link: null
title: 珠峰架构师成长计划
description: 传统方法是，QQ用户将自己的QQ号和密码，告诉百度，后者就可以读取用户的相册了。这样的做法有以下几个严重的缺点
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=117 sentences=291, words=1610

1. 应用场景

- 比如百度希望能获取QQ用户的相册
- 问题是只有得到用户的授权，QQ才会同意让百度读取这些照片
- 那么，我们怎样获得用户的授权呢？

传统方法是，QQ用户将自己的QQ号和密码，告诉百度，后者就可以读取用户的相册了。这样的做法有以下几个严重的缺点

1. 百度为了后续的服务，会保存用户的密码，这样很不安全
2. 百度不得不部署密码登录，而我们知道，单纯的密码登录并不安全
3. 百度拥有了获取用户储存在QQ上所有资料的权力，用户没法限制百度获得授权的范围和有效期
4. 用户只有修改密码，才能收回赋予百度的权力。但是这样做，会使得其他所有获得用户授权的第三方应用程序全部失效
5. 只要有一个第三方应用程序被破解，就会导致用户密码泄漏，以及所有被密码保护的数据泄漏。

> OAuth就是为了解决上面这些问题而诞生的。

2. 名词

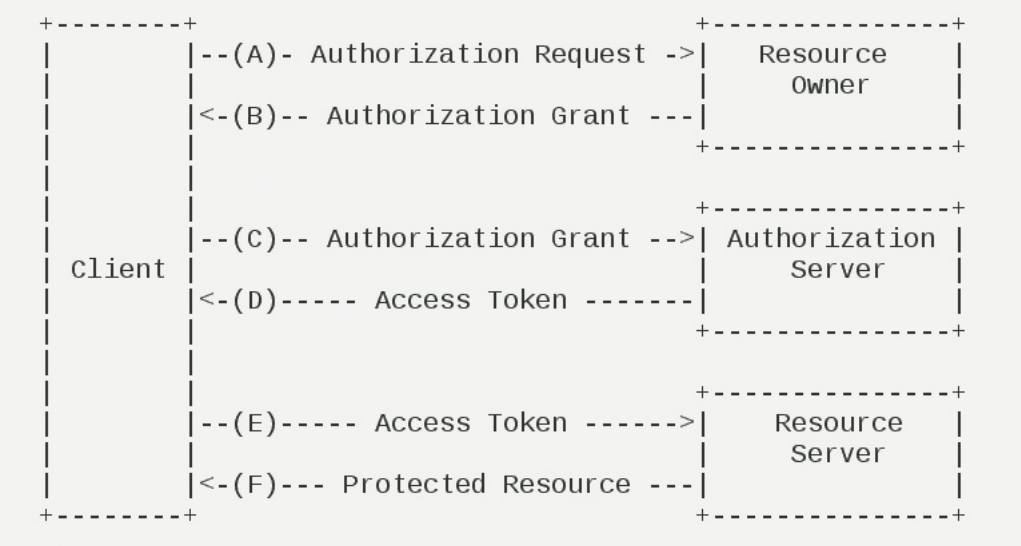
1. client: 第三方应用程序，本文中又称客户端，即上个例子中的"百度"
2. Resource Owner：资源所有者，本文中又称"QQ用户"（user）。
3. User Agent：用户代理，本文中就是指浏览器。
4. http service: 提供服务的HTTP服务提供商， 即上个例子中的"QQ服务器"
5. Authorization server：认证服务器，即服务提供商专门用来处理认证的服务器。
6. Resource server：资源服务器，即服务提供商存放用户生成的资源的服务器。它与认证服务器，可以是同一台服务器，也可以是不同的服务器。

> OAuth的作用就是让"客户端"安全可控地获取"用户"的授权，与"服务商提供商"进行互动。

3.设计思路

- OAuth在"客户端"与"服务提供商"之间，设置了一个授权层（authorization layer）。
- "客户端"不能直接登录"服务提供商"，只能登录授权层，以此将用户与客户端区分开来。
- "客户端"登录授权层所用的令牌（token），与用户的密码不同。用户可以在登录的时候，指定授权层令牌的权限范围和有效期。
- "客户端"登录授权层以后，"服务提供商"根据令牌的权限范围和有效期，向"客户端"开放用户储存的资料。

