

嵌入式系統設計

Embedded System Design

Lab 5: Segue and Navigation Controller

蕭安紜助教製作

一. 實驗目的

了解如何利用 Segue 與 Navigation Controller 做出介面之間的切換，以及如何利用 segue 做不同頁面間資料的傳遞及同步。

二. 實驗需求

環境：macOS 13.4 或以上

IDE：Xcode 14.0 或以上

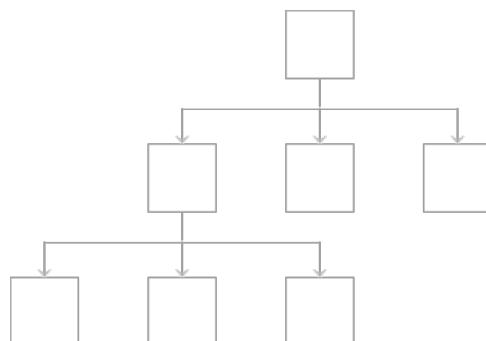
語言：Swift 5

三. Screen Navigation

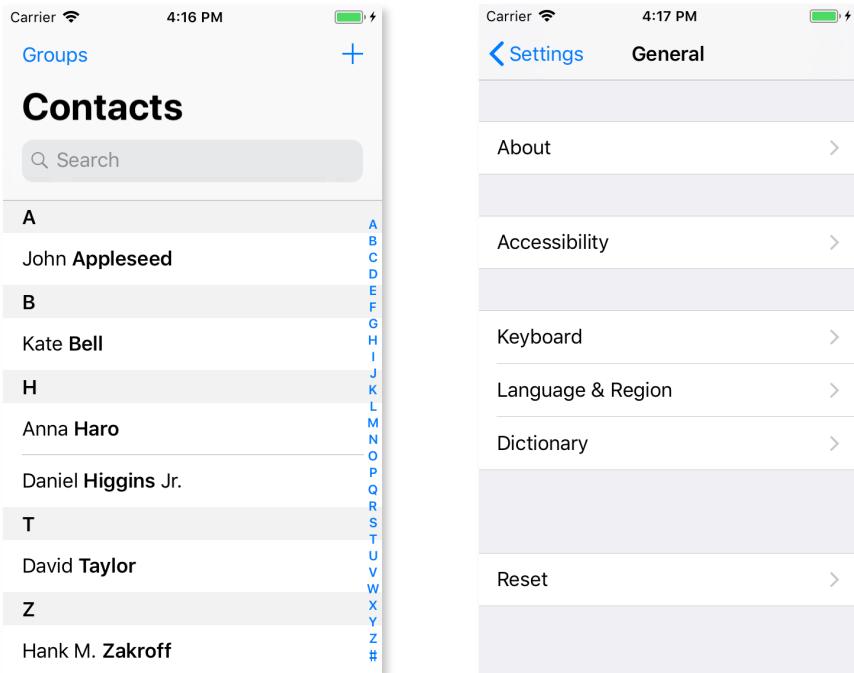
在不同的介面之間切換，稱為 navigation，navigation 與 app 的架構是連動的，app 透過螢幕間的切換讓使用者了解 app 的架構，如果頁面隨意切換沒有系統，會導致使用者迷失，不知道自己在哪個位置，或不知道要如何找到某個功能。建議使用內建標準的 navigation 套件來實作，因為這是使用者最熟悉的切換/互動方式，比較容易理解，也不容易混淆。常見的 navigation 分為三種：

1. Hierarchical Navigation

階層式的navigation架構，每一個介面只會有一個路徑可以到達，如果要去到不同的介面，需要回到上一層的介面，重新選擇下一個要去的介面。可使用內建的 **UINavigationController** 達成。

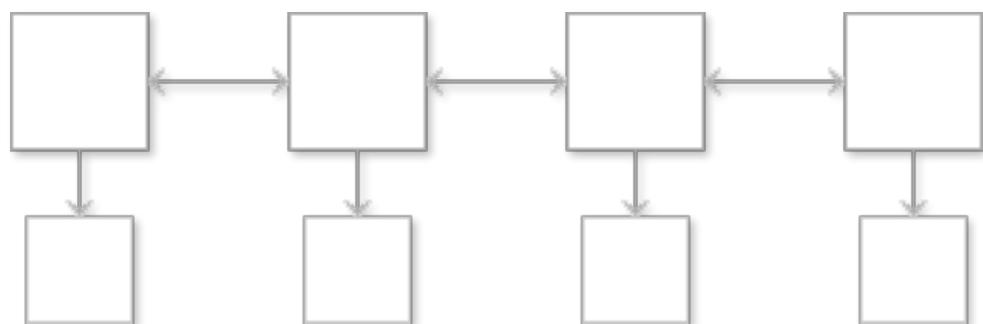


iOS 的「設定」與「通訊錄」 app 就是使用 hierarchical navigation。

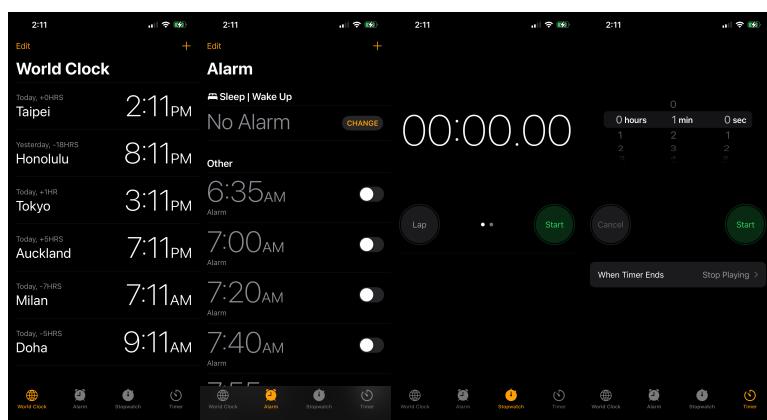


2. Flat Navigation

平行的navigation架構，直接在不同的介面切換，可使用內建的 **UITabBarController** 或是 **UIPageViewController** 達成。

