

行動裝置數位匯流應用

與應用程式互動：基本元件

王昱景 Brian Wang

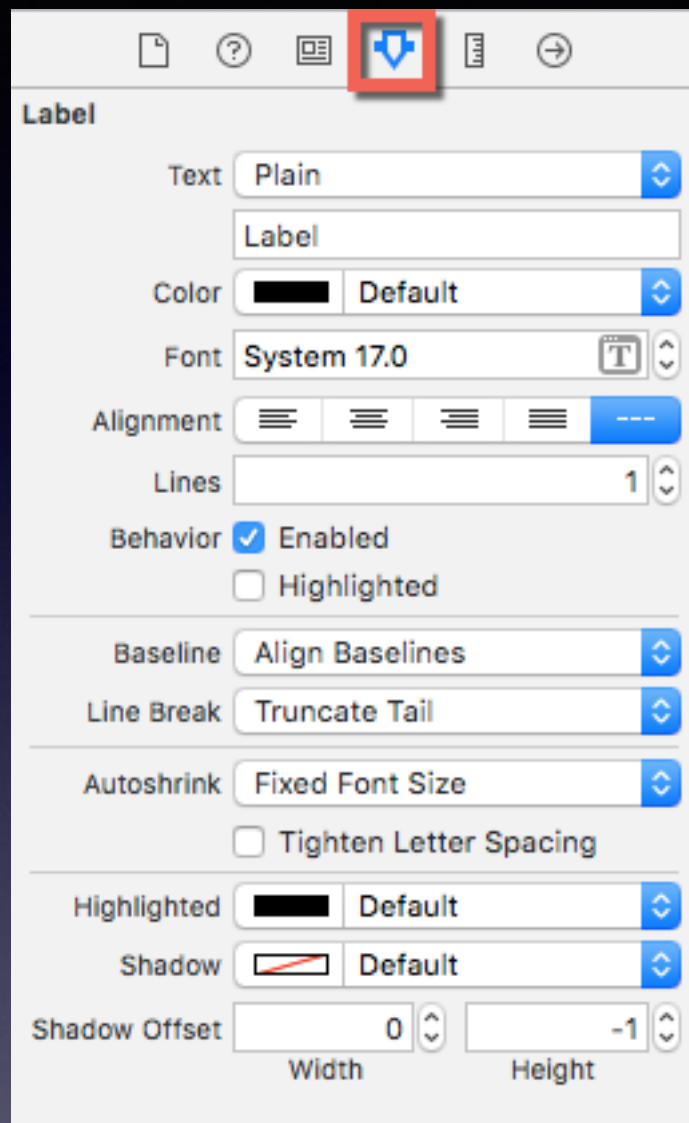
brian.wang.frontline@gmail.com

Label 元件

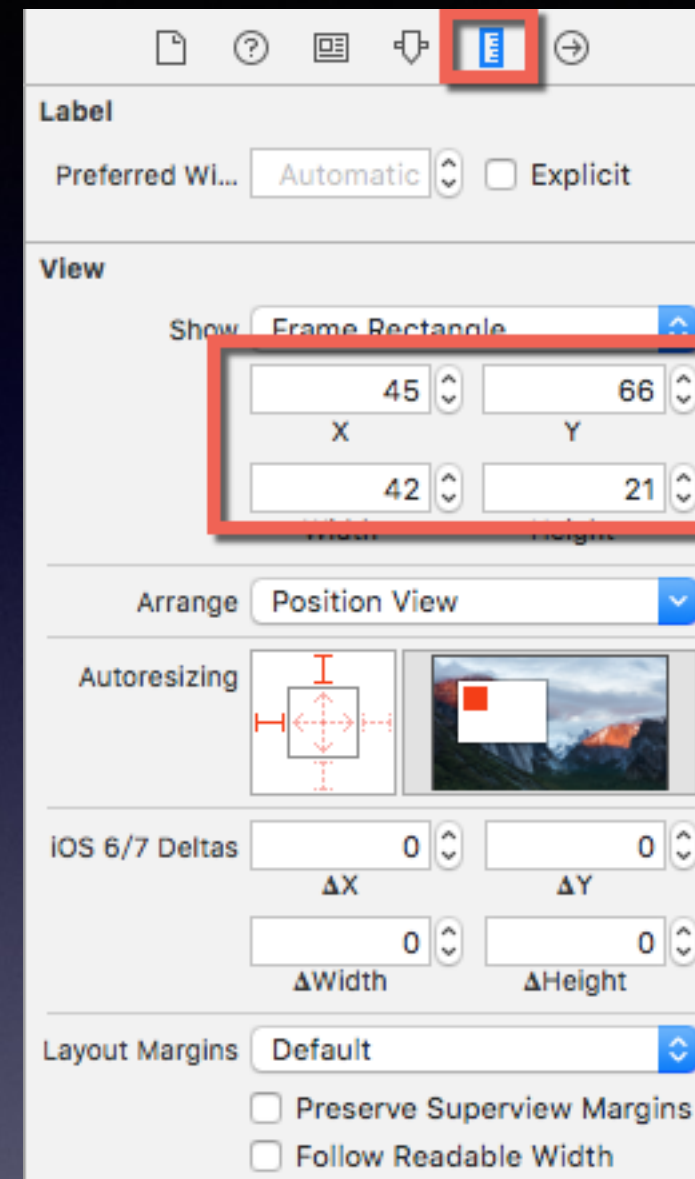
- iOS App 的介面元件位於 Main.storyboard 版面配置檔中
- 最主要的功能是顯示應用程式與使用者互動的介面
- Label 元件是最基礎的介面元件
- 功能是在畫面上顯示訊息，但不允許輸入資料

Label 元件常用屬性

Text	設定顯示的文字特性及內容
Color	可由右方下拉式選單選取文字顏色。預設值為黑色
Font	設定文字的字體及大小。預設字體為 System，預設大小為 17
Alignment	設定文字的排列方式。預設值為靠左排列
Lines	設定顯示的文字列數，預設值為 1。若此屬性值設為 0，表示顯示列數不受任何限制
Line Breaks	設定文字太長，無法完全顯示時的處理方式
Shadow	設定文字陰影的顏色。預設值為 Default，表示無陰影
Shadow Offset	設定文字陰影的偏移量，Horizontal 為水平偏移量，Vertical 為垂直偏移量。此屬性只有在 Shadow 不為 Default 時才有效
X	設定元件的水平位置
Y	設定元件的垂直位置
Height	設定元件的高度
Width	設定元件的寬度

