Socket.IO Programming using Swift for iOS/OSX Environment (1)

Introduction to Swift and iOS GUI

一. 實驗目的

了解Xcode使用方式，Swift的基本語法以及iOS的基本GUI使用方式

二. Swift 3基本語法介紹

1. 名詞對照
   - 變數：variable
   - 函數：function
   - 型別：type
   - 類別：class
   - 類別的變數：property
   - 類別的函數：method
2. 變數宣告

```
var i: Int = 0
```

   - 完整的變數宣告，包含
     變數名稱：i
     變數型別：Int
     變數初始值：0

```
var i = 0
```

   - 省略變數型別，”i” 之變數型別與 ”=“ 後面之 ”0“ 相同

3. 常數宣告

```
let pi: Double = 3.14159
```

   - 完整的常數宣告，包含
     常數名稱：pi
     常數型別：Double
     常數值：3.14159

```
let pi = 3.14159
```

   - 省略變數型別，“ pi ” 之變數型別與 ” = “ 後面之 ” 3.14159 “ 相同

4. 函數宣告

```
func appendStringWithNumber(string: String, number: Int) -> String
{
    let newString = string + String(number)
    return newString
}
```

- 函數名稱：appendStringWithNumber
  函數參數數量：2
  函數參數名稱：string, number
  參數型別：String, Int
  函數回傳型別：String

```
func printHello(name:String)
{
    print("hello, "+name+"!")
}
```

- 無回傳函數，等價於：

```
func printHello(name:String) -> Void
{
    print("hello, "+name+"!")
}
```

5. 函數呼叫

```
printHello(name: "Jack")
```

- 單一參數呼叫方法

```
appendStringWithNumber(string: "No.", number: 1)
```

- 多參數呼叫方法，注意，第二個變數開始輸入方式為 ” 函數參數名稱: 輸入變數 ”
- 在此， appendStringWithNumber 之第二個參數的名稱為 number ，所以 ( ) 內第二塊要填 “ number: 1 “

6. Class 與 繼承

```
class ViewController: UIViewController
{

    var name: String = "Jack"

    override func viewDidLoad() {
        super.viewDidLoad()
        self.printName()
    }

    func printName()
    {
        print(self.name)
    }
}
```

- 宣告一個名叫 ViewController 的 Class ，其繼承 UIViewController 這個 Class
- ViewController 包含一個名為 name 的 property ，型態為 String

- override 表示此 method 取代 parent class 原有的 method
- super 表示parent class，通常在取代parent class函數時，保險起見會先call parent class 的同一個函數，以免遺漏 parent class 本來該做的動作
- 在 class 內呼叫自身的 method 或是 property 時，要加 self。 self.name 及 self.printName( ) 表示指定此

7. String

在Swift中，可用不同的方式將多個變數組成一個新的 String

- String Concatenating：利用 + 號連接 String

```
let newString = "i = "+String(i)+", pi = "+String(pi)
```

- String Interpolation(建議)：利用\( )嵌入其他型別物件

```
let newString = "i = \(i), pi = \(pi)"
```

- Formatted String：與c語言類似的formatted string用法

```
let newString = String(format: "i = %d, pi = %f", i, pi)
```

8. 命名規則

- 專案名稱大寫開頭
- 變數/函數名稱小寫開頭，單字首大寫
  - myString
  - myString.containsString("s")
- 變數類別(Class Name) 開頭大寫
- 變數名稱盡量完整表達參數用途、型別

9. Interface Builder attributes

@IBOutlet：

Interface Builder內的物件可連結至這個變數

```
@IBOutlet var text : UITextField!
```

- Interface Builder 內可有 UITextField 連結到 text 這個變數

@IBAction：

Interface Builder 中的動作可連結至這個參數

```
@IBAction func sendClicked(sender : UIButton)
```

- Interface Builder 中 UIButton 類別的物件可將其動作連結至 sendClick 這個函數
- 發動這個動作的 UIButton 本身會被當作參數傳入 sendClicked 函數當作sender

