会话技术

会话技术初步认识

会话技术介绍

web 会话可简单理解为：用户开一个浏览器，访问某一个 web 站点，在这个站点点击多个超链接，访问服务器多个 web 资源，然后关闭浏览器，整个过程称之为一个会话。

HTTP 协议的特点是无状态/无连接，当一个浏览器连续多次请求同一个 web 服务器时，服务器是无法区分多个操作是否来自于同一个浏览器（用户）。会话技术就是通过 HTTP 协议想办法让服务器能够识别来自同一个浏览器的多次请求，从而方便浏览器（用户）在访问同一个网站的多次操作中，能够持续进行而不需要进行额外的身份验证。

会话技术分类

1） cookie 技术
   Cookie 是在 HTTP 协议下，服务器或脚本可以维护客户工作站上信息的一种方式。Cookie 是由 Web 服务器保存在用户浏览器（客户端）上的小文本文件(HTTP 协议响应头)，它可以包含有关用户的信息。无论何时用户链接到服务器（HTTP 请求携带数据），Web 站点都可以访问 Cookie 信息
2） session 技术
   Session 直接翻译成中文比较困难，一般都译成时域。在计算机专业术语中，Session 是指一个终端用户与交互系统进行通信的时间间隔，通常指从注册进入系统到注销退出系统之间所经过的时间。以及如果需要的话，可能还有一定的操作空间。Session 技术是将数据保存到服务器端，无论何时用户链接到服务器，Web 站点都可以访问 Session 信息：SESSION 技术的实现是依赖 COOKIE 技术的。

两种会话技术区别

1） 安全性方面
   a） Session 存储服务器端，安全性高
   b） Cookie 存储浏览器端，安全性低
2） 数据大小方面
   a） Cookie 的数量和大小都有限制（20 个/4K）
   b） Session 数据存储不限
3） 可用数据类型
   a） Cookie 只能存储简单数据，数值/字符串
   b） Session 可以存储复杂数据（自动序列化）

4） 保存位置方面
   a） Cookie 保存在浏览器上
   b） Session 保存在服务器上

COOKIE 的基本使用

COOKIE 原理

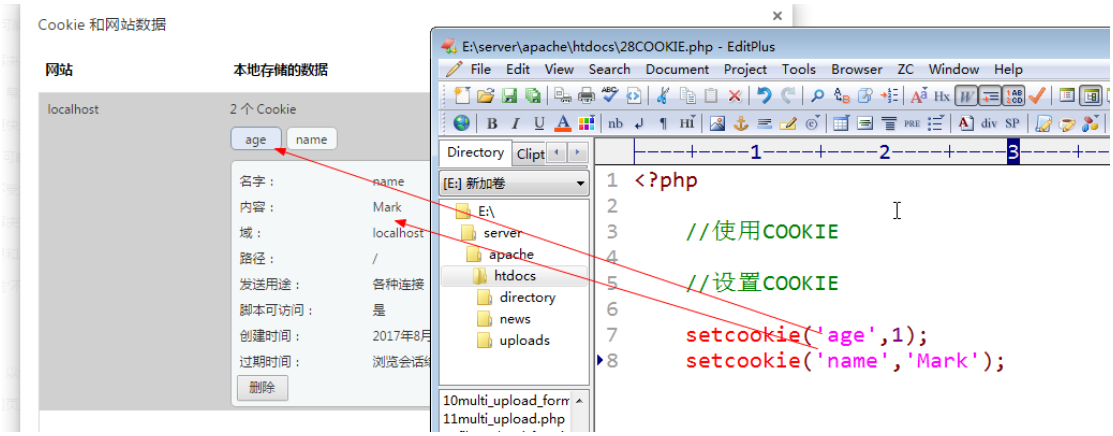
COOKIE 技术：服务器将数据通过 HTTP 响应存储到浏览器上，浏览器可以在以后携带对应的 COOKIE 数据访问服务器。

1、 第一次请求时，PHP 通过 setcookie 函数将数据通过 http 协议响应头传输给浏览器
2、 浏览器在第一次响应的时候将 Cookie 数据保存到浏览器
3、 浏览器后续请求同一个网站的时候，会自动检测是否存在 Cookie 数据，如果存在将在请求头中将数据携带到服务器
4、 PHP 执行的时候会自动判断浏览器请求中是否携带 Cookie，如果写到，自动保存到$_COOKIE 中
5、 利用$_COOKIE 访问 Cookie 数据

设置 COOKIE 信息

Setcookie 函数用来设定 COOKIE 信息
Setcookie(名字，值)

