SuperWebview 开发指南

本文档面向所有使用该 SDK 的开发人员、测试人员、管理人员以及对此感兴趣的其他用户。阅读该文档要求用户熟悉 Android 应用开发，了解 APICloud 平台，如果能对 HTML/CSS/JavaScript 有一定了解则更好。

目录

第一章 简介

SuperWebview 是 APICloud 官方推出的另一项重量级 API 生态产品，以 SDK 方式提供，致力于提升和改善移动设备 Webview 体验差的整套解决方案。APP 通过嵌入 SuperWebview 替代系统 Webview，即可在 Html5 中使用 APICloud 平台现有的所有端 API，以及包括增量更新、版本管理、数据云、推送云、统计分析、积木式模块化开发、所有已经聚合的开放平台服务等在内的云服务能力，增强用户体验，解决移动设备 Webview 使用过程中出现的兼容能力弱、加载速度慢、功能缺陷等任何问题，帮助开发者解决使用 Webview 过程中的所有痛点。

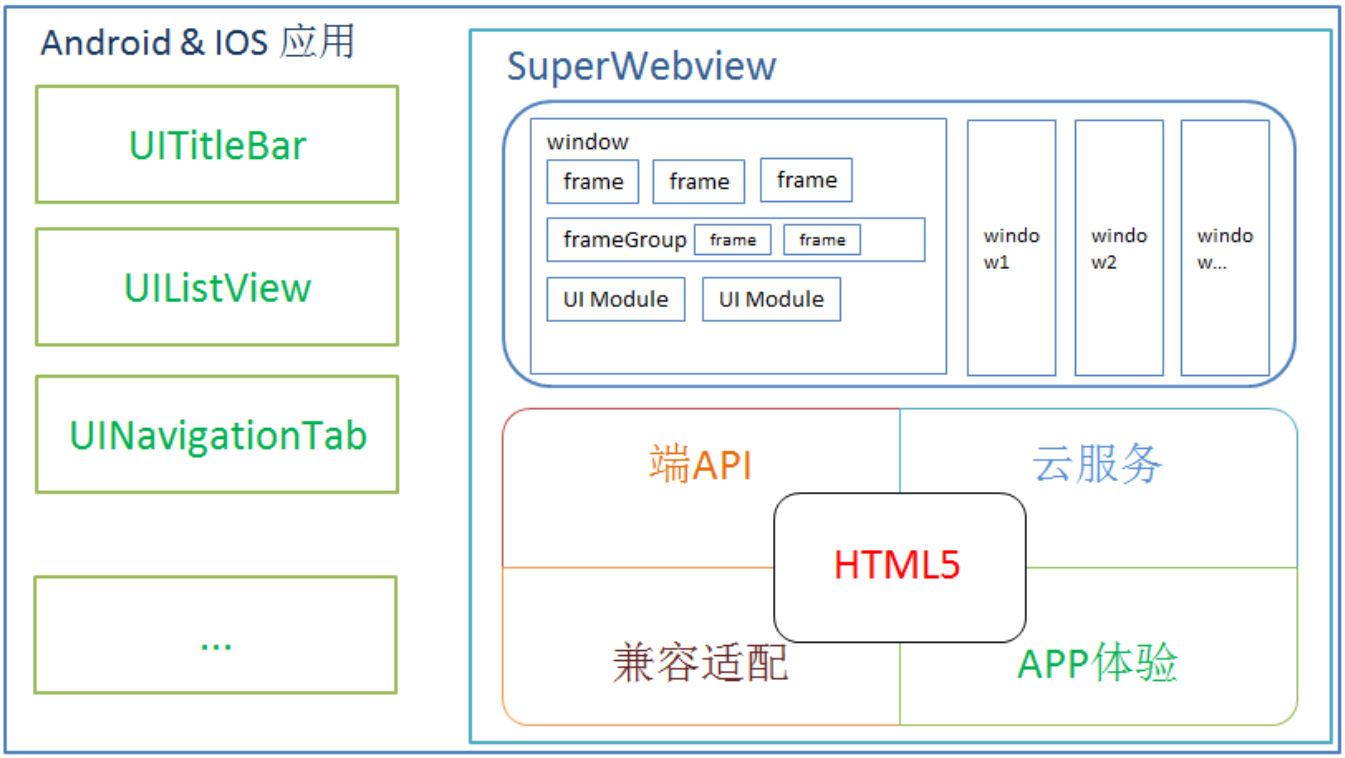
SuperWebview 继承 APICloud 终端引擎的包括跨平台能力，模块扩展机制，生命周期管理，窗口系统，事件模型，APP 级别的用户体验等在内的所有优秀能力，并且全面打通 Html5 与 Android 和 IOS 之间的交互，同步提供 APICloud 平台最新的 API 技术能力和服务，APICloud 团队将保持对 SuperWebview 的持续更新和优化，兼容 Html5 的新特性，持续推出优质服务。

第二章 架构设计

2.1 架构设计

SuperWebview 是 APICloud 终端引擎另一种形态下的开放 API，提供“NativeView 占主导，SuperWebview 层叠辅助”的混合能力，旨在帮助企业或者个人开发者已有的 Android 和 IOS 项目提供基于 Html5 能力的快速业务扩展，无缝使用 APICloud 云端一体能力提供的优质技术服务，同时保证最优的用户体验。您可以将 APP 中的某个或多个模块使用 SuperWebview 进行实现，您甚至可以将 SuperWebview SDK 当作独立的 APP 快速开发框架在混合开发中使用。结合 APICloud 终端引擎的跨平台能力和各项云服务能力，解决诸如跨平台，增量更新，快速版本迭代等 APP 开发过程中常见的痛点。