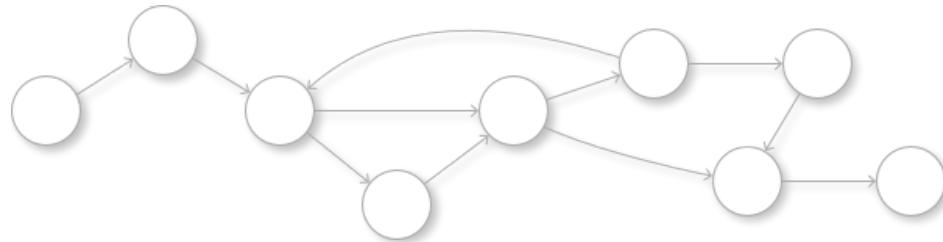


iOS 的「鬧鐘」與「照片」 app 就是使用 flat navigation。



3. Content-Driven or Experience-Driven Navigation

在不同功能的介面間隨意切換，通常是根據介面的內容來決定接下來可以 navigate 到哪一個介面。通常遊戲、書籍會使用這種方式。



三種 navigation 方式也可以混用，只要是符合邏輯，能讓使用者理解的方式都可以。需要注意不要讓 navigation 的方式影響到介面顯示的內容，或是讓使用者從內容分心。

延伸閱讀：[Navigation](#)、[Modality](#)

四. Segue

Segue 用來定義在 storyboard 檔案中不同 ViewController 間的切換。Segue 的起點 (Trigger Object) 可以是一個 button 、 table view cell 或是 gesture recognizer 。而 segue 的終點一定是一個 view controller 。當使用者與segue起點的物件互動時，segue就會自己觸發，進行頁面的切換。

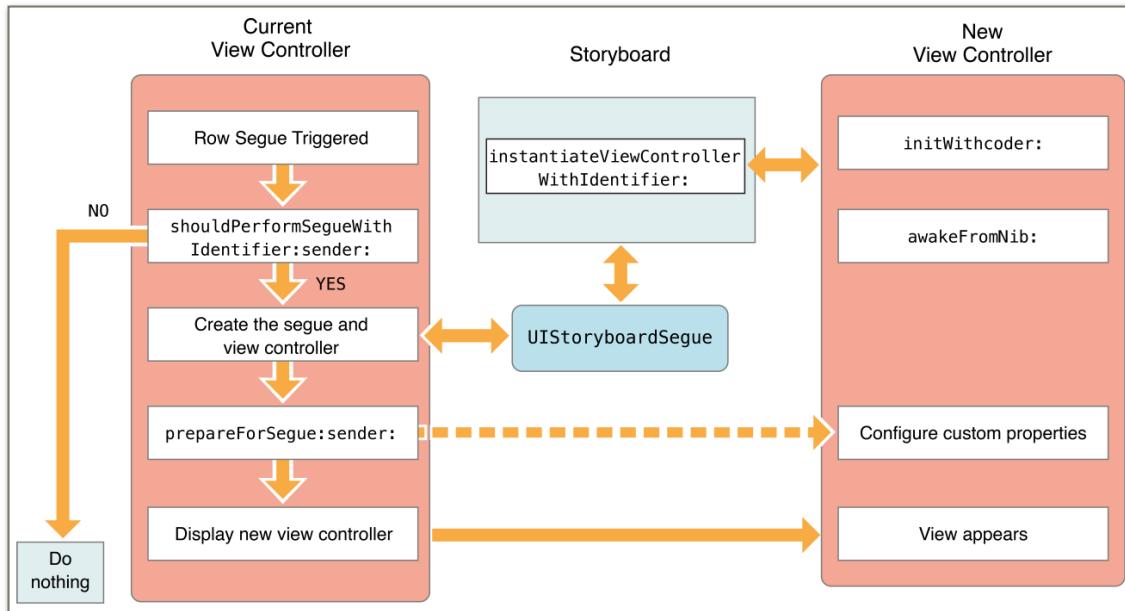


由於 segue 是連接兩個介面的橋樑，因此我們可以透過 segue 取得兩端的 view controller (segue.source , segue.destination)，進而同步兩個介面的資料。

* segue 的起始點也可以是 view controller ，不過這樣就無法透過使用者互動觸發，必須要用 code 觸發 (`performSegue(withIdentifier:sender:)`) 。

1. Segue

一般的 Segue 是定義從此介面切換至新介面，我們從 storyboard 上，run time 時，當使用者與 segue 的起點互動，segue 會自己觸發，產生新的 view controller 介面，並且切換至新的介面。iOS 會透過 call 發動 segue 的 view controller 的 method 來通知 segue 被觸發，讓 view controller 準備切換到下一個介面前的動作。segue 觸發到切換至新的介面的流程可見下圖。



由圖可見，當prepareForSegue: sender: 被 call 時，新的 view controller 已經產生了，我們可以利用 override 這個 method 來給新的介面一些初始值。

*注意：segue 一定會 initial 一個新的 view controller 出來，所以須小心不要讓 segue 的方向在 storyboard 裡形成一個迴圈，這樣會讓使用者一直產生新的介面出來，最後導致記憶體不夠，app 被強制關閉。

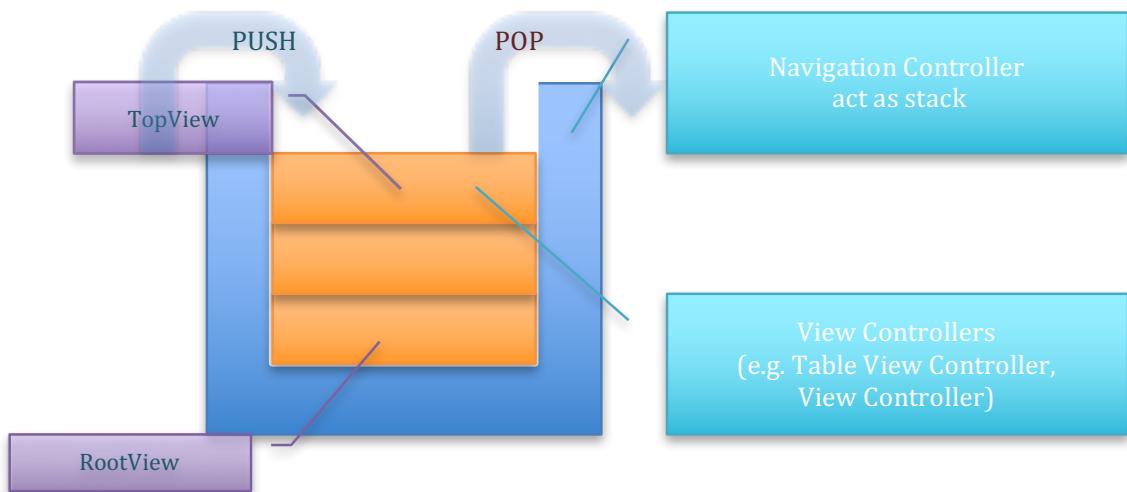
2. Unwind Segue

unwind segue 是用來離開(關閉)介面的 segue，unwind segue 的起始點一樣可以是 button 、 table view cell 或是 gesture recognizer。不過終點一定要是一個 view controller 內的 unwind action **method**。當 unwind segue 被觸發後，iOS 系統會自動尋找當下 **view controller hierarchy** (可以想成是被蓋在目前介面後面的所有其他的介面) 中，包含這個 method 的 view controller 當作 **destination view controller**，並且結束掉目前的介面以及從 destination view controller 到目前介面中間所有經過的介面。

