支付宝、微信支付是开发中经常需要用到的功能，那么如何集成支付功能到应用中呢？支付结果亦是异步的，又如何传递结果呢？带着这些疑问让我们一步步走进这学习探索之路。
在这探索之前，想问下看客朋友是否查看过官方的电池状态监控示例。如果还没有的朋友建议先看一下官方的[源码](https://link.zhihu.com/?target=https://links.jianshu.com/go?to=https%3A%2F%2Fgithub.com%2Fflutter%2Fplugins%2Ftree%2Fmaster%2Fpackages%2Fbattery" \t "https://zhuanlan.zhihu.com/p/_blank)!
当你对官方源码都有所了解后，那下面的内容就容易理解多了。一般情况下，原生与Flutter通讯，我们经常使用MethodChannel处理方法调用，有时候需要事件监控，那就需要EventChannel来处理。说到这里，头脑里是否有一个大致流程构思了？！
首先，集成支付SDK，这个就不多说了。
其次，把支付方法封装起来。
最后，把支付方法提供给Flutter调用，事件响应给Flutter。
**1.支付宝支付辅助类代码**

class AliPayPlugin private constructor() {

 companion object {

 private var alipay: AliPayPlugin? = null

 @JvmStatic

 fun getInstance(): AliPayPlugin? {

 if (alipay == null) {

 synchronized(AliPayPlugin::class.java) {

 alipay = AliPayPlugin()

 }

 }

 return alipay

 }

 }

 @Suppress("unchecked\_cast")

 @SuppressLint("HandlerLeak")

 private val mHandler = object : Handler() {

 override fun handleMessage(msg: Message) {

 with(LocalBroadcastManager.getInstance(GenydfApplication.instance)) {

 val intent = Intent().apply {

 action = OnlinePayPlugin.ACTION\_ONLINE\_PAY\_RESULT\_NOTIFIER

 putExtra("payType", "ALIPAY")

 }

 when (msg.what) {

 0x0000A -> with(PayResult(msg.obj as Map<String, String>)) {

 /\*\* 对于支付结果，请商户依赖服务端的异步通知结果。同步通知结果，仅作为支付结束的通知。 \*\*/

 val resultInfo = if (!TextUtils.isEmpty(result)) result else memo// 同步返回需要验证的信息

 // 判断resultStatus 为9000则代表支付成功

 when {

 TextUtils.equals(resultStatus, "9000") -> intent.apply {

 putExtra("message", "支付宝支付成功")

 putExtra("state", 1)

 }

 TextUtils.equals(resultStatus, "6001") -> intent.apply {

 putExtra("message", "您取消了支付宝支付")

 putExtra("state", 0)

 }

 else -> intent.apply {

 putExtra("message", "支付失败，原因：$resultInfo")

 putExtra("state", -1)

 }

 }

 }

 }

 sendBroadcast(intent)

 }

 }

 }

 /\*\*

 \* 支付宝支付业务示例

 \* @param activity 上下文对象

 \* @param orderInfo 订单信息，来自服务器

 \*/

 fun startPay(activity: Activity, orderInfo: String): Unit = Thread {

 val result = PayTask(activity).payV2(orderInfo, true)

 val msg = Message()

 msg.what = 0x0000A

 msg.obj = result

 mHandler.sendMessage(msg)

 }.start()

}

/\*\*

 \* 支付结果实体类

 \* @param rawResult 支付Map数据结果

 \*/

class PayResult(rawResult: Map<String, String>?) {

 /\*\*

 \* @return the resultStatus

 \*/

 var resultStatus: String? = null

 private set

 /\*\*

 \* @return the result

 \*/

 var result: String? = null

 private set

 /\*\*

 \* @return the memo

 \*/

 var memo: String? = null

 private set

 init {

 rawResult?.let {

 for (key in rawResult.keys) {

 when {

 TextUtils.equals(key, "resultStatus") -> resultStatus = rawResult[key]

 TextUtils.equals(key, "result") -> result = rawResult[key]

 TextUtils.equals(key, "memo") -> memo = rawResult[key]

 }

 }

 }

 }

 override fun toString(): String = "resultStatus={$resultStatus};memo={$memo};result={$result}"

}

**2.微信支付辅助类代码**

/\*\*

 \* 微信插件

 \*/

class WechatPlugin private constructor() {

 companion object {

 private var wechatPlugin: WechatPlugin? = null

 @JvmStatic

 fun getInstance(): WechatPlugin? {

 if (wechatPlugin == null) {

 synchronized(WechatPlugin::class.java) {

 wechatPlugin = WechatPlugin()

