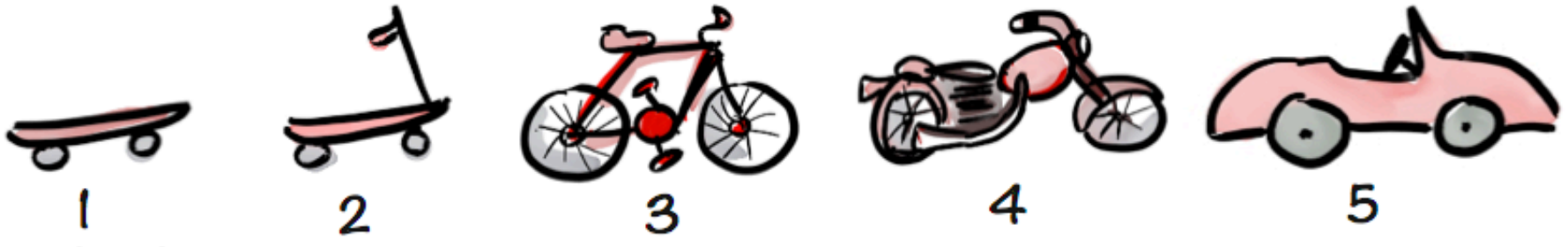
iOS App Creation



Ryan Chung

課程大綱

- 多國語言支援
  - 翻譯文字 NSLocalizedString
- APP通知
  - 使用者通知 User Notification
  - 10秒後觸發 Time-based Trigger
  - 圖文通知 NotificationAttachment
- 清單過濾與排序
  - 課程過濾 ForEach Filter
  - 課程排序 Sort $0>$1
  - 滑動刪除 onDelete
  - 互動選單 contextMenu
- @的使用與資料流動
- APP實機測試
  - 開發者帳號申請
  - 相關設定
- APP上架準備事項
  - 測試注意事項
  - 上架用圖片 Icon
- APP審核準則
  - 寫在前面
  - 安全
  - 效能
  - 設計
  - 法律

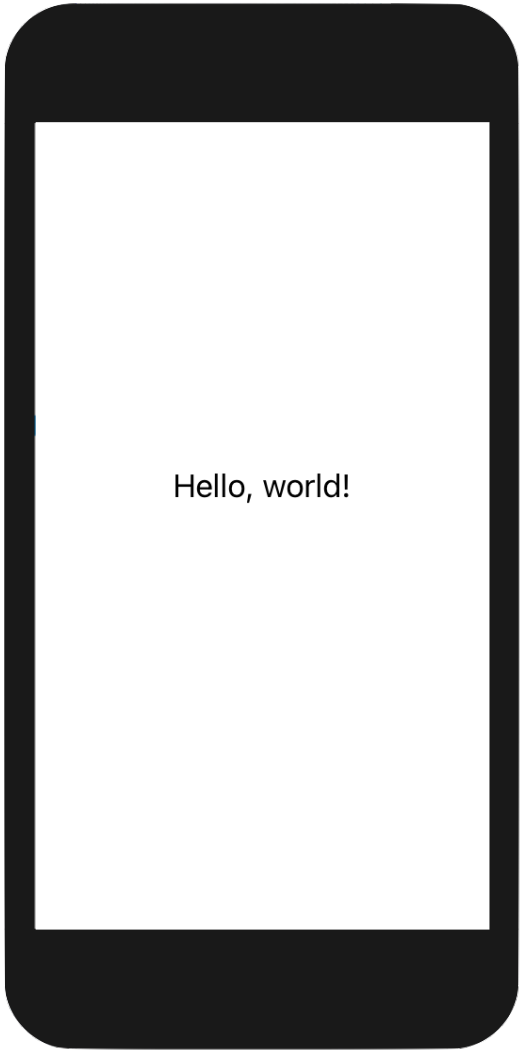
多國語言支援

- 介面文字隨使用者手機改變
- 匯出檔案讓翻譯者使用

修改欲多國語言顯示的文字

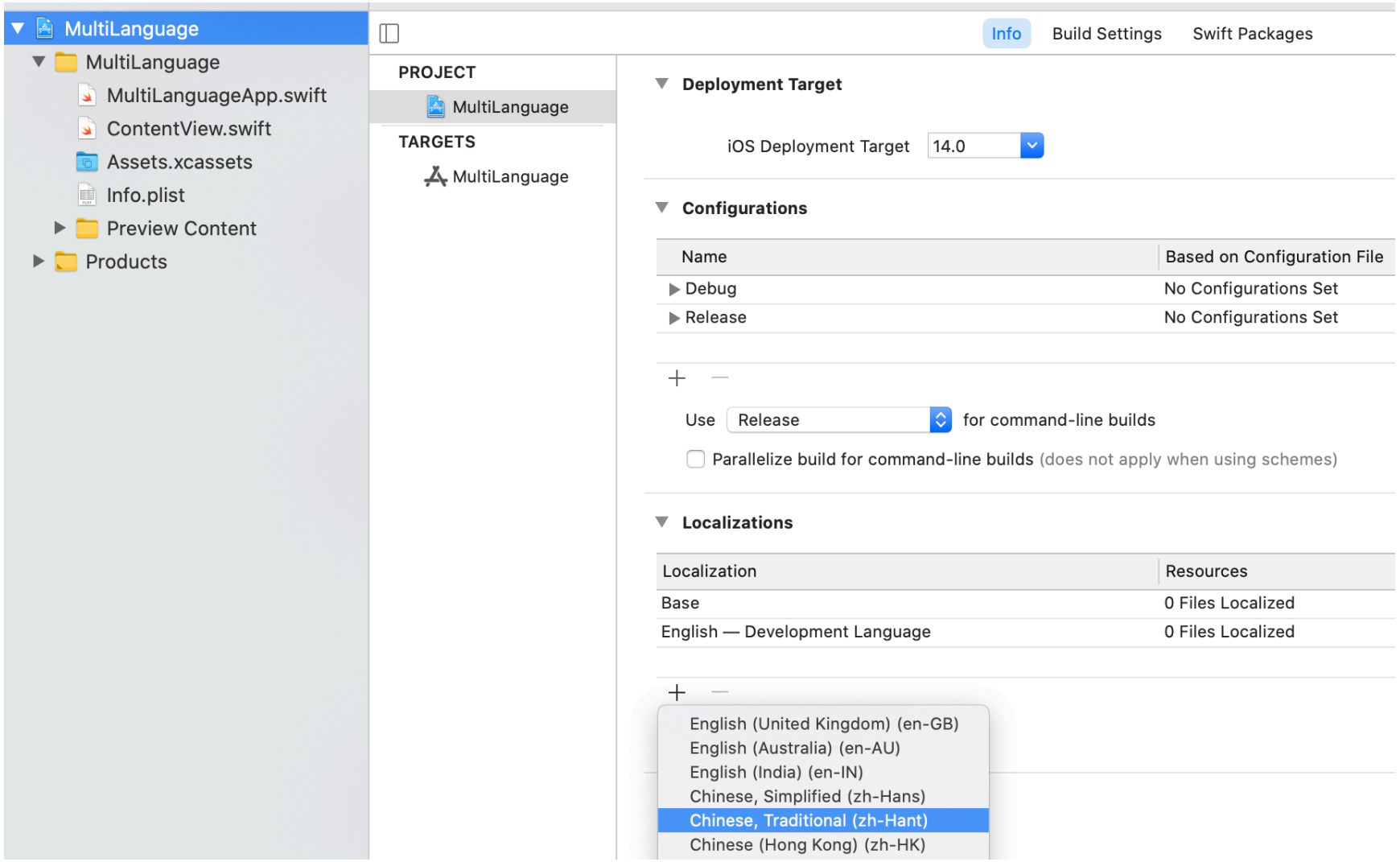
- NSLocalizedString( )

```
struct ContentView: View {
    var body: some View {
        Text(NSLocalizedString("Hello, world!",
              comment: "Hello, world!"))
            .padding()
    }
}
```



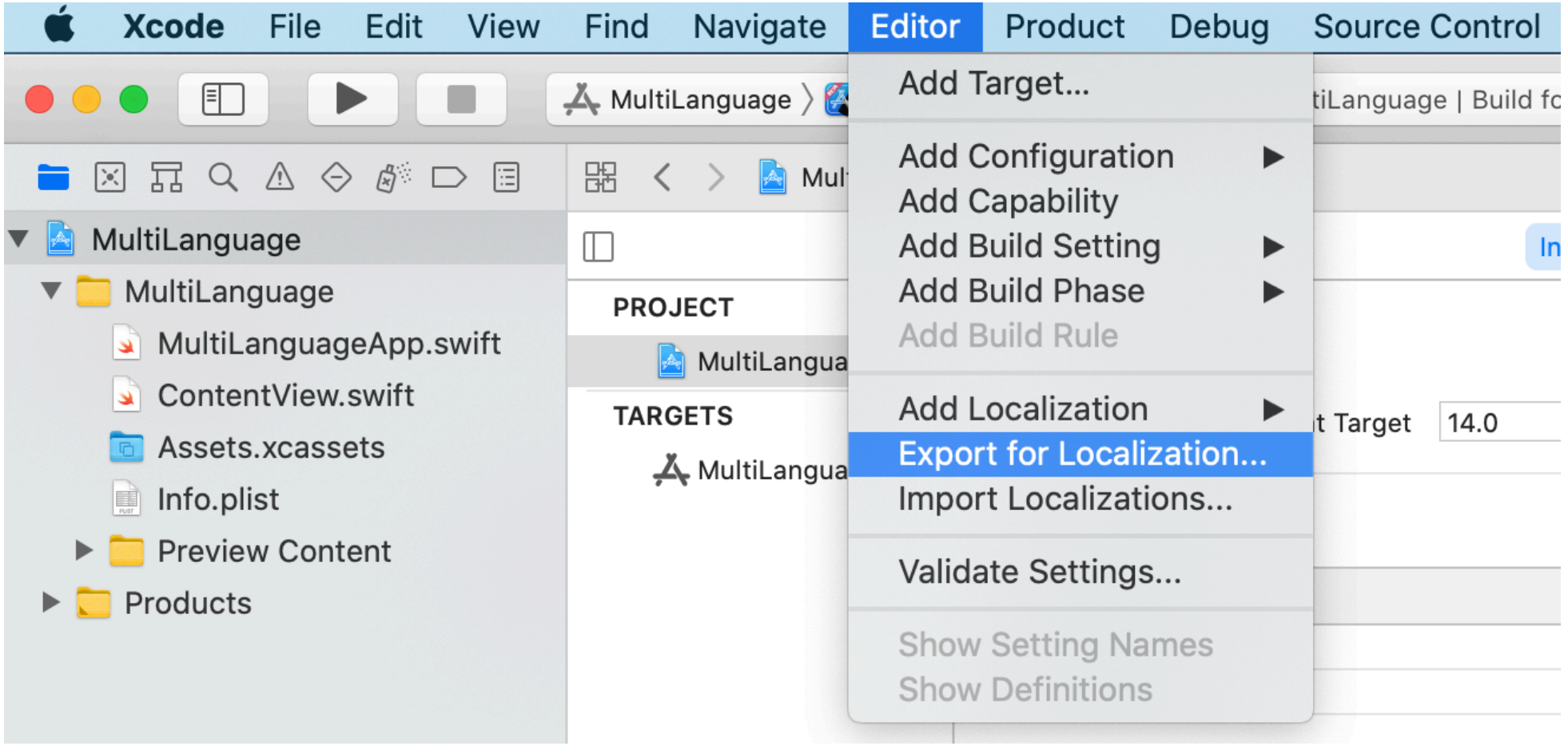
修改專案設定

- Project -> Info -> Localizations -> + 繁中



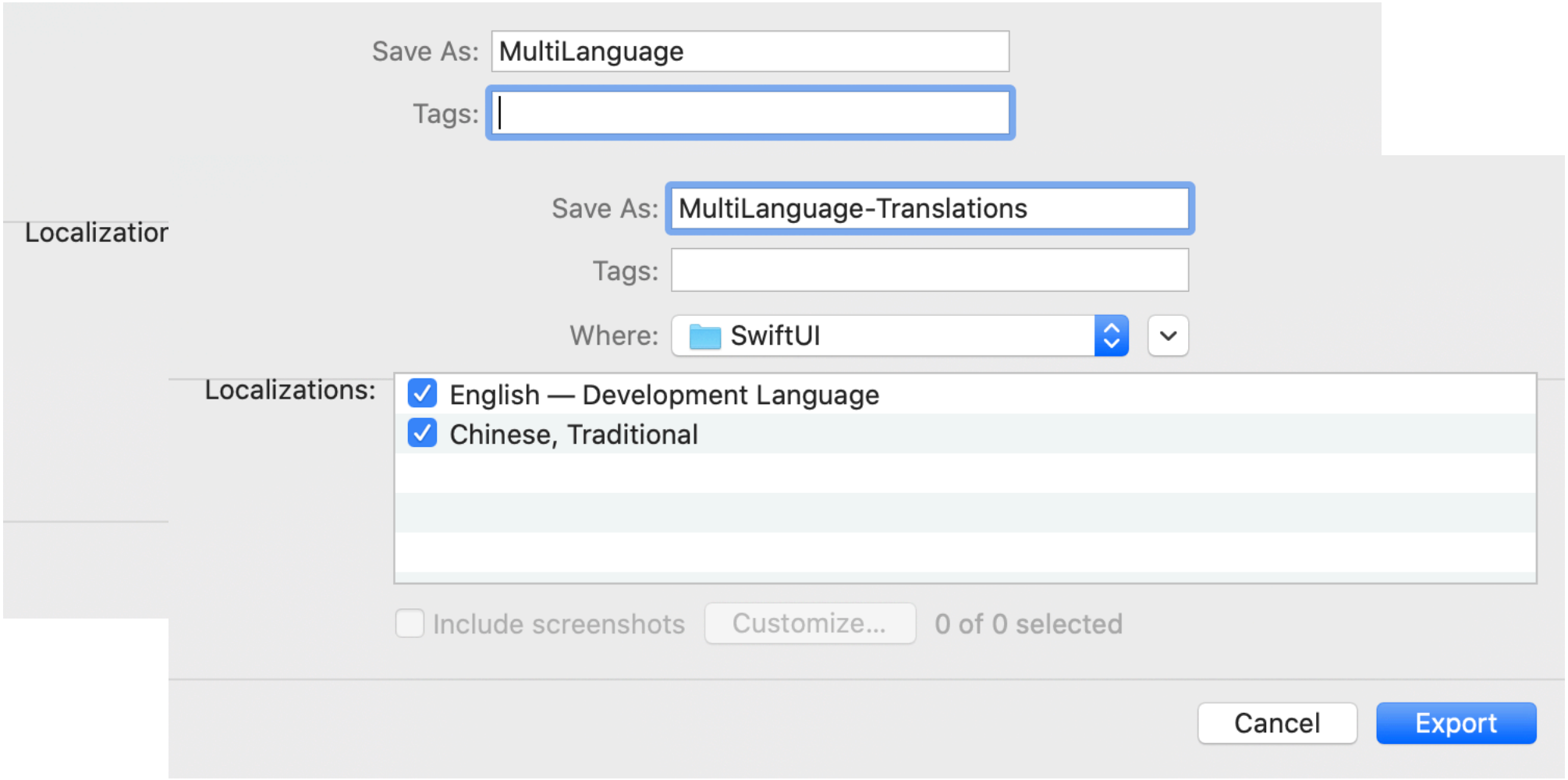
產生翻譯用檔案

- Editor -> Export for Localization...



