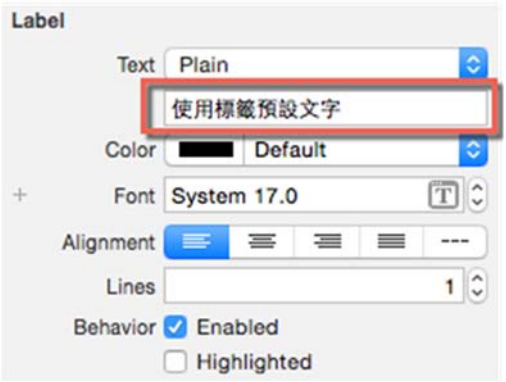
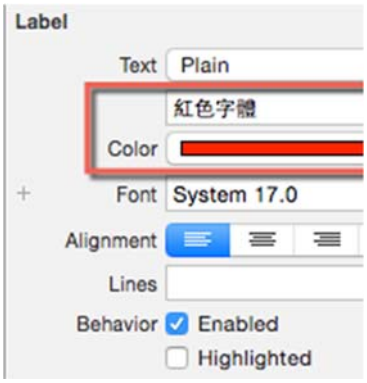
5-1

解 答

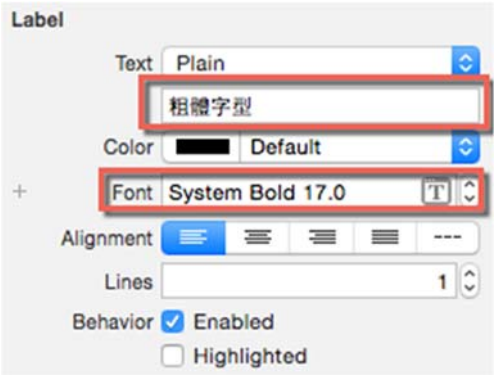
第一個標籤： 使用標籤預設文字



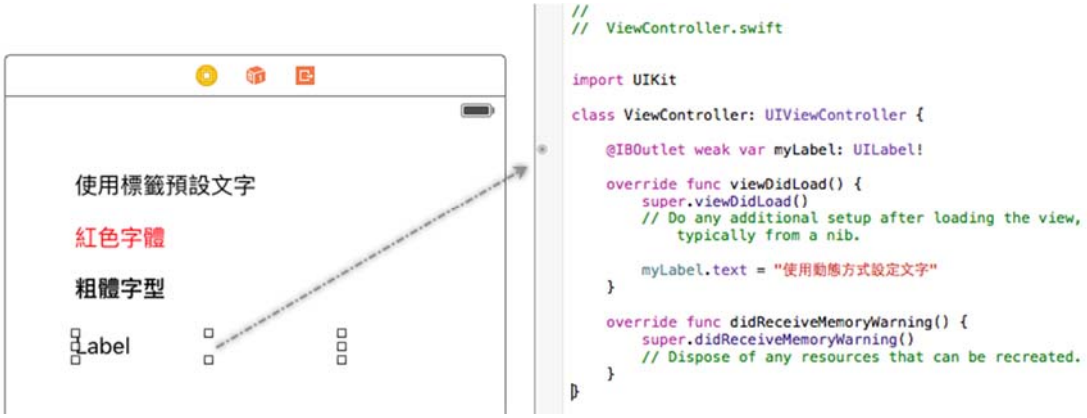
第二個標籤： 紅色字體



第三個標籤： 粗體字型



第四個標籤： 使用動態方式設定文字



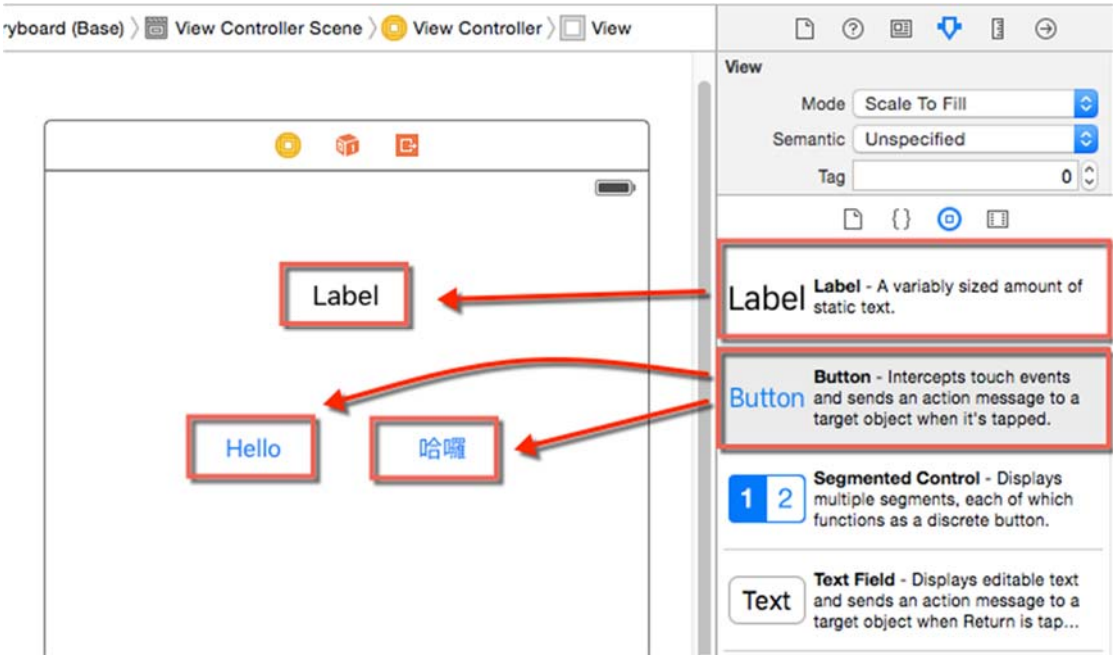

```
    @IBOutlet weak var myLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()

        myLabel.text = "使用動態方式設定文字"

    }
```

5-2

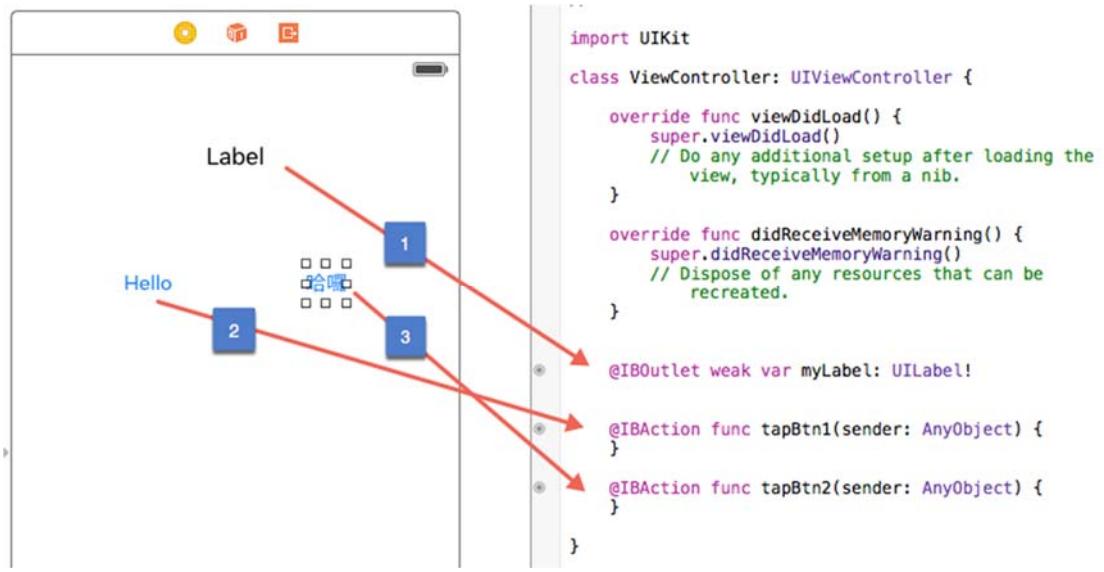
解 答

使用兩個【BUTTON】程式　　執行時，動態顯示【BUTTON】名稱為「Hello」以及「哈囉」。，當按下其中之一的【BUTTON】後，即變更【BUTTON】為指定【Label】名稱。

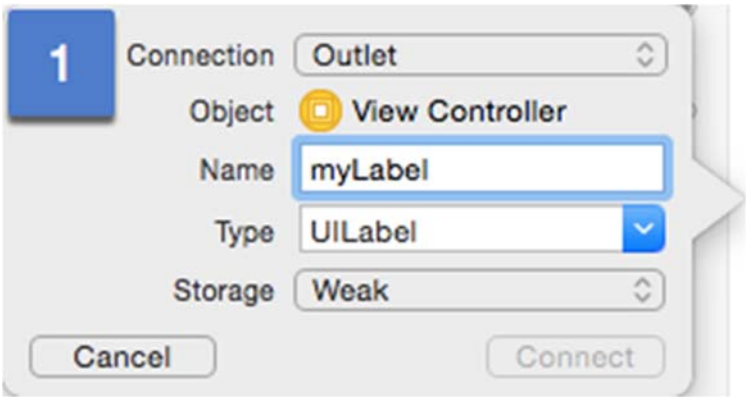
1. 分別將【文字標籤】及【操作按鈕】建立在 IB 畫面中。
2. 更改【操作按鈕】名稱為「Hello」以及「哈囉」。



