打包

作者：fengsharp

QQ：493894652

Email：csdnfor@126.com

2016/3/30

目 录

1. Cocos2dx打包命令 1

2. IOS打包 1

3. IOS免许可打入真机 1

4. Android打包 4

4.1. 要求 4

4.2. 打包命令 4

4.3. 定制打包命令的生成文件 4

4.4. 打包 5

4.4.1. AndroidManifest文件配置 5

4.4.2. Class文件Cpp编译配置 6

4.4.3. create秘钥文件 8

4.4.4. Gradle配置 9

4.4.5. 执行cocos命令来生成apk文件 10

4.4.6. cocos compile打包定制项目 10

4.5. 需要注意 11

5. 标题1 12

5.1. 标题2 12

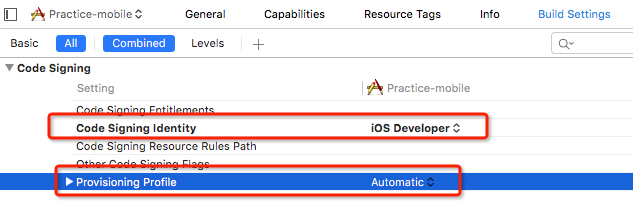
5.1.1. 标题3 12

# Cocos2dx打包命令

Cocos2dx提供了内置的打包命令，常常使用cocos2dx的打包命令来打android包。

# IOS打包

1. 设置好许可文件



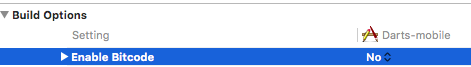
1. 插入ios设备
2. Product->Scheme->Edit Scheme... 确定是release还是debug的包

Macintosh HD:private:var:folders:v5:ttx1ql7d7pz05lthhpc844bw0000gn:T:WizNote:2de4998f-d195-49f1-b676-a0208f6f2a6d:index_files:9176622.png

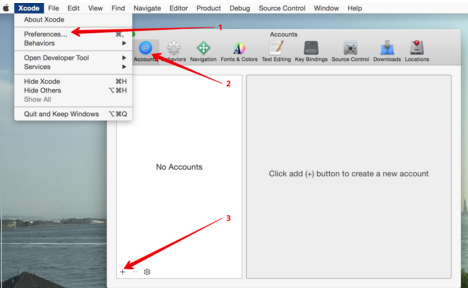
1. 执行 Product->Archive(档案)。等待编译链接结束后，会弹出界面（以后可以在Window->Orgnaizer中找到）。
2. 在对应的列表中，选择一个目标，右单击选择 Show in Finder
3. 右单击文件（\*.xcarchive），选择打开包内容
4. 在ITunes中，右单击目标文件，选择Show in Finder，即可找到对应的ipa文件了.