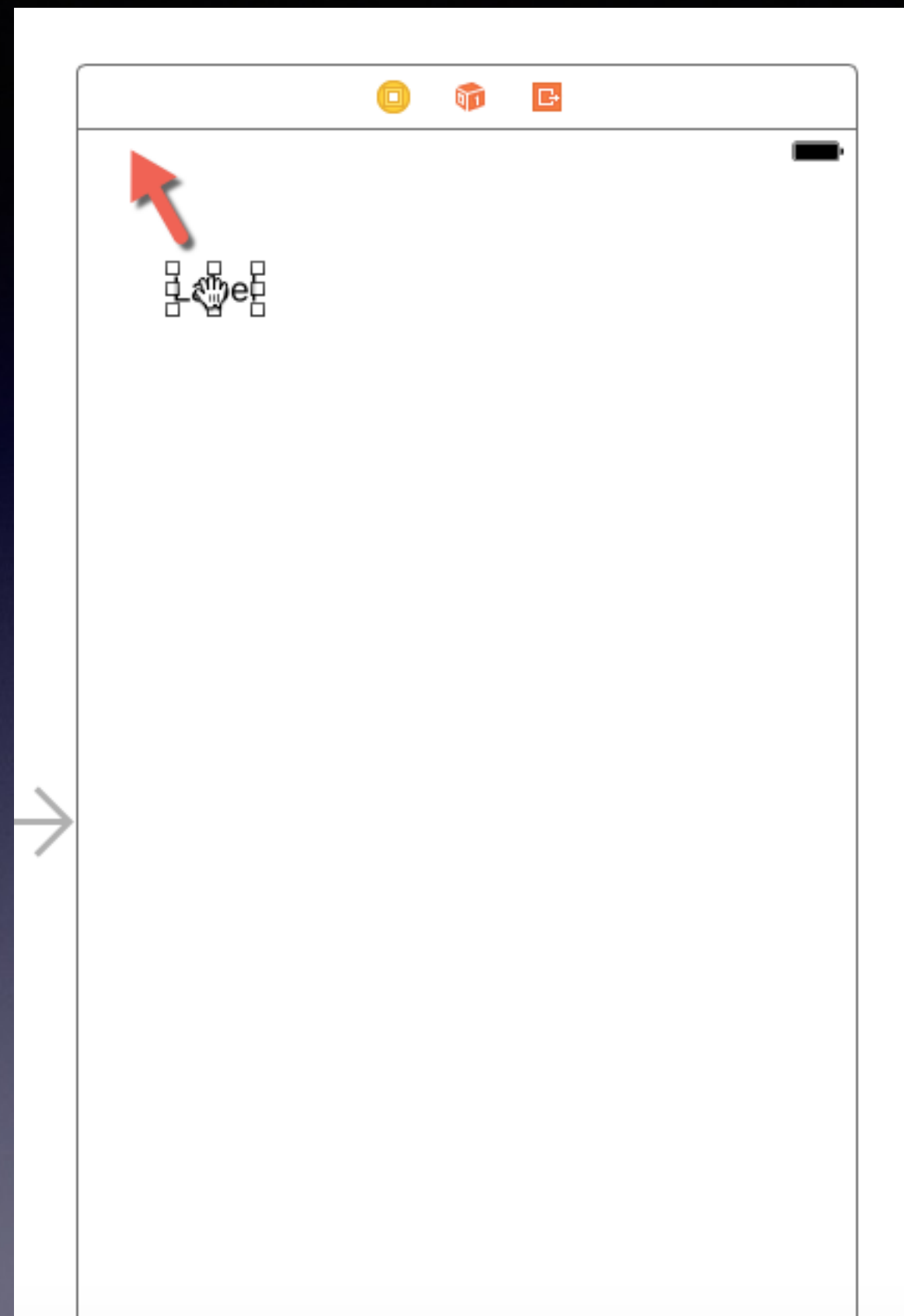


屬性面板

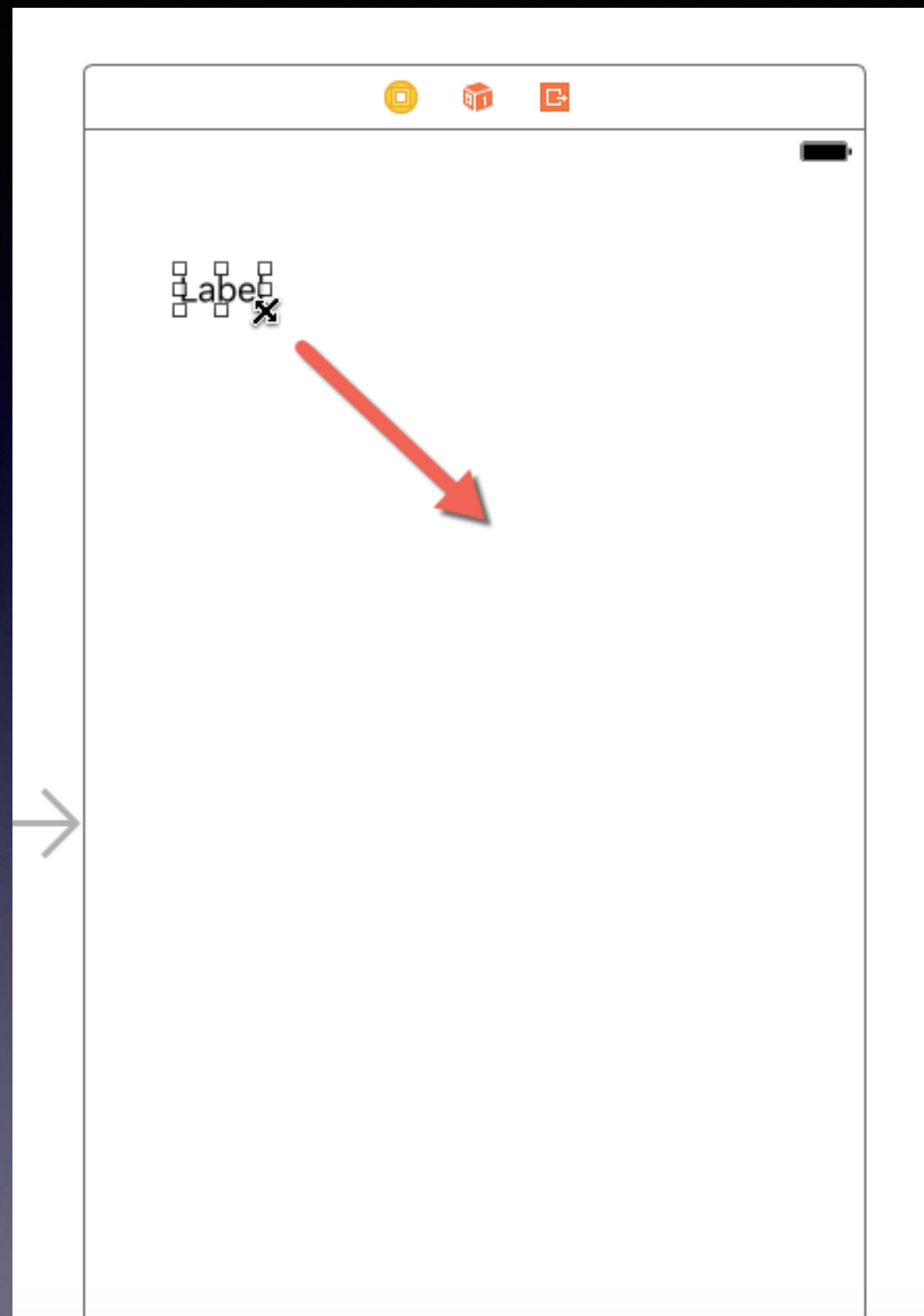


Size 面板

拖曳滑鼠移動位置



拖曳滑鼠改變大小



- Text 屬性
 - Text 屬性預設值為 Plain，只能顯示簡單形式的文字
 - 如果要詳細設定各種文字屬性，可切換到 Attributed 模式

Label

Text Attributed

系統字體 標準體 17.0

Label

Lines 1

Behavior ☒ Enabled
☐ Highlighted

Baseline Align Baselines

Line Break Truncate Tail

Autoshrink Fixed Font Size
☐ Tighten Letter Spacing

Highlighted ☐ Default

Shadow ☐ Default

Shadow Offset Width 0 Height -1

View

Content Mode Left

Fonts


Collection	Family	Typeface	Size
All Fonts	手札體-繁		17
繁體中文	宋體-繁		9
Favorites	娃娃體-繁		10
Recently Used	雅痞-繁		11
現代	黑體-繁		12
等寬字	圓體-繁		13
傳統	楷體-繁		14
網頁	翩翩體-繁		18
趣味	魏碑-繁		24
PDF	蘋方-繁		36
Windows Office 相容	儷宋 Pro		48