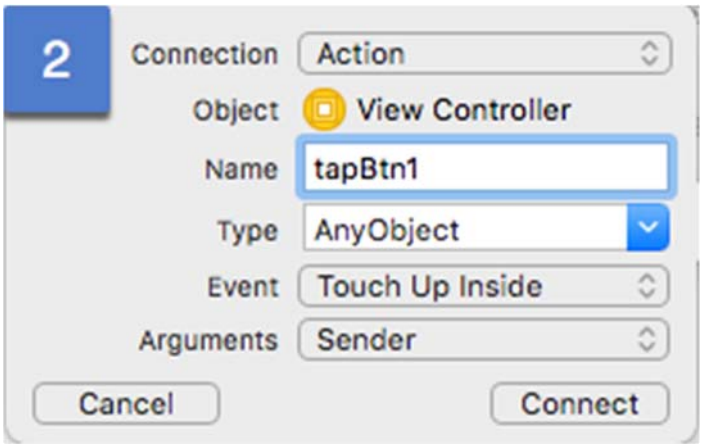
3. 在 IB 視窗上分別點選【文字標籤】【方框 1】、【操作按鈕】【方框 2】及【操作按鈕】【方框 3】，依序按住【contral】鍵，用滑鼠拖曳【操作按鈕】到右邊視窗與【程式碼】連結。



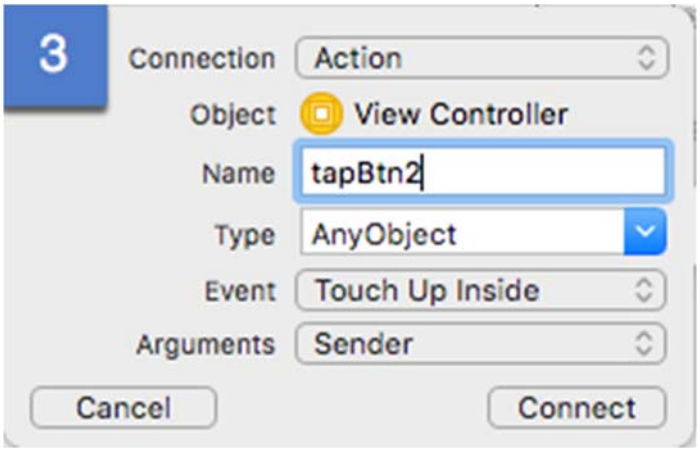
4. 在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myLabel】以及在「Type」欄位點選【UIButton】後，按【Connect】按鈕。



5. 在「Connection」欄位點選【Action】、「Name」欄位設定名稱【tapBtn1】，「Tyep」欄位點選【AnyObject】以及在「Event」欄位點選【Touch Up Inside】後，按【Connect】按鈕。



6. 在「Connection」欄位點選【Action】、「Name」欄位設定名稱【tapBtn2】，「Tyep」欄位點選【AnyObject】以及在「Event」欄位點選【Touch Up Inside】後，按【Connect】按鈕。



最後，我們在【方框 1】及【方框 2】中，鍵入學過的 swift 的【文字標籤】程式碼。

```
@IBOutlet weak var myLabel: UILabel!

@IBAction func tapBtn1(sender: AnyObject) {
    myLabel.text = "Hello"
}

@IBAction func tapBtn2(sender: AnyObject) {
    myLabel.text = "哈囉"
}
```

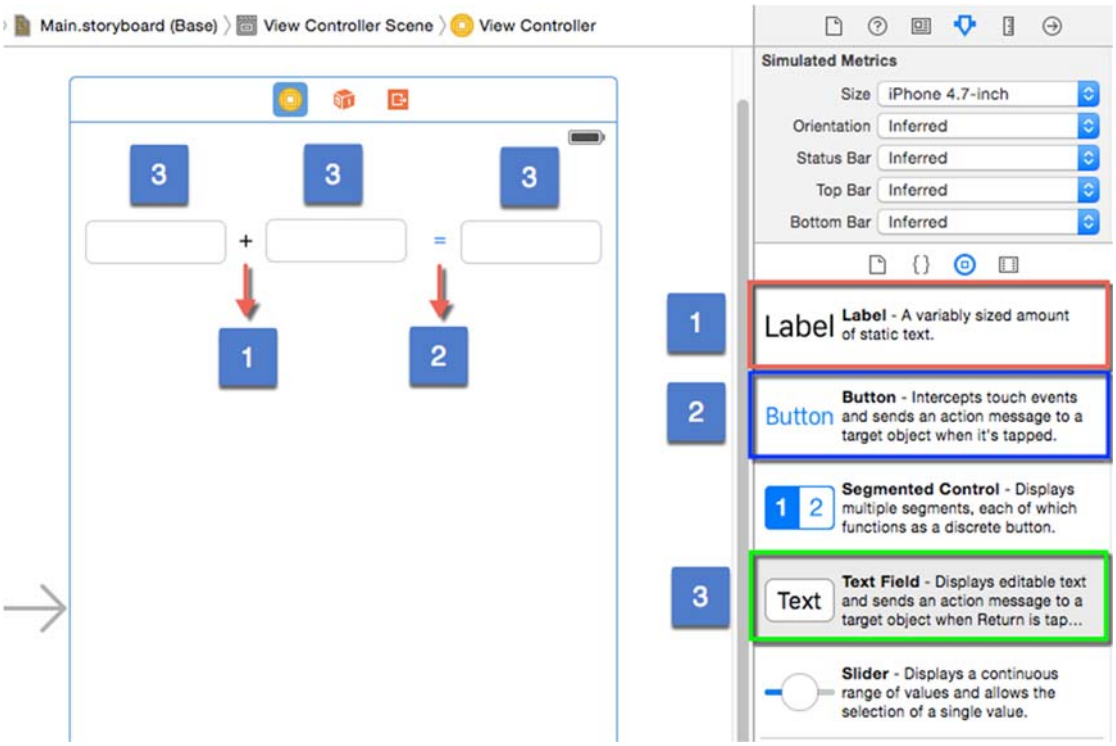
1

2

5-3

解 答

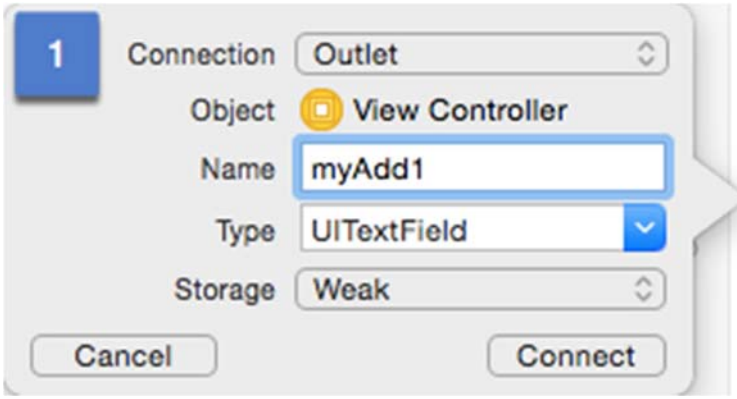
1. 將【操作按鈕】建立在畫面中。



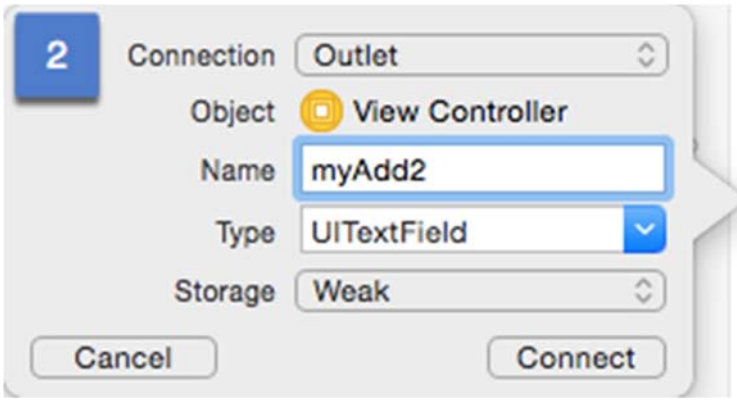
2. 在 IB 視窗上點選【文字欄位】，接著按住【contral】鍵，用滑鼠拖曳【文字欄位】依序到右邊視窗與【程式碼】連結。

```
1  @IBOutlet weak var myAdd1: UITextField!
2  @IBOutlet weak var myAdd2: UITextField!
3  @IBOutlet weak var mySum: UITextField!
4  @IBAction func tapBtn(sender: AnyObject) {
       var myValue = 0

5      myValue = Int(myAdd1.text!)! + Int(myAdd2.
           text!)!

       mySum.text = "\(myValue)"

   }
```

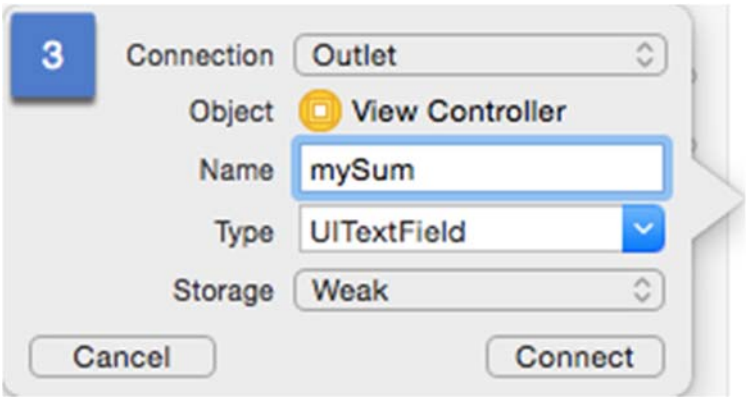
3. 在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myAdd1】以及在「Type」欄位點選【UITexField】後，按【Connect】按鈕。



