

二次开发指南

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 ThinkSNS2.0，一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

序言.....	2
目录.....	3
1.简介.....	5
1.1 什么是 ThinkSNS2.0.....	5
完整的微博系统.....	5
内置接入多个应用平台，无限应用扩展.....	5
支持 WAP，提供手机客户端.....	5
多系统账号登录支持.....	5
集成社交媒体平台.....	5
插件、应用双重扩展机制.....	6
1.2 编写目的.....	6
2.架构设计.....	6
2.1 设计目标.....	6
2.2 核心架构及目录结构.....	7
概览.....	7
术语解释.....	7
微博.....	8
漫游平台.....	8
目录结构.....	8
2.3 应用架构及目录结构.....	9
3.开发指南.....	10
3.1 命名规范与编码规范.....	10
3.2 使用函数库、类库和 Widget.....	10
使用系统函数库.....	10
使用服务.....	10
使用公共 Model.....	10
使用 Widget.....	10
使用第三方类库.....	11
3.3 使用弹出窗、提示消息、编辑器.....	11
弹出窗.....	11
提示消息.....	12
编辑器.....	12
3.4 ThinkPHP 开发指南.....	13
3.5 应用开发.....	13
开发流程.....	13
开发需求和开发目标.....	13
创建数据表.....	14
创建应用目录.....	15
开始编程.....	16
增加积分消费功能.....	17
增加发通知功能.....	18

增加发送动态功能.....	18
增加微博分享功能.....	19
增加应用后台.....	20
制作安装/卸载包.....	21
调试.....	23
3.6 勋章开发.....	23
什么是勋章?	23
开发流程.....	23
初始化数据.....	23
继承实现 BaseMedal 抽象类.....	24
回调方法.....	25
安装、调试、运行.....	25
注意事项.....	25
3.7 Widget 开发.....	25
命名规范.....	25
开发流程.....	25
异步加载.....	26
3.8 模板开发.....	26
教你看每个前台页面所对应的模板文件.....	26
模板路径以及目录结构.....	26
基本的模板修改方法.....	27
4.附录.....	30
4.1 全局变量.....	30
4.2 常量.....	30
4.3 函数库.....	30
function.php.....	30
extends.php.....	30
4.4 类库.....	31
服务.....	31
公共 Model.....	32
Widget.....	32

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，ThinkSNS2.0 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，ThinkSNS2.0 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，ThinkSNS2.0 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

ThinkSNS2.0 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，方便用户的信息管理。

插件、应用双重扩展机制

ThinkSNS 从来都是二次开发者的利器，2.0 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 ThinkSNS2.0 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。

ThinkSNS2.0 内核使用优化的 ThinkPHP 框架，ThinkPHP 框架的绝大多数规范和参考对 ThinkSNS2.0 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

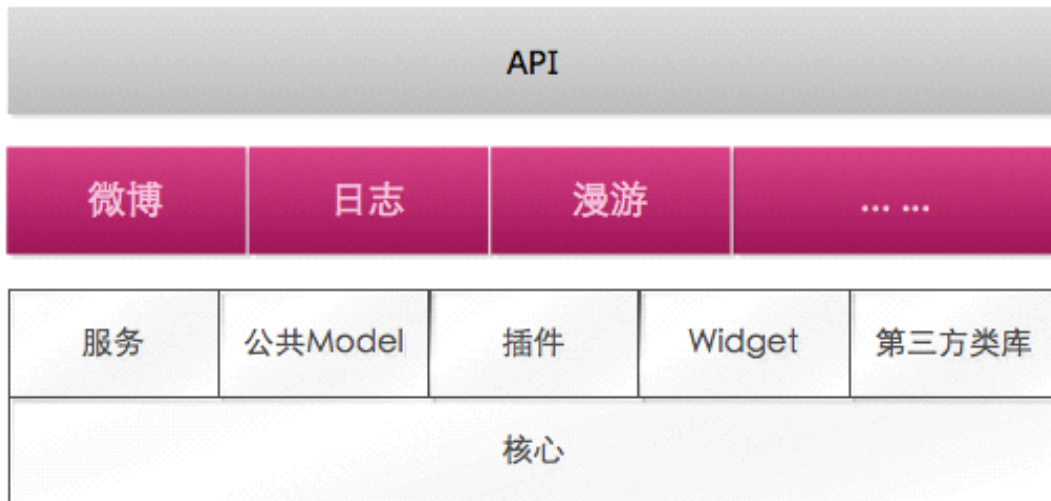


图 1 ThinkSNS2.0 核心结构图

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 ThinkSNS2.0 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

ThinkSNS2.0

└─ _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:

ThinkSNS2.0

└─ 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
ThinkSNS2.0 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>
```