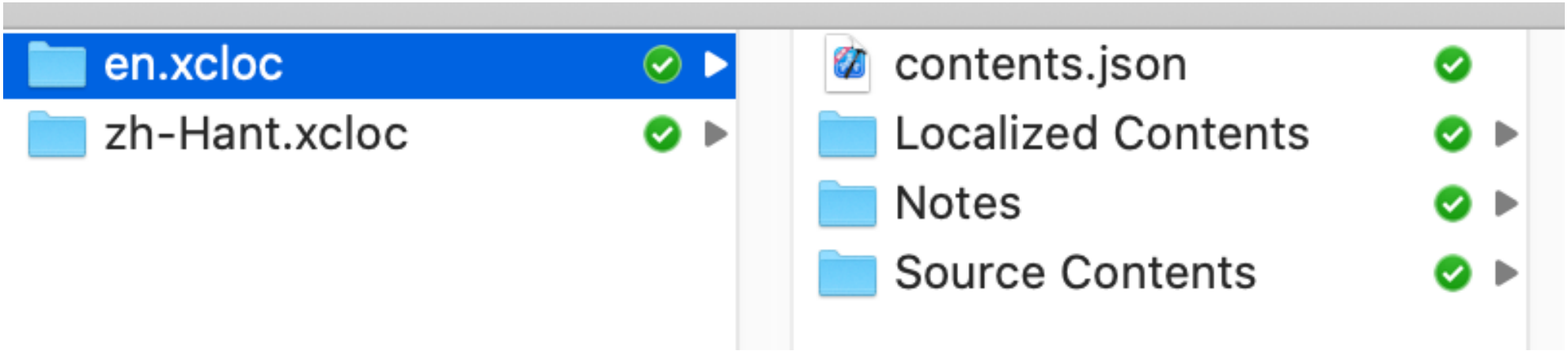
修改欲存檔之資料夾名稱

- 命名後匯出



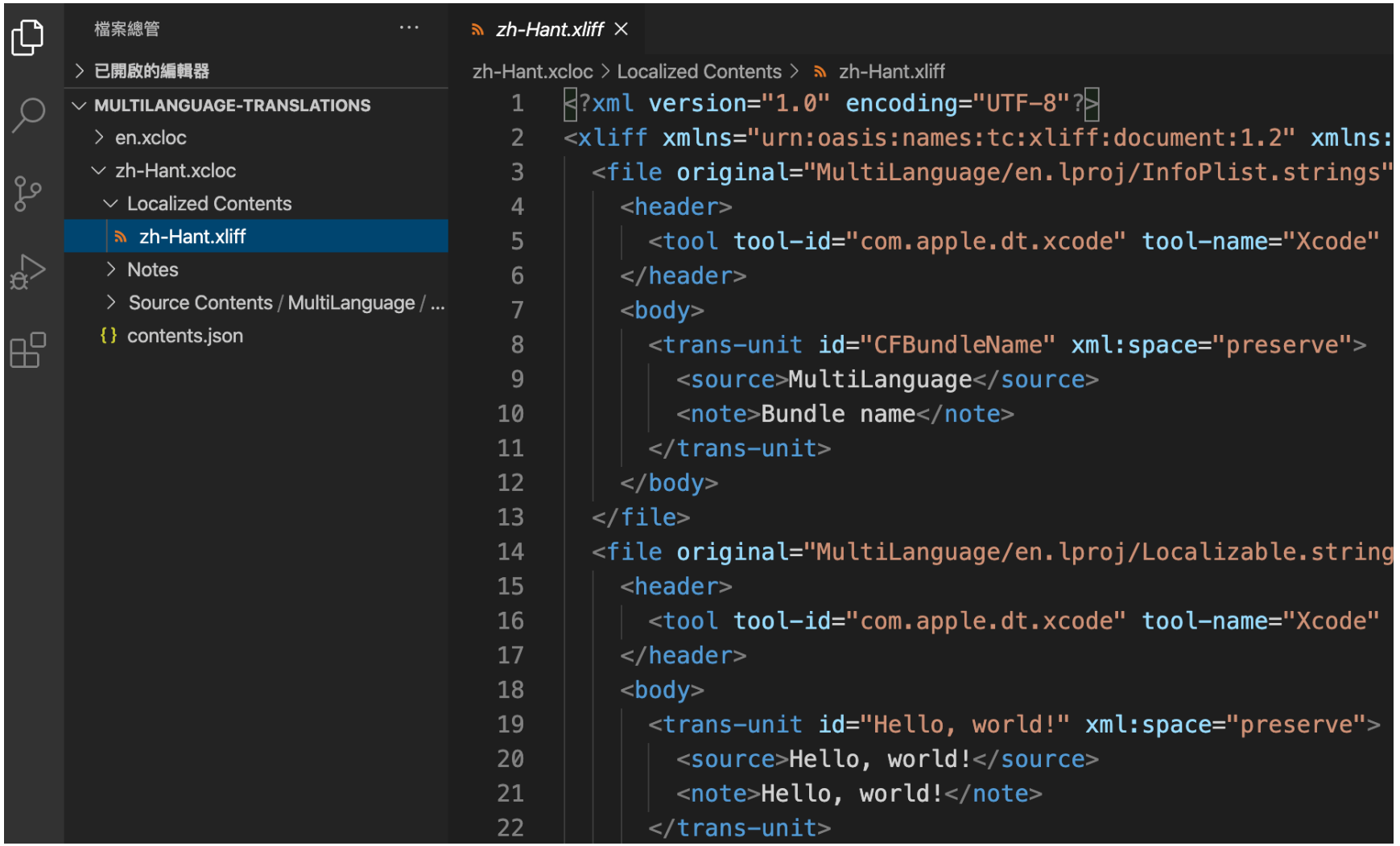
翻譯資料夾

- 檢視資料夾內容
  - contents.json 基本資訊
  - Localized Contents資料夾 主要工作項目
  - Notes 參考資訊
  - Source Contents 相關資訊



打開zh-Hant.xloc資料夾

- Localized Contents -> zh-Hant.xliff



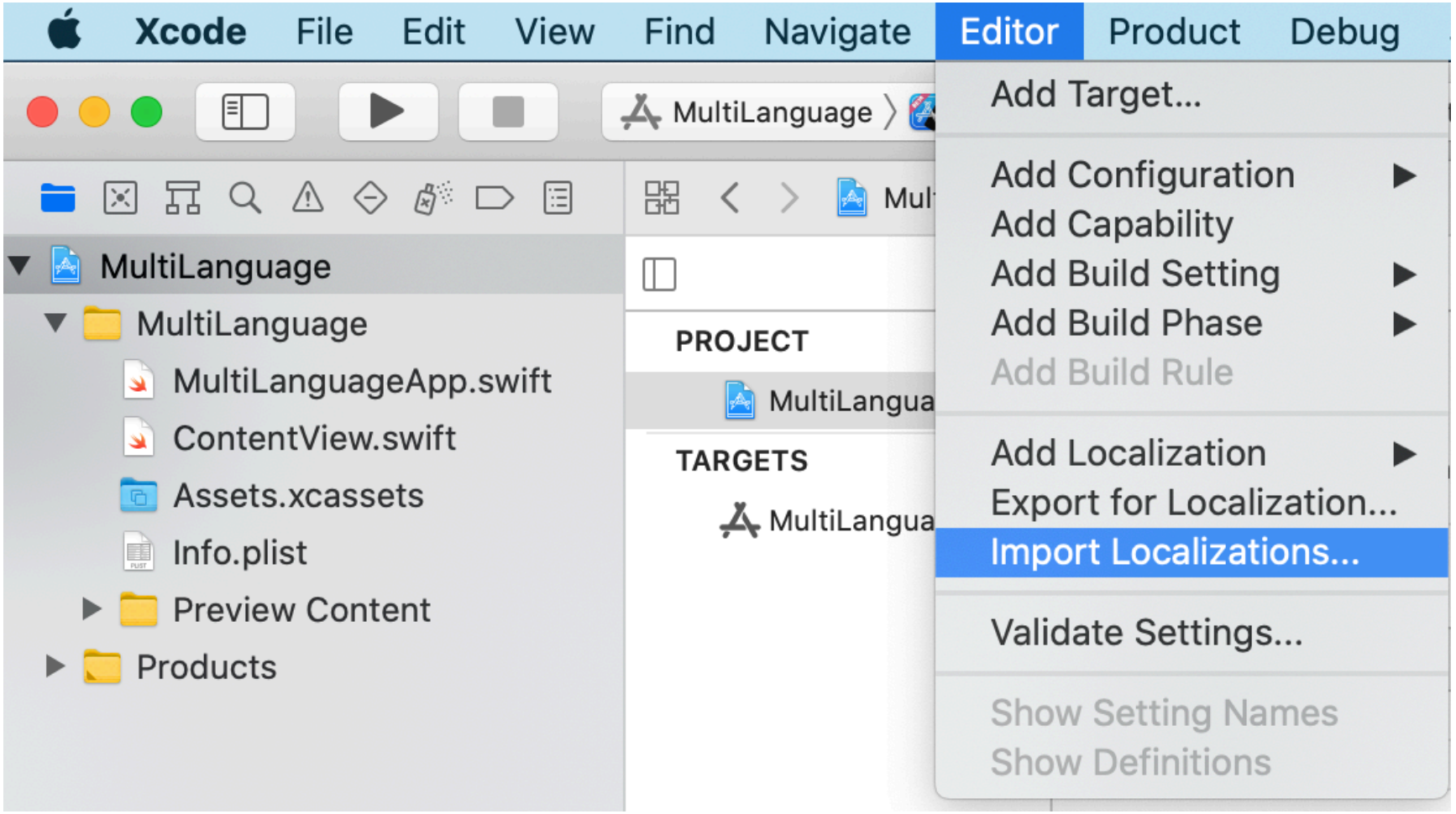
在所有source標籤下增加target

- 將翻譯內容放置於target標籤中

```
<?xml version="1.0" encoding="UTF-8"?>
...
<body>
        <trans-unit id="CFBundleName" xml:space="preserve">
            <source>MultiLanguage</source>
            <target>多國語言顯示</target>
             <note>Bundle name</note>
        </trans-unit>
...
<header>
...
</header>
<body>
        <trans-unit id="Hello, world!" xml:space="preserve">
            <source>Hello, world!</source>
            <target>新世界你好!</target>
            <note>Hello, world!</note>
        </trans-unit>
</body>
</file>
</xliff>
```

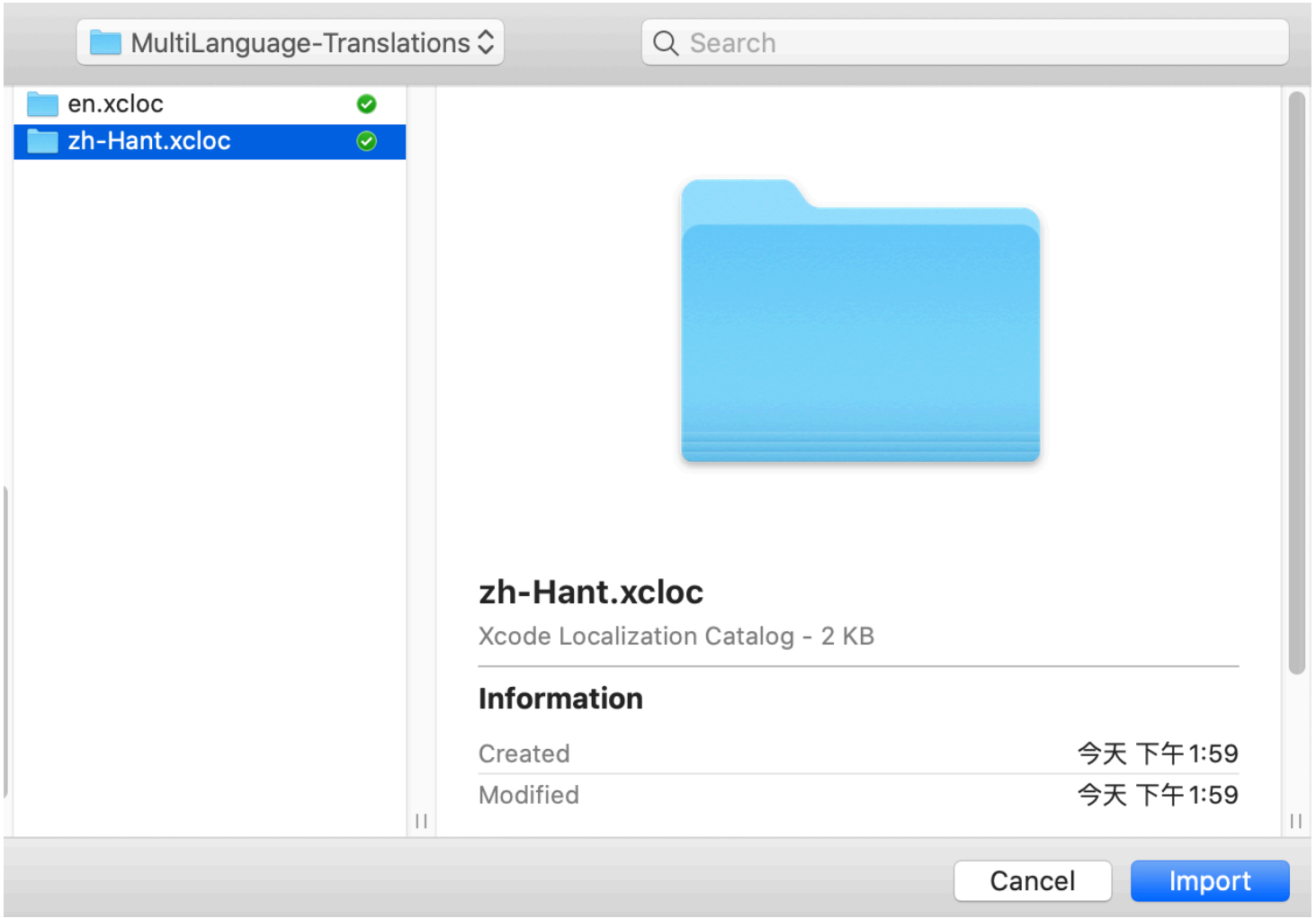
匯回翻譯成果

- Editor -> Import Localizations...



