### ucenter会员同步登录通信原理：

<http://www.jb51.net/article/54297.htm>

UCenter是Comsenz旗下各个产品之间信息直接传递的一个桥梁,通过UCenter 站长可以无缝整合Comsenz系列产品,实现用户的一站式登录以及社区其他数据的交互。其实何止是comsenz系列产品，可以这么说整合了 ucenter 就等于整合了整个php 界的所有php 程序。

1. 用户登录discuz，通过logging.php文件中的函数uc\_user\_login对post过来的数据进行验证，也就是对username和password进行验证。
2. 如果验证成功，将调用位于uc\_client下client.php文件中的函数uc\_user\_synlogin，在这个函数中调用 uc\_api\_post('user', 'synlogin', array('uid'=>$uid))。
3. 然后这个函数后向Ucenter的index.php传递数据,index.php接受传递的数据，获得model为user，action为synlogin的值。
4. 然后Ucenter的index.php调用control目录下的user.php类中的onsynlogin方法，通过foreach循环，以javascript的方式通知uc应用列表中开启同步登陆的应用进行同步登录；即通过get方式传递给各个应用目录中api下的uc.php一些数据。
5. uc.php接收通知并处理get过来的数据，并在函数synlogin（位于uc.php中）通过函数\_authcode加密数据（默认以UC\_KEY作为密钥），用函数\_setcookie设置cookie。

6、各个应用用对应的密钥解码上面设置的cookie，得到用户id等数据；通过这个值来判断用户是否经过其它应用登录过，从而让用户可以自动登陆。

应用程序的logging.php ------>uc\_client中的client.php------>Ucenter------>其他应用程序中的api/uc.php。

其实Ucenter实现同步登陆的原理就是cookie，一个应用登陆成功之后，向Ucenter传递数据，让Ucenter通知其他的应用也设置cookie，这样用户在访问其他应用的时候通过已经设置好的cookie实现自动登陆。

大致步骤 ，首先要安装 ucenter 然后把uc\_client 这个文件夹复制到自己的项目里面去 ，然后呢在配置几个文件

client.php相当于函数库

uc.php相当于回调文件

config.php 是配置文件

当你有2个应用都设置了同步登陆之后 当你登陆一个应用 然后执行

include './config.inc.php';

include './uc\_client/client.php';

$usernames="kyee";

$passwords="123456";

list($uid, $username, $password, $email) = uc\_user\_login($usernames, $passwords);

if($uid > 0) {

setcookie("username",$username,time()+intval(24\*3600));

echo uc\_user\_synlogin($uid);

echo '登录成功';

} elseif($uid == -1) {

echo '用户不存在,或者被删除';

} elseif($uid == -2) {

echo '密码错误';

} else {

echo '未定义';

}

uc\_user\_synlogin() 这个函数 代表着 要同步登陆到其他所有开启同步登陆的函数 uc自己会在后台把所有开启同步登陆的应用都给循环遍历一遍 然后 在页面上输出

<script type="text/javascript" src="http://rayibeauty.ck101.com/api/uc.php?time=1408327309&code=bc6bFLa6WH343nin2GAn%2F82Y9cnCennPk1gcLGYHdQF4wsXsOSdTyqBb2Nuoxe0UJqzWMWncdx%2FfQ1GK6FS%2BqJqi2AxVG2Oq1pD9c1wZy%2BgjXs7qo4mm2sxFVHwW7JnjKGPDkVdDqtYeybkSISz7yrdb0ZFuXH2yr3Cq" reload="1"></script>

<script type="text/javascript" src="http://kibeauty.ibeauty.tw/api/uc.php?time=1408327309&code=206flCqeb%2Faft%2FDFPno9Bvqsb1b0o6XTZdIByOoD7EC11vMrjzC7PaKLo0LF3tGiHwlwZkwdW5VDHq866MGulsco5nekfkL341VWp7BPabnZPNtgG7m4jZpfdx6nVP0LTJLYI%2BkebI7uLm58atk8Ex4sKBj%2FfDkjH%2F8z" reload="1"></script>

类似这种的js代码 ，就是发送给每个开启同步登陆的应用， 然后每个开启同步登陆的应用的回调文件 uc.php 接受到后会进行解密，解密好后其实你就可以自己来写代码了，这个uc.php回调文件的代码不一定非要按照他们的格式来写 ，你也可以自己写你自己的代码 。

其实UC的原理很简单 ，就是某个应用登陆后，然后后台轮询发送给同步登陆的应用的回调文件 ，回调文件接收到用户ID之后，生成cookie或者session然后进入登陆模式。

### ucenter整合

http://www.cnblogs.com/linzhenjie/archive/2012/08/25/2655572.html

### json详解

<http://www.jb51.net/article/59875.htm>

<1>

{“key”:”value”} key只能用双引号

<2>

$a = '{ "name" : "qttc", "url" : "www.jb51.net"  }';  
$a = json\_decode($a);  
print\_r($a);

打印结果：

stdClass Object  
(  
    [name] => qttc  
    [url] => www.jb51.net  
)

结论：使用json对象时，name相当于属性，$a->name来获取qttc值；

<3>

JSON.parse($res) //JSON必须大写

后台echo json\_encode($str),前台ajax接收res，需要用json.parse(res)将json字符串解析js值或对象，才能在前端输出使用。

### session保存7天

if (isset($\_POST['remember']) && *trim*($\_POST['remember']) == 1) {  
 *ini\_set*('session.gc\_maxlifetime', 7 \* 24 \* 3600);  
 *setcookie*(*session\_name*(), *session\_id*(), *time*() + 7 \* 24 \* 3600, '/');  
}

