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 iOS 应用开发，了解 APICloud 平台，如果能对 HTML/CSS/JavaScript 有一定了解则更好。

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、所有已经聚合的开方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。

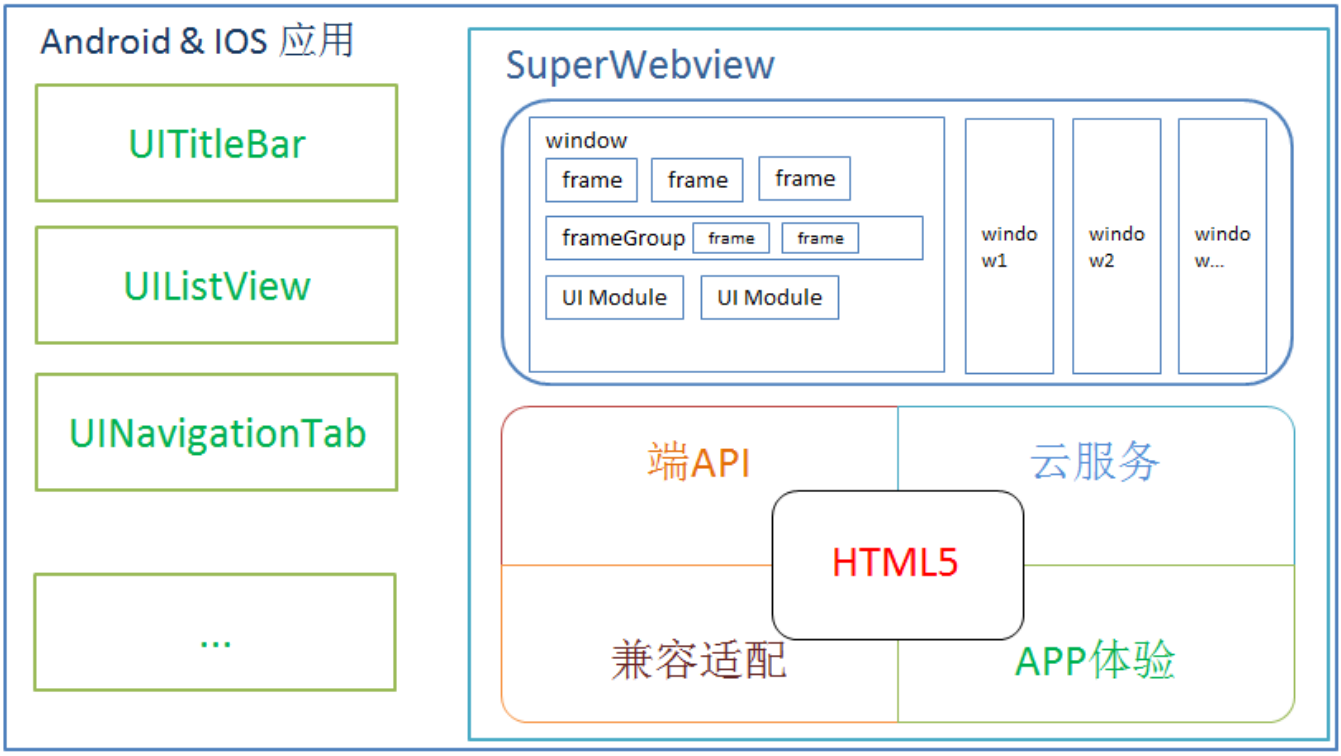
SuperWebview 继承 APICloud 终端引擎的包括跨平台能力，模块扩展机制，生命周期管理，窗口系统，事件模型，APP 级别的用户体验等在内的所有优秀能力，并且全面打通 Html5 与 Android 和 IOS 之间的交互，同步提供 APICloud 平台最新的 API 技术能力和服务，APICloud 团队将保持对 SuperWebview 的持续更新和优化，兼容 Html5 的新特性，持续推出优质服务。

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 支持 iOS7.0 及以上系统，支持 armv7、arm64 处理器架构。
2)、本 SDK 要求使用 xcode6 及以上版本

