boreal-kiss.net

日本語 / English

Contact: info [AT] boreal-kiss.com

Technical topics about Cocoa and Cocoa touch.

How to create universal static libraries on Xcode 4: the traditional way

by borealkiss

[Note on 09.04.2011: This article will not be helpful unless you really need universal static libraries for distribution purposes etc. Instead, using workspace is a better way for daily development.]

Introduction

Creating static libraries universal for iPhone device and simulator is a little bit tricky. On Xcode 3.x, the major way takes something like the following steps:

- Create a build for the device (the armv6 and armv7 architecture).
- Create a build for the simulator (the i386 architecture).
- Merge the above builds to create a universal build using lipo.

Since Xcode, 4 a new command called archiving (Product > Archive) is introduced as a way to distribute applications and/or static libraries. At glance this is suitable to creating a universal libraries but is not. One can not select this option for the simulator environment (you will notice it is always grayed out). As such we still have to rely on the traditional way to create universal libraries ([1]).

In this article I will show you how to create such a fat static library on Xcode 4. Because the place for product builds of each project are completely changed from those of Xcode 3.x, we have to make a slight effort tailoring it to the Xcode 4.

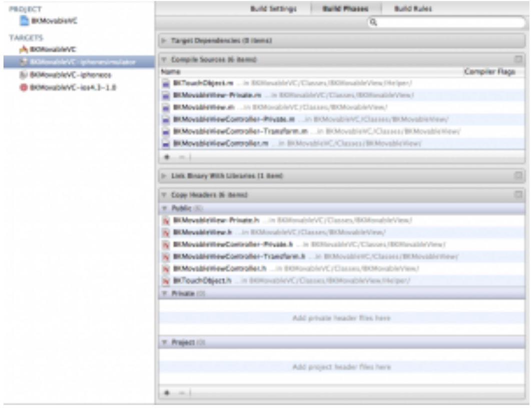
How to do

In the following example I will use my project called BKMovableVC. On your workspace you can use whatever name you war of course.

Step 1

Create a new *Cocoa Touch Static Library* target (File > New > New Target > Framework & Library) for the simulator environment. Naming is arbitrary as long as you can recognize (I named here BKMovableVC-iphonesimulator). Add related source files to the *Compile Source* pane and headers to *Copy Header* pane. For headers you can choose three possible places to put in, Public , Private, and Project (default is Project). For example if you choose Public, all headers will be automatically copied to the build directory, which is easy for me to continue following works.

[Note on 04.05.2011: There is a bug on Xcode 4; when all panes (*Public*, *Private*, and *Project*) in the *Copy Header* pane are closed or never opened before, pushing the add button causes a crash. Open one of them before adding files.]



BKMovableVC

BKMovableVC
BKMovableVC-iphonesimulator
BKMovableVC-iphoneos
BKMovableVC-ios4.3-1.0

Target Dependencies (0 items)

Compile Sources (6 items)

Name	Compiler Flags
BKTouchObject.m ...in BKMovableVC/Classes/BKMovableView/Helper/	
BKMovableView-Private.m ...in BKMovableVC/Classes/BKMovableView/	
BKMovableView.m ...in BKMovableVC/Classes/BKMovableView/	
BKMovableViewController-Private.m ...in BKMovableVC/Classes/BKMovableView/	
BKMovableViewController-Transform.m ...in BKMovableVC/Classes/BKMovableView/	
BKMovableViewController.m ...in BKMovableVC/Classes/BKMovableView/	

Link Binary With Libraries (1 item)

Copy Headers (6 items)

Public (6)

BKMovableView-Private.h ...in BKMovableVC/Classes/BKMovableView/
BKMovableView.h ...in BKMovableVC/Classes/BKMovableView/
BKMovableViewController-Private.h ...in BKMovableVC/Classes/BKMovableView/
BKMovableViewController-Transform.h ...in BKMovableVC/Classes/BKMovableView/
BKMovableViewController.h ...in BKMovableVC/Classes/BKMovableView/
BKTouchObject.h ...in BKMovableVC/Classes/BKMovableView/Helper/

