Xcode 4.x command line tools reference

By Oleksandr Dodatko, 2011-05-03

Table of Contents

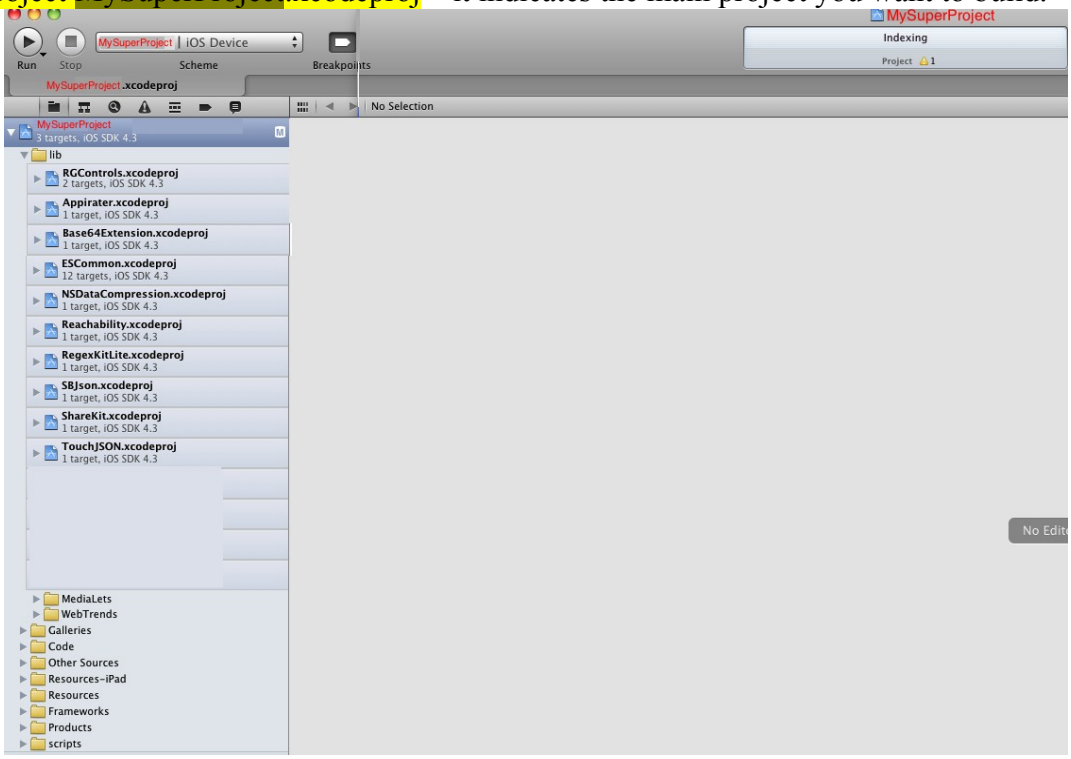
Building the project

In order to build the iPhone/iPad project from the command line, you should use the “xcodebuild” tool [1] which is shipped with the iOS SDK.

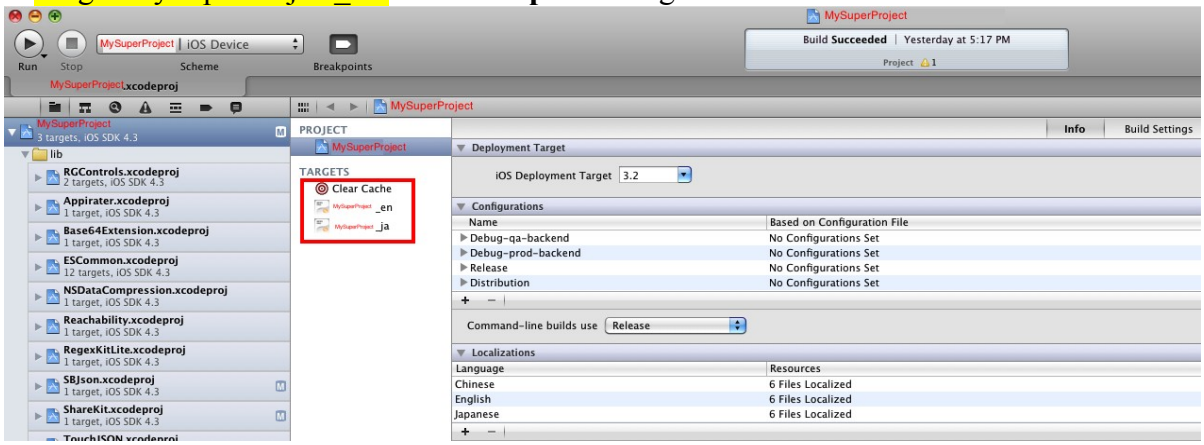
Let's start with the MySuperProject project build command example.
xcodebuild -project MySuperProject.xcodeproj -target MySuperProject_en -configuration Release -parallelizeTargets -sdk iphoneos4.3 clean build

Now let's consider its components one by one.

1. -project MySuperProject.xcodeproj – it indicates the main project you want to build.



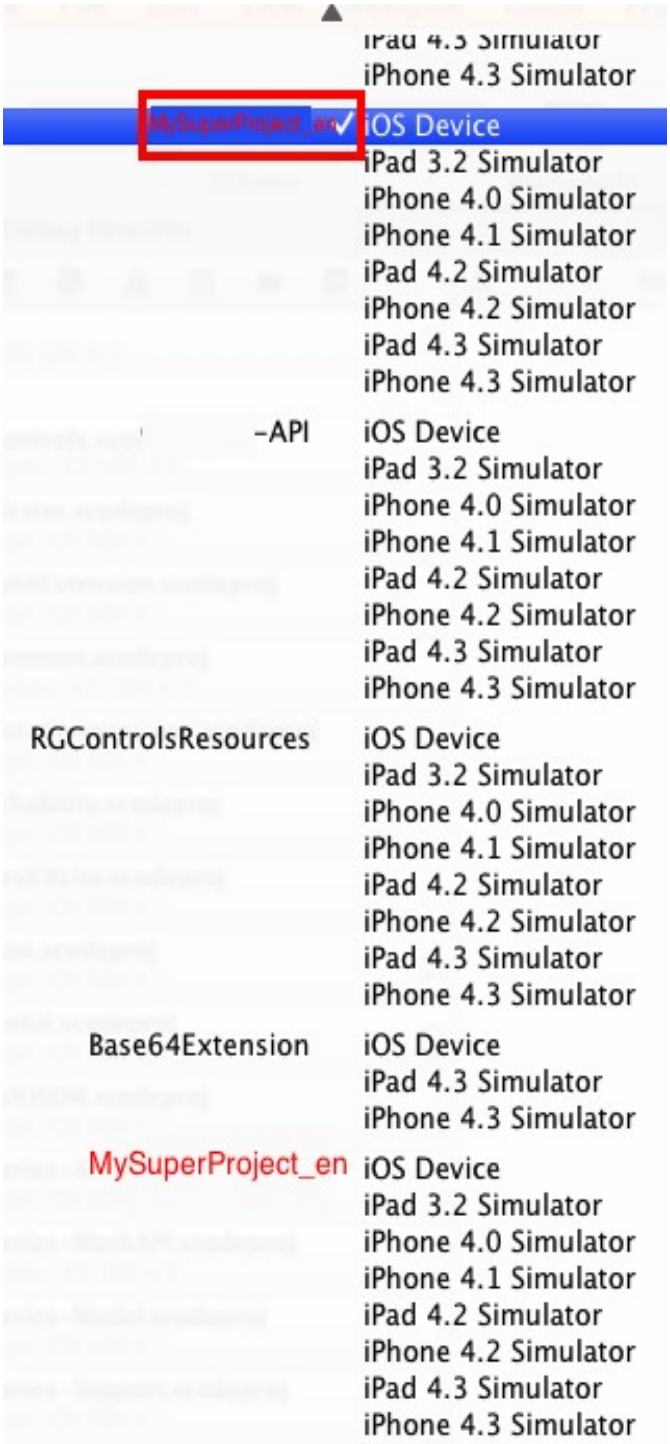
2. -target MySuperProject_en – select a **specific** target to build.



