OS 4.2 编写真机和模拟器通用的 framework（静态库）

分类：Iphone2012-12-13 17:31 5323人阅读 评论(3) 收藏

OS 中静态库的方式有*.a 或*.framework。个人感觉不论是.a 还是.framework 其实无外于对源码进行隐藏的一种表现方式。年多时在项目中，往往对于积累沉垫下来很少改动的代码（高内聚，低耦合的代码），为了增加重用性，常常都进行库的封存。

至 WIN 上封装为 DLL（动），LIB（静）。在 LINUX 上封装为.SO（动），.a（静）。

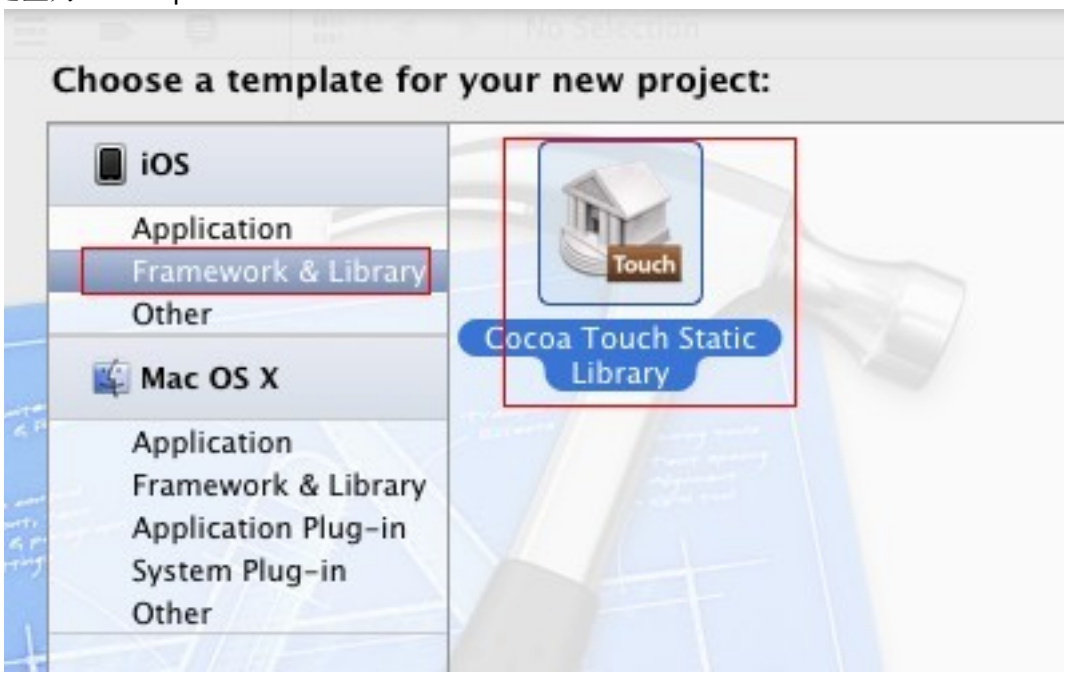
更于介绍对类库的 framework 编写，这里将 http://blog.csdn.net/fengsh998/article/details/8278978 中的 ocsqlite.h 和 ocsqlit的源码制作成静态库(framework)

废话少说，开始编写 IOS 的 framework 类库。

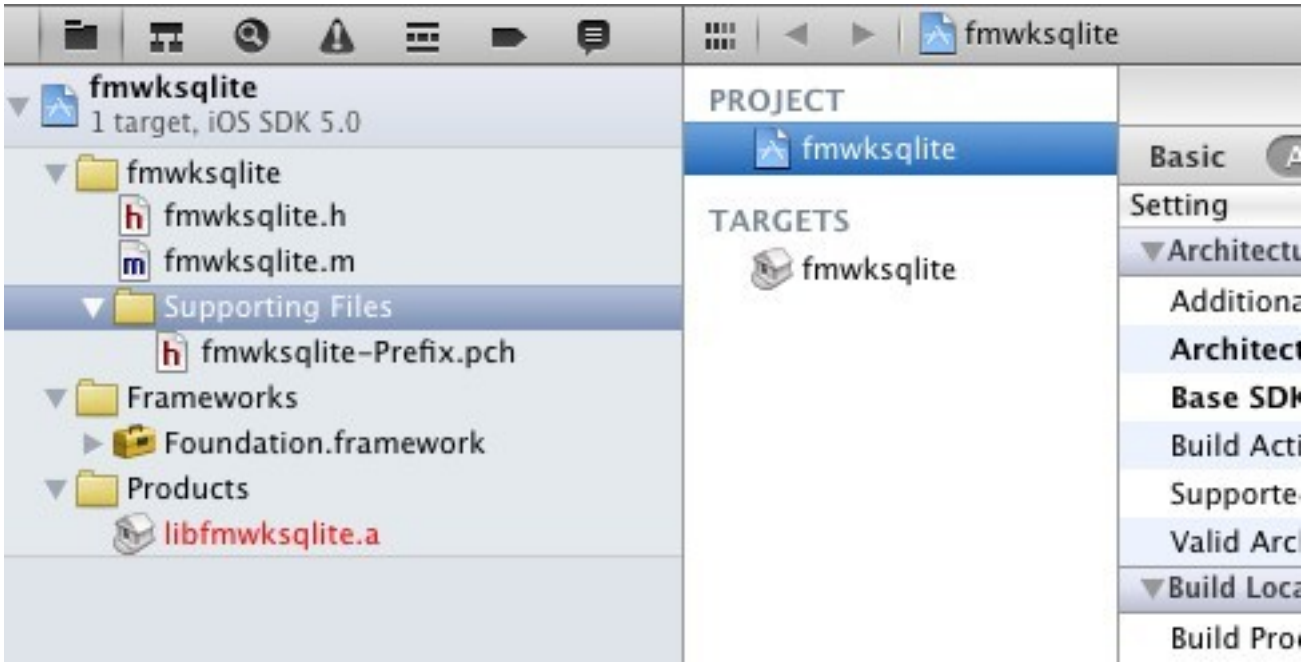
平台：VM+MAC OS10.6+XCode4.2

、新建一个静态库工程。file--new--project，弹出框中选择 Ios--framework&library 中的 cocoa touch static library.点击 NEX输入 product name