### php命名规则

函数：用\_连接

变量：用小驼峰

类：大驼峰

方法：小驼峰

属性：小驼峰

### 用&&做if判断

原始：

**if ($a>0){**

**$b='This is test';**

**}**

用&&做：

$a>0 && $b=’ **This is test**’

计算机会先判断$a 是否为真，如果是，则执行后面的语句，如果否，后面的语句就没有执行的必要了；

### 关于BOM头

1、为文件添加BOM头：fwrite($resource, chr(0xEF).chr(0xBB).chr(0xBF));

2、在php文件中添加BOM头：

echo "\xEF\xBB\xBF"; // UTF-8 BOM

或 print(chr(0xEF).chr(0xBB).chr(0xBF));

3、详细：在Windows下用记事本之类的程序将文本文件保存为UTF-8格式时，记事本会在文件头前面加上几个不可见的字符（EF BB BF），就是所谓的BOM（Byte order Mark）。当系统看到这玩意的时候，就会觉得你这个文件是UTF-8编码的。

但PHP不能识别BOM头，所以在读取、包含或者引用这些文件时，会把BOM作为该文件开头正文的一部分。

### 获取文件后缀名的方法

1)、pathinfo($filename , PATHINFO\_EXTENSION);//推荐

pathinfo()详解：

$path\_parts = pathinfo('/www/htdocs/inc/lib.inc.php');  
  
echo $path\_parts['dirname']; 输出：/www/htdocs/inc  
echo $path\_parts['basename']; 输出：lib.inc.php  
echo $path\_parts['extension']; 输出：php  
echo $path\_parts['filename']; 输出：lib.inc

$options：

PATHINFO\_DIRNAME 输出：/www/htdocs/inc

PATHINFO\_BASENAME 输出：lib.inc.php

PATHINFO\_EXTENSION 输出：php

PATHINFO\_FILENAME 输出：lib.inc

basename指整个文件名包括filename和extension

2)、end(explode(".",$filename))

3)、strrchr($filename,”.”);//带“.”的后缀名

4)、substr($filename , strtpos($filename , ’.’)+1)

5)、substr( strrchr($filename , ’.’) , 1 )

### 正则表达式

^与$ 都是指原字符串的开头与结尾，并不是匹配到的结果的开头和结尾

要匹配的字符串要在原字符串的开头或结尾

例：$str = "http://www.baidu.com/index/home/test.php";  
$num=*preg\_match*('/\/([^\/]+\.[a-z]+)[^\/]\*$/',$str,$match);  
echo $num.'<br>';  
*print\_r*($match);

此处不能加^ , ^指的是$str的开头

### 获取url地址信息的方法

$url = "http://www.baidu.com/index/home/test.php";

#### 获取文件名

正则：

preg\_match('/\/([^\/]+\.[a-z]+)[^\/]\*$/',$url,$match)

$match打印结果为：

Array(  
 [0] => /test.php?act=add  
 [1] => test.php  
 )

$path = "/testweb/home.php";

basename($path); //获得home.php

basename($path,’.php’) //获得home

相反dirname是获得目录名

dirname($path); //输出 /testweb (最后没有/)

#### 2) 获取扩展名

**parse\_url();**

parse\_url($url) 本函数解析一个 URL 并返回一个关联数组，包含在 URL 中出现的各种组成部分。

$arr = parse\_url($url);

结果为一个数组



$str = explode(‘.’,$arr[‘path’])；

$ext = $str[1];//获得扩展名

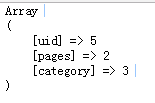
**parse\_str(string $str [, array &$arr ] )**

$str = ‘uid=5&pages=2&category=3’;

parse\_str($\_SERVER['QUERY\_STRING'],$arr);

print\_r($arr)

结果：



### php事务处理

MYSQL的事务处理主要有两种方法。

1、用begin,rollback,commit来实现（推荐）

begin 开始一个事务

rollback 事务回滚

commit 事务确认

2、直接用set来改变mysql的自动提交模式

MYSQL默认是自动提交的，也就是你提交一个QUERY，它就直接执行！我们可以通过

set autocommit=0 禁止自动提交

set autocommit=1 开启自动提交

来实现事务的处理。

当你用 set autocommit=0 的时候，你以后所有的SQL都将做为事务处理，直到你用commit确认或rollback结束。

注意当你结束这个事务的同时也开启了个新的事务！按第一种方法只将当前的作为一个事务！

开启InnoDB引擎：

MYSQL中只有INNODB和BDB类型的数据表才能支持事务处理！其他的类型是不支持的！  
\*\*\*：一般MYSQL数据库默认的引擎是MyISAM,这种引擎不支持事务！如果要让MYSQL支持事务，可以自己手动修改:  
方法如下：1.修改c:\appserv\mysql\my.ini文件，找到skip-InnoDB,在前面加上#，后保存文件。  
2.在运行中输入：services.msc,重启mysql服务。  
3.到phpmyadmin中，mysql->show engines;(或执行mysql->show variables like 'have\_%'; ),查看InnoDB为YES,即表示数据库支持InnoDB了。  
也就说明支持事务transaction了。  
4.在创建表时，就可以为Storage Engine选择InnoDB引擎了。如果是以前创建的表，可以使用mysql->alter table table\_name type=InnoDB;  
或 mysql->alter table table\_name engine=InnoDB;来改变数据表的引擎以支持事务。

