微信授权登录

参考：慕课网 - 微信授权登录
制作日期：2018-7-18 - 2018-07-19
制作人：小桅[yw_forgit@163.com]

1、登录方式

微信公众平台（微信公众号）
微信开放平台

两种登录方式：

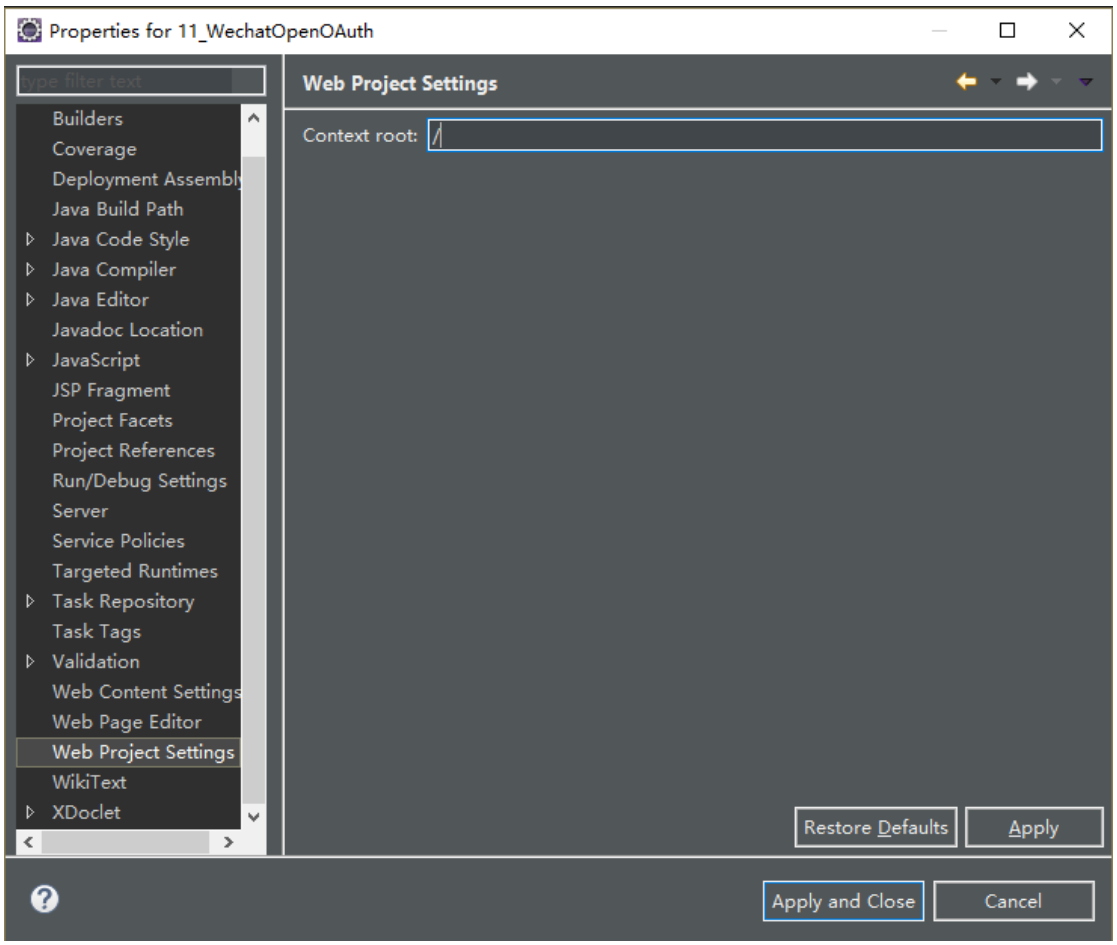
1）没有自己的账号体系，直接拉取微信用户信息来进行网站登录
下面只实现这一种

2）有自己的账号体系，授权成功后需要绑定自己的账号
这种与第一种一样，只是最后加上账号绑定而已

2、基于微信公众号授权登录

新建新项目，web2.5

改部署到根目录，不改也可以



授权回调页面域名

填写授权之后回调页面的域名（即回调的接口在该域名下）
测试号的填写：

订阅号等的填写，了解一下即可

网页服务	网页授权	网页授权获取用户基本信息		未获得
	基础接口	判断当前客户端版本是否支持指定JS接口	无上限	已获得
		获取jsapi_ticket	0/1000000	已获得
	分享接口	获取"分享到朋友圈"按钮点击状态及自定义分享内容接口		未获得
		获取"分享给朋友"按钮点击状态及自定义分享内容接口		未获得
		获取"分享到QQ"按钮点击状态及自定义分享内容接口		未获得
		获取"分享到腾讯微博"按钮点击状态及自定义分享内容接口		未获得
	图像接口	拍照或从手机相册中选图接口	无上限	已获得
		预览图片接口	无上限	已获得
		上传图片接口	无上限	已获得
		下载图片接口	无上限	已获得
	音频接口	开始录音接口	无上限	已获得
		停止录音接口	无上限	已获得
		播放语音接口	无上限	已获得
		暂停播放接口	无上限	已获得
		停止播放接口	无上限	已获得

点击“修改”，弹框

文档

选中菜单

关于特殊场景下的静默授权

1、上面已经提到，对于以snsapi_base为scope的网页授权，就静默授权的，用户无感知；

2、对于已关注公众号的用户，如果用户从公众号的会话或者自定义菜单进入本公众号的网页授权页，即使是scope为snsapi_userinfo，也是静默授权，用户无感知。

具体而言，网页授权流程分为四步：

1、引导用户进入授权页面同意授权，获取code

2、通过code换取网页授权access_token（与基础支持中的access_token不同）

3、如果需要，开发者可以刷新网页授权access_token，避免过期

4、通过网页授权access_token和openid获取用户基本信息（支持UnionID机制）

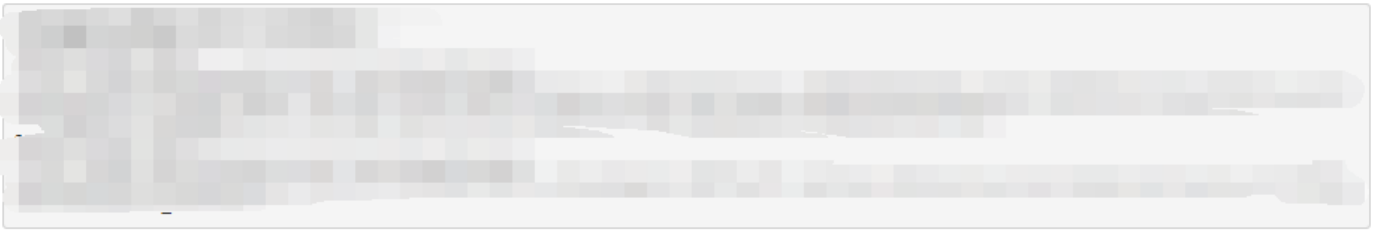
第一步：用户同意授权，获取code

在确保微信公众账号拥有授权作用域（scope参数）的权限的前提下（服务号获得高级接口后，默认拥有scope参数中的snsapi_base和snsapi_userinfo），引导关注者打开如下页面：

```
https://open.weixin.qq.com/connect/oauth2/authorize?appid=APPID&redirect_uri=REDIRECT_URI&response_type=code&scope=SCOPE&state=STATE#wechat_redirect
```