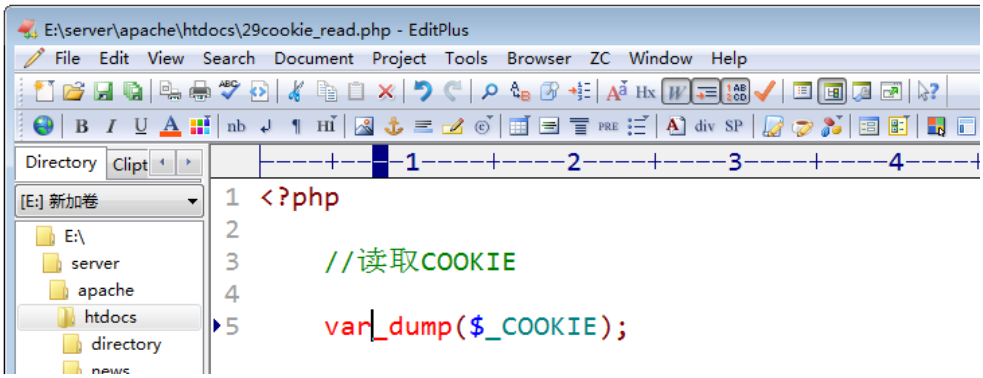
1）cookie 名的设置：字符串，第一个参数
2）cookie 值的设置：第二个参数
3）cookie 值的类型要求：必须是简单类型中的整数或者字符串



读取 COOKIE 信息

1）$_COOKIE 数组的使用

```
array(2) { ["age"]=> string(1) "1" ["name"]=> string(4) "Mark" }
```



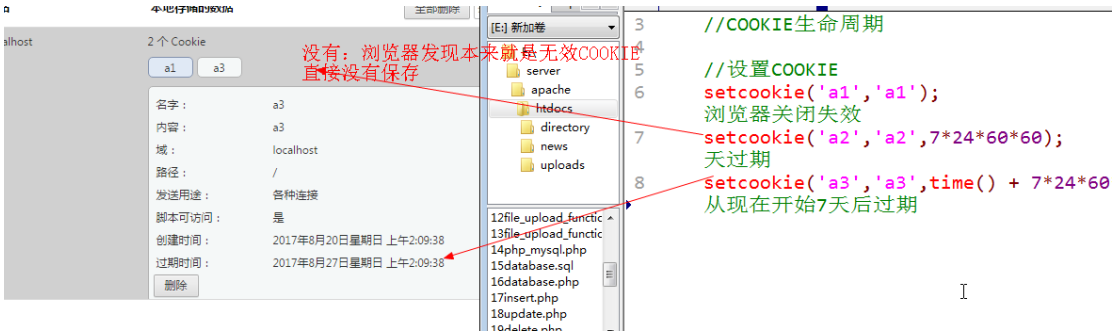
COOKIE（会话技术）能够实现跨脚本共享数据

COOKIE 高级使用

COOKIE 生命周期

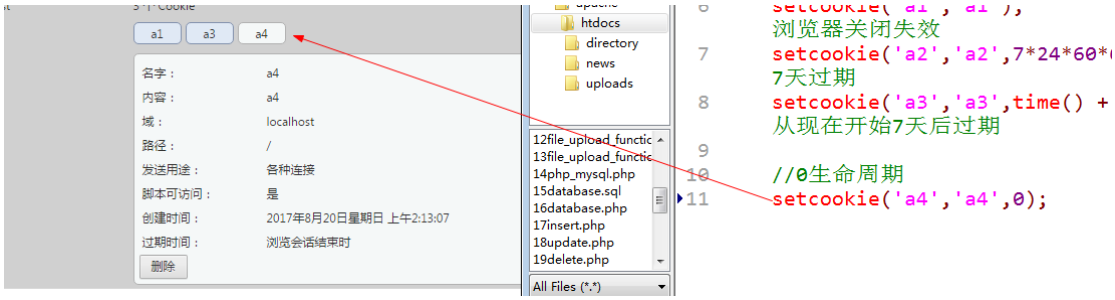
COOKIE 生命周期：COOKIE 在浏览器生存时间（浏览器在下次访问服务器的时候是否携带对应的 COOKIE）

1）默认（不设定）时的生命周期：不设定周期默认是关闭浏览器（会话结束）
2）设定为一个常规日期戳的周期：通过 setcookie 第三个参数可以限定生命周期，是用时间戳来管理，从格林威治时间开始