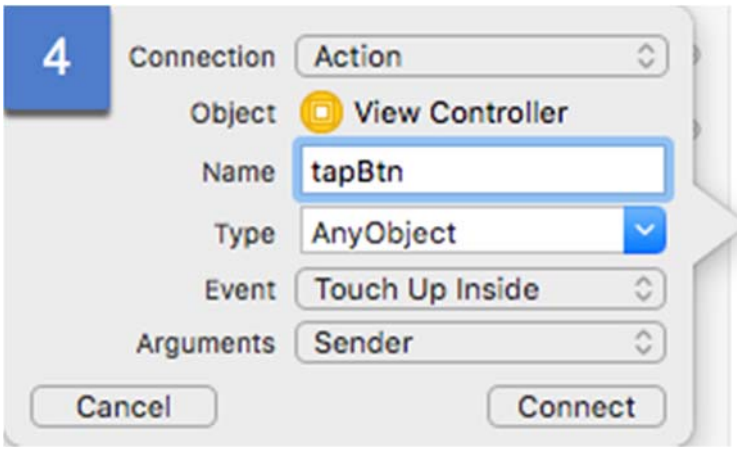
4. 在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myAdd2】以及在「Type」欄位點選【UITexField】後，按【Connect】按鈕。



5. 在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【mySum】以及在「Type」欄位點選【UITexField】後，按【Connect】按鈕。



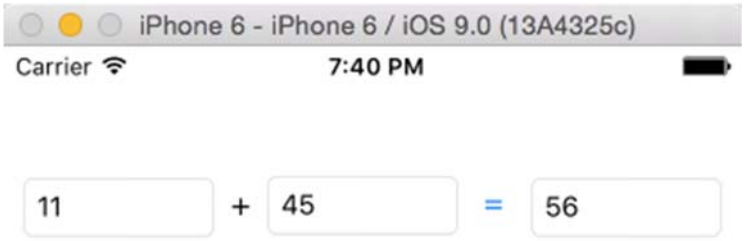
6. 在「Connection」欄位點選【Action】、「Name」欄位設定名稱【tapBtn】以及在「Event」欄位點選【Touch Up Inside】後，按【Connect】按鈕。



最後，我們在【方框 5】中設定 swift 的程式碼。

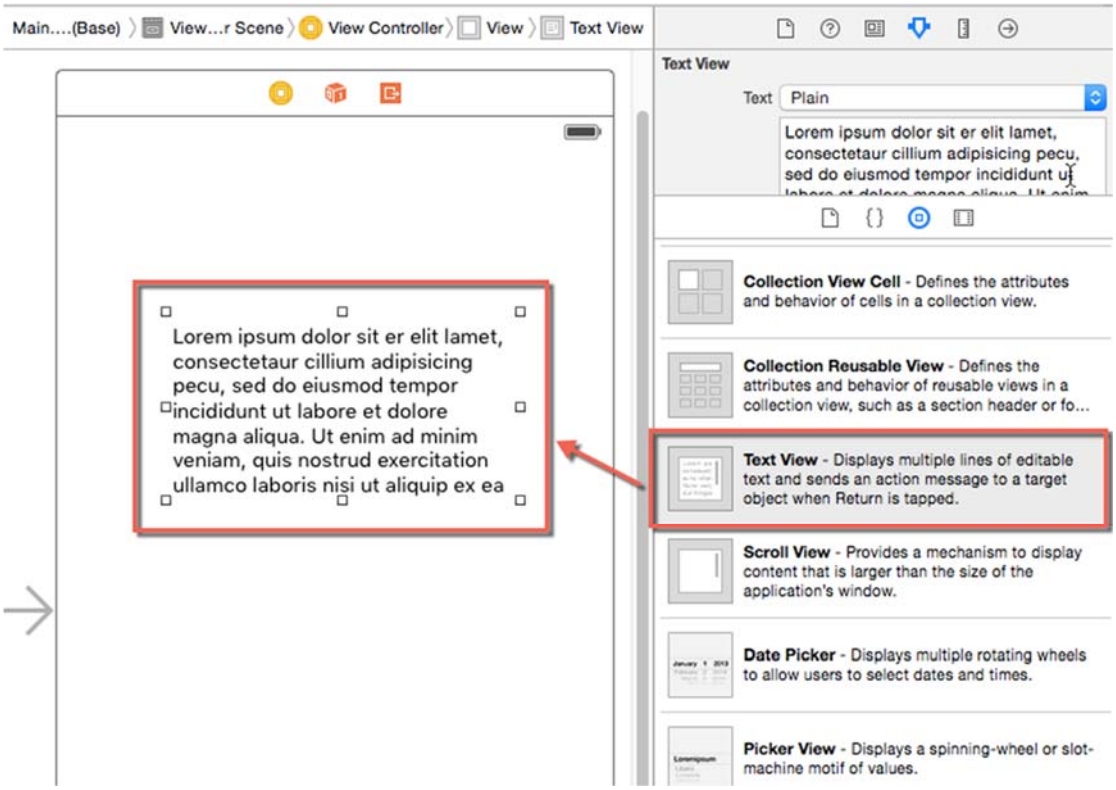
```
var myValue = 0;
myValue = Int(myAdd1.text!)! + Int(myAdd2.text!)!;
mySum.text = "\(myValue)";
```

輸入完成後，請按「＝」符號進行加總。

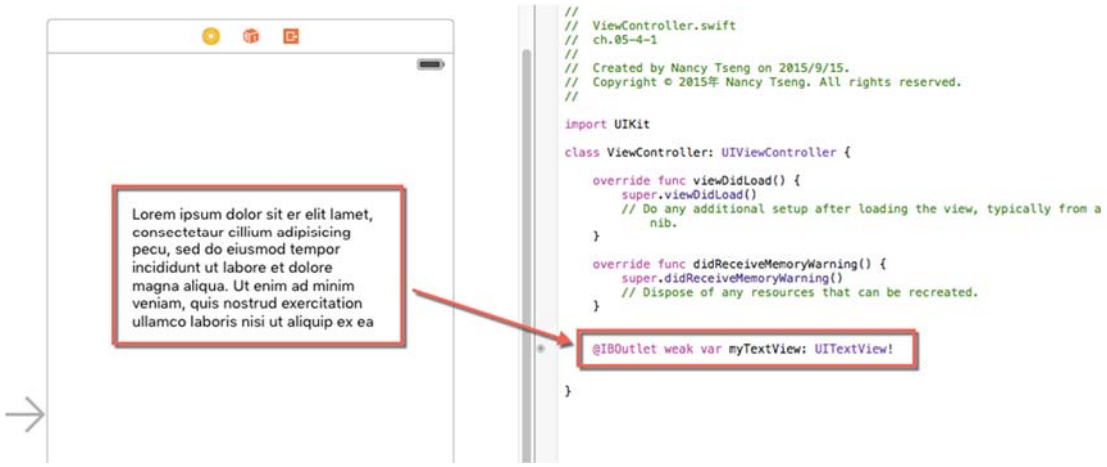


5-4 解 答

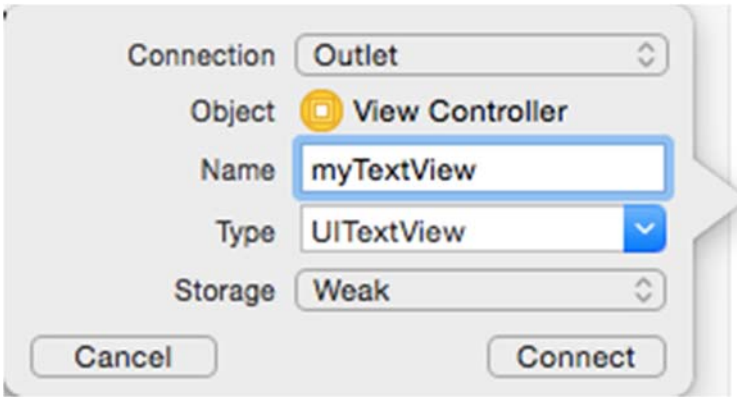
1. 將【文字方塊】建立在畫面中。



2. 在 IB 視窗上點選【文字方塊】，接著按住【control】鍵，用滑鼠拖曳【到右邊視窗與【程式碼】連結。



3. 在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myTextView】以及在「Type」欄位點選【UITexView】後，按【Connect】按鈕。