```
</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')->isLoggedIn();`

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 ThinkSNS2.0 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。ThinkSNS2.0 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 应用开发

看了前面的介绍和说明,相信大家已经蠢蠢欲动。本章以开发礼物应用为例为大家展示开发一个应用的全过程。如果你是 ThinkPHP 开发高手,你会发现开发一个应用其实就是创建一个新的项目,这也是 ThinkSNS2.0 的魅力所在:应用独立化。

开发流程

ThinkSNS2.0 应用开发的一般流程:

1. 创建数据库和数据表(没有数据库操作可略过)
2. 项目命名并创建项目目录
3. 创建控制器类
4. 创建模型类
5. 创建模板文件
6. 运行和调试

开发需求和开发目标

我们的礼物应用需要实现以下的功能:

1. 用户可以给关注的人或粉丝发送礼物
2. 用户可以看到自己送出去的全部礼物
3. 用户可以看到别人送给自己的全部礼物
4. 用户可以回赠礼物给好友

5. 用户发送礼物时能同时发送通知信息
6. 用户发送礼物成功后可发送微博分享消息
7. 管理员可在后台增加, 修改, 删除礼物及礼物种类
8. 管理员可在后台配置礼物的积分消费种类: 积分? 经验?
9. 实现完善的积分消费功能, 如果用户积分不足则不能赠送

创建数据表

根据上面的需求分析可知, 我们需要设计三张数据表, 一个用来保存礼物的分类信息, 另一个用来保存礼物的信息, 还有一张用来保存用户之间的送礼记录。增加数据表请注意表的命名格式: 数据库表前缀+表名, 其中数据库表前缀在 `config.inc.php` 里的 `DB_PREFIX` 常量已经定义。

增加礼物分类表:

字段	类型	说明
id	int	
name	varchar	分类名
status	tinyint	是否启用 0 禁用 1 启用 (默认)
cTime	int	创建时间

增加两个默认分类: 热门礼物、最新上架

增加礼物信息表:

字段	类型	说明
id	int	
categoryId	int	分类 ID, 对应上面的分类表的 ID
name	varchar	礼物名
num	int	库存
price	int	价格
img	varchar	礼物图片
status	tinyint	是否启用 0 禁用 1 启用 (默认)
cTime	int	创建时间

在这里我们当然也要预先准备好部分礼物, 即初始化的礼物信息。

增加送礼记录表:

字段	类型	说明
id	int	
fromUserId	int	送礼人 ID
toUserId	int	送礼对象
giftPrice	int	价格
giftImg	varchar	礼物图片
sendInfo	text	附加信息
sendWay	tinyint	赠送的方式: 1 公开 2 私下 3 匿名
cTime	int	创建时间

到此, 数据表设计完毕。

在应用程序目录下的 `/gift/Appinfo/` 文件夹里的 `install.sql` 文件包含了上述建表和插入预定义数据的 sql 语句。

创建应用目录

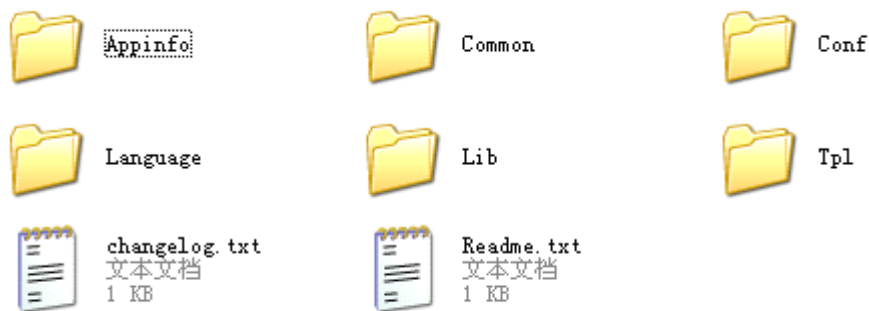


图 4 应用目录结构

如上图所示，我们先在 **/apps/** 目录下增加礼物目录 **gift**，然后在 **/apps/gift/** 目录下创建上图所示的目录（请注意大小写）。

- **应用程序目录 Lib:**

应用程序目录下存放的是 Action 控制器和 Model 模型的文件。

- **应用模板目录 Tpl:**

由于本次礼物应用只有一套模板，因此我们直接在 **Tpl/** 目录创建默认模板目录 **default/**，然后在 **default/** 目录下添加我们需要的 **Index** 模板和一个公共模板目录 **Public/**。