- Font 屬性
 - Font 屬性的預設字體為 System，此種模式只能改變字體大小，預設字體大小為 17
 - 如果要改字體及字型，可在下拉選單修改 Font 欄位為 Custom，此時 Family 及 Style 會改變為可設定，即可更改字體及字型

Label

Text

Color

Font 

Font

Family

Style

Size

☐ Autoshrink ☐ Fixed Font Size

☐ Tighten Letter Spacing

Highlighted

Shadow

Shadow Offset


Width Height

View

Label

Text

Color

Font 

Font

Family

Style

Size

☐ Autoshrink ☐ Fixed Font Size

☐ Tighten Letter Spacing

Highlighted

Shadow

Shadow Offset

Width Height

View

- 文字陰影
 - Shadow 屬性的預設值為 Default，表示無陰影
 - 若將 Shadow 屬性改為指定顏色，文字就會顯示陰影效果
 - Horizontal 及 Vertical 欄位可設定陰影的水平及垂直位移

新增範例專案

- 建立應用程式的第一個步驟就是新增專案
- 啟動 Xcode 後，點按 Create a new Xcode Project，或執行功能表 File / New / Project



Welcome to Xcode

Version 8.0 (8A218a)



Get started with a playground

Explore new ideas quickly and easily



Create a new Xcode project

Create an app for iPhone, iPad, Mac, Apple Watch or Apple TV.



Check out an existing project

Start working on something from an SCM repository.

No Recent Projects

Open another project...

- 在 Choose a template for your new project 對話方塊點選 iOS
- 下方選擇 Application 的 Single View Application 模板
- 按 Next 鈕繼續

Choose a template for your new project:

iOS

1

tvOS

macOS

Cross-platform

Filter

Application

1

Single View
Application

2



Game



Master-Detail
Application



Page-Based
Application



Tabbed
Application



Sticker Pack
Application



iMessage
Application

Framework & Library



Cocoa Touch
Framework



Cocoa Touch
Static Library



Metal Library

Cancel

Previous

3
Next

- Product Name 欄位輸入專案名稱 Label
- Organization Name 輸入 UCH
- Organization Identifier 輸入 tw.edu.uch
- Language 欄位由下拉選單點選 Swift
- 按 Next 鈕繼續

Choose options for your new project:

Product Name:

Label

1

Team:

Add account...

Organization Name:

UCH

2

Organization Identifi...

tw.edu.uch

Bundle Identifier:

tw.edu.uch.Label

Language:

Swift

3

Devices:

iPhone

- ☐ Use Core Data
- ☒ Include Unit Tests
- ☒ Include UI Tests

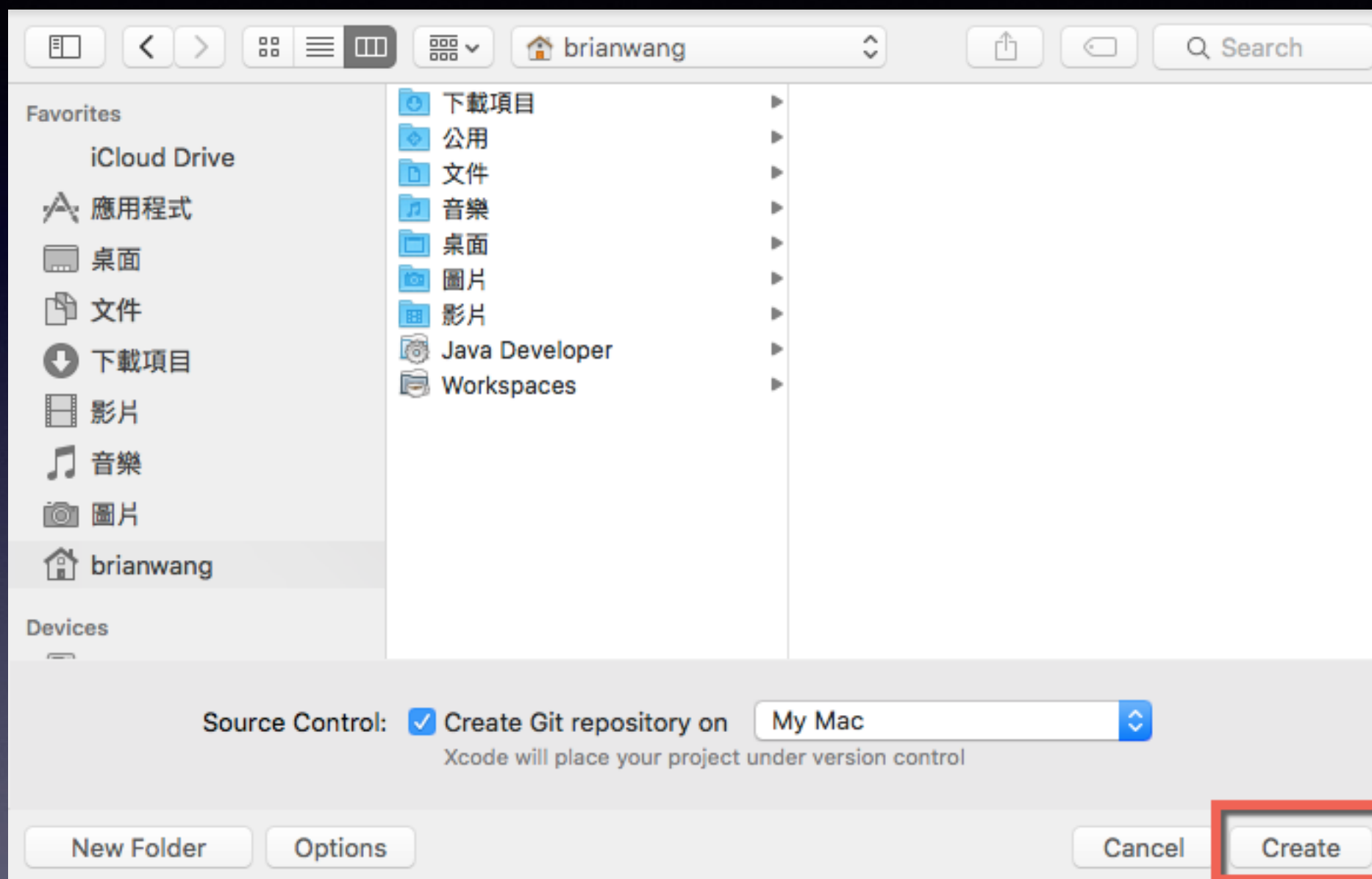
4

Cancel

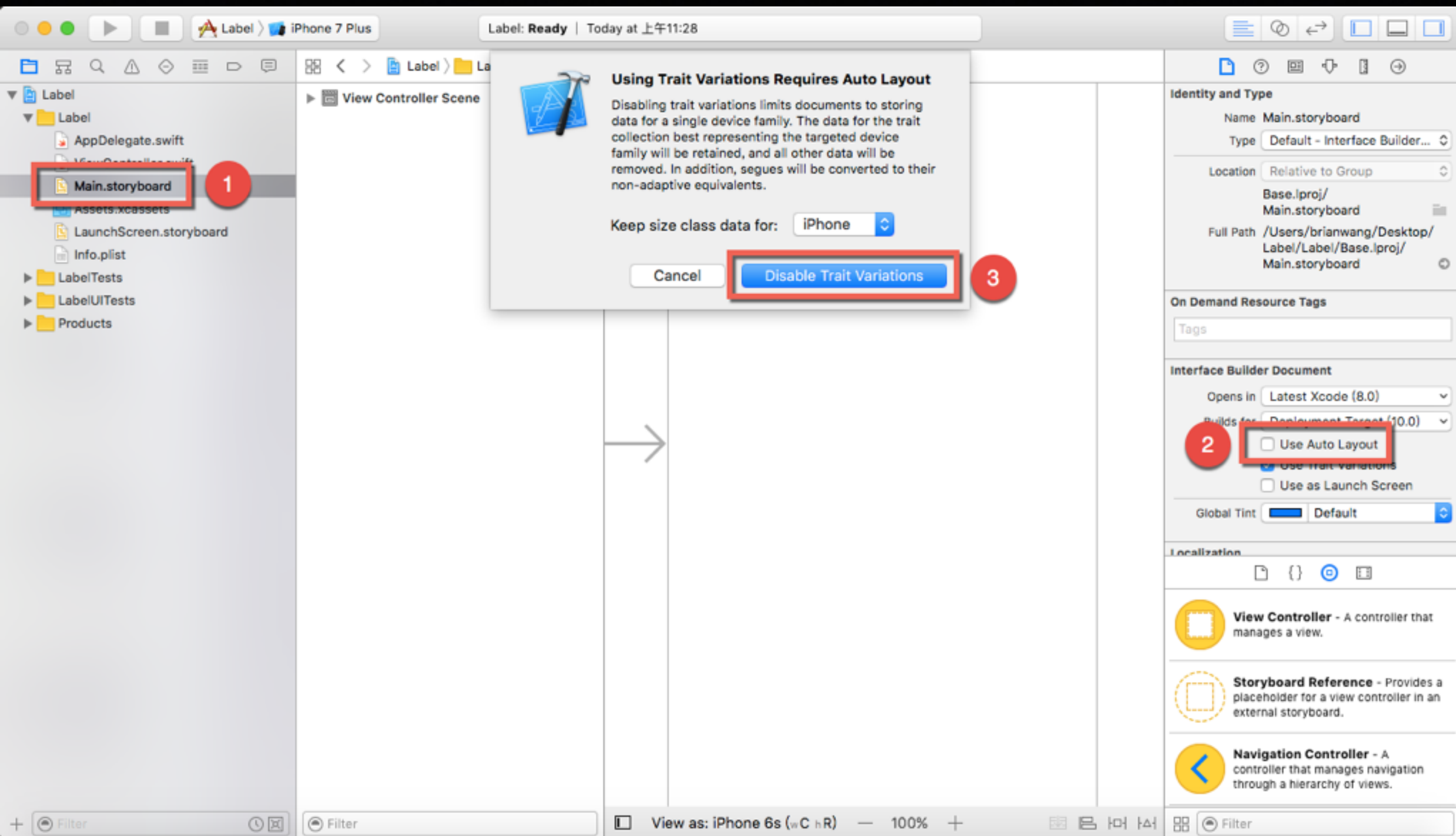
Previous

Next

- 接著選擇儲存專案資料夾
- 按 Create 建立專案



- 於檔案瀏覽區點選 Main.storyboard
- 於屬性設定區按  圖示
- 取消核選 Use Auto Layout 項目
- 再於對話方塊中按 Disable Size Classes 鈕



拖曳建立 Label

- 切換回 Main.storyboard 版面配置檔
- 由元件區拖曳 Label 元件到介面設計區適當位置



Identity and Type

Name Main.storyboard

Type Default - Interface Builder...

Location Relative to Group

Base.lproj/
Main.storyboard

Full Path /Users/brianwang/Desktop/
Label/Label/Base.lproj/
Main.storyboard

On Demand Resource Tags

Tags

Interface Builder Document

Opens in Latest Xcode (8.0)

Builds for Deployment Target (10.0)

☐ Use Auto Layout

☐ Use Trait Variations

☐ Use as Launch Screen

Global Tint  Default

Localization




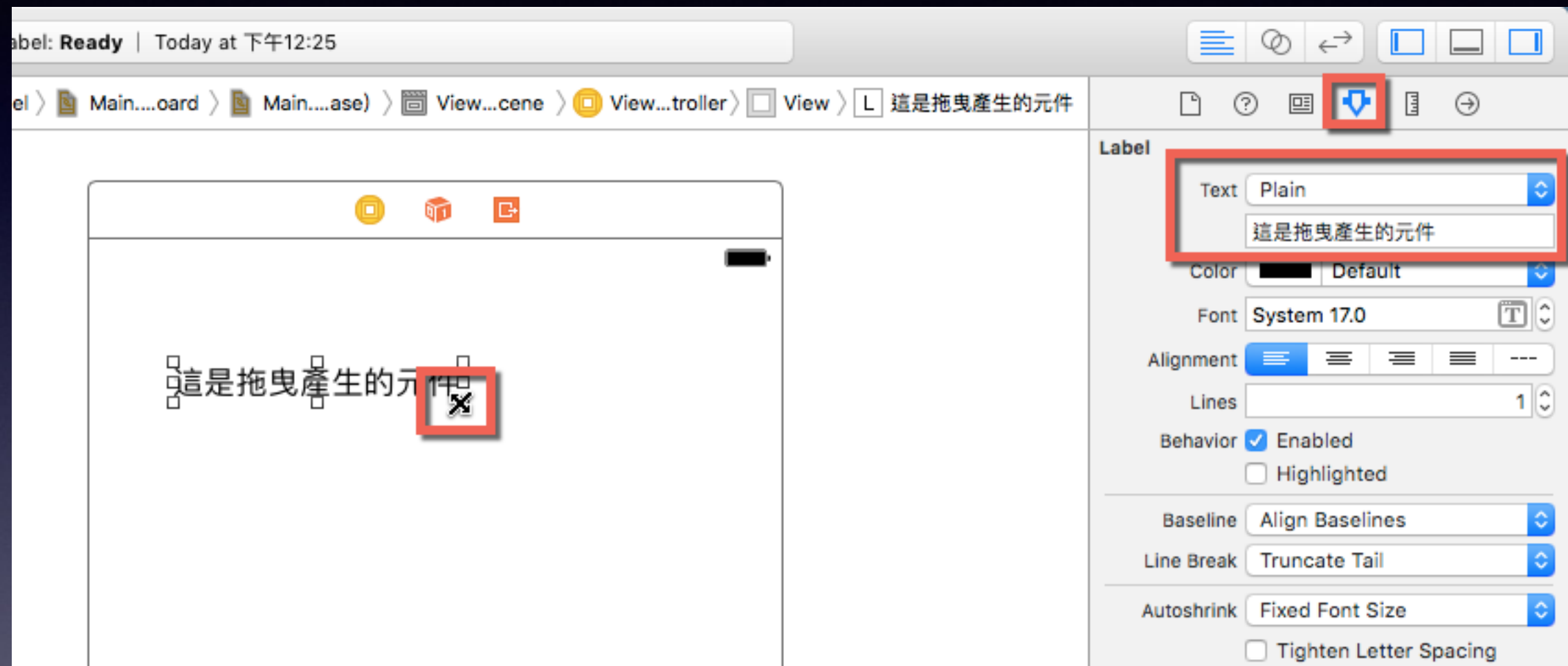
Object - Provides a template for objects and controllers not directly available in Interface Builder.

Label

Label - A variably sized amount of static text.