SuperWebview 整体 API 开放架构设计如下：



2.2 基本能力

SuperWebview 在继承系统 Webview 接口能力的基础上，主要提供以下功能的接口：

1、API 访问权限控制管理功能
2、Android/IOS 与 Html5 之间事件/数据交互功能
3、Web 与 Native 界面直接的混合布局和混合渲染功能
4、加速数据加载、点击响应和滚动速度
5、常用手势支持、界面切换动画
6、访问资源控制管理功能
7、执行 Html5 中指定 Javascript 脚本功能
8、模块扩展功能，该功能继承自 APICloud 终端引擎的模块扩展能力
9、Android&IOS 开发中常用的网络请求框架，缓存管理等工具接口
10、统一的生命周期管理，窗口系统，用户体验

第三章 使用准备和流程

3.1 运行环境

3.1.1 软硬件环境及要求

1)、本 SDK 要求最低 Android 系统版本为：2.3，Google API 级别为 9

2）、本 SDK 要求 APP 开启硬件加速，如关闭，会对渲染效率有一定的影响
3）、本 SDK 要求 Android 开发环境配置 Android SDK（ADT）的最低版本为 5.0，Google API 级别为 21
4）、本 SDK 依赖 Google 提供的最新版的兼容包，即最新版的 android-support-v4.jar

3.1.2 必须的权限：

使用 SuperWebviewSDK 的 APP 项目，必须在 AndroidManifest 中申请如下权限：

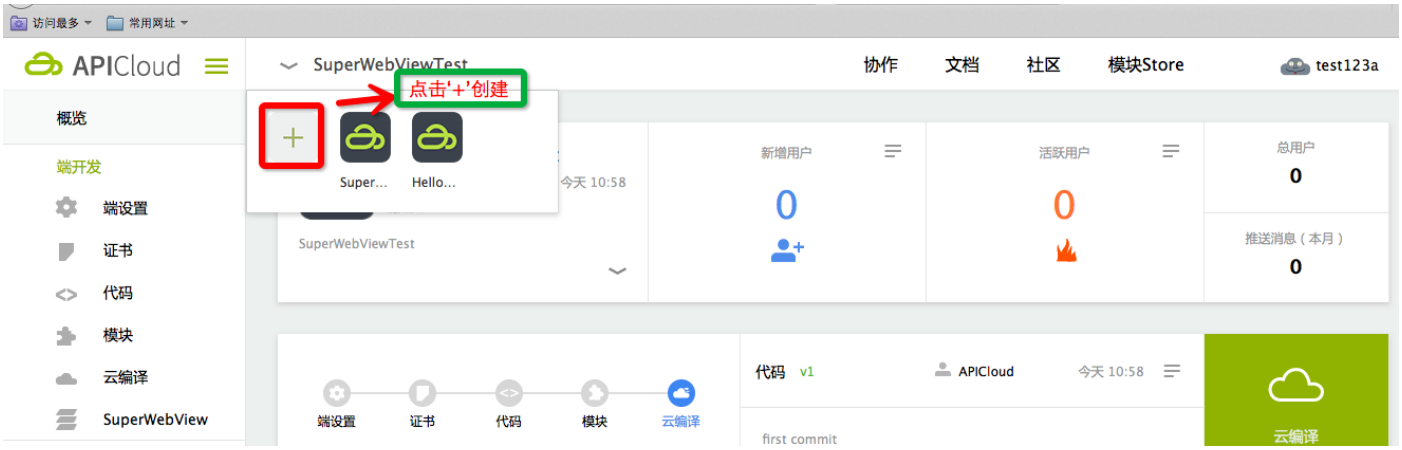
```
<!-- 访问网络  -->
<uses-permission  android:name="android.permission.INTERNET" />
<!-- 写外部存储  -->
<uses-permission  android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!-- 获取运营商信息，用于支持提供运营商信息相关的接口-->
<uses-permission  android:name="android.permission.ACCESS_NETWORK_STATE"  />
<!-- 访问 wifi 网络信息 -->
<uses-permission  android:name="android.permission.ACCESS_WIFI_STATE"  />
<!-- 读取手机当前的状态-->
<uses-permission  android:name="android.permission.READ_PHONE_STATE" />
```

