第六周
網路程式
Part 1

shenfive@gmail.com
申潤五

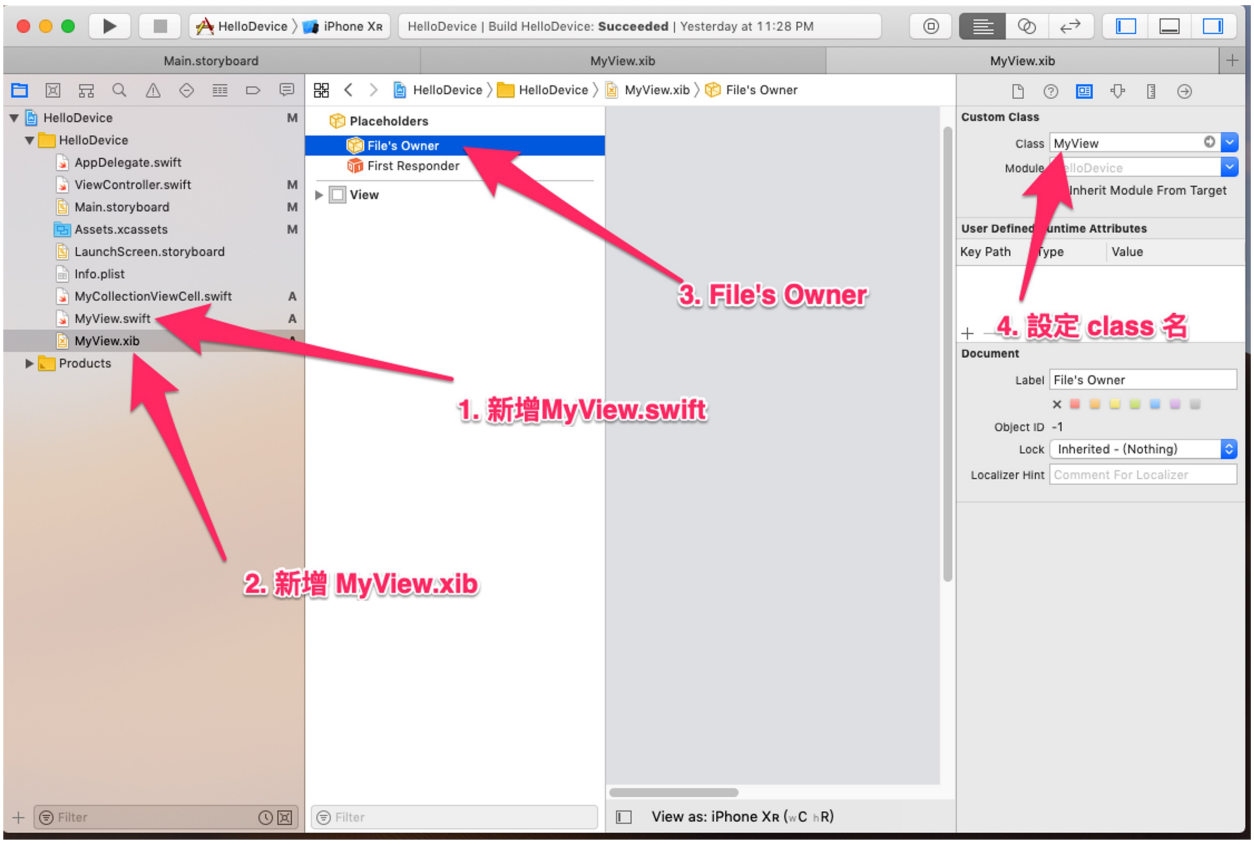
今天的目標

1. 使用 XIB 自訂 UIView
2. Objective-C 與 Switf 混用
3. WKWebView 建立網站
4. 圖片的使用與下載圖片
5. 呼叫網路 API
6. Cocoa pods

