

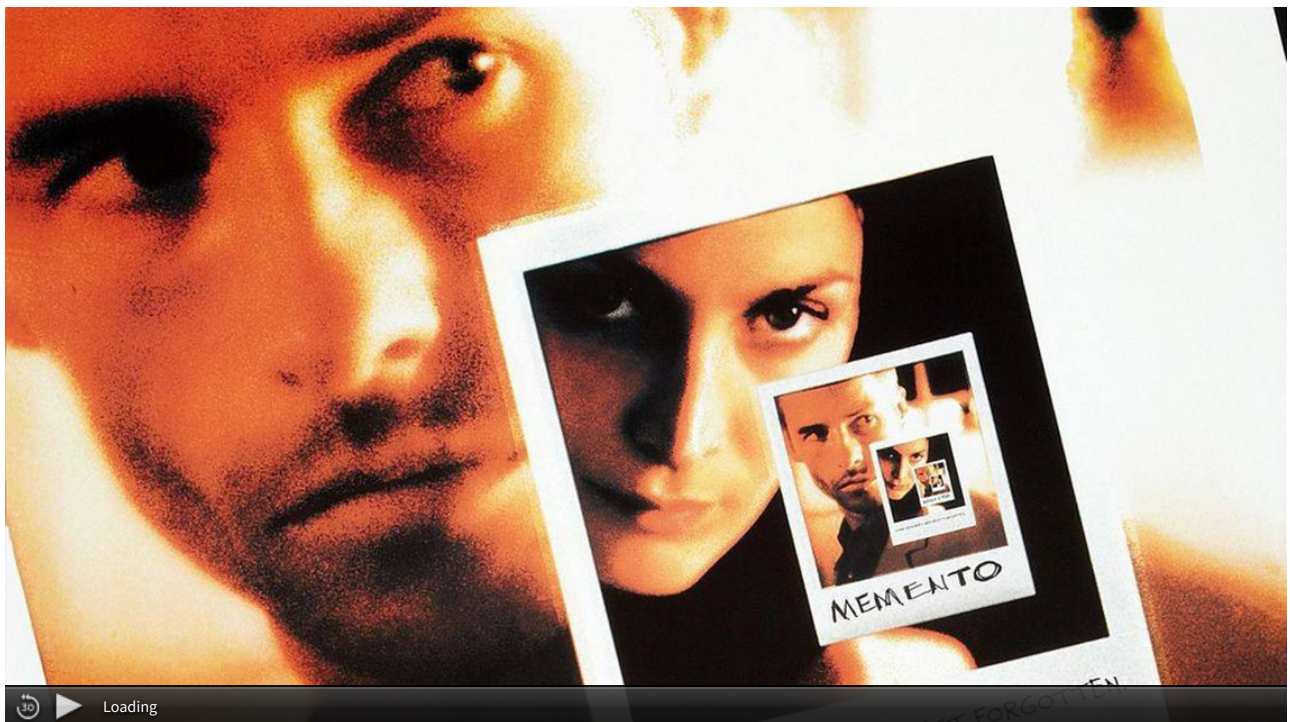
## 19-让我知道你是谁：HTTP的Cookie机制

在之前的第13、14讲中，我曾经说过，HTTP是“无状态”的，这既是优点也是缺点。优点是服务器没有状态差异，可以很容易地组成集群，而缺点就是无法支持需要记录状态的事务操作。

好在HTTP协议是可扩展的，后来发明的Cookie技术，给HTTP增加了“记忆能力”。

### 什么是Cookie？

不知道你有没有看过克里斯托弗·诺兰导演的一部经典电影《记忆碎片》（Memento），里面的主角患有短期失忆症，记不住最近发生的事情。



比如，电影里有个场景，某人刚跟主角说完话，大闹了一通，过了几分钟再回来，主角却是一脸茫然，完全不记得这个人是谁，刚才又做了什么，只能任人摆布。

这种情况就很像HTTP里“无状态”的Web服务器，只不过服务器的“失忆症”比他还要严重，连一分钟的记忆也保存不了，请求处理完立刻就忘得一干二净。即使这个请求会让服务器发生500的严重错误，下次来也会依旧“热情招待”。

如果Web服务器只是用来管理静态文件还好说，对方是谁并不重要，把文件从磁盘读出来发走就可以了。但随着HTTP应用领域的不断扩大，对“记忆能力”的需求也越来越强烈。比如网上论坛、电商购物，都需要“看客下菜”，只有记住用户的身份才能执行发帖子、下订单等一系列会话事务。