三. 實驗步驟

1. 下載Xcode

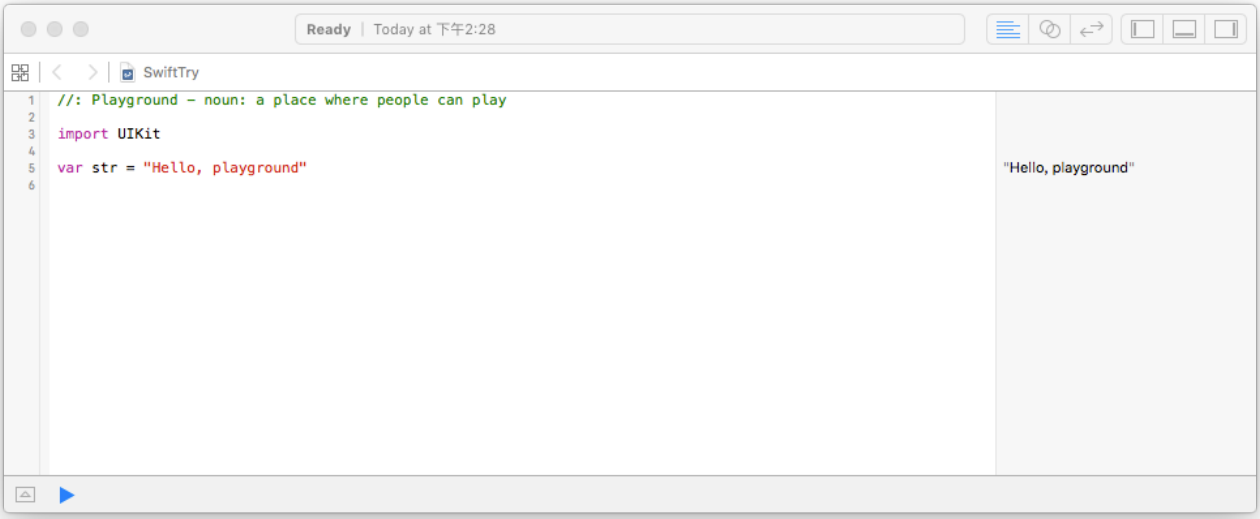
(1) 開啟 App Store 程式，找到Xcode程式下載

2. 利用Playground熟悉Xcode使用

(1) 開啟Xcode，左上角menubar，選擇 File -> New -> Playground

(2) Name 填 SwiftTry ， Platform 選擇 iOS，選擇 Next，選取桌面儲存

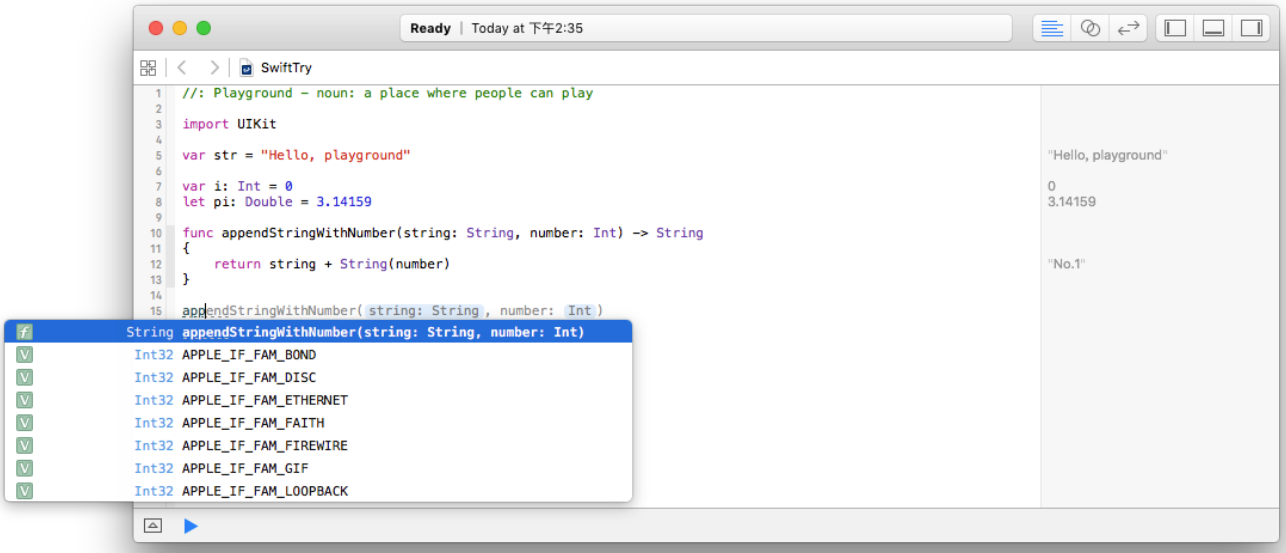
(3) Playground畫面左邊會即時顯示每行Code的回傳值



(4) 嘗試**自己鍵入**前述介紹內的程式碼，熟悉 Xcode 的 code editor 用法：

```
var i: Int = 0
let pi: Double = 3.14159
let string1 = "i = "+String(i)+", pi = "+String(pi)
let string2 = "i = \(i), pi = \(pi)"
let string3 = String(format: "i = %d, pi = %f", i, pi)

func appendStringWithNumber(string: String, number: Int) -> String
{
    return string + String(number)
}

appendStringWithNumber("No.", number: 1)


func printHello(name:String)
{
    print("hello, "+name+"!")
}

printHello("Jack")


class ViewController: UIViewController
{

    var name: String = "Jack"

    override func viewDidLoad() {
        super.viewDidLoad()
        self.printName()
    }

    func printName()
    {
        print(self.name)
    }
}
```

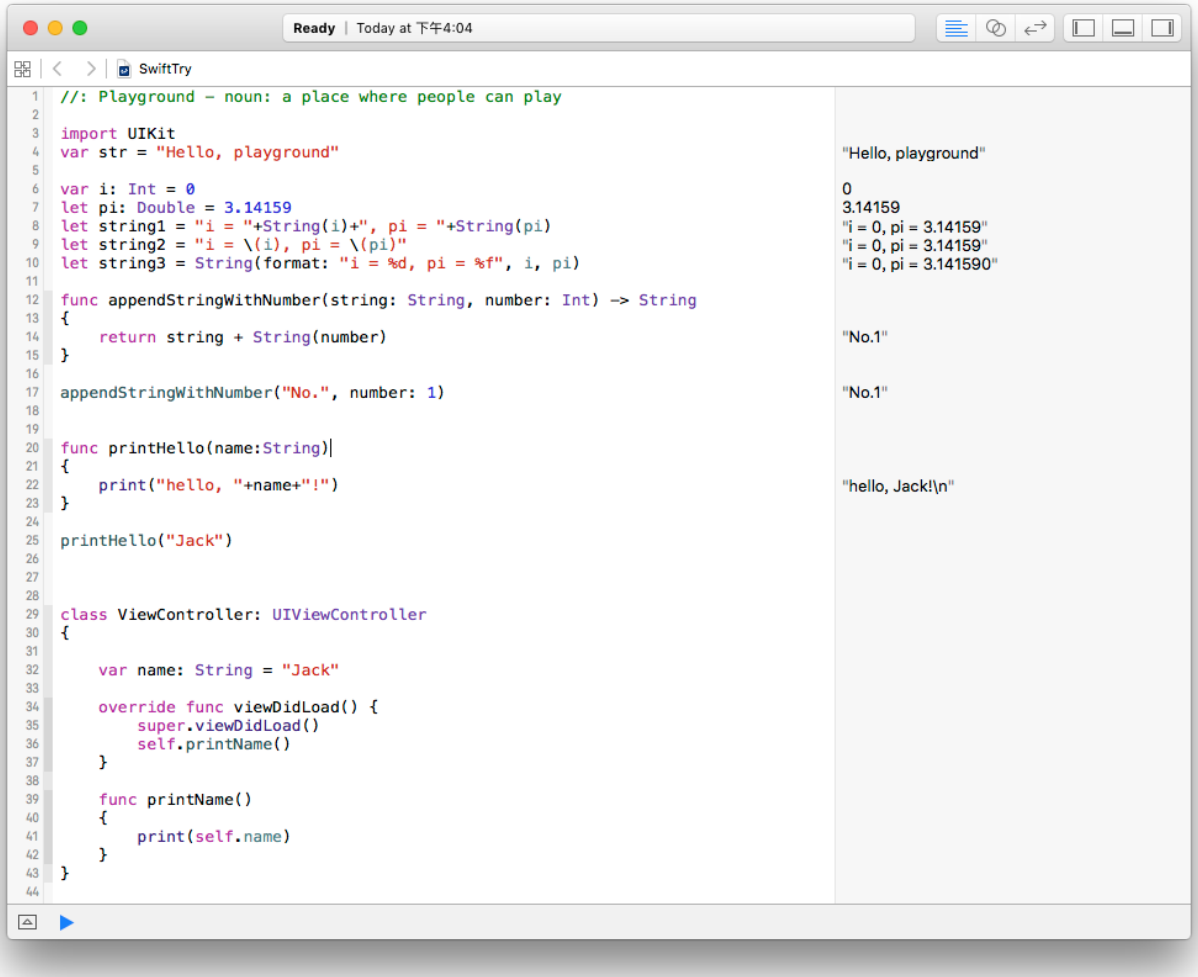
- Tips: 利用 ↑ 、 ↓ 、 enter 鍵選取自動完成



- Tips: 利用tab鍵跳轉參數填入區

```
appendStringWithNumber( string: String , number: Int )
```

