微信网页授权调试步骤

1. **公众号：**

* **使用微信公众账号测试号：**

因为只有已经认证的服务号才拥有网页授权（获取用户信息）的权限，订阅号是没有该权限的。微信提供了测试账号，可以进行网页授权。

测试号申请链接：

<https://mp.weixin.qq.com/debug/cgi-bin/sandboxinfo?action=showinfo&t=sandbox/index>

* **配置测试账号：**

1. **接口配置信息：**

**（配置完后，先别提交，因为SpringBoot工程中还没有进行Token验证，微信公众号需要通过“提交”按钮验证所填写外网服务器的正确性）**

* **URL：**外网的IP + 测试Token的URL

eg：http://chrissell.natapp1.cc/sell/wechat/wx.do

* **Token：**可以随意填写，只要保证在SpringBoot工程中，测试代码使用的Token为此处填写的数值即可。

eg：chris

1. **JS接口安全域名：**

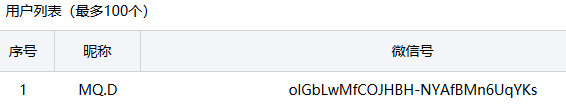
只填写外网服务器的域名即可，不需要填写<http://、Https://>

eg：chrissell.natapp1.cc

1. **测试号二维码：**

直接使用微信扫描测试号的二维码，关注该测试号，用于查看自己微信的Openid。用于验证SpringBoot工程所获取的Openid是否正确。

eg：



1. **网页服务->网页账号：网页授权获取用户的基本信息**

填写外网服务器的域名，不需要填写<http://、Https://>。

**作用：**

1. 当用户通过外网服务器访问微信公众号的web应用时，用户先访问到web应用。
2. 再由web应用到微信公众号进行授权，获取用户的Openid。
3. 其次，微信公众号需要有一个回调页面，把获取到的Openid等数据传回该回调页面，而这里填写的外网服务器域名就是该回调页面所在的服务器。
4. 最后，需要手工将用户请求中的redirect\_uri参数和获取到的Openid拼接，然后微信公众平台会将页面跳转到指定的redirect\_uri页面。

eg：跳转之后的页面URL：redirect\_uri + Openid

http://sell.com/#/?openid=olGbLwMfCOJHBH-NYAfBMn6UqYKs

1. **SpringBoot工程:**

* **进行Token验证**：

1. 将附录一中的代码添加到SpringBoot工程中；
2. 直接运行SpringBoot工程；
3. 在微信公众号的接口信息配置界面（填写Token的地方），点击提交按钮，就可以通过验证。此时公众号配置已经完成。