 }

 }

 return wechatPlugin

 }

 }

 private val wxApi: IWXAPI? by lazy {

 val api = WXAPIFactory.createWXAPI(GenydfApplication.instance, BuildConfig.WX\_APPID, false)

 api.registerApp(BuildConfig.WX\_APPID)

 api

 }

 /\*\* 微信APP是否已经安装 \*\*/

 val isWXAppInstalled: Boolean by lazy { wxApi?.isWXAppInstalled ?: false }

 /\*\*

 \* 发送请求

 \* @param req 要请求的对象数据

 \*/

 fun sendRequest(req: BaseReq?) = wxApi?.sendReq(req)

 /\*\*

 \* 微信登录

 \* @param activity 上下文对象

 \*/

 fun login(activity: Activity) {

 if (wxApi?.isWXAppInstalled != true) {

 Toast.makeText(activity, "您的设备未安装微信，请安装后再登录！", Toast.LENGTH\_SHORT).show()

 return

 }

 with(SendAuth.Req()) {

 scope = "snsapi\_userinfo"

 state = "${System.currentTimeMillis()}"

 wxApi?.sendReq(this)

 }

 }

 /\*\*

 \* 回调，需要在WxEntryActivity中调用

 \* @param intent

 \* @param handler

 \*/

 fun handleIntent(intent: Intent?, handler: IWXAPIEventHandler) {

 wxApi?.handleIntent(intent, handler)

 }

}

/\*\* 微信支付 \*\*/

class WechatPayPlugin private constructor() {

 companion object {

 private var wechatPay: WechatPayPlugin? = null

 @JvmStatic

 fun getInstance(): WechatPayPlugin? {

 if (wechatPay == null) {

 synchronized(WechatPayPlugin::class.java) {

 wechatPay = WechatPayPlugin()

 }

 }

 return wechatPay

 }

 }

 /\*\*

 \* 发送支付请求

 \* @param params 支付请求参数

 \*/

 fun startPay(params: WechatPayParams): Unit = with(PayReq()) {

 appId = params.appId

 partnerId = params.partnerId

 prepayId = params.prepayId

 packageValue = params.packageValue

 nonceStr = params.nonceStr

 timeStamp = params.timeStamp

 sign = params.sign

 WechatPlugin.getInstance()?.sendRequest(this)

 }

}

/\*\*

 \* 微信支付参数实体类

 \* @param appId APP-ID

 \* @param packageValue 包名

 \* @param partnerId 合作者ID

 \* @param prepayId 准备的订单ID

 \* @param sign 签名字符串

 \* @param nonceStr 随机字符串

 \* @param timeStamp 时间戳

 \*/

data class WechatPayParams(

 @JvmField

 var appId: String? = BuildConfig.WX\_APPID,

 @JvmField

 var packageValue: String? = "Sign=WXPay",

 @JvmField

 var partnerId: String?,

 @JvmField

 var prepayId: String?,

 @JvmField

 var sign: String? = null,

 @JvmField

 var nonceStr: String? = null,

 @JvmField

 var timeStamp: String? = null

)

3.在线支付插件，按照V2版本插件编写的。
下面代码中利用MethodChannel为Flutter提供调用方法，使用EventChannel为支付结果响应事件，其中这里用的是LocalBroadcastManager进行支付结果通知，我们可以把通知的优先级提高一点，防止收到通知太慢的情况发生。

class OnlinePayPlugin : FlutterPlugin, ActivityAware, EventChannel.StreamHandler, MethodChannel.MethodCallHandler {

 companion object {

 private const val M\_NAME = "com.test.trade/online-pay-plugin"

 private const val E\_NAME = "com.test.trade/online-pay-plugin/pay-result"

 /\*\* 在线支付结果通知 \*\*/

 const val ACTION\_ONLINE\_PAY\_RESULT\_NOTIFIER = "com.test.trade.online\_pay.result"

 }

 private var applicationContext: Context? = null

 private var activity: Activity? = null

 private var payResultNotifierReceiver: BroadcastReceiver? = null

 private var methodChannel: MethodChannel? = null

 private var eventChannel: EventChannel? = null

 override fun onAttachedToEngine(binding: FlutterPlugin.FlutterPluginBinding) {

 onAttachedToEngine(binding.applicationContext, binding.binaryMessenger)

 }

 private fun onAttachedToEngine(applicationContext: Context, messenger: BinaryMessenger) {

 this.applicationContext = applicationContext

 methodChannel = MethodChannel(messenger, M\_NAME).apply {

 setMethodCallHandler(this@OnlinePayPlugin)

