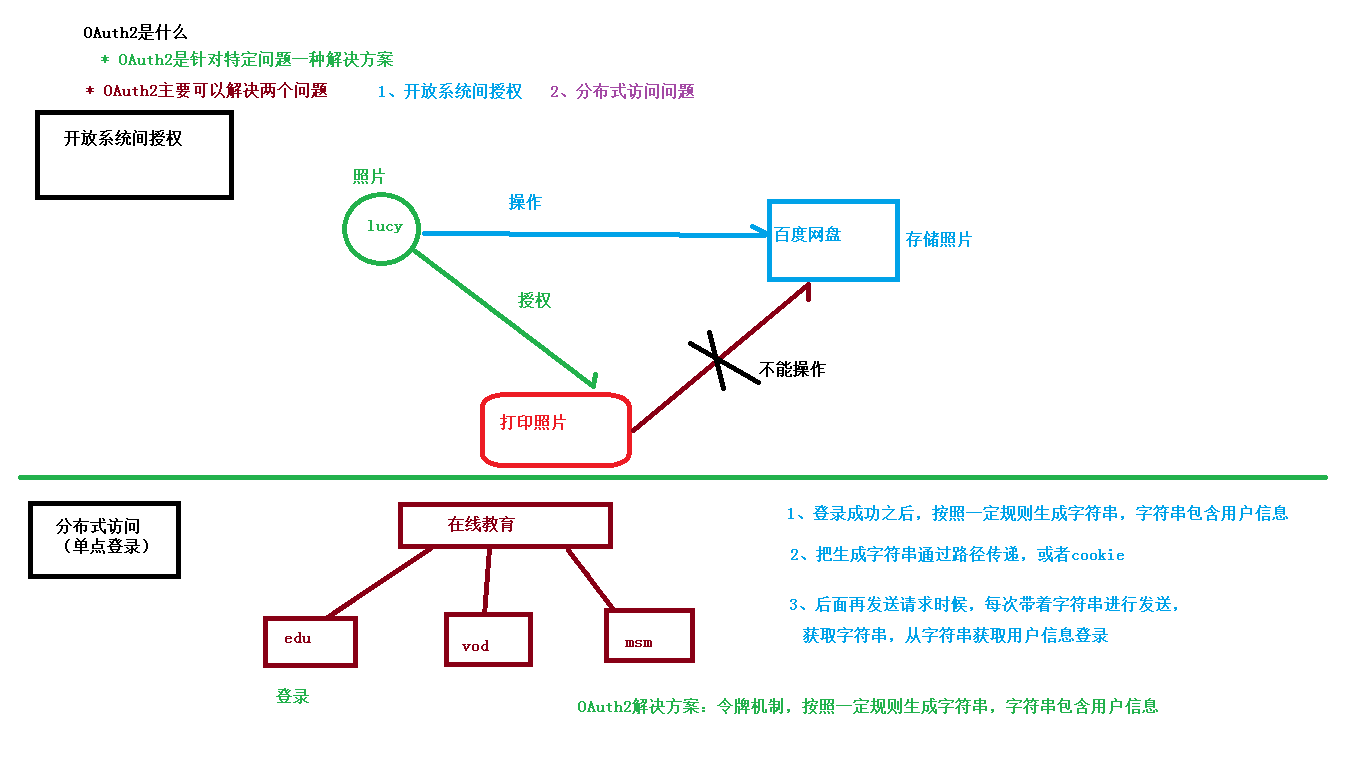
# 1 OAuth2

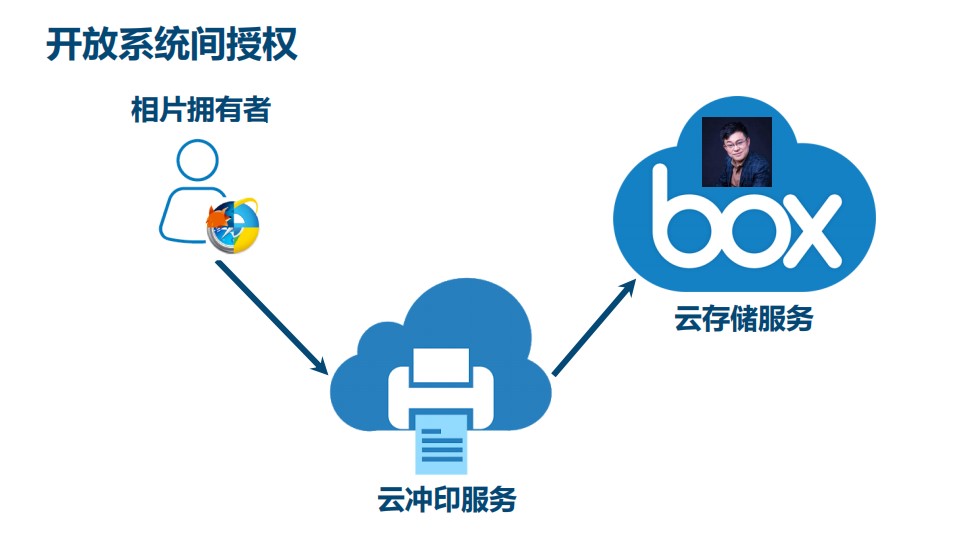
## OAuth2的使用场景



### 一、OAuth2解决什么问题

#### 1、OAuth2提出的背景

照片拥有者想要在云冲印服务上打印照片，云冲印服务需要访问云存储服务上的资源



#### 2、图例

资源拥有者：照片拥有者

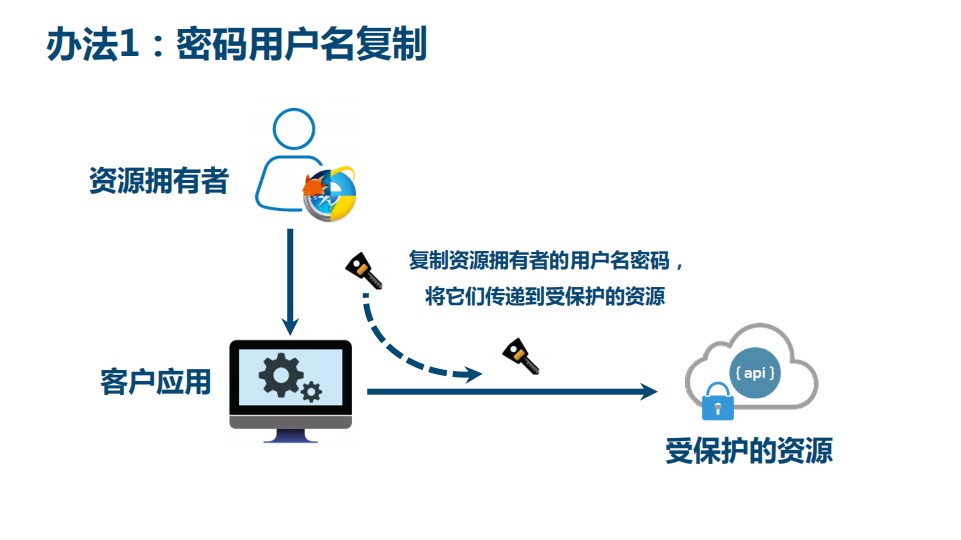
客户应用：云冲印

受保护的资源：照片