这里为 fmwksqlite



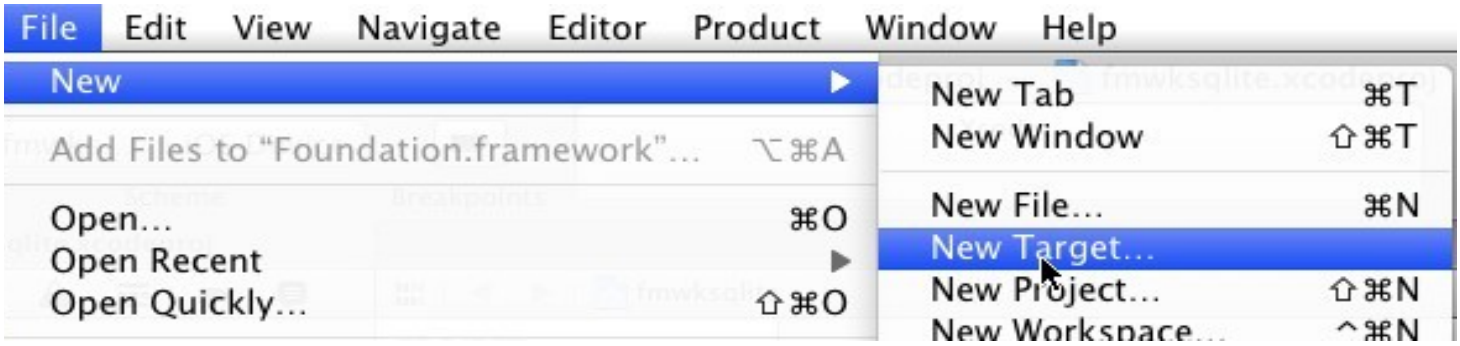
新建好的工程如下图：



图中默认情况下建的是.a 的静态库工程。可以从 products 中看到最终输出的是 libfmwksqlite.a 的静态库文件。

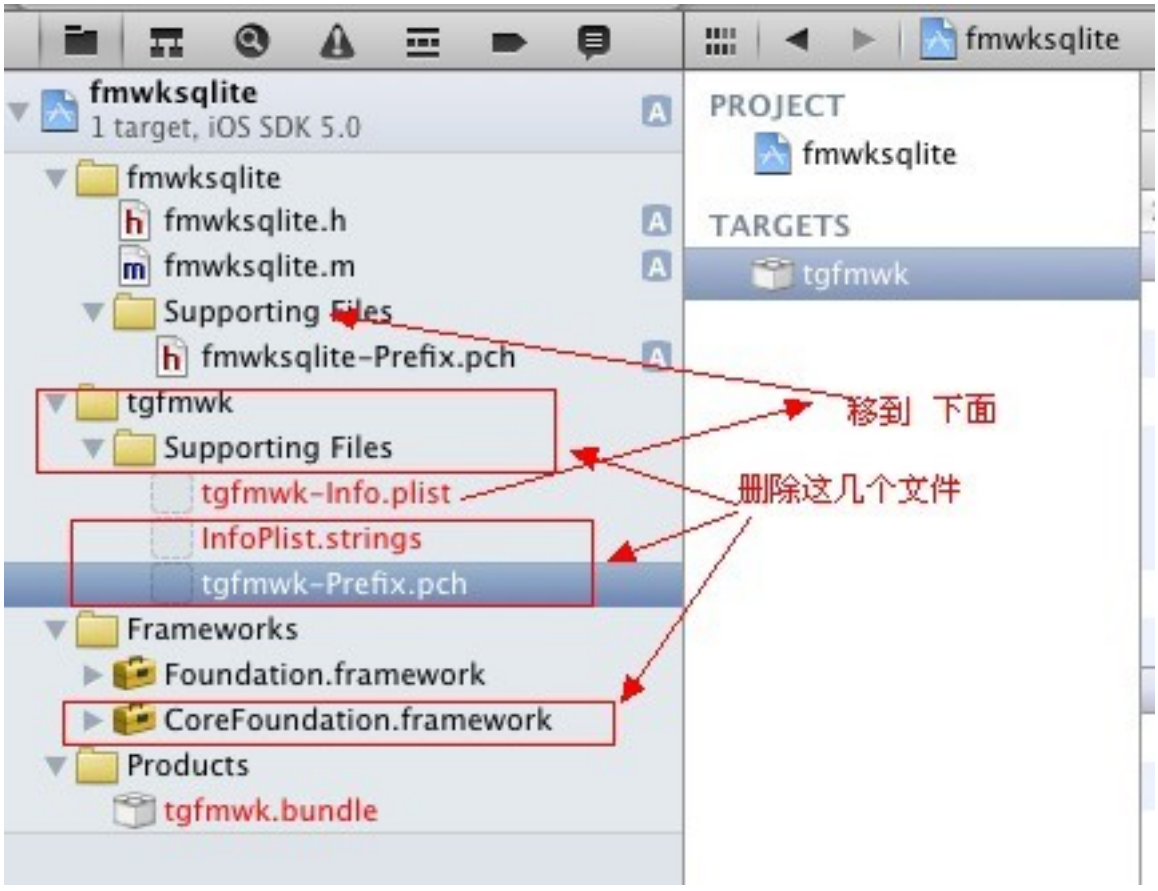
由于 IOS 直接生成提.a 静态库，因此要手动把库修改为 framework。因此需要把当前的 targets 文件删除。如图：