自訂 Xib 的 UIView

1. 開發元件而不是開發應用
2. 多人開發分工
3. 如果沒有 Xib (UI 檔)，就只是一個 UIView class 檔案了

1. 新增一個自訂 cocoa class 的 Swift 檔，繼承 UIView，如 MyView.swift
2. 新增一個 xib (view) 檔，建議使用與物件名相同的名字如 MyView.xib
3. 修改 MyView.xib 中的 FilesOwner 設定對應的 Class Name



設定 MyView

```
class MyView: UIView {
    var view:UIView!
    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }
    required init?(coder aDecoder: NSCoder) { //一定要寫的建構器
        super.init(coder: aDecoder)
        setup()
    }
    func setup() {
        view = loadViewFromNib()
        view.frame = bounds
        view.autoresizingMask = [ UIView.AutoresizingMask.flexibleWidth,
            UIView.AutoresizingMask.flexibleHeight ] // 一些歷史原因，必需要上的設定
        addSubview(view) //把 由 UI檔作出來的 View 加入成自己的 SubView
    }

    func loadViewFromNib() -> UIView { //由 Xib 檔，製作一個UIView
        let nib = UINib(nibName: "MyView", bundle: nil )
        let view = nib.instantiate(withOwner: self, options: nil)[0] as! UIView
        return view
    }
}
```

自訂建構器

Swift 與 Objective-C 混用

為什麼要混用？

因為有人只會用 Objective-C

因為有很多寫好的 Objectiv-C

因為要用底層 C 寫的程式庫

1. 把 Objective-C 用各種方法加入專案，例入：拉進去
2. 建立 Bridging-Header.h 檔
3. 使用 Objective-C 的 Class 與方法
4. 在Objective-C 中 import [App]-Swift.h

在使用網路之前

一旦使用網路後，若網路沒有信息，必需先處理

不過最好是有網路再使用

要偵測網路也不太難，但要懂一些網路知識，如Wi-Fi，3G/4G，Socket.....

若不想研究，就用 Apple 幫我們寫的類別 Reachability

https://developer.apple.com/library/archive/samplecode/Reachability/Introduction/Intro.html

可惜，Apple 是用 Objective-C 寫的，所以我們就要學習如何和 Swift 混用

```
#import "Reachability.h"
```

Header 檔 Sample

```
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        print(getNetworkStatus())
    }


    func getNetworkStatus()->Bool{
        if Reachability(hostName:
"www.apple.com")?.currentReachabilityStatus().rawValue == 0 {
            return false
        }else{
            return true
        }
    }
}
```

使用 Reachability 取得網路的 Status

WKWebview

1. WKWebview 其實就是瀏覽器元件
2. 早期版本的元件是 UIWebView，建議沒有特別理由就以 WKWebview 取代，不要使用
3. WK 代表 WebKit 開源 Project 即 Safari 所使用的核心，Chrome/Opera 的核心 也是 WebKit 的分支

記得 import Webkit，要在專案層引入

```
import WebKit

class ViewController: UIViewController {

    @IBOutlet var wkWebView: WKWebView!

    override func viewDidLoad() {
        super.viewDidLoad()
        if let url = URL(string: "https://www.ichih.com/"){
            let request = URLRequest(url: url)
            wkWebView.load(request)
        }
    }
}
```

