iWebSNS

主程序开发与改进规范（beta 版）

2010 年 1 月

聚易开放式技术实验室

序言

为了聚易平台能更好的开发与改进，特别作出如下规范，以方便开发人员更好的了解平台的架构和开发流程，也是为了平台以后能更好的修改和维护，请开发者遵守以下规范。本文档还不是很完善，开发者在开发的过程中，可能会遇到这样和那样的问题，出现问题后，大家应及时交流，通过讨论来制定一种合理的解决方案，从而使我们开发手册更加完善，形成一套完善的开发手册，最后感谢所有关注和支持 iWebSNS 的朋友，如果您有什么疑问或建议，请在我们社区的群组里找到相应的官方群组并提交你的疑问或建议，我们会一直留意提交的问题并及时更改，如果是提交 bug，请同时注明你的测试环境，方便我们发现问题所在，我们社区的网址是：http://tech.jooyea.com。

简介：

什么是 Iweb SNS？

iweb SNS, 是一款功能强大且易于扩展的 LAMP 开源 SNS(社交网络)软件。
作为一款大型高并发高负载的开源 SNS 软件，它基于 iweb SuperInteraction(简称 iweb SI)框架开发。借助 iwebSI 平台，站点可以轻松获得支持热插拔及快速增加新节点的集群计算与处理能力（分布式计算与存储/高可用性/负载均衡），以 方便管理 web2.0 类站点持续增长的数据量。SI 的 web 层、db 层负载均衡，基于内存的分布式缓存系统、dfs（分布式文件系统）、分布式数据存储等 可以轻松支持站点拥有服务于百万甚至千万级庞大用户群的能力，并且不管这些交互式服务的请求是来自计算机还是移动终端。
另外，Jooyea 技术团队还提供了一个轻量级的支持库，这使 iWebSNS 也可以轻松部署在虚拟主机上。
iWebSNS 为站点用户提供一个友好易用的个人信息和消息管理 ajax 界面。通过它，建站者可以轻松构建一个以好友关系为核心的交流网络，让站点用户可 以通过空间、日志、动态、好友圈、群组、相册、站内信、留言板、心情等功能模块记录、展示和分享生活；了解好友动态。
iweb SNS 功能及特点：
1、通过成本更低廉并且更优秀扩展能力的 SI 平台库支持，iweb SNS 可轻松应对庞大的流量与新增数据。
2、高性能的编译型模板引擎和 app 工具插件、功能扩展插件使系统具有良好的扩展性和伸缩性。
3、灵活的社交关系模型，这意味用户能便捷的通过好友发展新的关系，通过社交关系图谱彼此可以相互订阅最新动态。
4、安全的隐私管理，用户可以对自己帐户的内容项目进行保密设置，或者针对特殊关系组进行保密设置。
5、多语言接口，安装相应的语言包，使用 iweb SNS 架设的站点内所有信息均可以实现多语言切换。
6、既时状态公开传讯，以便及时让好友了解自己的最新动态。
7、相册图片及日志分享
8、心情分享
9、群组或圈子
10、通知与请求
11、丰富简练的用户概况

编写目的：

本文档主要是针对内部开发人员和主程序代码贡献者编撰的，供大家形成统一的开发风格，利于技术人员的阅读和开发，实现产品的易于维护性，强壮性。

构架说明：

主目录结构：

┌─action 进行表单处理的程序目录

├─defaultview 系统恢复文件目录

├─doc 系统备份 SQL 的文件和系统安装数据库的 SQL 文件目录

├─foundation 系统函数库目录

├─install 系统安装目录

├─iweb_mini_lib 系统核心包目录

├─iweb_si_lib 高级系统核心包目录

├─langpackage 系统语言包目录

├─models 系统模型文件目录

├─modules 编译后模块文件目录【此目录编译生成的】

├─plugins 插件扩展目录

├─servtools 系统服务工具目录

├─skin 系统皮肤目录

├─sysadmin 后台管理目录

├─templates 模板目录

├─uiparts UI 分解文档目录

├─uploadfiles 上传目录

├configuration.php 系统配制文件

├do.php Action 处理转换入口文件

├modules.php 模块转换入口文件

├home.php 用户主页文件【此文件是编译生成文件】

├index.php 系统程序入口文件【此文件是编译生成文件】

└main.php 系统主页文件【此文件是编译生成文件】

系统架构

系统是基于 MVC 的模式架构思想开发的，现介绍一下平台的 MVC 架构。

MVC（Model-View-Controller，模型—视图—控制器模式）

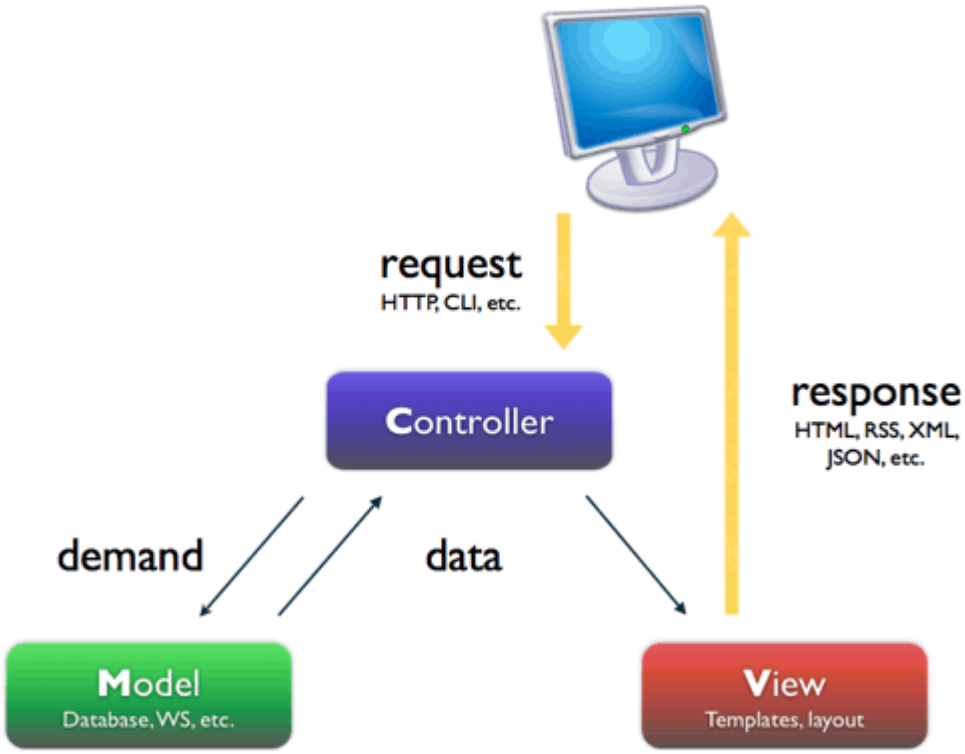


图 1

目录结构对应说明：

Controller(控制器)：

主要包括：对用户的请求进行拦截转发处理

do.php：主要是拦截 Action 请求，通过转发实现交互处理

格式：do.php?act=msg_crt

处理：通过 act 传过来的参数，查询系统定义的 Action 资源表，找到相应的处理模块，然后由相应模块处理完成后，执行回调函数，再次查询资源表，转向相应文件页面。