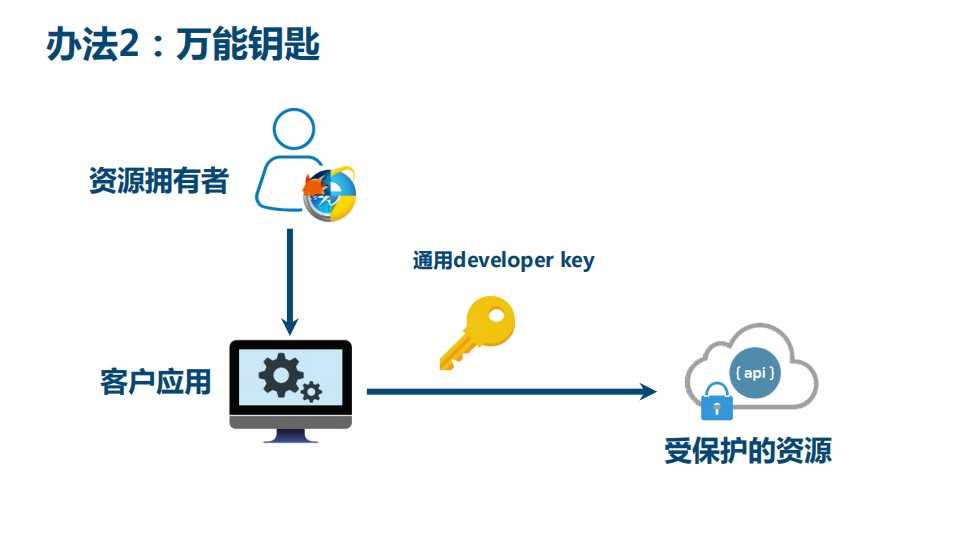
#### 3、方式一：用户名密码复制

适用于同一公司内部的多个系统，不适用于不受信的第三方应用



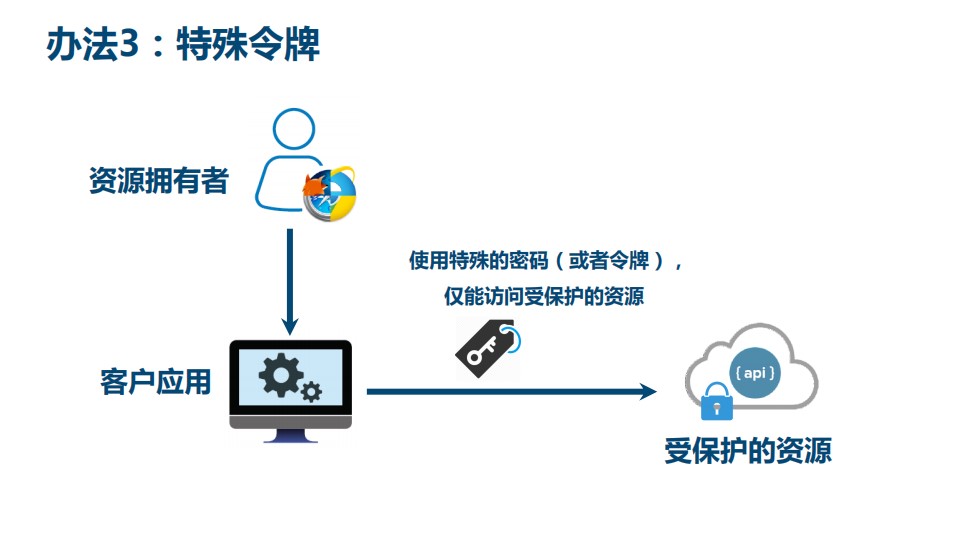
#### 4、方式二：通用开发者key

适用于合作商或者授信的不同业务部门之间



#### 5、方式三：办法令牌

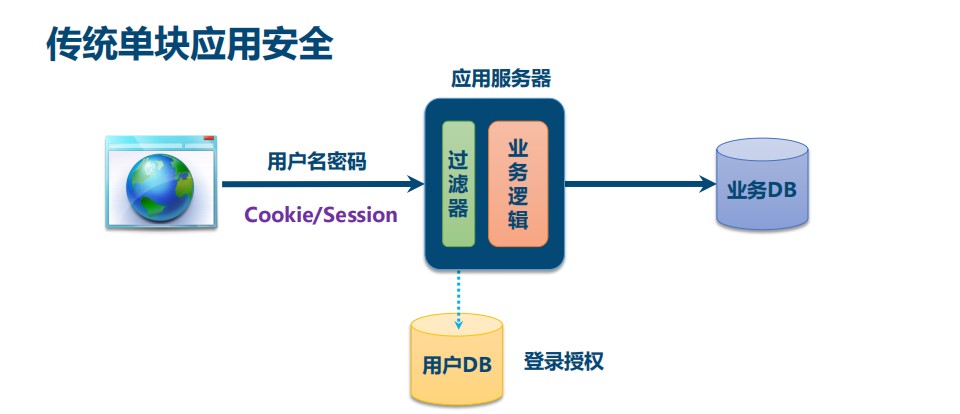
接近OAuth2方式，需要考虑如何管理令牌、颁发令牌、吊销令牌，需要统一的协议，因此就有了OAuth2协议