- **公共函数目录 Common:**

这里可以存放本次应用的公共函数，文件名为 **common.php**，该文件会在执行过程中自动加载，这些函数在 **Action/** 和 **Model/** 目下的文件里可以直接使用，无需再次引入文件。同时我们还可以直接使用 ThinkSNS2.0 里的公共函数，这些函数保存在 **/core/sociax/** 文件夹下的 **fuctions.php** 和 **extend.php** 里，它包括我们最常用的数据处理或者数据获得，如 **h()**、**t()** 过滤函数，**getUserName()** 函数（在模板文件的调用方法为：**{ \$userId|getUserName }**）、**getUserFace()** 函数和 **friendlyDate()** 函数（用法同 **getUserName()**）。以上的几个函数推荐大家使用。

- **应用配置目录 Conf:**

如果应用里有需要修改系统默认的常量或者在里面添加项目需要的一些配置参数，就需要在项目的 **Conf/** 目录下面，创建一个名称是 **config.php** 的配置文件，该文件也会自动加载。公共函数目录和应用配置目录非必需的目录，视应用开发决定是否需要。本次礼物开发中只用到公共函数文件夹。

- **语言目录 Language:**

用以保存应用的系统通知模板和动态模板。本次礼物开发中将用到系统通知模板。在 **Language/** 目录下创建中文语言包目录 **cn/**，然后添加通知模板文件 **notify.php**（动态是同样的道理，只是文件名为：**feed.php**）。

- **安装包目录 Appinfo:**

用于在 ThinkSNS 后台的应用安装/卸载操作，必须包含 **info.php** 文件（用以获取应用信息），如果含有 **install.php** 或 **uninstall.php** 文件，则会在应用安装、卸载时自动调用。

开始编程

- **公共函数层**

在 **Common/** 目录下, 新建 **common.php** 文件 (注意文件名的大小写), 写入所需函数, 如获取应用配置参数的函数:

```
function getConfig($key=NULL) {
    [.....]
}
```

详细代码请参见该文件。

这个文件里的函数在 **Action/**、**Model/** 和 **Tpl/** 目录下的文件都可以直接使用。

- **MODEL 层**

在 **Lib/Model/** 目录下, 新建三个文件 **GiftCategoryModel.class.php**, **GiftModel.class.php** 和 **UserGiftModel.class.php**, 分别创建代码如下:

```
class GiftCategoryModel extends Model{ }
class GiftModel extends Model{ }
class UserGiftModel extends Model{ }
```

通过以上步骤, 我们现在已将数据库表和模型类建立了关联关系。在项目的 **Action** 类和 **Model** 类中, 已可以直接对数据库进行相关操作。

有些人习惯在 **Model** 类里封装一些方法, 这样更符合 **MVC** 规范, 也有一些人习惯把方法都写在 **Action** 里, 方便程序的阅读, 这主要看个人习惯。本次开发将基本方法写进入 **Model** 层里。详细代码请参见相应文件。

- **ACTION 层**

在 **Lib/Action/** 目录下, 建立 **IndexAction.class.php**, 并创建代码如下:

```
class IndexAction extends Action{ }
```

详细代码参加该文件。

- **模板层**

我们在 **Tpl/default/** 目录下需要两个文件夹: **Index** 和 **Public**。**Index** 文件夹名是与 **IndexAction.class.php** 文件对应, 表示 **Index** 文件夹下面的文件都是 **IndexAction.class.php** 文件所需要的全部模板。应用的公共文件我们一般放到 **Tpl/default/Public/** 目录下, 比如礼物应用的礼物图片、JS 文件、CSS 文件、应用头文件 **_mainNav.html** 等。当然, 这些目录的设置完全由开发者自己决定, 只要保证调用文件的路径正确就可以了。

我们先来看看 ThinkSNS2.0 模板文件的一般结构, 开发人员引入本应用自己的样式文件和 JS 文件后可直接在画布层里增加相应的显示代码即可:

```
<!-- 引入系统头文件-->
<include file="__THEME__/_header" />

<!-- 引入样式文件区-->
<!-- 引入 JS 文件区-->

<!-- 引入应用头文件-->
<include file="../../Public/_mainNav" />

<!-- 内容 begin -->
```



```

<div class="content">
  <div class="main no_1">
    <div class="mainbox">
      <!-- 画布 begin -->
      <div class="mainbox_C no_r">
        [.....]
      </div>
      <!-- 画布 end -->
      <div class="c"></div><!-- 分隔层-->
    </div>
  </div>
</div>
<!-- 内容 end -->