如图：

图 2

modules.php：主要是控制前台模块的显示

通过 app 传过来的参数，查询 modules 资源表，找到相应的资源文件，转向对应资源文件进行处理。如图：

图 3

Model(模型)：

主要包括：各个功能模块以及对数据库底层的操作

Action 目录：负责系统添加、修改、删除请求的处理模块文件

存放了（图 2）中的对应 Action 资源文件，实现与数据库的交互处理，使数据永久保存在数据库中，主要体现在对数据库的添加、修改、删除上。

Models 目录：系统的所有模型文件

存放了（图 3）中的 modules 资源文件的 PHP 代码部分，实现对数据库的访问，主要体现是对数据库的查询上，是实现编译生成可执行文件的 PHP 代码部分。

View(视图)：

主要包括：布局模板文件、皮肤（CSS 和相关图片、动画以及 JS）

Templates 目录：所有静态模板文件

此目录下的所有文件和目录结构的

Skin 目录：皮肤文件目录

说明：Models 目录和 Templates 目录的结构是完全一样的，而且必需完全一样，这是系统进行编译生成可执行文件的基础。

系统编译流程

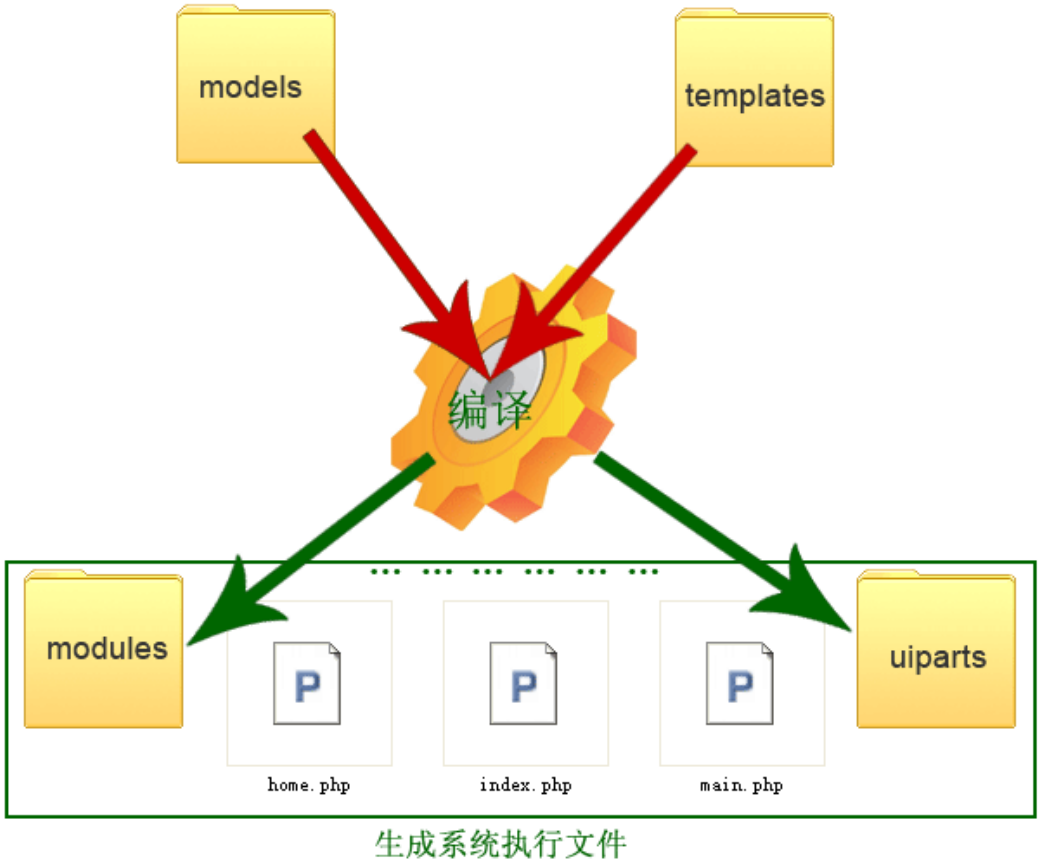


图 4

Models 目录：

此目录存放系统功能模块的 PHP 代码部分，也是开发者开发和修改的根源部分。

templates 目录：

此目录存放系统的静态布局模板文件，文件包含系统规定的一些标签，以实现系统动态编译成执行文件。

标签的实用说明将在下面介绍。

三个核心显示文件：

Home.php SNS 用户的个人主页文件

Index.php SNS 系统的程序入口文件

Main.php SNS 系统主页文件

开发者如何添加自己的功能模块？

清楚了 MVC 的架构和系统的编译原理，开发者应该明确开发一个自己的功能模块要考虑 MVC 这三部分操作：

1、 开发自己的 Action 功能模块，一个单独的模块应该独立建立一个文件夹，以方便系统的开发和维护，

实现对数据信息的永久保存，这里主要是实现对数据的添加、修改和删除操作。

2、 开发自己的信息展示页面，由于系统是通过编译处理的，所以开发者应该：

a) 先在 models 目录下建立信息提取的 PHP 代码断文件，这里要根据页面的具体要求，看是否要添加上权限验证代码。

b) 然后在 templetes 对应的目录下，建立自己的视图代码部分，编写模板文件要注意模板文件对标签的要求，此文件主要实现对信息的显示。

3、 在 do.php 和 modules.php 拦截容器里配制上自己的资源信息，通过后台进行编译处理，生成可执行文件，从而完成自己的功能模块的添加。

a) do.php 拦截容器的信息配制要求：在$actarrry 数组中加入 'login'=>array('action/login_act.php','main.php') ,文件的路径是根目录的相对路径,,关键词 login 代表 act 请求参数，数组的第一项，指对请求参数要进行的模块处理文件，数组的第二项，指在进行完第一项处理后要转向的处理。

b) modules.php 拦截容器的信息配制要求：在$appArray 数组中加入 'start'=>'modules/start.php'，关键词 start 代表 app 请求参数，内容 value 对应的是要处理显示信息模块文件（这是文件编译后的模块文件）。