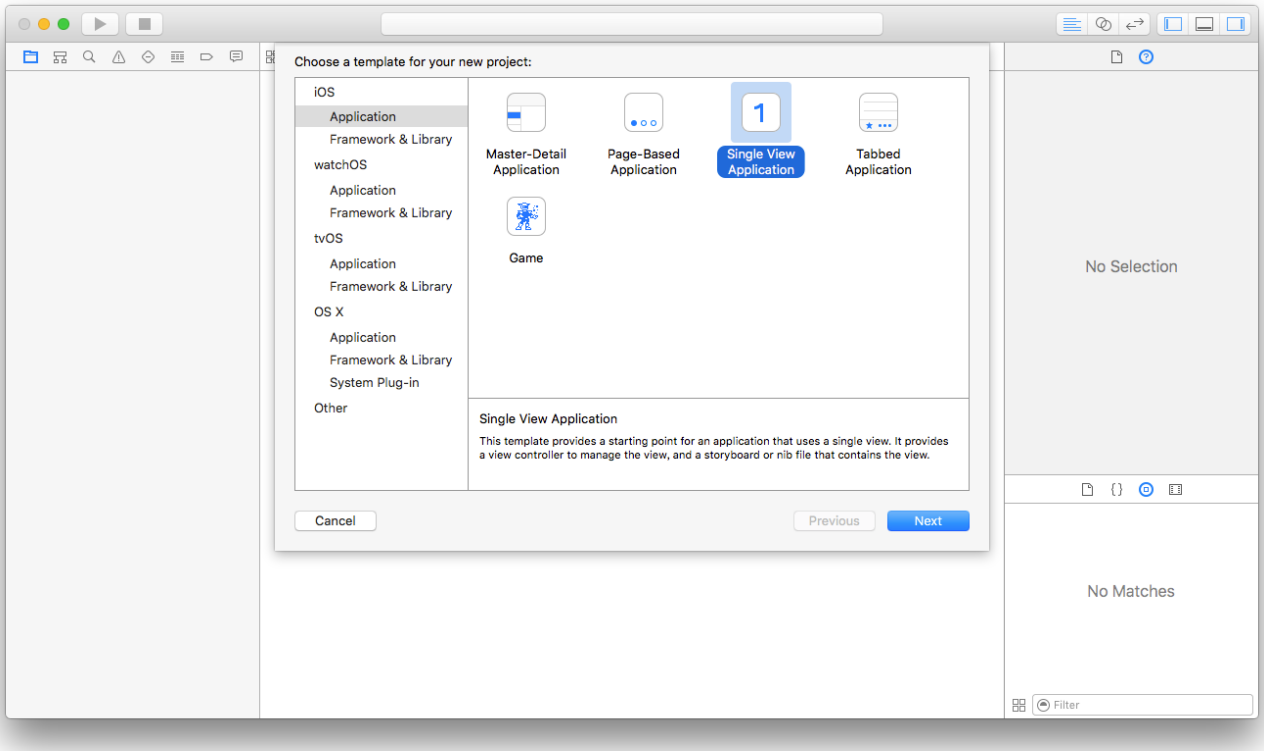
(5) Playground 結果如下



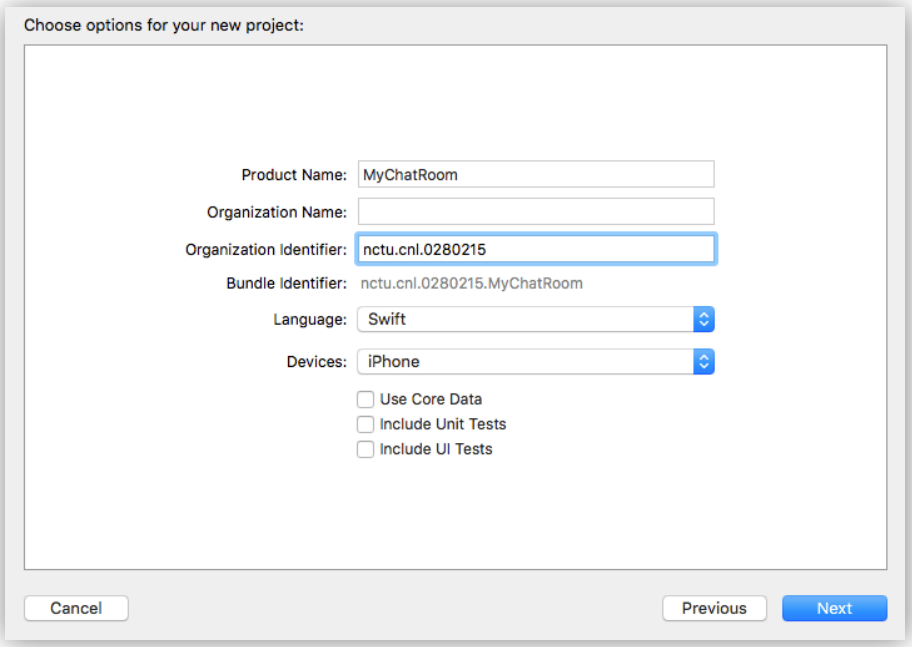
3. 建立 iOS 專案 - MyChatRoom

(1) 開啟Xcode，左上角menubar，選擇 File -> New -> Project

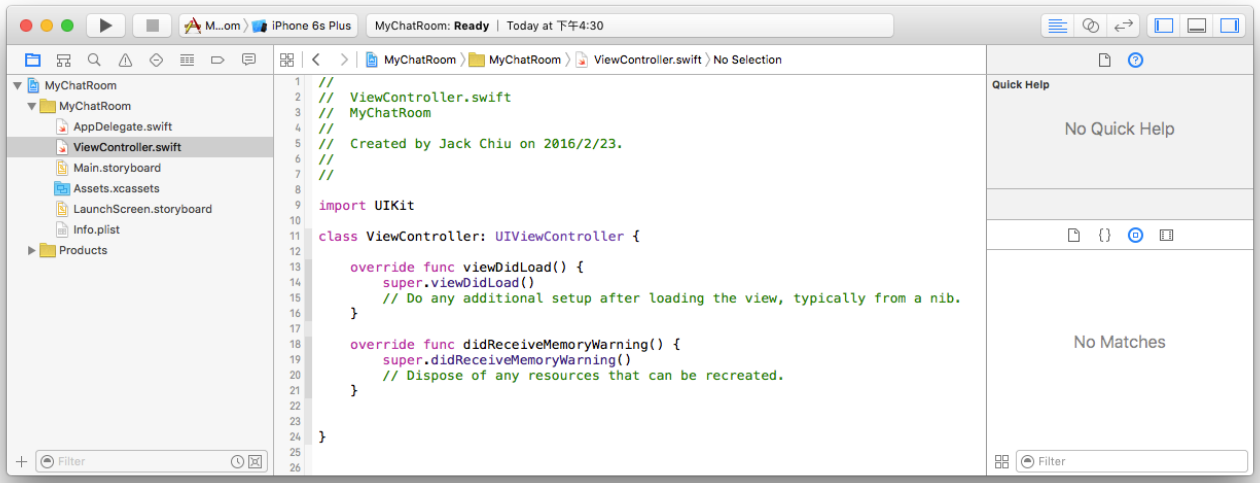
(2) 左方選擇 iOS 內的 Application ， 右方選擇 Single View Application，按 Next



(3) Product Name 輸入 MyChatRoom
