ThinkCMF

目录

ThinkCMF文档

ThinkCMF是一款基于ThinkPHP+MySQL开发的中文内容管理框架。ThinkCMF提出灵活的应用机制，框架自身提供基础的管理功能，而开发者可以根据自身的需求以应用的形式进行扩展。每个应用都能独立的完成自己的任务，也可通过系统调用其他应用进行协同工作。在这种运行机制下，开发商城应用的用户无需关心开发SNS应用时如何工作的，但他们之间又可通过系统本身进行协调，大大的降低了开发成本和沟通成本。

此项目用于编写ThinkCMF完全开发手册，我们希望通过git,让更多对CMF感兴趣的朋友参与开发文档的完善。本文档使用MD格式文档编写，方便生成pdf,epub,mobi及html的发行版。我们会定期生成相应的格式。

在阅读本手册前请先仔细阅读ThinkPHP3.2.3完全开发手册
http://www.kancloud.cn/manual/thinkphp/1678

在线阅读：ThinkCMF文档

参加步骤

1. 在GitHub上 `fork` 到自己的仓库，如 `your_username/cmfx_doc` ，然后 `clone` 到本地，并设置用户信息。

```
$ git clone git@github.com:your_username/cmfx_doc.git
$ cd cmfx_doc
$ git config user.name "yourname"
$ git config user.email "your email"
```

- 修改代码后提交，并推送到自己的仓库。

```
$ #do some change on the content
$ git commit -am "Fix issue #1: change helo to hello"
$ git push
```

- 在 GitHub 网站上提交 pull request。
- 定期使用项目仓库内容更新自己仓库内容。

```
$ git remote add upstream https://github.com/your_username/cmfx_doc

$ git fetch upstream

$ git checkout master
```

```
$ git rebase upstream/master

$ git push -f origin master
```

序言

版权声明

捐赠我们

ThinkCMF一直秉承ThinkPHP大道至简的理念，坚持做最简约的ThinkPHP开源软件！
您的每一份帮助都将支持ThinkCMF做的更好，走的更远！
我们一直在坚持不懈地努力，并坚持让ThinkCMF完全开源免费，您的帮助将使我们更有动力和信心^_^！

支付宝捐赠：(用手机支付宝扫描二维码支付)ThinkCMF支付宝捐赠二维码

您的每一份捐赠将用来：

1. 深入ThinkCMF核心的开发
2. 做丰富的应用

3. 设计更爽的用户界面
4. 吸引更多的模板开发者和应用开发者
5. 奖励更多优秀贡献者
6. 点站内自己有兴趣的广告，也是对我们的帮助哟！~~

关于ThinkCMF

ThinkCMF是一款基于ThinkPHP+MySQL开发的中文内容管理框架。ThinkCMF提出灵活的应用机制，框架自身提供基础的管理功能，而开发者可以根据自身的需求以应用的形式进行扩展。每个应用都能独立的完成自己的任务，也可通过系统调用其他应用进行协同工作。在这种运行机制下，开发商场应用的用户无需关心开发SNS应用时如何工作的，但他们之间又可通过系统本身进行协调，大大的降低了开发成本和沟通成本。

MVC分层模式

使用MVC应用程序被分成三个核心部件：模型（M）、视图（V）、控制器（C），他不是一个新的概念，只是ThinkCMF将其发挥到了极致。

用户管理

ThinkCMF内置了灵活的用户管理方式，并可直接与第三方站点进行互联互通，如果你愿意甚至可以对单个用户或群体用户的行为进行记录及分享，为您的运营决策提供有效参考数据。

云端部署

通过驱动的方式可以轻松支持云平台的部署，让你的网站无缝迁移，内置已经支持SAE、BAE，正式版将对云端部署进行进一步优化。

安全策略

提供的稳健的安全策略，包括备份恢复，容错，防治恶意攻击登陆，网页防篡改等多项安全管理功能，保证系统安全，可靠，稳定的运行。

应用模块化

提出全新的应用模式进行扩展，不管是你开发一个小功能还是一个全新的站点，在ThinkCMF中你只是增加了一个APP，每个独立运行互不影响，便于灵活扩展和二次开发。

免费开源

代码遵循Apache2开源协议，免费使用，对商业用户也无任何限制

README

ThinkCMF是一款基于PHP+MYSQL开发的中文内容管理框架。ThinkCMF提出灵活的应用机制，框架自身提供基础的管理功能，而开发者可以根据自身的需求以应用的形式进行扩展。每个应用都能独立的完成自己的任务，也可通过系统调用其他应用进行协同工作。在这种运行机制下，开发商场应用的用户无需关心开发SNS应用时如何工作的，但他们之间又可通过系统本身进行协调，大大的降低了开发成本和沟通成本。

官网:http://www.thinkcmf.com

文档:http:\/\/www.thinkcmf.com\/document

更新日志

X2.2.0

[Core]

- 增加wind.js可以自定义重写js，css的版本
- 增加$_GET统一urldecode
- 增加cdn支持
- 增加微信浏览器判断
- 增加验证码生成安全性
- 增加 `__WEB_ROOT__` 模板常量
- 增加跨主题调用模板功能
- 增加include,extend,block,layout的支持
- 统一修复session用法问题
- 统一分页变量名为$page
- 统一修复系统所有变量未定义错误
- 优化前台手机模式下的分页
- 优化分页类
- 升级ueditor
- 修复编辑器不能插入动态地图问题
- 默认取消后台多语言功能

[Admin]

- 增加后台管理员搜索功能

- 增加友情链接图标上传
- 增加列表批量删除提示
- 增加邮件发送测试
- 优化发件箱连接方式用下拉列表选择
- 优化后台登录用户名cookie保存30天
- 优化已经登录直接跳转到后台首页
- 修复后台导航添加时卡死问题
- 修复插件更新配置时，配置文件状态不更新问题
- 修复后台留言列表留言时间错误

[Asset]

- 增加七牛cdn整体解决方案
- 增加统一上传限制,根据上传文件类型设置大小限制
- 增加上传文件名以应用名为前缀
- 优化文件上传统一使用plupload上传控件
- 优化七牛路径前缀
- 修复七牛上传bad token
- 修复编辑器的附件不能上传歌曲,歌词等文件类型
- 修复文章内容里图片上传时返回域名问题
- 修复七牛华北分区不能上传

[User]

- 增加后台用户搜索功能
- 增加前台未登录redirect

[Portal]

- 增加后台文章批量复制功能
- 增加面包屑功能
- 增加文章可以自己定义模板
- 更改后台文章列表，以posts表为主表显示
- 更改article控制器参数(cid,id),id是posts表的主键id,cid是分类id

- 修复文章编辑界面没有取消审核功能
- 修复禁用的文章，在上一篇下一篇里还是会出现

[Install]

- 增加安装程序优化验证必须模块
- 安装程序优化，再次安装时清除data\/conf\/config.php文件

X2.1.0

- 修复前台导航缓存问题
- 修复个人中心mysql5.7下保存失败
- 文章评论插件化
- 增加系统评论插件
- 修复simplebootx模板config文件变量不对应
- 优化用户激活流程，防止已激活用户和被禁用用户两次发送激活邮件
- 更正数据库前台导航分类active注释
- 修复手机模板开启时跳转页面模板路径判断错误
- 修复重置密码后，重置密码链接仍可打开问题
- 增加后台评论管理查看原文功能
- 增强后台登录接口安全性
- 修复前台分页当前页选中问题
- 增加全局路由,强化URL美化功能
- 修复导航添加时导航分类选择问题

祝新年愉快，合家欢乐！

X2.0.0

- 更改HomeBaseController.class.php文件名为HomebaseController.class.php
- 移动Common里的Portal model到Portal下；
- 规范前后台模板目录，themes ,admin\/themes；
- 注意以上升级，有助于您升级到最新版本
- 增加对PHP7的支持

- 增强验证码易识别性；
- 增加后台从菜单栏点击立即刷新选项卡功能；
- 增加前台模板多语言，插件多语言；
- 增加后台多语言基础功能，语言包稍后升级完善；
- 优化后台模板文件；
- 优化后台登录界面；
- 优化前台登录、注册、找回密码界面；
- 优化common.js,frontend.js,规范一系列js-xxx-xxx命名和功能实现；
- 优化文章编辑页布局和css；
- 修复模板中U方法大小写错误；
- 修复ip获取，防止代理访问；
- 修复success,error跳转页无手机模板问题；
- 升级百度编辑器；
- 去除前台各处同意网站条款；
- 去除文章访问次数统计的ip限制；

X1.6.1

- 修复登录时仍然可以打开登录和注册界面
- 修复后台文章分类列表，点添加子类到添加分类界面父级分类选择错误
- 修复后台文章分类模板修复后不更新问题
- 修复编辑器里图片上传， 在文件存储选择七牛时，图片title,alt属性不对
- 增加对php格式模板文件的支持
- 优化sp_get_menu方法,id为空时，默认为主菜单
- 修复后台模板缺失<\/head>问题
- 修复Portal应用下文章，页面，和分类不存在时，无404状态码
- 增加对模板继承标签tc_extend的支持
- 优化会员的拉黑起用功能，改为实时验证
- 增加url模式更改后出现不能访问问题的提示和解决方法
- 修复文章，幻灯片添加重复提交问题

X1.6.0

- 增加静态缓存
- 增加form提交状态判断，防止连续提交
- 增加后台被禁用角色和用户登录提示
- 增加文章可以添加到多个分类
- 增加管理员停用启用功能
- 增加文章来源版权申明
- 增加文章内容页面二维码
- 增加验证码统一判断方法
- 优化验证码自动刷新
- 优化后台样式
- 修复邮件配置更新时，不会立刻更新问题
- 修复后台手动新加的菜单时没有同步到auth_rule表
- 修复角色禁止后登陆报错
- 修复角色无法删除问题
- 修复后台视频上传出错！请注意上传大小限制,php.ini post_max_size,upload_max_filesize
- 修复后台菜单过多不显示
- 修复后台管理登录可能会被暴力破解
- 修复后台菜单列表层级问题
- 修复启用后被删除的插件执行报错

X1.5.0

- 增加插件机制
- 增加编辑器附件上传功能
- 核心升级至thinkphp 3.2.3，必须开启php_pdo_mysql扩展
- 优化系统权限管理，增加auth+rbac混合认证模式
- 增加文件存储扩展支持，默认支持本地和七牛云存储
- 增加手机模板支持

- 增加手机模板侦测后台开启关闭功能，默认关闭手机模板侦测
- 增加MUI手机开发框架
- 增加评论时间间隔设置
- 增加视频插入
- 增加去除模板文件里面的html空格与换行
- 增加后台管理员列表分页
- 增加文章页上一篇、下一篇功能
- 优化菜单管理方式，采用增加文件方式菜单管理，方便程序升级
- 优化导航鼠标划过下拉菜单
- 优化管理员信息编辑,增加字段过滤
- 优化非后台入口登录跳转到首页
- 修复simplebootx搜索链接错误
- 修复ucenter各种问题
- 修复后台邮件发件人无法设置
- 修复入口文件SITE_PATH常量部分服务器异常
- 修复sae头像裁剪
- 修复分类编辑时层级出错
- 修复备份还原数据为空
- 统一所有模块模板路径分割符为V
- 删除thinkphp Vendor目录第三方类库
- 移除SendMail方法
- 替换scandir方法为sp_scan_dir
- 统一SAE判断方法
- 统一ajaxReturn为thinkphp 3.2.*以后用法，如果一直用thinkphp 3.0以前的用法，扩展时注意用sp_ajax_return()做一下升级

注：后台模板分割符已经统一为V，原来的类似Admin\Main.index.html文件已经改为Admin\Main\index.html请后台开发时注意

ThinkCMF全体贡献者祝大家2015年大吉大利，开心幸福！《给你一个吻》

X1.3.0

- 统一Action为Controller
- 增加文章搜索功能
- 增加前台编辑器
- 增加模板常量**STATICS**
- 增加最后评论时间写入
- 修复leuu bug
- CommonModel _before_write 数据过滤bug
- 后台评论管理,默认所有评论
- 文章分类path更新优化
- simplebootx模板文章页css样式优化
- 修复分页类bug
- 后台文章编辑所有文章链接错误
- 修复bug#4验证码不显示
- 优化bug#3页面使用LEUU函数后每个页面都查询
- 优化公共模型的调用方法，以兼容php5.3.0-5.3.2
- 修复sp_sql_posts_bycatid和sp_sql_posts_paged_bycatid两个方法where语句问题
- 修复bluesky主题分页样式问题
- 修复文章推荐，置顶bug

X1.2.0

- url美化
- 特殊用户名过滤
- 增加推荐，置顶功能
- 幻灯片隐藏显示功能
- 广告隐藏显示功能
- 友情链接隐藏显示功能
- 评论计数

X1.1.0

全新的ThinkPHP 3.2.2架构，使用php命名空间，让开发快起来吧！

- 统一Member应用为User,合并前台会员和后台管理员
- 完善用户中心，会员登录注册
- 增加编辑头像，绑定账号，我的评论，我的收藏
- 增加文章点赞，收藏，查看功能，可与其它应用共用
- 增强文章评论功能，方便多应用共用
- 优化留言功能，增强安全性
- 优化前台模板，增加多个实用组件，方便以后复用
- 增加后台风格切换功能；
- 增加后台风格bluesky
- 优化后台菜单使用方式
- 优化数据库中一些不规范字段
- 增加前台标签库TagLibHome，统一include标签为tc_include

X1.0.0

全新的ThinkPHP 3.2.2架构，使用php命名空间，让开发快起来吧！

- 统一前后台UI框架为simpleboot(bootstrap 2.3.2 ThinkCMF优化版)
- 集成Ucenter
- 增加文章评论功能
- 增加留言功能
- 全面支持SAE云平台
- 增加文章内分页功能
- 升级后台编辑器到Ueditor最新版本
- 优化后台ajax提交，未登陆时自动退出
- 优化后台所有文章按发布时间递减排序
- 修复后台密码会偶然不对的错误
- 修复SAE，linux下类库加载失败

- 修复ueditor chrome模板功能bug
- 修复文件上传bug

X1.0.0 alpha2

- 修复SAE，linux下类库加载失败
- 修复ueditor chrome模板功能bug
- 修复文件上传bug

X1.0.0 alpha

全新的ThinkPHP 3.2.2架构，使用php命名空间，让开发快起来吧！

- 集成Ucenter
- 增加文章评论功能
- 增加留言功能
- 全面支持SAE云平台
- 增加文章内分页功能
- 升级后台编辑器到Ueditor最新版本
- 优化后台ajax提交，未登陆时自动退出
- 优化后台所有文章按发布时间递减排序
- 修复后台密码会偶然不对的错误

INSTALL

安装请执行http://yourdomain/

ThinkCMF 免责声明

1、利用 ThinkCMF 构建的网站的任何信息内容以及导致的任何版权纠纷和法律争议及后果，ThinkCMF 官方不承担任何责任。

2、您一旦安装使用ThinkCMF，即被视为完全理解并接受本协议的各项条款，在享有上述条款授予的权力的同时，受到相关的约束和限制。

ThinkCMF 使用建议

请在您的网站首页加上ThinkCMF相关链接，O(∩_∩)O~ ！

捐赠**ThinkCMF**

http://www.thinkcmf.com/donate/index.html

您的每一份帮助都将支持ThinkCMF做的更好，走的更远！

ThinkCMF 正在为你开放更多....

最后文件编辑时间: 2017-02-03 17:55:31

关于BootStrap

Bootstrap是Twitter推出的一个开源优秀的前端响应式开发框架。它由Twitter的设计师Mark Otto和Jacob Thornton合作开发。ThinkCMF首发版本将全面支持BootStrap，ThinkCMF不仅创造了第一个中文内容管理框架，还致力于将BootStrap本土化。

Bootstrap是基于jQuery框架开发的，它在jQuery框架的基础上进行了更为个性化和人性化的完善，形成一套自己独有的网站风格，并兼容大部分jQuery插件。

Bootstrap中包含了丰富的Web组件，根据这些组件，可以快速的搭建一个漂亮、功能完备的网站。其中包括以下组件：下拉菜单、按钮组、按钮下拉菜单、导航、导航条、面包屑、分页、排版、缩略图、警告对话框、进度条、媒体对象等;

ThinkCMF演示站http://demo.thinkcmf.com就是完全基于Bootstrap2.3.2开发，对于ThinkCMF的前端开发，我们推荐您使用bootstrap,但不限制必须使用。我们自己完善的simpleboot开发框架，完全基于bootstrap 2.3.2,但它拥有更多的组件，同时支持IE7+，我们已经在ThinkCMF的public目录下集成了simpleboot，你在开发中完全可以直接使用，不用再次安装。

最后文件编辑时间: 2017-02-03 17:55:31

基础

开发规范

ThinkCMF完全按ThinkPHP开发规范进行

开发前请详细阅读：
http://www.kancloud.cn/manual/thinkphp/1687

ThinkCMF特别开发规范

1. 应用后台控件器放在应用Controller目录中，最好不要存放在Admin/Controller下，保证应用模块独立
2. 应用后台控件器命名以****adminController.class.php或者Admin****Controller.class.php命名的是后台Controller, 在后台菜单导入时会自动识别
3. 应用后台控件器方法命名：用户无法访问的内部方法，请以下划线（_）开头；
4. 附件保存路径，要相对于upload目录，只保存之后路径
5. 使用I函数获取post和get的数据
6. 模板中php代码注释都使用/**/的方式，//这种方式一定不要用，否则debug关闭后会有各种问题！
7. 各种php文件最好别加?>结束，防止响应多余字符

最后文件编辑时间: 2017-02-03 17:55:31

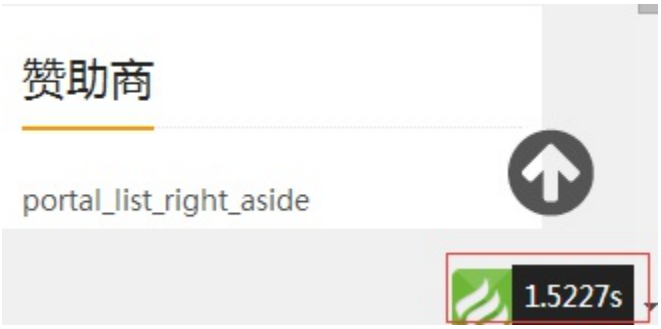
调试模式

ThinkCMF调试模式的开关在程序根目录index.php文件里

```
13                  $array[$k] = stripslashesRecursive
14              }
15          }
16          return $array;
17      }
18      $_GET = stripslashesRecursive($_GET);
19      $_POST = stripslashesRecursive($_POST);
20  }
21  //开启调试模式
22  define("APP_DEBUG", true);
23  //网站当前路径
24  define('SITE_PATH', getcwd());
25  //项目路径，不可更改
26  define('APP_PATH', SITE_PATH . '/application/');
27  //项目相对路径，不可更改
28  define('SPAPP_PATH',    SITE_PATH.'/simplewind/');
```

APP_DEBUG默认是开启的，方便开发者调试；
开发完成可以改成false,关闭调试模式，进入生产环境！

调试模式下会有如下的图标：



关闭调试模式后它就会消失！

最后文件编辑时间: 2017-02-03 17:55:31

应用

应用就是 `application` 目录下的一个模块，它是独立于其它模块存在的，有自己独立的运行空间；

应用采用MVC的结构:

拿Blog应用举例:

```
Blog
  Controller 控制器目录（必备）
  Common 函数库（可选）
  Conf 配置（可选）
  Lang 多语言包（可选）
  Menu 后台菜单（可选）
  Model 模型（可选）
  nav.php 前台导航文件（可选）
```

而在 CMF 中应用的 View 是独立于应用之外的，它分为前台view 和后台view, 分别存放在 `themes/前台主题/应用名` ，和 `admin/themes/后台主题/应用名` 下；

创建一个Blog应用

- 在 `application/Common/Conf/config.php` 文件 `MODULE_ALLOW_LIST` 数组里加上应用名Blog
- 在 `application` 下创建上面讲到的 `Blog` 目录结构；
- 创建一个控制器,在模块 `Controller` 目录下创建一个 `IndexController.class.php` 文件

```php
<?php
namespace Blog\Controller;

use Common\Controller\HomebaseController;

class IndexController extends HomebaseController{

    // 首页
    public function index(){
      echo "this is blog index !";
    }

}
```

- 访问 http://你的域名/index.php?g=blog&m=index&a=index;

> 注意控制器的命名规范：
>
> 控制器类的命名方式是：控制器名（驼峰法，首字母大写）+Controller （如IndexController）;

