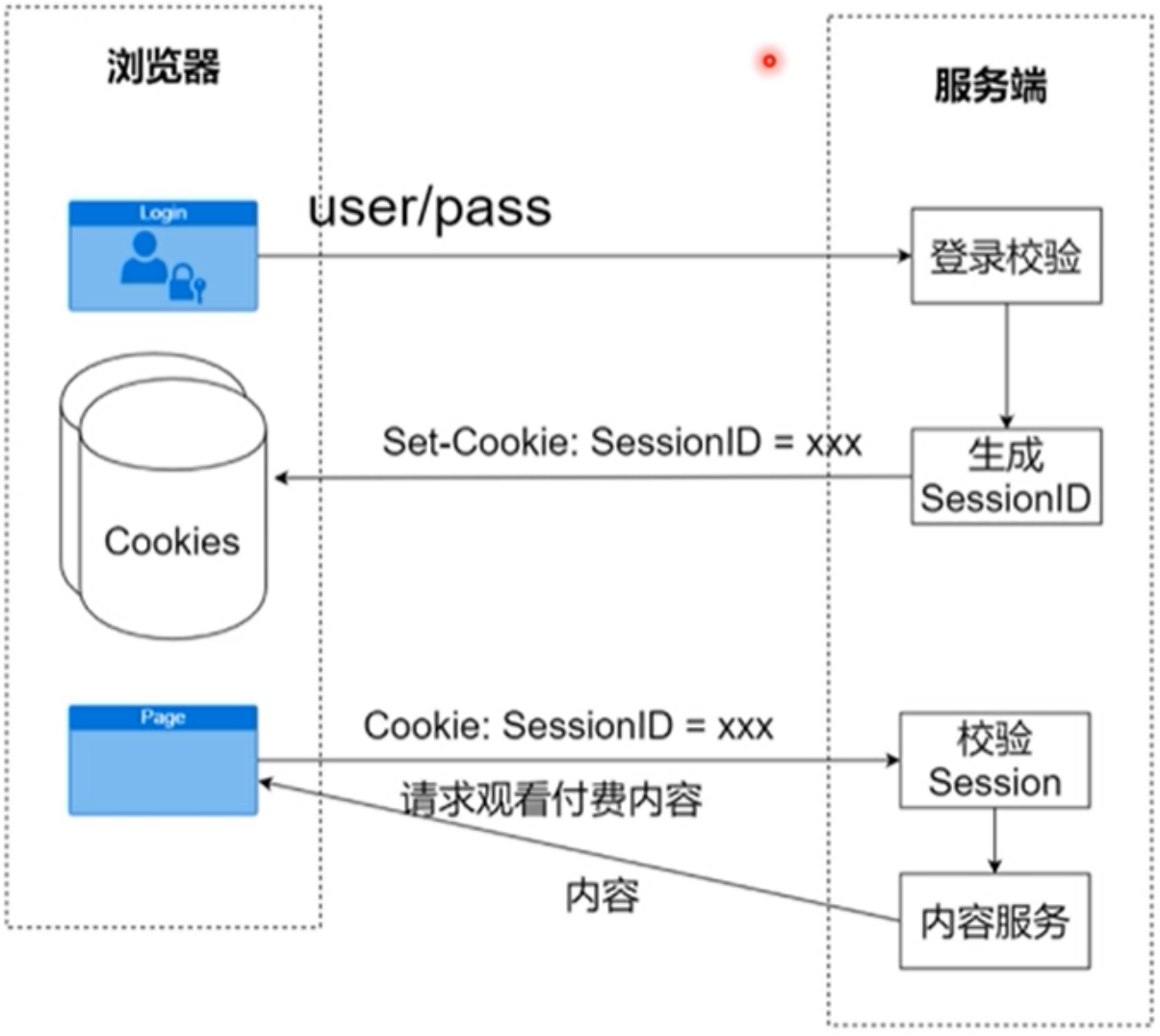
课程目标

session和storage

登录场景(理解session和cookie)



