BUILD FRAMEWORK iOS DOCUMENTATION

January 14, 2015

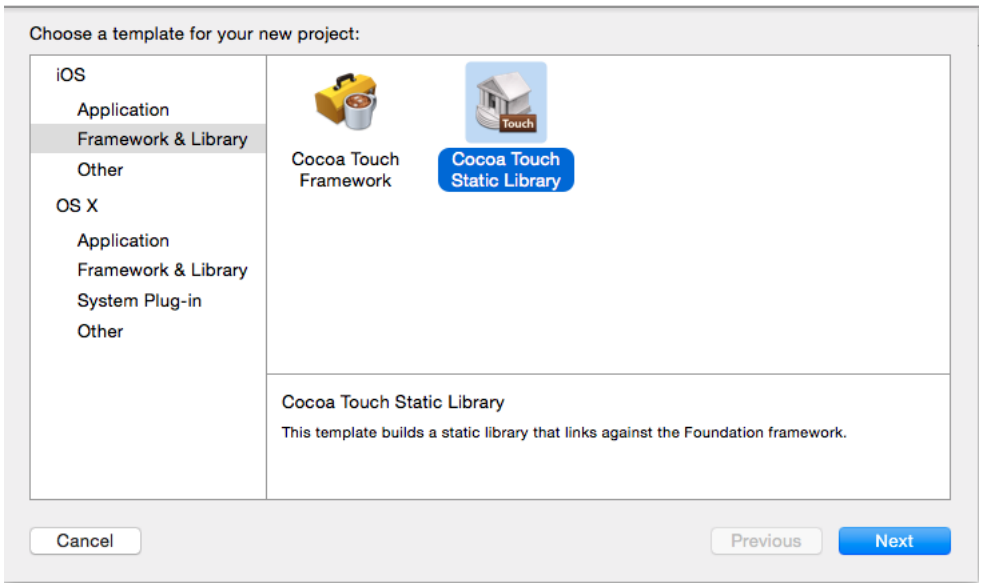
Contents

Chapter 1

How to build LooketAppsSDK

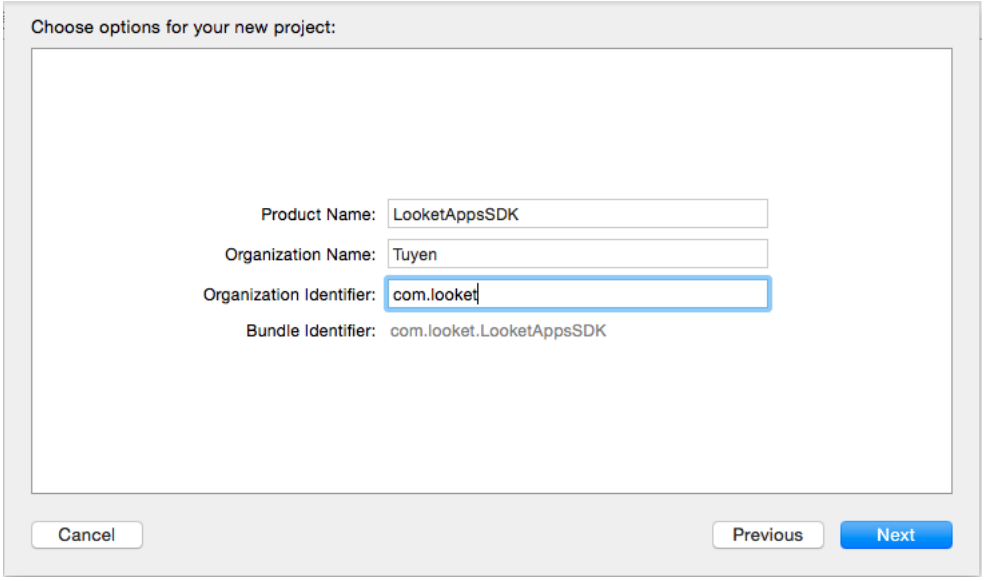
1.1 Create library project in XCode

- Open XCode



- Create a **Cocoa Touch Static Library** by clicking **File** > **New** > **Project**
- Under the iOS group select **Framework & Library** then select **Cocoa Touch Static Library**
- Click **Next**