那该怎么样让原本无“记忆能力”的服务器拥有“记忆能力”呢？

看看电影里的主角是怎么做的吧。他通过纹身、贴纸条、立拍得等手段，在外界留下了各种记录，一旦失忆，只要看到这些提示信息，就能够在头脑中快速重建起之前的记忆，从而把因失忆而耽误的事情继续做下去。

HTTP的Cookie机制也是一样的道理，既然服务器记不住，那就在外部想办法记住。相当于是服务器给每个

客户端都贴上一张小纸条，上面写了一些只有服务器才能理解的数据，需要的时候客户端把这些信息发给服务器，服务器看到Cookie，就能够认出对方是谁了。

## Cookie的工作过程

那么，Cookie这张小纸条是怎么传递的呢？

这要用到两个字段：响应头字段**Set-Cookie**和请求头字段**Cookie**。

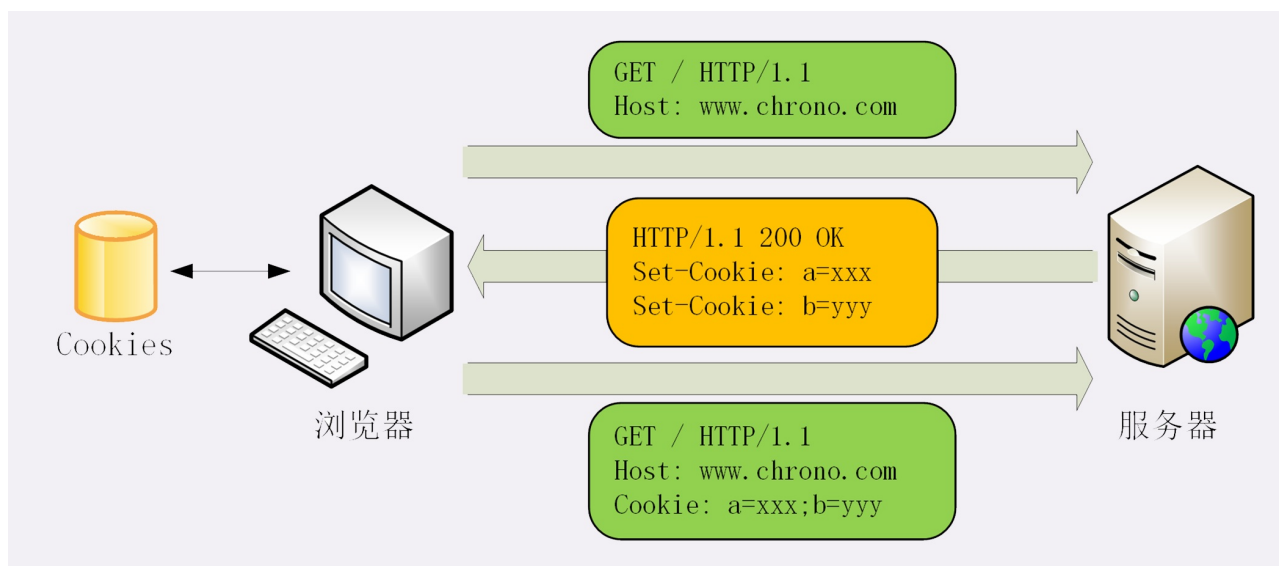
当用户通过浏览器第一次访问服务器的时候，服务器肯定是不知道他的身份的。所以，就要创建一个独特的身份标识数据，格式是“**key=value**”，然后放进Set-Cookie字段里，随着响应报文一同发给浏览器。

浏览器收到响应报文，看到里面有Set-Cookie，知道这是服务器给的身份标识，于是就保存起来，下次再请求的时候就自动把这个值放进Cookie字段里发给服务器。

因为第二次请求里面有了Cookie字段，服务器就知道这个用户不是新人，之前来过，就可以拿出Cookie里的值，识别出用户的身份，然后提供个性化的服务。

不过因为服务器的“记忆能力”实在是太差，一张小纸条经常不够用。所以，服务器有时会在响应头里添加多个Set-Cookie，存储多个“key=value”。但浏览器这边发送时不需要用多个Cookie字段，只要在一行里用“;”隔开就行。