```

```
<!--引入系统尾文件-->
```

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

详细内容请参阅具体代码。下面介绍模板文件中几处有学习价值的代码。

1. ThinkSNS2.0 全面使用 jQuery 技术, jQuery 是一个不错的轻量级的 JS 框架, 能帮助我们快速的开发 JS 应用, 并在一定程度上改变了我们写 JavaScript 代码的习惯。ThinkSNS2.0 在头文件里已经引入 jQuery 库, 开发人员不必重复引入。
2. **Tp1/default/Index/**目录下的 **index.html** 文件, 这是礼物中心的模板, 里面选择发送好友的功能用到好友选择 widget, 只需要在模板里增加 `{:W("SelectFriend")}`, 然后用 `$_POST['fri_ids']` 就可以获得选择的好友 ID。
3. 因为我们是使用 `findPage()` 方式获取收到和送出的礼物列表, 故返回的数据已经有分页数了, 只要在模板合适的地方增加如下的代码即可:

```
<div id="Pagination" class="pagination">{$gifts.html}</div>
```

增加积分消费功能

上面的程序基本完成了礼物的全部操作, 接下来我们要增加积分消费功能。至于礼物消费的积分类型, 是积分或者还是经验, 可以在后台设置, 详细请看后台程序说明。首先在发送礼物页面增加显示当前用户所拥有多少积分的功能, 我们只要在

IndexAction.class.php文件的 `index()` 方法里增加以下程序即可:

```
//获取当前用户的积分
```

```
$money = X('Credit')->getUserCredit($this->mid);
```

```
$moneyType = getConfig('credit');
```

```
$this->assign('money',$money[$moneyType]);
```

在发送礼物的模板里增加以下一行代码:

```
我目前拥有的 {$money.alias}是: {$money.credit}
```

然后在发送礼物的函数里 (**UserGiftModel.class.php** 的 `sendGift()` 方法) 增加扣除所选礼物的相应积分程序:

```
//扣除相应积分
```

```
$giftPrice = intval($giftInfo['price']);
```

```
$prices = $userNum*$giftPrice;
```

```

$moneyType = getConfig('credit');
//积分操作
$setCredit = X('Credit');
//检测积分是否足够
$userCredit = $setCredit->getUserCredit($fromUid);
if($userCredit[$moneyType]['credit']<$prices){
    return $userCredit[$moneyType]['alias'].'不足, 赠送失败~';
}
$setCredit->setUserCredit($fromUid, array($moneyType=>$prices), -1);

```

这里调用了公共类库 **services** 中的积分操作类, 即 `X('Credit')`。

到这里积分消费功能增加完毕。

增加发通知功能

首先需要增加礼物的通知模板。在 **Language/** 目录下创建一个文件夹 **cn**, 再在 **cn** 文件夹下创建通知模板文件 **notify.php**, 代码如下:

```

<?php
return array(
    'gift_send' => array(
        'title' => '{actor}给您送了一个礼物',
        'body' => $img.<br />'.$sendback.<br />
        <div class="quote">
        <p><span class="quoteR">'.'$content.'</span></p>
        </div><br />
        <a href="'.U('gift/Index/receivebox').'" target="_blank">
        去看看
        </a>',
    ),
);
?>

```

在发送礼物成功后增加以下程序:

```

//给接收人发送通知
$this->__doNotify($toUser,$sendInfo,$giftInfo,$fromUid,$appId);

```

`__doNotify()` 方法的实现请参阅具体代码。

这里调用了公共类库 **services** 中的系统通知类, 即 `X('Notify')`。

到这里发通知功能完成了。

增加发送动态功能

发送动态的机制与发送通知的机制完全相同, 区别只有如下两点:

- 通知使用 **/Language/cn/feed.php**, 而动态使用 **/Language/cn/notify.php**
- 通知服务使用 `X('Notify')` 或 `service('Notify')`, 而动态使用 `X('Feed')` 或 `service('Feed')`

由于礼物应用没有用到动态，这里也不再赘述。

增加微博分享功能

首先我们需要在“管理后台 - 内容 - 模板管理”增加一个礼物赠送的模板：

名称：	<input type="text" value="gift_send_weibo"/>	*
	使用英文，保证唯一性。建议：“类型_动作”，如“blog_add”或“credit_blog_add”	
别名：	<input type="text" value="礼物赠送"/>	
	简单达意，如“发表博客”	
标题模板：	<input type="text"/>	
	使用“{”和“}”包含变量名，如“{actor}做了{action}”	
内容模板：	<input type="text" value="我送给 {user} 一份礼物：【{title}】{content} 参与送礼 {url}"/>	
	使用“{”和“}”包含变量名，如“{actor}做了{action}增加了{volume}个{credit_type}”	
语言：	<input type="text" value="zh"/>	*
	与全局语言包一致，如“en”、“zh”等	
模板类型：	<input type="text" value="gift"/>	
	如“blog”、“credit”等	
模板类型2：	<input type="text" value="weibo"/>	
	备用类型，可留空。如“credit_blog”等	
是否默认缓存：	<input type="radio"/> 是 <input checked="" type="radio"/> 否	
	是否使用默认的模板缓存系统	
<input type="button" value="提交"/>		