Button - Intercepts touch events and

- 點按屬性設定區上方  鈕設定元件屬性
- Text 屬性文字輸入“這是拖曳產生的元件”
- Color 屬性選擇藍色
- 拖曳元件右下方的小方塊改變元件顯示大小



以程式碼建立 Label 元件

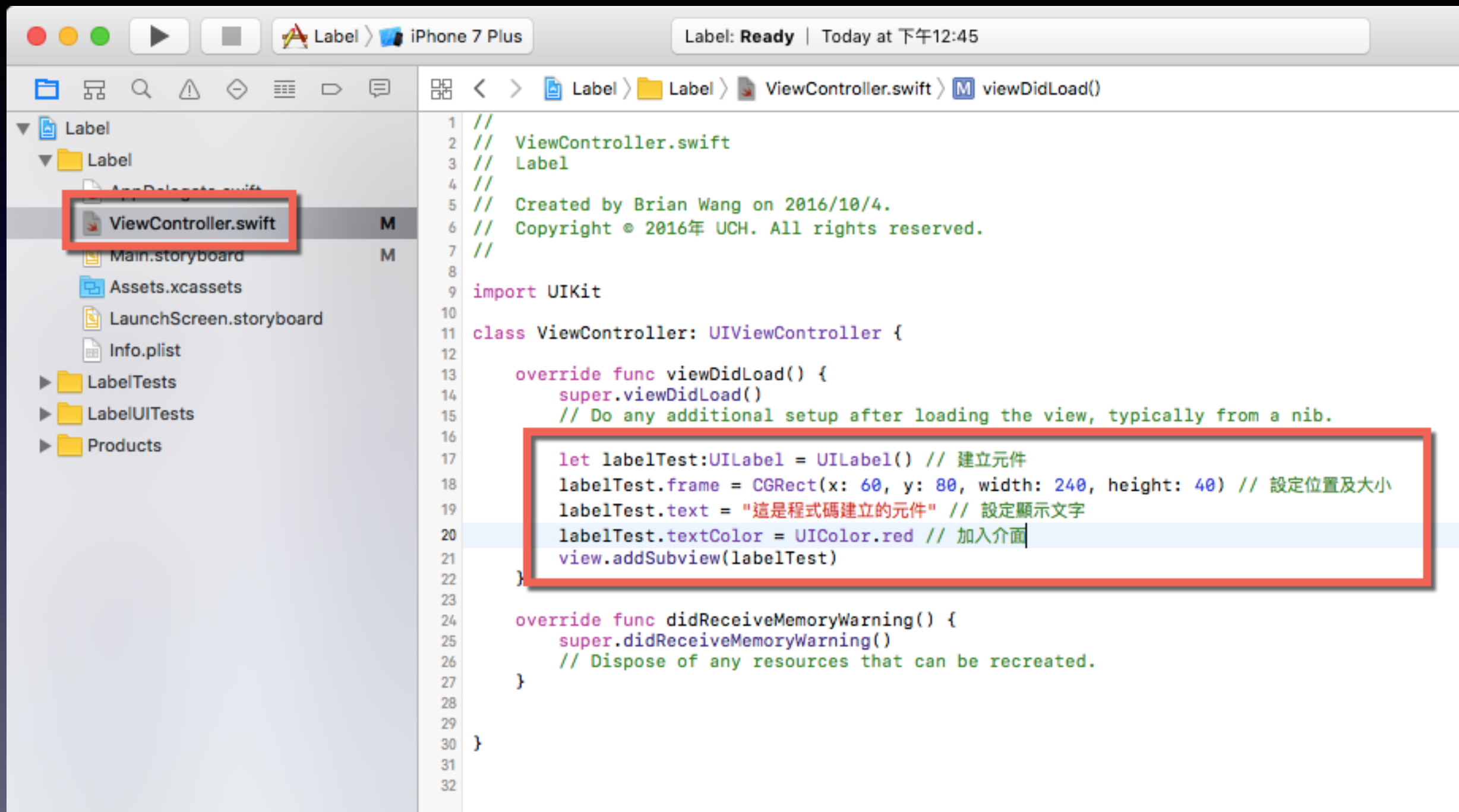
- 除了可使用元件區拖曳元件到介面設計區的方式建立元件，也可用程式碼建立元件
- 使用程式碼建立元件時新增元件更有彈性

- 首先宣告一個變數作為元件名稱
 - `var 變數名稱 : 元件名稱 = 元件類別 ()`
 - Label 元件的元件類別為 UILabel
 - `var labelTest : UILabel = UILabel ()`

- 接著以 frame 屬性設定元件的位置及大小
 - 變數名稱.frame = CGRect (x:x座標, y:y座標, width:寬度, height:高度)
 - labelTest.frame = CGRect (x:50, y:100, width:200, height:30)

- 至此已新增完成一個 Label 元件
- 可依需要設定元件屬性值
 - 變數名稱.屬性名稱 = 屬性值
 - labelTest.text = "測試元件"

- 最後將元件加入介面配置，元件才會在螢幕中顯示
 - `view.addSubview (變數名稱)`
 - `view.addSubview (labelTest)`



文字元件

- Label 元件只能顯示文字資訊，無法讓使用者輸入資料
- 要讓使用者輸入資料必須使用文字元件
- 文字元件包括只能輸入單列文字的 Text Field 元件及可輸入多列文字的 Text View 元件

Text Field 元件常用屬性

Text	設定元件輸入文字的特性及內容
Color	可由右方下拉式選單選取文字顏色。預設值為黑色
Font	設定文字的字體及大小。預設字體為 System，預設大小為 14
Alignment	設定文字的排列方式。預設值為靠左排列
Background	設定元件的背景圖片
Disabled	設定元件無作用時的背景圖片
Border Style	設定元件外框的形狀。預設值為橢圓形
Clear Button	設定清除元件文字按鈕的時機。預設值為 Never appears

- 操作元件文字
 - 應用程式執行時，先點選 Text Field 元件，再點按元件中的文字，元件下方會出現選取及貼上功能選單

- Border Style 屬性
 - Border Style 屬性設定元件外框的形狀
 - 有 4 種設定值：無外框、方形 1、方形 2、橢圓形

- Clear Button 屬性
 - Clear Button 設定清除元件文字按鈕的時機
 - 有 4 種設定值
 - Never appears：永遠不顯示。此為預設值
 - Appears while editing：元件處於編輯狀態時才顯示
 - Appears unless editing：當使用者正在編輯時才顯示
 - Is always visible：永遠都顯示

Text View 元件常用屬性

Text	設定元件輸入文字的特性及內容
Color	可由右方下拉式選單選取文字顏色。預設值為黑色
Font	設定文字的字體及大小。預設字體為 System，預設大小為 14
Alignment	設定文字的排列方式。預設值為靠左排列
Editor	設定文字是否可編輯。此屬性是核取方塊，預設值為可編輯
Selectable	設定文字是否可選取。此屬性是核取方塊，預設值為可選取
Background	設定元件的背景圖片

- 元件文字預設值
 - 使用者由元件區拖曳 Text View 元件到介面設計區時，預設已有一大篇文字，執行時在元件內滑動就可捲動文字
 - 選取、複製等文字操作的方式與 Text Field 元件相同

- 元件背景顏色
- Text View 元件沒有 Border Style 屬性設定邊界，預設值是沒有邊界
- 當 Text View 元件沒有文字時，在介面上完全看不見
- Text View 元件用於讓使用者輸入資料時，通常會為其設定背景色，讓使用者清楚元件範圍