### 二、现代微服务安全

除了开放系统授权，OAuth2还可以应用于现代微服务安全

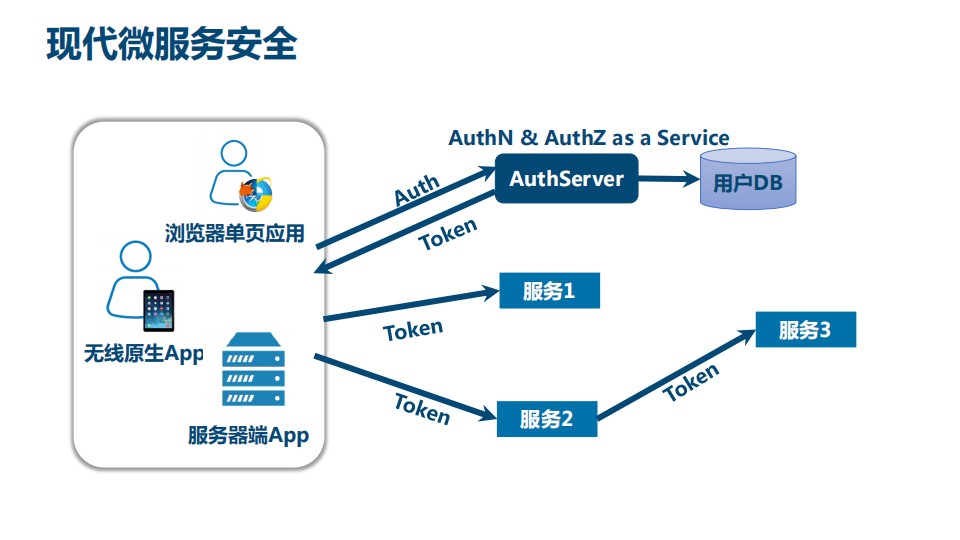
#### 1、传统单块应用的安全



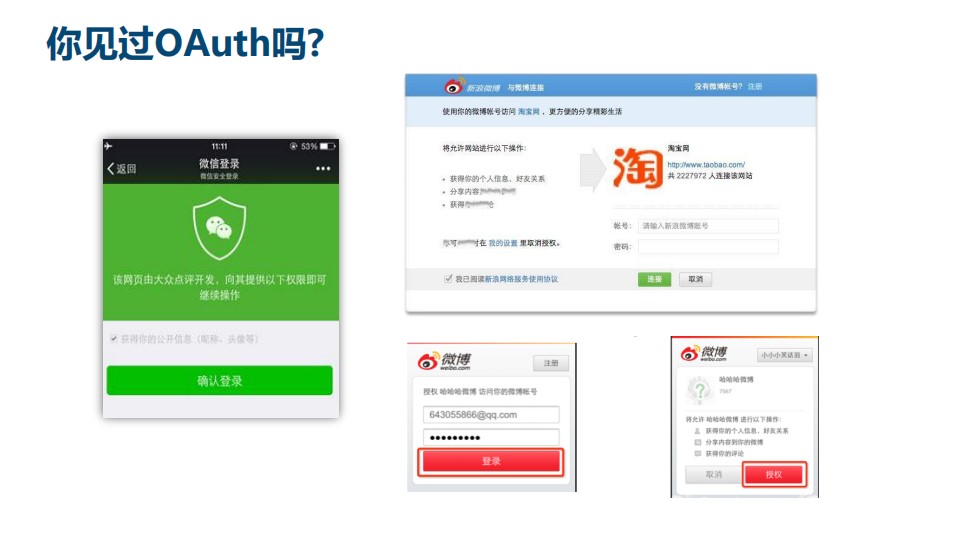
#### 2、现代微服务安全

现代微服务中系统微服务化以及应用的形态和设备类型增多，不能用传统的登录方式

核心的技术不是用户名和密码，而是token，由AuthServer颁发token，用户使用token进行登录



#### 3、典型的OAuth2应用



### 三、总结



**四、OAuth2最简向导**

川崎高彦：OAuth2领域专家，开发了一个OAuth2 sass服务，OAuth2 as Service，并且做成了一个公司