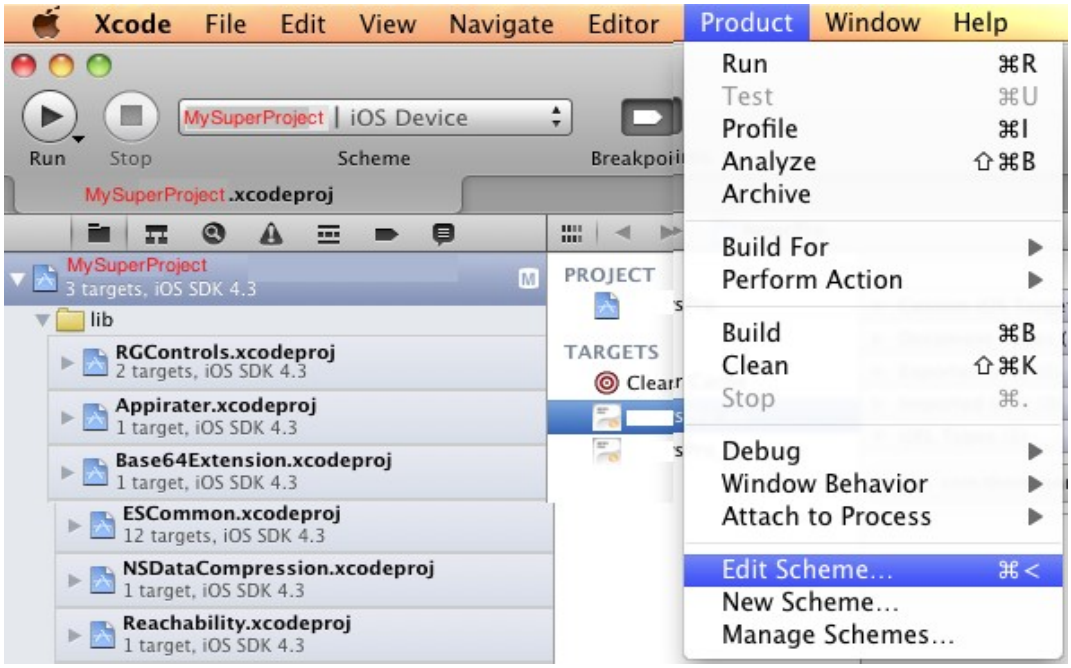
3. -alltargets – an alternative to a previous command. Builds all targets on the list.

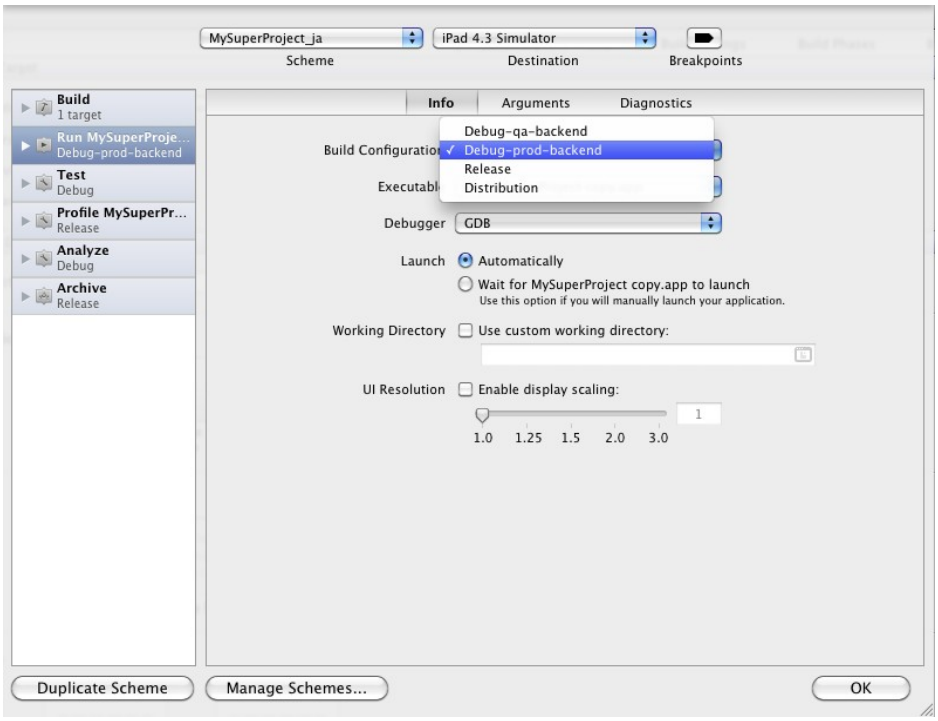
Note: xCode 4.x and up shows also the targets of the referenced projects. This command line option does not apply to them. It builds ONLY the targets of the project, passed as the **-project** argument.

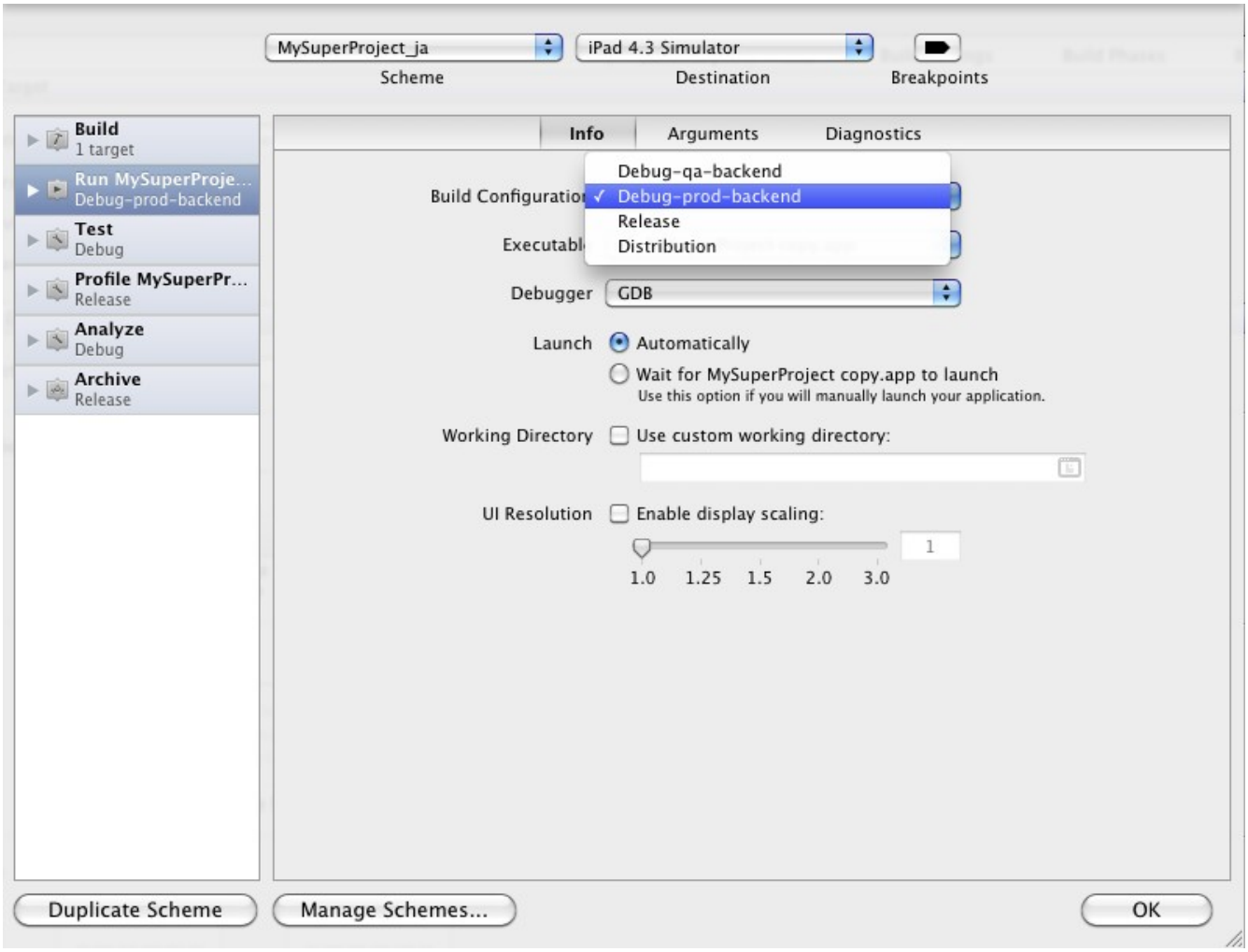
4. -activetarget – uses a selected target. Usually It is not versioned by svn/git. Please use one of the specified above instead.



