自我練習題

1. 前面曾提到此計算器的 App 還有一些 Bugs 存在，例如連續按下運算符號，將使數字運算後結果為 0，請修改此一 Bug。

20 CHAPTER

計算器 (Mac 版本)

前一章已實作一個可在 iOS 裝置上執行的計算器 App，本章將以同樣的功能，製作一個可以在 Mac OS 上執行的 App。

20.1 建立一個計算器的專案

首先開啟 Xcode，選取 Create a new Xcode project，然後選取 macOS -> Application -> Cocoa。如圖 20-1 所示：

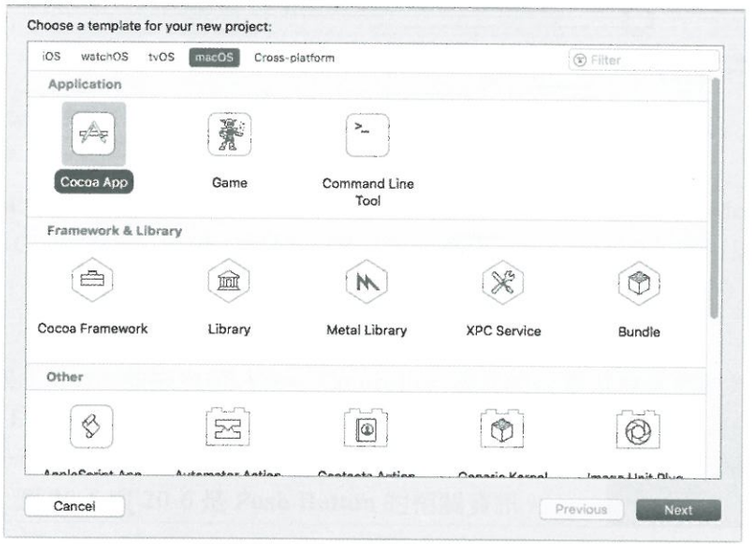

圖 20-1

按下 Next。接著是專案的名稱設定，按照圖 20-2 的設定即可。

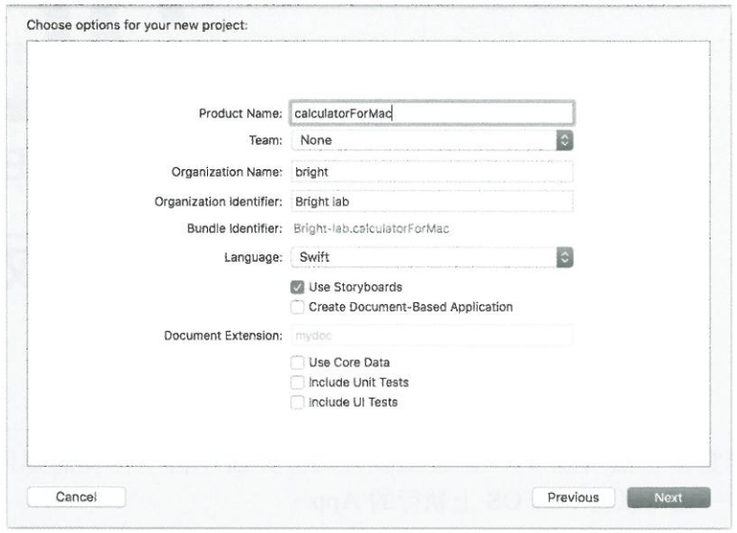

圖 20-2

完成後再按下 Next。此時會有一視窗指示使用者選擇專案欲儲存位置，最後按下 Create。

一個能在 OS X 上執行的空白 App 就建立完成了，如圖 20-3 所示。

