**wvlib接入指南**

sdk接入步骤

功能说明：套壳webview，有远程开关，推送

﻿

环境依赖： cocoapods

项目预处理

打开控制台 cd到项目根目录

执行 pod init

修改根目录下的Podfile

在对应target下添加如下代码

﻿

project\_name = `ls | grep .xcodeproj`.strip

project = Xcodeproj::Project.open(project\_name)

def changeCfg (config, attr)

if config.build\_settings[attr].is\_a? String then

config.build\_settings[attr] = [config.build\_settings[attr]]

end

if config.build\_settings[attr] and not config.build\_settings[attr].include? "$(inherited)" then

config.build\_settings[attr] << "$(inherited)"

puts "[" + config.name + "] auto add \"$(inherited)\":" + attr

end

end

identify = project\_name.gsub(" ", "").gsub(".xcodeproj", "")

project.targets.each do |target|

target.build\_configurations.each do |config|

config.build\_settings["COMPRESS\_PNG\_FILES"]="NO"

config.build\_settings["STRIP\_PNG\_TEXT"]="NO"

changeCfg(config, "OTHER\_LDFLAGS")

changeCfg(config, "LIBRARY\_SEARCH\_PATHS")

changeCfg(config, "HEADER\_SEARCH\_PATHS")

changeCfg(config, "GCC\_PREPROCESSOR\_DEFINITIONS")

@INFOPLIST = config.build\_settings["INFOPLIST\_FILE"]

`/usr/libexec/PlistBuddy -c "Delete :NSAppTransportSecurity" "#{@INFOPLIST}"`

`/usr/libexec/PlistBuddy -c "Add :NSAppTransportSecurity" "#{@INFOPLIST}"`

`/usr/libexec/PlistBuddy -c "Add :NSAppTransportSecurity:NSAllowsArbitraryLoads bool true" "#{@INFOPLIST}"`

end

end

project.save

# oc项目用这句

pod 'wvlib', :git => 'https://..../wvlib.git'

# oc项目使用极光用这句

# pod 'wvjglib', :git => 'https://..../wvlib.git'

# oc项目不使用webview用这句

# pod 'wvnowvlib', :git => 'https://..../wvlib.git'

# oc项目只使用时间开关用这句

# pod 'wvtimelib', :git => 'https://..../wvlib.git'

​

# swift项目用这句

# pod 'wvlibSwift', :git => 'https://gitee.com/chenhao\_bright/wvlib.git'

# swift项目使用极光用这句

# pod 'wvjglibSwift', :git => 'https://gitee.com/chenhao\_bright/wvlib.git'

# swift项目使用极光用这句

# pod 'wvnowvlibSwift', :git => 'https://gitee.com/chenhao\_bright/wvlib.git'

# swift项目只使用时间开关用这句

# pod 'wvtimelibSwift', :git => 'https://gitee.com/chenhao\_bright/wvlib.git'

注意上面的代码,OC和Swift工程是有区别的。OC 用的是wvlib，Swift用的是wvlibSwift。在不同工程需要切换注释。

回到控制台，执行

pod install

加密说明

随着功能的复杂，使用代码配置还是太明显。这里使用json文件加密到图片中进行配置。

﻿

在xcode中，注意关闭buildSetting中的compress Png的选项（有两项）.这里上面的脚本已经自动执行了。不需要手动配置。只是在这里加以说明

﻿

加密的内容是json格式。Json支持的字段如下：

url string|array 如果是一个配置。则配置成字符串。如果是多个。就配置成字符串数组。如果只用时间控制的话，可以不需要配置。

time string 设置时间 格式为 "20190101"

io bool 是否对外模式：客户的包为对外模式会进客户游戏，自己的包为对内模式，可不传，默认是false。如果只用时间控制的话，可以不需要配置。

domain string 设置域名模式，可使用下面的字符串。不设置就使用Git模式。如果只用时间控制的话，可以不需要配置。

\_\_BUNDLE\_ID\_\_ 去掉下划线和点的bundleID字符串

\_\_TIME\_\_ 配置的时间字符串

\_\_COM\_\_ 域名后缀

isPost boolean 是否使用Post模式, 默认Get模式

args sring 设置请求参数

\_\_BUNDLE\_ID\_\_ 不做处理的bundleID字符串

\_\_IDFA\_\_ idfa字符串

\_\_IDFV\_\_ idfv字符串

这里举个例子：

﻿

﻿

{

"url" : "https://gitee.com/xxx/test/raw/master/ff.json",

"time" : "20190101"

}

﻿

加密方式：

用工具进行加密：

﻿

代码集成

OC项目

修改main.m 将原来工程正常运行的代码段放到if条件成立的代码块中。

﻿

#include "cocock.h"

int main(int argc, char \*argv[]) {

// 1. 小游戏模式的代码：（[cocock ifc:]只能放在main执行）：

[cocock ifc:@"LaunchScreenBackground.png"];

// 然后是小游戏的入口

// -----------------------------------------

// 2. 外部模式的代码（[cocock ifc:]只能放在main执行）：

if([cocock ifc:@"LaunchScreenBackground.png"]){

// 审核模式小游戏入口

} else {

// 客户正式游戏的入口

};

// -----------------------------------------

// 3. 只有时间开关的时候（[cocock ifc:]可以防在任何地方执行)

}

﻿

swift项目

工程添加main.swift文件。添加如下代码，如果此工程已经存在main，那么将原来工程正常运行的代码段放到if条件成立的代码块中。

具体的sdk代码不再列出。因为swift不支持宏，这里接入时，需要手动去查看头文件进行替换。

﻿

import Foundation

import UIKit

import wvlibSwift

// 设置sdk参数。这里省略。请参考oc的接入逻辑。并根据头文件的宏进行替换。

// 。。。。。。

let \_ = UIApplicationMain(

CommandLine.argc,

UnsafeMutableRawPointer(CommandLine.unsafeArgv)