Organization identifier 輸入 nctu.cnl.”你的學號”
Language 選擇 Swift
取消勾選 Use Core Data, include Unit Tests, include UI Test
按下 Next，存在桌面



(4) 點選左方 Project navigator 中的 ViewController.swift，可看到以下程式碼

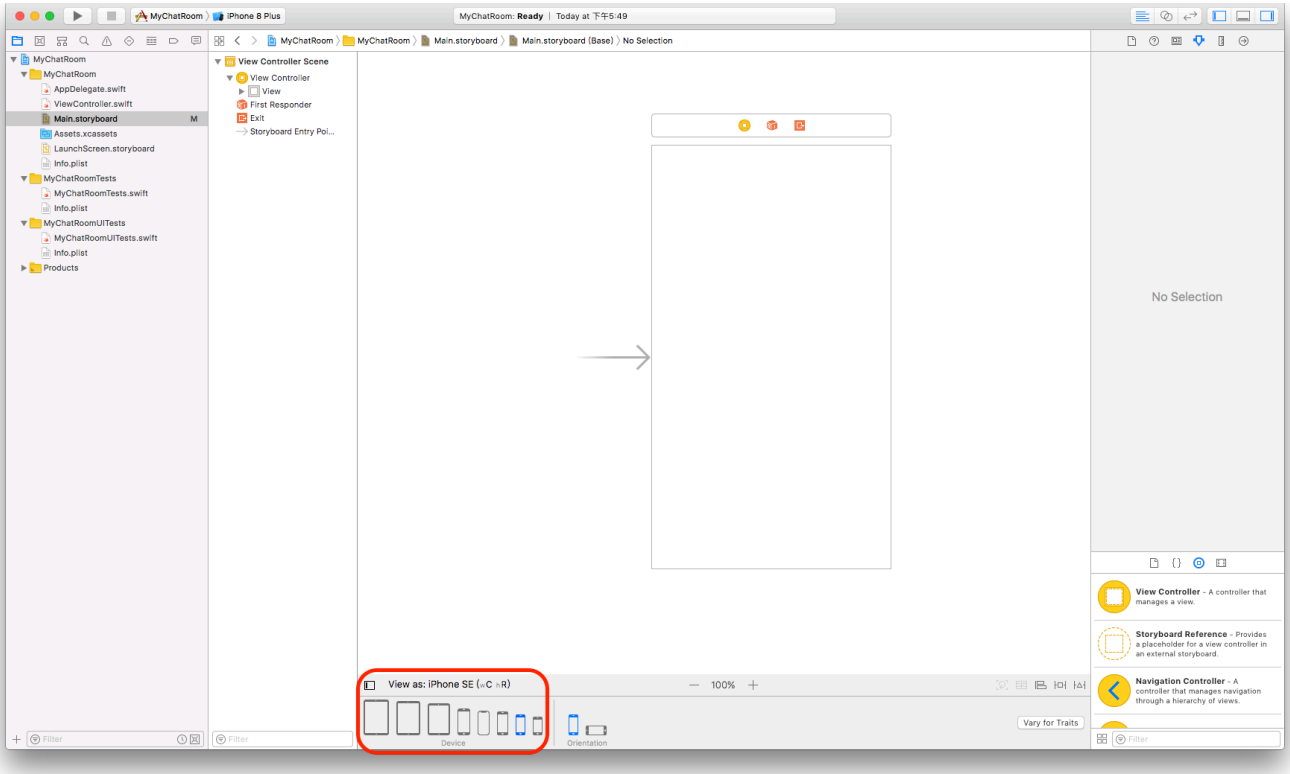


一個 View Controller 可以控制App的一個頁面。我們先在 Storyboard 中先拉好這頁使用者會用到的 GUI 物件，接下來在 View Controller 中用程式碼控制這些物件的作用、互動。