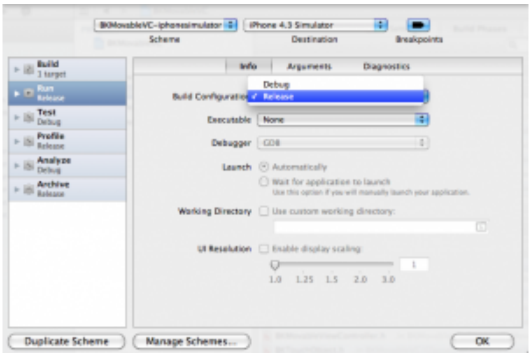
Private (0)

Add private header files here

Project (0)

Add project header files here

From the *Edit Scheme* pane (Product > Edit Scheme), change its build configuration to *Release*. This enables you to create a *Release* build product once the build operation is done. This build operation supposes to be done on the simulator architecture (i386).



Scheme: BKMovableVC-iphonesimulator
Destination: iPhone 4.3 Simulator

Build 1 target
Run Release
Test Debug
Profile Release
Analyze Debug
Archive Release

Build Configuration: Debug / ✓ Release
Executable: None
Debugger: GDB
Launch: ● Automatically ○ Wait for application to launch
Use this option if you will manually launch your application.
Working Directory: □ Use custom working directory:
UI Resolution: □ Enable display scaling 1.0 1.25 1.5 2.0 3.0 1

Step 2

Create a new static library target for the iPhone device environment. Naming is arbitrary but must be different from that of the simulator (I named here BKMovableVC-iphoneos). Add related source files and headers and change its build configuration to *Release*. Every step is just the same as that of the simulator except for the build architecture; this build operation supposes to be done on the device architecture (armv6 and armv7).

Now we have two targets, one for the device and the other for the simulator. When you build a product, the product will be placed under the following directory:

```
~/Library/Developer/Xcode/DerivedData/
```

This place is shared by build products of all projects created on Xcode 4, which is quite different from the legacy place found on Xcode 3.x (each project has its own build directory). For example, if the name of the project is BKMovableVC, the *Release* build product for the simulator will be placed at

```
~/Library/Developer/Xcode/DerivedData/BKMovableVC-xxxxxxx/
        Build/Products/Release-iphonesimulator/
        libBKMovableVC-iphonesimulator.a
```

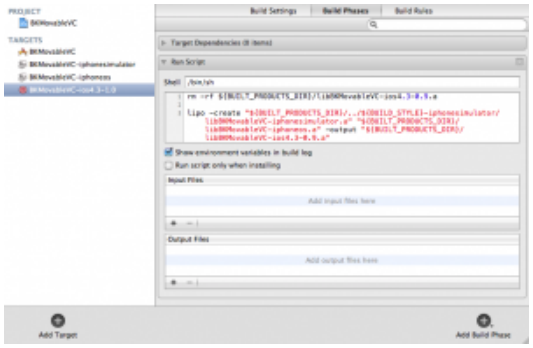
and that for the device will be found at

```
~/Library/Developer/Xcode/DerivedData/BKMovableVC-xxxxxxx/
        Build/Products/Release-iphoneos/
        libBKMovableVC-iphoneos.a
```

where BKMovableVC-xxxxxxx is a unique directory name automatically assigned by Xcode 4.

Step 3

Let’s merge them using the lipo command. Create a new Aggregate target (File > New > New Target > Other). I named it here BKMovableVC-ios4.3-0.9. On the *Build Pheses* pane select *Add Run Script* from the *Add Build Phase* button on the bottom-left corner.