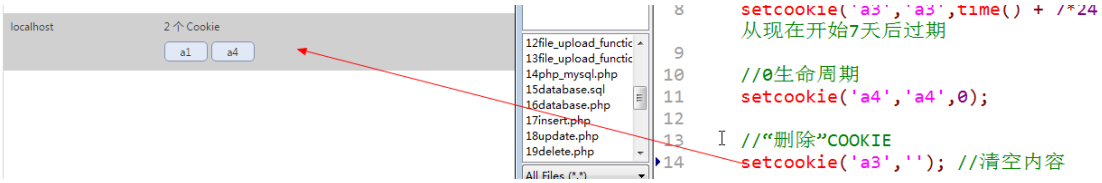
浏览器关闭后重新查看

3）设定为“0”的周期：在第三个参数设定生命周期的时候，用 0 代替时间戳：表示就是普通设置，会话结束过期

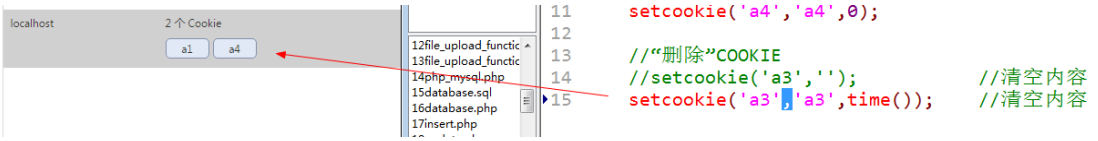


4）删除一个 cookie 的做法：服务器没有权限去操作浏览器上的内容（不可能删除）。可以通过设定生命周期来让浏览器自动判定 COOKIE 是否有效：无效就清除

4.1 清空 COOKIE 数据内容



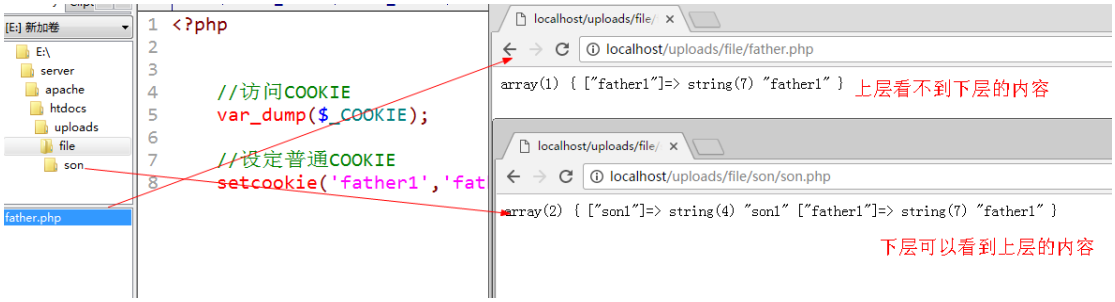
4.2 设定时间戳过期



COOKIE 作用范围

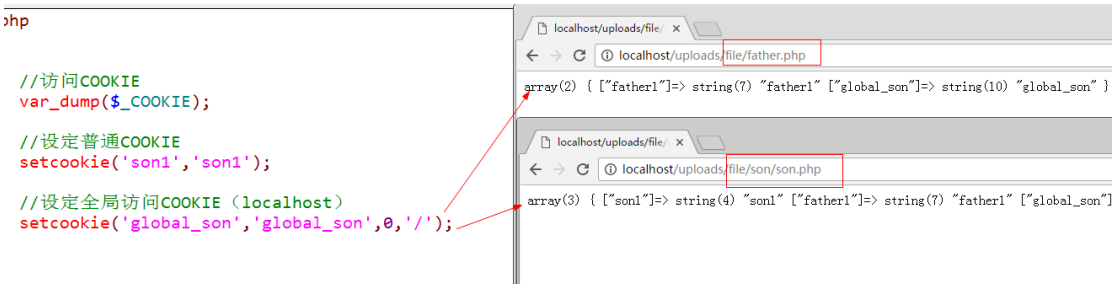
作用范围：不同的文件夹层级中，设定的 COOKIE 默认是在不同的文件夹下有访问限制。上层文件夹中设定的 COOKIE 可以在下层（子文件夹）中访问，而子文件夹中设定的 COOKIE 不能在上层文件夹中访问。

1）默认（不设定）的范围：就是使用 COOKIE 默认的作用范围



2）设定为“/”的含义：告知浏览器当前 COOKIE 的作用范围是网站根目录

Setcookie(名字,值,生命周期,作用范围)

