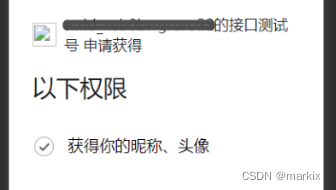
17 微信OAuth2授权登录

相信你在使用微信的时候，打开某些网页时，经常会弹出一个“xxx申请获取你的信息，是否同意？”的提示。类似这样：



当你点击"同意"时，网页应用便能获取你微信的用户信息。这个流程也是通过OAuth2实现的。

在专栏第15篇 OAuth2登录中我们提到，OAuth2登录的实现原理就是 Client获取用户授权，得到令牌，通过令牌获取用户信息（资源）。再在本地构建用户登录认证信息，维持用户会话状态，以此达到登录的目的。

同理，我们也可以借助微信OAuth2网页授权实现登录。在微信OAuth2网页授权中，我们的项目就相当于是Client，通过微信用户授权，最终取到用户信息，在本地构建用户登录认证信息，维持用户会话状态，达到登录的目的。

so，本文便尝试使用 SpringSecurity OAuth2 Client模块 接入 微信OAuth2网页授权，实现登录。

微信网页授权 接入文档：https://developers.weixin.qq.com/doc/offiaccount/OA\_Web\_Apps/Wechat\_webpage\_authorization.html

准备

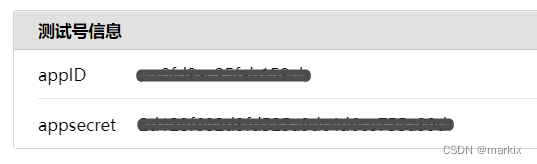
微信网页授权需要使用 公众号，门槛较高，一般在开发阶段我们就使用测试号。

申请接口测试号（沙盒号） https://developers.weixin.qq.com/doc/offiaccount/Basic\_Information/Requesting\_an\_API\_Test\_Account.html

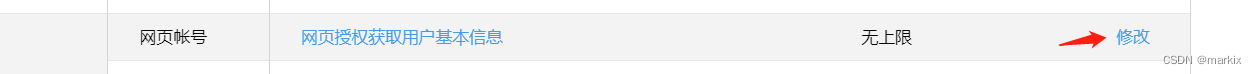
由于用户体验和安全性方面的考虑，微信公众号的注册有一定门槛，某些高级接口的权限需要微信认证后才可以获取。

所以，为了帮助开发者快速了解和上手微信公众号开发，熟悉各个接口的调用，推出了微信公众帐号测试号，通过手机微信扫描二维码即可获得测试号。

得到测试号appID和appsecret，在后续配置中会用到。



接着在测试号页面配置下回调地址：





微信网页授权，需要在微信客户端（APP）打开对应的网页，而我们开发是在本地，APP怎么能访问我们本地的网页服务呢？

这里有两种方法，1.微信开发者工具，类似在本地运行APP。 2.本地穿透，将本地的服务暴露到外网，这需要通过特定软件实现。比如 Natapp。

本文使用微信开发者工具来进行测试。其下载地址为： https://developers.weixin.qq.com/doc/offiaccount/OA\_Web\_Apps/Web\_Developer\_Tools.html

为帮助开发者更方便、更安全地开发和调试基于微信的网页，推出了 web 开发者工具。它是一个桌面应用，通过模拟微信客户端的表现，使得开发者可以使用这个工具方便地在 PC 或者 Mac 上进行开发和调试工作。

实战开发

参看 微信网页授权文档

授权请求

https://open.weixin.qq.com/connect/oauth2/authorize

?appid=wx0fd9ac25fab159eb

&redirect\_uri=http%3A%2F%2F127.0.0.1%3A8080

&response\_type=code

&state=001

&scope=snsapi\_base#wechat\_redirect

code->token 请求

https://api.weixin.qq.com/sns/oauth2/access\_token

?grant\_type=authorization\_code

&appid=wx0fd9ac25fab159eb

&secret=2d128f682d3fd525a9dc4d6ce755a39d

&code=061EhN000PhwPO1ctC000pawTC3EhN0x

token->userinfo 请求

https://api.weixin.qq.com/sns/userinfo

?access\_token=62\_s0SK3CpWAHn1n4DqUUFJKqfUkYYmX62TI5DBrmRjHg3jeNeP-T\_gOhw\_uW9RSElNJKLILcoTi\_IxkL6sR3ES7vRPx4aBOGz6aoY\_StMfwtg