再融资的过程中为了向投资人解释OAuth2是什么，于是写了一篇文章，《OAuth2最简向导》

## 02-OAuth2的正式定义

### 一、什么是OAuth2

#### 1、OAuth2正式定义



#### 2、令牌的核心



#### 3、OAuth2的历史



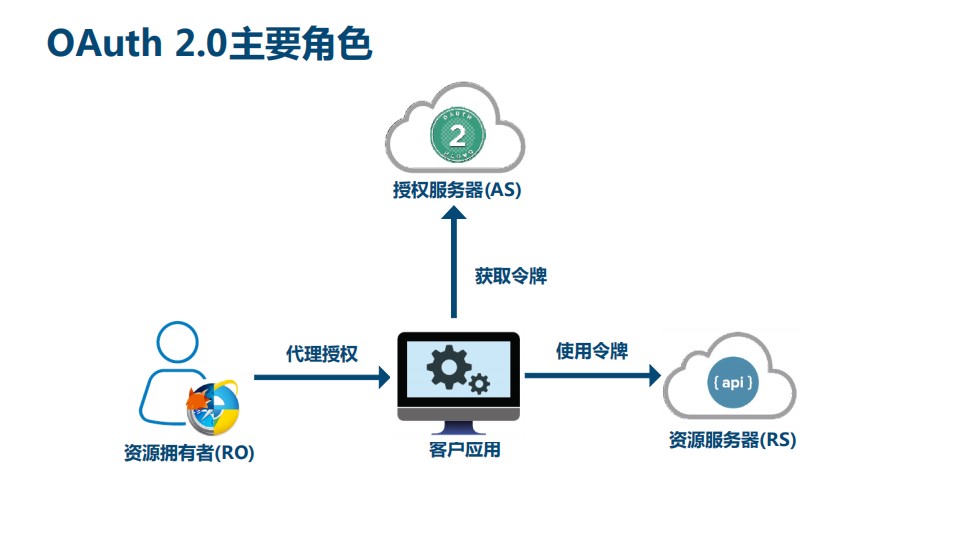
#### 4、OAuth2的优势



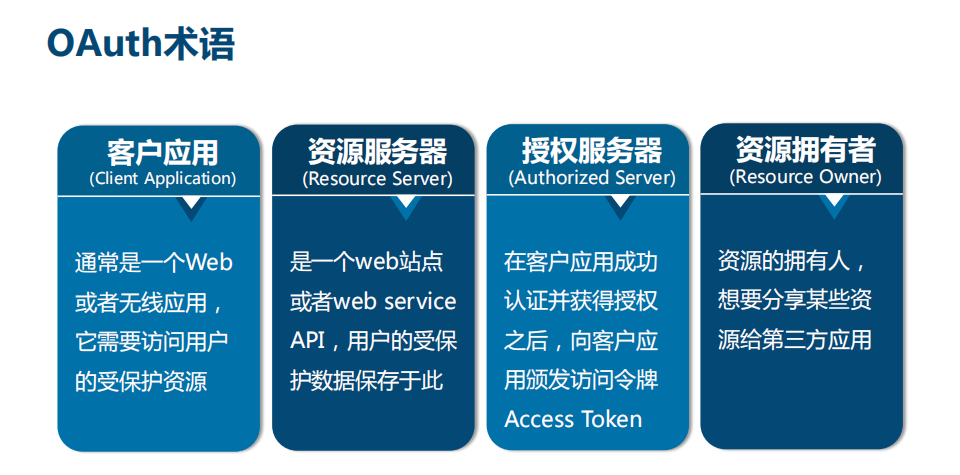
#### 5、OAuth2的不足



#### 6、Auth2涉及的角色



#### 7、OAuth2术语

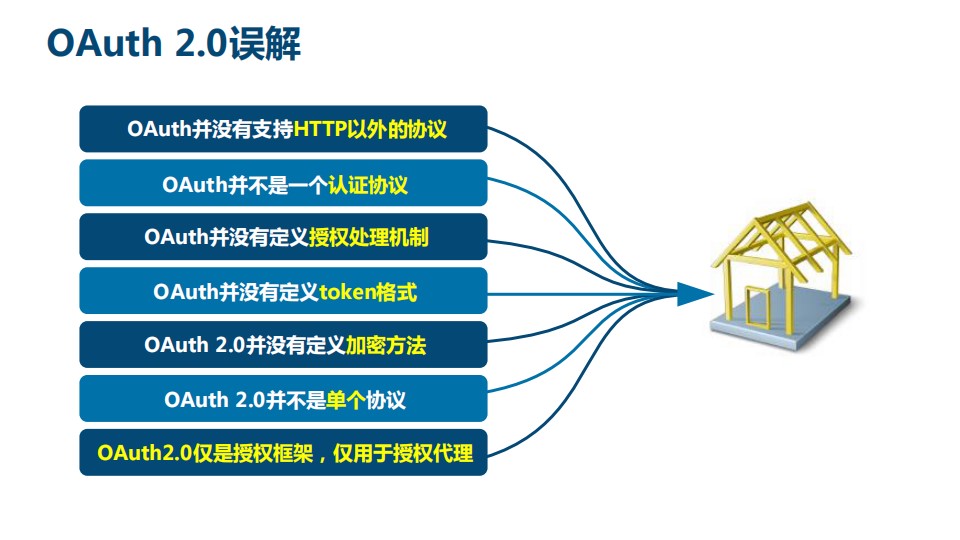