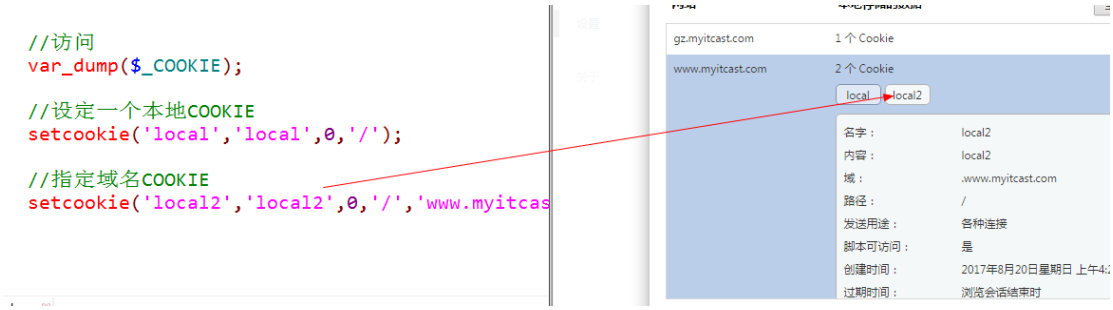


COOKIE 跨子域

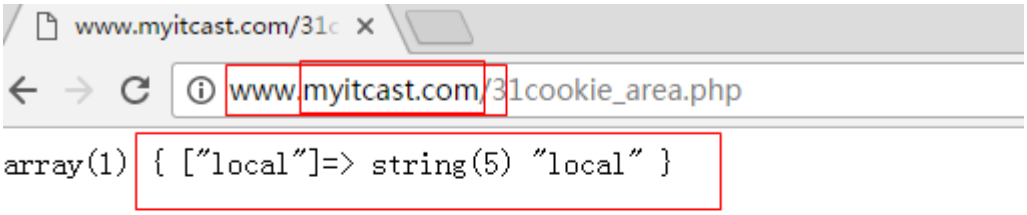
跨子域：在同一级别域名下，myitcast.com（一级域名），可以有多个子域名（www.myitcast.com 和 gz.myitcast.com），他们之间是搭建在不同的服务器上（不同文件夹：E:/server/apache/htdocs 和 E:/web），但是可以通过 COOKIE 设置实现对应的 COOKIE 共享

访问。但是默认是不允许跨域名访问的。

1）设定 cookie 的有效域名：不同的域名（包含主机）之间不能共享 COOKIE
可以通过 setcookie 的第五个参数来控制
Setcookie(名字,值,生命周期,作用范围,有效域名)



2）不设定时的默认有效域名



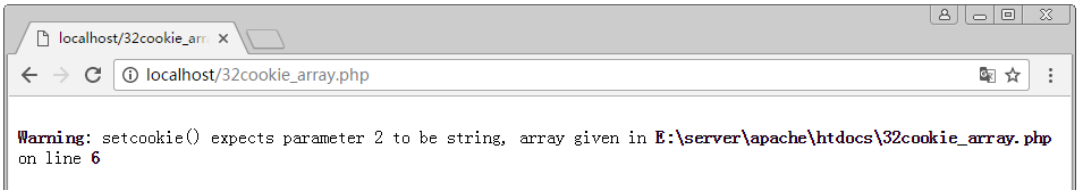
3）跨子域的设定方法：在设定域名访问的时候用设定上级域名即可：myitcast.com，这个是有所有以 myitcast.com 结尾的网站都可以共享 COOKIE

COOKIE 数组数据

COOKIE 本身只支持简单数据（数字或者字符串），能够保留的数据本身有限，也不成体系。

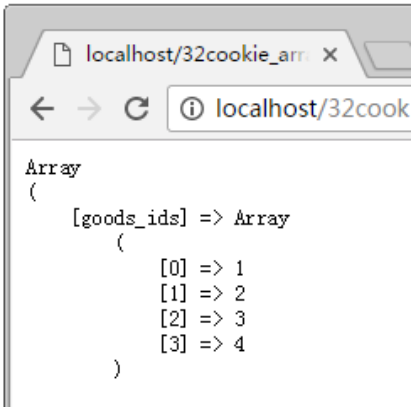
如果需要使用 COOKIE 来保留一组数据的化，想办法凑成数组。（COOKIE 不支持数组）

```
//使用"数组"保存COOKIE

//尝试保存数组
setcookie('goods_ids',array(1,2,3,4,5));
```



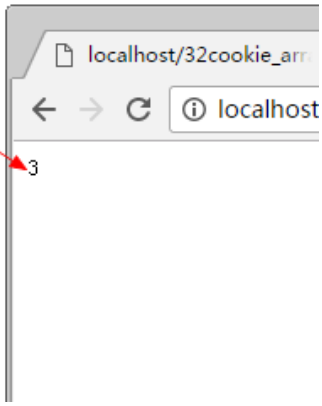
1）设置形式：setcookie('c1[k1]', 值)

```
//伪装数组
setcookie('goods_ids[0]',1);
setcookie('goods_ids[1]',2);
setcookie('goods_ids[2]',3);
setcookie('goods_ids[3]',4);

echo '<pre>';
print_r($_COOKIE);
```



2）读取形式：$_COOKIE['c1']['k1']

```
//伪装数组
setcookie('goods_ids[0]',1);
setcookie('goods_ids[1]',2);
setcookie('goods_ids[2]',3);
setcookie('goods_ids[3]',4);

echo '<pre>';
//print_r($_COOKIE);

echo $_COOKIE['goods_ids'][2];
```