4. 工作流程



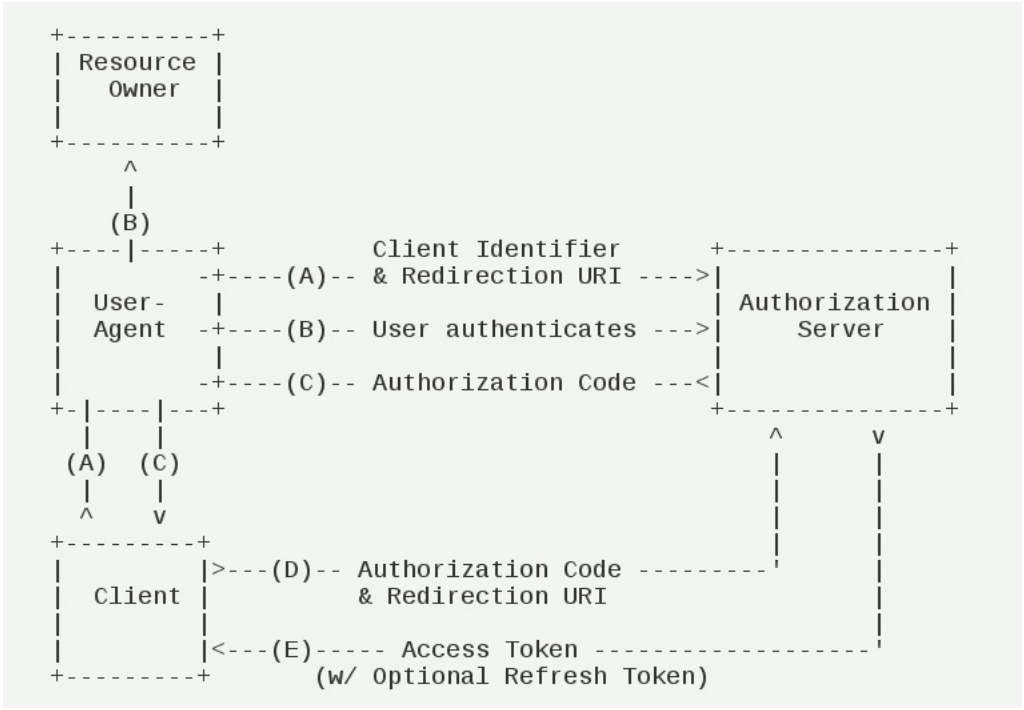
- A 用户打开客户端以后，客户端要求用户给予授权。
- B 用户同意给予客户端授权。
- C 客户端使用上一步获得的授权，向认证服务器申请令牌。
- D 认证服务器对客户端进行认证以后，确认无误，同意发放令牌。
- E 客户端使用令牌，向资源服务器申请获取资源。
- F 资源服务器确认令牌无误，同意向客户端开放资源。