#### 8、OAuth2令牌的类型



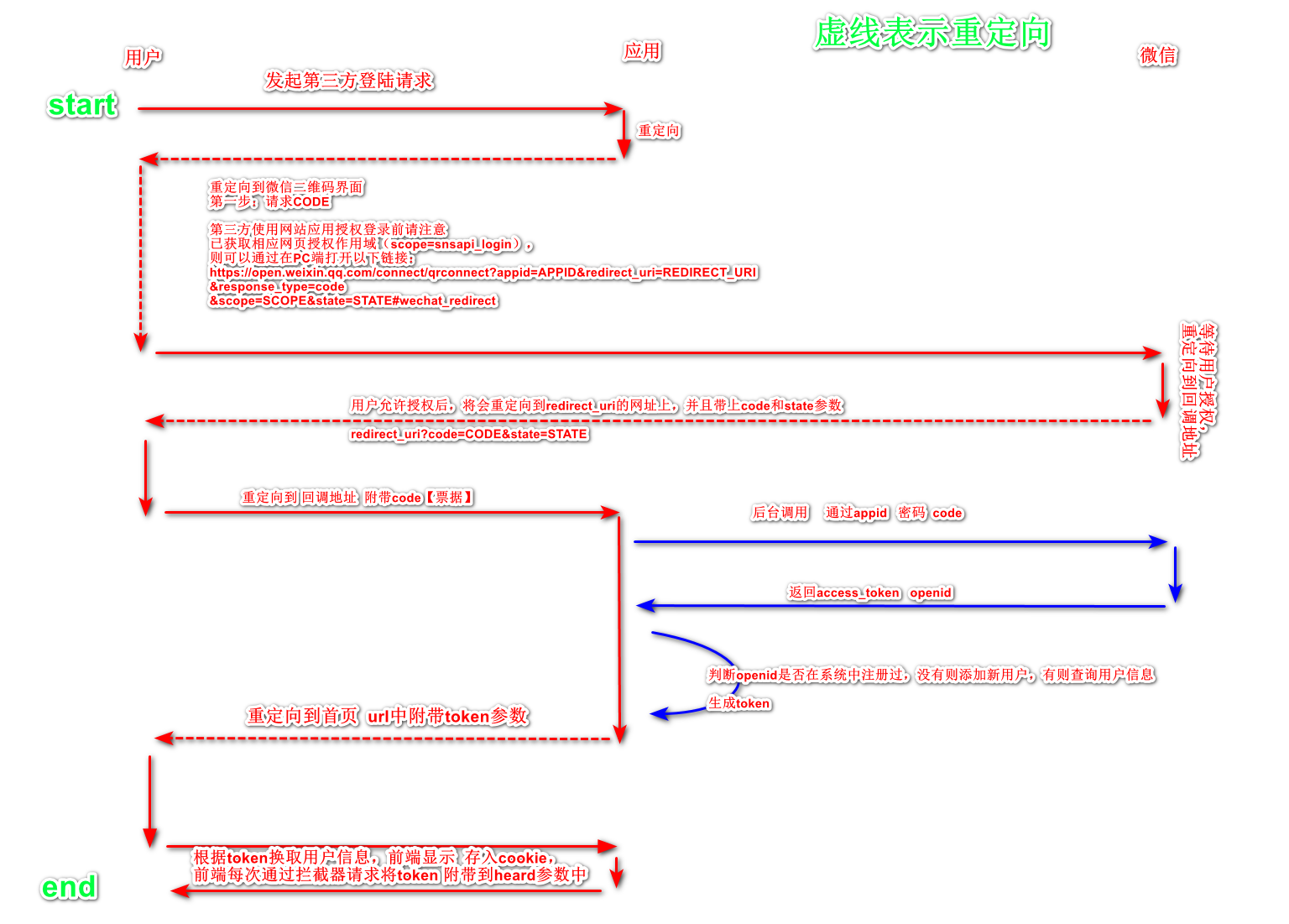
#### 9、OAuth2的误解



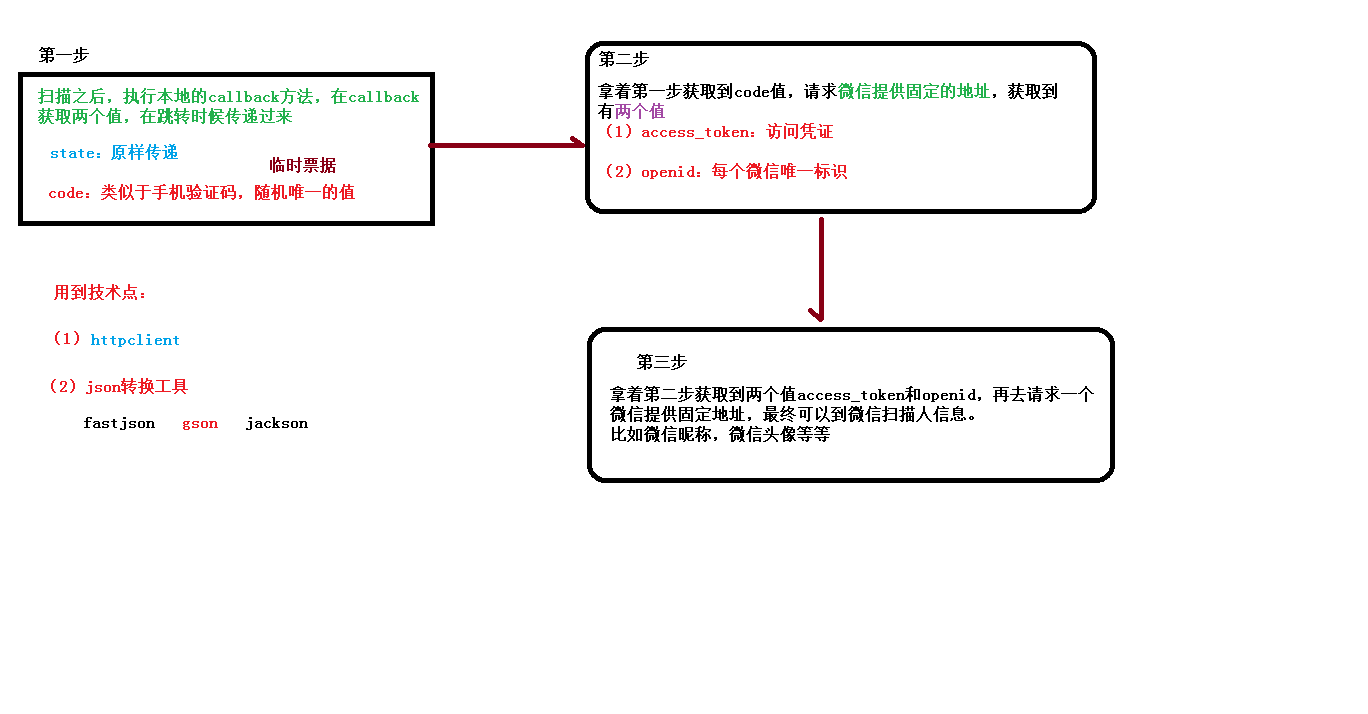
### 二、回顾



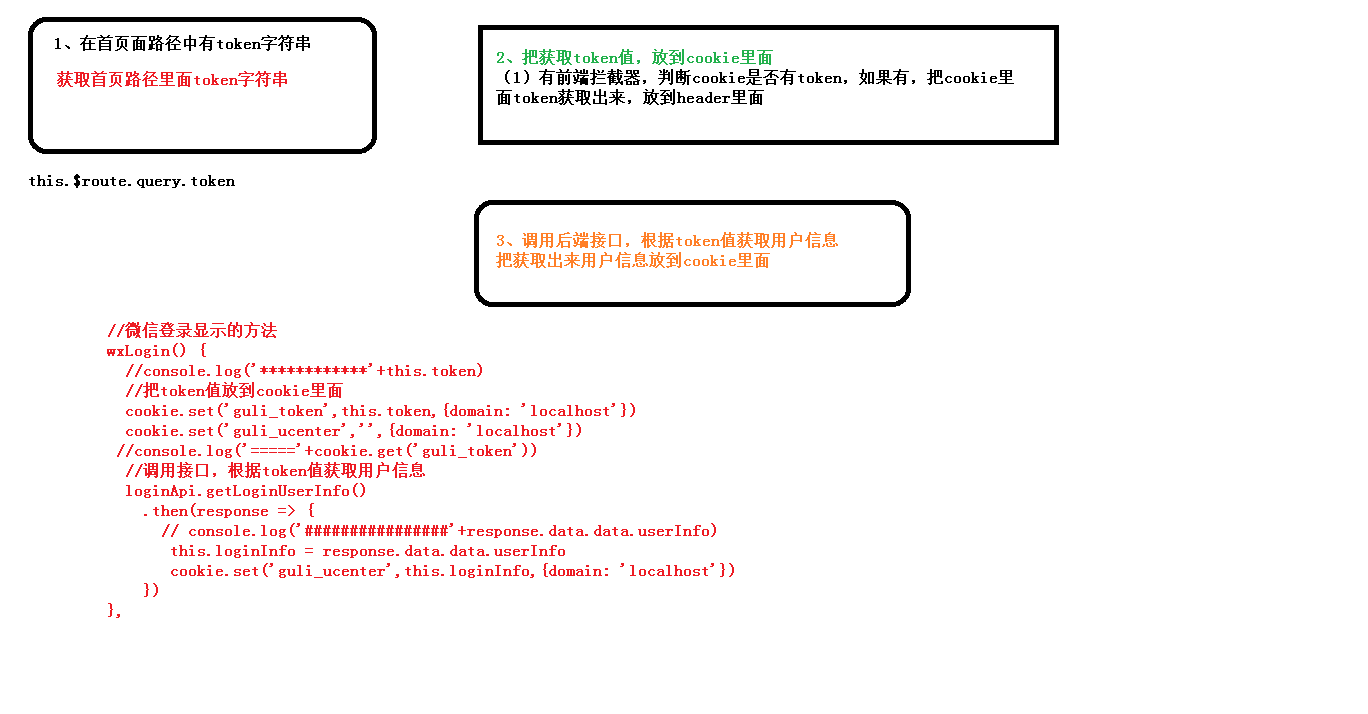
# 2 微信扫描登录

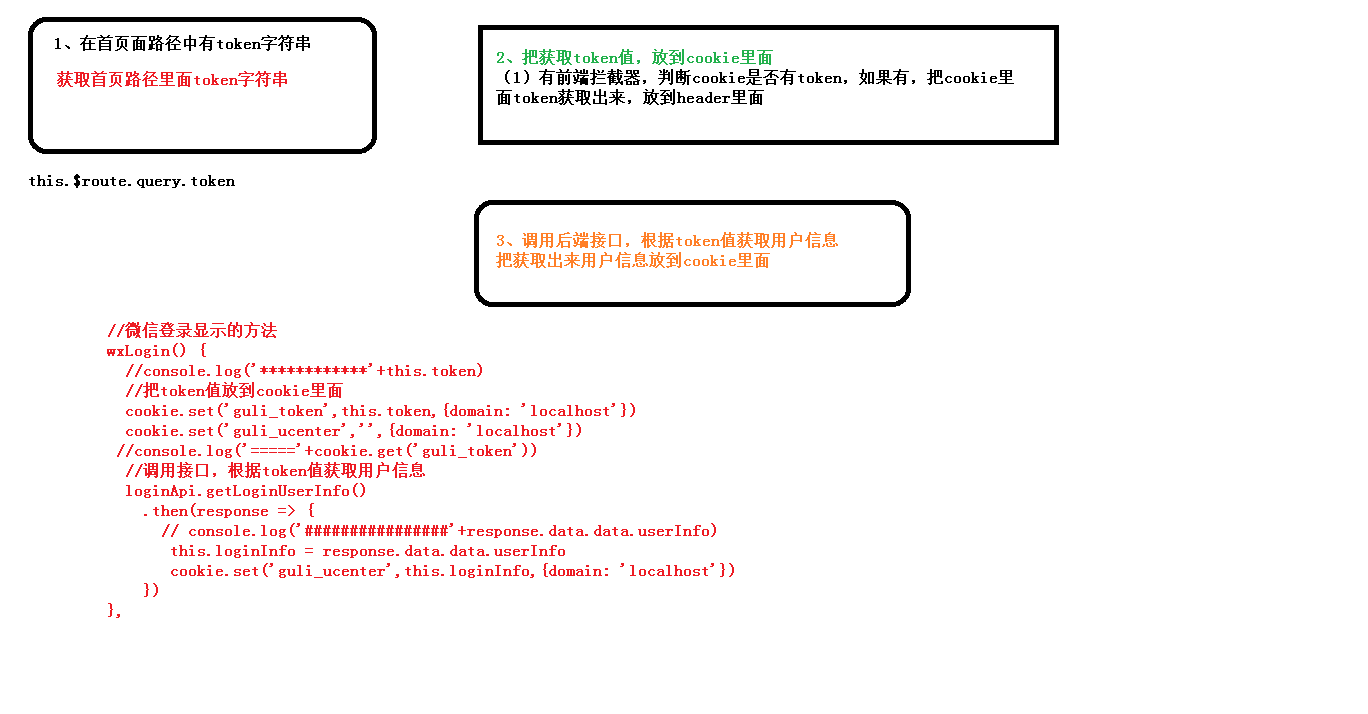












## 生成授权URL

### 一、准备工作

<https://open.weixin.qq.com>