Shell /bin/sh

```
rm -rf ${BUILT_PRODUCTS_DIR}/libBKMovableVC-ios4.3-0.9.a

lipo -create "${BUILT_PRODUCTS_DIR}/../${BUILD_STYLE}-iphonesimulator/
    libBKMovableVC-iphonesimulator.a" "${BUILT_PRODUCTS_DIR}/
    libBKMovableVC-iphoneos.a" -output "${BUILT_PRODUCTS_DIR}/
    libBKMovableVC-ios4.3-0.9.a"
```

☑ Show environment variables in build log
□ Run script only when installing

Now we are about to run some scripts on the Run Script pane but here is a problem. How do we direct lipo to work with builds that are placed under the directory whose name is automatically assigned by Xcode 4 (e.g., BKMovableVC-xxxxxxx for the above case)? Fortunately it is easily solved by the Xcode 4 environment variable called ${BUILT_PRODUCTS_DIR}. This variable represents the path to the directory for a current build product. For example, when we create a *Release* build product for the simulator explained above, ${BUILT_PRODUCTS_DIR} corresponds to the following path:

```
~/Library/Developer/Xcode/DerivedData/BKMovableVC-xxxxxxx/
        Build/Products/Release-iphonesimulator/
```

Let's continue. Write the following script on the *Run Script* pane. This lets lipo create a merged build product called libBKMovableVC-ios4.3-0.9.a (you can give it whatever name you want). If there is already the product with the same name lipo deletes it before the merge operation as shown in the first line.

```
rm -rf ${BUILT_PRODUCTS_DIR}/libBKMovableVC-ios4.3-0.9.a

lipo -create "${BUILT_PRODUCTS_DIR}/../${BUILD_STYLE}-iphonesimulator/libBKMovableVC-iphonesimulator.a" \
"${BUILT_PRODUCTS_DIR}/libBKMovableVC-iphoneos.a" -output \
"${BUILT_PRODUCTS_DIR}/libBKMovableVC-ios4.3-0.9.a"
```

Here is one trick. In this script the path representation to previously created build products such as libBKMovableVC-iphonesimulator.a looks odd because of "/../". This is because the environment variable ${BUILT_PRODUCTS_DIR} depends on a current build phase (iPhone device, simulator, *Debug*, *Release*). For example, if this script runs for the *Release* build of iPhone device, the variable ${BUILT_PRODUCTS_DIR} will represent

```
~/Library/Developer/Xcode/DerivedData/BKMovableVC-xxxxxxx/
        Build/Products/Release-iphoneos/
```

However the build product for the simulator, libBKMovableVC-iphonesimulator.a, is not placed under this path, because it was created through a different build phase.

Implementation

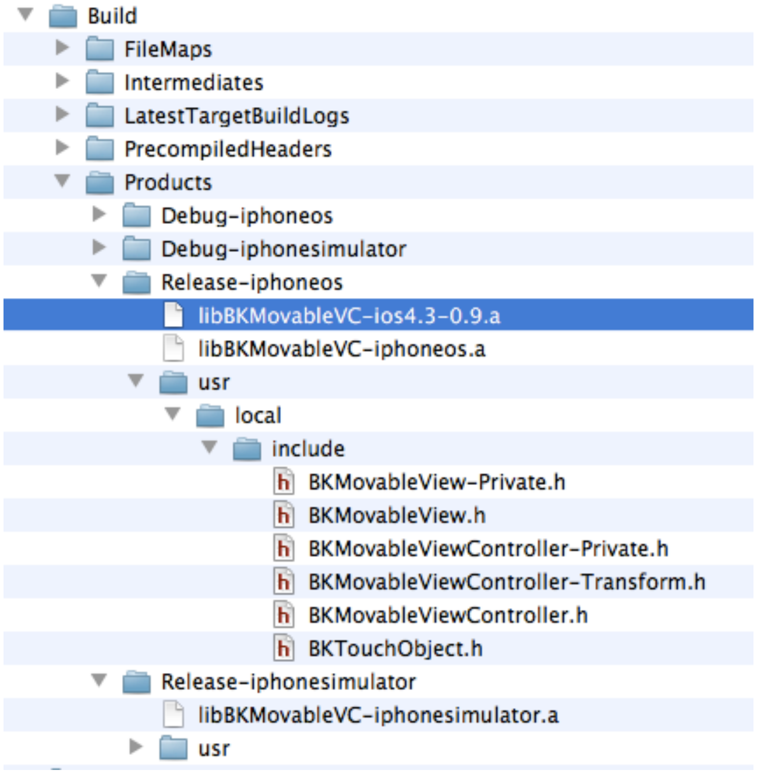
Let's do all the flows in order. Do the following:

1. Build the target BKMovableVC-iphonesimulator on the simulator. The build phase must be *Release*.
2. Build the target BKMovableVC-iphoneos on the device. The build phase must be *Release*.
3. Build the target BKMovableVC-ios4.3-0.9. on the device. The build phase must be *Release*.

Then you will have a static library universal for the device and simulator in the following directory:

```
~/Library/Developer/Xcode/DerivedData/BKMovableVC-xxxxxxx/
        Build/Products/Release-iphoneos/
        BKMovableVC-ios4.3-0.9.a
```

You will also have header files depending upon your configuration (see the text above).



```
▼ Build
    ▶ FileMaps
    ▶ Intermediates
    ▶ LatestTargetBuildLogs
    ▶ PrecompiledHeaders
    ▼ Products
        ▶ Debug-iphoneos
        ▶ Debug-iphonesimulator
        ▼ Release-iphoneos
              libBKMovableVC-ios4.3-0.9.a
              libBKMovableVC-iphoneos.a
            ▼ usr
                ▼ local
                    ▼ include
                          BKMovableView-Private.h
                          BKMovableView.h
                          BKMovableViewController-Private.h
                          BKMovableViewController-Transform.h
                          BKMovableViewController.h
                          BKTouchObject.h
        ▼ Release-iphonesimulator
              libBKMovableVC-iphonesimulator.a
            ▶ usr
```

Footnotes

1. Of course there could be a way to create a universal library on one scheme by tuning build settings but I have not found it by myself nor on the web. [↩]

Published: March 15, 2011
Filed Under: Uncategorized
Tags: iPhone : Xcode

32 Responses to "How to create universal static libraries on Xcode 4: the traditional way"

1. *Ambrose* says:
April 16, 2011 at 5:08 am

Great post! Clear, precise and most of all, it works! Thanks for saving my time. Keep up the good work!

Reply

2. *Delta5* says:
May 3, 2011 at 8:26 pm

This tutorial looks great….. but xCode 4 crashes every time I try to add to the the 'Copy Headers' portion of the build Phases. What version of xCode were you using? I am using Version: 4.0 (4A304a).

Reply

- *borealkiss* says:
May 4, 2011 at 1:48 am

Delta5, it’s an actual bug for Xcode 4 I have already experienced. When all panes (Public, Private, Project) in Copy Headers are closed (it’s never opened before), Xcode 4 crashes. Open one of them before adding files. Then push the add button.

Reply

3. *Andy* says:
   May 4, 2011 at 1:24 pm

   This is great and thank you so much! Question -> the library that I produce needs to then be included in another project. In Xcode 3.2, I just included the library that was generated straight away in my other project and, as long as the “library project” was built first, the other project would automatically link in the newest version of the library. Is there any way in Xcode 4 such that, after building the library, I can just open my other project that depends on the built library and have it build? Or, do I need to manually (or, write a script; still manually since you need to run the script) move the newly generated .a file into my other project each and every time I re-generate the .a file?

   Reply

   - *borealkiss* says:
     May 5, 2011 at 2:15 am

     Andy, use workspace that automatically build necessary libraries among projects. In the case in my blog post, you have to something manually, i.e., update libraries in one project first, then use it in another.

     Reply

4. *Devang* says:
   May 19, 2011 at 3:45 am

   Nice Article. I am able to get the products built as described, but am not getting the headers in the Release-iphoneos folder. Any suggestions/ideas on where i should get back and double check the things.
   Thanks!