5. -configuration Release – specifying a desired configuration for selected target/targets

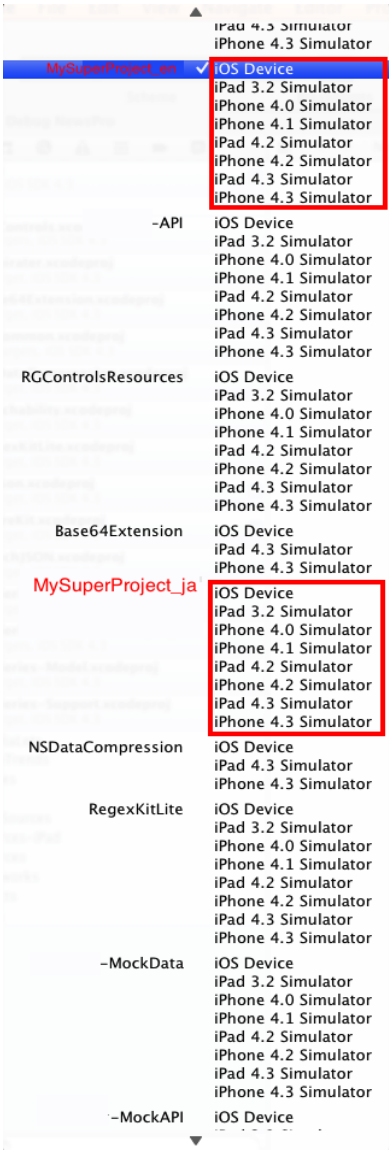






6. -activeconfiguration – uses a selected configuration. Usually It is not versioned by svn/git. Please use one of the specified above instead.

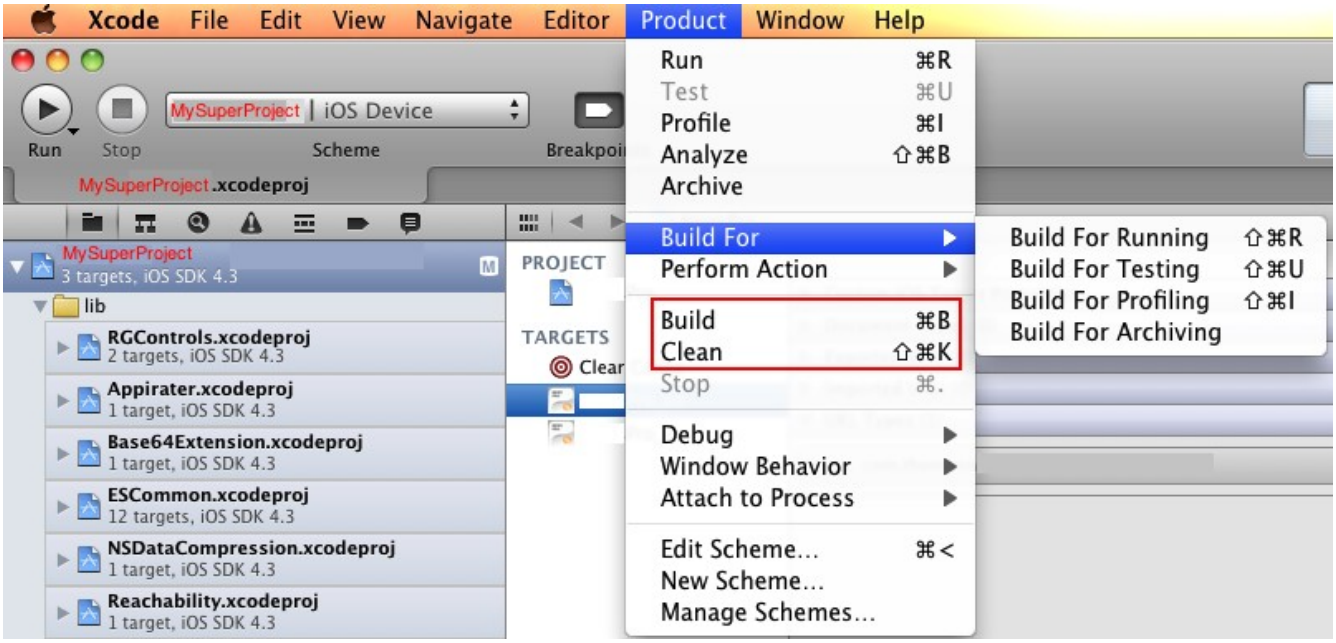
7. -sdk – iphone/ipad, device/simulator build.



If not specified – the default one (the sdk from the user settings) will be used.

8. -parallelizeTargets – enable build speed optimizations if possible

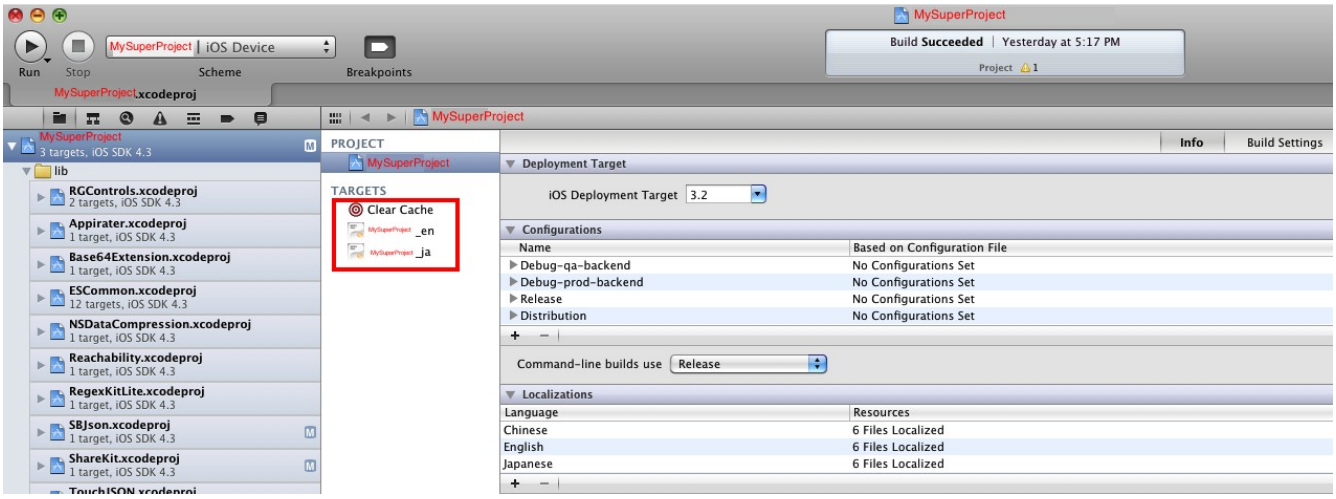
9. clean build – build actions. The mentioned actions are most often used with the continious integraion server. At the developer machine you'll most likely use Xcode GUI.

