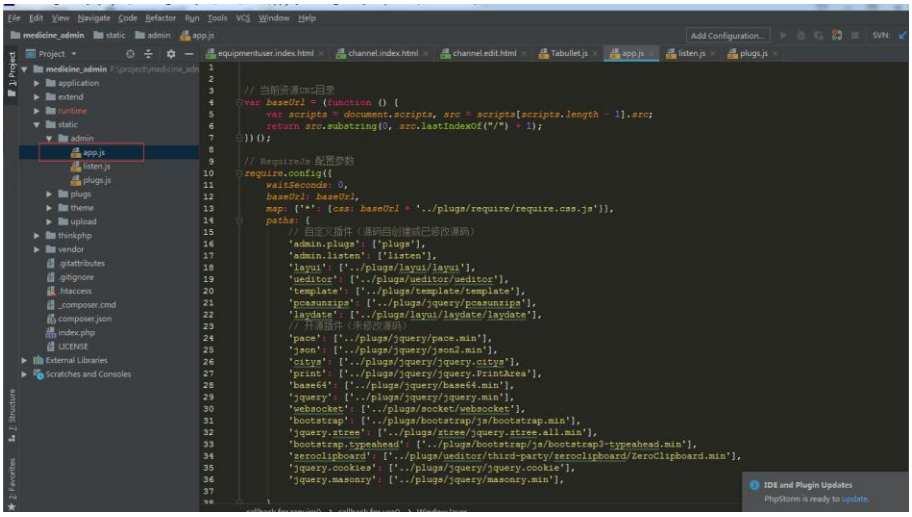
一、Thinkadmin 原理

1.thinkadmin 后台 js 插件的加载机制

Require.js 的简单介绍 :(https://www.jb51.net/article/117676.htm)。后台是利用 require.js 加载插件，加载的插件如下：(已经在 app.js 中配置过的文件)。define 来定义模块， require 来加载模块。(require.js 官网地址：http://www.requirejs.cn/) 关于此插件的加载原理



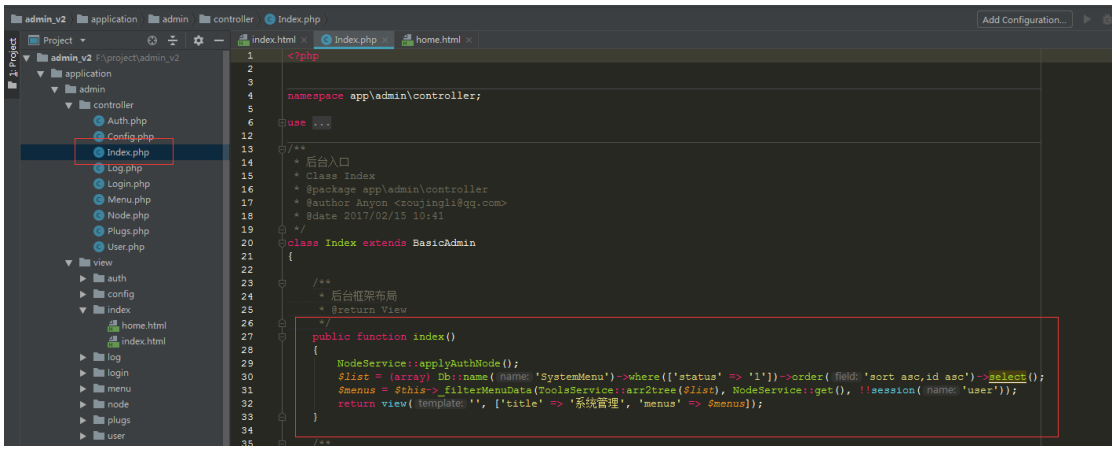
Require.js 加载配置项：(static\admin\app.js)