#### 1、注册

#### 2、邮箱激活

#### 3、完善开发者资料

#### 4、开发者资质认证

准备营业执照，1-2个工作日审批、300元

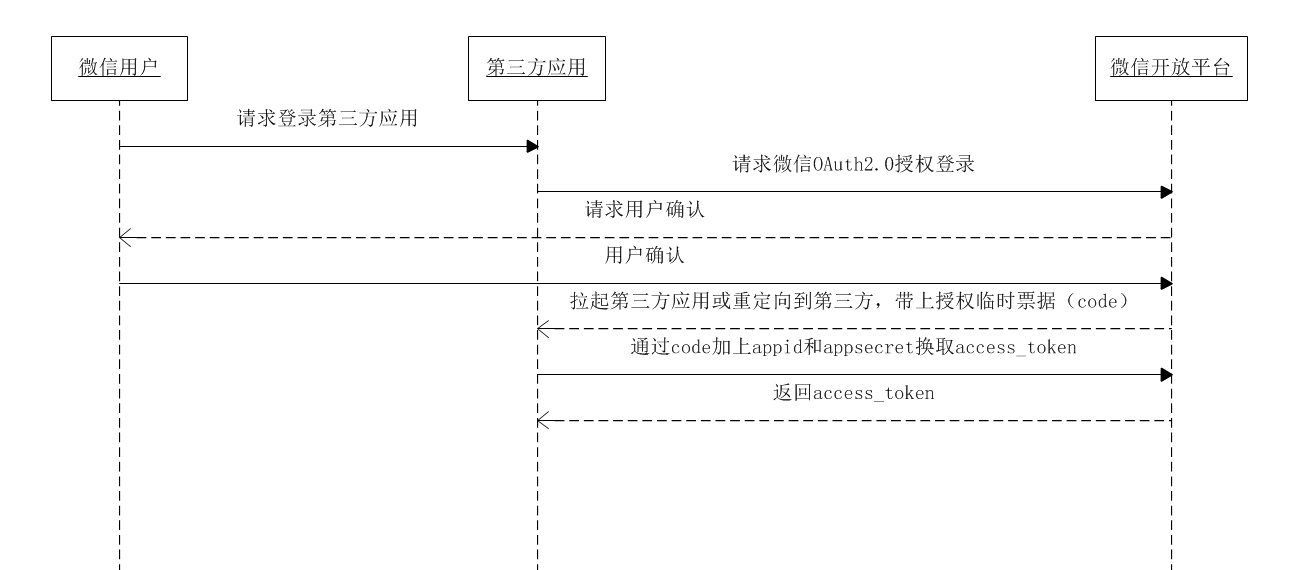
#### 5、创建网站应用

提交审核，7个工作日审批

#### 6、熟悉微信登录流程

参考文档：<https://open.weixin.qq.com/cgi-bin/showdocument?action=dir_list&t=resource/res_list&verify=1&id=open1419316505&token=e547653f995d8f402704d5cb2945177dc8aa4e7e&lang=zh_CN>