|  |
| --- |
| **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* transaction--1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** $conn = mysql\_connect('localhost','root','root') or die ("数据连接错误!!!"); mysql\_select\_db('test',$conn); mysql\_query("set names 'GBK'"); //使用GBK中文编码; //开始一个事务 mysql\_query("BEGIN"); //或者mysql\_query("START TRANSACTION"); $sql = "INSERT INTO `user` (`id`, `username`, `sex`) VALUES (NULL, 'test1', '0')"; $sql2 = "INSERT INTO `user` (`did`, `username`, `sex`) VALUES (NULL, 'test1', '0')";//这条我故意写错 $res = mysql\_query($sql); $res1 = mysql\_query($sql2);  if($res && $res1){ mysql\_query("COMMIT"); echo '提交成功。'; }else{ mysql\_query("ROLLBACK"); echo '数据回滚。'; } mysql\_query("END");  **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* transaction--2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** /\*方法二\*/ mysql\_query("SET AUTOCOMMIT=0"); //设置mysql不自动提交，需自行用commit语句提交 $sql = "INSERT INTO `user` (`id`, `username`, `sex`) VALUES (NULL, 'test1', '0')"; $sql2 = "INSERT INTO `user` (`did`, `username`, `sex`) VALUES (NULL, 'test1', '0')";//这条我故意写错 $res = mysql\_query($sql); $res1 = mysql\_query($sql2);  if($res && $res1){ mysql\_query("COMMIT"); echo '提交成功。'; }else{ mysql\_query("ROLLBACK"); echo '数据回滚。'; } mysql\_query("END"); //事务处理完时别忘记mysql\_query("SET AUTOCOMMIT=1");自动提交 |

### COOKIE中不能存数组

必须先将数组serialize()串行化成字符串，才能setcookie（）;

但是SESSION可以存数组；

### 删除数组元素方法

$array = array(0 => "a", 1 => "b", 2 => "c");

unset($array[1]);

array\_splice($array,0,1);//array\_splice($arr,$offset,$len[,$replace])从$offset处开始去除，去除长度是$len,若有$replace，则用它代替去掉的元素。

### 关于绝对路径、相对路径

realpath(dirname(\_\_FILE\_\_) . '/../../')；指返回上两级目录； （好像有几个../就会从后查出几个目录）

### file\_get\_contents()发送请求

file\_get\_contents($filename);//将文件读入一个字符串

file\_get\_contents($url);//可以发送get和post请求

1. 发送get请求

$url = 'http://www.baidu.com';  
$result = *file\_get\_contents*($url);

echo $result;  
//若得到的是个xml格式的字符串，可以将其转成xml对象  
//$xml = *simplexml\_load\_string*($result);

1. 发送post请求

$urn = 'http://www.example.com';

//post需要发送的数据

$data = array(

'para1' => 'example1',

'para2' => 'example2'

);

//定义post设置数组

$options = array(

'Http' => array(

'method' => 'POST',

'header' => 'Content-type: application/x-www-form-urlencoded',

'content' => http\_build\_query($data) //生成URL-encode之后的请求字符串,就是?后的参数

)

);

//创建资源流上下文,上下文可理解为发送请求所需要的环境

$context = stream\_context\_create($options);//返回的是资源

$result = file\_get\_content($url, false, $context);

echo $result;

### file\_put\_contents()

$data是个字符串；

file\_put\_contents($filepath,$data);若$filepath不存在则会自动创建

### ASKII编码转换

ord():返回字符串首个字母的ASKII值。

chr():将ASKII码转成字符。

### 字符串函数

#### http\_bulid\_query() 生成 URL-encode 之后的请求字符串

$data = array('foo'=>'bar',

'baz'=>'boom',

'cow'=>'milk',

'php'=>'hypertext processor');

echo http\_build\_query($data) . "\n";

输出：foo=bar&baz=boom&cow=milk&php=hypertext+processor

echo http\_build\_query($data, '', '&amp;');

输出：foo=bar&amp;baz=boom&amp;cow=milk&amp;php=hypertext+processor

#### strcmp($str1,$str2) 和strcasecmp($str1,$str2)、strncmp($str1,$str2,$len)字符串比较函数

三个函数比较相似，按照字节的ASCII值从两个字符串的首字节开始比较，如果相等则进入下一个字节的比较，直至结束比较。

例：echo strcmp('Hello','hEllo'); //输出-1 ，第一个字节就不相等，后面就不用比了

echo strcmp('hEllo','Hello'); //输出1

区别：strcmp()区分大小写，strcasecmp()不区分大小写；

strcmp()返回值只有0，+1，-1；

strcasecmp()返回值是将字符串全部转化成小写，在根据ASKII码值的差值计算得来。

例：echo strcasecmp('A','1');//ASKII码表里1和a的差是48；所以返回值是 -48

echo strcasecmp('a','1');//返回值也是 -48

strncmp($str1,$str2,$len) 能指定最大比较长度，用法跟strcmp()相同

#### strtr（$str,$arr）转换字符串中特定字符

<?php

$arr = array("Hello" => "Hi", "world" => "earth");

echo strtr("Hello world",$arr); //输出 Hi,earth

?>

#### printf() 和sprintf()

printf() 函数输出格式化的字符串。

sprint()函数把格式化的字符串写入变量中。

$number = 123;

$txt = sprintf("%f",$number);

Echo $txt;

Sprintf()详细解释：

sprintf("%格式化类型","$str1","$str2")





例子1： %   '（补位值） 宽度值  格式化类型

<?php

$a="abcdef";

$b="abcdef";

$c="1234";

echo sprintf("%'x13.2f",$c);

// 效果为：xxxxxx1234.00

?>

例子2： %    宽度值  格式化类型 （无需补位）

<?php

$a="abcdef";

$b="abcdef";

$c="1234";

echo sprintf("[%-6s]",$c); //结果是：[1234 ]

echo sprintf("[%-4s]",$c); //结果是：[1234]

echo sprintf("[%4.2s]",$c); //结果是：[ 12]

echo sprintf("[%08s]",$a)

