身份认证和授权

身份认证在很多领域得以应用，它用以验证当前用户的身份，例如：

互联网的认证

用户密码登录

手机接收验证码登录

网银的U盾

用户授予第三方应用访问某些资源的权限即为授权，实现认证和授权的前提是需要一种媒介来标记访问者的身份。实现授权的方式有cookie、session、token、OAuth。它能够存储一些身份信息或身份标记，确保当前访问资源的为合法用户。

在互联网应用中，一般网站会有两种模式：游客模式和登录模式。游客和登录后的模式所能看到的资源内容是有区别的，比如对文章的收藏和点赞通常是需要登录的。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），这个令牌用以表明用户的身份，之后每次浏览器发送请求都会带上该令牌，这样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。这样服务端自然无法确认当前访问者的身份信息，无法辨认请求是否为同一个用户。因此为了进行会话跟踪，就必须主动维护状态，用户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，它会在浏览器的下一次向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允许共享使用（设置domain）

cokie的作用

实现http连接的记忆功能

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 。

maxAge cookie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是一种记录服务器和客户端会话状态的机制

session是基于cookie实现的，session存储在服务器端，sessionid会存储到客户端的cookie中

session 认证流程：

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

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

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

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

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）

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：

客户端使用用户名跟密码请求登录

服务端收到请求，去验证用户名与密码

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
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
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
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

Token 和 Session 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。

使用 cookie 时需要考虑的问题

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

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
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。
当多个应用要共享 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。
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
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

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)
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