2、把原来的 targets 删除了，需要新建一个新的 targets。file-new-targets

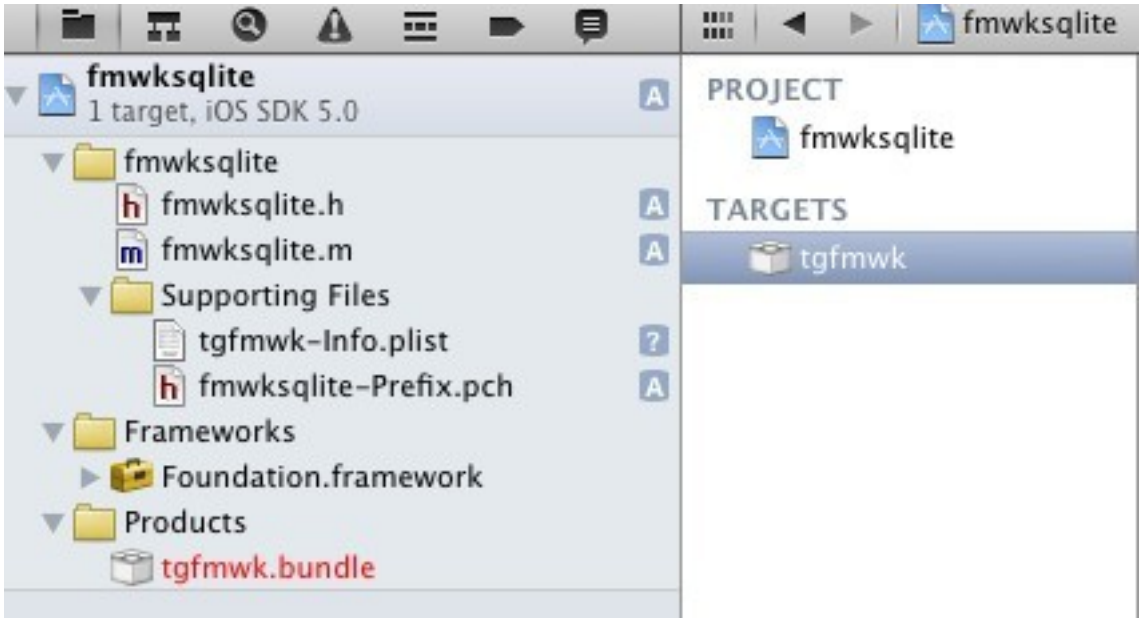


单出的框中选择 mac osx 下的 framework&library 中的 bundle （因为 IOS TOUCH 中没有 Bundle 所在使用 MAC 下的 Bur
长改制），输入 product name，这里可以输入和工程相同的名称（这里可以减少一点点麻烦），个人不太喜欢按步造班，
自己另取一个 targets 名称，这里为 tgfwmk。

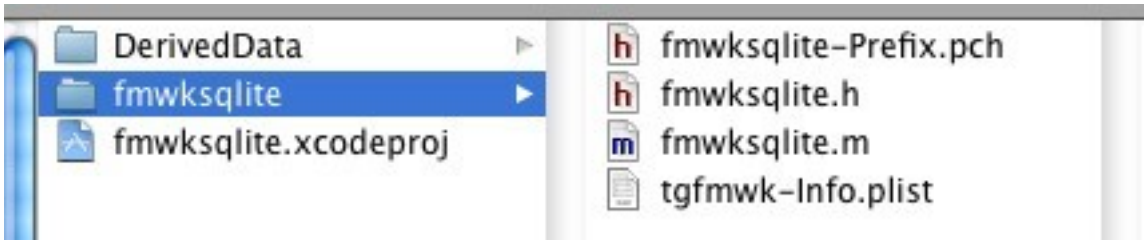
新建完成如图：



把新建的 target 产生的多余文件删除，当前没有用到所以就删除咯。只需要保留 targets 的配置文件即可。删除后如图：



文件夹中的文件：



将新建的 targets 进行修改几个必要项。（关键设置）

详细步骤：

1、选中 tgfmwk 中的 Build settings 。

2、修改 Build settings 页中的 Base SDK 的值为 least ios5.0(注我的 SDK 是 5.0 的，根据按装的 SDK 来定)

3、修改 Build settings 页中的 Architectures 的值改为 Standard(armv7)

4、修改 Build settings 页中的 Build Active Architecture only 值为 NO

5、修改 Build settings 页中的 Link With Standard Libraries 值改为 NO

6、修改 Build settings 页中的 Mach-O Type 为 Relocatable Object File

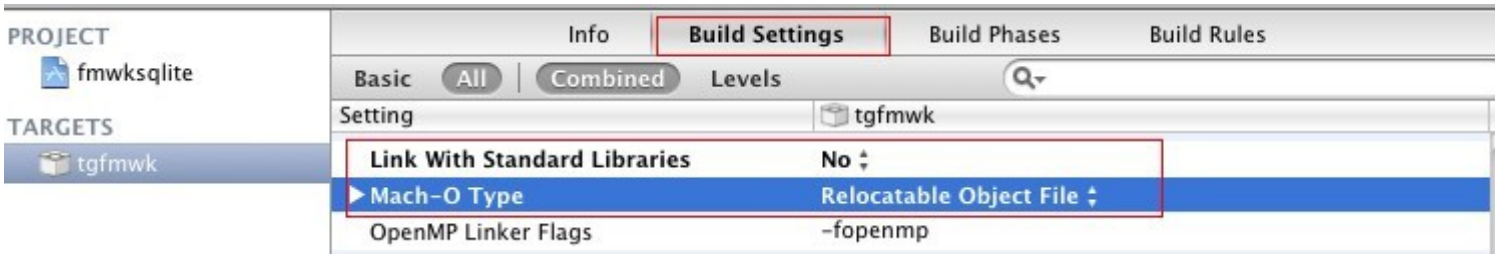
7、修改 Build settings 页中的 Wrapper Extension 为 framework

