core data

第一次使用Core Data就上手

林哲緯 2021/06/16 17:49:17

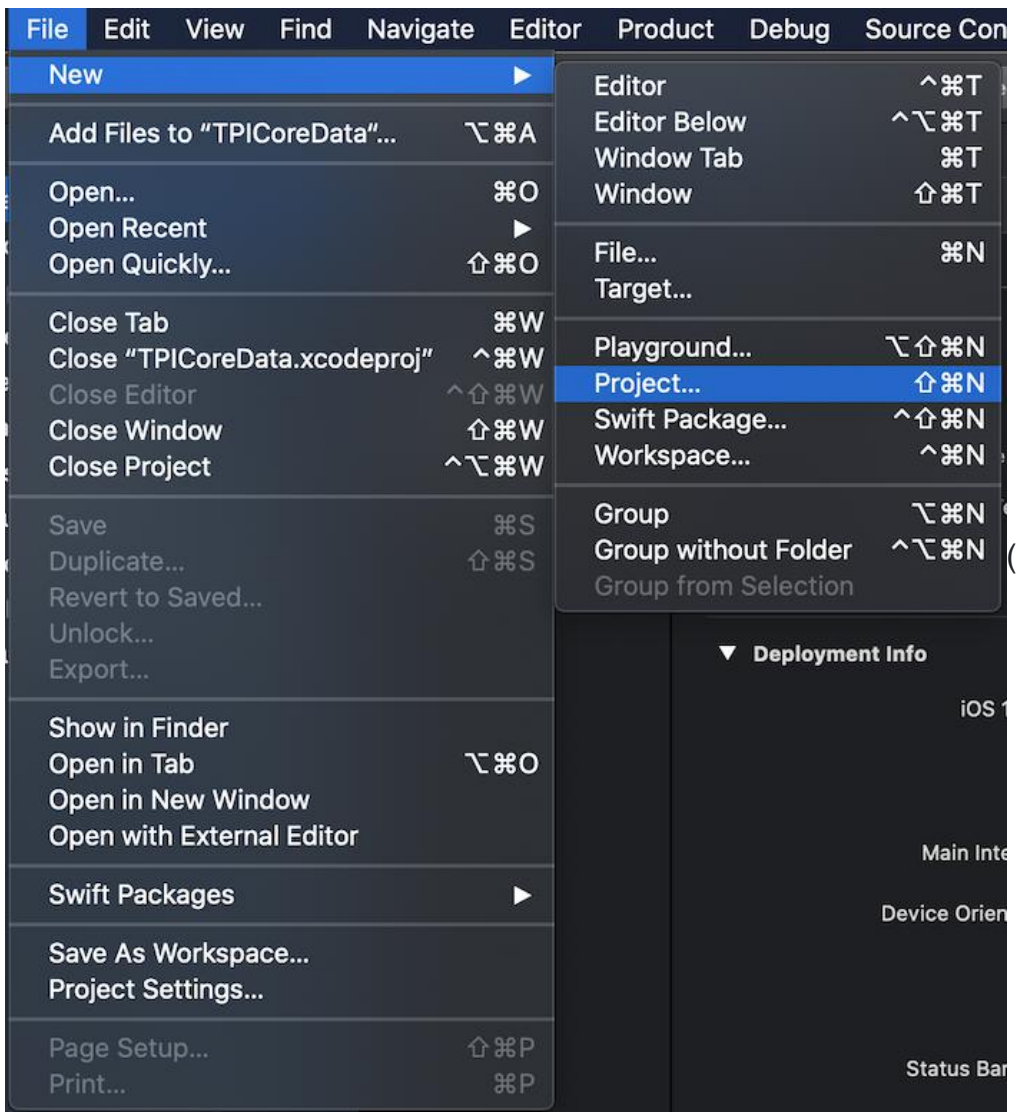
第一次使用Core Data就上手

前言：

在iOS APP有很多資料庫的應用，如：SQLite,FMDB,Realm,Firebase等，其實在iOS的開發工具的Xcode中就有一個簡化了資料庫的處理，讓你不用了解 SQL 指令也可以快速的為應用程式建立並使用資料庫框架：Core Data 。