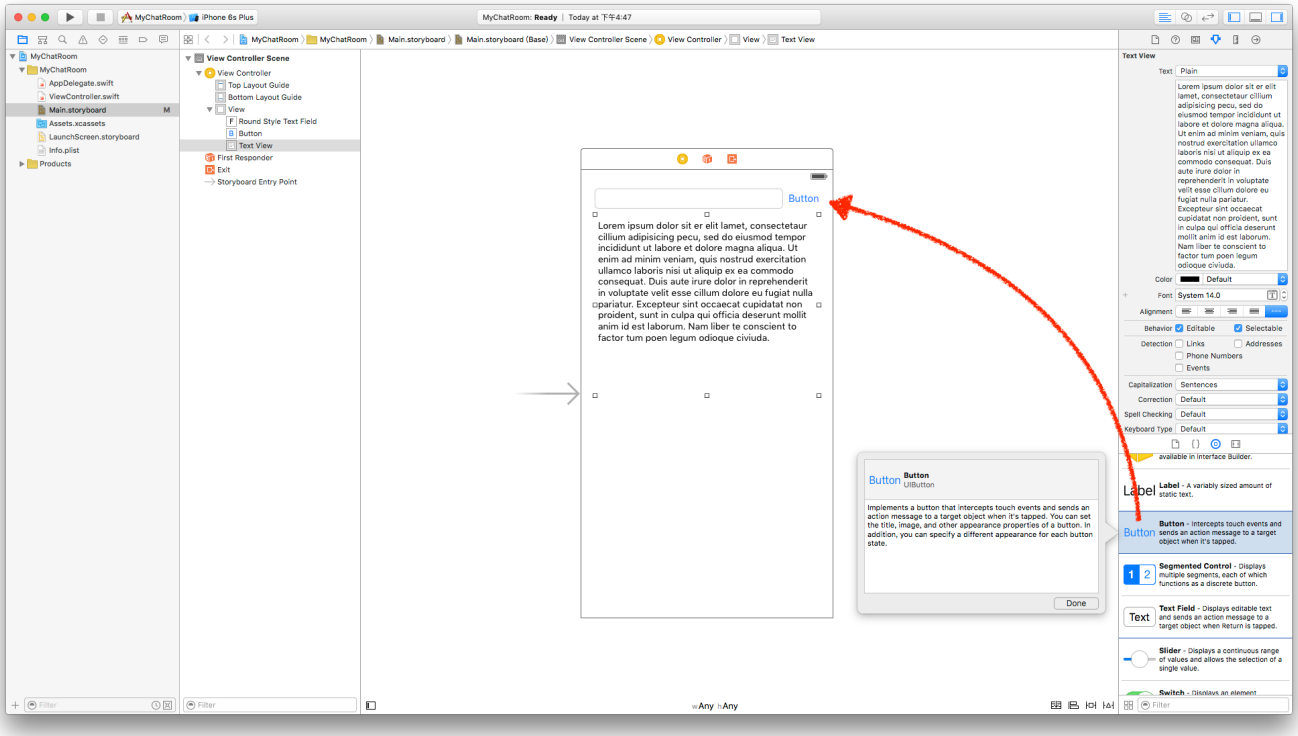
4. Xcode Interface Builder 與程式 GUI Layout

(1) 點選專案左方 Project navigator 中的 Main.storyboard 開始編輯程式使用者界面

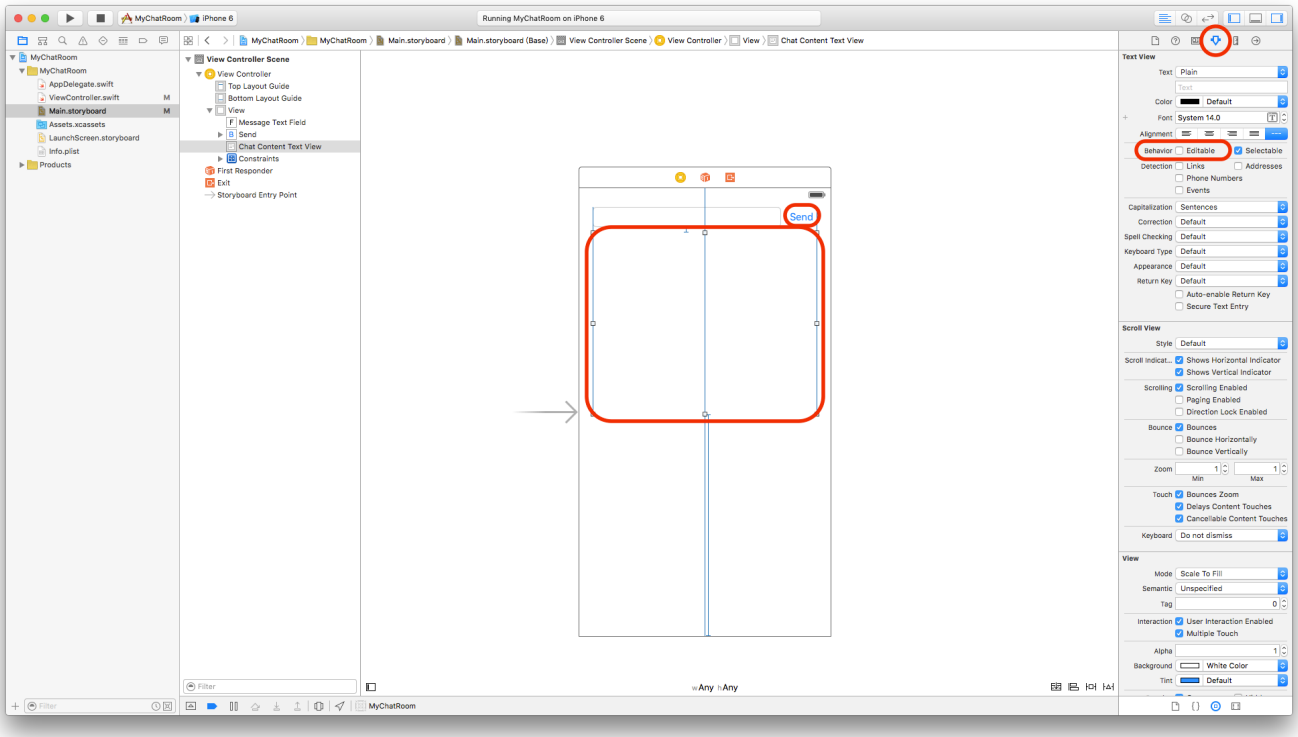
(2) 如圖，選取ViewController對應到的介面，並在下方將其介面大小改成適當大小，方便 layout