- Change where the project will be saved and give it the name **LooketAppsSDK**

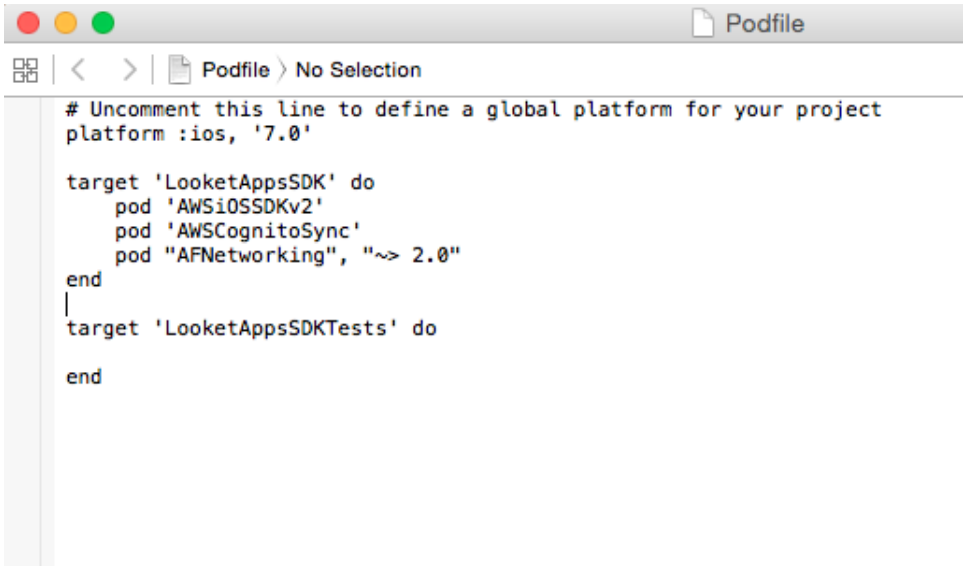


- Change the values of the other fields if necessary
- Then click **Create**

1.2 Install dependent library using CocoaPod

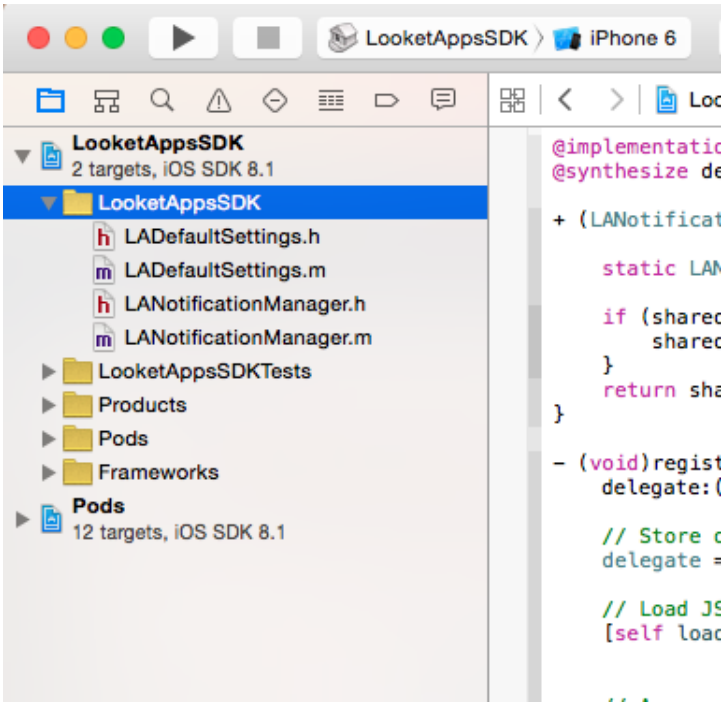
- Open **Terminal**
- **cd** to folder of your project.
- Type command "**pod init**" to create **Podfile** file
- Open **Podfile** file using command "**open -a Xcode Podfile**"
- After **Podfile** file is opened, enter the name of dependent libraries into this file.