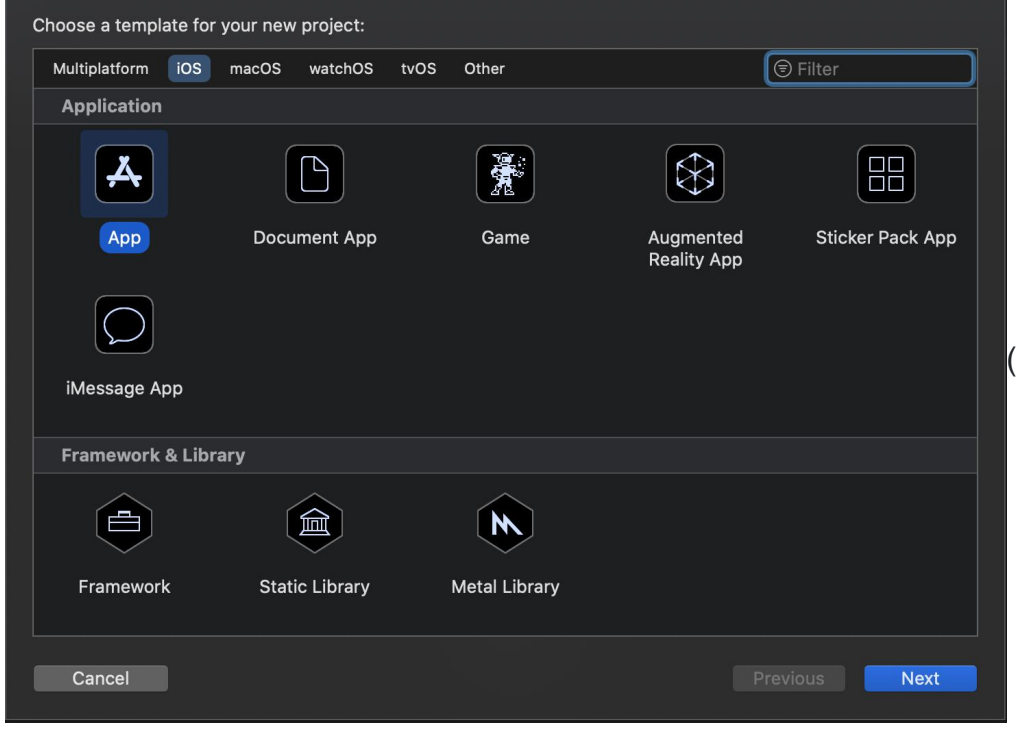
新增專案：

打開Xcode -> File -> New -> Project ...



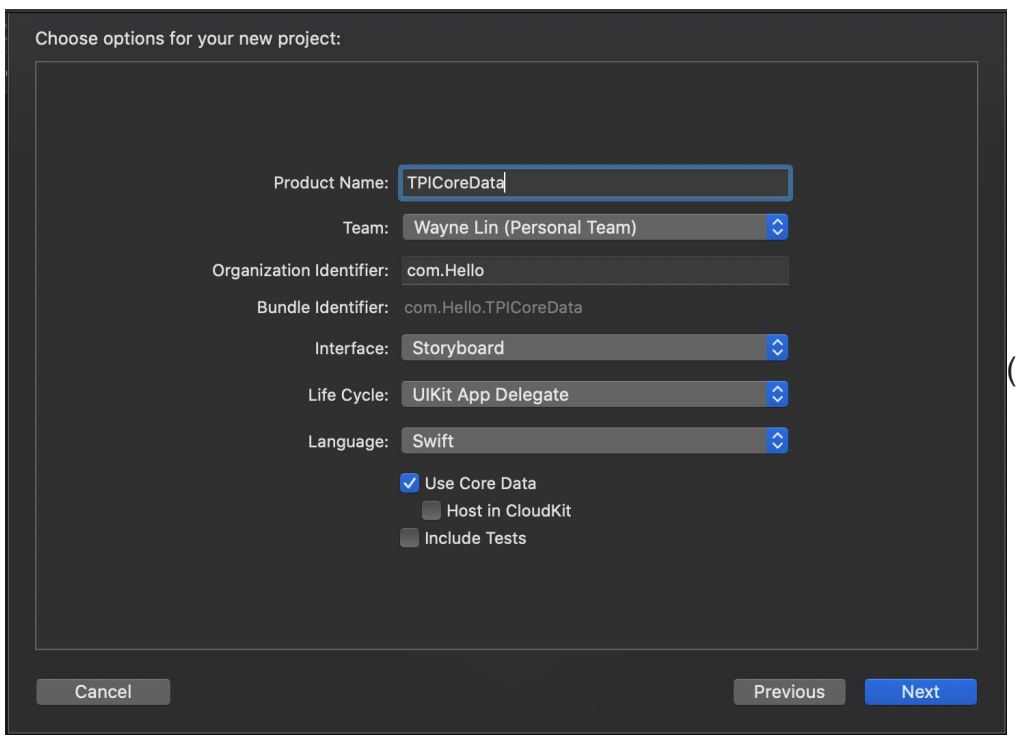
(圖1)