**获取access\_token时序图**



### 二、后端开发

#### 1、添加配置

application.properties添加相关配置信息

# 微信开放平台 appid

wx.open.app\_id=你的appid  
# 微信开放平台 appsecret  
wx.open.app\_secret=你的appsecret  
# 微信开放平台 重定向url  
wx.open.redirect\_url=http://你的服务器名称/api/ucenter/wx/callback

#### 2、创建常量类

创建util包，创建ConstantPropertiesUtil.java常量类

package com.guli.ucenter.util;

@Component  
//@PropertySource("classpath:application.properties")  
public class ConstantPropertiesUtil implements InitializingBean *{* @Value*(*"${wx.open.app\_id}")  
 private String appId;  
 @Value("${wx.open.app\_secret}")  
 private String appSecret;  
 @Value("${wx.open.redirect\_url}")  
 private String redirectUrl;  
 public static String WX\_OPEN\_APP\_ID;  
 public static String WX\_OPEN\_APP\_SECRET;  
 public static String WX\_OPEN\_REDIRECT\_URL;  
  
 @Override  
 public void afterPropertiesSet() throws Exception {  
 WX\_OPEN\_APP\_ID = appId;  
 WX\_OPEN\_APP\_SECRET = appSecret;  
 WX\_OPEN\_REDIRECT\_URL = redirectUrl;  
 }  
}

#### 3、创建controller

guli-microservice-ucenter微服务中创建api包

api包中创建WxApiController

package com.guli.ucenter.controller.api;

@CrossOrigin  
@Controller//注意这里没有配置 @RestController  
@RequestMapping("/api/ucenter/wx")  
public class WxApiController {  
 @GetMapping("login")  
 public String genQrConnect(HttpSession session) {  
 // 微信开放平台授权baseUrl  
 String baseUrl = "https://open.weixin.qq.com/connect/qrconnect" +  
 "?appid=%s" +  
 "&redirect\_uri=%s" +  
 "&response\_type=code" +  
 "&scope=snsapi\_login" +  
 "&state=%s" +  
 "#wechat\_redirect";  
 // 回调地址  
 String redirectUrl = ConstantPropertiesUtil.WX\_OPEN\_REDIRECT\_URL; //获取业务服务器重定向地址  
 try {  
 redirectUrl = URLEncoder.encode(redirectUrl, "UTF-8"); //url编码  
 } catch (UnsupportedEncodingException e) {  
 throw new GuliException(20001, e.getMessage());  
 }  
 // 防止csrf攻击（跨站请求伪造攻击）  
 //String state = UUID.randomUUID().toString().replaceAll("-", "");//一般情况下会使用一个随机数  
 String state = "imhelen";//为了让大家能够使用我搭建的外网的微信回调跳转服务器，这里填写你在ngrok的前置域名  
 System.out.println("state = " + state);  
 // 采用redis等进行缓存state 使用sessionId为key 30分钟后过期，可配置  
 //键："wechar-open-state-" + httpServletRequest.getSession().getId()  
 //值：satte  
 //过期时间：30分钟  
  
 //生成qrcodeUrl  
 String qrcodeUrl = String.format(  
 baseUrl,  
 ConstantPropertiesUtil.WX\_OPEN\_APP\_ID,  
 redirectUrl,  
 state);  
 return "redirect:" + qrcodeUrl;//重定向到微信二维码扫描地址进行扫描登陆，登陆成功后通过回调请求到自己的网站接口并附带code  
 }  
}

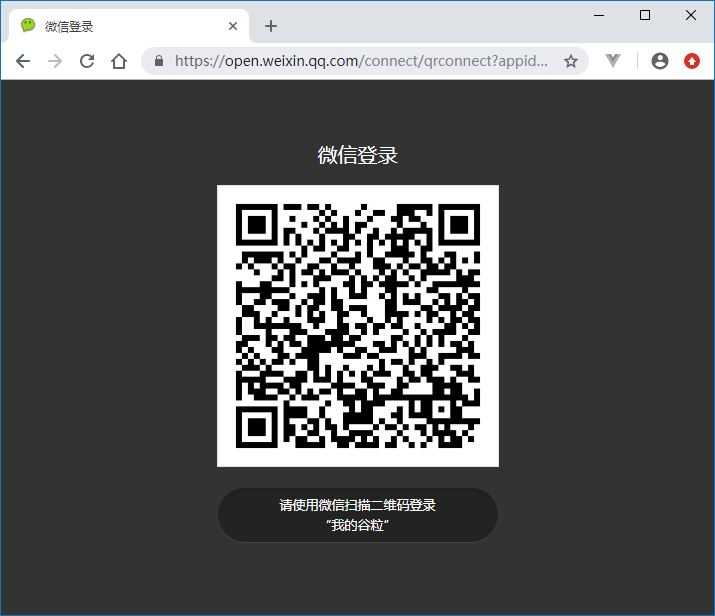
授权url参数说明

| **参数** | **是否必须** | **说明** |
| --- | --- | --- |
| appid | 是 | 应用唯一标识 |
| redirect\_uri | 是 | 请使用urlEncode对链接进行处理 |
| response\_type | 是 | 填code |
| scope | 是 | 应用授权作用域，拥有多个作用域用逗号（,）分隔，网页应用目前仅填写snsapi\_login即 |
| state | 否 | 用于保持请求和回调的状态，授权请求后原样带回给第三方。该参数可用于防止csrf攻击（跨站请求伪造攻击），建议第三方带上该参数，可设置为简单的随机数加session进行校验 |

#### 4、测试

访问：[http://localhost:8201/api/ucenter/wx/login](http://localhost:8104/api/ucenter/wx/login)

访问授权url后会得到一个微信登录二维码



用户扫描二维码会看到确认登录的页面



用户点击“确认登录”后，微信服务器会向谷粒学院的业务服务器发起回调，因此接下来我们需要开发回调controller

**注意：**如果没有正确的配置业务服务器的回调url，则会看到以下错误提示



## 开发回调URL

### 准备工作



#### 1、全局配置的跳转路径

# 微信开放平台 重定向url

wx.open.redirect\_url=http://回调地址/api/ucenter/wx/callback

#### 2、修改当前项目启动端口号为8150

#### 3、测试回调是否可用

在WxApiController中添加方法

@GetMapping("callback")

public String callback(String code, String state, HttpSession session) {

//得到授权临时票据code

System.out.println("code = " + code);

System.out.println("state = " + state);

