插件开发指南

ThinkSNS V2.5

智士软件（北京）有限公司

ZhiShiSoft (Beijing) Co,. Ltd.

序言

各位亲爱的用户朋友们，经历了很长时间的等待，我们终于发布了全新的 ThinkSNS2.5，一直以来都有一批忠实的用户陪伴在我们身边，让我们即使在很困难的情况下也没有忘记自己的使命，感谢你们一直以来的支持，新的产品虽然还不够完善，相信经过我们共同的努力，一定可以把她打造成一流的社区软件！

更多关于产品的信息和反馈，请登录我们的官方社区：http://t.thinksns.com.

目录

1.简介

1.1 什么是 ThinkSNS

ThinkSNS V2.5 是由智士软件（北京）有限公司开发的一款开源社区软件，定位于基于微博客的多应用 SNS 系统，帮助用户打造注重交流、沟通的垂直类社区网站，新版本的 ThinkSNS 具备如下诸多新特性。

完整的微博系统

微博作为一个新兴的工具，在近些年获得了极大的用户增长，随着门户微博的推出，微博功能已经成为了网站的标准功能，快速，小巧，易用是它的特点，ThinkSNS2.5 提供了一

套标准的微博程序，涵盖了微博全方面的功能，并将微博作为其他应用互通的基础应用，让用户在使用过程中能够获得统一的使用体验。

内置接入多个应用平台，无限应用扩展

为了让用户获得更多可用的应用，我们提供了多种应用接入方式，并内置了多个应用平台，站长无需自己开发，就可以获得海量应用，让用户有更多的选择。

支持 WAP，提供手机客户端

移动互联网是大势所趋，ThinkSNS2.5 不仅提供了对 WAP 浏览的支持，还将免费提供应用最广的 2 大手机平台— iPhone 和 Andoid — 的客户端产品，让每一个网站都能搭上移动互联网的快车。

多系统账号登录支持

用户注册是用户参与网站互动、发表内容时必不可少的步骤，但往往由于步骤的繁琐导致潜在用户的流失，ThinkSNS2.5 内置多个知名站点的账户登录支持（新浪微博、豆瓣等），用户只需一步操作就可以快速进入社区参与讨论，让繁琐的步骤不再成为交流的隔阂。

集成社交媒体平台

ThinkSNS2.5 紧密集成各大社交网站（新浪微博等），利用大网站超高的人气，带动新生社区的发展，只需要简单的绑定，就可以让用户将信息在各个网站之间同步，方面用户的信息管理。

插件、应用双重扩展机制

ThinkSNS 从来都是二次开发者的利器，2.5 版本更是具备了小插件与大型应用的双重扩展机制，应用的开发和扩展更加独立、灵活，同时开发的文档，各种接口也日趋完善，非常易于做二次开发，让站长很容易的做出更为个性化的网站。

1.2 编写目的

本文档旨在帮助开发者快速理解 ThinkSNS2.5 的系统架构和开发方法（应用、插件、模板），为二次开发提供参考。

ThinkSNS2.5 内核使用优化的 ThinkPHP 框架，ThinkPHP 框架的绝大多数规范和参考对 ThinkSNS2.5 都有效，建议您同时参阅 ThinkPHP 文档（http://thinkphp.cn/Manual）。

2.架构设计

2.1 设计目标

- 核心高效稳定
- 应用独立开发部署
- 插件平行扩展
- 功能平行扩展

2.2 核心架构及目录结构

概览

图 1 ThinkSNS2.5 核心结构图

如图 1 所示，核心与服务、公共 Model、插件、Widget、第三方类库共同构成了系统的大根基，其他所有应用都其上构建。

术语解释

- **核心**：源自 ThinkPHP 框架，为系统提供 MVC 分离、底层数据库支持等核心功能，并提供诸多便捷的类库和函数库供系统其他部分使用。位于**/core/**目录。
- **服务**：一组全局通用的类库，实现对特定功能的封装。位于**/addons/services/**目录，如邮件发送（Mail）、用户认证（Passport）等。
- **公共 Model**：一组全局通用模型。位于**/addons/models/**目录，如附件模型（AttachModel）、地区模型（AreaModel）等。
- **插件**：为实现某种功能而增加的程序文件。位于**/addons/plugins/**目录，包括第三