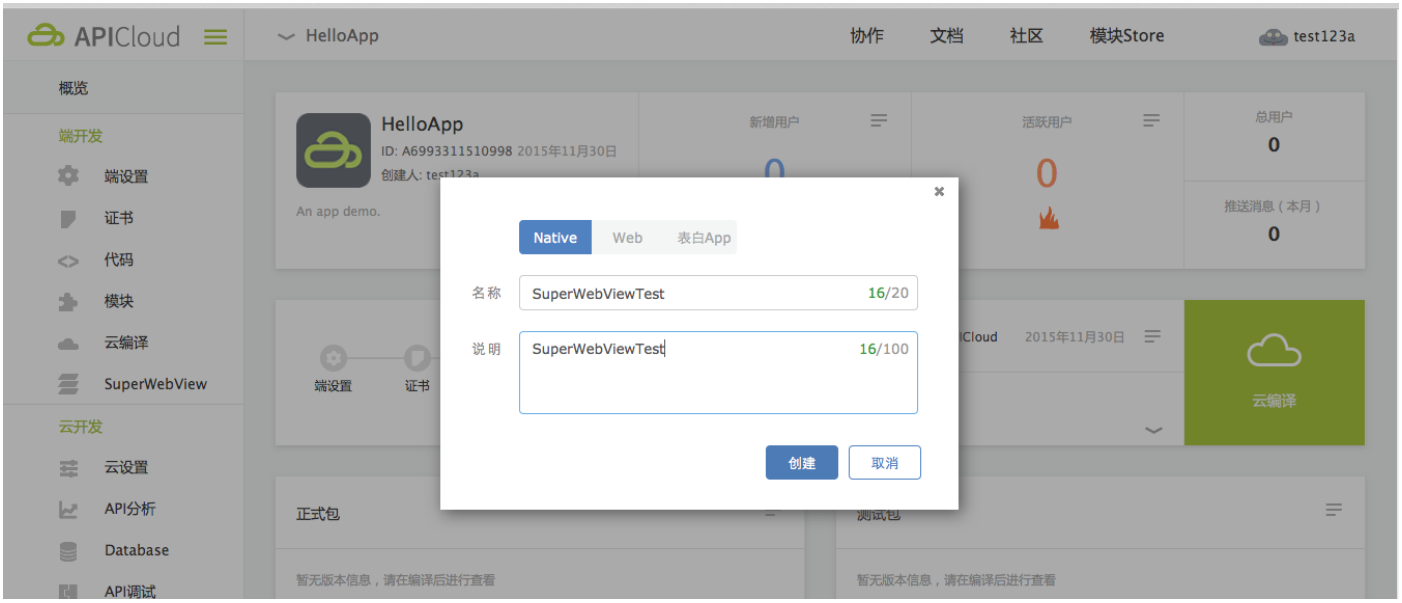
3.2 获取 SuperWebview SDK

3.2.1 创建和编译 SDK

1）、登录 APICloud 官网：http://www.apicloud.com，注册成为 APICloud 开发者

2）、进入控制台，在“概览”界面新建 APP 项目，如截图：





3)、点击控制台左侧的“模块”选项卡，导航至模块选择界面，如截图：

勾选您的项目中将要用到的模块，如果不需要，则略过此步骤

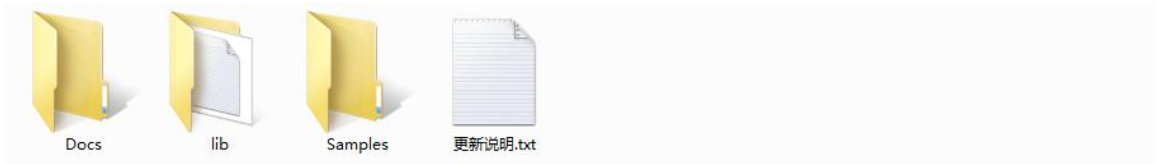
4)、点击控制台左侧的“SuperWebView”选项卡，导航至 SDK 编译界面，勾选您需要编译的平台，如截图：

5)、点击“编译 SDK”按钮，开始进行 SDK 的编译，等待片刻，编译完成后，页面中将展示 SDK 的下载链接，如下图：

6)、点击其中的下载链接，下载对应平台的 SDK 包至本地，并解压

3.3 SuperWebview SDK 包简介

本 SDK 为一个压缩文件包，其中包含 SDK 包一份（lib）、示例代码工程一份（Samples）、文档包一份（Docs），可能还包含一份更新说明。基本目录结构如下：



目录说明：

3.3.1 lib 目录

lib 目录下包含您在 APICloud 网站控制台编译的 SDK 包的所有资源，包括代码的 jar 包，so 库，必须的资源文件，AndroidManifest.xml 文件，在使用过程中需要将这些文件依次拷贝到您的 APP 项目中。lib 目录下共包含 libs、res、以及 AndroidManifest.xml 文件，可能还包含一个 assets 目录：

其中：

libs 目录下为 APICloud 引擎及所勾选模块的 jar 和 so 库；
res 目录下为 APICloud 引擎及所勾选的模块所需的资源文件，包括 xml，图片等文件；
assets 目录下为 APICloud 引擎及所勾选模块需要的 assets 类型资源文件；
AndroidManifest.xml 文件包含了 APICloud 引擎及所勾选的模块所需的权限、Activity、Service 等配置

3.3.2 Docs 目录

Docs 目录下包含本“开发者使用指南”以及一份 html 格式的详细 API 帮助文档，在开发过程中可随时参考该文档，获取满足您 APP 业务所需的各种 API。

3.3.3 Samples 目录

Samples 目录下为使用本 SDK 的几个不同场景下的 Demo，包含详细的代码注释。包含以下第四章中代码示例中的项目 ProjectFirst。

第四章 开始嵌入 SDK 到 APP

以下操作过程中，假设您现有或者新建的 APP 项目名称为“ProjectFirst”。
解压下载得到的 SDK 包到本地，得到 lib、Docs、Samples 等文件夹

4.1 添加 SDK 到 APP 工程

1、将 lib/libs 目录下的 so 库和 Jar 包拷贝到 ProjectFirst 中对应的 libs 目录下及 armeabi 目录中
2、将 lib/res 目录下的所有资源文件拷贝到 ProjectFirst 中对应的 res 同名目录中，注意 values 类型资源的合并
3、将 lib/AndroidManifest.xml 文件中的所有 permission、activity、provider、service、receiver 等全部拷贝到 ProjectFirst 中对应的 AndroidManifest 中
4、如果有 lib/assets 目录，则将 lib/assets 目录下的所有资源文件拷贝到 ProjectFirst 中对应的 assets 目录中

4.2 开始使用 API

4.2.1 初始化 SDK

SuperWebview SDK 中的所有 API 必须在显式的调用初始化函数后才能使用，建议在您项目入口 Application 的 onCreate 函数中调用 APICloud.initialize(Context)进行初始化。如果您的项目没有 Application，可以新建一个类继承自 Application，并在 AndroidManifest 中配置该类。比如 MyApplication：

```
package com.sdk.samples;

import android.app.Application;
import com.uzmap.pkg.openapi.APICloud;
```

```
public class MyApplication extends Application{

    @Override
    public void onCreate() {
        super.onCreate();
        APICloud.initialize(this);//初始化APICloud，SDK中所有的API均需要初始化后方可调用执行
    }
}
```

同时在 AndroidManifest 的 application 节点配置该类为其 name 属性：

```
    <application
        android:name="com.sdk.samples.MyApplication"
        android:icon="@drawable/ic_logo"
        android:label="@string/app_name">
```