選擇iOS -> App -> Next



(圖2)

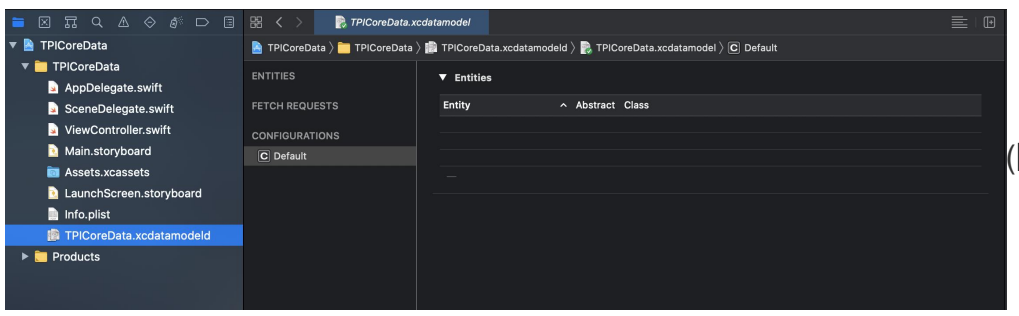
輸入專案名稱：TPICoreData -> 將Use Croe Data 選項打勾 -> Next



(圖3)

專案中的檔案列表多了一個檔案：TPICoreData.xcdatamodeld

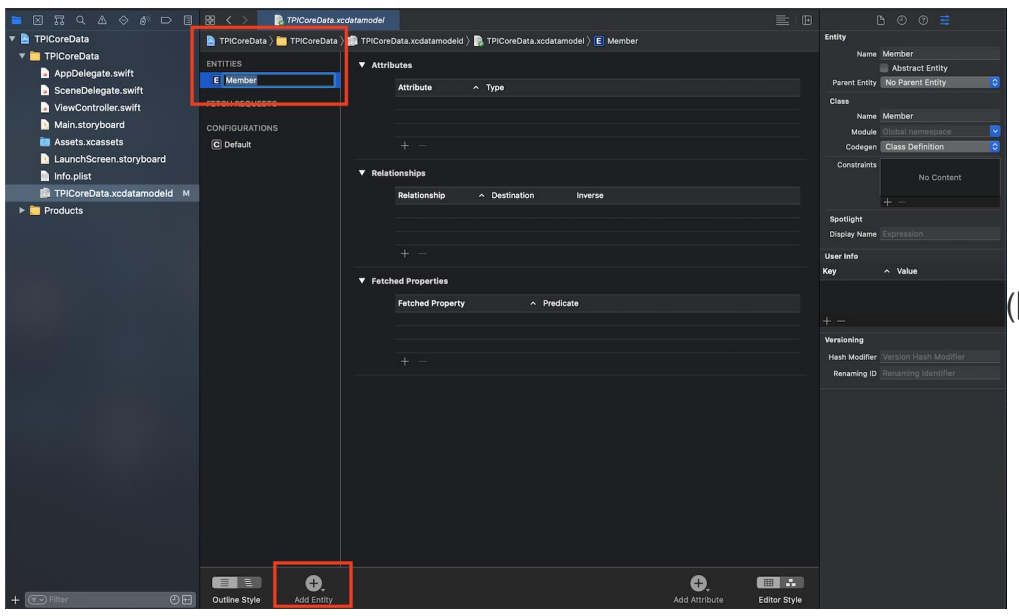
這個就是我們可以用來設定Entity和Attribute的檔案



(圖4)

設定Entity和Attribute：

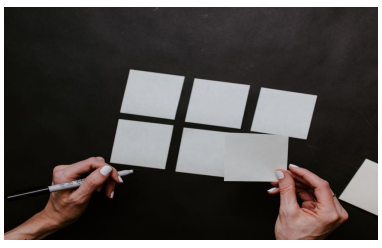
點選下方Add Entity 並將新增在 Entities 的 Entity 並重新命名為 Member



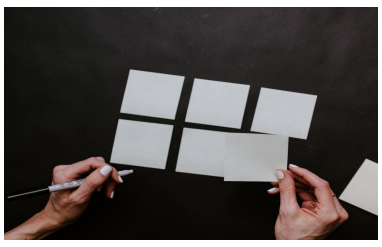
(圖5)

