微信APP支付服务端和Android 端详解及其demo

最近在开发APP微信支付和支付宝支付，Android 端和后端都是我自己开发的，发现两家公司的文档都不是很友好，特别是微信，接触过或者开发过的人都应该有所体会。因此我特意把开发的过程梳理了，做下记录，方便以后可能还用得到，同时也方便后来的一些开发者，希望如此吧。文章较长，耐心看吧，因为这篇文章涉及到了服务端和安卓端的开发。如果你是服务端开发者，那就只需要看服务端部分，如果是Android开发者，就只需要看Android部分即可。

这篇整理的是APP微信支付服务端和Android 端，文末有服务端和Android demo的下载链接。

1. 准备工作

微信支付开发需要用到应用id（appid）、商户号（mch\_id）和秘钥(key)，为了获取这几个重要的参数，需要我们做以下几个步骤获取。如果已经有了这几个参数则可以直接跳过这一步。

1.创建应用（获取appid）

要开发APP微信支付，需要在微信开放平台（[http://open.weixin.qq.com](http://open.weixin.qq.com/)）上创建应用以获得应用id。微信有几个平台，一定要搞清楚，否则开发过程会觉得很混乱。最好先把这几个平台的作用和几个重要的名词搞清楚，官方的链接：<https://pay.weixin.qq.com/wiki/doc/api/app/app.php?chapter=2_2>

注册/登录开发平台后进入管理中心创建应用，根据提示填写相应资料后提交审核，审核结果一般三天工作日左右可以查看。注意应用签名不要搞错，推荐使用官方工具生成。



审核通过之后，在管理中心页面即可看到成功创建的应用，点击查看进入详情页面即可看到appid，如图：



2.申请商户号

商户号通过微信支付成功申请开通后获得，通过后可以在申请的邮件中找到，也可以在微信商户平台/账户中心/账户设置/API安全中找到，就不贴图了。

3.设置秘钥

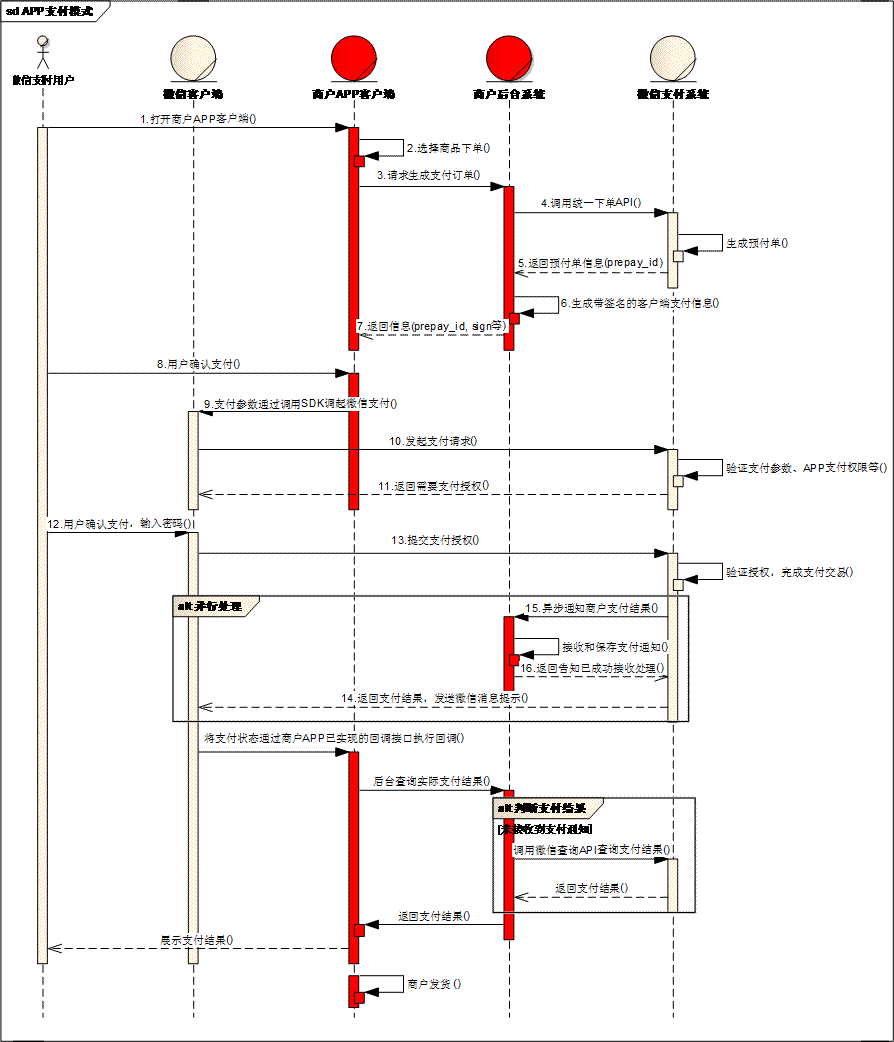
秘钥由我们自己设置,可以在微信商户平台按照提示要求进行设置。也可按一下路径设置：微信商户平台(pay.weixin.qq.com)-->账户中心-->账户设置-->API安全-->密钥设置