?>

第一行：宽度为6，$c只有4个字符，宽度不够，右边自动扩充（扩充再多也只显示一个空格）。因为前面有个 – 号，所以向右扩充

第三行：4代表总宽度为4 , 2代表显示的字符串长度为2，所以显示12，总长度为4，所以向左扩充（默认向左）2个空格，但扩充再多也只显示一个空格。

第四行：0代表填充字符，8代表总宽度，结果为 [00abcdef] (默认向左填充)

#### hash（）

hash($algo,$str)

$algo为加密算法，常用的有sha1、MD5、sha256

Sha1有专门的php函数: Sha1($str)

#### hash\_hmac($algo,$data,$key)

生成带有秘钥的哈希值

$algo:哈希算法名，md5,sha256等

$data:加密的内容

$key:密钥

#### base64\_encode(string $data)

base64除了可以用来加密，它还可以把某些二进制数转成普通字符用于网络传输。由于这些二进制字符在传输协议中属于控制字符，不能直接传送，所以需要转换一下。虽然图片可能直接传输，但是我们也可以将它变成字符串直接放在源码里，而不需要浏览器在读取到源码后再从服务器上下载。

#### urlencode()

urlencode():除了 -\_. 之外的所有非字母数字字符都将被替换成百分号（%）后跟两位十六进制数,空格编码为+。

#### iconv()

string iconv ( string in\_charset, string out\_charset, string str )

第二个参数可以增加两个后缀：//TRANSLIT 和 //IGNORE，其中 //TRANSLIT 会自动将不能直接转化的字符变成一个或多个近似的字符，//IGNORE 会忽略掉不能转化的字符，而默认效果是从第一个非法字符截断（即遇到不能转化的字符，之后的所有字符都不能输出）。

iconv(‘utf-8’,’gb2312//IGNORE’,$str);

#### string mb\_convert\_encoding (string str, string to\_encoding [, mixed from\_encoding])

可以指定多种输入编码，它会根据内容自动识别,但是执行效率比iconv差太多；

注：一般情况下用 iconv，只有当遇到无法确定原编码是何种编码，或者iconv转化后无法正常显示时才用mb\_convert\_encoding 函数.

#### str\_getcsv() 解析csv字符串为数组

|  |
| --- |
| array str\_getcsv ( string $input [, string $delimiter = "," [, string $enclosure = '"' [, string $escape = "\\" ]]] )  不同于explode（），可以设置包裹字符，将被包裹的内容看成整体，不被$delimiter分割  input 待解析的字符串  delimiter 设定字段界定符(仅单个字符)  enclosure 设定字段包裹字符（仅单个字符）  escape 设置转义字符（仅单个字符），默认为反斜线（\）  $str = "中国,广东省,广州市,天河区,'113.329884,23.154799',1,'2016-01-01 12:00:00','1,2,3,4,5,6'";  $arr = str\_getcsv($str, ',', "'");  print\_r($arr);  输出：  Array  (  [0] => 中国  [1] => 广东省  [2] => 广州市  [3] => 天河区  [4] => 113.329884,23.154799  [5] => 1  [6] => 2016-01-01 12:00:00  [7] => 1,2,3,4,5,6  ) |

#### str\_word\_count($str)

统计字符串中含有的单词数

### 系统函数

#### addslashes($str)

返回字符串，当你要向数据库插入数据时,需要在某些字符前加上了反斜线。这些字符是单引号（'）、双引号（"）、反斜线（\）与 NUL（NULL 字符）。

#### htmlspecialchars()将特殊字符转换为 HTML 实体，

转换规则：

& ->&amp

“ ->&quot

‘ ->&039或&apos

< ->&lt

* ->&gt

例：



#### htmlentities()转换所有具有HTML实体的字符，用法跟htmlspecialchars()一样。

#### strip\_tags()去除空字符、HTML 和 PHP 标签

#### set\_include\_path(get\_include\_path().PATH\_SEPARATOR…)

下次include文件时，添加过的包含目录内的文件可以不加文件名

1）ini\_set(‘include\_path’, 'd:\www\mysite\test’);

设置之后，之后include或require此目录下的文件是就不用带目录了

ini\_set() : 为一个配置选项设置值 修改php.ini文件

这个选项会在脚本运行时保持新的值，并在脚本结束时恢复。

或者

2）set\_include\_path(get\_include\_path().PATH\_SEPARATOR.PUBLIC\_DIR);

PATH\_SEPARATOR： windows下为 ; 号 linux下为 : 号

[get\_include\_path()](http://php.net/manual/zh/function.get-include-path.php) - 获取当前的 include\_path 配置选项

**补充说明：**当使用include()或require()函数包含文件的时候,程序首先以include\_path设置的路径作为参考点去找文件,如果找不到,则以程序自身所在的路径为参考点去找所要的文件,如果都找不到,则出错.

#### http\_build\_query($data)

将$data数组的键名和键值用“=”连接后再用”&”连接，可用于拼接url地址的参数部分

<?php  
$data = array('foo'=>'bar',  
              'baz'=>'boom',  
              'cow'=>'milk',  
              'php'=>'hypertext processor');  
  
echo http\_build\_query($data) . "\n";  
echo http\_build\_query($data, '', '&amp;');  
  
?>

#### $\_SERVER

<http://www.5idev.com/p-php_server_php_self.shtml>

$\_SERVER[‘REQUEST\_URI’]**=**$\_SERVER['PHP\_SELF']. $\_SERVER[‘QUERY\_STRING’]