> 控制器文件的命名方式是：类名+class.php（类文件后缀）（如 IndexController.class.php);

如何获取

ThinkCMF获取方式：

> 官方网站 :http://www.thinkcmf.com可获取最新版本
> git@osc :http://git.oschina.net/thinkcmf/ThinkCMFX
> github :https://github.com/thinkcmf/cmfx

最后文件编辑时间: 2017-02-03 17:55:31

环境要求

ThinkCMFX可以支持Windows/Unix服务器环境，需要PHP5.3.0以上版本支持， 可运行于包括Apache、IIS和nginx在内的多种WEB服务器和模式，支持Mysql、 MsSQL、PgSQL、Sqlite、Oracle、Ibase、Mongo以及PDO等多种数据库和连接，推荐LAMP构架。框架本身没有什么特别模块要求，具体的应用系统运行环境要求视开发所涉及的模块。 ThinkCMF底层运行的内存消耗极低，而本身的文件大小也是轻量级的，因此不会出现 空间和内存占用的瓶颈。对于刚刚接触PHP或者ThinkCMF的新手，我们推荐使用集成开发 环境WAMPServer（wampserver是一个集成了Apache、PHP和MySQL的开发 套件，而且支持不同PHP版本、MySQL版本和Apache版本的切换）来使用ThinkCMF 进行本地开发和测试。

安装使用

上传你的代码，直接在浏览器中输入你的域名或IP（例如： `www.yourdomain.com` ）,安装程序会自动执行安装。期间系统会提醒你输入数据库信息以完成安装，安装完成后建议删除 `application` 目录下的 `Install` 。

安装完成，一定把 `data/conf/db.php` 文件做个备份！否则大神也救不了你！

ThinkCMF目录结构：

```
|--admin                    /管理后台URL重定向目录，你可以将文件夹名改为任何你喜欢的
    |--themes               /后台模板文件目录
|--application              /应用目录
|--data                     /各类数据存放目录，包括缓存数据
|--simplewind               /核心包，无特殊情况请勿改动
|--public                   /静态文件存放包，包含bootstrap资源
|--themes                   /前台模板文件目录
```

application 目录结构：

```
|--application
  |--Admin                  /后台管理应用
  |--Api                    /公共接口
  |--Asset                  /资源管理应用
  |--Comment                /评论应用
  |--Common                 /应用公共模块
  |--Portal                 /门户应用
```

应用的目录结构规范：

举例应用Portal

```
|--Portal
  |--Controller             /必须目录，存放应用的操作模块如：/IndexController.class.php
  |--Conf                   /可选，应用配置文件存放目录，如应用无配置文件则不需要
  |--Common                 /可选，应用函数库，如无则不需要
```

配置

ThinkPHP支持多种配置格式,ThinkCMF默认使用php数组的形式,其它使用方式请参考:
http://www.kancloud.cn/manual/thinkphp/1678

控制器

ThinkCMF目前分为两种控制器，前台和后台控制器；

无论是前台，还是后台控制器都位于应用的Controller目录下.
前台控制器一般继承 `Common\Controller\HomebaseController` ,如果你需要用户登录才访问或操作此控制器，就要继承 `Common\Controller\MemberbaseController` ;

我们来创建一个前台控制器：
在Blog应用Controller目录下创建一个IndexController.class.php文件

```
<?php
namespace Blog\Controller;
use Common\Controller\HomebaseController;

class IndexController extends HomebaseController{
    public function index(){
      echo "this is blog index !";
    }
}
```

访问地址：http://你的域名/?g=blog&m=index&a=index;

如果只让用户在登录时才能访问此控制器，只需把HomebaseController换成MemberbaseController；

```
<?php
namespace Blog\Controller;
use Common\Controller\MemberbaseController;

class IndexController extends MemberbaseController{
    public function index(){
      echo "this is blog index !";
    }
}
```

这里 IndexController 下的所有方法，用户只能在登录后才能访问，否则会报错，让用户登录；

创建后台控制器：

在Blog应用Controller目录下创建一IndexadminController.class.php文件（注意：这里有文件命名规则，以****adminController.class.php或 Admin****Controller.class.php命名的是后台Controller,在后台菜单导入时会自动识别；）

```
<?php
namespace Blog\Controller;
```

```
use Common\Controller\AdminbaseController;

class IndexadminController extends AdminbaseController{
    public function index(){
        echo "this is admin controller!";
    }
}
```

访问地址：http://你的域名/?g=blog&m=indexadmin&a=index，这里你一定要先登录后台，才能访问；

如果你想这个控制不用管理员登录也能被访问到，只给 `IndexadminController` 加个空的 `_initialize()` 方法；

```
<?php
namespace Blog\Controller;
use Common\Controller\AdminbaseController;

class IndexadminController extends AdminbaseController{

    //初始化，这里不执行父类的初始化方法
    public function _initialize(){}

    public function index(){
        echo "this is admin controller!";
    }
}
```

URL生成

ThinkCMF遵循**ThinkPHP**的**url**生成方法：

U方法的定义规则如下（方括号内参数根据实际应用决定）：

U('地址表达式',['参数'],['伪静态后缀'],['是否显示域名'])

```
U('Blog/Index/index') // 生成Blog应用Index控制器的index操作的URL地址
U('Portal/Article/index?id=1') // 生成Portal应用Article控制器的index操作 并且参数id为1的URL地址
U('Portal/Article/index',array('id'=>1) // 生成Portal应用Article控制器的index操作 并且参数id为1的URL地址
U('User/index') // 生成当前应用的User控制器的index操作的URL地址
```

参数：
U方法的第二个参数支持数组和字符串两种定义方式，如果只是字符串方式的参数可以在第一个参数中定义，例如：

```
U('Blog/Index/index',array('cat'=>1,'status'=>1))
U('Blog/Index/index','cat=1&status=1')
U('Blog/Index/index?cat=1&status=1')
```

添加生成带域名的 url,只要把第四个参数设置为 true

```
U('Blog/Index/index','cat=1&status=1',true,true)
```

leuu/UU方法:

为了配合后台设置的 url 美化规则，cmf 增加了 leuu/UU两个方法，UU方法只是 leuu 的别名用法一样；
leuu 的参数列表和 U 方法一样，只是要配合后台设置的 url美化规则才能生效，如没有规则，leuu 其实就是 U 方法；

url的美化规则设置：

进入ThinkCMF后台,设置->网站信息->URL美化

原始网址规则：应用名（小写）/控制器名/操作名?参数 如：portal/list/index?id=1 显示网址：英文字母加数字，不带后缀；

如：

原始网址规则：portal/list/index?id=1

显示网址：cases

```
leuu("portal/list/index",array('id'=>1)) //生成的 url为/cases.html
```

列出常用的优化方案：

```
portal/list/index?id=1 news   http://demo.thinkcmf.com/news.html
portal/list/index?id=2 discovery http://demo.thinkcmf.com/discovery.html
portal/page/index?id=2 contacts http://demo.thinkcmf.com/contacts.html
portal/page/index?id=14 about http://demo.thinkcmf.com/about.html
portal/article/index article/:id\d http://demo.thinkcmf.com/article/1.html
portal/list/index list/:id\d http://demo.thinkcmf.com/list/1.html
```

注意：进行URL美化之后，要进入ThinkCMF后台，进行清除缓存操作后，刷新前台，才能看到美化效果。

AJAX返回

请参考：http://www.kancloud.cn/manual/thinkphp/1719

跳转和重定向

请参考：http://www.kancloud.cn/manual/thinkphp/1720

输入变量

请参考：http://www.kancloud.cn/manual/thinkphp/1721

请求类型

请参考：http://www.kancloud.cn/manual/thinkphp/1722

空操作

请参考：http://www.kancloud.cn/manual/thinkphp/1723

空控制器

请参考：http://www.kancloud.cn/manual/thinkphp/1724

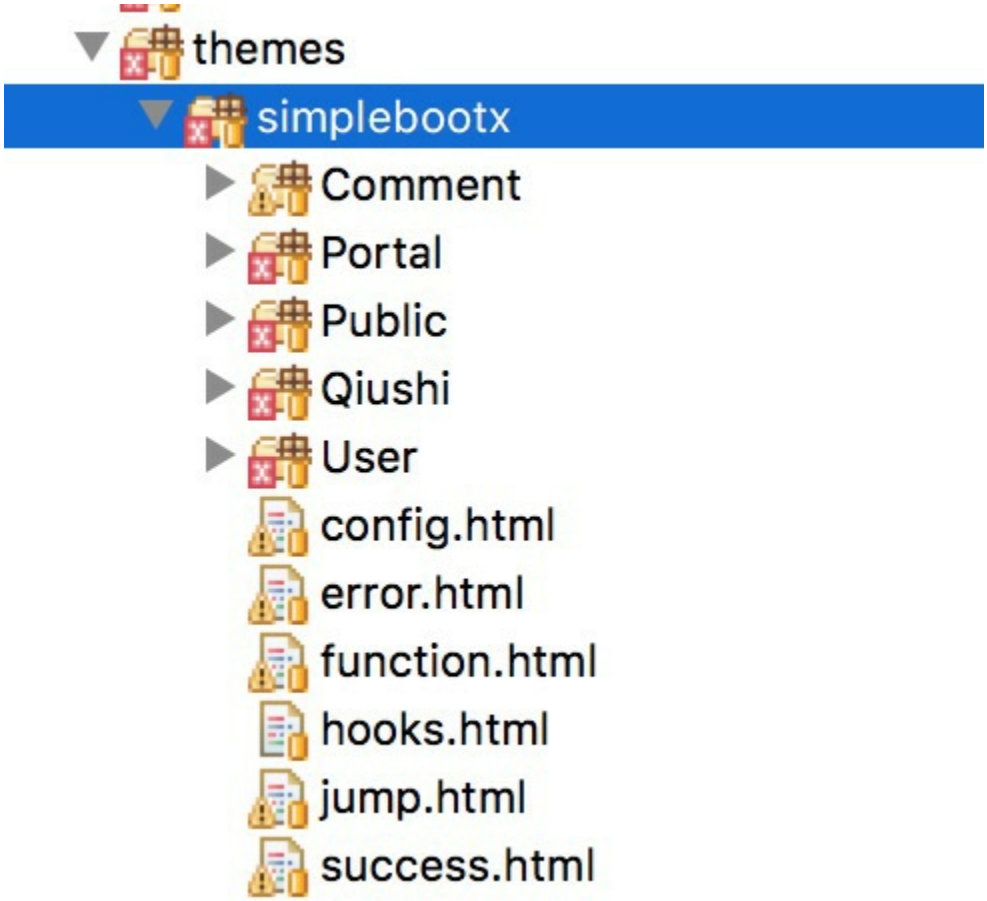
视图

模板定义

在 cmf中模板就是一个 html 文件，可分为前台模板文件和后台模板文件；

前台模板位于 `themes` 目录下，后台模板位于 `admin/themes` 目录下，前后台都是多主题机制的,可以分开设置不同的主题；

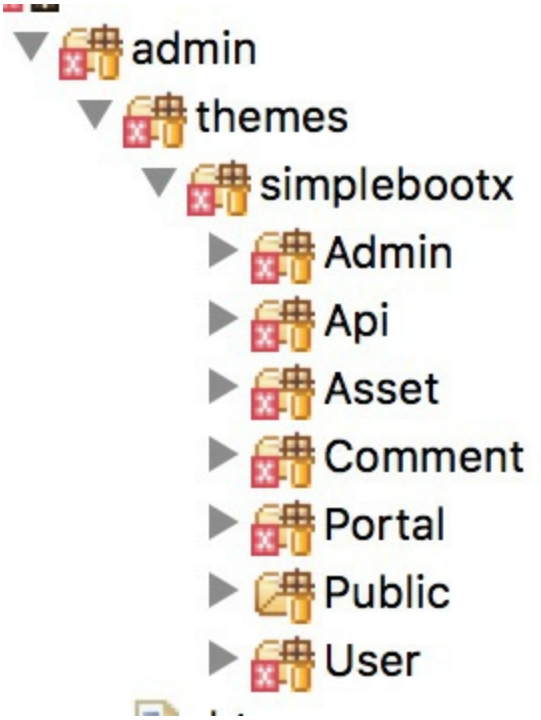
前台默认模板是 `simplebootx` ,以后也可能会换，我们先以这个为例；



这是前台模板的结构；应用之间彼此分开，Portal目录下就对应的是application/Portal应用的模板文件；

一个模板文件大致路径是这样的:themes/主题名/应用名/控制器名/操作名+(.html) 或者 themes/主题名/应用名/自定义模板名+(.html)，如：themes/simplebootx/User/Profile/password.html,themes/simplebootx/Portal/index.html;

后台默认模板也是simplebootx只是和前台的名字一样而已，实质它是后台主题；



和前台的模式基本一样，一个后台模板文件大致路径是这样的: admin/themes/主题名/应用名/控制器名/操作名+(.html) 或者 admin/themes/主题名/应用名/自定义模板名+(.html)，如：admin/themes/simplebootx/Admin/Link/add.html;

『themes/主题名/应用名』和『admin/themes/主题名/应用名』这一部分基本是固定的，开发者最好不要去修改，也最好不要跨应用调用模板，这会让结构很乱，不方便维护；

应用名以后的部分，开发者可以用控制器的display方法指定模板的具体文件名，display用法我们在模板渲染部分会详细说明；

模板主题

cmf分为前台模板主题和后台模板主题，两都分开独立设置，它们也在不同的目录下，前台模板在themes下，后台模板在 `admin/themes` 下；

目前cmf默认的前后台模板主题分别是 `simplebootx` 和 `simplebootx` ，路径分别是 `themes/simplebootx` , `admin/themes/simplebootx` ；

所以这种架构形式下，前后台是都支持多主题的，如果我想增加的主题，只要把默认主题复制后，目录名改一下放在前后台主题目录下，就是一个新的主题，当然你完全可以从零开始写，只要确保文件名对应就可以了；要注意模板命名格式，英文字母加数字的形式；

目前前台主题切换是可以在后台管理里完成的，打开后台管理，在设置->网站信息->模板方案里选择自己要的主题就可以了；

注意你在打开模板方案里可能会发现_en-us,_mobile,_mobile_en-us结尾的类似模板，这些都是相应模式下的模板，比如，你后台设置的是simplebootx,如果是手机用户系统会自动判断，并使用simplebootx_mobile 模板，如果用户是英文的系统也会自动判断使用simplebootx_en-us模板；

后台主题切换，暂时没有做到后台，不过你可以在后台设置->网站信息->后台风格,切换你的后台风格，如果你愿意自己再做套后台模板的话，你可以在配置文件 `application/Common/Conf/config.php` 里找到 `SP_ADMIN_DEFAULT_THEME` 配置项，把simplebootx

换成你自己的后台主题名，这样以后如果你还想用官方的后台主题，还可以再换回来！后台模板制作我们并不会提供教程，不过作为高手你懂得！

模板赋值

模板赋值就是在控制器里把控制器的变量传递给模板，对于系统变量你不用赋值我们可以通过特殊的标签在模板里输出,变量赋值我们都通过assign方法进行赋值；所有assign方法，都必须在display,show,fetch方法执行前调用；

1. 传递一个$name到模板

```
$this->assign('name','this is name');
```

这样就可以在模板使用$name了，直接输出变量可以{$name}
也可以直接在php标签里使用这个变量；

```
<php>
echo $name;
</php>
```

2. 传递一个数组

```
$user=array(
 'name'=>'Dean',
 'email'=>'cmf@simplewind.net',
 'phone'=>'15121010086'
);
$this->assign($user);
```

模板里就被传递了$name,$email,$phone 三个变量了，你可以直接输出{$name},{$email},{$phone}

获取内容

还有一种调用模板的情况是我们只想渲染后返回模板渲染后的数据而不是直接输出，这时我们会用fetch方法；

fetch的用法和display 完全一样，只是不直接输出了；

```
//不带任何参数
$content=$this->fetch();
```

此种方式系统会自动判断模板路径，并渲染出模板内容，此种方式模板路径是：主题名/应用名/控制器名/操作名+模板文件后缀名；

```
$content=$this->fetch('edit');
```

此种方式表示调用此控制器下的edit操作的模板；

```
$content=$this->fetch(':index');
```

此种方式表示调用此应用下index控制器的模板；

```
$str_content='this is a template file content,{$name}';
$content=$this->fetch('',$str_content);
```

此种方式是表示直接渲染传入的字符串里的模板内容；

通过以上方式得到渲染后的模板内容后，你可以对此内容做进一步处理；

模板渲染

在控制器里模板渲染最常用的方法是display,在CMF中支持如下的模板渲染方式：

```
//不带任何参数
$this->display();
```

此种方式系统会自动判断模板路径，并渲染出模板内容，此种方式模板路径是：主题名/应用名/控制器名/操作名+模板文件后缀名；

```
$this->display('edit');
```

此种方式表示调用此控制器下的edit操作的模板；

```
$this->display(':index');
```

此种方式表示调用此应用下index控制器的模板；

还有一种使用场景，就是有时我们可能会把模板存在数据库，或缓存等地方，并没有任何模板文件，只有模板内容，这时我们就要用到另一种方法show了；

```
//$content 是数据库中的模板内容
$this->show($content);
```

通过方法，会对模板内容进行渲染，并输出。

模型

模型定义

模型定义

为方便框架的后续升级，**thinkcmf**与**thinkphp**框架模型定义方法完全一致。 模型类并非必须定义，只有当存在独立的业务逻辑或者属性的时候才需要定义。

模型类通常需要继承系统的\Think\Model类或其子类，下面是一个Home\Model\UserModel类的定义：

```
namespace Home\Model;
use Think\Model;
class UserModel extends Model {
}
```

模型类的作用大多数情况是操作数据表的，如果按照系统的规范来命名模型类的话，大多数情况下是可以自动对应数据表。 模型类的命名规则是除去表前缀的数据表名称，采用驼峰法命名，并且首字母大写，然后加上模型层的名称（默认定义是Model），例如： |模型名| 约定对应数据表（假设数据库的前缀定义是 cmf_）| |----|----| |UserModel| cmf_user| 如果你的规则和上面的系统约定不符合，那么需要设置Model类的数据表名称属性，以确保能够找到对应的数据表。

1. 数据表定义 在ThinkPHP的模型里面，有几个关于数据表名称的属性定义：

属性	说明
tablePrefix	定义模型对应数据表的前缀，如果未定义则获取配置文件中的DB_PREFIX参数
tableName	不包含表前缀的数据表名称，一般情况下默认和模型名称相同，只有当你的表名和当前的模型类的名称不同的时候才需要定义。
trueTableName	包含前缀的数据表名称，也就是数据库中的实际表名，该名称无需设置，只有当上面的规则都不适用的情况或者特殊情况下才需要设置。
dbName	定义模型当前对应的数据库名称，只有当你当前的模型类对应的数据库名称和配置文件不同的时候才需要定义。

举个例子来加深理解，例如，在数据库里面有一个cmf*users*表，而我们定义的模型类名称是*UserModel*，按照系统的约定，这个模型的名称是*User*，对应的数据表名称应该是*cmf_user*（全部小写），但是现在的数据表名称是*cmf_users*，因此我们就需要设置*tableName*属性来改变默认的规则（因为*cmf*程序默认已经在配置文件里面定义了*DB_PREFIX* 为 *cmf*）。

```
namespace Home\Model;
use Think\Model;
class UserModel extends Model {
    protected $tableName = 'users';
}
```

注意这个属性的定义不需要加表的前缀cmf_

如果我们需要CategoryModel模型对应操作的数据表是 top_user，那么我们只需要设置数据表前缀即可：

```
namespace Home\Model;
use Think\Model;
class UserModel extends Model {
    protected $tablePrefix = 'top_';
}
```

如果你的数据表直接就是user，而没有前缀，则可以设置tablePrefix为空字符串。

```
namespace Home\Model;
use Think\Model;
class UserModel extends Model {
    protected $tablePrefix = '';
}
```

没有表前缀的情况必须设置，否则会获取当前配置文件中的 DB_PREFIX。 而对于另外一种特殊情况，我们需要操作的数据表是top_categories，这个时候我们就需要定义 trueTableName 属性

```
namespace Home\Model;
use Think\Model;
class CategoryModel extends Model {
    protected $trueTableName = 'top_categories';
}
```

注意trueTableName需要完整的表名定义。 除了数据表的定义外，还可以对数据库进行定义（用于操作当前数据库以外的数据表），例如 top.top_categories：

```
namespace Home\Model;
use Think\Model;
class CategoryModel extends Model {
    protected $trueTableName = 'top_categories';
    protected $dbName = 'top';
}
```