若提示“该链接无法访问”，请检查参数是否填写错误，是否拥有scope参数对应的授权作用域权限。

尤其注意：由于授权操作安全等级较高，所以在发起授权请求时，微信会对授权链接做正则强匹配校验，如果链接的参数顺序不对，授权页面将无法正常访问



尤其注意：跳转回调redirect_uri，应当使用https链接来确保授权code的安全性。

参数说明

参数	是否必须	说明
appid	是	公众号的唯一标识
redirect_uri	是	授权后重定向的回调链接地址， 请使用 urlEncode 对链接进行处理
response_type	是	返回类型，请填写code
scope	是	应用授权作用域，snsapi_base （不弹出授权页面，直接跳转，只能获取用户openid），snsapi_userinfo （弹出授权页面，可通过openid拿到昵称、性别、所在地。并且， 即使在未关注的情况下，只要用户授权，也能获取其信息 ）
state	否	重定向后会带上state参数，开发者可以填写a-zA-Z0-9的参数值，最多128字节
#wechat_redirect	是	无论直接打开还是做页面302重定向时候，必须带此参数

下图为scope等于snsapi_userinfo时的授权页面：

如果用户已经关注了，就不会弹框

用户同意授权后

如果用户同意授权，页面将跳转至 redirect_uri/?code=CODE&state=STATE。

```
code说明 ： code作为换取access_token的票据，每次用户授权带上的code将不一样，code只能使用一次，5分钟未被使用自动过期。
```

错误返回码说明如下：

返回码	说明
10003	redirect_uri域名与后台配置不一致
10004	此公众号被封禁
10005	此公众号并没有这些scope的权限
10006	必须关注此测试号
10009	操作太频繁了，请稍后重试
10010	scope不能为空
10011	redirect_uri不能为空
10012	appid不能为空
10013	state不能为空
10015	公众号未授权第三方平台，请检查授权状态
10016	不支持微信开放平台的Appid，请使用公众号Appid

第二步：通过code换取网页授权access_token

首先请注意，这里通过code换取的是一个特殊的网页授权access_token,与基础支持中的access_token（该access_token用于调用其他接口）不同。公众号可通过下述接口来获取网页授权access_token。如果网页授权的作用域为snsapi_base，则本步骤中获取到网页授权access_token的同时，也获取到了openid，snsapi_base式的网页授权流程即到此为止。

尤其注意：由于公众号的secret和获取到的access_token安全级别都非常高，必须只保存在服务器，不允许传给客户端。后续刷新access_token、通过access_token获取用户信息等步骤，也必须从服务器发起。

请求方法

```
获取code后，请求以下链接获取access_token:  https://api.weixin.qq.com/sns/oauth2/access_token?
appid=APPID&secret=SECRET&code=CODE&grant_type=authorization_code
```

参数说明

请求参数

参数	是否必须	说明
appid	是	公众号的唯一标识
secret	是	公众号的appsecret
code	是	填写第一步获取的code参数
grant_type	是	填写为authorization_code

返回说明

正确时返回的JSON数据包如下：

```
{ "access_token":"ACCESS_TOKEN",
"expires_in":7200,
"refresh_token":"REFRESH_TOKEN",
"openid":"OPENID",
"scope":"SCOPE" }
```

返回参数

参数	描述
access_token	网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
expires_in	access_token接口调用凭证超时时间，单位（秒）
refresh_token	用户刷新access_token
openid	用户唯一标识，请注意，在未关注公众号时，用户访问公众号的网页，也会产生一个用户和公众号唯一的OpenID
scope	用户授权的作用域，使用逗号（,）分隔

错误时微信会返回JSON数据包如下（示例为Code无效错误）：

```
{"errcode":40029,"errmsg":"invalid code"}
```

第三步：刷新access_token（如果需要） 我不做这步

由于access_token拥有较短的有效期，当access_token超时后，可以使用refresh_token进行刷新，refresh_token有效期为30天，当refresh_token失效之后，需要用户重新授权。

请求方法

```
获取第二步的refresh_token后，请求以下链接获取access_token：
https://api.weixin.qq.com/sns/oauth2/refresh_token?appid=APPID&grant_type=refresh_token&refresh_token=REFRESH_TOKEN
```

参数	是否必须	说明
appid	是	公众号的唯一标识
grant_type	是	填写为refresh_token
refresh_token	是	填写通过access_token获取到的refresh_token参数

返回说明

正确时返回的JSON数据包如下：

```
{ "access_token":"ACCESS_TOKEN",
"expires_in":7200,
"refresh_token":"REFRESH_TOKEN",
"openid":"OPENID",
"scope":"SCOPE" }
```

参数	描述
access_token	网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
expires_in	access_token接口调用凭证超时时间，单位（秒）
refresh_token	用户刷新access_token
openid	用户唯一标识
scope	用户授权的作用域，使用逗号（,）分隔

错误时微信会返回JSON数据包如下（示例为code无效错误）：

```
{"errcode":40029,"errmsg":"invalid code"}
```

unionid 是判断是否是同一个用户的关键标志，但必须绑定微信开放平台，不管是公众号、移动应用、网站应用等 unionid 都是一样的。

第四步：拉取用户信息(需scope为 snsapi_userinfo)

如果网页授权作用域为snsapi_userinfo，则此时开发者可以通过access_token和openid拉取用户信息了。

请求方法

```
http: GET（请使用https协议） https://api.weixin.qq.com/sns/userinfo?access_token=ACCESS_TOKEN&openid=OPENID&lang=zh_CN
```

参数说明

请求参数

参数	描述
access_token	网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
openid	用户的唯一标识
lang	返回国家地区语言版本，zh_CN 简体，zh_TW 繁体，en 英语

返回说明

正确时返回的JSON数据包如下：

```
{    "openid":" OPENID",
" nickname": NICKNAME,
"sex":"1",
"province":"PROVINCE"
"city":"CITY",
"country":"COUNTRY",
"headimgurl":
"http://thirdwx.qlogo.cn/mmopen/g3MonUZtNHkdmzicIlibx6iaFqAc56vxLSUfpb6n5WKSYVY0ChQKkiaJSgQ1dZuTOgvLLrhJbERQQ4eMsv84eavHiaiceqxibJxCfHe/46",
"privilege":[ "PRIVILEGE1" "PRIVILEGE2"     ],
"unionid": "o6_bmasdasdsad6_2sgVt7hMZOPfL"
}
```

返回参数

参数	描述
openid	用户的唯一标识
nickname	用户昵称
sex	用户的性别，值为1时是男性，值为2时是女性，值为0时是未知
province	用户个人资料填写的省份
city	普通用户个人资料填写的城市
country	国家，如中国为CN
headimgurl	用户头像，最后一个数值代表正方形头像大小（有0、46、64、96、132数值可选，0代表640*640正方形头像），用户没有头像时该项为空。若用户更换头像，原有头像URL将失效。
privilege	用户特权信息，json 数组，如微信沃卡用户为（chinaunicom）
unionid	只有在用户将公众号绑定到微信开放平台帐号后，才会出现该字段。

只要绑定了同一个开放平台的公众号（一个或多个）获取到的unionid都是一样的

错误时微信会返回JSON数据包如下（示例为openid无效）：

```
{"errcode":40003,"errmsg":" invalid openid "}
```