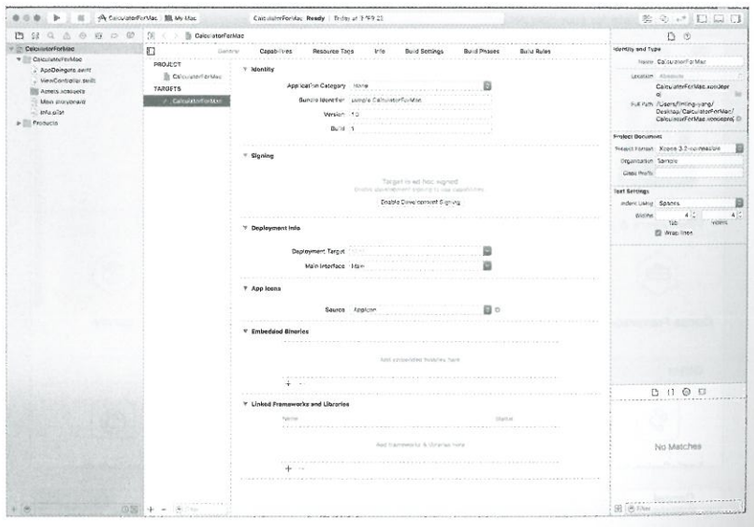

圖 20-3

20.2 UI 設計

同樣先在左側的導覽列中打開 Main.storyboard，開始排列計算器的使用者介面，需要注意的是，這裡的介面與製作 iOS App 介面有所不同，如圖 20-4 所示。

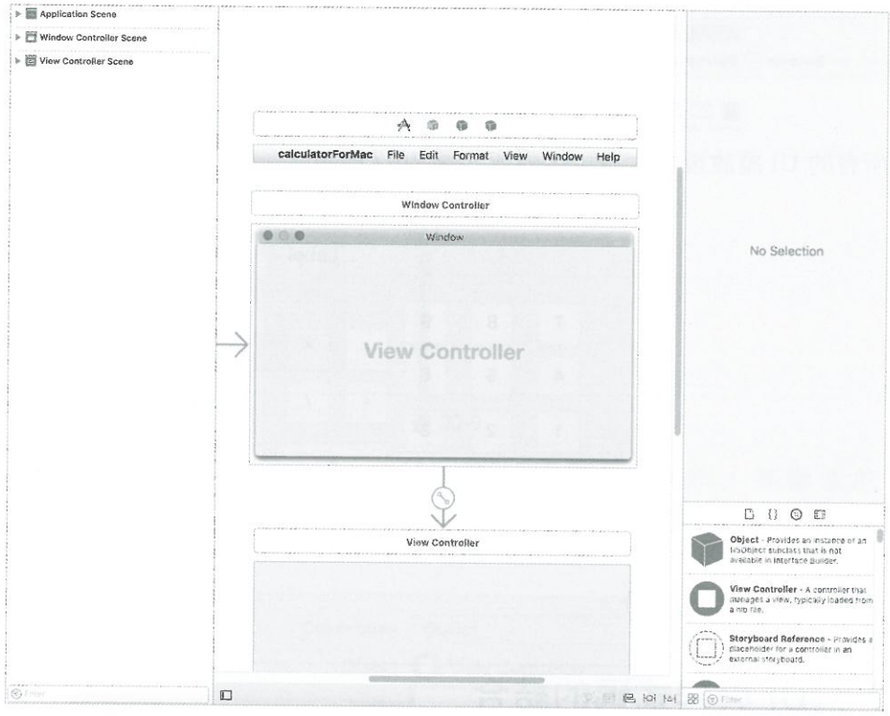

圖 20-4

不同於 iOS 初始的單一 View Controller，在 OS X 上多了 Main Menu 以及 Window Controller，不過，目前我們只要注意 View Controller 就可以。此處也會看到 UI 元件庫的不同，雖然要製作的是相同的範例，但使用的 UI 元件會略有不同。

首先選取 Label，並拖曳至 View Controller 適當的位置擺放。同樣地，再選取 Push Button 拖曳至 View Controller 適當的位置，並修改按鈕的名稱，可參閱前一章建立 Label 和 Button 方式。我們可以從屬性的欄位得知或其相關的資訊，圖 20-5 與 20-6 是 Push Button 的相關資訊。

