【前端面试】session、cookie、token的区...

前言

session、cookie、token 这几个概念在面试中可以说是经常出现的，因为通过这个概念基本上可以了解到你对网络请求或者权限管理这一块是否有过了解和实际的应用。很多小伙伴可能喜欢死记硬背，以此来应付面试！

但是这几个东西不光面试中经常问到，它们在我们的实际项目中也是应用广泛的。所以我们有必要了解它们，并且学会如何使用它们。

今天就来理一理这三者之间的关系！

1.为什么会有它们?

我们都知道 HTTP 协议是无状态的，所谓的无状态就是客户端每次想要与服务端通信，都必须重新与服务端链接，意味着请求一次客户端和服务端就连接一次，下一次请求与上一次请求是没有关系的。

这种无状态的方式就会存在一个问题：如何判断两次请求的是同一个人？就好比用户在页面 A 发起请求获取个人信息，然后在另一个页面同样发起请求获取个人信息，我们如何确定这俩个请求是同一个人发的呢?

为了解决这种问题，我们就迫切需要一种方式知道发起请求的客户端是谁？此时，cookie、token、session 就出现了，它们就可以解决客户端标识的问题，在扩大一点就是解决权限问题。

它们就好比让每个客户端或者说登录用户有了自己的身份证，我们可以通过这个身份证确定发请求的是谁！

2.什么是 cookie?

cookie 是保存在客户端或者说浏览器中的一小块数据，大小限制大致在 4KB 左右，在以前很多开发人员通常用 cookie 来存储各种数据，后来随着更多浏览器存储方案的出现，cookie 存储数据这种方式逐渐被取代，主要原因有如下：

- cookie 有存储大小限制，4KB 左右。
- 浏览器每次请求会携带 cookie 在请求头中。

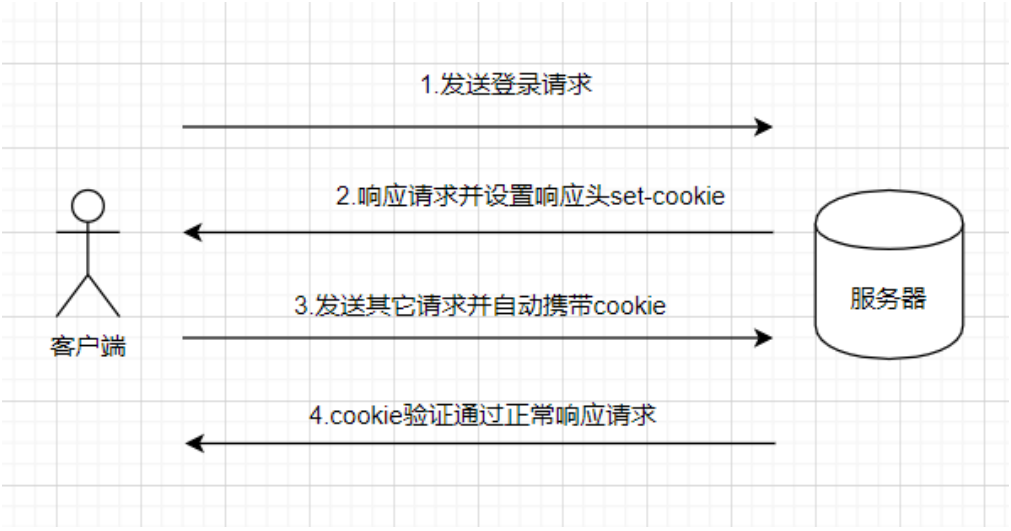
- 字符编码为 Unicode，不支持直接存储中文。
- 数据可以被轻易查看。

cookie 主要有以下属性：

属性名称	属性含义
name	cookie 的名称
value	cookie 的值
comment	cookie 的描述信息
domain	可以访问该 cookie 的域名
expires	cookie 的过期时间，具体某一时间
maxAge	cookie 的过期时间，比如多少秒后 cookie 过期。
path	cookie 的使用路径，
secure	cookie 是否使用安全协议传输，比如 SSL 等
version	cookie 使用的版本号
isHttpOnly	指定该 Cookie 无法通过 JavaScript 脚本拿到，比如 Document.cookie 属性、XMLHttpRequest 对象和 Request API 都拿不到该属性。这样就防止了该 Cookie 被脚本读到，只有浏览器发出 HTTP 请求时，才会带上该 Cookie。

我们介绍了 cookie，那么我们是如何通过 cookie 来实现用户确定或者权限的确定呢?