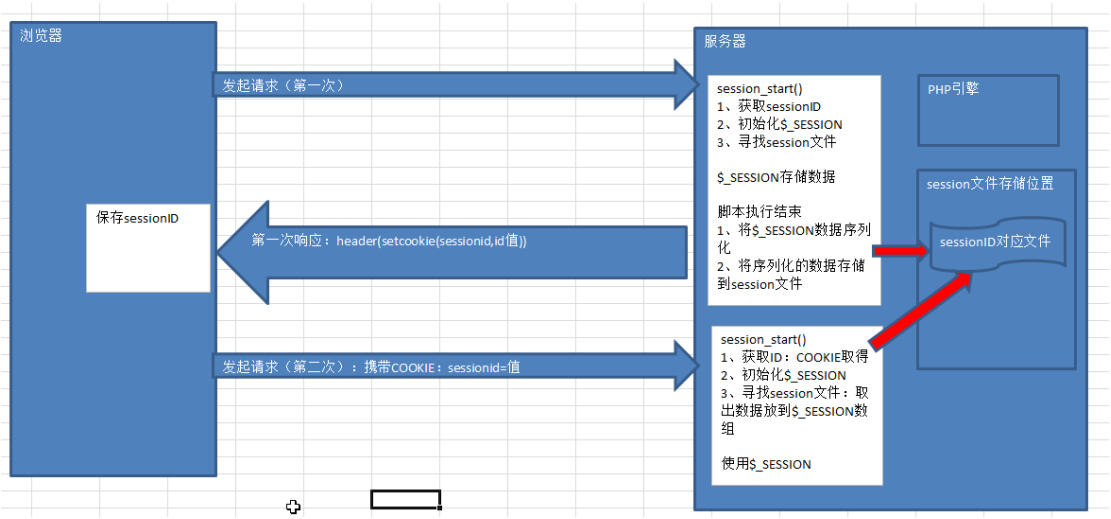
会话技术

SESSION 基本使用

SESSION 原理

Session 与浏览器无关，但是与 Cookie 有关。

1、 PHP 碰到 session_start()时开启 session 会话，会自动检测 sessionID
   a) 如果 Cookie 中存在，使用现成的
   b) 如果 Cookie 中不存在，创建一个 sessionID，并通过响应头以 Cookie 形式保存到浏览器上
2、 初始化超全局变量$_SESSION 为一个空数组
3、 PHP 通过 sessionID 去指定位置（session 文件存储位置）匹配对应的文件
   a) 不存在该文件：创建一个 sessionID 命名文件
   b) 存在该文件：读取文件内容（反序列化），将数据存储到$_SESSION 中
4、 脚本执行结束，将$_SESSION 中保存的所有数据序列化存储到 sessionID 对应的文件中

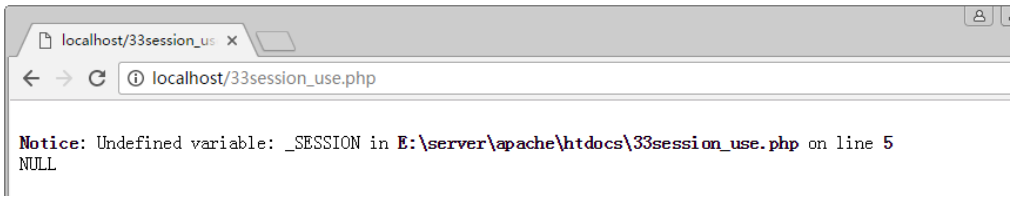


SESSION 基本使用

启用 session，任何时候都需要开启 session（脚本使用到$_SESSION 就开启一次）

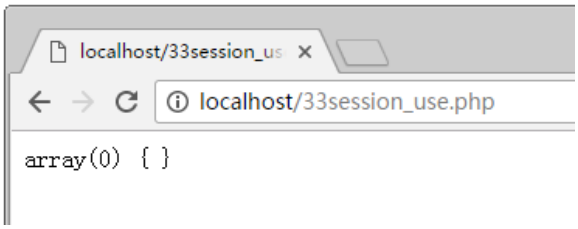
$_SESSION 是通过 session_start()函数的调用才会定义的，没有直接定义

```php
//session基本使用

var_dump($_SESSION);
```



Session 使用需要开启 session_start

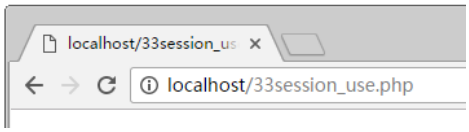
```php
//开启session
session_start();

var_dump($_SESSION);
```



设置 SESSION 信息

如果想存储数据到 session 中，那么只要不断给$_SESSION 数组添加元素即可

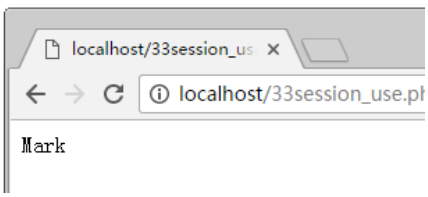
```php
//设置session数据
$_SESSION['name'] = 'Mark';
$_SESSION['hobby'] = array('basketball','football');
```