在组装通知数据前，初始化一个获赠对象变量：

```
// 赠送的对象名称 用于公开赠送微博
```

```
$toUserName = NULL;
```

在组装公开赠送的通知信息中添加如下代码，用以@ 到各个获赠用户：

```
// 赠送对象名称
```

```
$toUserName .= '@'.getUser_name($fid).' ';
```

在通知信息组装结束后，添加如下代码，将微博信息暂存于SESSION中：

```
// 公开则发微薄
```

```
if($toUserName) {
    $_SESSION['gift_send_weibo'] = urlencode ( serialize ( array(
        'user'      => $toUserName,
        'title'     => $giftInfo['name'],
        'content'   => $data['content'],
        'url'       => U('gift/Index/index', array(
            'uid'      => $fid,
            'type'     => 1,
            'type_data'=>realityImageURL($giftInfo['img']))),
    ));
}
```

```
}
```

在**Lib/Index/ IndexAction.class.php**文件里的**sendbox()**函数内添加如下代码:

```
//判断是否有公开赠送信息, 存在, 则赋值给模板, 用于发微薄
if(isset($_SESSION['gift_send_weibo'])&&!empty($_SESSION['gift_send_weibo'])) {
    $this->assign('tpl_data', $_SESSION['gift_send_weibo']);
    unset($_SESSION['gift_send_weibo']);
}
```

在**Tpl/default/Index/**目录下送出的礼物**sendbox.html**模板文件里, 添加WeiboWidget引用(只有发送微博的信息存在时才加载), 如下:

```
<if condition="$tpl_data">
{:W('Weibo',array(
    'tpl_name'=>'gift_send_weibo',
    'button_title'=>'分享',
))}
<script>_widget_weibo_start(' ', '{$tpl_data}');</script>
</if>
```

到这里公开赠送发微博功能完成了。

增加应用后台

ThinkSNS2.0 为应用增加管理后台的方法很简单, 直接在应用的**Action/**目录下增加**AdminAction.class.php**文件, 然后导入并继承**AdministratorAction**类即可完成权限管理:

```
<?php
//引入后台管理类
import('admin.Action.AdministratorAction');
class AdminAction extends AdministratorAction {
    function _initialize(){
        // 管理权限判定
        parent::_initialize();
        [.....]
    }
    [.....]
}
?>
```

注意: 在_initialize函数里, 首先要执行parent::_initialize(), 进行管理权限判定。更多详细代码请见具体文件。

接着我们增加对应的管理页面的模板, **Tpl/default/**目录下增加**Admin/**目录, 然后在该目录下增加**_header.html**, **_footer.html**, **_tab.html**, **index.html**, **giftlist.html**, **category.html**, **edit_gift_tab.html**, **edit_category_tab.html**几个模板文件。模板开发方式同前台的模板一样, 在此不再重复。

制作安装/卸载包

在应用的目录下增加安装信息目录，命名为**Appinfo**，然后在该目录下增加应用图标（**ico_app.gif**、**ico_app_large.gif**）、应用配置文件（**info.php**）、应用安装文件（**install.php**）、应用数据库文件（**install.sql**）和应用卸载文件（**uninstall.php**）。

- **配置文件内容：**

```
<?php
if (!defined('SITE_PATH')) exit();

return array(
    // 应用名称 [必填]
    'NAME'          => '礼物',
    // 应用简介 [必填]
    'DESCRIPTION'   => '礼物赠送',
    // 托管类型 [必填] (0:本地应用, 1:远程应用)
    'HOST_TYPE'     => '0',
    // 前台入口 [必填] (格式: Action/act)
    'APP_ENTRY'     => 'Index/index',
    // 应用图标 [必填]
    'ICON_URL'      => SITE_URL . '/apps/gift/Appinfo/ico_app.gif',
    // 应用图标 [必填]
    'LARGE_ICON_URL' => SITE_URL . '/apps/gift/Appinfo/ico_app_large.gif',
    // 后台入口 [选填]
    'ADMIN_ENTRY'   => 'Admin/index',
    // 统计入口 [选填] (格式: Model/method)
    'STATISTICS_ENTRY' => 'GiftStatistics/statistics',
    [.....]
);
?>
```

注意：其中的 `SITE_URL` 是站点根目录网址，如果直接使用它本身的话程序会自动解释成绝对网址，还有一个好处就是网站转移的时候这些信息都不需要修改，程序会自动解释出新的网址赋值给它。当然，你也可以使用绝对网址。