- Install dependent library using command "**pod install**"
- After finished installation, open project by clicking **LooketAppsSDK.xcworkspace** file



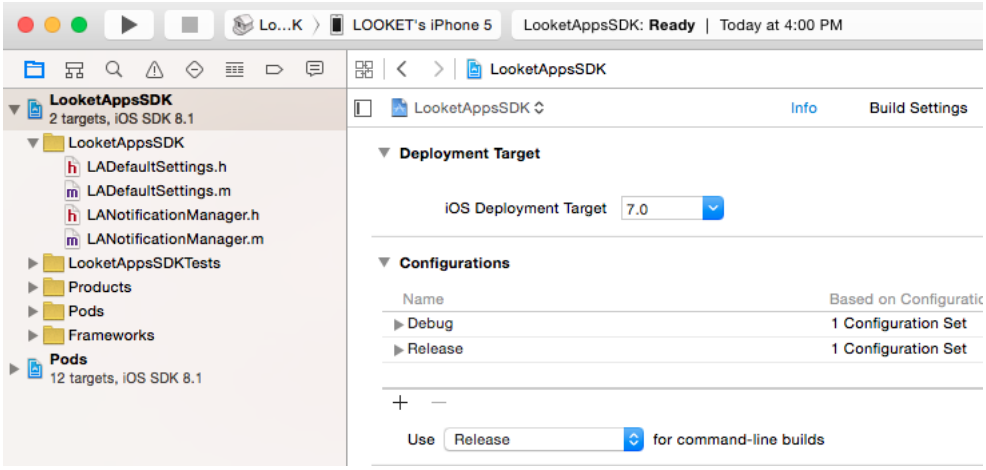
1.3 Write code to framework

- Write code to project. There have 4 files: **LADefaultSettings.h, LADefaultSettings.m, LANotificationManager.h, LANotificationManager.m**