4. 最後，我們設定 swift 的程式碼，分別用設置在編輯區中。

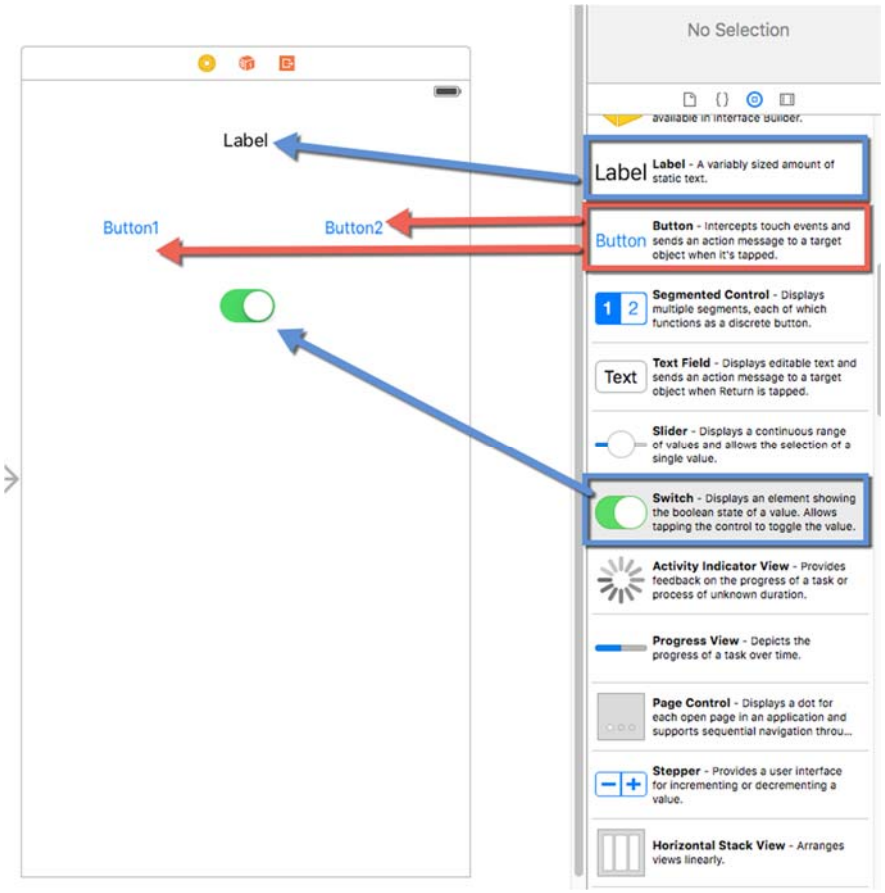
```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.

        var i:Int = 0;
        myTextView.editable = false;
        myTextView.text = "";

1       //使用 for 迴圈
         myTextView.text = myTextView.text.stringByAppendingString("\n使用 for 迴圈\n");
        for (i=0; i<10; i++) {

            myTextView.text = myTextView.text.stringByAppendingString("Hello!");
        }

2       //使用 While 迴圈
        myTextView.text = myTextView.text.stringByAppendingString("\n\n使用 While 迴圈\n");
        i=0;
        while (i<10) {
            myTextView.text = myTextView.text.stringByAppendingString("你好!");
            i++;
        }

3       //使用 do-While 迴圈
         myTextView.text = myTextView.text.stringByAppendingString("\n\n使用 do-While 迴圈\n");
        i=0;
        repeat {
            myTextView.text = myTextView.text.stringByAppendingString("厚嗨啦!");
            i++;
        } while (i<10);
```

5. 執行後畫面，可得出不同方式設定【程式碼】，但卻可以得到相同結果。

很神奇吧！
這是一個很棒的開始，慢慢透過各項演練會發現，愈來愈能夠更得心應手唷。
加油囉！

5-5 解 答

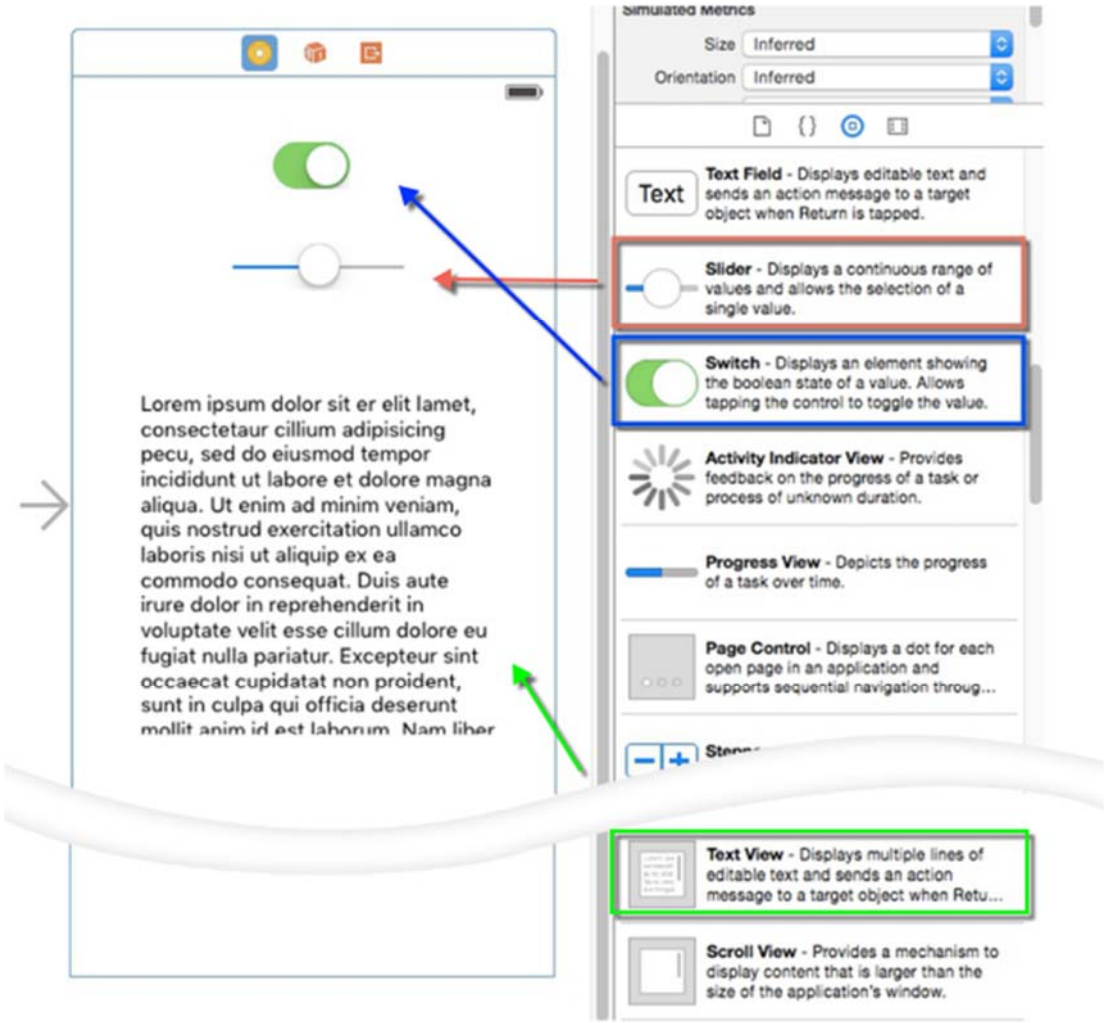
1. 將【文字方塊】建立在畫面中。



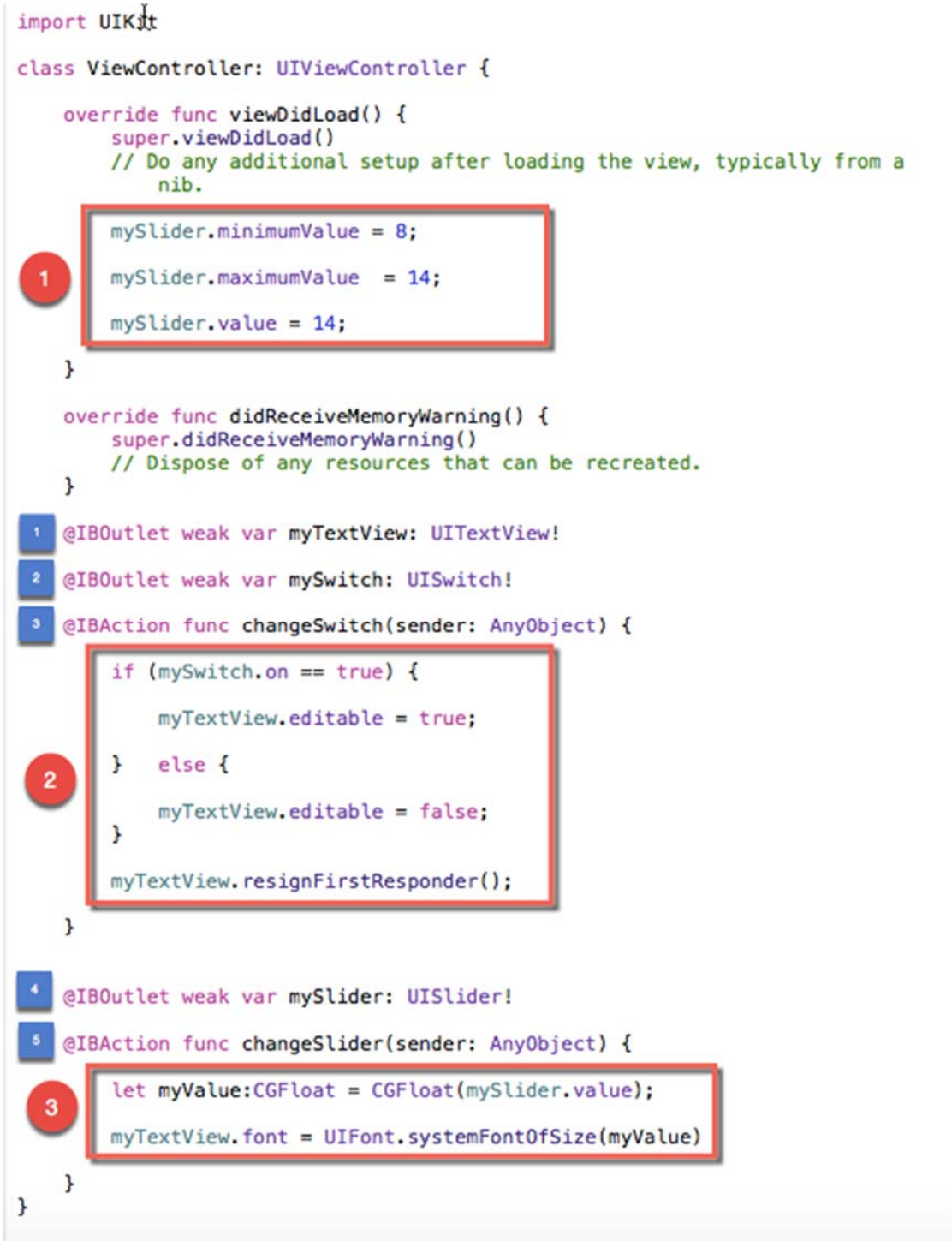
```swift
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

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

    @IBOutlet weak var myLabel: UILabel!
    @IBOutlet weak var myBtn1: UIButton!
    @IBOutlet weak var myBtn2: UIButton!
    @IBOutlet weak var mySwitch: UISwitch!

    @IBAction func changeSwitch(sender: AnyObject) {

        if (mySwitch.on == true) {
            myLabel.text = "Button 1 is ON"
            myBtn1.hidden = false
            myBtn2.hidden = true
        } else {
            myLabel.text = "Button 2 is ON"
            myBtn1.hidden = true
            myBtn2.hidden = false
        }

    }
}
```

