使用CLI編譯XCode Project

編譯ipa檔案
自動上傳至TestFlight

Author: BY

v 0.0.1

基本環境

- 安裝Xcode (本篇實作環境為 Xcode 8)
- 安裝cocoapods
- 安裝git
- 使用到的command library有
  - xcodebuild => Build xcode and ipa
  - altool => 上傳至TestFlight
  - security => sign certificate

設定CERTIFICATE和PROVISIONING PROFILE

- 至Apple Developer Console 下載
  - Certificate for Distribution
  - Provisioning Profile for Distribution
- 執行 「security find-identity -p codesigning -v」查看validation certificate

✳ 注意

✳ Certificat

IMPORT CERTIFICATE

- 使用CLI匯入新的Certifiate檔案

- security import \
  <Certificate Path> \
  -k <defualt login keychain> \
  -P <Certrificate password> \
  -T /usr/bin/codesign

```
chenboyude-MBP:Keychains chenboyu$ security import \
> ~/Download/certificate.p12 \
> -k ~/Library/keychains/login.keychain \
> -P 1234 \
> -T /usr/bin/codesign
```

執行 PROVISIONING PROFILE

- 點雙擊 「.mobileprovision」就會完成匯入步驟
- Provisioning Profile位置會在「~/Library/MobileDevice/Provisioning Profiles/」
- Provisioning Profile的檔案名稱，就是所屬的UUID

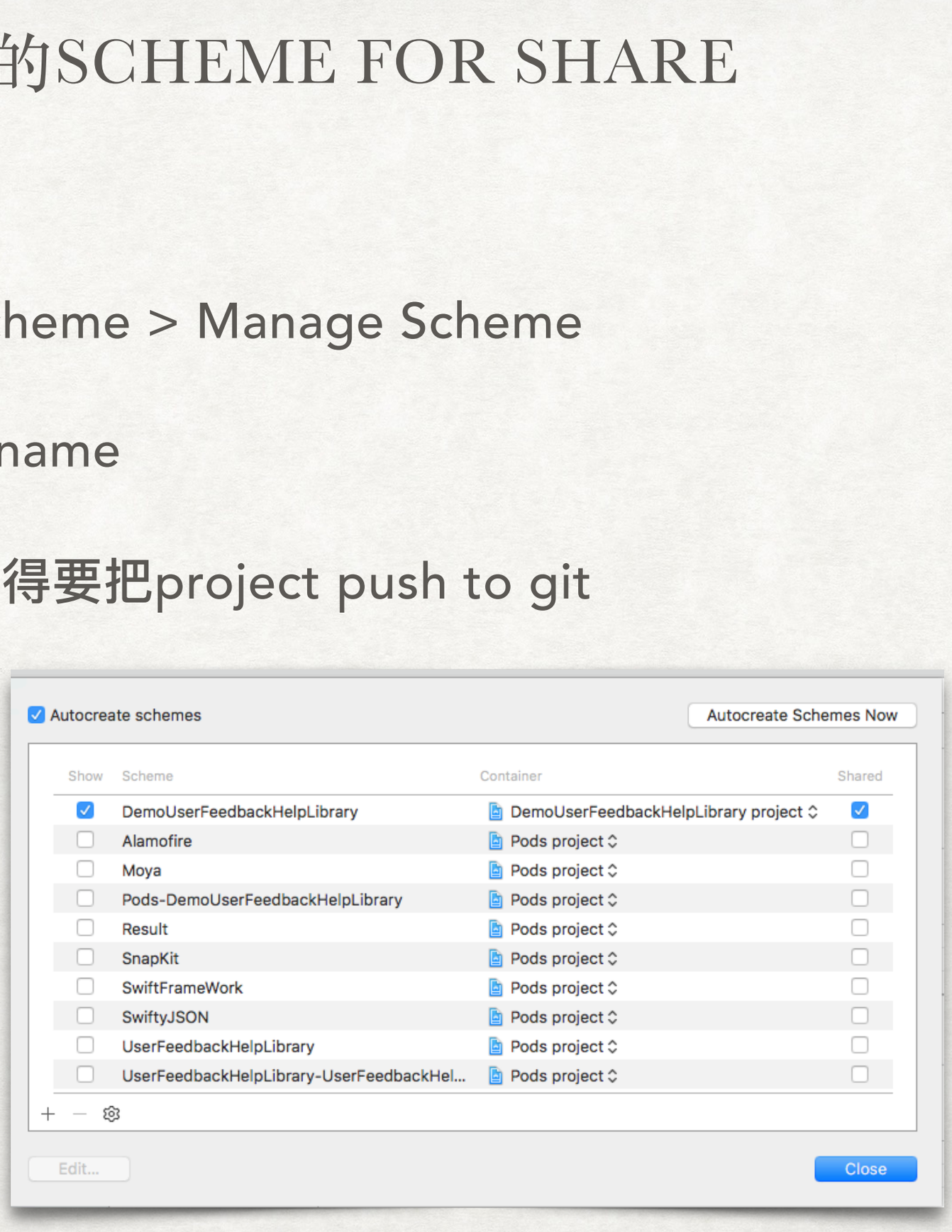
```
chenboyude-MBP:Provisioning Profiles chenboyu$ pwd
/Users/chenboyu/Library/MobileDevice/Provisioning Profiles
chenboyude-MBP:Provisioning Profiles chenboyu$ ls
364cd99d-5f76-4a85-84f5-94e928ae00c1.mobileprovision
4dccc985-5639-4339-a5e5-365299cae09e.mobileprovision
c0b19354-392c-4112-b14c-483e6f6b409d.mobileprovision
e0b9147a-f369-4776-ba79-4171a70705e8.mobileprovision
fe70da67-722b-4f9d-84d7-5de64a98f842.mobileprovision
chenboyude-MBP:Provisioning Profiles chenboyu$
```