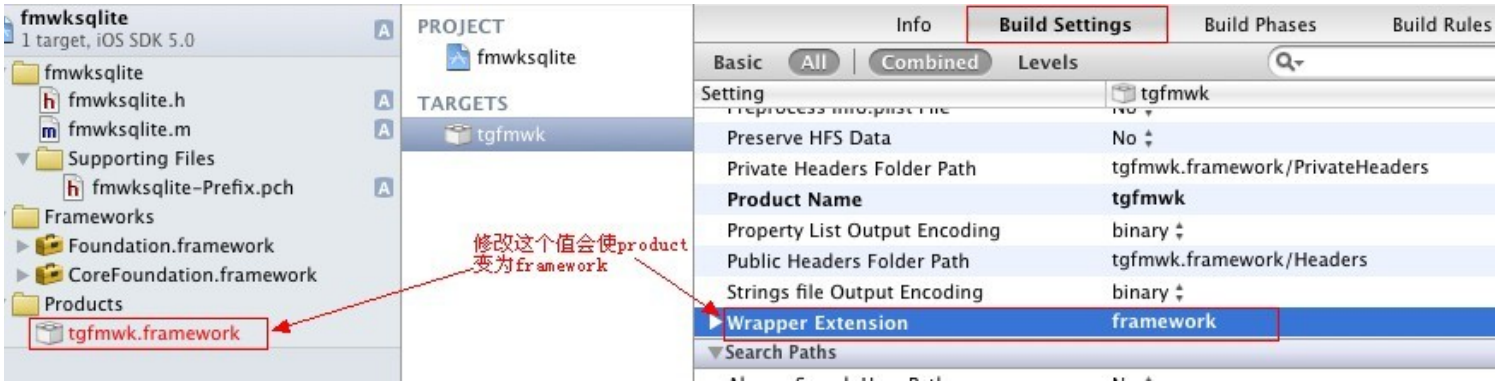
8、修改 Build settings 页中的 Dead Code Stripping 为 NO

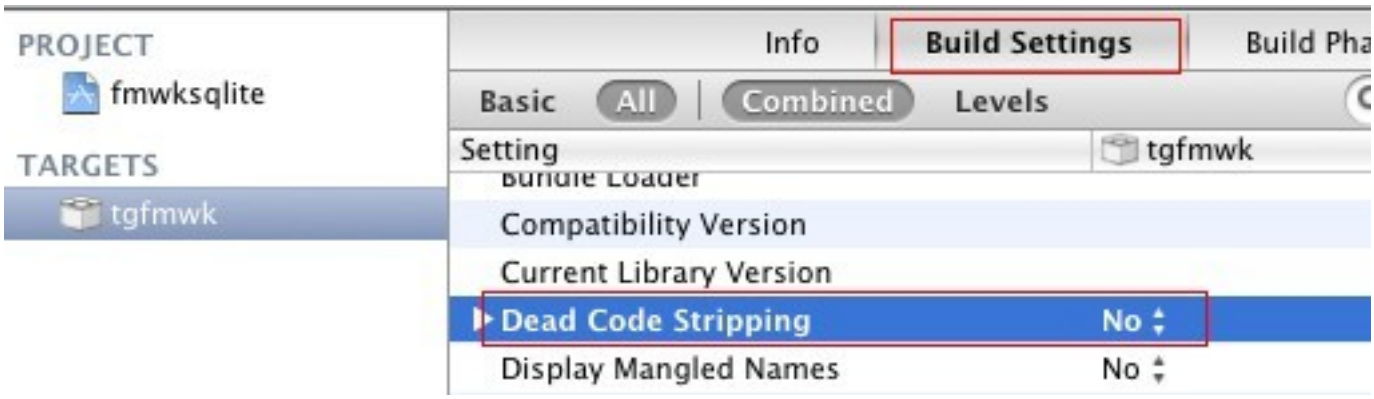
下面的步聚是由于新建的 targets 文与项目名称不对应导致要手动修改的地方，如果是同名就不用理会下面的步骤了)

9、修改 Build settings 页中的 info.plist File 为 fmwksqlite/tgfmwk-info.plist 未改前为 tgfmwk/tgfmwk-info.plist（因为我将这个文件移到了项目目录下）

10、修改 Build settings 页中的 GCC_PREFIX_HEADER 改为 fmwksqlite/fmwksqlite-Prefix.pch 未改前为 tgfmwk/tgfmwk-Prefix.pch