5-6 解 答

1. 將【文字方塊】建立在畫面中。

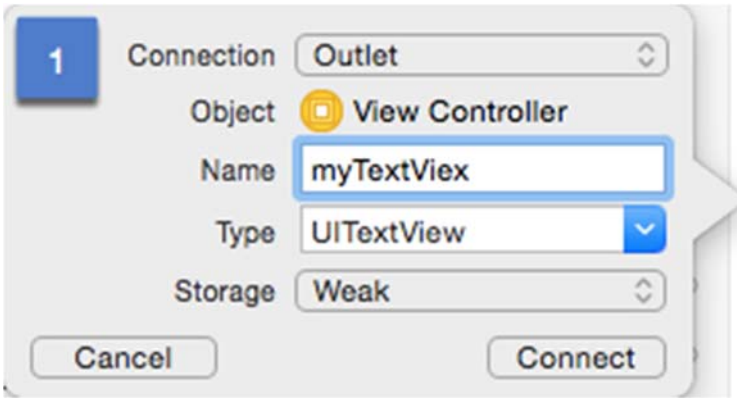


2. 在 IB 視窗上點選【文字方塊】，接著按住【control】鍵，用滑鼠拖曳【到右邊視窗與【程式碼】連結。

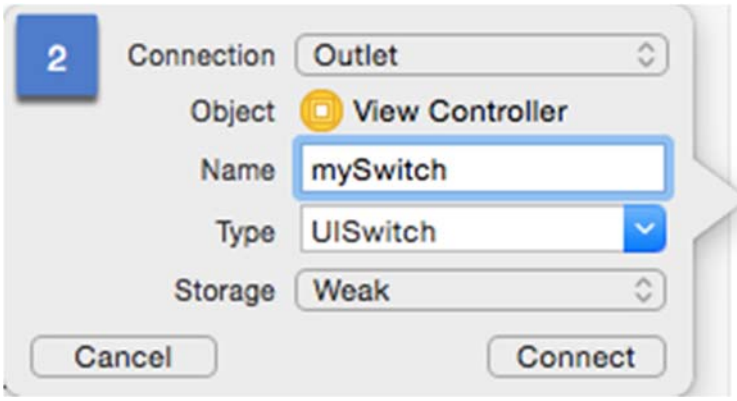


```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a
            nib.

        mySlider.minimumValue = 8;

1       mySlider.maximumValue  = 14;

        mySlider.value = 14;

    }

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

1   @IBOutlet weak var myTextView: UITextView!
2   @IBOutlet weak var mySwitch: UISwitch!
3   @IBAction func changeSwitch(sender: AnyObject) {

        if (mySwitch.on == true) {

            myTextView.editable = true;

2       }   else {

            myTextView.editable = false;
        }

        myTextView.resignFirstResponder();

    }

4   @IBOutlet weak var mySlider: UISlider!
5   @IBAction func changeSlider(sender: AnyObject) {

        let myValue:CGFloat = CGFloat(mySlider.value);

3       myTextView.font = UIFont.systemFontOfSize(myValue)

    }
}
```

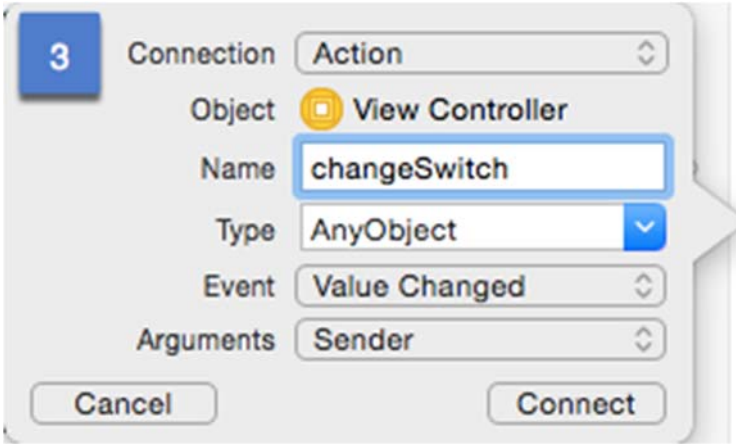
3. 拖曳【Text View】，在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myTextView】以及在「Type」欄位點選【UITexView】後，按【Connect】按鈕。



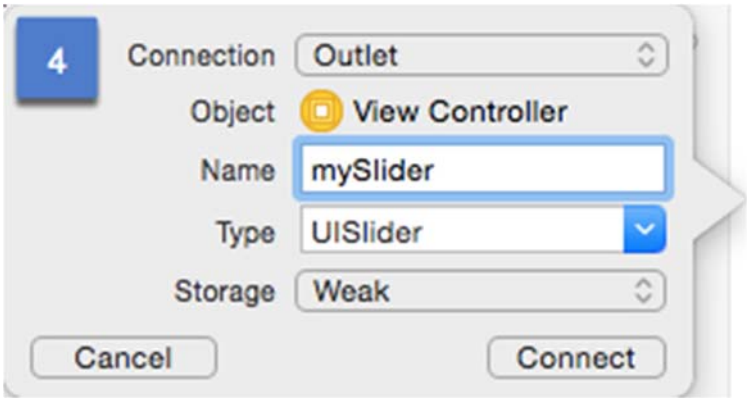
拖曳【Switch】後，在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【mySwitch】以及在「Type」欄位點選【UISwitch】後，按【Connect】按鈕。



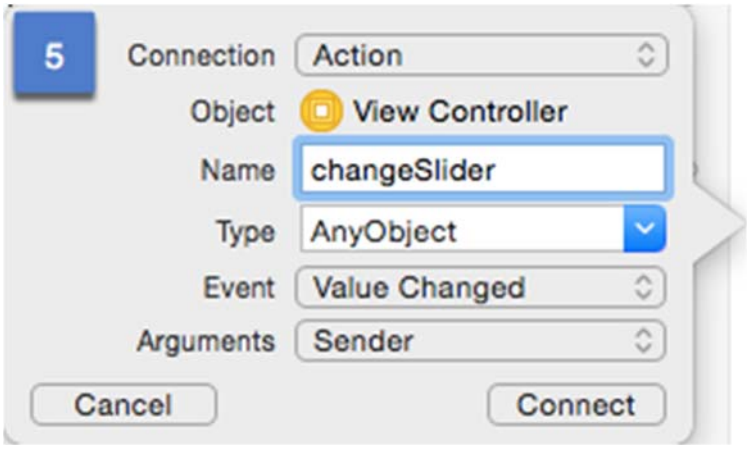
再拖曳一次【Switch】後，在「Connection」欄位點選【Action】、「Name」欄位設定名稱【changeSwitch】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。



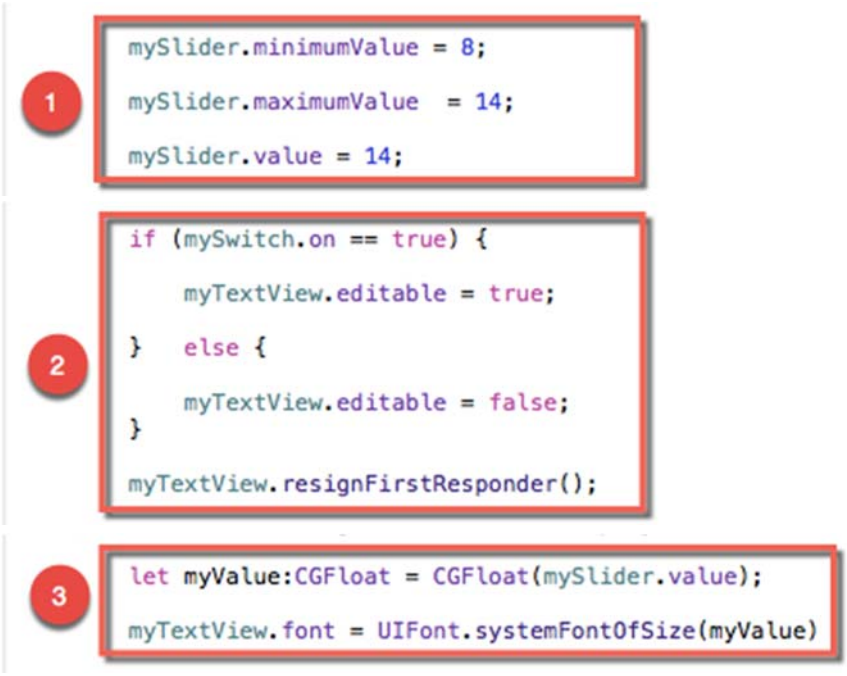
拖曳【Slider】後，在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【mySlider】以及在「Type」欄位點選【UISilder】後，按【Connect】按鈕。