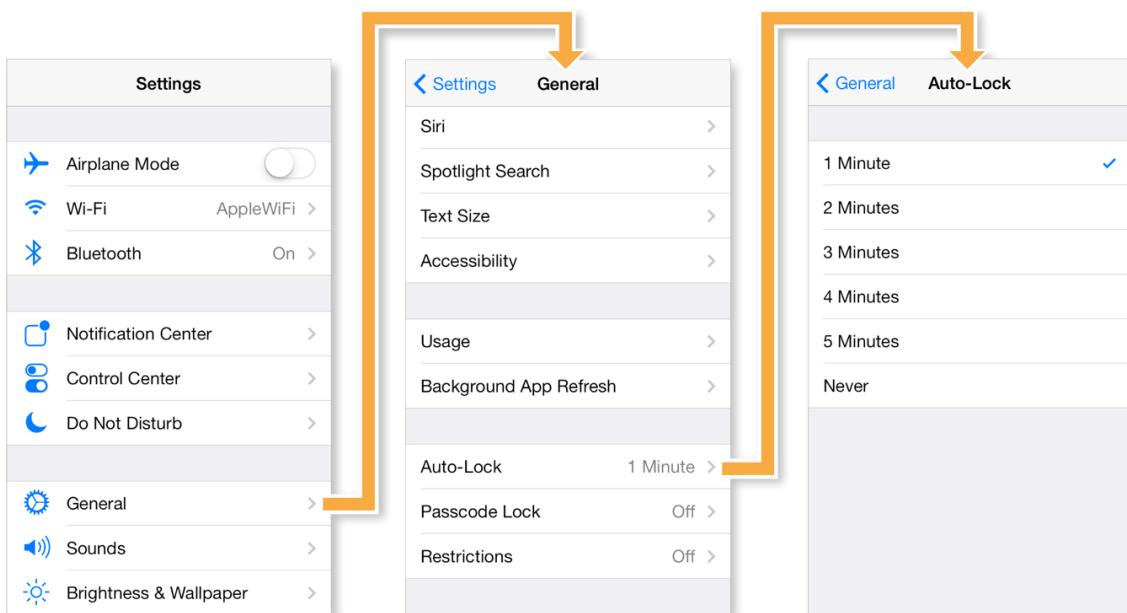
延伸閱讀：[Using Segues](#)

五. Navigation Controller

Navigation controller 可以讓 programmer 簡單的實作前述的 hierarchical navigation，navigation controller 使用 stack 的架構來管理一群 view controller。負責 view controller 的顯示以及不同 view controller 之間的切換。Stack 的最上方為 top view，也就是目前顯示的頁面，將 hierarchy 下一層的 view controller push 進 stack 即為顯示下一層的介面，要回到上一層的介面，則是將目前的介面 pop 出 stack。

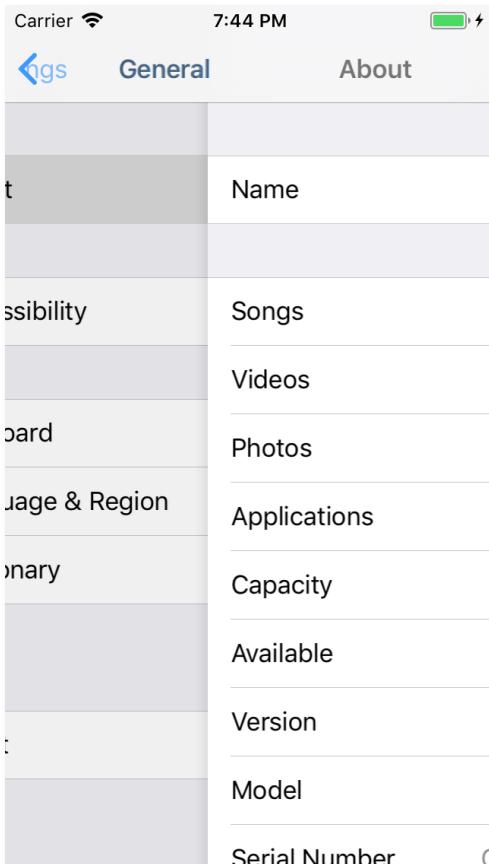


在顯示上，navigation controller 包含了一個 navigation bar，我們可以將整個介面包含 navigation bar 視為整個 navigation controller 管理的介面，而 navigation bar 下方的空間，則由 navigation controller 決定要顯示哪一個 view controller，最後 view controller 再決定自己的介面要顯示什麼內容。



Navigation controller 會根據目前顯示的 view controller 更新上方的 navigation bar，包含自動加入回到上一層的 back button，或是顯示 view controller 的標題，又或者是自動加入 view controller 定義要顯示在 bar 上面的按鈕。

我們也可從 push 或 pop 的 transition 動畫中，觀察出 hierarchical navigation 的 app 架構：



若在 storyboard 中使用 navigation controller 搭配 segue，新的 view controller 會自動 push 到 navigation controller 的 stack 中。

延伸閱讀：[UINavigationController](#)

