session就是一种工作机制，是一个全局变量

Session 的工作机制是：为每个访问者创建一个唯一的 id (UID)，并基于这个 UID 来存储变量。

session用来存储用户登录信息和用在跨页面传值。

1）常用在用户登录成功后，将用户登录信息赋值给session；

2）用在验证码图片生成，当随机码生成后赋值给session。

session在web开发中具有非常重要的份量。它可以将用户正确登录后的信息记录到服务器的内存中，当用户以此身份访问网站的管理后台时，无需再次登录即可得到身份确认。而没有正确登录的用户则不分配session空间，即便输入了管理后台的访问地址也不能看到页面内容。通过session确定了用户对页面的操作权限。

使用session的时候，通过什么来表示当前用户，从而与其他用户进行区分:

sessionid，通过session\_id()函数可以取得当前的session\_id。

开始 PHP Session

在您把用户信息存储到 PHP session 中之前，首先必须启动会话。

注释：session\_start() 函数必须位于 <html> 标签之前：

<?php session\_start(); ?>

<html>

<body>

</body>

</html>

上面的代码会向服务器注册用户的会话，以便您可以开始保存用户信息，同时会为用户会话分配一个 UID。

存储和取回 session 变量的正确方法是使用 PHP $\_SESSION 变量：

<?php

session\_start();

// store session data

$\_SESSION['views']=1;

?>

如果您希望删除某些 session 数据，可以使用 unset() 或 session\_destroy() 函数。

unset() 函数用于释放指定的 session 变量：

<?php

unset($\_SESSION['views']);

?>

您也可以通过 session\_destroy() 函数彻底终结 session：

<?php

session\_destroy();

?>

注释：session\_destroy() 将重置 session，将失去所有已存储的 session 数据。

session\_start() 会创建新会话或者重用现有会话。 如果通过 GET 或者 POST 方式，或者使用 cookie 提交了会话 ID， 则会重用现有会话。

当会话自动开始或者通过 session\_start() 手动开始的时候， PHP 内部会调用会话管理器的 open 和 read 回调函数。

PHP 会自动反序列化数据并且填充 $\_SESSION 超级全局变量。

要想使用命名会话，请在调用 session\_start() 函数 之前调用 session\_name() 函数。

如果启用了 session.use\_trans\_sid 选项， session\_start() 函数会注册一个内部输出管理器， 该输出管理器完成 URL 重写的工作。

bool session\_start ([ array $options = [] ] )

options:

此参数是一个关联数组，如果提供，那么会用其中的项目覆盖 会话配置指示 中的配置项。此数组中的键无需包含 session. 前缀。

返回值:

成功开始会话返回 TRUE ，反之返回 FALSE

//Example #1 page1.php

<?php

// page1.php

session\_start();

echo 'Welcome to page #1';

$\_SESSION['favcolor'] = 'green';

$\_SESSION['animal'] = 'cat';

$\_SESSION['time'] = time();

// 如果使用 cookie 方式传送会话 ID

echo '<br /><a href="page2.php">page 2</a>';

// 如果不是使用 cookie 方式传送会话 ID，则使用 URL 改写的方式传送会话 ID

echo '<br /><a href="page2.php?' . SID . '">page 2</a>';

?>

Session 函数 ：

session\_abort — Discard session array changes and finish session

session\_cache\_expire — 返回当前缓存的到期时间

session\_cache\_limiter — 读取/设置缓存限制器

session\_commit — session\_write\_close 的别名

session\_create\_id — Create new session id

session\_decode — 解码会话数据

session\_destroy — 销毁一个会话中的全部数据

session\_encode — 将当前会话数据编码为一个字符串

session\_gc — Perform session data garbage collection

session\_get\_cookie\_params — 获取会话 cookie 参数

session\_id — 获取/设置当前会话 ID

session\_is\_registered — 检查变量是否在会话中已经注册

session\_module\_name — 获取/设置会话模块名称

session\_name — 读取/设置会话名称

session\_regenerate\_id — 使用新生成的会话 ID 更新现有会话 ID

session\_register\_shutdown — 关闭会话

session\_register — Register one or more global variables with the current session

session\_reset — Re-initialize session array with original values

session\_save\_path — 读取/设置当前会话的保存路径

session\_set\_cookie\_params — 设置会话 cookie 参数

session\_set\_save\_handler — 设置用户自定义会话存储函数

session\_start — 启动新会话或者重用现有会话

session\_status — 返回当前会话状态

session\_unregister — Unregister a global variable from the current session

session\_unset — 释放所有的会话变量

session\_write\_close — Write session data and end session

//Example #2 page2.php

<?php

// page2.php

session\_start();

echo 'Welcome to page #2<br />';

echo $\_SESSION['favcolor']; // green

echo $\_SESSION['animal']; // cat

echo date('Y m d H:i:s', $\_SESSION['time']);