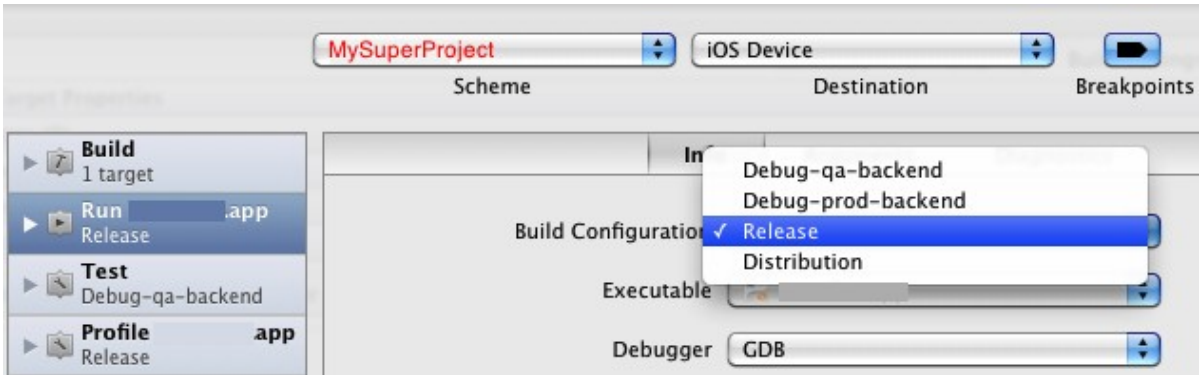


Getting available build settings

1. Getting the list of targets and configurations

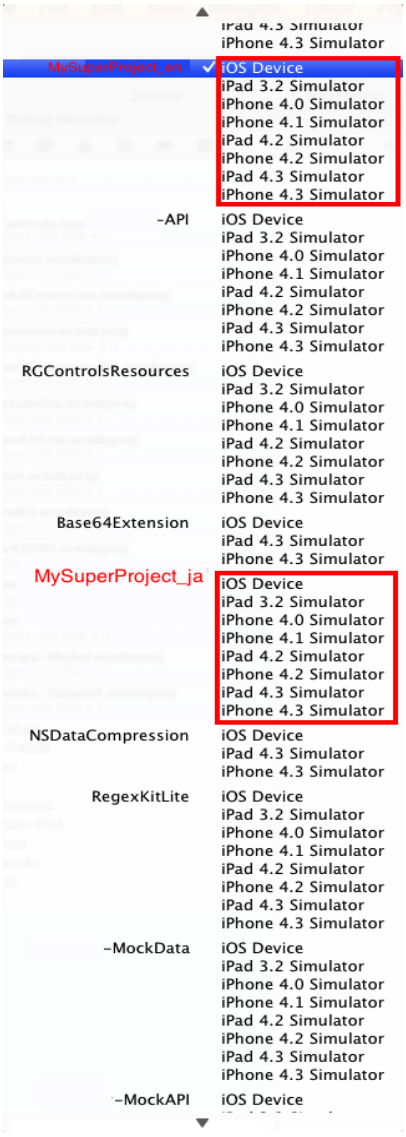
```
Oleksandr_Dodatko$ xcodebuild -project MySuperProject.xcodeproj -list
Information about project "MySuperProject":
    Targets:
        Clear Cache
        MySuperProject_en
        MySuperProject_ja

    Build Configurations:
        Debug-qa-backend
        Debug-prod-backend
        Release
        Distribution

    If no build configuration is specified "Release" is used.
```





2. Getting list of available SDKs.
   Use the -showsdks flag. Use it without specifying other options

```
:~ Oleksandr_Dodatko$ xcodebuild -showsdks
Mac OS X SDKs:
        Mac OS X 10.6                   -sdk macosx10.6

iOS SDKs:
        iOS 4.3                 -sdk iphoneos4.3

iOS Simulator SDKs:
        Simulator - iOS 3.2             -sdk iphonesimulator3.2
        Simulator - iOS 4.0             -sdk iphonesimulator4.0
        Simulator - iOS 4.1             -sdk iphonesimulator4.1
        Simulator - iOS 4.2             -sdk iphonesimulator4.2
        Simulator - iOS 4.3             -sdk iphonesimulator4.3
```



Iphone SDK installation rule

Install most recent versions of the XCode to the standard (/Developer/) directory.
Reinstall older versions (if required) to the directories of your choice.

This will save you from unwanted command line tools related problems.

Running compiled iphone/ipad applications on the simulator without XCode.

We can use the iphonesim [2] program to do that.
iphonesim launch "$TEST_PROJECTS_PATH/MyProjectTest/build/Release-iphonesimulator/MyProjectTest.app" 4.2 ipad

```
:~ Oleksandr_Dodatko$ iphonesim
Usage: iphonesim <options> <command> ...
Commands:
  showsdks
  launch <application path> [sdkversion] [family] [uuid]
```

The arguments are case sensitive.

Application argument

Note : application path MUST be a full path. Relative paths are not supported.

SDK argument

SDK version can be obtained with the following command : iphonesim showsdks

```
:~ Oleksandr_Dodatko$ iphonesim showsdks
[DEBUG] Simulator SDK Roots:
[DEBUG] 'Simulator - iOS 3.2' (3.2)
        /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator3.2.sdk
[DEBUG] 'Simulator - iOS 4.0' (4.0)
        /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.0.sdk
[DEBUG] 'Simulator - iOS 4.1' (4.1)
        /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.1.sdk
[DEBUG] 'Simulator - iOS 4.2' (4.2)
        /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.2.sdk
[DEBUG] 'Simulator - iOS 4.3' (4.3)
        /Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk
```

Family argument

At the moment only 2 options are available :

- iphone
- ipad

"iphone" value is default one.

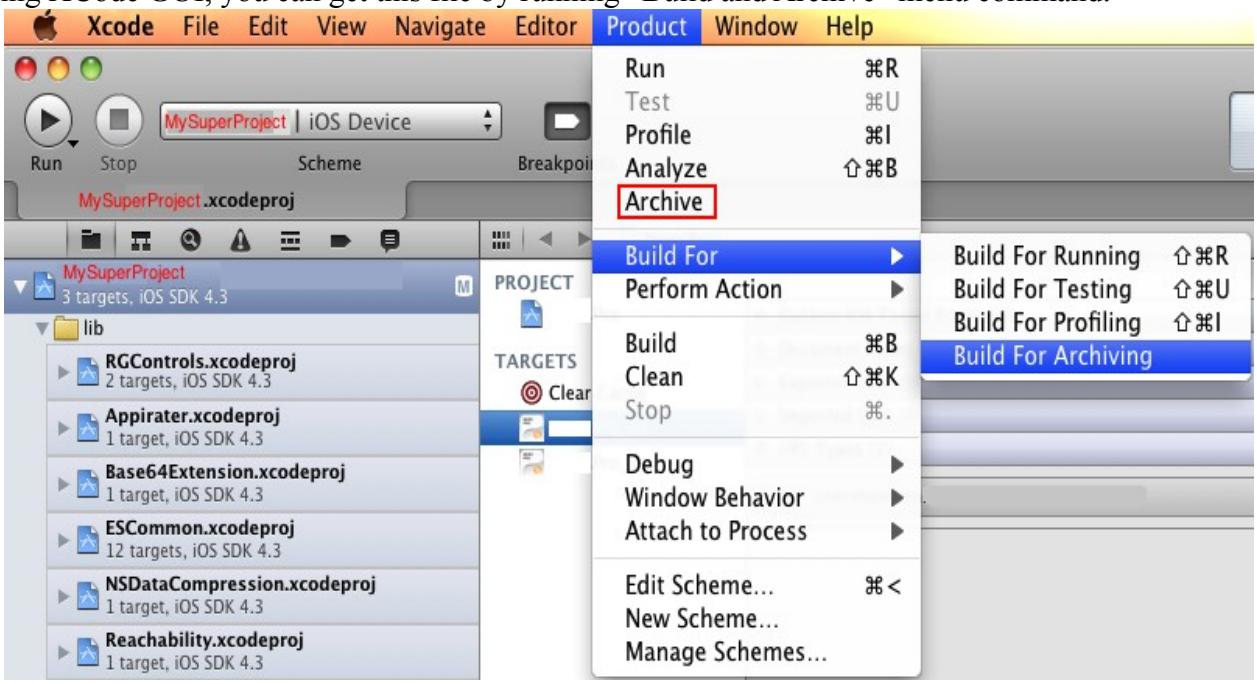
Deploying applications for QA (creating *.ipa files)

*.ipa files are used to install your compiled applications without deploying them to the app store. However, the devices count is limited to 100 and they have to be registered with your provisioning and developer profile.