我画了一张图来描述这个过程，你看过就能理解了。

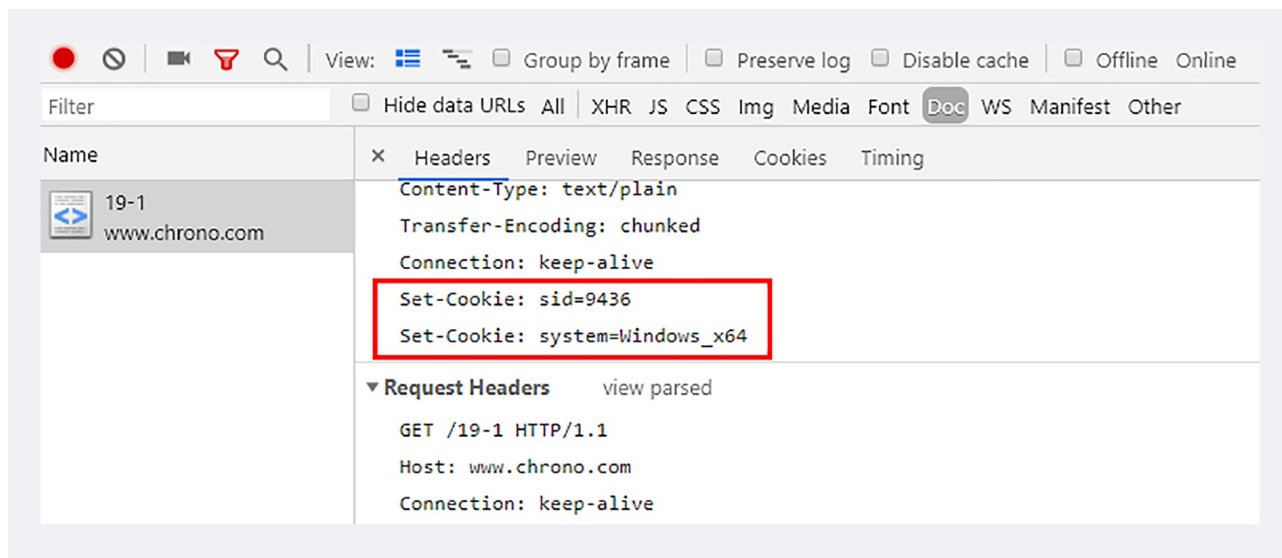


从这张图中我们也能够看到，Cookie是由浏览器负责存储的，而不是操作系统。所以，它是“浏览器绑定”的，只能在本浏览器内生效。

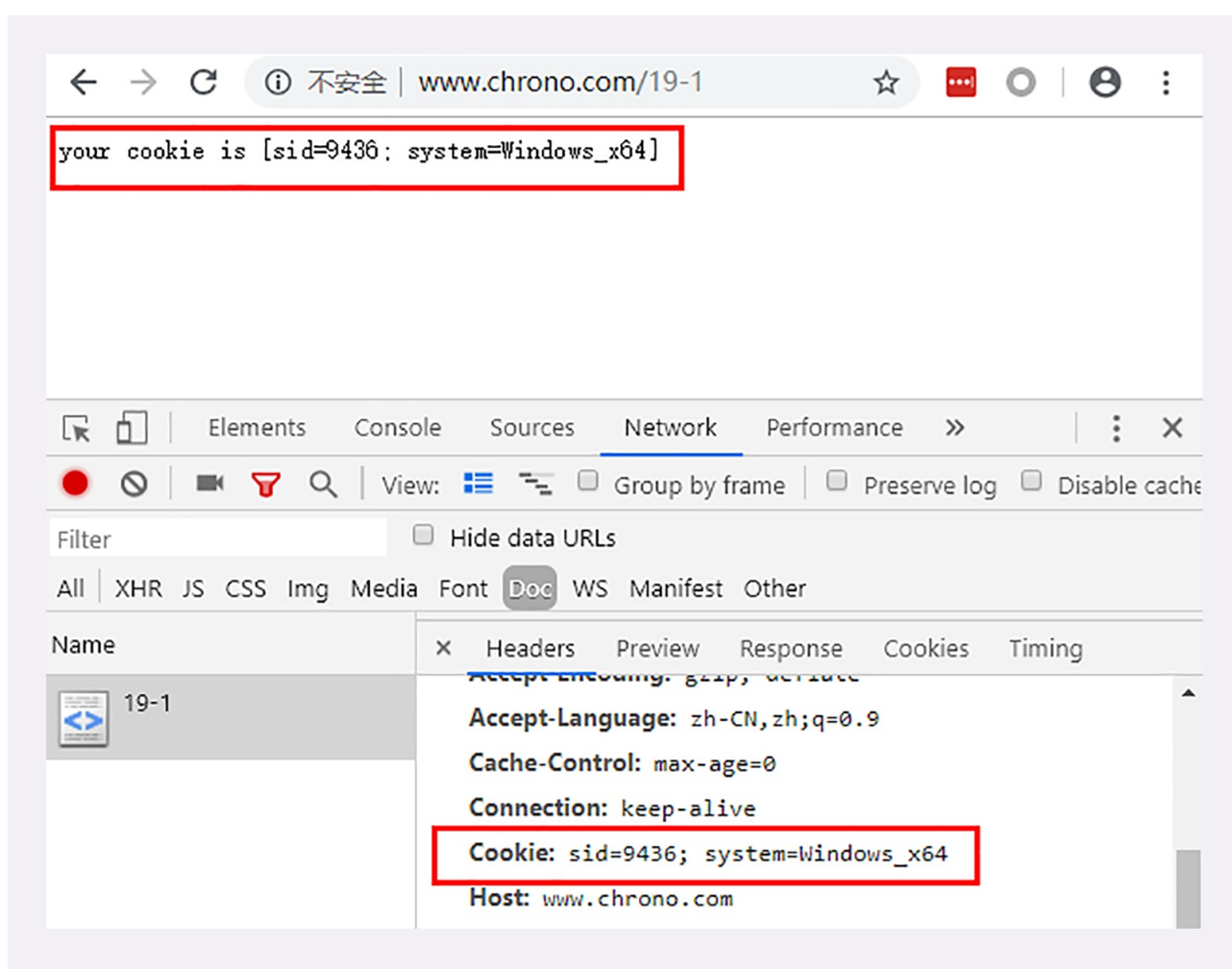
如果你换个浏览器或者换台电脑，新的浏览器里没有服务器对应的Cookie，就好像是脱掉了贴着纸条的衣服，“健忘”的服务器也就认不出来了，只能再走一遍Set-Cookie流程。

在实验环境里，你可以用Chrome访问URI“/19-1”，实地看一下Cookie工作过程。