5. 客户端的授权模式

- 客户端必须得到用户的授权（authorization grant），才能获得令牌（access token）
- OAuth 2.0定义了四种授权方式。

5.1 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。

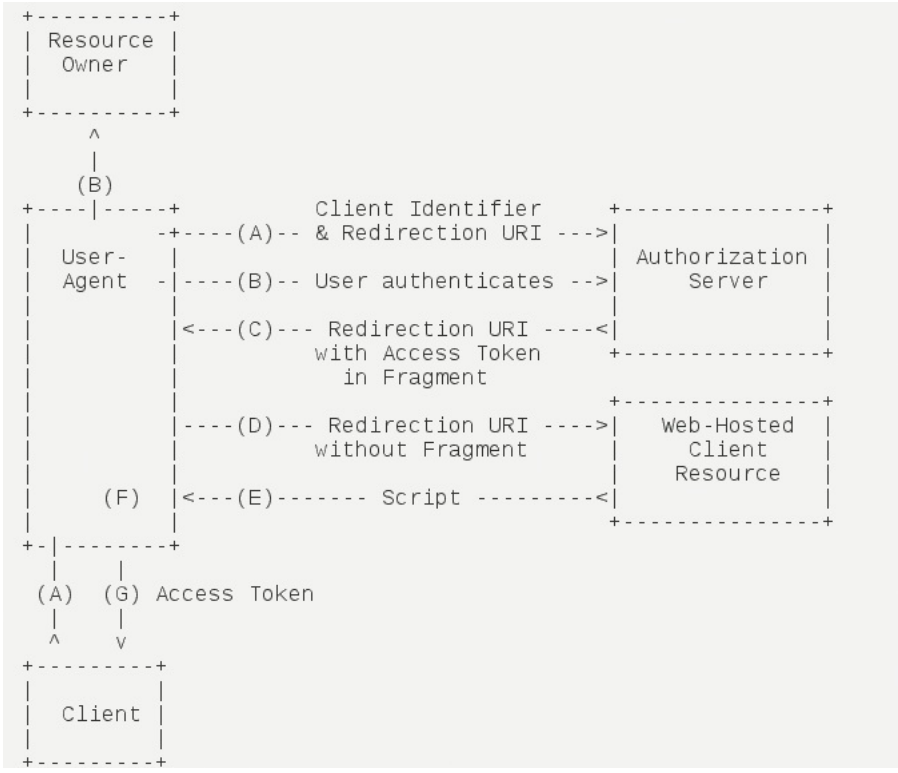


它的步骤如下：

- 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）

5.2 简化模式

简化模式（implicit grant type）不通过第三方应用程序的服务器，直接在浏览器中向认证服务器申请令牌，跳过了"授权码"这个步骤，因此得名。所有步骤在浏览器中完成，令牌对访问者是可见的，且客户端不需要认证。

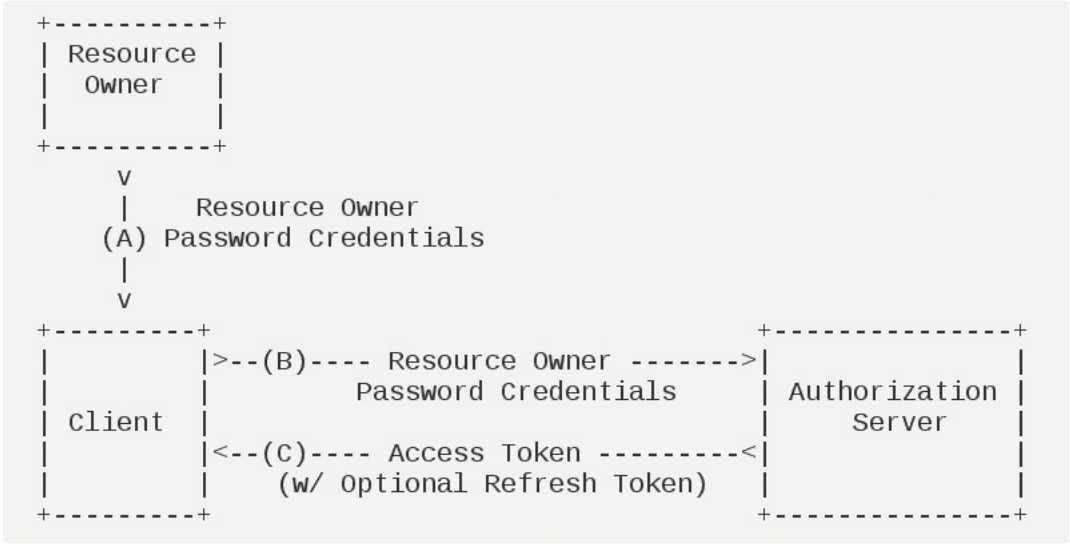


它的步骤如下：

- A 客户端将用户导向认证服务器。
- B 用户决定是否给于客户端授权。
- C 假设用户给予授权，认证服务器将用户导向客户端指定的"重定向URI"，并在URI的Hash部分包含了
- D 浏览器向资源服务器发出请求，其中不包括上一步收到的Hash值。
- E 资源服务器返回一个网页，其中包含的代码可以获取Hash值中的令牌。
- F 浏览器执行上一步获得的脚本，提取出令牌。
- G 浏览器将令牌发给客户端。