   Reply

   - *borealkiss* says:
     May 19, 2011 at 4:00 am

     Devang, make sure you put headers into the Copy Headers pane (Public) in your device target. The detail how to copy headers is described in Step 1.

     Reply

5. *Jamie Chapman* says:
   May 27, 2011 at 3:38 am

   Fantastic post, thanks so much! Really helped me with my static library woes in Xcode 4.

   Reply

6. *Erde* says:
   July 3, 2011 at 6:12 am

   Nice post, but i’m having some trouble. I did everything as you stated it, no how should i do to get the .a file. I went to: /Library/Developer/Xcode/DerivedData/MyProject/
   Build/Products/Release-iphoneos but i didn’t find any .a file. What should i do to generate this file, should i only run the project? Could the file be in another directory? Thanks.

Reply

- *borealkiss* says:
  July 3, 2011 at 6:26 am

  Erde, make sure you are running a static library target which includes some source files to be archived. Once again try only Step 1 and run the static library target in order to confirm you are actually creating a .a file.

  Reply

  - *Erde* says:
    July 3, 2011 at 7:50 am

    what do you mean by 'run the static library target' ? Run the project on the device? i did step one again with no success after running the project.

    Reply

    - *Erde* says:
      July 3, 2011 at 7:54 am

      the only thing i find in the directory is 2 files one with my project's name and the other one MyProjectName.app.dSYM. Thanks for your time. By the way i'm doing Step 2 because i'm running on a device.

      Reply

    - *borealkiss* says:
      July 3, 2011 at 8:04 am

      Set the proper scheme corresponding to your static library target and build it. It seems you only built the project scheme for your app.

      Reply

      - *Erde* says:
        July 3, 2011 at 8:16 am

        you were absolutely right i was changing the scheme to the hole project and not to the static library i just created. Thanks, you saved my day!

7. *how to generate the .a file when creating static library iphone | taking a bite into Apple* says:
   July 3, 2011 at 3:06 pm

   [...] Do i have to do something special to generate this file. I'm following the steps in this [...]

   Reply

8. *ankit* says:
   July 5, 2011 at 7:00 am

   How do i use this Library in my other project.. sorry for the lame question 😛

   Reply

   - *borealkiss* says:
     July 5, 2011 at 7:18 am

Just add the static library on your project with header files and necessary resources.

Reply

9. *slushduck* says:
   July 11, 2011 at 5:50 pm

   Thanks very much! This was the best resource I've found and was able to create the static lib with little difficulty!
   Quick question though: once I've got the includes and the .a file, how does one go about packing it up into a nice Framework file?

   Thanks again!

   Reply

   - *slushduck* says:
     July 11, 2011 at 6:18 pm

     Alright seems like a framework file is just a directory structure with no extra magic required. Simply organizing the files as below did the trick:

     MyLib.framework [add the .framework extension]
     -Headers [Add this directory]
     [put headers here]
     MyLib. [copy the static lib here and remove the .a extension]

     Cheers!

     Reply

     - *sambit* says:
       December 14, 2011 at 2:03 am

       Ok you have converted the .a file to .framework extension. Can you please tell me step by step procedure to do this. I have successfully got the libMyframe.a file. In the framework I have also added few images also. Now my problem is I have to convert this .a extension to . framework extension and add this to my project and even I need to distribute this framework. Please tell me mouse click by click step to do the conversion work till adding it to my project successfully.

       Reply

       - *borealkiss* says:
         December 22, 2011 at 6:04 am

         Creating static libraries is the goal. Using frameworks is not allowed on iOS. You can do it only on Mac platform.

         Reply

10. *kra* says:
    July 24, 2011 at 1:51 pm

    That used to work fine in xcode 4.0 but now just boke with 4.1.
    When building for the simulator, xcode skips the device part and you can't create the fat library.
    Weird part is that when you build for the device, it works fine… I'm trying to tweak the build settings to get that working again, but i've had no luck so far

    Reply

11. *kra* says:
[July 24, 2011 at 2:54 pm]

Ok, so I'm not quite sure I understand what has changed in XCode4.1, but it just won't build as is.

