# 关于本文档

作者：[**Kenvix**](http://zhizhe8.net)

文档修改时间：2016-5-8

# 2.插件基础结构说明

### 2.1.插件目录结构说明

所有插件都应该放到 /plugins/ 文件夹，在该目录下 创建一个 *插件名* 的文件夹，

然后在这个文件夹里放置一个 *插件名*.php就成为一个可以被系统扫描到的插件了。

打包插件时，请打包插件目录。

### 2.2.插件文件结构说明

|  |  |  |
| --- | --- | --- |
| **插件文件** | **选择性** | **描述** |
| [*插件名*.php](#_2.3.插件名.php说明) | 必须 | 插件核心文件，会被系统加载 |
| [*插件名*\_desc.php](#_2.4.插件名_desc.php说明) | 必须 | 插件自述文件，会被系统加载 |
| *插件名*\_setting.php | 可选 | 插件设置文件，管理员可见 |
| [*插件名*\_callback.php](#_2.7.插件名_callback.php_说明) | 可选 | 系统操作插件时的回调文件，[用法参见**2.7**](#_2.7.插件名_callback.php_说明) |
| [*插件名*\_show.php](#_2.6.插件名_show.php_插件名_public.php_插件名) | 可选 | 插件前台页面，所有登录的用户可见，[用法参见 **2.6**](#_2.6.插件名_show.php_插件名_public.php_插件名) |
| [*插件名*\_public.php](#_2.6.插件名_show.php_插件名_public.php_插件名) | 可选 | 插件的公共页面，即使用户不登陆也可查看，[用法参见 **2.6**](#_2.6.插件名_show.php_插件名_public.php_插件名) |
| [*插件名*\_private.php](#_2.6.插件名_show.php_插件名_public.php_插件名) | 可选 | 插件的私有页面，只有管理员可以查看，[用法参见 **2.6**](#_2.6.插件名_show.php_插件名_public.php_插件名) |
| [*插件名*\_vip.php](#_2.6.插件名_show.php_插件名_public.php_插件名) | 可选 | 插件的VIP页面，只有VIP和管理员可以查看，[用法参见 **2.6**](#_2.6.插件名_show.php_插件名_public.php_插件名) |
| [*插件自定义页面*.php](#_2.8.插件自定义页面.php_说明) | 可选 | 插件自定义的页面，[用法参见**2.8**](#_2.8.插件自定义页面.php_说明) |
| *插件自定义文件* | 可选 | 插件自定义的文件，自由使用，无限制 |

**不兼容性**

因为安全原因，取消了原来的核心插件功能。

**如何兼容旧版云签到**

参见旧版插件开发文档，按旧版插件模式写一份插件自述即可

**保证插件安全**

除非你打算直接让用户访问这个php文件，否则，所有的文件开头都应该加上：

<?php if (!defined('SYSTEM\_ROOT')) { die('Insufficient Permissions'); } ?>

### 2.3.*插件名*.php说明

你可以使用 addAction() 将你想要挂载的函数挂载到挂载点上，以 调试信息插件 为例

<?php

//创建一个名为wmzz\_debug\_system1的函数

function wmzz\_debug\_system1() {

$GLOBALS['wmzz\_debug\_time'] = microtime();

}

//将wmzz\_debug\_system1()挂载到 header 挂载点

addAction('header','wmzz\_debug\_system1');

如果要挂载的挂载点会向被挂载的函数传入参数（如error），则需要使该函数能够接收传入的参数，如：

function fyy\_errorlog\_take1($errno,$errstr,$errfile,$errline,$errnoo) {

echo $errstr;

}

addAction('error','fyy\_errorlog\_take1');

写函数名建议以 *前缀\_插件名\_函数名* 的形式命名，避免与其他函数发生冲突导致程序崩溃.

如果你**不打算兼容PHP5.3 以下版本**，则还可以使用命名空间定义函数；计划用于挂载点的函数则可使用闭包，如：

//这和上面的代码是等效的

addAction('error', function($errno,$errstr,$errfile,$errline,$errnoo) {

echo $errstr;

});