系统的规则下，tableName会转换为小写定义，但是trueTableName定义的数据表名称是保持原样。因此，如果你的数据表名称需要区分大小写的情况，那么可以通过设置trueTableName定义来解决。

1. 这里谈下THINKPHP的自动验证和自动完成功能（thinkcmf也是完全一样的） 假设上文我们已经定义了Home\Model\UserModel.class.php 代码完善后如下:

```
namespace Home\Model;
```

```php
use Think\Model;
class UserModel extends Model {
  //自动完成
  protected $_auto = array (
           array('status','1'),  // 新增的时候把status字段设置为1
           array('random_str','creat_str',1,'function'), // 新增时自动完成random_str
        );
  //自动校验
  protected $_validate = array(
      array('verify','require','验证码必须！'), //默认情况下用正则进行验证
      array('name','','帐号名称已经存在！',0,'unique',1), // 在新增的时候验证name字段是否唯一

      array('value',array(1,2,3),'值的范围不正确！',2,'in'), // 当值不为空的时候判断是否在一个范围内
      array('name','check_name','密码格式不正确',0,'function'), // 自定义函数验证名称是否为admin
    );

  /*
  *为自动校验定义的校验方法 检测名称是否为admin
  *@prama $name 名字
  */
  function check_name($name){
      return $name=='admin'?false:true;
  }
  /*
  *为自动完成定义的完成方法//创建随机字符串用来
  *@prama $length 随机字符串长度
  */
  function creat_str($length=6){
      return sp_random_string($length);
  }
  /*
   // 随机字符串生成 cmf自带函数;
   // @param int $len 生成的字符串长度
   // @return string

  function sp_random_string($len = 6) {
      $chars = array(
              "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k",
              "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v",
              "w", "x", "y", "z", "A", "B", "C", "D", "E", "F", "G",
              "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R",
              "S", "T", "U", "V", "W", "X", "Y", "Z", "0", "1", "2",
              "3", "4", "5", "6", "7", "8", "9"
      );
      $charsLen = count($chars) - 1;
      shuffle($chars);    // 将数组打乱
      $output = "";
      for ($i = 0; $i < $len; $i++) {
          $output .= $chars[mt_rand(0, $charsLen)];
```

```
        }
        return $output;
    }
    */
}
```

参考文献

1. http://www.kancloud.cn/manual/thinkphp/1728
2. http://www.thinkcmf.com/topic/topic/index/id/432.html

文档问题联系iwzh

模型实例化

为方便框架的后续升级，**thinkcmf**与**thinkphp**框架模型操作方法完全一致。 所以这里是支持**thinkphp**原生的**M**方法和**D**方法来实例化模型**;**

实例化模型的好处： 可以实现自动完成，自动校验功能。使用模型封装的方法。分离M层和C层；

M方法：

M(['模型名'],['数据表前缀'],['数据库连接信息']);

```
//实例化模型
$User=M('User');
// 执行具体的数据操作
$User->select();
```

M方法仅支持基本的CURD操作;但是性能会较D方法高。 如果你的模型类有自己的业务逻辑，*M*方法是无法支持的，就算是你已经定义了具体的模型类，*M*方法实例化的时候是会直接忽略。 **M**方法的特殊用法： $model=M();//实例化空模型；

$model->query('select * from cmf_user where id=1');//使用原生sql语句进行查询id为1的用户

D方法:

假设当前模块为Home; D('User');D('Home/User');是一样的；

如果在Linux环境下面，一定要注意D方法实例化的时候的模型名称的大小写。 D方法可以自动检测模型类，如果存在自定义的模型类，则实例化自定义模型类，如果不存在，则会实例化系统的\Think\Model基类，同时对于已实例化过的模型，不会重复实例化。 D方法的参数就是模型的名称，并且和模型类的大小写定义是一致的，例如： 参数 实例化的模型文件 User 对应的模型类文件的 \Home\Model\UserModel.class.php UserType 对应的模型类文件的 \Home\Model\UserTypeModel.class.php

用法示例:

```
//实例化模型
$User = D('User');
// 相当于 $User = new \\Home\\Model\\UserModel();
// 执行具体的数据操作
$User->select();
//$User->diy_select();//diy_select方法是您在\\Home\\Model\\UserModel.class.php中自定义的方法。
```

当 *\Home\Model\UserModel* 类不存在的时候，*D*函数会尝试实例化公共模块下面的 *\Common\Model\UserModel* 类 D方法还可以支持跨模块调用，需要使用：

```
//实例化Admin模块的User模型
D('Admin/User');
//实例化Extend扩展命名空间下的Info模型
D('Extend://Editor/Info');
```

注意：跨模块实例化模型类的时候 不支持自动加载公共模块的模型类

M方法和**D**方法的区别

1. M方法不用加载具体模型类效率更高。但仅能实现基础的CURD;
2. D方法会先实例化具体的模型类，找不到后自动调用M方法来实例化模型类

名词解释

1. CURD(Create,Update,Read,Delete),数据库的增删查改操作

参考文献：

1. http://www.kancloud.cn/manual/thinkphp/1729

连接数据库

一、全局配置定义

```
return array(
    'DB_TYPE' => 'mysql',
    'DB_HOST' => '127.0.0.1',
    'DB_NAME' => 'thinkcmf',
    'DB_USER' => 'root',
    'DB_PWD' => 'root',
    'DB_PORT' => '3306',
    'DB_PREFIX' => 'cmf_',
    //密钥
    "AUTHCODE" => '2oF6v4m67BdQnfwMSf',
    //cookies
    "COOKIE_PREFIX" => '03a9uH_',
);
```

如果我们已经在配置文件中配置了额外的数据库连接信息，例如：

```
return array(
    //数据库1
    'DB_TYPE' => 'mysql',
    'DB_HOST' => '127.0.0.1',
    'DB_NAME' => 'thinkcmf',
    'DB_USER' => 'root',
    'DB_PWD' => 'root',
    'DB_PORT' => '3306',
    'DB_PREFIX' => 'cmf_',
    //密钥
    "AUTHCODE" => '2oF6v4m67BdQnfwMSf',
    //cookies
    "COOKIE_PREFIX" => '03a9uH_',
    //数据库2
    'DB_CONFIG2' => array(
        'db_type'  => 'mysql',
        'db_user'  => 'root',
        'db_pwd'   => 'root',
        'db_host'  => '127.0.0.1',
        'db_port'  => '3306',
        'db_name'  => 'repast'
       ),
    );
```

二、实例化定义

```
$User = M('User','cmf_','DB_CONFIG2');
```

作者：小夏

更多模型

ThinkCMF支持所有ThinkPHP模型用法,其它使用方式请参考:
http://www.kancloud.cn/manual/thinkphp/1727

系统函数

Comments()

> V1.1.1新增

```
Comments($table,$post_id,$params=array())
```

功能:

生成评论组件

参数:

`$table` :评论对象所在的表,不带表前缀

`$post_id` :评论对象的id

`$params` :额外参数,类型为数组,目前只支持tpl参数

模板中使用

```
<!-- 评论文章表里的某个ID为$object_id的文章-->
{:Comments("posts",$object_id)}

<!--
    评论文章表里的某个ID为$object_id的文章,
    并且使用mycomments模板(此模板位于当前主题 Comment目录下)
-->
{:Comments("posts",$object_id,array('tpl'=>'mycomments'))}
```

hook()

> X1.4.0新增

```
hook($hook,$param)
```

功能：

添加钩子

参数：

`$hook` :钩子名称
`$param` :传入参数，默认为空数组

返回：

无

模板使用：

```
{:hook('footer')}
{:hook('sider',array('text'=>'hello ThinkCMF')}
```

控制器方法里使用:

```
//不带参数
hook('your_hook_name');

//带参数
hook('your_hook_name',array('text'=>'Hello ThinkCMF'));
```

leuu()/UU()

UU和leuu是ThinkCMF X1.2增加有url美化方法,两个用法完全一样，UU就是leuu的别名；

用法和U方法一样；对于需要美化的url要用到这个方法；

leuu方法的定义规则如下（方括号内参数根据实际应用决定）：

```
leuu('地址表达式',['参数'],['伪静态后缀'],['显示域名'])
```

地址表达式的格式定义如下：

```
[应用/控制器/操作#锚点@域名]?参数1=值1&参数2=值2...
leuu('portal/list/index',array('id'=>1))
```

leuu方法要配置后台设置的url规则才能完成url的美化；如果没有相应规则，则和U方法生成的url一样；

url的美化规则写法： 进入ThinkCMF后台 后台 设置->网站信息->URL美化

原始网址规则：应用名（小写）/控制器名/操作名?参数 如：portal/list/index?id=1 显示网址：英文字母加数字，不带后缀；

列出常用的优化方案：

```
portal/list/index?id=1 news http://demo.thinkcmf.com/news.html
portal/list/index?id=2 discovery http://demo.thinkcmf.com/discovery.html
portal/page/index?id=2 contacts http://demo.thinkcmf.com/contacts.html
portal/page/index?id=14 about http://demo.thinkcmf.com/about.html
portal/article/index article/:id\d http://demo.thinkcmf.com/article/1.html
portal/list/index list/:id\d http://demo.thinkcmf.com/list/1.html
```

网站信息 URL美化 添加URL规则

排序	ID	原始网址	显示网址	状态
0	1	portal/list/index?id=1	news	已启用
0	2	portal/list/index?id=2	discovery	已启用
0	4	portal/page/index?id=2	contacts	已启用
0	5	portal/page/index?id=14	about	已启用
0	6	portal/article/index	article/:id\d	已启用
0	7	portal/list/index	list/:id\d	已启用

sp_asset_relative_url()

> V1.1.1新增

```
sp_asset_relative_url($asset_url)
```

功能:

获取文件相对路径

参数:

`$asset_url` :文件URL

返回

类型字符串,带协议的绝对地址直接返回原来的值,否则会转化为相对于系统upload 目录的文件路径

实例

```
<?php
$file='/data/upload/1.png';//文件路径
$path=sp_asset_relative_url($file);//转化
echo $path;//输出数据库保存的文件路径（都是相对于data/upload文件夹的）,结果为1.png
```

sp_auth_check()

> X1.5.0新增

```
sp_auth_check($uid,$name=null,$relation='or')
```

功能：
用户权限验证

参数：
`$uid` : 当前登录用户或者管理员的id
`$name` :需要验证的规则列表,支持逗号分隔的权限规则或索引数组,默认为当前url
`$relation` :如果为 'or' 表示满足任一条规则即通过验证;如果为 'and'则表示需满足所有规则才能通过验证

返回：
类型boolean 通过验证返回true;失败返回false

使用：

```
sp_auth_check(2);
sp_auth_check(2,'admin/ad/index');
sp_auth_check(2,array('admin/ad/index'));
sp_auth_check(2,'admin/ad/index,admin/ad/add','and');
```

sp_authcode()

> V1.1.1新增

```
<?php
$string='1324123i412qewrwerqwe';
$string=sp_authcode($string);//解密字符串
echo $string;//输出解密后的字符串
?>
```

sp_authencode()

> V1.1.1新增

```
<?php
$string='666666';
$string=sp_authencode($string);//加密字符串
echo $string;//输出加密后的字符串
?>
```

sp_check_lang()

> X2.2.0新增

```
sp_check_lang()
```

功能: 判断当前的语言包，并返回语言包名

参数: 无

返回： 类型string 包名,如:zh-cn

使用：

```
$lang=sp_check_lang();
```

sp_check_mobile_verify_code()

> X2.3.0新增(暂时无法使用)

```
sp_check_mobile_verify_code($mobile='',$verifycode='')
```

功能： 手机验证码检查，验证完后销毁验证码增加安全性

参数： `$mobile`：手机号 `$verifycode`:验证码 返回： 类型boolean true：手机验证码正确，false：手机验证码错误

使用：

```
$is_right_mobile_code = sp_check_mobile_verify_code('15121000000','123456');
```

最后文件编辑时间: 2017-02-03 17:55:31

sp_check_user_action()

> X1.1新增

```
sp_check_user_action($object,$count_limit,$ip_limit,$expire)
```

功能:

检查用户对某个url,内容的可访问性，用于记录如是否赞过，是否访问过等等;开发者可以自由控制，对于没有必要做的检查可以不做，以减少服务器压力

参数说明：

`$object`：访问对象的id,格式：不带前缀的表名+id;如posts1表示xx_posts表里id为1的记录;如果object为空，表示只检查对某个url访问的合法性
`$count_limit`：访问次数限制,如1，表示只能访问一次
`$ip_limit`：ip限制,false为不限制，true为限制
`$expire`：距离上次访问的最小时间单位s，0表示不限制，大于0表示最后访问$expire秒后才可以访问

返回：

`true`可访问， `false`不可访问

sp_check_verify_code()

> X1.6.0新增

```
sp_check_verify_code()
```

功能：
验证码检查，验证完后销毁验证码增加安全性

参数：
无

返回：
类型boolean true|false;

使用：

```
<?php
if(!sp_check_verify_code()){
    echo '验证码不正确';
}
```

> 注：表单提交时验证码name为verify；支持POST和GET方法

sp_clear_cache()

```
sp_clear_cache()
```

功能:

清除缓存

参数:

无

返回:

无

```
<?php
sp_clear_cache();//无返回值
?>
```

sp_content_page()

> X1.0新增

```
sp_content_page($content,$pagetpl')
```

功能：
对文本内容进行分页处理，返回当前页的内容，和分页html

参数：
`$content` :要处理的文本内容；里面含有百度编辑器的分页标记;
`$pagetpl` :分页模板；默认值{first}{prev}{list}{next}{last}

返回：
类型数组

```
array(
    "content"=>"",//当前页内容
    "page"=>""产生的分页html
);
```

sp_compare_password()

> X2.1.0新增

```
sp_compare_password($password,$password_in_db)
```

功能：

CMF密码比较方法,所有涉及用户密码比较的地方都用这个方法

参数：

`$password`：用户输入的密码
`$password_in_db`:数据库保存的经过加密后的密码串

返回：

类型boolean,密码相同，返回true

使用：

```
$is_right = sp_compare_password('123456','###adfasfasdkfjaslkdjflksajerwqe');
```

sp_get_asset_upload_path()

> X1.6.0新增

```
sp_get_asset_upload_path($file,$style='')
```

功能：

转化数据库保存的文件路径，为可以访问的url

参数：

`$file`：数据库保存的文件路径 `$style`:样式(七牛)

返回：

类型string,文件可以访问的url

使用：

```
$url = sp_get_asset_upload_path('portal/23232.png');
```

sp_get_cmf_settings()

> X1.2新增

```
sp_get_cmf_settings($key);
```

功能：
获取cmf的设置；如果key为空则返回所有设置，如果key不为空，则返回相应key的设置

参数：
`$key` :默认为空，设置的key

返回：
如果key为空则返回所有设置，如果key不为空，则返回相应key的设置

sp_get_comments()

> X1.2新增

```
sp_get_comments($tag,$where);
```

功能：
获取评论；

参数：
`$tag` :查询标签，默认：field:*;limit:0,5;order:createtime desc;
`$where` :查询where数组，按照thinkphp where array格式；

返回：
数组，评论

sp_get_current_admin_id()

> X1.4.0新增

```
sp_get_current_admin_id()
```

功能：
获取当前登录管理员id,同get_current_admin_id()

参数：
无

返回：
类型int,管理员的id

sp_get_current_theme()

> X2.2.0新增

```
sp_get_current_theme($default_theme='')
```

功能：

获取当前主题名

参数：

`$default_theme`：指定的默认主题

返回：

类型string,主题名

使用：

```
$theme = sp_get_current_theme();
```

sp_get_current_user()

```
sp_get_current_user()
```

功能：
获取当前登录用户信息，包括users表里详细信息；

参数：
无

返回：
数组，用户包括users表里详细信息

sp_get_current_userid()

```
sp_get_current_userid()
```

功能：
获取当前登录用户ID

参数：
无

返回：
int,当前登录的用户id,如果未登录返回0

sp_get_favorite_key()

> X1.0新增

```
sp_get_favorite_key($table,$object_id)
```

功能：
用于生成收藏内容用的安全key,收藏时必用

参数：
`$table` :收藏内容所在表，不带表前缀 `$object_id` :收藏内容的id

返回：
类型string,收藏内容用的安全key

sp_get_file_download_url()

> X2.2.0新增

```
sp_get_file_download_url($file,$expires=3600)
```

功能：

获取文件下载链接

参数：

`$file`：数据库保存的文件路径

`$expires`：文件过期时间(七牛)

返回：

类型string,文件下载链接

使用：

```
$url = sp_get_file_download_url('portal/23232.png');
```

sp_get_file_extension()

> X2.2.0新增

```
sp_get_file_extension($filename)
```

功能：

获取文件扩展名

参数：

`$filename`：文件名

返回：

类型string,文件扩展名

使用：

```
$suffix = sp_get_file_extension('23232.png');
```

sp_get_host()

> X1.0新增

```
sp_get_host()
```

功能：
返回带协议的域名

参数：
无

返回：
类型string
带协议的域名,如http://www.thinkcmf.com

sp_get_image_preview_url()

> X2.2.0新增

```
sp_get_image_preview_url($file,$style='watermark')
```

功能：

获取图片预览链接

参数：

`$file`：数据库中保存的文件名 `$style`:样式(七牛)

返回：

类型string,图片预览链接

使用：

```
$url = sp_get_image_preview_url('portal/23232.png');
```

sp_get_image_url()

> X2.2.0新增

```
sp_get_image_url($file,$style='')
```

功能：

转化数据库保存图片的文件路径，为可以访问的url

参数：

`$file` : 数据库中保存的文件名

`$style` :样式(七牛)

返回：

类型string,图片可以访问的url

使用：

```
$url = sp_get_image_url('portal/23232.png');
```

sp_get_menu()

```
sp_get_menu($id,$menu_root_ul_id,$filetpl,$foldertpl,$ul_class,$li_class,$menu_root_ul_class,$showlevel,$dropdown)
```

功能:

生成指定ID的导航

参数:

`$id` :导航id
`$menu_root_ul_id` :菜单根节点ul标签的id属性值
`$filetpl` :没有子菜单的菜单的html模板
`$foldertpl` :有子菜单的菜单的html模板
`$ul_class` :内部ul标签的class属性值
`$li_class` :内部li标签的class属性值
`$menu_root_ul_class` :菜单根节点ul标签的class属性值
`$showlevel` :菜单根节点ul标签的class属性值 `$dropdown` :含有子菜单的li标签的class属性值,用于控制多级菜单的折叠

模板中用法:

```
<php>
    $menu_root_ul_id="main-menu";
    $filetpl="<a href='\$href' target='\$target'>\$label</a>";
    $foldertpl="<a class='dropdown-toggle' href='\$href' target='\$target'>\$label</a>";

    $ul_class="dropdown-menu" ;/*内部ul标签的class属性值*/
    $li_class="" ;/*内部li标签的class属性值*/
    $menu_root_ul_class="nav";/*菜单根节点ul标签的class属性值*/
    $showlevel=6;/*显示菜单的层级*/
    $dropdown='dropdown';/*含有子菜单的li标签的class属性值,用于控制多级菜单的折叠*/
</php>

{:sp_get_menu("main",$menu_root_ul_id,$filetpl,$foldertpl,$ul_class,$li_class,$menu_root_ul_class,$showlevel,$dropdown)}
```

```
<!--生成的代码如下: -->
<ul class="nav">
    <li class="active" id="menu-item-1"><a href="/" target="">首页</a></li>
    <li class="" id="menu-item-11"><a href="" target="">产品与服务</a></li>
    <li class="dropdown" id="menu-item-12">
        <a class="dropdown-toggle" href="" target="">企业新闻</a>
```

```
            <ul class="dropdown-menu">
              <li class="" id="menu-item-11">
                  <a href="" target="">产品与服务</a>
              </li>
            </ul>
    </li>
</ul>
```

sp_get_plugin_class()

> X1.4.0新增

```
sp_get_plugin_class($name)
```

功能：
获取插件类的类名

参数：
`$name`:插件类名

返回：
类型string
如：plugins\Demo\DemoPlugin

sp_get_plugin_config()

> X1.4.0新增

```
sp_get_plugin_config($name)
```

功能：
获取插件配置

参数：
`$name`:插件名

返回：
类型数组，插件配置信息

sp_get_relative_url()

> X1.0新增

```
sp_get_relative_url($url)
```

功能：
获取域名后的地址，如http://thinkcmf.com/news.html,转化后为/news.html

参数：
`$url` :要转化的url

返回：
类型string;
域名后的地址，如http://thinkcmf.com/news.html,转化后为/news.html

sp_get_theme_path()