举例说明：

我们以 helloWord 模块做一个实例：

1、 我们先建立一张表 helloword 表 ，以实现对数据的存储，此实例只实现了很简单的应用，主要是为了演示如何添加自己定义的功能模块。建表代码如下：

创建数据表

```
CREATE TABLE `helloWord` (
   `id` int(10) NOT NULL AUTO_INCREME
NT,
   `msg` varchar(200) DEFAULT NULL,
   PRIMARY KEY (`id`)
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

2、 在 action 目录下建立一个文件夹 hellowrod,在此目录下建立文件 helloword_add.action.php，helloword_del.action.php 文件,同时在 foundation 目录下建立一个 model_helloword.php 文件【说明：格式为 model_模块名.php 文件，主要有以下目的：把对模块的操作封装成对应的函数，方便以后内部的修改和添加，同时对开发 API 和方便外部调用打下基础】

3、 Helloword_add.action.php 文件代码如下：

```
<?php
```

```
require("foundation/arecent_affair.php");
require("foundation/aanti_refresh.php");
require("foundation/aintegral.php");
//引入语言包 此实例暂不加入
//变量取得
$hello_msg=short_check(get_args("hello_msg"));
//防止重复提交
$t_blog=$tablePreStr."helloword";
$dbo = new dbex;
//读写分离定义函数
dbtarget('w',$dbServs);
$sql="insert into $t_blog (msg)"
 ." values ('".$hello_msg."')";
 if($dbo->exeUpdate($sql)){
        action_return(1,"添加测试成功！",");
 }
 //回应信息
 action_return(0,"添加测试失败测！",");
?>
```

Helloword_del.action.php 文件代码如下：

```
 $hello_id=short_check(get_args("hello_id"));
 //数据表定义区
 $t_blog=$tablePreStr."helloword";
 $dbo = new dbex;
 //读写分离定义函数
 dbtarget('w',$dbServs);
 $sql="delete from $t_blog    where id= $hello_id ";
 if($dbo->exeUpdate($sql)){
        action_return(1,"删除测试成功！",");
 }
```

```
//回应信息
action_return(0,"删除测试失败测！",");
```

4、 在 models\modules 目录下建立 helloword 目录，在此目录下建立 helloword_list.php 文件,实现对数据库信息的提取，代码如下：

```
<?php
//数据表定义区
$ses_uid=get_sess_userid();
$is_self_mode='partLimit';
require("foundation/auser_validate.php");
require("foundation/fpages_bar.php");
//数据表定义区
$t_hello = $tablePreStr."helloWord";
$dbo=new dbex();
//读写分离定义方法
dbtarget('r',$dbServs);
$sql="select * from $t_hello";
$hello_list= $dbo->getRs($sql);
?>
```

5、 在目录templates\default\modules目录下建立helloword目录,在此目录下建立文件helloword_list.html文件，templates\default 目录下的结构要和 models 目录下的结构完全一样，并且文件名也要一样，只是文件的扩展名不同，helloword_list.html 文件的代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title></title>
<link rel="stylesheet" type="text/css" href="skin/default/css/row_menus.css">
<link rel="stylesheet" type="text/css" href="skin/default/css/right.css">
</head>
```

```
<body ><h1 style="text_align:center">HelloWord</h1>
<div class='main_right_title'>
<table width="100%">
<tr><td>编号</td><td>信息</td><td>操作</td></tr>
{sta:foreach($hello_list as $rs)[loop]}
<tr><td>{echo:$rs['id'];/}</td><td>{echo:$rs['msg'];/}</td><td><a
href="do.php?act=hello_del&hello_id={echo:$rs['id'];/}">删除</a></td></tr>
{end:foreach/}
</table>
</div>
<form action="do.php?act=hello_add" method="post">
信息:<input name="hello_msg"/><br/>
<input type="submit" value="提交"/>
</form>
</body>
</html>
```

6、 完成以上文件后，在 do.php 文件注册上对应的资源信息,即在$actArray 数组中加入如下信息

'hello_add'=>array('action/helloword/helloword_add.action.php','modules.php?app=hello'),'hello_del'=>array('action/helloword/helloword_del.action.php','modules.php?app=hello')

7、 在 modules.php 文件中注册上对应的视图文件信息,即在$appArray 数组中加入

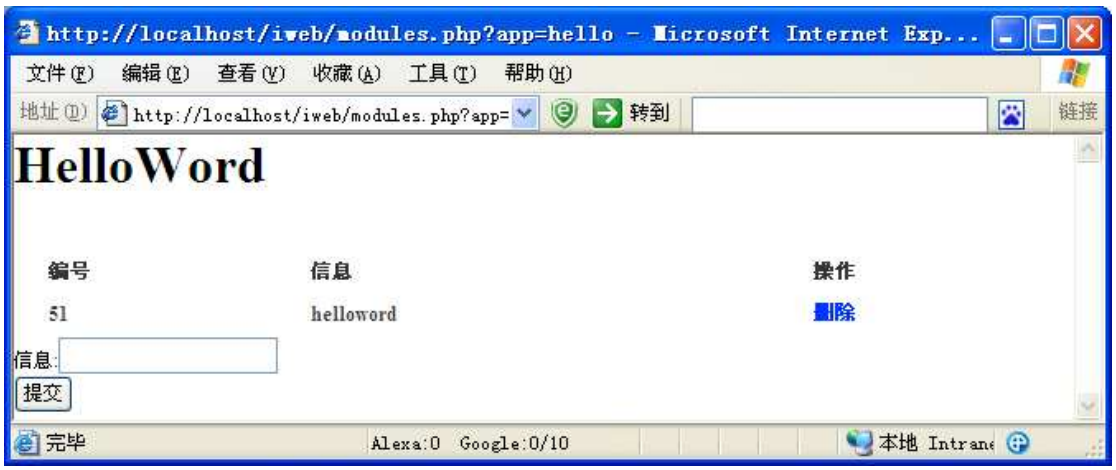
'hello'=>'modules/helloword/helloword_list.php'信息，【注：modules/helloword/helloword_list.php 文件指的系统编译生成的文件路径】

8、 到后台找到自己定义的模板，选中进行编译，然后登录系统，在地址栏中输入，

http://localhost/iweb/modules.php?app=hello ，系统将通过 modules.php 拦截容器转向视图页面

在信息输入框中输入,helloword 点击提交，系统将交给 do.php 拦截容器处理，然后从注册的资源信息中找到对应的处理模块进行处理，处理完毕后，系统调用回调函数，进行信息提示如图：