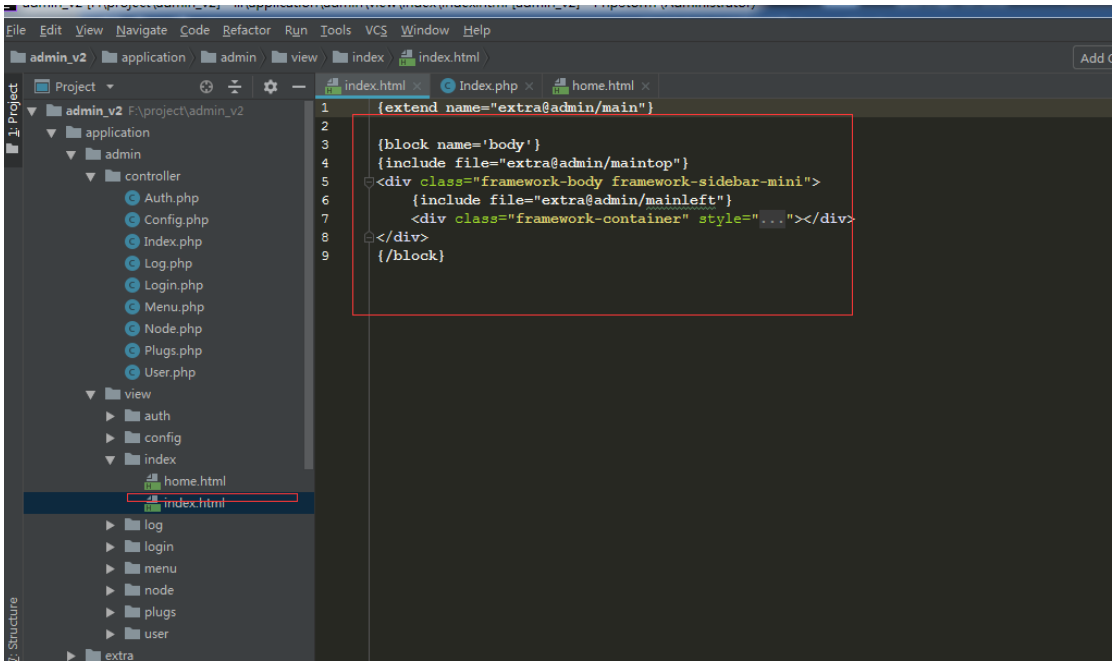
2.后台框架布局

后台框架布局分三部分：内容部（3）分利用 js 替换；（实现方式在 plug.js 脚本中）。登录之后 utl：testv2.kangkn.cn/admin.html#/admin/index/home，此时访问的控制器始终是：admin/index/index。#后面的全是参数。（接收 url

提交的参数:数组中有一个是 “/admin/index/home” => “”）



前端代码：



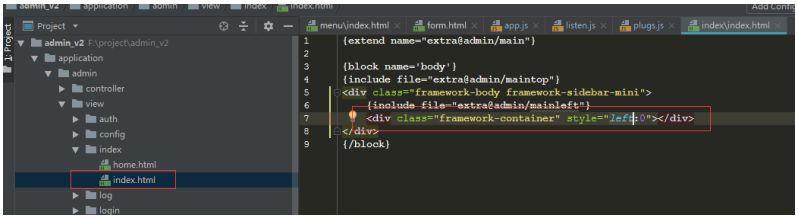
3.后台路由重定向问题

输入域名 testv2.kangkn.cn，默认访问 “testv2.kangkn.cn/index/index/index/”

Php 脚本重新定向到：testv2.kangkn.cn/iadmin/login.html(访问的是 testv2.kangkn.cn/iadmin/login/index)，即登录页面。点击登录之后，自动加 “#”（此段 js 代码在 plugs.js 脚本中）。例如 testv2.kangkn.com 进入登录页面，点击登录之后，在脚本 plugs.js 中域名后加上 “#” 以及/admin/index/home（后台首页）