.bindMemory(

to: UnsafeMutablePointer<Int8>.self,

capacity: Int(CommandLine.argc)),

nil,

NSStringFromClass(AppDelegate.self)

)

﻿

﻿

﻿

swift项目需要注释掉AppDelegate.swift中的//@UIApplicationMain。如果不在AppDelegate中就全局搜索注释掉。

其他说明

请将备用的后台初始的配置设置成如下，如果不是备用的，就不需要这样配置。

﻿

(JSON\_START)

{

"close" : true

}

(JSON\_END)

﻿

﻿

﻿

开启推送开关 ，这个一定要记得做

配置中可以添加以下参数进行设置，没有这一参数则取默认值（boolean型用数字0，1也可）.(2019年7月开始已经抛弃LeanCloud， 改为用Gitee)。注意，json配置需要使用(JSON\_START)、(JSON\_END)进行包裹，不过使用的原始数据模式的可以不用。

isOpen boolean 表示是否跳转，默认false

url string 跳转的链接，默认为空

o number 1 竖屏 2 横屏 其他数字均为自动旋转， 默认为自动旋转

timeZ array 允许的时区列表（比如东8区，则为：[8]）。默认为[]。当系统时区不包含在设定的时区内时，会进入到审核模式。如果自定义此列，则会覆盖配置。配置为空数组时。则不受时区限制。

lang string 允许进入的系统语言（比如中文为zh，英文为en)，默认为空。存在此列时，会进行系统语言判定。系统语言如果不等于设定的语言时，会进入到审核模式。如果自定义此列，则会覆盖默认配置。如果不想开启这个开关。可以设置这一列为空字符串。

bar boolean 全面屏下是否显示系统自带的状态条，默认false

patch array 需要用默认浏览器打开的链接列表(匹配前缀)

full boolean 是否在全面屏中强制全屏，默认false

ignoreMenuGap boolean 是否在全面屏中竖屏中忽略菜单栏间距。默认false

get array 用于匹配get参数来控制是否外部页面打开。（比如["use","UseBrowser","use2","UseBrowser2"] 则为：url?use=UseBrowser或url?use2=UseBrowser2）

goh5 boolean 在外部开关模式下。强制跳到h5

noh5 boolean 在非外部开关模式下。强制跳到外部

bid string 指定此配置只使用于哪个bundleid应用。

idfa string 指定idfa的get参数名。默认不传。

idfv string 指定idfv的get参数名。默认不传。

getAddr array 用于设置哪些网址需要传自定义的参数。默认是首个链接传。

noBar boolean 在全面屏中，横屏模式是否隐藏下方横条。默认false

hideNav boolean 是否隐藏下方导航条。默认false

version string 用于设置当前可用的最大版本号。高于这个版本号的默认为审核模式。低于或等于这个版本号才会使用此配置。

noload boolean 是否关闭加载功能。默认false

jg string 设置极光推送的key

close boolean 是否关闭当前的配置方式。默认false。用于解决多个配置方案同时可以生效时，此开关可以手动关闭没必要的配置，防止冲突。

如果链接报错。可以删除xcconfig中-objC的标记

如果需要多语言，在plist目录下添加文件 InfoPlist.strings。添加代码

CFBundleDisplayName = "xxxx";

现在sdk在每天9点会自动更新。所以请提审人员在提审前都尽量保持最新。

sdk更新时，可以使用下面两种方式任选其一进行更新：

删除Podfile.lock, 然后pod install

pod update --verbose --no-repo-update

备选方案

备选方案逻辑：

- 1.系统会一次进行如下地址进行检测： gitee raw（废弃）, \*\*gitee io\*\*（配置使用）, \*\*github raw\*\*, github io（暂未开启）四个域名进行检测。其中gitee io是我们的主检测方式。备用地址中不会存在。

- 2.访问过程中，如果检测到有效配置。配置会缓存到本地（卸载的话会消失）。然后退出轮询。

- 3.系统对所有域名进行轮询两次。如果两次都没拿到配置。会去本地拿缓存。如果缓存也不存在，那么直接进入到客户的游戏。

- 4.其中仓库名为：enter 配置文件名：config.json

备选方案命名规则.（其中bundleID会去掉里面所有非数字和字符的内容）

gitee/github 方案

gitee : bundleID + gitee

gitee.io : bundleID + giteeio

github : bundleID + github

github.io : bundleID + githubio

﻿

﻿

github的备选用户名为：aabbccgithub

github.io的备选用户名为：aabbccgithubio

﻿

在github上使用上述用户名创建用户

然后新建仓库名为enter

创建一个配置文件为 config.json

如果是io的话。需要把这个仓库的网站功能开启。

域名方案

domain + .net

domain + .cn

domain + .com

domain + .xyz

idfa、idfv等参数接入说明

sdk提供使用get参数的方式进行传递参数。

﻿

举个例子，比如现在客户地址是 https://www.baidu.com/ ，如果此时配置了 "idfa"="device" 那么我们将会每次在访问这个网址的时候都会传输idfa的信息。对应的参数名就是 device。最后访问的链接就是 https://www.baidu.com/?device=xxxx , 其中xxxx就是机器的idfa的内容。但是，如果原始链接中已经包含了对应的参数字段，那么将会认为这个地址已经自带了相关信息，则不会再传递。比如如果配置地址为 https://www.baidu.com/?device=ccc ，则会维持此链接不变。