# 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

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

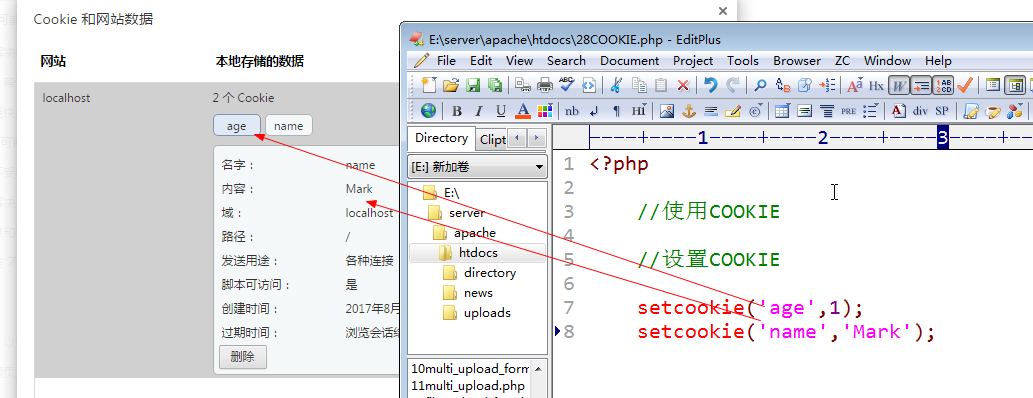
Setcookie函数用来设定COOKIE信息

Setcookie(名字，值)