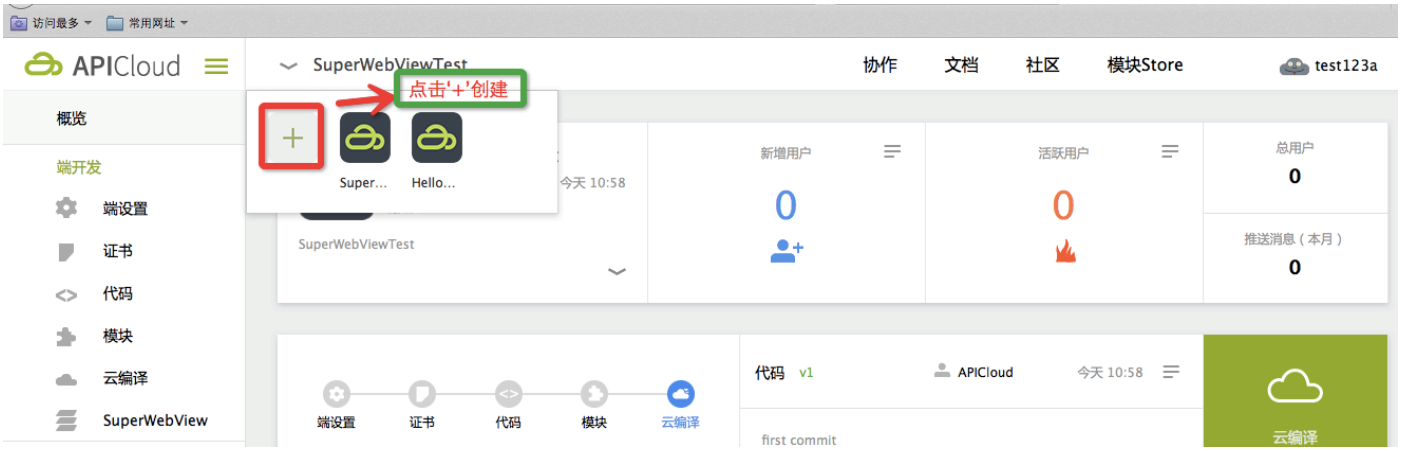
3.1.2 工程设置：

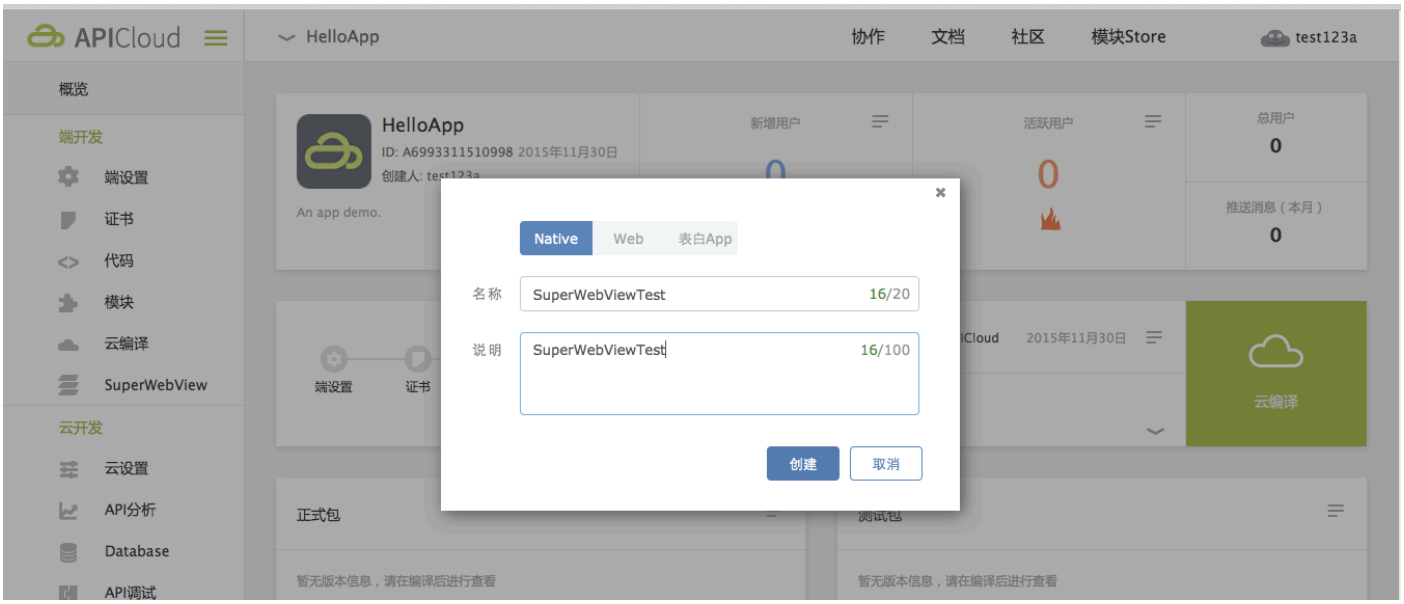
1)、找到项目工程的 TARGETS -> Build Phases -> Link Binary With Libraries，添加 SDK 用到的必需的库 libz.dylib、libicucore.dylib、libstdc++.dylib。若使用了模块可能还需要额外添加一些相应的库。
2)、找到项目工程的 TARGETS -> Build Settings -> Other Linker Flags，添加-ObjC 关键字。
3)、若是 Xcode7，找到项目工程的 TARGETS -> Build Settings -> Enable Bitcode，设置为 NO。
4)、 添加了模块用到的第三方 framework 到工程后，若编译时报 ld: framework not found xxx 之类的错误，那么找到项目工程的 TARGETS -> Build Settings -> Framework Search Paths，添加一下 framework 库所在的目录路径。