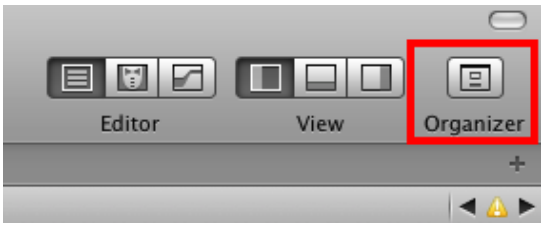
However, they are just great for the in-house applications testing. The benefits are :

1. Mac OS X enabled workstations are not required for testers
2. QA team members do not have to touch the developer tools (versions control, xcode, etc.). Just during the ordinary desktop applications development process
3. QA team members do not have to deal with the apple provisioning profiles.

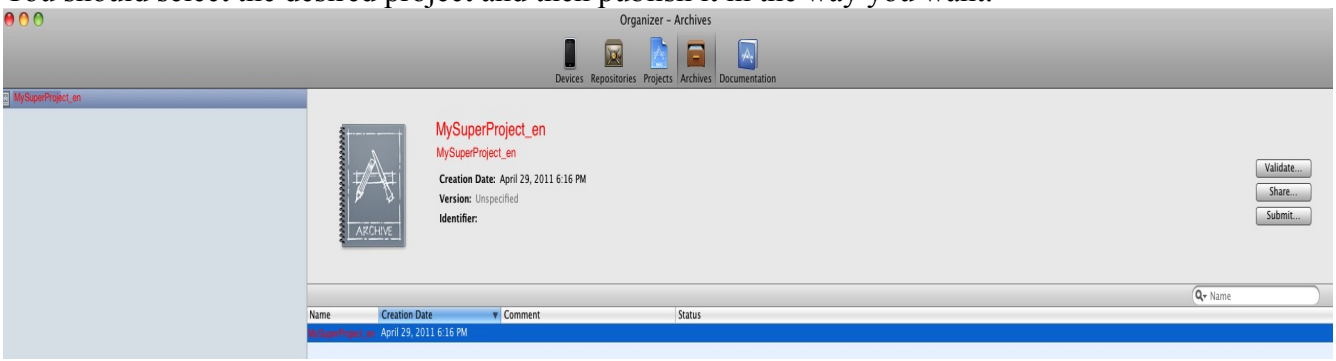
Using XCode GUI, you can get this file by running "Build and Archive" menu command.

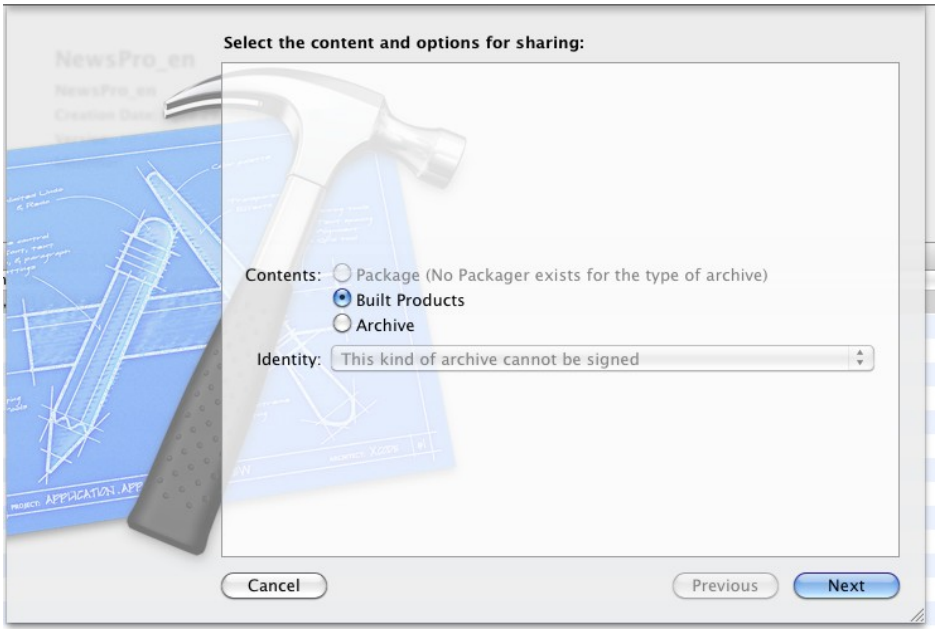


Build results can be found in the **organizer** window. It can be accessed with the following menu item :



You should select the desired project and then publish it in the way you want.





To find out more about provisioning, you should consider apple documentation. [3], [4].
To find out more about continious integration, see [5]

Build instructions, described here, can be found at stackoverflow.com – [6], [7]. However, they are subject to change as the newer versions of the XCode are released.

Here is a command for the MySuperProject.app packaging.

```
/usr/bin/xcrun -sdk iphoneos PackageApplication -v $BUILT_PRODUCTS_DIR/MySuperProject.app -o $PROJECT_ROOT/deployment/MySuperProject.ipa --sign iPhone Developer: Oleksandr Dodatko (ABCD123456) --embed $PROJECT_ROOT/certificates/MySuperProjectiPad.mobileprovision
```

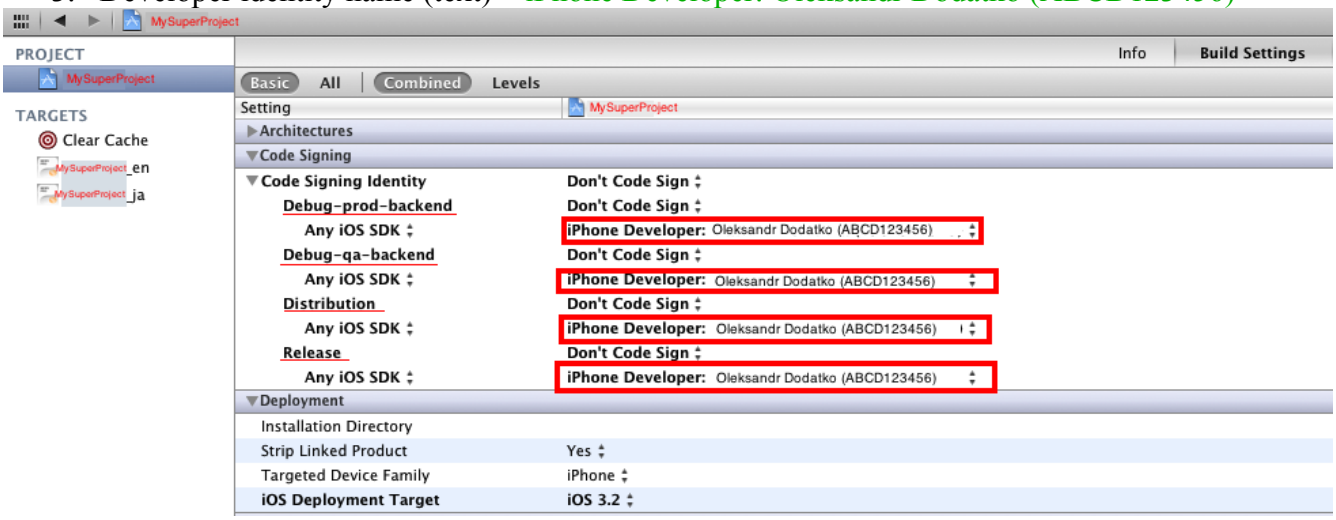
All you have to do is :

1. Setup all required certificates for your build machine (see apple documentation on provisioning).
2. Build the application for the desired iOS device configuration.
3. Use the **/usr/bin/xcrun** command to digitally sign the application.