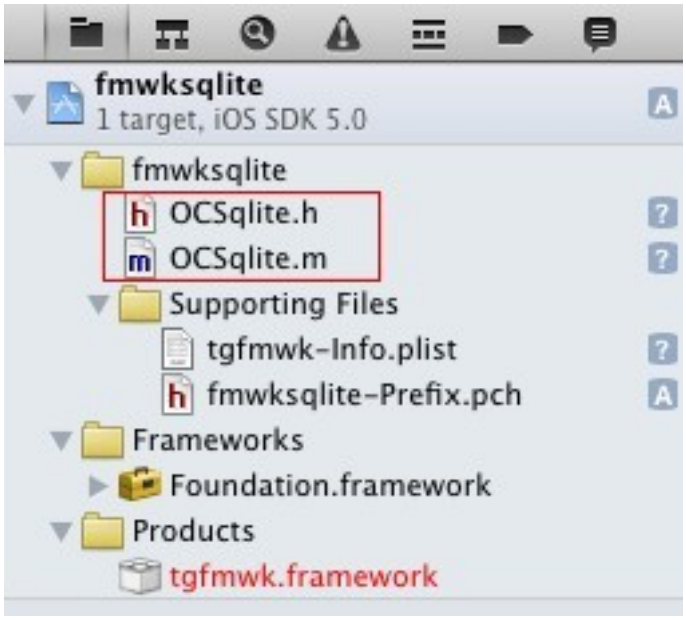


以上修改完成后就可以选模拟器平台运行还是设备运行了。这里先选模拟器(iphonesimulator)点击 RUN。正常情况下应该是
行成功。

但这个时候只是产生了一个空的 tgfmwk.framework 里面并没有库文件和头文件输出。

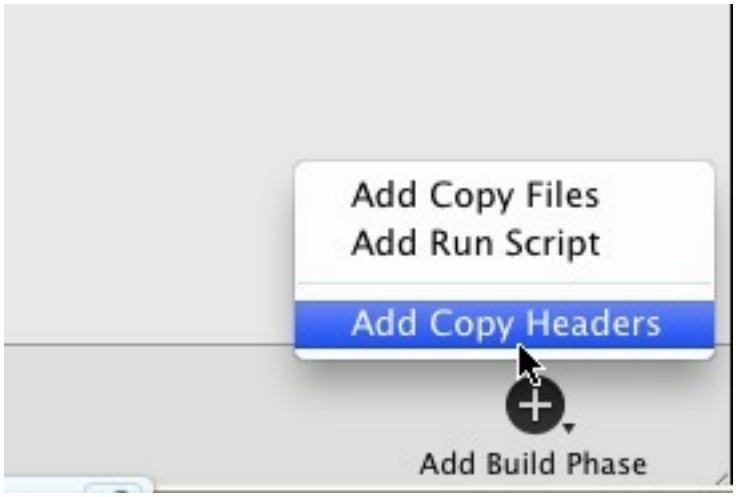
下面是如何添加类文件以编译产生相应的库文件。

把默认的 fmwksqlite.h 和 fmwksqlite.m 文件删除。添加 ocsqlite.h 和 ocsqlite.m 文件进来

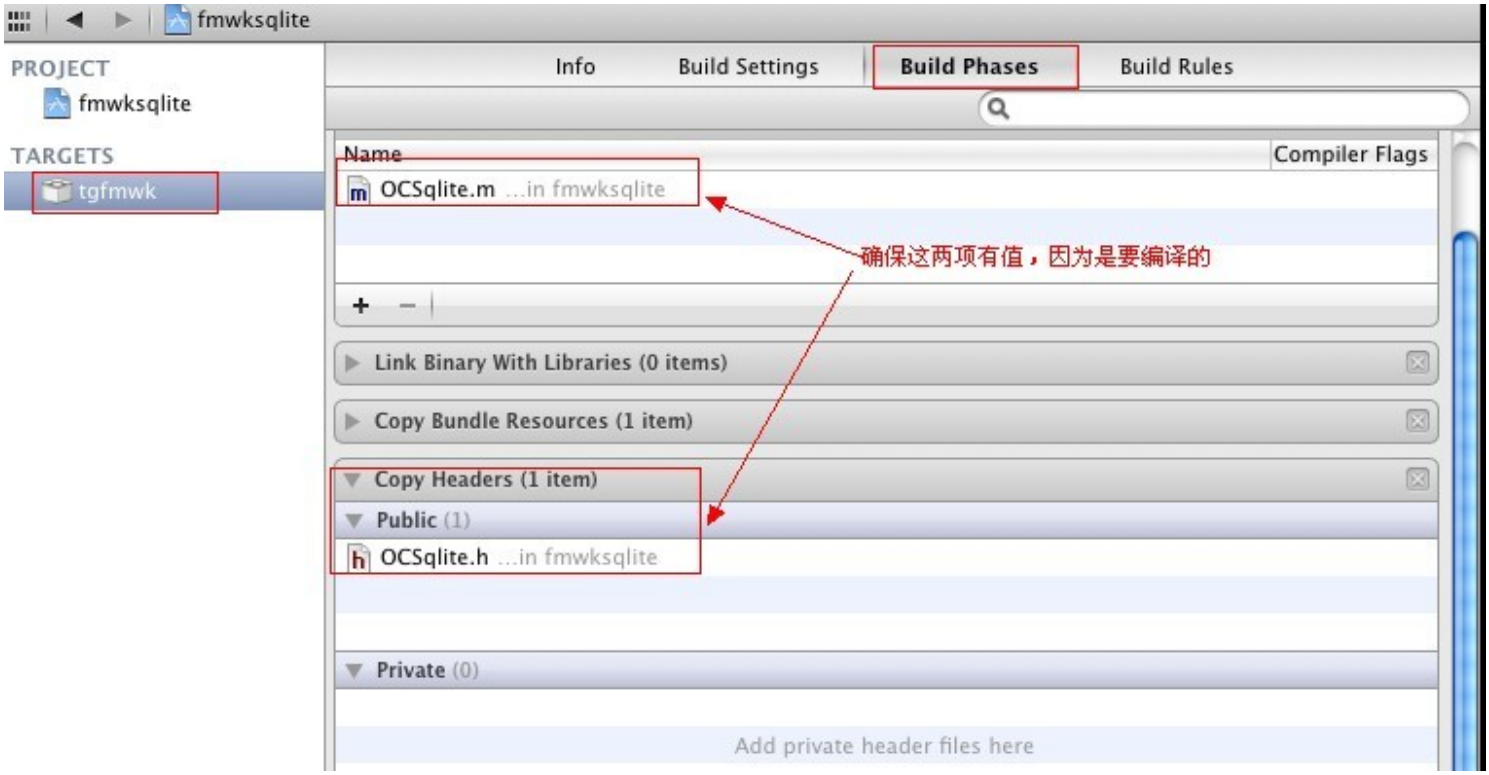


3、添加编译文件和头文件。操作如下：

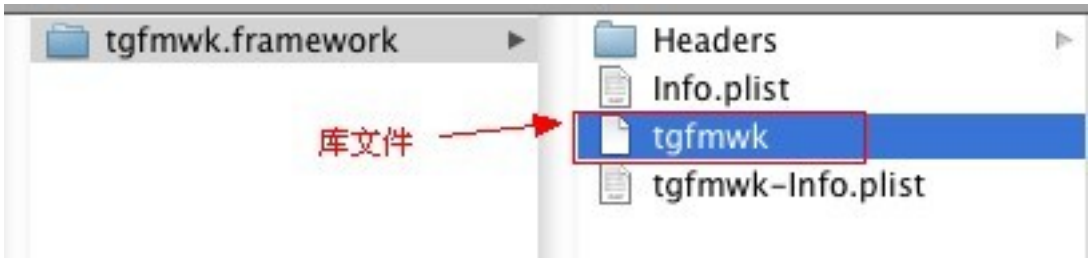
打开 tgfmwk 下的 build parses 点击 add bulids parses 弹出中选择 add copy headers



然后在头文件项中添加 ocsqlite.h 为 public headers



上面 OK 之后就可以编译，成功后，将会在 Build/products/debug-iphonesimulator/tgfmwk.framework 下产生库文件。

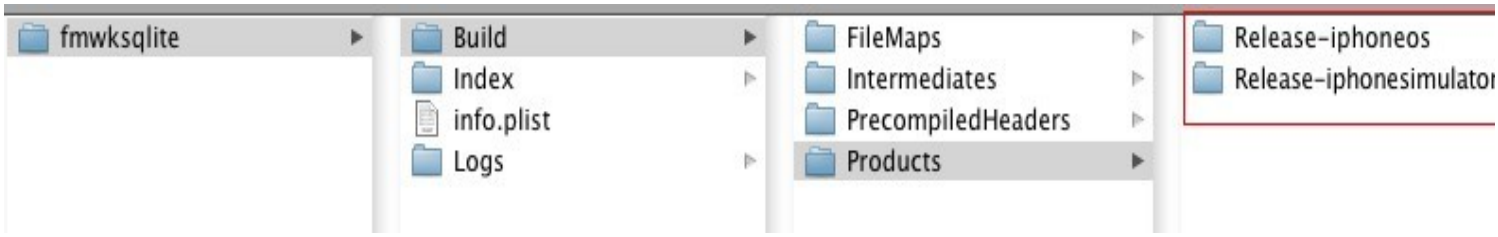


到这里一个模拟器版本的 framework 就制作 OK。使用时只需要将 tgfmwk.framework 整个文件夹 COPY 到相应的项目下就可使用了。正真发布时最好使用 Release 版本。

下面介绍一下如何使用整个 framework 即可模拟器使用也可以真机使用。

演示前先输入 iphoneos(真机)和 iphonesimulator(模拟器)版本的 tgfmwk.framework，这里使用的是 Release 版。(product--scheme)

输出后可以在 DeviceData/fmwksqlite/build/products/