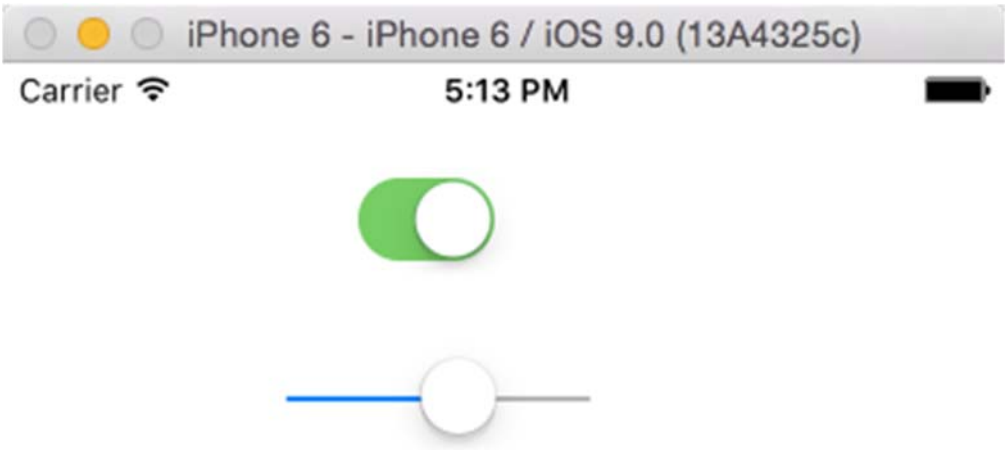
再拖曳一次【Slider】，在「Connection」欄位點選【Action】、「Name」欄位設定名稱【changeSlider】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。



4. 最後，我們設定 swift 的程式碼，分別用設置在【紅圈一】、【紅圈二】及【紅圈三】編輯區中。



5. 執行後畫面，可得出不同方式設定【程式碼】，但卻可以得到相同結果。

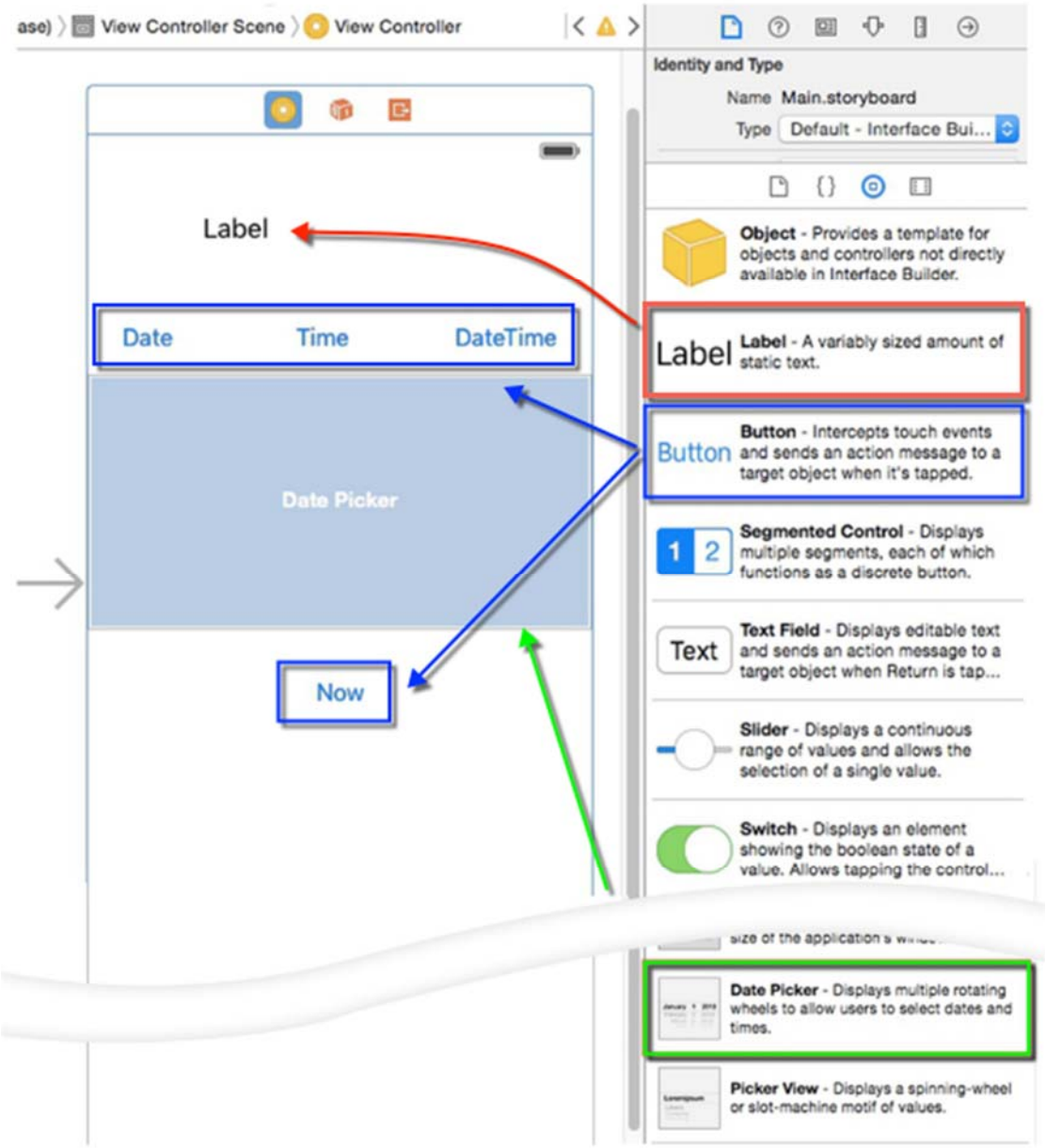


Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Nam liber te conscient to factor tum poen legum odioque civiuda.

瞧瞧～發現了神奇的變化！
當我們將前面所學演練發揮巧思結合後，透過啟發性的學習，將不同的功能綜合在一起，產生多元的操作賦予新的生命；建議讀者們，動動腦找答案會發現更有趣唷。有關於詳細的 swift 語法，建議讀者們可以參閱第三章相關說明，更能清楚瞭解其中奧妙之處。
Fighting！

5-7 解 答

1. 將【文字標籤】、【按鈕】及【日期及時間方塊】建立在 IB 畫面中。



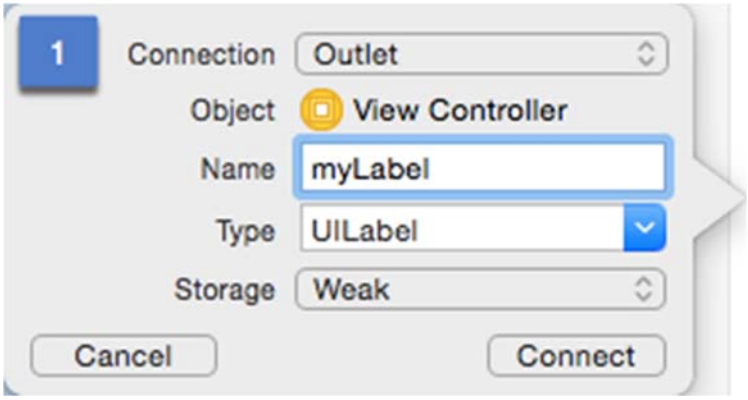
2. 在 IB 視窗上一一的點選【Label】、【Button】及【Date Picker】(為以下畫面藍色方框)，接著按住【control】鍵，用滑鼠拖曳到右邊視窗與【程式碼】連結。

```
1  @IBOutlet weak var myLabel: UILabel!
2  @IBOutlet weak var myDatePicker: UIDatePicker!
1  var myType = 3
3  @IBAction func changeDatePicker(sender: AnyObject) {
       let df = NSDateFormatter();
       switch (myType) {
       case 1:
           df.dateFormat = "yyyy/MM/dd";
       break;
2      case 2:
           df.dateFormat = "HH:mm";
       break;
       case 3:
           df.dateFormat = "yyyy/MM/dd HH:mm";
       break;
       default:
           df.dateFormat = "yyyy/MM/dd HH:mm";
       }

       myLabel.text = df.stringFromDate(myDatePicker.date);
   }
4  @IBAction func tapDate(sender: AnyObject) {
3      myType = 1
       changeDatePicker(myLabel)
   }
5  @IBAction func tapTime(sender: AnyObject) {
4      myType = 2
       changeDatePicker(myLabel)
   }
6  @IBAction func tapDateTime(sender: AnyObject) {
5      myType = 3
       changeDatePicker(myLabel)
   }
7  @IBAction func tapNow(sender: AnyObject) {
6      myDatePicker.date = NSDate()
   }
}
```