4.下载证书

我们服务端开发用的是官方封装的SDK，支付部分需要用到证书，如果所有逻辑都是自己写的话也可以不用，但是与支付相关的其他接口就必须用到了,所以干脆都用证书的方式进行开发。在微信商户平台(pay.weixin.qq.com)-->账户中心-->账户设置-->API安全这个页面有下载的按钮，也不贴图了。

1. 梳理流程

先看下官方文档中交互时序图及其说明，这些官方文档都有说明了，但我还是想补充些说明。



商户系统和微信支付系统主要交互说明：

步骤1：用户在商户APP中选择商品，提交订单，选择微信支付。

步骤2：商户后台收到用户支付单，调用微信支付统一下单接口。

步骤3：统一下单接口返回正常的prepay\_id，再按签名规范重新生成签名后，将数据传输给APP。参与签名的字段名为appid，partnerid，prepayid，noncestr，timestamp，package。注意：package的值格式为Sign=WXPay

步骤4：商户APP调起微信支付。

步骤5：商户后台接收支付通知。

步骤6：商户后台查询支付结果。

**补充说明：**

1.商户服务端主要负责步骤2、步骤3中的签名、步骤5和步骤6的结果处理。在不少的应用中展示支付结果是不依赖于步骤6的，APP端调起支付后在回调类中直接展示支付结果，这样商户服务端也可以不需要步骤6。虽然这样基本都是没什么问题的，但是最好是按照官方的要求，展示支付结果要依赖于商户后台查询的支付结果。

2.APP端主要负责步骤1和步骤4，如果支付结果依赖于商户后台的查询结果，则还需要步骤6。

3.步骤3这一步骤，很多人在这里被坑了，一定不要用预支付接口返回来的签名，需要重新生成，而且参与的字段有且只有上面提到的6个，而且都是小写，其中时间戳的单位为秒—10位数，时间戳可以重新获取。

1. 服务端的代码（为了简化，这里的支付结果不依赖于商户后台的查询结果）

服务端开发的有两种方式，其一是可以按照官方文档，所有逻辑都由自己来写，其二是使用官方封装的SDK开发。我选择的是第二种，这个方便一些。

1.开发环境

开发工具 ：IntelliJ IDEA

构建工具 ：Maven

2.引入SDK依赖

2.1 Maven 构建

在pop.xml中添加

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

2.2如果使用Grade 构建，则在模块build.grade文件中的dependencies 范围内添加：

compile group: 'com.github.wxpay', name: 'wxpay-sdk', version: '0.0.3'

2.3如果是eclipse 或者 IDEA用Grade 构建的话，可以直接下载jar 再导入，至于怎么导入jar 包，网上有很多教程，这里略过。当然Maven也是可以jar包的，只是比较麻烦些，在我的支付宝支付开发文章里有maven 环境下导入jar包的教程。