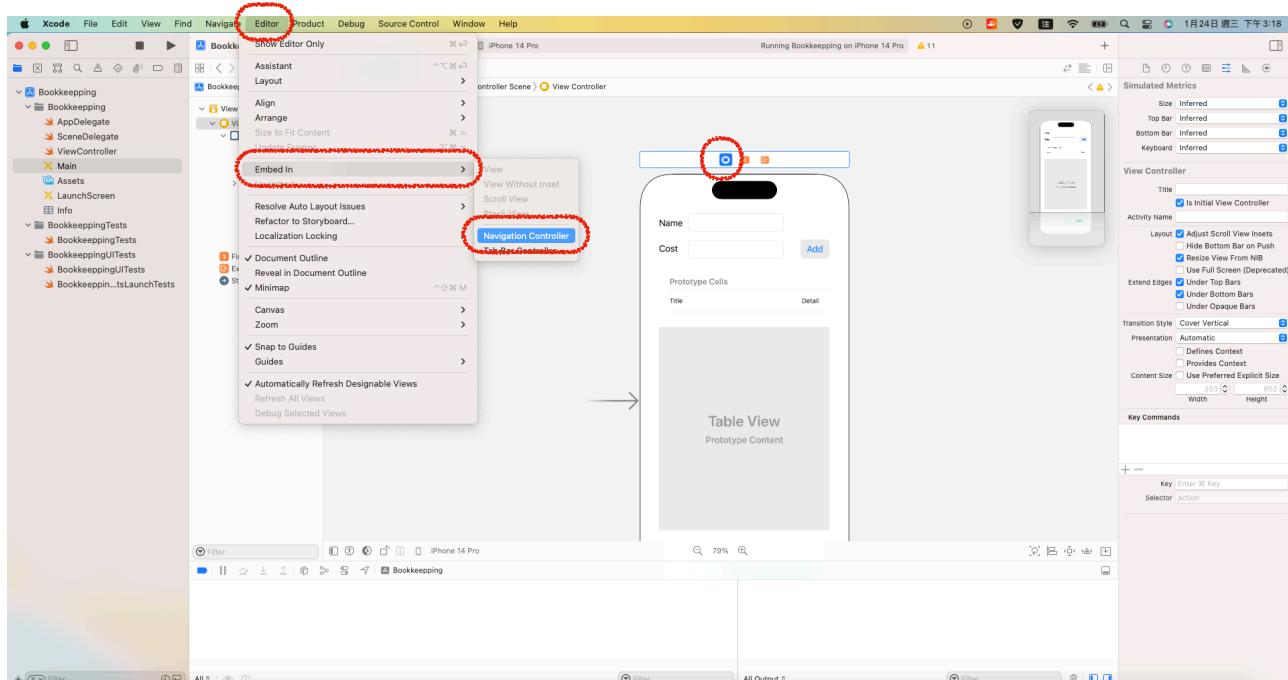
六. 實驗步驟

1. 準備基礎 Bookkeeping 專案

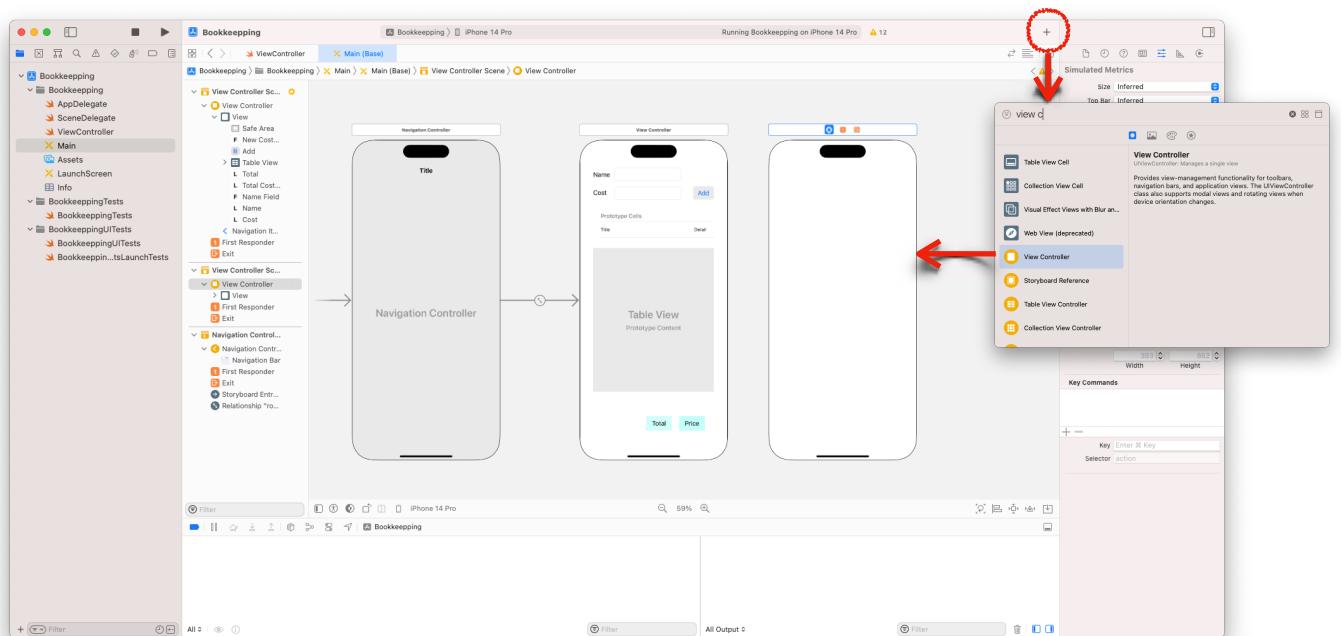
(1) 請依照 Lab 4 準備好基本的 Bookkeeping 專案

2. 新增 Bookkeeping App 第二個介面

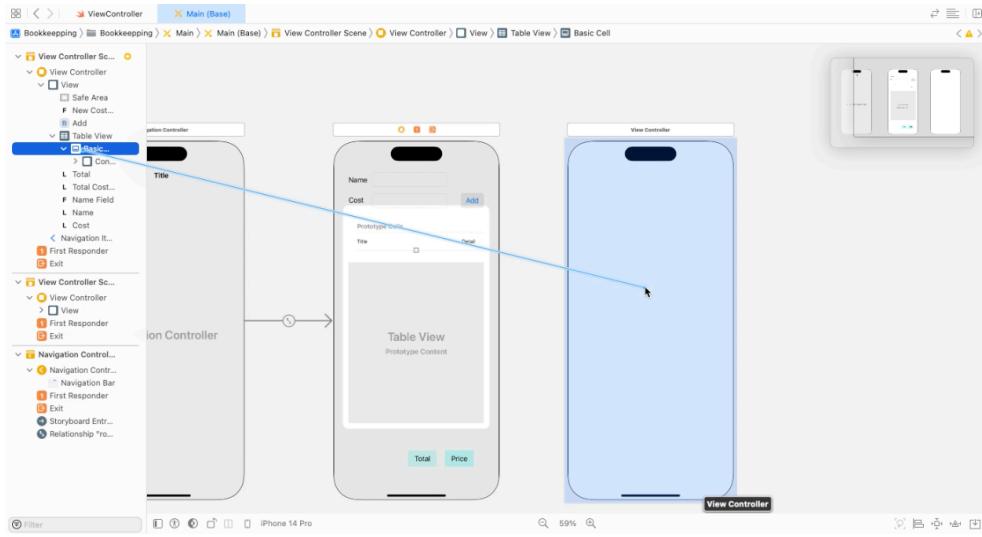
(1) 點選 Storyboard 上的 ViewController頁面，在上方 menu bar 選取 Editor -> Embed In -> Navigation Controller