方平台登陆插件和勋章两种。

- **Widget**：一组可以在任意 HTML 页面调用的代码块。位于**/addons/widgets/**目录，包括评论（Comment）、选择好友（SelectFriend）等。
- **第三方类库**：其他开源的第三方类库。位于**/addons/libs/**目录，如 phpmailer 等。
- **应用**：实现特定功能的独立模块，基于上述的系统结构构建。位于**/apps/**目录，如日志（Blog）、相册（Photo）等。
- **API**：应用程序编程接口（Application Programming Interface）。位于**/api/**目录，如微博 API、用户资料 API 等。

微博

微博虽然也是以系统独立应用的身份出现（如图 1 所示），但它还肩负着系统核心应用的责任。许多系统元素完全构建于微博应用之上，如微博 Widget（即“分享”功能）是直接操作微博，而 WAP 应用则是完全使用微博 API 架构。

漫游平台

系统应用中另一个特例是漫游应用。漫游应用来源于康盛的漫游平台，由于漫游平台的 URL 根据 UCHome 的目录结构生成，这与 ThinkSNS 的目录结构完全不同，而且还需兼顾到漫游平台的可移植性，所以集成的漫游应用不走标准的 ThinkSNS2.5 核心，而是使用漫游应用内部已实现的一个最简版框架。

如果您是模版开发人员，那您得特别留意！因为您在公共目录下修改完头部和左侧的页面后，还要修改漫游下的头部和左侧（位于**/apps/myop/themes/classic/**目录）。

目录结构

ThinkSNS2.5

```
├─ _runtime                  ----------运行时缓存
├─ addons                    ----------扩展库
|   ├─ libs                  ----------第三方类库
|   ├─ models                ----------公共 Model
|   ├─ plugins               ----------插件
|   |     ├─ login           ----------第三方平台登录插件
|   |     ├─ Medal           ----------勋章
|   |     └ Tags             ----------标签
|   ├─ services              ----------系统服务
|   └ widgets          ----------系统 Widget
├─ api                       ----------API 库
├─ apps                      ----------系统应用
|   ├─ admin                 ----------管理后台
|   ├─ home                  ----------Home 应用
|   ├─ myop                  ----------漫游应用
```

```
|   ├─ wap                  ----------手机 WAP 端
|   └─ weibo                ----------微博应用
├─ core                     ----------核心
|   ├─ sociax               ----------系统核心文件
|   ├─ ThinkPHP             ----------ThinkPHP 核心
|   └─ sociax.php           ----------核心引导文件
├─ data                     ----------站点数据
├─ install                  ----------系统安装文件
├─ public
|   ├─ admin                ----------管理后台的样式
|   ├─ js                   ----------系统 JS 库
|   └─ themes               ----------系统模板
├─ access.php                        ----------节点权限控制文件
├─ cleancache.php                    ----------缓存清理文件
├─ config.inc.php          ----------站点配置文件
├─ index.php               ----------站点入口文件
├─ shorturl.php            ----------短地址文件
└─ thumb.php               ----------自动缩略图生成文件
```

2.3 应用架构及目录结构

应用的目录位置及结构：

```
ThinkSNS2.5
    ├─ apps
        ├─ app
            ├─ Appinfo      ----------安装信息、安装\卸载执行文件、图标
            ├─ Common       ----------函数库 common.php
            ├─ Conf         ----------    项目配置 config.php
            ├─ Language     ----------通知、动态的语言包
            ├─ Lib
            |   ├─ Action ----------操作类库
            |   ├─ Model    ----------模型类库
            |   └─ Widget ----------插件库
            └─ Tpl          ----------模板、css、js 文件
```

- 入口文件

  ThinkSNS2.5 只有一个公共入口文件，即 ThinkSNS 目录下的 **index.php**.

