简单聊一聊Cookie、Session、Token、JWT的区别和作用

> 链接：https://segmentfault.com/a/1190000021810849

要解释这方面的知识，先得简单说一下什么是认证（Authentication），通俗地讲就是验证当前用户的身份，证明“你是你自己”（比如：你每天上下班打卡，都需要通过指纹打卡，当你的指纹和系统里录入的指纹相匹配时，就打卡成功），比如互联网中的认证：用户密码登录，邮箱发送登录链接，手机号接受验证码，只要你能收到邮件或者验证码就默认你是主人。

再谈谈什么是授权（Authorization）： 用户授予第三方应用访问该用户某些资源的权限就是授权。打个比方：你在安装手机应用的时候，APP 会询问是否允许授予权限（访问相册、地理位置等权限），你在访问微信小程序时，当登录时，小程序会询问是否允许授予权限（获取昵称、头像、地区、性别等个人信息）等

一般实现实现授权的方式有：cookie、session、token。 再聊聊什么是凭证Credentials

实现认证和授权的前提是需要一种媒介（证书） 来标记访问者的身份，打个比方，在互联网应用中，一般网站（如掘金）会有两种模式，游客模式和登录模式。游客模式下，可以正常浏览网站上面的文章，一旦想要点赞/收藏/分享文章，就需要登录或者注册账号。当用户登录成功后，服务器会给该用户使用的浏览器颁发一个令牌（token），这个令牌用来表明你的身份，每次浏览器发送请求时会带上这个令牌，就可以使用游客模式下无法使用的功能。

什么是cookie?

HTTP 是无状态的协议（对于事务处理没有记忆能力，每次客户端和服务端会话完成时，服务端不会保存任何会话信息）：每个请求都是完全独立的，服务端无法确认当前访问者的身份信息，无法分辨上一次的请求发送者和这一次的发送者是不是同一个人。所以服务器与浏览器为了进行会话跟踪（知道是谁在访问我），就必须主动的去维护一个状态，这个状态用于告知服务端前后两个请求是否来自同一浏览器。而这个状态需要通过 cookie 或者 session 去实现。cookie 存储在客户端： cookie 是服务器发送到用户浏览器并保存在本地的一小块数据，它会在浏览器下次向同一服务器再发起请求时被携带并发送到服务器上。

cookie 是不可跨域的： 每个 cookie 都会绑定单一的域名，无法在别的域名下获取使用，一级域名和二级域名之间是允许共享使用的（靠的是 domain）。

cookie 重要的属性属性说明name=value键值对，设置 Cookie 的名称及相对应的值，都必须是字符串类型。

如果值为 Unicode 字符，需要为字符编码。 如果值为二进制数据，则需要使用 BASE64 编码。domain指定 cookie 所属域名，默认是当前域名path指定 cookie 在哪个路径（路由）下生效，默认是 '/'。如果设置为 /abc，则只有 /abc 下的路由可以访问到该 cookie，如：/abc/read。maxAgecookie 失效的时间，单位秒。如果为整数，则该 cookie 在 maxAge 秒后失效。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。

比 expires 好用。expires过期时间，在设置的某个时间点后该 cookie 就会失效。一般浏览器的 cookie 都是默认储存的，当关闭浏览器结束这个会话的时候，这个 cookie 也就会被删除secure该 cookie 是否仅被使用安全协议传输。安全协议有 HTTPS，SSL等，在网络上传输数据之前先将数据加密。默认为false。当 secure 值为 true 时，cookie 在 HTTP 中是无效，在 HTTPS 中才有效。httpOnly如果给某个 cookie 设置了 httpOnly 属性，则无法通过 JS 脚本 读取到该 cookie 的信息，但还是能通过 Application 中手动修改 cookie，所以只是在一定程度上可以防止 XSS 攻击，不是绝对的安全。

什么是 Session?

session 是另一种记录服务器和客户端会话状态的机制

注意：session 是基于 cookie 实现的，session 存储在服务器端，sessionId 会被存储到客户端的cookie 中

session 认证流程：

1:用户第一次请求服务器的时候，服务器根据用户提交的相关信息，创建对应的 Session

2:请求返回时将此 Session 的唯一标识信息 SessionID 返回给浏览器

3:浏览器接收到服务器返回的 SessionID 信息后，会将此信息存入到 Cookie 中，同时 Cookie 记录此 SessionID 属于哪个域名

4:当用户第二次访问服务器的时候，请求会自动判断此域名下是否存在 Cookie 信息，如果存在自动将 Cookie 信息也发送给服务端，服务端会从 Cookie 中获取 SessionID，再根据 SessionID 查找对应的 Session 信息，如果没有找到说明用户没有登录或者登录失效，如果找到 Session 证明用户已经登录可执行后面操作。 根据以上流程可知，SessionID 是连接 Cookie 和 Session 的一道桥梁，大部分系统也是根据此原理来验证用户登录状态。