微信支付SDK Jar包下载链接：https://download.csdn.net/download/m\_sicily/10643021

3.设置配置文件，在调用官方封装的SDK时需要

import com.github.wxpay.sdk.WXPayConfig;import java.io.ByteArrayInputStream;import java.io.File;import java.io.FileInputStream;import java.io.InputStream;public class WXConfigUtil implements WXPayConfig { private byte[] certData; public WXMyConfigUtil() throws Exception { String certPath = "证书地址";//从微信商户平台下载的安全证书存放的路径 File file = new File(certPath); InputStream certStream = new FileInputStream(file); this.certData = new byte[(int) file.length()]; certStream.read(this.certData); certStream.close(); } @Override public String getAppID() { return “填写你的appid”; } //parnerid @Override public String getMchID() { return “填写商户id”; } @Override public String getKey() { return “填写API密钥”; } @Override public InputStream getCertStream() { ByteArrayInputStream certBis = new ByteArrayInputStream(this.certData); return certBis; } @Override public int getHttpConnectTimeoutMs() { return 8000; } @Override public int getHttpReadTimeoutMs() { return 10000; }}4.后端生成的预支付订单号以及异步回调

4.1控制层

import com.example.wxpay.module.service.WXserviceImpl;

import com.example.wxpay.module.util.WxMD5Util;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/v1/weixin")

public class WXController {

@Autowired

private WXserviceImpl wxPayService;

//微信预支付接口

@PostMapping("/apppay.json")

public Map<String, String> wxPay(/\*@RequestParam(value = "userId") String user\_id,

@RequestParam(value = "totalFee") String total\_fee,

@RequestParam(required = false, value = "couponId", defaultValue = "") String coupon\_id\*/

) throws Exception {

String total\_fee = "1"; //微信支付钱的单位为分

String user\_id = "liming"; //如用户id等额外参数

String attach = "{\"user\_id\":\"" + user\_id + "\"}";

//请求预支付订单

Map<String, String> result = wxPayService.dounifiedOrder(attach, total\_fee);

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

WxMD5Util md5Util = new WxMD5Util();

//返回APP端的数据

//参加调起支付的签名字段有且只能是6个，分别为appid、partnerid、prepayid、package、noncestr和timestamp，而且都必须是小写

//参加调起支付的签名字段有且只能是6个，分别为appid、partnerid、prepayid、package、noncestr和timestamp，而且都必须是小写

//参加调起支付的签名字段有且只能是6个，分别为appid、partnerid、prepayid、package、noncestr和timestamp，而且都必须是小写

map.put("appid", result.get("appid"));

map.put("partnerid", result.get("mch\_id"));

map.put("prepayid", result.get("prepay\_id"));

map.put("package", "Sign=WXPay");

map.put("noncestr", result.get("nonce\_str"));

map.put("timestamp", String.valueOf(System.currentTimeMillis() / 1000));//单位为秒

// 这里不要使用请求预支付订单时返回的签名

// 这里不要使用请求预支付订单时返回的签名

// 这里不要使用请求预支付订单时返回的签名

map.put("sign", md5Util.getSign(map));

map.put("extdata", attach);

return map;

}

/\*\*

\* 订单支付异步通知

\*/

@RequestMapping(value = "/notify.json", method = {RequestMethod.GET, RequestMethod.POST})

public String wxPayNotify(HttpServletRequest request, HttpServletResponse response) {

String resXml = "";

try {

InputStream inputStream = request.getInputStream();

//将InputStream转换成xmlString

BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));

StringBuilder sb = new StringBuilder();

String line = null;

try {

while ((line = reader.readLine()) != null) {

sb.append(line + "\n");

}

} catch (IOException e) {

System.out.println(e.getMessage());

} finally {

try {

inputStream.close();

} catch (IOException e) {

e.printStackTrace();

}

}

resXml = sb.toString();

String result = wxPayService.payBack(resXml);

return result;

} catch (Exception e) {

System.out.println("微信手机支付失败:" + e.getMessage());