下面是合并的关键，

先来看一下 framework 中的 tgfmwk

打开终端。输入 pwd 先看一下当前所在的目当路径。使用 ls 查看当前目录下的文件夹。

cd /users/[用户名]/Desktop/fmwksqlite/DerivedData/fmwksqlite/build/products/release-iphoneos/tgfmwk.framework

然后使用 lipo -info tgfmwk 查看一下库的信息。

看到输出 Non-fat file: tgfmwk is architecture: armv7

这里表明只支持真机。

```
f-fshmatomac:tgfmwk.framework ffsh$ lipo -info tgfmwk
Non-fat file: tgfmwk is architecture: armv7
f-fshmatomac:tgfmwk.framework ffsh$
```

司样的再来看看模拟器的。

:d /users/[用户名]/Desktop/fmwksqlite/DerivedData/fmwksqlite/build/products/release-iphonesimulator/tgfmwk.framework

然后使用 lipo -info tgfmwk 查看一下库的信息。

昏到输出 Non-fat file: tgfmwk is architecture: i386

```
f-fshmatomac:~ ffsh$ cd desktop/fmwksqlite/derivedata/fmwksqlite/build/products
/release-iphonesimulator/tgfmwk.framework
f-fshmatomac:tgfmwk.framework ffsh$ lipo -info tgfmwk
Non-fat file: tgfmwk is architecture: i386
f-fshmatomac:tgfmwk.framework ffsh$
```