读取 SESSION 信息

$_SESSION 就是一个数组，存储什么数据，什么方式存的，就是可以通过什么方式访问什么数据

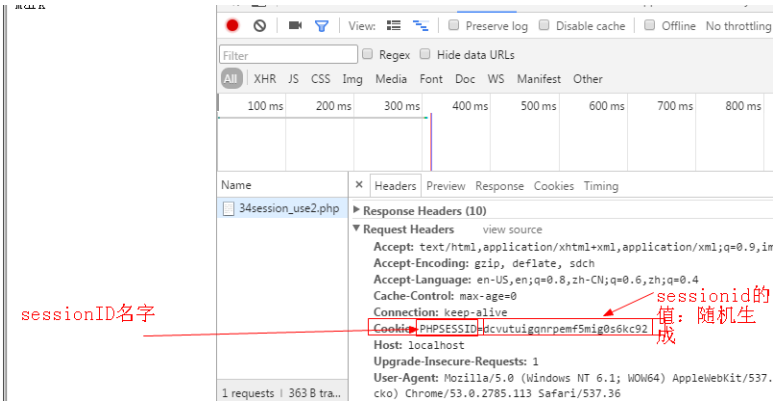
```php
12
13      //设置session数据
14      $_SESSION['name'] = 'Mark';
15      $_SESSION['hobby'] = array('basketball','football');
16
17      //访问session数据
18      echo $_SESSION['name'];
```



会话技术的本质是为了实现跨脚本共享数据：在一个脚本中定义数据，在另外一个脚本中保

存数据

```
5
6       //开启session
7       session_start();
8
9       //访问
0       echo $_SESSION['name'];
```



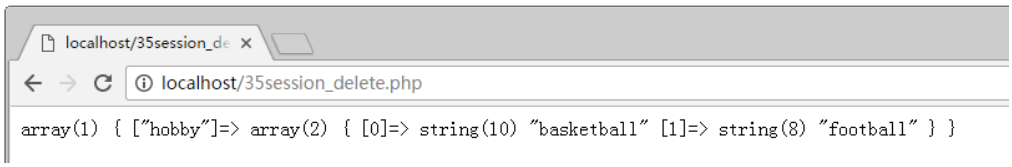
SESSION 基本使用

删除 session 就是将 session 数据清理掉（$_SESSION 拿不到）

删除一个 SESSION 信息

Unset($_SESSION[元素下标])删除指定 session 数据

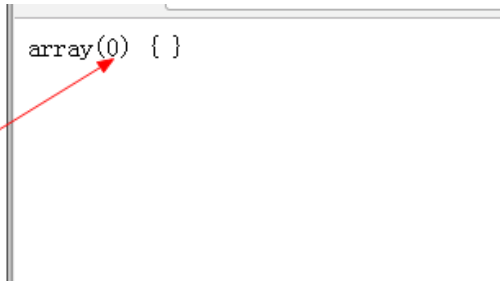
```
//读取session
var_dump($_SESSION);

//删除数据
unset($_SESSION['name']);
```



删除全部 SESSION 信息

删除全部数据就是让$_SESSION 变成一个空数组

```
//删除数据
unset($_SESSION['name']);

//删除全部数据
$_SESSION = array();
```



SESSION 基本使用

SESSION 相关配置

SESSION 基础配置

1）session.name：session 名字，保存到 COOKIE 中 sessionID 对应的名字

```
1415
1416 ; Name of the session (used as cookie name).
1417 ; http://php.net/session.name
1418 session.name = PHPSESSID
1419
```

2）session.auto_start：是否自动开启 session（无需手动 session_start()），默认是关闭的

```
1419
1420 ; Initialize session on request startup.
1421 ; http://php.net/session.auto-start
1422 session.auto_start = 0
1423
```

3）session.save_handler：session 数据的保存方式，默认是文件形式

```
1367
1368 [Session]
1369 ; Handler used to store/retrieve data.
1370 ; http://php.net/session.save-handler
1371 session.save_handler = files
1372
```

4）session.save_path：session 文件默认存储的位置

```
398 ; does not overwrite the process's umask.
399 ; http://php.net/session.save-path
400 ;session.save_path = "/tmp"
401
```

没有开启：说明借用操作系统的临时文件存储

使用系统的文件夹存储不安全，需要指定对应存储路径

```
1398 ; does not overwrite the process's umask.
1399 ; http://php.net/session.save-path
1400 ;session.save_path = "/tmp"
1401 session.save_path = "E:/server/sessions"
1402
```

SESSION 常用配置

1）session.cookie_lifetime：PHPsessionID 在浏览器端对应 COOKIE 的生命周期，默认是会话结束

```
1424
1425 ; Lifetime in seconds of cookie or, if 0, until browser is restarted.
1426 ; http://php.net/session.cookie-lifetime
1427 session.cookie_lifetime = 0
```