这个没用到

附：检验授权凭证（access_token）是否有效

请求方法

```
http: GET（请使用https协议） https://api.weixin.qq.com/sns/auth?access_token=ACCESS_TOKEN&openid=OPENID
```

参数说明

参数	描述
access_token	网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
openid	用户的唯一标识

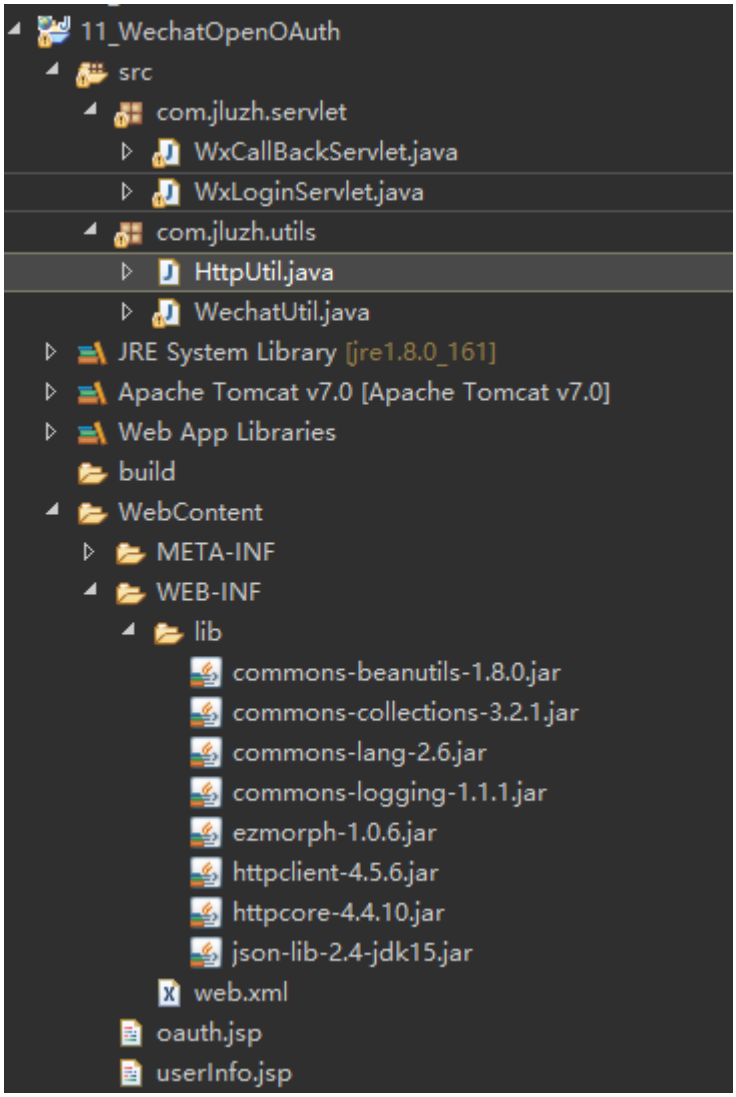
返回说明
正确的JSON返回结果：

```
{ "errcode":0,"errmsg":"ok"}
```

错误时的JSON返回示例：

```
{ "errcode":40003,"errmsg":"invalid openid"}
```

代码



web.xml 默认

web.xml | WxLoginServlet.java | WxCallBackServlet.java | WechatUtil.java | HttpUtil.java | oauth.jsp | userInfo.jsp

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">
    <display-name>11_WechatOpenOAuth</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

引导用户进入授权页面，直接重定向到指定的 url 即可

WxLoginServlet.java | WxCallBackServlet.java | WechatUtil.java | HttpUtil.java | oauth.jsp | userInfo.jsp

```java
package com.jluzh.servlet;

import java.io.IOException;

/**
 * 直接使用注解（servlet3.0特性），无需在web.xml文件配置映射等，如果在web.xml有配置则注解无效，但是别忘了要继承HttpServlet
 * @author WIN
 *
 */
@WebServlet("/wxLogin")
public class WxLoginServlet extends HttpServlet {

    @Override
    /**
     * 由我们的页面发送get请求，来到这里
     */
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 1、引导用户进入授权页面同意授权
        String oAuthUrl = WechatUtil.guideOAuthUrl();
        // 直接重定向给微信，微信会弹出授权确认框给用户确认授权，用户同意，微信就会重定向到我们的回调url（REDIRECT_URI）
        resp.sendRedirect(oAuthUrl);
    }

}
```

用户同意授权（或者静默授权）回调的接口

WxLoginServlet.java | WxCallBackServlet.java | WechatUtil.java | HttpUtil.java | oauth.jsp | userInfo.jsp

```java
package com.jluzh.servlet;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.jluzh.utils.WechatUtil;

import net.sf.json.JSONObject;

@WebServlet("/wxCallBack")
public class WxCallBackServlet extends HttpServlet {

    @Override
    /**
     * 回调ur（http://test.mykjianhu.com/wxCallBack）是微信重定向过来的，而重定向都是get请求，故到这里
     */
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 获取用户授权之后的code。如果用户同意授权，页面将跳转至 redirect_uri/?code=CODE&state=STATE。
        String code = req.getParameter("code");
        // 2、通过code换取网页授权access_token
        JSONObject tokenInfo = WechatUtil.getAccessToken(code);
        String accessToken = tokenInfo.getString("access_token"); // 网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
        String openid = tokenInfo.getString("openid"); // 用户唯一标识，请注意，在未关注公众号时，用户访问公众号的网页，也会产生一个用户和公众号唯一的OpenID

        // 4、通过网页授权access_token和openid获取用户基本信息
        JSONObject userInfo = WechatUtil.getUserInfo(accessToken, openid);
        req.setAttribute("userInfo", userInfo);
        // 转发
        req.getRequestDispatcher("/userInfo.jsp").forward(req, resp);
    }
}
```