選取翻譯資料夾

- zh-Hant.xcloc

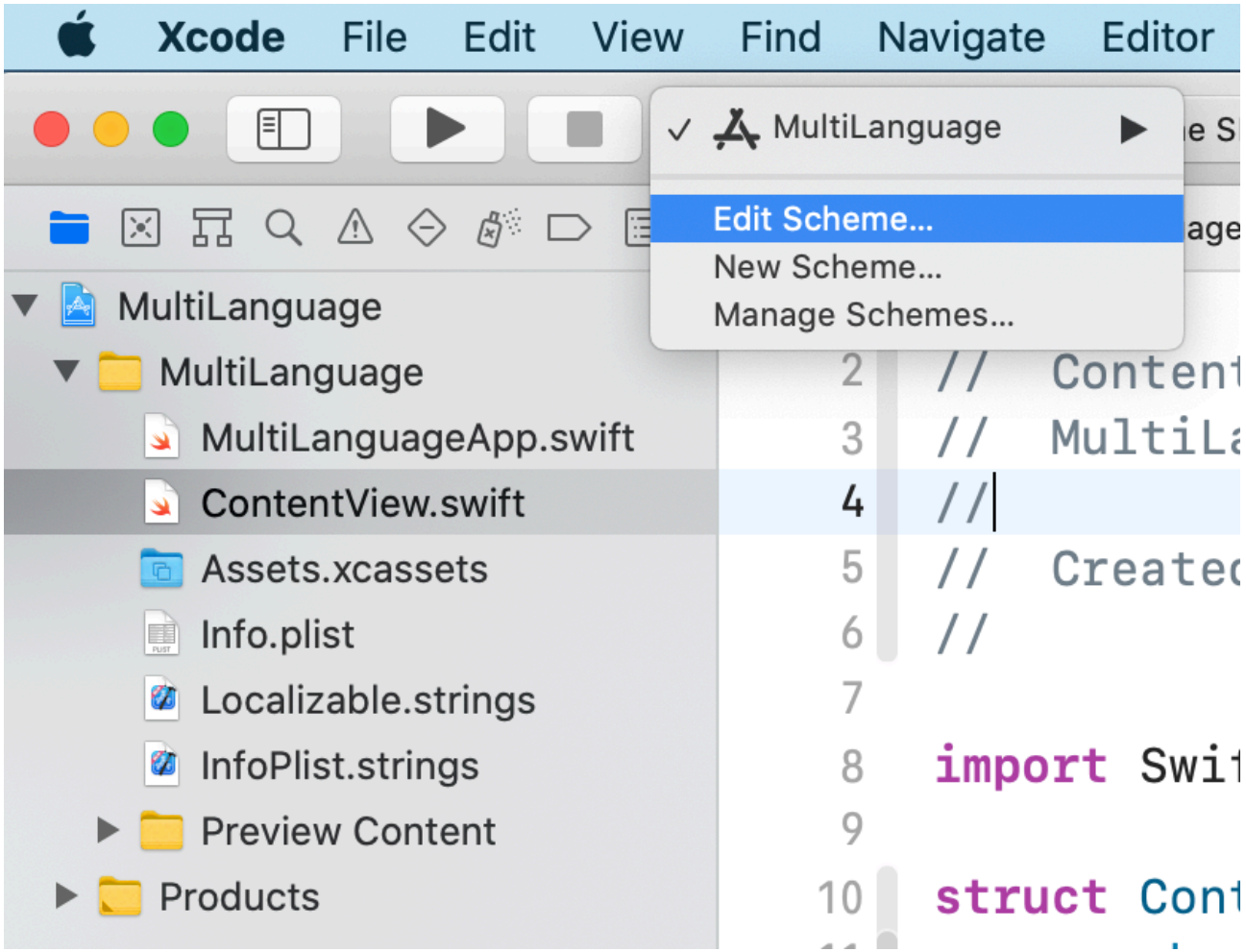


確認是否有出現翻譯文字

- APP內介面文字
- APP名稱

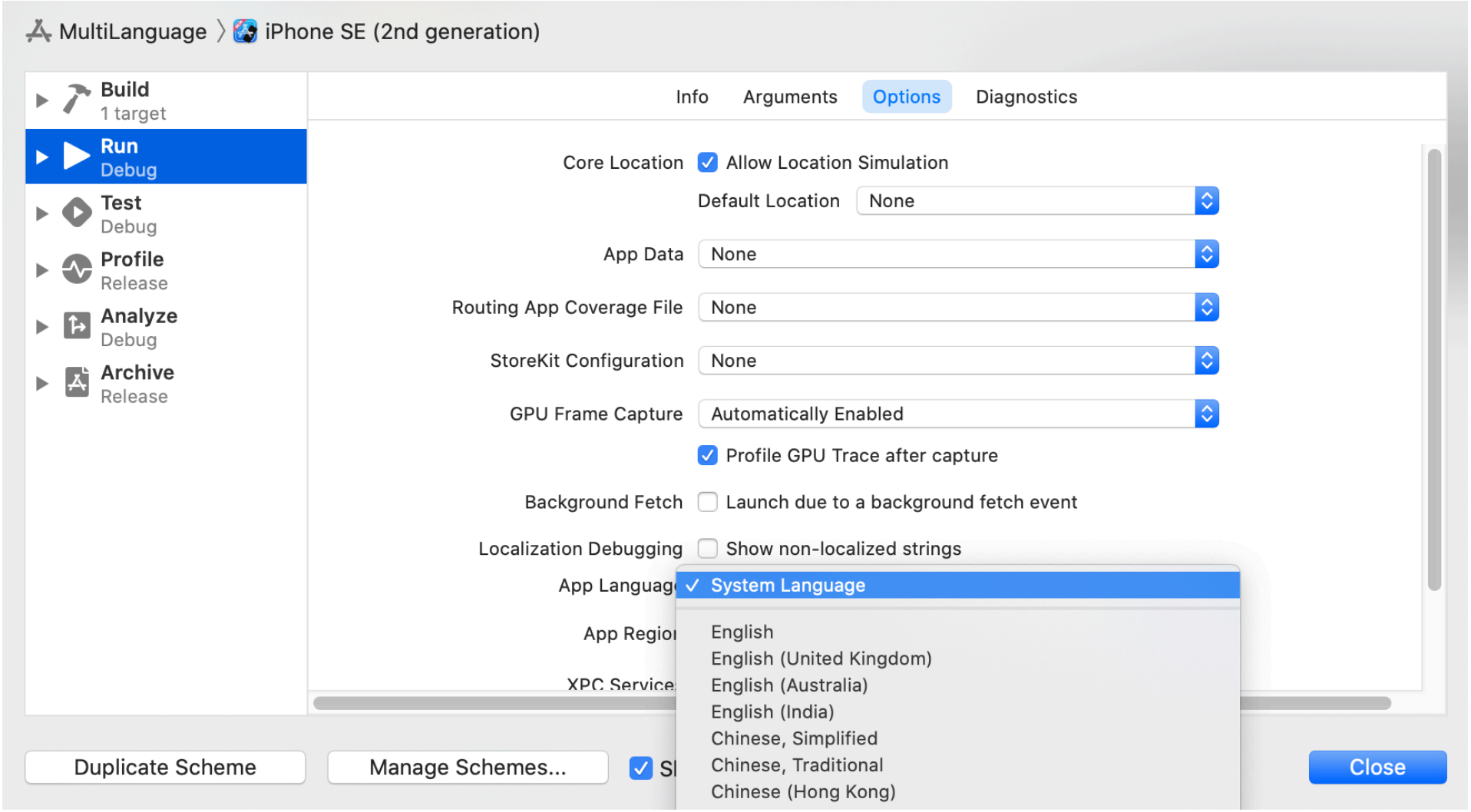
設定執行時顯示語言

- Target -> Edit Scheme...



設定執行時顯示語言

- Run -> Options -> App Language

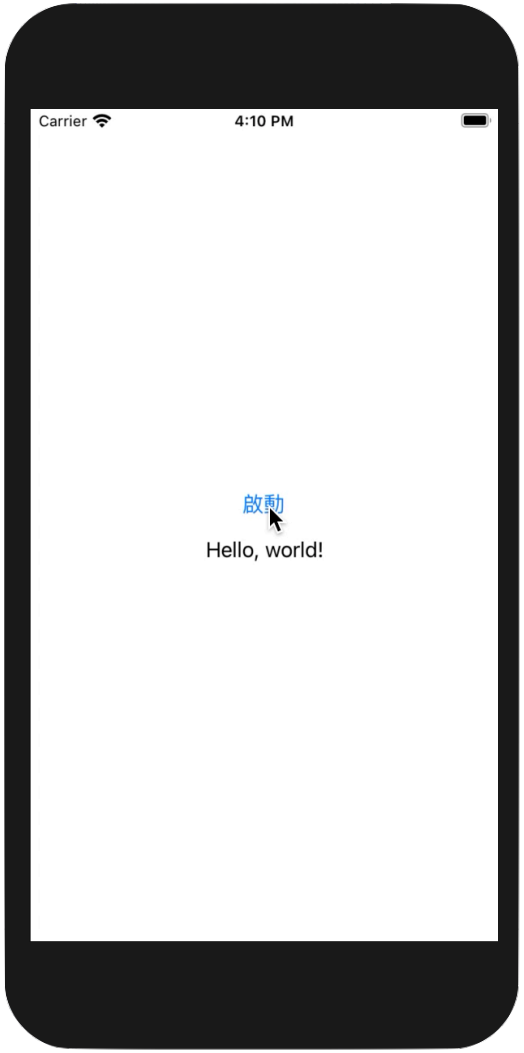


App 通知

- 本地端通知 Local Notification
  - App本身觸發執行
    - 時間觸發 Time-based Trigger
    - 行事曆觸發 Calendar-based Trigger
    - 地點觸發 Location-based Trigger
- 遠端通知 Remote Notification
  - 於伺服器應用程式啟動發出
  - 送至APNS(Apple Push Notification Service)

實作使用者通知

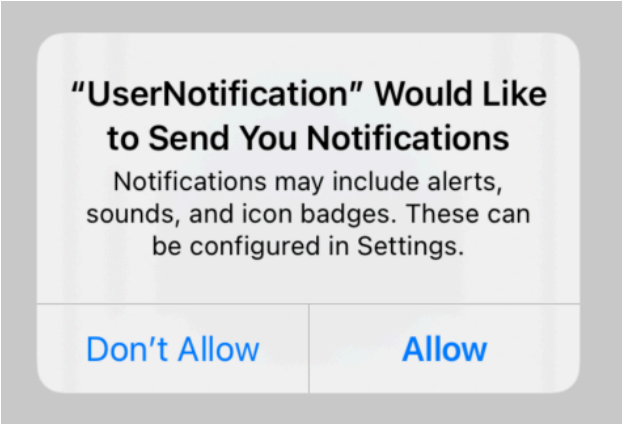
- 10秒後App主動通知
- 點擊通知可進入App



取得使用者允許

- UNUserNotificationCenter

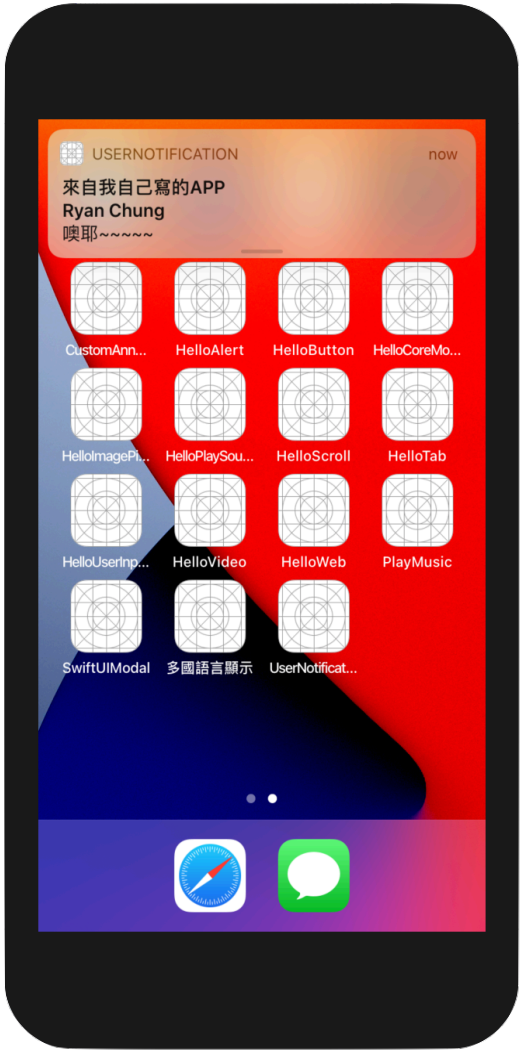
```
struct ContentView: View {
    var body: some View {
        VStack{
            Button(action:{

            }){
                Text("啟動")
            }
            Text("Hello, world!").padding()
        }.onAppear {
               UNUserNotificationCenter.current()
               .requestAuthorization(options: [.alert,.sound,.badge]){
                (granted, error) in
                if granted{
                    print("User notification set!")
                }else{
                    print("User Do Not allowed!!")
                }
            }
        }
    }
}
```