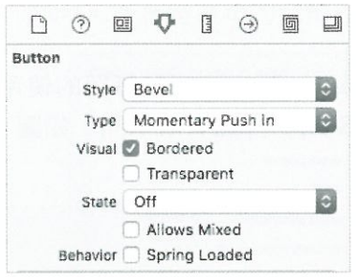

圖 20-5

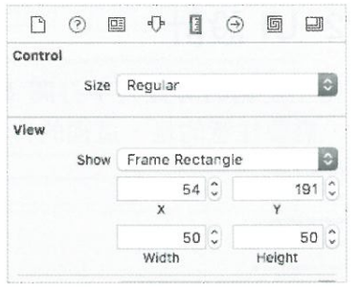

圖 20-6

將所有的 UI 擺放後，大概可排列成如圖 20-7 所示。

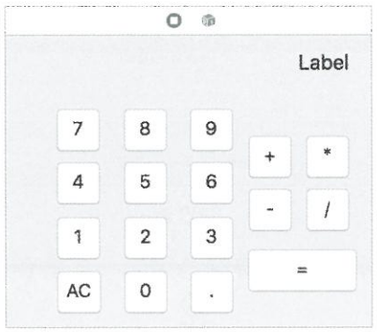

圖 20-7

如此便完成了 UI 的部分。

20.3 UI 與程式碼的結合

和 iOS 計算器的範例相似，按下視窗右上角，如圖 20-8 中的 按鈕。

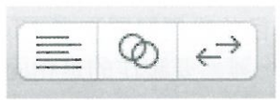

圖 20-8

將 UI 元件與程式碼建立連結。從左側的 Main.storyboard 點選 UI 元件，按住 control 並拖曳至右側的程式碼中，如圖 20-9 所示。

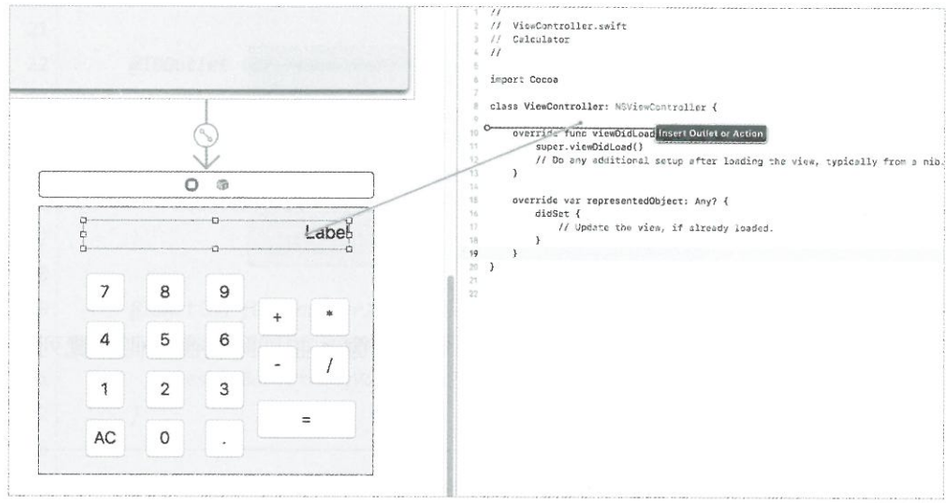

圖 20-9

要注意的是，在 OS X 中，Label 的連結如圖 20-10，其繼承的是 NSTextField，這與 iOS 的 UILabel 有所不同，所以在稍後程式碼的部分也略有所差異。請在 Name 填入 resultBar。