例如：testv2.kangkn.cn/admin.html#/admin/index/home

利用 ajax 访问 testv2.kangkn.com/admin/index/home 获取返回的 html 标签， 然后替换或添加内容



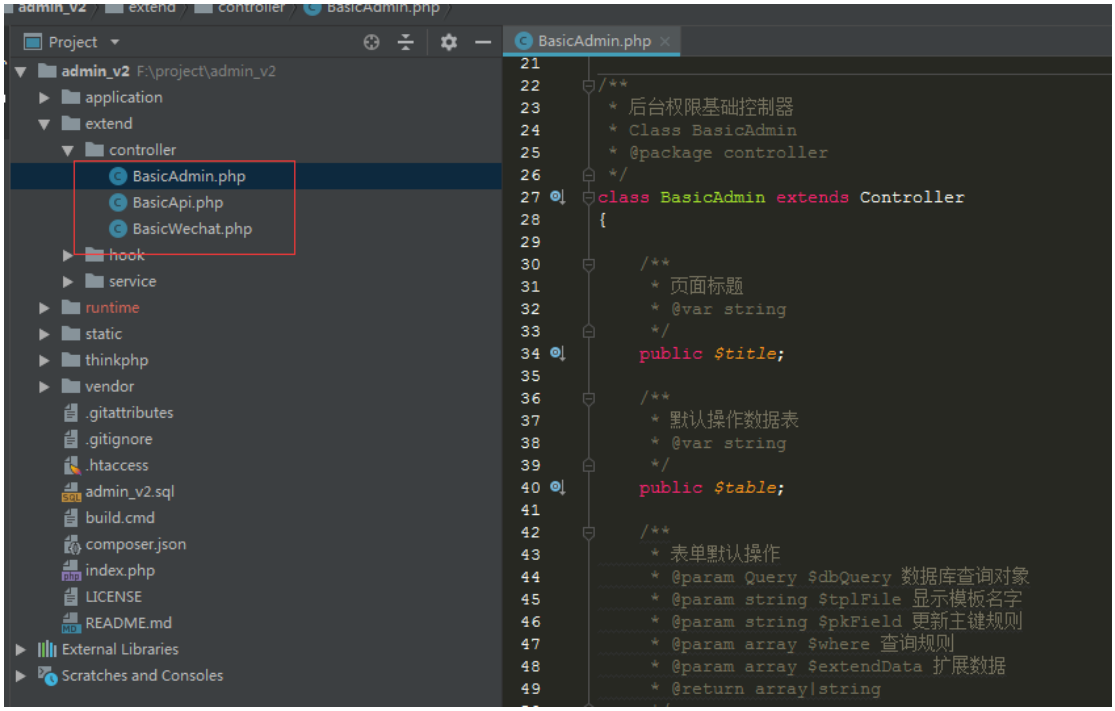
4.权限验证

点击登录之后，所有用户权限保存在 session 以及信息保存在 session 中，根据 session 来判定权限。每次更改 url 都会访问 admin/index/index 控制器，显示页面，并且权限判定。

```
     * @return View
     */
    public function index()
    {
        NodeService::applyAuthNode();
        $list = (array) Db::name( name: 'SystemMenu')->where(['status' => '1'])->order( field: 'sort asc,id asc')->select();
        $menus = $this->_filterMenuData(ToolsService::arr2tree($list), NodeService::get(), !!session( name: 'user'));
        return view( template: '', ['title' => '系统管理', 'menus' => $menus]);
    }
```

二、service 层封装函数

后台控制器基类中的函数常用函数



BasicAdmin 中两个常用封装函数；

1._form（）

注意 回调函数

```
    /**
 * 表单默认操作
 * @param Query $dbQuery 数据库查询对象
 * @param string $tplFile 显示模板名字
 * @param string $pkField 更新主键规则
 * @param array $where 查询规则
 * @param array $extendData 扩展数据
 * @return array|string
 */
protected function _form($dbQuery = null, $tplFile = '', $pkField
= '', $where = [], $extendData = [])
```

2.分页_list()

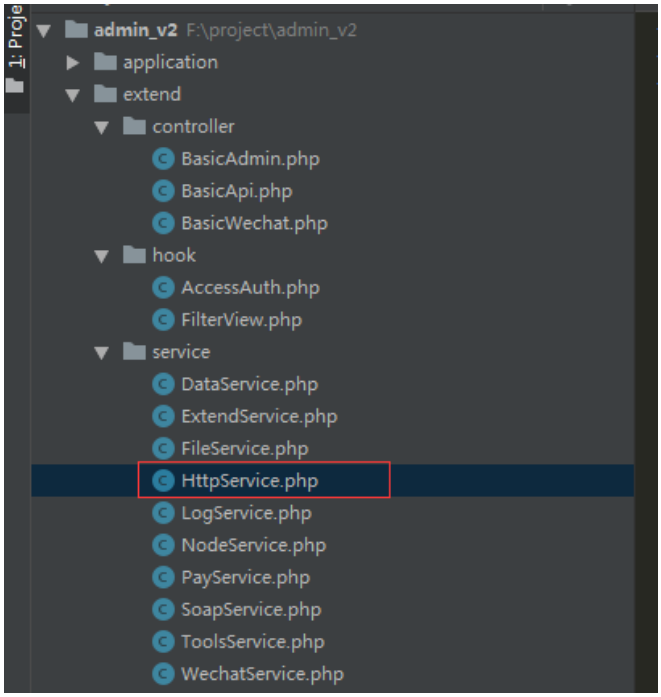
注意回函数

```
/**
 * 列表集成处理方法
 * @param Query $dbQuery 数据库查询对象
 * @param bool $isPage 是启用分页
 * @param bool $isDisplay 是否直接输出显示
 * @param bool $total 总记录数
 * @param array $result
 * @return array|string
 */
protected function _list($dbQuery = null, $isPage = true,
$isDisplay = true, $total = false, $result = [])
```