String result = "<xml>" + "<return\_code><![CDATA[FAIL]]></return\_code>" + "<return\_msg><![CDATA[报文为空]]></return\_msg>" + "</xml> ";

return result;

}

}

}

4.2服务层：

import java.util.Map;

public interface WXservice {

Map<String, String> dounifiedOrder(String attach, String total\_fee) throws Exception;

String payBack(String notifyData);

}

import com.example.wxpay.module.util.WXConfigUtil;

import com.example.wxpay.module.util.WxMD5Util;

import com.github.wxpay.sdk.WXPay;

import com.github.wxpay.sdk.WXPayUtil;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Service;

import java.util.HashMap;

import java.util.Map;

@Service

public class WXserviceImpl implements WXservice {

private static final Logger logger = LoggerFactory.getLogger("MainLogger");

public static final String SPBILL\_CREATE\_IP = "x.x.x.x";

public static final String NOTIFY\_URL = "http://域名/v1/weixin/notify.json";

public static final String TRADE\_TYPE\_APP = "APP";

@Override

public Map<String, String> dounifiedOrder(String attach, String total\_fee) throws Exception {

WxMD5Util md5Util = new WxMD5Util();

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

WXConfigUtil config = new WXConfigUtil();

WXPay wxpay = new WXPay(config);

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

//生成商户订单号，不可重复

String out\_trade\_no = "wxpay" + System.currentTimeMillis();

data.put("appid", config.getAppID());

data.put("mch\_id", config.getMchID());

data.put("nonce\_str", WXPayUtil.generateNonceStr());

String body = "订单支付";

data.put("body", body);

data.put("out\_trade\_no", out\_trade\_no);

data.put("total\_fee", total\_fee);

//自己的服务器IP地址

data.put("spbill\_create\_ip", SPBILL\_CREATE\_IP);

//异步通知地址（请注意必须是外网）

data.put("notify\_url", NOTIFY\_URL);

data.put("trade\_type", TRADE\_TYPE\_APP);

//附加数据，在查询API和支付通知中原样返回，该字段主要用于商户携带订单的自定义数据

data.put("attach", attach);

String sign1 = md5Util.getSign(data);

data.put("sign", sign1);

try {

//使用官方API请求预付订单

Map<String, String> response = wxpay.unifiedOrder(data);

System.out.println(response);

String returnCode = response.get("return\_code"); //获取返回码

//若返回码为SUCCESS，则会返回一个result\_code,再对该result\_code进行判断

if (returnCode.equals("SUCCESS")) {//主要返回以下5个参数

String resultCode = response.get("result\_code");

returnMap.put("appid", response.get("appid"));

returnMap.put("mch\_id", response.get("mch\_id"));

returnMap.put("nonce\_str", response.get("nonce\_str"));

returnMap.put("sign", response.get("sign"));

if ("SUCCESS".equals(resultCode)) {//resultCode 为SUCCESS，才会返回prepay\_id和trade\_type

//获取预支付交易回话标志

returnMap.put("trade\_type", response.get("trade\_type"));

returnMap.put("prepay\_id", response.get("prepay\_id"));

return returnMap;

} else {

//此时返回没有预付订单的数据

return returnMap;

}

} else {

return returnMap;

}

} catch (Exception e) {

System.out.println(e);

//系统等其他错误的时候

}

return returnMap;

}

/\*\*

\* 支付结果通知

\* 官方文档 ：https://pay.weixin.qq.com/wiki/doc/api/app/app.php?chapter=9\_7&index=3

\*

\* @param notifyData 异步通知后的XML数据

\* @return

\*/

@Override

public String payBack(String notifyData) {

WXConfigUtil config = null;

try {

config = new WXConfigUtil();

} catch (Exception e) {

e.printStackTrace();

}

WXPay wxpay = new WXPay(config);

String xmlBack = "";