流程:浏览器输入账号密码登录之后，服务端校验账号密码的正确性，服务端会生成一个会话记录，用户id 登录时间，和数据库时间，但是这不会给到客户端，给到客户端只是一个SessionID，也就是客户端的cookie。当用户登录进来之后，来到一个需要付费的页面。发送一个一个请求到server端，server端校验的SessionId 查看客户端传过来的sessionID，比对session库（存会话的表）中是否有这么一个sessionID 是否在有效时间内 是否有这个权限。如果有 才返回相应的内容给到客户端。这就是session和cookie的一个场景。
总结：session的代表的是一次会话 服务端存储的是一条会话记录，给到客户端的是一个会话的唯一标识，给到客户端用cookie存储。

- Session代表一次会话
- SessionID是这一次会话的唯一标识
- Cookie是浏览器用于存储少量数据的存储手段

思考:

1. 为什么不让用户每次输入用户名和密码？ 因为用户麻烦又不安全。
2. 为什么浏览器是无状态的？因为这和之前我们说的restful规范。client端只做资源的表示。

cookie Session Storge和 local Storge

	Cookie	Local Storage	Session Storage
容量	4kb	10mb	5mb
作用域	同源	同源	当前网页
过期时间	手动设置	永久	当前网页关闭
位置	浏览器/服务端	浏览器	浏览器

作用域：都需要是同源。

实战的session/cookie

```
const exprss = require('express')
const app1 = exprss()
const app2 = exprss()
```

```
4
5  app1.set('etag',false)
6  app2.set('etag',false)
7  app1.get('/',(req,res)=>{
8   //原始的设置cookie的方法
9   // res.setHeader('Set-Cookie','abc=ddsaddgggggggghashgdhgsahghagdh')
10   //express中的设置cookie
11   // res.cookie('abc','dsddddddd',{maxAge: 60 * 1000, httpOnly: true})
12   res.send('ok1')
13  })
14
15  app2.get('/',(req,res)=>{
16   res.send('ok2')
17  })
18
19
20
21  app1.listen(3000,()=>{
22   console.log('进入3000服务')
23
24  })
25
26  app2.listen(3001,()=>{
27   console.log('进入3001服务')
28
29  })
```