BasicApi 控制器：写接口的时候用到

BasicWechat 控制器：微信模块继承的控制器

关于 http 封装的函数:



3.模拟 http 请求（get）

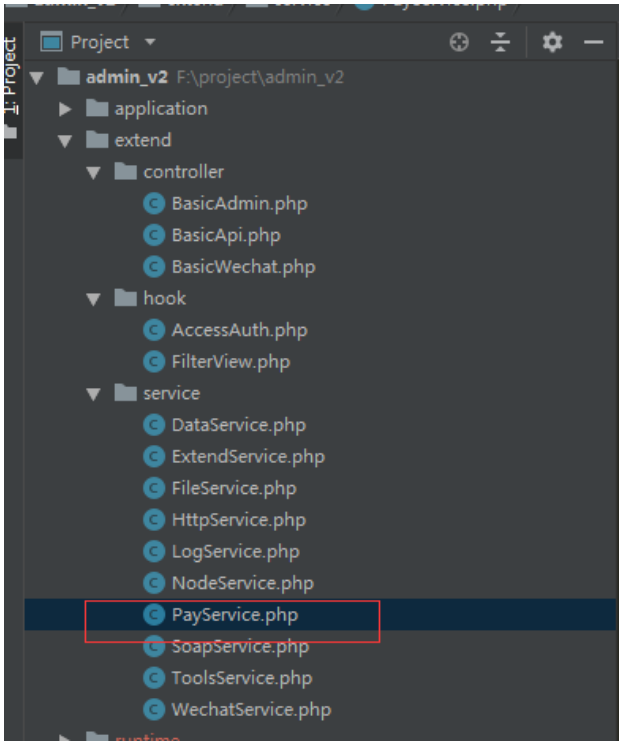
```
/**
 * HTTP GET 请求
 * @param string $url 请求的 URL 地址
 * @param array $data GET 参数
 * @param int $second 设置超时时间（默认 30 秒）
 * @param array $header 请求 Header 信息
 * @return bool|string
 */
public static function get($url, $data = [], $second = 30,
$header = [])
```

4.模拟 http 请求（post）

```
/**
 * POST 请求（支持文件上传）
 * @param string $url HTTP 请求 URL 地址
 * @param array|string $data POST 提交的数据
 * @param int $second 请求超时时间
```

```
 * @param array $header 请求 Header 信息
 * @return bool|string
 */
static public function post($url, $data = [], $second = 30,
$header = [])
```

微信支付接口的封装：



5. 查询订单是否已经支付 isPay(*$order_no*)

```


/**
 * 查询订单是否已经支付
 * @param string $order_no
 * @return bool
 */
public static function isPay($order_no)
```

6.创建微信二维码支付（扫码支付模式二）

```

/**
 *  创建微信二维码支付（扫码支付模式二）
 * @param WechatPay $pay 支付 SDK
```

```
     * @param string $order_no 系统订单号
     * @param int $fee 支付金额
     * @param string $title 订单标题
     * @param string $from 订单来源
     * @return false|string
     */
    public static function createWechatPayQrc(WechatPay $pay,
    $order_no, $fee, $title, $from = 'wechat')
```