Basically, when building for device (iOS device selected in the scheme drop down), everything goes fine, Xcode is running properly both targets (device and sim).
When building for simulator, Xcode is skipping the device target. Looks like the only defined arch is i386, the best thing I could get is have the device build… for i386. Not very helpful.

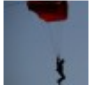
The way I fixed our project for now is to do the following:
- keep both device/sim targets as is
- update the aggregate target to NOT depend on the device target
- update the post build script to explicitly build the device target, so that would be something like:

```
xcodebuild -configuration "${CONFIGURATION}" -target "Native Target Name" -sdk "iphoneos" ${ACTION}
RUN_CLANG_STATIC_ANALYZER=NO BUILD_DIR="${BUILD_DIR}" BUILD_ROOT="${BUILD_ROOT}"
```

It definitely lacks the elegance of your current solutions, but it works.

I'd be very glad if anybody has any idea how to fix these issues…

Reply

12.  *kra* says:
July 24, 2011 at 2:56 pm

Oh, and one last thing.
${BUILD_STYLE} was deprecated a long time ago for ${CONFIGURATION}.
Looks like apple decided to drop it altogether, so your lipo command line should be update with the new variable name.

Reply

13. *Jimmy Koerting* says:
July 28, 2011 at 11:20 am

First of all – thanks for your nice howto here.
I did like you showed it here and the creation of the lib worked like charme. But when I try to use it, the using project fails with 'Symbol(s) not found for architecture armv7′ (and armv6 or i386 if I build with simulator):

Undefined symbols for architecture armv7:
"_myfuncition", referenced from:
-[useLibraryTestViewController buttonClicked:] in useLibraryTestViewController.o

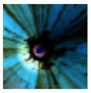
I did tried this NOT with objective-c code, but with c++ code. I'm new to both c++ and xcode but I want to include a given c++ library into a objective-c project for the iphone.

Can you give me a hint why this happens? mylib.a is listed it the 'Link Binary With Libraries' pane and the header file is there too, so I have no clue what's wrong.

Any hint is really welcome!!

Jimmy

Reply

    ○  *borealkiss* says:
    July 28, 2011 at 4:58 pm

It seems your library was not compiled for armv7. Searching the error messages on google will give you more info.

Reply

14. *Jimmy Koerting* says:
July 29, 2011 at 8:27 am

    That is the fun – it is. One build is done for armv6+armv7 and the other is for i386. This is why I'm confused. Could there be something from with my c++ code that raises such error. E.g. that my c++ file is not exposing the function, and this gives this misleading message? On the other hand, the compiler does not complain anything in my code, but right code does not say right code for THIS work, correct?

    Again, any further idea would be of help.

    Jimmy

    Reply

15. *Oscar Salvador* says:
December 5, 2011 at 7:15 pm

    Hi!. I found this tutorial and after reading the comments I was really excited. However i got stock when I tried to change the scheme and then I would get tons of errors and could not build. If I change the scheme again no errors. What am i doing wrong?

    Reply

16. *Oscar Salvador* says:
December 5, 2011 at 7:17 pm

    this is for step 1

    Reply

17. *Yogan* says:
December 9, 2011 at 9:49 am

    Great post!!!!! All works.

    Reply

18. *sambit* says:
December 14, 2011 at 1:47 am

    I was successful till getting the .a file in the derived data folder.I also included the lib…. .a file to the link binary with libraries.But next when I compiled I got an error of i386.Can somebody tell me what to do. Have I added the framework properly or something more has to be done

    Reply

19. *sambit* says:
December 14, 2011 at 1:53 am

    After creating the lib…. .a file next is how to convert it into .framework extension .

    Reply

Leave a Comment

Name: *Required*

Email: *Required, not published*

Homepage:

Comment:

Post Comment

« Previous Post

Next Post »

Elsewhere

Search