跨域去发送cookie是一件非常困难的事情，一般是跨二级域去发cookie

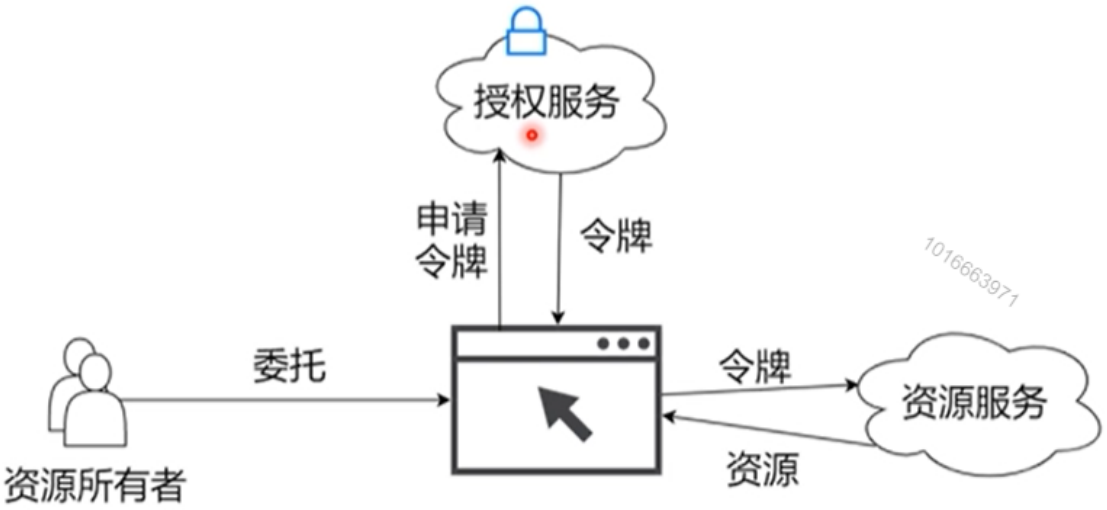
实战localstorage /session storage

观察相关使用，思考应用场景（这个就不演示代码了）

token

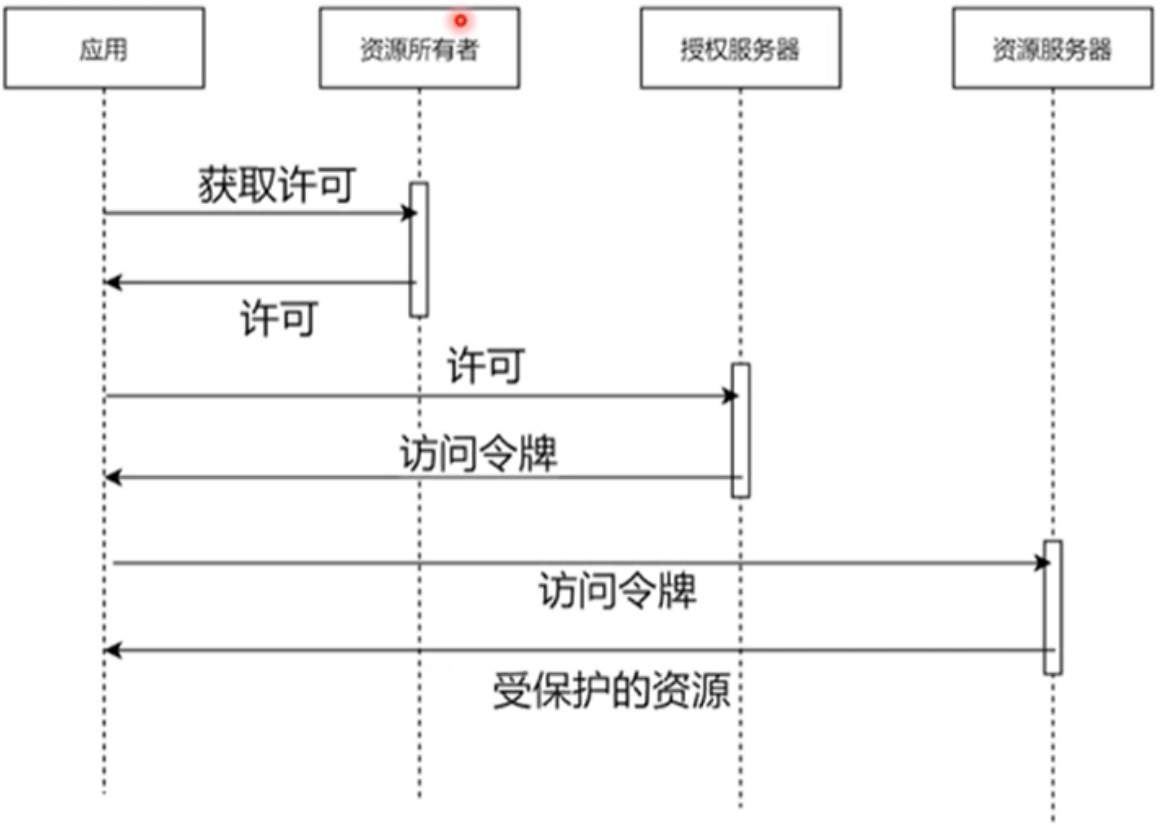
相关资料查询:http://www.ruanyifeng.com/blog/2019/04/oauth-grant-types.html

