浅谈 cookie 和 session、sessionStroage 的关系

cookie 会每次随 http 请求一起发送。

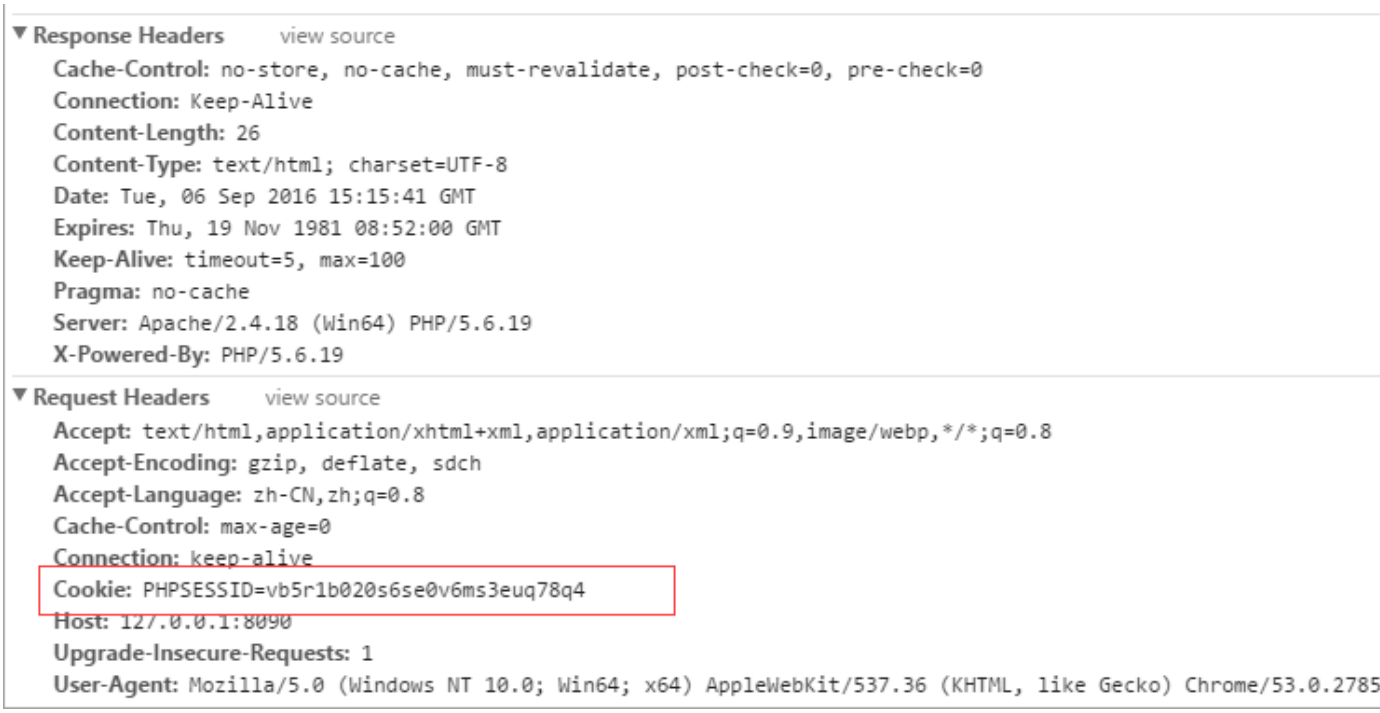
seesionStorage 的数据不会跟随 HTTP 请求一起发送到服务器，只会在本地生效，并在关闭标签页后清除数据

```php
<?php

session_start();

echo $sid = session_id();

$_SESSION['name']='zhezhao';

$_SESSION['age']=23;
```

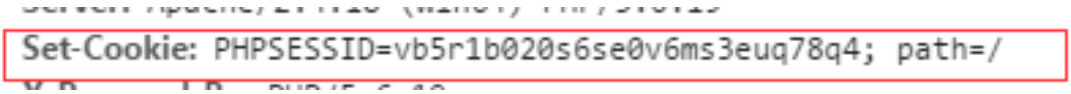
首次访问

我们可以看到，首次访问该页面，请求的 http 协议头并没有包含 cookie 信息；返回的 http 协议头包含了设置 cookie 信息的内容。设置 cookie：PHPSESSID 的值为 session_id()的内容，有效期为当前会话，关闭浏览器则该 cookie：PHPSESSID 失效。