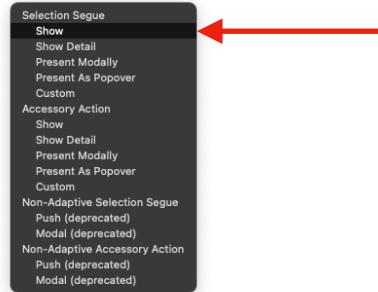
(2) 從 Object Library 拉入新的 View Controller 介面



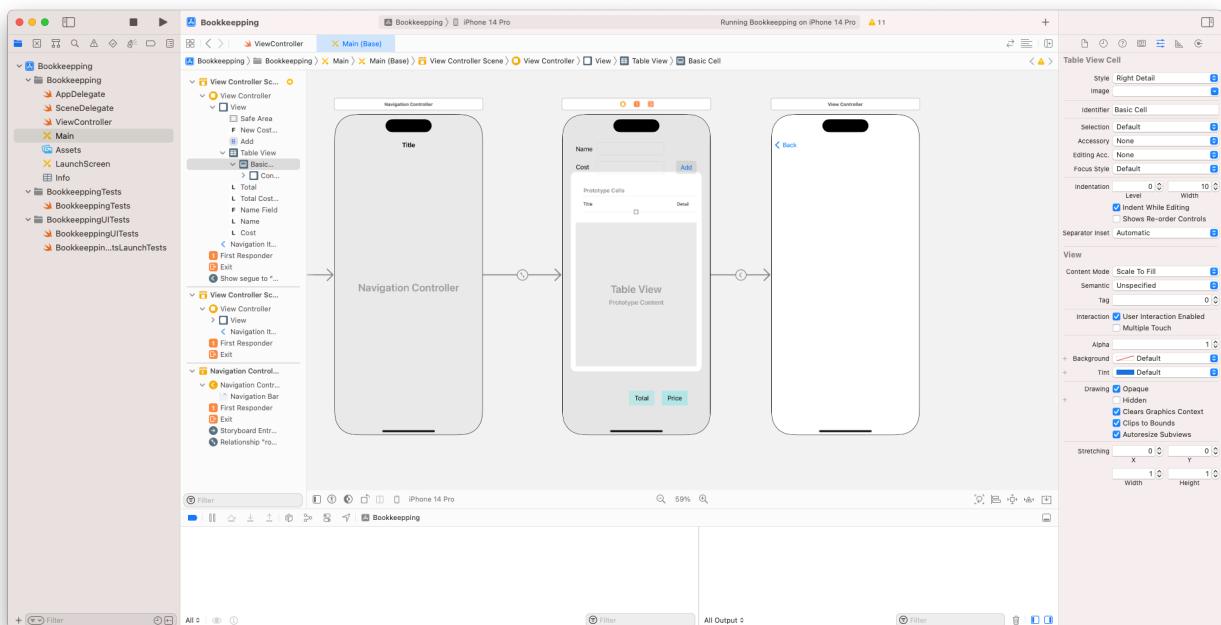
(3) 選取左邊你所使用的 table view cell，右鍵拖曳到新的介面上