4.2.2 使用 SuperWebview

新建 Activity 类，继承自 ExternalActivity（ExternalActivity 的帮助说明请参考第五章重要 API 说明中的介绍，或者阅读 API 文档中的详细介绍）。假设 ProjectFirst 项目中某版块是通过使用 Android 标准 API 中的 Webview 加载远程服务器的 Html5 页面所实现，该版块所在 Activity 为 WebPageModule 类，则将该类改造为继承自 ExternalActivity 类即可，或者自行新建 WebPageModule 类继承自 ExternalActivity：

```
package com.sdk.samples.apicloud;

import android.os.Bundle;
import com.uzmap.pkg.openapi.ExternalActivity;

/**
 *
 * 您原项目中加载远程Html5页面的版块，现用SuperWebview替换<br>
 * 使用SuperWebview的Activity，需继承自ExternalActivity
 * @author dexing.li
 *
 */
public class WebPageModule extends ExternalActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

同时，在 AndroidManifest 中配置 WebPageModule：

```
<activity android:name="com.sdk.samples.apicloud.WebPageModule"
          android:screenOrientation="portrait"
          android:windowSoftInputMode="adjustResize"
          android:theme="@android:style/Theme.Translucent.NoTitleBar"

    android:configChanges="orientation|locale|keyboardHidden|screenLayout|scre
enSize|smallestScreenSize|keyboard" />
```

此外，您可以通过重写 ExternalActivity 下的 shouldOverrideUrlLoading，onPageStarted，onPageFinished 等函数，实现 SuperWebview 与您项目原有业务的对接。

4.2.3 Html5 代码的编写和配置

在 ProjectFirst 项目的 assets 目录下新建名为 widget 的目录，并在该目录下新建名为 config 的 xml 文件，config.xml 文件要求编码为无 BOM 头 UTF-8 编码，如图：

关于 widget 目录的详细介绍，请登录 APICloud 网站参考《Widget 包结构说明》文档。
关于 config.xml 文件的详细说明及配置，请登录 APICloud 网站参考《APICloud 应用配置说明》文档。
配置完成后，开始编写第一张 Html 页面，例如 index.html，并将其在 config.xml 中配置为 content 字段，即声明该 Html 为 SuperWebview 的默认入口 Html 页：

```
<?xml version="1.0" encoding="UTF-8"?>
<widget id="A0000000000000" version="0.0.1">
  <name>ProjectFirst</name>
  <content src="index.html"/>
  <access origin="*"/>
</widget>
```

index.html 的详细代码请阅读 Samples 中的代码，其他 Html 代码的编写，按照您的产品设计进行即可。

关于 Html 代码的编写，也可通过下载 APICloud 开发工具进行项目的创建，编码，调试，开发完成后直接将整个项目的代码拷贝到 ProjectFirst 中的 assets/widget 目录下即可。使用 APICloud 开发工具开发，可为配合第七章高级功能中的某些操作提供基础。

4.2.4 启动 SuperWebview

在 ProjectFirst 中任意界面中，通过响应某可点击界面元素的点击事件，启动 WebPageModule 这个 Activity 即可：