開啟一個頁面

WKWebView的基本功能與屬性

1. loadRequest()
加載請求

2. goBack()
網頁到上一頁

3. goForward()
網頁到下一頁

4. reload()
網頁重新加載

5. stopLoading()
網頁停止加載

6. title
網頁標題

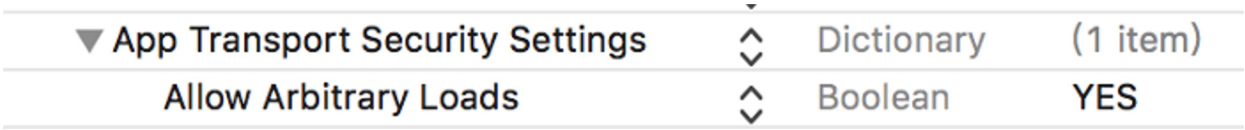
7. canGoBack
網頁是否能夠後退（回上一頁）

8. canGoForward
網頁是否能夠前進（回下一頁）

9. estimatedProgress
網頁加載中的當前進度

用他們來實作一個可以用的瀏覽器？

—



若要加載的網頁網址為 http 開頭，必須在 Info.plist 內加入

才能正確地加載網頁。

額外提醒

—

下載一張圖片

1. 先下載成 Data
2. 再用 UIImage

```
let imageData = try Data(contentsOf: imageURL)

imageView.image = UIImage(data: imageData)
```

但是，有可能出錯，要使用 try Catch

---

try catch

凡是 method 說明後面帶 throws 的都有可能出錯，此時就必需使用 try catch

catch 會回傳一種或多種 error ，如果只有一種，可省略一切參數

```
import UIKit

class ViewController: UIViewController {
    @IBOutlet var imageView: UIImageView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let imageAddress =
"https://shop.r10s.jp/aikimania/cabinet/new/sw-
1758_sp.jpg"
        if let imageURL = URL(string: imageAddress){
            do{
                let imageData = try Data(contentsOf:
imageURL)
                imageView.image = UIImage(data: imageData)
            }catch{
                print(error.localizedDescription)
            }
        }
    }
}
```

下載一張圖

更好的下載圖片方式 GCD

由於畫面有許多圖片組成之前的做法會等網路上所有的圖片都下載，才顯示圖片，一般來說，在網路順暢時，需要個零點幾到數秒，這聽起來沒有什麼，但若是用在如 TableView，一上滑就要等個半秒一秒，使用者就會感到卡，這也是手感不佳的由來，一般來說，我們就會用到多線程來下載圖片，所以就要用 DispatchQueue 物件來執行程式

GCD DispatchQueue

```
DispatchQueue.global().async {
        // 要執行的程式
}
```

這樣 {} 中間的程式就會在背景執行

主執行緒

```
DispatchQueue.main.async {
        // 要執行的程式
}
```

凡是與畫面有關的，都必需寫在主 Queue 中執行

```
        let start = Date().timeIntervalSince1970
        super.viewDidLoad()

            let imageAddress =
"https://img.muah.blog/uploads/20180516133305_82.jpg"
            if let imageURL = URL(string: imageAddress){
                DispatchQueue.global().async {
                    do{
                        let imageData = try Data(contentsOf: imageURL)
                        DispatchQueue.main.async {
                            self.imageView.image = UIImage(data: imageData)
                            print(Date().timeIntervalSince1970 - start)
                        }
                    }catch{
                        print(error.localizedDescription)
                    }
                }
            }
        print(Date().timeIntervalSince1970 - start)
```

理想的載圖程序

設定圖片其他項目

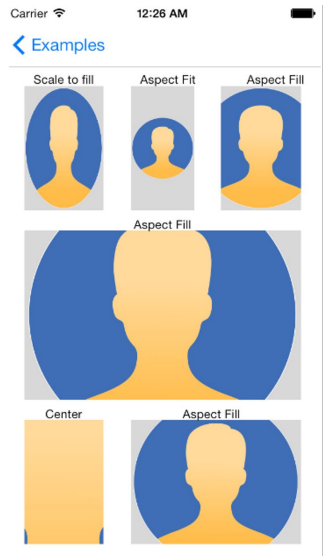
圖片常用設定項目

填充模式

圓角，圓形

外框邊線

陰影



ContentMode

clipsToBounds

clipsToBounds 畫面是否會延伸到邊框外

陰影一定會延伸到邊框外，所以無法開啟

若要同時使用邊框與陰影，就要外面再來一層 View

或多層 Layer 解決問題

```
imageView.clipsToBounds = true
 imageView.layer.cornerRadius = 15        //圓角
 imageView.layer.borderColor = UIColor.red.cgColor //邊線色
 imageView.layer.borderWidth = 5          //邊線
```