(4) 選擇 show



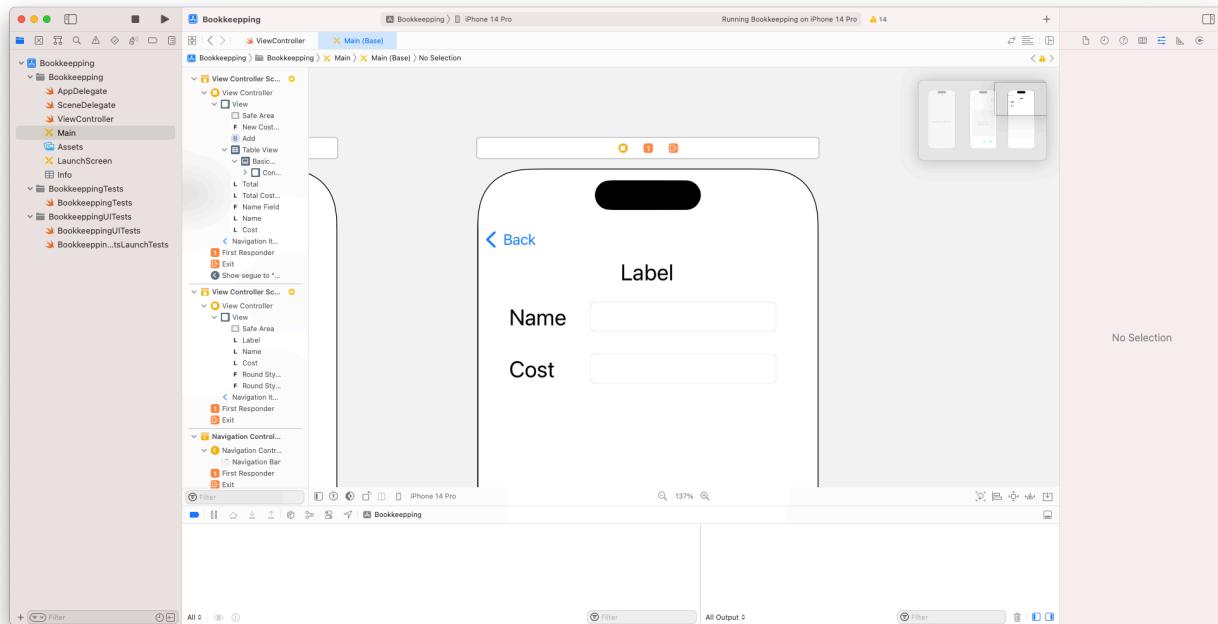
(5) 可看到 storyboard 上已經建立好兩個介面間的 segue 了



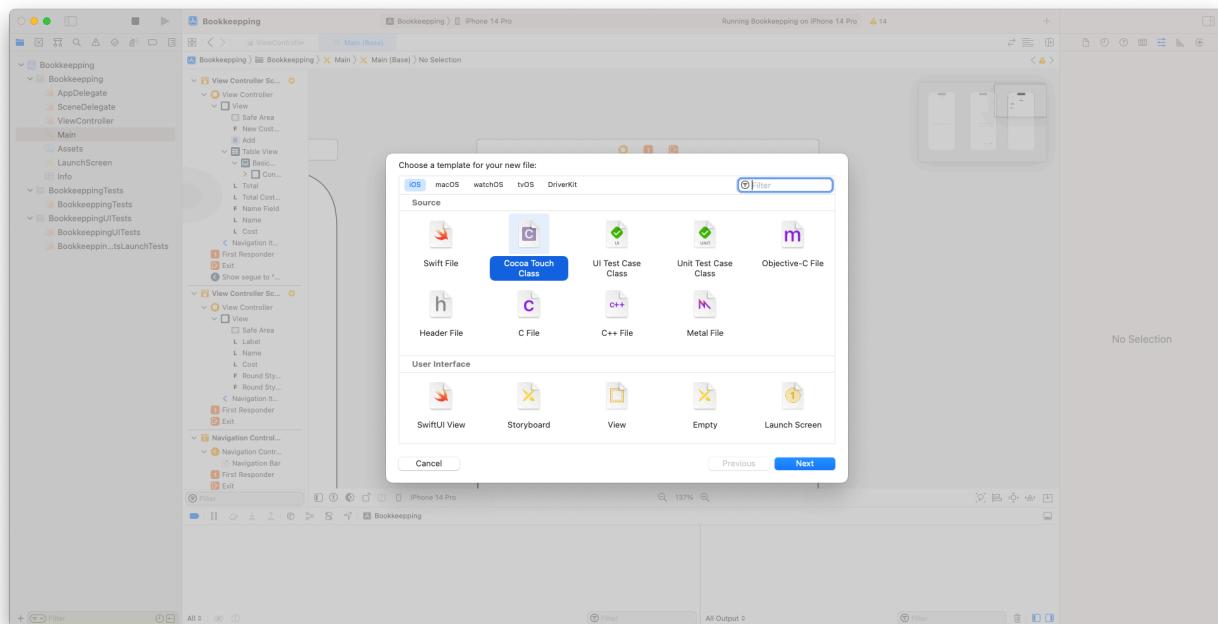
(6) 執行程式，點選 cell 後，可順利切換到下一页，並且按 back 可回到上一页

3. 在第二頁個介面上顯示詳細的消費資料

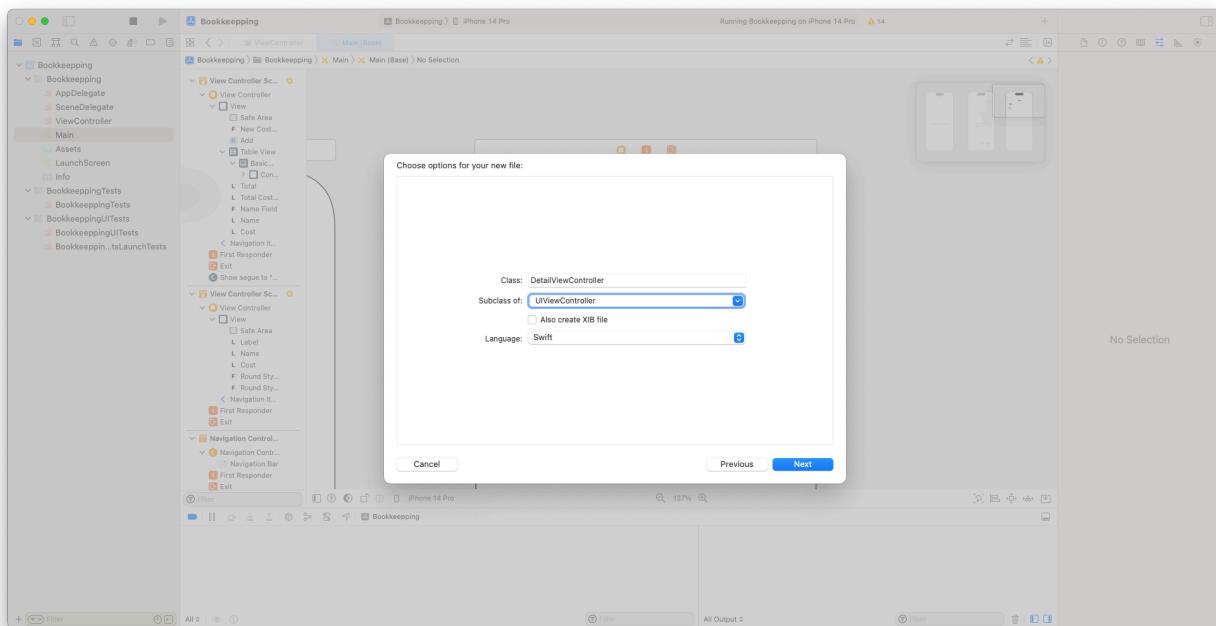
- (1) 在第二個介面上，從 object library 中拉入三個 label，兩個 text field 用來顯示 date、name、cost 三種資料，name 與 cost 是可以修改的 text field



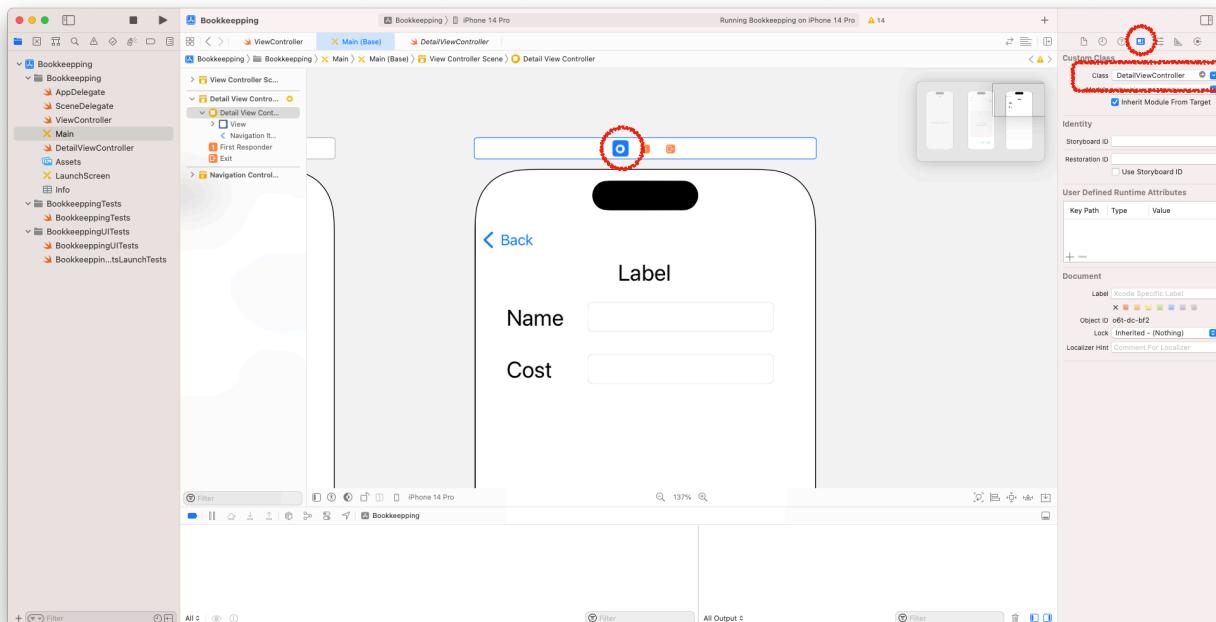
- (2) 為了可以用程式碼來控制新的介面顯示資料，我們需要一個新的 view controller class 來控制第二個介面，在上方 menu bar 上點選 File -> New File，選擇 Cocoa Touch Class，按 Next