1）cookie名的设置：字符串，第一个参数

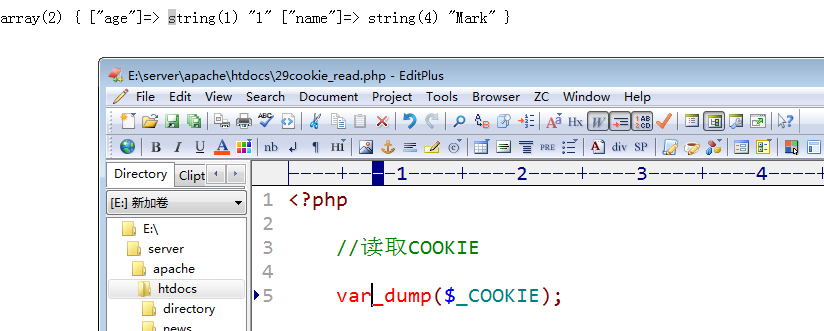
2）cookie值的设置：第二个参数

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



COOKIE（会话技术）能够实现跨脚本共享数据

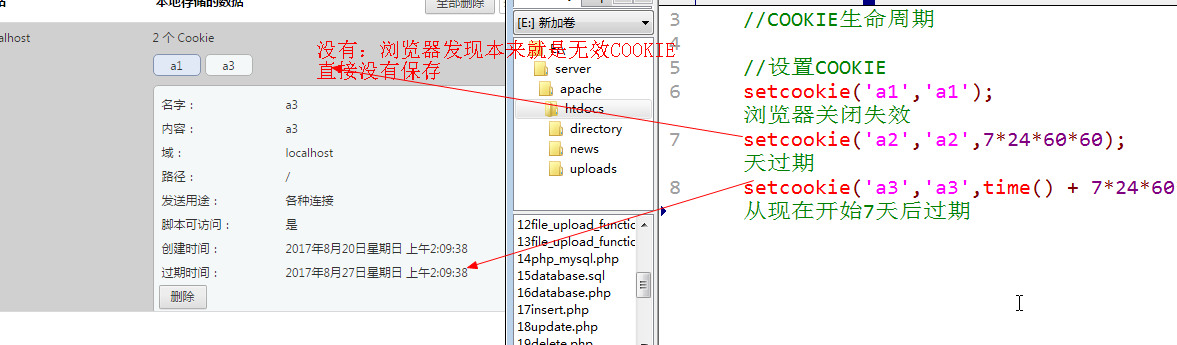
## COOKIE高级使用

### 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）

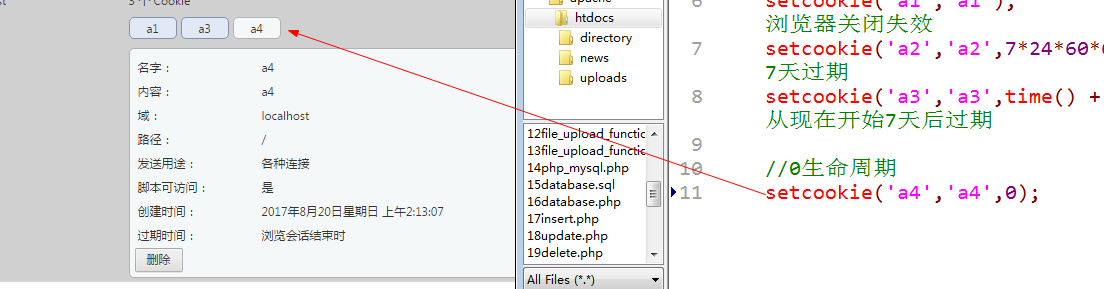
2）设定为一个常规日期戳的周期：通过setcookie第三个参数可以限定生命周期，是用时间戳来管理，从格林威治时间开始