- URL 模式

  URL 的访问方式是 **index.php?app=APP_NAME&mod=Action&act=function**

- 函数库

  应用自身的函数库放在该应用目录下的 **Common/common.php** 里即可，这里面的函数会随该应用一起加载，可在该应用内随意调用。系统函数库请参阅附录的“函数库”。

- 模板

```
<include file="__THEME__/header" />
  <!-- 内容 begin -->
```

```
<div class="content no_bg">
  <div class="main no_l">
    <div class="mainbox">
      <!-- 画布 begin -->
      <div class="mainbox_C">
        [……]
      </div>
      <!-- 画布 end -->
    </div>
  </div>
</div>
<!-- 内容 end -->
<include file="__THEME__/footer" />
```

- 应用的样式文件统一存放在的 Tpl 下的 **Public/**目录，通过`../Public/xxx.css` 引用，应用的 JS 文件统一放到应用项目下的 **Tpl/**下的 **Public/js/**目录，通过`../Public/js/xxx.js` 引用。

3.开发指南

3.1 命名规范与编码规范

参考 ThinkPHP 的命名与编码规范：http://thinkphp.cn/Manual/20

3.2 使用函数库、类库和 Widget

使用系统函数库

系统函数位于**/core/sociax/functions.php** 和**/core/sociax/extend.php** 文件，为全局有效函数，可以直接调用。

如获取用户昵称的方法：`$uname = getUserName($uid);`

使用服务

服务位于**/addons/services/**目录下，通过 `service('serviceName')->method($param);` 来使用服务。

如验证用户是否登录的方法：`$is_logged = service('Passport')->isLogged();`

使用公共 Model

公共 Model 位于**/addons/models/**目录下，通过 `model('modelName')->method($param);`来使用公共 Model。

如获得地区列表的方法：`$area_list = model('Area')->getAreaList();`

使用 Widget

Widget 位于**/addons/widgets/**目录下，通过 `W('widgetName',array('param'=>'value'))`来调用 Widget。一般 Widget 是用在页面中，所以调用方法为：`{:W('widgetName',array('param'=>'value'))}`。

如在页面中展示可能认识的人的方法：`{:W('RelatedUse',array('uid'=>'1'))}`

使用第三方类库

第三方类库一般放置在**/addons/libs/**目录下，使用前通过 `include_once` 等函数将文件引入即可。

3.3 使用弹出窗、提示消息、编辑器

在 ThinkSNS2.5 的世界里，jQuery 库是我们默认的 JS 框架，它在页面头部自动载入。ThinkSNS2.5 在 jQuery 的基础上对弹出窗、提示消息和 KISSY 编辑器进行了封装，本节主要介绍它们的使用方法（jQuery 库官方文档：http://docs.jquery.com/Main_Page）。

弹出窗

图 2 弹出窗效果图

如图 2 所示，弹出窗至少包含标题、关闭按钮和内容三部分，为保证用户操作的连贯性，一般会在内容部分添加“确定”和“取消”按钮。

- 弹出窗的调用方法：

```
<script>
function yourFunc() {
```

```
        ui.box.load(your_url, {title:'这里是标题'});
    }
    </script>
```

- 在 your_url 中放置弹出窗的内容。注意：“确定”和“取消”按钮也是内容的一部分！
- 关闭弹出窗：

```
<script>
function close() {
    ui.box.close();
}
</script>
```

提示消息

图 3 提示消息效果图

调用方法：

```
<script>
function success() {
    ui.success("更新完成");
}
function error() {
    ui.error("新 Email 已存在");
}
</script>
```

编辑器

编辑页中使用编辑器需要以下四步：

1. 引入编辑器样式（已在前台页面头部加载，勿重复引入）。如下所示（请直接复制！）：

```
<link href="__PUBLIC__/js/editor/editor/theme/base-min.css" rel="stylesheet"/>
<!--[if lt IE 8]>
<link href="__PUBLIC__/js/editor/editor/theme/cool/editor-pkg-min-mhtml.css"
rel="stylesheet"/>
<![endif]-->
<!--[if gte IE 8]><!-->
<link
href="__PUBLIC__/js/editor/editor/theme/cool/editor-pkg-min-datauri.css"
rel="stylesheet"/>
<!--<![endif]-->
```

2. 引入编辑器 JS 文件。由于需要加载的 JS 文件较多，所以可以通过引入一个通用文

件的方式引入所有编辑器 JS 文件：

```
<include file="__THEME__/editor" />
```

3. 定义 textarea（宽度和高度请在此处定义）：

```
<textarea id="idOfTextArea" style="width:650px;height:350px"></textarea>
```