```
@Override
public void onClick(View v) {
    //实现由之前的加载远程webapp的体验，转向由SuperWebview提供的原生APP体验
    if(v == mButtonWeb){
        Intent intent = new Intent(getContext(), WebPageModule.class);
        getContext().startActivity(intent);
    }
}
```

启动完 Activity 后，接下来 Html 页面的加载，渲染，逻辑执行等，SuperWebview 会自动帮您完成。
通过 SuperWebview 提供的 APICloud 终端引擎的能力，您基于 Html 的页面，可以无缝使用 APICloud 云端一体能力中提供的所有 API，达到原生 APP 级别的用户体验

接下来您可以通过结合第五章中的重要 API 说明以及 SuperWebview 的 API 文档来拓展您 APP 的各种场景下使用 SuperWebview 的能力。

第五章 重要 API 说明

本章节提供 SuperWebview SDK 中几个重要 API 的简单帮助说明，详细 API 文档请参考附件中的 API 文档包。本 SDK 开放的 API 主要位于“com.uzmap.pkg.openapi.*”以及“com.uzmap.pkg.uzkit.*”路径下。

“com.uzmap.pkg.openapi.*”中主要包含提供 SuperWebview SDK 能力的 Activity 等 UI 组件相关的类，主要包括 4 个重要的开放类：APICloud，ExternalActivity，Html5EventListener，WebViewProvider

“com.uzmap.pkg.uzkit.*”中主要包含一些工具类，如 HTTP 网络请求，Html5 相关工具函数，缓存处理工具等。

5.1 openapi 相关 API 类简介

类	描述及函数原型
APICloud	提供了所有使用 SuperWebview 的静态方法，在使用本 SDK 下的所有函数前，必须调用 APICloud.initialize(Context)函数初始化终端引擎
ExternalActivity	SuperWebview 开放的基本 UI 组件 Activity，您项目中任何使用 SuperWebview 能力的 Activity 必须继承自该 Activity。该类继承自系统 Activity，主要包括以下开放 API： 1）、evaluateJavascript(String) 向某个 window 或者 frame 执行一段 JS 脚本 2）、addHtml5EventListener(Html5EventListener)

	设置一个 Html5 事件监听器，监听来自 Html5 页面广播出来的事件 3）、removeHtml5EventListener(Html5EventListener) 移除一个 Html5 事件监听器 4）、removeAllHtml5EventListener() 移除所有 Html5 事件监听器 5）、sendEventToHtml5(eventName, extra) 发送一个事件并广播给当前所有已加载的 Html5 页面，如果其中有页面监听了该事件，它将收到广播 6）、onProgressChanged(WebViewProvider, progress) 当 Html5 页面的加载进度发生变化时，引擎将通过该接口回调给应用 7）、onPageStarted(WebViewProvider, url, pageIcon) 当某个 Html5 页面开始加载时，引擎将通过该接口回调给应用 8）、onPageFinished(WebViewProvider, url) 当某个 Html5 页面加载结束时，引擎将通过该接口回调给应用 9）、shouldOverrideUrlLoading(WebViewProvider, url) 当引擎内部某个 webview 即将请求加载一个 url 时，将通过该接口通知应用，如果应用拦截并自行处理，引擎将不再处理该请求。 10）、onReceivedTitle(WebViewProvider, title) 当引擎内部某个 webview 收到 Html5 页面标题时，将通过该接口回调给应用 11）、shouldForbiddenAccess(host, module, api) 当某 host 地址的 html 页面即将访问某 api 时，将通过该接口通知应用，应用可以决定是否拦截，控制其不允许访问 12）、如果启动 ExternalActivity 实现的子类 Activity 时，Intent 中传入了“startUrl”字段，那么该 Activity 将自动以该 startUrl 配置的 URL 作为首页加载入口。默认情况下使用 widget/config.xml 中的 content 字段配置作为入口
Html5EventListener	Html5 页面事件监听器，该监听允许您监听来自 Html5 页面通过 api.sendEvent 发出的自定义事件广播，实现原生应用与 Html5 之间的数据交互及解耦。 可通过 ExternalActivity.addHtml5EventListener(Html5EventListener)进行注册监听 Html5 事件将通过 Html5EventListener.onReceive(WebViewProvider, Object)回调
WebViewProvider	APICloud 终端引擎中 SuperWebview 的代理类，内部包含一个 APIWebview 的实例。该类以代理的形式开放了 webview 的诸多 API，如： 1）、getName() 获取 webview 所在 frame 的 name，与 api.openFrame 时传入的 name 对应，主 window 所在 frame 的 name 为固定值“main” 2）、getWinName() 获取 webview 所在 window 的 name，与 api.openWin 时传入的 name 对应，根 window 的 name 为固定值“root”

	3）、evaluateJavascript(script) 向该 webview 执行一段 js 脚本 4）、loadUrl(url) 请求该 webview 加载一条 url 5）、stopLoading() 要求当前 webview 停止加载 6）、reload() 要求 webview 重新加载当前页面 7）、goBack() 要求当前 webview 回退历史记录 8）、goForward() 要求当前 webview 前进历史记录 …. 等数十个 API

5.2 uzkit 相关 API 类简介

类	描述及函数原型
UZUtility	静态工具函数库，提供与 Html5 处理相关，APP 相关的工具函数
request.APICloudHttpClient	APICloud 终端引擎全局标准 HTTP 请求处理类，包含： request(Request) 插入一个 http 请求 cancelRequests(Object) 取消某个 http 请求 download(HttpDownload) 插入一个 http 下载请求 cancelDownload(Object) 取消某个 http 下载请求 disPlayImage(ImageOption, ImageView) 请求显示远程服务器上的某张图片资源到 ImageView …. 等诸多 HTTP 数据请求相关的函数，非常方便客户端向服务器发起数据请求
request.Request	所有类型 HTTP 请求的超类
request.HttpGet	HTTP 的 GET 请求类。通过 APICloudHttpClient.request(HttpGet)发起
request.HttpPost	HTTP 的 POST 请求类。通过 APICloudHttpClient.request(HttpPost)发起
request.HttpDownload	HTTP 的 GET 请求类，通常为请求下载非文本类型数据的大文件。 通过 APICloudHttpClient.download(HttpDownload)发起
request.HttpDelete	HTTP 的 DELETE 请求类。通过 APICloudHttpClient.request(HttpDelete)发起
request.HttpPut	HTTP 的 PUT 请求类。通过 APICloudHttpClient.request(HttpPut)发起
request.HttpHead	HTTP 的 HEAD 请求类。通过 APICloudHttpClient.request(HttpHead)发起

request.HttpOptions	HTTP 的 OPTIONS 请求类。通过 APICloudHttpClient.request(HttpOptions)发起
request.HttpPatch	HTTP 的 PATCH 请求类。通过 APICloudHttpClient.request(HttpPatch)发起
request.HttpTrace	HTTP 的 TRACE 请求类。通过 APICloudHttpClient.request(HttpTrace)发起
request.HttpParams	发起 POST/PUT/PATCH 请求时用于配置提交的数据
request.RequestCallback	HTTP 请求的回调，通过 request.setCallback(callback)使用
request.HttpResult	HTTP 请求的结果，通过 RequestCallback.onFinish(HttpResult)回调

第六章 其他

6.1 关于代码混淆：

如果开发者在发布版本时需要混淆自己的代码，请在混淆文件（一般默认为 Android 工程下 proguard-project.txt 或者 proguard.cfg 文件）中添加如下说明：