5.3 密码模式

- 密码模式（Resource Owner Password Credentials Grant）中，用户向客户端提供自己的用户名和密码。客户端使用这些信息，向"服务商提供商"索要授权。
- 在这种模式中，用户必须把自己的密码给客户端，但是客户端不得储存密码。这通常用在用户对客户端高度信任的情况下，比如客户端是操作系统的一部分，或者由一个著名公司出品。而认证服务器只有在其他授权模式无法执行的情况下，才能考虑使用这种模式。

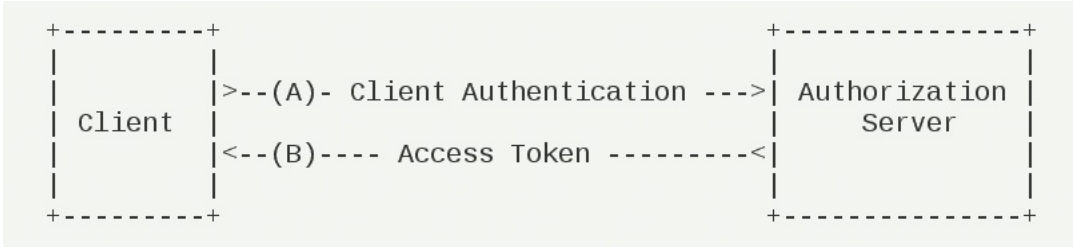


它的步骤如下：

- A 用户向客户端提供用户名和密码。
- B 客户端将用户名和密码发给认证服务器，向后者请求令牌。
- C 认证服务器确认无误后，向客户端提供访问令牌。

5.4 客户端模式

- 客户端模式（Client Credentials Grant）指客户端以自己的名义，而不是以用户的名义，向"服务提供商"进行认证。
- 严格地说，客户端模式并不属于OAuth框架所要解决的问题。在这种模式中，用户直接向客户端注册，客户端以自己的名义要求"服务提供商"提供服务，其实不存在授权问题。