# IOS免许可打入真机

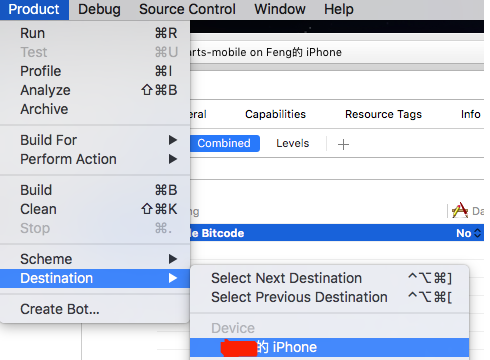
1. Build Setting中

​

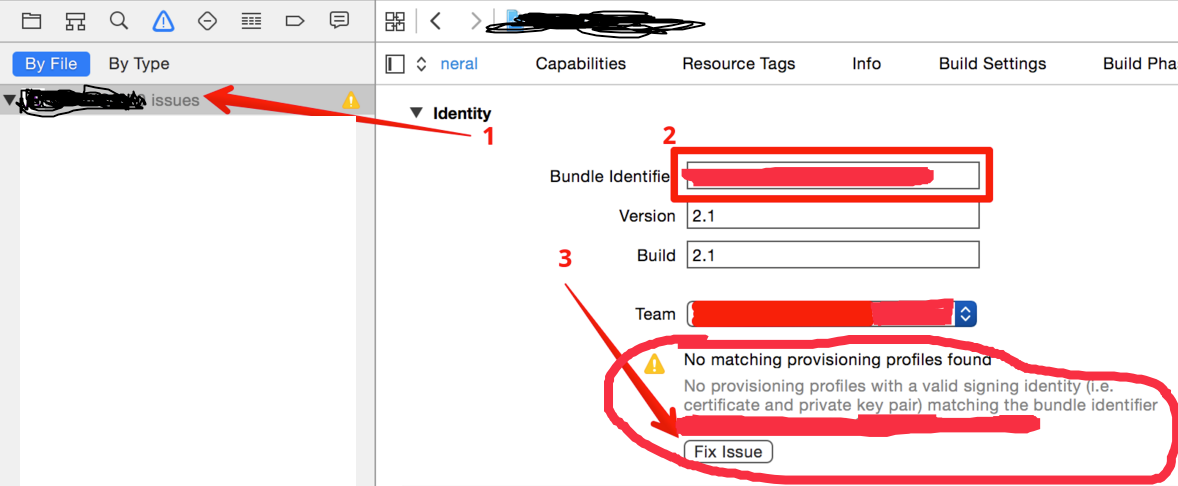
1. 账号设置



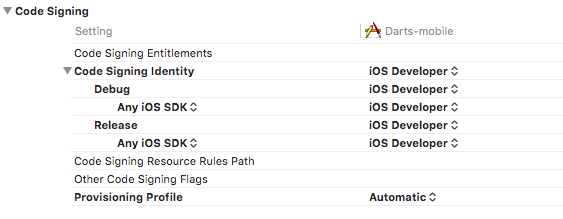
1. 选择自己的设备

​ 

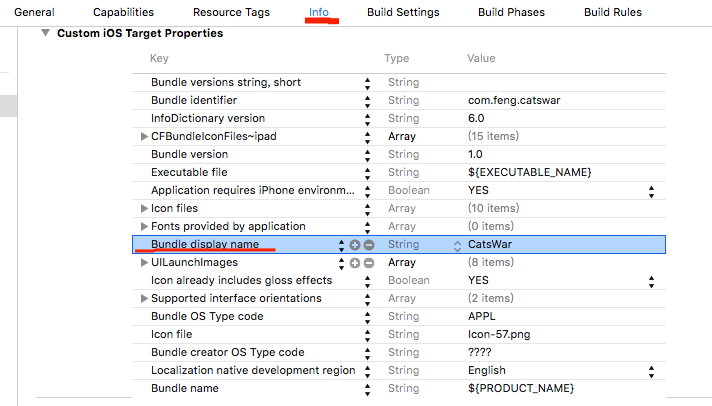
1. ​​工程中进行如下设置

  
该步非常重要，【Fix Issue】后，弹出界面中create一个profile

1. 保险起见，可以Build Settings中进行设置

​ 

6. app显示的名称设置



# Android打包

## 要求

配置了NDK cocos等命令，该配置在源码解压根目录中的【steup.py】，执行后就配置完成了。

## 打包命令

debug: cocos compile -p android --android-studio

release:cocos compile -p android -m release --android-studio

## 定制打包命令的生成文件

将[build\_android.py]（在文档所在目录下可以找到）文件替换【/Applications/Cocos/frameworks/cocos2d-x-3.8.1/tools/cocos2d-console/plugins/project\_compile】中的文件。

打release包的生成文件名称带有时间戳。

## 打包

### AndroidManifest文件配置



比较重要的有：

1.包命的配置

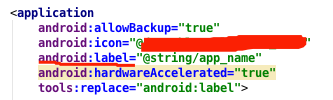
2.横屏竖屏的配置 landscape portrait

3.app权限的配置（比如网络，电话的访问权限）

4.图片资源的配置

5.其他一些配置

**android:label="@string/app\_name"  
android:hardwareAccelerated="true"  
tools:replace="android:label"**>



这个application中的label标记非常重要，在加载和卸载的时候会显示.