7.创建微信 **JSAPI** 支付签名包

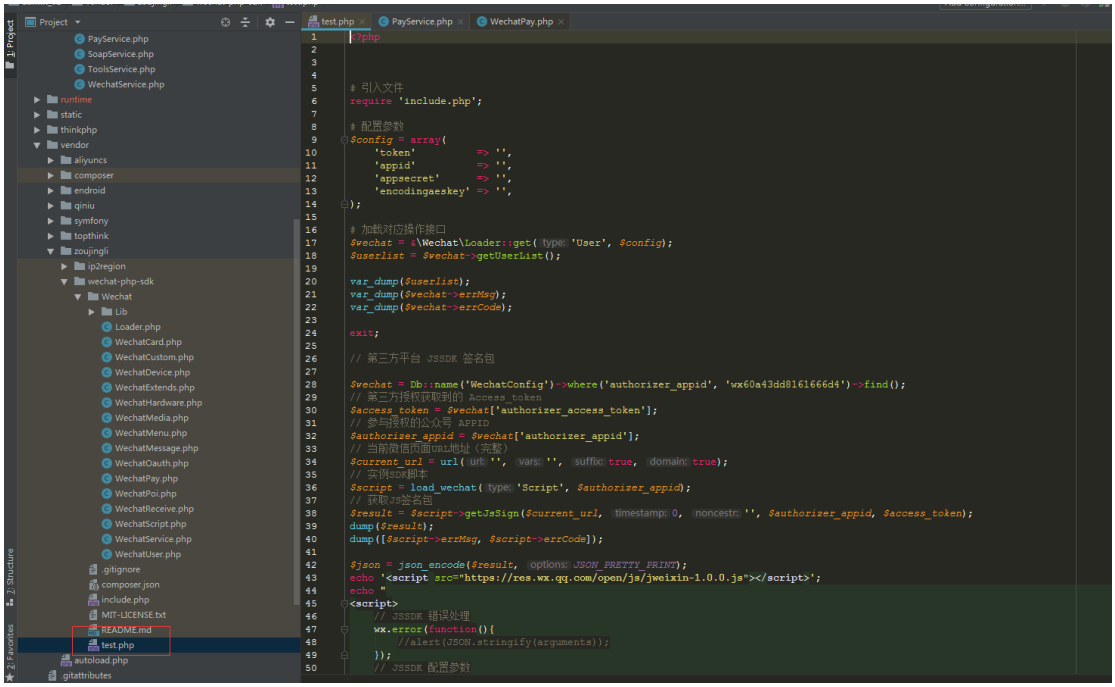
```
    /**
     * 创建微信 JSAPI 支付签名包
     * @param WechatPay $pay 支付 SDK
     * @param string $openid 微信用户 openid
     * @param string $order_no 系统订单号
     * @param int $fee 支付金额
     * @param string $title 订单标题
     * @return bool|array
     */
    public static function createWechatPayJsPicker(WechatPay $pay,
    $openid, $order_no, $fee, $title)
```

```
    /**
     * 微信退款操作
     * @param WechatPay $pay 支付 SDK
     * @param string $order_no 系统订单号
     * @param int $fee 退款金额
     * @param string|null $refund_no 退款订单号
     * @param string $refund_account
     * @return bool
     */
    public static function putWechatRefund(WechatPay $pay, $order_no,
    $fee = 0, $refund_no = null, $refund_account = '')
```

```
    /**
     * 创建微信预支付码
```

```
 * @param WechatPay $pay 支付 SDK
 * @param string $openid 支付者 Openid
 * @param string $order_no 实际订单号
 * @param int $fee 实际订单支付费用
 * @param string $title 订单标题
 * @param string $trade_type 付款方式
 * @param string $from 订单来源
 * @return bool|string
 */
public static function createWechatPrepayid(WechatPay $pay,
$openid, $order_no, $fee, $title, $trade_type = 'JSAPI', $from =
'wechat')
```

支付 demo



此脚本主要封装的关于文件的函数：七牛云，oss

8.七牛云上传：

```
/**
 * 七牛云存储
 * @param string $filename
 * @param string $content
 * @return array|null
 */
public static function qiniu($filename, $content)
```

9.阿里云 oss 上传

```
/**
 * 阿里云 OSS
 * @param string $filename
 * @param string $content
 * @return array|null
 */
public static function oss($filename, $content)
```

10.下载文件到本地

```
/**
 * 下载文件到本地
 * @param string $url 文件 URL 地址
 * @param bool $isForce 是否强制重新下载文件
 * @return array|null;
 */
public static function download($url, $isForce = false)
```

后台控制器基类中的函数常用函数(BaseApi)