首次访问时服务器会设置两个Cookie。



然后刷新这个页面，浏览器就会在请求头里自动送出Cookie，服务器就能认出你了。

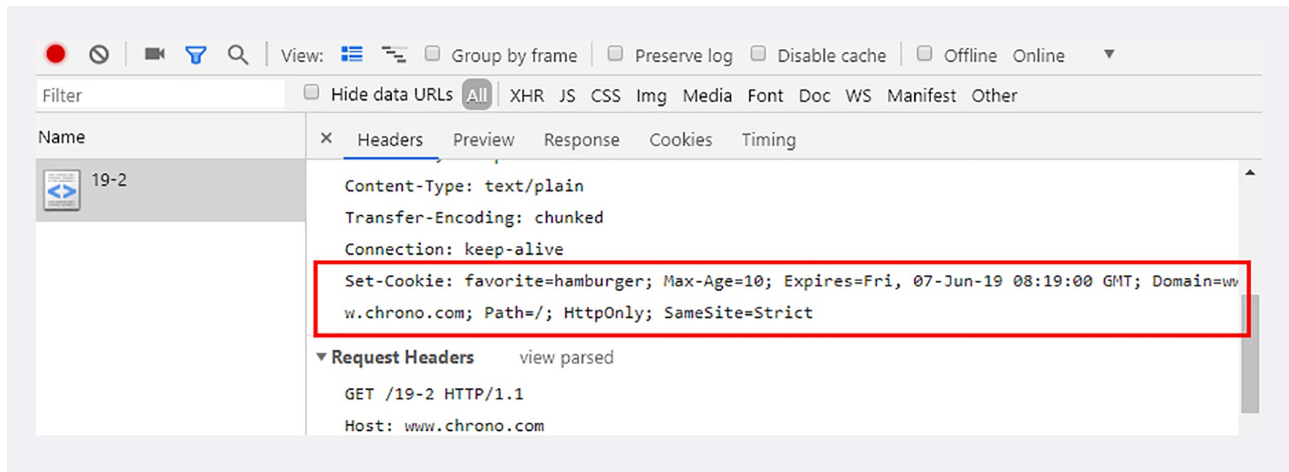


如果换成Firefox等其他浏览器，因为Cookie是存在Chrome里的，所以服务器就又“蒙圈”了，不知道你是谁，就会给Firefox再贴上小纸条。

## Cookie的属性

说到这里，你应该知道了，Cookie就是服务器委托浏览器存储在客户端里的一些数据，而这些数据通常都会记录用户的关键识别信息。所以，就需要在“key=value”外再用一些手段来保护，防止外泄或窃取，这些手段就是Cookie的属性。