我们就以一个普通网站的用户登录操作以及后续操作为例，主要过程可以简单用下图表示：



从上图中可以看到使用 cookie 进行用户确认流程是比较简单的，大致分为以下几步：

1. 客户端发送请求到服务端（比如登录请求）。
2. 服务端收到请求后生成一个 session 会话。

3. 服务端响应客户端，并在响应头中设置 Set-Cookie。Set-Cookie 里面包含了 sessionId，它的格式如下：Set-Cookie: value[; expires=date][; domain=domain][; path=path][; secure]。其中 sessionId 就是用来标识客户端的，类似于去饭店里面，服务员给你一个号牌，后续上菜通过这个号牌来判断上菜到哪里。
4. 客户端收到该请求后，如果服务器给了 Set-Cookie，那么下次浏览器就会在请求头中自动携带 cookie。
5. 客户端发送其它请求，自动携带了 cookie，cookie 中携带有用户信息等。
6. 服务端接收到请求，验证 cookie 信息，比如通过 sessionId 来判断是否存在会话，存在则正常响应。

cookie 主要有以下特点：

- cookie 存储在客户端
- cookie 不可跨域，但是在如果设置了 domain，那么它们是可以在一级域名和二级域名之间共享的。

3.什么是 session?

在上一节中，我们通过 Cookie 来实现了用户权限的确认，在其中我们提到了一个词：session。顾名思义它就是会话的意思，session 主要由服务端创建，主要作用就是保存 sessionId，用户与服务端之间的权限确认主要就是通过这个 sessionId。

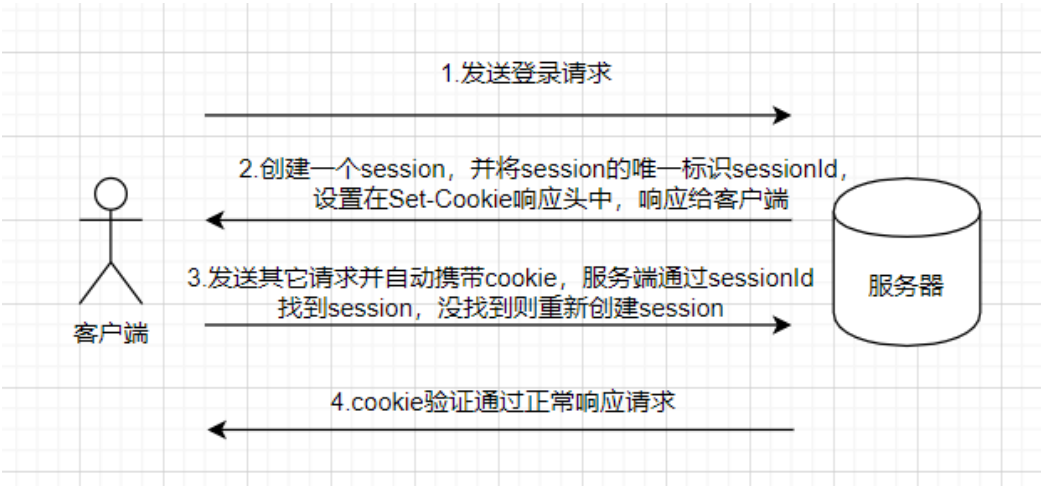
简单描述下 session：

> session 由服务端创建，当一个请求发送到服务端时，服务器会检索该请求里面有没有包含 sessionId 标识，如果包含了 sessionId，则代表服务端已经和客户端创建过 session，然后就通过这个 sessionId 去查找真正的 session，如果没找到，则为客户端创建一个新的 session，并生成一个新的 sessionId 与 session 对应，然后在响应的时候将 sessionId 给客户端，通常是存储在 cookie 中。如果在请求中找到了真正的 session，验证通过，正常处理该请求。

总之每一个客户端与服务端连接，服务端都会为该客户端创建一个 session，并将 session 的唯一标识 sessionId 通过设置 Set-Cookie 头的方式响应给客户端，客户端将 sessionId 存到 cookie 中。

通常情况下，我们 cookie 和 session 都是结合着来用，当然你也可以单独只使用 cookie 或者单独只使用 session，这里我们就将 cookie 和 session 结合着来用。

我们可以在修改一下整个请求过程图，如下所示：



4.cookie 和 session 的区别?