4. 调用编辑器加载函数。

```
<script>
$(document).ready(function(){
    loadEditor("idOfTextArea");
});
</script>
```

展示页中使用编辑器需要以下两步：

1. 引入编辑器样式（已在前台页面头部加载，勿重复引入）。如下所示（请直接复制！）：

```
<link href="__PUBLIC__/js/editor/editor/theme/base-min.css" rel="stylesheet"/>
<!--[if lt IE 8]>
<link href="__PUBLIC__/js/editor/editor/theme/cool/editor-pkg-min-mhtml.css"
rel="stylesheet"/>
<![endif]-->
<!--[if gte IE 8]><!-->
<link
href="__PUBLIC__/js/editor/editor/theme/cool/editor-pkg-min-datauri.css"
rel="stylesheet"/>
<!--<![endif]-->
```

2. 将内容用`<div class="ke-post"></div>`包裹起来：

```
<div class="ke-post">{$content}</div>
```

3.4 ThinkPHP 开发指南

ThinkPHP 官方文档 – 开发指南：http://thinkphp.cn/Manual/50
ThinkPHP 官方文档 – 模板指南：http://thinkphp.cn/Manual/194

3.5 插件开发

什么是插件？

插件位于**/addons/plugins/**目录下

插件是在不修改任何应用-核心代码的情况下扩展某些功能。具备可启动可关闭特性。TS 的插件机制是基于 Hook 的，所以每个插件实现的 hook 必然会在制定位置执行一次。多个插件同时实现同一个钩子时也会有顺序执行。可以将插件想象成一个在核心制定位置中让二次开发工作独立开发的小片段。在这个片段里，所有的掌控都是在开发员手上

开发流程

1. 选择插件开发方式，以及告知插件系统需要的一些信息
2. 实现插件所希望实现的钩子
3. 安装、调试、运行

实现插件 Addons 抽象类的描述对象

Addons 是插件的描述抽象对象，为了方便开发，在这个基础上再做了一层类型封装：

- SimpleAddons：简单插件
- NormalAddons：复杂的插件

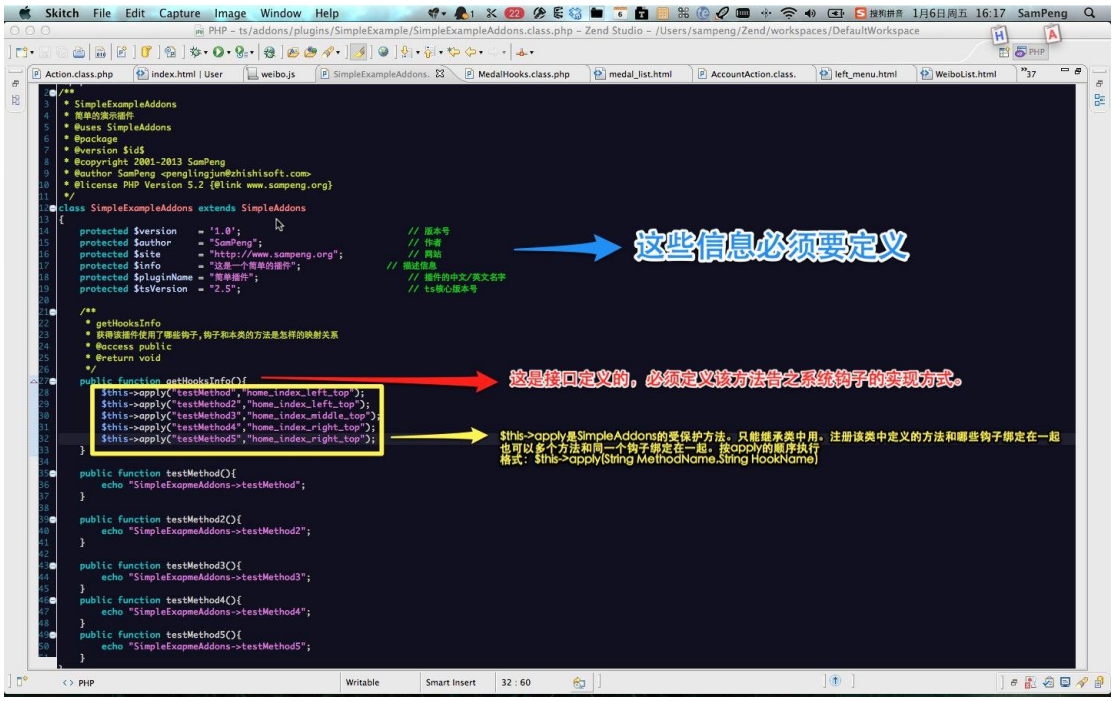
在**/addons/plugins/**插件目录下创建一个希望做的插件，这里我们叫做SimpleExample。这个就标示了该插件的名称为 SimpleExample.系统将以该名称为唯一标示名