In order to sign the app, you have to specify:

1. Path to the built app (bundle directory) – $BUILT_PRODUCTS_DIR/MySuperProject.app
2. Path to the result file (file) – $PROJECT_ROOT/deployment/MySuperProject.ipa

3. Developer identity name (text) – iPhone Developer: Oleksandr Dodatko (ABCD123456)

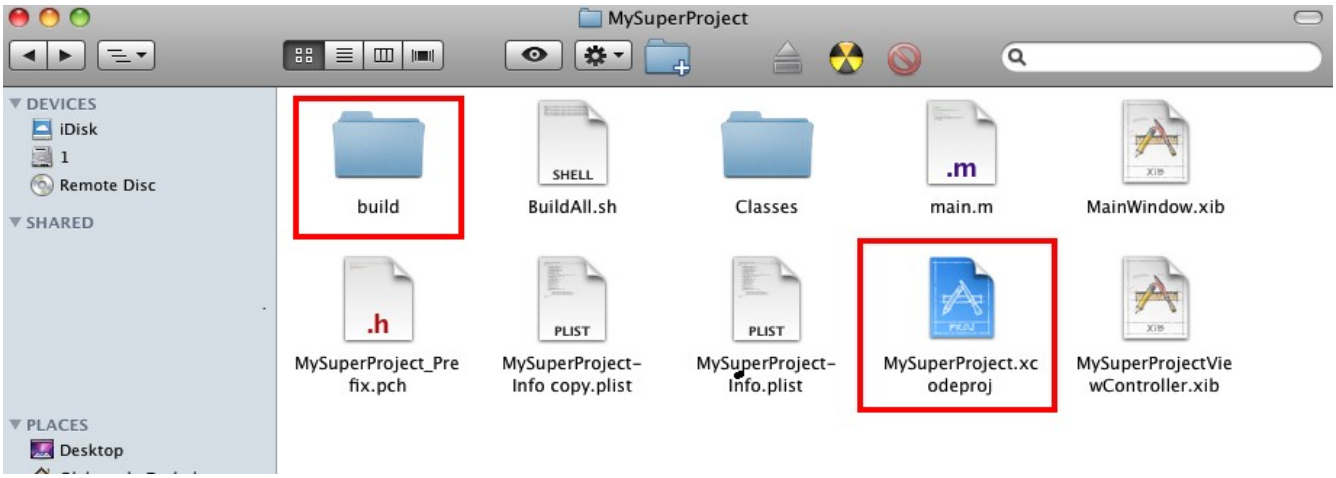


4. Path to the provisioning certificate file (file) – $PROJECT_ROOT/certificates/MySuperProjectiPad.mobileprovision

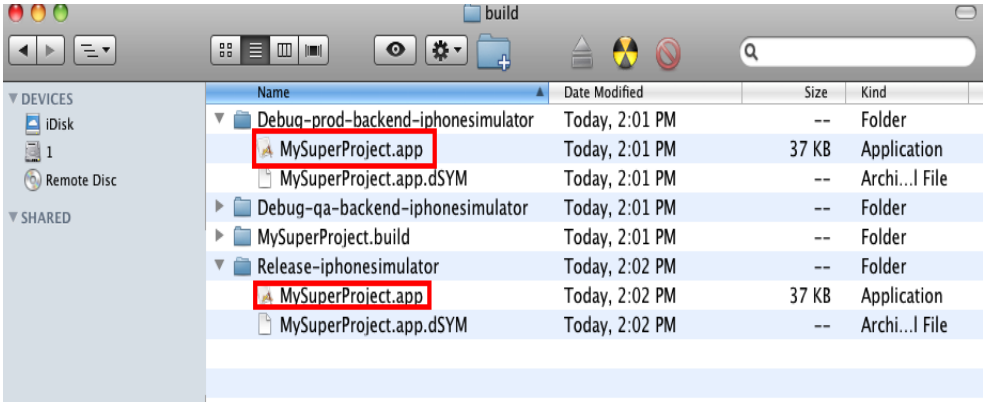
Changes in the build system of XCode 4.0.2

Traditional build directory anatomy

Before xCode 4.0.2 the “build” directory was created at the project directory (at the same directory as ***MySuperProject.xcodeproj*** in our case).



The **build** directory contained some directories, named in the same way as the targets. And those target-named subdirectories contained built apps.



New build directory anatomy

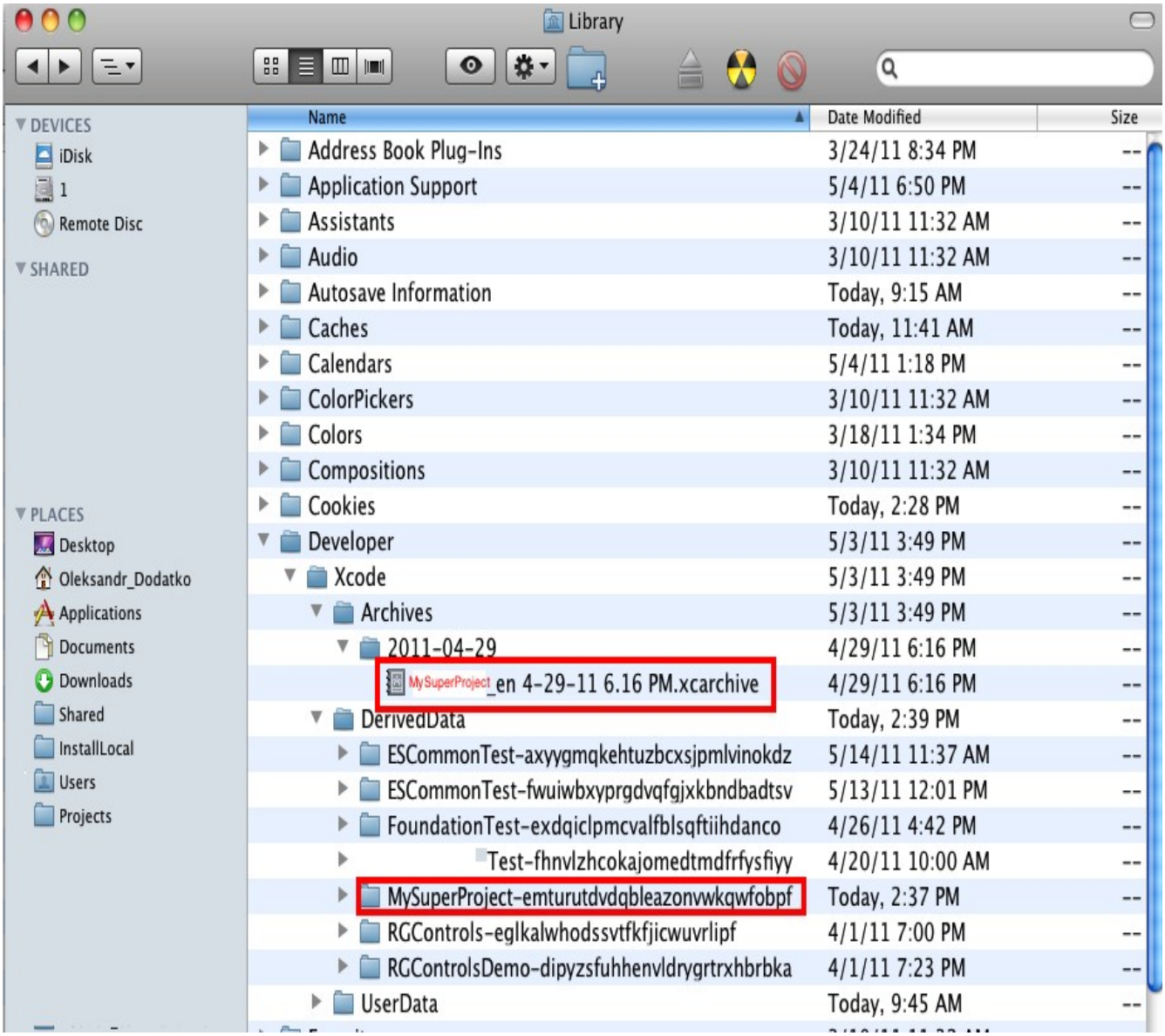
Since this version (xCode4.0.2) build directory is no more created at the common location. Now build results of all projects are stored at **~/Library/Developer/Xcode/DerivedData**

```
:~ Oleksandr_Dodatko$ cd ~/Library/Developer/Xcode/DerivedData
epuadniw0063:DerivedData Oleksandr_Dodatko$ ls
FoundationTest-exdqiclpmcvalfblsqftiihdanco
MySuperProjectTest-fhnvlzhcokajomedtmdfrfysfiyy
MySuperProject-efyflrltwszrpeecwijzbewsbpuy
RGControls-eglkalwhodssvtfkfjicwuvrlipf
RGControlsDemo-dipyzsfuhhenvldrygrtrxhbrbka
```

The project name is followed by the GUID identifier in order to distinguish projects with the same name in different locations. In order to determine the location of your particular project, you can use the compiler output of the xCode GUI.

```
CompileC /users/Oleksandr_Dodatko/Library/Developer/Xcode/DerivedData/MySuperProject-
efyflrltwszrpeecwijzbewsbpuy/Build/Intermediates/MySuperProject.build/Release-
iphonesimulator/MySuperProject_en.build/Objects-normal/i386/main.o main.m normal i386 objective-c
com.apple.compilers.gcc.4_2
…….....
…….....
…….....
```

Build directory has somewhat different structure comparing with the one from previous versions. It contains some intermediate build data and logs as well. However, most of them are not that much human readable.