* **获取Openid：**

基本步骤：

1. **配置Nginx：**

位置：/usr/local/nginx/conf/nginx.conf

修改server中的

1. **server\_name:** 之后微信公众号要跳转的域名。

eg：sell.com

1. **location/sell/**：本地代码调试时，使用的链接。

proxy\_pass <http://SpringBoot>工程所在机器的IP:端口/web应用的context\_path

eg：

<http://192.168.43.149:8080/sell/>

1. **配置前端项目：**
2. 修改前端源码的配置文件：

位置:/opt/code/sell\_fe\_buyer/config/index.js

修改build下的Url的sellUrl、OpenidUrl。

* sellUrl（web主页链接）：

<http://sell.com>

* OpenidUrl（获取Openid的链接）：

http://chrissell.natapp1.cc/sell/wechat/authorize

1. 重新编译：

返回/opt/code/sell\_fe\_buyer/，使用npm run build编译

1. 将编译后的文件复制到原来的Nginx前端的根目录下：

命令：cp -r dist/\* /opt/data/wwwroot/sell/

（Nginx不需要重启）

1. **SpringBoot工程添加获取Openid的代码：**

将附录二得代码直接加入SpringBoot工程中即可。然后运行SpringBoot工程。

1. **用户访问链接：（测试是否能够实现页面跳转，并获取到Openid）**

外网服务器域名 + web应用中获取Openid的controller路径 + returnUrl

可以将百度替换成自己的外网服务器中的web应用地址，就可以实现从微信客户端访问web应用。

**eg：微信客户端访问链接，跳转到百度**

1. 外网服务器域名：

<http://chrissell.natapp1.cc>

1. web应用中获取Openid的controller路径：

/sell/wechat/authorize

1. returnUrl：

<http://www.baidu.com>

1. 拼接结果：（最总，用户在微信上点击的链接）

<http://chrissell.natapp1.cc/sell/wechat/authorize?returnUrl=http://www.baidu.com>

1. 用户最终的访问页面为：百度

该页面的URL：

https://www.baidu.com/?openid=olGbLwMfCOJHBH-NYAfBMn6UqYKs

**附参考链接：**

1. 微信开放文档：网页授权

https://developers.weixin.qq.com/doc/offiaccount/OA\_Web\_Apps/Wechat\_webpage\_authorization.html

1. 微信网页授权，第三方SDK：

https://github.com/Wechat-Group/WxJava

**附录一：验证Token**

package com.wechatorder.demo.ConnectMP;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.PrintWriter;  
import java.security.MessageDigest;  
import java.security.NoSuchAlgorithmException;  
import java.util.Arrays;  
import java.util.Enumeration;  
  
/\*\*  
 \* ====== 测试类：公众号链接（只运行一次，用于配置测试公众号时使用） =====  
 \* 用于链接微信测试号的：接口配置，  
 \* URL：http://chrissell.natapp1.cc/sell/wechat/wx.do  
 \* Token：chris  
 \* 一、验证Token步骤：  
 \* （1）在微信测试号上填写对应的接口配置：上面的URL、Token  
 \* （2）运行IntelliJ中的应用  
 \* （3）在微信测试号上，点击提交（保存），即可通过配置  
 \* Created by Chris on 2020/3/9.  
 \*/  
@RequestMapping("/wechat")  
@Controller  
@Slf4j  
public class VerifyMpToken {  
  
 private static String *WECHAT\_TOKEN* = "chris";  
 private static String *token* = "chris";  
  
 @RequestMapping(value = "/wx.do")  
 public void get(HttpServletRequest request, HttpServletResponse response) throws Exception {  
 *log*.error("WechatController ---- WechatController");  
 System.*out*.println("========WechatController========= ");  
 *log*.info("请求进来了...");  
  
 Enumeration pNames = request.getParameterNames();  
 while (pNames.hasMoreElements()) {  
 String name = (String) pNames.nextElement();  
 String value = request.getParameter(name);  
 // out.print(name + "=" + value);  
  
 String log = "name =" + name + " value =" + value;  
 // log.error(log);  
 }  
  
 String signature = request.getParameter("signature");/// 微信加密签名  
 String timestamp = request.getParameter("timestamp");/// 时间戳  
 String nonce = request.getParameter("nonce"); /// 随机数  
 String echostr = request.getParameter("echostr"); // 随机字符串  
 PrintWriter out = response.getWriter();  
  
 if (*checkSignature*(signature, timestamp, nonce)) {  
 out.print(echostr);  
 }  
 out.close();  
 out = null;  
 }  
  
 /\*\*  
 \* 校验签名  
 \*/  
 public static boolean checkSignature(String signature, String timestamp, String nonce) {  
 System.*out*.println("signature:" + signature + "timestamp:" + timestamp + "nonc:" + nonce);  
 String[] arr = new String[] { *token*, timestamp, nonce };  
 // 将token、timestamp、nonce三个参数进行字典序排序  
 Arrays.*sort*(arr);  
 StringBuilder content = new StringBuilder();  
 for (int i = 0; i < arr.length; i++) {  
 content.append(arr[i]);  
 }  
 MessageDigest md = null;  
 String tmpStr = null;  
  
 try {  
 md = MessageDigest.*getInstance*("SHA-1");  
 // 将三个参数字符串拼接成一个字符串进行sha1加密  
 byte[] digest = md.digest(content.toString().getBytes());  
 tmpStr = *byteToStr*(digest);  
 } catch (NoSuchAlgorithmException e) {  
 e.printStackTrace();  
 }  
  
 content = null;  
 // 将sha1加密后的字符串可与signature对比，标识该请求来源于微信  
 System.*out*.println(tmpStr.equals(signature.toUpperCase()));  
 return tmpStr != null ? tmpStr.equals(signature.toUpperCase()) : false;  
 }  
  