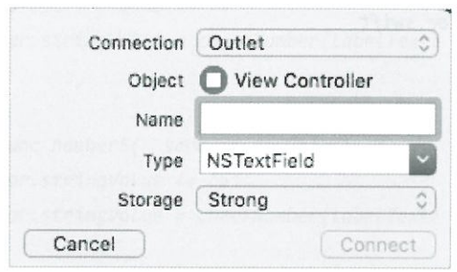

圖 20-10

而 Push Button 的連結如圖 20-11，注意要將 Connection 欄位值改為 Action。並請在 Name 欄位填入適當的名稱，如 number1、number2, ...等等。請參閱以下的完整程式碼。

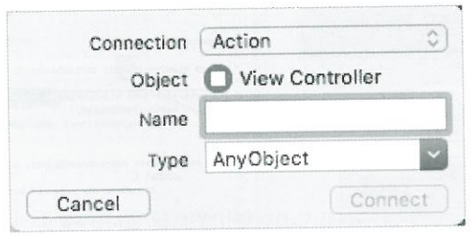

圖 20-11

完成所有 UI 元件的連結後，就可以切回單一視窗，並打開視窗左側導覽列的 ViewController.swift。

20.4 完整程式碼

與 iOS 專案不同之處在於 import Cocoa 以及繼承 NSViewController，不過在本範例中，程式碼大至相同，只在 Label 字串的程式碼略有所不同，在 OS X 上使用 resultBar.stringValue，而非 resultBar.text!，完整程式碼如下：

```
01  //
02  //  ViewController.swift
03  //  Calculator
04  //
05
06  import Cocoa
07  enum Sign {
08      case nothing
09      case plus
10      case minus
11      case multi
12      case division
13  }
14
15  class ViewController: NSViewController {
16      let formatter = NumberFormatter()
17      var firstNumber: Double = 0.0
```



```
18      var secondNumber: Double = 0.0
19      var finalNumber: Double = 0.0
20      var currentSign = Sign.nothing
21
22      @IBOutlet var resultBar: NSTextField!
23
24      @IBAction func number1(_ sender: AnyObject) {
25          resultBar.stringValue += "1"
26          resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
27      }
28
29      @IBAction func number2(_ sender: AnyObject) {
30          resultBar.stringValue += "2"
31          resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
32      }
33
34      @IBAction func number3(_ sender: AnyObject) {
35          resultBar.stringValue += "3"
36          resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
37      }
38
39      @IBAction func number4(_ sender: AnyObject) {
40          resultBar.stringValue += "4"
41          resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
42      }
43
44      @IBAction func number5(_ sender: AnyObject) {
45          resultBar.stringValue += "5"
46          resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
47      }
48      @IBAction func number6(_ sender: AnyObject) {
49          resultBar.stringValue += "6"
50          resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
51      }
52      @IBAction func number7(_ sender: AnyObject) {
53          resultBar.stringValue += "7"
54          resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
55      }
56      @IBAction func number8(_ sender: AnyObject) {
```

```
        resultBar.stringValue += "8"
        resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
    }
    @IBAction func number9(_ sender: AnyObject) {
        resultBar.stringValue += "9"
        resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
    }
    @IBAction func number0(_ sender: AnyObject) {
        resultBar.stringValue += "0"
        if !(resultBar.stringValue.contains(".")){
            resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
        }
    }
    @IBAction func point(_ sender: AnyObject) {
        if !(resultBar.stringValue.contains(".")){
            resultBar.stringValue += "."
        }
    }
    @IBAction func plus(_ sender: AnyObject) {
        resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
        firstNumber = Double(resultBar.stringValue)!
        resultBar.stringValue = "0"
        currentSign = Sign.plus
    }
    @IBAction func minus(_ sender: AnyObject) {
        resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
        firstNumber = Double(resultBar.stringValue)!
        resultBar.stringValue = "0"
        currentSign = Sign.minus
    }
    @IBAction func multi(_ sender: AnyObject) {
        resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
        firstNumber = Double(resultBar.stringValue)!
        resultBar.stringValue = "0"
        currentSign = Sign.multi
    }
    @IBAction func division(_ sender: AnyObject) {
        resultBar.stringValue = checkNumber(labelText: resultBar.stringValue)
        firstNumber = Double(resultBar.stringValue)!
```



```
        resultBar.stringValue = "0"
        currentSign = Sign.division
    }
    @IBAction func AC(_ sender: AnyObject) {
        firstNumber = 0
        secondNumber = 0
        resultBar.stringValue = "0"
        currentSign = Sign.nothing
    }
    @IBAction func equal(_ sender: AnyObject) {
        switch currentSign {
        case .plus:
            secondNumber = Double(resultBar.stringValue)!
            finalNumber = firstNumber + secondNumber
            resultBar.stringValue = checkNumber(labelText: String(finalNumber))
            currentSign = Sign.nothing
        case .minus:
            secondNumber = Double(resultBar.stringValue)!
            finalNumber = firstNumber - secondNumber
            resultBar.stringValue = checkNumber(labelText: String(finalNumber))
            currentSign = Sign.nothing
        case .multi:
            secondNumber = Double(resultBar.stringValue)!
            finalNumber = firstNumber * secondNumber
            resultBar.stringValue = checkNumber(labelText: String(finalNumber))
            currentSign = Sign.nothing
        case .division:
            secondNumber = Double(resultBar.stringValue)!
            finalNumber = firstNumber / secondNumber
            resultBar.stringValue = checkNumber(labelText: String(finalNumber))
            currentSign = Sign.nothing
        case .nothing:
            break
        }
    }
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        resultBar.stringValue = "0"
```

```
    }

    override var representedObject: Any? {
        didSet {
            // Update the view, if already loaded.
        }
    }

    func checkNumber(labelText: String) -> String {
        var stringOfNumber: NSNumber
        stringOfNumber = formatter.number(from: labelText)!
        return String(describing: stringOfNumber)
    }
}
```