The other directory, which may be interesting for continuous integration, is **Archives**. It stores the archives, made with the help of the XCode GUI (the Organizer).

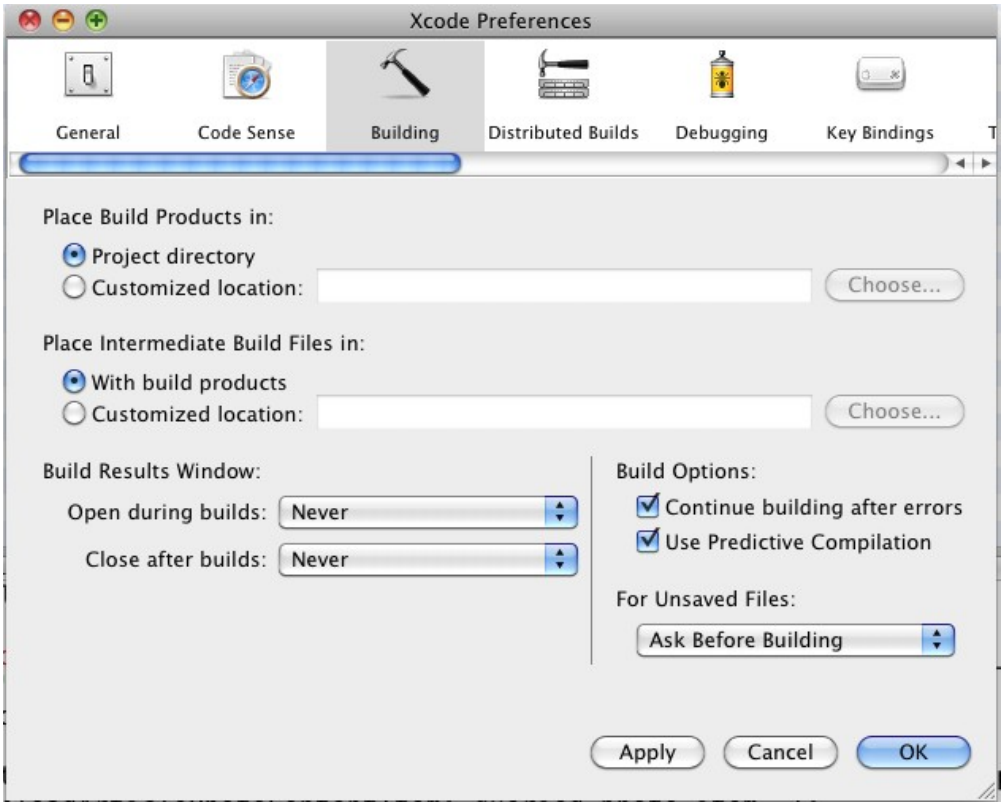
So, the final path for our app bundle will be

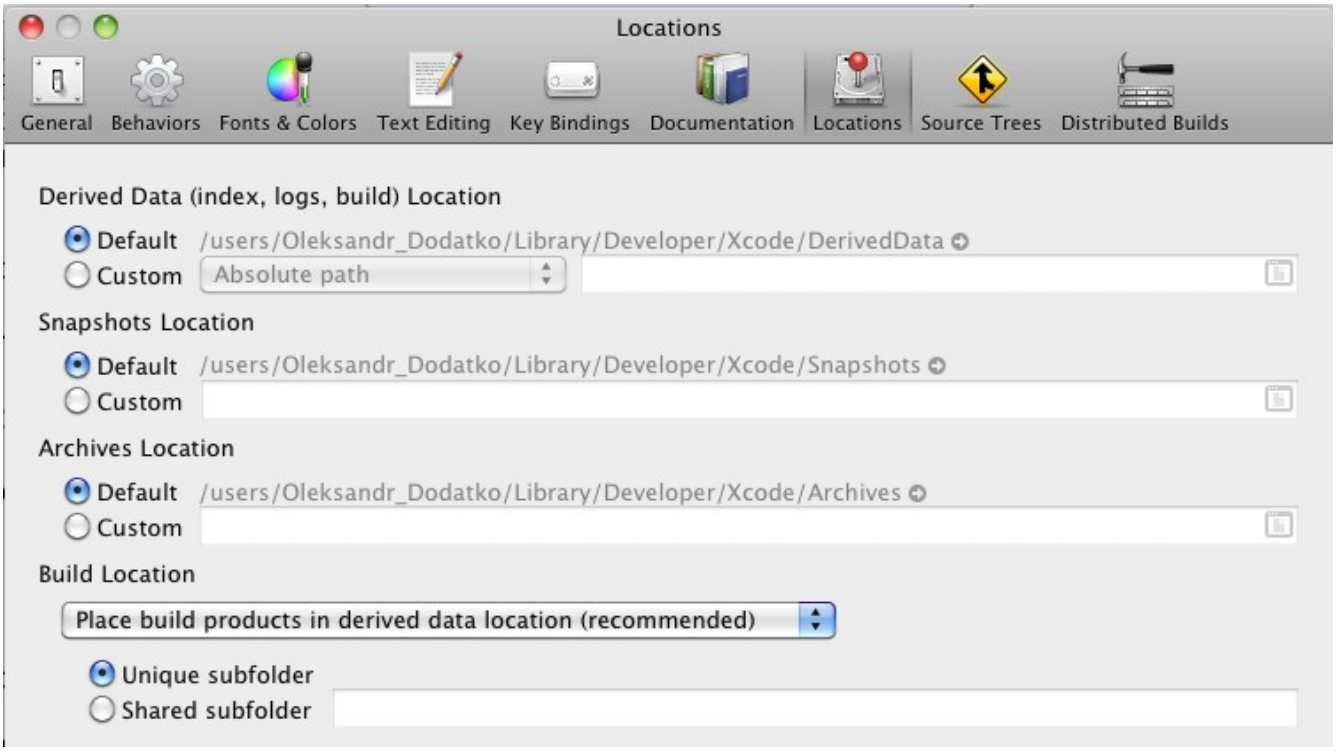
```
~/Library/Developer/Xcode/DerivedData/MySuperProject-efyflrltwszrpeecwijzbewsbpuy/Build/Products/Debug-qa-backend-iphonesimulator
```

Note : **iphonesim** and **xcrun** tools will still work correctly with those *.app bundles.

Build directories settings

However, those build locations can be changed within the XCode. In order to do it, go to the XCode preferences





So, you can change the build directory to be in the same place as before (beneath the *.xcodeproj directory). Still, the build directory structure has changed since Xcode4. So, integration scripts that rely on “build” directory structure will still be broken.