2）session.cookie_path：sessionID 在浏览器存储之后允许服务器访问的路径（COOKIE 有作用范围）

```
1429 ; The path for which the cookie is valid.
1430 ; http://php.net/session.cookie-path
1431 session.cookie_path = /
1432
```

网站根目录

3）session.cookie_domain：COOKIE 允许访问的子域（COOKIE 可以跨子域）

```
1432
1433 ; The domain for which the cookie is valid.
1434 ; http://php.net/session.cookie-domain
1435 session.cookie_domain =
1436
```

默认的PHPsessionID只能当前主机访问

配置的两种形式

1）php.ini 中配置：全局配置，修改 php.ini 中的配置项

```
398 ; does not overwrite the process's umask.
399 ; http://php.net/session.save-path
400 ;session.save_path = "/tmp"
401 session.save_path = "E:/server/sessions"
402
```

2）脚本中配置：PHP 可以通过 ini_set 函数来在运行中设定某些配置项（只会对当前运行的脚本有效），把这种配置称之为项目级

@Ini_set(‘session.save_path’,’E:/server/sessions’);

SESSION 基本使用

Session 删除是指删除 session 数据，$_SESSION 中看不到而已；销毁 session 是指删除 session 对应的 session 文件。

销毁 SESSION

系统提供一个函数：session_destroy()，会自动根据 session_start 得到的 sessionID 去找到指

定的 session 文件，并把其删除。

```
//销毁session
session_destroy();
```

SESSION 基本使用

SESSION 垃圾回收机制

垃圾回收机制原理

session 会话技术后，session 文件并不会自动清除，如果每天有大量 session 文件产生但是又都是失效的，会增加服务器的压力和影响 session 效率。
垃圾回收，是指 session 机制提供了一种解决垃圾 session 文件的方式：给 session 文件指定周期，通过 session 文件最后更改时间与生命周期进行结合判定，如果已经过期则删除对应的 session 文件，如果没有过期则保留。这样就可以及时清理无效的僵尸文件，从而提升空间利用率和 session 工作效率。

1、 任何一次 session 开启（session_start），session 都会尝试去读取 session 文件
2、 读取 session 文件后，有可能触发垃圾回收机制（在 session 系统中也是一个函数：自己有一定几率调用）
3、 垃圾回收机制会自动读取所有 session 文件的最后编辑时间，然后加上生命周期（配置文件）与当前时间进行比较（所有 session 文件）
    a) 过期：删除
    b) 有效：保留

垃圾回收参数设置

1）session.gc_maxlifetime = 1440：规定的 session 文件最大的生命周期是 1440 秒，24 分钟
2）session.gc_probability = 1：垃圾回收概率因子（分子）
3）session.gc_divisor = 1000：垃圾回收概率分母

默认的触发概率是 1/1000

```
; Production Value: 1
; http://php.net/session.gc-probability
session.gc_probability = 1

; Defines the probability that the 'garbage collection' process is starte
; session initialization. The probability is calculated by using the foll
; gc_probability/gc_divisor. Where session.gc_probability is the numeratc
; session.gc_divisor is the denominator in the equation. Setting this val
; when the session.gc_divisor value is 100 will give you approximately a
; the gc will run on any give request. Increasing this value to 1000 will
; a 0.1% chance the gc will run on any give request. For high volume proc
; this is a more efficient approach.
; Default Value: 100
; Development Value: 1000
; Production Value: 1000
; http://php.net/session.gc-divisor
session.gc_divison = 1000

; After this number of seconds, stored data will be seen as 'garbage' anc
; cleaned up by the garbage collection process.
; http://php.net/session.gc-maxlifetime
session.gc_maxlifetime = 1440
```