&openid=oQHEX6mZRvnrWVWg8EmvNhCkhWq8

&lang=zh\_CN

可以看出，微信OAuth2网页授权和标准OAuth2有很多出入，比如client\_id 变成了appid，client\_secret变成了secret，还有响应内容不同等等差别，这些差异致使我们无法直接使用 SpringSecurity OAuth2 Client 模块，所以我们需要对原有的实现进行改造兼容。

搭建示例

我们在 专栏第15篇 OAuth2登录 中示例的基础上，进行改造兼容。

常量类

public interface WechatConstants {

/\*\*

\* 配置文件中的 registration id

\*/

String REG\_ID = "weixin-app";

String PARAM\_APP\_ID = "appid";

String PARAM\_SECRET = "secret";

String PARAM\_SUFFIX = "#wechat\_redirect";

String PARAM\_OPENID = "openid";

String PARAM\_LANG = "lang";

}

授权请求的参数处理器。兼容微信网页授权

public class WeiXinOAuth2AuthorizationRequestCustomizer implements Consumer<OAuth2AuthorizationRequest.Builder> {

@Override

public void accept(OAuth2AuthorizationRequest.Builder builder) {

builder.authorizationRequestUri(CustomUriFunction.INS);

}

private static class CustomUriFunction implements Function<UriBuilder, URI> {

private static final CustomUriFunction INS = new CustomUriFunction();

@Override

public URI apply(UriBuilder uriBuilder) {

URI uri = uriBuilder.build();

String query = uri.getQuery();

if(query.contains(WechatConstants.REG\_ID)) {

// 特殊处理 weixin 的授权请求。

// 将 client\_id 改为 appid

// url 末尾增加 #wechat\_redirect

String reqUri = uri.toString()

.replaceAll(OAuth2ParameterNames.CLIENT\_ID, WechatConstants.PARAM\_APP\_ID)

.concat(WechatConstants.PARAM\_SUFFIX);

uri = URI.create(reqUri);

}

return uri;

}

}

}

code->token 请求的参数处理器。兼容微信网页授权

public class WeiXinOAuth2AuthorizationCodeGrantRequestEntityConverter extends OAuth2AuthorizationCodeGrantRequestEntityConverter {

@Override

public RequestEntity<?> convert(OAuth2AuthorizationCodeGrantRequest authorizationCodeGrantRequest) {

ClientRegistration clientRegistration = authorizationCodeGrantRequest.getClientRegistration();

if(WechatConstants.REG\_ID.equals(clientRegistration.getRegistrationId())){

// 微信的请求特殊处理

MultiValueMap<String, String> queryParameters = this.buildWechatQueryParameters(authorizationCodeGrantRequest);

URI uri = UriComponentsBuilder.fromUriString(clientRegistration.getProviderDetails().getTokenUri())

.queryParams(queryParameters)

.build()

.toUri();

return RequestEntity.get(uri).build();

}

return super.convert(authorizationCodeGrantRequest);

}

private MultiValueMap<String, String> buildWechatQueryParameters(OAuth2AuthorizationCodeGrantRequest authorizationCodeGrantRequest) {

ClientRegistration clientRegistration = authorizationCodeGrantRequest.getClientRegistration();

OAuth2AuthorizationExchange authorizationExchange = authorizationCodeGrantRequest.getAuthorizationExchange();

MultiValueMap<String, String> parameters = new LinkedMultiValueMap<>();

parameters.add(OAuth2ParameterNames.GRANT\_TYPE, authorizationCodeGrantRequest.getGrantType().getValue());

parameters.add(OAuth2ParameterNames.CODE, authorizationExchange.getAuthorizationResponse().getCode());