> X1.1新增

```
sp_get_theme_path()
```

功能：

获取当前模板的地址

参数：

无

返回：

类型string

当前模板的地址,如当前模板是 `simplebootx` ,则返回 `/themes/simplebootx/`

sp_get_user_avatar_url()

> X2.2.0新增

```
sp_get_user_avatar_url($avatar)
```

功能：

获取用户头像地址

参数：

`$avatar`：数据库中保存的用户头像文件路径

返回：

类型string,用户头像地址

使用：

```
$url = sp_get_user_avatar_url('avatart/2231rweqrqwer.png');
echo $url;//输出用户头像地址
```

sp_get_users()

> X1.2新增

```
sp_get_users($tag,$where)
```

功能：

获取符合条件的用户列表；

参数：

`$tag` :查询标签，默认：field:*;limit:0,8;order:create_time desc;
`$where` :查询where数组，按照thinkphp where array格式；

返回：

数组，符合条件的用户列表

使用

```
    //获取最新注册的8个新用户
    $last8_users = sp_get_users('field:*;limit:0,8;order:create_time desc;');

    // 打印用户列表
    print_r($last8_users);
```

sp_getad()

```
sp_getad();
```

功能:

根据广告名称获取广告内容

参数:

`$ad_name` :广告名称

返回:

格式 string,广告内容

示例:

```
<?php
$ad_name='top_ad';
$ad_content=sp_getad('top_ad');    //获取广告内容
echo $ad_content; //输出广告内容
```

模板中用法:

```
<div>
    {:sp_getad('top_ad')} <!--top_ad是后台设置的广告名称-->
</div>
```

sp_getslide()

```
sp_getslide($slide,$limit=5,$order = "listorder ASC")
```

功能:

根据幻灯片标识获取所有幻灯片

参数:

`$slide` :幻灯片标识,后台可以设置 `$limit` :最多显示几张幻灯片 `$order` :按什么字段(slide表的字段)排序

返回

数组,符合条件的幻灯片列表

示例:

```
    $slides=sp_getslide('top_slide');    //top_slide是你在后台创建的幻灯片标识
    print_r($slides);       //打印出获取的结果
```

模板中用法:

```
<php>
    $slides=sp_getslide('top_slide');
    print_r($slides);
</php>
<foreach name="slides" item="vo">
    <!--{$vo.slide_id}幻灯片id-->
    <!--{$vo.slide_cid}幻灯片分类id,对应于slide_cat表里的cid-->
    <!--{$vo.slide_name}幻灯片名称-->
    <!--{$vo.slide_pic}幻灯片图片路径相对于upload文件夹/data/upload/-->
    <!--{$vo.slide_url}幻灯片链接地址-->
    <!--{$vo.slide_des}幻灯片描述-->
    <!--{$vo.slide_content}幻灯片内容-->
    <a href="{$vo.slide_url}" target="_blank" title="{$vo.slide_content}">
        <img src="/data/upload/{$vo.slide_pic}"/>
    </a>
</foreach>
```

sp_is_mobile()

> 1.4.0新增

```
sp_is_mobile()
```

功能：

判断是否为手机访问

参数：

无

返回：

boolean, true:是手机访问,false:不是

使用:

```
    if(sp_is_mobile()){
        echo "我是手机用户呢!";
    }else{
        echo "我不是手机用户呀,那就是电脑用户呀!";
    }
```

模板里使用

```
    <if condition="sp_is_mobile()">
        <div>我是手机用户呢!</div>
    <else/>
        <div>我不是手机用户呀,那就是电脑用户呀!</div>
    </if>
```

sp_getcontent_imgs()

示例：

```
<?php
$content='html格式内容';
$imgs=sp_getcontent_imgs($content);    //获取内容中图片信息
print_r($imginfo);
?>
```

返回字段说明：
`title` :图片的title属性
`alt` :图片的alt属性
`src` :图片的资源路径

sp_is_user_login()

X1.0新增

sp_is_user_login() 功能：

判断用户是否已经登录

参数：

无

返回：

类型布尔 true/false

sp_is_weixin()

> X2.2.0新增

```
sp_is_weixin()
```

功能：

判断是否为微信访问

参数：

无

返回：

类型boolean,true为微信访问

使用：

```
$is_weixin = sp_is_weixin();
```

最后文件编辑时间: 2017-02-03 17:55:31

sp_param_lable()

```
<?php
$result=sp_param_lable('id:2;cat:home');
print_r($result);
?>
```

sp_password()

```php
<?php
$pw='simplewind';
$afpw=sp_password($pw);//加密字符串
echo $afpw;//输出加密后的字符串
?>
```

sp_plugin_url()

> X1.4.0新增

```
sp_plugin_url($url,$param,$domain)
```

功能：
生成访问插件的url

参数：
`$url` : url 格式：插件名://控制器名/方法
`$param` :额外参数，默认为空数组
`$domain` :是否添加域名，默认false

返回： 类型url

模板使用：

```
{:sp_plugin_url('Demo://Index/index',array('id'=>2),true)}
{:sp_plugin_url('Demo://List/index',array('id'=>2))}
```

sp_random_string()

```php
<?php
$random=sp_random_string();//不指定位数，默认为6位
echo $random;
//或者
$random=sp_random_string(8);//指定返回8位随机字符串
echo $random;
?>
```

sp_scan_dir()

> X1.4.0新增

```
sp_scan_dir($pattern,$flags)
```

功能：
替代scan_dir的方法

参数：
`$pattern` :检索模式 搜索模式 *.txt,*.doc; (同glog方法)
`$flags` :返回模式 同glog方法

返回：
类型数组

使用方法：

```
//扫描application目录
$files=sp_scan_dir('application/*');//返回application目录下所有文件
```

最后文件编辑时间: 2017-02-03 17:55:31

sp_send_email()

> X1.0新增

```
sp_send_email($address,$subject,$message)
```

功能：
发送邮件

参数：
`$address` :收件人地址
`$subject` :邮件主题
`$message` :邮件内容

返回：
类型数组，发送状态和信息

```
array(
    "error"=>"1",//有错误
    "message"=>""错误信息
);
array(
    "error"=>"0",//成功发送，无错误
);
```

sp_set_dynamic_config()

> V1.1.1新增

```
<?php
$data=array("URL_HTML_SUFFIX"=>".html");
$result=sp_set_dynamic_config($data);
?>
```

sp_set_option()

> X2.2.0新增

```
sp_set_option($key,$data)
```

功能：

设置系统配置，通用

参数：

`$key`：配置的键名,英文下划线小写,最好加上自己的应用或插件名作为前级 `$data`:配置的值，数组

返回：

类型boolean,true设置成功

使用：

```
$result = sp_set_option('portal_custom_settings',array('test'=>1));
```

sp_update_current_user()

> X1.0新增

```
sp_update_current_user($user)
```

功能：
更新session里当前登录用户的信息

参数：
`$user`:当前登录用户的最新信息

返回：
无

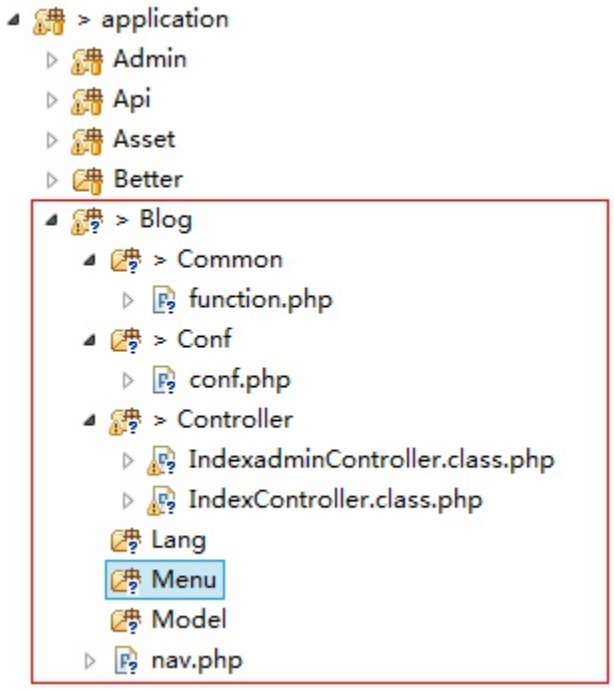
系统扩展

ThinkCMF应用开发流程

以下教程适用于ThinkCMFX系列版本 以Blog模块为例(相关问题已经更新)

1. 在application目录下创建Blog目录 Blog结构:

```
Blog
Controller 控制器目录
Common 函数库
Conf 配置
Lang 多语言包
Menu 后台菜单
Model 模型
nav.php 前台导航
```



Blog结构

同时在application/Common/Conf/config.php里的MODULE_ALLOW_LIST加上你新加的Blog

2. 创建一个前台控件器（Controller） 在模块Controller目录下创建一个IndexController.class.php文件

```
<?php
namespace Blog\Controller;
use Common\Controller\HomebaseController;

class IndexController extends HomebaseController{
   function index(){
     echo "this is blog index !";
   }
```

```
}
```

前台Controller一般都要继承HomebaseController

3. 为前台 IndexController的index方法创建一个模板

   i. 修改IndexController.class.php

```
<?php
namespace Blog\Controller;
use Common\Controller\HomebaseController;

class IndexController extends HomebaseController{
  function index(){
    $this->display(":index");
  }
}
```

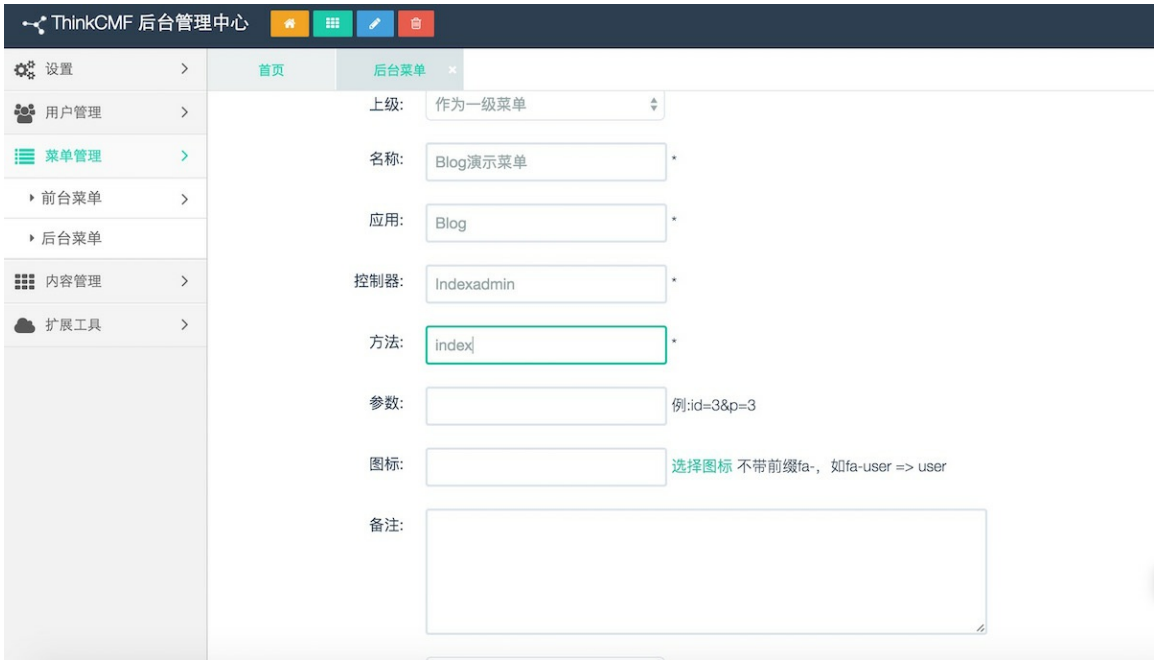
   ii. 在程序前台当前模板目录下创建Blog目录(假设当前模板是simplebootx，当前模板目录./themes/simplebootx) 在Blog目录下创建index.html文件

4. 在浏览器里运行http://你的域名/index.php?g=blog&m=index&a=index
   到此为止，一个应用基本创建完成

5. 创建一个后台控件器（Controller）在模块Controller目录下创建一IndexadminController.class.php文件(注意：这里有文件命名规则，以****adminController.class.php或者Admin****Controller.class.php命名的是后台Controller, 在后台菜单导入时会自动识别）

```
<?php
namespace Blog\Controller;
use Common\Controller\AdminbaseController;

class IndexadminController extends AdminbaseController{

    function index(){
      $this->display();
    }
}
```

后台Controller一般都要继承AdminbaseController 登陆后台，在后台菜单管理添加一个后台菜单



项目，模块，方法一一定要保证和程序代码里的大小写一致性

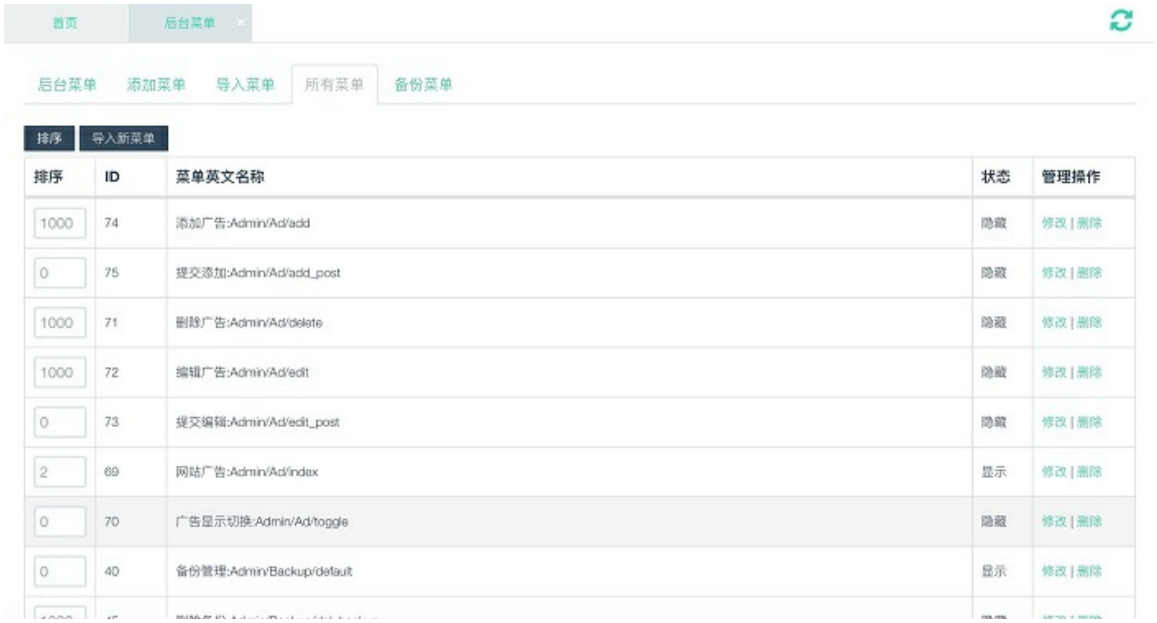
6. 创建index方法模板
   在程序当前后台模板目录下创建 `Blog` 目录(当前后台模板为 `simplebootx` ，当前后台模板目录 `admin/themes/simplebootx`) 创建Indexadmin/index.html（后台模板目录分隔符是自己可以在应用配置里自己定义的， `'TMPL_FILE_DEPR'=> '/'` ，新建应用默认为/）
7. 刷新后台，点击在5步骤里添加的后台菜单
8. Blog应用开发流程完毕

后台隐藏的后台菜单管理功能

应用Controller目录下文件命名规则，以****adminController.class.php或者Admin****Controller.class.php命名的是后台controller, 在后台菜单导入时会自动识别

以下几个功能是debug开启时才会出现的：

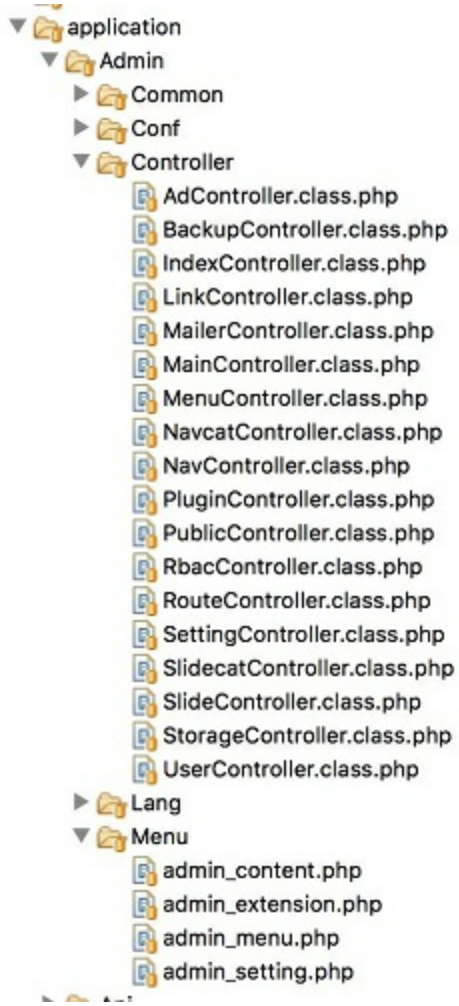
1. 所有菜单



这个是用来管理后台所有菜单的，在这里可以很方便的对那些“未知”的菜单进行编辑

2. 导入新菜单 这个功能就是所有菜单里那个“导入新菜单”的按钮 它用于导入所有你新增的后台方法（可以通过url访问的公共方法）， 这样就不用一个个添加你新加的方法了

3. 备份菜单 你会发现在CMF的每个应用下都会有个Menu目录 当你点击“备份菜单”以后，再到你系统的应用Menu下你会发现会增加很多个文件



这些文件就是对应Admin应用下的每个控制器的方法，里面会记录一些url参数

4. 恢复菜单 那这个功能就是对应于“备份菜单”的，它会将备份菜单 功能生成的文件，恢复到数据库；

> 注意：
> a)这个功能只会增加数据库里没有的菜单，不会删除！
> b)同时你也可以修改备份文件里的菜单名称，在恢复时会更新到数据库；
> c)不会修改你菜单层级关系！

几个重要基类

在ThinkCMFX系列版本的application/Common/Controller下有以下几个基类：

```
|-AppframeController     – ThinkCMF框架控件器基类，继承至Controller
  |-AdminbaseController   – ThinkCMF框架后台控件器基类，继承至AppframeController
  |-HomebaseController     – ThinkCMF框架前台控件器基类，继承至AppframeController
  |-MemberbaseController – ThinkCMF框架会员控件器基类，继承至HomebaseController
```

对于AdminbaseController和HomebaseController它们一个重要的方法display,分别用于管理后台和前台的模板显示，如果你的controller继承了AdminbaseController，在你调用display方法时，它会帮你去找admin/themes目录下的文件，同样HomebaseController也会帮你去找themes目录下的文件，这样就很好的实现了前后台模板的分离。

对于MemberbaseController，继承这个类的Controller会帮你判断会员的相关的操作，比如用户是否已经登陆，用户是否有权限访问此url。

所以想开发应用的同学，一定要在自己创建Controller之前想好你的Controller要完成什么功能，再去让它extends相应的基类。

引入第三方库

ThinkCMF第三方类库在simplewind/Core/Library/Vendor

如果你的第三方类库都放在Vendor目录下面，并且都以.php为类文件后缀，也没用采用命名空间的话，那么可以使用系统内置的vendor函数简化导入。例如，我们来导入二维码操作库phpqrcode,把 phpqrcode放到Vendor目录下面，这个时候phpqrcode主文件的路径就是simplewind/Core/Library/Vendor/phpqrcode/phpqrcode.php，我们使用vendor 方法导入只需要使用：

```
vendor('phpqrcode.phpqrcode');//导入类库
$QRcode = new \QRcode();//实例化，注意加\
```

这样就导入phpqrcode类库了，并完成了实例化

vendor方法也可以支持和import方法一样的基础路径和文件名后缀参数，例如：

```
vendor('phpqrcode.phpqrcode','simplewind/Core/Library/Vendor','.php');
```

vendor方法：

```
vendor($class, $baseUrl, $ext)
```

功能：
快速导入第三方框架类库，所有第三方框架的类库文件统一放到系统的simplewind/Core/Library/Vendor目录下面

参数：
$class:类库,如phpqrcode.phpqrcode
$baseUrl:基础目录，默认simplewind/Core/Library/Vendor
$ext:类库后缀,默认为.php

