序言

PREFACE

打從以 Objective C 程式語言開始撰寫 iOS App 開始，繼而使用 Swift 程式語言，我的感覺是 Apple 公司無不替開發者著想，以親和力相當高的界面和強大的軟體架構來開發 iOS App，因此，有相當多的開發者也紛紛的加入開發的行列，期望有好的作品置放於 Apple store，有朝一日成為大富翁。

本書基於能夠讓有志開發 iOS App 的讀者，能夠在短時間撰寫自已的 iOS App，因此本書可以說是筆者另一拙著:「學會 Swift 4 的 21 堂課」之續集，當您有了 Swift 的基本知識後，進而探討如何撰寫有關 iOS App 的相關元件，然後整合一些元件加以實作屬於自已的 iOS App。

本書共分二部份，第一部份是 iOS App 相關元件的實作。此部分主要在探討建立 iOS App 會用及的相關 UI，再以此為基礎加以整合應用，這將在 Part II 加以討論之。第二部份是 iOS App 實作。此部分包含兩個章節，分別是實作提醒事項 App 和天氣 App，將第一部分所論及的一些 UI 元件做整合，期許讀者對製作 iOS 的 App 有一初步的概念和認識。

努力學習加上毅力，相信成功就離你不遠，親愛的讀者們讓我們一起共勉，明天一定會更好。

mjtsai168@gmail.com

1

CHAPTER

Xcode 介面介紹

1.1 認識 Xcode

Xcode 是在 Mac 電腦下開發應用程式的主要工具，就如同 Microsoft 的 Visual Studio。

Xcode 可以幫助您快速開發一個 iOS 的應用程式。而且 iPhone 跟 iPad 也都可以使用 Xcode Xcode 版來開發。早期的版本，程式碼是在 Xcode 編輯，介面建置則需透過 Interface Builder 編輯，現在最新的版本 Xcode 中，已經將 Interface Builder 整合到 Xcode 內，因此開發上會更方便許多。

您可以在根目錄下的 Developer/Applications 找到 Xcode.app，將它拖曳至 mac 底下的 Dock 列，並將其固定在 Dock 列上，以方便之後開啟 Xcode。