点击确定，信息将显示出来如图：



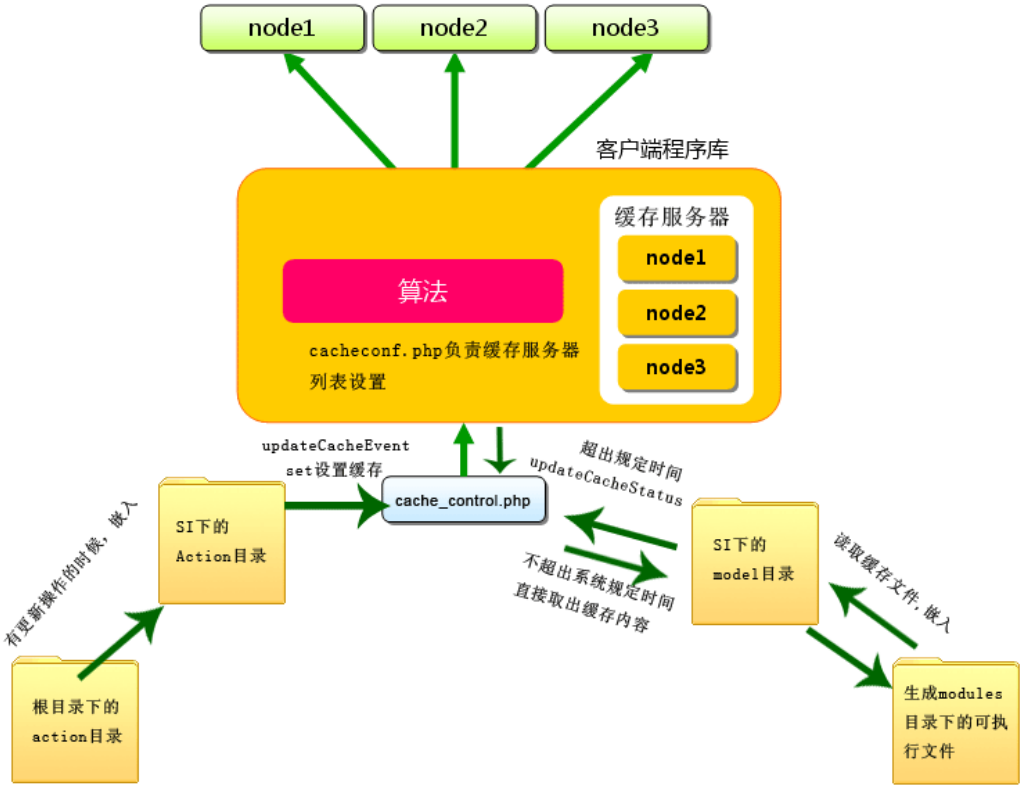
然后点击删除，同样系统将再次交于 do.php 拦截容器处理，然后从注册的资源信息中找到对应的处理模块，更行相应的处理，处理完毕后，调用回调函数处理，系统会提示如下：

点击确定后，系统再次回到视图页面。

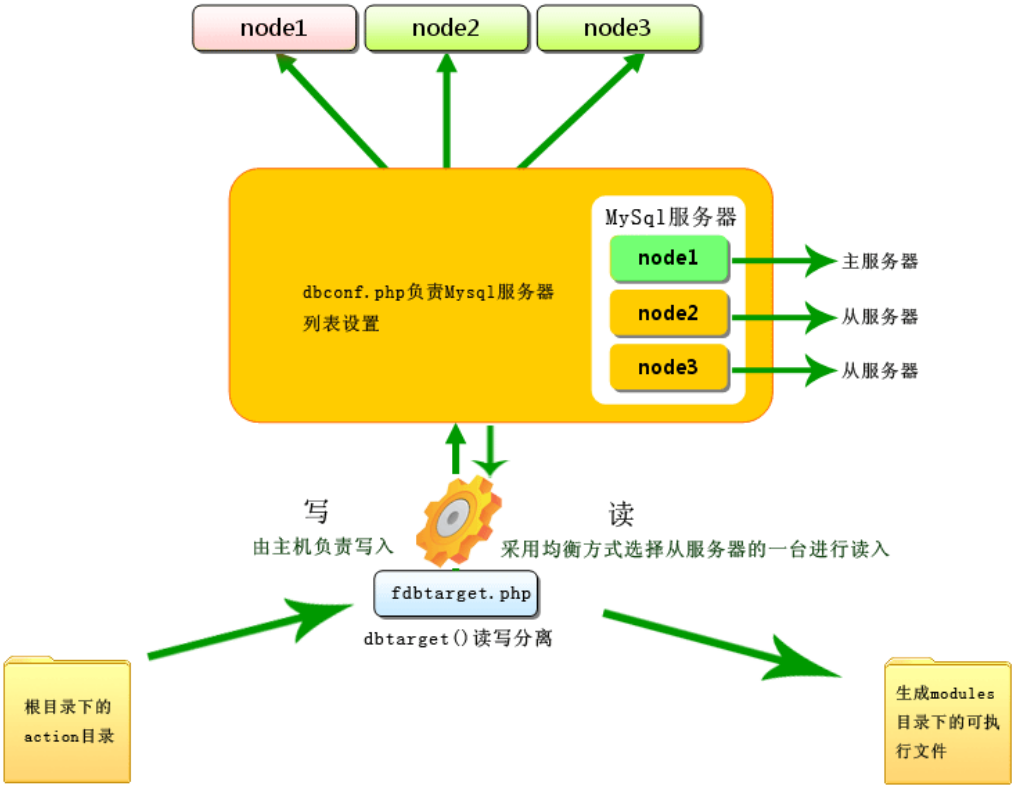
恭喜你，一个小的功能模块就添加上了。

核心包 iweb_si_lib

作为首款为解决高度交互与海量数据负载类Internet 及移动互联网站点而设计的基础架构规范， iwebSI平台规范将庞大在线用户数的站点所需要的集群与分布式技术（分布式计算与存储/高可用性/负载均衡等技术）屏蔽在SI 平台框架内部。基于SI 架构，将会使开发团队无须反复重构系统架构，及编写大量复杂代码以适应在线用户规模的快速增长，而可以将精力放在改善用户体验，改进业务逻辑等方面。当系统升级为可获取iweb SuperInteraction平台库支持时，本目录用于存放iwebSI相关的库支持文件，内核工作原理如下图：