再次访问



然后我们刷新一下，再访问该页面



此时，浏览器的 http 请求头中包含了 cookie：PHPSESSID 的信息。

`session_start();`函数会检查请求头中是否包含 PHPSESSID 的 cookie 信息，如果有则将其作为 session 的唯一标识，即 session_id()的值。如果没有 PHPSESSID 的 cookie 信息，则重新生成一个 session_id()并将该值设置为 cookie：PHPSESSID 的值。

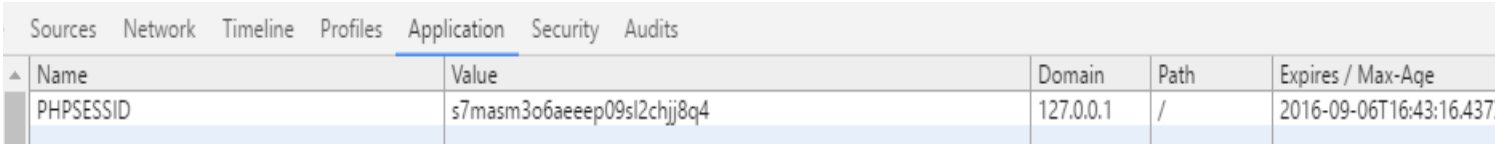
首次访问该页面时，并没有任何 cookie 信息，所以此时 php 生成了 session_id 并在返回的 http 头文件中设置了 PHPSESSID 这个 cookie 的值。

再次访问页面的时候，http 请求头文件中会包含该域名下的 cookie，所以 php 就获取到了 PHPSESSID 的值，将其作为 session_id()的值。

下面我们来解释一下，为什么 session 默认的有效期是浏览器的当前会话，关闭浏览器即失效；以及如何设置 session 的有效期？

session 的默认有效期

在浏览器首次访问页面的时候，session_start()判断没有 PHPSESSID 这个 cookie，所以自己生成了一个 session_id()并将其设置为 PHPSESSID 的值。



我们注意到并没有设置 cookie 的有效期，所以使用的是 cookie 的默认有效期，即浏览器的当前会话，关掉浏览器，则 PHPSESSID 这个 cookie 就会失效，重新打开该页面的时候，http 请求头就没有 PHPSESSID 这个 cookie 了，相当于一个新用户来访问。

所以 session 的默认过期时间和 cookie 的默认过期时间是一致的。

修改 session 的失效时间

通过上面的分析，我们发现 php 在自动设置 PHPSESSID 这个 cookie 的值的时候，没有设置过期时间，所以 cookie 过期，session 就过期了。

我们可以手动设置 PHPSESSID 这个 cookie 的过期时间，也就相当于设置了 session 的过期时间。

```php
<?php

session_start();

echo $sid = session_id();

setcookie('PHPSESSID', session_id(), time() + 60*60);   //设置过期时间为 1 小时

$_SESSION['name']='zhezhao';

$_SESSION['age']=23;
```

php 对 session 的处理机制

session 在 php 中是通过文件存储的，存储路径可以在 php 的配置文件中找到

```
session.save_path ="c:/wamp64/tmp"
```

通过上面的分析，我们知道默认情况下，关闭浏览器。PHPSESSID 这个 cookie 的值就会失效，再次访问该站点的时候，会重新生成一个 PHPSESSID 的值。然而，此时原来的 PHPSESSID 这个 cookie 的值虽然已经失效，但是 session_id 对应的存储 session 的文件可能还没有被删除，我们如果知道原来 session_id 的值的话，可以通过 `session_id("原来的 session_id 的值")`这个函数来实现恢复 session 的操作的。

那么存储 session 的文件在什么时候会被删除呢？
它和下面几个 php 的配置参数有关

```
session.gc_maxlifetime = 1440

session.gc_probability = 1

session.gc_divisor = 1000
```

gc_maxlifetime 表示 session 的过期时间，从最后一次访问 session 开始算起，超过这个时间就会被当做系统垃圾，启动 gc（垃圾清理）机制。