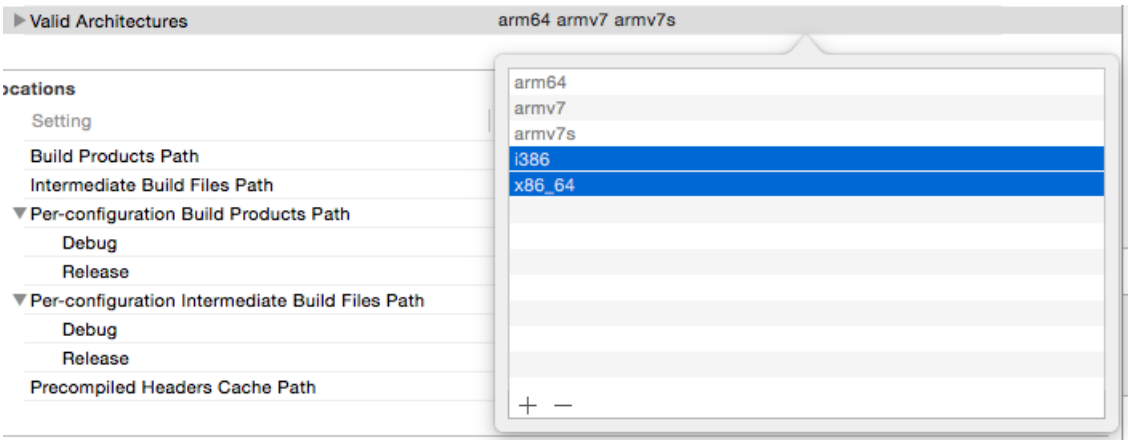


1.4 Configure Project

- iOS deployment target is **7.0**

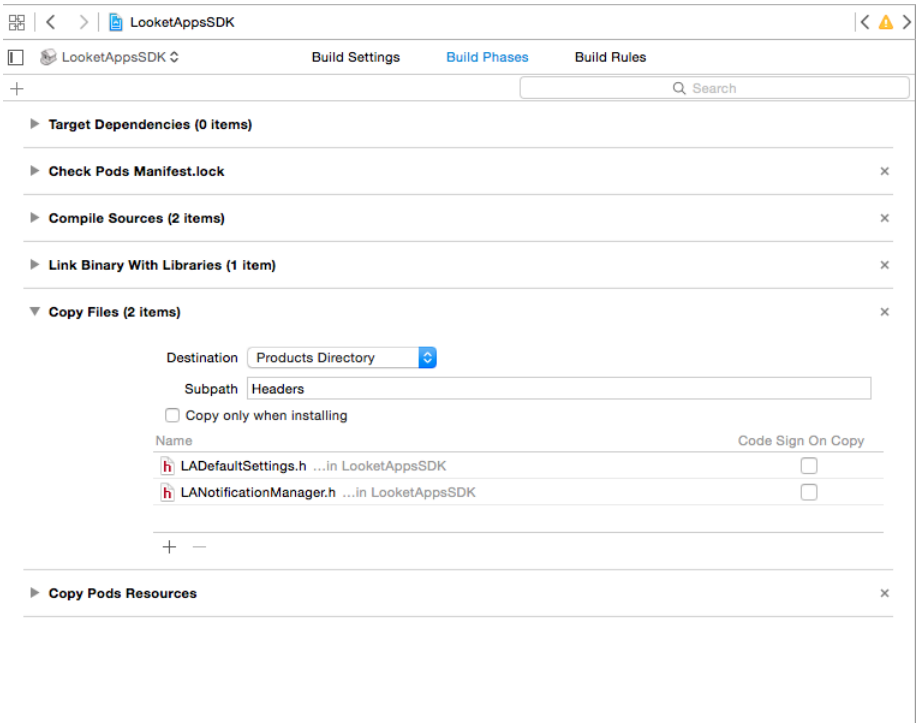


- Add *i*386 and *x*86_64 to "**Valid Architecture**". This configuration allows framework to work on many environments (real devices and simulators).



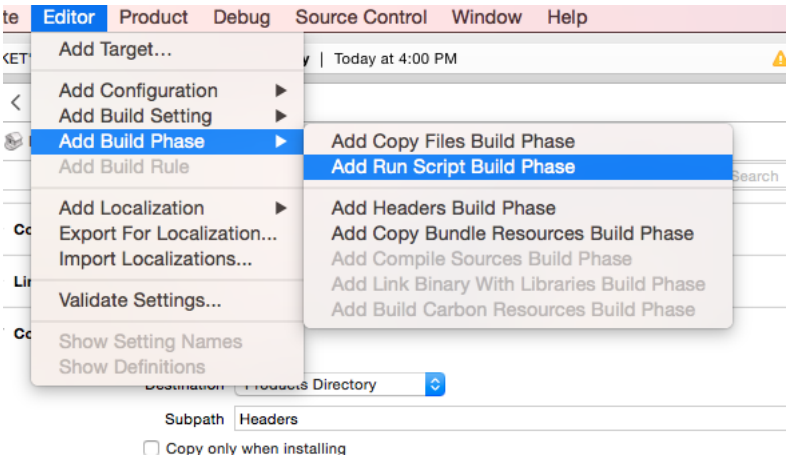
- Select target **LooketAppsSDK** and Click on **Build Phases** and then expand **Copy Files**

- In the field **Subpath**, rename to **Headers**

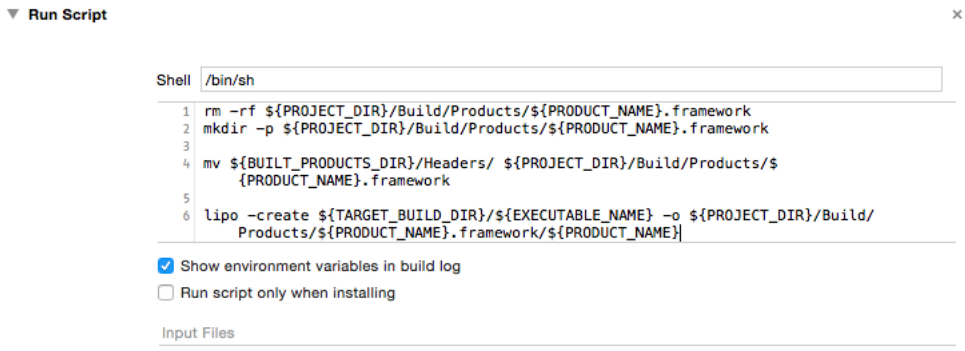


- Click + symbol and add header files. See picture above:

- While still looking at the **Build Phases** screen in Xcode, choose **Editor > Add Build Phase > Add Run Script Build Phase**