}

### 二、后台开发

#### 1、添加依赖

<!--httpclient-->

*<*dependency*>  
 <*groupId*>*org.apache.httpcomponents*</*groupId*>  
 <*artifactId*>*httpclient*</*artifactId*>  
</*dependency*>*<!--commons-io-->  
*<*dependency*>  
 <*groupId*>*commons-io*</*groupId*>  
 <*artifactId*>*commons-io*</*artifactId*>  
</*dependency*>*<!--gson-->  
*<*dependency*>  
 <*groupId*>*com.google.code.gson*</*groupId*>  
 <*artifactId*>*gson*</*artifactId*>  
</*dependency*>*

#### 2、创建httpclient工具类

放入util包

HttpClientUtils.java

#### 3、创建回调controller方法

在WxApiController.java中添加如下方法

*/\*\**

*\** ***@param*** *code  
 \** ***@param*** *state  
 \** ***@return*** *\*/*@GetMapping*(*"callback"*)*public String callback*(*String code, String state*) {* //得到授权临时票据code  
 System.*out*.println*(*code*)*;  
 System.*out*.println*(*state*)*;  
 //从redis中将state获取出来，和当前传入的state作比较  
 //如果一致则放行，如果不一致则抛出异常：非法访问  
 //向认证服务器发送请求换取access\_token  
 String baseAccessTokenUrl = "https://api.weixin.qq.com/sns/oauth2/access\_token" +  
 "?appid=%s" +  
 "&secret=%s" +  
 "&code=%s" +  
 "&grant\_type=authorization\_code";  
 String accessTokenUrl = String.*format(*baseAccessTokenUrl,  
 ConstantPropertiesUtil.*WX\_OPEN\_APP\_ID*,  
 ConstantPropertiesUtil.*WX\_OPEN\_APP\_SECRET*,  
 code*)*;  
 String result = null;  
 try *{* result = HttpClientUtils.*get(*accessTokenUrl*)*;  
 System.*out*.println*(*"accessToken=============" + result*)*;  
 *}* catch *(*Exception e*) {* throw new GuliException*(*20001, "获取access\_token失败"*)*;  
 *}* //解析json字符串  
 Gson gson = new Gson*()*;  
 HashMap map = gson.fromJson*(*result, HashMap.class*)*;  
 String accessToken = *(*String*)* map.get*(*"access\_token"*)*;  
 String openid = *(*String*)* map.get*(*"openid"*)*;  
 //查询数据库当前用用户是否曾经使用过微信登录  
 Member member = memberService.getByOpenid*(*openid*)*;  
 if *(*member == null*) {* System.*out*.println*(*"新用户注册"*)*;  
 //访问微信的资源服务器，获取用户信息  
 String baseUserInfoUrl = "https://api.weixin.qq.com/sns/userinfo" +  
 "?access\_token=%s" +  
 "&openid=%s";  
 String userInfoUrl = String.*format(*baseUserInfoUrl, accessToken, openid*)*;  
 String resultUserInfo = null;  
 try *{* resultUserInfo = HttpClientUtils.*get(*userInfoUrl*)*;  
 System.*out*.println*(*"resultUserInfo==========" + resultUserInfo*)*;  
 *}* catch *(*Exception e*) {* throw new GuliException*(*20001, "获取用户信息失败"*)*;  
 *}* //解析json  
 HashMap*<*String, Object*>* mapUserInfo = gson.fromJson*(*resultUserInfo, HashMap.class*)*;  
 String nickname = *(*String*)* mapUserInfo.get*(*"nickname"*)*;  
 String headimgurl = *(*String*)* mapUserInfo.get*(*"headimgurl"*)*;  
 //向数据库中插入一条记录  
 member = new Member*()*;  
 member.setNickname*(*nickname*)*;  
 member.setOpenid*(*openid*)*;  
 member.setAvatar*(*headimgurl*)*;  
 memberService.save*(*member*)*;  
 *}* //*TODO 登录* return "redirect:http://localhost:3000";  
*}*

#### 4、业务层

业务接口：MemberService.java

Member getByOpenid(String openid);

业务实现：MemberServiceImpl.java

@Override

public Member getByOpenid*(*String openid*) {* QueryWrapper*<*Member*>* queryWrapper = new QueryWrapper*<*>*()*;  
 queryWrapper.eq*(*"openid", openid*)*;  
 Member member = baseMapper.selectOne*(*queryWrapper);  
 return member;  
*}*

## 返回JWT令牌

### 一、整合JWT令牌

#### 1、callback中生成jwt

在WxApiController.java的callback方法的最后添加如下代码

// *生成jwt*

String token = JwtUtils.*getJwtToken(*member.getId*()*, member.getNickname*())*;  
//存入cookie  
//CookieUtils.setCookie(request, response, "guli\_jwt\_token", token);  
//因为端口号不同存在跨域问题，cookie不能跨域，所以这里使用url重写  
return "redirect:http://localhost:3000?token=" + token;

#### 2、前端打印token

在layout/defaullt.vue中打印获取的token值

export default {

created() {

console.log(this.$route.query.token)

}

}

## 04-前端整合和显示用户登录信息

### 1、修改default.vue页面脚本

export default {

  data() {

    return {

      token: '',

      loginInfo: {

        id: '',

        age: '',

        avatar: '',

        mobile: '',

        nickname: '',

        sex: ''

      }

    }

  },

  created() {

    this.token = this.$route.query.token;//获取url中token值

    if (this.token) {

      this.wxLogin()

    }

    this.showInfo()

  },

  methods: {

    showInfo() {

      //debugger

*var* jsonStr = cookie.get("guli\_ucenter");

      if (jsonStr) {

        this.loginInfo = JSON.parse(jsonStr)

      }

    },

    logout() {

      //debugger

      cookie.set('guli\_ucenter', "", {domain: 'localhost'})

      cookie.set('guli\_token', "", {domain: 'localhost'})

      //跳转页面

      window.location.href = "/"

    },

    wxLogin() {

      if (this.token == '') return

      //把token存在cookie中、也可以放在localStorage中

      cookie.set('guli\_token', this.token, {domain: 'localhost'})

      cookie.set('guli\_ucenter', '', {domain: 'localhost'})

      //登录成功根据token获取用户信息

      userApi.getLoginInfo().then(*response* *=>* {//拦截器会自动把token存入header中

        this.loginInfo = *response*.data.data.item

        //将用户信息记录cookie

        cookie.set('guli\_ucenter', this.loginInfo, {domain: 'localhost'})

      })

    }

  }

}