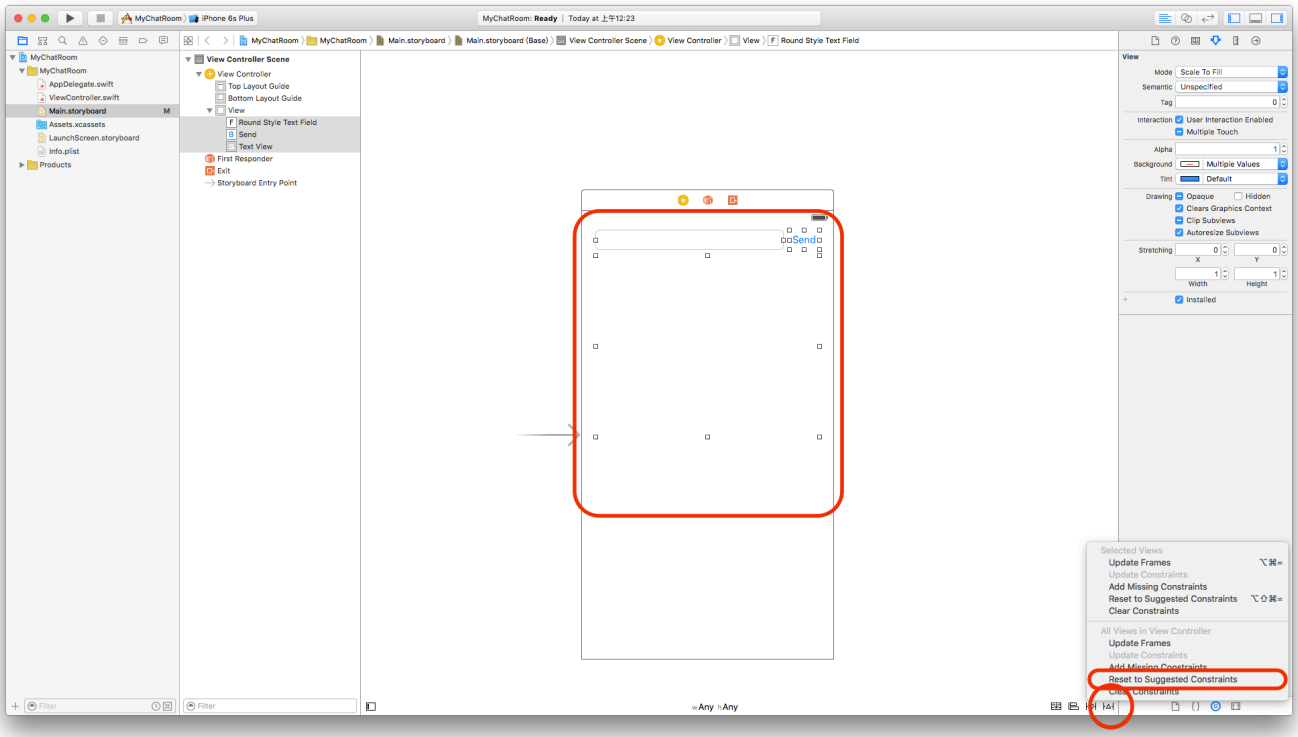
(3) 從右下方Show Object Library 中找到 Text Field (UITextField) 、 Button (UIButton) 、 Text View (UITextView) 三種物件，拉入畫面中，排成想要的排列。



(4) 點選Button物件，利用右方 Attributes inspector 將按鈕名稱改成 Send ，同樣方式，將 Text View 的預設內容刪除，並將 Text View 的 Editable 取消選取。

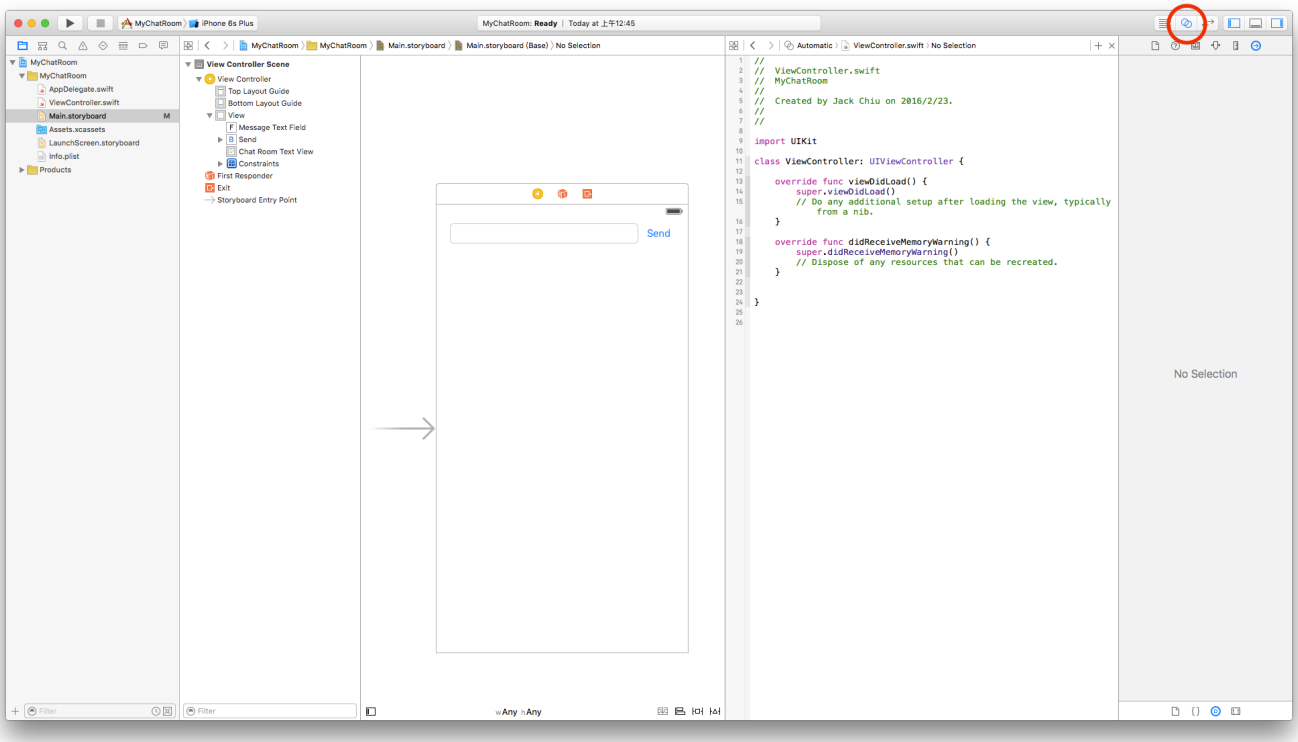


(5) **固定物件 Layout** 。 利用鍵盤 Command + A 選取全部物件，點選右下方 Resolve Auto Layout Issue -> Reset to Suggested Constraints