輸入單列及多列文字

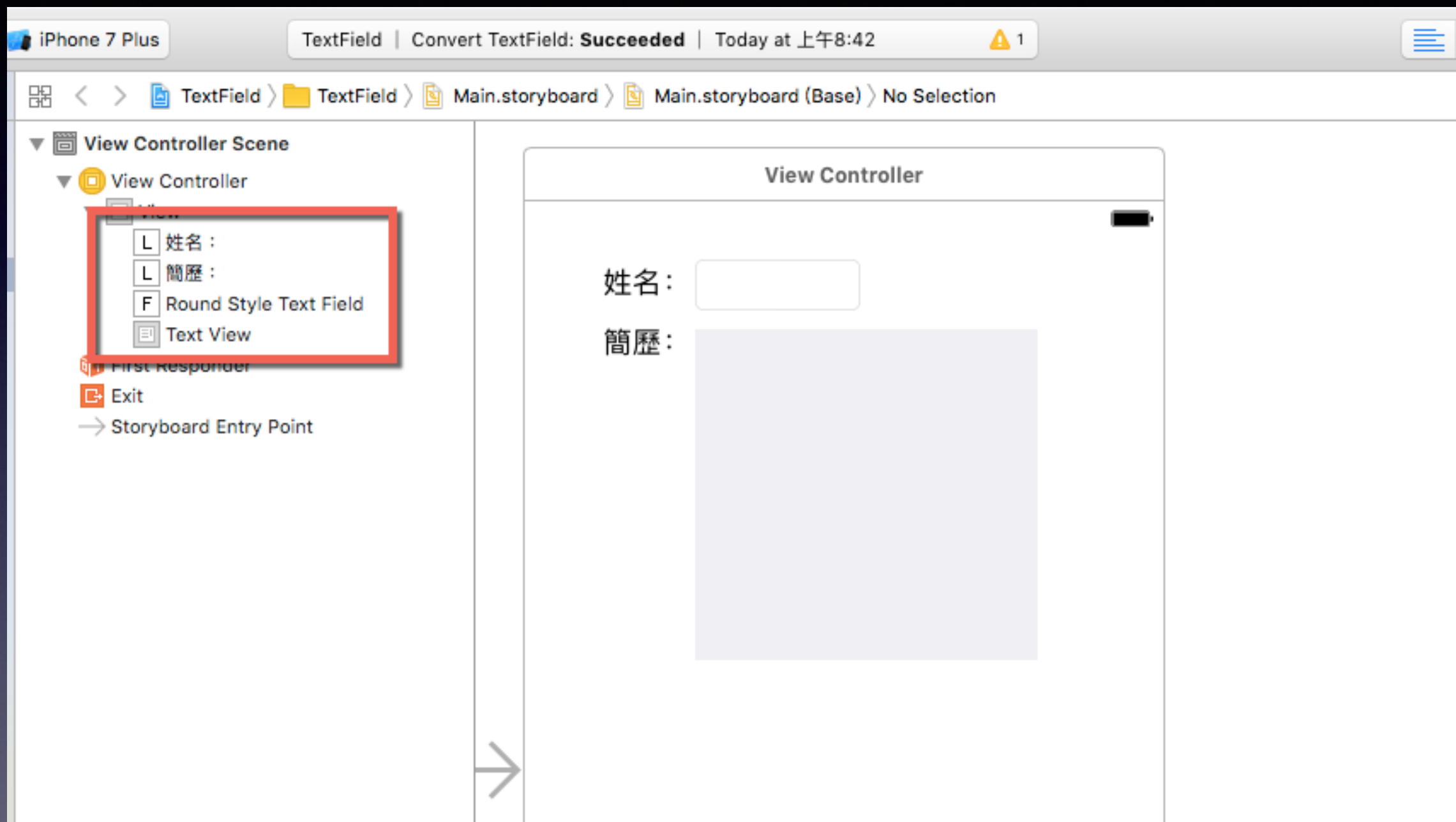
- 建立名稱為 TextField 的 Single View 專案
- 姓名 欄位利用 Text Field 元件讓使用者輸入單列資料
- 簡歷 欄位利用 Text View 元件讓使用者輸入多列資料

姓名：

簡歷：

- 介面配置

- 拖曳 2 個 Label 元件，將其文字改為 姓名 及 簡歷
- 拖曳 Text Field 元件置於 姓名 欄位右方
- 拖曳 Text View 元件置於 簡歷 欄位右方，並清除其預設文字，再設定 Background 屬性為淡藍色



Button 元件

- Button 元件的功能就是建立按鈕
- 使用者按下介面中的按鈕，應用程式根據按鈕特性給予回應與使用者互動

Button 元件常用屬性

Type	設定按鈕顯示型態：System 表示以文字顯示，此為預設值
Title	設定顯示的文字特性及內容
Font	設定文字的字體及大小。預設字體為 System，預設大小為 15
Text Color	可由右方下拉式選單選取文字顏色。預設值為藍色
Shadow Color	設定文字陰影的顏色。預設值為灰色
Image	設定按鈕圖片
Button 的 Background	設定按鈕的背景圖片
Shadow Offset	設定文字陰影的偏移量，Horizontal 為水平偏移量，Vertical 為垂直偏移量
Enable	設定按鈕是否有作用。這是核取方塊，預設值為有作用
View 的 Background	設定按鈕的背景顏色

- 元件的繼承關係
 - Button 元件繼承 Control 類別
 - Control 又繼承 View 類別
 - View 類別也有 Background 屬性，功能是設定按鈕的背景顏色