注意：接口控制器没有集成 thinkphp 中的 controller，并且封装了 success 与 error 方法

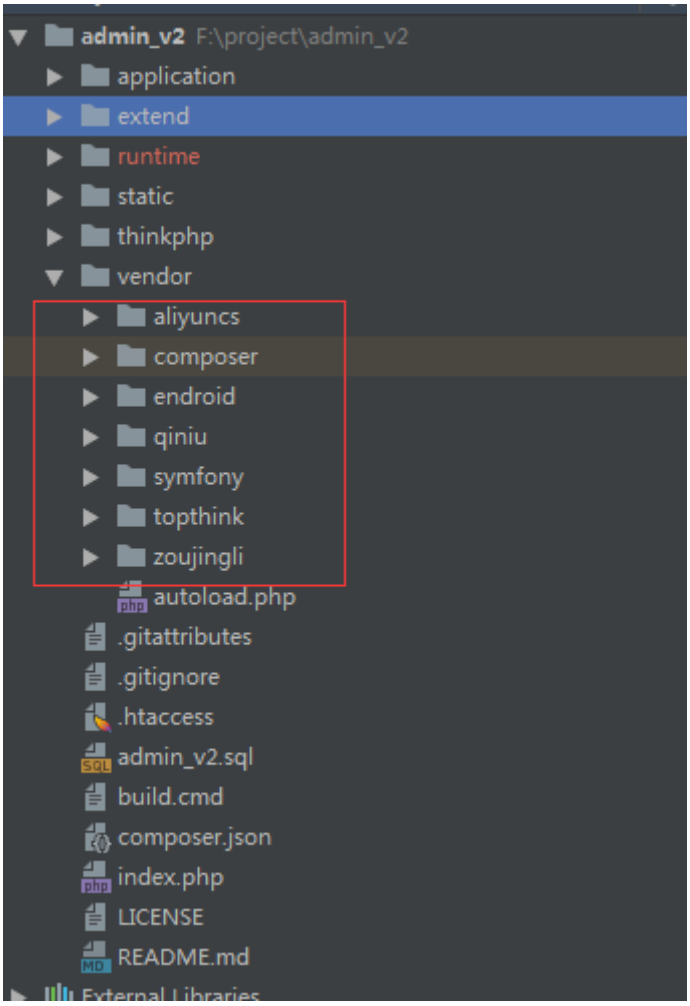
11.接口基类封装方法（success 与 error）

```
/**
 * 返回成功的操作
 * @param mixed $msg 消息内容
 * @param array $data 返回数据
 * @param integer $code 返回代码
 */
protected function success($msg, $data = [], $code = 1)
```

```
/**
 * 返回失败的请求
 * @param mixed $msg 消息内容
 * @param array $data 返回数据
 * @param integer $code 返回代码
 */
protected function error($msg, $data = [], $code = 0)
```

三、php 插件

Thinkadmin 所有的插件：



四、公共函数 common.php

写接口用到的数据加密函数

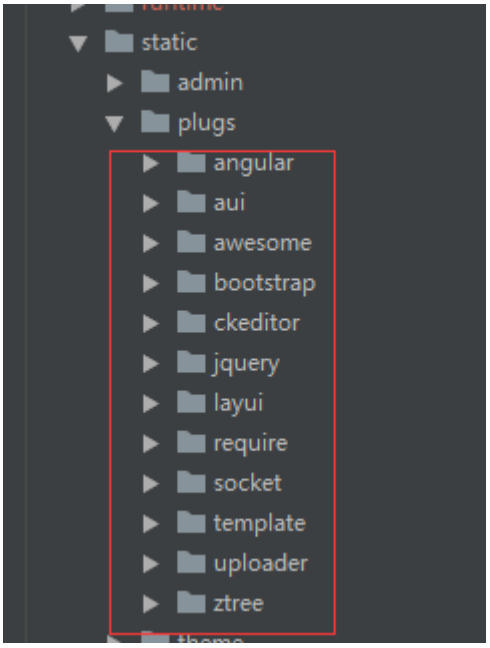
1.加密函数 encode()

```
/**
 * UTF8 字符串加密
 * @param string $string
 * @return string
 */
function encode($string)
```

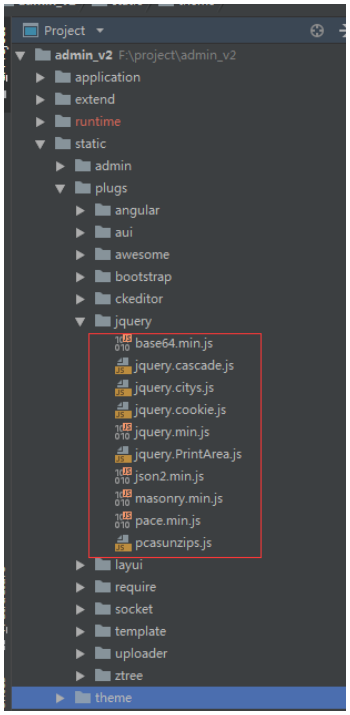
2.解密函数 decode()

```
/**
 * UTF8 字符串解密
 * @param string $string
 * @return string
 */
function decode($string)
```

五、前端插件



Angular:js 框架（参考网址 https://www.runoob.com/angularjs/angularjs-intro.html）
Layui：（产考网址 https://www.layui.com/demo/util.html）



城市三级联动：http://jquerywidget.com
与 jq 操作 cookie：https://www.cnblogs.com/webcome/p/5470975.html