Map<String, String> notifyMap = null;

try {

notifyMap = WXPayUtil.xmlToMap(notifyData); // 调用官方SDK转换成map类型数据

if (wxpay.isPayResultNotifySignatureValid(notifyMap)) {//验证签名是否有效，有效则进一步处理

String return\_code = notifyMap.get("return\_code");//状态

String out\_trade\_no = notifyMap.get("out\_trade\_no");//商户订单号

if (return\_code.equals("SUCCESS")) {

if (out\_trade\_no != null) {

// 注意特殊情况：订单已经退款，但收到了支付结果成功的通知，不应把商户的订单状态从退款改成支付成功

// 注意特殊情况：微信服务端同样的通知可能会多次发送给商户系统，所以数据持久化之前需要检查是否已经处理过了，处理了直接返回成功标志

//业务数据持久化

System.err.println("支付成功");

logger.info("微信手机支付回调成功订单号:{}", out\_trade\_no);

xmlBack = "<xml>" + "<return\_code><![CDATA[SUCCESS]]></return\_code>" + "<return\_msg><![CDATA[OK]]></return\_msg>" + "</xml> ";

} else {

logger.info("微信手机支付回调失败订单号:{}", out\_trade\_no);

xmlBack = "<xml>" + "<return\_code><![CDATA[FAIL]]></return\_code>" + "<return\_msg><![CDATA[报文为空]]></return\_msg>" + "</xml> ";

}

}

return xmlBack;

} else {

// 签名错误，如果数据里没有sign字段，也认为是签名错误

//失败的数据要不要存储？

logger.error("手机支付回调通知签名错误");

xmlBack = "<xml>" + "<return\_code><![CDATA[FAIL]]></return\_code>" + "<return\_msg><![CDATA[报文为空]]></return\_msg>" + "</xml> ";

return xmlBack;

}

} catch (Exception e) {

logger.error("手机支付回调通知失败", e);

xmlBack = "<xml>" + "<return\_code><![CDATA[FAIL]]></return\_code>" + "<return\_msg><![CDATA[报文为空]]></return\_msg>" + "</xml> ";

}

return xmlBack;

}

}

MD5加密：

import com.github.wxpay.sdk.WXPayConstants;

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Arrays;

import java.util.Map;

import java.util.Set;

public class WxMD5Util {

public String getSign(Map<String, String> data) throws Exception {

WXConfigUtil config = new WXConfigUtil();

Set<String> keySet = data.keySet();

String[] keyArray = keySet.toArray(new String[keySet.size()]);

Arrays.sort(keyArray);

StringBuilder sb = new StringBuilder();

for (String k : keyArray) {

if (k.equals(WXPayConstants.FIELD\_SIGN)) {

continue;

}

if (data.get(k).trim().length() > 0) // 参数值为空，则不参与签名

sb.append(k).append("=").append(data.get(k).trim()).append("&");

}

sb.append("key=").append(config.getKey());

MessageDigest md = null;

try {

md = MessageDigest.getInstance("MD5");

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

byte[] array = new byte[0];

try {

array = md.digest(sb.toString().getBytes("UTF-8"));

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

StringBuilder sb2 = new StringBuilder();

for (byte item : array) {

sb2.append(Integer.toHexString((item & 0xFF) | 0x100).substring(1, 3));

}

return sb2.toString().toUpperCase();

}

}

后期我会把查询订单等功能也加上。

1. 安卓端

官方文档

快速接入文档1：

<https://open.weixin.qq.com/cgi-bin/showdocument?action=dir_list&t=resource/res_list&verify=1&id=1417751808&token=61982a3f18457e55b6afa7bb0103887f08dc7c90&lang=zh_CN>

开发步骤文档2: <https://pay.weixin.qq.com/wiki/doc/api/app/app.php?chapter=8_5>

1. 根据文档2的步骤创建应用获得AppId。
2. 根据文档1的步骤在项目中引入支付SDK，注意新版和旧版包名的不同。
3. 在AndroidManifest.xml 清单配置文件中添加必要的权限

<uses-permission android:name="android.permission.INTERNET"/>  
  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>  
  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>  
  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>  
  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