然后在 SimpleExample 中建立一个插件描述文件:SimpleExampleAddons.class.php.

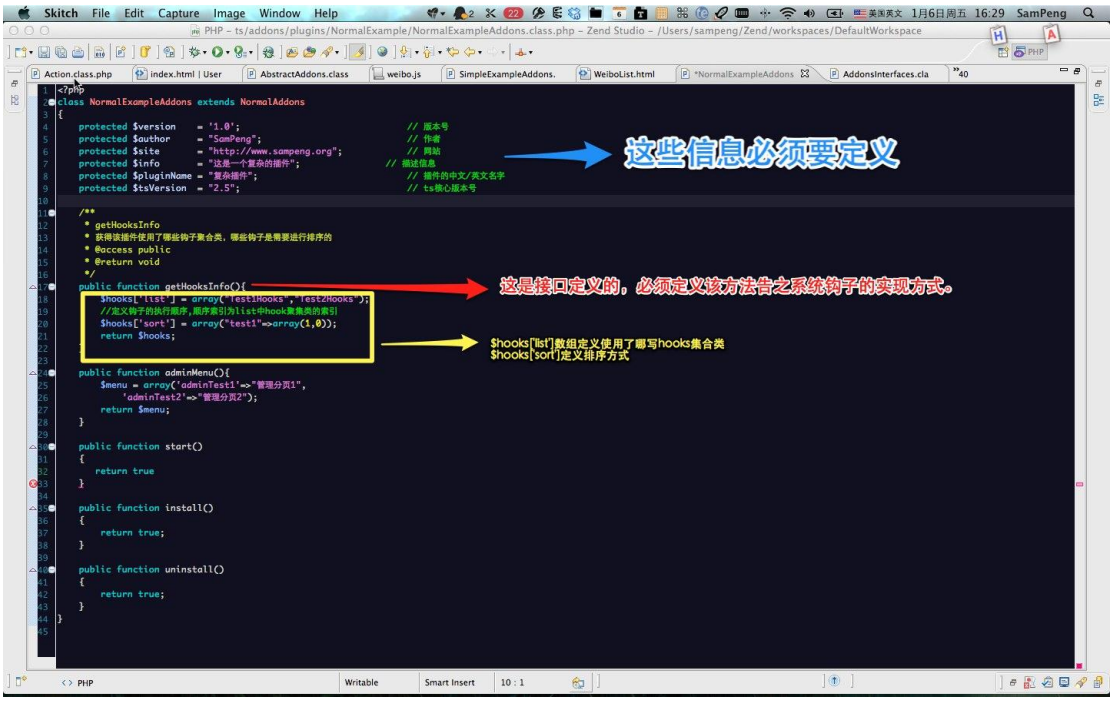
注意：该名字一定要是 插件名字+Addons+.class.php 这样的标准格式

最后在该描述文件中告之想做的插件的一些信息。

简单钩子的描述方式和开发如图：



复杂插件的描述信息如图：



请注意，不论简单还是复杂插件，必须实现以下接口：

- start() :在后台启动该插件时需要做的初始化动作。返回 true 和 false。通常是用来启动插件时检测功能依赖
- install():在后台第一次启动插件时需要做的初始化动作。注意：和 start 不同的是，install 只在第一次启动时运行，而 start 是每次停用后启用时运行。通常在这里定义插件需要的数据库操作
- uninstall():在后台卸载插件时需要做的卸载操作。通常定义该插件卸载时清除数据库的操作。由于是数据库操作，请谨慎操作。