然而处于性能考虑，session 过期之后，并不是会被立刻删除。而是有一定的概率（gc_probability/gc_divisor）被删除，默认情况下是 0.1% ，也就是说有 0.1% 的概率，启动 gc 流程，清理 cookie。每一次 php 请求，就会有 1/1000 的概率发生回收；1000 次 php 请求，就会进行一次垃圾回收。

如果将 gc_divisor 的值也设置为 1 的话，启动 gc 的概率为 1。也就是说，只要到达了 gc_maxlifetime 所设定的时间，就会立刻被清理。

重要补更

很多网站都有 7 天免登录功能，这种情况下 cookie 和 session 的生存时间都应该为 7 天。

设置 cookie 的有效期为 7 天

```
setcookie('PHPSESSID', session_id(), time() + 60*60*24*7);  //过期时间为7天
```

上面讲过，session 是以文件的形式存储在服务器上的，php 通过 session.gc_maxlifetime 参数来决定 session 文件的过期时间，过期之后就可能会被清理。所以，我们需要将 session 的过期时间设置为 7 天。

```
session.gc_maxlifetime = 604800;
```

通过 get 方式传递 session_id

设置方法

修改 php.ini 的如下配置，并重启 web 服务器

```
session.use_trans_sid = 1;

session.use_only_cookies = 0;
```

通过上文，我们知道 SESSION 默认依赖 cookie 所传递的 session_id 的唯一标识 SESSION。如果客户端禁用 cookie 时，我们还可以通过 get 的方式传递 session_id，使 SESSION 正常工作。
修改 php.ini 生效之后，PHP 编译器在将 php 文件进行解析时会做如下操作：

1. 在所有的站内 url 后面添加 PHPSESSIOD 参数，对于 http://www.baidu.com 这种站外 url 无效 。
2. 在 session_start();命令执行之前，会自动执行如下操作 session_id($_GET[“PHPSESSID”]) 。

```
http://192.168.20.131/one.php?PHPSESSID=eckva0me0t1d4t4i3m8f496rb3
```

如何判断浏览器是否禁用了 cookie 呢？

```
echo defined('SID')?'true':'false';
```

可以通过 SID 是否被定义来判断，让 cookie 被禁用时，SID 常量才会被定义。通过输出我们可以发现 SID 的值就是通过 URL 进行传递的 `PHPSESSID=eckva0me0t1d4t4i3m8f496rb3` 字符串。

事实上，cookie 被禁用之后，SID 常量就会被定义。但是要按照上面修改完 php.ini 的参数之后，才能 SID 才能作为 session_id 被 SESSION 使用。

在《HTTP 权威指南》的第 11 章，有讲到在 cookie 之前的一些用户识别机制。

> 本章对 HTTP 识别用户的几种技巧进行了总结。HTTP 并不是天生就具有丰富的识别特性的。早期的 Web 站点设计者们（他们都是些注重实际的人）都有自己的用户识别技术。每种技术都有其优势和劣势。本章我们将讨论下列用户识别机制。
>
> - 承载用户身份信息的 HTTP 首部。
> - 客户端 IP 地址跟踪，通过用户的 IP 地址对其进行识别。
> - 用户登录，用认证方式来识别用户。
> - 胖 URL，一种在 URL 中嵌入识别信息的技术。

上面所讲到的通过 URL 传递 seesion_id 实际上就是书中所讲的“胖 URL”技术，该技术存在以下缺点。

> - 无法共享 URL
>
> 胖 URL 中包含了与特定用户和会话有关的状态信息。如果将这个 URL 发送给其他人，可能就在无意中将你积累的个人信息都共享出去了。
> - 破坏缓存
>
> 为每个 URL 生成用户特有的版本就意味着不再有可供公共访问的 URL 需要缓存了。
> - 额外的服务器负荷
>
> 服务器需要重写 HTML 页面使 URL 变胖。
> - 逃逸口
>
> 用户跳转到其他站点或者请求一个特定的 URL 时，就很容易在无意中“逃离”胖 URL 会话。只有当用户严格地追随预先修改过的链接时，胖 URL 才能工作。如果用户逃离此链接，就会丢失他的进展（可能是一个已经装满了东西的购物车）信息，得重新开始。
> - 在会话间是非持久的
>
> 除非用户收藏了特定的胖 URL，否则用户退出登录时，所有的信息都会丢失。