外框與圓角

—

```
imageView.clipsToBounds = false
imageView.layer.shadowRadius = 20        //陰影
imageView.layer.shadowOpacity = 0.6;
imageView.layer.shadowColor = UIColor.gray.cgColor
imageView.layer.shadowOffset = CGSize(width: 10, height: 10)
```

陰影

—

```
containerView.clipsToBounds = false
containerView.backgroundColor = UIColor.clear
containerView.layer.shadowRadius = 20        //陰影
containerView.layer.shadowOpacity = 0.6;
containerView.layer.shadowColor = UIColor.gray.cgColor
containerView.layer.shadowOffset = CGSize(width: 10, height: 10)

imageView.clipsToBounds = true
imageView.layer.cornerRadius = imageView.frame.width / 2        //圓角
imageView.layer.borderColor = UIColor.red.cgColor //邊線色
imageView.layer.borderWidth = 5        //邊線
```

同時外框與陰影

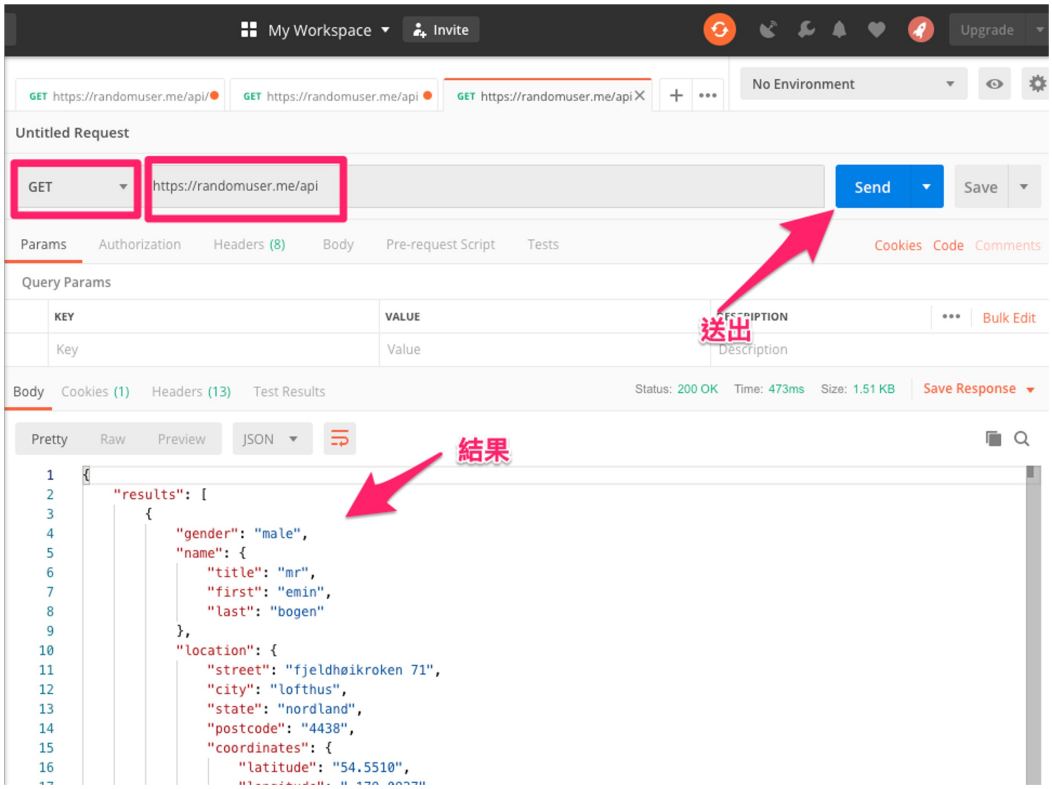
使用 Post Man 來測 API

安裝 POST MAN

依指示 測 API

方法：GET， https://randomuser.me/api

沒有別的參數



PostMan

我們使用的 API

https://randomapi.com/

應用程式介面 (Application Programming Interface)，在現行網際網路發達的時代，因應網路平台的多樣性的發展，許多大型網站平台逐漸開放物件功能和資訊與其它網站進行串接、共享，不僅能接觸整合多元資訊，更可以提升跨平台使用者的友善性，達到平台多元發展的市占性，而大型平台將這些開放分享的資訊功能開發打包成可物件連結的應用程式介面，提供其它開發者進行應用串接，便是API的核心架構。

我們使用的是 Restful 風格的 API

我們取得的資料 ~ JSON

JSON（JavaScript Object Notation，JavaScript物件表示法，讀作「Jason」）是一種由道格拉斯·克羅克福特構想和設計、輕量級的資料交換語言，該語言以易於讓人閱讀的文字為基礎，用來傳輸由屬性值或者序列性的值組成的資料物件。儘管JSON是JavaScript的一個子集，但JSON是獨立於語言的文字格式，並且採用了類似於C語言家族的一些習慣。

線上解析：

http://json.parser.online.fr/

如何取得資料？

1. 可如同我們取得圖片一樣，由下載點取得DATA，也就是Data(contentofURL:.....但其功能較為底層，不容易對應到 API 的各項參數設定
