

公众号_网页端

公众号_网页端

- 一、课前准备
- 二、课堂主题
- 三、课堂目标
- 四、知识要点
 - 1. 网页授权
 - 2. 微信JSSDK

一、课前准备

- 预习Oauth2原理 http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html

二、课堂主题

对接公众号服务器端接口

- 消息接口
- 服务器端接口的调用

三、课堂目标

1. 掌握公众号网页授权方法
2. 掌握Oauth2原理
3. 掌握JSSDK调用方法

四、知识要点

1. 网页授权

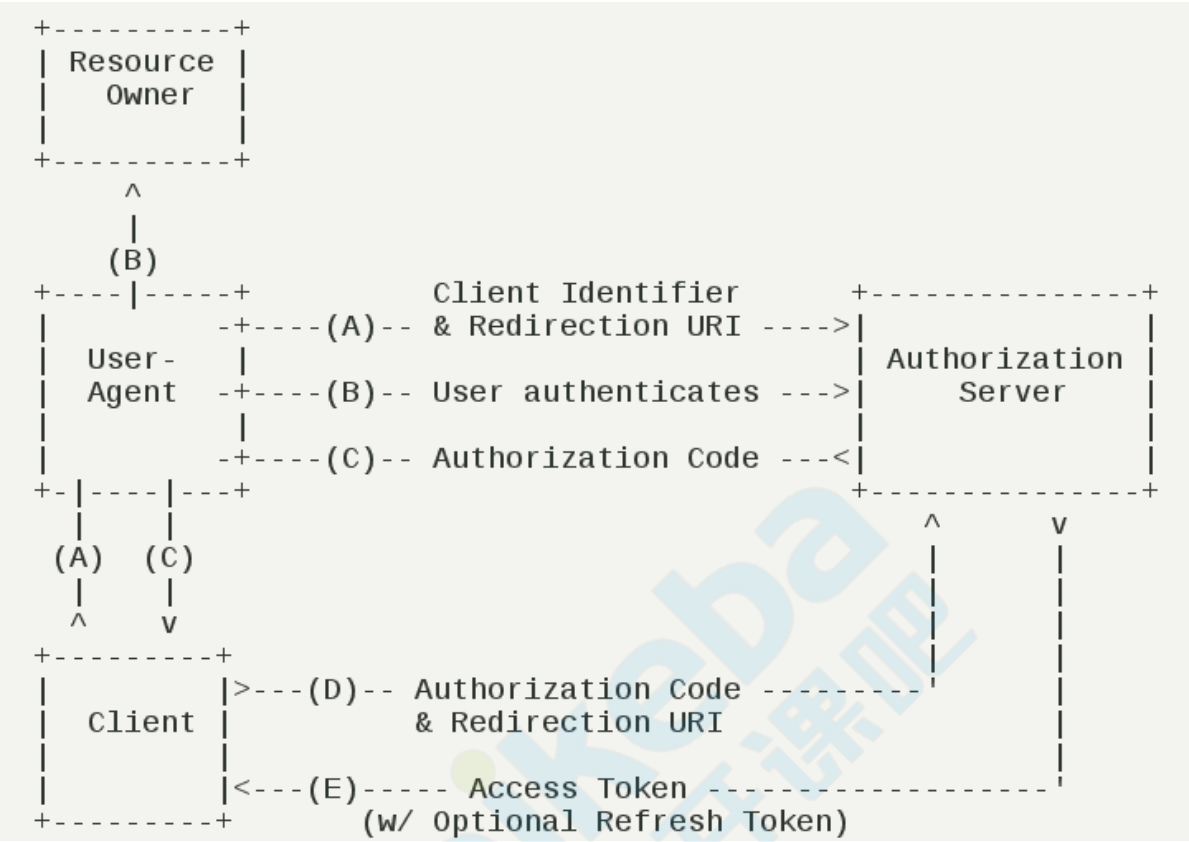
官方资料 https://mp.weixin.qq.com/wiki?t=resource/res_main&id=mp1421140842