数据库的负载平衡工作原理如下图：



核心包 iweb_mini_lib

是完全模拟在没有 Si 核心包的情况下建立的一个小的核心包，此包不实现多服务器缓存机制，以及多数据库服务器的均衡负载功能。另外也是为了系统更方便进行平滑切换。

根目录下的 action 目录和 models 目录下的文件编写规范

此规范是针对流程上的规范，命名和代码规范详见下面的规范内容。

Action 目录下文件的规范：

```
<?php
    //引入公共方法【主要有用户权限认证，防刷新等模块，以及在 foundation 下自己定义的功能模块】
    require("foundation/aanti_refresh.php");
    ……相关引入代码
    //引入相关语言包
    $a_langpackage=new albumlp;
    ……相关处理代码
    //取得相关变量
    $album_id = short_check(get_args('album_id')); // get_args ($name)已经封装了$_GET[]和$_POST[]的获得
```

```
$visitor_id=get_sess_userid();//get_sess_userid()已经封装了$_SESSION[]
//数据表定义区
$t_album_comment = $tablePreStr."album_comment";//获得真正的表名
$dbo = new dbex;//取得数据库操作对象
//读写分离定义函数，这是实现负载均衡的基础
dbtarget('r',$dbServs);
//读写分离定义函数
dbtarget('w',$dbServs);
……相应代码
if($dbo -> exeUpdate($sql)){//如果操作成功
    //缓存功能区，这是实现缓存处理的基础。
    require($webRoot.$baseLibsPath."action/alb_com_list_update.action.cache.php");
}
    //回调函数
    action_return(1,$a_langpackage->a_add_suc,"-1");//此函数的定义在 do.php 中
```

```
action_return($status,$message,$target)
//关于 action_return(参数 1，参数 2，参数 3)的说明：
//参数 1 的默认值为 1 表示成功，如果非 1 表示失败;参数 2 表示系统要提示的信息，如果非空则系统会提
示参数 2 设置的信息，空则不提示信息，默认值为空;参数 3 如果是-1 返回上一页地址，0 则关闭本页面，
空则返回到 do.php 拦截容器，配制资源信息的第二项的地址，非以上情况则返回当前设定的地址。
```

```
}
?>
```

Models 目录下文件:

```
<?php
    //引入语言包
    $a_langpackage=new albumlp;
    //引入相应模块页面
    require("foundation/auser_mustlogin.php");
    //变量取得
    $album_id=short_check(get_args('album_id'));
```

```
//数据表定义区
$t_album = $tablePreStr."album";
$dbo = new dbex; //取得数据库操作对象
……相应代码
//读写分离定义函数
dbtarget('r',$dbServs);
……相应代码
//缓存功能区，这是实现缓存处理的基础。
require($webRoot.$baseLibsPath."action/alb_com_list_update.action.cache.php");
?>
```

语言包的抽取标准：

模板文件内需要抽成语言包的内容，直接提取到 langpackage->语言文件夹下，语言包的文件命名一般是以模块来命名的，代码要封装成类的形式，变量是以类的属性体现。变量的命名一般是$+文件名的前两位首字母_对应表示内容的英文单词。

布局模板文件标签规范：

模板编写一定要严格使用下列标签，首尾标签一定要注意空格，防止标签不能被正常编译。例如{inc: 写成{ inc: 以及;/} 一定不要在这三个标签之间加入空格和或丢掉/，以上情况系统将不能正常编译。

标签规范		
书写标签	转换标签	实用说明
{inc:require($inc_header);/}	<?php require($inc_header);?>	引入外部文件，可以实用require、require_once、include include_once
{echo:$none_data;/}	<?php echo $none_data;?>	输出标签 可输出任何已定义的变量
{echo:lp{pu_no_user};/}	<?php echo $pu_langpackage->pu_no_user;?>	引用语言包变量信息标签
{echo:str_replace("{b_com_num}",$rs['comments'],lp{b_com_num});/}	<?php echo str_replace("{b_com_num}",$rs['com	调用函数标签 参数调用格式 {变量},数组名

	ments'],$b_langpackage->b_com_num);?>	['index'],lp(变量名) lp:语言包内容
{sta:foreach($blog_rs as $rs)[loop]} {end:foreach/}	<?php foreach($blog_rs as $rs){?> <?php }?>	Foreach 标签
{sta:if(111=222)[exc]} {end:if/}	'<?php if(111=222){?> <?php }?>	If 标签

注：绿色标注部分为可替换内容

插件规范：请参阅插件规范文档

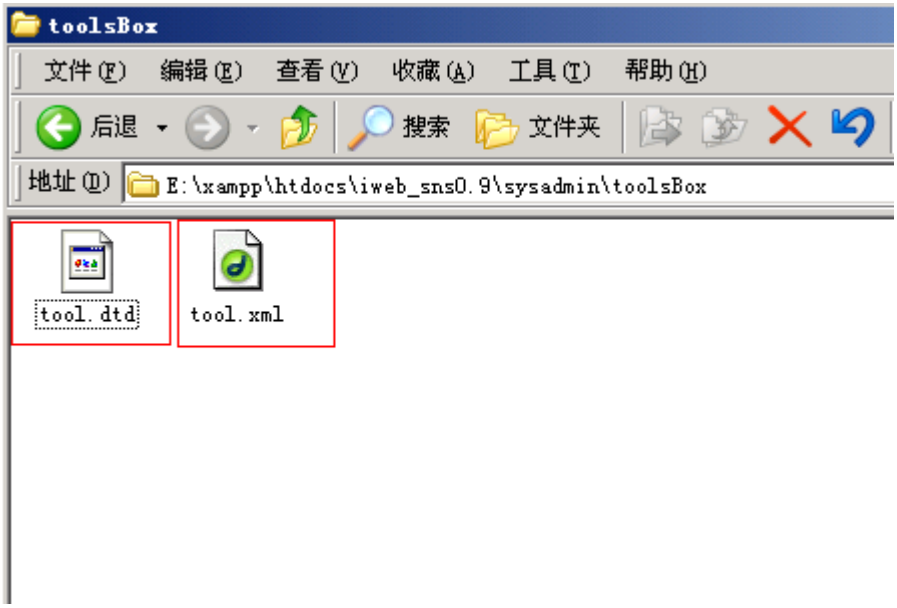
工具箱的说明文档:

工具箱是管理员在网站后台所使用的对网站进行管理的工具，他存在于后台目录（sysadmin）->toolsBox 目录中，工具箱的运行是基于 front_end.php （后台容器）来调用的，每个工具箱都有属于自己单独的索引值和目录，并且这两者的值是相同的。所以当工具箱的索引值（目录）相同时，您就需要手动更改，使得每个工具箱的使用互不影响。