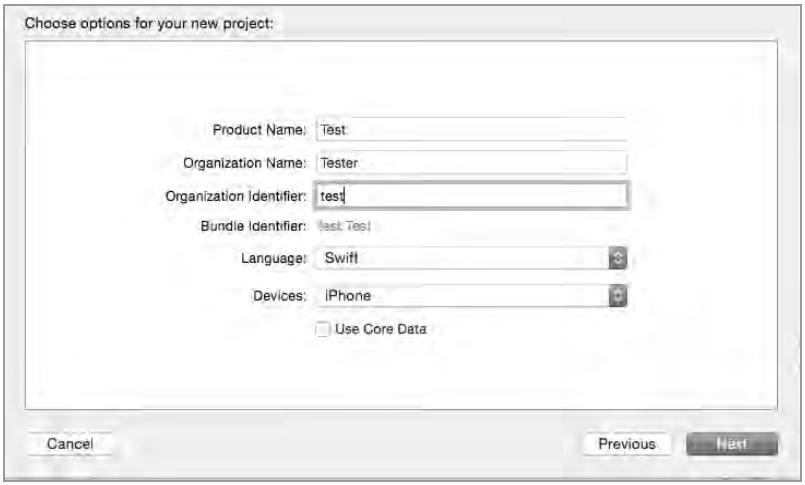
當您一開啟 Xcode，會出現以下的畫面。

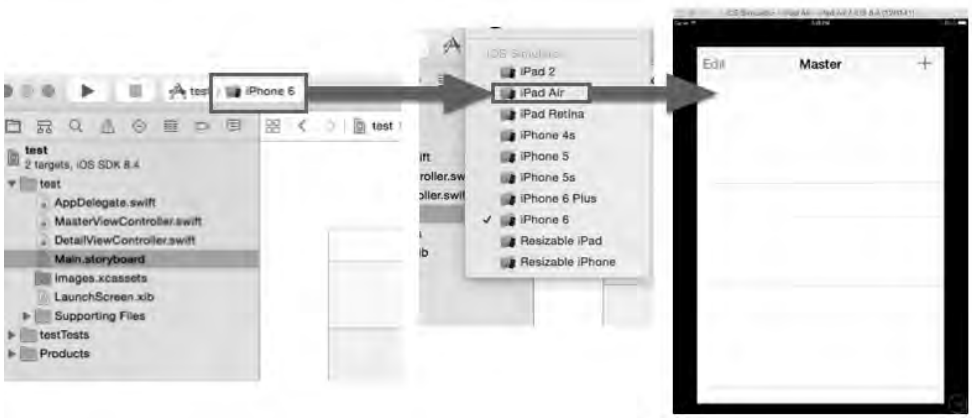
如上圖，如果您之前已經有開啟過的專案，在上圖的右方會出現您之前開過的專案歷史紀錄，這可幫助您快速開啟你之前的專案，如果你要建立新專案，可以點選左方的“Create a new Xcode project”來建立一個新的專案。

左方選擇 iOS 內的 Application，右方會出現多種專案的 template。

1.2 樣版介紹

Master-Detail Application





提供 master-detail application，使用者可以用 navigation controller 控制資料呈現的方式。呈現的資料方式如下圖，其中可選擇在不同裝置上呈現。

OpenGL Game

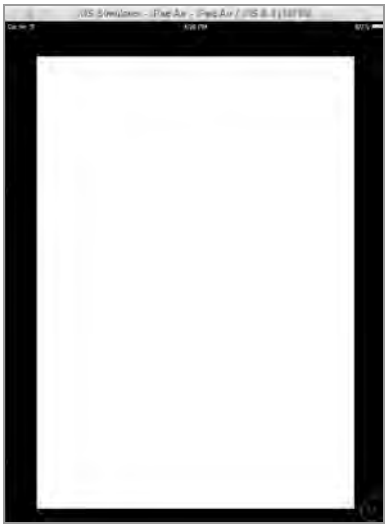
OpenGL Game 此樣版提供大量的遊戲開發所需的函式庫，OpenGL ES 主要是針對嵌入式系統環境，如手機、PDA；目前很多精美的 2D、3D 遊戲幾乎都是使用 OpenGL ES，而最新的版本是 OpenGL ES 2.0，在此不詳細贅述 OpenGL，有興趣的讀者可參考相關書籍。

Page-Based Application

提供了 page based controller，使用者可以利用 page view 來呈現，使用者可以製作一個可以像翻書一樣的動作，在畫面左右滑動，翻到下一頁，或是往回上一頁的動作，而翻轉的過程中，會呈現有如真實翻動書本的動畫。

Single View Application

提供了一個 view controller，使用者可以管理 view 的呈現內容，這個 view 就有如一張空白紙，讓您能夠在上面設計按鈕的擺設位置、圖片的呈現大小、文字...等，這些都可以透過 storyboard 視覺化管理 view 的呈現內容。