XCode variables

However, those settings can be retrieved during the XCode build. The full list of the XCode variables can be found at the Apple Documentation. [8]

You can dump all of them by adding a “Run script” step to your target. Here is an example for the mentioned options :

```
# dumps xcode variables
DUMP_FILE=~/XCODE_VARIABLES.txt
rm $DUMP_FILE

echo TARGET_BUILD_DIR        -- $TARGET_BUILD_DIR >> $DUMP_FILE
echo BUILT_PRODUCTS_DIR      -- $BUILT_PRODUCTS_DIR >> $DUMP_FILE
echo PROJECT_DIR             -- $PROJECT_DIR >> $DUMP_FILE
echo CONFIGURATION_BUILD_DIR -- $CONFIGURATION_BUILD_DIR >> $DUMP_FILE
echo INSTALL_DIR             -- $INSTALL_DIR >> $DUMP_FILE
echo OBJROOT                 -- $OBJROOT >> $DUMP_FILE

exit 0
```

Xcode3 default output

```
TARGET_BUILD_DIR --
/Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest/build/Debug-
iphonesimulator
BUILT_PRODUCTS_DIR --
/Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest/build/Debug-
iphonesimulator
PROJECT_DIR -- /Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot test/ESCommonTest
CONFIGURATION_BUILD_DIR -- /Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot
test/ESCommonTest/build/Debug-iphonesimulator
INSTALL_DIR -- /tmp/ESCommonTest.dst/users/Oleksandr_Dodatko/Applications
OBJROOT -- /Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest/build
```

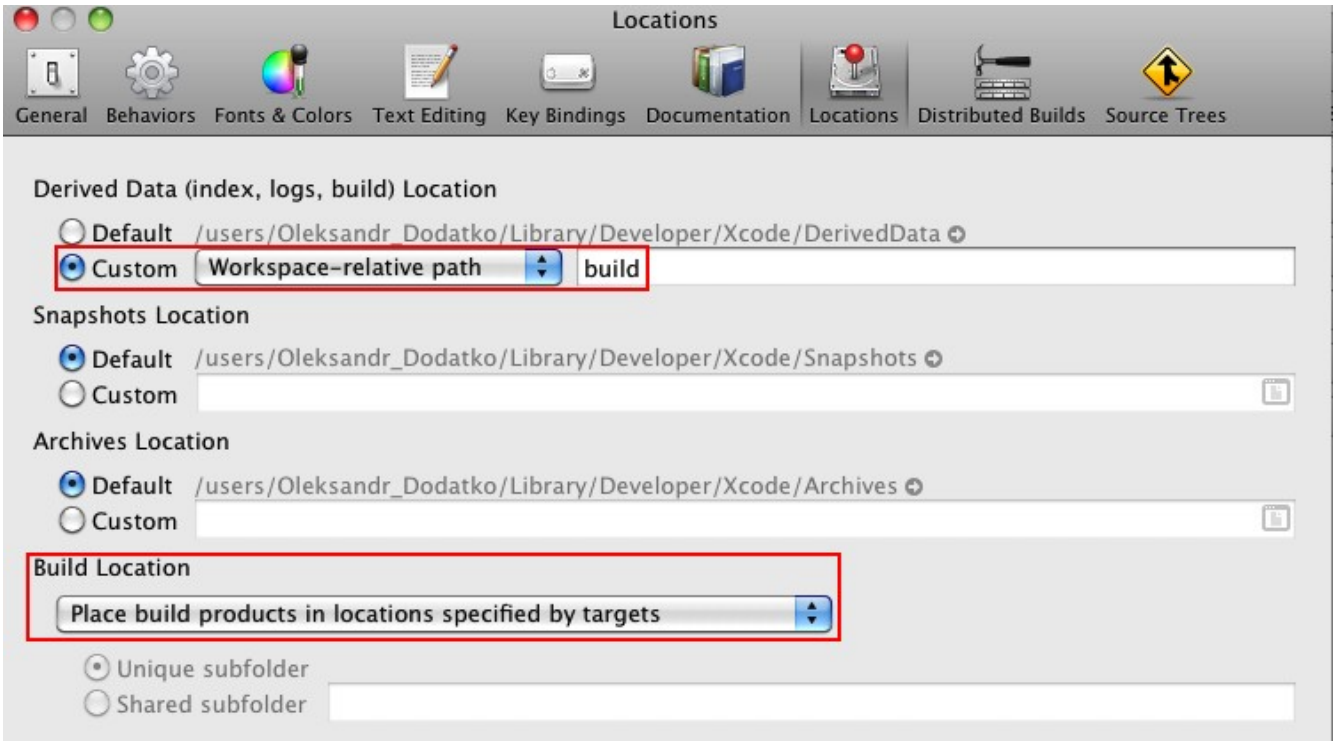
Xcode4 default output

```
TARGET_BUILD_DIR -- /users/Oleksandr_Dodatko/Library/Developer/Xcode/DerivedData/ESCommonTest-axyygmqkehtuzbcxsjpmlvinokdz/Build/Products/Debug-iphonesimulator

BUILT_PRODUCTS_DIR -- /users/Oleksandr_Dodatko/Library/Developer/Xcode/DerivedData/ESCommonTest-axyygmqkehtuzbcxsjpmlvinokdz/Build/Products/Debug-iphonesimulator

PROJECT_DIR -- /Users/Oleksandr_Dodatko/Projects/ThomsonReuters/ReutersInsider/iPhone/trunk/test/ESCommonTest

CONFIGURATION_BUILD_DIR -- /users/Oleksandr_Dodatko/Library/Developer/Xcode/DerivedData/ESCommonTest-axyygmqkehtuzbcxsjpmlvinokdz/Build/Products/Debug-iphonesimulator

INSTALL_DIR -- /tmp/ESCommonTest.dst/Applications

OBJROOT -- /users/Oleksandr_Dodatko/Library/Developer/Xcode/DerivedData/ESCommonTest-axyygmqkehtuzbcxsjpmlvinokdz/Build/Intermediates
```

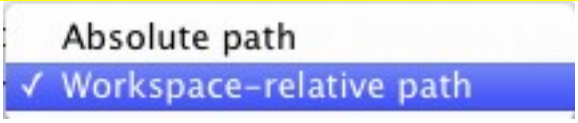
Continuous integration considerations

As the result of this research, I can suggest you using XCode variables to access your built applications and not rely on the directories structure as it's subject to change with new releases of the XCode.

However, if you have some legacy scripts that rely on the build directory structure, you have to configure XCode to make them work.



The upper settings stands for the build directory location.
We should choose "**workspace-relative path**"->"**build**" to preserve the Xcode3 behavior.
However, it can be a good solution to use custom path and rely on it in your scripts regardless of default path entries proposed by Apple.



The lower settings stand for the build directory structure. Xcode4 supports two modes :

In order to achieve legacy behavior, we need to select "**locations specified by targets**" option.

Now that we have set up XCode, we can launch the script and check produced variables.

Xcode4 legacy variables

```
TARGET_BUILD_DIR --
/Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest/build/Debug-
iphonesimulator
BUILT_PRODUCTS_DIR --
/Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest/build/Debug-
iphonesimulator
PROJECT_DIR -- /Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest
CONFIGURATION_BUILD_DIR --
/Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest/build/Debug-
iphonesimulator
INSTALL_DIR -- /tmp/ESCommonTest.dst/Applications
OBJROOT -- /Users/Oleksandr_Dodatko/Projects/MySuperProjectRoot/test/ESCommonTest/build
```

As you can see, they are identical to those from Xcode3. So, you can still use your old scripts.

However, it is STRONGLY RECOMMENDED to rewrite the scripts as Apple may remove these settings in future releases of the XCode.

Bibliography

1: , , , http://www.manpagez.com/man/1/xcodebuild/
2: Jeff Haynie, , , git://github.com/jhaynie/iphonesim.git
3: , , ,
http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/iphone_development/145-Distributing_Applications/distributing_applications.html
4: , , ,
http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/iphone_development/128-Managing_Devices_and_Digital_Identities/devices_and_identities.html
5: , , , http://nachbaur.com/blog/how-to-automate-your-iphone-app-builds-with-hudson
6: , , , http://blog.octo.com/en/automating-over-the-air-deployment-for-iphone/
7: , , , http://stackoverflow.com/questions/2664885/xcode-build-and-archive-from-command-line
8: Apple. Inc, Xcode Build Setting Reference, 2010,
http://developer.apple.com/library/mac/#documentation/DeveloperTools/Reference/XcodeBuildSettingRef/1-Build_Setting_Reference/build_setting_ref.html