工具箱主体文件包括两部分，例图 1。

（1）tool.xml 配置文件，它是工具箱组件最重要的配置信息。

（2）tool.dtd 规范，它主要是对于 tool.xml 主配置文件的约束条件，用于检查工具箱组件的合法性。



例图 1

下面主要讲述 tool.xml 配置文件的各个节点的功能和作用，这里以“测试数据清理器”的工具组件为例，此工具可以直接从后台下载获取到，或者登录到 jooyea.net 下载专区进行下载。例图 2

工具箱列表

名称	工具编号	时间	作者	操作
测试数据清理器	clear_test	2009-12-29	nswe	下载

提示信息

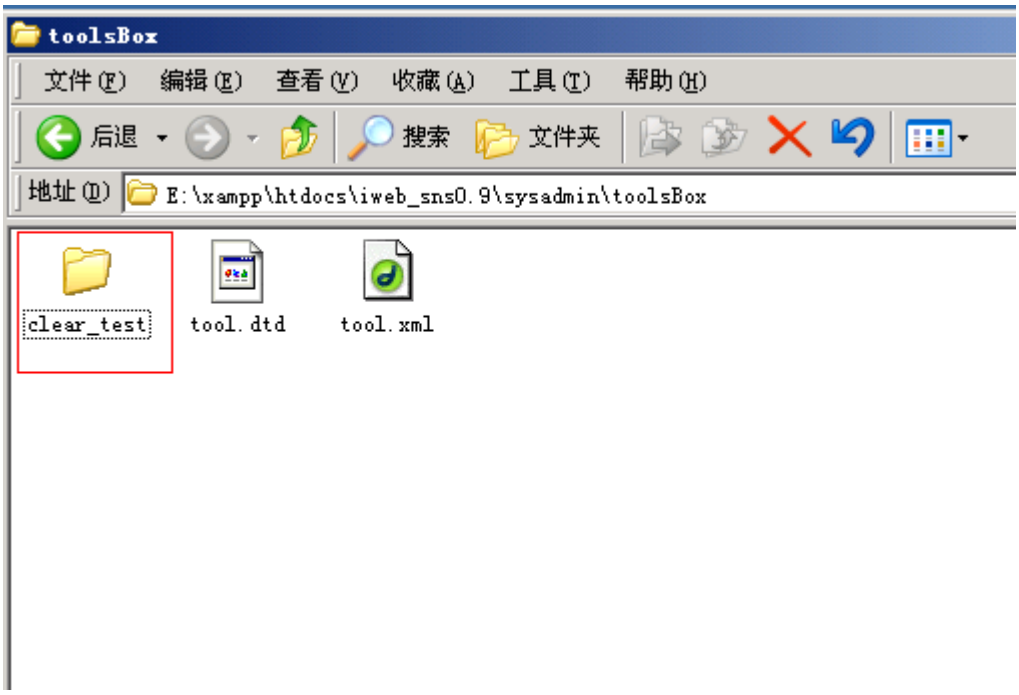
1，当前列出了您的系统中没有集成的工具组件，点击 "下载" 即可把他们下载到后台目录(sysadmin)->toolsBox目录中，并在 "工具箱列表" 中使用。

2，工具组件的编号是唯一的，如果您的系统中没有列表中的某个组件，却又下载不了，请您修改后台目录(sysadmin)->toolsBox目录中的 "tool.xml" 文件。

3，下载最新的模板需要产品授权，详情请登陆我们的官方主站：http://www.jooyea.net

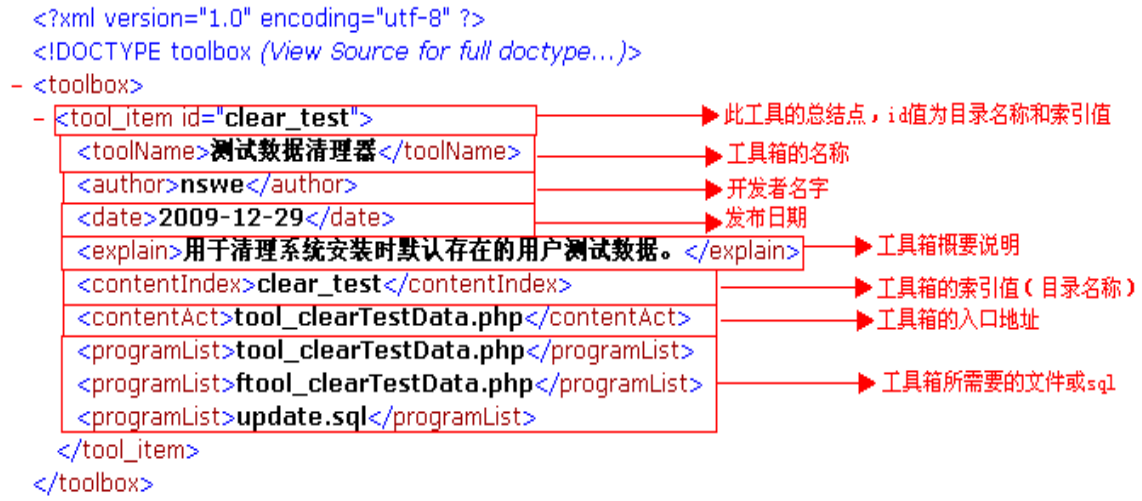
例图 2

“测试数据清理器”下载后，查看工具箱目录（sysadmin->toolsBox）后台目录。此时会发现，此工具的目录已经被动态创建，目录名称（索引值）为 clear_test，此工具所需的程序文件都在此文件夹内。例图 3



例图 3

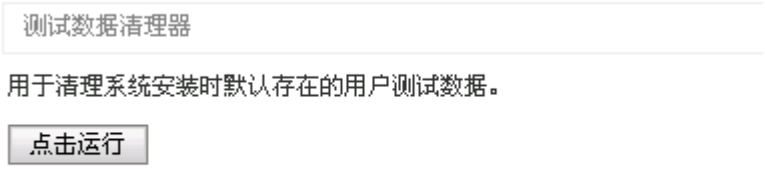
打开 tool.xml 文件，可以看到如下信息：例图 4



例图 4

可以看到“测试数据清理器”的工具配置信息，都包含在节点<tool_item id=”clear_test”>中。每个<tool_item>节点内包含的就是一个独立的工具箱数据信息。