可见要想真机和模拟器都可以使用该 framework 需要将这两个版本的 tgfmwk 文件进行合并。

使用命令 lipo -create xxxx/tgfmwk xxxxx/tgfmwk -output tgfmwknew

其中 xxxx 表示路径，一个是真机的 tgfmwk 所在路径，一个是模拟器 tgfmwk 文件所在路径，然后输出 tgfmwknew 新的文

输出后，只需要将这个 tgfmwknew 重命名为 tgfmwk 然后复盖原来的 tgfmwk 文件即可。把复盖后的这个 tgfmwk.framewo

考出来到具体使用的项目就可以实现真机和模拟器通用的 framework 了。

昏一下 tgfmwknew 的信息。

显示：Architectures in the fat file: tgfmwknew are : armv7 i386

可见这个合并的文件已具备了 armv7（真机）和 i386（虚拟机）的能力。

```
f-fshmatomac:tgfmwk.framework ffsh$ lipo -create tgfmwk /users/ffsh/desktop/fmwk
sqlite/derivedata/fmwksqlite/build/products/release-iphoneos/tgfmwk.framework/t
gfmwk -output tgfmwknew
f-fshmatomac:tgfmwk.framework ffsh$ lipo -info tgfmwknew
Architectures in the fat file: tgfmwknew are: armv7 i386
f-fshmatomac:tgfmwk.framework ffsh$
```