完成後，按下執行，便可以順利地在 OS X 上執行計算器。圖 20-12 是此計算器的初始畫面。

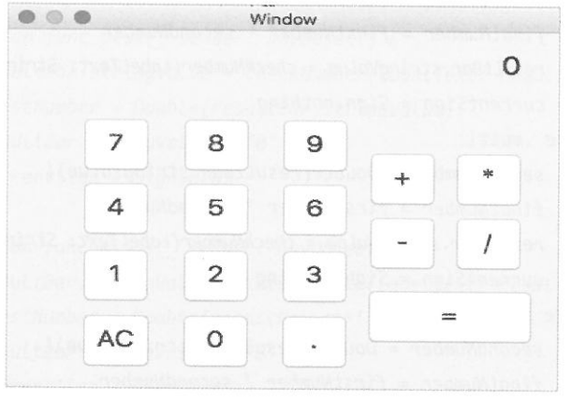

圖 20-12

21 CHAPTER

在 iOS 裝置上製作隨機顯示圖片的 App

本章仿照第 19 章來建立一個新的 iOS 專案，命名為 NextRandomImage。此 App 的概念是，按下按鈕後，畫面上的圖片會隨機換成儲存在 App 中圖片集的其中一張，另外也會提示出，目前顯示的圖片是哪一張圖片。這個簡單的 App 會用到以下幾種 UI 元件：Navigation Bar、Image View、Button、Label。

21.1 UI 設計

圖 21-1 是這個 App 大略的 UI 設計，您不必按照此元件排列方式，大可發揮您的美學佈局。

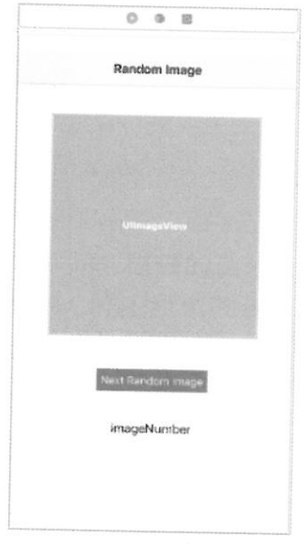

圖 21-1