// 类似 page1.php 中的代码，你可能需要在这里处理使用 SID 的场景

echo '<br /><a href="page1.php">page 1</a>';

?>

Example #3 覆盖 Cookie 超时时间设定

<?php

// 设置 cookie 的有效时间为 1 天

session\_start([

'cookie\_lifetime' => 86400,

]);

?>

Example #4 读取会话之后立即关闭会话存储文件

<?php

// 如果确定不修改会话中的数据，

// 我们可以在会话文件读取完毕之后立即关闭它

// 来避免由于给会话文件加锁导致其他页面阻塞

session\_start([

'cookie\_lifetime' => 86400,

'read\_and\_close' => true,

]);

Note: 要使用基于 cookie 的会话， 必须在输出开始之前调用 session\_start() 函数。

我们来手动设置 Session 的生存期：

＜?php

session\_start();

// 保存一天

$lifeTime = 24 \* 3600;

setcookie(session\_name(), session\_id(), time() + $lifeTime, "/");

?＞

其实 Session 还提供了一个函数 session\_set\_cookie\_params(); 来设置 Session 的生存期的，该函数必须在 session\_start() 函数调用之前调用：

＜?php

// 保存一天

$lifeTime = 24 \* 3600;

session\_set\_cookie\_params($lifeTime);

session\_start();

$\_SESSION["admin"] = true;

?＞

如果客户端使用 IE 6.0 ， session\_set\_cookie\_params(); 函数设置 Cookie 会有些问题，所以我们还是手动调用 setcookie 函数来创建 cookie。

setcookie()

或

session\_set\_cookie\_params($lifeTime)

如何修改SESSION的生存时间：

答:方法1:将php.ini中的session.gc\_maxlifetime设置为9999重启apache

方法2:$savePath = “./session\_save\_dir/”;

$lifeTime = 小时 \* 秒;

session\_save\_path($savePath);

session\_set\_cookie\_params($lifeTime);

session\_start();

方法3:setcookie() and session\_set\_cookie\_params($lifeTime);

设置或读取session之前，需要做什么 ：

可以直接在php.ini中开启session.auto\_start=1或者在页面头部用session\_start();

开启session，session\_start()前面不能有任何输出，包括空行。

session\_start()是session机制的开始，它有一定概率开启垃圾回收,因为session是存放在文件中，PHP自身的垃圾回收是无效的，SESSION的回收是要删文件的，这个概率是根据php.ini的配置决定的。session会判断当前是否有$\_COOKIE[session\_name()];session\_name()返回保存session\_id的COOKIE键值，这个值可以从php.ini找到：session.name = PHPSESSID //默认值

1. session.use\_cookies:默认值为"1",代表SessionID使用Cookie来传递，反之就是用Query\_String来传递。
2. Session.name:这个就是SessionID储存的变量名称，可能是Cookie来传递，也可能是Query\_String来传递，默认值是"PHPSESSID"
3. Session.cookie\_lifetime:这个代表SessionID在客户端Cookie储存的时间，默认值是“0”，代表浏览器一关闭，SessionID就作废，就是因为这个原因，所以Session不能永久使用。
4. Session.gc\_maxlifetime:这个是Session数据在服务器端储存的时间，如果超过这个时间，那么Session数据就自动删除。

**session\_name()**

session\_name() 存取目前 session 名称。

语法: string session\_name(string [name]);

本函数可取得或者重新配置目前 Session 的名称。若无参数 name 则表示单单获取目前 Session 名称，加上参数则表示将 Session 名称设为参数 name。

如果建立时是这样seesion\_register("test");那么就可以用seesion\_name("test")来找到这。session\_id()这个函数是用来为会话进行编号就好像PHP数组一样支持字符和数字标识。

#### session\_name() 与 session\_id() 的区别

* session\_name() 默认为 "PHPSESSID"
* 而 session\_id() 是 一次HTTP 请求,服务器得到的 $\_POST['PHPSESSID'] 或者 $\_GET['PHPSESSID'] 或者 $\_COOKIE['PHPSESSID']；

如果你在 session\_start() 前调用了 session\_name('SID'); 那么正常情况下（客户端支持Cookie时）, 会给客户端发送 Set-Cookie: SID=(session\_id 的值);

使用session的步骤：

1. 启动session：

使用session\_start()函数来启动。

2. 注册会话：

直接给$\_SESSION数组添加元素即可。

3. 使用会话：

判断session是否为空或者是否已经注册，如果已经存在则像普通数组使用即可。

4. 删除会话：

1.可以使用unset删除单个session；

2.使用$\_SESSION=array()的方式，一次注销所有的会话变量；

3.使用session\_destroy()函数来彻底销毁session。