1.$.form 通用表单数据插件

```
// 在内容区显示视图
$.form.show(html);

// 以 hash 打开网页
$.form.href(url, obj);

// 刷新当前页面
$.form.reload();

// 异步加载的数据
$.form.load(url, data, type, callback, loading, tips, time);

// 加载 HTML 到目标位置
$.form.open(url, data, callback, loading, tips);

// 打开一个 iframe 窗口
```

```
$.form.iframe(url, title);

// 加载 HTML 到弹出层
$.form.modal(url, data, title, callback, loading, tips);
```

2.$.msg 通用消息弹出提示插件

```
// 显示正在加载中的提示
var index = $.msg.loading(msg, callback);

// 显示失败类型的消息
var index = $.msg.error(msg, time, callback);

// 显示成功类型的消息
var index = $.msg.success(msg, time, callback);

// 确认对话框
var index = $.msg.confirm(msg, ok, no);

// 弹出警告消息框
var index = $.msg.alert(msg, callback);

// 状态消息提示
var index = $.msg.tips(msg, time, callback);

// 关闭弹出层
$.msg.close(index);
```

3.$.vali 通用表单验证插件

```
// 手动实现表单验证
$(html 表单选择器).vali(function(ret){
        console.log(ret); // 通过验证后获取的数据
});

// 初始化自动表单验证
$.vali.listen();
```

4.引入框架封装的的文本编辑器

```
require(['ckedit'], function () {
      Var editor = window.createEditor('[name="content"]');
}
```

5.引入框架封装的日期插件

```
require(['jquery', 'laydate'], function () {

}
```

六、 html 标签

1.data-load 标准异常网络请求（其中菜单不会发生改变）

- data-load 指定 get 网络请求地址
- data-tips 指定异步网络请求 tips 提示
- data-confirm 是否启用提示确认框

```
// 只加载内容到内容区
<a data-load="__URL__">加载 html 到内容区</a>

// 加载内容到内容区，同时显示提示
<a  data-load="__URL__"  data-tips='正在加载，请稍候...'>加载 html
到内容区</a>

// 加载内容到内容区，在加载前提示用户是否要继续操作
<a data-load="__URL__" data-confirm='确定要删除数据吗?' >删除数据
<a>
```

如果返回的数据是 `html`(也就是 `string` 类型) 则会将数据插入到内容区域;
如果返回的数据是 `JSON` 格式，则会使用消息自动处理机制，调用 `$.msg.auto` 方法来处理。

Ps：js 注册的监听事件在 listen.js 脚本中

2.data-open 内容区打开新页（其中菜单会发生改变）

- data-open 指定 get 网络请求地址

```
// 只加载内容到内容区
<a data-open="__URL__">加载 html 到内容区</a>
```

如果返回的数据是 `html`(也就是 `string` 类型) 则会将数据插入到内容区域;
如果返回的数据是 `JSON` 格式，则会使用消息自动处理机制，调用 `$.msg.auto` 方法来处理。

3.data-modal Modal 弹出层

- data-modal 指定 get 网络请求地址

```
// 只加载内容到内容区
<a data-modal="__URL__">加载 html 到 modal</a>
```

如果返回的数据是 `html`(也就是 `string` 类型) 则会将数据插入到内容区域;
如果返回的数据是 `JSON` 格式，则会使用消息自动处理机制，调用 `$.msg.auto` 方法来处理。

4.data-reload 强制刷新内容区

强制刷新内容区

```
<a data-reload='true'>刷新</a>
```

5.data-update 更新数据

- data-update 指定需要操作的数据记录 ID，多个以英文逗号分割
- data-field 设置需要更新的字段名称
- data-value 设置将需要更新的字段更新的值
- data-action 设置服务端处理的 URL 地址

```
<a data-update="1,3,4" data-field='status' data-value='0' data-action='http://demo.thinkadmin.top/forbid.html'>禁用</a>
```

针对 `ThinkAdmin` 删除记录的处理

- 服务端对 `data-field=deleted` 有特殊处理过，

  如果对应表有 `is_deleted` 字段则为软删除，否为硬删除。

```
 <a      data-update="1,3,4"      data-field='delete'      data-action='http://demo.thinkadmin.top/del.html'>删除</a>
```