```
-libraryjars libs/apiEngine {verion}.jar
-dontwarn    com.uzmap.pkg.*
-keep class    com.uzmap.pkg.** { *;}
```

其中，{version}为引擎 jar 包对应的版本

6.2 SDK 固有资源说明及替换

原则上，SuperWebviewSDK 中包含的任何 XML、图片等资源文件，均需要原封不动的拷贝到您的项目中，如果熟悉 APICloud 平台的开发者，则可在此基础上替换或者更改这些资源，实现 UI 效果的自由定制。

6.3 一些开放源码

即将开放

第七章 高级功能

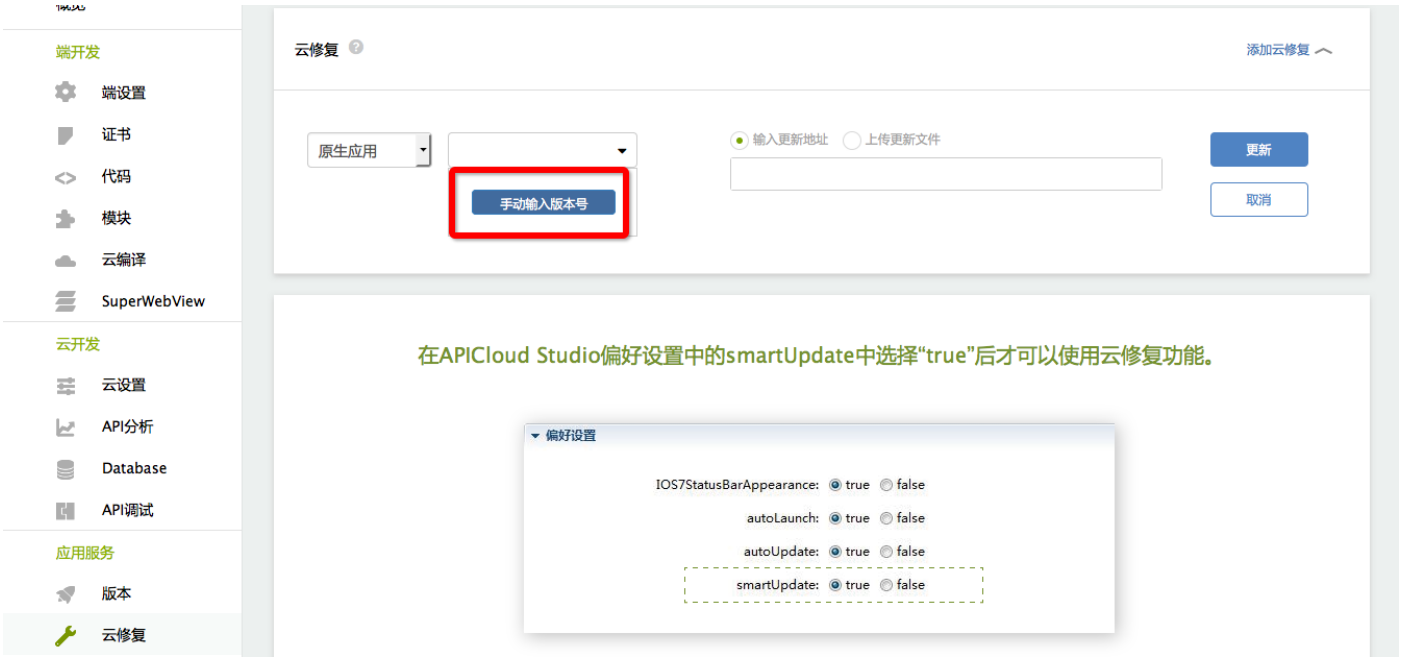
7.1 增量更新（云修复）

注意：使用云修复功能的 SuperWebview，您 widget 的 config.xml 中必须开启“云修复功能”，即 config.xml 中必须显式的配置：<preference name="smartUpdate" value="true"/>为 true

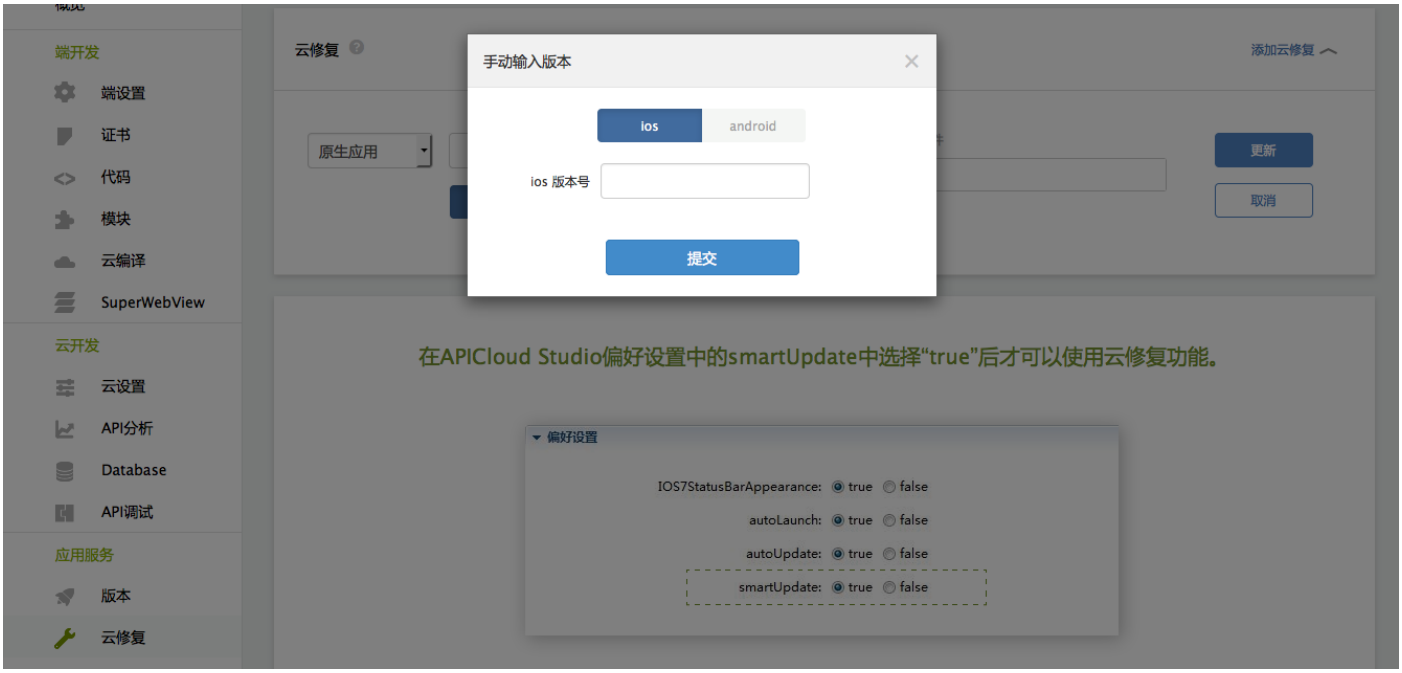
具体的使用流程：