WechatUtil.java

共有 4 个步骤，但是我们只做 3 步

WxLoginServlet.java | WxCallBackServlet.java | WechatUtil.java | HttpUtil.java | oauth.jsp | userInfo.jsp

```java
package com.jluzh.utils;

import java.io.IOException;
import java.net.URLEncoder;

import org.apache.http.client.ClientProtocolException;

import net.sf.json.JSONObject;

public class WechatUtil {
    /** 公众号的唯一标识 */
    private static final String APPID = "wx4[redacted]c8d";
    /** 密钥 */
    private static final String SECRET = "2a1c29[redacted]2102b";

    // >> 页面授权所需参数
    /** 授权后重定向的回调链接地址，请使用 urlEncode 对链接进行处理；
     * 并且需要在公众平台的接口权限设置“网页授权获取用户基本信息”，填写该回调用接口的域名；
     * 应当使用https确保授权code的安全性 */
    private static final String REDIRECT_URI = "http://test.r[redacted].com/wxCallBack";
    /** 应用授权作用域：
     * snsapi_base（不弹出授权页面，直接跳转，只能获取用户openid）；
     * snsapi_userinfo（弹出授权页面，可通过openid拿到昵称、性别、所在地。并且， 即使在未关注的情况下，只要用户授权，也能获取其信息 ）。
        关于特殊场景下的静默授权：
        1）上面已经提到，对于以snsapi_base为scope的网页授权，就静默授权的，用户无感知；
        2）对于已关注公众号的用户，如果用户从公众号的会话或者自定义菜单进入本公众号的网页授权页，即使是scope为snsapi_userinfo，也是静默授权，用户无感知。 */
    private static final String SCOPE = "snsapi_userinfo";
    /** 重定向后会带上state参数，开发者可以填写a-zA-Z0-9的参数值，最多128字节；
     * 据说这个参数很重要，见https://blog.csdn.net/weixin_37242696/article/details/80243325 */
    private static final String STATE = "test";
    /** 微信OAuth2.0网页授权的url（授权之后，可以获取到用户详细信息） */
    private static final String OAUTH_URL = "https://open.weixin.qq.com/connect/oauth2/authorize?appid=APPID&redirect_uri=REDIRECT_URI&response_type=code&scope=SCOPE&state=STATE#wechat_redirect";

    // >> 换取网页授权access_token所需参数
    /** 填写上一步获取的code参数 */
    private static String CODE = "";
    /** 换取网页授权access_token的url */
    private static final String ACCESS_TOKEN_URL = "https://api.weixin.qq.com/sns/oauth2/access_token?appid=APPID&secret=SECRET&code=CODE&grant_type=authorization_code";

    // >> 获取用户信息所需参数
    /** 上一步获取的access_token */
    private static String ACCESS_TOKEN = "";
    /** 上一步获取的用户的openid */
    private static String OPENID = "";
    /** 获取用户信息的url，要求使用get请求、https */
    private static final String USER_INFO_URL = "https://api.weixin.qq.com/sns/userinfo?access_token=ACCESS_TOKEN&openid=OPENID&lang=zh_CN";
```

```java
/*网页授权流程分为四步：
1、引导用户进入授权页面，用户同意授权，获取code
2、通过code换取网页授权access_token（与基础支持中的access_token(这token是获取素材等资源使用的)不同）
3、如果需要，开发者可以刷新网页授权access_token，避免过期（我们省略）
4、通过网页授权access_token和openid获取用户基本信息（需前面的scope为 snsapi_userinfo；支持UnionID机制）*/

/**
 * 1、引导用户进入授权页面（返回引导的url，引导在servlet层做）
 * @return
 * @throws IOException
 */
public static String guideOAuthUrl() throws IOException {
    System.out.println("1、引导用户进入授权页面开始");

    String url = OAUTH_URL.replace("APPID", APPID).replace("REDIRECT_URI", URLEncoder.encode(REDIRECT_URI)).replace("SCOPE", SCOPE).replace("STATE", STATE);
    return url;
}

/**
 * 2、通过code换取网页授权access_token（与基础支持中的access_token(这token是获取素材等资源使用的)不同）
 * @param code 用户同意授权，获取code，上一步直接可以从request获取到
 * @return 返回获取的所有结果
 * @throws ClientProtocolException
 * @throws IOException
 */
public static JSONObject getAccessToken(String code) throws ClientProtocolException, IOException {
    System.out.println("用户同意授权，获取到code=" + code);

    CODE = code; // 为了维护方法的接口性和所需参数的明显性，故皮一下
    String url = ACCESS_TOKEN_URL.replace("APPID", APPID).replace("SECRET", SECRET).replace("CODE", CODE);
    JSONObject jsonObj = HttpUtil.doGetJsonObj(url);

    System.out.println("2、通过code换取网页授权access_token成功。" + jsonObj);
    return jsonObj;
}

/**
 * 4、通过网页授权access_token和openid获取用户基本信息（需前面的scope为 snsapi_userinfo；支持UnionID机制）
 * @param accessToken
 * @param openid
 * @return
 * @throws ClientProtocolException
 * @throws IOException
 */
public static JSONObject getUserInfo(String accessToken, String openid) throws ClientProtocolException, IOException {
    ACCESS_TOKEN = accessToken;
    OPENID = openid;
    String url = USER_INFO_URL.replace("ACCESS_TOKEN", ACCESS_TOKEN).replace("OPENID", OPENID);
    JSONObject jsonObj = HttpUtil.doGetJsonObj(url);

    System.out.println("4、通过网页授权access_token和openid获取用户基本信息 成功。" + jsonObj);
    return jsonObj;
}
```

get 请求工具

```java
package com.jluzh.utils;

import java.io.IOException;

public class HttpUtil {

    /**
     * get请求，返回JsonObj格式的数据
     * @param url
     * @return
     * @throws ClientProtocolException
     * @throws IOException
     */
    public static JSONObject doGetJsonObj(String url) throws ClientProtocolException, IOException {
        JSONObject jsonObj = null;
        // 创建默认的
        CloseableHttpClient httpClient = HttpClients.createDefault();
        // get方法请求url
        HttpGet httpGet = new HttpGet(url);
        // 发送请求
        CloseableHttpResponse response = httpClient.execute(httpGet);
        // 获取请求返回的数据
        HttpEntity entity = response.getEntity();
        if(entity != null) {
            // 转成字符串，设置下编码，不然会乱码
            String result = EntityUtils.toString(entity, "UTF-8");
            // 转成json对象的形式，利于获取属性
            jsonObj = JSONObject.fromObject(result);
        }
        // 释放连接
        httpGet.releaseConnection();
        return jsonObj;
    }
}
```