正确配置好 xml 文档，把所需的程序文件放到指定的目录里后，登录后台，在“工具箱”->“工具箱列表”可以直接看到当前系统可以使用的工具箱组件了，例图 5



例图 5

所以您要是想自己开发“工具箱”确实是很简单的，您只需配置 xml 文档，并把编写的程序文件放到指定的目录里就可以在后台直接使用了。

系统编码规范：

1、 文件的编码要求，必需是 UTF-8 编码格式保存。

2、 文件夹命名要求，名称必需用英文同时能反应此文件夹下文件整体功能。

3、 文件命名要求：

a) Action 目录：模块名_功能名.action.php

b) Foundation 目录：首字母(a:代码片断 f:函数 c:类)模块名_功能名.(calss?).php

c) Model 目录：模块名_功能名.php

4、 内部编码命名要求：

a) 变量：使用英文命名或者缩写（3-4 字母），单词之间用’_’连接,全部小写。

b) 常量：常量的定义要求字母全大写

c) 函数：通常每个方法和函数都是执行一个动作的，所以对它们的命名应该清楚的说明它们是做什么的，格式为动词+名词的格式，判断类返回值应为 bool 型 例如：function is_Login(){}

d) 类：使用大写字母作为词的分隔，其他的字母均使用小写，名字的首字母使用大写，不要使用下划线('_')。

e) 接口：使用大写字母作为词的分隔，其他的字母均使用小写，名字的首字母使用大写，不要使用下划线('_')

5、 编写原则：

a) 变量一定要先定义后使用。

b) 能编写成函数的不要写成代码断，能写成类的不要编写成分散的函数。

c) 编写成 a)原则目的是为了更好的以后扩展和修改，以实现程序以后在变更时，我们只需要修改对应的函数和对应的类就行，而无需了解引用它们的地方，实现系统的快速的修改和升级。

d) 在函数编写时，要写清楚函数的使用规则和说明，以及的参数的含义和注明开发者和修改时间。

e) 编写代码一定缩进。

f) 在程序中，不要定义无意义的变量，如$a $b 致使程序晦涩难懂。

g) 数据在添加和修改时一定要验证数据的有效性，保持和数据库中定义的一致性。

h) 在程序开发中，对$_GET $_POST 都已进行了封装，统一使用 get_args($name)方法调用。

i) 对于 session 的使用，开发者要做到，能不使用 session 就尽最大可能的不使用,以提高程序的运行速度，系统对 session 已经进行了封装，对有些 session 进行了固定封装，请开发者尽可能的使用系统定义的，如果是非系统固定定义的 session，请使用基本的封装函数 set_session($key,$value)及 get_session($key)方法。封装的主要目的是防止 session 串用，便于统一管理和修改。