1. 在调起支付之前一定记得要注册微信，否则无法调起支付。注册微信有几种方式，其中一种如下：IWXAPI wxapi = WXAPIFactory.createWXAPI(this, APP\_ID,false);
2. 注意回调的类的位置和类名有严格规定，不能更改。微信支付回调的类类名必须是WXPayEntryActivity，类的路径必须是:应用包名+wxipa，否则微信将无法回调。如下图：



1. 在AndroidManifest.xml声明回调类的时候务必加上android:exported="true"，否则调不起支付界面。

<activity

android:name=".wxapi.WXPayEntryActivity"

android:exported="true"

android:launchMode="singleTop" />

1. 如果代码要混淆，加上以下代码以避免相关类被混淆

-keep class com.tencent.mm.opensdk.\*\* {

\*;

}

-keep class com.tencent.wxop.\*\* {

\*;

}

-keep class com.tencent.mm.sdk.\*\* {

\*;

}

1. 运行APP的时候，需要正式打包后的APP，或者在配置文件中指定相应的签名文件，否则支付返回都是-1。
2. 应该避免用户快速点击支付按钮。
3. 贴上Android端主要代码：

调起支付代码

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import com.hingin.l1.hiprint.common.utils.LogUtil;

import com.tencent.mm.opensdk.modelpay.PayReq;

import com.tencent.mm.opensdk.openapi.IWXAPI;

import com.tencent.mm.opensdk.openapi.WXAPIFactory;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.IOException;

import okhttp3.Call;

import okhttp3.Callback;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.Response;

public class WXPayActivity extends AppCompatActivity {

public static final String APP\_ID = "你的AppId";

OkHttpClient okHttpClient = new OkHttpClient.Builder().build();

private static final String TAG = "PayActivity";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_wxpay);//界面只是一个按钮

final IWXAPI wxapi = WXAPIFactory.createWXAPI(this, APP\_ID,false);

final Button button = findViewById(R.id.button);

button.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

button.setEnabled(false);

String url = "商户服务端接口";

Request request = new Request.Builder().url(url).build();

Call call = okHttpClient.newCall(request);

call.enqueue(new Callback() {

@Override

public void onFailure(Call call, IOException e) {

button.setEnabled(true);

Toast.makeText(WXPayActivity.this, "请求失败", Toast.LENGTH\_LONG).show();

}

@Override

public void onResponse(Call call, Response response) throws IOException {

if (response.isSuccessful()) {

try {

JSONObject jsonObject = new JSONObject(response.body().string());

LogUtil.INSTANCE.i(TAG,jsonObject.toString());

int code = jsonObject.getInt("code");

if (code == 0) {

JSONObject data = jsonObject.getJSONObject("data");

String appId = data.getString("appid");

String partnerId = data.getString("partnerid");

String prepayId = data.getString("prepayid");

String packageValue = data.getString("package");

String nonceStr = data.getString("noncestr");

String timeStamp = data.getString("timestamp");

String extData = data.getString("extdata");

String sign = data.getString("sign");

PayReq req = new PayReq();

req.appId = appId;

req.partnerId = partnerId;

req.prepayId = prepayId;

req.packageValue = packageValue;

req.nonceStr = nonceStr;

req.timeStamp = timeStamp;

req.extData = extData;

req.sign = sign;

Toast.makeText(WXPayActivity.this, "调起支付结果:" + wxapi.sendReq(req), Toast.LENGTH\_LONG).show();

// boolean result = wxapi.sendReq(req);

} else {

Toast.makeText(WXPayActivity.this, "数据出错", Toast.LENGTH\_LONG).show();

}

} catch (JSONException e) {

e.printStackTrace();

}

}

button.setEnabled(true);

}

});

}

});