- **安装文件内容：**

数据库操作的语句均统一放在数据库文件 **install.sql** 中，**install.php** 的功能就是执行该文件：

```
<?php
if (!defined('SITE_PATH')) exit();

header('Content-Type: text/html; charset=utf-8');

$sql_file = APPS_PATH . '/gift/Appinfo/install.sql';
//执行sql文件
$res = M('')->executeSqlFile($sql_file);
if(!empty($res)){//错误
```

```

        echo $res['error_code'];
        echo '<br />';
        echo $res['error_sql'];
        //清除已导入的数据
        include_once(APPS_PATH.'/gift/Appinfo/uninstall.php');
        exit;
    }
?>

```

- **数据库文件内容:**

首先, 写入该应用三张数据表的添加语句; 然后, 再加入应用系统配置信息和微博模板的插入语句, 若有设定积分规则的应用, 也加上积分规则的插入语句, 可以参见日志、相册等应用的数据库文件, 代码如下:

```
SET FOREIGN_KEY_CHECKS=0;
```

[……三张数据表及其预设信息的sql语句]

```

#添加ts_system_data数据
REPLACE INTO `ts_system_data` (`uid`,`list`,`key`,`value`,`mtime`)
VALUES
    (0, 'gift', 'credit', 's:5:"score\";', '2010-12-24 11:22:17');

#模板数据
DELETE FROM `ts_template` WHERE `name` = 'gift_send_weibo';
INSERT INTO `ts_template` (`name`,`alias`,`title`,`body`,`lang`,`type`,`
`type2`,`is_cache`,`ctime`)
VALUES
    ('gift_send_weibo', '礼物赠送', '', '我送给{user} 一份礼物:【{title}】
{content} 参与送礼{url}', 'zh', 'gift', 'weibo', 0, 1290417734);

```

这里可以直接使用由工具导出的 sql 文件和语句。

- **卸载文件内容:**

删除应用的数据表、应用系统配置信息和微博模板。若应用还设置了积分规则, 则也要同时删除其积分规则, 可以参见日志、相册等应用的卸载文件。

```

<?php
if (!defined('SITE_PATH')) exit();

$db_prefix = C('DB_PREFIX');

$sql = array(
    // gift数据
    "DROP TABLE IF EXISTS `${db_prefix}gift`;",
    "DROP TABLE IF EXISTS `${db_prefix}gift_category`;",
    "DROP TABLE IF EXISTS `${db_prefix}gift_user`;",
    // ts_system_data数据
    "DELETE FROM `${db_prefix}system_data` WHERE `list` = 'gift'",

```

```
// 模板数据
"DELETE FROM `{ $db_prefix }template` WHERE `name` = 'gift_send_weibo';",
);

foreach ($sql as $v)
    M('')->execute($v);
?>
```

至此，安装\卸载包制作完成。

登录后台后进入“应用->应用管理->添加应用”点击“安装”应用就可以了。然后更新一下系统缓存就可以在前台的应用列表中看到我们的礼物应用了。

调试

要调试我们的有程序，有很多种方法，在此我们推荐几种常用的调试方法。

- 用ThinkPHP 自带的`dump()`方法把过程变量打印出来，看是否与预想的结果一样
- 使用ThinkPHP 的`getLastSql()`方法取得最后的SQL语句并用`dump()`打印出来，检查SQL语句是否正确
- 在**Conf/config.php**文件里增加`DEBUG_MODE`常量并赋值为`true`，即把ThinkPHP 的调试模式打开（注：修改后需要清理缓存）