 }

 eventChannel = EventChannel(messenger, E\_NAME).apply {

 setStreamHandler(this@OnlinePayPlugin)

 }

 }

 override fun onDetachedFromEngine(binding: FlutterPlugin.FlutterPluginBinding) {

 methodChannel?.setMethodCallHandler(null)

 methodChannel = null

 eventChannel?.setStreamHandler(null)

 eventChannel = null

 }

 override fun onAttachedToActivity(binding: ActivityPluginBinding) {

 onAttachedToActivity(binding.activity)

 }

 private fun onAttachedToActivity(activity: Activity) {

 this.activity = activity

 }

 override fun onDetachedFromActivity() {}

 override fun onReattachedToActivityForConfigChanges(binding: ActivityPluginBinding) {}

 override fun onDetachedFromActivityForConfigChanges() {}

 override fun onListen(arguments: Any?, events: EventChannel.EventSink?) {

 payResultNotifierReceiver = object : BroadcastReceiver() {

 override fun onReceive(context: Context?, intent: Intent?) {

 if (intent?.action == ACTION\_ONLINE\_PAY\_RESULT\_NOTIFIER) {

 val payType = intent.getStringExtra("payType")

 val message = intent.getStringExtra("message")

 val state = intent.getIntExtra("state", -1)

 events?.success(mapOf(

 "type" to payType,

 "state" to state,

 "description" to message))

 }

 }

 }

 if (applicationContext != null && payResultNotifierReceiver != null)

 LocalBroadcastManager.getInstance(applicationContext!!).registerReceiver(payResultNotifierReceiver!!,

 IntentFilter(ACTION\_ONLINE\_PAY\_RESULT\_NOTIFIER))

 }

 override fun onCancel(arguments: Any?) {

 if (applicationContext != null && payResultNotifierReceiver != null)

 LocalBroadcastManager.getInstance(applicationContext!!).unregisterReceiver(payResultNotifierReceiver!!)

 payResultNotifierReceiver = null

 }

 override fun onMethodCall(call: MethodCall, result: MethodChannel.Result) {

 when (call.method) {

 "startPay" -> startOnlinePay(call, result)

 else -> result.notImplemented()

 }

 }

 /\*\*

 \* 启动在线支付

 \* @param call 方法Call

 \*/

 private fun startOnlinePay(call: MethodCall, result: MethodChannel.Result) {

 when (call.argument<String>("type")) {

 "ALIPAY" -> { //支付宝支付

 val payOrderInfo = call.argument<String>("payInfo")

 if (payOrderInfo.isNullOrEmpty()) {

 result.error("ONLINE\_PAY\_ARGUMENTS\_ERROR", "错误：支付参数不能为空", null)

 return

 }

 this.activity?.let { AliPayPlugin.getInstance()?.startPay(it, payOrderInfo) }

 result.success(null)

 }

 "WECHAT\_PAY" -> WechatPayPlugin.getInstance()?.startPay( //微信支付

 Gson().fromJson<WechatPayParams>(call.argument<String>("payInfo"), WechatPayParams::class.java))

 else -> result.error("ONLINE\_PAY\_TYPE\_ERROR", "错误：未知支付类型", null)

 }

 }

}

微信**WXPayEntryActivity .java**的处理

public class WXPayEntryActivity extends Activity implements IWXAPIEventHandler {

 private WechatPlugin wechatPlugin;

 @Override

 public void onCreate(Bundle savedInstanceState) {

 super.onCreate(savedInstanceState);

 wechatPlugin = WechatPlugin.getInstance();

 if (wechatPlugin != null)

 wechatPlugin.handleIntent(getIntent(), this);

 }

 @Override

 protected void onNewIntent(Intent intent) {

 super.onNewIntent(intent);

 setIntent(intent);

 if (wechatPlugin == null)

 wechatPlugin = WechatPlugin.getInstance();

 if (wechatPlugin != null)

 wechatPlugin.handleIntent(getIntent(), this);

 }

 @Override

 public void onReq(BaseReq req) {

 }

 @Override

 public void onResp(BaseResp resp) {

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

 Intent intent = new Intent();

 intent.setAction(OnlinePayPlugin.ACTION\_ONLINE\_PAY\_RESULT\_NOTIFIER);

 intent.putExtra("payType", "WECHAT\_PAY");

 switch (resp.errCode) {

 case 0: //支付成功

 intent.putExtra("message", "微信支付成功");

 intent.putExtra("state", 1);

 break;

 case -1: //支付错误

 intent.putExtra("message", "微信支付失败, 错误：" + resp.errStr);

 intent.putExtra("state", -1);

 break;

 case -2: //用户取消

 intent.putExtra("message", "您已取消微信支付");

 intent.putExtra("state", 0);

 break;