浏览器关闭后重新查看

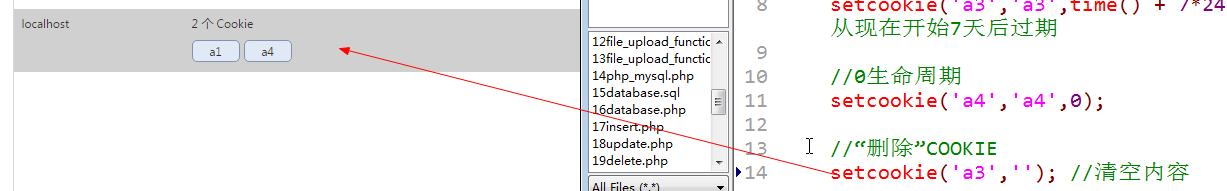


3）设定为“0”的周期：在第三个参数设定生命周期的时候，用0代替时间戳：表示就是普通设置，会话结束过期

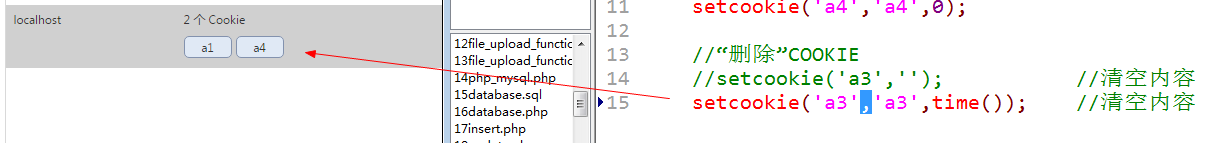


4）删除一个cookie的做法：服务器没有权限去操作浏览器上的内容（不可能删除）。可以通过设定生命周期来让浏览器自动判定COOKIE是否有效：无效就清除

4.1 清空COOKIE数据内容



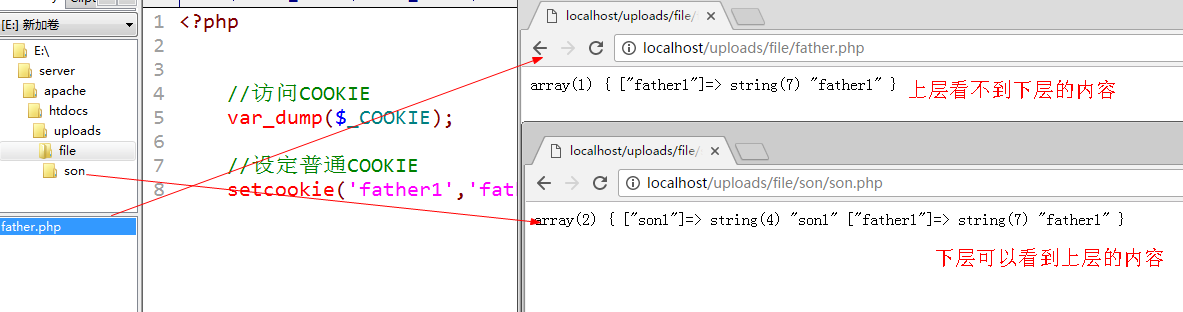
4.2 设定时间戳过期



### COOKIE作用范围

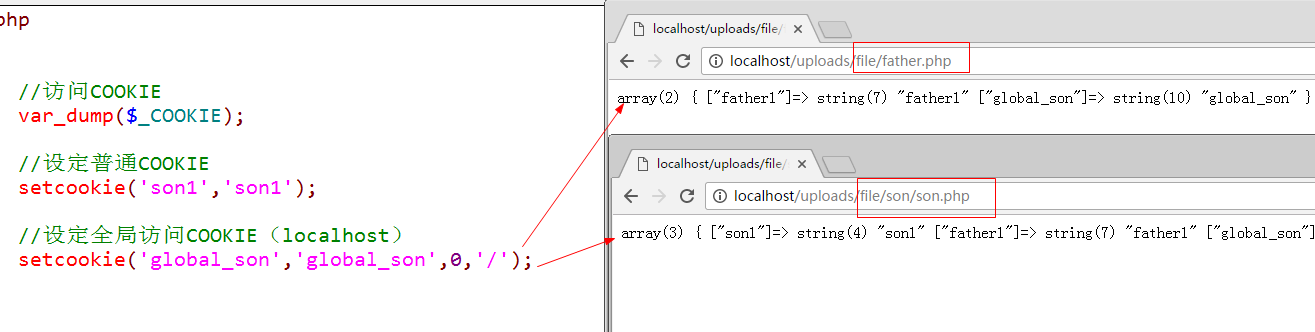
作用范围：不同的文件夹层级中，设定的COOKIE默认是在不同的文件夹下有访问限制。上层文件夹中设定的COOKIE可以在下层（子文件夹）中访问，而子文件夹中设定的COOKIE不能在上层文件夹中访问。

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)



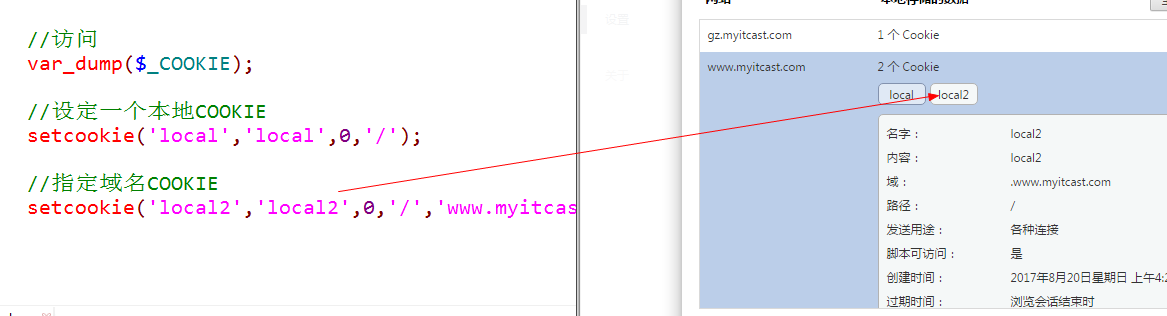
### COOKIE跨子域

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](http://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