3.6 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人员手上

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

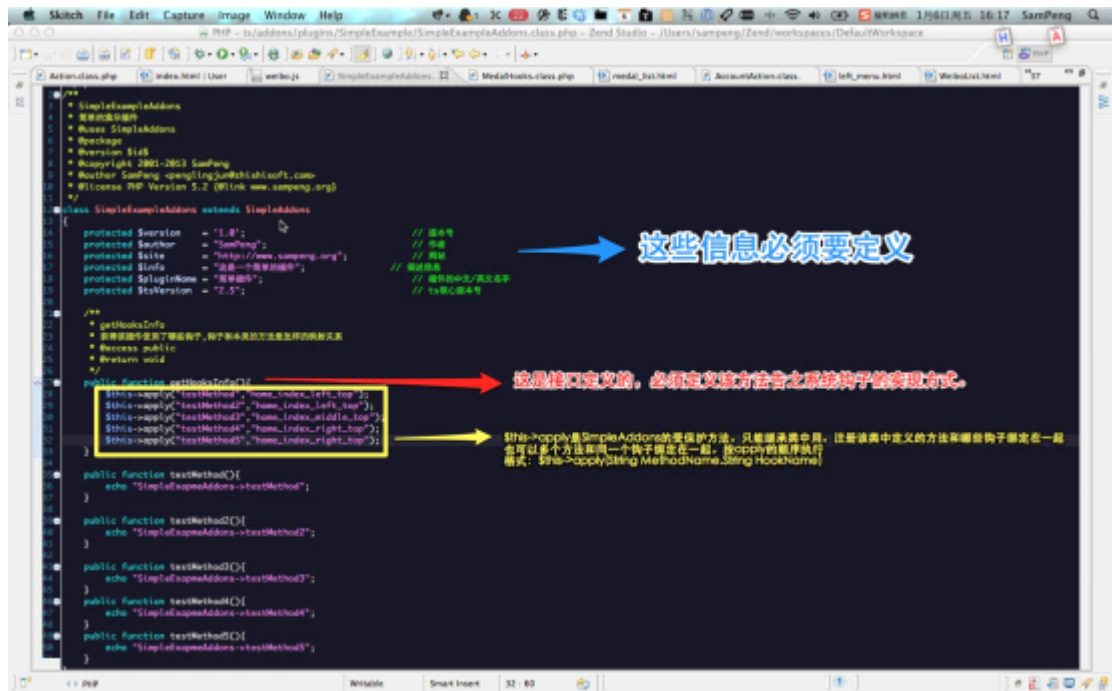
- NormalAddons: 复杂的插件

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。

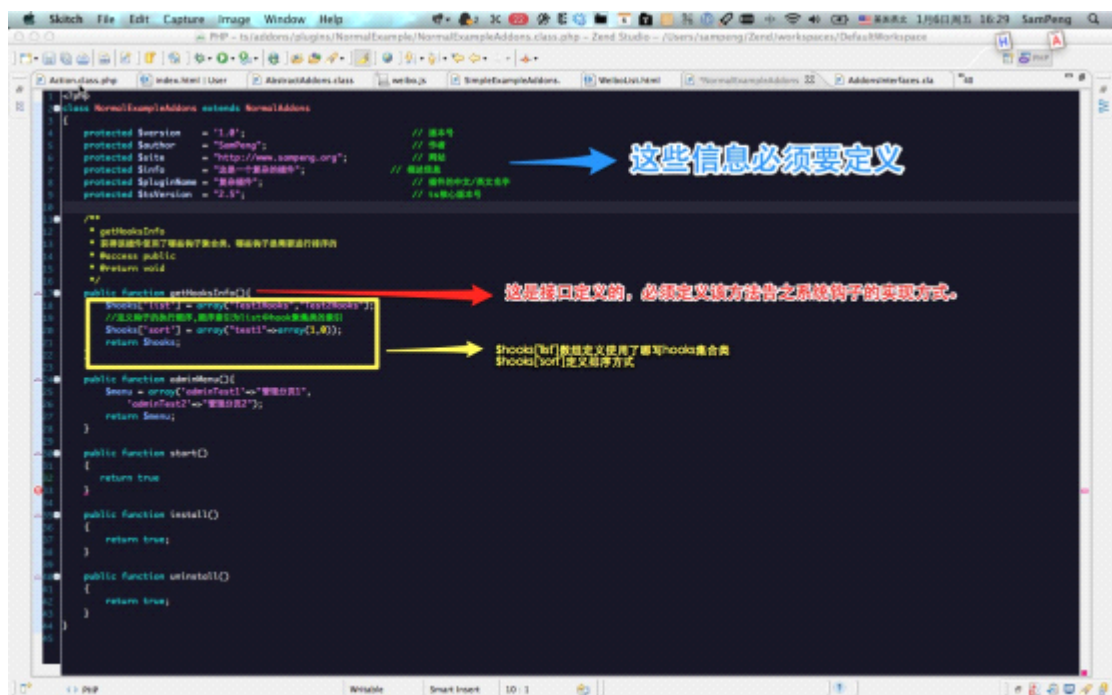
注意: 该名字一定要是 插件名字+Addons+.class.php 这样的标准格式

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：

```
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
class TestHooks extends Hooks
{
    public function home_index_middle_top()
    {
        echo "TestHooks";
    }
    public function home_index_left_top()
    {
        echo "TestHooks";
    }
    public function home_index_right_top()
    {
        echo "TestHooks";
    }

    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';
    }
}

```

实现的抽象类

钩子名称和类方法名相同

钩子要执行的业务逻辑

在实现插件过程中的几个注意事项

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。

安装、调试、运行

勋章的安装方法为：管理后台 - 扩展 - 插件 - 勋章 - 安装 XX。

勋章有四个展示位置：首页左侧、个人主页右侧、设置-勋章管理-我的勋章、设置-勋章管理-勋章设置，请同时测试此 4 个位置。

注意事项

- 用户浏览首页和个人主页都会展示勋章，所以务必保证勋章逻辑的效率
- 用户获得勋章后增加积分：`X('Credit')->setUserCredit($uid,'add_medal');` 用户丢失勋章后减少积分：`X('Credit')->setUserCredit($uid,'delete_medal');`