最后文件编辑时间: 2017-02-03 17:55:31

模板

基础

全局变量

ThinkCMF封装了前台模板开发时常用的一些变量，这些变量是全局的，你在前台模板任何时候都能直接调用：

```
{$site_name}                    /站点名称
{$site_host}                    /站点域名
{$site_root}                    /安装目录
{$site_icp}                     /备案信息
{$site_admin_email}             /管理员邮箱
{$site_tongji}                  /页面统计代码
{$site_seo_title}               /SEO标题
{$site_seo_keywords}            /SEO关键字
{$site_seo_description}         /SEO描述
{$site_copyright}               /版权信息
```

模板结构

ThinkCMF模板目录结构如下：

```
|--themes
  |--simplebootx
    |--Portal
      |--index.html
```

模板开发时我们仅需要在 `themes` 目录下创建一个模板文件夹，假设为 `simplebootx`， `simplebootx` 文件夹下我们需要创建一个应用名称的文件夹，比如 `Portal` 。一般情况下我们还会创建一个 `Public` 文件夹用来存放公共的css，js等资源文件。

```
|--themes
    |--simplebootx                          //模板目录
        |--Comment
            |--comment.html           //评论模板，{:Comments()}中会调用
            |--index.html             //用户中心评论模板（链接:comment/comment/index）
        |--Portal
            |--404.html               //错误模板
            |--article.html           //默认文章页模板
            |--contact.html          //联系我们页面模板，可以后台页面编辑里更改，只需写文件名
contact
            |--index.html            //首页模板
            |--list_masonry.html         //文章列表页瀑布流模板
            |--list.html                //文章列表页模板
            |--page.html            //默认页面模板
            |--product.html        //产品列表页模板，可以在后台分类编辑里设置列表页模板，只需
写文件名product
            |--search.html            //文章搜索页模板
        |--Public                      //模板公共资源目录
        |--User
            |--Profile
                |--avatar.html  //头像编辑界面
                |--bang.html    //第三方账号绑定界面
                |--edit.html    //资料编辑界面
                |--password.html //密码修改界面
            |--active.html            //用户激活模板
            |--center.html         //用户中心模板
            |--disable.html           //用户未激活，重发激活邮件模板
            |--favorite.html       //我的收藏模板
            |--forgot_password.html //忘记密码模板
            |--index.html           //用户主页，公开主页
            |--login.html           //用户登录模板
            |--password_reset.html  //密码重置模板
            |--register.html         //用户注册模板
        |--config.html             //模板配置文件
```

```
        |--jump.html                //系统跳转页模板
        |--error.html                 //系统action错误模板
        |--success.html              //系统action操作成功模板
```

模板注释

js注释：

```
<script>
/* js注释*/
</script>
```

css注释

```
<style>
/*css注释*/
</style>
```

html注释

```
<div>
<!--html注释-->
</div>
```

php标签里的注释

```
<php>
/*一定要用这个方法，用//会在debug关时出问题*/
</php>
```

模板常量

`__ROOT__` :
网站根目录，不带/;

`__WEB_ROOT__` : 网站资源根目录,不带/,如果以前版本用 `__ROOT__` 来定位网站资源,x2.2.0以后最好用这个常量,方便以后cdn切换

`__TMPL__` :
当前模板根目录，带/;
如:前台 simplebootx 模板根目录是/themes/simplebootx/
后台 simplebootx模板根目录是/admin/themes/simplebootx/

`__PUBLIC__` :
public目录,不带/;

前台模板多语言

ThinkCMF前台模板多语言是使用多模板的方式来实现的,如:当前模板是 `simplebootx` ,如果想开启英文前台模板的话,就只要加一个模板名为 `simplebootx_en-us` 模板就可以了;

前台模板多语言实现原理:

ThinkCMF在前台控制器加载模板文件时,会根据当前用户的浏览器语言或者用户指定的语言来加载模板文件,如果是中文用户就加载 simplebootx 里的模板文件,如果是英文用户就加载 simplebootx_en-us 里的模板文件;每个模板里数据调用是独立的,你可以在不同模板里做不同的配置,以调用不同的语言的内容;

为什么选用多模板形式实现前台多语言?

很多用户会疑问,这不是会增加维护的难度吗?为什么不用语言包的形式呢?

维护难度当然会增加,做一个模板和做两个模板是不一样的时间,但你想英文模板和中文模板无论从内容还风格都有可能会不同,一个模板,你要考虑很多布局上兼容的问题,同时,如果想对不同语言的用户做不同的体验上的界面设计,一个模板肯定无法满足,所以多模板形式才是前台多语言最好的选择,当然你在模板里也可以使用应用里设置的语言包.

多语言切换

只要在 url后带上 `?l=语言包` ,如 `?l=en-us` , `?l=zh-cn`

例子

默认语言为zh-cn，可用根据需要开发多个语言包,如后台设置模板为simplebootx，在themes目录下simplebootx目录为zh-cn语言模板，simplebootx_en-us为en-us语言模板，simplebootx_zh-tw对应的就是zh-tw语言包，simplebootx_mobile则为移动版模板，移动模板+"_language"则为移动版对应语言包，如simplebootx有三个语言版本，则对应目录为 y `simplebootx // PC中文默认模板` `simplebootx_en-us // PC端zh-us语言模板` `simplebootx_zh-tw // PC端zh-tw语言模板` `simplebootx_mobile // 移动端中文默认模板` `simplebootx_mobile_en-us // 移动端en-us语言模板` `simplebootx_mobile_zh-tw // 移动端zh-tw语言模板`

> 注:感谢 `@gai871013`

变量输出

请参考：http://www.kancloud.cn/manual/thinkphp/1794

使用函数

请参考：http://www.kancloud.cn/manual/thinkphp/1796

默认值输出

请参考：http://www.kancloud.cn/manual/thinkphp/1797

使用运算符

请参考：http://www.kancloud.cn/manual/thinkphp/1798

三元运算

请参考：http://www.kancloud.cn/manual/thinkphp/1802

包含文件

tc_include

tc_include是前台模板包含文件的方法，用于替换原来的include的标签，实现更多功能

用法：

```
<tc_include file="Public:nav"/> <!--加载的是当前模板的Public/nav.html-->
<tc_include file="Public/nav"/> <!--加载的是当前模板的Public/nav.html-->
<tc_include file="Portal/sidebar"/> <!--加载的是当前模板的Portal/sidebar.html-->
<tc_include file="User/Profile/nav"/> <!--加载的是当前模板的User/Profile/nav.html-->
```

原样输出

请参考：http://www.kancloud.cn/manual/thinkphp/1820

模板标签

作为内容管理框架，ThinkCMF允许app开发者自定义任意标签，ThinkCMF标签使用花括号作为定界符，如{$name}。

ThinkCMF系统内置的标签有：

`<php></php>` 用来在模板中执行php的代码，示例代码：

```
<php>echo "这个是在模板中执行的php程序"</php>
```

`<foreach></foreach>` 遍历标签，示例代码：

```
<php>$title=array("简介","产品","新闻");</php>          //遍历一个一维数组
<foreach name='title' item='vo'>
<div style="样式">
<a href="#">{$vo}</a>
</div>
</foreach>

<php>                          //遍历一个二维数组
$article = array
(
  "0"=>array
  (
  "title"=>"title1",
  "content"=>"content1",
  "date"=>"date1",
  ),
  "1"=>array
  (
  "title"=>"title2",
  "content"=>"content2",
  "date"=>"date2",
  )
);
</php>
<foreach name='article' item='vo'>
<div style="样式">
<a href="#">{$vo.title}</a>
<p>{$vo.content}</p>
</div>
</foreach>
```

`<volist></volist>` 按条件遍历，示例代码：

```
<php>                          //遍历一个二维数组
```

```
$article = array
(
  "0"=>array
  (
  "title"=>"title1",
  "content"=>"content1",
  "date"=>"date1",
  ),
  "1"=>array
  (
  "title"=>"title2",
  "content"=>"content2",
  "date"=>"date2",
  )
);
</php>
<volist name='article' id='vo'>
<div style="样式">
<a href="#">{$vo.title}</a>
<p>{$vo.content}</p>
</div>
</volist>
```

`<for></for>` 循环标签，实例代码：

```
<for start="开始值" end="结束值" name="循环变量名" >
 循环语句。。。
 </for>
```

tc_include

tc_include是前台模板包含文件的方法，用于替换原来的include的标签，实现更多功能

用法：

```
<tc_include file="Public:nav"/> <!--加载的是当前模板的Public/nav.html-->
<tc_include file="Public/nav"/> <!--加载的是当前模板的Public/nav.html-->
<tc_include file="Portal/sidebar"/> <!--加载的是当前模板的Portal/sidebar.html-->
<tc_include file="User/Profile/nav"/> <!--加载的是当前模板的User/Profile/nav.html-->
```

foreach

foreach标签

属性
`name` :表示数据源;
`item` :表示循环变量;

```
<foreach name="list" item="vo" >
   {$vo.name} {$key}
</foreach>
```

更改属性key的变量名：

```
<foreach name="list" item="vo" key="k">
   {$vo.name} {$k}
</foreach>
```

volist

请参考：http://www.kancloud.cn/manual/thinkphp/1805

php

请参考：http://www.kancloud.cn/manual/thinkphp/1819

if else

请参考：http://www.kancloud.cn/manual/thinkphp/1811

for

请参考：http://www.kancloud.cn/manual/thinkphp/1807

switch

请参考：http://www.kancloud.cn/manual/thinkphp/1808

比较标签

请参考：http://www.kancloud.cn/manual/thinkphp/1809

范围判断标签

请参考：http://www.kancloud.cn/manual/thinkphp/1810

Present标签

请参考：http://www.kancloud.cn/manual/thinkphp/1812

Empty标签

请参考：http://www.kancloud.cn/manual/thinkphp/1813

Defined标签

请参考：http://www.kancloud.cn/manual/thinkphp/1814

Assign标签

请参考：http://www.kancloud.cn/manual/thinkphp/1815

Define标签

请参考：http://www.kancloud.cn/manual/thinkphp/1816

标签嵌套

请参考：http://www.kancloud.cn/manual/thinkphp/1817

前端组件

js-count-btn

前台数量操作组件

```
<a href="这里定义自己的url" class="js-count-btn">
    <i class="fa fa-thumbs-up"></i>
    <span class="count">0</span>
</a>
```

功能：
加上js-count-btn类名的a标签为可以实现数量增加的 ajax操作；ajax执行成功返回后对其内类名包含count的标签进行数量加1操作; ajax 请求的 url 为 a标签的 href 属性；

js-favorite-btn

前台收藏组件，其它应用可以公用

```
<a href="{:U('user/favorite/do_favorite')}" class="js-favorite-btn" data-title="收藏的内容标题" data-url="收藏的内容的url" data-key="{:sp_get_favorite_key('收藏内容所在表',收藏内容的id)}"></a>

<!--如文章收藏：-->
<a href="{:U('user/favorite/do_favorite',array('id'=>$object_id))}" class="js-favorite-btn" data-title="{$post_title}" data-url="{:U('article/index',array('id'=>$tid))}" data-key="{:sp_get_favorite_key('posts',$object_id)}">
  <i class="fa fa-star-o"></i>
</a>
```

```
sp_get_favorite_key($table,$object_id)
```

`$table`：收藏内容所在的表，不带表前缀的表名称，如cmf_posts应该改为“posts”;
`$object_id`：收藏内容的id:

a标签属性说明

data-title:收藏的内容标题；

data-url:收藏内容的url；

data-key:安全key，用sp_get_favorite_key方法生成，防止有人提交错误数据；

最后文件编辑时间: 2017-02-03 17:55:31

js-ajax-dialog-btn

ajax 对话框组件

```
<a href="你的url" class="js-ajax-dialog-btn" data-msg="确定还原吗？">还原</a>
```

功能：
加上js-ajax-dialog-btn类名的a标签在单击后，会出现一个对话框，提示用户一些信息(信息内容为data-msg的属性值)，用户确定后会进行一个 ajax 请求，请求的 url为

href 属性的值，请求成功返回后，会刷新当前界面。

最后文件编辑时间: 2017-02-03 17:55:31

js-ajax-delete

ajax 删除组件

```
<a href="你的删除url" class="js-ajax-delete" data-msg="确定删除它吗？">删除</a>
```

功能：

加上js-ajax-delete类名的a标签在单击后，会出现一个对话框，提示用户一些信息(信息内容为data-msg的属性值)，用户确定后会进行一个 ajax 请求，请求的 url为

href 属性的值，请求成功返回后，会刷新当前界面。

js-date

日期选择组件

```
<input class="js-date" type="text" id="input-birthday" placeholder="2013-01-04" name="b
irthday">
```

功能：

加上js-date类名的text input标签在用户输入时间时，会弹出一个日历让用户选择日期

最后文件编辑时间: 2017-02-03 17:55:31

js-datetime

时间选择组件

```
<input class="js-datetime" type="text" id="input-datetime" placeholder="2013-01-04 09:20
" name="datetime">
```

功能：

加上js-datetime类名的text input标签在用户输入时间时，会弹出一个日历让用户选择时间

js-ajax-form

ajax表单组件

```
<form class="js-ajax-form" action="{:U('user/login/dologin')}" method="post">
    <label for="input_username">账号</label>
    <input type="text" id="input_username" name="username" placeholder="请输入用户名或者邮箱" class="span3">

    <label for="input_password">密码</label>
    <input type="password" id="input_password" name="password" placeholder="请输入密码" class="span3">

    <label for="input_verify">验证码</label>
    <input type="text" id="input_verify" name="verify"  placeholder="请输入验证码" class="span3">
    {:sp_verifycode_img('length=4&font_size=15&width=100&height=35&charset=1234567890')}

    <button class="btn btn-primary js-ajax-submit" type="submit" data-wait="1500">确定</button>
</form>
```

功能：

加上类名为 js-ajax-form 的 form标签，配合类名为 js-ajax-submit 的提交按钮，在用户单击提交按钮时，会以 ajax 的方式提交表单，提交的地址为 form 的 action属性，提交方法为 form的 method 属性，凡是在 此form 里且有 name 属性的表单元素都会被提交;

提交成功返回后，如果返回结果中有 referer 字段，页面会跳转到 referer 表示的地址。如果 没有 referer 或者其为空，则会刷新当前页，或者等待一定时间(data-wait的值，单位 ms)

后，再刷新当前页。

公共模板

菜单导航制作

```
sp_get_menu($id,$menu_root_ul_id,$filetpl,$foldertpl,$ul_class,$li_class,$menu_root_ul_class,$showlevel,$dropdown)
```

功能:

生成指定ID的导航

参数:

`$id` :导航id
`$menu_root_ul_id` :菜单根节点ul标签的id属性值
`$filetpl` :没有子菜单的菜单的html模板
`$foldertpl` :有子菜单的菜单的html模板
`$ul_class` :内部ul标签的class属性值
`$li_class` :内部li标签的class属性值
`$menu_root_ul_class` :菜单根节点ul标签的class属性值
`$showlevel` :菜单根节点ul标签的class属性值 `$dropdown` :含有子菜单的li标签的class属性值,用于控制多级菜单的折叠

模板中用法:

```
<php>
    $menu_root_ul_id="main-menu";
    $filetpl="<a href='\$href' target='\$target'>\$label</a>";
    $foldertpl="<a class='dropdown-toggle' href='\$href' target='\$target'>\$label</a>";

    $ul_class="dropdown-menu" ;/*内部ul标签的class属性值*/
    $li_class="" ;/*内部li标签的class属性值*/
    $menu_root_ul_class="nav";/*菜单根节点ul标签的class属性值*/
    $showlevel=6;/*显示菜单的层级*/
    $dropdown='dropdown';/*含有子菜单的li标签的class属性值,用于控制多级菜单的折叠*/
</php>

{:sp_get_menu("main",$menu_root_ul_id,$filetpl,$foldertpl,$ul_class,$li_class,$menu_root_ul_class,$showlevel,$dropdown)}
```

```
<!--生成的代码如下: -->
<ul class="nav">
    <li class="active" id="menu-item-1"><a href="/" target="">首页</a></li>
    <li class="" id="menu-item-11"><a href="" target="">产品与服务</a></li>
    <li class="dropdown" id="menu-item-12">
        <a class="dropdown-toggle" href="" target="">企业新闻</a>
```

```
            <ul class="dropdown-menu">
              <li class="" id="menu-item-11">
                  <a href="" target="">产品与服务</a>
              </li>
            </ul>
    </li>
</ul>
```

幻灯片制作

在后台扩展工具》幻灯片分类 添加分类标识为"portal_index"的分类，然后在此分类添加幻灯片；

```
cat_name：幻灯片类型名称
cat_idname：幻灯片标示
slide_name：幻灯片名称
slide_pic：幻灯片图片地址
slide_url：幻灯片URL
slide_des：幻灯片描述
slide_content：幻灯片内容

<php>
$home_slides=sp_getslide("portal_index");
</php>

<foreach name="home_slides" item="vo">
{$vo.slide_name}
<a href="{$vo.slide_url}">
    <img src="{:sp_get_asset_upload_path($vo['slide_pic'])}" alt="">
</a>
</foreach>
```

广告位制作

在文章底部加一个广告：

到后台扩展工具->网站广告->添加广告名称 为'portal_article_bottom'的广告，同时加上广告代码；

模板里代码如下：

```
<div>
    {:sp_getad("portal_article_bottom")}
</div>
```

友情链接制作

```
sp_getlinks();
```

功能:

获取所有友情连接

参数:

无

返回

数组,友情链接列表

示例:

```
$links=sp_getlinks();         /*获取友情链接*/
print_r($links);             /*打印出获取的结果*/
```

模板中用法:

```
<php>
    $links=sp_getlinks();
</php>
<foreach name="links" item="vo">
    {$vo.link_url} <!--链接地址-->
    {$vo.link_name} <!--链接名称-->
    {$vo.link_image} <!--链接图片,一般是网站logo图片地址-->
    {$vo.link_target} <!--打开方式-->
    {$vo.link_description} <!--描述-->

    /*常见用法*/
    <a href="{$vo.link_url}" target="{$vo.link_target}">
        <notempty name="vo.link_image">
                <img src="{:sp_get_image_url($vo['link_image'])}"><!-- 链接图片 -->
        </notempty>
        {$vo.link_name}
    </a>
</foreach>
```

添加留言控件

在模板中加入以下代码：

```
<form class="js-ajax-form" role="form" method="post" action="{:U('api/guestbook/addmsg')}">
    <label>姓名</label>
    <input type="text" class="span3" placeholder="Your name" name="full_name">
    <label>邮箱</label>
    <input type="email" class="span3" placeholder="Email address" name="email">
    <label>内容</label>
    <textarea class="span6" placeholder="Write message here..." name="msg"></textarea>
    <label>验证码</label>
    <input type="text" class="span3" placeholder="please enter the code" name="verify" autocomplete="off">
   {:sp_verifycode_img('length=4&font_size=20&width=238&height=34&font_color=&background=','style="cursor: pointer;vertical-align:top;" title="点击获取"')}
   <button type="submit" class="btn btn-primary js-ajax-submit">发送留言</button>
</form>
```

如何收藏

收藏组件，其它应用可以公用

```
<a href="{:U('user/favorite/do_favorite')}" class="js-favorite-btn" data-title="收藏的内容标题" data-url="收藏的内容的url" data-key="{:sp_get_favorite_key('收藏内容所在表',收藏内容的id)}"></a>

<!--如文章收藏：-->
<a href="{:U('user/favorite/do_favorite',array('id'=>$object_id))}" class="js-favorite-btn" data-title="{$post_title}" data-url="{:U('article/index',array('id'=>$tid))}" data-key="{:sp_get_favorite_key('posts',$object_id)}">
  <i class="fa fa-star-o"></i>
</a>
```

```
sp_get_favorite_key($table,$object_id)
```

参数1：收藏内容所在的表，不带表前缀的表名称，如cmf_posts应该改为“posts”;

参数2：收藏内容的id:

data-title:收藏的内容标题；

data-url:收藏内容的url；

data-key:安全key，用sp_get_favorite_key方法生成，防止有人提交错误数据；

最后文件编辑时间: 2017-02-03 17:55:31

点赞组件

点赞组件

```
<a href="{:U('article/do_like',array('id'=>$object_id))}" class="js-count-btn"><i class=
"fa fa-thumbs-up"></i><span class="count">{$post_like}</span></a>
```