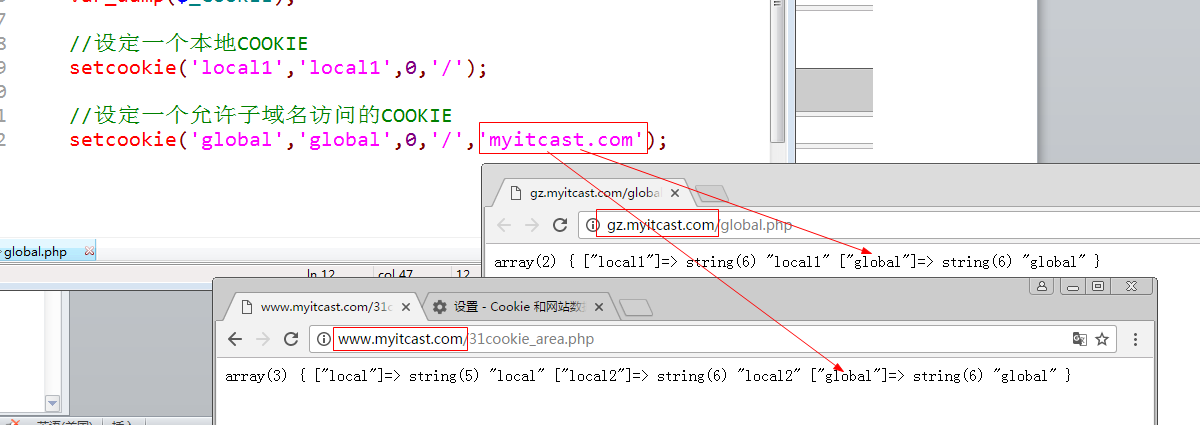
Setcookie(名字,值,生命周期,作用范围,有效域名)



2）不设定时的默认有效域名

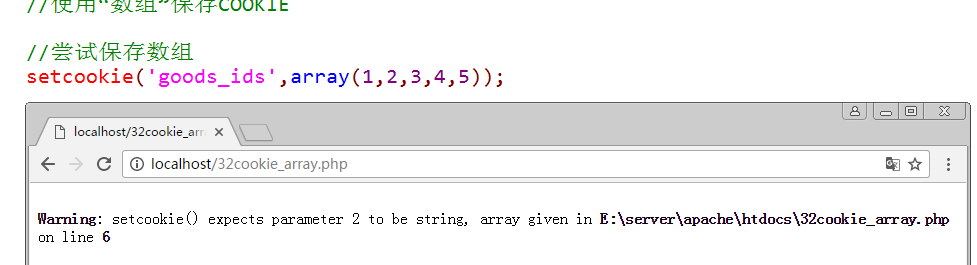


3）跨子域的设定方法：在设定域名访问的时候用设定上级域名即可：myitcast.com，这个是有所有以myitcast.com结尾的网站都可以共享COOKIE

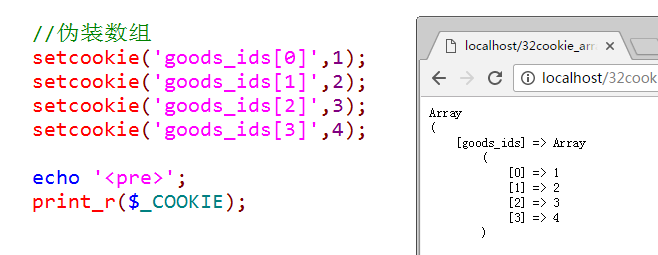


### COOKIE数组数据

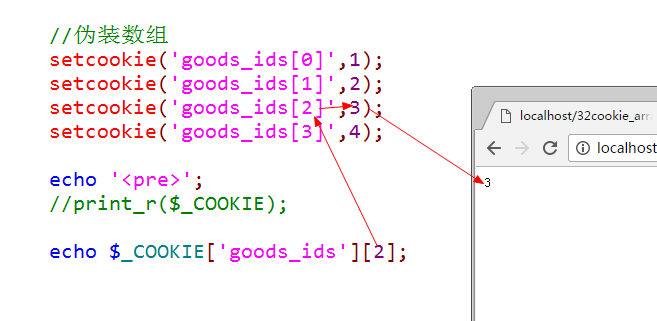
COOKIE本身只支持简单数据（数字或者字符串），能够保留的数据本身有限，也不成体系。如果需要使用COOKIE来保留一组数据的化，想办法凑成数组。（COOKIE不支持数组）



1）设置形式：setcookie('c1[k1]', 值)



2）读取形式：$\_COOKIE['c1']['k1']