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

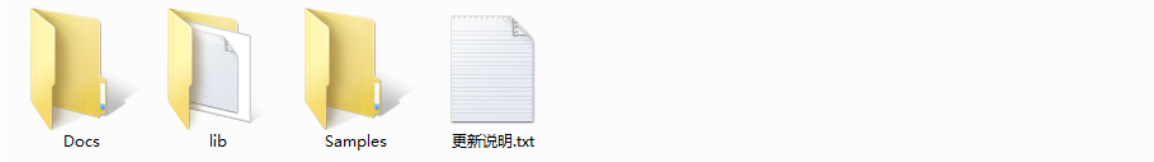
4)、点击控制台左侧的“SuperWebView”选项卡，导航至 SDK 编译界面，勾选您需要编译的平台，如截图：

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，如截图：

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，包括引擎和模块的库，以及使用到的资源文件，在使用过程中需要将这些文件及文件夹加到您的工程中。lib 目录下共包含 Engine、Modules 目录，可能还有 Info.plist 文件。

其中：
Engine 目录下为 APICloud 引擎及其头文件；
Modules 目录下为所勾选的模块及其所需的资源文件；
Info.plist 为一些模块如微信、qq 等需要用到的配置信息，需要将其中的内容拷贝或合并到您工程的 Info.plist 里面

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/Engine 目录下的库和头文件添加到 ProjectFirst 工程中，添加时选择 Create groups 选项。
2、将 lib/Modules 目录下的所有文件添加到 ProjectFirst 工程中，添加时选择 Create groups 选项，再把该目录下的所有文件夹也添加到工程中，添加时务必选择 Create folder references 选项。
3、找到项目工程的 TARGETS ->Build Phases ->Link Binary With Libraries，添加 SDK 用到的必需的库 libz.dylib、libicucore.dylib、libstdc++.dylib。
4、找到项目工程的 TARGETS->Build Settings->Other Linker Flags，添加-ObjC 关键字。
5、若是 Xcode7，找到项目工程的 TARGETS -> Build Settings->Enable Bitcode，设置为 NO。