下面这个截图是实验环境“/19-2”的响应头，我来对着这个实际案例讲一下都有哪些常见的Cookie属性。



首先，我们应该设置Cookie的生存周期，也就是它的有效期，让它只能在一段时间内可用，就像是食品的“保鲜期”，一旦超过这个期限浏览器就认为是Cookie失效，在存储里删除，也不会发送给服务器。

Cookie的有效期可以使用Expires和Max-Age两个属性来设置。

“Expires”俗称“过期时间”，用的是绝对时间点，可以理解为“截止日期”（deadline）。“Max-Age”用的是相对时间，单位是秒，浏览器用收到报文的时间点再加上Max-Age，就可以得到失效的绝对时间。

Expires和Max-Age可以同时出现，两者的失效时间可以一致，也可以不一致，但浏览器会优先采用Max-Age计算失效期。

比如在这个例子里，Expires标记的过期时间是“GMT 2019年6月7号8点19分”，而Max-Age则只有10秒，如果现在是6月6号零点，那么Cookie的实际有效期就是“6月6号零点过10秒”。

其次，我们需要设置Cookie的作用域，让浏览器仅发送给特定的服务器和URI，避免被其他网站盗用。

作用域的设置比较简单，“Domain”和“Path”指定了Cookie所属的域名和路径，浏览器在发送Cookie前会从URI中提取出host和path部分，对比Cookie的属性。如果不满足条件，就不会在请求头里发送Cookie。

使用这两个属性可以为不同的域名和路径分别设置各自的Cookie，比如“/19-1”用一个Cookie，“/19-2”再用另外一个Cookie，两者互不干扰。不过现实中为了省事，通常Path就用一个“/”或者直接省略，表示域名下的任意路径都允许使用Cookie，让服务器自己去挑。

最后要考虑的就是Cookie的安全性了，尽量不要让服务器以外的人看到。

写过前端的同学一定知道，在JS脚本里可以用document.cookie来读写Cookie数据，这就带来了安全隐患，有可能会导致“跨站脚本”（XSS）攻击窃取数据。

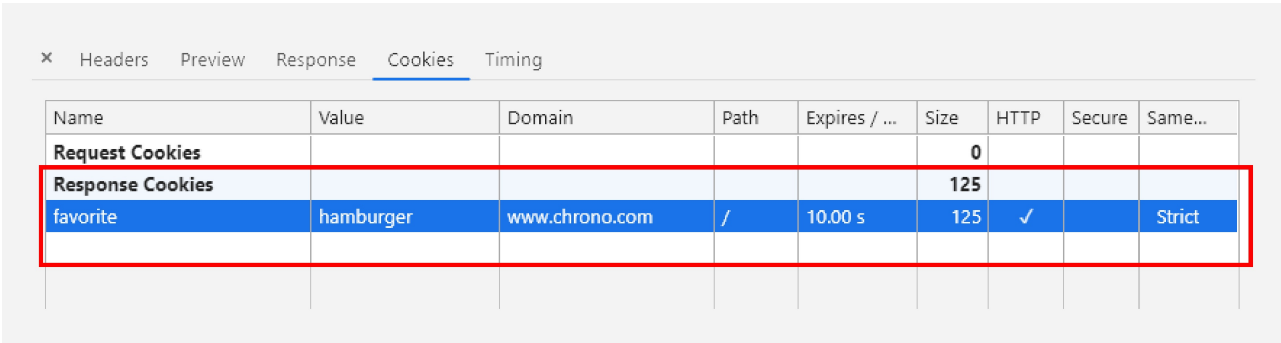
属性“HttpOnly”会告诉浏览器，此Cookie只能通过浏览器HTTP协议传输，禁止其他方式访问，浏览器的JS引擎就会禁用document.cookie等一切相关的API，脚本攻击也就无从谈起了。