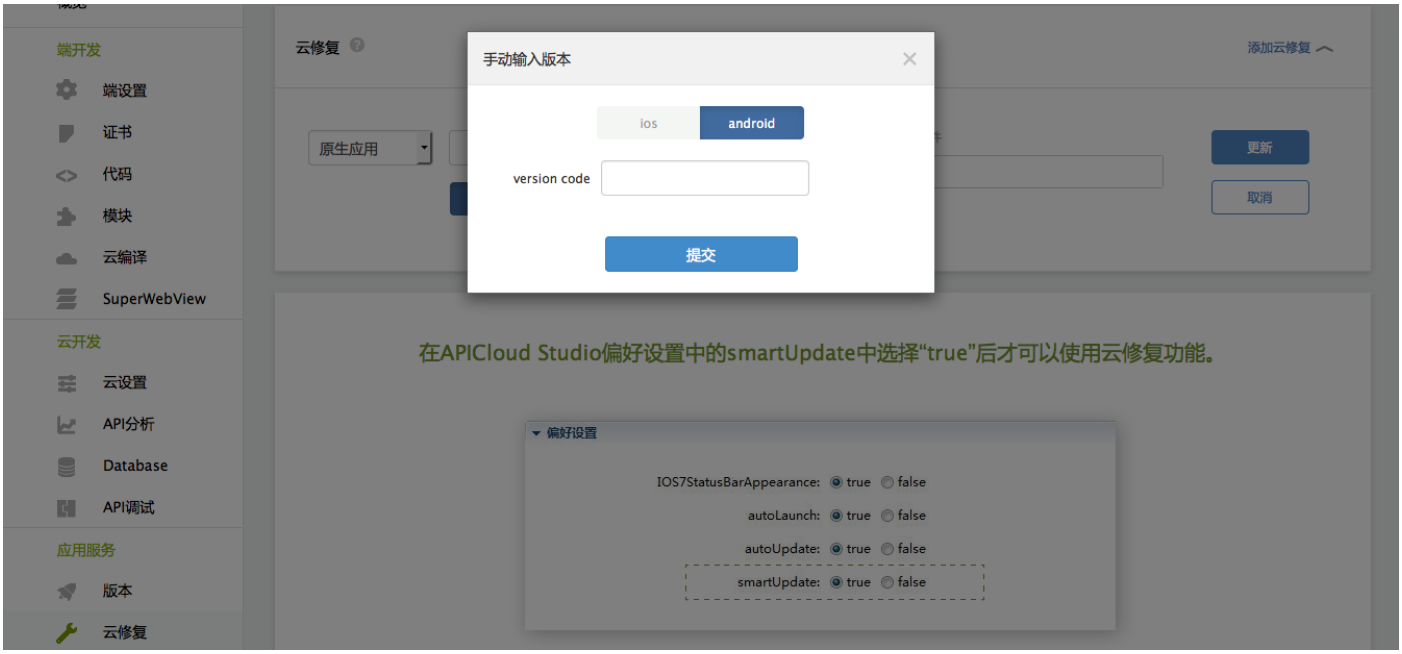
1)、在左侧控制台选择“云修复”，进入云修复页面，在下拉列表中选择“原生应用”，如截图：

2)、选择“原生应用”后，点击输入框提示需要“手动输入版本号”，如截图：

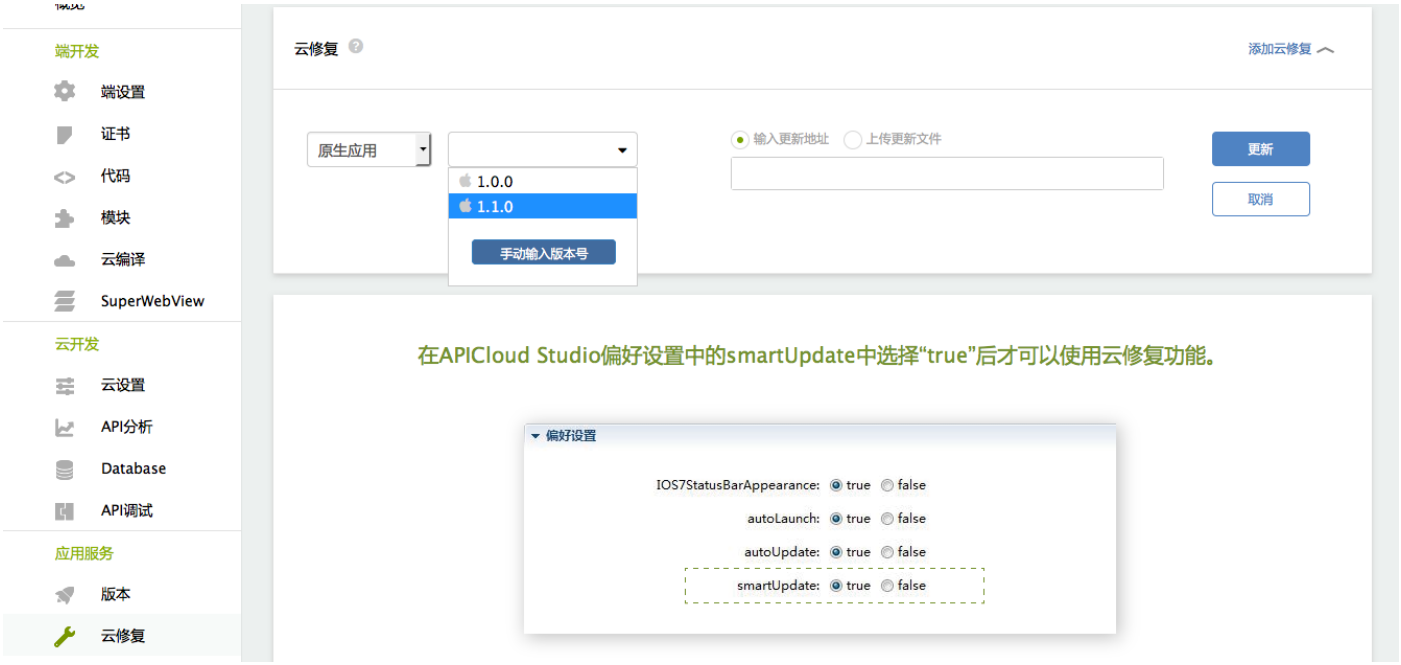


3）、点击“手动输入版本号”，如下图所示.
其中，IOS 要求输入您应用的版本号（即 PList 文件中的 CFBundleShortVersionString 字段值）；
Android 要求输入您应用的 versionCode（即 AndroidManifest.xml 文件中的“android:versionCode”字段值）；





4）、以 IOS 版本为例：输入版本号之后，需要选择相对应的版本进行云修复操作，如截图：



选择“1.1.0”作为修复版本，如截图：

5）、发布修复包：

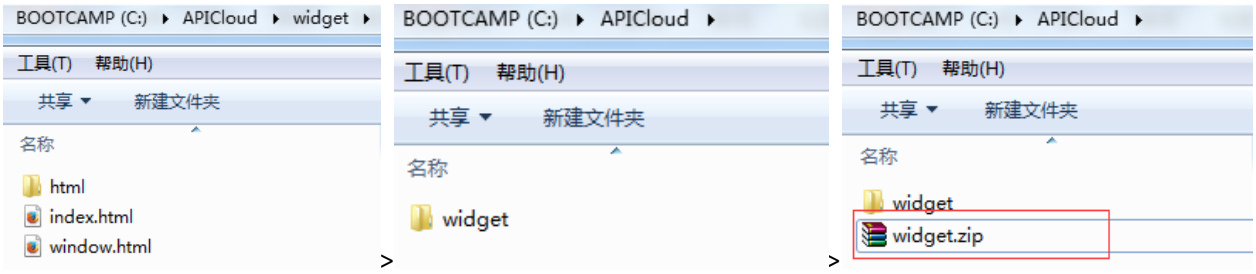
1、支持两种云修复方式：提示修复、静默修复。

提示修复：为您的某个版本应用发布提示修复包后，用户在打开应用时，将收到更新提示，如果用户选择更新，SuperWebview 将自动进行下载修复，如果用户选择不更新，则忽略。