查看 PROVISIONING PROFILE

- 執行「security cms -D -i <provisioning profile.mobileprovision>」
- 可以查看到
  - TeamID
  - AppIDName
  - Application Identifier =>包含Bundle name

設定PROJECT的SCHEME FOR SHARE

- Open Xcode with project
- In tool bar > Product > Scheme > Manage Scheme
- Check Shared for project name
- 如果要完整的自動測試，記得要把project push to git



USE CLI TO BUILD PROJECT

- 必要command 「xcodebuild」
  - 指定 Workspace 「-workspace <AppName.xcworkspace>」
  - 指定 Project 「-project <AppName.xcodeproj>」
  - Workspace 和Project 只能選其中一個
  - 指定Scheme 「-scheme AppName」通常為AppName
  - 指定Configuration 「-configuration Release/Debug」
  - Build actions
    - Clean : 清除專案
    - Build : 編譯
    - Archive : 匯出ipa

查看PROJECT環境

- 移動到Project底下，執行「xcodebuild -list」
- 可以查看 AppName, Targets, Schemes, Configurations

```
chenboyude-MBP:DemoUserFeedbackHelpLibrary chenboyu$ xcodebuild -list
Information about project "DemoUserFeedbackHelpLibrary":
    Targets:
        DemoUserFeedbackHelpLibrary

    Build Configurations:
        Debug
        Release

    If no build configuration is specified and -scheme is not passed then "Release" is used.

    Schemes:
        DemoUserFeedbackHelpLibrary

chenboyude-MBP:DemoUserFeedbackHelpLibrary chenboyu$
```

取得專案的TEAM ID

- 查訊此專案的Team ID，執行

  「xcodebuild -workspace <.xcworkspace> \
  -scheme <AppName> \
  -showBuildSettings | grep 'DEVELOPMENT_TEAM'」

```
chenboyude-MBP:DemoUserFeedbackHelpLibrary chenboyu$ xcodebuild \
> -workspace DemoUserFeedbackHelpLibrary.xcworkspace \
> -scheme DemoUserFeedbackHelpLibrary \
> -showBuildSettings | grep 'DEVELOPMENT_TEAM'
    DEVELOPMENT_TEAM = 483VY7RC75
chenboyude-MBP:DemoUserFeedbackHelpLibrary chenboyu$
```

執行BUILD PROJECT

- Build Project且順帶執行Clean指令
- 需要帶入參數
  - CODE_SIGN_IDENTITY
  - PROVISIONING_PROFILE_SPECIFIER
- 執行

「xcodebuild -workspace <.xcworkspace> \
-scheme <AppName> \
-configuration <ReleaselDebug> \
clean build \
CODE_SIGN_IDENTITY= "< 填入Certificate的名稱>" \
PROVISIONING_PROFILE_SPECIFER="<填入Provisioning Profile 檔案名稱>"
」

```
chenboyude-MBP:~ chenboyu$ xcodebuild \
> -workspace UserFeedbackHelpLibrary.xcworksspace \
> -scheme UserFeedbackHelpLibrary \
> -configuration Release \
> clean build \
> CODE_SIGN_IDENTITY="iPhone Developer: Chien LinFu (AH9D5P56KS)" \
> PROVISIONING_PROFILE_SPECIFER="4dccc985-5639-4339-a5e5-365299cae09e"
```

BUILD ARCHIVE

- 在Build ipa之前需要先build出archive檔案
- 大致上跟build project一樣的command
  - archivePath：可以設定輸出.xcarchive路徑
- 執行