編輯Button Action

- UNNotificationRequest

```
struct ContentView: View {
    var body: some View {
        VStack{
            Button(action:{
                let content = UNMutableNotificationContent()
                content.title = "來自我自己寫的APP"
                content.subtitle = "Ryan Chung"
                content.body = "噢耶~~~~~"
                content.sound = UNNotificationSound.default

                let trigger = UNTimeIntervalNotificationTrigger(
                              timeInterval: 10,
                              repeats: false)
                let request = UNNotificationRequest(
                              identifier: "Notify Test from Ryan",
                              content: content, trigger: trigger)
                UNUserNotificationCenter.current()
                .add(request, withCompletionHandler: nil)
            }){
                Text("啟動")
            }
            Text("Hello, world!").padding()
```



發通知不附圖，此風不可長

- 在通知中夾帶圖片
- 點擊後可放大

UNNotificationAttachment

```
Button(action:{
          let content = UNMutableNotificationContent()
          content.title = "來自我自己寫的APP"
          content.subtitle = "Ryan Chung"
          content.body = "噢耶~~~~~"
          content.sound = UNNotificationSound.default

          if let image = UIImage(named: "mobiledevtw"){
                let tempDirURL = URL(fileURLWithPath: NSTemporaryDirectory(),
                    isDirectory: true)
                let tempFileURL = tempDirURL.appendingPathComponent("mobiledevtw.png")

                try? image.jpegData(compressionQuality: 1.0)?.write(to: tempFileURL)
                if let mobiledevImage = try?
                    UNNotificationAttachment(identifier: "mobiledevtw",
                                             url: tempFileURL, options: nil){
                content.attachments = [mobiledevImage]
             }
          }

          let trigger = UNTimeIntervalNotificationTrigger(timeInterval: 10, repeats: false)
          let request = UNNotificationRequest(identifier: "Notify Test from Ryan",
                        content: content, trigger: trigger)
           UNUserNotificationCenter.current().add(request, withCompletionHandler: nil)
       }){
                Text("啟動")
       }
```

清單排序與過濾

- 課程過濾
  - 只顯示特色課程
  - 只顯示低於12hr的課程
- 課程排序
  - 依首字字母排序
  - 依課程時數排序
  - 特色課程放在前面

清單過濾

- 過濾
  - 關關：是否只顯示特色課程
  - 加號減號：設定只顯示低於特定時數的課程

CourseListView

- 調整資料結構與內容

```
struct Course: Identifiable{
        var id = UUID()
        var name:String
        var image:String
        var description:String
        var isFeature:Bool
        var hours:Int
    }

    var courses = [
        Course(name: "Intro to AI", image: "ai", description: "人工智慧介紹",
               isFeature:true, hours:12),
        Course(name: "Intro to Python", image: "python", description: "Python介紹",
               isFeature:true, hours:18),
        Course(name: "Intro to Data Science", image: "data-science", description: "資料科學介紹",
               isFeature:false, hours:12),
        Course(name: "Intro to Azure", image: "azure", description: "微軟Azure介紹",
               isFeature:false, hours:6),
        Course(name: "Intro to AWS", image: "aws", description: "亞馬遜AWS介紹",
               isFeature:false, hours:6),
    ]
```

SettingView

- 寫入AppStorage

```
struct SettingView: View {
    @State var colorArray:Array = [255.0,255.0,255.0]
    @State var sliderValue = 0.0
    @AppStorage("UserName") var UserName:String = ""
    @AppStorage("showFeatureOnly") var showFeatureOnly = false
    @AppStorage("showHoursLowerThanOnly") var showHoursLowerThanOnly:Int = 12

    var body: some View {
        NavigationView{
            Form{
                Section(header: Text("使用者名稱")) {
                    TextField("請輸入您的名字", text: $UserName)
                }
                Section(header: Text("只顯示特色課程")){
                    Toggle(isOn:$showFeatureOnly){
                        Text("是否 : (\(String(showFeatureOnly)))")
                    }
                }
                Section(header: Text("課程時數")){
                    Stepper("只顯示 \(showHoursLowerThanOnly) 小時以下課程",
                        onIncrement: {
                            showHoursLowerThanOnly += 6
                        }, onDecrement: {
                            showHoursLowerThanOnly -= 6
                        })
                }
            }
            .navigationBarTitle("Settings 設定")
        }
    }
}
```

CourstListView

- 在結構中加入AppStorage讀取

```
struct CourseListView: View {

    @State var showDetailView = false
    @State var selectedCourse:Course?

    @AppStorage("showFeatureOnly") var showFeatureOnly = false
    @AppStorage("showHoursLowerThanOnly") var showHoursLowerThanOnly:Int = 12
```

CourseListView

- 在結構中加入一個函數：shouldShowItem
  - 如果是只要顯示特色課程的話，就只抓出isFeature為真的
  - 只抓出課程時數小於等於使用者設定的時數值

```
private func shouldShowItem(course:Course)->Bool{
        return (showFeatureOnly ? course.isFeature : true) &&
               (course.hours<=showHoursLowerThanOnly)
    }
```

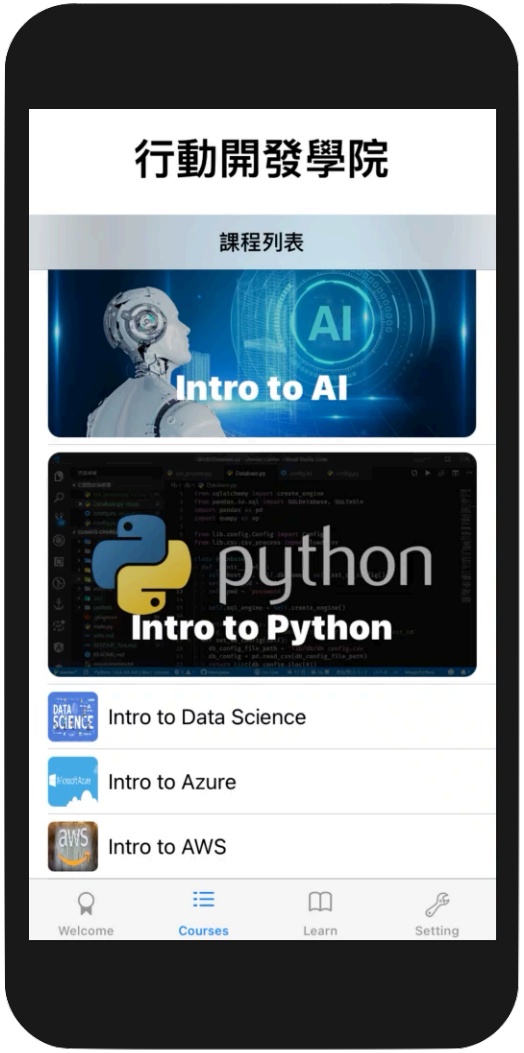
CourseListView

- 利用shouldShowItem函數來過濾

```
var body: some View {
        NavigationView{
            List{
                ForEach(courses){
                    courseItem in
                    if self.shouldShowItem(course: courseItem){
                        if courseItem.isFeature{
                            FullImageRow(thisCourse: courseItem)
                                .onTapGesture {
                                    self.showDetailView = true
                                    self.selectedCourse = courseItem
                                }
                        }else{
                            BasicImageRow(thisCourse: courseItem)
                                .onTapGesture {
                                    self.showDetailView = true
                                    self.selectedCourse = courseItem
                                }
                        }
                    }
                }
             }
            .sheet(item: self.$selectedCourse){ thisCourse in
                CourseDetailView(course: thisCourse)
            }
            .navigationBarTitle("課程列表")
        }
 }
```

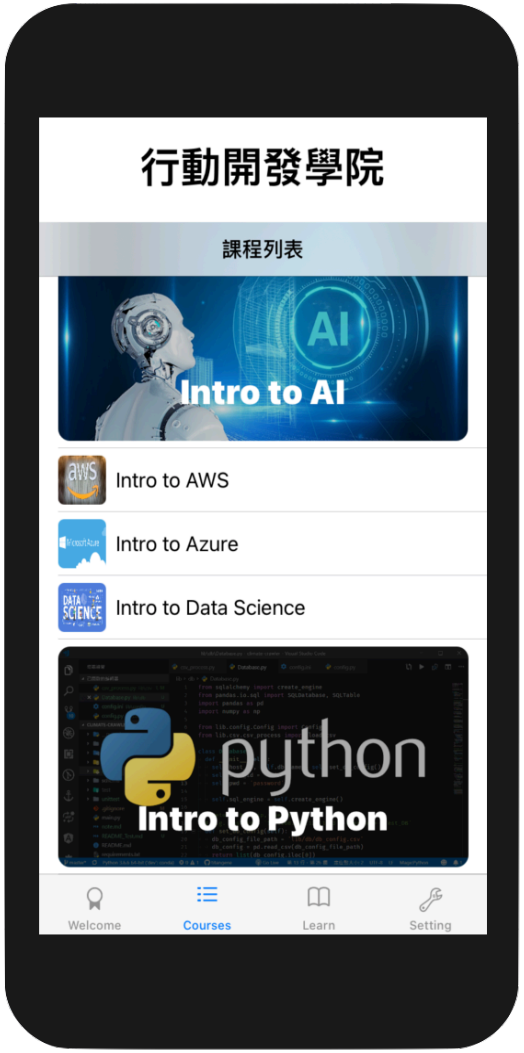
測試

- 只顯示特色課程
- 只顯示特定時數以下的課程



清單排序

- 課程排序
  - 依首字字母排序
  - 依課程時數排序
  - 特色課程放在前面



DisplayOrderType.swift

- 新增檔案

```
import Foundation

enum DisplayOrderType: Int, CaseIterable {
    case featureFirst = 0
    case alphabetical = 1
    case shortHoursFirst = 2

    init(type:Int){
        switch type {
        case 0: self = .featureFirst
        case 1: self = .alphabetical
        case 2: self = .shortHoursFirst
        default:self = .featureFirst
        }
    }

    var text:String {
        switch self {
        case .featureFirst: return "Feature First"
        case .alphabetical: return "Alphabetical"
        case .shortHoursFirst: return "Short Hours First"
        }
    }

    func predicate()->((Course, Course)->Bool){
        switch self {
        case .alphabetical: return {$0.name < $1.name}
        case .featureFirst: return {$0.isFeature && !$1.isFeature}
        case .shortHoursFirst: return {$0.hours < $1.hours}
        }
    }
}
```

SettingView

- 將Picker選項改為剛才設計的enum

```
import SwiftUI

struct SettingView: View {
    @State var colorArray:Array = [255.0,255.0,255.0]
    @State var sliderValue = 0.0

    @AppStorage("UserName") var UserName:String = ""

    @AppStorage("displayOrder") var displayOrder = DisplayOrderType.featureFirst
    @AppStorage("showFeatureOnly") var showFeatureOnly = false
    @AppStorage("showHoursLowerThanOnly") var showHoursLowerThanOnly:Int = 12

    var body: some View {
        NavigationView{
            Form{
                Section(header: Text("使用者名稱")) {
                    TextField("請輸入您的名字", text: $UserName)
                }
                Section(header: Text("課程排序")){
                    Picker(selection:$displayOrder,label:Text("依據")){
                        ForEach(DisplayOrderType.allCases,id:\.self){
                            orderType in
                            Text(orderType.text)
                        }
                    }
                }
```

CourseListView

- 在結構中增加AppStorage取用

```
struct CourseListView: View {

    @State var showDetailView = false
    @State var selectedCourse:Course?

    @AppStorage("displayOrder") var displayOrder = DisplayOrderType.featureFirst
    @AppStorage("showFeatureOnly") var showFeatureOnly = false
    @AppStorage("showHoursLowerThanOnly") var showHoursLowerThanOnly:Int = 12
```

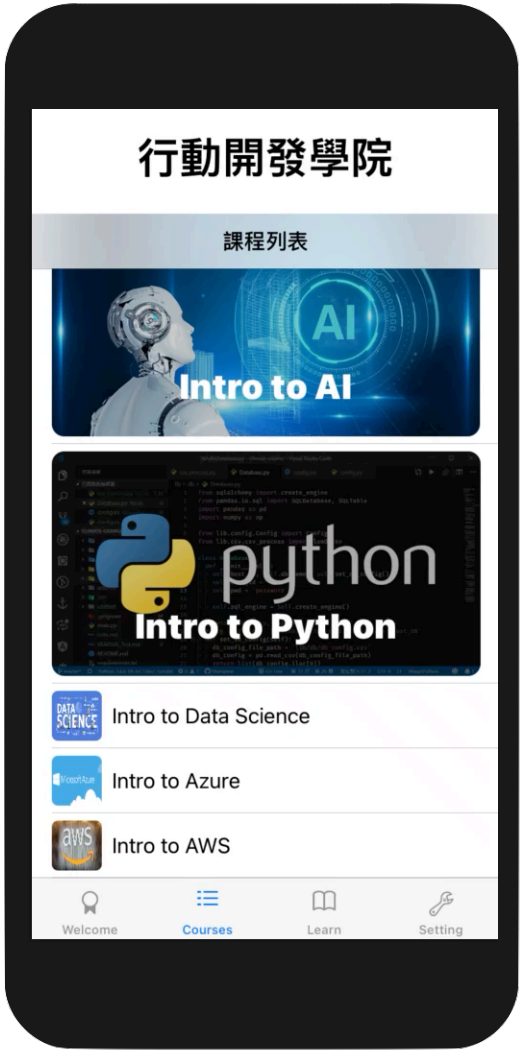
CourseListView

- 在List的ForEach套用predicate方法於排序

```
var body: some View {
        NavigationView{
            List{
                ForEach(courses.sorted(by: displayOrder.predicate())){
                    courseItem in
                    if self.shouldShowItem(course: courseItem){
                        if courseItem.isFeature{
                            FullImageRow(thisCourse: courseItem)
                                .onTapGesture {
                                    self.showDetailView = true
                                    self.selectedCourse = courseItem
                                }
                        }else{
                            BasicImageRow(thisCourse: courseItem)
                                .onTapGesture {
                                    self.showDetailView = true
                                    self.selectedCourse = courseItem
                                }
                        }
                    }
                }
            }
        }
```