# 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。

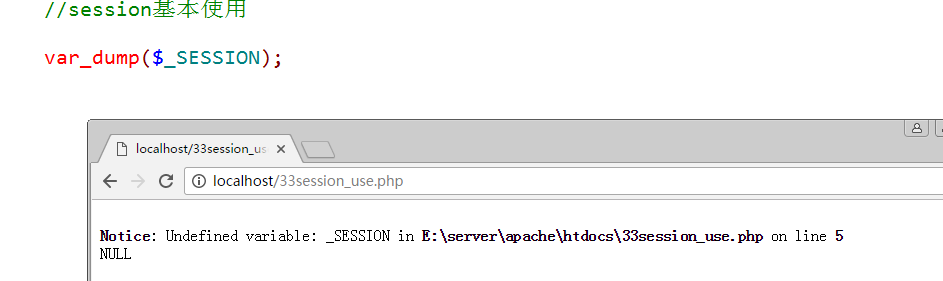
1. PHP碰到session\_start()时开启session会话，会自动检测sessionID
   1. 如果Cookie中存在，使用现成的
   2. 如果Cookie中不存在，创建一个sessionID，并通过响应头以Cookie形式保存到浏览器上
2. 初始化超全局变量$\_SESSION为一个空数组
3. PHP通过sessionID去指定位置（session文件存储位置）匹配对应的文件
   1. 不存在该文件：创建一个sessionID命名文件
   2. 存在该文件：读取文件内容（反序列化），将数据存储到$\_SESSION中
4. 脚本执行结束，将$\_SESSION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