 }

 LocalBroadcastManager.getInstance(this).sendBroadcast(intent);

 }

 finish();

 }

}

**4.Flutter部分实现，对接MethodChannel和EventChannel，类使用单例模式。**

/// 在线支付插件, 单例类

class OnlinePayPlugin {

 static const MethodChannel \_methodChannel =

 const MethodChannel('com.test.trade/online-pay-plugin');

 static const EventChannel \_eventChannel =

 const EventChannel('com.test.trade/online-pay-plugin/pay-result');

 static OnlinePayPlugin \_instance;

 factory OnlinePayPlugin() {

 if (\_instance == null) \_instance = OnlinePayPlugin();

 return \_instance;

 }

 Stream<OnlinePayResultInfo> \_onOnlinePayResultEvent;

 /// 开始支付

 /// + `info` 支付信息

 void startPay(OnlinePayInfo info) => \_methodChannel.invokeMethod('startPay', {

 'type': info.type == PayType.AliPay ? 'ALIPAY' : 'WECHAT\_PAY',

 'payInfo': info.payArguments is String

 ? info.payArguments

 : json.encode(info.payArguments)

 });

 /// 接收支付结果的事件

 Stream<OnlinePayResultInfo> get onOnlinePayResultEvent {

 if (\_onOnlinePayResultEvent == null) {

 \_onOnlinePayResultEvent = \_eventChannel

 .receiveBroadcastStream()

 .map((dynamic data) => OnlinePayResultInfo.fromJson(data));

 }

 return \_onOnlinePayResultEvent;

 }

}

/// 支付类型

enum PayType {

 ///支付宝支付

 AliPay,

 /// 微信支付

 WechatPay

}

///支付信息

class OnlinePayInfo {

 /// 支付类型

 PayType type;

 /// 支付参数

 dynamic payArguments;

 OnlinePayInfo({@required this.type, @required this.payArguments});

 Map<String, dynamic> toJson() => {

 'type': type.toString(),

 'payArguments':

 payArguments is String ? payArguments : json.encode(payArguments)

 };

}

/// 微信支付参数

class WechatPayArgumentsInfo {

 /// 应用ID

 String appId;

 /// 常量值：Sign=WXPay

 String packageValue;

 /// 合作者ID

 String partnerId;

 /// 预付订单ID

 String prepayId;

 /// 签名字符串

 String sign;

 /// 随机字符串

 String nonceStr;

 /// 时间戳

 String timeStamp;

 WechatPayArgumentsInfo();

 factory WechatPayArgumentsInfo.fromJson(Map<String, dynamic> json) =>

 WechatPayArgumentsInfo()

 ..appId = json['appId'] as String

 ..packageValue = json['packageValue'] as String

 ..partnerId = json['partnerId'] as String

 ..prepayId = json['prepayId'] as String

 ..sign = json['sign'] as String

 ..nonceStr = json['nonceStr'] as String

 ..timeStamp = json['timeStamp'] as String;

 Map<String, dynamic> toJson() => {

 'appId': appId,

 'packageVakue': packageValue,

 'partnerId': partnerId,

 'prepayId': prepayId,

 'sign': sign,

 'nonceStr': nonceStr,

 'timeStamp': timeStamp

 };

}

/// 支付结果状态

enum OnlinePayResultState {

 /// 支付成功

 Success,

 /// 支付失败

 Fail,

 /// 支付被取消

 Cancel

}

/// 在线支付结果信息

class OnlinePayResultInfo {

 /// 支付类型

 PayType type;

 /// 支付状态

 OnlinePayResultState state;

 /// 支付结果描述

 String description;

 OnlinePayResultInfo();

 factory OnlinePayResultInfo.fromJson(Map<String, dynamic> json) =>

 OnlinePayResultInfo()

 ..type = json['type'] == 'ALIPAY' ? PayType.AliPay : PayType.WechatPay

 ..state = json['state'] == 1

 ? OnlinePayResultState.Success

 : (json['state'] == -1

 ? OnlinePayResultState.Fail

 : OnlinePayResultState.Cancel)

 ..description = json['description'] as String;

 Map<String, dynamic> toJson() => {

 'type': type.toString(),

 'state': state.toString(),

 'description': description

 };

}

**5.调用测试**

void test() async {

 await OnlinePayPlugin().startPay(.....);

 var result = await OnlinePayPlugin().onOnlinePayResultEvent.single;

 debugPrint(result);

}