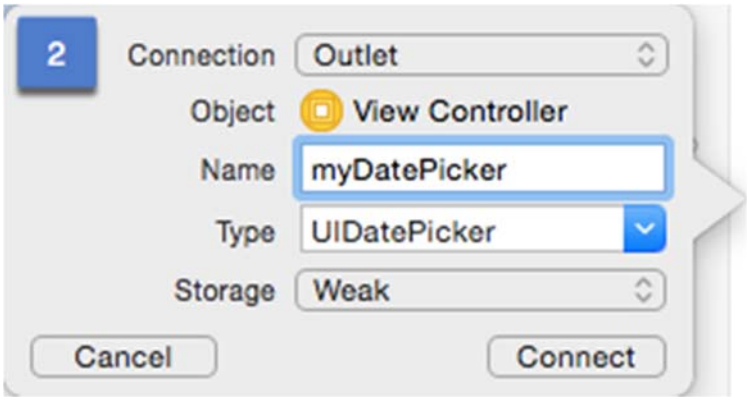
分解如下

3. 由【物件區】選取之各個元件，設定名稱如下：

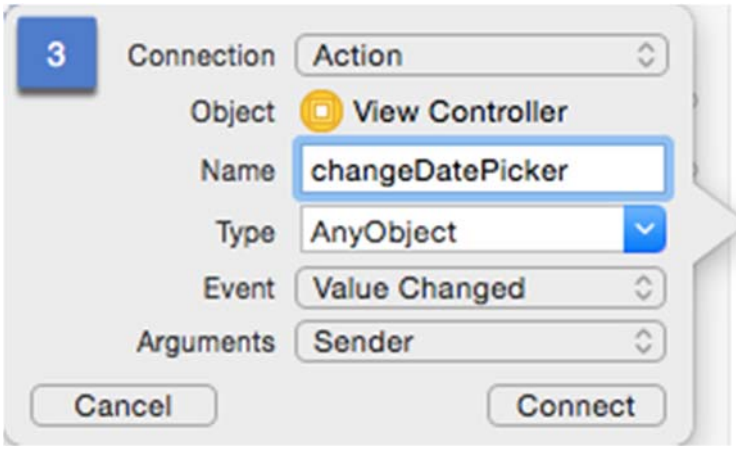
拖曳【Label】在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myLabel】以及在「Type」欄位點選【UILabel】後，按【Connect】按鈕。



拖曳【Date Picker】在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myDatePicker】以及在「Type」欄位點選【UIDatePicker】後，按【Connect】按鈕。

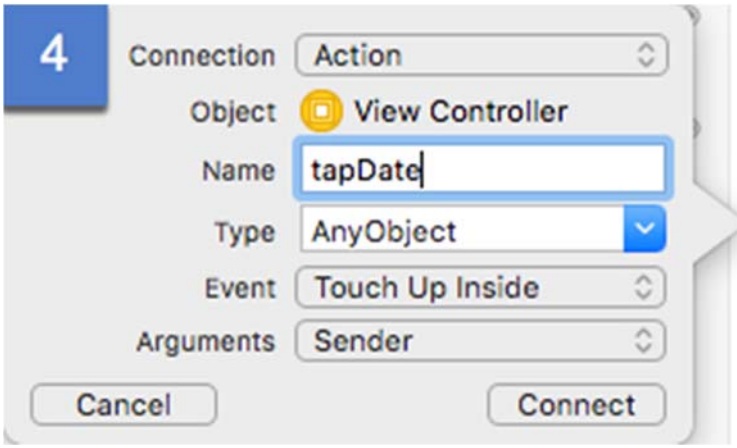


再拖曳一次【Date Picker】於「Connection」欄位點選【Action】、「Name」欄位設定名稱【changeDatePicker】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。

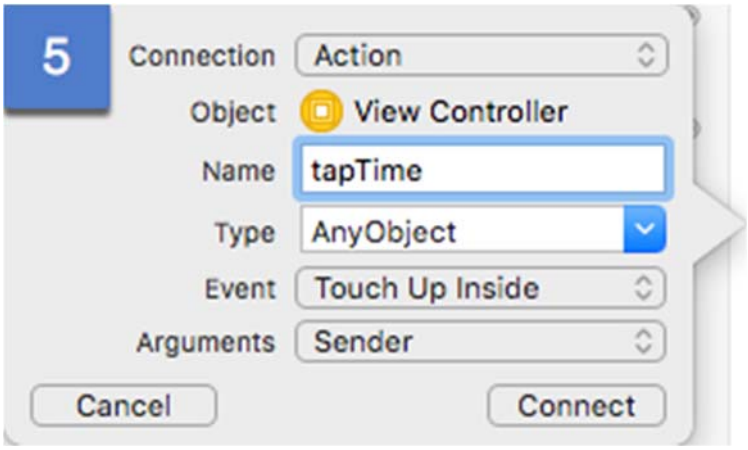


接下來，我們分別拖曳四個【Button】在 IB 畫面中，其名稱更改為「Date」、「Time」、「DateTime」及「Now」各個設定名稱如下：

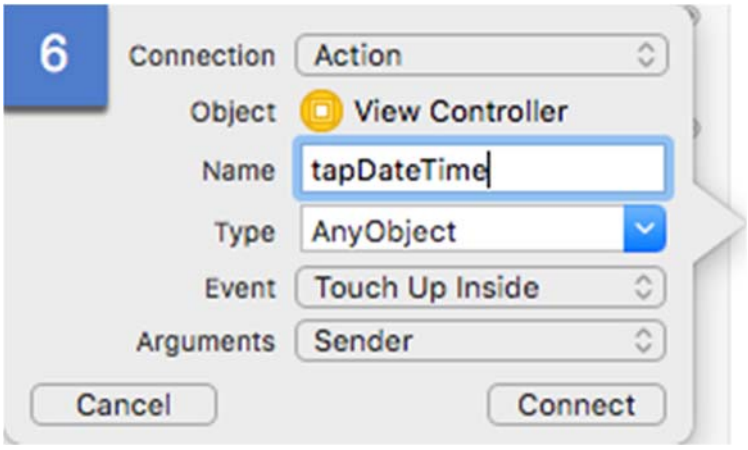
「Date」功能選項設定，在「Connection」欄位點選【Action】、「Name」欄位設定名稱【myDate】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。



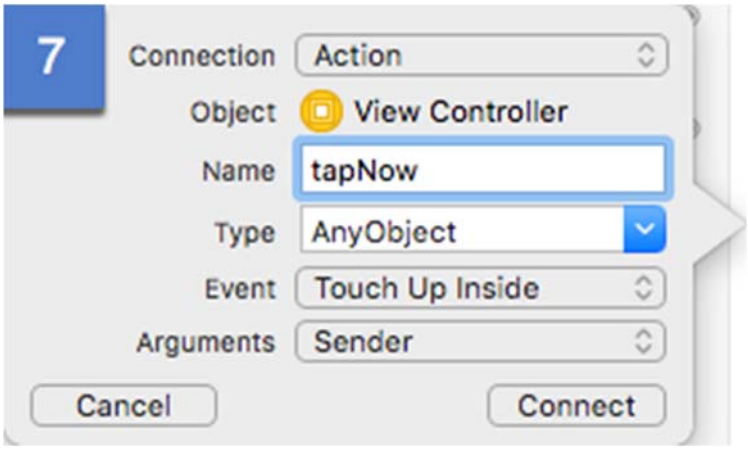
「Time」功能選項設定，在「Connection」欄位點選【Action】、「Name」欄位設定名稱【myTime】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。



「DateTime」功能選項設定，在「Connection」欄位點選【Action】、「Name」欄位設定名稱【myDateTime】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。



「Now」功能選項設定，在「Connection」欄位點選【Action】、「Name」欄位設定名稱【myNow】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。



4. 最後，我們設定 swift 的程式碼，分別用設置在【紅圈一】到【紅圈五】編輯區中。

【紅圈一】

```
var MyType = 3;
```

【紅圈二】

```
let df = NSDateFormatter();
switch (MyType) {
case 1:
        df.dateFormat = "yyyy/MM/dd";
        break;
case 2:
        df.dateFormat = "HH:mm";
        break;
case 3:
        df.dateFormat = "yyyy/MM/dd HH:mm";
        break;
default:
        df.dateFormat = "yyyy/MM/dd HH:mm";
}

myLabel.text = df.stringFromDate(myDatePicker.date);
```