$\_SERVER['PHP\_SELF']: host后面的，?号前面的部分

http://www.5idev.com/php/ ：/php/index.php

http://www.5idev.com/php/index.php ：/php/index.php

http://www.5idev.com/php/index.php?test=foo ：/php/index.php

http://www.5idev.com/php/index.php/test/foo ：/php/index.php/test/foo

注意：它不包含url中？及后面的参数

$\_SERVER[‘REQUEST\_URI’] ：host后面的所有

例：http://www.5idev.com/php/index.php?test=foo ：/php/index.php?test=foo

$\_SERVER[‘QUERY\_STRING’]: ？后面的内容

例：http://www.5idev.com/php/index.php?test=foo ： test=foo

$\_SERVER['HTTP\_HOST'] : Host 头部的内容。

显示：1270.0.01:8080

$\_SERVER['HTTP\_REFERER']  链接到当前页面的前一页面的 URL 地址。

$\_SERVER[‘SCRIPT\_NAME’] 当前的文件名

例：http://www.5idev.com/php/index.php?test=foo ： /php/index.php

//获取完整的url  
echo 'http://'.$\_SERVER['HTTP\_HOST'].$\_SERVER['REQUEST\_URI'];  
echo 'http://'.$\_SERVER['HTTP\_HOST'].$\_SERVER['PHP\_SELF'].'?'.$\_SERVER['QUERY\_STRING'];  
#http://localhost/blog/testurl.php?id=5

$\_SERVER[‘DOCUMENT\_ROOT’] 获取当前文件所在的目录，且都是左斜杠/

相比str\_replace('\\','/',dirname(\_\_FILE\_\_)) 速度更快

#### $GLOBALS['HTTP\_RAW\_POST\_DATA']

$GLOBALS : 一个包含了全部变量的全局组合数组。变量的名字就是数组的键。

基本上$GLOBALS['HTTP\_RAW\_POST\_DATA'] 和 $\_POST是一样的。但是如果post过来的数据不是PHP能够识别的，你可以用$GLOBALS['HTTP\_RAW\_POST\_DATA']来接收，比如 text/xml 或者 soap 等等。

知识点：

1，Content-Type仅在取值为application/x-www-data-urlencoded和multipart/form- data两种情况下，PHP才会将http请求数据包中相应的数据填入全局变量$\_POST

2，PHP不能识别的Content-Type类型的时候，会将http请求包中相应的数据填入变量$HTTP\_RAW\_POST\_DATA

3, 只有Content-Type不为multipart/form-data的时候，PHP不会将http请求数据包中的相应数据填入php: //input，否则其它情况都会。填入的长度，由Content-Length指定。

4，只有Content-Type为application/x-www-data-urlencoded时，php://input数据才 跟$\_POST数据相一致。

5，php://input数据总是跟$HTTP\_RAW\_POST\_DATA相同，但是php://input 比$HTTP\_RAW\_POST\_DATA更凑效，且不需要特殊设置php.ini

6，PHP会将PATH字段的query\_path部分，填入全局变量$\_GET。通常情况下，GET方法提交的http请求，body为空。

#### usleep()

usleep(microseconds);延迟代码执行若干微秒。

### 数字函数

#### number\_format(number,decimals)

Number:要格式化的数字

Decimals:规定多少个小数

例：echo number\_format("5000000",2);

’结果：5,000,000.00

#### intval( $var )

获取数据的整数值；

#### bcadd( $a,$b,$scale)

将两个高精度数字相加

$scale : 设置小数点的位数；等于直接保留相应的小数点位数后再进行计算

结果：$a+$b;

其他类似函数：

bcsub — 将两个高精度数字相减；

bcmul — 将两个高精度数字相乘

bcdiv — 将两个高精度数字相除

bccomp — 比较两个高精度数字，返回-1, 0, 1

bcscale — 配置默认小数点位数

注意点：关于设置的位数，超出部分是丢弃掉，而不是四舍五入。

#### uniqid() 获取一个带前缀、基于当前时间微秒数的唯一ID。

### 时间函数

#### strtotime()

strtotime(‘today’)：当天00:00:00的时间戳，所以time()一定比strtotime(‘today’)大

strtotime('-10 minute')：距time()十分钟前的时间戳；

### 数组函数

#### array\_rand(array $input [, int $num\_req = 1 ])

从数组中取出一个或多个随机单元，返回键名key，而不是返回数组

#### implode(‘,’,$arr)

当$arr中只有一个元素时，最终的字符串是不加分隔符‘,’的;

#### array\_map()

array array\_map ( [callable](http://php.net/manual/zh/language.types.callable.php) $callback , array $array1 [, array $... ] )

array\_map()：返回数组，是为 **array1** 每个元素应用 **callback**函数之后的数组。（不改变原数组）

**callback** 函数形参的数量和传给 array\_map() 数组数量，两者必须一样。

多数组举例：

function show\_Spanish($n, $m)  
{  
    return("The number $n is called $m in Spanish");  
}

$a = array(1, 2, 3, 4, 5);

$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array\_map("show\_Spanish", $a, $b);

print\_r($c);

输出：

Array

(

[0] => The number 1 is called uno in Spanish

[1] => The number 2 is called dos in Spanish

[2] => The number 3 is called tres in Spanish

[3] => The number 4 is called cuatro in Spanish

[4] => The number 5 is called cinco in Spanish

)

#### array\_filter()

array array\_filter ( array $array [, [callable](http://php.net/manual/zh/language.types.callable.php) $callback [, int $flag = 0 ]] )

依次将 **array** 数组中的每个值传递到 **callback** 函数。如果 **callback** 函数返回 true，则 **array** 数组的当前值会被包含在返回的结果数组中。数组的键名保留不变。

返回：过滤后的新数组

情况1：

function odd($var)  
{  
        return($var & 1);  
}

$array1 = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);

print\_r(array\_filter($array1, "odd"));

输出：

Array

(

[a] => 1

[c] => 3

[e] => 5

)

情况2：

如果没有提供 **callback** 函数， 将删除 **array** 中所有等值为 **FALSE** 的条目。更多信息见[转换为布尔值](http://php.net/manual/zh/language.types.boolean.php#language.types.boolean.casting)。