Oauth :用户在互联网获取资源的开放(委托形式) 授权标准



资源所有者，把相关资源委托给第三方授权服务，这个资源是在第三方不是在资源所有者身上。资源所有者首先要去第三方授权服务申请令牌(短信 微信授权 账号密码) 返回令牌是给到client端，这里说的web浏览器。浏览器拿到令牌然后去服务端验证，验证成功则返回相关的资源服务。这就是Oauth授权服务，总结一下就是去 授权服务拿令牌，再用令牌换服务。

思考：如果资源所有者是服务？



资源的所有者也是一个服务，应用需要去拿资源所有者的资源。比如：资源的所有者是腾讯，应用需要要去拿腾讯拿用户的基本信息，应用先要用户授权，才能获取许可,拿到授权

许可再去腾讯的授权服务拿访问令牌.....

摘要算法

我们拿到访问令牌。这个令牌是通过一些摘要算法生成令牌，也就是token(用账号密码登录，也是而可以生成token)。我们看到的token都是经过摘要算法加密后的。

Token Table

如何保证token唯一?

- 散列算法:sha(client_id + rnd +timestamp) 客户ID + 随机数+时间戳 生成的id应该重复就少了

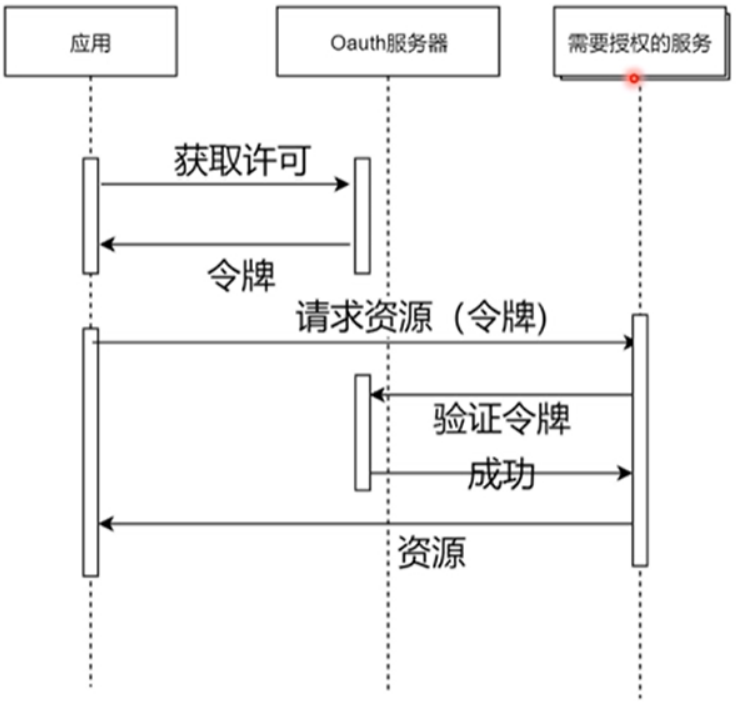
id	client_id	expire
d41d8cd98f……	100	2020-8-16 17:04:42

单点登录系统原理

单点登录(SSO)

用户通过一个单点登录的平台(例如：sso)，再去登录不同的系统。现在比较用的比较多的微信登录，我不管登录什么系统，都先登录微信，由微信去处理这些资源，如申请令牌等等.程序员的话，很多都是用github去处理这些资源。这就是所谓的单点登录。

Oauth with SSO



Oauth协议和sso单点登录。应用先去的Oauth服务获取许可，例如你应用是游戏，Oauth服务器是腾讯的，这个应用先去腾讯获取许可(前提是这个应用是腾讯有备案)，许可之后腾讯返回令牌，应用拿到这个令牌再去自己的授权服务验证(这个的令牌是腾讯生成的，只能是的腾讯去验证，腾讯验证完后告诉授权服务结果，结果可行，授权服务则返回相关资源，反之则不返回。)

课程小结:

- Cookie/Session/Storage: 状态同步
- Oauth: 资源开放
- SSO: 业务整合

像github开放资源可以使用Oauth SSO是业务整合的