「xcodebuild -workspace <.xcworkspace> \
-scheme <AppName> \
-archivePath <OutputPath>/<Name>.xcarchive
-configuration <ReleaselDebug> \
clean archive \
CODE_SIGN_IDENTITY= "< 填入Certificate的名稱>" \
PROVISIONING_PROFILE_SPECIFER="<填入Provisioning Profile 檔案名稱>"
」

```
chenboyude-MBP:~ chenboyu$ xcodebuild \
> -workspace UserFeedbackHelpLibrary.xcworkspace \
> -archivePath ~/Desktop/UserFeedbackHelpLibrary/build/UserFeedbackHelpLibrary.xcarchive \
> -scheme UserFeedbackHelpLibrary \
> -configuration Release \
> clan archive \
> CODE_SIGN_IDENTITY="iPhone Developer: Chien LinFu (AH9D5P56KS)" \
> PROVISIONING_PROFILE_SPECIFER="4dccc985-5639-4339-a5e5-365299cae09e"
```

EXPORT IPA：OPTION PLIST FILE

- Build ipa的檔案前，需要先創建一個export 環境的plist檔案
- plist檔案裡面要包含兩個Key：
  - method類型有：app-store, ad-hoc, enterprise, or development
  - teamID：可以由Certificate 取得 or 使用xcodebuild 取得

```
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
                "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
    <dict>
            <key>method</key>
            <string>${METHOD}</string>
            <key>teamID</key>
            <string>${DEVELOPMENT_TEAM}</string>
    </dict>
</plist>
```

<exportOptions>.plist

BUILD ipa

- 產生ipa時候需要用到 xcarchive 和 option plist 檔案，才能夠Build spa
- 用到參數有
  - archivePath ：xcarchive位置
  - exportOptionsPlist：option plist 位置
  - exportPath：ipa輸出的位置
- 執行

「xcodebuild -exportArchive \
-archivePath <OutputPath>/<Name>.xcarchive \
-exportOptionsPlist <option>.plist \
-exportPath <export ipa path>/」

```
chenboyude-MBP:~ chenboyu$ xcodebuild -exportArchive \
> -archivePath UserFeedbackHelpLibrary/build/UserFeedbackHelpLibrary.xcarchive \
> -exportOptionsPlist UserFeedbackHelpLibrary/build/option.plist \
> -exportPath UserFeedbackHelpLibrary/build/
```

SETTING UPLOAD TO APPLE TestFlight

- 上傳至Apple TestFlight會用到兩個檔案
  - altool：Application Loader Tool
  - iTMSTransporter：altool裡面會使用
- altool路徑會在
- 「/Applications/Xcode.app/Contents/Applications/Application\ Loader.app/Contents/Frameworks/ITunesSoftwareService.framework/Versions/A/Support/altool」
- iTMSTransporter路徑資料夾會在
- 「/Applications/Xcode.app/Contents/Applications/Application\ Loader.app/Contents/itms」

SETTING UPLOAD TO APPLE TestFlight

- 為了在任何路徑底下都可以使用，所以進行檔案連結
- 執行
- 「sudo -S ln -s <altool path> /usr/local/bin」
- 「sudo -S ln -s <itms path> /usr/local」

UPLOAD TO APPLE TestFlight

- 上傳至Apple TestFlight需要Apple的帳號密碼
- 密碼可以使用keychain製作，本範例不使用這種方法，而是使用較不安全的明碼密碼
- 「altool --upload-app \
  -f <ipa path>.ipa \
  -u <account> \
  -p <password>」

```
chenboyude-MBP:itms chenboyu$ altool --upload-app \
> -f UserFeedbackHelpLibrary/build/UserFeedbackHelpLibrary.ipa \
> -u abce \
> -p 1234
```

SECURITY UNLOCK KEYCHAIN

- 在build情況下都會需要授權Security，當使用CLI去執行所有過程時，就要進行Unlock keychain的命令
- 通常預設的keychain路徑會在「~/Library/Keychains/login.keychain」
- 執行
- security set-key-partition-list -S apple-tool:,apple: -s -k <System passwod> <default login keychain>
- security default-keychain -d user -s <default login keychain>
- security unlock-keychain -p <System passwod> <default login keychain>

把自動載入PROVISIONING PROFILE改成手動

- 由於Xcode 8使用Automatically manage singing，但在全自動的環境底下，許要把這個更改為Manual
- 可以使用以下指令去更改
- sed -i '' 's/ProvisioningStyle = Automatic;/ProvisioningStyle = Manual;/' <project>.xcodeproj/project.pbxproj

```
};
8BCC97B71BC38461003093AC = {
    CreatedOnToolsVersion = 7.0.1;
    DevelopmentTeam = 483VY7RC75;
    LastSwiftMigration = 0800;
    ProvisioningStyle = Automatic;
    SystemCapabilities = {
        com.apple.Keychain = {
            enabled = 1;
        };
        com.apple.Push = {
            enabled = 1;
        };
    };
};
```

```
};
8BCC97B71BC38461003093AC = {
    CreatedOnToolsVersion = 7.0.1;
    DevelopmentTeam = 483VY7RC75;
    LastSwiftMigration = 0800;
    ProvisioningStyle = Manual;
    SystemCapabilities = {
        com.apple.Keychain = {
            enabled = 1;
        };
        com.apple.Push = {
            enabled = 1;
        };
    };
};
```