分解组件：

```
<!-- 点赞链接 须传入文章id -->
href="{:U('article/do_like',array('id'=>$vo['object_id']))}"
```

js类名：

js-count-btn

此组件可复用，只要链接是数量操作，开发者可以在自己的应用里添加相应操作

最新评论组件制作

```
<php>$last_comments=sp_get_comments("field:*;limit:0,5;order:createtime desc;");</php>
<foreach name="last_comments" item="vo">
    <div class="comment-ranking-inner">
    <i class="fa fa-comment"></i>
    <a href="{:U('user/index/index',array('id'=>$vo['uid']))}">{$vo.full_name}:</a>
    <span>{$vo.content}</span>
    <a href="__ROOT__/{$vo.url}#comment{$vo.id}">查看原文</a>
    <span class="comment-time">{:date('m月d日  H:i',strtotime($vo['createtime']))}
    </span>
    </div>
</foreach>
```

最新加入组件制作

```
<php>$last_users=sp_get_users("field:*;limit:0,8;order:create_time desc;");</php>
<ul class="list-unstyled tc-photos margin-bottom-30">
<foreach name="last_users" item="vo">
    <li>
        <a href="{:U('user/index/index',array('id'=>$vo['id']))}">
        <img alt="" src="{:U('user/public/avatar',array('id'=>$vo['id']))}">
        </a>
    </li>
</foreach>
</ul>
```

本站用户登录模板制作

模板文件：User/login.html

```
<form class="js-ajax-form" action="{:U('user/login/dologin')}" method="post">
    <label for="input_username">账号</label>
    <input type="text" id="input_username" name="username" placeholder="请输入用户名或者邮箱" class="span3">

    <label for="input_password">密码</label>
    <input type="password" id="input_password" name="password" placeholder="请输入密码" class="span3">

    <label for="input_verify">验证码</label>
    <input type="text" id="input_verify" name="verify"  placeholder="请输入验证码" class="span3">
    {:sp_verifycode_img('length=4&font_size=15&width=100&height=35&charset=1234567890')}

    <button class="btn btn-primary js-ajax-submit" type="submit">确定</button>
</form>
```

本站用户注册模板制作

模板文件：User/register.html

```
<form class="form-horizontal js-ajax-form" action="{:U('user/register/doregister')}" me
thod="post">
    <label class="control-label" for="input_username">账号</label>
    <input type="text" id="input_username" name="username" placeholder="请输入账号" class
="span3">

    <label class="control-label" for="input_email">邮箱</label>
    <input type="text" id="input_email" name="email" placeholder="请输入邮箱" class="spa
n3">

    <label class="control-label" for="input_password">密码</label>
    <input type="password" id="input_password" name="password" placeholder="请输入密码"
class="span3">

    <label class="control-label" for="input_repassword">重复密码</label>
    <input type="password" id="input_repassword" name="repassword" placeholder="请输入
重复密码" class="span3">

    <label class="control-label" for="input_verify">验证码</label>
    <input type="text" id="input_verify" name="verify" placeholder="请输入验证码" class="
span3">
    {:sp_verifycode_img('length=4&font_size=15&width=100&height=35&charset=1234567890')
}

    <button class="btn btn-primary js-ajax-form" type="submit" data-wait="1500">确定注册
</button>
</form>
```

忘记密码模板制作

模板文件：User/forgot_password.html

```
<form class="form-horizontal js-ajax-form" action="{:U('user/login/doforgot_password')}"
 method="post">
    <label class="control-label" for="input_email">注册邮箱</label>
    <input type="email" id="input_email" name="email" class="span3">

    <label class="control-label" for="input_verify">验证码</label>
    <input type="text" id="input_verify" name="verify" class="span3">
    {:sp_verifycode_img('length=4&font_size=15&width=100&height=35&charset=1234567890')
}
    <label class="control-label" for="input_repassword"></label>

    <button class="btn btn-primary js-ajax-form" type="submit">确定</button>
</form>
```

密码重置模板制作

模板文件：User/password_reset.html

```
<form class="form-horizontal js-ajax-form" action="{:U('user/login/doforgot_password')}"
 method="post">
    <label class="control-label" for="input_email">注册邮箱</label>
    <input type="email" id="input_email" name="email" class="span3">

    <label class="control-label" for="input_verify">验证码</label>
    <input type="text" id="input_verify" name="verify" class="span3">
    {:sp_verifycode_img('length=4&font_size=15&width=100&height=35&charset=1234567890')
}

    <button class="btn btn-primary js-ajax-form" type="submit">确定</button>
</form>
```

评论组件

显示评论组件：

```
{:Comments("posts",$object_id)}
<!-- 评论文章表里的某个id为$object_id的文章-->
```

Comments方法说明：

`参数1`：评论内容所在的表，不带表前缀的表名称，如cmf_posts应该改为“posts”;

`参数2`：评论内容的id:

`参数3`：数组，目前支持tpl参数，如array("tpl"=>"comment_custom"),这样设置就会加载模板目录Comment/coment_custom.html这个模板。

评论模板：
默认评论模板文件：Comment/comment.html

```
<br>
<h3>评论</h3>
<div class="comment-area">

    <hr>
    <form class="form-horizontal comment-form" action="{:U('comment/comment/post')}" me
thod="post">
      <div class="control-group">
          <div class="comment-postbox-wraper">
              <textarea class="form-control comment-postbox" placeholder="Write your co
mment here" style="min-height:90px;"  name="content"></textarea>
          </div>
      </div>

      <div class="control-group">
          <button type="submit" class="btn pull-right btn-primary js-ajax-submit"><i cl
ass="fa fa-comment-o"></i> 发表评论</button>
      </div>

      <input type="hidden" name="post_table" value="{:sp_authencode('posts')}"/>
      <input type="hidden" name="post_id" value="{$post_id}"/>
      <input type="hidden" name="to_uid" value="0"/>
      <input type="hidden" name="parentid" value="0"/>
    </form>

    <script class="comment-tpl" type="text/html">
        <div class="comment" data-username="{$user.user_nicename}" data-uid="{$user.id}"
>
          <a class="pull-left" href="{:U('user/index/index',array('id'=>$user['id']))}">
```

```
            <img class="media-object avatar" src="{:U('user/public/avatar',array('id'=>
$user['id']))}" class="headicon"/>
          </a>
          <div class="comment-body">
            <div class="comment-content"><a href="{:U('user/index/index',array('id'=>$u
ser['id']))}">{$user.user_nicename}</a>:<span class="content"></span></div>
            <div><span class="time">刚刚</span> <a onclick="comment_reply(this);" href=
"javascript:;"><i class="fa fa-comment"></i></a></div>
          </div>
          <div class="clearfix"></div>
        </div>
    </script>

    <script class="comment-reply-box-tpl" type="text/html">
        <div class="comment-reply-submit">
                    <div class="comment-reply-box">
                        <input type="text" class="textbox" placeholder="回复">
                    </div>
                    <button class="btn pull-right" onclick="comment_submit(this);">回复<
/button>
                </div>
    </script>


    <hr>
    <div class="comments">
    <foreach name="comments" item="vo">
         <div class="comment" data-id="{$vo.id}" data-uid="{$vo.uid}" data-username="{$
vo.full_name}"  id="comment{$vo.id}">
          <a class="pull-left" href="{:U('user/index/index',array('id'=>$vo['uid']))}">
            <img class="media-object avatar" src="{:U('user/public/avatar',array('id'=>
$vo['uid']))}" class="headicon"/>
          </a>
          <div class="comment-body">
            <div class="comment-content"><a href="{:U('user/index/index',array('id'=>$v
o['uid']))}">{$vo.full_name}</a>:<span>{$vo.content}</span></div>
            <div><span class="time">{:date('m月d日  H:i',strtotime($vo['createtime']))}
</span> <a onclick="comment_reply(this);" href="javascript:;"><i class="fa fa-comment">
</i></a></div>

            <if condition="!empty($vo['children'])">
                <foreach name="vo.children" item="voo">
                    <div class="comment" data-id="{$voo.id}" data-uid="{$voo.uid}" data
-username="{$voo.full_name}" id="comment{$voo.id}">
                      <a class="pull-left" href="{:U('user/index/index',array('id'=>$vo
o['uid']))}">
                        <img class="media-object avatar" src="{:U('user/public/avatar',
array('id'=>$voo['uid']))}" class="headicon"/>
                      </a>
                      <div class="comment-body">
```

```
                        <div class="comment-content"><a href="{:U('user/index/index',ar
ray('id'=>$voo['uid']))}">{$voo.full_name}</a>:<span>回复 <a href="{:U('user/index/index
',array('id'=>$voo['to_uid']))}">{$parent_comments[$voo['parentid']]['full_name']}</a>
{$voo.content}</span></div>
                        <div><span class="time">{:date('m月d日  H:i',strtotime($voo['cr
eatetime']))}</span> <a onclick="comment_reply(this);" href="javascript:;"><i class="fa
 fa-comment"></i></a></div>
                    </div>
                    <div class="clearfix"></div>
                </div>
            </foreach>

        </if>


        </div>
        <div class="clearfix"></div>
      </div>
    </foreach>
    </div>

</div>
```

进阶

七牛图片处理

七牛提供了强大的图片处理功能,ThinkCMF内部提供了七牛良好的支持,只要在后台"文件存储"里进行简单的设置就可以把全站的图片上传七牛了,前台使用七牛的强大api就可以对图片进行各种处理,如放大缩小,缩略图,加水印等.

ThinkCMF内部保存的文件路径是相对路径,假如你在七牛空间有个图片访问地址是 `http://78re52.com1.z0.glb.clouddn.com/resource/gogopher.jpg` ,那么你七牛空间的根目录就是 `http://78re52.com1.z0.glb.clouddn.com/` ,而在 CMF 的数据库里保存的是 `resource/gogopher.jpg` ,在前台模板你只要 sp_get_image_url()就可以获取到图片在七牛的访问地址了.具体用法如下:

```
<img src="{:sp_get_image_url('resource/gogopher.jpg')}"/>
<!--这里输出的img地址就是http://78re52.com1.z0.glb.clouddn.com/resource/gogopher.jpg-->
```

> 使用七牛的api生成图片300x300的缩略图

```
<img src="http://78re52.com1.z0.glb.clouddn.com/resource/gogopher.jpg?imageView2/1/w/300/h/300" />
```

> 使用CMF标签生成七牛图片300x300的缩略图

```
<img src="{:sp_get_image_url('resource/gogopher.jpg','?imageView2/1/w/300/h/300')}" />
```

所以图片处理的关键还是七牛的 api,ThinkCMF只是集成了七牛的用法,sp_get_image_url这个方法就是把数据库里存的相对图片路径转化为可以访问的路径;

```
sp_get_image_url($file,$style)
```

参数:

`$file` :数据库中保存的图片路径,是相对路径;

`$style` :图片显示样式,这个参数只在文件存储类型是七牛时才有用

在模板里显示一个400x300的图片缩略图

```
<img src="{:sp_get_image_url('resource/gogopher.jpg','?imageView2/1/w/400/h/300')}" />
```

如果你想生成其它尺寸的图片,只要改变七牛 api 里的 `w` 和 `h` 的大小就可以了!

这么牛的图片处理方式,只要把在cmf后台做一下设置就可以了,同时七牛也提供了免费的使用额度,对于流量小的企业站基本够用了,同时也保证了网站流畅性,ThinkCMF用户也可以享受ThinkCMF七牛专用优惠码,想再打个折就到ThinkCMF官网首页找优惠码吧!

门户应用

基础

主程序结构

```
|--Portal
    |--Common
        |--funtion.php                 //应用函数库
    |--Conf
        |--config.php                 // 应用配置文件
    |--Controller
        |--ArticleController.class.php   //文章内页控制器
        |--ListController.class.php      //文章列表控制器
        |--IndexController.class.php     //应用首页控制器
        |--PageController.class.php    //页面控件器
    |--Lang                             //多语言文件
    |--Menu                            //1.0.4版本目录，默认为空用来存放备份的此应用的后台菜
单
    |--nav.php                     //1.0.4新加，用于开发者控制返回前台菜单的文件
```

模板结构

```
|--themes
    |--simplebootx                          //模板目录
        |--Comment
            |--comment.html           //评论模板，{:Comments()}中会调用
            |--index.html            //用户中心评论模板（链接:comment/comment/index）
        |--Portal
            |--404.html              //错误模板
            |--article.html          //默认文章页模板
            |--contact.html         //联系我们页面模板，可以后台页面编辑里更改，只需写文件名
contact
            |--index.html           //首页模板
            |--list_masonry.html         //文章列表页瀑布流模板
            |--list.html               //文章列表页模板
            |--page.html            //默认页面模板
            |--product.html        //产品列表页模板，可以在后台分类编辑里设置列表页模板，只需
写文件名product
            |--search.html           //文章搜索页模板
        |--Public                     //模板公共资源目录
        |--User
            |--Profile
                |--avatar.html  //头像编辑界面
                |--bang.html    //第三方账号绑定界面
                |--edit.html    //资料编辑界面
                |--password.html //密码修改界面
            |--active.html            //用户激活模板
            |--center.html        //用户中心模板
            |--disable.html           //用户未激活，重发激活邮件模板
            |--favorite.html      //我的收藏模板
            |--forgot_password.html //忘记密码模板
            |--index.html          //用户主页，公开主页
            |--login.html          //用户登录模板
            |--password_reset.html  //密码重置模板
            |--register.html        //用户注册模板
        |--config.html            //模板配置文件
        |--jump.html              //系统跳转页模板
        |--error.html               //系统action错误模板
        |--succes.html            //系统action操作成功模板
```

函数库

sp_get_all_child_terms()

> X2.2.0新增

```
sp_get_all_child_terms($term_id)
```

功能：

指定分类下的所有子分类

参数：

`$term_id`：分类id

返回：

类型array,指定分类下的所有子分类

使用：

```
$terms = sp_get_all_child_terms(1);
```

sp_get_breadcrumb()

> X2.2.0新增

```
sp_get_breadcrumb($term_id)
```

功能：

获取面包屑数据

参数：

`$term_id`：当前文章所在分类id,或者当前分类的id

返回：

类型array,面包屑数据

使用：

```
$breadcrumb = sp_get_breadcrumb(3);
```

sp_sql_posts()

```
sp_sql_posts($tag,$where)
```

功能：
查询文章列表，不做分页

参数：
`$tag` :查询语句（见$tag规则）
`$where` :查询条件，（暂只支持数组），格式和thinkphp where方法一样；

返回： array 文章列表

示例：

```
<?php
$tag='cid:6;field:post_title,post_content;order:listorder asc';
$posts=sp_sql_posts($tag);
print_r($posts);
$smeta=json_decode($vo['smeta'],true); //smeta处理方法，将其转化为数组
?>
```

`$tag` 规则：

cid 分类id；

field 需要取出的内容，默认取出所有信息；order排序方式，可根据任何取出的字段排序，默认为按发布时间排序。

```
field可选参数：
term_id      文章分类id
post_author      文章作者id，后台管理员，对应于表users里的ID;
post_keywords
post_date      文章发布日期 格式2014-01-01 00:00:00
post_content      文章内容
post_title      文章标题
post_excerpt      文章摘要
post_modified      文章更新日期 格式2014-01-01 00:00:00
smeta      文章扩展属性，以json格式保存，如属性thumb文章缩略图
user_nicename      管理员昵称
user_email      管理员邮箱
```

模板中用法：

```
<php>
```

```
$posts=sp_sql_posts('cid:6;field:post_title,post_content;order:listorder asc');
</php>

<foreach name="posts" item="vo">  /* 遍历数组 */
    {$vo.term_id }<br>
    {$vo.post_author }<br>
    {$vo.post_keywords }<br>
    {$vo.post_date }<br>
    {$vo.post_content }<br>
    {$vo.post_title }<br>
    {$vo.post_excerpt }<br>
    {$vo.post_modified}<br>
    {$vo.user_nicename }<br>
    {$vo.user_email }<br>
    <php>
    $smeta=json_decode($vo['smeta'],true);/* 把smeta转化成数组 */
    </php>
    <img src="{:sp_get_asset_upload_path($smeta['thumb'])}"/>
</foreach>
```

sp_sql_post()

示例：

```
<?php
$id=1000;    //posts表里的文章id
$posts=sp_sql_post($article,'field:post_title,post_content;');
print_r($posts);
//field 的可选参数已在sp_sql_posts()中说明
//smeta 处理方法已在sp_sql_posts()中说明
```

sp_get_term()

```
sp_get_term($term_id)
```

功能：
返回指定分类

参数：
`$term_id` :分类id

返回：
类型数组,符合条件的分类

示例：

```
<?php
 $term_id=1;
 $term=sp_get_term($term_id ); //获取分类信息
 print_r($term); //打印出分类信息
?>
```

返回数组说明：

`term_id` :分类id
`name` :分类名称
`taxonomy` :分类的类型，用字符串表示，article表示文章
`description` : 分类描述
`parent` :分类父级id,terms表中的term_id
`seo_title`
`seo_keywords`
`seo_description`
`list_tpl` :分类列表页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为list
`one_tpl` :分类单文章页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为article

sp_get_terms()

```
sp_get_terms($tag)
```

功能：
返回符合条件的所有分类

参数：
`$tag` :查询标签，以字符串方式传入,
例："ids:1,2;field:term_id,name,description,seo_title;limit:0,8;order:path asc,listorder desc;where:term_id>0;"

```
ids:调用指定id的一个或多个数据,如 1,2,3
field:调用terms表里的指定字段,如(term_id,name...) 默认全部，用*代表全部
limit:数据条数,默认值为10,可以指定从第几条开始,如3,8(表示共调用8条,从第3条开始)
order:排序方式，如：path desc,listorder asc;
where:查询条件，字符串形式，和sql语句一样
```

返回：
类型数组,符合条件的所有分类

示例：

```
<php>
    $tag='ids:1,2;field:post_date,post_content;limit:10;order:post_date DESC;';
    /*
    $tag规则：ids分类id,以逗号隔开；field需要取出的内容，默认取出所有信息；order排序方式，可根据任何取出的字段排序，默认为按发布时间排序。
    $tag这是一个格式化的字符串，这种方式最大的好处就是方便日后扩展，它格式如《field+冒号+field支持的值，以英文逗号隔开+分号》
    */

    /*关于此函数的field:
    .它的可选值是：
   term_id 分类id
   name 分类名称
   taxonomy 分类的类型，用字符串表示，article表示文章
   description 分类描述
   parent 分类父级id,terms表中的term_id
   path 用于无限级分类的path,如0-1-29
   seo_title
   seo_keywords
   seo_description
   list_tpl 分类列表页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为list
   one_tpl 分类单文章页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为article
   listorder 分类排序
```

```
    status 分类状态0已回收，1正常
     */
     $terms=sp_get_terms($tag);
</php>
<foreach name="terms" item="vo">

    {$vo.name }<br>
    {$vo.taxonomy }<br>
    {$vo.seo_title }<br>
    {$vo.seo_keywords }<br>
    {$vo.seo_description}<br>

</foreach>
```

sp_get_child_terms()

```
sp_get_child_terms($term_id)
```

功能：
返回指定分类下的子分类

参数：
`$term_id` :分类id

返回：
类型数组,指定分类下的子分类

示例：

```
<?php
 $term_id=1;
 $terms=sp_get_child_terms($term_id ); //获取子分类信息
 print_r($terms); //打印出子分类信息
?>
```

返回数组item说明：
`term_id` :分类id
`name` :分类名称
`taxonomy` :分类的类型，用字符串表示，article表示文章
`description` :分类描述
`parent` :分类父级id,terms表中的term_id
`seo_title`
`seo_keywords`
`seo_description`
`list_tpl` :分类列表页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为list
`one_tpl` : 分类单文章页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为article

模板中用法：

```
<php>
    $term_id=1;
     $terms=sp_get_child_terms($term_id ); //获取子分类信息
</php>
<foreach name="terms" item="vo">
    {$vo.name}<!--打印出分类名称 -->

</foreach>
```

sp_sql_posts_paged()

```
sp_sql_posts_paged($tag,$pagesize,$pagetpl)
```

功能：
文章分页查询方法

参数：
`$tag` :查询标签,以字符串方式传入,例："field:post_title,post_content;limit:0,8;order:post_date desc,listorder desc;where:id>0;"

```
field:调用post指定字段,如(id,post_title...) 默认全部
limit:数据条数,默认值为10,可以指定从第几条开始,如3,8(表示共调用8条,从第3条开始)
order:排序方式，如：post_date desc
where:查询条件，字符串形式，和sql语句一样
$pagesize:每页显示文章数
$pagetpl:分页模板，例："{first}{prev}{liststart}{list}{listend}{next}{last}"
```

返回：
类型数组,带分页数据的文章列表

示例：

```
<? php
$tag='cid:6;field:post_title,post_content;order:listorder asc';
$content=sp_sql_posts_paged($tag);
$posts=$content['posts'];
$pager=$content['page'];
?>
```