4.2 开始使用 API

4.2.1 初始化 SDK

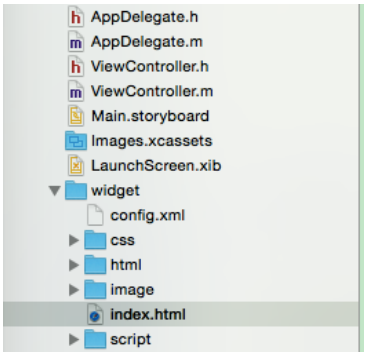
SuperWebview SDK 中的所有 API 必须在显式的调用初始化函数后才能使用，建议在您项目入口 AppDelegate 的-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中进行初始化：

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [[APIManager manager] initSDKWithLaunchOptions:launchOptions];

    ViewController *controller = [[ViewController alloc] init];
    UINavigationController *navi = [[UINavigationController alloc]
initWithRootViewController:controller];
    self.window = [[UIWindow alloc] initWithFrame:[UIScreen mainScreen].bounds];
    self.window.rootViewController = navi;
    [self.window makeKeyAndVisible];
    return YES;
}
```

4.2.2 Html5 代码的编写和配置

我们采用 widget 的形式来管理 html 代码，每个 APICloud 应用都有一个 widget 网页包，在前面章节获取 SDK 的页面中也可以获取该应用的 widget。这里我们 Samples 里面已经有了 widget 网页包，拷贝过来，将 widget 目录添加到工程中，添加时选择 Create folder references 选项。



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。
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替换掉之前的 widget。

4.2.3 启动 SuperWebview

这里我们在 ViewController 的视图中添加一个 button，在 button 的点击事件中来启动，其中 widget://为一个相对路径，表示 widget 的根目录，开发者也可以直接使用文件的绝对路径。至于选择 APIWindowContainer 还是 APIWidgetContainer 则取决于您当前项目的实际情况，APIWindowContainer 是一个 UIViewcontroller 对象，而 APIWidgetContainer 是一个 UINavigationController 对象：

```
- (IBAction)openSuperWebView:(UIButton *)button {
    button.highlighted = NO;

    // 这里的widget://表示widget的根目录路径
    NSString *url = @"widget://index.html";
    APIWindowContainer *windowContainer = [APIWindowContainer
windowContainerWithUrl:url name:@"root" userInfo:nil];
    [windowContainer startLoad];
    [self.navigationController pushViewController:windowContainer
animated:YES];
    self.windowContainer = windowContainer;
}
```

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 功能

本章节提供 SuperWebview SDK 中几个重要 API 的示例说明，详细 API 文档请参考附件中的 API 文档包。

5.1 在 html 页面中执行脚本

创建一个 APIWindowContainer 或者 APIWidgetContainer 对象后，可以在原生和 html 页面需要交互的时候通过 execScript:window:frame: 方法在指定的页面中执行脚本:

```
- (void)execScript {
    NSString *script = @"alert('在main页面执行脚本测试');";
    [self.windowContainer execScript:script window:@"root" frame:@"sudoku"];
}
```

5.2 事件机制

原生和前端 html 数据交互还可以使用事件机制，两者之间可以相互监听和发送事件。APIEventCenter 类提供了事件处理。

1)、发送一个事件给 html 页面，html 页面里面通过 api.addEventListener 方法来监听指定的事件：

```
- (void)sendEvent {
    [[APIEventCenter defaultCenter] sendEventWithName:@"showAlert"
userInfo:@{@"msg":@"成功发送事件给js"}];
}
```

2)、接收 html 页面里面发出的事件，html 页面里面通过 api.sendEvent 方法发送事件：

```
//注册监听
[[APIEventCenter defaultCenter] addEventListener:self
selector:@selector(handleEvent:) name:@"abc"];

//监听方法
- (void)handleEvent:(APIEvent *)event {
    NSString *msg = [NSString stringWithFormat:@"收到来自Html5的%@事件，传入的参
数为:%@", event.name, event.userInfo];
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"" message:msg
delegate:nil cancelButtonTitle:@"确定" otherButtonTitles:nil, nil];
    [alert show];
}
```

3)、移除指定事件或者全部事件的监听：

```
//移除指定事件监听
[[APIEventCenter defaultCenter] removeEventListener:self name:@"abc"];