6、 数据库设计和编码规范

a) 编码统一为 UTF-8。

b) 表名命名是由表前缀和表示实体意义的单词组成，单词之间用”_”连接。

c) 字段命名的规则，以实体意义的小写英文单词组成，单词之间用”_”连接

d) 在定义字段类型时，要先考虑字段的范围，最大最小值，然后尽可能的选择最优的类型，方便程序的调用，提高查询速度。

e) 表与表的设计时，应考虑到范式的要求，要做到第二范式，最好能做到第三范式的要求。以减少数据有冗余、更新、添加、删除异常。（二、三范式详见附录参考）

详细目录结构：

Action 目录

┌─album 相册表单处理目录

├─blog 日志表单处理目录

├─group 群组表单处理目录

├─mood 心情表单处理目录

├─msgboard 留言板表单处理目录

├─msgscrip 小纸条表单处理目录

├─mypals 朋友圈表单处理目录

├─privacy 隐私表单处理目录

├─pubtools 公用工具目录

├─users 用户设置表单处理目录

├login_act.php 用户登录 action

├logout_act.php 用户退出 action

└reg_act.php 用户注册 action

Defaultview 目录

┌─c_files 编译结果恢复目录

├─models 视图代码段恢复目录

├─skin 皮肤文件恢复目录

└─tpl 模板文件恢复目录

Doc 目录

┌iweb_sns.sql 系统安装 SQL 文件

└isns_*_*_*.sql 系统备份 SQL 文件

Foundation 目录

┌aanti_refresh.php 防刷新程序

├achange_lp.php 检测语言

├adata_count.php 用户数据统计

├agrade.php 等级处理程序

├aintegral.php 积分规则

├ainfo_collect.php 信息采集

├aoffline.php 站点离线控制代码

├arecent_affair.php 纪录和评论程序

├asession.php 用于 SI 库添加多服务器时的 SESSION 共享配置

├auser_mustlogin.php 权限控制，必须登录

├auser_validate.php 用户检测

├ctime.class.php 时间类库与 mysql 兼容

├cupload.class.php 文件上传处理类

├fcontent_format.php 格式化处理函数

├fdnurl_aget.php 动态生成 URL

├fgetandpost.php 封装 get 和 post

├fmain_target.php 取得 app/act/url 的封装函数

├fpages_bar.php 分页函数

├freq_filter.php post,get 对象过滤通用函数

├fsession.php 封装 session

├fsqlseletiem_set.php 解析 sql 中的取得字段函数

├fstring.php 字符串截取函数

├ftpl_compile.php 模板编译函数

├module_album.php 相册处理模块

├module_affair.php 动态处理模块

├module_blog.php 日志处理模块

├module_group.php 群组处理模块

├module_lang.php 切换语言模块

├module_mood.php 心情处理模块

├module_mypals.php 朋友圈处理模块

├module_remind.php 获取提醒方法

├module-restore.php 回复模块

├module-poll.php 投票模块

├module_share.php 分享处理模块

├module_scrip.php 小纸条处理模块

└module_users.php 用户设置处理模块

iweb_mini_lib 目录

┌─action 更新缓存目录【实现平滑处理】

├─model 列表缓存目录【实现平滑处理】

├─conf 配制文件目录

├cdbex.class.php SQL 统一处理类

├fdbtarget.php 数据库连接统一管理方法

├integral.php 默认整型变量文件

└libs_inc.php 库支持文件

iweb_si_lib 目录

┌─action 事件缓存目录

├─model 列表缓存目录

├─conf 配制文件目录

├cache_control.php 封装更新缓存事件和状态函数

├cdbex.class.php SQL 统一处理类

├fdbtarget.php 数据库连接统一管理方法

├libs_inc.php 库支持文件

└memcached-client.php 内存缓存类文件

langpackage 目录

┌─ft 繁体包目录

└─zh 中文包目录

models 目录

┌─modules 系统模块 php 目录

├─uiparts UI 片段文档 PHP 目录

├home.php 个人主页 PHP 模板文件

├index.php 系统入口 PHP 模板文件

└main.php 系统主页 PHP 模板文件

modules 目录【此目录是编译生成目录】

┌─album 相册模块目录

├─blog 日志模块目录

├─friend 右侧朋友圈展示模块目录

├─group 群组模块目录

├─guest 右侧访客展示目录

├─mood 心情模块目录

├─msgboard 留言板模块目录

├─msgscrip 小纸条模块目录

├─mypals 朋友圈模块目录

├─poll 投票目录

├─privacy 隐私模块目录

├─pubtools 公用工具模块目录

├─recentaffair 动态模块目录

├─restore 回复模块目录

├─share 分享目录

├─uiparts 片段文件目录

├─userapp 用户组件目录

├─users 用户设置模块目录

├default.php

├homestart.php

├invite.php

└start.php

plugins 目录

┌Actions.dtd 规范 Actions.xml 文件的 DTD 文档

├Actions.php 插件 Action 统一处理文件

├ActionUrl.php 插件路径处理文件

├createplugin.php 插件文件配制信息生成器

├includes.php 插件基础文件（Session 数据库支持 GET 和 POST 的封装 调用 API 代理函

数）

┗plugin.php 插件拦截容器

servtools 目录

┏ajax_client AJAX 文件目录

┣error 错误信息提示目录

┣flash_mod 上传照片的 Flash 组件

┣img_cut 图像剪切目录

┣simeditor 小型编辑器目录

┣calendar.js 日历工具代码文件

┣error.php 错误信息跳转处理文件

┣md5.js MD5 加密算法

┗veriCodes.php 图片验证码生成文件

skin 目录

default 默认皮肤文件夹

sysadmin 目录

┏CSS 后台样式目录

┣images 后台图片文件目录

┣img_cut 图片处理目录

┣js 后台统一 JS 目录

┣authorization 授权码目录

┣proxy 远程代理配制信息目录

┣skin 后台皮肤目录

┣toolsbox 工具箱目录

┣action_comp.php 模板编译文件

┣admin_pw_change.php 管理员密码修改

┣affair_del.action.php 心情删除文件

┣affair_list.php 心情列表文件

┣album_del.action.php 相册相关删除

┣album_list.php 相册列表显示

├album_lock.action.php 相册锁定文件

├blog_del.action.php 日志相关删除

├blog_list.php 日志列表文件

├blog_lock.action.php 日志锁定文件

├change_tmp.php 模板修改文件

├check_login.php 登录检测文件

├com_del.action.php 评论删除文件

├comment_list.php 评论列表显示文件

├database.recover.php 数据恢复文件

├database.save.php 数据备份文件

├group_list.php 群组列表文件

├group_lock.action.php 群组定文件

├includes.php 后台统一包含文件

├integral.php 积分设定文件

├login.php 后台登录文件

├login_out.php 后台退出文件

├main.php 后台界面文件

├manage_template.php 模板管理文件

├manager_sort.action.php 好友处理文件

├manager_sort.php 好友显示页面

├member_del.action.php 会员删除文件

├member_list.php 会员列表文件

├member_lock.action.php 会员锁定文件

├menu.php 后台管理菜单文件

├mess_del.action.php 评论删除文件

├msgboard_list.php 留言管理文件

├photo_del.action.php 图片删除文件

├photo_list.php 图片管理文件

├photo_lock.action.php 图片锁定程序文件

├right.php 后台欢迎界面程序

├session_check.php 权限检测文件

├subject_del.action.php 话题删除脚本程序

├subject_list.php 话题管理文件

├table.array.php 数据库表名文件数组文件

├tmp_list.php 模板列表展示文件

├UI_restore.action.php 系统修复处理文件

├UI_restore.php 系统修复文件

└user_info.php 用户信息显示文件

templates 模板目录

defualt 默认模板文件夹

uiparts UI 分解文档目录【此目录是编译生成目录】

┌bread_guide.php 导航分解文件

├footor.php 文件底部分解文件

├homeheader.php 个人主页头部分解文件

├homeleft.php 个人主页左侧分解文件

├mainheader.php 系统主页头部分解文件

└mainleft.php 系统主页左侧分解文件

uploadfiles 上传目录

┌─album 相册存放目录

├─group_logo 群组 logo 目录

└─photo_store 其它图片目录

附录

关于二、三范式的参考：

第二范式（2NF）：如果关系模式 R（U，F）中的所有非主属性都完全依赖于任意一个候选关键字，则称关系 R 是属于第二范式的。

例：选课关系 SCI（SNO，CNO，GRADE，CREDIT）其中 SNO 为学号， CNO 为课程号，GRADEGE 为成绩，CREDIT 为学分。 由以上条件，关键字为组合关键字（SNO，CNO）在应用中使用以上关系模式有以下问题：

a.数据冗余，假设同一门课由 40 个学生选修，学分就 重复 40 次。

b.更新异常，若调整了某课程的学分，相应的元组 CREDIT 值都要更新，有可能会出现同一门课学分不同。

c.插入异常，如计划开新课，由于没人选修，没有学号关键字，只能等有人选修才能把课程和学分存入。

d.删除异常，若学生已经结业，从当前数据库删除选修记录。某些门课程新生尚未选修，则此门课程及学分记录无法保存。

原因：非关键字属性 CREDIT 仅函数依赖于 CNO，也就是 CREDIT 部分依赖组合关键字（SNO，CNO）而不是完全依赖。

解决方法：分成两个关系模式 SC1（SNO，CNO，GRADE），C2（CNO，CREDIT）。新关系包括两个关系模式，它们之间通过 SC1 中的外关键字 CNO 相联系，需要时再进行自然联接，恢复了原来的关系

第三范式（3NF）：如果关系模式 R（U，F）中的所有非主属性对任何候选关键字都不存在传递信赖，则称关系 R 是属于第三范式的。

例：如 S1（SNO，SNAME，DNO，DNAME，LOCATION） 各属性分别代表学号，

姓名，所在系，系名称，系地址。

关键字 SNO 决定各个属性。由于是单个关键字，没有部分依赖的问题，肯定是 2NF。但这关系肯定有大量的冗余，有关学生所在的几个属性 DNO，DNAME，LOCATION 将重复存储，插入，删除和修改时也将产生类似以上例的情况。

原因：关系中存在传递依赖造成的。即 SNO -> DNO。 而 DNO ->SNO 却不存在，DNO -> LOCATION, 因此关键辽 SNO 对 LOCATION 函数决定是通过传递依赖 SNO ->

LOCATION 实现的。也就是说，SNO 不直接决定非主属性 LOCATION。

解决目地：每个关系模式中不能留有传递依赖。

解决方法：分为两个关系 S（SNO，SNAME，DNO），D（DNO，DNAME，LOCATION）

注意：关系 S 中不能没有外关键字 DNO。否则两个关系之间失去联系。