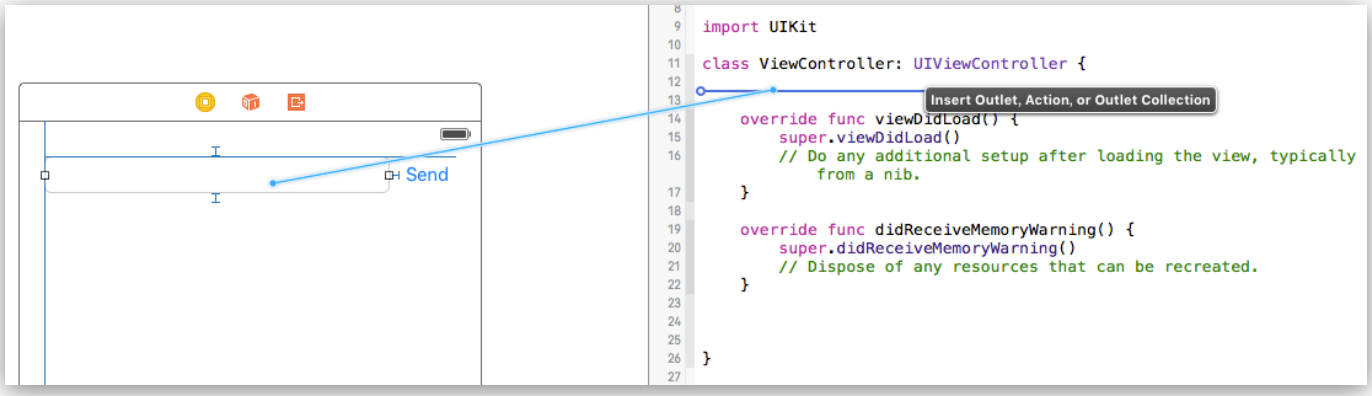
5. GUI 與 Code 的連結

(1) 點選右上方 Show the Assistant Editor ， Xcode 介面右側會顯示現在編輯的介面所對應到的View Controller Class 檔案。



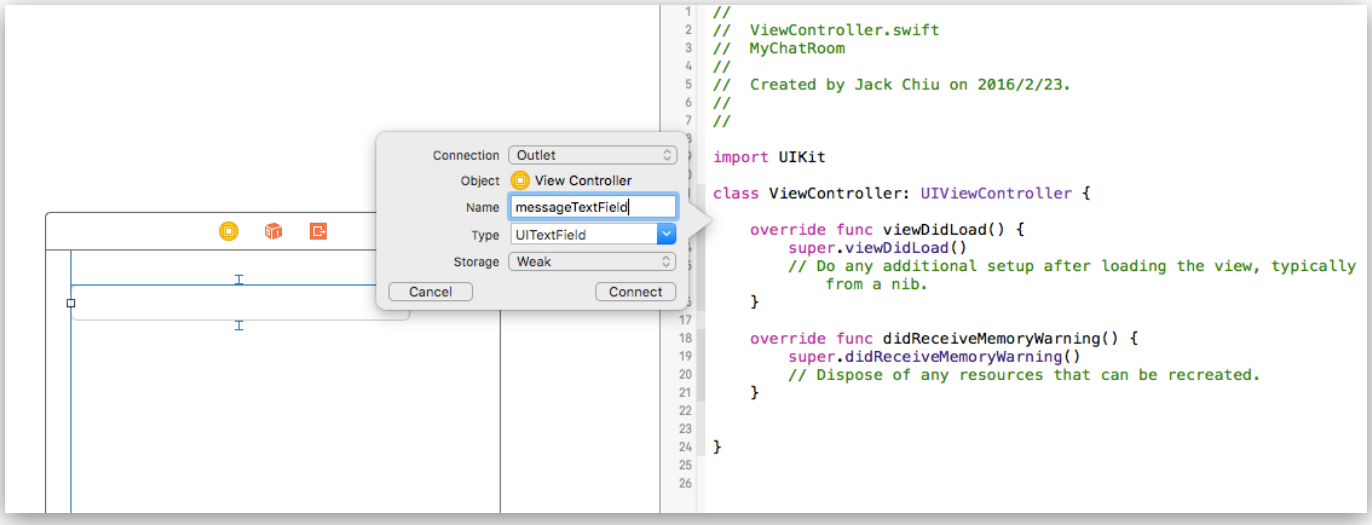
(2) **將 GUI 物件連結到 ViewController 的 Property (IBOutlet)**。在左方 Interface Builder 中選取 Text Field ，按住鍵盤 control 鍵，用滑鼠拖移致右方的程式碼中

“class ViewController: UIViewController {“ 與 “override func viewDidLoad() {“ 的中間。



(3) 在彈出的框框中如以下設定，按下 Connect

- Connection: Outlet
- Name: messageTextField
- Type: UITextField (預設值，確認是否為物件的Class)
- Storage: Weak (預設值)



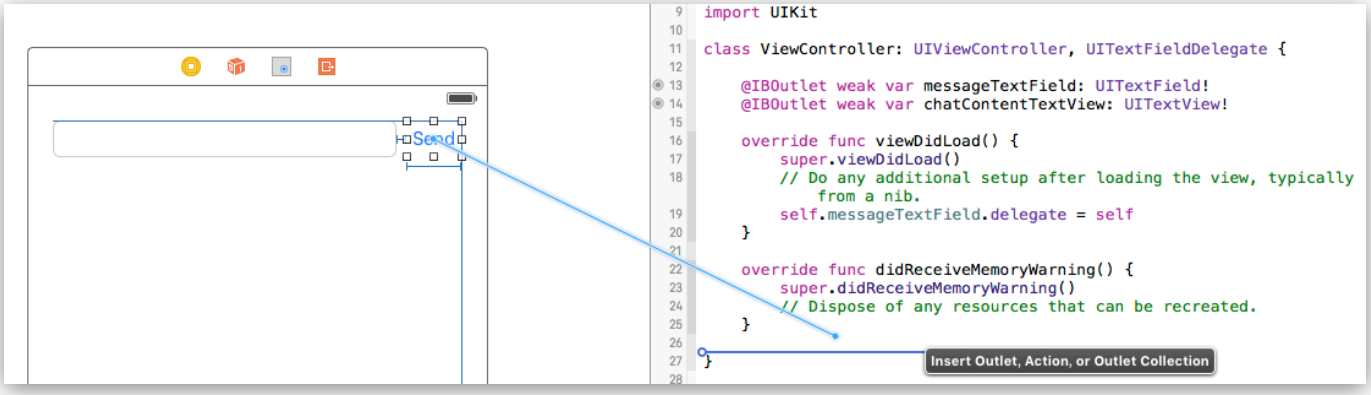
(4) 我們在介面上拉出的 Text Field 即會連結到 ViewController 中名為 messageTextField 的 Property 上。在 ViewController 中，self.messageTextField 就代表左方介面中剛剛連結的那個 Text Field 物件

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var messageTextField: UITextField!