实现插件所希望实现的钩子

1. 以简单插件为例。在 getHooksInfo()中 apply 后的方法名，在本类中同样定义即可。

第一步：

```
public function getHooksInfo(){
    $this->apply("testMethod","home_index_left_top");
    $this->apply("testMethod2","home_index_left_top");
    $this->apply("testMethod3","home_index_middle_top");
    $this->apply("testMethod4","home_index_right_top");
    $this->apply("testMethod5","home_index_right_top");
}
```

第二步：

```php
public function testMethod(){
    echo "SimpleExapmeAddons->testMethod";
}

public function testMethod2(){
    echo "SimpleExapmeAddons->testMethod2";
}

public function testMethod3(){
    echo "SimpleExapmeAddons->testMethod3";
}
public function testMethod4(){
    echo "SimpleExapmeAddons->testMethod4";
}
public function testMethod5(){
    echo "SimpleExapmeAddons->testMethod5";
}
```

以上简单插件的功能就定义完毕。可以进行调试和发布了

2. 以复杂插件的钩子实现为例。
   - 定义 hooks 集合类：在插件目录中建立 hooks 目录。然后在其中定义一个 TestHooks.class.php. 请注意文件名格式。
   - 实现 TestHooks.class.php


```php
<?php
class Test1Hooks extends Hooks
{
    public function home_index_middle_top()
    {
        echo "Test1Hooks";
    }
    public function home_index_left_top()
    {
        echo "Test1Hooks";
    }
    public function home_index_right_top()
    {
        echo "Test1Hooks";
    }

    public function home_account_tab($param){
        $param[0][]=array('act'=>'TestTab','name'=>'测试tab','param'=>array('addon'=>'NormalExample','hook'=>'testPage'));
    }

    public function testPage(){
    }

    public function adminTest1(){
        echo 123;
    }
    public function adminTest3($param){
        $result = &$param[0];
        $result['status'] = false;
        $result['info'] = 'test';
    }
}
```

在实现插件过程中的几个注意项

关于输入输出

- 输入：参数只有一个,在定义方法时的参数名字可以自己定义。推荐使用$param。所有钩子传入的参数都在该数组中
- 输出：简单的可以 echo，比如 ajax 操作。复杂的像 Action 一样可以 assign 也可以 display。

注意，display 的 html 文件固定在插件目录下的 html 目录中。display 方法必须指定 html 文件名(如：$this->display('test'))。如果 html 目录不存在，请自行创建。

关于插件中的路径问题

- $this->url 指通过 url 访问到该插件的目录。如在插件中使用自己要用到的一些图片或者样式。可以$this->assign('url_path',$this->url.'/html/');
- $this->path 指物理绝对地址。通常用来 require 插件内的辅助对象使用。
- 其他路径和 Thinksns 中常量保持一致

关于插件的请求问题

- 表单请求。插件中经常会用到 ajax 请求或者 post 请求。可以使用 Addons::createAddonUrl($addonsName,$hookName,$param)方法生成一个插件请求路径
- 页面请求。有两种方式。第一种。自行定义一个页面，这个页面是由各个公共部分组成。把希望为空的地方放一个 Addons::addonsHook($addonsName,$name,$param=array())方法产生的一插件中的制定方法--调用。另一种方式：直接请求 Addons::createAddonsShow($addonsName,$hookName,$param=array())方法.这个请求实际会生成一个页面。有系统公共头和公共底部的空页面。会调用$addonsName 中的 $hookName 方法。如果$hookName 方法是 display 一个页面。那么一个插件的请求页面也就实现了。

后台的开发

- 告知系统该插件使用了插件:
  在 XXXAddons.class.php 插件信息类中书写 adminMenu()方法。该方法没有参数。
  返回数组 array('方法名'=>'在后台显示时的 tab 名称。')如果是多个，就是多个 tab;
  方法名指的是简单钩子中的 XXXAddons.class.php 中的任意方法。复杂插件的 XXXHooks.class.php 中的任意方法。建议如果用复杂插件，单独用一个 AdminHooks.class.php。
- 插件管理页面中需要用到提交地址或者 ajax。使用 Addons::adminUrl("同 adminMenu 中的返回值的 key 一样，是一个方法名");
- 插件管理页中需要生成一个新的页面时，使用 Addons::adminPage(同 adminMenu 中的返回值的 key 一样，是一个方法名"));
- 所有的插件页面都还是放在插件的 html 目录中