$entry = array(  
             0 => 'foo',  
             1 => false,  
             2 => -1,  
             3 => null,  
             4 => ''  
          );  
  
print\_r(array\_filter($entry));

输出：

Array

(

[0] => foo

[2] => -1

)

#### end($arr)

end() 将 array 的内部指针移动到最后一个单元并返回其值。

$arr = array('a','b','c','d','e');  
echo *end*($arr);//输出 e

#### var\_export($arr)

此函数返回关于传递给该函数的变量的结构信息，它和 var\_dump() 类似，不同的是其返回的表示是合法的 PHP 代码。

例：



#### array\_diff()

Array\_diff($arr1,$arr2)对比返回在array1但不在array2中的值，

图中蓝色部分：

Arr1

Arr2

#### array\_walk()

使用用户自定义函数对数组中的每个元素做回调处理

bool array\_walk ( array &$array , callable $funcname [, mixed $userdata = NULL ] )

回调函数一般接收两个参数：

function test($val,$key){//第一个参数是数组的值，第二个是对应的key

}

例：

|  |
| --- |
| $arr = [1,2,3,4,5];  function add(&$v,$k)  {  $v = $v+1;  }  array\_walk($arr,'add');  print\_r($arr);  //结果：    若是个匿名函数  $add = function (){  …  }  array\_walk($arr,$add)//此时第二个参数用$add; |

#### array\_multisort（）对多维数组排序

bool array\_multisort ( array &$arr [, mixed $arg = SORT\_ASC [, mixed $arg = SORT\_REGULAR [, mixed $... ]]] )

|  |
| --- |
| $a1 = array('a'=>1,'b'=>8,'c'=>5,'d'=>8);  $a2 = array('e'=>2,'f'=>2,'g'=>3,'h'=>5);  array\_multisort（$a1,$a2）;    分析：a和e永远对应，c和g永远对应。。。可以把每个数组想像成数据库表的一列。每一行的对应关系不会因为排序改变。第二个数组会跟着第一个数组的排序而排序 |

参数：每个数组后面都能加两个参数，默认是SORT\_ASC ，SORT\_REGULAR；

若第一个数组按升续，第二个按降续，则第二个数组首先会按照跟$arr原有的关联关系排序，当第一个数组有相同值的情况下，对应第二个数组才会按降续排列

|  |
| --- |
| $guys =[  ['name'=>'jim','score'=>14,'grade'=>2],  ['name'=>'jack','score'=>12,'grade'=>3],  ['name'=>'tom','score'=>34,'grade'=>3],  ['name'=>'kaka','score'=>9,'grade'=>5],  ['name'=>'john','score'=>12,'grade'=>4],  ];  foreach ($guys as $row) {  $arr['name'][]=$row['name'];  $arr['score'][]=$row['score'];  $arr['grade'][]=$row['grade'];  }  array\_multisort($arr['score'],SORT\_DESC,$arr['grade'],$arr['name']); |

#### compact()

建立一个数组，包含变量名和他的值

|  |
| --- |
| $name = 'jim';  $age = 12;  $gender = 'male';  $arr = compact('name','age','gender');  print\_r($arr); |
| 2 方便生成一个关联数组  $name = [‘tom’,’lili’,’john’];  print\_r(compact(‘name’));  结果： |

#### array\_merge()的那些坑

1. 键名相同的话，后面的数组会将前面数组中键相同的值覆盖。

|  |
| --- |
| $arr1 = array('a'=>'1','b'=>'2');  $arr2 = array('a'=>'1','b'=>'3','c'=>'2');  $ret = array\_merge($arr1,$arr2);  print\_r($ret);  Array  (  [a] => 1  [b] => 3  [c] => 2  ) |

1. 数组中的键名为数字时(索引数组也一样)，会格式化键名并保留所有键值。

|  |
| --- |
| $arr1 = array(1=>'1',2=>'2');  $arr2 = array(1=>'1',2=>'3',6=>'2');  $ret = array\_merge($arr1,$arr2);  print\_r($ret);  Array  (  [0] => 1  [1] => 2  [2] => 1  [3] => 3  [4] => 2  ) |

1. 若想要键名重复的时候保留前者，使用 +

|  |
| --- |
| $arr1 = array('a'=>1,'b'=>2);  $arr2 = array('a'=>1,'b'=>3,'c'=>6);  $ret = $arr1+$arr2;  print\_r($ret);  Array  (  [a] => 1  [b] => 2  [c] => 6  ) |

注：array\_merge()后面覆盖前面，+ 前面的覆盖后面

#### array\_sum() 数组中所有值的总和

#### array\_key\_exists(): 判断某个数组中是否存在指定的 key

### 文件函数

#### glob()

glob函数能快速查询指定目录文件

$array = glob('\*.\*');//查当前目录下的所有文件，以索引数组形式返回

$array = glob('\*.php');//查询当前目录下以.php结尾的文件

$files = glob('../05-15/1\*.php');// 在指定目录搜索以1开的php文件

#### file()

file($filename);//将整个文件读入一个索引数组中，每一行为一个元素。

#### getcwd() 获取当前运行脚本的目录 get current directory

getcwd()返回的是运行它的文件所在的目录，会因为包含而变化；而\_\_DIR\_\_返回的是getcwd()所在的那个文件的目录，不会因为包含而变化

#### scandir($path) 返回指定目录中的文件和目录的数组。

php读取出一个文件夹及其子文件夹下所有文件的方法:

|  |
| --- |
| function scanFile($path) {  global $result;  $files = scandir($path);  foreach ($files as $file) {  if ($file != '.' && $file != '..') {  if (is\_dir($path . '/' . $file)) {  scanFile($path . '/' . $file);  } else {  $result[] = basename($file);  }  }  }  return $result;  } |