Cookie 和 Session 的区别

安全性： Session 比 Cookie 安全，Session 是存储在服务器端的，Cookie 是存储在客户端的。

存取值的类型不同：Cookie 只支持存字符串数据，想要设置其他类型的数据，需要将其转换成字符串，Session 可以存任意数据类型。

有效期不同： Cookie 可设置为长时间保持，比如我们经常使用的默认登录功能，Session 一般失效时间较短，客户端关闭（默认情况下）或者 Session 超时都会失效。

存储大小不同： 单个 Cookie 保存的数据不能超过 4K，Session 可存储数据远高于 Cookie，但是当访问量过多，会占用过多的服务器资源。

什么是 Token（令牌）

Acesss Token全称，访问资源接口（API）时所需要的资源凭证，简单 token 的组成： uid(用户唯一的身份标识)、time(当前时间的时间戳)、sign（签名，token 的前几位以哈希算法压缩成的一定长度的十六进制字符串）。

Token的特点：

服务端无状态化 可扩展性好 支持移动端设备 安全 支持跨程序调用

token 的身份验证流程：

1.客户端使用用户名跟密码请求登录 2.服务端收到请求，去验证用户名与密码 3.验证成功后，服务端会签发一个 token 并把这个 token 发送给客户端 4.客户端收到 token 以后，会把它存储起来，比如放在 cookie 里或者 localStorage 里 客户端每次向服务端请求资源的时候需要带着服务端签发的 token 5.服务端收到请求，然后去验证客户端请求里面带着的 token ，如果验证成功，就向客户端返回请求的数据

注意：每一次请求都需要携带 token，需要把 token 放到 HTTP 的 Header 里 基于 token 的用户认证是一种服务端无状态的认证方式，服务端不用存放 token 数据。用解析 token 的计算时间换取 session 的存储空间，从而减轻服务器的压力，减少频繁的查询数据库 token 完全由应用管理，所以它可以避开同源策略。

再介绍一种token，refresh token。 refresh token 是专用于刷新 access token 的 token。如果没有 refresh token，也可以刷新 access token，但每次刷新都要用户输入登录用户名与密码，会很麻烦。有了 refresh token，可以减少这个麻烦，客户端直接用 refresh token 去更新 access token，无需用户进行额外的操作。

注意：Access Token 的有效期比较短，当 Acesss Token 由于过期而失效时，使用 Refresh Token 就可以获取到新的 Token，如果 Refresh Token 也失效了，用户就只能重新登录了。

Refresh Token 及过期时间是存储在服务器的数据库中，只有在申请新的 Acesss Token 时才会验证，不会对业务接口响应时间造成影响，也不需要向 Session 一样一直保持在内存中以应对大量的请求。

Token和Ssession的区别

Session 是一种记录服务器和客户端会话状态的机制，使服务端有状态化，可以记录会话信息。而 Token 是令牌，访问资源接口（API）时所需要的资源凭证。Token 使服务端无状态化，不会存储会话信息。

Session 和 Token 并不矛盾，作为身份认证 Token 安全性比 Session 好，因为每一个请求都有签名还能防止监听以及重放攻击，而 Session 就必须依赖链路层来保障通讯安全了。如果你需要实现有状态的会话，仍然可以增加 Session 来在服务器端保存一些状态。

所谓 Session 认证只是简单的把 User 信息存储到 Session 里，因为 SessionID 的不可预测性，暂且认为是安全的。而 Token ，如果指的是 OAuth Token 或类似的机制的话，提供的是 认证 和 授权 ，认证是针对用户，授权是针对 App 。其目的是让某 App 有权利访问某用户的信息。这里的 Token 是唯一的。不可以转移到其它 App上，也不可以转到其它用户上。Session 只提供一种简单的认证，即只要有此 SessionID ，即认为有此 User 的全部权利。是需要严格保密的，这个数据应该只保存在站方，不应该共享给其它网站或者第三方 App。所以简单来说：如果你的用户数据可能需要和第三方共享，或者允许第三方调用 API 接口，用 Token 。如果永远只是自己的网站，自己的 App，用什么就无所谓了。

什么是 JWT?

JSON Web Token（简称 JWT）是目前最流行的跨域认证解决方案。是一种认证授权机制。

JWT 是为了在网络应用环境间传递声明而执行的一种基于 JSON 的开放标准（RFC 7519）。JWT 的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源。比如用在用户登录上。

可以使用 HMAC 算法或者是 RSA 的公/私秘钥对 JWT 进行签名。因为数字签名的存在，这些传递的信息是可信的。

阮一峰老师的 JSON Web Token 入门教程 讲的非常通俗易懂，这里就不再班门弄斧了 生成JWT，jwt.io/www.jsonwebtoken.io/

JWT的认证流程：

1.用户输入用户名/密码登录，服务端认证成功后，会返回给客户端一个 JWT

2.客户端将 token 保存到本地（通常使用 localstorage，也可以使用 cookie）

3.当用户希望访问一个受保护的路由或者资源的时候，需要请求头的 Authorization 字段中使用Bearer 模式添加 JWT，其内容看起来是下面这样 Authorization: Bearer复制代码 服务端的保护路由将会检查请求头 Authorization 中的 JWT 信息，如果合法，则允许用户的行为