npm库 <https://github.com/node-webot/wechat-oauth>

阮一峰的OAuth2 http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html

OAuth2.0的授权码模式

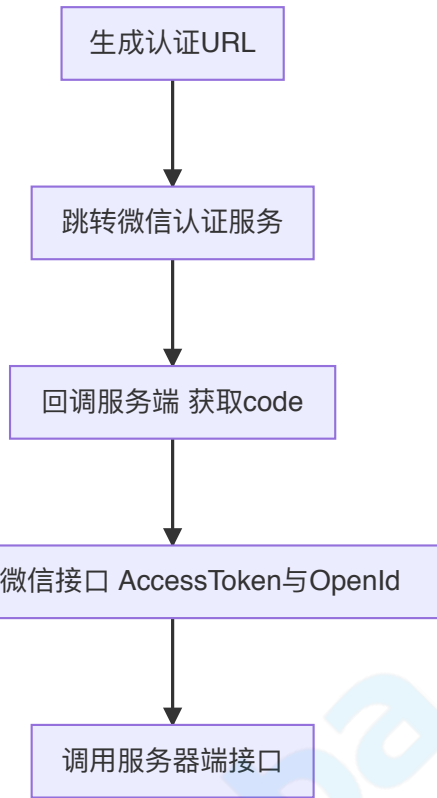
授权码模式（authorization code）是功能最完整、流程最严密的授权模式。它的特点就是通过客户端的后台服务器，与"服务提供商"的认证服务器进行互动。



- (A) 用户访问客户端，后者将前者导向认证服务器。
- (B) 用户选择是否给予客户端授权。
- (C) 假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个授权码。
- (D) 客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌。这一步是在客户端的后台的服务器上完成的，对用户不可见。
- (E) 认证服务器核对了授权码和重定向URI，确认无误后，向客户端发送访问令牌（access token）和更新令牌（refresh token）。

基于SPA的网页授权流程 <https://www.jianshu.com/p/27b8069b4178>

- 获取用户信息 - 相当于普通网页的用户登录



1. 配置网页回调

号	网页授权获取用户基本信息	无上限	修改
---	--------------	-----	----

OAuth2.0网页授权

授权回调页面域名:

用户在网页授权页同意授权给公众号后，微信会将授权数据传给一个回调页面，回调页面需在此域名下，以确保安全可靠。沙盒号回调地址支持域名和ip，正式公众号回调地址只支持域名。

2. 配置JS安全接口

JS接口安全域名 [修改](#)

设置JS接口安全域名后，通过关注该测试号，开发者即可在该域名下调用微信开放的JS接口，请阅读[微信JSSDK开发文档](#)。

域名 josephxia.free.idcfengye.com

1 网页端

```
<h3>网页授权</h3>
<hr>
<cube-button v-on:click="auth">微信登录</cube-button>
<cube-button v-on:click="getUser">获取用户信息</cube-button>
<hr>
```

```
// wechat/index.html
async auth () {
  window.location.href = '/wxAuthorize'
},
```

1. 初始化OAuth

```
// index.js
const OAuth = require('co-wechat-oauth');
const oauth = new OAuth(conf.appid, conf.appsecret);
```

2. 生成用户URL

```
// index.js
// 生成引导用户点击的 URL
router.get('/wxAuthorize', async (ctx, next) => {
  const state = ctx.query.id
  // const target = ctx.href
  console.log('ctx...' + ctx.href)
  // 目标地址
  const { protocol, hostname } = new URL(ctx.href)
  const redirect = `${protocol}://${hostname}/wxCallback`
  const scope = 'snsapi_userinfo'

  var url = oauth.getAuthorizeURL(redirect, state, scope);
  console.log('url' + url)
  ctx.redirect(url)
})
```

3. 获取用户回调 AccessToken与OpenId

```
// index.js
// 获取AccessToken
router.get('/wxCallback', async (ctx, next) => {
  const code = ctx.query.code // 授权码
  console.log('getAccessToken', code)
  var token = await oauth.getAccessToken(code);
  var accessToken = token.data.access_token;
  var openid = token.data.openid;
  console.log('getAccessToken....')
  console.log('accessToken', accessToken)
  console.log('openid', openid)
  // ctx.body = token

  ctx.redirect('/?openid=' + openid)
})
```

