微信支付的开发文档：

https://pay.weixin.qq.com/wiki/doc/apiv3/open/pay/chapter2\_8\_1.shtml

一、注册商户号

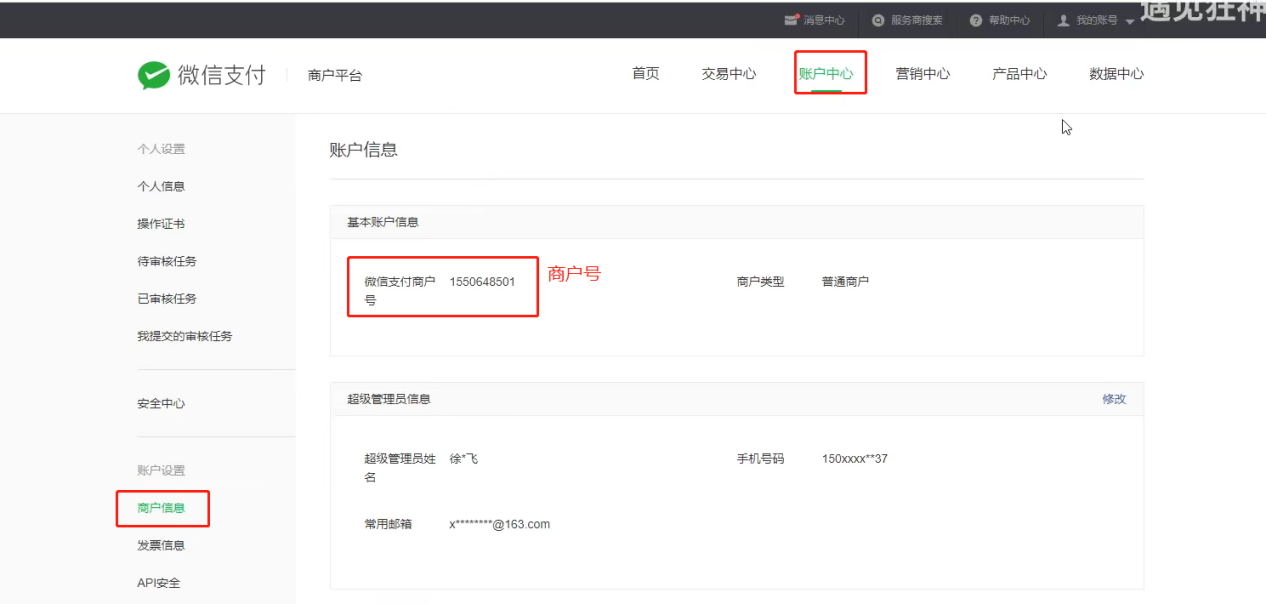
<https://pay.weixin.qq.com/index.php/core/home/login?return_url=%2F>