// 微信特定参数：appid、secret

parameters.add(WechatConstants.PARAM\_APP\_ID, clientRegistration.getClientId());

parameters.add(WechatConstants.PARAM\_SECRET, clientRegistration.getClientSecret());

return parameters;

}

}

code->token 请求的响应处理器。兼容微信网页授权

public class WeiXinMapOAuth2AccessTokenResponseConverter implements Converter<Map<String, String>, OAuth2AccessTokenResponse> {

private static final Converter<Map<String, String>, OAuth2AccessTokenResponse> DELEGATE = new MapOAuth2AccessTokenResponseConverter();

@Override

public OAuth2AccessTokenResponse convert(Map<String, String> tokenResponseParameters) {

// 兼容微信响应中没有 tokenType 值

if(tokenResponseParameters.get(OAuth2ParameterNames.TOKEN\_TYPE) == null){

tokenResponseParameters.put(OAuth2ParameterNames.TOKEN\_TYPE, OAuth2AccessToken.TokenType.BEARER.getValue());

}

return DELEGATE.convert(tokenResponseParameters);

}

}

token->userinfo 请求的参数处理器。兼容微信网页授权

public class WeiXinOAuth2UserRequestEntityConverter extends OAuth2UserRequestEntityConverter{

@Override

public RequestEntity<?> convert(OAuth2UserRequest userRequest) {

ClientRegistration clientRegistration = userRequest.getClientRegistration();

if(WechatConstants.REG\_ID.equals(clientRegistration.getRegistrationId())){

// 微信的请求特殊处理

MultiValueMap<String, String> queryParameters = this.buildWechatQueryParameters(userRequest);

URI uri = UriComponentsBuilder.fromUriString(clientRegistration.getProviderDetails().getUserInfoEndpoint().getUri())

.queryParams(queryParameters)

.build()

.toUri();

return RequestEntity.get(uri).build();

}

return super.convert(userRequest);

}

private MultiValueMap<String, String> buildWechatQueryParameters(OAuth2UserRequest userRequest) {

MultiValueMap<String, String> parameters = new LinkedMultiValueMap<>();

parameters.add(OAuth2ParameterNames.ACCESS\_TOKEN, userRequest.getAccessToken().getTokenValue());

// 微信特定参数：openid、lang

parameters.add(WechatConstants.PARAM\_OPENID, userRequest.getAdditionalParameters().get(WechatConstants.PARAM\_OPENID).toString());

parameters.add(WechatConstants.PARAM\_LANG, "zh\_CN");

return parameters;

}

}

改造原 SecurityConfiguration 的 configure 方法

上述的兼容性改造类，最终都需要配置进对应的调用类才能生效。配置方法如下：

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

.anyRequest().authenticated()

.and()

// 支持 OAuth2 登录

.oauth2Login()

.authorizationEndpoint(t -> {

DefaultOAuth2AuthorizationRequestResolver requestResolver = new DefaultOAuth2AuthorizationRequestResolver(

getClientRegistrationRepository(http), OAuth2AuthorizationRequestRedirectFilter.DEFAULT\_AUTHORIZATION\_REQUEST\_BASE\_URI);

// 授权请求的自定义处理逻辑！

requestResolver.setAuthorizationRequestCustomizer(new WeiXinOAuth2AuthorizationRequestCustomizer());

t.authorizationRequestResolver(requestResolver);

})

.tokenEndpoint(t -> {

DefaultAuthorizationCodeTokenResponseClient tokenResponseClient = new DefaultAuthorizationCodeTokenResponseClient();

// token请求的自定义处理逻辑！

tokenResponseClient.setRequestEntityConverter(new WeiXinOAuth2AuthorizationCodeGrantRequestEntityConverter());

OAuth2AccessTokenResponseHttpMessageConverter converter = new OAuth2AccessTokenResponseHttpMessageConverter();

// 兼容 微信的 text/plain 响应类型

converter.setSupportedMediaTypes(Arrays.asList(MediaType.APPLICATION\_JSON, MediaType.TEXT\_PLAIN, new MediaType("application", "\*+json")));

// 兼容 微信的 响应内容

converter.setTokenResponseConverter(new WeiXinMapOAuth2AccessTokenResponseConverter());