测试垃圾回收效果：

1、 修改生命周期为 2 分钟，120 秒

2、 修改触发几率：100%

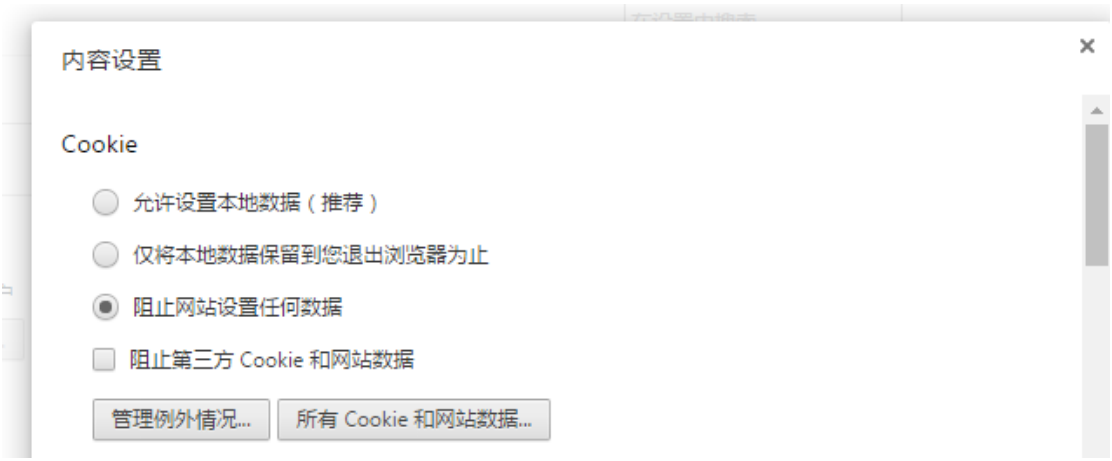
```
; Development Value: 1000
; Production Value: 1000
; http://php.net/session.gc-divisor
session.gc_divisor = 1

; After this number of seconds, stored data will be seen as 'ga
; cleaned up by the garbage collection process.
; http://php.net/session.gc-maxlifetime
session.gc_maxlifetime = 120
```

SESSION 基本使用

禁用 COOKIE 后如何使用 SESSION

禁用 COOKIE 不能使用 SESSION 原因

Session 技术需要利用到 COOKIE 技术来保存 sessionID，从而使得 PHP 能够在跨脚本的时候得到相同的 sessionID，从而访问同一个 session 文件。



解决思路：最终让 session_start 在开启之前拿到原来的 sessionID（另外一个脚本的）

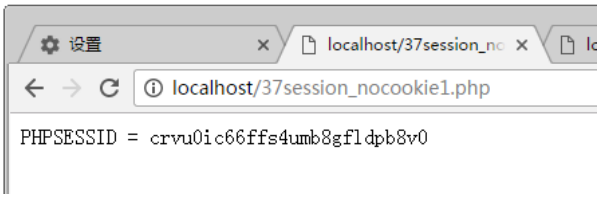
实现无 COOKIE 使用 SESSION

在 PHP 中，想要解决没有 COOKIE 也实现 session 技术的方式有两种：

方案 1：可以利用 PHP 提供的 session 函数：session_id 和 session_name 来获得和设置 sessionID 或者 name 从而解决 session_start 产生新 sessionID 的情况（手动操作）：

1、 在 session 保存数据的脚本中获取 sessionID 和名字

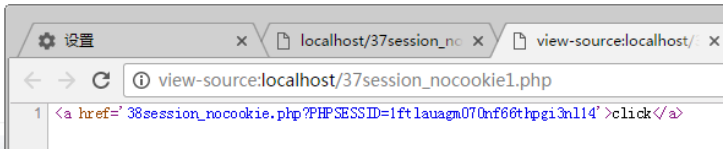
```
//获取sessionID和名字
$id = session_id();          //获取是在session_start之后
$name = session_name();      //拿到名字（php.ini session.name）
echo $name . ' = ' . $id;
```




```
//设置内容
$_SESSION['name'] = 'mark';
```

2、 想办法将数据传递给另外一个脚本：表单传值（URL 或者 form 表单）

```
//设置内容
$_SESSION['name'] = 'mark';

//传递给另外一个脚本
echo "<a href='38session_nocookie.php?{$name}={$id}'>click</a>";
```



3、 在需要使用到 session 的脚本中，先接收数据

```php
//访问session
//接收数据
$name = session_name();
$id = $_GET[$name];
```

4、 组织 session_start 产生新的 ID，告诉它已经存在：session_id($id)

```php
//接收数据
$name = session_name();
$id = $_GET[$name];

//设定sessionID
session_id($id);


//开启session
```

方案 2：可以利用 session 集中已经提供的解决方案自动操作（配置）
原因 1：默认 session 配置只允许使用 COOKIE 保存 sessionID：cookie_ony
原因 2：默认关闭了其他能够传送数据的方式，只保留了 COOKIE

1、 修改 PHP 配置文件，开启其他方式传输 sessionID，关闭只允许使用 COOKIE 传输功能

```
; not the end all be all of session hijacking defense, but it's a g
; http://php.net/session.use-only-cookies
session.use_only_cookies = 1
```

需要关闭：从1变成0

```
;   always using URL stored in browser's history or bookmarks.
; http://php.net/session.use-trans-sid
session.use_trans_sid = 0

; Select a hash function for use in generating session ids.
```

开启其他方式保存sid，从0变成1

2、 一旦配置开启，PHP 会自动将 sessionID 和 session 名字在其他位置绑定数据，同时还会在 session_start 的时候，考虑其他方式传递（表单）的数据，而不是只有 COOKIE

```php
//保存session数据
session_start();
$_SESSION['age'] = 40;

echo "<a href='40session_nocookie4.php'>click</a>";
```