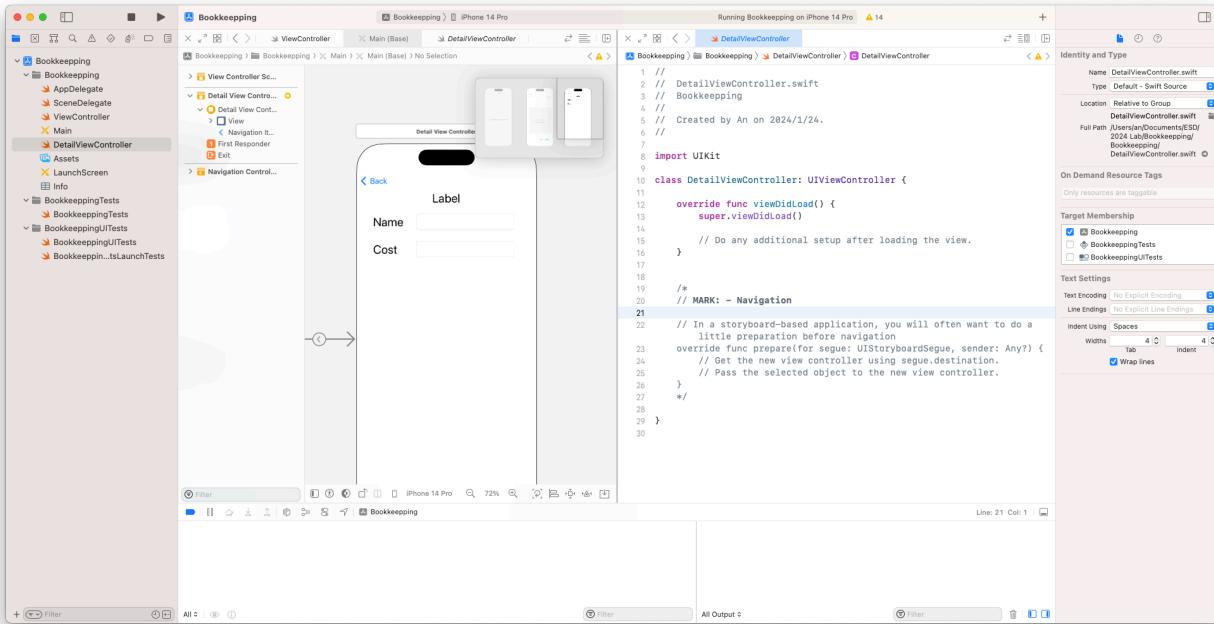
(3) 選擇 subclass UIViewController，將新的 class 命名為 DetailViewController，按下 Next，將新的檔案存在與其他 .swift 檔案同一個資料夾



(4) 將介面與 class 做連結，選取第二個介面的 view controller 介面，在右方 identity inspector 中把 Class 改成我們剛剛新增的 DetailViewController



(5) 點選右上方的 Assistant editor，當我們在左側畫面點選第二個介面的 view controller 時，可在右方的 editor 看到對應的 DetailViewController class 的檔案（如果沒有自動開啟檔案，請在如圖所示的 editor 上方選取 Automatic）



(6) 將 code 會用到的 UI 右鍵拖曳到 class 內，建立 IBOutlet

```

@IBOutlet weak var dateLabel: UILabel!
@IBOutlet weak var nameField: UITextField!
@IBOutlet weak var costField: UITextField!

```

(7) 新增一個**class property** 變數 **data**，用來存這個介面要顯示的那筆資料，型態與我們之前用來存資料的 dictionary 一樣為 [String:Any]。由於要顯示的 data 是由第一個介面決定的，所以在 class initial 的時候會是空的，但是在使用時一定會有值(不然無法顯示資料)，在此我們可使用 **Implicitly Unwrapped Optional** 來宣告它 (可參考 Lab 1)

```

var data: [String:Any]!

```

由於，第二個介面要顯示的資料是由第一個頁面中使用者選擇的資料決定的，因此 **data** 這個變數需要由第一個介面設定，我們可透過 **segue** 來搭起兩個介面的橋樑，傳遞資料

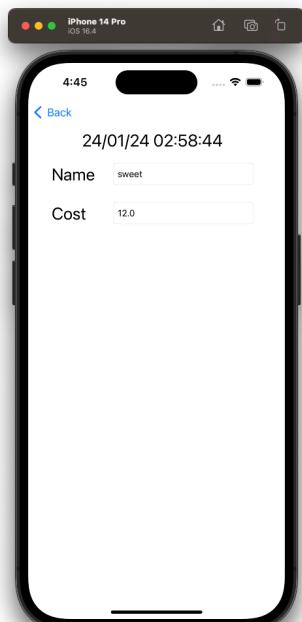
- (8) 將要顯示的 data 從第一個頁面傳到第二個頁面。回到 ViewController.swift 程式碼，我們可利用 override 「`prepare(for segue: UIStoryboardSegue, sender: Any?)`」 這個函數，定義在 segue 執行前需要的準備動作，將選取到的這筆資料設定到 detail view controller 的 data property 上

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
  
    //利用segue找到它連到的viewcontroller  
    //並且確定type是DetailViewController  
    guard let detailVC = (segue.destination as?  
DetailViewController) else { return }  
  
    //取得使用者點選的位置  
    guard let selectedIndex =  
tableView.indexPathForSelectedRow?.row else { return }  
  
    //取得點選到的data  
    let selectedData = dataArray[selectedIndex]  
  
    //將data設定到第二個頁面的property上  
    detailVC.data = selectedData  
  
}
```

- (9) 回到 DetailViewController.swift，在介面初始時，把介面填上要顯示的資料，請自行完成 `viewDidLoad()` 內的 code，將 data 的資料顯示在介面的 label 及 text field 上

```
override func viewDidLoad() {  
    super.viewDidLoad()  
  
