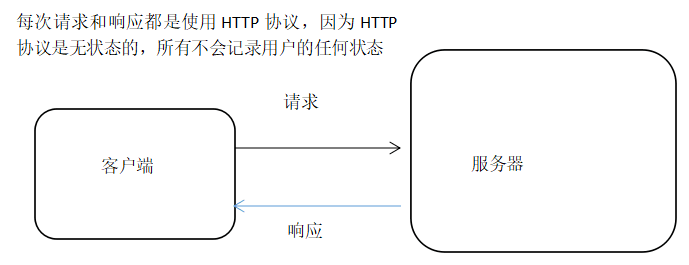
# 会话控制

在计算机中，客户端和服务器之间的通讯就是会话。

而对会话进行控制就是会话控制。



## COOKIE

### COOKIE的工作原理

COOKIE是在HTTP协议下，服务器或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，使Web每个站点都可以访问COOKIE信息。

### COOKIE使用方法

* 添加COOKIE值：
* setcookie(‘下标’,’值’,’有效时间’,’/’);
* 参数四：/ ->有效作用域，表示当前网站下好使
* 修改COOKIE值：
* setcookie(‘下标’,’新值’,’有效时间’,’/’);
* 删除COOKIE值：
* setcookie(‘下标’,’空值’,’过期时间’,’/’)
* 查看COOKIE值：
* COOKIE所有的值都是依赖于系统提供的$\_COOKIE预定义数组变量。所有COOKIE设置的值都会成为该数组的一个元素
* 查看COOKIE值，使用var\_dump()或者print\_r()。如果使用特定的COOKIE的值可直接加下标使用，$\_COOKIE[‘下标’];
* $\_COOKIE是一个可以跨页面的系统预定义变量
* 注意：$\_COOKIE在设置当前页面时无法读取信息。
* COOKIE的缺点：
* 相对不安全。COOKIE信息容易被盗取，容易泄露个人隐私。
* 它必须依赖于客户端的运行(客户端可以关闭COOKIE)。

## SESSION

### SESSION会话控制原理

session的操作和数组一模一样。

注意：所有的session的操作必须在页面开始部分使用session\_start()开启SESSION的会话。

### SESSION使用方法

* 添加session
* $\_SESSION[‘下标’] = 值;
* 修改session
* $\_SESSION[‘下标’] = 新值;
* 删除session
* 删除session的变量值(保留数据类型)
* $\_SESSION = array();
* 删除所有SESSION(包括文件)，一般使用在退出操作中
* 使客户端COOKIE中的SESSIONID过期
* 清空页面SESSION的变量值
* 摧毁服务器的SESSION文件

session\_destroy()

### SESSION相关的配置文件信息

* session.save\_handler = files 设定SESSION 文件的存储方式。
* files以文件存储
* session.save\_path = “c:/wamp64/tmp” 设定SESSION文件的存放路径。
* session.use\_cookies = 1 设定是否使用COOKIE来存储SESSION的ID。
* session.use\_only\_cookies = 1 设定SESSION是否只用COOKIE来存储ID。
* session.name = PHPSESSID 设置存储SESSIONID的下标名称。
* 同时也可以通过session\_name() 来获取设置的值。
* session.auto\_start = 0 是否自动开启session。
* 如果开启会让类无法使用，因为类的加载必须在session之前。
* session.cookie\_lifetime = 0 设定session使用COOKIE的有效生存时间。
* 0表示关闭浏览器前有效。
* session.cookie\_path = / 设置session使用COOKIE的有效路径。
* session.cookie\_domain = 设置session使用COOKIE的有效域名。
* 空表示当前域名
* session.cookie\_httponly = 设置SESSION使用COOKIE是否指允许使用HTTP协议访问。
* session.serialize\_handler = php SESSION串行化的方式。
* session.gc\_probability = 1 垃圾回收几率的除数。
* session.gc\_divisor = 1000 垃圾回收几率的被除数。
* session.gc\_maxlifetime = 1440 session垃圾回收的最大时间。
* session.use\_trans\_sid = 0 SESSION是否使用传送方式传session的ID。
* session.hash\_function = 0 SESSION 文件名的加密方式。
* url\_rewriter.tags = "a=href,area=href,frame=src,input=src,form=fakeentry"

设定session.use\_trans\_sid 可用的标签。

### SESSION与COOKIE的区别

将两者比作办会员卡，COOKIE就好比办好会员卡后，卡自己拿着。而SESSION好比办好卡后，卡放到店里面到时候用时报自己的手机号即可。

### SESSION工作原理

ID存放到COOKIE文件中，然后文件存入到服务器里。因此COOKIE被禁用的话那SESSION也用不了了。但是我们可以通过另一种方式来实现(url的传递)

### 基于URL的SESSION的会话原理

在第一个页面获取对应的SESSION的ID，然后通过超链接的方式(自动和手动)将ID传送到下一个需要使用SESSIONID的页面，这种方式就是基于URL传送的SESSION会话。

* 手动：

手动比较麻烦，就是一个页面一个页面的获取传送因此推荐使用自动。

* 自动：

自动需要修改配置文件，session.use\_trans\_sid = 1 开启自动传送方式 session.use\_only\_cookies= 0 关闭session仅用COOKIE方式进行工作的选项。

## 如何在多台web服务器上共享session

分析:从负载均衡的角度讲，一台web服务器存储的session只存才自己的服务器上，不能让其他服务器共享，这样会降低用户的体验效果。因为session只存在自己的服务器上无法共享，那么我们就要想办法将session存在多态服务器都能获取的地方即可。

方法：

1：如果session的东西不多我们可以用cookie来代替。毕竟cookie是存在客服端的。但弊端是很明显的，第一不安全，第二存储的数据大小有限，第三会影响页面的访问速度

2：session的id还是存在cookie中，但原存服务器上的数据让其存到数据库中。这样也能实现，但问题也随着而来你，小的访问是可以，但如果高并发呢？无疑是加重了数据库的压力。

3：存在mamcache或者redis中，两者都存在内存中，并且对高并发的情况的又很合适。