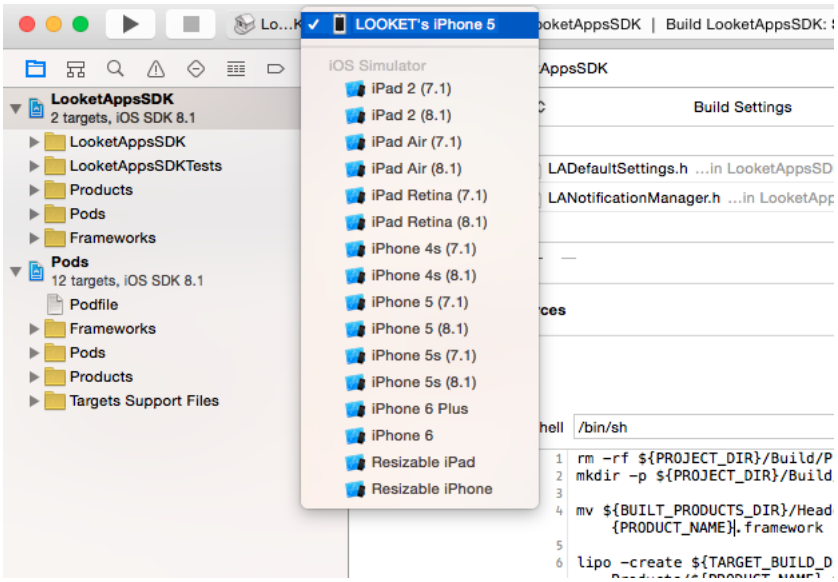


- Expand **Run Script**, add script as picture below:

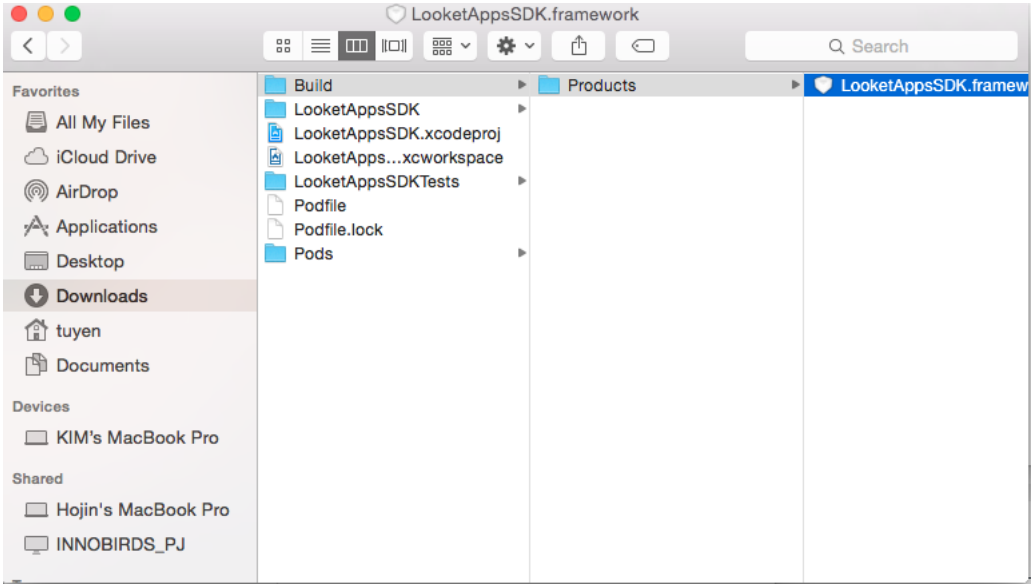


1.5 Build framework

- Before building project, select **iOS Device**. This step allows framework to work on real device.



- Build project by **Product** > **Build**
- After building, framework will appear at folder $PROJECT\_DIR/Build/Products/$