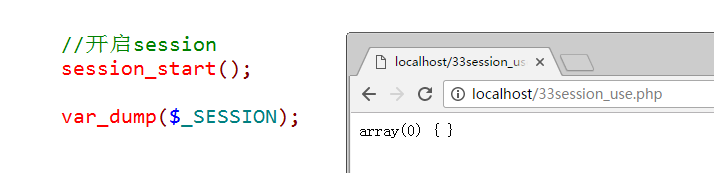
## SESSION基本使用

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

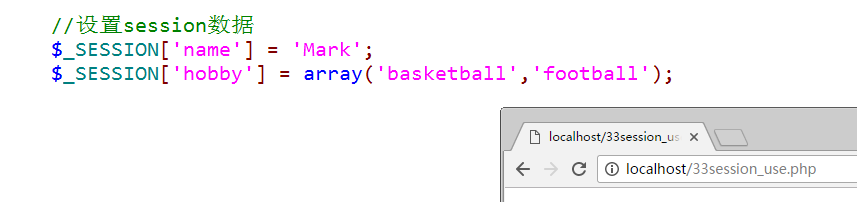


Session使用需要开启session\_start



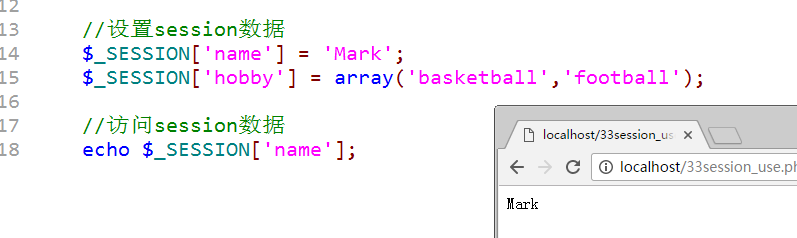
### 设置SESSION信息

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

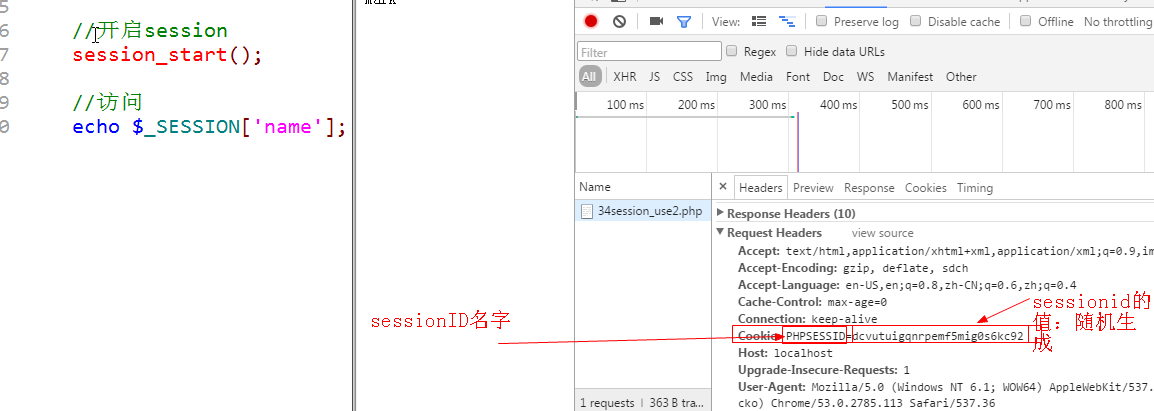


### 读取SESSION信息

$\_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



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

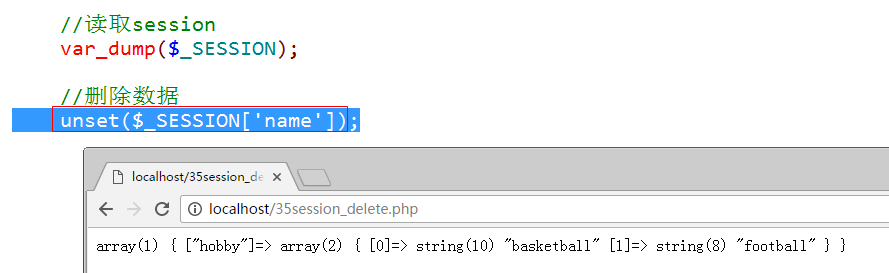


## SESSION基本使用

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

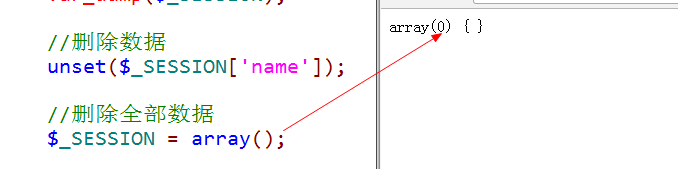
### 删除一个SESSION信息

Unset($\_SESSION[元素下标])删除指定session数据



### 删除全部SESSION信息

删除全部数据就是让$\_SESSION变成一个空数组

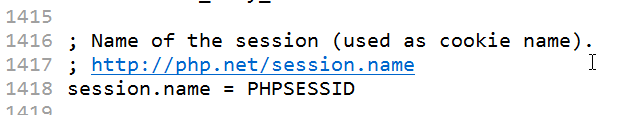


## SESSION基本使用

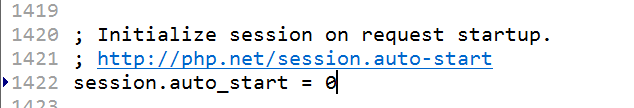
### SESSION相关配置

#### SESSION基础配置

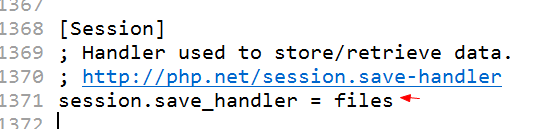
1）session.name：session名字，保存到COOKIE中sessionID对应的名字



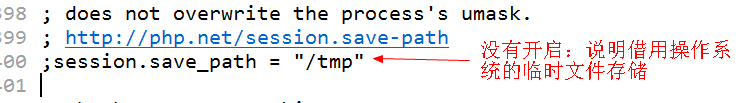
2）session.auto\_start：是否自动开启session（无需手动session\_start()），默认是关闭的



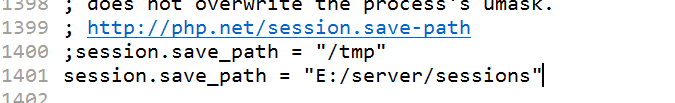
3）session.save\_handler：session数据的保存方式，默认是文件形式