另一个属性“SameSite”可以防范“跨站请求伪造”（XSRF）攻击，设置成“SameSite=Strict”可以严格限定Cookie不能随着跳转链接跨站发送，而“SameSite=Lax”则略宽松一点，允许GET/HEAD等安全方法，但禁止POST跨站发送。

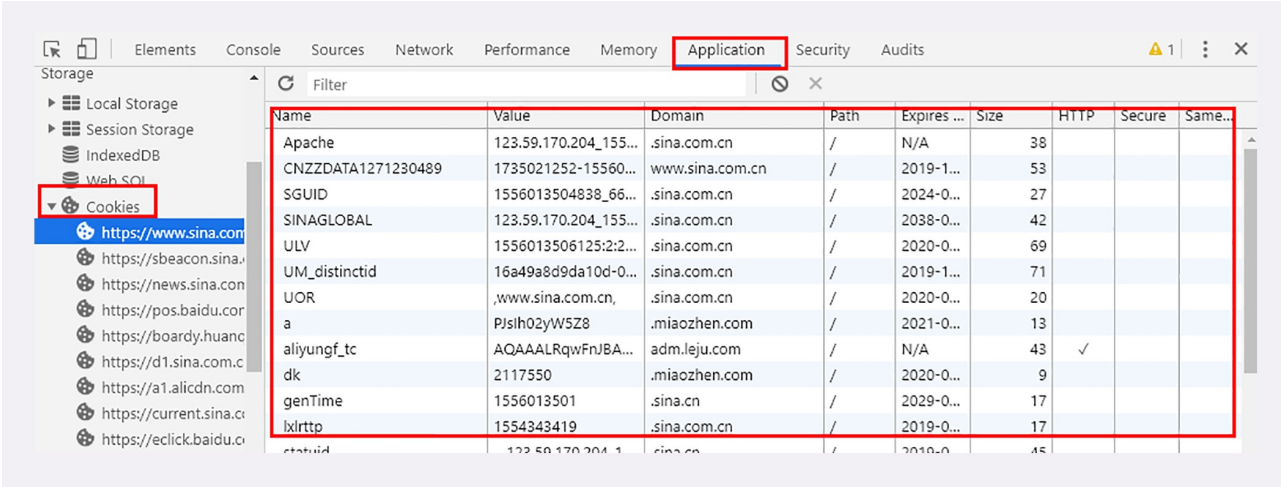


还有一个属性叫“Secure”，表示这个Cookie仅能用HTTPS协议加密传输，明文的HTTP协议会禁止发送。但Cookie本身不是加密的，浏览器里还是以明文的形式存在。

Chrome开发者工具是查看Cookie的有力工具，在“Network-Cookies”里可以看到单个页面Cookie的各种属性，另一个“Application”面板里则能够方便地看到全站的所有Cookie。



Name	Value	Domain	Path	Expires / ...	Size	HTTP	Secure	Same...
Request Cookies					0			
Response Cookies					125			
favorite	hamburger	www.chrono.com	/	10.00 s	125	✓		Strict



Name	Value	Domain	Path	Expires ...	Size	HTTP	Secure	Same...
Apache	123.59.170.204_155...	.sina.com.cn	/	N/A	38			
CNZZDATA1271230489	1735021252-15560...	www.sina.com.cn	/	2019-1...	53			
SGUID	1556013504838_66...	.sina.com.cn	/	2024-0...	27			
SINAGLOBAL	123.59.170.204_155...	.sina.com.cn	/	2038-0...	42			
ULV	1556013506125:2:...	.sina.com.cn	/	2020-0...	69			
UM_distinctid	16a49a8d9da10d-0...	.sina.com.cn	/	2019-1...	71			
UOR	.www.sina.com.cn,	.sina.com.cn	/	2020-0...	20			
a	PJslh02yW5Z8	.miaozhen.com	/	2021-0...	13			
aliyun_gf_tc	AQAAALRqwFnJBA...	adm.leju.com	/	N/A	43	✓		
dk	2117550	.miaozhen.com	/	2020-0...	9			
genTime	1556013501	.sina.cn	/	2029-0...	17			
lxlrtp	1554343419	.sina.com.cn	/	2019-0...	17			