其他接口

- 在插件中希望访问到某个制定插件的指定方法
  Addons::addonsHook($addonName,$methodName,$param=null);
- 在插件中需要产生一个 ajax 请求，并且该请求是请求到自身或者其他插件的实现中（非 admin），Addons::createAddonUrl($addonName,$methodName,$param=null);
- 官方发布版本的钩子不够用怎么办？在任意位置中调用
  Addons::hook($name,$param=array());清除缓存后，该钩子位生效

3.7 Widget 开发

命名规范

由于 Widget 为全局通用，可能在任意页面中被引用，所以必要的命名规范尤为重要。为避免 DOM ID 和 JS 的命名冲突，Widget 页面的所有节点 ID、JS 变量、JS 函数必须使用“_widget_widget_name_”前缀（如：_widget_medal_domid）。

开发流程

Widget 扩展用于在页面根据需要输出不同的内容，其定义位于**/addons/widgets/**目录。

Widget 类库必须继承 Widget 类，并且必须实现 render()方法，例如：

```
class ShowCommentWidget extends Widget{
    public function render($data){
        return '这是最新的'.$data['count'].'条评论信息';
    }
}
```

render()方法必须使用 return 返回要输出的字符串信息，而不是直接输出。Widget 也可以调用 Widget 类的 renderFile()方法，渲染模板后进行输出。例如：

```
class ShowCommentWidget extends Widget{
    public function render($data){
        $content = $this->renderFile('tpl.html',$data);
        return $content;
    }
}
```

异步加载

您只需组装好$_REQUEST 后，将 URL 引导至 **index.php?app=home&mod=Widget&act=renderWidget** 即可实现 Widget 的异步加载。其中$_REQUEST['name']为 Widget 名，$_REQUEST['param']为先 serialize 再 urlencode 的 Widget 参数。

4.附录

ThinkPHP 官方文档的附录（http://thinkphp.cn/Manual/218）对常量、配置、函数库和类库都有非常完备的说明，本附录仅说明 ThinkSNS2.5 特有的全局变量、常量、函数库和类库。

4.1 全局变量

$ts：存储全局信息的数组，包括站点信息、用户信息、当前节点信息、用户的应用信息、当前广告信息、页脚文章信息等。

代码中通过 `global $ts;` 声明即可使用，模板中可以直接通过 `{$ts['param']}` 调用。

4.2 常量

- **SITE_PATH**: 系统根目录
- **SITE_URL**: 站点根 URL
- **APPS_PATH**: **/apps/**目录
- **ADDON_PATH**: **/addons/**目录
- **UPLOAD_PATH**: **/data/upload/**目录，所有的上传文件都存放于此

4.3 函数库

系统函数库位于**/core/sociax/functions.php**和**/core/sociax/extend.php**两个文件，这里仅列举常用函数，全部函数和及参数说明请参阅文件注释。

function.php

- **model**: 实例化公共 model
- **service**: 实例化服务
- **widget**: 实例化 Widget

extends.php

- **canAccess**: 根据 access.inc.php 检查是否有权访问当前节点
- **convert_ip**: 获取给定 IP 的物理地址
- **desdecrypt**: DES 解密函数
- **desencrypt**: DES 加密函数
- **format**: 格式化微博,替换表情/@用户/话题
- **formatComment**: 格式化评论，替换表情和@用户
- **friendlyDate**: 友好的时间显示
- **get_client_ip**: 获取客户端 IP 地址
- **getAppAlias**: 根据应用名获取应用别名
- **getFollowState**: 获取关注状态
- **getFrom**: 获取微博来源
- **getLocation**: 根据给定的省市的代码获取实际地址
- **getOnlineUserCount**: 获取当前在线用户数(有效期 15 分钟)
- **getSex**: 根据 sexid 获取性别