Swift JSON Decode with

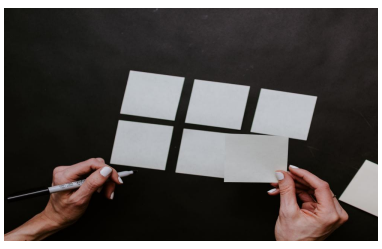
最新文章



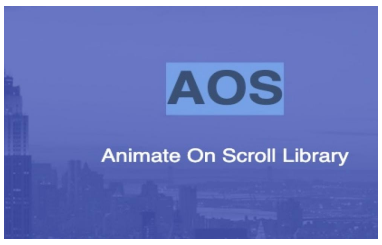
Angular Custom Modal -共用元件實作 - 彈窗 (3)



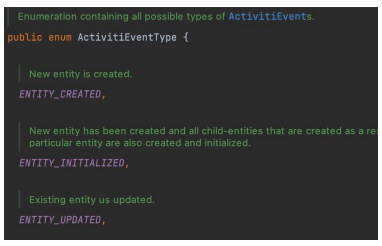
Angular Custom Modal - 共用元件實作 - 彈窗 (2)



Angular Custom Modal - 共用元件實作 - 彈窗 (1)



AOS-Animate 套件運用

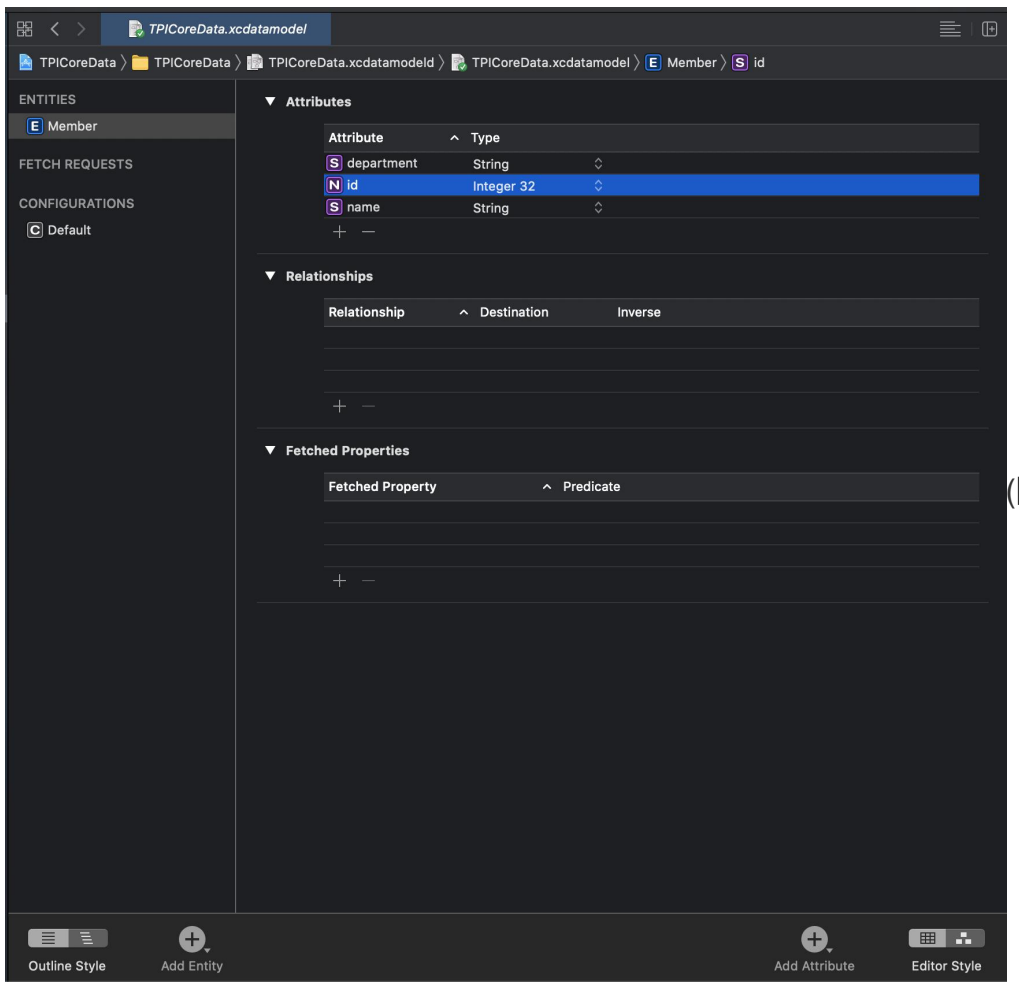


Activiti事件監聽入門

我把CSS變成Photoshop了！我跟mix-blend-mode剛認識