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。

3.8 模板开发

本节主要包括：

- 教你看每个前台页面所对应的模板文件
- 模板路径以及目录结构
- 基本的模板修改方法
- 新风格包机制说明

教你看每个前台页面所对应的模板文件

以登录页和首页为例，其他页面以此类推：(这里用颜色区分每级目录)

登陆页路径：`http://t.thinksns.com/index.php?app=home&mod=Public&act=login`

登陆页目录：`/apps/home/Tpl/default/Public/login.html`

首页路径：`http://t.thinksns.com/index.php?app=home&mod=User&act=index`

首页目录：`/apps/home/Tpl/default/User/index.html`

注：某些页面会使用 `include` 的方式载入，这时您只能通过代码找到相应模板文件了。

模板路径以及目录结构

样式模板存放路径为 `/public/themes/weibo/`，其中的 `classic` 是模板包，其中存放的是默认模板，以下所有的说明都是针对默认模板。

模板修改过程中需要特别注意：

- 应用有自己的样式文件。如日志：`/apps/blog/Tpl/default/Public/blog.css`
- 修改前请做备份
- 修改站点头部或个人首页的左侧后，还需要修改漫游下的站点头部和左侧（位于 `/apps/myop/themes/weibo/` 目录下的 `header.html` 和 `apps.html` 文件）
- **新增：从 ThinkSNS V2.5 起，风格包下新增应用模版目录，可以在风格包中重新定义应用模版。**

如果风格包中存在应用模版文件，则以风格包中为准，如果不存在仍然去应用中寻找如：

登陆页路径：`http://t.thinksns.com/index.php?app=home&mod=Public&act=login`

原登陆页模版：`/apps/home/Tpl/default/Public/login.html`

如存在新文件：`/public/themes/weibo/apps/home/Public/login.html`
则以新文件为准，如果不存在以原文件为准。

目录结构:

ThinkSNS2.0	
└─ public	
└─ themes	
└─ classic	
└─ apps	----- 应用模版风格包
└─ home	----- home 应用
└─ Index	----- 应用下的 Index 模块
└─ weibo	----- weibo 应用
└─ images	----- 图片
└─ apps.html	----- 首页左侧
└─ header.html	----- 站点头部 (修改后需清理缓存)
└─ footer.html	----- 站点底部 (修改后需清理缓存)
└─ layout.css	----- 布局样式
└─ public.css	----- 公共样式

基本的模板修改方法

推荐的软件 and 工具:

- CSS 样式修改: Dreamweaver / EditPlus / Notepad++ 等 (新手推荐使用 Dreamweaver)
- 截图取色: FSCapture
- Firefox 插件: Firebug (使用方法: <http://getfirebug.com/wiki/index.php/FAQ>)

模板制作方法:

- 加入背景图 (**layout.css**)

```
.body_bg {  
    background:url("http://ww2.sinaimg.cn/large/3eb8a9d2jw6dbums1mdboj.jpg") no-repeat fixed center 0 #564E4B;  
}
```

如果不想让背景图片固定把 `fixed` 去掉就 OK 了。

- 去掉顶部蓝色背景图 (**layout.css**)

```
.header_bg {z-index:9;}
```

效果如下:



- 去掉底部背景 (**layout.css**)
`.footer_bg {clear:both;}`
- 修改首页左侧背景为透明 (**layout.css**)
`.content {
 background:url("http://timg.sjs.sinajs.cn/t3/skin/skin_diy/images/black.png") repeat-y scroll 0 0 transparent;
 _background:none;
 _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://timg.sjs.sinajs.cn/t3/skin/skin_diy/images/black.png',
 sizingMethod='crop');
 clear:both;
 margin:15px auto;
 min-height:580px;
 text-align:left;
}`

上面 `.content` 样式的背景是 PNG 透明图片, IE6 默认是不认的, 后面的一长串 `_filter` 是写给 IE6 的 hack, 这里的图片地址一定要写绝对地址, 所以修改是需要更换两个图片路径。

更换 Logo 是同样的方法, 需要更换如下两个标红的地址:

```
.header .logo {
    float:left;
    margin:15px 0 0;
    width:200px;
    background:url('images/logo.png') no-repeat;
    _background:none;
    _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='http://t.
thinksns.com/public/themes/classic/images/logo.png', sizingMethod='crop');
}
```


ThinkSNS 2.0 智士软件（北京）有限公司

效果如下：



- 修改首页左侧文字的样式 (**layout.css**)

```
.user_app{color:#eee}
.user_app a {color:#eee;}
```

效果如下：



现在一个简单的模板就制作完成啦，如果您懂一些 html 和 CSS 的基础知识，模板制作还是非常简单的！

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 ThinkSNS2.0 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
- **SystemPoppedomService**: 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