$tag规则：

cid 分类id；

field 需要取出的内容，默认取出所有信息；order排序方式，可根据任何取出的字段排序，默认为按发布时间排序。

field可选参数：

term_id 文章分类id

post_author 文章作者id，后台管理员，对应于表users里的ID;

post_keywords

post_date 文章发布日期 格式2014-01-01 00:00:00

post_content 文章内容

post_title 文章标题

post_excerpt 文章摘要

post_modified 文章更新日期 格式2014-01-01 00:00:00

smeta 文章扩展属性，以json格式保存，如属性thumb文章缩略图

user_nicename 管理员昵称

user_email 管理员邮箱

模板中用法：

```
<php>
$content=sp_sql_posts_paged('cid:6;field:post_title,post_content;order:listorder asc');
</php>

<foreach name="content['posts']" item="vo">  /* 遍历数组 */
    {$vo.term_id }<br>
    {$vo.post_author }<br>
    {$vo.post_keywords }<br>
    {$vo.post_date }<br>
    {$vo.post_content }<br>
    {$vo.post_title }<br>
    {$vo.post_excerpt }<br>
    {$vo.post_modified}<br>
    {$vo.user_nicename }<br>
    {$vo.user_email }<br>
    <php>
    $smeta=json_decode($vo['smeta'],true);/* 把smeta转化成数组 */
    </php>
    <img src="/data/upload/{$smeta.thumb}"/>
</foreach>

<div>{$content.page}</div><!--分页-->
```

sp_sql_page()

```
sp_sql_page($id)
```

功能：
获取指定id的页面

参数：
`$id` :页面的id

返回：
类型数组,符合条件的页面

示例：

```
<?php
$ID=1000;    //
$page=sp_sql_page($ID);
print_r($page);
```

sp_sql_posts_bycatid()

> X1.2新增

```
sp_sql_posts_bycatid($cid,$tag,$where)
```

功能：
根据分类文章分类ID 获取该分类下所有文章（包含子分类中文章），调用方式同sp_sql_posts

参数：
`$cid` :分类id
`$tag` :查询标签，以字符串方式传入,例："order:post_date desc,listorder desc;"field:调用post指定字段,如(id,post_title...) 默认全部 ； limit:数据条数,默认值为10,可以指定从第几条开始,如3,8(表示共调用8条,从第3条开始);order:推荐方式(post_date) (desc/asc/rand())
`$where` :按照thinkphp where array格式

返回：
类型数组,符合条件的文章列表

sp_sql_posts_paged_bycatid()

> X1.2新增

```
sp_sql_posts_paged_bycatid($cid,$tag,$pagesize,$pagetpl)
```

功能：
根据分类文章分类ID 获取该分类下所有文章（包含子分类中文章），已经分页，调用方式同sp_sql_posts_paged

参数：
`$cid` :分类id
`$tag` :查询标签,以字符串方式传入,例："field:post_title,post_content;limit:0,8;order:post_date desc,listorder desc;where:id>0;"

```
field:调用post指定字段,如(id,post_title...) 默认全部
limit:数据条数,默认值为10,可以指定从第几条开始,如3,8(表示共调用8条,从第3条开始)
order:排序方式，如：post_date desc
where:查询条件，字符串形式，和sql语句一样
$pagesize:每页显示文章数
$pagetpl:分页模板，例："{first}{prev}{liststart}{list}{listend}{next}{last}"
```

返回：
类型数组,符合条件的文章列表，及分页html

```
array(
    'content'=>'',//符合条件的文章列表
    'page'=>''//分页html
)
```

模板使用方法：

```
<div class="main-title">
    <php>
        $result=sp_sql_posts_paged_bycatid($cat_id,"",20);
    </php>
</div>

<volist name="result['posts']" id="vo">
<php>
    $smeta=json_decode($vo['smeta'], true);
</php>

<div class="list-boxes">
```

```
    <h2><a href="{:leuu('article/index',array('id'=>$vo['tid']))}">{$vo.post_title}</a>
</h2>
    <p>{$vo.post_excerpt}</p>
    <div>
        <div class="pull-left">
            <div class="list-actions">
            <a href="javascript:;"><i class="fa fa-eye"></i><span>{$vo.post_hits}</span>
</a>
            <a href="{:U('article/do_like',array('id'=>$vo['object_id']))}" class="J_co
unt_btn"><i class="fa fa-thumbs-up"></i><span class="count">{$vo.post_like}</span></a>
            <a href="{:U('user/favorite/do_favorite',array('id'=>$vo['object_id']))}" c
lass="J_favorite_btn" data-title="{$vo.post_title}" data-url="{:U('portal/article/index'
,array('id'=>$vo['tid']))}" data-key="{:sp_get_favorite_key('posts',$vo['object_id'])}">

                <i class="fa fa-star-o"></i>
            </a>
            </div>
        </div>
        <a class="btn btn-warning pull-right" href="{:leuu('article/index',array('id'=>
$vo['tid']))}">查看更多</a>
    </div>
</div>
</volist>

<div class="pagination">
    <ul>
        {$result['page']}
    </ul>
</div>
```

门户模板制作

文章列表页制作

可用变量

```
{$term_id}<br><!--分类id-->
{$cat_id}<br><!--分类id,同上-->
{$name}<br><!--分类名称-->
{$taxonomy}<br><!--分类的类型，用字符串表示，article表示文章-->
{$description}<br><!--分类描述-->
{$parent}<br><!--分类父级id,terms表中的term_id-->
{$seo_title}<br>
{$seo_keywords}<br>
{$seo_description}<br>
{$list_tpl}<br><!--分类列表页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为list-->

{$one_tpl}<br><!--分类单文章页的模板，对应于模板目录下Portal/文件名+.html，文件名默认为artic
le-->
```

例子：

```
<php>
    $lists = sp_sql_posts_paged("cid:$cat_id;order:post_date DESC;",10);
</php>
<volist name="lists['posts']" id="vo">
    <php>
    $smeta=json_decode($vo['smeta'], true);
    </php>

<div class="list-boxes">
    <h2><a href="{:leuu('article/index',array('id'=>$vo['object_id'],'cid'=>$vo['term_i
d']))}">{$vo.post_title}</a></h2>
    <p>{$vo.post_excerpt}</p>
    <div>
    <div class="pull-left">
        <div class="list-actions">
    <a href="javascript:;"><i class="fa fa-eye"></i><span>{$vo.post_hits}</span></a>
    <a href="{:U('article/do_like',array('id'=>$vo['object_id']))}" class="js-count-btn
"><i class="fa fa-thumbs-up"></i><span class="count">{$vo.post_like}</span></a>
    <a href="{:U('user/favorite/do_favorite',array('id'=>$vo['object_id']))}" class="js-
favorite-btn" data-title="{$vo.post_title}" data-url="{:U('portal/article/index',array(
'id'=>$vo['object_id'],'cid'=>$vo['term_id']))}" data-key="{:sp_get_favorite_key('posts'
,$vo['object_id'])}">i class="fa fa-star-o"></i></a>
    </div>
    </div>
    <a class="btn btn-warning pull-right" href="{:leuu('article/index',array('id'=>$vo[
'object_id'],'cid'=>$vo['term_id']))}">查看更多</a>
</div>
```

```
</div>
</volist>

<div class="pagination">
    <ul>
    {$lists['page']}
    </ul>
</div>
```

文章内页制作

可用变量

```
{$article_id} 当前文章id
{$term_id} 文章分类id，对应于表terms里的term_id;
{$post_author}<!-- 文章作者id，后台管理员，对应于表users里的ID;--><br>
{$post_keywords}<br><!---->
{$post_date}<br><!-- 文章发布日期 格式2014-01-01 00:00:00-->
{$post_content}<br><!-- 文章内容-->
{$post_title}<br><!-- 文章标题-->
{$post_excerpt}<br><!-- 文章摘要-->
{$post_modified}<br><!-- 文章更新日期 格式2014-01-01 00:00:00-->
{$smeta}<br><!-- 文章扩展属性，以json格式保存，如属性thumb文章缩略图，以后还会增加很多-->
{$user_nicename}<br><!-- 管理员昵称-->
{$user_email}<br><!-- 管理员邮箱-->
```

对变量smeta的转化

```
<php>
  /*处理smeta*/
  $smeta=json_decode($smeta,true);/*把smeta转化成数组*/
</php>
{$smeta.thumb}<br><!-- thumb文章缩略图，以后还会增加很多-->

<img src="{:sp_get_asset_upload_path($smeta['thumb'])}"/>
```

页面制作

可用变量

```
{$post_author}<br><!-- 页面作者id，后台管理员，对应于表users里的ID;-->
{$post_keywords}<br><!---->
{$post_date}<br><!-- 页面发布日期 格式2014-01-01 00:00:00-->
{$post_content}<br><!-- 页面内容-->
{$post_title}<br><!-- 页面标题-->
{$post_excerpt}<br><!-- 页面摘要-->
{$post_modified}<br><!-- 页面更新日期 格式2014-01-01 00:00:00-->
{$smeta}<br><!-- 页面扩展属性，以json格式保存，如属性thumb页面缩略图，以后还会增加很多-->
```

对变量smeta的转化

```
<php>
/*处理smeta*/
$smeta=json_decode($smeta,true);/*把smeta转化成数组*/
</php>
{$smeta.thumb}<br><!-- thumb页面缩略图，以后还会增加很多-->

<img src="/data/upload/{$smeta.thumb}"/>
```

获取文章的各种方式

```
sp_sql_posts()
```

示例：

```
<?php
$tag='cid:6;field:post_title,post_content;order:listorder asc';
$posts=sp_sql_posts($tag);
print_r($posts);
$smeta=json_decode($vo['smeta'],true); //smeta处理方法，将其转化为数组
?>
```

$tag规则：

```
cid 分类id;
field 需要取出的内容，默认取出所有信息；order排序方式，可根据任何取出的字段排序，默认为按发布时间排序。
field可选参数：
term_id      文章分类id
post_author      文章作者id，后台管理员，对应于表users里的ID
post_keywords      文章关键词
post_date      文章发布日期 格式2014-01-01 00:00:00
post_content      文章内容
post_title      文章标题
post_excerpt      文章摘要
post_modified      文章更新日期 格式2014-01-01 00:00:00
smeta      文章扩展属性，以json格式保存，如属性thumb文章缩略图
user_nicename      管理员昵称
user_email      管理员邮箱
```

模板中用法：

```
<php>
$posts=sp_sql_posts('cid:6;field:post_title,post_content;order:listorder asc');
</php>

<foreach name="posts" item="vo">  /* 遍历数组 */
    {$vo.term_id }<br>
    {$vo.post_author }<br>
    {$vo.post_keywords }<br>
    {$vo.post_date }<br>
    {$vo.post_content }<br>
    {$vo.post_title }<br>
    {$vo.post_excerpt }<br>
```

```
    {$vo.post_modified}<br>
    {$vo.user_nicename }<br>
    {$vo.user_email }<br>
    <php>
    $smeta=json_decode($vo['smeta'],true);/* 把smeta转化成数组 */
    </php>
    <img src="{:sp_get_asset_upload_path($smeta['thumb'])}"/>
</foreach>
```

```
sp_sql_posts_paged()
```

示例：

```
<?php
$tag='cid:6;field:post_title,post_content;order:listorder asc';
$content=sp_sql_posts_paged($tag);
$posts=$content['posts'];
$pager=$content['page'];
?>
```

$tag规则：

```
cid 分类id;
field 需要取出的内容，默认取出所有信息；order排序方式，可根据任何取出的字段排序，默认为按发布时间排序。
field可选参数:
term_id     文章分类id
post_author     文章作者id，后台管理员，对应于表users里的ID
post_keywords     文章关键词
post_date     文章发布日期 格式2014-01-01 00:00:00
post_content     文章内容
post_title     文章标题
post_excerpt     文章摘要
post_modified     文章更新日期 格式2014-01-01 00:00:00
smeta     文章扩展属性，以json格式保存，如属性thumb文章缩略图
user_nicename     管理员昵称
user_email     管理员邮箱
```

模板中用法：

```
<php>
$content=sp_sql_posts_paged('cid:6;field:post_title,post_content;order:listorder asc');
</php>

<foreach name="content['posts']" item="vo">  /* 遍历数组 */
    {$vo.term_id }<br>
    {$vo.post_author }<br>
```

```
    {$vo.post_keywords }<br>
    {$vo.post_date }<br>
    {$vo.post_content }<br>
    {$vo.post_title }<br>
    {$vo.post_excerpt }<br>
    {$vo.post_modified}<br>
    {$vo.user_nicename }<br>
    {$vo.user_email }<br>
    <php>
    $smeta=json_decode($vo['smeta'],true);/* 把smeta转化成数组 */
    </php>
    <img src="{:sp_get_asset_upload_path($smeta['thumb'])}"/>
</foreach>

<div>{$content.page}</div><!--分页-->
```

热门文章组件制作

热门文章

```
<!--获取最热的5条文章-->
<php>$hot_articles=sp_sql_posts("cid:$portal_index_lastnews;field:post_title,post_excer
pt,object_id,term_id,smeta;order:post_hits desc;limit:5;"); </php>
<ul class="unstyled">
 <foreach name="hot_articles" item="vo">
  <php>$top=$key<3?"top3":"";</php>
  <li class="{$top}">
      <i>{$key+1}</i>
      <a title="{$vo.post_title}" href="{:leuu('article/index',array('id'=>$vo['object_
id'],'cid'=>$vo['term_id']))}">{$vo.post_title}</a></li>
 </foreach>
</ul>
```

seo优化

全站seo:

```
{$site_seo_title}              <!--SEO标题-->
{$site_seo_keywords}           <!--SEO关键字-->
{$site_seo_description}        <!--SEO描述-->
```

文章列表页：

```
{$seo_title}                    <!--SEO标题-->
{$seo_keywords}                 <!--SEO关键字-->
{$seo_description}              <!--SEO描述-->
```

文章页：

```
 {$post_title}    <!-- 文章标题-->
 {$post_keywords} <!--关键字-->
 {$post_excerpt}  <!-- 文章摘要-->
```

页面：

```
 {$post_title}    <!-- 文章标题-->
 {$post_keywords} <!--关键字-->
 {$post_excerpt}  <!-- 文章摘要-->
```

文章相册制作

```
<php>$smeta=json_decode($vo['smeta'], true);</php>
<div id="imgs" class="imgs">
    <foreach name="smeta['photo']" item="vo"> <
        <img src="{:sp_get_asset_upload_path($vo['url'])}" />
    </foreach>
</div>
作者：小夏
```

文章列表推荐功能制作

在模版调用推荐功能：recommended，1推荐 0不推荐

我们在模板只要判断recommended等于1时给标题个样式变红就是推荐的文章

```
<div class="span9">
  <div class="">
      <php>
          $lists = sp_sql_posts_paged("cid:$cat_id;order:istop desc , post_modified des
c;",10);
      </php>
      <volist name="lists['posts']" id="vo">
      <php>
          $smeta=json_decode($vo['smeta'], true);
      </php>

      <div class="list-boxes">
          <php>$recommended_style=$vo['recommended']==1?"color:red;":"";</php>
          <h2><a href="{:leuu('article/index',array('id'=>$vo['tid'],'cid'=>$vo['term_i
d']))}" style="{$recommended_style}">{$vo.post_title}</a></h2>
          <p>{$vo.post_excerpt|msubstr=0,256}</p>
          <div>
              <div class="pull-left">
                  <div class="list-actions">
                  <a href="javascript:;"><i class="fa fa-eye"></i><span>{$vo.post_hits}
</span></a>
                  <a href="{:U('article/do_like',array('id'=>$vo['object_id']))}" class
="js-count-btn"><i class="fa fa-thumbs-up"></i><span class="count">{$vo.post_like}</span
></a>
                  <a href="{:U('user/favorite/do_favorite',array('id'=>$vo['object_id'])
)}" class="js-favorite-btn" data-title="{$vo.post_title}" data-url="{:U('portal/article
/index',array('id'=>$vo['tid']))}" data-key="{:sp_get_favorite_key('posts',$vo['object_
id'])}">
                      <i class="fa fa-star-o"></i>
                  </a>
                  </div>
              </div>
              <a class="btn btn-warning pull-right" href="{:leuu('article/index',array(
'id'=>$vo['tid'],'cid'=>$vo['term_id']))}">查看更多</a>
          </div>
      </div>
      </volist>

  </div>
  <div class="pagination"><ul>{$lists['page']}</ul></div>
</div>
```

文章列表置顶功能制作

先在后台选择置顶功能

在模板的排序是用istop,post_modified降序

```
<div class="">
    <php>
        $lists = sp_sql_posts_paged("cid:$cat_id;order:istop desc , post_modified desc;"
,10);
    </php>
    <volist name="lists['posts']" id="vo">
    <php>
        $smeta=json_decode($vo['smeta'], true);
    </php>

    <div class="list-boxes">
        <h2><a href="{:leuu('article/index',array('id'=>$vo['tid'],'cid'=>$vo['term_id']
))}">{$vo.post_title}</a></h2>
        <p>{$vo.post_excerpt|msubstr=0,256}</p>
        <div>
            <div class="pull-left">
                <div class="list-actions">
                <a href="javascript:;"><i class="fa fa-eye"></i><span>{$vo.post_hits}</
span></a>
                <a href="{:U('article/do_like',array('id'=>$vo['object_id']))}" class="
js-count-btn"><i class="fa fa-thumbs-up"></i><span class="count">{$vo.post_like}</span>
</a>
                <a href="{:U('user/favorite/do_favorite',array('id'=>$vo['object_id']))}
" class="js-favorite-btn" data-title="{$vo.post_title}" data-url="{:U('portal/article/i
ndex',array('id'=>$vo['tid']))}" data-key="{:sp_get_favorite_key('posts',$vo['object_id'
])}">
                    <i class="fa fa-star-o"></i>
                </a>
                </div>
            </div>
```

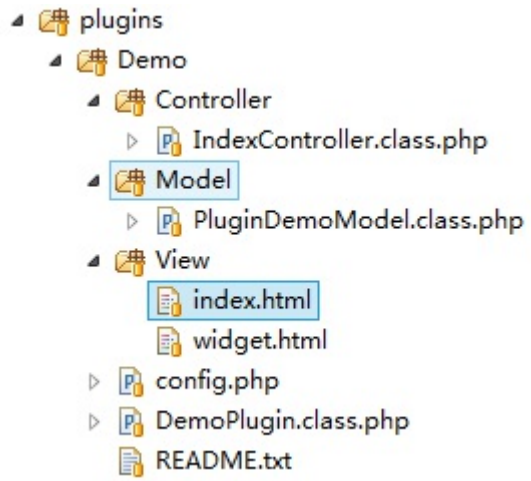
```
            <a class="btn btn-warning pull-right" href="{:leuu('article/index',array('i
d'=>$vo['tid'],'cid'=>$vo['term_id']))}">查看更多</a>
        </div>
    </div>
    </volist>

</div>
```

插件

插件是用于实现简单的显示及数据处理的功能扩展。插件是可以开启关闭的，但不会影响原有系统的代码；

插件结构：

插件结构图



以Demo插件为例：

Controller //控制器目录

Model //插件自定义模型目录 View //模板目录，可以配置多个主题 config.php //模板配置文件

DemoPlugin.class.php 插件主文件，格式为：插件名+Plugin.class.php

README.txt 说明文件

lisense.txt 授权协议文件

插件钩子

钩子是插件执行的触发器；插件就像挂在钩子上的东西；

插件只有实现相应钩子方法，并安装启用成功后才能执行；

ThinkCMF系统内置了很多钩子；http://www.thinkcmf.com/document/hooks.html

开发者也可以用hook('test')方法在控制器只加入钩子，让你的应用具有更好的扩展性；

同时也可以模板里加入钩子{:hook('footer')}；

钩子也支持传入参数hook('footer',array('test'=>1));

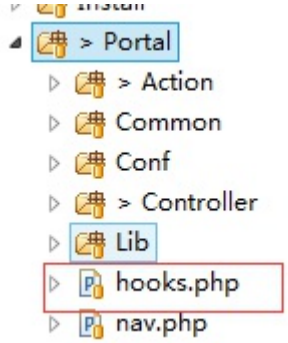
向系统暴露你的钩子

就是把你的钩子在相应的文件里列出来，系统会来检测。 暴露应用控制器钩子：

在你的应用根目录加上hooks.php文件

文件中返回此应用所有钩子数组就可以了；

如portal应用：



hooks.php文件内容

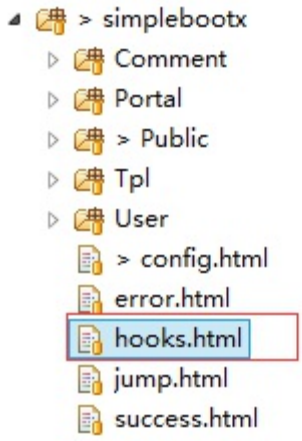
```
<?php
return array(
    //'test',
);
```

