关于新手必须要理解的几个名词，cookie、session和token

标签：for 重写 访问 server 失败 内存溢出 打开 存在 好文

以下要说的，虽然不是开发过程中必须会遇到的，但却是进阶之路上必须要掌握的，一些涉及到状态管理与安全的应用当中尤为重要。

我之前虽略有学习，但也是东拼西凑临时看的一点皮毛，所以在这个假期利用一点时间，整合一篇博文出来，方便以后自己温故，当然能对新入行的朋友有些许帮助，那是最好的了。

好，废话结束，下面咱们开始。

我们首先分别说一下cookie、session和token各是什么，以及使用方法，然后再说一下他们的关系和区别。

一、Cookie

如果是从来没接触过cookie的童鞋，可以先去http://www.runoob.com/js/js-cookies.html文档看一下，很通俗易懂的，里面介绍了cookie的基础属性以及使用cookie时的三种封装，十分钟搞定了再回来。

由于HTTP是一种无状态的协议，服务器单从网络连接上无从知道客户身份。怎么办呢？就给客户端们颁发一个通行证吧，每人一个，无论谁访问都必须携带自己通行证。这样服务器就能从通行证上确认客户身份了。这就是Cookie的工作原理。

Cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用response向客户端浏览器颁发一个Cookie。客户端浏览器会把Cookie保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该Cookie一同提交给服务器。服务器检查该Cookie，以此来辨认用户状态。服务器还可以根据需要修改Cookie的内容。

除了name（名）和value（值），cookie还有以下一些可选属性，用来控制cookie的有效期，作用域，安全性等：

expires属性

指定了cookie的生存期，默认情况下cookie是暂时存在的，他们存储的值只在浏览器会话期间存在，当用户退出浏览器后这些值也会丢失，如果想让cookie存在一段时间，就要为expires属性设置为未来的一个用毫秒数表示的过期日期或时间点，expires默认为设置的expires的当前时间。现在已经被max-age属性所取代，max-age用秒来设置cookie的生存期。

如果max-age属性为正数，则表示该cookie会在max-age秒之后自动失效。浏览器会将max-age为正数的cookie持久化，即写到对应的cookie文件中。无论客户关闭了浏览器还是电脑，只要还在max-age秒之前，登录网站时该cookie仍然有效。

如果max-age为负数，则表示该cookie仅在本浏览器窗口以及本窗口打开的子窗口内有效，关闭窗口后该cookie即失效。max-age为负数的Cookie，为临时性cookie，不会被持久化，不会被写到cookie文件中。cookie信息保存在浏览器内存中，因此关闭浏览器该cookie就消失了。cookie默认的max-age值为-1。

？ 如果max-age为0，则表示删除该cookie。cookie机制没有提供删除cookie的方法，因此通过设置该cookie即时失效实现删除cookie的效果。失效的Cookie会被浏览器从cookie文件或者内存中删除。

如果不设置expires或者max-age这个cookie默认是Session的，也就是关闭浏览器该cookie就消失了。

这里要说明一下：Session的cookie在ie6下，如果用户实在网页上跳转打开页面或新开窗口（包括target="_blank"，鼠标右键新开窗口），都是在同一个Session内。如果用户新开浏览器程序或者说是进程再打开当前的页面就不是同一个Session。其他浏览器只要你Session存在，还是同一个Session，cookie还能共享。

domain属性

domain属性可以使多个web服务器共享cookie。domain属性的默认值是创建cookie的网页所在服务器的主机名。不能将一个cookie的域设置成服务器所在的域之外的域。
例如让位于a.sodao.com的服务器能够读取b.sodao.com设置的cookie值。如果b.sodao.com的页面创建的cookie把它的path属性设置为"/"，把domain属性设置成".sodao.com"，那么所有位于b.sodao.com的网页和所有位于a.sodao.com的网页，以及位于sodao.com域的其他服务器上的网页都可以访问这个cookie。

path属性

它指定与cookie关联在一起的网页。在默认的情况下cookie会与创建它的网页，该网页处于同一目录下的网页以及与这个网页所在目录下的子目录下的网页关联。

secure属性

它是一个布尔值，指定在网络上如何传输cookie，默认是不安全的，通过一个普通的http连接传输；

HttpOnly属性

HttpOnly 属性限制了 cookie 对 HTTP 请求的作用范围。特别的，该属性指示用户代理忽略那些通过"非 HTTP" 方式对 cookie 的访问（比如浏览器暴露给js的接口）。注意 HttpOnly 属性和 Secure 属性相互独立：一个 cookie 既可以是 HttpOnly 的也可以有 Secure 属性。
在前段时间的项目中我就用js去读取一个cookie，结果怎么都取不到这个值，最后查证这个cookie是httpOnly的，花了近2个小时，悲剧了。

cookie的传输

浏览器将cookie信息以`name-value`对的形式存储于本地，每当请求新文档时，浏览器将发送Cookie，目的是让Server可以通过HTTP请求追踪客户。所以从WEB性能的角度来说我们要尽量的减小cookie，以达到传输性能的最大化。

cookie的编码和解码

由于cookie的名/值中的值不允许包含分号，逗号和空格符，为了最大化用户代理和服务器的兼容性，任何被存储为 cookie 值的数据都应该被编码，例如用我们前端熟知的js全局函数encodeURIComponent编码和decodeURIComponent解码。

cookie作为客户端存储

前面说了每当请求新文档时，浏览器将发送Cookie到服务器，导致WEB性能下降。所以不建议将cookie作为客户端存储一种实现方案，替代方案参见：JavaScript本地存储实践（html5的localStorage和ie的userData）等。

cookie的同名问题

同名的 cookie，不同的 domain 或不同的 path，属不同的 cookie；同名的 cookie，相同的 domain 且相同的 path，不同的 expires，属同一个 cookie。

cookie的web安全方面

Cookie 的HttpOnly 属性是Cookie 的扩展功能，它使JavaScript 脚本无法获得Cookie。其主要目的为防止跨站脚本攻击（Cross-sitescripting，XSS）对Cookie 的信息窃取。发送指定HttpOnly 属性的Cookie 的方法

Set-Cookie: name=value; HttpOnly

Cookie 的secure 属性用于限制Web 页面仅在HTTPS 安全连接时，才可以发送Cookie。

基本上可以用到的cookie相关的知识就是以上这些，接下来我们说一下session。