### 对象函数

#### get\_object\_vars()

获取对象属性组成的关联数组；

在类的内部使用，可以获取到除静态属性外的所有属性；

在类的外部使用，只能获取到公共属性

|  |
| --- |
| class Test  {  private $age =12;  protected $gender = 'female';  public $name = 'John';  static $height = 176;  public function test()  {  print\_r(get\_object\_vars($this));  echo '<hr>';  }  }  $test = new Test();  $test->test();  $res = get\_object\_vars($test);  print\_r($res); |

### $GLOBALS与global区别详解

源：http://www.thinkphp.cn/code/1018.html

主题：global $var 其实就是$var = &$GLOBALS['var']。调用外部变量的一个别名而已。

官方解释：$GLOBALS['var'] 是外部的全局变量$var本身。

global $var 是外部$var的同名引用或者指针。（错误：是个别名引用而已，非指针！！！）

例1：

<?php

$var1 = 1;

$var2 = 2;

function test() {

$GLOBALS['var2'] = &$GLOBALS['var1'];

}

test();

echo $var2;

?>

正常打印结果为1

-------------------------------------------------------------------------------

例2：

<?php

$var1 = 1;

$var2 = 2;

function test(){

global $var1, $var2;

$var2 = &$var1;

echo $var2;

$var2 = 'snsgou.com';

}

test(); // 输出 1

echo $var2; // 输出 2

echo $var1; // 输出 snsgou.com

?>

注：test()函数中的$var1，$va2都是局部变量，只不过是加了global关键字后，分别引用指向全局变量$var1，$va2了，当 $var2 = &$var1; 时，局部变量$var2不再指向全局变量$val2，而重新指向全局变量$var1，换句话来说，局部变量$var2的改变，不会再影响到全局变量$val2，而会影响到重新指向的全局变量$val1。

-------------------------------------------------------------------------------

例3：

$var1 = 1;

function test(){

global $var1;

unset($var1);

}

test();

echo $var1;//打印结果为1

证明删除的只是别名，$GLOBALS['var']的引用，其本身的值没有受到任何的改变。即断开了引用与全局变量之间的联系；

### 关于引用

$a="hihaha";  
$b= &$a;  
$c="eita";  
$b=$c;  
//$a、$b都指向‘hihaha’ ，当$b的值变为eita时，即是将原先内存上的‘hihaha’给抹去，重新赋上新值eita  
echo $a;// shows "eita"  
  
$a="hihaha";  
$b= &$a;  
$c="eita";  
$b= &$c;//$b对$a的引用断开了，重新指向$c的内存空间  
echo$a;// shows "hihaha"  
  
  
$a="hihaha";  
$b= &$a;  
$b=null;//$b指向$a的内存空间，$b=null,即把$a的内存空间内容给清空了  
echo$a;// shows nothing (both are set to null)  
  
$a="hihaha";  
$b= &$a;  
$a=null;  
echo$b;// shows nothing (both are set to null)  
  
$a="hihaha";  
$b= &$a;  
unset($b);//unset 只是断开了$b到$a内存空间的连接，不同于$b=null  
echo$a;// shows "hihaha"  
  
$a="hihaha";  
$b= &$a;  
$c="eita";  
$a=$c;//$a内存空间的值变成了 eita，$b也同样指向这个内存空间；  
echo$b;// shows "eita"  
  
$a="hihaha";  
$b= &$a;  
$c="eita";  
$a= &$c;//这个有别于上面的。$a现在不指向'hihaha'内存地址了，转而指向eita的内存地址，但是$b并没有变，依旧指向hihaha的地址  
echo$b;// shows "hihaha"  
  
  
$a="hihaha";  
$b= &$a;  
unset($a);//$a 断开了与‘hihaha’地址的连接，但$b并没有  
echo$b;// shows "hihaha"

### 定界符

<<<START //后面不能有空格

<body>

<!--12321-->

Hello,{$name}!

Hello,$name!

</body>

START; //最后一行要另起一行，顶格，前后都不能有任何字符。

实现：

可用来输出大段的html和javascript脚本

1.PHP定界符的作用就是按照原样，包括换行格式什么的，输出在其内部的东西；

2.在PHP定界符中的任何特殊字符都不需要转义；

3.PHP定界符中的PHP变量会被正常的用其值来替换。

### php闭包 Closure用法：

例1：

|  |
| --- |
| $f = function () {  return 100;  };  function B(Closure $callback)  {  return $callback();  }  $a = B($f);  print\_r($a);//输出100 |

实现闭包：将匿名函数在普通函数中当做参数传入，也可以被返回。这就实现了一个简单的闭包。

|  |
| --- |
| function printStr() {  $func = function( $str ) {  echo $str;  };  $func( ' hello my girlfriend ! ' );  }  printStr();//输出 hello my girlfriend ! |

连接闭包和外界变量的关键字：USE

|  |
| --- |
| function getMoney() {  $rmb = 1;  $dollar = 8;  $func = function() use ( $rmb ) {  echo $rmb;  echo $dollar;  };  $func();  }  getMoney();  //输出：1 $dollar读取不到  用了use并不能改变此变量，它仅仅是变量的一个副本clone而已，若想完全引用变量，前加&； |

调用类里的匿名函数：

|  |
| --- |
| class A {  public static function testA() {  return function($i) { //返回匿名函数  return $i+100;  };  }  }  function B(Closure $callback)  {  return $callback(200);  }  $a = B(A::testA());  print\_r($a);//输出 300 |