//移除所有事件监听
[[APIEventCenter defaultCenter] removeEventListener:self];
```

第六章 其他

6.1 一些开放源码

即将开放

第七章 高级功能

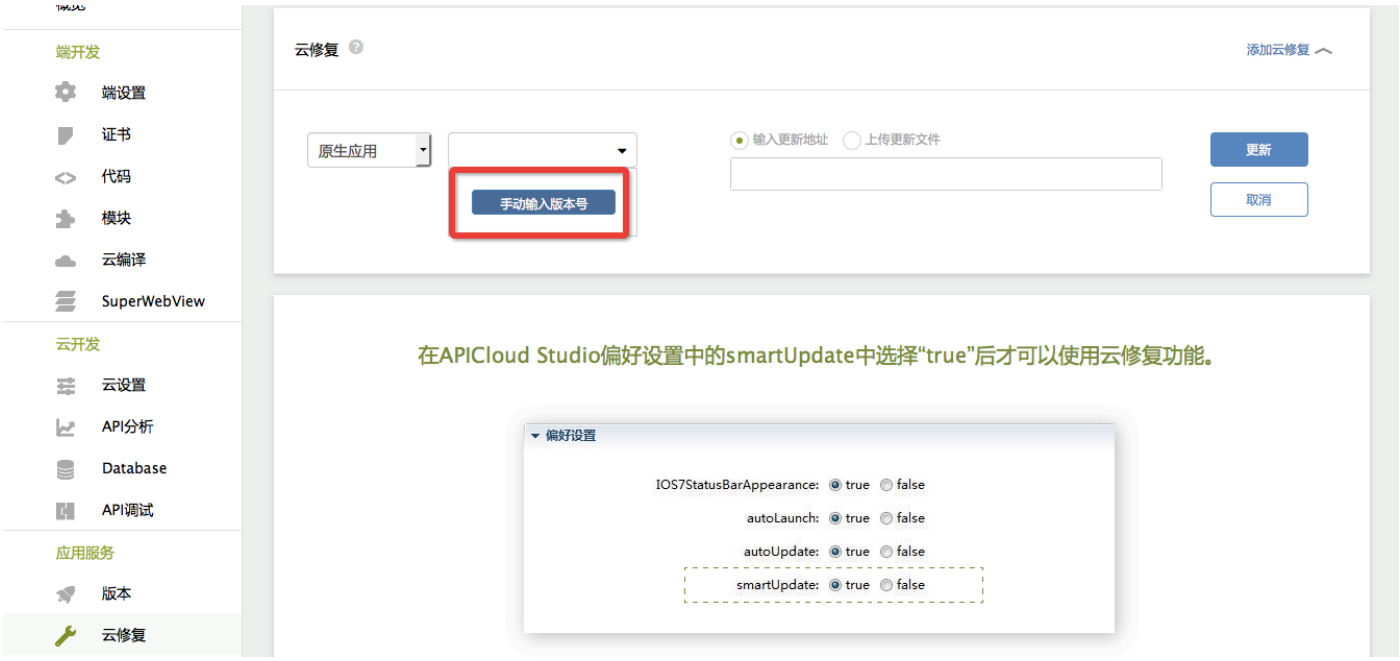
7.1 增量更新（云修复）

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

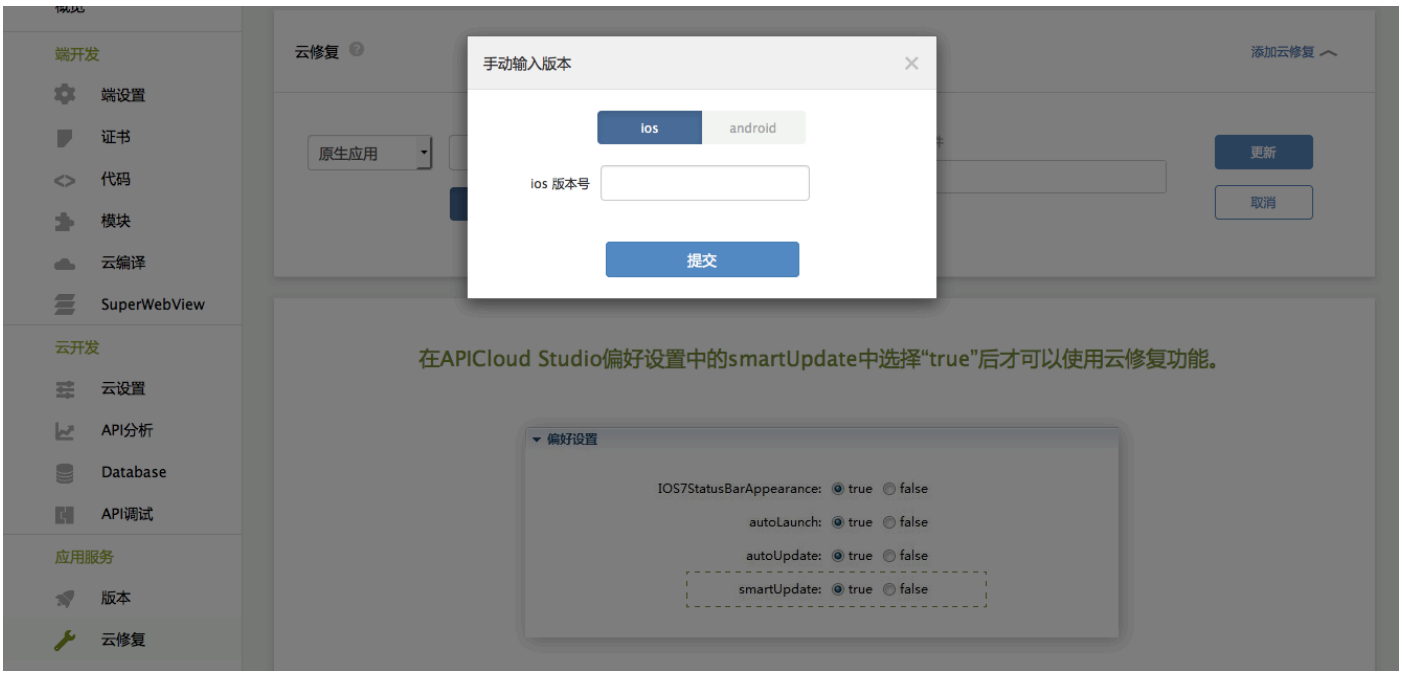
具体的使用流程：