RestTemplate restTemplate = new RestTemplate(Arrays.asList(new FormHttpMessageConverter(), converter));

restTemplate.setErrorHandler(new OAuth2ErrorResponseErrorHandler());

tokenResponseClient.setRestOperations(restTemplate);

t.accessTokenResponseClient(tokenResponseClient);

})

.userInfoEndpoint(t -> {

DefaultOAuth2UserService userService = new DefaultOAuth2UserService();

// userinfo请求的自定义处理逻辑！

userService.setRequestEntityConverter(new WeiXinOAuth2UserRequestEntityConverter());

MappingJackson2HttpMessageConverter converter = new MappingJackson2HttpMessageConverter();

// 兼容 微信的 text/plain 响应类型

converter.setSupportedMediaTypes(Arrays.asList(MediaType.APPLICATION\_JSON, MediaType.TEXT\_PLAIN, new MediaType("application", "\*+json")));

RestTemplate restTemplate = new RestTemplate(Arrays.asList(converter));

restTemplate.setErrorHandler(new OAuth2ErrorResponseErrorHandler());

userService.setRestOperations(restTemplate);

t.userService(userService);

})

.and()

.csrf().disable()

;

}

application.yml

server:

port: 8080

spring:

security:

oauth2:

client:

registration: # 定义应用信息

weixin-app:

clientName: 微信帐号登录

client-id: xxx #测试号的appid

client-secret: xxx #测试号的secret

clientAuthenticationMethod: basic

authorizationGrantType: authorization\_code

redirectUri: 'http://127.0.0.1:8080/login/oauth2/code/{registrationId}'

scope: snsapi\_userinfo

provider: weixin

provider: # 定义授权服务器信息

weixin:

authorizationUri: https://open.weixin.qq.com/connect/oauth2/authorize

tokenUri: https://api.weixin.qq.com/sns/oauth2/access\_token

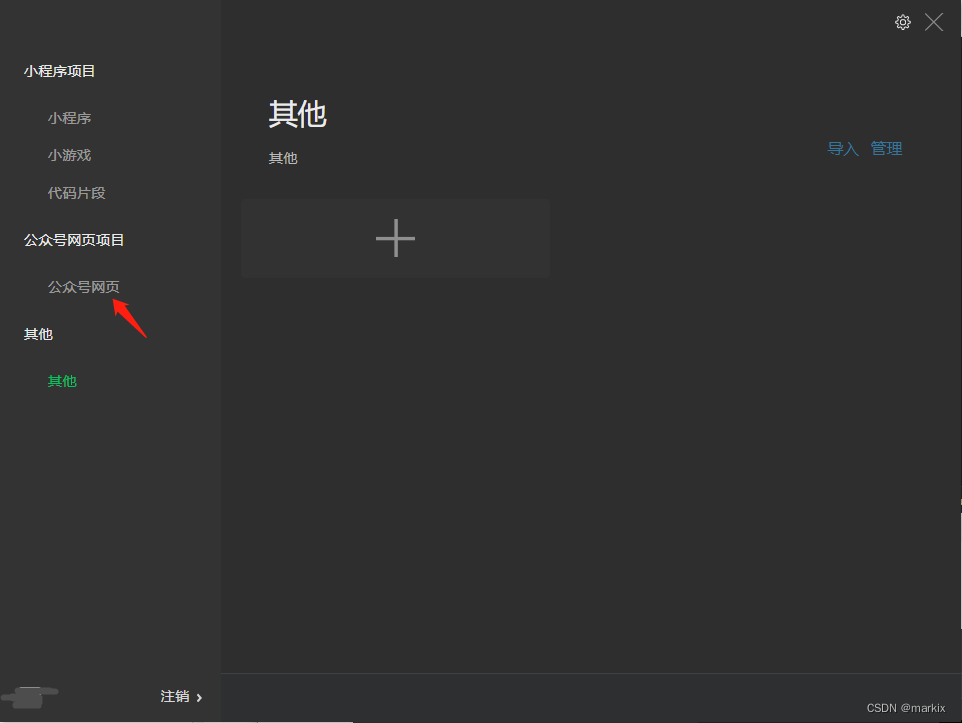
userInfoUri: https://api.weixin.qq.com/sns/userinfo