准备引导进入授权的页面

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<!-- 移动端自动适应 -->
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<title>Insert title here</title>
</head>
<body>
<!-- 超链接的请求是get -->
<a href="/wxLogin" >微信授权登录</a>
</body>
</html>
```

授权成功，转发的页面

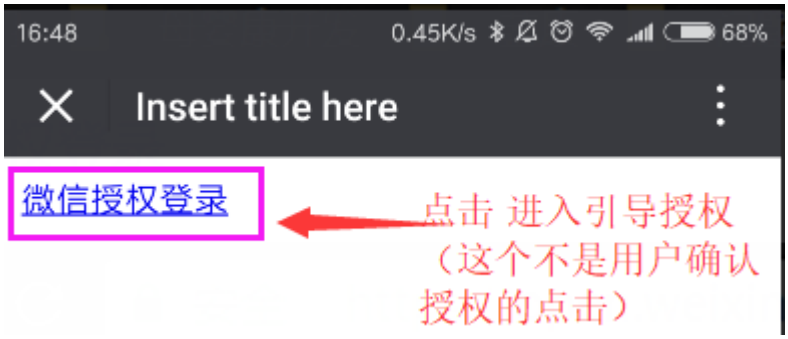
```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
${userInfo}
</body>
</html>
```

测试

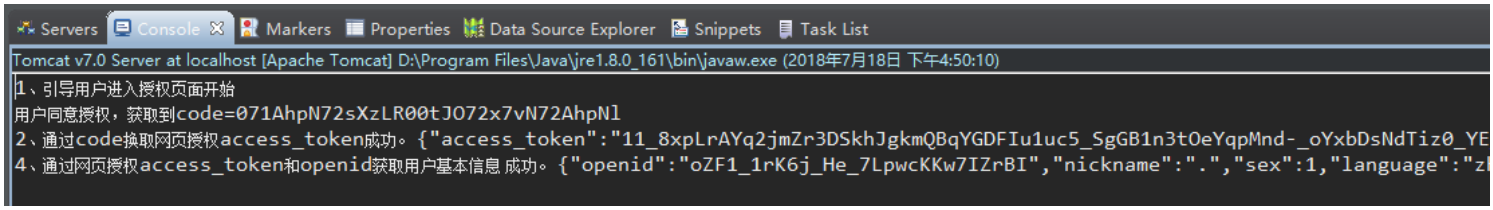
电脑端测试

提示要用微信打开

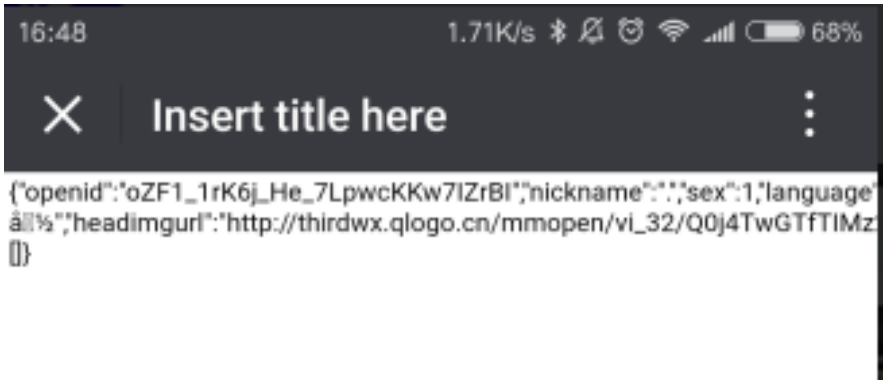
手机端打开，然后点击“微信授权登录”



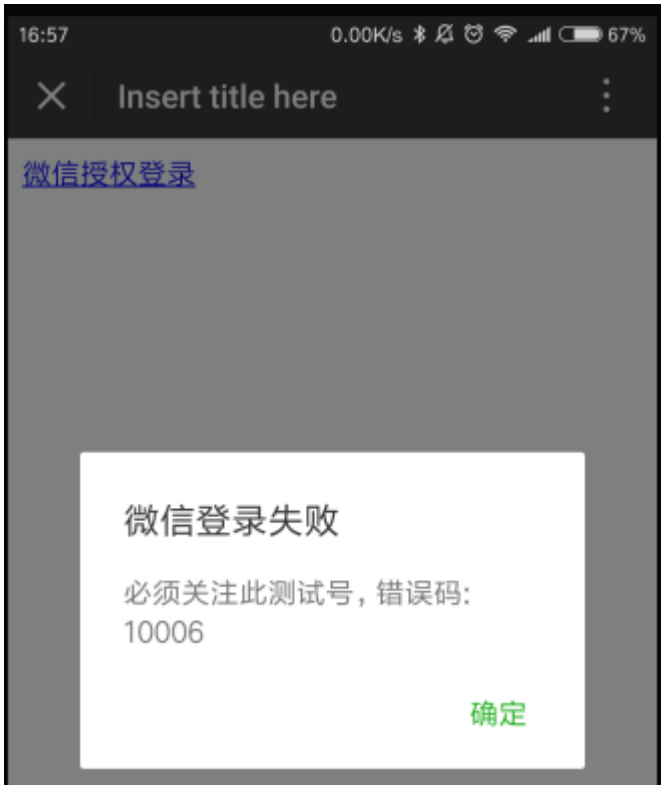
因为我已经关注了，故会静默授权，无需我确认，日志可以看到三步都走过了



然后页面跳转，显示用户信息



测试号、订阅号都无法测试弹框授权（未关注才会有弹框授权）的情况，如图，需要使用认证的服务号



3、基于微信开放平台实现授权登录

开发门槛比较高，必须是企业，个人无法开通开放平台。
步骤一样，只是链接可能会有点不一样（就引导授权的链接不一样）

1）网站应用

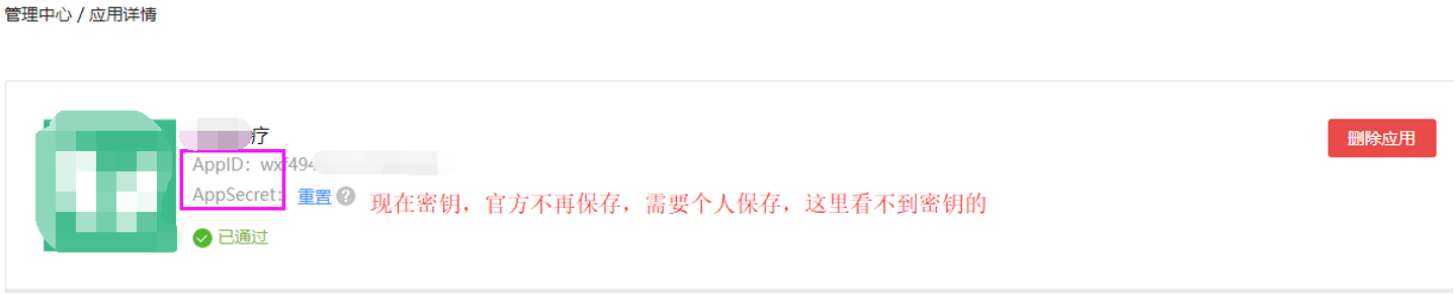
创建网站应用

登录微信开放平台，创建网站应用

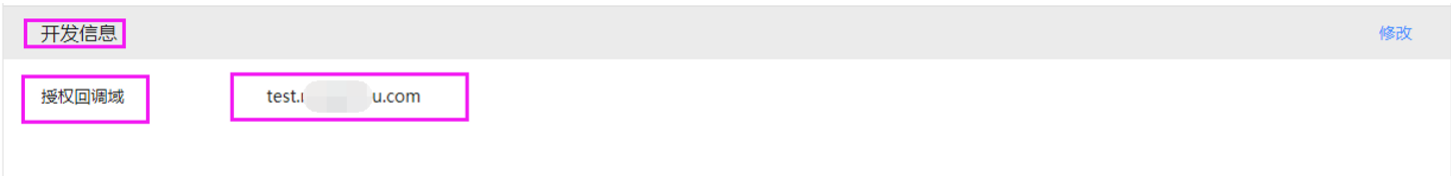
填写信息，这需要企业的信息，我是用了公司已有的网站应用进行测试

这个就是创建好的，点击 查看

获取查看 AppID 和 AppSecret（官方不保存）

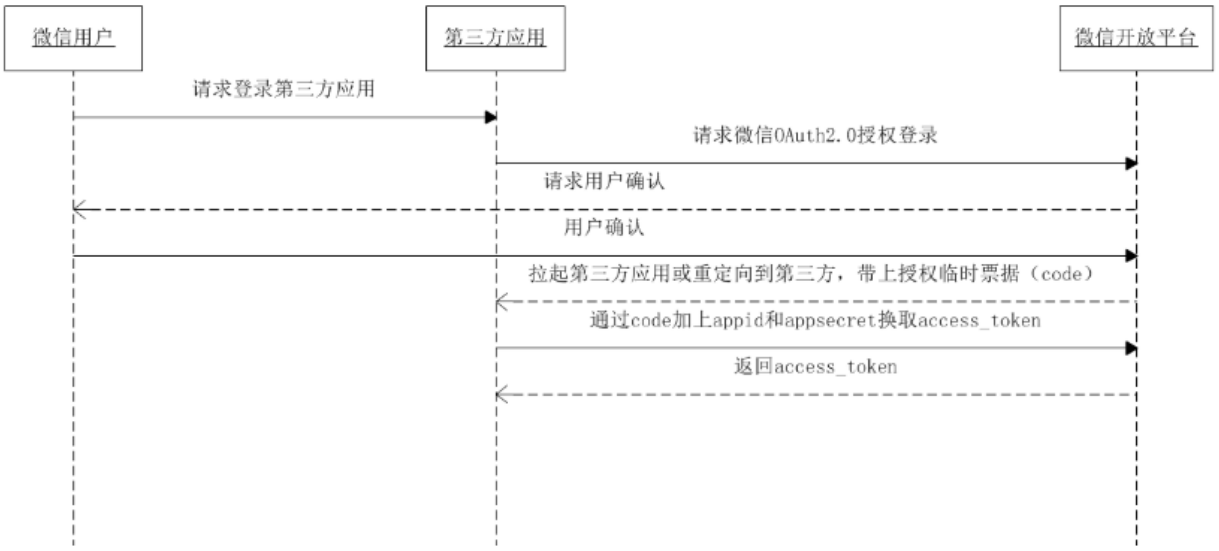


往下拉，修改授权回调域，即授权成功之后，回调的接口必须在该域名下，才能回调成功。



文档

获取access_token时序图：



第一步：请求CODE

第三方使用网站应用授权登录前请注意已获取相应网页授权作用域（scope=snsapi_login），则可以通过在PC端打开以下链接：

https://open.weixin.qq.com/connect/qrconnect?appid=APPID&redirect_uri=REDIRECT_URI&response_type=code&scope=SCOPE&state=STATE#wechat_redirect

若提示“该链接无法访问”，请检查参数是否填写错误，如redirect_uri的域名与审核时填写的授权域名不一致或scope不为snsapi_login。

参数说明

链接与公众号授权的不一样，参数也有不一样的地方

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

返回说明

用户允许授权后，将会重定向到redirect_uri的网址上，并且带上code和state参数