4）session.save\_path：session文件默认存储的位置

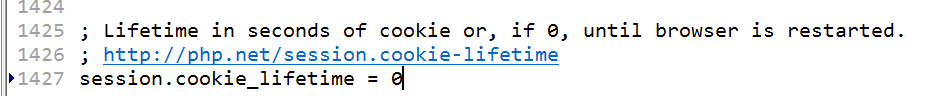


使用系统的文件夹存储不安全，需要指定对应存储路径

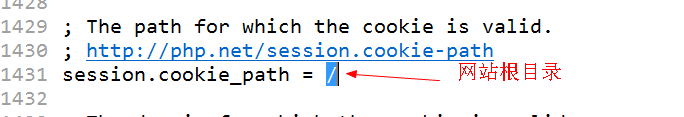


#### SESSION常用配置

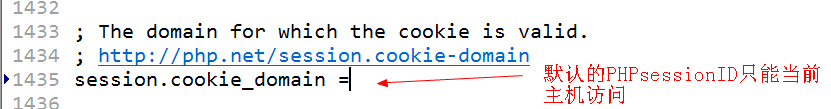
1）session.cookie\_lifetime：PHPsessionID在浏览器端对应COOKIE的生命周期，默认是会话结束



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）

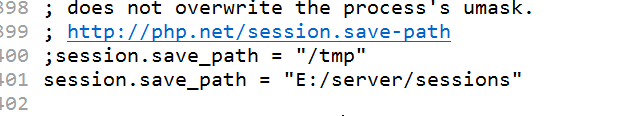


3）session.cookie\_domain：COOKIE允许访问的子域（COOKIE可以跨子域）



#### 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



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

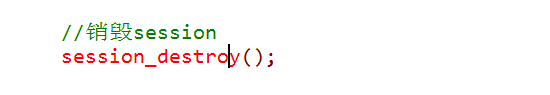
@Ini\_set(‘session.save\_path’,’E:/server/sessions’);

