## 什么是会话控制

出现原因:http是无状态协议的,每次都是不同的服务器请求,无法维护两个请求事务之间的关系.

会话控制:能够在网站中跟踪一个变量,通过对变量的跟踪,就可以使多个请求事物之间建立联系,根据授权和用户身份显示不同的内容,不同页面

cookie:是服务器设置在客户端的文本文件,保存客户端的个人信息.

## 什么是Cookie:

1. cookie是在http协议下,服务器或脚本在客户端保存维护信息的一种方式
2. cookie是web服务器保存在客户端的一个很小的文本文件,保存客户端的信息,提高网页的处理效率,降低服务器负担
3. cookie保存在客户端,通常在浏览器的cookie临时文件夹中,可以手动的删除.如果cookie太多超过系统的允许范围,系统将自动删除

## cookie的工作原理

1. 当用户访问基于php技术的网站时,在php中可以使用setcookie()函数生成一个cookie,系统经过处理,将cookie发送到客户端并保存在C:\Documents and Setting\用户名\Cookies目录下
2. cookie是http标头的一部分,在它之前不能有任何输出，空格和换行都不可以
3. 当客户端再次访问网站时,浏览器会自动把cookie目录下与该站点对应的cookie信息发送到服务器,服务器将自动把客户端传来的cookie转化成php的变量,.php5中存放在$\_COOKIE全局数据中
4. 接受和处理cookie.使用$\_COOKIE获取

## COOKIE的参数设定

setcookie(键，值，有效期=0，有效路径=‘’，有效域名=‘’，是否仅安全连接传输=false，是否HTTPonly=false)

1、 键：设置cookie的名字，数据是在响应阶段被传输到浏览器端的

2、 值：设置cookie的值

Tip：增加，修改，删除，都使用该函数setcookie()完成！

3、 有效期：默认浏览器关闭（会话周期结束）时，COOKIE失效。也称之为：会话COOKIE

（临时COOKIE）

允许设置COOKIE的有效期，语法上，使用setcookie的第三个参数进行设置，通过unix时间戳（从1970年到现在的秒数，PHP函数time()来获取当前时间戳。）来设置COOKIE的有效期。

当用户下次访问您的站点的时候，浏览器会先检查您站点的所有cookie,如果某个cookie已经过期，浏览器不会把这个cookie随页面请求一起发送给服务器，而是删除这个已经过期的cookie。时间点是从cookie被创建时候当前时间开始计算。

谁来判断该COOKIE是否过期？(是否在有效期内)

浏览器 OR 服务器？

是浏览器判断该COOKIE是否过期的。

在设置COOKIE时，同时将COOKIE的有效期，告知（传输）到浏览器端。

如何删除COOKIE：

Time()-1: 删除COOKIE通用做法，强制设为过期。

注意: 当cookie被删除时,它的值在当前页面仍然有效.如果要把cookie设置成在浏览器关闭后就失效.那么可以把expiretime的值设为0,或者不设此值

4、 path有效路径（使用较少）：默认仅在当前目录及其后代目录有效。使用空字符串表示

默认值！通常如果需要设置，就是将COOKIE设置为 / 根目录下有效，表示整站有效。

5、 有效域（使用频率较高）：某个域名下设置的COOKIE，仅仅可以在当前域名下所使用

COOKIE，支持在一级域名内（所有的二级域名之间）进行COOKIE数据共享。

6、 secure是否仅安全连接（https）传输：

告知浏览器，我们所设置的cookie，是否仅仅在https这个协议下，才会被发送到浏览器端！ 用 参数 第六个完成设置：默认值false，表示都可以（http和https）。

7、 HTTPONLY：

所设置好的COOKIE，是否仅仅在http请求时所使用。

COOKIE还可以通过浏览器端的脚本（javascript）获取到。

function getCookie(){

alert(document.cookie);

}

如果需要禁止浏览器端其他脚本使用该cookie，则使用第七个参数即可，默认false，设置为true表示除了http请求，其它地方（javascript）不可以用

## COOKIE的有效期：

设置方法如下：  
　　setcookie("cookie\_name", "abcd", time()+3600);  
　　解释：这个cookie有3600秒的有效期；  
　　cookie有效期在退出作用域那一刻开始生效；  
　　重新访问又有3600秒，设置cookie时，不会考虑之前的是否设置cookie和cookie是否有效，可以认为是强制设置。  
　　如果关闭浏览器重新打开又重新算（当然这个要在浏览器端设置了关闭浏览器立即失效的开关）。cookie其实就是从打开页面产生cookie开始算起，如果你打开页面产生cookie的时间有1800秒了，假如设置的3600秒失效，再次刷新页面（不重新设置cookie的情况下）就剩1800秒了