```
redirect_uri?code=CODE&state=STATE
```

若用户禁止授权，则重定向后不会带上code参数，仅会带上state参数

```
redirect_uri?state=STATE
```

第二步：通过code获取access_token

通过code获取access_token

```
https://api.weixin.qq.com/sns/oauth2/access_token?
appid=APPID&secret=SECRET&code=CODE&grant_type=authorization_code
```

参数说明 请求参数

参数	是否必须	说明
appid	是	应用唯一标识，在微信开放平台提交应用审核通过后获得
secret	是	应用密钥AppSecret，在微信开放平台提交应用审核通过后获得
code	是	填写第一步获取的code参数
grant_type	是	填authorization_code

返回说明

正确的返回： 返回参数

```
{
"access_token":"ACCESS_TOKEN",
"expires_in":7200,
"refresh_token":"REFRESH_TOKEN",
"openid":"OPENID",
"scope":"SCOPE",
"unionid": "o6_bmasdasdsad6_2sgVt7hMZOPfL"
}
```

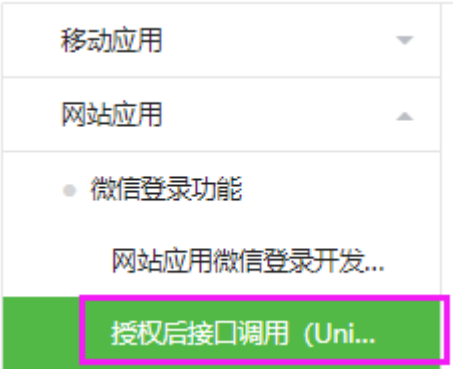
参数说明

参数	说明
access_token	接口调用凭证
expires_in	access_token接口调用凭证超时时间，单位（秒）
refresh_token	用户刷新access_token
openid	授权用户唯一标识
scope	用户授权的作用域，使用逗号（,）分隔 这个字段就是标志着同一个用户
unionid	当且仅当该网站应用已获得该用户的userinfo授权时，才会出现该字段。

错误返回样例：

```
{"errcode":40029,"errmsg":"invalid code"}
```

第三步：获取用户信息



获取用户个人信息（UnionID机制）

接口说明

此接口用于获取用户个人信息。开发者可通过OpenID来获取用户基本信息。特别需要注意的是，如果开发者拥有多个移动应用、网站应用和公众帐号，可通过获取用户基本信息中的unionid来区分用户的唯一性，因为只要是同一个微信开放平台帐号下的移动应用、网站应用和公众帐号，用户的unionid是唯一的。换句话说，同一用户，对同一个微信开放平台下的不同应用，unionid是相同的。请注意，在用户修改微信头像后，旧的微信头像URL将会失效，因此开发者应该自己在获取用户信息后，将头像图片保存下来，避免微信头像URL失效后的异常情况。

请求说明

```
http请求方式: GET
https://api.weixin.qq.com/sns/userinfo?access_token=ACCESS_TOKEN&openid=OPENID
```

参数说明

请求参数

参数	是否必须	说明
access_token	是	调用凭证
openid	是	普通用户的标识，对当前开发者帐号唯一
lang	否	国家地区语言版本，zh_CN 简体，zh_TW 繁体，en 英语，默认为zh-CN

返回说明

正确的Json返回结果：

```
{
"openid":"OPENID",
"nickname":"NICKNAME",
"sex":1,
"province":"PROVINCE",
"city":"CITY",
"country":"COUNTRY",
"headimgurl":
"http://wx.qlogo.cn/mmopen/g3MonUZtNHkdmzicIlibx6iaFqAc56vxLSUfpb6n5WKSYVY0ChQKkiaJSgQ1dZuTOgvLLrhJbERQQ4eMsv84eavHiaiceqxibJxCfHe/0",
"privilege":[
"PRIVILEGE1",
"PRIVILEGE2"
],
"unionid": " o6_bmasdasdsad6_2sgVt7hMZOPfL"