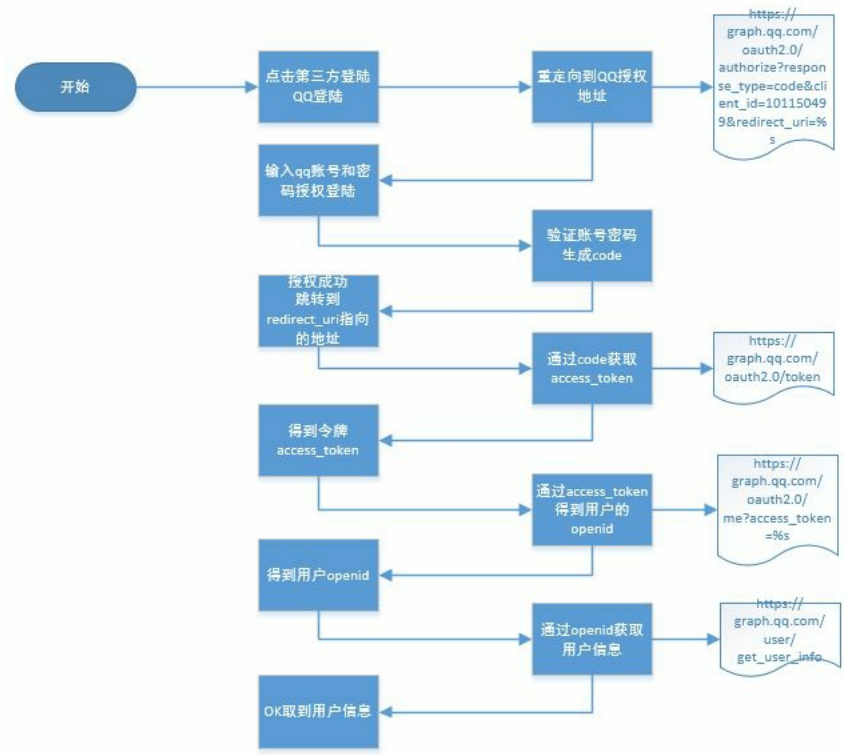
它的步骤如下：

- A 客户端向认证服务器进行身份认证，并要求一个访问令牌。
- B 认证服务器确认无误后，向客户端提供访问令牌。

6. 接入QQ

- QQ网站接入流程 (http://wiki.connect.qq.com/%E7%BD%91%E7%AB%99%E6%8E%A5%E5%85%A5%E6%B5%81%E7%A8%8B)
- 使用Authorization Code获取Access Token (http://wiki.connect.qq.com/%E4%BD%BF%E7%94%A8authorization_code%E8%8E%B7%E5%8F%96access_token)

6.1 接入流程



1. 开发者注册
2. 放置QQ登录按钮
3. 获取Access Token
4. 获取用户的OpenID
5. 调用OpenAPI访问修改用户信息

6.2 QQ开发平台的前置条件

- QQ
- 可以通过域名访问的服务器
- 关于备案
- 服务器域名环境

6.3 安装服务器

```
apt update
apt upgrade
wget -qO- https:
source  /root/.bashrc
nvm install stable
npm install pm2 -g
apt install nginx
```

6.4 添加应用

- 域名http://front.zhufengpeixun.cn (http://front.zhufengpeixun.cn)
- 回调地址http://front.zhufengpeixun.cn/user/callback (http://front.zhufengpeixun.cn/user/callback)
- 网站名称 珠峰前端精华
- 网站备案号
- 网站图标

6.5 编写应用

6.5.1 QQ连接

```
https:
```

6.5.2 获取数据流程

```
const client_id='101499238';
const client_secret="1af7ef89da1cbb3dae41465a1865a523";
const redirect_uri='http://front.zhufengpeixun.cn/user/callback';
const code='57C038D3AD7C5588570FF8F723EB57DE';
const request = require('then-jsonp');
const axios=require('axios');
const querystring=require('querystring');
const access_token='0CF4BD6B1BA83606E20218E6BA16FF4A';
const refresh_token='7E3C2110B9813BF714BA212094699F7F';
const openid='0E302B779D8D4C72D21E3172C015682B';
const options={
    grant_type: 'authorization_code',
    client_id,
    client_secret,
    code,
    redirect_uri
}
const query=querystring.stringify(options);

 (async function () {
    const response=await axios.get(`https://graph.qq.com/oauth2.0/token?${query}`);
    const data=response.data;
    const result=querystring.parse(data);
    return result;
})().then(ret => console.log(ret));

(async function () {
    const response=await request('GET',`https://graph.qq.com/oauth2.0/me?access_token=${access_token}`,{callbackName:"callback"});
    return response;
})().then(ret => console.log(ret));

(async function () {
    const options={
        grant_type: 'refresh_token',
        client_id,
        client_secret,
        refresh_token
    }
    const query=querystring.stringify(options);
    const response=await axios.get(`https://graph.qq.com/oauth2.0/token?${query}`);
    const data=response.data;
    const result=querystring.parse(data);
    return result;
})().then(ret => console.log(ret));

(async function () {
    const options={
        access_token,
        oauth_consumer_key:client_id,
        openid
    }
    const query=querystring.stringify(options);
    const response=await axios.get(`https://graph.qq.com/user/get_user_info?${query}`);
    const data=response.data;
    return data;
})().then(ret => console.log(ret));
```

- qq-client (https://gitee.com/zhufengpeixun/qq-client)

7.开发自己的Oauth系统

7.1 开发客户端

auth-client (https://gitee.com/zhufengpeixun/auth-client)

7.1.1 config.js

```
module.exports={
    appId: '38aa47d5-da52-417c-a742-1a341a689fcc',
    appKey: 'e49b9b51-7348-4d1e-be59-7bb3125d9337',
    redirect_uri:'http://localhost:3000/user/callback',
    fetchAccessTokenUrl: 'http://localhost:4000/oauth2.0/token',
    fetchOpenIdUrl: 'http://localhost:4000/oauth2.0/me',
    getUserInfoUrl:'http://localhost:4000/user/get_user_info'
}
```