前面两节我们介绍了 cookie 和 session，它们两者之间主要是通过 sessionId 关联起来的，所以我们总结出：sessionId 是 cookie 和 session 之间的桥梁。我们日常的系统中如果在鉴权方面如果使用的是 cookie 方式，那么大部分的原理就和我们前面说的一样。

或者我们可以换个说法，session 是基于 cookie 实现的，它们两个主要有以下特点：

- session 比 cookie 更加安全，因为它是存在服务端的，cookie 是存在客户端的。
- cookie 只支持存储字符串数据，session 可以存储任意数据。
- cookie 的有效期可以设置较长时间，session 有效期都比较短。
- session 存储空间很大，cookie 有限制。

系统想要实现鉴权，可以单独使用 cookie，也可以单独使用 session，但是建议结合两者使用。

5.token 是什么?

前面我们说的 sessionId 可以叫做令牌，令牌顾名思义就是确认身份的意思，服务端可以通过令牌来确认身份。

cookie+session 是实现认证的一种非常好的方式，但是凡事都有两面性，它们实现的认证主要有以下缺点：

- 增加请求体积，浪费性能，因为每次请求都会携带 cookie。
- 增加服务端资源消耗，因为每个客户端连接进来都需要生成 session，会占用服务端资源的。
- 容易遭受 CSRF 攻击，即跨站域请求伪造。

那么为了避免这些缺点，token 方式的鉴权出现了，它可以说是一个民间的认证方式，但是不得不说它带来了非常多的好处。

token 的组成：

> token 其实就是一串字符串而已，只不过它是被加密后的字符串，它通常使用 uid(用户唯一标识)、时间戳、签名以及一些其它参数加密而成。我们将 token 进行解密就可以拿到诸如 uid 这类的信息，然后通过 uid 来进行接下来的鉴权操作。

token 是如何生成的：

前面我们说 cookie 是服务端设置了 set-cookie 响应头之后，浏览器会自动保存 cookie，然后下一次发送请求的时候会自动把 cookie 携带上。但是我们说 cookie 算是一种民间的实现方式，所以说浏览器自然不会对它进行成么处理。token 主要是由服务器生成，然后返回给客户端，客户端手动把 token 存下来，比如利用 localstorage 或者直接存到 cookie 当中也行。

token 认证流程：

1. 客户端发起登录请求，比如用户输入用户名和密码后登录。
2. 服务端校验用户名和密码后，将用户 id 和一些其它信息进行加密，生成 token。
3. 服务端将 token 响应给客户端。
4. 客户端收到响应后将 token 存储下来。
5. 下一次发送请求后需要将 token 携带上，比如放在请求头中或者其它地方。
6. 服务端 token 后校验，校验通过则正常返回数据。

用图表示大致如下：

总结

虽然前面解释 cookie、session、token 用了不少口舌，但是归根结底啊，它们的目的都是一样的：**鉴权和认证。**

鉴权认证方式	特点	优点	缺点
cookie	1.存储在客户端。 2.请求自动携带 cookie。 3.存储大小 4KB。	1.兼容性好，因为是比较老的技术。 2.很容易实现，因为 cookie 会自动携带和存储。	1.需要单独解决跨域携带问题，比如多台服务器如何共享 cookie。 2.会遭受 CSRF 攻击。 3.存储在客户端，不够安全。
session	1.存储在服务端。 2.存储大小无限制。	1.查询速度快，因为是个会话，相当于是在内存中操作。 2.结合 cookie 后很容易实现鉴权。 3.安全，因为存储在服务端。	1.耗费服务器资源，因为每个客户端都会创建 session。 2.占据存储空间，session 相当于存储了一个完整的用户信息。
token	1.体积很小。 2.自由操作存储在哪里。	1.安全，因为 token 一般只有用户 id，就算被截取了也没什么用。 2.无需消耗服务器内存资源，它相当于只存了用户 id，session 相当于存储了用户的所有信息。 3.跨域处理较为方便，比如多台服务器之间可以共用一个 token。	1.查询速度慢，因为 token 只存了用户 id，每次需要去查询数据库。

总结下来就是：session 是空间换时间，token 是时间换空间。