    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.
    }

}
```

(5) 將 Interface Builder 中的 Text View 利用同樣方式連結到 ViewController 的 property ，名為 chatContentTextView

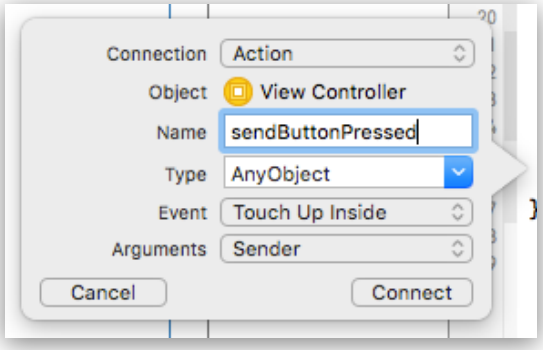
```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var messageTextField: UITextField!
    @IBOutlet weak var chatContentTextView: UITextView!

    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.
    }

}
```

(6) **將 GUI 物件的事件連結到 ViewController 的 method (IBAction)**。在左方 Interface Builder 中選取 “Send” Button，按住鍵盤 control ，用滑鼠拖移致右方的程式碼中最後的 ”}” 上方。



(7) 在彈出的框框中如以下設定，按下 Connect

- Connection: Action
- Name: sendButtonPressed
- Type: AnyObject (預設值)
- Event: Touch Up Inside (預設值，可改成你需要的Button動作)
- Arguments: Sender (預設值)



(8) 我們在介面的 “Send” button 若有 Touch Up Inside (點擊) 的事件發生時，會自動呼叫 ViewController 中名為 sendButtonPressed 的 method。 “Send” button 會將自己本身當作 method 參數傳入，所以在 sendButtonPressed function 中 sender 即代表程式介面上觸發事件(被點擊)的那個 “Send” button。

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var messageTextField: UITextField!
    @IBOutlet weak var chatContentTextView: UITextView!
    var i = 0

    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.
    }

    @IBAction func sendButtonPressed(_ sender: Any) {
    }

}
```

6. 在 View Controller 設計程式流程與處理介面的互動

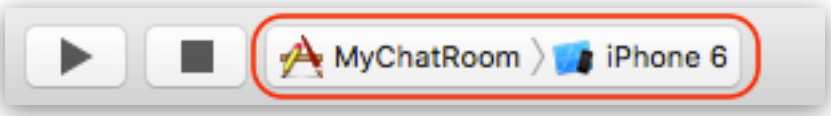
(1) 點選右上方 Show Standard Editor 返回單一檔案編輯畫面，在左邊 Project navigator 中選取 ViewController.swift 編輯

(2) 在 View Controller 中加入 i 這個 Int property，並在 sendButtonPressed中加入以下程式碼：

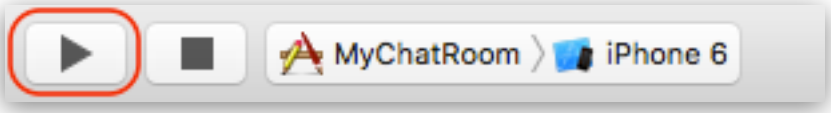
```
//取得欲輸入chat room 的文字
let message = self.messageTextField.text!

//留言計數+1
self.i = self.i + 1

//合成新的 chat room 內容
let newChatContent = "[\(self.i)] \(message)\n\(self.chatContentTextView.text!)"

//更新 chat room 內容
self.chatContentTextView.text = newChatContent

//清空輸入框
self.messageTextField.text = ""
```

7. 建置、執行與測試專案

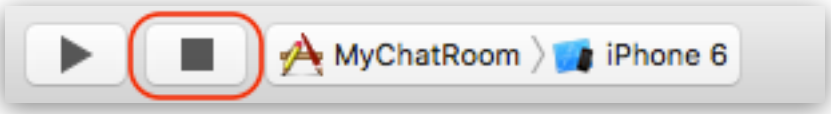
(1) 在左上方點選 Set the active scheme ， 選擇想要建置且執行的目標，在此選擇任一 iOS Simulator。



(2) 點選左上方 Build and then run the current scheme ，來建置且執行專案

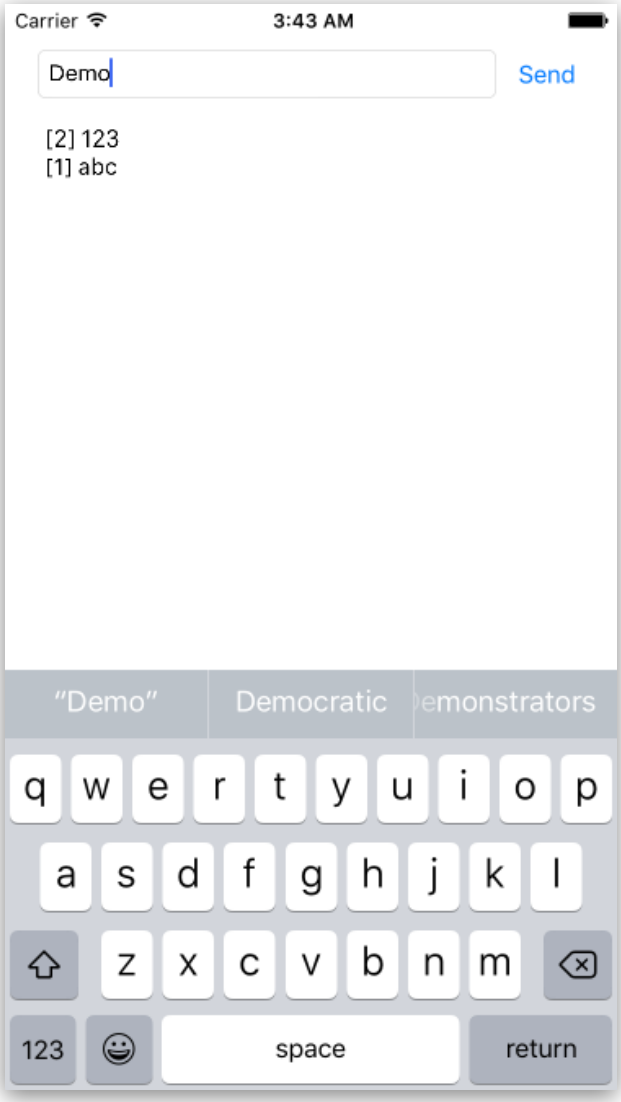


(3) 點選左上方 Stop the running scheme or application ，來停止執行中的程式



8. 執行結果

(1) 在 Text Field 輸入文字後，按下 Send 按鈕，會將輸入內容傳到下方的 Text View ，並且加上編號



四. Demo

請 Demo 實驗步驟最後的執行結果。