Atitit session机制的实现web

目录

[1. Sessionid的发送 1](#_Toc23690)

[1.1. session大部分情况下基于cookie实现。 1](#_Toc11445)

[1.2. 基于url的session 1](#_Toc31283)

[1.3. http head的session 1](#_Toc8424)

[1.4. 表单保存法隐藏域 1](#_Toc25518)

[2. Session的服务的实现 2](#_Toc2396)

[3. Sessionid的客户端保存 2](#_Toc22081)

# Sessionid的发送

## session大部分情况下基于cookie实现。

面试官说的没错，session大部分情况下基于cookie实现。

客户端给服务端维持联系靠cookie，服务端生成一个shit用cookie给客户端拿着。并不矛盾。另外还有

## 基于url的session

第二种：通过重写URL。如果浏览器不支持cookies，可以自己编程使用URL重写的方式实现session（访问页面的时候在地址栏里面，URL后会跟上sessionID，效果见展示图2）。

## http head的session

## 表单保存法隐藏域

可以，在每条url中都加上session字段，这种做法现在在很多老式WAP网站上依然能够见到，因为当时的具有WAP浏览器的手机因为内存小的可怜无法存储cookie。还有一种办法就是每个页面都加一个hidden隐藏域，value写上session，总之方法很多，灵活变通。

只要传递sessoinid即可。。

# Session的服务的实现

器。服务器还是会发Set-Cookie的Header，只是客户端忽略，那每次访问服务器都是一个新的Session，服务器没法和旧的Session关联上。

# Sessionid的客户端保存

****javascript:alert (document. cookie)****

ETag 已经是最要脸的追踪手段了好吗。。。  
试试 authentication cookie

<iframe src**=**"http://session:<cookie>@zhihu.com/">

最简单的办法，重新定向  
登陆之后网址变成[http://www.xxx.com/](https://link.zhihu.com/?target=http://www.xxx.com/" \t "_blank)？session=xxxxxxxxxxxxxxxx  
还有一个就是用户再次点击的问题，以前的方法是查找所有a href，form修改，这个太麻烦了。  
其实用referer就好了，在nginx里判断referer有没有session，然后处理下，应用代码可以不做任何修改。  
还有这种方式容易泄露，可以跟ip，agent绑定，即session记录操作的ip，如果ip，浏览器变了就重新生成session

|  |  |
| --- | --- |
| ****int maxAge**** | ****该Cookie失效的时间，单位秒。如果为正数，则该Cookie在maxAge秒之后失效。如果为负数，该Cookie为临时Cookie，关闭浏览器即失效，浏览器也不会以任何形式保存该Cookie。如果为0，表示删除该Cookie。默认为–1**** |

|  |  |
| --- | --- |
| String domain | 可以访问该Cookie的域名。如果设置为“.google.com”，则所有以“google.com”结尾的域名都可以访问该Cookie。注意第一个字符必须为“.” |