JWT的使用方式 1.客户端收到服务器返回的 JWT，可以储存在 Cookie 里面，也可以储存在 localStorage 当用户希望访问一个受保护的路由或者资源的时候，可以把它放在 Cookie 里面自动发送，但是这样不能跨域，所以更好的做法是放在 HTTP 请求头信息的 Authorization 字段里，使用 Bearer 模式添加 JWT 2.跨域的时候，可以把 JWT 放在 POST 请求的数据体里。 3.通过URL

Token和JWT的区别：

相同点： 都是访问资源的令牌 都可以记录用户的信息 都是使服务端无状态化 都是只有验证成功后，客户端才能访问服务端上受保护的资源

不同点：

Token：服务端验证客户端发送过来的 Token 时，还需要查询数据库获取用户信息，然后验证 Token 是否有效。

JWT：将 Token 和 Payload 加密后存储于客户端，服务端只需要使用密钥解密进行校验（校验也是 JWT 自己实现的）即可，不需要查询或者减少查询数据库，因为 JWT 自包含了用户信息和加密的数据。

再总结一些实用session，cookie，token，Jwt需要注意的问题

使用 cookie 时需要考虑的问题： 1.因为存储在客户端，容易被客户端篡改，使用前需要验证合法性

2.不要存储敏感数据，比如用户密码，账户余额 使用 httpOnly 在一定程度上提高安全性 尽量减少 cookie 的体积，能存储的数据量不能超过 4kb

3.设置正确的 domain 和 path，减少数据传输 cookie 无法跨域

4.一个浏览器针对一个网站最多存 20 个Cookie，浏览器一般只允许存放 300 个Cookie 移动端对 cookie 的支持不是很好，而 session 需要基于 cookie 实现，所以移动端常用的是 token

使用 session 时需要考虑的问题： 1.将 session 存储在服务器里面，当用户同时在线量比较多时，这些 session 会占据较多的内存，需要在服务端定期的去清理过期的 session

2.当网站采用集群部署的时候，会遇到多台 web 服务器之间如何做 session 共享的问题。因为 session 是由单个服务器创建的，但是处理用户请求的服务器不一定是那个创建 session 的服务器，那么该服务器就无法拿到之前已经放入到 session 中的登录凭证之类的信息了。

3.多个应用要共享 session 时，除了以上问题，还会遇到跨域问题，因为不同的应用可能部署的主机不一样，需要在各个应用做好 cookie 跨域的处理。

4.sessionId 是存储在 cookie 中的，假如浏览器禁止 cookie 或不支持 cookie 怎么办？ 一般会把 sessionId 跟在 url 参数后面即重写 url，所以 session 不一定非得需要靠 cookie 实现

5.移动端对 cookie 的支持不是很好，而 session 需要基于 cookie 实现，所以移动端常用的是 token

使用 token 时需要考虑的问题

1.如果你认为用数据库来存储 token 会导致查询时间太长，可以选择放在内存当中。比如 redis 很适合你对 token 查询的需求。 token 完全由应用管理，所以它可以避开同源策略 token 可以避免 CSRF 攻击(因为不需要 cookie 了)

2.移动端对 cookie 的支持不是很好，而 session 需要基于 cookie 实现，所以移动端常用的是 token

使用 JWT 时需要考虑的问题：

1.因为 JWT 并不依赖 Cookie 的，所以你可以使用任何域名提供你的 API 服务而不需要担心跨域资源共享问题（CORS）

2.JWT 默认是不加密，但也是可以加密的。生成原始 Token 以后，可以用密钥再加密一次。 JWT 不加密的情况下，不能将秘密数据写入 JWT。 JWT 不仅可以用于认证，也可以用于交换信息。有效使用 JWT，可以降低服务器查询数据库的次数。

3.JWT 最大的优势是服务器不再需要存储 Session，使得服务器认证鉴权业务可以方便扩展。但这也是 JWT 最大的缺点：由于服务器不需要存储 Session 状态，因此使用过程中无法废弃某个 Token 或者更改 Token 的权限。也就是说一旦 JWT 签发了，到期之前就会始终有效，除非服务器部署额外的逻辑。

3.JWT 本身包含了认证信息，一旦泄露，任何人都可以获得该令牌的所有权限。为了减少盗用，JWT的有效期应该设置得比较短。对于一些比较重要的权限，使用时应该再次对用户进行认证。 JWT 适合一次性的命令认证，颁发一个有效期极短的 JWT，即使暴露了危险也很小，由于每次操作都会生成新的 JWT，因此也没必要保存 JWT，真正实现无状态。

4.为了减少盗用，JWT 不应该使用 HTTP 协议明码传输，要使用 HTTPS 协议传输。

说了很多，都是关于知识面的总结，更多的是靠自己平时开发实践过程中的总结以及学习，前端的世界，知识无穷无尽，面对特殊的疫情期间，要继续努力努力，逆战向前，迎难而上！