由于 OCSqlite 这个类使用到了 libsqlite3.dylib ，所在在调用 framework 库时也需要添加这个动态库。

本节完，下节将介绍.a 的静态库。.a 的比 framework 相对少了不少工作。呵呵。

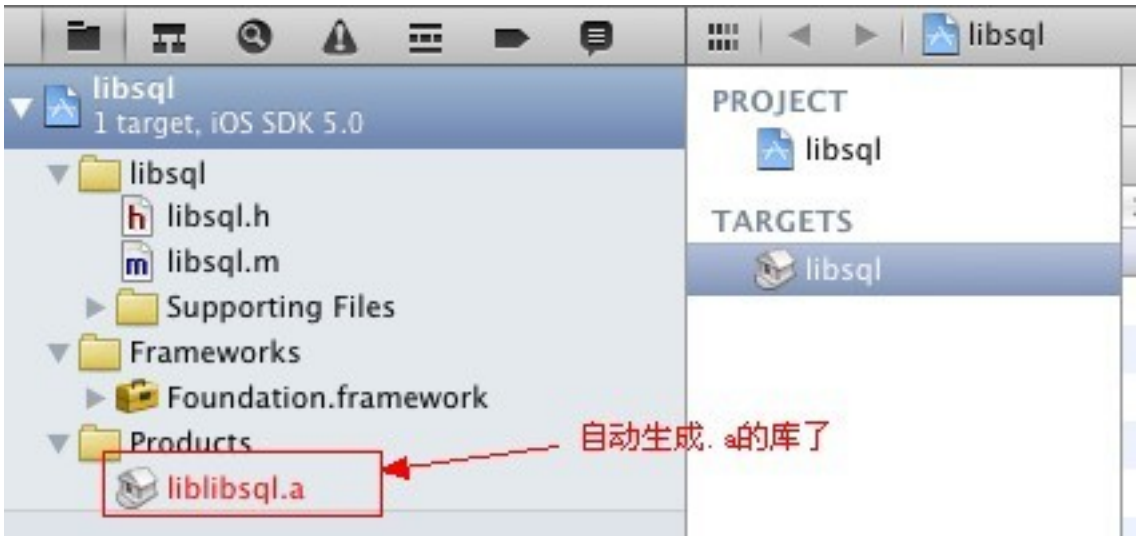
OS 4.2 编写通用的静态库.a 文件

分类：Iphone2012-12-13 18:19 6133人阅读 评论(2) 收藏

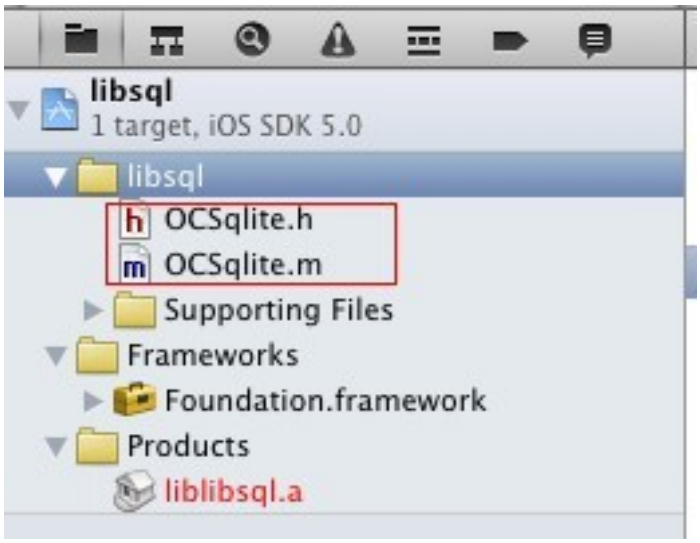
OS 产生.a 的静态库，比起.framework 相对简单了好些。

下面介绍一下具体生成步骤：

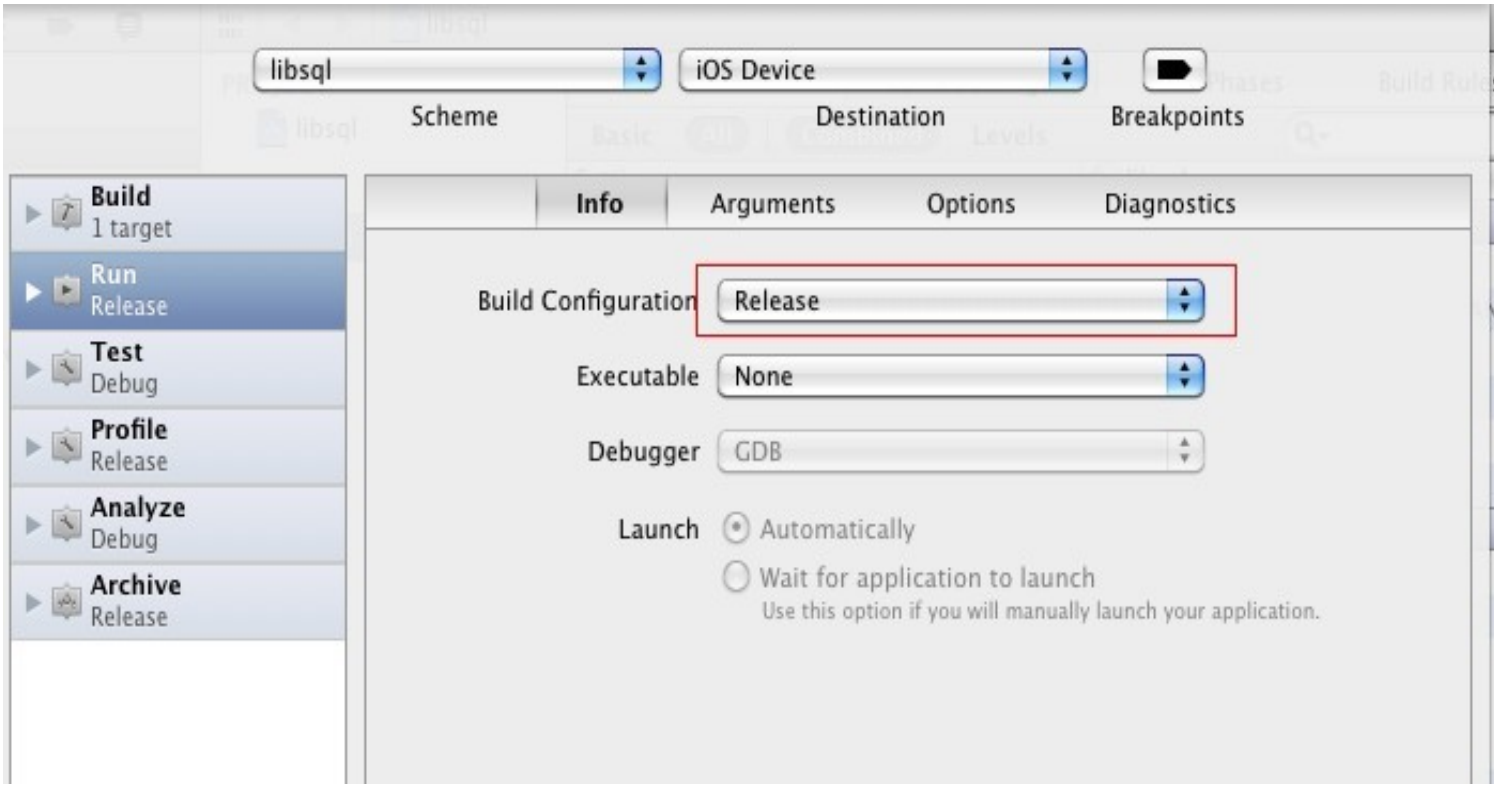
、新建一个 framework&library 库。IOS 下的 cocoa touch static library。然后输入 product name 为 libsql



、把 libsql.h 和 libsql.m 删除。导入 ocsqlite.h 和 ocsqlite.c（文件见 http://blog.csdn.net/fengsh998/article/details/827897



、修改 scheme，设为 release 版本。



OK，选译 ios device 编译运行。成功后将在目录的 build/products/release-iphoneos/下产生一个 liblibsql.a 文件。

主，这里产生的是真机使用的.a 文件。

选译 iphonesimulator 进行编译一次，同样会在 build/products/release-iphonesimulator/下产生一个 liblibsql.a 文件。

这里是虚拟机使用的.a 文件。

下面来看一下这两个文件有什么不同之处，使用 lipo -info 命令。

打开终端。

进入到相应的目录。

真机的：liblibsql.a 文件信息。

nput file liblibsql.a is not a fat file

Non-fat file: liblibsql.a is architecture: armv7

如图：

```
Last login: Thu Dec 13 01:08:01 on ttys000
f-fshmatomac:~ ffsh$ cd desktop/libsql/derivedata/libsql/build/products/release
-iphoneos
f-fshmatomac:release-iphoneos ffsh$ lipo -info liblibsql.a
input file liblibsql.a is not a fat file
Non-fat file: liblibsql.a is architecture: armv7
f-fshmatomac:release-iphoneos ffsh$
```

模拟器的：liblibsql.a 文件信息。

nput file liblibsql.a is not a fat file

Non-fat file: liblibsql.a is architecture: i386

如图：

```
f-fshmatomac:~ ffsh$ cd desktop/libsql/derivedata/libsql/build/products/release
-iphonesimulator
f-fshmatomac:release-iphonesimulator ffsh$ lipo -info liblibsql.a
input file liblibsql.a is not a fat file
Non-fat file: liblibsql.a is architecture: i386
f-fshmatomac:release-iphonesimulator ffsh$
```

如果使用真机和模拟器通用，则需要将这两个文件合并，使用命令 lipo

create xxxx/liblibsql.a xxxxx/liblibsql.a -output libsql.a

同样可以使用 lipo -info 来查看这个合并的 libsql.a

可以看到 architectures in the fat file: libsql.a are: i386 armv7

如图：

```
f-fshmatomac:release-iphonesimulator ffsh$ lipo -create liblibsql.a /users/ffsh/
desktop/libsql/derivedata/libsql/build/products/release-iphoneos/liblibsql.a -o
utput libsql.a
f-fshmatomac:release-iphonesimulator ffsh$ lipo -info libsql.a
Architectures in the fat file: libsql.a are: i386 armv7
f-fshmatomac:release-iphonesimulator ffsh$
```

OK,打包完成。这个静态库比起 framework 是不是简单了许多呢。完工，如果想打包 framework 请参考：

http://blog.csdn.net/fengsh998/article/details/8290687