## 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。



## 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）
   1. 过期：删除
   2. 有效：保留

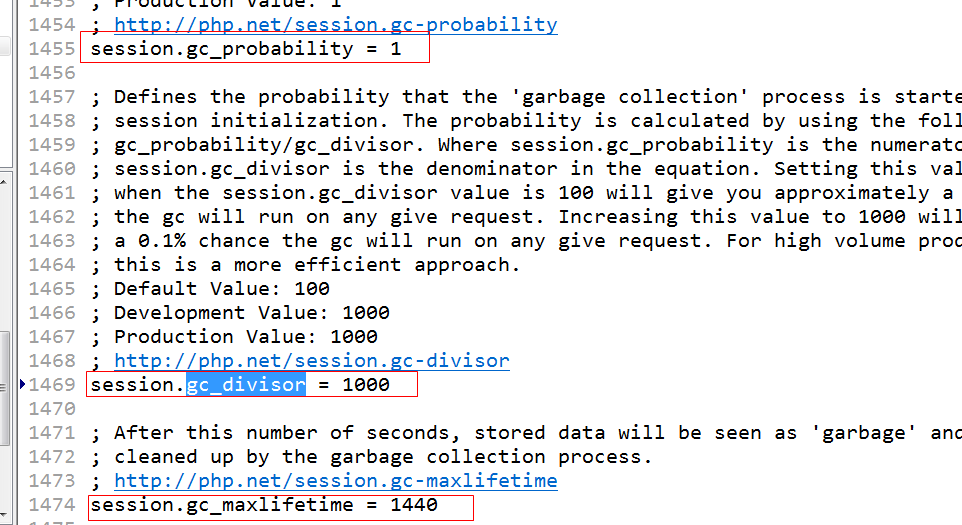
#### 垃圾回收参数设置

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

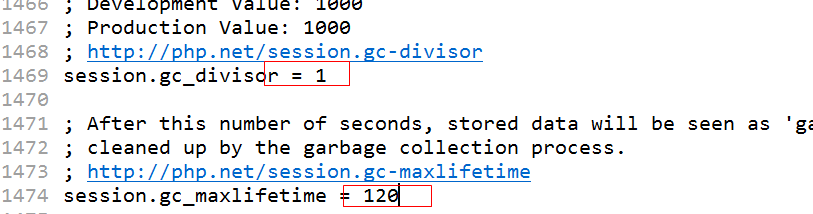
默认的触发概率是1/1000



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%

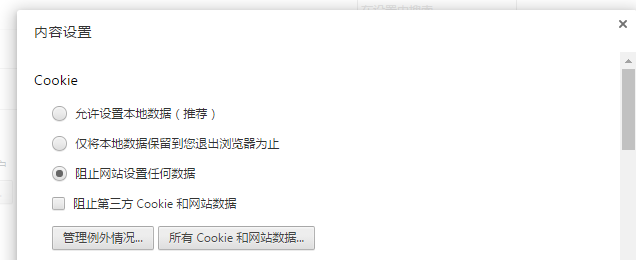


## SESSION基本使用

### 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。



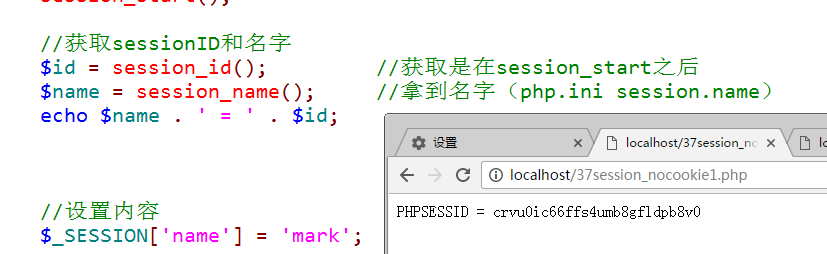
解决思路：最终让session\_start在开启之前拿到原来的sessionID（另外一个脚本的）

#### 实现无COOKIE使用SESSION

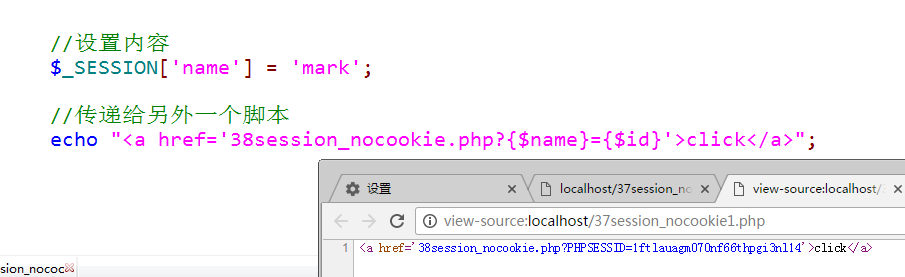
在PHP中，想要解决没有COOKIE也实现session技术的方式有两种：

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）：

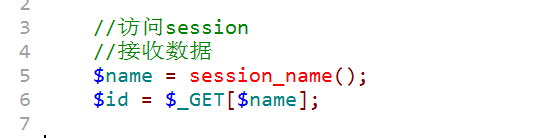
1、 在session保存数据的脚本中获取sessionID和名字



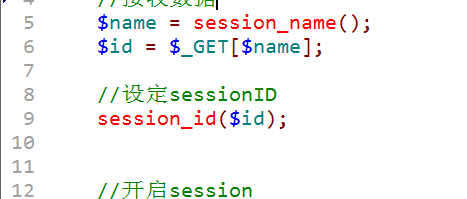
2、 想办法将数据传递给另外一个脚本：表单传值（URL或者form表单）



3、 在需要使用到session的脚本中，先接收数据



4、 组织session\_start产生新的ID，告诉它已经存在：session\_id($id)

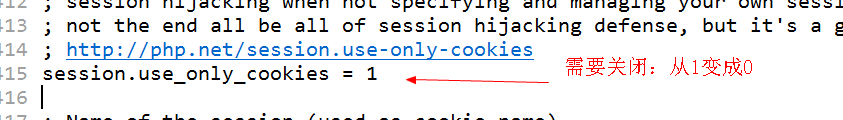


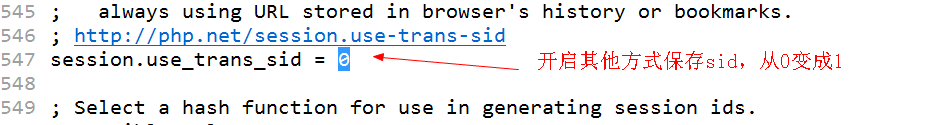
方案2：可以利用session集中已经提供的解决方案自动操作（配置）

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





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