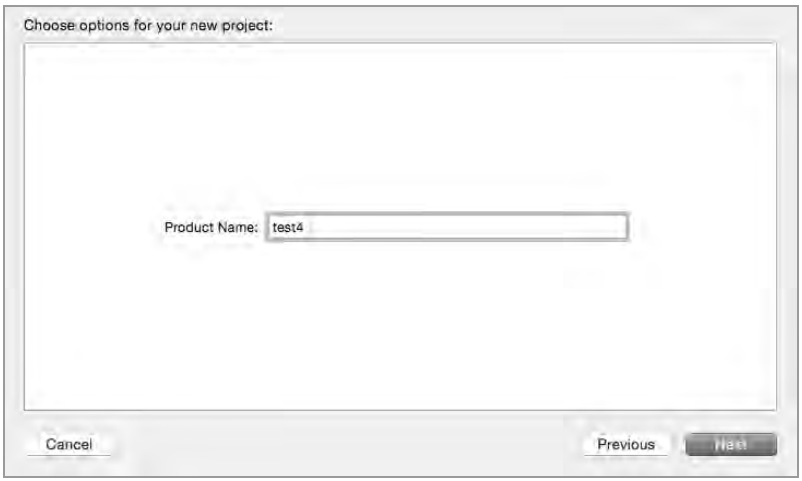


Tabbed Application

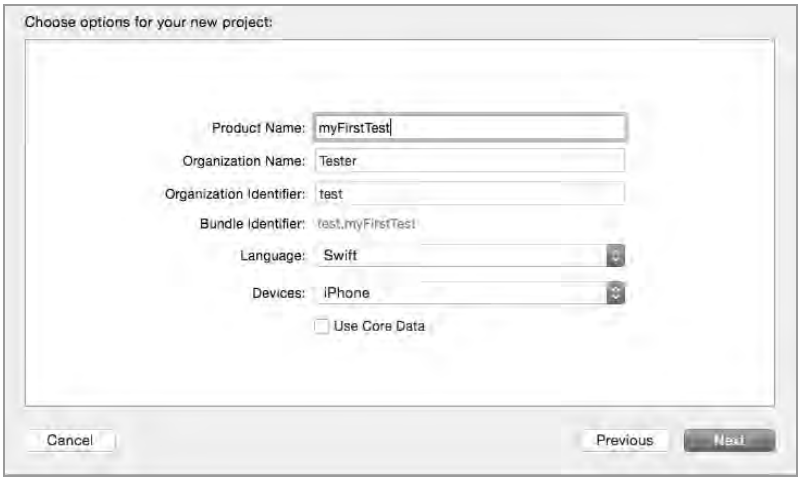
提供了一個 tab bar controller，使用者可以控制 tab bar item 管理每個 view 呈現的內容。Tab bar 固定在螢幕的下方，每個 bar 代表不同的項目，使用者可以快速的切換 bar 瀏覽不同的資料，最常見的例子是 App Store，下方的 Tab bar 有 Featured、Category、Top25、Search、Updates，此外 Tab bar 也可以結合 Navigation 來呈現資料。

Empty Application

此專案樣版只提供一個 window 的元件，可支援 Core Data，在新增專案時，可以看到”Use Core Data”這個選項；Core Data 有許多好用的 API，資料的存取可以使用 Core Data，當有很多資料要讓使用者新增、修改、刪除時，都可以利用 Core Data 存取，類似將資料存在 SQL Server。



接著建立簡單的範例，以 Single View Application 建立我們第一個專案，請輸入專案名稱(Product Name)為 myFirstTest，Organization Identifier 與 Organization Name 都是自行定義，Device 選擇 iPhone，接著按下 Next。



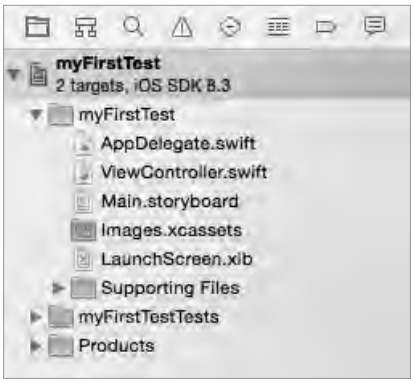
接著選擇專案的儲存的位置，此先將專案範例儲存在桌面，最後按下 Create，如下圖。

1.3 專案導覽

完成專案的建立，會出現如下圖的畫面，在 Xcode 的編輯畫面中主要可以分為下列幾個主要的區域，

左方為 navigation area，可以瀏覽目前專案中的檔案，中間為 editor area，主要是編輯程式碼以及 UI 介面，右方為 utility area，分別可以設定 UI 元件、controller、IBAction、IBOutlet 等細部的設定。

當專案建立完成時，可以在左方的 navigation area 看到已經幫您建立好了一些檔案說明如下：在 myFirstTest 資料夾下包含了所有的 Swift 檔，以及應用程式介面設計的.storyboard 檔。



名稱	說明
Supporting File (支援檔案)	包含了主程式(main)及應用程式的標頭檔、圖片和其他應用程式需要的檔案資料。
Products	這裡是放您最後完成的專案檔案的地方，檔名為.app 結尾，檔案的名稱也會跟您的.app 檔名相同。

您可以看到您目前專案類的檔案架構，在檔案架構的上方，可以看到一排工具列有幾個圖案的按鈕，由左而又分別為 Project、Symbol、Search、Issue、Test、Debug、Breakpoint、Log。