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="<http://schemas.android.com/apk/res/android>"

    xmlns:tools="<http://schemas.android.com/tools>"

    package="com.feng.catswar"

    android:installLocation="auto">

    <uses-feature android:glEsVersion="0x00020000" />

    <application

        android:allowBackup="true"

        android:label="@string/app\_name"

        android:hardwareAccelerated="true"

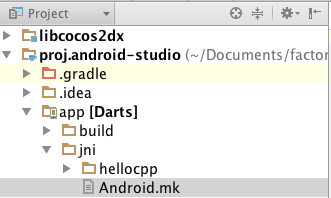
        tools:replace="android:label"

        android:icon="@mipmap/ic\_launcher">

另外记着修改资源字符串app\_name的值

### Class文件Cpp编译配置

1.找到Android.mk文件



修改内容为：

LOCAL\_PATH := $(call my-dir)

include $(CLEAR\_VARS)

$(call import-add-path,$(LOCAL\_PATH)/../../../cocos2d)

$(call import-add-path,$(LOCAL\_PATH)/../../../cocos2d/external)

$(call import-add-path,$(LOCAL\_PATH)/../../../cocos2d/cocos)

LOCAL\_MODULE := cocos2dcpp\_shared

LOCAL\_MODULE\_FILENAME := libcocos2dcpp

MY\_SRC\_FILES := $(wildcard $(LOCAL\_PATH)/../../../Classes/\*.cpp)

MY\_SRC\_FILES := $(MY\_SRC\_FILES:$(LOCAL\_PATH)/%=%)

LOCAL\_SRC\_FILES := hellocpp/main.cpp

LOCAL\_SRC\_FILES += $(MY\_SRC\_FILES)

LOCAL\_C\_INCLUDES := $(LOCAL\_PATH)/../../../Classes

# \_COCOS\_HEADER\_ANDROID\_BEGIN

# \_COCOS\_HEADER\_ANDROID\_END

LOCAL\_STATIC\_LIBRARIES := cocos2dx\_static

# \_COCOS\_LIB\_ANDROID\_BEGIN

# \_COCOS\_LIB\_ANDROID\_END

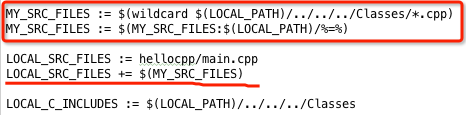
include $(BUILD\_SHARED\_LIBRARY)

$(call import-module,.)

# \_COCOS\_LIB\_IMPORT\_ANDROID\_BEGIN

# \_COCOS\_LIB\_IMPORT\_ANDROID\_END

特别注意：



2.Application.mk文件修改为

APP\_STL := gnustl\_static

APP\_CPPFLAGS := -frtti -DCC\_ENABLE\_CHIPMUNK\_INTEGRATION=1 -std=c++11 -fsigned-char

APP\_LDFLAGS := -latomic

APP\_CPPFLAGS += -fexceptions

APP\_ABI := armeabi

NDK\_TOOLCHAIN\_VERSION = 4.8

APP\_PLATFORM := android-9

ifeq ($(NDK\_DEBUG),1)

APP\_CPPFLAGS += -DCOCOS2D\_DEBUG=1

APP\_OPTIM := debug

else

APP\_CPPFLAGS += -DNDEBUG

APP\_OPTIM := release

endif

其中

APP\_CPPFLAGS += -fexceptions 是打开编译器的一场捕获

APP\_ABI := armeabi 指定程序运行体系架构

NDK\_TOOLCHAIN\_VERSION = 4.8 因为使用了C++11的特性，所以指定GCC4.8来编译

APP\_PLATFORM := android-9 设置SDK的API版本

android 2.2 - api version 8

android 2.3 - api version 9

android 4.2.2 - api version 17

### create秘钥文件

输入命令(validity为有效期）

keytool -genkey -alias darts -keyalg RSA -validity 20000 -keystore darts.keystore

输入密钥库口令

123456

您的名字与姓氏是什么?

[fengsharp]: fengsharp

您的组织单位名称是什么?

[fengsharp]: fengsharp

您的组织名称是什么?

[fengsharp]: fengsharp

您所在的城市或区域名称是什么?

[beijign]: beijing

您所在的省/市/自治区名称是什么?

[beijign]: beijing

该单位的双字母国家/地区代码是什么?

[86]: 86

CN=fengsharp, OU=fengsharp, O=fengsharp, L=beijing, ST=beijing, C=86是否正确?