元件的繼承關係會
呈現於屬性面板

Button 繼承 Control

Control 繼承 View

Button

Type System

State Config Default

Title Plain

確定

Font System 15.0

Text Color Default

Shadow Color

Image Default Image

Background Default Background Image

Shadow Offset 0 0

Width Height

☐ Reverses On Highlight

Drawing ☐ Shows Touch On Highlight

☒ Highlighted Adjusts Image

☒ Disabled Adjusts Image

Line Break Truncate Middle

Control

Alignment

Horizontal

Vertical

State ☐ Selected

☒ Enabled

☐ Highlighted

View

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

Interaction ☒ User Interaction Enabled

☐ Multiple Touch

Alpha 1

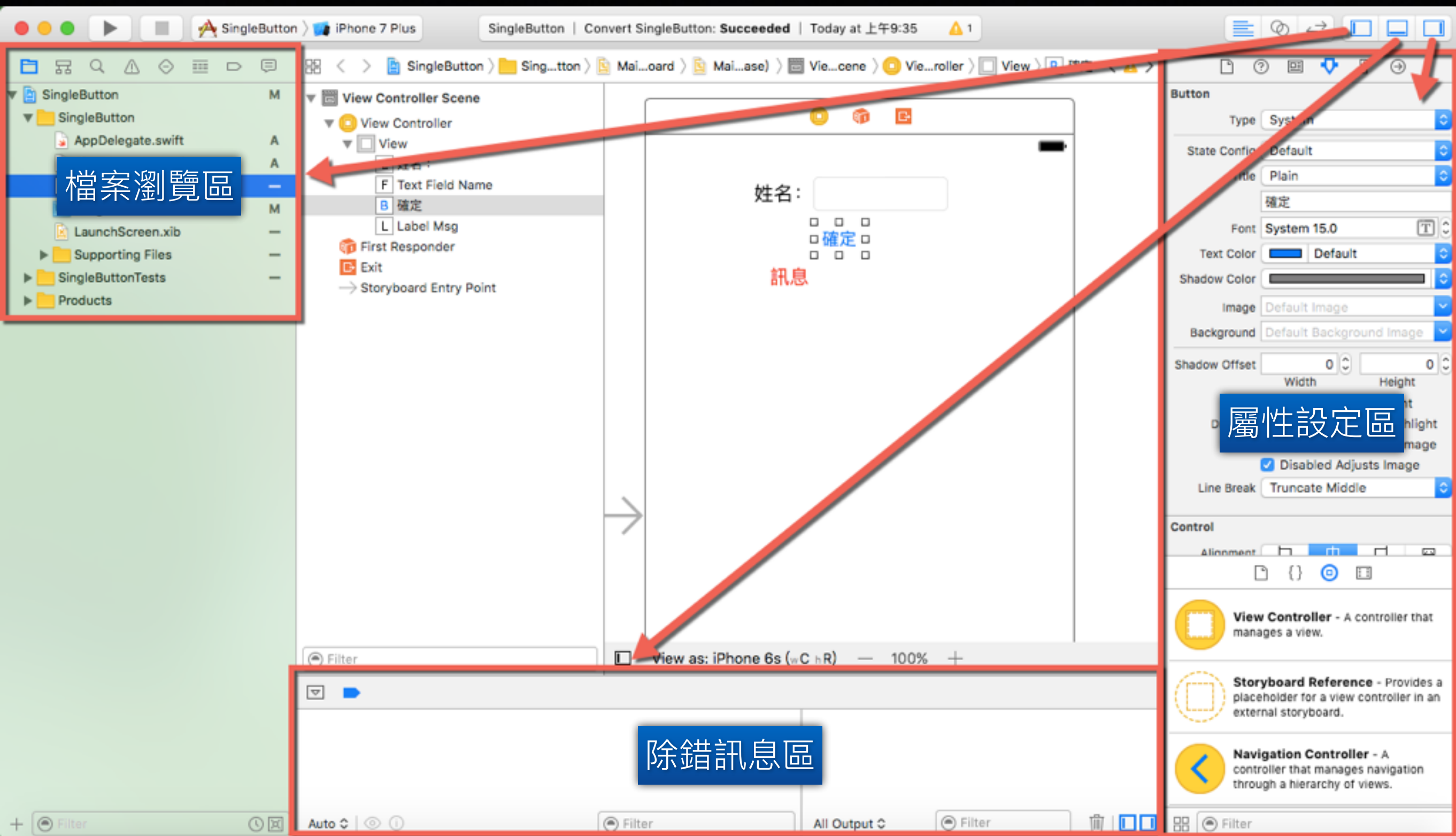
Background

Tint Default

Button 的 Background 屬性

View 的 Background 屬性

- 按鈕文字及圖形
 - Button 元件的 Title 屬性設定按鈕文字，Image 屬性設定按鈕圖形
 - 兩個屬性都設定的話，文字與圖形都會顯示，圖形在左，文字在右
 - 按鈕文字預設為藍色，但加入圖形後文字顏色會自動變為淺灰色看不清楚，需自行修改 Text Color 屬性為其他較明顯的顏色



Button 元件觸發事件的處理程序

- 建立需要使用的元件與程式碼檔案連結並命名
- 建立按鈕元件觸發事件與程式碼檔案連結並命名
- 在觸發事件中撰寫處理程式碼

歡迎訊息

- 建立名稱為 SingleButton 的 Single View 專案
- 若未輸入姓名就按 確定 鈕，會顯示必須輸入姓名的提示訊息
- 輸入姓名後按 確定 鈕，則顯示歡迎訊息

姓名:

確定

訊息

- 在 Text Field 元件上按滑鼠右鍵，拖曳快顯功能表 New Referencing Outlets 項目右方的圓點到程式碼適當位置
- 在連結對話方塊中 Name 欄位輸入 textFieldName 作為元件名稱
- 按 Connect 鈕完成操作

- 以相同方式建立 Label 元件連結，名稱為 labelMsg
- 在 Button 元件上按滑鼠右鍵，拖曳快顯功能表 Touch Up Inside 項目右方的原點到程式碼適當位置
- 在連結對話方塊中 Name 欄位輸入 sureClick 做為方法名稱
- Type 欄位選取 UIButton，按 Connect 鈕完成操作

```
1 import UIKit
2
3 class ViewController: UIViewController {
4     @IBOutlet var textFieldName: UITextField! //連結姓名輸入欄位
5     @IBOutlet var labelMsg: UILabel! //連結訊息欄位
6
7     //建立按鈕 Touch Up Inside 事件
8     @IBAction func sureClick(_ sender: UIButton) {
9         if textFieldName.text != "" {
10             labelMsg.text = (textFieldName.text)! + "，歡迎光臨！"
11         } else {
12             labelMsg.text = "必須輸入姓名！"
13         }
14     }
15
16     override func viewDidLoad() {
17         super.viewDidLoad()
18         // Do any additional setup after loading the view, typically from a nib.
19     }
20
21     override func didReceiveMemoryWarning() {
22         super.didReceiveMemoryWarning()
23         // Dispose of any resources that can be recreated.
24     }
25
26 }
27
28
29
```

多按鈕共用事件處理方法

- 建立共用事件處理方法
- 共用事件處理方法應用：電話鍵盤

電話鍵盤

- 建立名稱為 MultiButton 的 Single View 專案
- 按下 0、1、2 ~ 9 按鈕，會在上方顯示電話號碼
- 如果輸入 10 個數字後按 確定 鈕，會顯示撥電話訊息
- 若未輸入 10 個數字就按 確定 鈕，會顯示提示訊息
- 按 清除 鈕會移除所有數字，以便重新輸入

View Controller



電話號...

1

2

3

4

5

6

7

8

9

確定

0

清除

顯示電話訊息

- 為了讓按鈕看起來較美觀，設定其背景色為黑色，文字為白色，文字大小為 22
- labelTel 是 Label 元件，開始時沒有內容，預設背景是白色，執行時看不到此元件，因此將背景設為灰色
- 按鈕 0 已建立 Touch Up Inside 事件連結 numberClick 方法
- 其他數字按鈕也都要連結到此方法

```
1 import UIKit
2
3 class ViewController: UIViewController {
4     @IBOutlet weak var labelTel: UILabel! //連結電話欄位
5     @IBOutlet weak var labelMsg: UILabel! //連結訊息欄位
6
7     //所有數字按鈕共用事件處理方法
8     @IBAction func numberClick(_ sender: UIButton) {
9         labelTel.text = labelTel.text! + sender.currentTitle!
10    }
11
12    @IBAction func sureClick(_ sender: UIButton) { //按 確定 鈕
13        if labelTel.text?.lengthOfBytes(using: String.Encoding.utf8) == 10 { //輸入10個數字
14            labelMsg.text = "撥打電話：" + labelTel.text!
15        } else {
16            labelMsg.text = "必須輸入10個數字！"
17        }
18    }
19
20    @IBAction func clearClick(_ sender: UIButton) { //按 清除 鈕
21        labelTel.text = ""
22    }
23
24    override func viewDidLoad() {
25        super.viewDidLoad()
26        // Do any additional setup after loading the view, typically from a nib.
27    }
28
29    override func didReceiveMemoryWarning() {
30        super.didReceiveMemoryWarning()
31        // Dispose of any resources that can be recreated.
32    }
33
34 }
35
36
37
```