7.1.2 utils.js

```
let {parse,format}=require('url');
exports.addQueryParamsToUrl=function (url,options) {
    let {protocol,host,pathname,query}=parse(url,true);
    Object.assign(query,options);
    return format({protocol,host,pathname,query});
}
```

7.1.3 model.js

```
let mongoose=require('mongoose');

mongoose.set('useFindAndModify',false);
const opts = { useNewUrlParser: true };
let conn=mongoose.createConnection('mongodb://localhost/client',opts);
let Schema=mongoose.Schema;

let UserSchema=new Schema({
    access_token: {type: String,required: true},
    refresh_token:String,
    username: {type:String,required:true},
    avatar:{type:String,required:true}
});

exports.User=conn.model('User',UserSchema);
```

7.1.4 app.js

```
const express=require('express');
const path=require('path');
const session=require('express-session');
const bodyParser=require('body-parser');
const app=express();
const logger=require('morgan');
const oauth=require('./routes/oauth2');
const user=require('./routes/user');
app.set('view engine','html');
app.set('views',path.resolve(__dirname,'views'));
app.engine('html',require('ejs').__express);

app.use(bodyParser.urlencoded({extends: true}));
app.use(logger('dev'));
app.use(session({
    secret: 'zfpx',
    saveUninitialized: true,
    resave:true
}));
app.use(function (req,res,next) {
    res.createError=function (status,message) {
        let error=new Error(message);
        error.code=1;
        error.status=status;
        error.message=message;
        return error;
    }
    req.session.user={
        "_id": "5b7f6f189d384c73a7ee9038",
        "username": "zhangsan",
        "password": "4",
        "avatar":"http://www.gravatar.com/avatar/93e9084aa289b7f1f5e4ab6716a56c3b"
    };
    next();
});
app.use('/oauth2.0',oauth);
app.use('/user',user);
app.use(function (err,req,res,next) {
    res.status(err.status||500).json({code:err.code,message:err.message});
});
app.listen(4000,() => {
    console.log(`服务已经在4000端口上启动`);
});
```

7.1.5 user.js

```
let express=require('express');
const axios=require('axios');
const {User}=require('../model');
const querystring = require('querystring');
const {appId,appKey,redirect_uri,fetchAccessTokenUrl,fetchOpenIdUrl,getUserInfoUrl}=require('../config');
const {addQueryParamsToUrl}=require('../utils');
let router=express.Router();
router.get('/login',function (req,res) {
    res.render('login');
});
router.get('/callback',async function (req,res) {
    let {code}=req.query;
    let options={
        grant_type: 'authorization_code',
        client_id: appId,
        client_secret: appKey,
        code,
        redirect_uri
    }
    let url=addQueryParamsToUrl(fetchAccessTokenUrl,options);
    let result=await axios.get(url);
    let {access_token,expires_in,refresh_token}=querystring.parse(result.data);

    url=addQueryParamsToUrl(fetchOpenIdUrl,{
        access_token
    });
    result=await axios.get(url);
    let start=result.data.indexOf('{');
    let end=result.data.lastIndexOf('}');
    result.data=result.data.slice(start,end+1);
    let {client_id,openid}=JSON.parse(result.data);
    url=addQueryParamsToUrl(getUserInfoUrl,{
        access_token,
        oauth_consumer_key: client_id,
        openid
    });
    result=await axios.get(url);
    let {username,avatar}=result.data;
    let user = await User.create({
        access_token,
        refresh_token,
        username,
        avatar
    });
    req.session.user=user;
    res.redirect('/');
});
module.exports=router;
```

7.1.6 login.html

```
<div class="row">
    <div class="col-md-12">
      <a href="http://localhost:4000/oauth2.0/authorize?
response_type=code&client_id=5b7f704beacb0274b068da17&redirect_uri=http%3A%2F%2Flocalhost%3A3000%2Fuser%2Fcallback&scope=list_album,get_user_info">
           <img src="/images/qq.png">
        a>
    div>
div>
<%include footer.html%>
```

7.1.7 index.html

```
<div class="row">
    <div class="col-md-12">
        你已经登录，欢迎你 <%=user.username%>
        <img src=""/>
    div>
div>
```