PHP手册上关于setcookie函数的expire参数有这样一个解释：  
If set to 0, or omitted, the cookie will expire at the end of the session (when the browser closes).   
如果设为0 或者忽略该参数, cookie将在session结束时(或关闭浏览器时)过期

## COOKIE的注意事项

1. setcookie()之前不能有任何的输出,空白和空格也不行(Setcookie()为头信息设置)
2. setcookie()之后在当前页面echo时不会输出,必须刷新或到下一个页面才会输出
3. 不同浏览对cookie的处理不同,客户端可以禁用cookie,而且对浏览器的数量有限制.一个浏览器可以最多创建300个cookie,每个不能超过4kb,每个站点最多设置20个
4. 避免过度依赖cookie,因为客户端会禁止掉cookie

## 什么是session?

1. session从用户访问页面开始,到断开与网站的连接为止,形成一个会话的生命周期.在会话期间,分配客户唯一的一个sessionID,用来标识当前用户,与其他用户进行区分
2. session会话时,session会分别保存在客户端和服务器端的两个文件,对于客户端:使用临时的cookie保存(cookie的名称为PHPSESSID)或者通过url字符串的形式传递.服务器端也以文本的形式保存在指定的session目录中
3. session通过id接受每一次访问的请求,从而识别当前用户,跟踪和保持用户的具体资料,以及session变量,可在session中存储数字或文字资料.比如session\_name.这些信息都保存在服务器端
4. sessionID可以作为会话信息保存到数据库中,进行session持久化.这样可以跟踪用户的登陆次数,在线与否,在线时间

## 如何销毁session

1、unset($\_SESSION['XXX']):用于删除单个session变量

注意:不要使用unset($\_SESSION),此函数不可用,会删除全局的$\_SESSION销毁

2、$\_SESSION=array():删除多个session变量，清空session里的变量

3、销毁session\_destroy():结束当前的会话,并清空会话中的所有资源.该函数不会unset(释放)和当前session相关的全局变量,也不会删除客户端的cookie

4、清空session值

session\_start(); 使用session前必须先开启session ！！！！！！！！

$\_SESSION=array();

## session的过期时间

session的过期时间由两方面决定的;

1存储在客户端的$\_COOKIE['PHPSESSID']的过期时间（默认cookie名称为PHPSESSID,可通过php.ini中的session.name修改。）

2.存储在服务器端的相对应的session文件（session文件名和上述cookie的值一一对应），默认为1440秒，即24分钟

两者的关系： 当执行session\_start()的时候，其实是做了两件事：

1，检查客户端发送过来的的所有cookie（当然也包括$\_COOKIE['PHPSESSID'], 如果有的话），根据$\_COOKIE['PHPSESSID']的值（这是由apache产生的随机字符串，如0lkbd2se458r600m2m7o1r4ic5）来访问 相对应的 session文件，这两者是一 一对应的关系。当然文件里面的值是经过序列化的

2，如果客户端没有传来$\_COOKIE['PHPSESSID']，就会有服务端产生一个随机的$\_COOKIE['PHPSESSID']并存储在客户端。

**需要理解：**

1、在session\_start()开启之后，当session数据对应的PHPSESSIONID的cookie已经超时，无法传回到服务器端，此时由于session开启，系统会自动再重新分配一个SESSIONID标识，SESSIONID默认是存储于浏览器端。

2、同时在session\_start开启之后，如果之前有存储过session，并且没有过期，则会获取到已经存好的session，如果没有存储过session，则session机制默认也会自动分配一个SESSIONID给浏览器，同时在服务器端生成对应的文件，此时的session是一个过期的会话

## 如何设置 session数据的属性？

设置COOKIE中session-ID这个COOKIE变量属性即可！

方案一，配置 php.ini

方案二，在脚本中，开启session之前使用函数进行配置

ini\_set('session.cookie\_lifetime','3600');

ini\_set('sesssion.cookie\_domain',' .baidu.com');

@session\_start();

采用下面的函数进行设置：

Session\_set\_cookie\_params(有效期，有效路径，有效域，是否仅安全连接传输，是否HTTPONLY)

实际环境中，很少改session的有效期。经常改有效域名。

session\_set\_cookie\_params(0,'/', '.baidu.com');

## Session数据区操作：

重写session的存储机制：

Session数据区默认以 文件的形式存储与服务器操作系统临时目录中！

当 session数据区过多时，文件形式的存储，操作速度变慢，磁盘的读写开销很大

重写session入数据库：

共6个函数

Session\_set\_save\_handler(开始函数，结束函数，读函数，写函数，删除函数，GC函数);