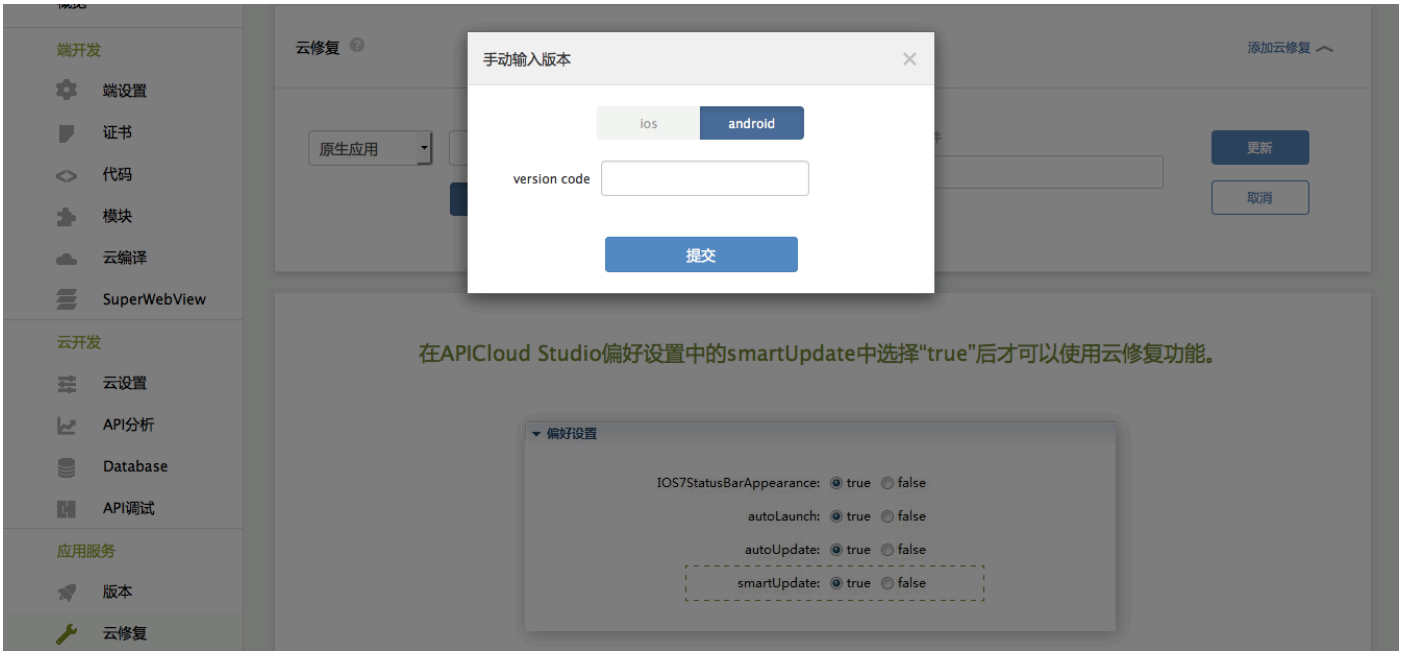
1)、在左侧控制台选择“云修复”，进入云修复页面，在下拉列表中选择“原生应用”，如截图：

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：

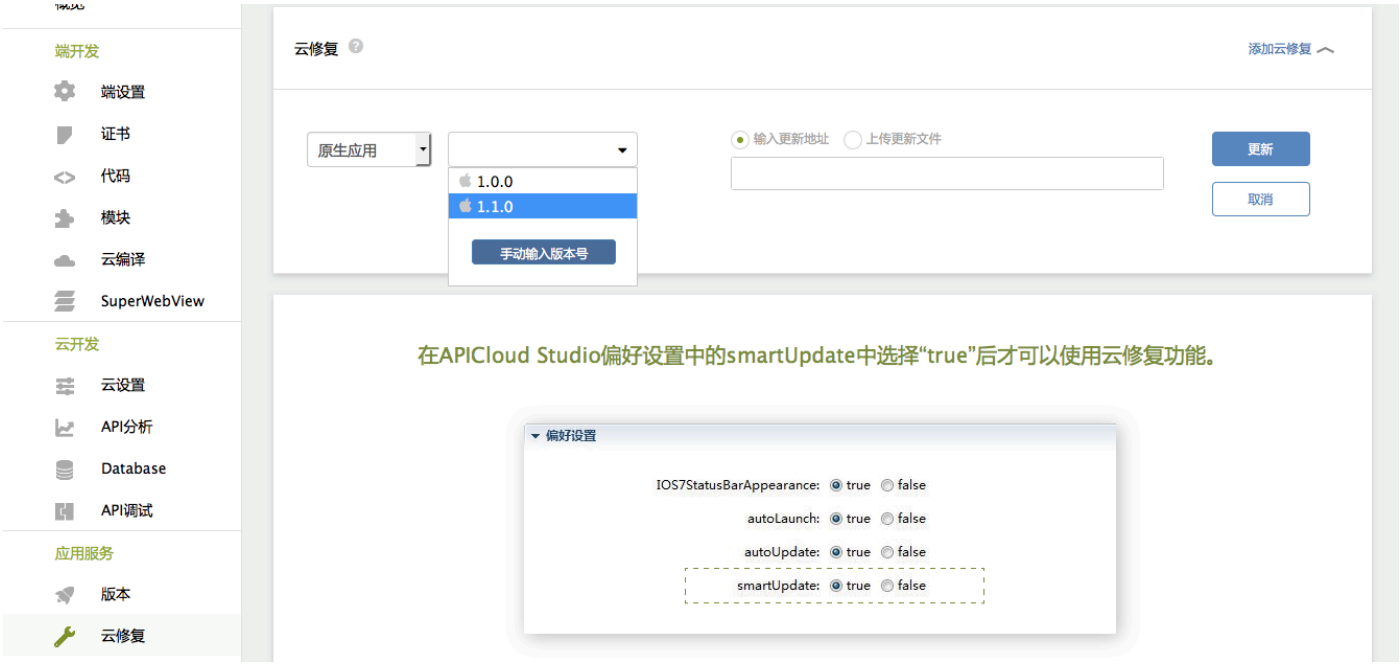


3）、点击“手动输入版本号”，如下图所示.
其中，IOS 要求输入您应用的版本号（即 Info.plist 文件中的 CFBundleShortVersionString 字段值）；