7.2 开发授权服务器

auth-server (https://gitee.com/zhufengpeixun/auth-server)

7.2.1 model.js

```
let mongoose=require('mongoose');

mongoose.set('useFindAndModify',false);
const opts = { useNewUrlParser: true };
let conn=mongoose.createConnection('mongodb://localhost/oauth',opts);
let Schema=mongoose.Schema;
let ObjectId=Schema.Types.ObjectId;
let ApplicationSchema=new Schema({
    appId: {type:String,required:true},
    appKey: {type:String,required:true},
    website: {type: String,required: true},
    redirect_uri: {type: String,required: true}
});

exports.Application=conn.model('Application',ApplicationSchema);

let UserSchema=new Schema({
    username: {type:String,required:true},
    password: {type: String,required: true},
    avatar:{type:String,required:true}
});

exports.User=conn.model('User',UserSchema);

let AuthorizationCodeSchema=new Schema({
    client_id: {type: String,required: true},
    user: {type: ObjectId,ref: 'User',required:true},
    permissions: {type: [{type:ObjectId,ref:'Permission'}],required: true},
    createAt:{type:Date,default:Date.now}
});

exports.AuthorizationCode=conn.model('AuthorizationCode',AuthorizationCodeSchema);

let PermissionSchema=new Schema({
    name: {type:String,required:true},
    scope: {type: String,required: true}
});

exports.Permission=conn.model('Permission',PermissionSchema);

let AccessTokenSchema=new Schema({
    client_id: {type: String,required: true},
    user: {type: ObjectId,ref: 'User',required:true},
    permissions: {type: [{type: ObjectId,ref: 'Permission'}],required: true},
    refresh_token:{type:String},
    createAt:{type:Date,default:Date.now}
});

exports.AccessToken=conn.model('AccessToken',AccessTokenSchema);
```

7.2.2 mock.js

```
let uuid=require('uuid');
let {Application,User,AuthorizationCode,Permission}=require('./model');

let appId=uuid.v4();
let appKey=uuid.v4();
let redirect_uri='http://localhost:3000/user/callback';
let encode_redirect_uri=encodeURIComponent(redirect_uri);
console.log(encode_redirect_uri);

let name="获取登录用户的相册列表";

let scope='list_album';
```

7.2.3 app.js

```
const express=require('express');
const path=require('path');
const session=require('express-session');
const bodyParser=require('body-parser');
const app=express();
const logger=require('morgan');
const oauth=require('./routes/oauth2');
const user=require('./routes/user');
app.set('view engine','html');
app.set('views',path.resolve(__dirname,'views'));
app.engine('html',require('ejs').__express);

app.use(bodyParser.urlencoded({extends: true}));
app.use(logger('dev'));
app.use(session({
    secret: 'zfpx',
    saveUninitialized: true,
    resave:true
}));
app.use(function (req,res,next) {
    res.createError=function (status,message) {
        let error=new Error(message);
        error.code=1;
        error.status=status;
        error.message=message;
        return error;
    }
    req.session.user={
        "_id": "5b7f6f189d384c73a7ee9038",
        "username": "zhangsan",
        "password": "4",
        "avatar":"http://www.gravatar.com/avatar/93e9084aa289b7f1f5e4ab6716a56c3b"
    };
    next();
});
app.use('/oauth2.0',oauth);
app.use('/user',user);
app.use(function (err,req,res,next) {
    res.status(err.status||500).json({code:err.code,message:err.message});
});
app.listen(4000,() => {
    console.log(`服务已经在4000端口上启动`);
});
```