1.1、点击成为商家



填写信息注册成功以后等待几个工作日审核通过以后就可以扫码进去了

1-2、商户号



1. 下载安装证书（参考官网链接提示）获取秘钥
2. 商户号开通产品，具备支付功能



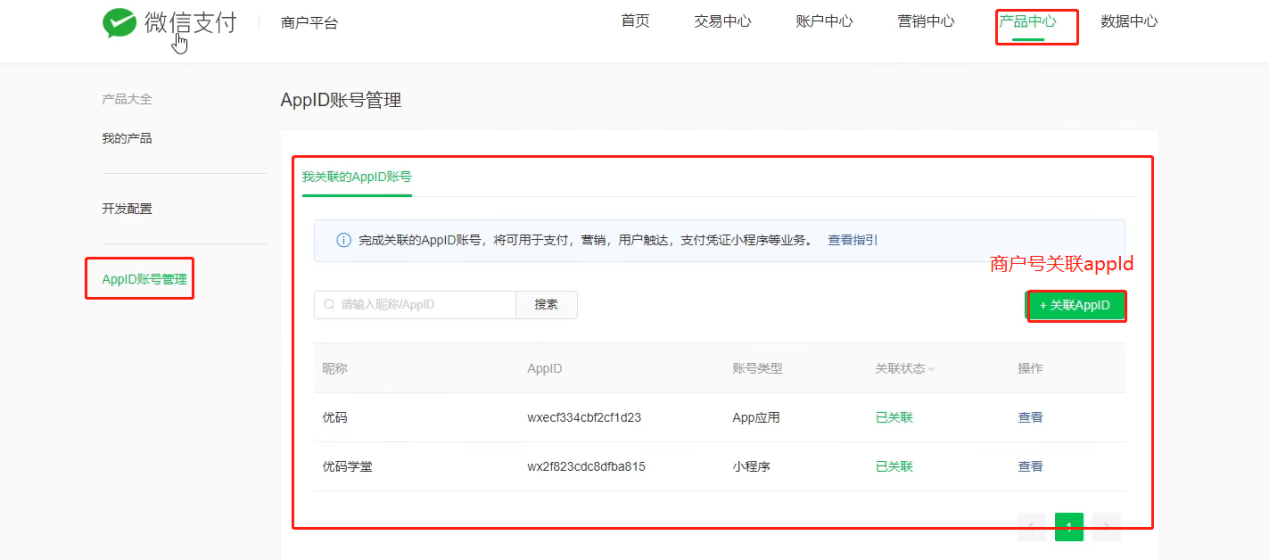
1. 配置回调地址



1. 申请企业级别的微信小程序（得需要上传营业执照），获取到appId

5-1、在小程序开发文档里面可以获取到appid，设置开发人员名单， 项目上线审核， 开通认证， 需要交300块钱的认证费

1. 微信商户号和小程序的绑定



微信小程序里面支付步骤

1. 创建订单

请求创建订单的API接口，（订单金额，收货地址，订单中的商品信息）发送到服务器

服务器响应的结果：订单编号

1. 订单预支付

请求订单预支付的API接口，把订单编号发送到服务器

服务器响应的结果：订单预支付的参数对象，里面包含了订单支付的必要参数信息

1. 发起微信支付

调用wx.requestPayment()这个API，发起微信支付，把步骤二得到的微信预支付作为参数传递给wx.requestPayment()

准备工作完之后就开始写代码了

1. v2 -- JSAPI 支付

1. 获取openid https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=4\_4

1-1、报错的话就只用最后一个依赖

<!--<dependency>  
 <groupId>com.github.tedzhdz</groupId>  
 <artifactId>wxpay-sdk</artifactId>  
 <version>3.0.10</version>  
 </dependency>  
 <dependency>  
 <groupId>com.github.binarywang</groupId>  
 <artifactId>weixin-java-pay</artifactId>  
 <version>3.4.0</version>  
 </dependency>-->

<dependency>  
 <groupId>com.github.wxpay</groupId>  
 <artifactId>wxpay-sdk</artifactId>  
 <version>0.0.3</version>  
 </dependency>

1-2、获取授权码 （查看上面的获取openid地址看官网）

1-3、拿着上步获取的code，调取接口获取openid

1. 拿到openid之后就可以调用微信的统一下单接口开始支付了

<https://pay.weixin.qq.com/wiki/doc/api/jsapi.php?chapter=9_1>

1. v2 -- native支付
2. 生成二维码 （下面是大概的代码示例）

Map paramMap = new HashMap(); // 请求微信接口生成二维码的参数封装  
paramMap.put("appid", “APPID”);  
paramMap.put("mch\_id", “商户号”);  
paramMap.put("nonce\_str", WXPayUtil.*generateNonceStr*()); // 随机字符串，微信提供的工具类生成的  
paramMap.put("body", “支付的简单介绍”);  
paramMap.put("out\_trade\_no", “订单号”); // 商户订单号  
//paramMap.put("total\_fee", order.getAmount().multiply(new BigDecimal("100")).longValue()+"");  
paramMap.put("total\_fee", "1"); //为了测试，统一写成这个值  
paramMap.put("spbill\_create\_ip", "127.0.0.1");  
paramMap.put("notify\_url", "http://guli.shop.com/api/order/weixinPay/weixinNotify"); // 扫码付款成功的回调地址，微信调用  
paramMap.put("trade\_type", "NATIVE"); // 支付方式  
//4 调用微信生成二维码接口,httpclient调用  
HttpClient client = new HttpClient("https://api.mch.weixin.qq.com/pay/unifiedorder"); // 工具类在最下面（httpClient工具类）  
//设置map参数（参数需要转换成xml格式，微信提供的工具类转成xml格式的参数）