}
```

返回参数

参数	说明
openid	普通用户的标识，对当前开发者帐号唯一
nickname	普通用户昵称
sex	普通用户性别，1为男性，2为女性
province	普通用户个人资料填写的省份
city	普通用户个人资料填写的城市
country	国家，如中国为CN
headimgurl	用户头像，最后一个数值代表正方形头像大小（有0、46、64、96、132数值可选，0代表640*640正方形头像），用户没有头像时该项为空
privilege	用户特权信息，json数组，如微信沃卡用户为（chinaunicom）
unionid	用户统一标识。针对一个微信开放平台帐号下的应用，同一用户的unionid是唯一的。

如果用户改了头像，链接会失效，建议保存头像回本地

建议：

开发者最好保存用户unionID信息，以便以后在不同应用中进行用户信息互通。

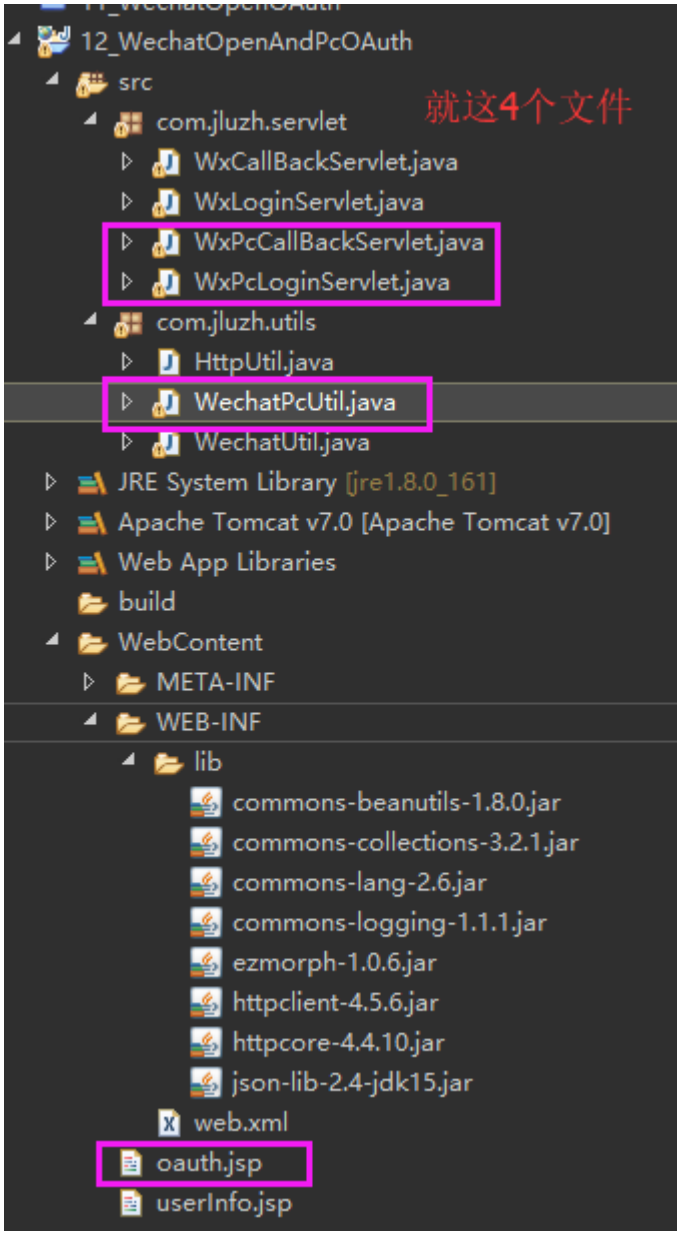
错误的Json返回示例：

```
{
"errcode":40003,"errmsg":"invalid openid"
}
```

代码

复制上一节的项目

注意名称都是带有 Pc 的哦，不要与上一节的代码搞混了，我就搞混过，报错 40029



WxPcLoginServlet.java | WechatPcUtil.java | WxPcCallBackServlet.java | oauth.jsp

```java
package com.jluzh.servlet;

import java.io.IOException;

/**
 * 直接使用注解（servlet3.0特性），无需在web.xml文件配置映射等，如果在web.xml有配置则注解无效，但是别忘了要继承HttpServlet
 * @author WIN
 *
 */
@WebServlet("/wxPcLogin")
public class WxPcLoginServlet extends HttpServlet {

    @Override
    /**
     * 由我们的页面发送get请求，来到这里
     */
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 1、请求CODE
        String oAuthUrl = WechatPcUtil.guideOAuthUrl();
        // 直接重定向给微信，微信会弹出授权确认框给用户确认授权，用户同意，微信就会重定向到我们的回调url（REDIRECT_URI）
        resp.sendRedirect(oAuthUrl);
    }

}
```

WxPcLoginServlet.java | WechatPcUtil.java | WxPcCallBackServlet.java | oauth.jsp

```java
package com.jluzh.utils;

import java.io.IOException;

/**
 * 开放平台的网站应用授权（这里简称PC）与公众号的类似，但是也有区别
 * @author WIN
 *
 */
public class WechatPcUtil {
    /** 开放平台 -- 网站应用 -- 的唯一标识 */
    private static final String APPID = "wxf49";
    /** 密钥 */
    private static final String SECRET = "8d8084f";

    // >> 1、请求CODE
    /** 授权后重定向的回调链接地址，请使用 urlEncode 对链接进行处理；
     * 并且需要在开放平台 -- 网站应用 点击该应用，填写 授权回调域 */
    private static final String REDIRECT_URI = "http://test.hu.com/wxPcCallBack";
    /** 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 */
    private static final String SCOPE = "snsapi_login";
    /** 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 */
    private static final String STATE = "test";
    /** 微信OAuth2.0第三方使用网站应用授权登录的url，链接与公众号的不一样 */
    private static final String OAUTH_URL = "https://open.weixin.qq.com/connect/qrconnect?appid=APPID&redirect_uri=REDIRECT_URI&response_type=code&scope=SCOPE&state=STATE#wechat_redirect";

    // >> 2、通过code获取access_token
    /** 填写上一步获取的code参数 */
    private static String CODE = "";
    /** 获取access_token的url，链接与公众号的一样 */
    private static final String ACCESS_TOKEN_URL = "https://api.weixin.qq.com/sns/oauth2/access_token?appid=APPID&secret=SECRET&code=CODE&grant_type=authorization_code";

    // >> 3、获取用户个人信息（UnionID机制）
    /** 上一步获取的access_token */
    private static String ACCESS_TOKEN = "";
    /** 上一步获取的用户的openid，虽然有UnionID（仅能用来判断是否是同一个微信用户），但是获取信息还是只能使用openid */
    private static String OPENID = "";
    /** 获取用户信息的url，要求使用get请求，链接与公众号的一样；
     * 请注意，在用户修改微信头像后，旧的微信头像URL将会失效，因此开发者应该自己在获取用户信息后，将头像图片保存下来，避免微信头像URL失效后的异常情况。 */
    private static final String USER_INFO_URL = "https://api.weixin.qq.com/sns/userinfo?access_token=ACCESS_TOKEN&openid=OPENID&lang=zh_CN";

    /**
     * 1、请求CODE
     * @return
     * @throws IOException
     */
    public static String guideOAuthUrl() throws IOException {
        System.out.println("1、请求CODE 开始");

        String url = OAUTH_URL.replace("APPID", APPID).replace("REDIRECT_URI", URLEncoder.encode(REDIRECT_URI)).replace("SCOPE", SCOPE).replace("STATE", STATE);
        return url;
    }
```

```java
/**
 * 2、通过code获取access_token（与基础支持中的access_token(这token是获取素材等资源使用的)不同）
 * @param code 用户同意授权，获取code，上一步直接可以从request获取到
 * @return 返回获取的所有结果
 * @throws ClientProtocolException
 * @throws IOException
 */
public static JSONObject getAccessToken(String code) throws ClientProtocolException, IOException {
    System.out.println("用户同意授权，获取到code=" + code);

    CODE = code; // 为了维护方法的接口性和所需参数的明显性，故废一下
    String url = ACCESS_TOKEN_URL.replace("APPID", APPID).replace("SECRET", SECRET).replace("CODE", CODE);
    JSONObject jsonObj = HttpUtil.doGetJsonObj(url);

    System.out.println("2、通过code获取access_token成功。" + jsonObj);
    return jsonObj;
}

/**
 * 3、获取用户个人信息（UnionID机制）
 * @param accessToken
 * @param openid
 * @return
 * @throws ClientProtocolException
 * @throws IOException
 */
public static JSONObject getUserInfo(String accessToken, String openid) throws ClientProtocolException, IOException {
    ACCESS_TOKEN = accessToken;
    OPENID = openid;
    String url = USER_INFO_URL.replace("ACCESS_TOKEN", ACCESS_TOKEN).replace("OPENID", OPENID);
    JSONObject jsonObj = HttpUtil.doGetJsonObj(url);