- **getShortUrl**: 获取给定 URL 的短地址（依赖于短网址插件）
- **getUids**: 获取给定字符串中被@用户的 uid 数组
- **getUserEmail**: 获取给定用户的 Email
- **getUserFace**: 获取用户头像
- **isBlackList**: 是否为黑名单成员
- **isEmailAvailable**: 检查 Email 是否可用
- **isSubmitLocked**: 检查表单是否提交
- **isfavorited**: 检查给定用户是否收藏给定微博
- **isValidEmail**: 检查 Email 地址是否合法
- **lockSubmit**: 锁定表单提交（防止重复提交表单）
- **object_to_array**: 通过循环遍历将对象转换为数组
- **real_strip_tags**: 对 strip_tags 函数的扩展，可以过滤 object, param, embed 等来自编辑器的标签
- **setOnline**: 将给定用户设为在线
- **unlockSubmit**: 表单解锁

4.4 类库

系统类库包括公共 Model、公共服务、公共插件和 Widget，分别位于**/addons/models/**, **/addons/services/**, **/addons/plugins/**, **/addons/widgets/**目录
这里仅列举类库的概览，详细实现和使用方法请参阅具体代码的注释

服务

- **CommentService**: 评论服务
- **CreditService**: 积分服务
- **FeedService**: 动态服务
- **MailService**: 邮件服务
- **NotifyService**: 通知服务
- **ObjectCacheService**: 对象缓存服务
- **PassportService**: 通行证服务
- **ShortUrlService**: 短地址服务
- **SystemPopedomService**: 系统权限服务
- **SystemService**: 系统信息服务
- **UserRegisterService**: 用户注册服务
- **ValidationService**: 验证服务
- **XattachService**: 附件服务
- **XdataService**: 系统配置数据服务

公共 Model

- **AppModel**: 应用模型

- **AreaModel**: 地区模型
- **AttachModel**: 附件模型
- **ExpressionModel**: 表情模型
- **FriendModel**: 好友模型
- **GlobalCommentModel**: 全局评论模型
- **InviteModel**: 邀请模型
- **MedalModel**: 勋章模型
- **MessageModel**: 短消息模型
- **MyopModel**: 漫游应用模型
- **TemplateModel**: 模板模型
- **UserCountModel**: 用户统计模型
- **UserGroupModel**: 用户组模型
- **UserModel**: 用户模型
- **UserDataModel**: 用户统计数据模型（V2.5 新增）
- **XdataModel**: Key-Value 引擎模型

Plugin

- **Login**: 外部登录插件：包括新浪微博登录、QQ 登录等
- **Medal**: 勋章插件
- **NewFeatures**: 新功能上线引导插件
- **ShareOther**: 向站外分享微博插件
- **ShortUrl**: 短网址插件（包括 t.cn、bit.ly、goo.gl、本地网址等）
- **SpaceAppShow**: 在空间展示应用数据插件
- **SpaceFollow**: 在空间展示关注我的人插件
- **SquareAppShow**: 在广场展示应用数据插件
- **SyncGroupWeibo**: 微博同步发布到微群插件
- **UserSpaceCard**: 个人小名片插件
- **UserVerified**: 微博身份认证插件
- **Visitor**: 显示最近访客插件
- **WeiboType**: 发布微博类型插件（默认表情、话题，可通过插件增加图片、视频、照片、文档等）

Widget

- **CommentWidget**: 评论 Widget
- **FollowGroupWidget**: 关注群组 Widget
- **GroupWeiboWidget**: 群组微博 Widget
- **HotTopicWidget**: 热门话题 Widget
- **MedalWidget**: 勋章 Widget
- **RelatedUserWidget**: 可能感兴趣的人 Widget
- **SelectFriendsWidget**: 选择好友 Widget
- **SelectUserGroupWidget**: 选择用户组 Widget

- **`UploadAttachWidget`**: 上传附件 Widget
- **`VisitorWidget`**: 最忌来访 Widget
- **`VoteAddWidget`**: 添加投票 Widget
- **`WeiboWidget`**: 发布微博 Widget（亦即：分享 Widget）
- **`WeiboListWidget`**: 微博列表 Widget