[否]: y

输入 <darts.keystore> 的密钥口令

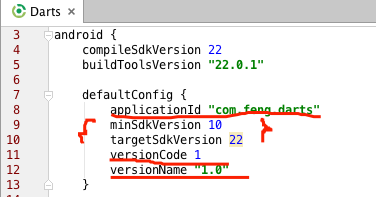
123456

输入 open . 打开当前的目录，找到【darts.keystore】文件。将该文件放到【proj.android-studio】目录下

### Gradle配置



比较常用配置



然后配置秘钥

signingConfigs {  
 release {  
 keyAlias **'catswar'**keyPassword **'123456'**storeFile file(**'../catswar.keystore'**)  
 storePassword **'123456'**}  
 debug {  
 keyAlias **'catswar'**keyPassword **'123456'**storeFile file(**'../catswar.keystore'**)  
 storePassword **'123456'**}  
}

注意这里的配置内容一定要和create时候的一致，否则无法生成apk，比如这里的keyAlias 设置为darts，而不是创建时候的darts.keystore。则编译链接没有问题，但是在最终生成apk文件的时候会报错。

### 执行cocos命令来生成apk文件

debug 命令

cocos compile -p android --android-studio

Android打包：

debug: cocos compile -p android --android-studio

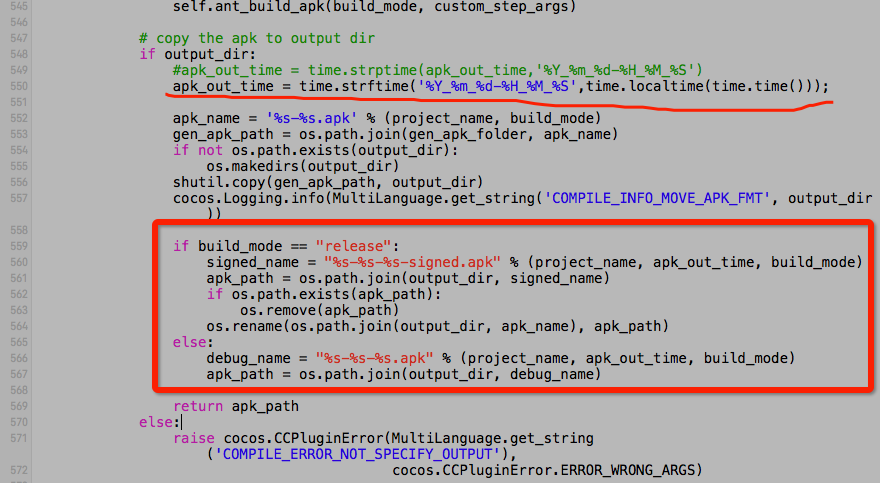
release:cocos compile -p android -m release --android-studio

### cocos compile打包定制项目

在打release包的时候希望apk文件的文件名称按照日期时间的形式，则需要修改cocos的打包命令

找到文件【/Applications/Cocos/frameworks/cocos2d-x-3.8.1/tools/cocos2d-console/plugins/project\_compile/build\_android.py】

主要修改：



apk\_out\_time = time.strftime('%Y\_%m\_%d-%H\_%M\_%S',time.localtime(time.time()));

if build\_mode == "release":

    signed\_name = "%s-%s-%s-signed.apk" % (project\_name, apk\_out\_time, build\_mode)

    apk\_path = os.path.join(output\_dir, signed\_name)

    if os.path.exists(apk\_path):

        os.remove(apk\_path)

    os.rename(os.path.join(output\_dir, apk\_name), apk\_path)

else:

    debug\_name = "%s-%s-%s.apk" % (project\_name, apk\_out\_time, build\_mode)

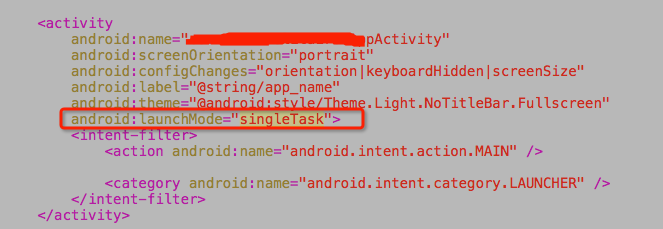
    apk\_path = os.path.join(output\_dir, debug\_name)

## 需要注意

在最新的Google手机上发现：

游戏正在运行，已经存在一个游戏实例，又生成一个新实例时候，会触发事件EVENT\_RENDERER\_RECREATED。比如游戏已经在后台运行，再从Google Play Store中打开游戏时候。这是因为安卓的默认启动模式是允许这样的。这样带来的问题是，当两个游戏实例存在时，用户通过安卓返回键退出时候，需要退出两次才真正退出游戏。

我们修改游戏的启动模式为singleTask则避开了这个问题。



# 标题1

## 标题2

### 标题3

#### 标题4