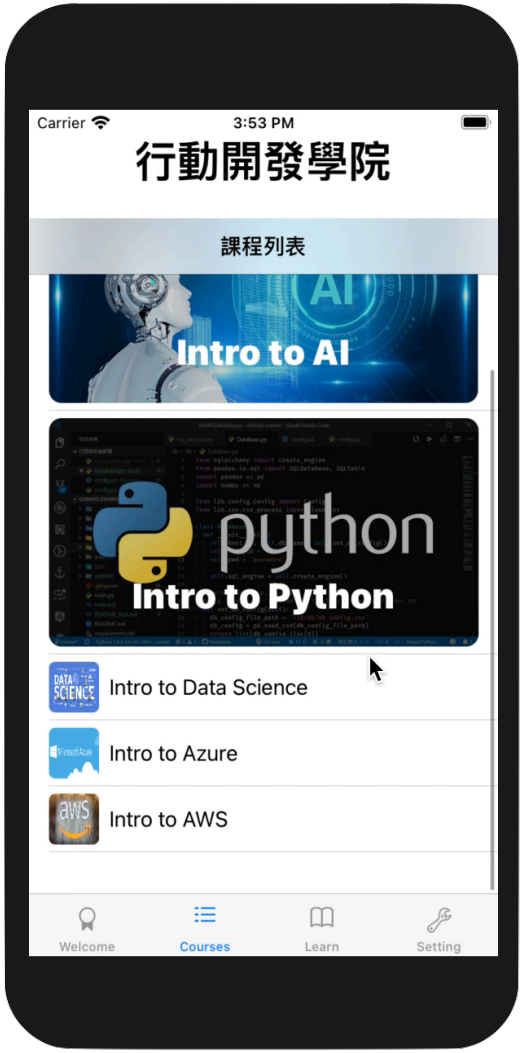
測試

- 三種排序
  - 特色課程優先
  - 照課程名稱A-Z排序
  - 照課程時數由小到大排序



項目可刪除

- 在課程項目上滑動
  - 可刪除該項目



CourseListView

- 將課程資料搬入CourseListView並加上@State

```
struct CourseListView: View {

    @State var showDetailView = false
    @State var selectedCourse:Course?

    @AppStorage("displayOrder") var displayOrder = DisplayOrderType.featureFirst
    @AppStorage("showFeatureOnly") var showFeatureOnly = false
    @AppStorage("showHoursLowerThanOnly") var showHoursLowerThanOnly:Int = 12

    @State var courses = [
        Course(name: "Intro to AI", image: "ai", description: "人工智慧介紹", isFeature:true, hours:12),
        Course(name: "Intro to Python", image: "python", description: "Python介紹", isFeature:true, hours:18),
        Course(name: "Intro to Data Science", image: "data-science", description: "資料科學介紹", isFeature:false,
hours:12),
        Course(name: "Intro to Azure", image: "azure", description: "微軟Azure介紹", isFeature:false, hours:6),
        Course(name: "Intro to AWS", image: "aws", description: "亞馬遜AWS介紹", isFeature:false, hours:6),
    ]
```

CourseListView

- 在ForEach後面加上onDelete方法

```
var body: some View {
        NavigationView{
            List{
                ForEach(courses.sorted(by: displayOrder.predicate())){
                    courseItem in
                    if self.shouldShowItem(course: courseItem){
                        if courseItem.isFeature{
                            FullImageRow(thisCourse: courseItem)
                                .onTapGesture {
                                    self.showDetailView = true
                                    self.selectedCourse = courseItem
                                }
                        }else{
                            BasicImageRow(thisCourse: courseItem)
                                .onTapGesture {
                                    self.showDetailView = true
                                    self.selectedCourse = courseItem
                                }
                        }
                    }
                }.onDelete { indexSet in
                    courses = courses.sorted(by: displayOrder.predicate())
                    courses.remove(atOffsets: indexSet)
                }
            }
            .sheet(item: self.$selectedCourse){ thisCourse in
                CourseDetailView(course: thisCourse)
            }
```

互動小選單

- 長按項目出現互動小選單
  - 可刪除特定課程
  - 可加入我的最愛

修改Course結構

- 新增 isFavorite預設值為false

```
struct Course: Identifiable{
    var id = UUID()
    var name:String
    var image:String
    var description:String
    var isFeature:Bool
    var hours:Int
    var isFavorite:Bool = false
}
```

CourseListView

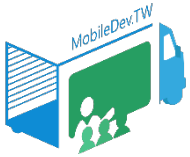
- 新增delete、setFavorite方法

```
private func shouldShowItem(course:Course)->Bool{
    return (showFeatureOnly ? course.isFeature : true) &&
           (course.hours<=showHoursLowerThanOnly)
}

private func delete(item course:Course){
    if let index = self.courses.firstIndex(where: {
        $0.id == course.id
    }){
        courses.remove(at: index)
    }
}

private func setFavorite(item course:Course){
    if let index = self.courses.firstIndex(where: {
        $0.id == course.id
    }){
        courses[index].isFavorite.toggle()
    }
}
```

List -> BasicImageRow

- 加入contextMenu

```
BasicImageRow(thisCourse: courseItem)
                .onTapGesture {
                    self.showDetailView = true
                    self.selectedCourse = courseItem
                }
                .contextMenu{
                    Button(action:{
                        self.delete(item: courseItem)
                    }){
                        HStack{
                            Text("刪除該課程")
                            Image(systemName: "trash")
                        }

                    }
                    Button(action:{
                        self.setFavorite(item: courseItem)
                    }){
                        HStack{
                            Text("加入/刪除我的最愛")
                            Image(systemName: "star")
                        }
                    }
                }
```



資料處理後，畫面呈現也要

- 更改BasicImageRow的呈現

```
struct BasicImageRow: View {
        var thisCourse:Course
        var body: some View {
            HStack{
                Image(thisCourse.image)
                    .resizable()
                    .frame(width: 40, height: 40)
                    .cornerRadius(5)
                Text(thisCourse.name)
                if thisCourse.isFavorite{
                    Spacer()
                    Image(systemName: "star.fill")
                        .foregroundColor(.yellow)
                }
            }
        }
    }
```

FullImageRow也想要這個效果

- 一樣加上contextMenu

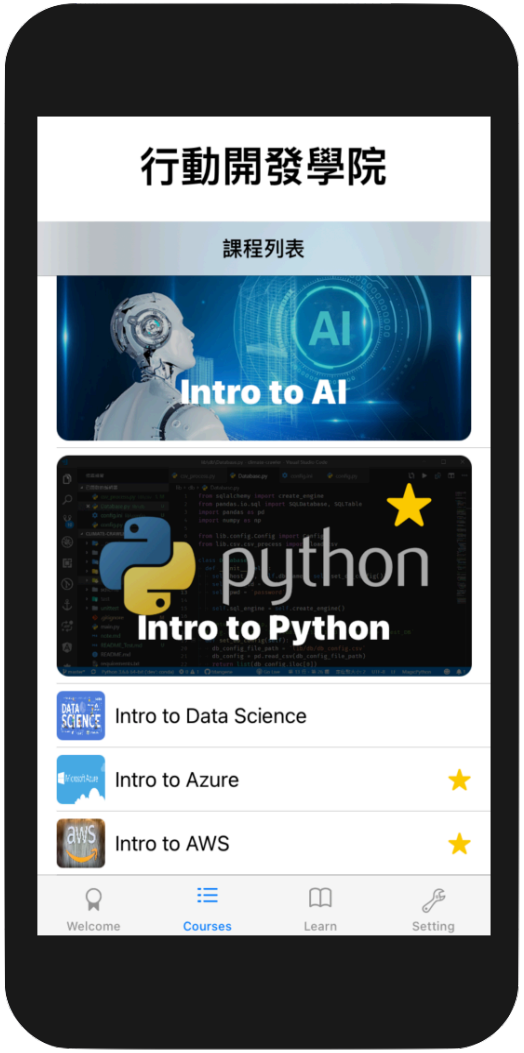
```
if courseItem.isFeature{
            FullImageRow(thisCourse: courseItem)
                .onTapGesture {
                    self.showDetailView = true
                    self.selectedCourse = courseItem
                }
                .contextMenu{
                    Button(action:{
                        self.delete(item: courseItem)
                    }){
                        HStack{
                            Text("刪除該課程")
                            Image(systemName: "trash")
                        }

                    }
                    Button(action:{
                        self.setFavorite(item: courseItem)
                    }){
                        HStack{
                            Text("加入/刪除我的最愛")
                            Image(systemName: "star")
                        }
                    }
                }
```

FullImageRow呈現方式也要修改

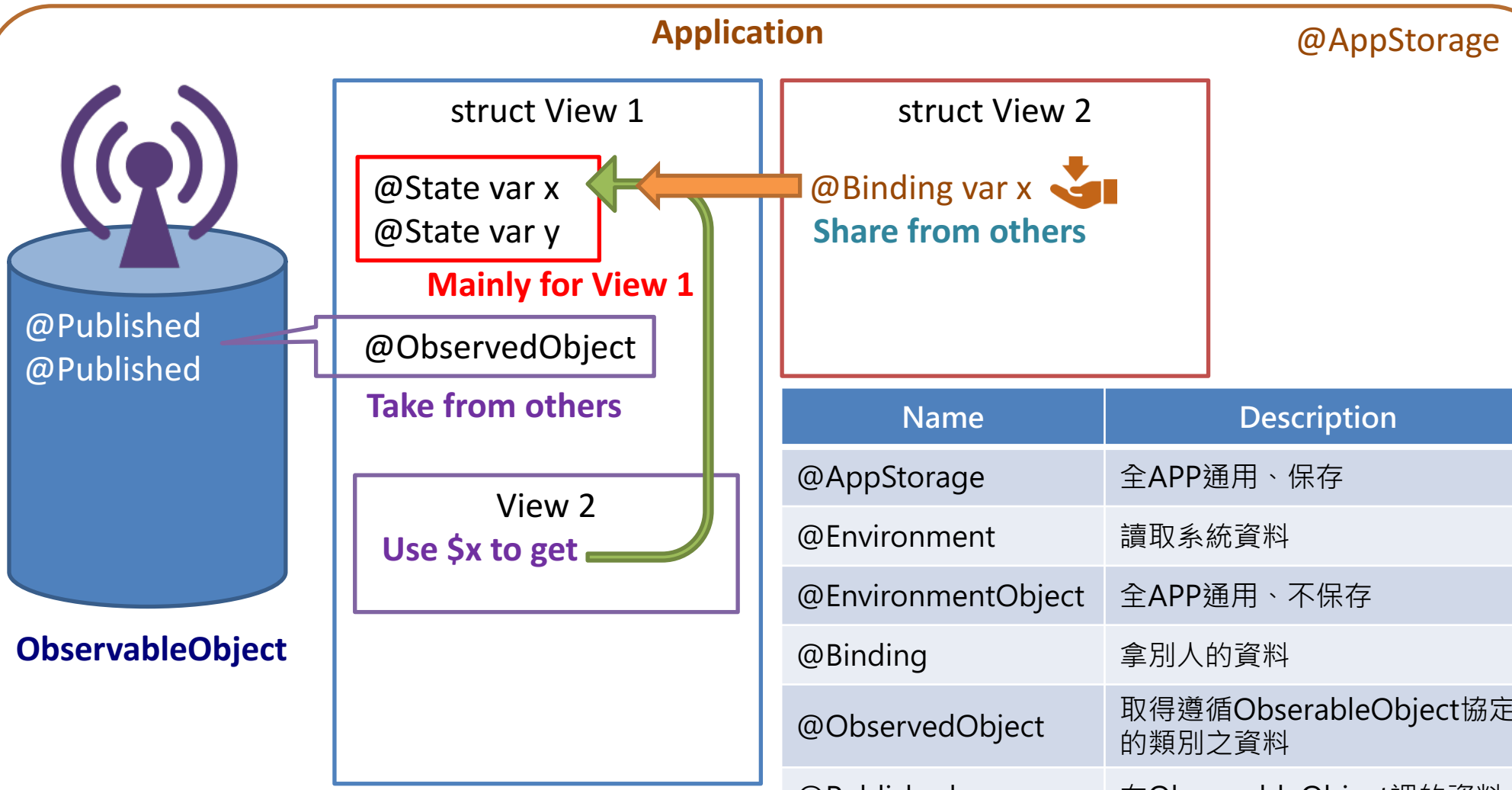
```
struct FullImageRow: View {
        var thisCourse:Course
        var body: some View {
            ZStack{
                Image(thisCourse.image)
                    .resizable()
                    .aspectRatio(contentMode: .fill)
                    .frame(height: 180)
                    .cornerRadius(10)
                    .overlay(
                        Rectangle()
                            .foregroundColor(.black)
                            .cornerRadius(10)
                            .opacity(0.2)
                    )
                Text(thisCourse.name)
                    .font(.system(.title))
                    .fontWeight(.black)
                    .foregroundColor(.white)
                    .offset(x: 0.0, y: 50.0)
                if thisCourse.isFavorite{
                    Spacer()
                    Image(systemName: "star.fill")
                        .foregroundColor(.yellow)
                        .offset(x: 120.0, y: -50.0)
                        .font(.largeTitle)
                }
            }
        }
    }
```