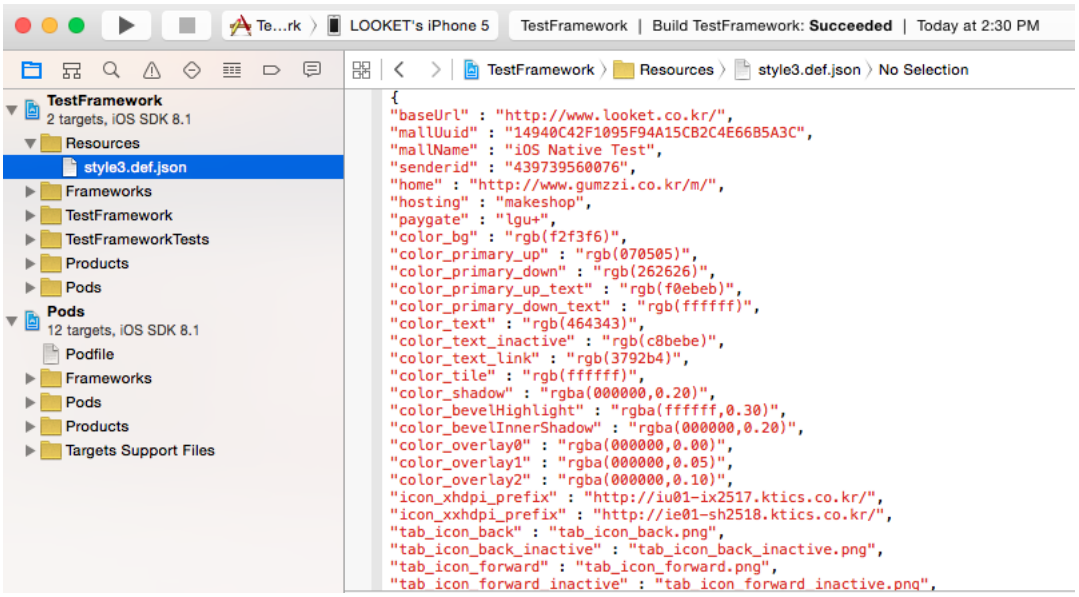

LooketAppsSDK.framework
Search
Favorites
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Downloads
tuyen
Documents
Devices
KIM's MacBook Pro
Shared
Hojin's MacBook Pro
INNOBIRDS_PJ
Build
LooketAppsSDK
LooketAppsSDK.xcodeproj
LooketApps...xcworkspace
LooketAppsSDKTests
Podfile
Podfile.lock
Pods
Products
LooketAppsSDK.framew