## Cookie的应用

现在回到我们最开始的话题，有了Cookie，服务器就有了“记忆能力”，能够保存“状态”，那么应该如何使用Cookie呢？

Cookie最基本的一个用途就是**身份识别**，保存用户的登录信息，实现会话事务。

比如，你用账号和密码登录某电商，登录成功后网站服务器就会发给浏览器一个Cookie，内容大概是“name=yourid”，这样就成功地把身份标签贴在了你身上。

之后你在网站里随便访问哪件商品的页面，浏览器都会自动把身份Cookie发给服务器，所以服务器总会知道你的身份，一方面免去了重复登录的麻烦，另一方面也能够自动记录你的浏览记录和购物下单（在后台数据库或者也用Cookie），实现了“状态保持”。

Cookie的另一个常见用途是**广告跟踪**。

你上网的时候肯定看过很多的广告图片，这些图片背后都是广告商网站（例如Google），它会“偷偷地”给你贴上Cookie小纸条，这样你上其他的网站，别的广告就能用Cookie读出你的身份，然后做行为分析，再推给你广告。

这种Cookie不是由访问的主站存储的，所以又叫“第三方Cookie”（third-party cookie）。如果广告商势

力很大，广告到处都是，那么就比较“恐怖”了，无论你走到哪里它都会通过Cookie认出你来，实现广告“精准打击”。

为了防止滥用Cookie搜集用户隐私，互联网组织相继提出了DNT（Do Not Track）和P3P（Platform for Privacy Preferences Project），但实际作用不大。

## 小结

今天我们学习了HTTP里的Cookie知识。虽然现在已经出现了多种Local Web Storage技术，能够比Cookie存储更多的数据，但Cookie仍然是最通用、兼容性最强的客户端数据存储手段。

简单小结一下今天的内容：

1. Cookie是服务器委托浏览器存储的一些数据，让服务器有了“记忆能力”；
2. 响应报文使用Set-Cookie字段发送“key=value”形式的Cookie值；
3. 请求报文里用Cookie字段发送多个Cookie值；
4. 为了保护Cookie，还要给它设置有效期、作用域等属性，常用的有Max-Age、Expires、Domain、HttpOnly等；
5. Cookie最基本的用途是身份识别，实现有状态的会话事务。

还要提醒你一点，因为Cookie并不属于HTTP标准（RFC6265，而不是RFC2616/7230），所以语法上与其他字段不太一致，使用的分隔符是“;”，与Accept等字段的“,”不同，小心不要弄错了。

## 课下作业

1. 如果Cookie的Max-Age属性设置为0，会有什么效果呢？
2. Cookie的好处已经很清楚了，你觉得它有什么缺点呢？

欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。



## == 课外小贴士 ==

- 01 Cookie 这个词来源于计算机编程里的术语“Magic Cookie”，意思是不透明的数据，并不是“小甜饼”的含义（虽然字面意如此）。
- 02 早期 Cookie 直接就是磁盘上的一些小文本文件，现在基本上都是以数据库记录的形式存放的（通常使用的是 Sqlite）。浏览器对 Cookie 的数量和大小也都有限制，不允许无限存储，一般总大小不能超过 4K。
- 03 如果不指定 Expires 或 Max-Age 属性，那么 Cookie 仅在浏览器运行时有效，一旦浏览器关闭就会失效，这被称为会话 Cookie (session cookie) 或内存 Cookie (in-memory cookie)，在 Chrome 里过期时间会显示为“Session”或“N/A”。
- 04 历史上还有“Set-Cookie2”和“Cookie2”这样的字段，但现在已经不再使用。

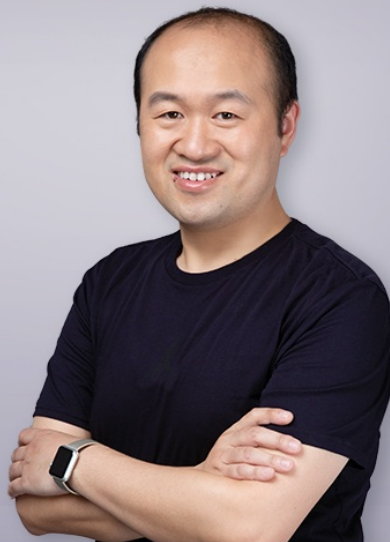
# 透视 HTTP 协议

深入理解 HTTP 协议本质与应用

罗剑锋

奇虎360技术专家

Nginx/OpenResty 开源项目贡献者



新版升级：点击「🔔 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

## 精选留言：

• 徐海浪 2019-07-10 08:59:13

1. 如果 Cookie 的 Max-Age 属性设置为 0，会有什么效果呢？

设置为0，服务器0秒就让Cookie失效，即立即失效，服务器不存Cookie。

2. Cookie 的好处已经很清楚了，你觉得它有什么缺点呢？

好处：方便了市民

缺点：方便了黑客:) [2赞]