2. 可以使用第三方套件，本課程使用Alamofire

如何解析 JSON 資料？

1. 我們可以用內建的 JSONDecoder 或 JSONSerialization，但建議是不要用，因為很難用.....是真的非常難用
2. 建議使用第三方的 Swifty JSON 來解析

三種第三方套件管理工具

- Cocoa pods
  - 流傳最廣的套件管理工具，Google 主流支持，大多數套件階有支援
  - 操作程序較麻煩，會改變專案結構檔 (改用Work Space 檔)
- Swift Package Manager SPM
  - Xcode 11 開始整合提供使用者視覺化介面，操作簡單易懂
    (第一次發布為 2016 年 Xcode 8 但只有 Command Line 工具，沒有使用者介面，大多數 Package 也未支援)
  - Apple 官方推薦，官方軟硬體變更第一時間支援 (例如 M1 的完整支援)
  - 去中心化的專案管理工具，必需事先知道原始碼的 git 位置
  - 支持的套件較少 (相對於 Cocoa pod)，問題陸續解決中
- Carthage

Cocoa pods

Google 特別為它拍了一個短劇

https://www.youtube.com/watch?v=iEAjvNRdZa0

這是一個好用的，第三方管理工具，用就對了

安裝與執行 CocoaPods，包括安裝 CocoaPods 一共有五個動作：

一、 安裝 CocoaPods
二、 初始化專案的 pod
三、 修改 Podfile
四、 pod install
五、 使用新的設定檔 .xcworkspace 開啟專案

說明如下：

一、 安裝 CocoaPods:

這需要用到Ruby，除非你的 Mac 有故意移除，或系統太舊，不然都應該已安裝好 Ruby 才對，所以 macOS 10.15 為例，只要一行指令就可以完成安裝了，由於是線上安裝，所以網際網路一定要是通的狀況下，執行【終端機】（在 finder 或 Spot Light 中搜尋一下【終端機】，或在應用程式上找一下就可以找到），然後打以下指令，並輸入系統密碼，就可以安裝了

sudo gem install cocoapods

安裝 Cocoapods

二、 初始化專案 pod

一樣要用命令例，若你不熟命令例，請把專案建在桌面上。其實也只有一行指令，必需移動到專案資料夾中，再執行

pod init

以上範例為專案在桌面上的Sample 的資料匣時的狀況，如果目錄中，出現一個 "Podfile" 的檔案，就表示設定成功了它就是這個專案的 Pod設定檔

三、 修改 Podfile

這個動作是要修改 Podfile，它是一個文字檔，若你很熟 vi 或其他的文編輯工具，就可以直接打開它，否則就像我一樣用 Xcode 打開它，同樣用指令

open -a Xcode Podfile

未來在開發各個子功能項目時，可能要在同一個地方，加上其他用到的 SDK，完整應該打什麼，可以到 cocoapod 網站去找

初始化 Cocoapods

三、 修改 Podfile

```
#範例
#Podfile

target 'downloadImage' do

  pod 'Kingfisher'
  use_frameworks!
end
```