4. 用户信息

```
// index.html
async getUser(){
  const qs = Qs.parse(window.location.search.substr(1))
  const openid = qs.openid
  const res = await axios.get(`/getUser`,{
    params:{
      openid
    }
  })
  console.log('User',res.data)
},
```

```
// index.js
router.get('/getUser', async (ctx, next) => {
  const openid = ctx.query.openid
  console.log('getUser', openid)
  var userInfo = await oauth.getUser(openid);
  console.log('userInfo:', userInfo)
  ctx.body = userInfo
})
```

5. AccessToken缓存问题

```
// mongo.js
// ClientAccessToken
// mongoose.js
const mongoose = require('mongoose')
const {
```

```

    Schema
  } = mongoose

  mongoose.connect('mongodb://localhost:27017/weixin', {
    useNewUrlParser: true
  }, () => {
    console.log('Mongodb connected..')
  })

  exports.ServerToken = mongoose.model('ServerToken', {
    accessToken: String
  });

  // ClientAccessToken
  schema = new Schema({
    access_token: String,
    expires_in: Number,
    refresh_token: String,
    openid: String,
    scope: String,
    create_at: String
  });
  // 自定义getToken方法
  schema.statics.getToken = async function (openid) {
    return await this.findOne({
      openid: openid
    });
  };
  schema.statics.setToken = async function (openid, token) {
    // 有则更新, 无则添加
    const query = {
      openid: openid
    };
    const options = {
      upsert: true
    };
    return await this.updateOne(query, token, options);
  };

  exports.ClientToken = mongoose.model('ClientToken', schema);

```

```
const OAuth = require('co-wechat-oauth');
const oauth = new OAuth(conf.appid, conf.appsecret,
  async function (openid) {
    return await ClientToken.getToken(openid)
  },
  async function (openid, token) {
    return await ClientToken.setToken(openid, token)
  }
)
```

2. 微信JSSDK

官方资料: https://mp.weixin.qq.com/wiki?t=resource/res_main&id=mp1421141115

npm库: <https://github.com/node-webot/co-wechat-api> (获取JSConfig)

是开发者在网页上通过JavaScript代码使用微信原生功能的工具包, 开发者可以使用它在网页上录制和播放微信语音、监听微信分享、上传手机本地图片、拍照等许多能力

- 运行于微信内置浏览器的网页
- 调用微信原生应用如: 拍照、语音、扫一扫
- 分享功能 查到的数据不同
- 图像接口
- 音频接口

```
// index.js
// 获取JSConfig
router.get('/getJsConfig', async ctx => {
  console.log('getJSSDK...', ctx.query)

  var res = await api.getJsConfig(ctx.query);
  console.log('res', res)
  ctx.body = res
})
```

```
// index.html
<script src="http://res.wx.qq.com/open/js/jweixin-1.6.0.js"></script>
```

```
// index.html
getJSConfig: async function () {
  console.log('wx:', wx)
  let res = await axios.get('/getJSConfig', {
    params: {
      url: window.location.href
    }
  })
}
```

```

    console.log('res.....', res.data)
    res.data.jsApiList = ['onMenuShareTimeline', 'onMenuShareAppMessage']
    //认证
    wx.config(res.data);
    wx.ready(function () {
        console.log('wx.ready.....')
    });
    // 获取网络地址
    wx.getNetworkType({
        success: function (res) {
            // 返回网络类型2g, 3g, 4g, wifi
            var networkType = res.networkType;
            console.log('getNetworkType...', networkType)
        }
    });
},

```

> jsApiList参考 https://mp.weixin.qq.com/wiki?t=resource/res_main&id=mp1421141115

五、拓展点、未来计划、行业趋势 (5min)

六、总结

[TOC]

七、作业

八、互动问答

webpack proxy 代理后端

后端提供静态服务 + webpack watch