圖示	名稱	說明
	專案導覽 (Project navigator)	預設的畫面，是瀏覽您目前專案文件的檔案。
	符號導覽 (Symbol navigator)	可以由這個視窗看到您目前專案的類別的繼承關係與方法的定義。
	搜尋導覽 (Search navigator)	可用來對整個專案做文字的搜尋，搜尋的結果均會在此專欄顯示。
	錯誤導覽 (Issue navigator)	專案編輯的過程發生的錯誤及警告訊息都會在此專欄顯示。
	測試導覽 (Test Navigator)	用來測試程式碼。
	除錯導覽 (Debug navigator)	用來顯示除錯過程所執行的程式碼。

圖示	名稱	說明
	中斷點導覽 (Breakpoint navigator)	用來顯示開發者自行設定的中斷點。
	記錄檔導覽 (Log navigator)	用來顯示編譯或執行過程中所產生的記錄檔。

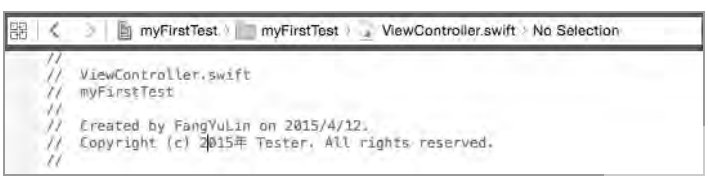
在專案導覽的下方可看到下圖的畫面，

主要是用來新增檔案和搜尋用的，說明如下表：

圖示	名稱	說明
	新增檔案 (Add a new file)	新增檔案到專案中。
	顯示最新檔案 (Show only recent files)	顯示最近開啟過的檔案。
	顯示版本控管的檔案狀態 (Show only files with source-control status)	如果您的程式碼是在版本控管的狀態下，在此會顯示出來。
	顯示符合檔案名稱的搜尋 (Show files with matching name)	輸入要搜尋的檔案名稱，只要符合的都會在此顯示。