}

}

回调代码

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import com.tencent.mm.opensdk.constants.ConstantsAPI;

import com.tencent.mm.opensdk.modelbase.BaseReq;

import com.tencent.mm.opensdk.modelbase.BaseResp;

import com.tencent.mm.opensdk.openapi.IWXAPI;

import com.tencent.mm.opensdk.openapi.IWXAPIEventHandler;

import com.tencent.mm.opensdk.openapi.WXAPIFactory;

public class WXPayEntryActivity extends Activity implements IWXAPIEventHandler {

private final String TAG = "WXPayEntryActivity";

private IWXAPI api;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//setContentView(R.layout.pay\_result);

api = WXAPIFactory.createWXAPI(this,”填写appid”);

api.handleIntent(getIntent(), this);

}

@Override

protected void onNewIntent(Intent intent) {

super.onNewIntent(intent);

setIntent(intent);

api.handleIntent(intent, this);

}

@Override

public void onReq(BaseReq req) {

}

@Override

public void onResp(BaseResp resp) {

Log.i(TAG,"errCode = " + resp.errCode);

//最好依赖于商户后台的查询结果

if (resp.getType() == ConstantsAPI.COMMAND\_PAY\_BY\_WX) {

Toast.makeText(WXPayActivity.this, "支付成功", Toast.LENGTH\_LONG).show(); }

}

finish();

}

1. 总结

凡事经历过了，觉得也就那么回事，但是第一次经历的时候，往往会觉得无厘头，特别是开发过程中出现问题的时候，有时还不好排查。有几个地方特别需要注意的虽然在代码中有注释了，但是还是想统一记录下。

服务端:

1. 如果开发哪功能出问题了，别急，先把该功能的相关文档详细过一遍，甚至多遍，这很重要，这能让你解决大部分问题了。错误的方式是，先去百度或者谷歌相关问题，因为这往往得不到你想要的答案，虽然官方文档有点坑，但是熟悉了之后就得还好吧。往往官方文档能给出最直接最准确的答案。
2. 检查申请的几个参数是有出入，特别是秘钥和应用签名。
3. 生成预支付订单后返回给移动端用的sign 参数一定不要用统一下单返回的sign,需要重新签名，而且签名的字段有且只有6个，而且都是小写，我就是在坑里滚了很久，我的key有大写的字母。
4. 两个单位值得注意下，涉及到时间戳的时候注意要用秒级(10位数),如果是毫秒级的，转换成秒级的；涉及到钱的要用分为单位，不能有小数点.
5. 订单号务必确保唯一，否则prepay\_id 返回null.

Android 端:

1. 在微信开放平台生成的应用签名不要搞错，最好用官方给的工具生成。
2. 在APP端调起支付的时候，需要正式打包的apk才能调起。或者在APP的build.grade文件中配置在线调式打包的签名和正式的一样，默认情况下在线打包用的是系统默认生成的签名，正式打包通常由开发者生成的新的签名。
3. 微信支付回调的类类名必须是WXPayEntryActivity，类的路径务必是:应用包名+wxipa，否则微信将无法回调（我记得以前开发微信登录分享等也有类似强制要求，这里我就不去验证了）。也就是说开发微信的相关功能的某些类必须在wxapi下，而且wxapi这个包必须在应用包下。我知道有点绕，查看上文的截图说明吧。这个只能说厂大了他们说了算吧
4. 配置文件中回调的activity 的exported 的属性必须为true ---android:exported="true"，name属性为exported不要改。

<activity

android:name=".wxapi.WXPayEntryActivity"

android:exported="true"

android:launchMode="singleTop" />

附链接：

服务端demo:

Github：<https://github.com/zouchuqu/wxpay>

csdn:

Android demo:

Github：<https://github.com/zouchuqu/wxdemo-Android.git>

CSDN: <https://download.csdn.net/download/m_sicily/10647840>

微信支付官方封装的SDK Jar包下载链接：

<https://download.csdn.net/download/m_sicily/10643021>

应用签名工具下载：

<https://open.weixin.qq.com/zh_CN/htmledition/res/dev/download/sdk/Gen_Signature_Android.apk>

欢迎关注我的公众号：kk鲁



或者我个人微信，开发过程遇到问题可以问我，我会试着尽量帮忙。