測試

- 刪除課程
- 加入/刪除我的最愛



Data flow between views & structs



Name	Description
@AppStorage	全APP通用、保存
@Environment	讀取系統資料
@EnvironmentObject	全APP通用、不保存
@Binding	拿別人的資料
@ObservedObject	取得遵循ObserableObject協定的類別之資料
@Published	在ObservableObject裡的資料
@State	一個View中可修改的資料

Mobile Development Academy

Struct VS. Class

- Struct
  - 簡潔快速
  - 概念上不能變更(但可以有@State變數)
- Class
  - 撰寫上常要考慮繼承、父類別子類別問題

App實機測試

- 無開發者帳號
  - 也可使用，但有數量限制
  - 部分功能受限
- 有開發者帳號
  - 遠端推播通知可
  - CloudKit
  - App上架

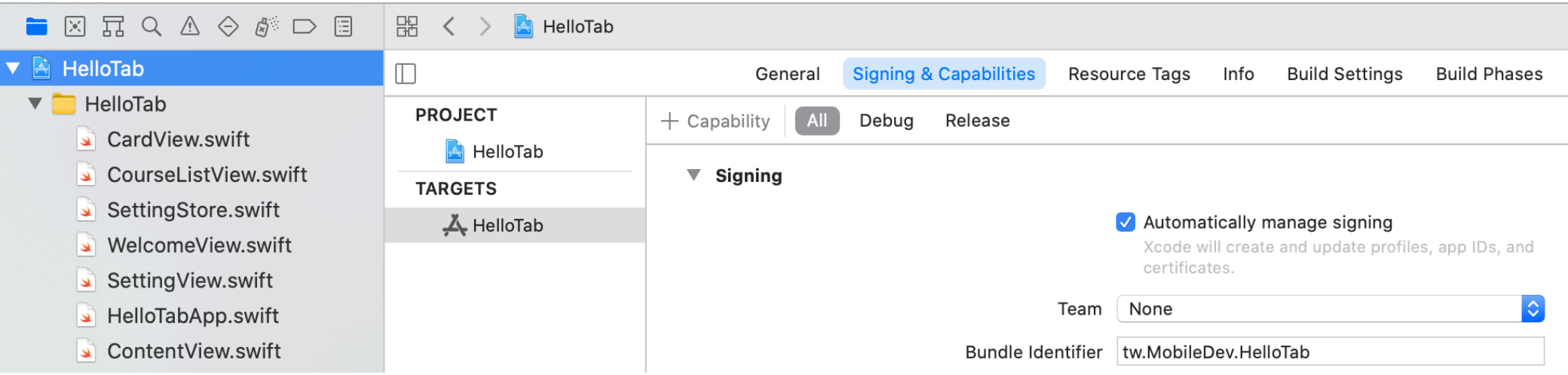
Cienpies Design

實機測試注意事項

- Bundle ID
  - 專案唯一識別
- 自動簽署 Automatic Signing
  - 請求開發者憑證
  - 建立App ID
  - 建立App 描述檔

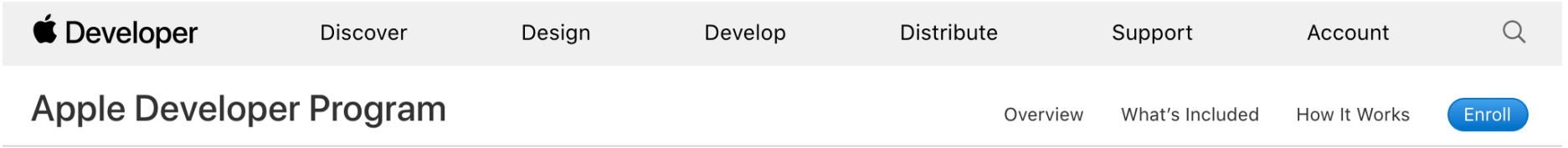
Automatic Signing

- 專案->Targets->Signing & Capabilities
- 勾選Automatically manage signing
- 選擇團隊
- 如無法自動，則須自行選擇Provision Profile



Developer Program

- 個人帳號 or 組織帳號



What You Need To Enroll

Enrolling as an Individual

If you are an individual or sole proprietor/single person business, get started by signing in with your Apple ID with two-factor authentication turned on. You'll need to provide basic personal information, including your legal name and address.

Enrolling as an Organization

If you're enrolling your organization, you'll need an Apple ID with two-factor authentication turned on, as well as the following to get started:

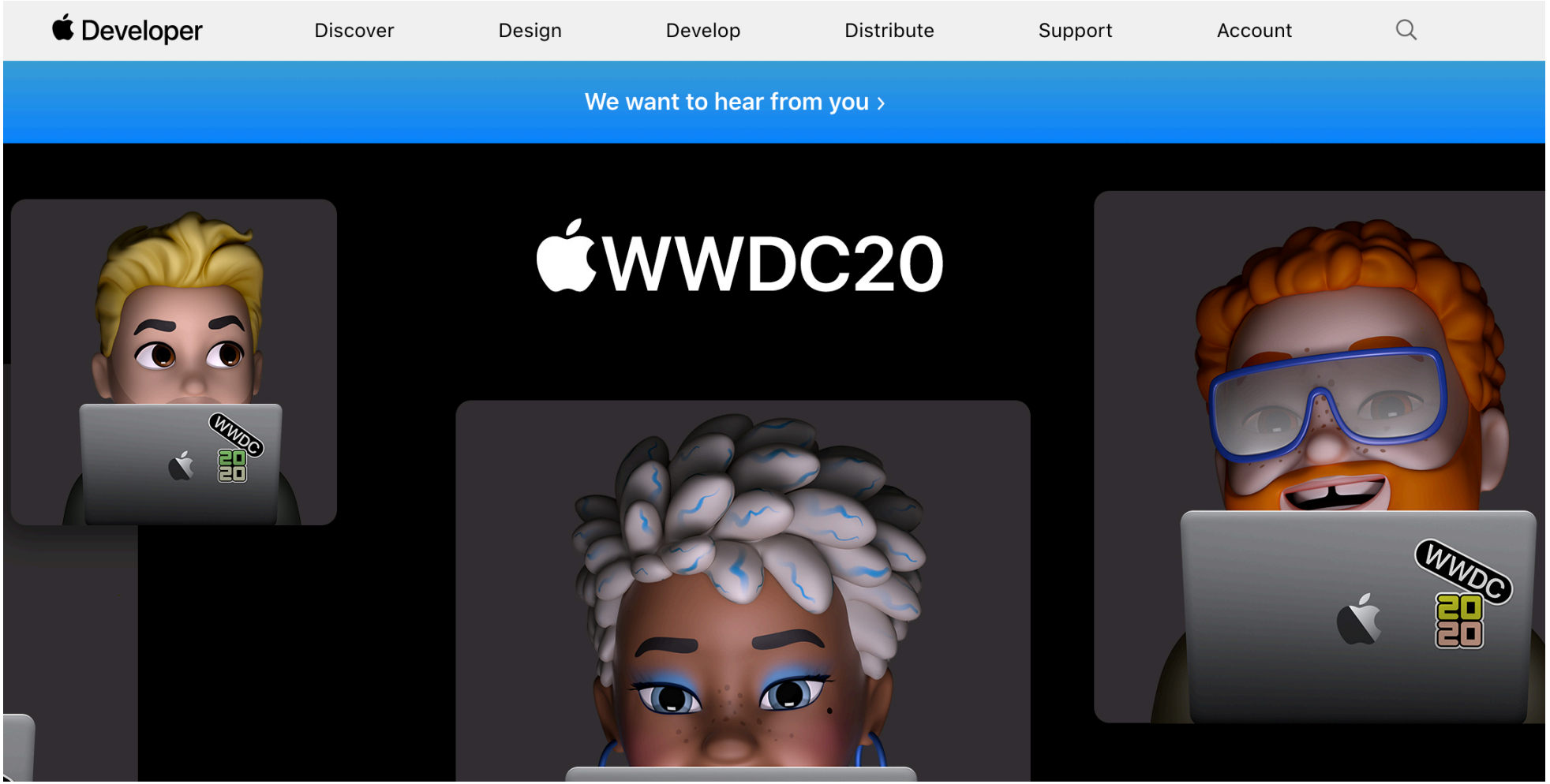
https://developer.apple.com/programs/enroll/

其他選擇

- Apple Developer Enterprise Program
  - 企業內部應用
  - 不想對外公開上架
- MFi Program
  - 開發周邊硬體配件
  - 須透過此方式取得認證Logo
- iOS Developer University Program
  - 課堂教學使用