// 也可以直接用工具类生成签名 WXPayUtil.generateSignature(paramMap, "商户key秘钥"); 签名 sign用微信提供的工具类生成

client.setXmlParam( WXPayUtil.*generateSignedXml*(paramMap, ”商户号key”) );

client.setHttps(true);  
client.post();  
//5 返回相关数据  
String xml = client.getContent();  
Map<String, String> resultMap = WXPayUtil.*xmlToMap*(xml); // 用工具类将相应的xml数据转换成map  
System.*out*.println("resultMap:"+resultMap);  
//6 封装返回结果集  
Map map = new HashMap<>();  
map.put("orderId", orderId);  
map.put("totalFee", order.getAmount());  
map.put("resultCode", resultMap.get("result\_code"));  
map.put("codeUrl", resultMap.get("code\_url")); //二维码地址

将codeUrl返给前端， 前端用 qrcodejs2 插件 生成支付二维码 即可用微信扫码付钱了

处理回调接口， 不然微信会在24小时之后将钱返回给用户

2、查询支付结果（也可以跳过此步骤直接看3，利用微信的异步回调 notify\_url）

//封装提交参数  
Map paramMap = new HashMap();  
paramMap.put("appid", “appid”);  
paramMap.put("mch\_id", “商户id”);  
paramMap.put("out\_trade\_no", “订单号”);  
paramMap.put("nonce\_str", WXPayUtil.generateNonceStr());  
  
//设置请求内容  
HttpClient client = new HttpClient("https://api.mch.weixin.qq.com/pay/orderquery");  
client.setXmlParam( WXPayUtil.generateSignedXml(paramMap, “商户秘钥key”) );  
client.setHttps(true);  
client.post();  
  
//4 得到微信接口返回数据  
String xml = client.getContent();  
Map<String, String> resultMap = WXPayUtil.xmlToMap(xml);  
System.out.println("支付状态resultMap:"+resultMap);

3、微信异步回调接口 notify\_url 告诉我们支付成功

@RequestMapping(“api/order/weixinPay/weixinNotify”)

public String wxpayCallback(String result\_code) {

if(“SUCCESS”.equels(result\_code)) {

// 此处支付成功之后就可以处理自己的业务了， 比如更新订单状态

// 前端写个定时器一直给前端查咱们的订单表，直到此处付款成功之后取// 消定时器

// 支付成功，给微信响应数据，让微信停止重复通知我们

Map<String, String> returnMap = new HashMap<>();

returnMap .put(“return\_code”, “SUCCESS”);

String params = WXPayUtil.mapToXml(returnMap);

return params ;

}

}

V2中JSAPI 和 Native 两种支付方式的流程

微信小程序支付（jsapi）的流程是 java统一下单， 拿到prepay\_id封装一个map给微信小程序，由微信小程序调取支付，java还得准备微信的下单成功的回调notify\_url

Pc扫码支付（native）的流程是java统一下单，生成二维码返给前端进行显示二维码，同时java也得准备微信的下单成功的回调notify\_url

httpClient工具类

import org.apache.http.Consts;  
import org.apache.http.HttpEntity;  
import org.apache.http.NameValuePair;  
import org.apache.http.client.ClientProtocolException;  
import org.apache.http.client.entity.UrlEncodedFormEntity;  
import org.apache.http.client.methods.\*;  
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;  
import org.apache.http.conn.ssl.SSLContextBuilder;  
import org.apache.http.conn.ssl.TrustStrategy;  
import org.apache.http.entity.StringEntity;  
import org.apache.http.impl.client.CloseableHttpClient;  
import org.apache.http.impl.client.HttpClients;  
import org.apache.http.message.BasicNameValuePair;  
import org.apache.http.ssl.SSLContexts;  
import org.apache.http.util.EntityUtils;  
  
import javax.net.ssl.SSLContext;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.security.KeyStore;  
import java.security.cert.CertificateException;  
import java.security.cert.X509Certificate;  
import java.text.ParseException;  
import java.util.HashMap;  
import java.util.LinkedList;  
import java.util.List;  
import java.util.Map;  
  
*/\*\*  
 \* http请求客户端  
 \*  
 \** ***@author*** *qy  
 \*  
 \*/*public class HttpClient {  
  
 private String url;  
 private Map<String, String> param;  
 private int statusCode;  
 private String content;  
 private String xmlParam;  
 private boolean isHttps;  
 private boolean isCert = false;  
 //证书密码 微信商户号（mch\_id）  
 private String certPassword;  
  
 public boolean isHttps() {  
 return isHttps;  
 }  
  
 public void setHttps(boolean isHttps) {  
 this.isHttps = isHttps;  
 }  
  
 public boolean isCert() {  
 return isCert;  
 }  
  
 public void setCert(boolean cert) {  
 isCert = cert;  
 }  
  
 public String getXmlParam() {  
 return xmlParam;  
 }  
  
 public void setXmlParam(String xmlParam) {  
 this.xmlParam = xmlParam;  
 }  
  
 public HttpClient(String url, Map<String, String> param) {  
 this.url = url;  
 this.param = param;  
 }  
  