告知session机制，在需要读写时，使用用户自定义的读写函数完成！

这个函数仅仅是设置告知，不是调用，以上6个函数，在session机制运行到某个时间点时，

才会被调用！

**垃圾回收操作： sessGC()**

垃圾：服务器上过时的session数据区。

垃圾如何判定？

如果一个session数据区已经超过多久没有使用（最后一次写操作）了，就是被视为垃圾数据。

该时间临界点：默认1440s。可以被配置：session.gc\_maxlifetime = 1440

判断过期条件：Last\_write < 当前时间-1440

垃圾如何删除？

在 session\_start()过程中，开启session机制过程中：有几率地执行 垃圾回收操作。

一旦执行，就会删除所有的过期的垃圾数据区。

默认的概率为1/1000。

可以设置该几率：

可能性：Session.gc\_probability

基数（除数）：Session.gc\_divisor

建议在脚本周期调整，使用函数ini\_set(),在开启session机制前完成

实现 sessGC()：PHP的session机制将 最大有效期作为参数，传递过来！

语法细节：

要注意先设置再开启session机制

PHP配置项：session.save\_handler改为 user： 表示用户自定义！

## 会话技术总结：

Session.save\_handler 存储处理器： files|user

Session.save\_path 存储地址。

Session.cookie\_XXX （lifetime，path，domain，secure，httponly）存储session-ID这个COOKIE变量的属性

Session.gc\_maxlifetime

Session.gc\_probability

Session.gc\_divisor

## Session，COOKIE联系和区别？

联系

都是会话技术。

Session基于COOKIE，session-ID存储于COOKIE中！

区别：

Cookie session

存储位置 浏览器端 服务器端

安全性 低 高

大小限制 有 没有

数据类型 字符串 除资源外的其它全部

有效期使用 长时间存 几乎不做持久化

## Session如何持久化？[理论]

1、Session-ID要持久化：

2、session\_set\_cookie\_params(3600);

服务器session数据区有效期修改：ini\_set(‘session.gc\_maxlifetime’, 3600);

## 浏览器禁用COOKIE，session是否可用？[理论]

COOKIE被禁用，session-Id不能存储和传输。

不可用！

理论上的解决方案：

通过 URL， 或者 POST数据数据向服务器端，每次传输session-ID！

例如下面的配置：php.ini

Session是否仅仅是用COOKIE完成传输session-ID：

session.use\_only\_cookie = 1 开启（在session\_start之前）

是否通过其他方式自动传输session-ID：

session.use\_trans\_sid = 0 开启

然后通过表单的隐藏域向服务器端，每次传输session-ID！即可

## session和cookie的额外的一些理解：

1. 由于HTTP协议是无状态的协议，所以服务端需要记录用户的状态时，就需要用某种机制来识具体的用户，这个机制就是Session.典型的场景比如购物车，当你点击下单按钮时，由于HTTP协议无状态，所以并不知道是哪个用户操作的，所以服务端要为特定的用户创建了特定的Session，用用于标识这个用户，并且跟踪用户，这样才知道购物车里面有几本书。这个Session是保存在服务端的，有一个唯一标识。在服务端保存Session的方法很多，内存、数据库、文件都有。集群的时候也要考虑Session的转移，在大型的网站，一般会有专门的Session服务器集群，用来保存用户会话，这个时候 Session 信息都是放在内存的，使用一些缓存服务比如Memcached之类的来放 Session。

2. 思考一下服务端如何识别特定的客户？这个时候Cookie就登场了。每次HTTP请求的时候，客户端都会发送相应的Cookie信息到服务端。实际上大多数的应用都是用 Cookie 来实现Session跟踪的，第一次创建Session的时候，服务端会在HTTP协议中告诉客户端，需要在 Cookie 里面记录一个Session ID，以后每次请求把这个会话ID发送到服务器，我就知道你是谁了。有人问，如果客户端的浏览器禁用了 Cookie 怎么办？一般这种情况下，会使用一种叫做URL重写的技术来进行会话跟踪，即每次HTTP交互，URL后面都会被附加上一个诸如 sid=xxxxx 这样的参数，服务端据此来识别用户。

3. Cookie其实还可以用在一些方便用户的场景下，设想你某次登陆过一个网站，下次登录的时候不想再次输入账号了，怎么办？这个信息可以写到Cookie里面，访问网站的时候，网站页面的脚本可以读取这个信息，就自动帮你把用户名给填了，能够方便一下用户。这也是Cookie名称的由来，给用户的一点甜头。

所以，总结一下：

Session是在服务端保存的一个数据结构，用来跟踪用户的状态，这个数据可以保存在集群、数据库、文件中；

Cookie是客户端保存用户信息的一种机制，用来记录用户的一些信息，也是实现Session的一种方式。