作者回复2019-07-10 12:24:34

√

• 业余草 2019-07-10 14:47:17

属性“HttpOnly”、“Secure”、“SameSite”很少见，老师可以给几个配套例子，后面答疑篇，可以来个攻防实战！

• chengzise 2019-07-10 14:15:37

老师好，我理解的广告跟踪那个原理是：用户访问A网站，A会给用户设置cookies（T），用户再次访问网站B时，浏览器会带上cookies（T），B网站就能够识别到被A标记的用户。这里A给用户设置的cookies里面是不是把domain设置为B网站？不然凭什么访问B网站的时候，浏览器会带上A设置的cookies。

• 响雨 2019-07-10 11:55:35

Max-Age=0，cookie就会马上失效了。cookie的明文存储肯定不安全的。

Flask开发中，Flask-session提供了将session\_id存储在cookie中的方法，这样cookie就不需要存储敏感数据，通过session\_id从后端获取session数据，然后session中存放用户的数据。

作者回复2019-07-10 12:19:57

√

• 小美 2019-07-10 11:05:59

比如域名A，是不是能获取到用户所有的cookie（包括B域名）这样不是很不安全



作者回复2019-07-10 12:28:09

这个就是跨站的问题，通常cookie都会设置作用域，浏览器会保证只发给对应的网站，不会那么简单地泄漏。

- 我行我素 2019-07-10 10:59:03

- 1.立即失效，即每次访问都会重新设置cookie
- 2.安全性问题，还有就是隐私的烦恼；

作者回复2019-07-10 12:20:23

√

- W.LI- 2019-07-10 09:24:21

老师好有个问题!不加SameSite的人话跨站请求时会把cookie信息带过去。除了cookie信息还有哪些在跨站时会被浏览器自动带过去啊?

作者回复2019-07-10 12:32:16

网络攻击方面研究的不深，无法给你完整的解答。

最常见的就是cookie了。

- W.LI- 2019-07-10 09:21:02

挺好的，cookie存放在流浪器端，不太安全。我是用java的现在用户数据大多存储在服务器缓存里了。一些不重要的数据可以放在cookie，cookie安全的那些参数，有利有弊吧。设置了设置了domain就没法跨越了，有些场景需要跨越的就不能设了，path太大全局可见不好，太小上级路径取不到就丢了。HttpOnly和SameSite过安全测评的时候都会要求加上。加了肯定会有不方便的地方，前端不怎么会。涨姿势了谢谢老师。

作者回复2019-07-10 12:24:24

不客气。

- cp3yqng 2019-07-10 08:17:25

域名+路径的方式存储cookie，感觉像只有一台业务服务器，那后台如何过分布式系统呢，用户中心是一个系统，核心业务是其他的系统，这里cookie肯定要共享，应该有一级域名和二级域名等等的概念吧，麻烦老师在解释解释。我本人是做移动端开发的，都是自己把token写在网络底层的请求头中，其实核心思想是一样的，但是缺点就是所有的域名里面都带token，这样也不好，好像还有优化的空间。

作者回复2019-07-10 08:58:25

只要cookie设置了domain和path属性，浏览器在访问uri时就会根据这两个属性有选择地发送cookie。

需要根据自己的业务需求，恰当设置cookie的作用域，太大太小都不好。

- 业余爱好者 2019-07-10 07:22:54

- 1.Max-Age: 是永久有效的意思。
- 2.cookie在浏览器端禁用，还有就是安全性，因为在本地是明文存储的

作者回复2019-07-10 08:56:31

1不对，max-age=0，就是立即过期，不允许缓存，只能在浏览器运行期间有效。

- 苦行僧 2019-07-10 06:21:57

cookie类似缓存，只不过是存储在客户端，敏感信息容易泄露，所以要慎用存储敏感信息

作者回复2019-07-10 08:55:09

对

- 大小兵 2019-07-10 01:17:35

要是能把session和token也说一下就好了

作者回复2019-07-10 07:39:42

这个不在http范围之内，而且篇幅有限，还望见谅。

- レイン小雨 2019-07-10 01:04:31

棒！

作者回复2019-07-10 07:39:08

thanks。