将匿名函数绑定到类里：

|  |
| --- |
| class A {  public $base = 100;  }  class B {  private $base = 1000;  }  $f = function () {  return $this->base + 3; //$this已经说明这个匿名函数要被绑定到类中去  };  $a = Closure::bind($f, new A);  print\_r($a());//输出 103  echo PHP\_EOL;  $b = Closure::bind($f, new B , 'B');  print\_r($b());//输出1003 |

|  |
| --- |
| class A {  public $base = 100;  }  class B {  private $base = 1000;  }  class C {  private static $base = 10000;  }  $f = function () {  return $this->base + 3;  };  $sf = static function() {  return self::$base + 3;  };  $a = Closure::bind($f, new A);  print\_r($a());//这里输出103,绑定到A类  echo PHP\_EOL;  $b = Closure::bind($f, new B , 'B');  print\_r($b());//这里输出1003，绑定到B类  echo PHP\_EOL;  $c = $sf->bindTo(null, 'C'); //注意这里：使用变量#sf绑定到C类，默认第一个参数为null  print\_r($c());//这里输出10003 |

终极实例：

|  |
| --- |
| /\*\*  \* 复制一个闭包，绑定指定的$this对象和类作用域。  \*  \* @author fantasy  \*/  class Animal {  private static $cat = "加菲猫";  private $dog = "汪汪队";  public $pig = "猪猪侠";  }  /\*  \* 获取Animal类静态私有成员属性  \*/  $cat = static function() {  return Animal::$cat;  };  /\*  \* 获取Animal实例私有成员属性  \*/  $dog = function() {  return $this->dog;  };  /\*  \* 获取Animal实例公有成员属性  \*/  $pig = function() {  return $this->pig;  };  $bindCat = Closure::bind($cat, null, new Animal());// 给闭包绑定了Animal实例的作用域，但未给闭包绑定$this对象（所以不能用$this,只能静态调取）  $bindDog = Closure::bind($dog, new Animal(), 'Animal');// 给闭包绑定了Animal类的作用域，同时将Animal实例对象作为$this对象绑定给闭包  $bindPig = Closure::bind($pig, new Animal());// 将Animal实例对象作为$this对象绑定给闭包,保留闭包原有作用域  echo $bindCat(),'<br>';// 输出：加菲猫，根据绑定规则，允许闭包通过作用域限定操作符获取Animal类静态私有成员属性  echo $bindDog(),'<br>';// 输出：汪汪队, 根据绑定规则，允许闭包通过绑定的$this对象(Animal实例对象)获取Animal实例私有成员属性  echo $bindPig(),'<br>';// 输出：猪猪侠, 根据绑定规则，允许闭包通过绑定的$this对象获取Animal实例公有成员属性 |

Closure::bind — 复制一个闭包，绑定指定的$this对象和类作用域。

|  |
| --- |
| 说明  public static Closure Closure::bind ( Closure $closure , object $newthis [, mixed $newscope = 'static' ] )  这个方法是 Closure::bindTo() 的静态版本。查看它的文档获取更多信息。  参数  closure  需要绑定的匿名函数。  newthis  需要绑定到匿名函数的对象，或者 NULL 创建未绑定的闭包。  newscope  想要绑定给闭包的类作用域，或者 'static' 表示不改变。如果传入一个对象，则使用这个对象的类型名。 类作用域用来决定在闭包中 $this 对象的 私有、保护方法 的可见性。（备注：可以传入类名或类的实例，默认值是 'static'， 表示不改变。）  返回值：  返回一个新的 Closure 对象 或者在失败时返回 FALSE |

### return

|  |
| --- |
| if(true){  reutn ;//表示执行退出；后面的代码就不再执行了  }  echo 11111; |

### 下载

1. file\_put\_contets($des,$data);直接将内容放进$des 文件内，页面没有任何提示，$des不存在会自动创建
2. header("Content-Disposition:attachment;filename=test.png");

echo $content;//$content是可以带header头的文件信息

此种方式，会在页面弹出另存为的窗口，可供选择存放目录

3）

### 类型转化

### 字符串转换为数值

1）如果该字符串没有包含 '.'，'e' 或 'E' 并且其数字值在整型的范围之内（由 PHP\_INT\_MAX 所定义），该字符串将被当成 integer 来取值。其它所有情况下都被作为 float 来取值。

2）该字符串的开始部分决定了它的值。如果该字符串以合法的数值开始，则使用该数值。否则其值为 0（零）。

|  |
| --- |
| <?php  $foo = 1 + "10.5"; // $foo is float (11.5)  $foo = 1 + "-1.3e3"; // $foo is float (-1299)  $foo = 1 + "bob-1.3e3"; // $foo is integer (1)  $foo = 1 + "bob3"; // $foo is integer (1)  $foo = 1 + "10 Small Pigs"; // $foo is integer (11)  $foo = 4 + "10.2 Little Piggies"; // $foo is float (14.2)  $foo = "10.0 pigs " + 1; // $foo is float (11)  $foo = "10.0 pigs " + 1.0; // $foo is float (11)  ?> |

### 转布尔值

**!!**$a :第一个!将$a装成相反的布尔值，两个！等于将$a转为布尔值。

### PHP代码优化测试

使用Benchmark测试；获取执行时间；

1）$row['id'] =0比 $row[id]=0 快

2）变量预定义比未定义的快

3）str\_replace比preg\_replace快

4）使用&引用更快

5）判断字符串长度用isset($str{15})比strlen($str) < 15更快，因为isset是语言结构，而strlen是函数，语言结构比函数快

6）$\_SERVER['DOCUMENT\_ROOT'] 比str\_replace('\\','/',dirname(\_\_FILE\_\_))快

7）$\_SERVER['REQUEST\_TIME'] 比time（）快很多

8）尽量使用单引号，因为双引号会检测里面是否含有变量

9）foreach用不到key的时候就不要加key