暴露你的模板钩子：

在你的模板根目录加上hooks.html文件；

在此文件中用英文逗号分开此模板所有的钩子就可以了；

如simplebootx模板：



hooks.html文件内容：

footer,footer_end

插件配置文件

插件配置文件是插件目录下的config.php，不需要配置的插件可以不添加此文件；

文件结构：

```
return array (
    'text' => array (// 在后台插件配置表单中的键名 ,会是config[text]
        'title' => '文本:', // 表单的label标题
        'type' => 'text',// 表单的类型：text,password,textarea,checkbox,radio,select等
        'value' => 'hello,ThinkCMF!',// 表单的默认值
        'tip' => '这是文本组件的演示' //表单的帮助提示
    ),
    'password' => array (// 在后台插件配置表单中的键名 ,会是config[password]
        'title' => '密码:',
        'type' => 'password',
        'value' => '',
        'tip' => '这是密码组件'
    ),
    'select' => array (// 在后台插件配置表单中的键名 ,会是config[select]
        'title' => '下拉列表:',
        'type' => 'select',
        'options' => array (//select 和radio,checkbox的子选项
            '1' => 'ThinkCMFX',// 值=>显示
            '2' => 'ThinkCMF',
            '3' => '跟猫玩糗事',
            '4' => '门户应用'
        ),
        'value' => '1',
        'tip' => '这是下拉列表组件'
    ),
    'checkbox' => array (
        'title' => '多选框',
        'type' => 'checkbox',
        'options' => array (
            '1' => 'genmaowan.com',
            '2' => 'www.thinkcmf.com'
        ),
        'value' => 1,
        'tip' => '这是多选框组件'
    ),
    'radio' => array (
        'title' => '单选框',
        'type' => 'radio',
        'options' => array (
            '1' => 'ThinkCMFX',
            '2' => 'ThinkCMF'
        ),
        'value' => '1',
```

```
            'tip' => '这是单选框组件'
        ),
        'textarea' => array (
            'title' => '多行文本',
            'type' => 'textarea',
            'value' => '这里是你要填写的内容',
            'tip' => '这是多行文本组件'
        )
);
```

插件类主文件

插件类主文件

文件位于插件根目录

命名格式：插件名+Plugin.class.php

```
<?php
// +----------------------------------------------------------------------
// | ThinkCMF [ WE CAN DO IT MORE SIMPLE ]
// +----------------------------------------------------------------------
// | Copyright (c) 2013-2014 http://www.thinkcmf.com All rights reserved.
// +----------------------------------------------------------------------
// | Author: Dean <zxxjjforever@163.com>
// +----------------------------------------------------------------------
namespace plugins\Demo;//Demo插件英文名，改成你的插件英文就行了
use Common\Lib\Plugin;

/**
 * Demo
 */
class DemoPlugin extends Plugin{//Demo插件英文名，改成你的插件英文就行了

    public $info = array(
        'name'=>'Demo',//Demo插件英文名，改成你的插件英文就行了
        'title'=>'插件演示',
        'description'=>'插件演示',
        'status'=>1,
        'author'=>'ThinkCMF',
        'version'=>'1.0'
    );

    public $has_admin=1;//插件是否有后台管理界面

    public function install(){//安装方法必须实现
        return true;//安装成功返回true，失败false
    }

    public function uninstall(){//卸载方法必须实现
        return true;//卸载成功返回true，失败false
    }

    //实现的footer钩子方法
    public function footer($param){
        $config=$this->getConfig();
        $this->assign($config);
        $this->display('widget');
    }
```

```
}
```

插件开发流程

1. 确定功能，先给插件起名，英文名和中文名 如：Demo,插件演示，然后在plugins目录里添加这个插件目录
2. 确定是否要后台配置文件，如果需要在插件根目录加上config.php，格式可看文档“插件配置文件”
3. 创建插件主类文件，查看“插件类主文件”
4. 确定是否要模板，如需要请根目录添加View目录
5. 确定是否可以外部访问，如需要请加Controller目录，再添加Controller文件
6. 给自己的模板（themes/simplebootx）添加钩子，把{:hook('钩子名')}放到模板相应位置，再在模板根目录添加hooks.html文件，让系统可以获取你模板里的钩子，如themes/simplebootx/hooks.html文件，文件中以英文逗号分隔钩子，不能有空格；hooks.html文件可以这样： footer_end,footer
7. 到后台扩展工具-》插件管理刷新界面就会看到你新添加的插件

最后文件编辑时间: 2017-02-03 17:55:31

插件控制器

插件写法：

```
<?php
namespace plugins\Demo\Controller; //Demo插件英文名，改成你的插件英文就行了
use Api\Controller\PluginController;//插件控制器基类，所有插件都要继承它

class IndexController extends PluginController{

    function index(){
        $this->display(":index");
    }

}
```

插件url生成方法，sp_plugin_url();

X1.4.0新增

```
sp_plugin_url($url,$param,$domain)
```

功能：
生成访问插件的url

参数：
`$url` : url 格式：插件名://控制器名/方法
`$param` :额外参数，默认为空数组
`$domain` :是否添加域名，默认false

返回：
类型url

模板使用：

```
{:sp_plugin_url('Demo://Index/index',array('id'=>2),true)}
{:sp_plugin_url('Demo://List/index',array('id'=>2))}
```

插件数据库模型

创建插件自定义模型

```
<?php
namespace plugins\Demo\Model;//Demo插件英文名，改成你的插件英文就行了
use Common\Model\CommonModel;//继承CommonModel
class PluginDemoModel extends CommonModel{ //Demo插件英文名，改成你的插件英文就行了,插件数据表最好加个plugin前缀再加表名,这个类就是对应“表前缀+plugin_demo”表

    //自动验证
    protected $_validate = array(
            //array(验证字段,验证规则,错误提示,验证条件,附加规则,验证时间)
            //array('ad_name', 'require', '广告名称不能为空! ', 1, 'regex', 3),
    );

    protected function _before_write(&$data) {
        parent::_before_write($data);
    }

    //自定义方法
    function test(){
        echo "hello";
    }
}
```

实例化模型：

```
$plugin_demo_model=D("plugins://Demo/PluginDemo");//实例化自定义模型PluginDemo
$plugin_demo_model->test();//调用自定义模型PluginDemo里的test方法

$users_model=D("Users");//实例化Common模块下的Users模型
//$users_model=D("Common/Users");//也可以这样实例化Common模块下的Users模型
$users=$users_model->limit(0,5)->select();

print_r($users);
```

插件后台管理控制器

1. 在插件类主文件里加has_admin为1的属性；
2. 默认后台管理控制器AdminIndex,默认方法index,不可更改
3. sp_get_current_admin_id();可获取后台管理员id，可用于判断是否登录

```
<?php
namespace plugins\Demo\Controller; //Demo插件英文名，改成你的插件英文就行了
use Api\Controller\PluginController;//插件控制器基类

class AdminIndexController extends PluginController{

    function _initialize(){
        $adminid=sp_get_current_admin_id();//获取后台管理员id，可判断是否登录
        if(!empty($adminid)){
            $this->assign("adminid",$adminid);
        }else{
            //TODO no login
        }
    }

    function index(){
        //$plugin_demo_model=D("plugins://Demo/PluginDemo");//实例化自定义模型PluginDemo
 ,需要创建plugin_demo表
        //$plugin_demo_model->test();//调用自定义模型PluginDemo里的test方法

        $users_model=D("Users");//实例化Common模块下的Users模型
        //$users_model=D("Common/Users");//也可以这样实例化Common模块下的Users模型
        $users=$users_model->limit(0,5)->select();

        $this->assign("users",$users);
        $this->display(":admin_index");
    }

}
```

后台管理

SMTP配置

发件人:
显示的发件人名字

邮箱地址: 表示显示发件人的邮箱地址，和下面的邮箱登录帐号最好一样；

SMTP服务器：
表示你的邮件服务器地址，不带http;如163,126邮箱分别是
smtp.163.com,smtp.126.com,smtp.qq.com

连接方式：
为空或"ssl"或"tls"(具体查看邮箱客户端设置帮助)

SMTP服务器端口:
正常连接方式为空的为25,连接方式为"ssl"的为465

邮箱登录帐号:
就是你的邮箱号

邮箱登录密码:
你的邮箱密码；(QQ邮箱为开启smtp邮箱时的第三方登录"授权码")

示例**1**：**QQ**邮箱
发件人:rainfer
邮箱地址:81818832@qq.com
SMTP服务器:smtp.qq.com
连接方式:ssl
SMTP端口:465
发件箱帐号:81818832@qq.com
发件箱密码:xxxxxxxxxxxx

最后文件编辑时间: 2017-02-03 17:55:31

忘记后台密码？

后台密码忘记了怎么办？

1.如果你已经在后台配置了，邮件发送功能且邮箱是你的真实邮箱，可以到前台登录页找回密码；

2.如果你是后台管理员，你可以使用 sp_password()方法生成一下新的密码；

你只要在任何一个前台可以访问控制器里,如
application/Portal/Controller/IndexController.class.php

```
<?php
namespace Portal\Controller;
use Common\Controller\HomebaseController;
/**
 * 首页
 */
class IndexController extends HomebaseController {

    public function index() {
        echo sp_password('666666');//这次一定要记清了，密码是6个6;
        exit;
        $this->display(":index");
    }

}
```

访问你的首页：得到密码后，把你刚刚修改的地方还原；

打开你的数据库管理功能，找到你的管理员那一列，把密码换进去！

3.如果以上方法都不行，那你只能找 ThinkCMF的官网人员了(如：Dean),不过他的收费很高，单次还原密码要1000元，看到这里你估计不会考虑了！

ps:让以后你还忘记密码！

 开源不容易，多少给点支持行不？别删链接呀也！

后台地址是啥？

后台地址是啥？

1.thinkcmf 默认后台地址是/admin

2.如果在后台开启后台地址加密码的功能，那地址就是/?g=admin&upw=系统为你生成的加密码

后台地址加密码开启了，但我没有记呀？

没办法其实 Dean 也无能为力呀！

好吧...我太好了

1.打开 data/conf/config.php文件，记得别用 windows 的记事本打开

2.找到SP_SITE_ADMIN_URL_PASSWORD，复制它的值

3.访问/?g=admin&upw={你第二步得到的值}

最后文件编辑时间: 2017-02-03 17:55:31

后台菜单管理

管理员权限管理

第三方登录配置

专题

多语言开发

概述

ThinkCMF已经默认开启了多语言的支持,后台以语言包形式实现多语言,前台以语言包和多模板的形式实现多语言.

语言包

ThinkCMF开启的语言有三个,分别是zh-cn,en-us,zh-tw,此项配置在application/Common/Conf/config.php配置文件下,可以通过更改 `LANG_LIST` 的值增加语言.

语言包分为框架核心语言包(在simplewind/Core/Lang目录下)和应用语言包(在每个应用的Lang目录下,如application/Portal/Lang),根据你设置的语言列表,在这些地方增加相应的语言包,就实现了多语言.

应用语言包具体位置说明:

语言包	文件位置	加载时间
应用公共语言包	application/Common/Lang/语言.php	都加载
应用语言包	application/应用名/Lang/语言.php	只在访问应用时加载
应用控制器语言包	application/应用名/Lang/语言目录/语言.php	只在访问控制器下方法时加载
应用控制器后台菜单语言包	application/应用名/Lang/语言目录/admin_menu.php	登录后台首页,和访问后台控制器时加载

语言文件定义

1.语言文件格式为PHP返回数组形式,如:

```
return array(
     'ADMIN_CENTER' => 'Admin Center',
     'WELCOME_USER' => 'Welcome, {$username}!',
     'REFRESH_CURRENT_PAGE' => 'Refresh Current Page',
     'WEBSITE_HOME_PAGE' => 'Website Home Page'
);
```

2.也可以在代码中动态设置

```
    L('ADMIN_CENTER','后台管理中心');
    $lang_admin_center = L('ADMIN_CENTER');
```

变量传入支持

1.在定义语言包时也支持变量,如:

```
return array(
    'WELCOME_USER' => 'Welcome, {$username}!',
);
```

2.在使用 L 方法获取时,可以传入$username 变量,如:

```
    $lang_welcome_user =  L('WELCOME_USER',array('username'=>'无敌小夏'));
    echo $lang_welcome_user; // 这里输出的字符串就是:Welcome,无敌小夏!
```

获取语言包设置的值

1.在PHP代码里使用 L 方法,如:

```
    $lang_admin_center = L('ADMIN_CENTER');
    echo $lang_admin_center; // 这里输出的值是:Admin Center
```

2.在模板里使用 L 方法,如:

```
    <!--以下输出的字符串也是:Admin Center-->
    {:L('ADMIN_CENTER')}

    <!--以下输出的字符串也是:Welcome,无敌小夏!-->
    {:L('WELCOME_USER',array('username'=>'无敌小夏'))}
```

前台模板多语言

ThinkCMF前台模板多语言是使用多模板的方式来实现的,如:当前模板是 `simplebootx` ,如果想开启英文前台模板的话,就只要加一个模板名为 `simplebootx_en-us` 模板就可以了;

前台模板多语言实现原理:

ThinkCMF在前台控制器加载模板文件时,会根据当前用户的浏览器语言或者用户指定的语言来加载模板文件,如果是中文用户就加载 simplebootx 里的模板文件,如果是英文用户就加载 simplebootx_en-us 里的模板文件;每个模板里数据调用是独立的,你可以在不同模板里做不同的配置,以调用不同的语言的内容;

为什么选用多模板形式实现前台多语言?

很多用户会疑问,这不是会增加维护的难度吗?为什么不用语言包的形式呢? 维护难度当然会增加,做一个模板和做两个模板是不一样的时间,但你想英文模板和中文模板无论从内容还风格都有可能会不同,一个模板,你要考虑很多布局上兼容的问题,同时,如果想对不同语言的用户做不同的体验上的界面设计,一个模板肯定无法满足,所以多模板形式才是前台多语言最好的选择,当然你在模板里也可以使用应用里设置的语言包.

输出**?l=en-us**就可以打开语言切换

默认语言为zh-cn，可用根据需要开发多个语言包,如后台设置模板为simplebootx，在themes目录下simplebootx目录为zh-cn语言模板，simplebootx_en-us为en-us语言模板，simplebootx_zh-tw对应的就是zh-tw语言包，simplebootx_mobile则为移动版模板，移动模板+"_language"则为移动版对应语言包，如simplebootx有三个语言版本，则对应目录为

```
simplebootx                // PC中文默认模板
simplebootx_en-us          // PC端zh-us语言模板
simplebootx_zh-tw          // PC端zh-tw语言模板
simplebootx_mobile         // 移动端中文默认模板
simplebootx_mobile_en-us   // 移动端en-us语言模板
simplebootx_mobile_zh-tw   // 移动端zh-tw语言模板
```

利用**Page**类和**limit**方法分页

```
$Wxch_indent = M("Wxch_indent"); // 实例化Wxch_indent对象
$count = $Wxch_indent->where($where)->count();// 查询满足要求的总记录数
$Page  = $this->Page($count,25);// 实例化分页类 传入总记录数和每页显示的记录数(25)
$show       = $Page->show("Admin");// 分页显示输出
// 进行分页数据查询 注意limit方法的参数要使用Page类的属性
$list  = $Wxch_indent->where('1')->order('id desc')
->limit($Page->firstRow . ',' . $Page->listRows)->select();
$this->assign('list',$list);// 赋值数据集
$this->assign('Page',$show);// 赋值分页输出
$this->display(":index"); // 输出模板
```

带入查询条件

```
$_GET = array_merge($_GET,$_POST);//分页带条件
$count = $Wxch_indent->where($where)->count();// 查询满足要求的总记录数
$Page  = $this->Page($count,25);// 实例化分页类 传入总记录数和每页显示的记录数(25)
$show       = $Page->show("Admin");// 分页显示输出
 // 进行分页数据查询 注意limit方法的参数要使用Page类的属性
$list  = $Wxch_indent->where($where)->order('id desc')
->limit($Page->firstRow . ',' . $Page->listRows)->select();
```

> 注意：$_GET会自己把查询的条件自动传进分页代码内

最后在视图就可以直接调用了

```
<div class="meneame">{$Page}</div>
```

缓存

ThinkPHP支持多种缓存方式,ThinkCMF同样适用,详细方式请参考:
http://www.kancloud.cn/manual/thinkphp/1834

安全

请参考:http://www.kancloud.cn/manual/thinkphp/1840

SESSION支持

请参考：http://www.kancloud.cn/manual/thinkphp/1872

Cookie支持

请参考：http://www.kancloud.cn/manual/thinkphp/1873

文件上传

验证码

1.生成验证码

生成验证码的函数:

```
sp_verifycode_img($imgparam,$imgattrs);
```

参数:

`$imgparam` :控制验证码的样式,(默认值 length=4&font_size=20&width=238&height=50&use_curve=1&use_noise=1)

`$imgattrs` :生成的验证码img标签的原生属性，除src,onclick之外都可以设置,(默认值:style="cursor: pointer;" title="点击获取")

返回:

包括验证码的html代码

模板里使用:

```
<!--输出一个验证码-->
{:sp_verifycode_img('length=4&font_size=14&width=100&height=34&charset=2345678&use_noise=1&use_curve=0')}
<!--输出结果-->
<img class="verify_img" src="/index.php?g=api&m=checkcode&a=index&length=4&font_size=14&width=100&height=34&charset=2345678&use_noise=1&use_curve=0" onclick="this.src='/index.php?g=api&m=checkcode&a=index&length=4&font_size=14&width=100&height=34&charset=2345678&use_noise=1&use_curve=0&time='+Math.random();" style="cursor: pointer;" title="点击获取"/>

<!--你可以在任何一个需要验证码的表单里生成一个验证码,同时为它增加一个name为vefify 的 input-->
<input type="text" id="input_verify" name="verify"  placeholder="验证码" class="form-control">
```

一个带验证码的表单

```
<!--这是一个完整的ThinkCMF登录的表单-->
<h2 class="text-center">用户登录</h2>
<form class="form-horizontal js-ajax-forms" action="{:U('user/login/dologin')}" method="post">
    <div class="form-group">
        <input type="text" id="input_username" name="username" placeholder="手机号/邮箱/用户名" class="form-control">
```

```
    </div>

    <div class="form-group">
        <input type="password" id="input_password" name="password" placeholder="密码" c
lass="form-control">
    </div>

    <div class="form-group">
        <div class="row">
            <div class="col-md-6">
                <input type="text" id="input_verify" name="verify"  placeholder="验证码"
 class="form-control">
            </div>
            <div class="col-md-6">
                {:sp_verifycode_img('length=4&font_size=14&width=100&height=34&charset=
2345678&use_noise=1&use_curve=0')}
            </div>
        </div>

    </div>

    <div class="form-group">
        <input type="hidden" name="redirect" value="{:I('get.redirect','')}">
        <button class="btn btn-primary btn-block js-ajax-submit" type="submit" style="m
argin-left: 0px">确定</button>
    </div>

    <div class="form-group" style="text-align: center;">
        <ul class="list-inline">
            <li><a href="{:leuu('user/register/index')}">现在注册</a></li>
            <li><a href="{:U('user/login/forgot_password')}">忘记密码</a></li>
        </ul>
    </div>
</form>
```

2.验证码验证

验证验证码的函数:

```
sp_check_verify_code($verifycode='')
```

参数

`$verifycode` :要验证的验证码,默认空

在控制器里使用

```
<?php
```

```php
namespace User\Controller;

use Common\Controller\HomebaseController;

class LoginController extends HomebaseController
{

    // 前台用户登录
    public function index()
    {
        //省略...
    }

    // 登录验证提交
    public function dologin()
    {
        //如果表单里验证码input的name是verify,可以省略参数
        //如果不是可以单独获取验证码传入参数验证
        if (! sp_check_verify_code()) {

            $this->error("验证码错误! ");
        }

        //省略...
    }
}
```

部署

迁移到正式环境

请执行以下步骤：

1.debug打开；

2.把整个程序打包；

3.传到服务器；

4.数据库备份到服务器；

5.改data/conf/db.php的数据库用户名密码；

6.测试；

7.debug关闭;

URL重写

请参考:http://www.kancloud.cn/manual/thinkphp/1866