### 2.4.*插件名*\_desc.php说明

用于自述插件，以调试信息插件为例说明说明：

<?php if (!defined('SYSTEM\_ROOT')) { die('Insufficient Permissions'); }

//没有标注必填的，都是选填，但是按照规范，请保留该键，只需将值留空即可

return array(

'plugin' => array(

'name' => '调试信息', //插件名称，必填

'version' => '2.0', //插件版本号

'description' => '在底部显示调试信息，并为开发新架构插件提供范例和详细的注释', //插件描述

'onsale' => true, //bool 插件是否已在产品中心上架

'url' => 'http://zhizhe8.net', //插件地址，比如哪里可以下载到这个插件

'for' => 'all', //适用的云签到版本，all为所有版本，版本后面跟+表示适用于该版本或更高版本，如V4.0+

'forphp' => 'all' //适用的PHP版本，如果定义了，系统就在安装和激活时进行版本对比，如果版本低于forphp，自动禁止下一步操作，all为所有版本

),

'author' => array(

'author' => 'Kenvix', //作者名称

'email' => 'kenvix@vip.qq.com', //作者邮箱

'url' => 'http://zhizhe8.net' //作者的个人网站

),

'view' => array(

//以下设置均只影响插件列表页面是否有对应按钮

'setting' => true, //bool 插件是否有设置页面，必填

'show' => true, //bool 插件是否有展示页面，必填

'vip' => false, //bool 插件是否有只给VIP看的页面，必填

'private' => false, //bool 插件是否有只给管理员看的页面，必填

'public' => false, //bool 插件是否有给任何人（包括未登录的）看的页面，必填

'update' => true, //bool 插件如果有新版本，是否在插件列表页面显示升级按钮

),

'page' => array(

//规定插件有哪些自定义页面，不需要自定义页面可留空

//自定义页面访问方式：index.php?mod=view:插件名:自定义页面名

//程序将自动在插件目录下寻找并加载 view\_自定义页面名.php

//任何人都能查看自定义页面，包括未登录的用户，因此你必须自己写好权限控制

'phpinfo' //定义一个名为phpinfo的自定义页面，位于/plugins/wmzz\_debug/view\_phpinfo.php

)

);

### 2.5.*插件名*\_setting.php说明

请参见“插件设置读写“

### 2.6.*插件名*\_show.php *插件名*\_public.php *插件名*\_vip.php *插件名*\_private.php 说明

这些页面都是由插件完全设计的页面，程序本身不会有任何输出内容

如需加载用户界面，请在输出开始前加上 loadhead(); 输出完成后加上 loadfoot();

范例：

<?php

loadhead('我是标题title'); //无参数则为无标题

echo 'hey';

loadfoot();

友情提示：loadfoot(); 不会输出云签到程序版权信息

如需访问插件前台页面，请往上查看“插件文件结构说明”

**Tip:** 你可以加入权限检查来实现只允许一部分人访问

### 2.7.*插件名*\_callback.php 说明

这是一个可选的文件，在进行安装、激活、卸载等操作时会自动调用

|  |  |
| --- | --- |
| **回调函数** | **描述** |
| callback\_install() | 在插件被安装时调用 |
| callback\_remove() | 在插件被卸载时调用，一般用于清扫垃圾（比如在数据库留下的东西）  若插件是激活的，则先调用callback\_inactive() |
| callback\_init() | 在插件被激活时调用 |
| callback\_inactive() | 在插件被禁用或卸载激活的插件时调用 |
| callback\_setting() | 在插件使用插件设置储存接口存储设置时调用，参见“[插件设置读写](#_3.1.4.2.自己编写插件设置存储接口) |

回调函数用法示例：

function callback\_init() {

msg('hello');

}

### 2.8.插件自定义页面.php 说明

你可以规定插件有哪些自定义页面，不需要自定义页面可留空

自定义页面访问方式：index.php?mod=view:*插件名*:*自定义页面名*

程序将自动在插件目录下寻找并加载 view\_*自定义页面名*.php

任何人都能查看自定义页面，包括未登录的用户，因此你必须自己写好权限控制

要使用此功能，你需要先在 *插件名*\_desc.php 中指定这个页面

然后在插件目录下新建 view\_*自定义页面名.php，即可使用本功能*

# 3.设置读写

灰底表示函数原型

@param 表示参数

@return 表示返回值

#### 3.1.1.读写全局设置

全局设置位于 tc\_options 表

##### 读取设置

@param string $name 设置项名称

@return string

public static function get($name)

举例：option::get('foo');

##### 改变或添加一个设置 (不存在时自动添加)

@param string $name 设置项名称

@param string $value 值

public static function set($name,$value)

举例：option::set('foo', '666');

##### 直接添加一个设置 (已存在时不操作)

@param string $name 设置项名称

@param string $value 值

public static function add($name,$value)

举例：option::add('foo', '666');