userNameAttribute: nickname

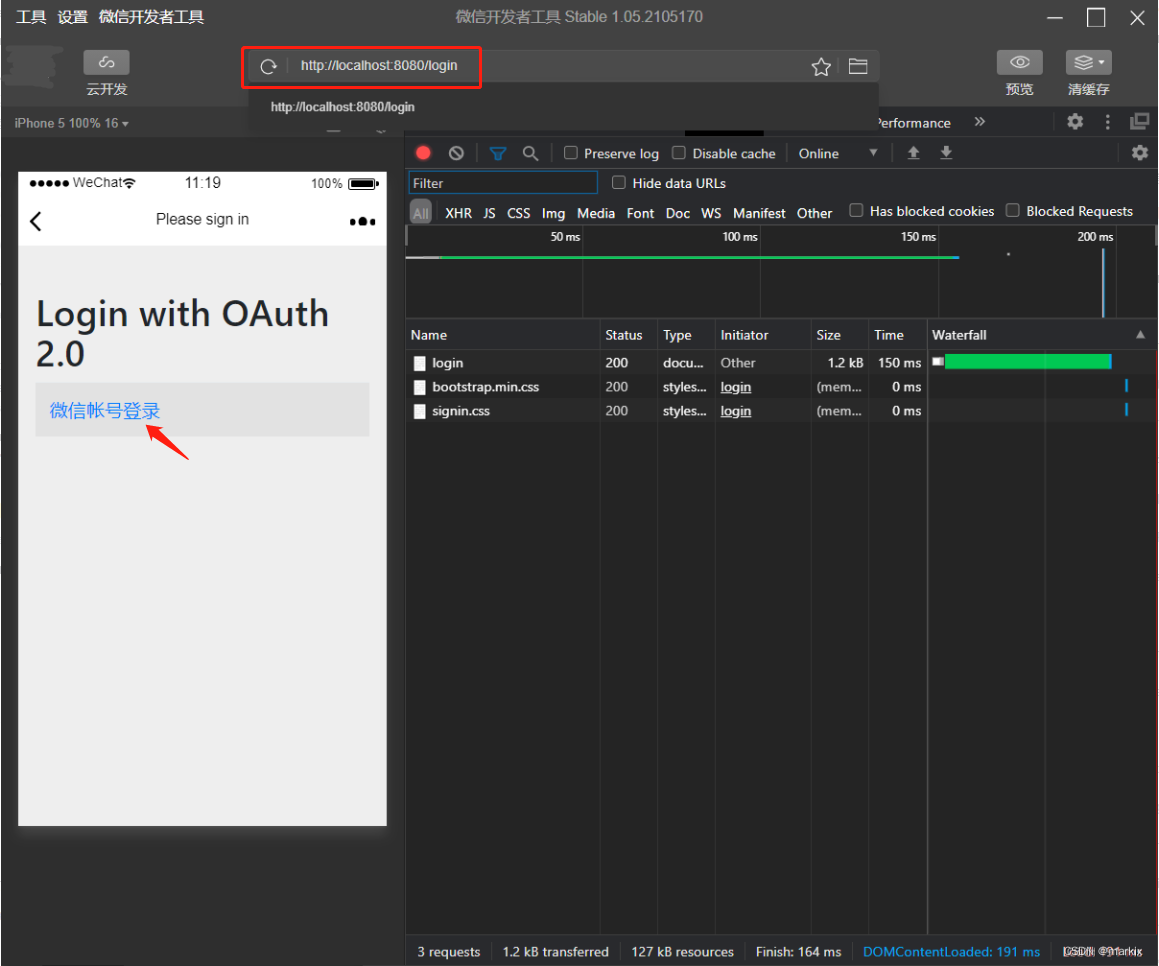
ok，搞定。启动服务。

测试

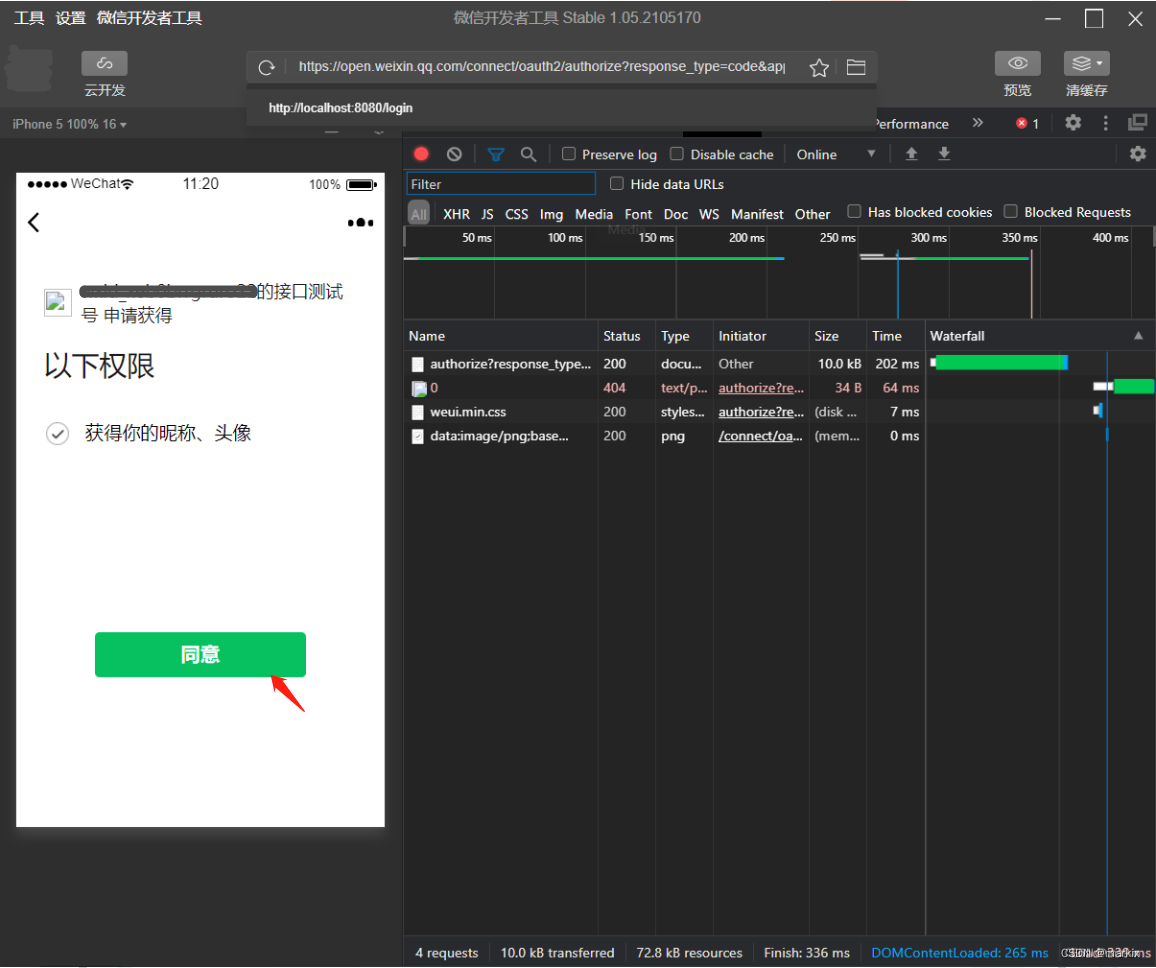
打开 微信开发者工具，登录，点击 公众号网页



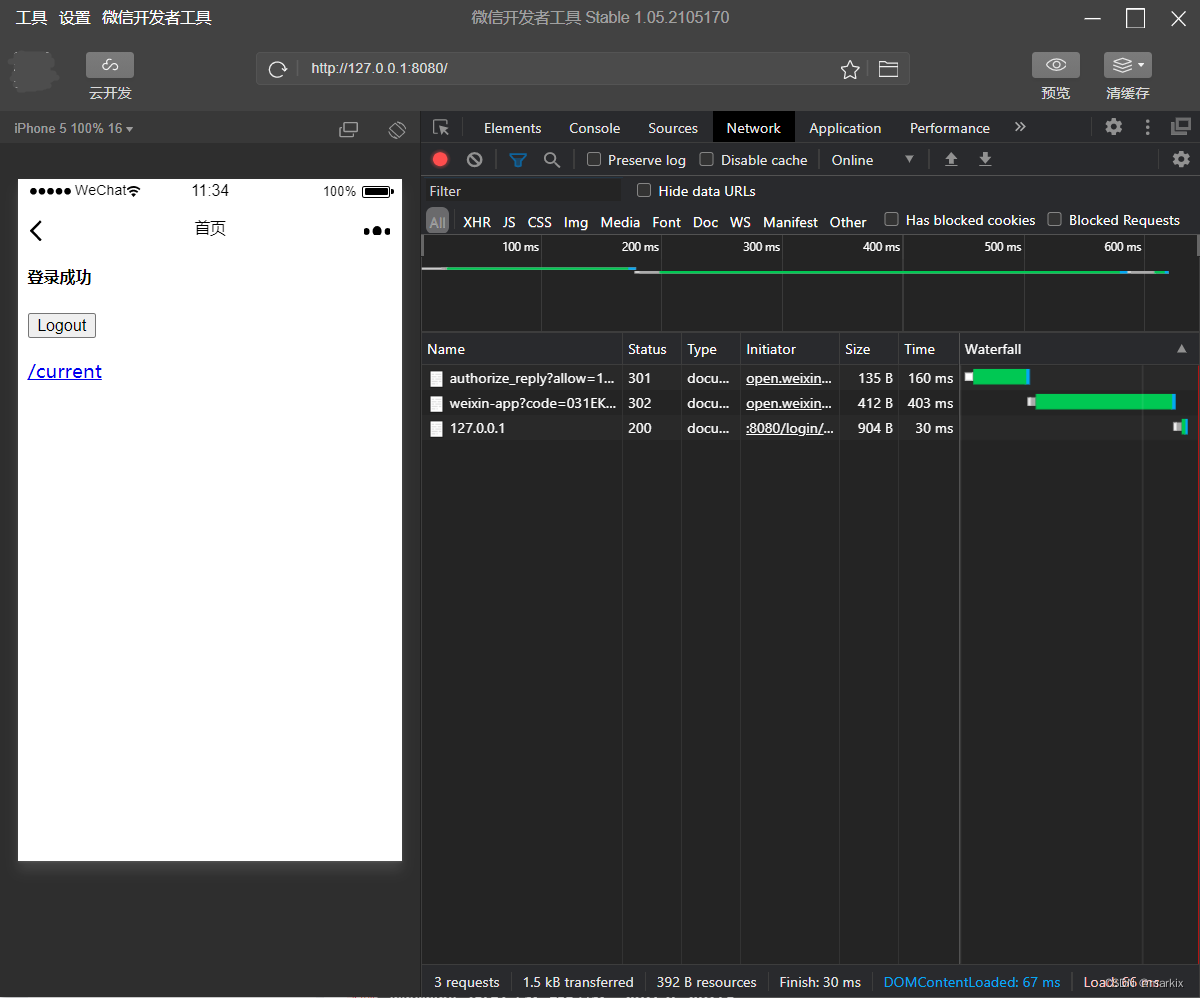
在地址栏输入 http://localhost:8080，点击 微信帐号登录



点击同意



登录成功



ok，大功告成。我们成功利用 微信网页授权 实现我们项目的登录。

reference:

集成微信公众号OAuth2.0授权

https://cloud.tencent.com/developer/article/1875293

整合企业微信扫码登录

https://cloud.tencent.com/developer/article/2134892

什么是内网穿透?

<https://natapp.cn/article/shenmeshineiwangchuantou>

NATAPP1分钟快速新手图文教程

<https://natapp.cn/article/natapp_newbie>

用Natapp(ngrok)进行微信本地开发调试

https://natapp.cn/article/wechat\_local\_debug