開發者官網

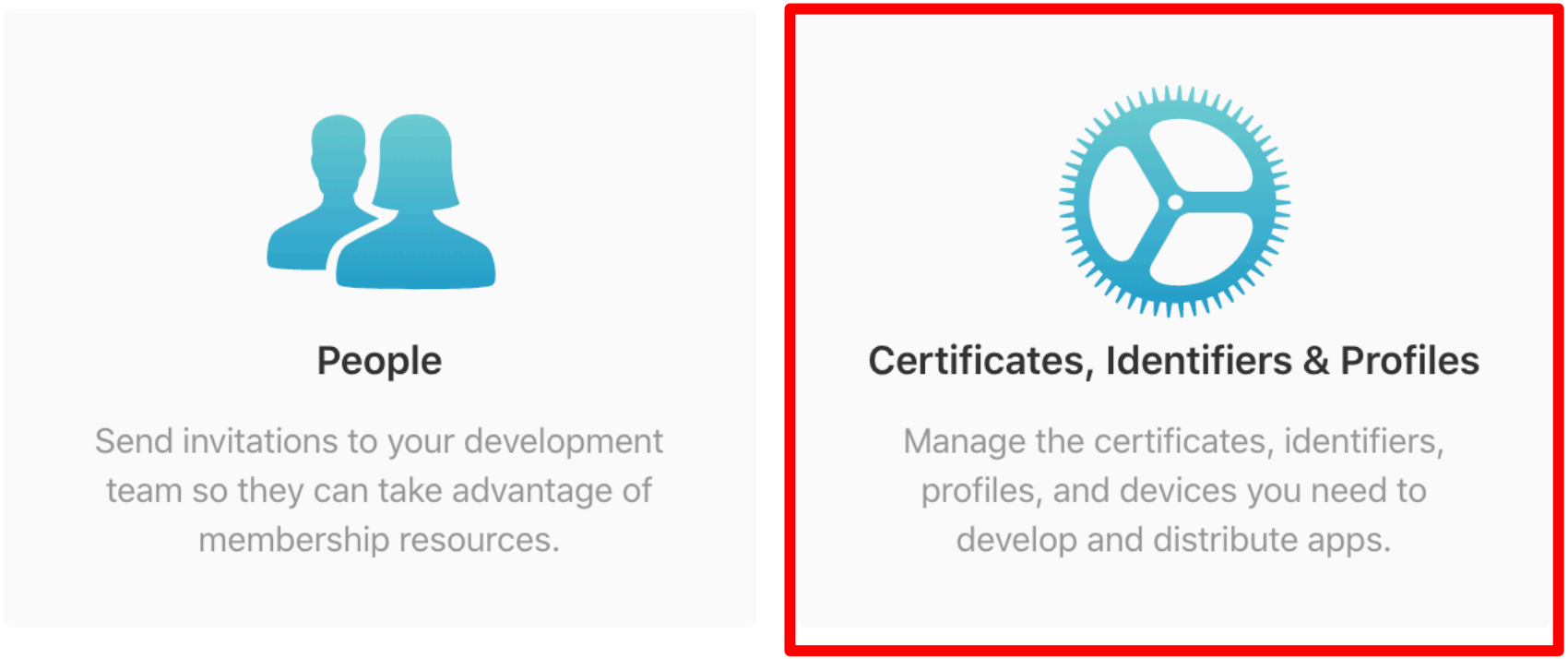
- 右上角Account登入



https://developer.apple.com/

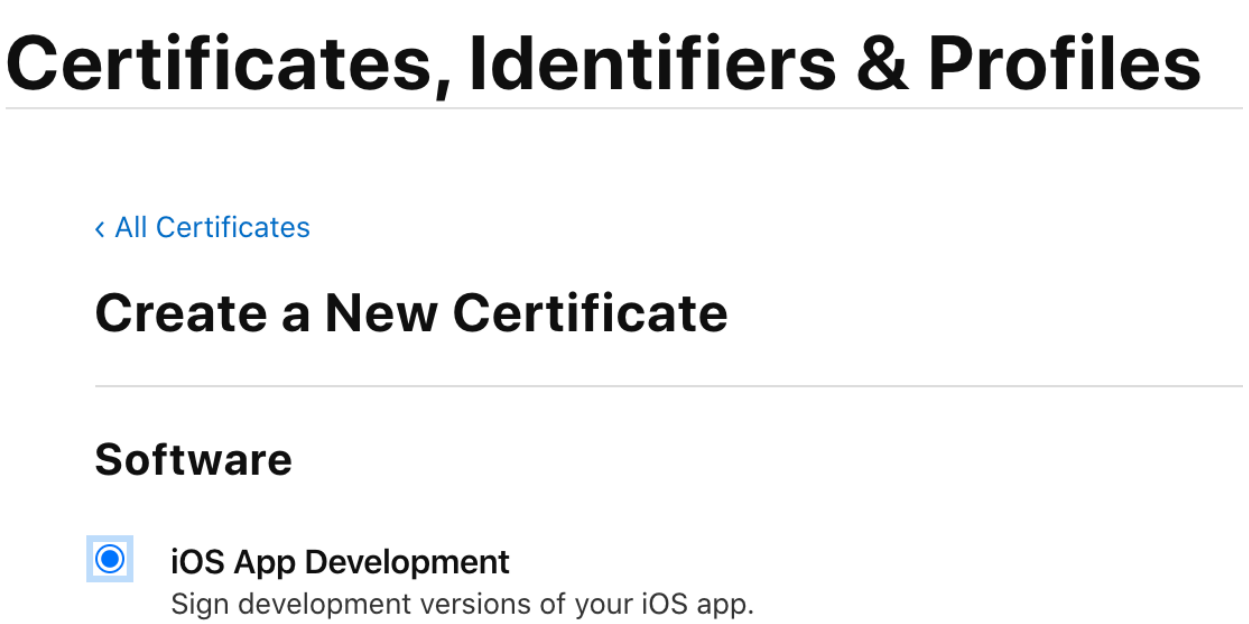
憑證取得

- Certificates, Identifiers & Profiles



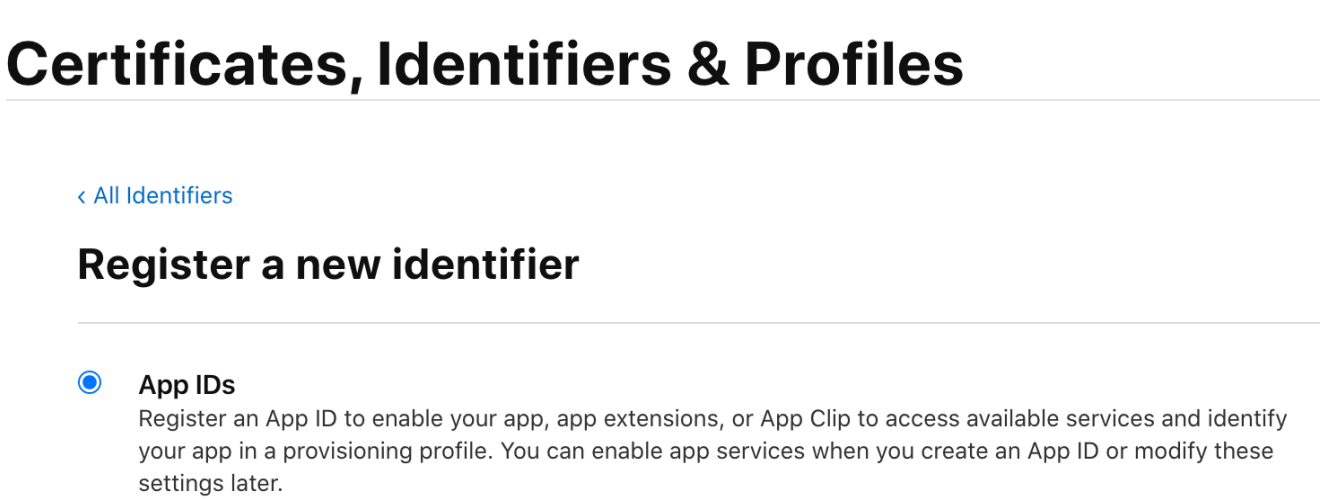
如無法自動使用實機

- 可以在這裡設定這幾個項目
  - Certificates
  - Identifiers
  - Devices
  - Profiles
  - Keys



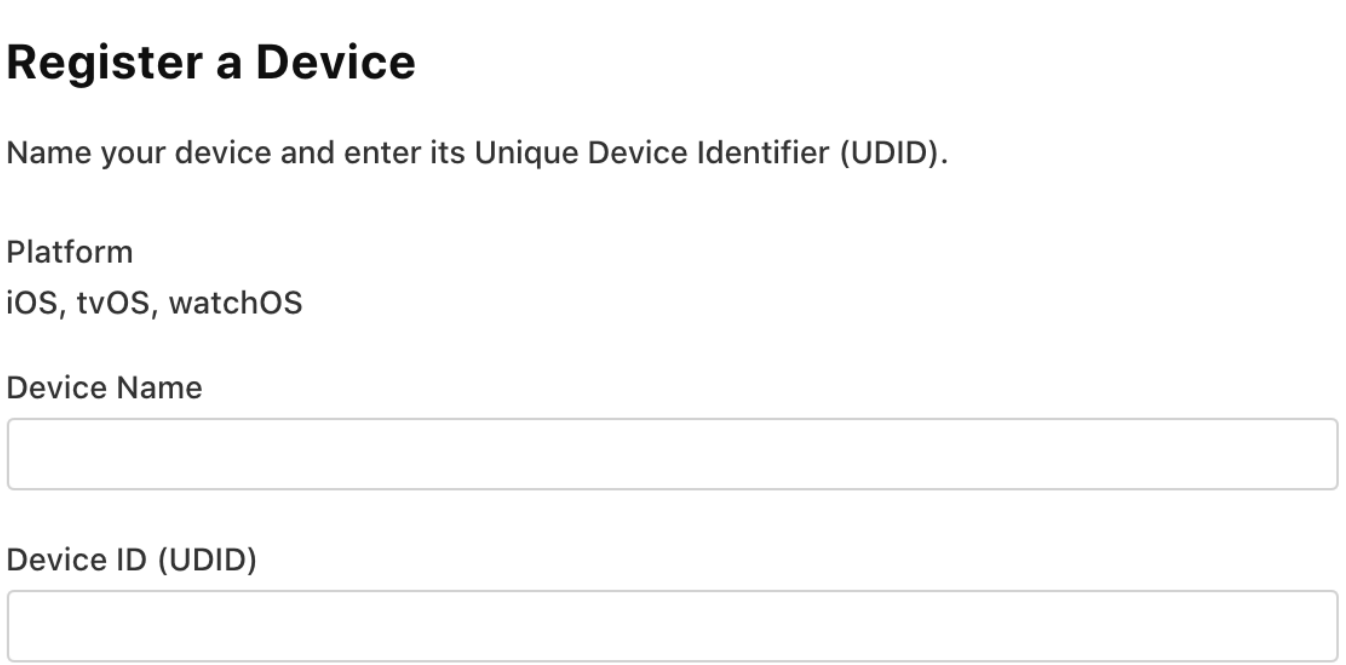
如無法自動使用實機

- 可以在這裡設定這幾個項目
  - Certificates
  - Identifiers
  - Devices
  - Profiles
  - Keys



如無法自動使用實機

- 可以在這裡設定這幾個項目
  - Certificates
  - Identifiers
  - Devices
  - Profiles
  - Keys

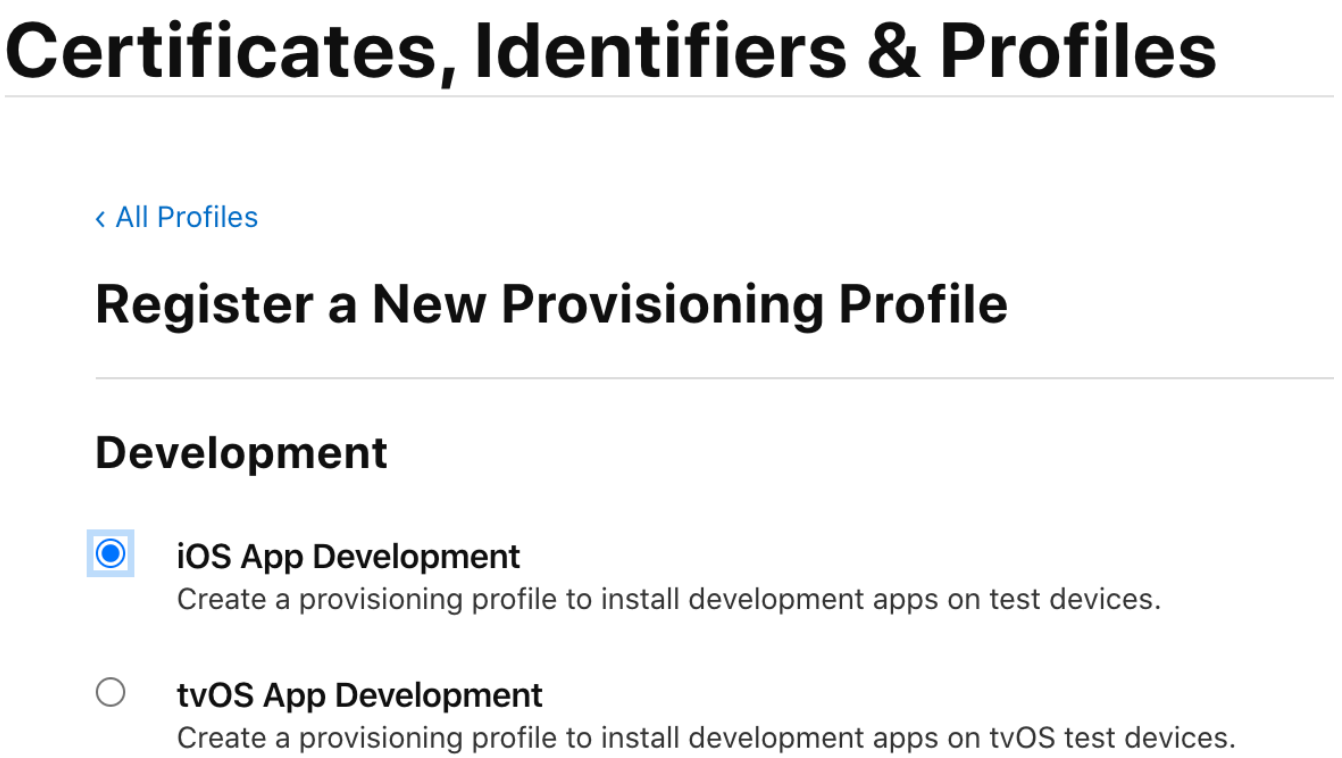


如何取得實機的UDID?

- 打開iTunes(音樂)
- 將iPhone接上電腦
- 點擊裝置
  - 上方裝置圖示旁小字
  - 點擊可切換顯示
    - 電話號碼/IMEI/ICCID
    - 型號/容量/電量
    - 序號/UDID/機型
  - 右鍵可拷貝

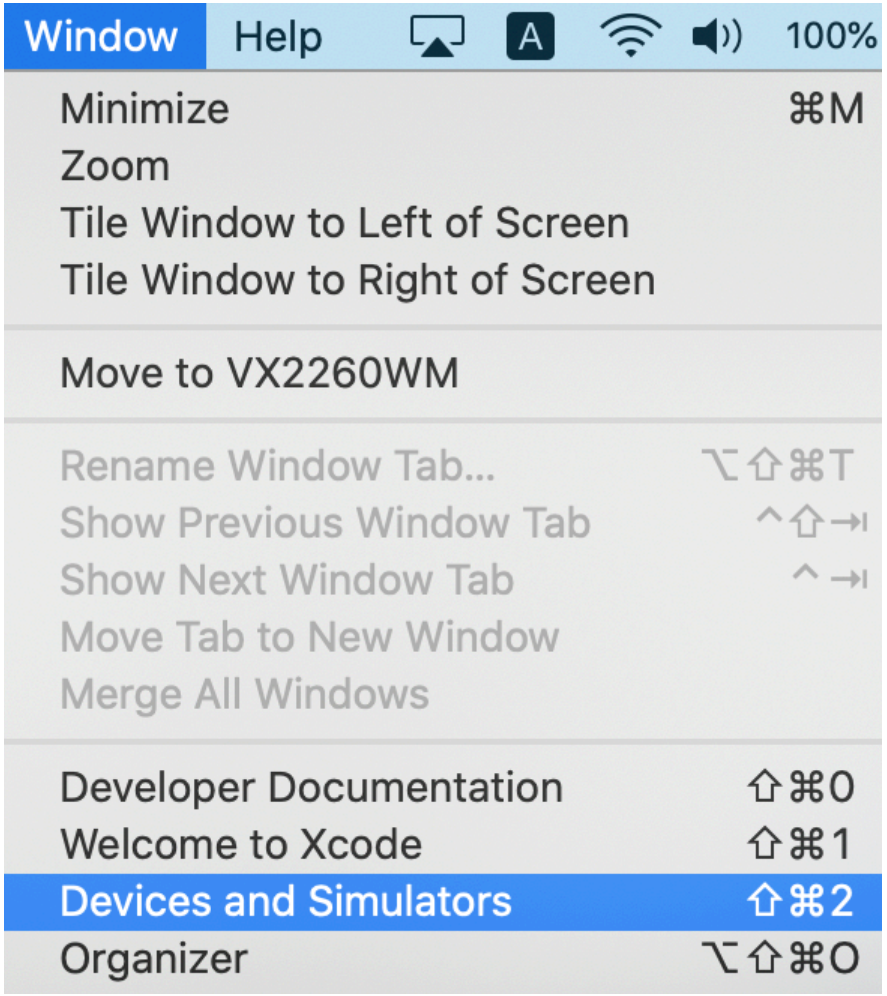
如無法自動使用實機

- 可以在這裡設定這幾個項目
  - Certificates
  - Identifiers
  - Devices
  - Profiles
  - Keys