    System.out.println("3、获取用户个人信息（UnionID机制）成功。" + jsonObj);
    return jsonObj;
}
```

WxPcLoginServlet.java | WechatPcUtil.java | WxPcCallBackServlet.java | oauth.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<!-- 移动端自动适应 -->
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<title>Insert title here</title>
</head>
<body>
<!-- 超链接的请求是get -->
<a href="/wxLogin" >微信公众平台授权登录</a>
<br>
<a href="/wxPcLogin" >微信开放平台授权登录</a>
</body>
</html>
```

就这个

测试

电脑浏览器访问，点击 “微信开放平台授权登录”

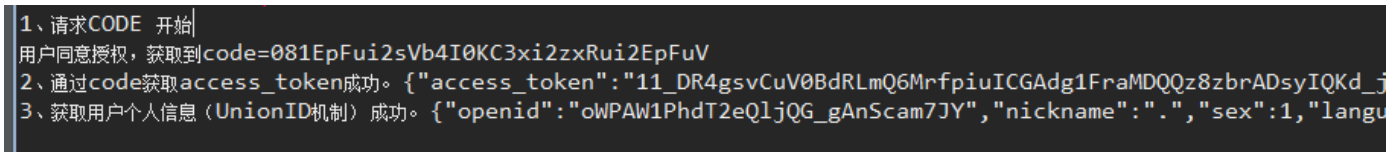
然后进入到微信授权页面

使用手机微信扫描上面的二维码，会弹出下面的界面，让用户授权

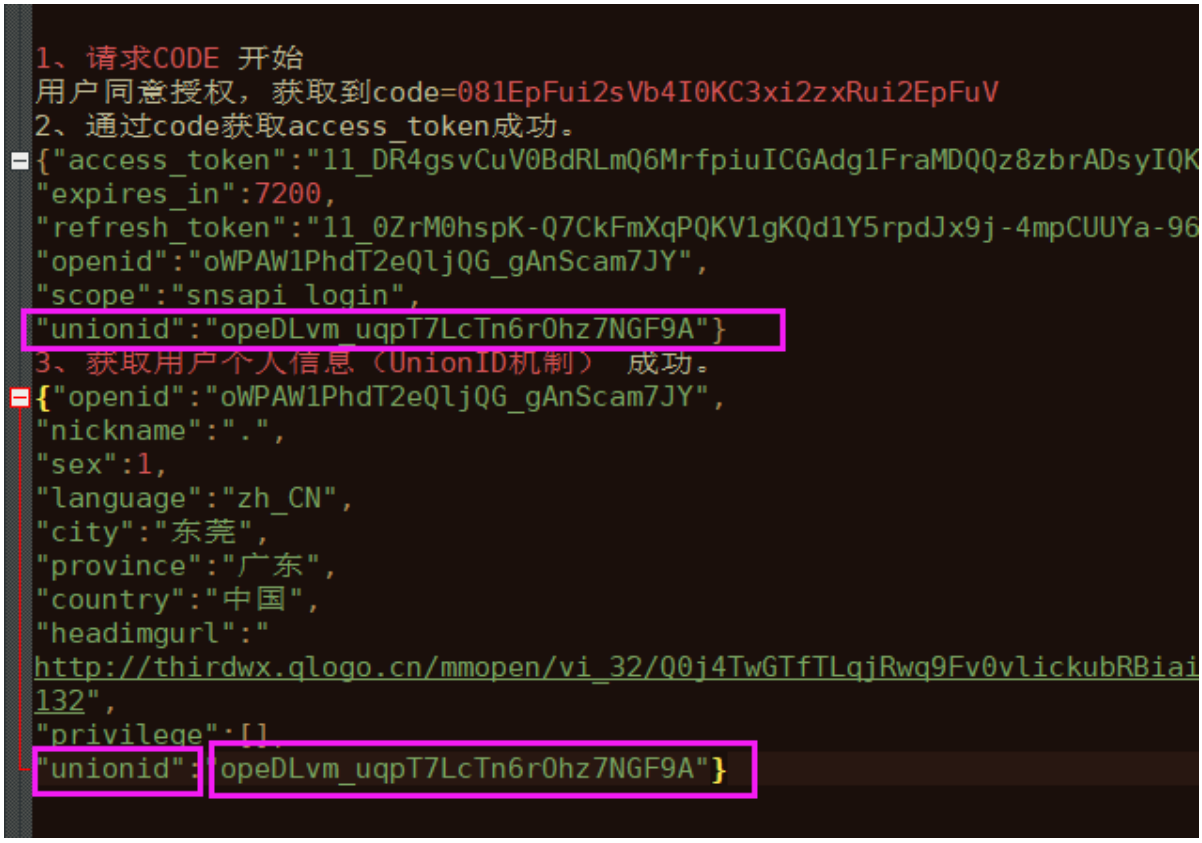
授权成功，浏览器页面提示扫码成功

然后我们的后台就会获取用户信息，信息如下

后台日志打印三步骤。



分析一下用户信息，记住 unionid



2）公众账号

登录微信开放平台，然后绑定公众号

回顾一下获取用户信息的接口，必须绑定公众号到开放平台，才会 unionid 字段返回

第四步：拉取用户信息(需scope为 snsapi_userinfo)

如果网页授权作用域为snsapi_userinfo，则此时开发者可以通过access_token和openid拉取用户信息了。

请求方法

```
http: GET（请使用https协议） https://api.weixin.qq.com/sns/userinfo?access_token=ACCESS_TOKEN&openid=OPENID&lang=zh_CN
```

参数说明

参数	描述
access_token	网页授权接口调用凭证,注意：此access_token与基础支持的access_token不同
openid	用户的唯一标识
lang	返回国家地区语言版本，zh_CN 简体，zh_TW 繁体，en 英语

返回参数

参数	描述
openid	用户的唯一标识
nickname	用户昵称
sex	用户的性别，值为1时是男性，值为2时是女性，值为0时是未知
province	用户个人资料填写的省份
city	普通用户个人资料填写的城市
country	国家，如中国为CN
headimgurl	用户头像，最后一个数值代表正方形头像大小（有0、46、64、96、132数值可选，0代表640*640正方形头像），用户没有头像时该项为空。若用户更换头像，原有头像URL将失效。
privilege	用户特权信息，json 数组，如微信沃卡用户为（chinaunicom）
unionid	只有在用户将公众号绑定到微信开放平台帐号后，才会出现该字段。

进行微信公众平台授权开发，上面已经做过了，这里就不重复了，然后对比一下微信公众号获取的 unionid 与开放平台的 unionid，会发现是一致的，但是各自的 openId 都是不一样的。

（其他我这里偷懒了，因为我用了公司的公众号，而且公司本来就保存了 unionid，而我之前也关注了，所以直接去数据库查一下对比就好了，省事点。）

unionid	openId
opeDLvm_uqpT7LcTn6rOhz7NGF9A	olqUu0mW25R_VprOP-_i9la1x7e4

nickName	sex	headimgurl
.	1	http://thirdwx.qlogo.cn/mmopen/vi_32/Q0j4TwG

country	province	city
中国	广东	东莞

4、微信公众号与微信开放平台关联整合

只要是同一个微信开放平台帐号下的公众号（一个或者多个），用户的 UnionID 是唯一的。换句话说， 同一用户，对同一个微信开放平台帐号下的不同应用（移动应用、网站应用、公众账号、小程序、第三方平台等），UnionID 是相同的。

目前微信开放平台所支持的应用：