 /\*\*  
 \* 将字节数组转换为十六进制字符串  
 \*  
 \* @param byteArray  
 \* @return  
 \*/  
 private static String byteToStr(byte[] byteArray) {  
 String strDigest = "";  
 for (int i = 0; i < byteArray.length; i++) {  
 strDigest += *byteToHexStr*(byteArray[i]);  
 }  
 return strDigest;  
 }  
  
 /\*\*  
 \* 将字节转换为十六进制字符串  
 \*  
 \* @param mByte  
 \* @return  
 \*/  
 private static String byteToHexStr(byte mByte) {  
 char[] Digit = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };  
 char[] tempArr = new char[2];  
 tempArr[0] = Digit[(mByte >>> 4) & 0X0F];  
 tempArr[1] = Digit[mByte & 0X0F];  
  
 String s = new String(tempArr);  
 return s;  
 }  
}

**附录二：获取Openid**

@Controller  
@RequestMapping("/wechat")  
@Slf4j  
public class WechatController {  
 //微信公众号  
 @Autowired  
 private WxMpService wxMpService;  
 //微信开放平台  
 @Autowired  
 private WxMpService wxOpenService;  
  
 /\*\*  
 \* 用户测试的URL：（直接跳转到百度）  
 \* http://chrissell.natapp1.cc/sell/wechat/authorize?returnUrl=http://www.baidu.com  
 \*  
 \* 跳转之后的百度页面的URL（会添加该用户的openid）：https://www.baidu.com/?openid=olGbLwMfCOJHBH-NYAfBMn6UqYKs  
 \*  
 \* @param returnUrl  
 \* @return  
 \* @throws UnsupportedEncodingException  
 \*/  
 @GetMapping("authorize")  
 public String authorize(@RequestParam("returnUrl") String returnUrl) throws UnsupportedEncodingException {  
 //1、配置  
  
 //2、调用方法  
 String url = "http://chrissell.natapp1.cc/sell/wechat/userInfo";  
 //构造网页授权url，含微信的网页授权部分 --- 可以对url进行编码：utf-8  
 String redirectUrl = wxMpService.oauth2buildAuthorizationUrl(url, WxConsts.OAuth2Scope.*SNSAPI\_BASE* , URLEncoder.*encode*(returnUrl));  
 return "redirect:" + redirectUrl;  
 }  
  
 /\*\*  
 \* 获取用户信息：openId  
 \* @param code  
 \* @param returnUrl  
 \* @return  
 \*/  
 @GetMapping("/userInfo")  
 public String userInfo(@RequestParam("code") String code,  
 @RequestParam("state") String returnUrl) {  
 WxMpOAuth2AccessToken wxMpOAuth2AccessToken = new WxMpOAuth2AccessToken();  
 try {  
 //当用户同意授权后，会回调所设置的url并把authorization code传过来，然后用这个code获得access token openid  
 wxMpOAuth2AccessToken = wxMpService.oauth2getAccessToken(code);  
 } catch (WxErrorException e) {  
 *log*.error("【微信网页授权】{}", e);  
 throw new SellException(ResultEnum.*WECHAT\_MP\_ERROR*.getCode(), e.getError().getErrorMsg());  
 }  
 String openId = wxMpOAuth2AccessToken.getOpenId();  
 return "redirect:" + returnUrl + "?openid=" + openId;  
 }  
  
}