##### 删除一个设置（不存在时不操作）

@param string $name 设置项名称

public static function del($name)

举例：option::del('foo');

#### 3.1.2.读写插件设置

插件设置的读写包括批量读写（即一次性读出插件的所有设置），和单个读写（读取一个设置）

这里的读写插件设置指的是使用标准API所存储的设置，数据存储于 tc\_plugins 表

使用标准API存储的设置将会在插件卸载时自动删除

插件也可以把设置存储成全局设置（储存到这里的设置将不会自动删除）

##### 3.1.2.1.批量读写设置

###### 批量获取设置

@param string $plug 插件标识符

@return array 设置数组

public static function pget($plug)

举例：option::pget('wmzz\_post');

###### 保存插件的所有设置

@param string $plug 插件标识符

@param string $value array 设置数组

public static function pset($plug , $value)

###### 删除插件的所有设置

@param string $plug 插件标识符

public static function pdel($plug)

举例：option:pdel('wmzz\_post');

##### 3.1.2.2.单个设置读写

###### 获取插件的一条设置

@param string $plug 插件标识符

@param string $name 设置项名称

@return string 设置值

public static function xget($plug , $name)

###### 保存插件的一条设置，不存在则添加之

注意：需要大量修改的请直接将设置保存到全局设置

@param string $plug 插件标识符

@param string $name 设置项名称

@param string $value 值

public static function xset($plug , $name , $value)

###### 删除插件的一条设置

@param string $plug 插件标识符

@param string $name 设置项名称

public static function xdel($plug , $name )

###### 直接添加插件的一条设置已存在则跳过

@param string $plug 插件标识符

@param string $name 设置项名称

@param string $value 值

public static function xadd($plug , $name , $value)

#### 3.1.3.用户设置读写

使用此功能可以简单地为特定用户保存特定的设置

###### 获取用户的设置

@param string $name 设置项名称

@param string $uid 用户UID，默认当前用户的UID

@return string|bool 不存在时返回false

public static function uget($name, $uid = '')

###### 改变用户的设置

@param string $name 设置名

@param string $value 设置值

@param string $uid 用户UID，默认当前用户的UID

public static function uset($name , $value , $uid = '')

###### 清除用户的所有设置

@param string $uid 用户UID，默认当前用户的UID

public static function udel($uid = '')

###### 清除用户的指定设置

@param string $name 设置项名称

@param string $uid 用户UID，默认当前用户的UID

public static function udela($name , $uid = '')

###### 添加一个用户的设置

添加时会自动检查有关设置是否已存在

@param string $name 设置项名称

@param string $value 值

@param string $uid 用户UID，默认当前用户的UID

public static function uadd($name , $value , $uid = '')

#### 3.1.4.在插件中读写设置

##### 3.1.4.1.使用内置API简单地存取插件的设置

该API使用非常方便，只需POST提交一个表单即可自动存储所提交的所有内容

权限：只允许管理员访问

POST提交地址：setting.php?plugin=*插件名*

然后你可以参考“[读写插件设置](#_3.1.2.读写插件设置)”来了解如何读写

##### 3.1.4.2.自己编写插件设置存储接口

上面的存储方式是完全由系统完成的，插件无法控制，如果你想要自行编写存储代码，则可以使用这个方法

权限：只允许管理员访问

POST提交地址：setting.php?setplug=*插件名*

然后插件会调用 *插件名*\_callback.php 的 callback\_setting() 函数

用法示例：

function callback\_setting() {

if(empty($\_POST['text']) || empty($\_POST['time'])) msg('缺少关键信息');

option::xset('wmzz\_autoremovebduss', 'time', intval($\_POST['time']));

option::xset('wmzz\_autoremovebduss', 'mail', (!empty($\_POST['mail']) ? '1' : '0'));

option::xset('wmzz\_autoremovebduss', 'text', $\_POST['text']);

option::xset('wmzz\_autoremovebduss', 'title', $\_POST['title']);

}

##### 3.1.4.3.插件设置页面

访问 index.php?plugin=*插件名* 所显示的页面

实际是 *插件名*\_setting.php

以调试信息插件为例：

<?php

/\*\*

\* 插件设置页面

\* 在插件列表页面点击设置插件会自动加载此文件

\* 已自动加载云签到UI

\*/

if (!defined('SYSTEM\_ROOT')) { die('Insufficient Permissions'); }

global $i;

echo '调试信息：<br/><br/>';

dump($i,true);

?>

# 4.其他功能

请自己翻看/lib/看吧