    // your code here  
}
```

- (10) 執行程式，檢查資料是否能正確顯示



4. 將修改過後的資料傳回第一個介面

- (1) 我們可使用 **unwind segue** 離開第二個介面回到第一個介面，並且抓到 **DetailViewController** 的資料更新回 **dataArray**。首先，我們需要在 **ViewController.swift** 先定義好從第二個介面回到第一個介面的 **unwind segue** 所需要執行的 **unwind action method**，它必須是一個 **IBAction** 函數

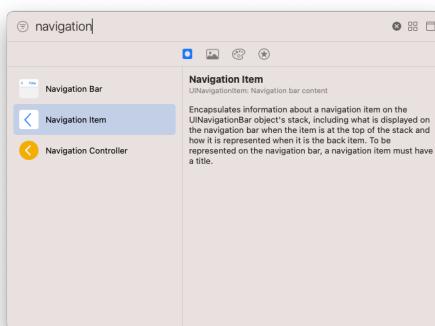
```
@IBAction func unwindFromDetailVC(segue:UIStoryboardSegue)
{
    //利用segue找到它來源的viewcontroller
    //並且確定type是DetailViewController
    guard let detailVC = segue.source as? DetailViewController else
    { return }

    //取得使用者點選的位置，當作欲修改data array的位置
    guard let selectedIndex =
    tableView.indexPathForSelectedRow?.row else {return}

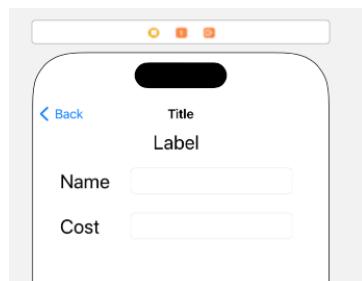
    //更新array中的dictionary
    dataArray[selectedIndex] = detailVC.data

    //更新對應的tableview的row
    tableView.reloadRows(at: [tableView.indexPathForSelectedRow!],
    with: .automatic)
}
```

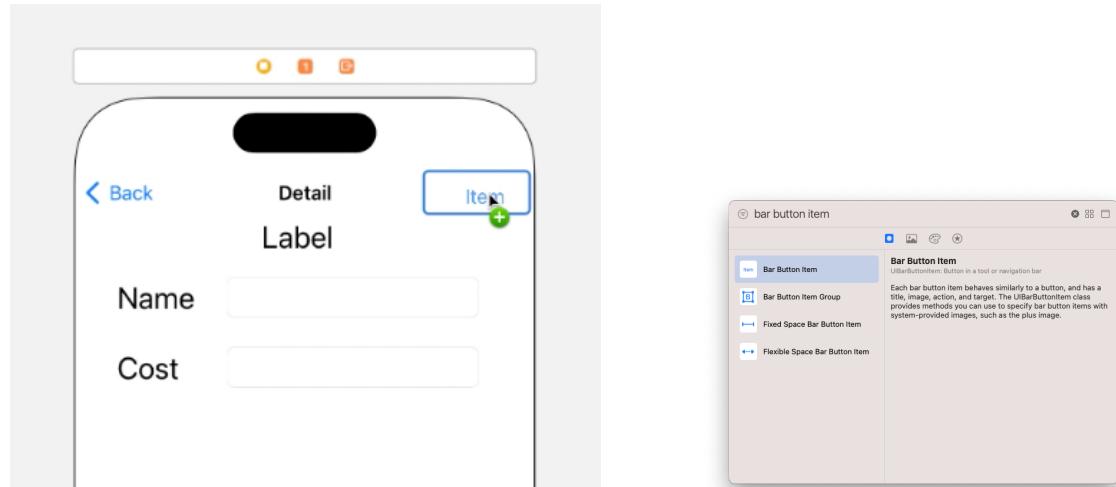
- (2) 接著，我們需要在第二個介面的上方的 **navigation bar** 加入一個 **save** 的按鈕，用它來觸發 **unwind segue**。欲增加 **navigation bar** 上物件，我們需要先將 **navigation item** 新增到介面上，在右下方 **object library** 中找到 **Navigation Item**，拉入 **DetailViewController** 介面上



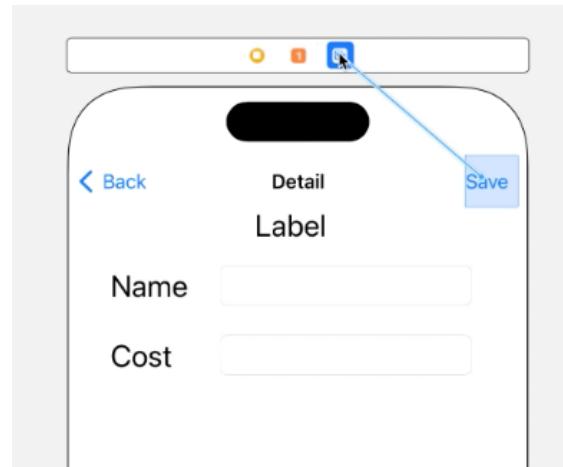
- (3) 成功完成後可以看到在 Back 的旁邊多了一個 Title



- (4) 點兩下 navigation bar 的 title，將其重新命名為 "Detail"，並且再從 object library 找到 bar button item，拉到 navigation bar 的右方，將其命稱改成 "Save"



- (5) 建立 unwind segue。control按住並使用右鍵拖曳 save 的 bar button item 到這個介面的 exit 放開



- (6) 選擇剛剛所建立的 unwindFromDetailVC method



- (7) 此時可執行看看程式，測試按下 save 後，會回到第一頁。不過如果在第二頁如果修改資料後，按下 save 不會更新第一頁的資料，因為我們在 unwind segue 執行之前，並沒有把 class 的 data property 更新。此時，我們可以再次利用 override DetailViewController 的 prepare(for segue: UIStoryboardSegue, sender: Any?) 這個函數，在 DetailViewController 執行 unwind segue 前更新 data property，**請自行完成更新 data 的 code**

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    // your code here  
  
    //將 nameField的字串抓出來轉換成String  
    //將 costField的字串抓出來轉換成Double  
    //更新 data property，讓第一頁ViewController抓到新的data資料  
    data["name"] = newName  
    data["cost"] = newCost  
}
```

- (8) 再次執行程式，可發現Detail介面更改的資料已經可以同步更改到第一個介面了，最後，**請自行補上更新總額及存檔的程式碼**

七. Demo

1. 請 Demo 最後的執行結果，可以在新的Detail介面顯示詳細資料，修該並儲存資料後，第一個介面會跟著更新，並存檔。