Android 要求输入您应用的 versionCode（即 AndroidManifest.xml 文件中的“android:versionCode”字段值）；





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：

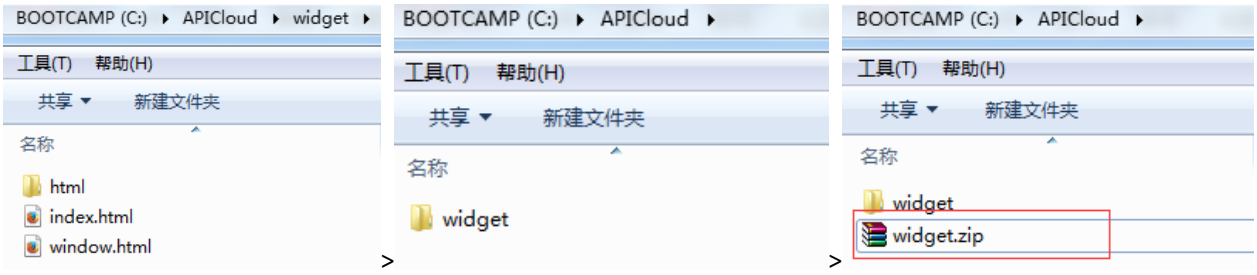


选择‘1.1.0’作为修复版本，如截图：

5）、APICloud 现支持两种云修复方式：提示修复、静默修复。

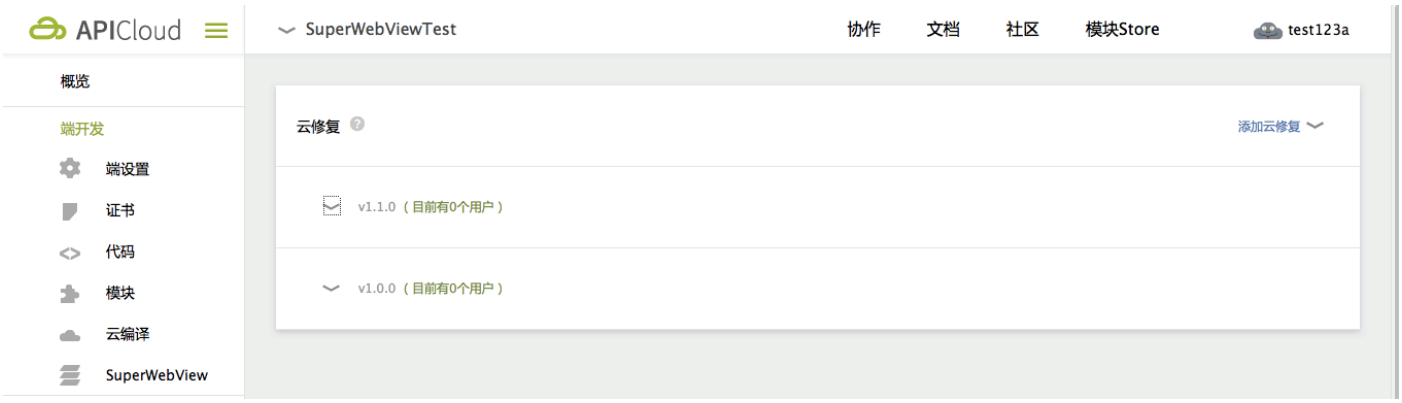
更新包上传方式分为：输入修复内容的 http 更新地址或直接上传更新压缩包；

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 APICloudStudio 中项目目录一致，并只保留更新的文件，然后将 widget 目录压缩成.zip 包即可：



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：



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/