静默修复：为您的某个版本应用发布静默修复包后，用户在打开应用时，SuperWebview 将在后台静默方式将更新包更新，更新完毕后会向 Html5 页面发出静默修复完成事件。用户在下次启动应用时，更新包将生效

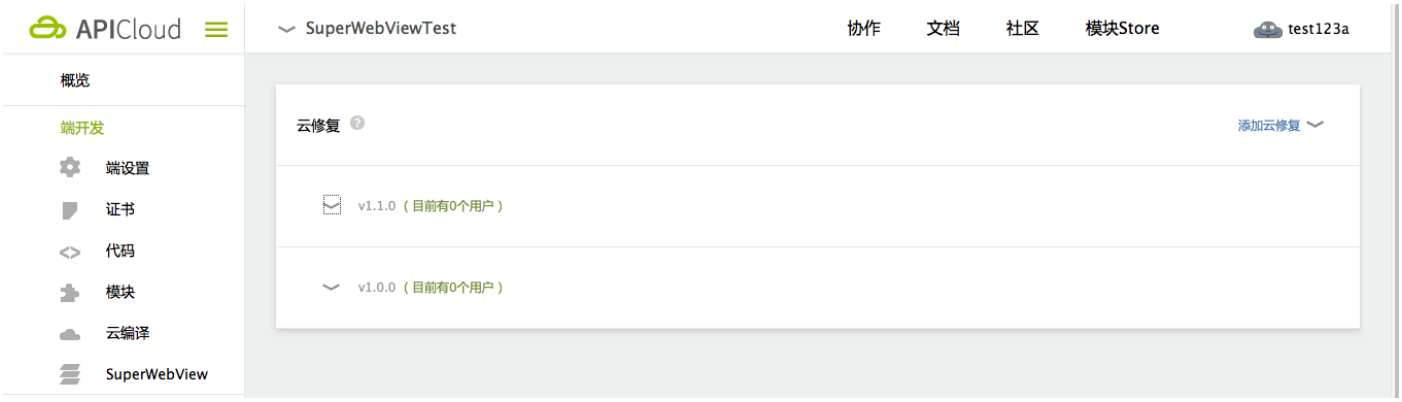
2、支持两种发布修复包方式：提供更新压缩包的远程 http 地址、直接上传更新压缩包；

更新压缩包格式要求为：根目录名称必须为 widget，子目录结构保持与项目中原 widget 目录一致，并只保留更新的文件，最后将 widget 目录压缩成 zip 格式的压缩包即可。大致的操作步骤如下图：



下面以提示修复为例，选择“提示修复”中“上传更新文件”如截图：

上传完成之后，点击右侧“更新”，成功之后会生成一条记录，如截图：



您可以为您应用的某个版本发布多个修复包，SuperWebview 将依照您的发布顺序，依次覆盖修复。
更新包发布完成后，用户再开启应用时，就会收到云修复

第八章 联系我们

如果以上信息无法帮助您解决在开发中遇到的具体问题，请通过以下方式联系我们：
Email：
Tel：
WebSite：http://community.apicloud.com/bbs/