四、執行 pod install

回到命令例，執行 pod install 指令

設定 Podfile 並安裝 Package

使用 Cocoa pod 專案

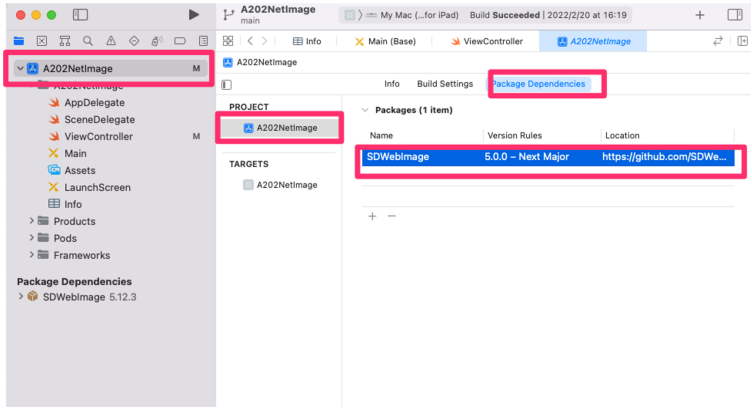
- 開啟時，使用白色的 work space 檔，取代 project 檔
- work space 檔會包括兩個 project ，一個是你的專案，另一個會包括所有的第三方檔與 cocoa pod 設定檔
- 若只開啟原來的 pod 檔，將無法有效的使用第三方的資料庫

使用 Swift Package Manager

1. 確認 Package 的 Git 網址
2. Xcode 開啟要用的專案
3. 下拉選單 [File] -> [Add Packages]
4. 在搜尋欄輸入 SPM 的 GitHub 網址
5. 選定版本
6. 按下 [Add Pagkages]
7. 完成

管理 SPM

1. 下拉選單 [File] -> [Packages] -> 各種管理
2. 在專案檔下，選擇 Package Dependencies -> 刪除或調整



Kingfisher

從網路上下載圖片到 App 顯示一直都是網路應用 App 中一個重要的功能項目。但有可能因為圖片重複下載而造成使用者流量問題或是網路環境不佳而發生些「意外」，於是善用快取(cache)的方式將圖片保留在記憶體或是儲存空間中便是一種解決之道。但對於一些開發者來說快取的應用並不那麼上手，好在有 Kingfisher 幫忙解決這樣問題。

Kingfisher 就會從網址下載圖片顯示，同時暫存於記憶體快取及硬碟快取之中，當這張圖需要在此顯示時就會從快取中取出而不會重新下載。

先用 Kingfisher 試一下

```
#Podfile

target 'downloadImage' do

  pod 'Kingfisher'
  use_frameworks!
end
```

```
import UIKit
import Kingfisher
class ViewController: UIViewController {
    @IBOutlet var imageView: UIImageView!
    @IBOutlet var containerView: UIView!
    override func viewDidLoad() {
        super.viewDidLoad()
        let start = Date().timeIntervalSince1970
        let imageAddress =
"https://img.muah.blog/uploads/20180516133305_82
.jpg"
        DispatchQueue.global().async {
            self.imageView.kf.setImage(with: URL(string:
imageAddress))
            print(Date().timeIntervalSince1970 - start)
        }
        print(Date().timeIntervalSince1970 - start)
    }
}
```

若一切準備好

1. 建立 Call API Model
2. 處理多線程 I/O
3. 分析 JSON
4. 做出畫面