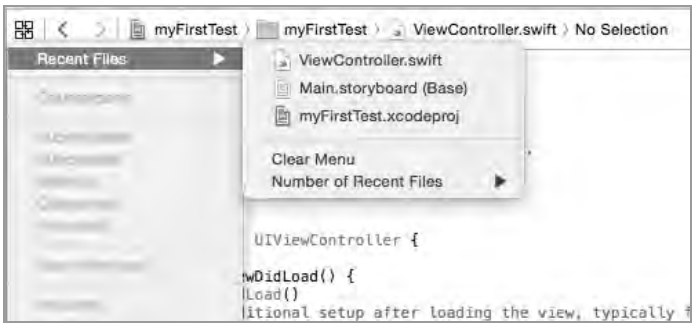
1.4 編輯區域

編輯區域主要就是撰寫程式碼的區塊，所以整個最大的區塊就是撰寫程式碼的區塊，但是在上方還是有幾個功能可以使用，功能區塊主要分為三個區塊，由左至右為：



相關檔案

點選該圖示後會出現您最近開過的檔案、未儲存的檔案、父類別等等。



檔案瀏覽

當您開啟多的檔案後，可以透過該按鈕快速的切換檔案。

檔案的目錄結構 myFirstTest myFirstTest ViewController.swift No Selection

您可以透過他了解到您的檔案目錄結構，也可以點選他來開啟其他的檔案。

1.5 除錯區域

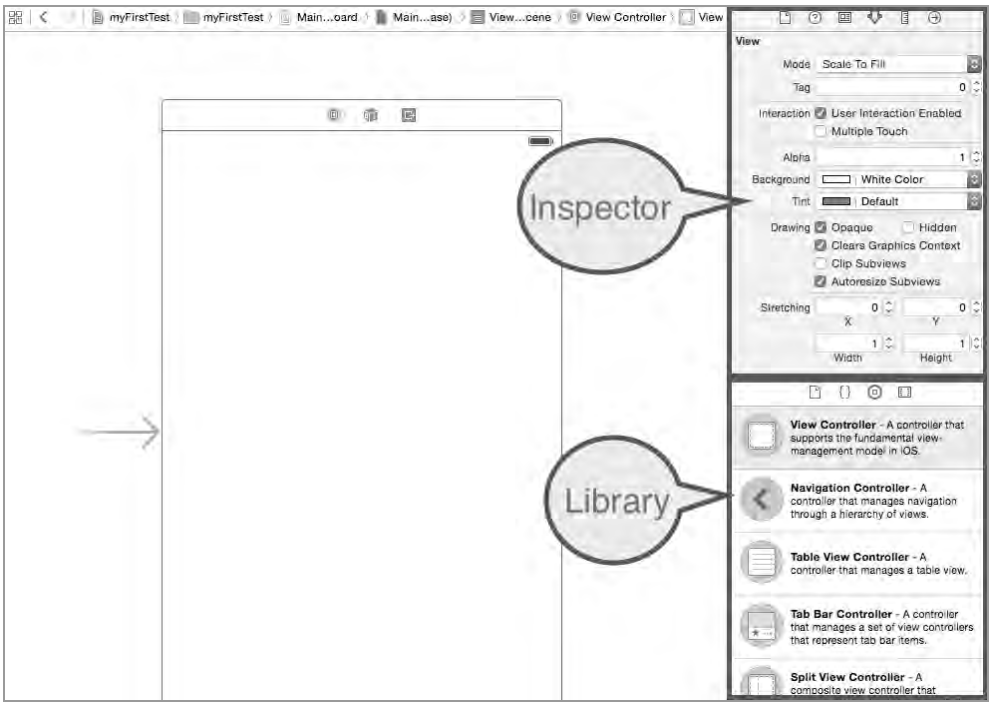
除錯區域主要用來顯示除錯的相關資訊。當您在專案中設定了中斷點，可以在除錯區域中看到您設定的中斷點的相關資訊。除錯區域可以分為兩個區塊，如下圖：

左方為變數的資訊，當您有設定中斷點時，可以看到您設定的中斷點，可以看到您中斷點的變數資訊；右方為程式錯誤訊息及您手動要印出((NSLog)訊息的地方。當您有多個中斷點時，上方可以使用除錯區域上方的功能來操作。如下圖，依序說明如下表：

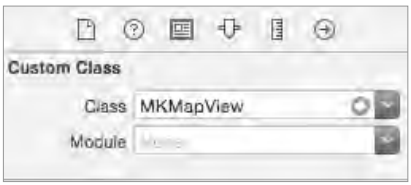
圖示	名稱	說明
	隱藏除錯區域 (Hide the Debug area)	顧名思義就是將除錯區域隱藏起來。
	繼續執行程式碼 (Continue Program execution)	當專案在編譯進行中，會切換成繼續執行程式碼。
	停止程式碼執行 (Pause Program execution)	停止編譯動作。
	跳過函式，但會執行 (Step over)	當您設定的中斷點，過程會在執行其他函式時，不會進入該函式內一行一行執行，而是直接到您設定的下一個中斷點。
	進入函式 (Step into)	當您設定的中斷點，過程會去執行其他的函式時，並進入該函式一行一行執行。
	跳出函式 (Step out)	在使用逐行偵錯時，進入處理函式後如果想要返回前一個 Call Stack 呼叫函式時使用。

1.6 元件庫視窗

首先來設計 iPhone 的 UI 介面，先點選 MainStoryboard_iPhone.storyboard，可以看到以下的畫面。左方即是一個空白的介面可以讓您設計，右方則有 UI 元件以及可以設定元件的屬性，其中右方又可分為 Inspector 與 Library，先來看 Library。



在 Library 視窗直接拖曳到 View 視窗，即可完成在畫面中新增不同元件的動作。完成元件新增的動作之後，可以對元件本身的屬性進行一些相關的設定，這些設定的內容於 Inspector 的視窗中，如右圖。



在這因為最常使用的是 Identity、Attribute、Size、Connection，因此主要是介紹這四個介紹。

圖示	名稱	說明
	Attribute	設定元件的外觀
	Connection	設定元件和程式碼的連接關係
	Size	設定元件的 width、height 和 x、y 座標
	Identity	設定物件的 Class

1.7 視窗的切換

編輯模式(Editor) 

