

# 支付宝移动开发框架 mPaas 研究报告

## 1. 关于 mPaas :

mPaas 是源于支付宝 App 的移动开发平台 , 为移动开发、测试、运营及运维提供的一站式解决方案 ;mPaas 提供了整套 UI 框架以及众多原生 API 接口实现, 用户可根据需求新增自定义接口实现 ;

## 2. mPaas 与原生交互实现 :

### 1) .JS 的调用实现 :

```
<body>
  <button id="J_Custom Jsapi_1">自定义 JSAPI-1</button>
  <button id="J_Custom Jsapi_2">自定义 JSAPI-2</button>
  <button id="J_Custom Jsapi_3">自定义 JSAPI-3</button>

  <script>
    function ready(callback) {
      // 如果 jsbridge 已经注入则直接调用
      if (window.AlipayJSBridge) {
        callback && callback();
      } else {
        // 如果没有注入则监听注入的事件 1.监听注入事件
        document.addEventListener('AlipayJSBridgeReady', callback, false);
      }
    }

    ready(function () {
      var jsapiNode = document.getElementById('J_Custom Jsapi_1');
      jsapiNode.addEventListener('click', function () {
        AlipayJSBridge.call('myApi01', {
          name: '测试01',
          content: '内容01' 2.统一使用 AlipayJSBridge,传入方法名和参数, 调用原生方法;
        }, function (result) {
          alert(JSON.stringify(result));
        });
      });
    });
  </script>
```

- JS 调用原生 :无需引入任何 js 文件, 只需注入 `AlipayJSBridgeReady` 即可 ;
- JS 调用原生 : 无需传入类名, 直接使用 `AlipayJSBridge` 传入方法名和参数, 即可触发原生接口, 完成调用 ;

## 2) .原生实现：

The screenshot shows the Xcode interface with the project structure on the left and the implementation of the native handler in the center.

**Project Structure (Left Panel):**

- mPaasDemo
  - AppDelegate.h
  - AppDelegate.m
  - MPLauncherAppDelegate.h
  - MPLauncherAppDelegate.m
  - DemoViewController.h
  - DemoViewController.m
  - MyJsApiHandler4.h (highlighted with a red box and labeled "具体实现类")
  - MyJsApiHandler4.m (highlighted with a red box)
  - MyJsApiHandler4Sms.h
  - MyJsApiHandler4Sms.m
  - MPH5WebViewController.h
  - MPH5WebViewController.m
  - H52Native.html
  - CustomJSAPI.html
  - Assets.xcassets
  - Main.storyboard
  - LaunchScreen.storyboard
  - Info.plist
  - main.m
- CustomJsApi.bundle
  - Poseidon-UserDefine-Extra-Config.plist (highlighted with a red box and labeled "创建自定义接口文件，完成实现类与方法关联绑定")

**Interface Builder (Right Panel):**

Key	Type	Value
Root	Dictionary	(1 item)
JsApiRuntime	Dictionary	(1 item)
JsApis	Array	(4 items)
Item 0	Dictionary	(2 items)
jsApi	String	myApi01
name	String	MyJsApiHandler4
Item 1	Dictionary	(2 items)
Item 2	Dictionary	(2 items)
Item 3	Dictionary	(2 items)

Red boxes and labels highlight the configuration in the Interface Builder:

- Red box around "jsApi" and "name" with label "关联绑定".
- Red box around "myApi01" with label "方法".
- Red box around "MyJsApiHandler4" with label "类名".

**Implementation (Center Panel):**

```
8
9 #import "MyJsApiHandler4.h" 实现类
10
11 @implementation MyJsApiHandler4 data: JS传入的参数
12 - (void)handler:(NSDictionary *)data context:(PSDContext *)context
    callback:(PSDJsApiResponseCallbackBlock)callback
13 {
14     [super handler:data context:context callback:callback];
15     PSDInvocationEvent * _event = [context valueForKey:@"_event"];
16     NSLog(@"调用接口方法名称: %@", _event.invocationName);
17     NSString * content = @"";
18     if ([data.allKeys containsObject:@"content"]) {
19         content = data[@"content"];
20     }
21     if ([content length] > 0) {
22         callback(@{
23             @"success":@YES,
24             @"message":content,
25             @"method":_event.invocationName JS调用的方法名
26         });
27     } else {
28         callback(
29             @{
30                 @"success":@NO,
31                 @"message":@"不是发送短信的接口",
32                 @"method":_event.invocationName
33             });
34     }
35 }
36
37 @end
```

Red boxes and labels highlight the implementation details:

- Red box around `MyJsApiHandler4` with label "实现类".
- Red box around `data` with label "data: JS传入的参数".
- Red box around `callback` with label "接口回调".

原生实现：

1. 创建 plist 文件，完成实现类与实现方法的关联绑定；

## 2. 编写具体实现：

- a) 所有实现类必须继承 `PSDJsApiHandler` 类；
- b) 完成具体实现：获取 JS 传参、方法名，实现具体业务逻辑；

## 3. mPaas 自带的崩溃处理机制：

mPaas 在接口具体实现类里，可自动拦截崩溃异常，如数组越界（实测没有给 JS 回调）、空指针、JSON 取值 Null 等，保证程序异常时不闪退，不影响用户正常操作；

## 4. 相关模块：

- a) 移动分析、实时发布、智能投放等均与 mPaas 官网控制台操作相关；  
所有崩溃日志、埋点数据、性能指标、热更新等只能在 mPaas 官方后台查看，对于统一壳意义不大；
- b) 社交分享、消息推送、设备标识、扫一扫、统一存储、定位等模块，统一壳基本已全部覆盖实现，没有借鉴意义；