Chapter 2

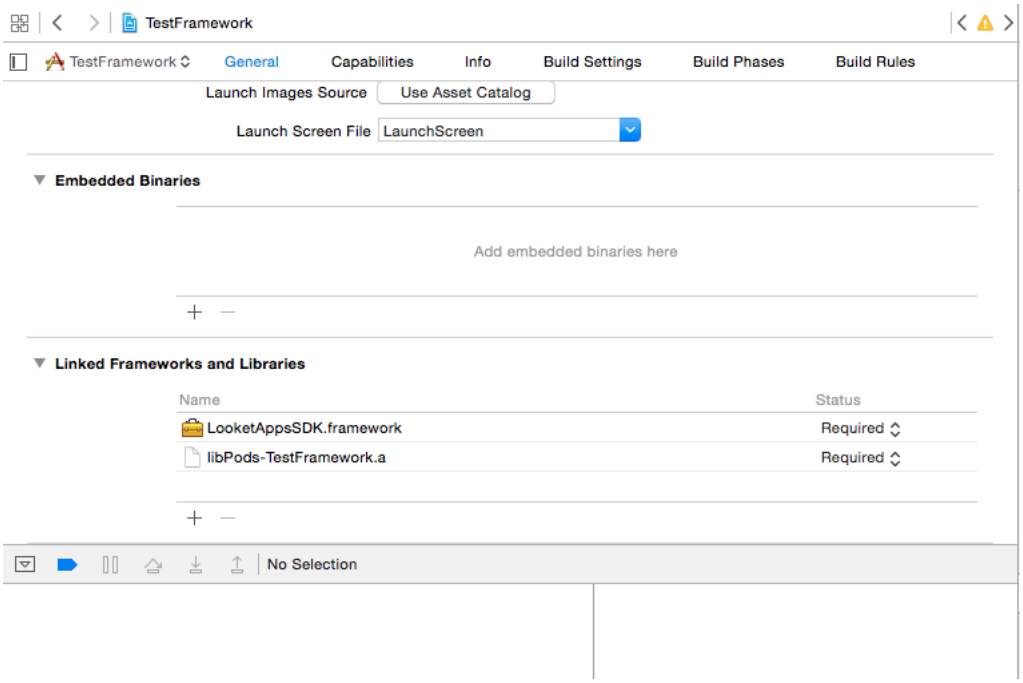
How to use LooketAppsSDK framework

2.1 Configurations on custom application

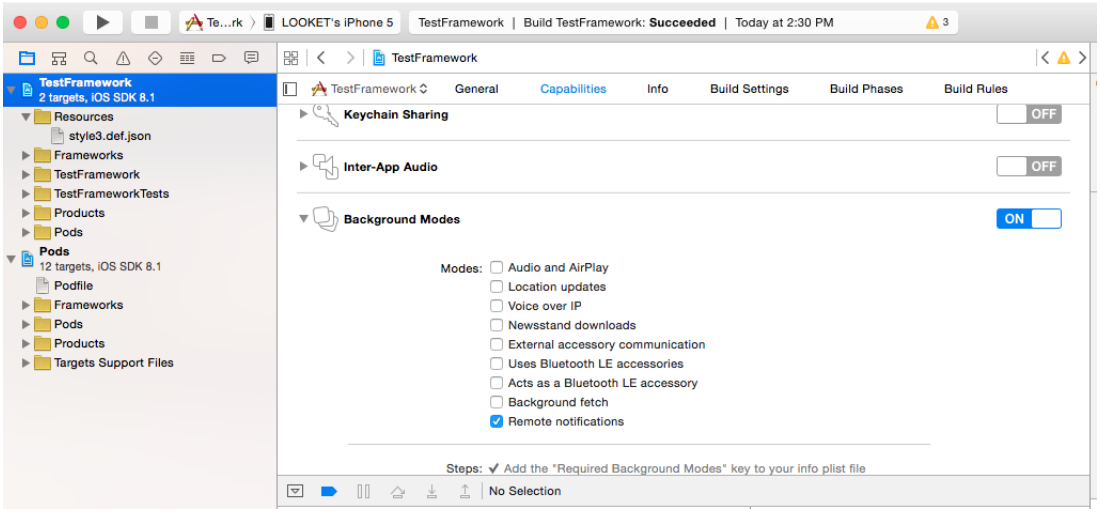
- Create a project called **TestFramework**
- Add resource file **style3.def.json**



- Add **LooketAppsSDK.framework** into project



- In **capabilities** page on project target, turn on **Background Modes** and select **Remote Notification**. This step allows custom application to handle received notification in background.



2.2 Install dependent library using CocoaPod

- We also need to install dependent library using cocoapod as instruction at section 1.2

2.3 Using LooketAppsSDK API in custom applications

In **Appdelegate.h** file:

- Include header files of SDK

```
#import <LooketAppsSDK/LANotificationManager.h>
#import <LooketAppsSDK/LADefaultSettings.h>
```

- Inherit protocol **LANotificationManagerDelegate**

```
@interface AppDelegate : UIResponder <UIApplicationDelegate,
    LANotificationManagerDelegate>

@property (strong, nonatomic) UIWindow *window;
@property (strong, nonatomic) ViewController *viewController;

@end
```

In **AppDelegate.m** file:

- In the callback function **application:didFinishLaunchingWithOptions:**, call API to register remote notification as bellow:

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(
    NSDictionary *)launchOptions {
    [[LANotificationManager sharedNotificationManager] registerRemoteNotification:
        application withOptions:launchOptions delegate:self];

    return YES;
}
```

- In the callback function **application: didRegisterForRemoteNotificationsWithDeviceToken**, call API to add device to server as bellow:

```
- (void)application:(UIApplication *)application
    didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    [[LANotificationManager sharedNotificationManager] addUserToServerWithDeviceToken:
        deviceToken];
}
```

- In the callback function **application:didReceiveRemoteNotification:**, call API to receive notification as bellow:

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(
    NSDictionary *)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))
    completionHandler {

    [[LANotificationManager sharedNotificationManager] didReceiveRemoteNotification:
        application userInfo:userInfo];

    //Tell the system that you're done.
    completionHandler(UIBackgroundFetchResultNewData);
}
```

- Besides, customer application can also implements Delegate functions:

```
- (void)handleOpenNotificationFromActive:(UIApplication *)application userInfo:(
    NSDictionary *)userInfo;
- (void)handleOpenNotificationFromInactive:(UIApplication *)application userInfo:(
    NSDictionary *)userInfo;
- (void)handleResponseObjectFromServer:(id)responseObject;
```