在右方 Inspector、Library 的上方可以看到 Editor View，說明如下表：

圖示	名稱	說明
	標準模式 (Standard)	這是最常使用的模式，整個區域只會顯示一個編輯程式碼的區塊。當您在編輯介面的時候，iPhone 的介面設計也會在這個區塊顯示。
	輔助模式 (Assistant)	輔助模式可同時開啟兩個檔案，您可同時編輯程式碼及介面的 Outlet 與 Action，在這個模式下很好用，可同時編輯兩個檔案，例如一個是.h 檔定義變數或方法時，可同時直接編輯.m 檔，這樣就不需要在尋找對應的.m 檔或是.h 檔。
	版本模式 (Version)	此模式可以看到.storyboard、.xib、.plist 檔案的原始碼，也可以讓您看到不同版本間的專案的差異，但前提是您的專案必須受到版本控管系統的控制。

視窗佈局 

視窗的佈局，顧名思義就是開發介面的的佈局，您會發現在一開始的時候，除錯區域並不會在建立專案後馬上開啟，要等到您編譯專案時，才會開啟除錯區域；除錯區域是可以隱藏的，因此專案導覽、Library、Inspector 也都是可以隱藏的，在此就不再多做說明，自行點選視窗的布局，很明顯就可以看出端倪。

圖示	說明
	出現 navigation 區域
	出現 debug 區域
	出現 utility 區域

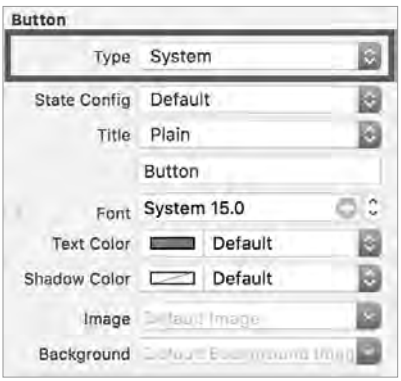
2 CHAPTER

UIButton

UIButton 是對使用者的動作，執行相對應程式碼的視圖。簡單地來說，就是當使用者按下按鈕，會去執行開發者為按鈕所寫的程式碼。接下來，我們要來簡單地實際操作一遍 UIButton。

Step 1 建立 Single View Application 專案，專案名稱為 UIButton。

Step 2 編輯 MainStoryboard.storyboard。UIButton 按鈕在前面已經有簡單的使用到，Main.storyboard 中加入 UIButton，在 Attribute Insepctor 可看到有很多屬性欄位。第一個可看到 Type 欄位，用來設定按鈕的型態。一共有 6 種形態。如下圖。



型態	說明
Custom	可以客製化的按鈕，例如要使用自行設計好的圖 片。
System	原本預設的圓角按鈕，裡面可以輸入文字

型態	說明
Detail Disclosure	代表後續還有詳細的資料
Info Light	資訊按鈕高亮度的版本
Info Dark	資訊按鈕顏色較深的版本
Add Contact	新增按鈕，表示要新增東西。

在此還是使用原本預設的按鈕做介紹；加入按鈕近來之後，可以使用輔助編輯的模式，建立按鈕的 IBAction 方法，如下圖，接著再開做 ViewController.swift 撰寫該方法的執行內容。

Step 3 編輯 ViewController.swift

加入以下程式碼

```
@IBAction func click (_ sender: Any){
    print("click")
}
```

Step 4 編譯專案這時編譯專案，點選 click 按鈕，在除錯區域就會顯示 NSLog 的訊息。

這樣的操作過程很簡單，但是不可能都如欲其按鈕是固定在這些位置上，在操作的設計上會有其他的設計，按鈕可能位置是動態的，而且按鈕執行方法內容並不一定要使用 IBAction。接下來就以程式碼的方式進行撰寫。