【紅圈三】

```
MyType = 1;
changeDatePicker(myLabel);
```

【紅圈四】

```
MyType = 2;
changeDatePicker(myLabel);
```

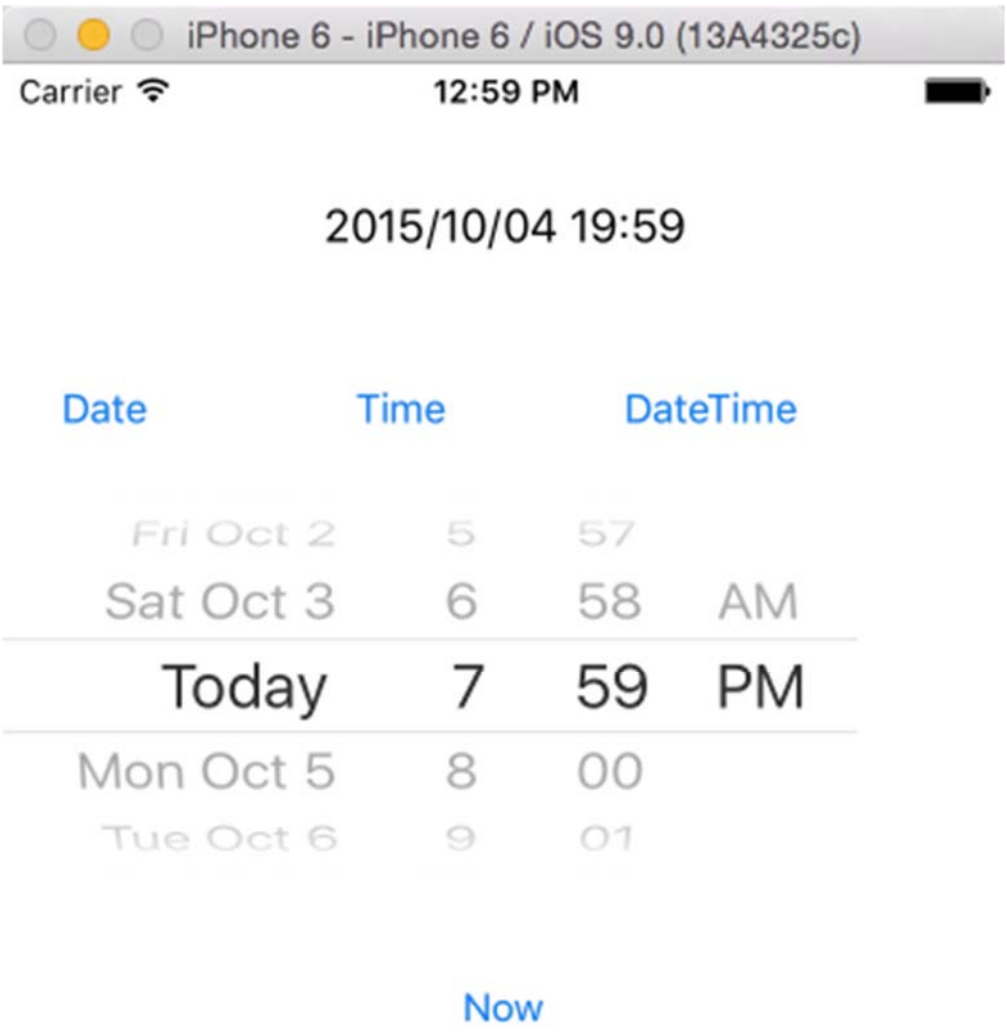
【紅圈五】

```
MyType = 3;
changeDatePicker(myLabel);
```

【紅圈六】

```
myDatePicker.date =  NSDate();
```

5. 執行後畫面，可得出不同方式設定【程式碼】，但卻可以得到相同結果。



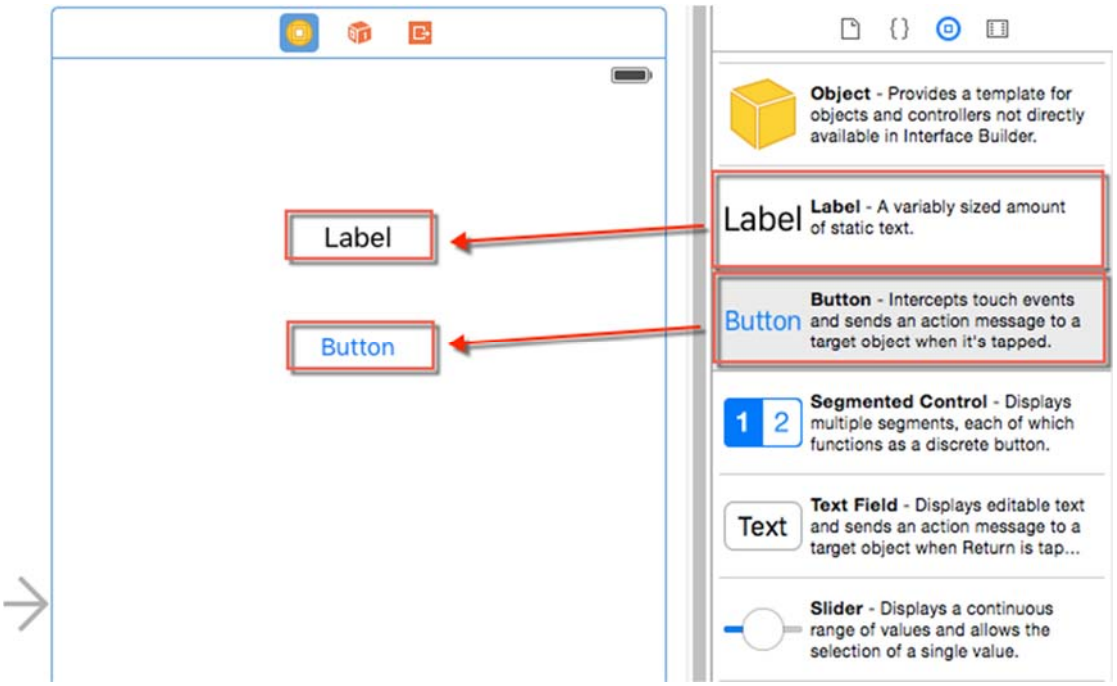
你也可以是高手！

在辛苦耕耘播種，必快樂地享受豐收的結果。透過在前面中幾節重覆性的練習後，現在身為讀者的乞應該是滿心歡喜的，迫不及待的想往下一節邁進。編編建議讀者們，別忘了有關於詳細的 swift 語法，還是可以多翻閱第三章相關說明唷。

休息一下，喝杯水吧！

5-8 解 答

1. 將【文字標籤】及【操作按鈕】建立在 IB 畫面中。



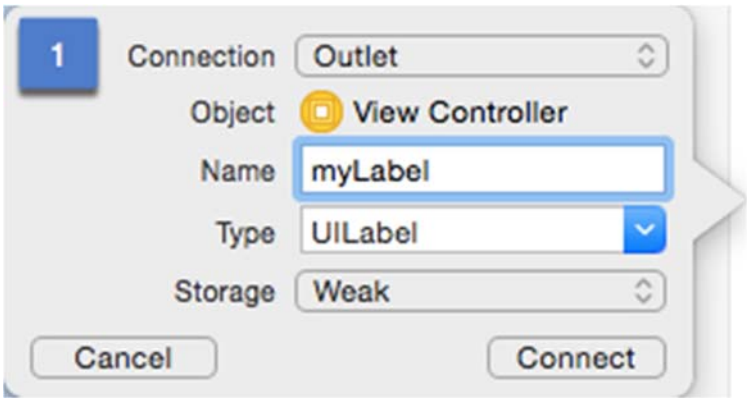
2. 在 IB 視窗上點選【Label】及【Button】(為以下畫面藍色方框)，接著按住【control】鍵，用滑鼠拖曳到右邊視窗與【程式碼】連結。

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

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

1   @IBOutlet weak var myLabel: UILabel!
2   @IBAction func tapBtn(sender: AnyObject) {
```

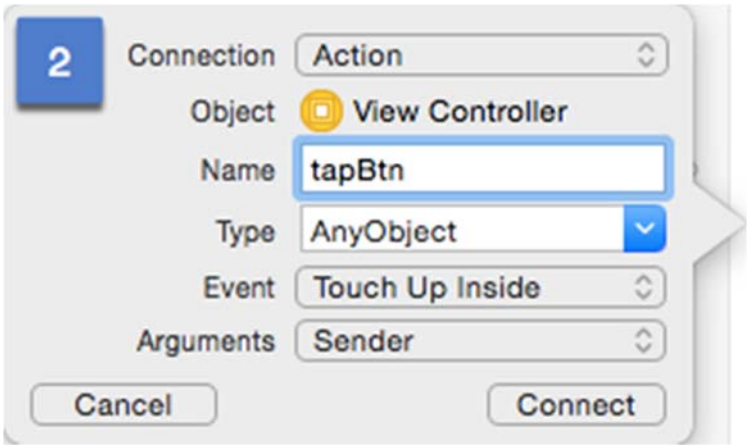
分解如下

3. 由【物件區】選取之各個元件，設定名稱如下：

拖曳【Label】在「Connection」欄位點選【Outlet】、「Name」欄位設定名稱【myLabel】以及在「Type」欄位點選【UILabel】後，按【Connect】按鈕。



拖曳【Button】於「Connection」欄位點選【Action】、「Name」欄位設定名稱【tapBtn】以及在「Type」欄位點選【AnyObject】後，按【Connect】按鈕。



4. 最後，我們在【紅框】中設定 swift 的程式碼在編輯區當中。

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

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

1   @IBOutlet weak var myLabel: UILabel!

2   @IBAction func tapBtn(sender: AnyObject) {

        let alertController = UIAlertController (title: "Alert Test", message: "OK
            Calcel Test", preferredStyle: .Alert)

        alertController.addAction(UIAlertAction(title: "YES", style: .Default,
            handler: {action in self.myLabel.text = "YES!" }   ))

1       alertController.addAction(UIAlertAction(title: "NO", style: .Default,
            handler: { action in self.myLabel.text = "NO!" }  ))

        alertController.addAction(UIAlertAction(title: "Cancel", style: .Default,
         handler: { action in self.myLabel.text = "Cancel!" }  ))

        presentViewController(alertController, animated: true, completion: nil)

    }
}
```

5. 執行畫面，出現經由【操作按鈕】「Button」後，出現「YES」、「NO」以及「Cancel」三者選項，讀者們可以試試當選取「OK」時，在我們撰寫的【程式碼】在【文字標籤】「Label」會出現什麼樣的變化結果。