7.2.4 oauth2.js

```
let express=require('express');
let uuid=require('uuid');
const querystring=require('querystring');
const {Application,Permission,AuthorizationCode,AccessToken}=require('../model');
let router=express.Router();
router.get('/authorize',async function (req,res,next) {

    let {response_type='code',client_id,redirect_uri,scope='get_user_info'}=req.query;
    if (!client_id) {
        return next(res.createError(400,'缺少 client_id 参数'));
    }
    if (!redirect_uri) {
        return next(res.createError(400,'缺少 redirect_uri 参数'));
    }
    redirect_uri=decodeURIComponent(redirect_uri);
    let client=await Application.findById(client_id);
    if (client.redirect_uri !== redirect_uri) {
        return next(res.createError(400,'传入的回调地址不匹配'));
    }
    let query={$or: scope.split(',').map(item => ({scope: item}))};
    let permissions=await Permission.find(query);
    res.render('authorize',{
        user:req.session.user,
        client,
        permissions
    });
});

router.post('/authorize',async function (req,res,next) {
    let {client_id,redirect_uri}=req.query;
    let {permissions}=req.body;
    if (!Array.isArray(permissions)) {
        permissions=[permissions]
    }
    let authorizationCode=await AuthorizationCode.create({
        client_id,
        user: req.session.user._id,
        permissions
    });
    res.redirect(`${redirect_uri}?code=${authorizationCode._id}`);
});

router.get('/token',async function (req,res,next) {
    let {grant_type,client_id,client_secret,code,redirect_uri}=req.query;
    let authorizationCode=await AuthorizationCode.findById(code);
    if (!authorizationCode) {
        return next(res.createError(400,'授权码错误'));
    }
    let accessToken=await AccessToken.create({
        client_id: authorizationCode.client_id,
        user:authorizationCode.user,
        refresh_token: uuid.v4(),
        permissions:authorizationCode.permissions
    });
    let result={access_token: accessToken._id.toString(),expires_in: 60*60*24*90};
    res.send(querystring.stringify(result));
});

router.get('/me',async function (req,res,next) {
    let {access_token}=req.query;
    if (!access_token) {
        return next(res.createError(400,'access_token未提供'));
    }
    let {client_id,user:openid} = await AccessToken.findById(access_token);
    let result={
        client_id,
        openid
    }
    res.send(`callback(${JSON.stringify(result)})`);
});
module.exports=router;
```

7.2.5 routes/users.js

```
let express = require('express');
const {User} = require('../model');
let router=express.Router();
router.get('/get_user_info',async function (req,res) {
    let {access_token,oauth_consumer_key,openid}=req.query;
    let user = await User.findById(openid);
    res.json(user);
});
module.exports=router;
```

7.2.6 authorize.html

```
<%include header.html%>
<div class="row">
    <div class="col-md-12">
       <form method="POST">
          <div class="row">
           <div class="col-md-6">
               <div class="panel panel-default">
                   <div class="panel-heading">
                       <h4 class="text-center">快速登录h3>
                   div>
                   <div class="panel-body text-center">
                       <img src="" style="display:inline-block" class="img-responsive" alt="头像">
                   div>
                   <div class="panel-footer text-center">
                     <input type="submit" class="btn btn-primary" value="授权并登录">
                   div>
               div>

           div>
           <div class="col-md-6">
             <ul class="list-group">
                <li class="list-group-item"><%=client.website%>将获得以下权限：li>
                <li class="list-group-item">
                  <div class="checkbox">
                         <label>
                          <input type="checkbox">全选
                        label>
                  div>
                  <%permissions.forEach(function(permission){%>
                     <div class="checkbox">
                         <label>
                          <input type="checkbox" name="permissions" value=""><%=permission.name%>
                        label>
                     div>
                  <%})%>
                li>
                <li class="list-group-item">授权后表明你已同意 QQ登录服务协议li>
             ul>
           div>
       div>
      form>
    div>
div>
<%include footer.html%>
```

8. 参考资料

- rfc6749 (http://www.rfcreader.com/#rfc6749)
- QQ互联 (https://connect.qq.com/)
- 开发攻略_Server-side的Step2和Step3 (http://wiki.connect.qq.com/%E5%BC%80%E5%8F%91%E6%94%BB%E7%95%A5_server-side#Step2.EF.BC.9A.E8.8E.B7.E5.8F.96AuthorizationCode)
- 使用Authorization Code获取Access Token (http://wiki.connect.qq.com/%E4%BD%BF%E7%94%A8authorization_code%E8%8E%B7%E5%8F%96access_token)
- 理解OAuth 2.0 (http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html)