Step 1 建立 Single View Application 專案，專案名稱為 UIButton2

Step 2 編輯 ViewController.swift

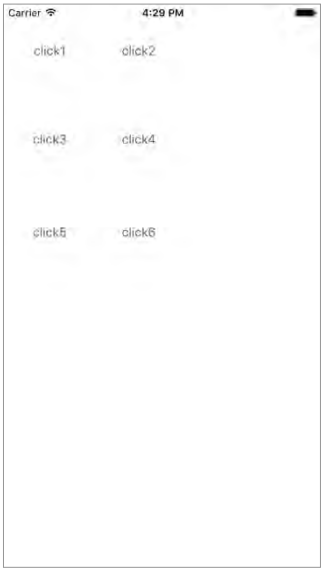
範例程式

```
01  import UIKit
02  class ViewController: UIViewController {
03      override func viewDidLoad() {
04          super.viewDidLoad()
05          //定義按鈕的 Type
06          let btn1 = UIButton(type: UIButtonType.system)
07          let btn2 = UIButton(type: UIButtonType.system)
08          let btn3 = UIButton(type: UIButtonType.system)
09          let btn4 = UIButton(type: UIButtonType.system)
10          let btn5 = UIButton(type: UIButtonType.system)
11          let btn6 = UIButton(type: UIButtonType.system)
12
13          //定義按鈕的位置與寬高
14          btn1.frame = CGRect(x: 5,y: 5,width: 100,height: 100)
15          btn2.frame = CGRect(x: 110,y: 5,width: 100,height: 100)
16          btn3.frame = CGRect(x: 5,y: 110,width: 100,height: 100)
17          btn4.frame = CGRect(x: 110,y: 110,width: 100,height: 100)
18          btn5.frame = CGRect(x: 5,y: 220,width: 100,height: 100)
19          btn6.frame = CGRect(x: 110,y: 220,width: 100,height: 100)
20
21          //定義按鈕的 Title 文字
22          btn1.setTitle("click1", for: UIControlState())
23          btn2.setTitle("click2", for: UIControlState())
24          btn3.setTitle("click3", for: UIControlState())
25          btn4.setTitle("click4", for: UIControlState())
26          btn5.setTitle("click5", for: UIControlState())
27          btn6.setTitle("click6", for: UIControlState())
28
29          //為所有的按鈕加入 action 方法
30          btn1.addTarget(self, action: #selector(ViewController.btn1Click(_:)),
31                         for: UIControlEvents.touchUpInside)
32          btn2.addTarget(self, action: #selector(ViewController.btn2Click(_:)),
33                         for: UIControlEvents.touchUpInside)
34          btn3.addTarget(self, action: #selector(ViewController.btn3Click(_:)),
```

```
35                          for: UIControlEvents.touchUpInside)
36          btn4.addTarget(self, action: #selector(ViewController.btn4Click(_:)),
37                          for: UIControlEvents.touchUpInside)
38          btn5.addTarget(self, action: #selector(ViewController.btn5Click(_:)),
39                          for: UIControlEvents.touchUpInside)
40          btn6.addTarget(self, action: #selector(ViewController.btn6Click(_:)),
41                          for: UIControlEvents.touchUpInside)
42
43          //將按鈕加入畫面
44          self.view.addSubview(btn1)
45          self.view.addSubview(btn2)
46          self.view.addSubview(btn3)
47          self.view.addSubview(btn4)
48          self.view.addSubview(btn5)
49          self.view.addSubview(btn6)
50      }
51
52      //為所有的按鈕加入 action 方法
53      @objc func btn1Click (_ sender:UIButton){
54          print("btn1 clicked")
55      }
56      @objc func btn2Click (_ sender:UIButton){
57          print("btn2 clicked")
58      }
59      @objc func btn3Click (_ sender:UIButton){
60          print("btn3 clicked")
61      }
62      @objc func btn4Click (_ sender:UIButton){
63          print("btn4 clicked")
64      }
65      @objc func btn5Click (_ sender:UIButton){
66          print("btn5 clicked")
67      }
68      @objc func btn6Click (_ sender:UIButton){
69          print("btn6 clicked")
70      }
71
72      override func didReceiveMemoryWarning() {
73          super.didReceiveMemoryWarning()
```

```
74          // Dispose of any resources that can be recreated.
75      }
76  }
```

Step 3 定義 6 個按鈕，型態都設為 System ，接著使用 CGRect 方法定義按鈕的位置與寬高，setTitle 是設定按鈕的 Title 文字，addTarget 裡的 forControlEvents:是設定按鈕的觸發事件為「 UIControlEvents.TouchUpInside 」，當按鈕被觸發時會執行 action:指定的方法。最後使用 addSubView 將按鈕加入畫面中。如右圖。在各個按鈕事件中，使用簡單的 print 讓我們知道按鈕是否觸發到對應的方法。



Step 4 編譯專案

編譯完成專案並點選模擬器中的按鈕，觀察除錯區域是否有顯示對應的事件。如下圖。