以程式碼建立多個按鈕元件

- 使用程式碼來建立元件可利用迴圈一次建立多個
- 在宣告變數時必須加入按鈕類型，否則加入介面時不會再介面中顯示按鈕

```
var 變數名稱 : UIButton = UIButton (type : 按鈕類型) as UIButton
變數名稱.frame = CGRect (x : x 座標, y : y 座標, width : 寬度, height : 高度)
變數名稱.屬性名稱 = 屬性值
... ..
view.addSubview (變數名稱)
```

```
var buttonTest : UIButton = UIButton(type : UIButtonType.System) as UIButton
buttonTest.frame = CGRect(x : 20, y : 50, width : 40, height : 35)
buttonTest.setTitle(“確定”, forState : UIControlState.Normal)
view.addSubview (buttonTest)
```


- 建立 Button 元件時通常會以 addTarget 方法加入事件處理程序


```
變數名稱.addTarget (self, action : "處理方法名稱 :",  
    forControlEvents : UIControlEvents.事件名稱)
```

```
func 處理方法名稱 (sender : UIButton) {  
    程式碼  
    ... ..  
}
```

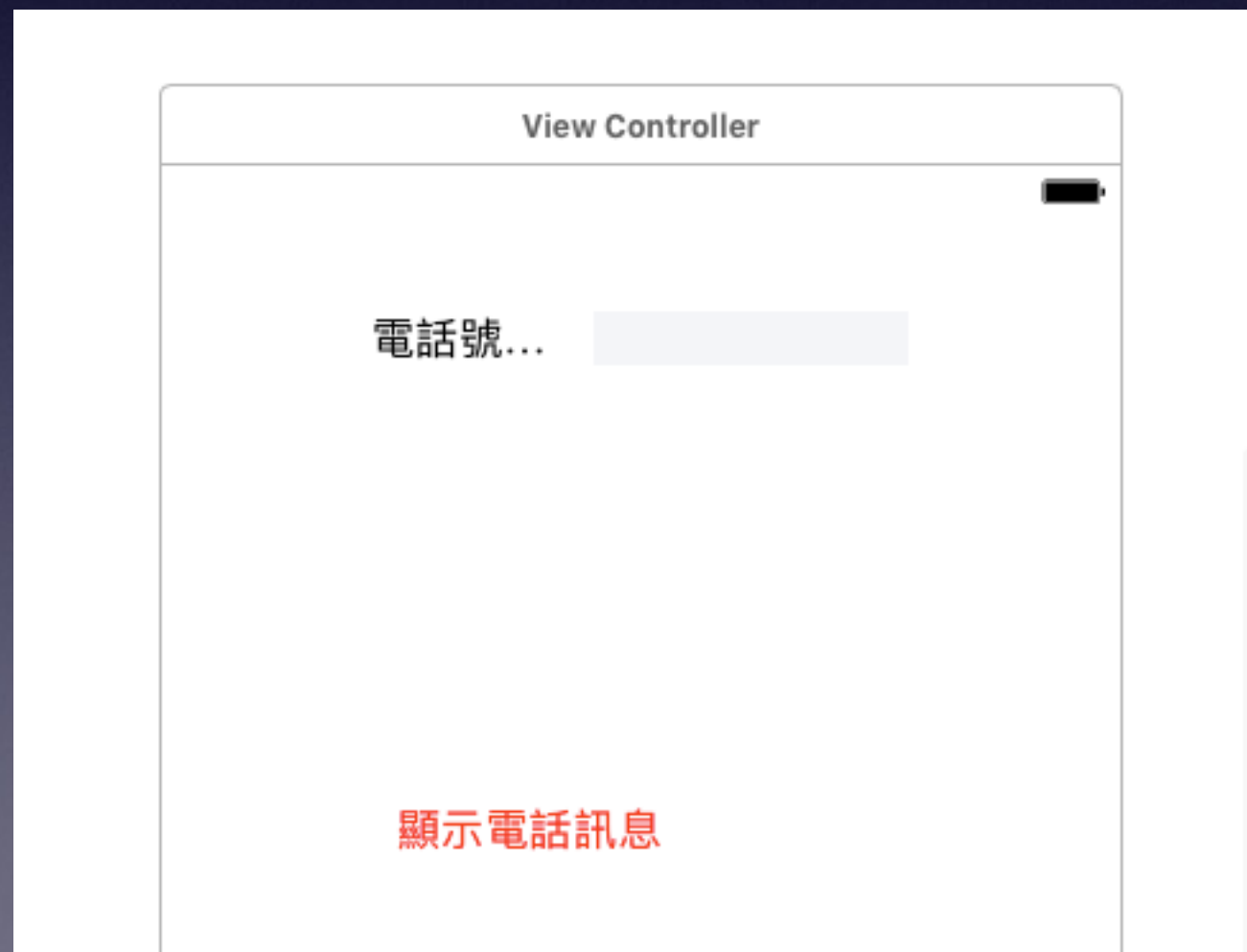
```
buttonTest.addTarget(self, action: "testClick:",  
    forControlEvents : UIControlEvents.TouchUpInside)
```

```
func testClick(sender: UIButton) {  
    var result = sender.currentTitle!  
}
```

```
for var i = 0; i < 10; ++i {  
    建立按鈕的程式碼  
}
```

電話鍵盤進階版

- 建立名稱為 ProgramButton 的 Single View 專案



```

import UIKit

class ViewController: UIViewController {
    @IBOutlet var labelTel: UILabel! //連結電話輸入欄位
    @IBOutlet var labelMsg: UILabel! //連結訊息欄位

    override func viewDidLoad() {
        super.viewDidLoad()

        //以程式建立12個按鈕
        for i in 0 ..< 12 += 1 {
            let x:Int = 80 + (i % 4) * 60 //按鈕水平坐標
            let y:Int = 95 + (i / 4) * 50 //按鈕垂直坐標
            let buttonNumber:UIButton = UIButton(type: UIButtonType.system) as UIButton //建立按鈕物件
            buttonNumber.frame = CGRect(x: x, y: y, width: 41, height: 35) //按鈕位置及大小
            buttonNumber.setTitleColor(UIColor.white, for: UIControlState()) //文字顏色
            buttonNumber.backgroundColor = UIColor.black //按鈕背景色
            buttonNumber.titleLabel?.font = UIFont(name: "System", size: 22.0) //字型大小
            if i==10 { //第11個是 清除 鈕
                buttonNumber.setTitle("清除", for: UIControlState())
                buttonNumber.addTarget(self, action:#selector(ViewController.clearClick(_:)), for:
                    UIControlEvents.touchUpInside)
            } else if i==11 { //第12個是 確定 鈕
                buttonNumber.setTitle("確定", for: UIControlState())
                buttonNumber.addTarget(self, action:#selector(ViewController.sureClick(_:)), for:
                    UIControlEvents.touchUpInside)
            } else {
                buttonNumber.setTitle("\(i)", for: UIControlState())
                buttonNumber.addTarget(self, action:#selector(ViewController.numberClick(_:)), for:
                    UIControlEvents.touchUpInside)
            }
            view.addSubview(buttonNumber)
        }

        func numberClick(_ sender:UIButton) {
            labelTel.text = labelTel.text! + sender.currentTitle!
        }

        func clearClick(_ sender:UIButton) {
            labelTel.text = ""
        }

        func sureClick(_ sender:UIButton) {
            if labelTel.text?.lengthOfBytes(using: String.Encoding.utf8) == 10 { //輸入10個數字
                labelMsg.text = "撥打電話：" + labelTel.text!
            } else {
                labelMsg.text = "必須輸入10個數字！"
            }
        }

        override func didReceiveMemoryWarning() {
            super.didReceiveMemoryWarning()
            // Dispose of any resources that can be recreated.
        }
    }
}

```


- 講義、範例程式下載：
- <https://github.com/ycwang812/UCH>