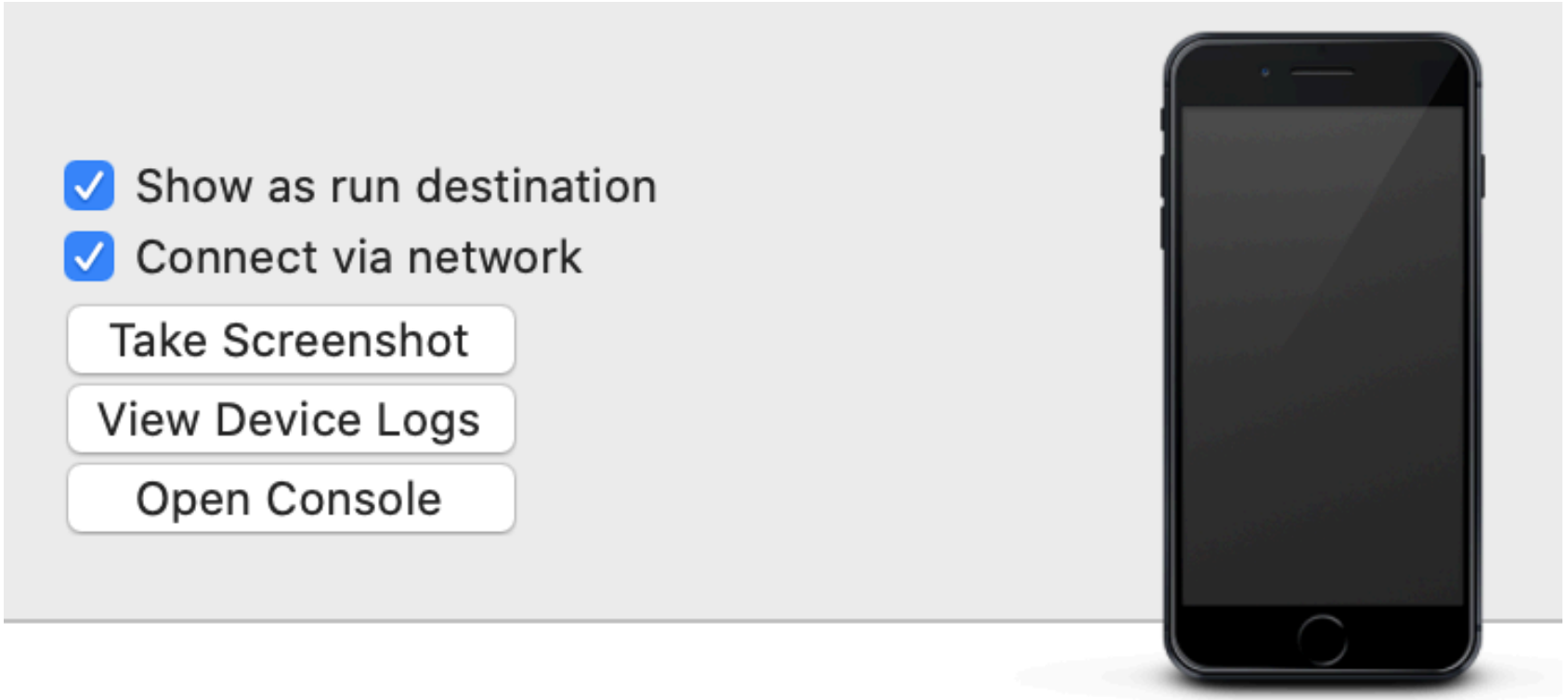
裝置設定

- Xcode -> Window -> Device and Simulators
- 第一次在實機測試
  - 設定->一般->裝置管理
  - 確認信任



無線測試

- 勾選 Connect via network



App 上架前

- 徹底測試
  - Preview -> 模擬器 -> 實機
  - 有網路/沒網路、有資料/沒資料
- 基本要求
  - 不會閃退
  - 可順利執行，沒有Bug
  - 內容與送審描述相符
  - 無違反善良風俗
  - 不可有Apple相關商標文字/圖案

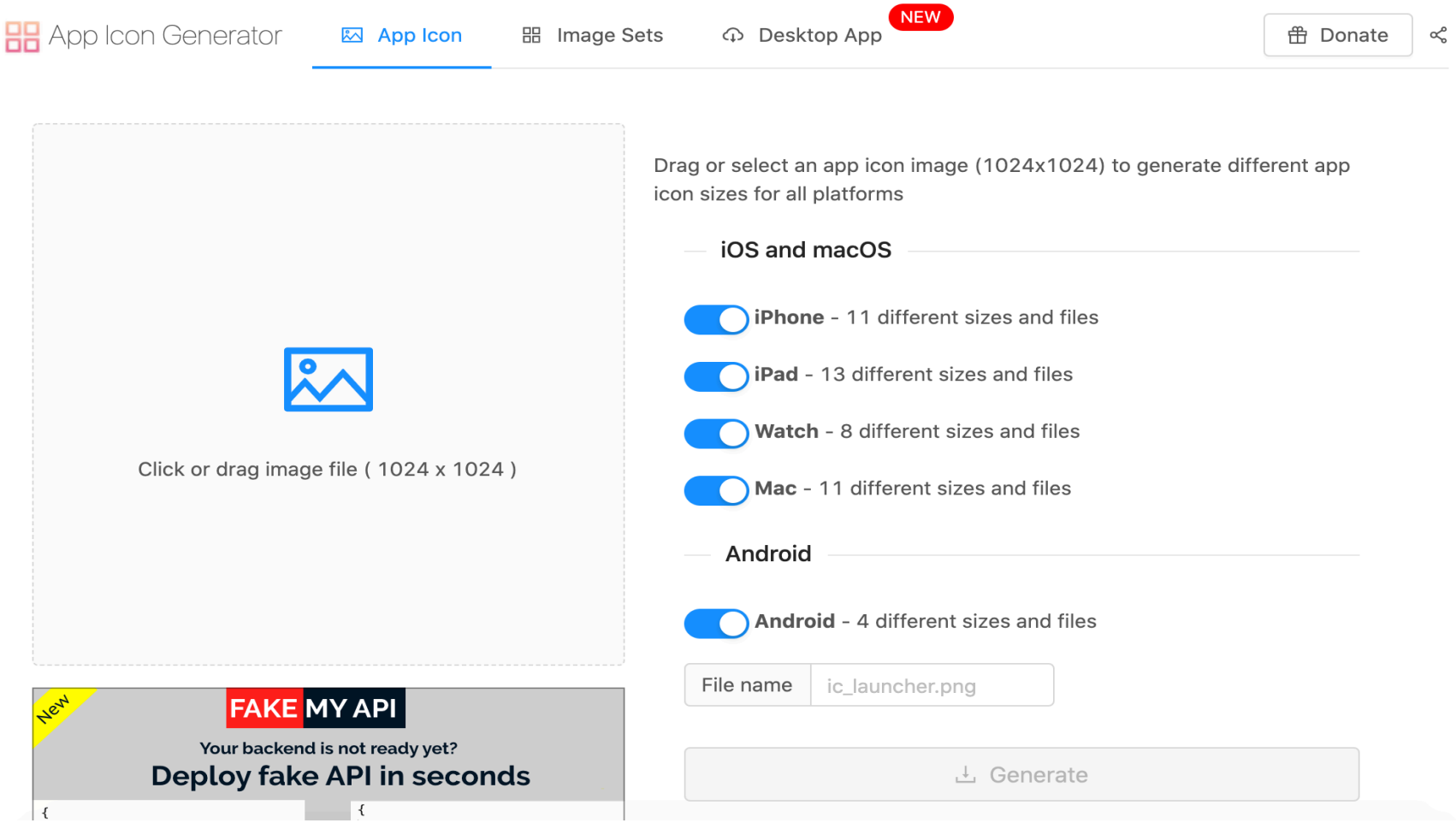
App上架前

- 符合一般iPhone使用者操作預期
  - 有使用者指引說明更好
- App中的每個連結都確定可順利執行有內容
- 沒有Placeholder、Beta字樣等開發中素材

Olivier Le Moal / Getty Images

產生上架要用的圖片

- App Icon以及螢幕截圖

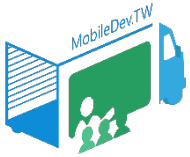


https://appicon.co/

App Store Review Guidelines

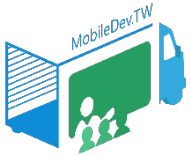
- Apple寫在最前面的提醒事項
  - 很多小孩子在用iPhone，請謹記在心
  - App Store是公開的大市場，如果只是要寫一些給家人朋友看的東西，有別的方式可以給他們
  - 我們心中有一把尺，你送過來我就會知道
  - 別想作弊，我們會把你跟App剔除
  - 這份文件持續會更新...

https://developer.apple.com/app-store/review/guidelines/



先確認的事項

- 不會閃退、沒有Bug
- 資訊完整
- 有留聯絡資訊給Apple
- 如果要登入，有留測試帳號密碼
- 如果需要搭配硬體，也請提供
- 確認你的後端持續運作中
- 多一點指引文件、功能解釋文件
- 符合普遍iOS裝置操作原則



安全性

- 沒有不當內容
- 對於使用者產生內容有篩選機制
  - 如何避免使用者大頭照上傳色情照片?
- 不會對使用者造成傷害
  - 錯誤的醫療數據
- 保護使用者隱私

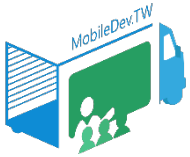
效能

- 完成測試，為發行版
- 不可以有什麼隱藏版功能
- 也不可以宣稱你沒有的功能
- 如有內部購買，請明確讓使用者知道
- 請用真實截圖
- 挑選適當分類
- 名稱獨特而不怪異
  - 不可包含價格、描述等不適當內容

https://developer.apple.com/app-store/categories/

設計

- 創新
- 有功能
- 以下這些很多了，不要送過來
  - fart, burp, flashlight, fortune telling, dating, and Kama Sutra
- 只是披著APP皮的網站也會被退

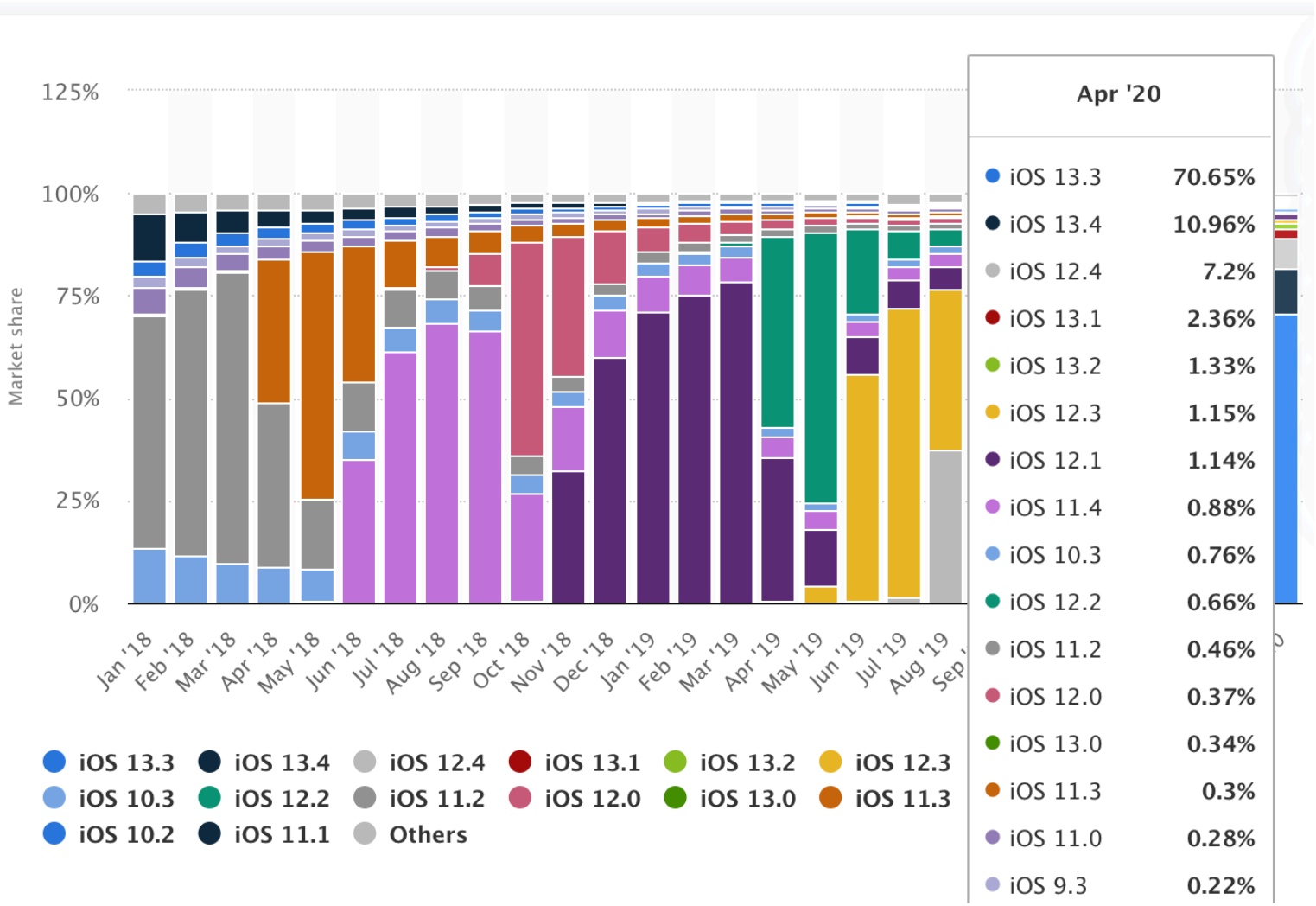


法律

- 隱私權聲明
- 只跟使用者要必要的資料
- 提供拒絕權限的使用者替代方案
  - 不提供定位存取的使用者可手動輸入地址

iOS OS Version Market Share

- iOS 13 : Over 85.64% (2020/04/20)



statista.com

Objective-C (1980 ~ 2016)

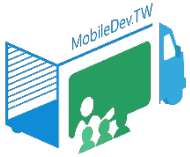
- 1980年代
  - Stepstone 公司的Brad Cox和Tom Love發明Objective-C
- 1988
  - Steve Jobs離開蘋果公司後成立 NeXT Computer 公司，並買下 Objective-C 語言的授權
- 1996
  - 蘋果公司宣布收購 NeXT Software 公司，NEXTSTEP/OPENSTEP環境成為蘋果作業系統下一個主要發行版本OS X的基礎，名為Cocoa
- 2006
  - 2006年7月蘋果全球開發者會議中，Apple宣布了 Objective-C 2.0
- 2010
  - 2010年7月，蘋果開發者工具部門總監克里斯·拉特納開始著手 Swift 程式語言的設計工作，一年時間完成基本架構
- 2014
  - 歷經4年的開發期，2014年6月對外發表

iOS Swift (2014 ~)

Version	Release Date
Swift 1.0	**September 9, 2014**
Swift 1.1	October 22, 2014
Swift 1.2	April 8, 2015
Swift 2.0	**September 21, 2015**
Swift 2.1	October 20, 2015
Swift 2.2	March 21, 2016

Version	Release Date
Swift 3.0	**September 13, 2016**
Swift 3.1	March 27, 2017
Swift 4.0	**September 19, 2017**
Swift 4.1	March 29, 2018
Swift 4.2	September 17, 2018

Version	Release Date
Swift 5.0	**March 25, 2019**
Swift 5.0.1	April 18, 2019
Swift 5.1	September 20, 2019
Swift 5.2	March 24, 2020
Swift 5.3	(March 25 , 2020 Announced)



ObjC -> Swift

- 冗長 -> 簡潔
- .h, .m -> .swift