 public HttpClient(String url) {  
 this.url = url;  
 }  
  
 public String getCertPassword() {  
 return certPassword;  
 }  
  
 public void setCertPassword(String certPassword) {  
 this.certPassword = certPassword;  
 }  
  
 public void setParameter(Map<String, String> map) {  
 param = map;  
 }  
  
 public void addParameter(String key, String value) {  
 if (param == null)  
 param = new HashMap<String, String>();  
 param.put(key, value);  
 }  
  
 public void post() throws ClientProtocolException, IOException {  
 HttpPost http = new HttpPost(url);  
 setEntity(http);  
 execute(http);  
 }  
  
 public void put() throws ClientProtocolException, IOException {  
 HttpPut http = new HttpPut(url);  
 setEntity(http);  
 execute(http);  
 }  
  
 public void get() throws ClientProtocolException, IOException {  
 if (param != null) {  
 StringBuilder url = new StringBuilder(this.url);  
 boolean isFirst = true;  
 for (String key : param.keySet()) {  
 if (isFirst)  
 url.append("?");  
 else  
 url.append("&");  
 url.append(key).append("=").append(param.get(key));  
 }  
 this.url = url.toString();  
 }  
 HttpGet http = new HttpGet(url);  
 execute(http);  
 }  
  
 */\*\*  
 \* set http post,put param  
 \*/* private void setEntity(HttpEntityEnclosingRequestBase http) {  
 if (param != null) {  
 List<NameValuePair> nvps = new LinkedList<NameValuePair>();  
 for (String key : param.keySet())  
 nvps.add(new BasicNameValuePair(key, param.get(key))); // 参数  
 http.setEntity(new UrlEncodedFormEntity(nvps, Consts.*UTF\_8*)); // 设置参数  
 }  
 if (xmlParam != null) {  
 http.setEntity(new StringEntity(xmlParam, Consts.*UTF\_8*));  
 }  
 }  
 private void execute(HttpUriRequest http) throws ClientProtocolException,  
 IOException {  
 CloseableHttpClient httpClient = null;  
 try {  
 if (isHttps) {  
 if(isCert) {  
 //设置成自己的证书cert目录路径，因为在退款的时候需要证书FileInputStream inputStream = new FileInputStream(new File(“别忘了设置”));  
 KeyStore keystore = KeyStore.*getInstance*("PKCS12");  
 char[] partnerId2charArray = certPassword.toCharArray();  
 keystore.load(inputStream, partnerId2charArray);  
 SSLContext sslContext = SSLContexts.*custom*().loadKeyMaterial(keystore, partnerId2charArray).build();  
 SSLConnectionSocketFactory sslsf =  
 new SSLConnectionSocketFactory(sslContext,  
 new String[] { "TLSv1" },  
 null,  
 SSLConnectionSocketFactory.*BROWSER\_COMPATIBLE\_HOSTNAME\_VERIFIER*);  
 httpClient = HttpClients.*custom*().setSSLSocketFactory(sslsf).build();  
 } else {  
 SSLContext sslContext = new SSLContextBuilder()  
 .loadTrustMaterial(null, new TrustStrategy() {  
 // 信任所有  
 public boolean isTrusted(X509Certificate[] chain,  
 String authType)  
 throws CertificateException {  
 return true;  
 }  
 }).build();  
 SSLConnectionSocketFactory sslsf = new SSLConnectionSocketFactory(  
 sslContext);  
 httpClient = HttpClients.*custom*().setSSLSocketFactory(sslsf)  
 .build();  
 }  
 } else {  
 httpClient = HttpClients.*createDefault*();  
 }  
 CloseableHttpResponse response = httpClient.execute(http);  
 try {  
 if (response != null) {  
 if (response.getStatusLine() != null)  
 statusCode = response.getStatusLine().getStatusCode();  
 HttpEntity entity = response.getEntity();  
 // 响应内容  
 content = EntityUtils.*toString*(entity, Consts.*UTF\_8*);  
 }  
 } finally {  
 response.close();  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 httpClient.close();  
 }  
 }  
 public int getStatusCode() {  
 return statusCode;  
 }  
 public String getContent() throws ParseException, IOException {  
 return content;  
 }  
}