6.data-href 打开新的网页

* 打开新的网页，与 `a` 标签的 `href` 类似

```
<a data-href='__URL__'>打开新页</a>
```

7.data-file 文件上传插件

以 **HTML** 属性的方式实现文件上传

- data-type 限定允许上传的文件类型

- data-field 绑定对应表单字段的 name 属性
- data-file 设置文件上传模式（one: 单文件上传, mut: 多文件上传）

```
<input name='file-field'>
<button type='button' data-file='one' data-field='file-field' d
ata-type='jpg,png,gif'>上传文件</button>
```

以 `JavaScript` 实现图片上传，`input` 上可以加 `data-type` 来设置上传文件的后缀。

```
<input name='file_field'>
<script>
    $(function(){
        $('[name="file_field"]').uploadOneImage();
    });
</script>
```

以 `JavaScript` 实现多图片上传，`input` 上可以加 `data-type` 来设置上传文件的后缀。

```
<input name='fiel_field'>
<script>
    $(function(){
         $('[name="file_field"]').uploadMultipleImage();
    });
</script>
```

8.data-iframe Iframe 打开网页

* 以 `iframe` 方式打开网页

```
<a data-iframe='__URL__'>打开新页</a>
```

9.data-icon 打开图标选择器

- 打开系统图标选择器, data-icon 的值用来指定表单对应字段名称

```
<input name='icon'>
<button data-icon='icon'>选择图标</button>
```

10.data-tips-image 点击图片展示图片

- 点击图片展示图片

```
// 展示 img 的 src 对应图片
<img data-tips-image  src='https://demo.thinkadmin.top/static/upload/f47b8fe06e38ae99/08e8398da45583b9.png'>

// 展示指定 src 的图片
<img data-tips-image="指定的图片 SRC 地址"  src='https://demo.thinkadmin.top/static/upload/f47b8fe06e38ae99/08e8398da45583b9.png'>
```

11.data-tips-text 鼠标悬停文字提示

- 鼠标悬停文字提示

```
<img data-tips-text='tips 文 字 ' src='https://demo.thinkadmin.top/static/upload/f47b8fe06e38ae99/08e8398da45583b9.png'>
```

12.data-phone-view 以手机模式显示

- 以 `iframe` 方式打开手机新页

```
<a data-phone-view='__URL__'>打开手机新页</a>
```

六、注意问题

1.分页问题

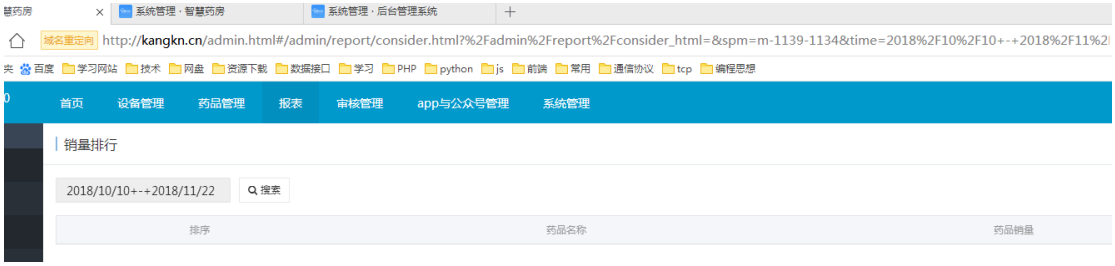
使用 thinkphp 类的分页，要达到与框架一样的分页效果，如下代码：

```
        }

        $list = $db->paginate(20,false,["query" =>$get]);
        $page= preg_replace(['|href="(.*?)"|','|pagination|'],['data-open="$1" href="javascript:void(0);"','pagination pull-right'],$list->render());
```

2.使用日期插件问题

分页+时间段搜索或出现如下图问题



原因 get 提交的请求有特殊字符（例如空格）。使用 urlencode（）函数解决此问题

```
        $get["time"]=trim($arr["0"])."-".trim($a
    $get["time"]=urlencode($get["time"]);
```

3.input 自动获取焦点问题

Input 框自动获取焦点，行内式：

```
<div class="layui-form-item">
  <label class="layui-form-label">联系手机</label>
  <div class="layui-input-block">
    <input type="tel" autofocus name="phone" value="13888888855" pattern="^1[3-9][0-9]{9}$" title="请输入联系手机" placeholder="请输入联系手机" class="layui-input validate-error"> == $0
    <span style="animation-duration: 0.2s; padding-right: 20px; color: rgb(169, 68, 66); position: absolute; right: 0px; font-
```

这种写法在第二次访问点击的时候不能自动获取焦点，应使用 js 获取焦点，如下。

```
    <script>window.form.render();
    $("input[name='barcode']").focus();
```