新增完 Entity之後，點選 Attributes 的 “＋” 分別新增三個Attribute：department、id、name

Type 為每一個Attribute的類型，分別設定為 String、Integer32、String

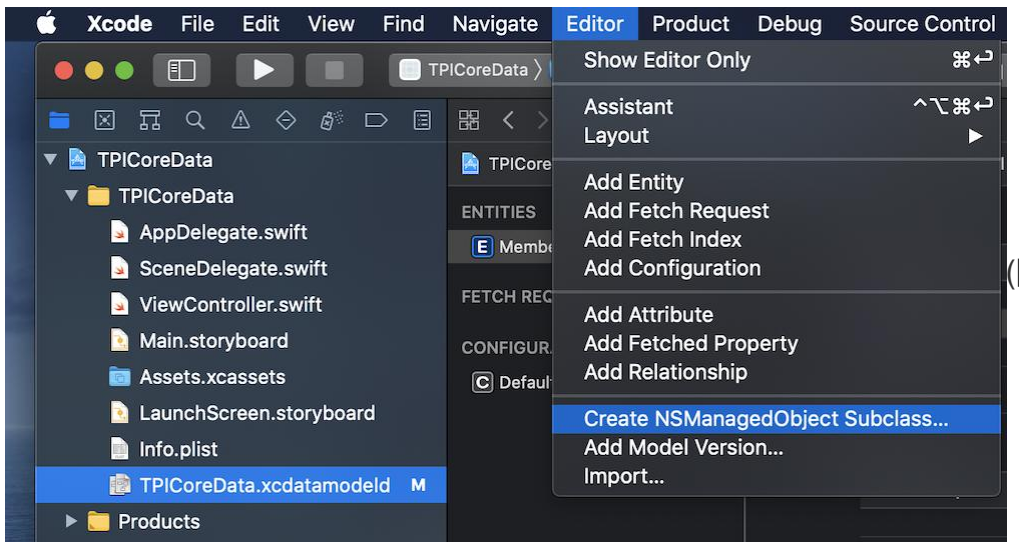


(圖6)

建立Entity的NSManagedObject類別：

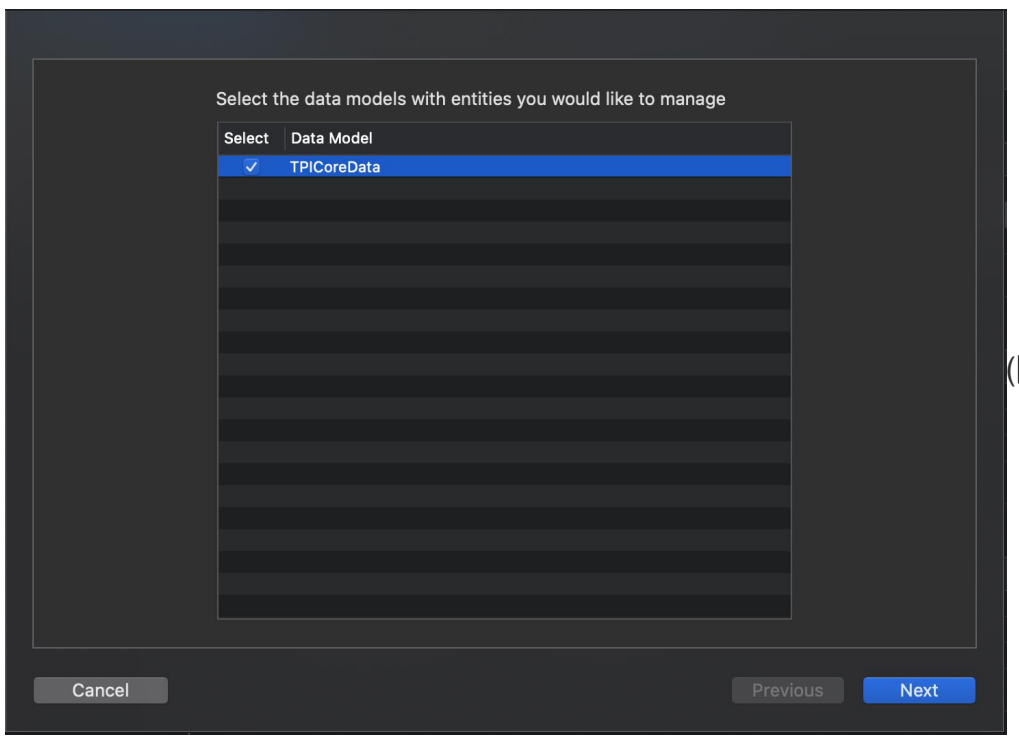
為了要讓建立的Entity能夠在code裡使用，我們要建立一個Entity的類別

選擇TPICoreData.xcdatamodeld -> 點選 Editor -> Create NSManagedObject Subclass...



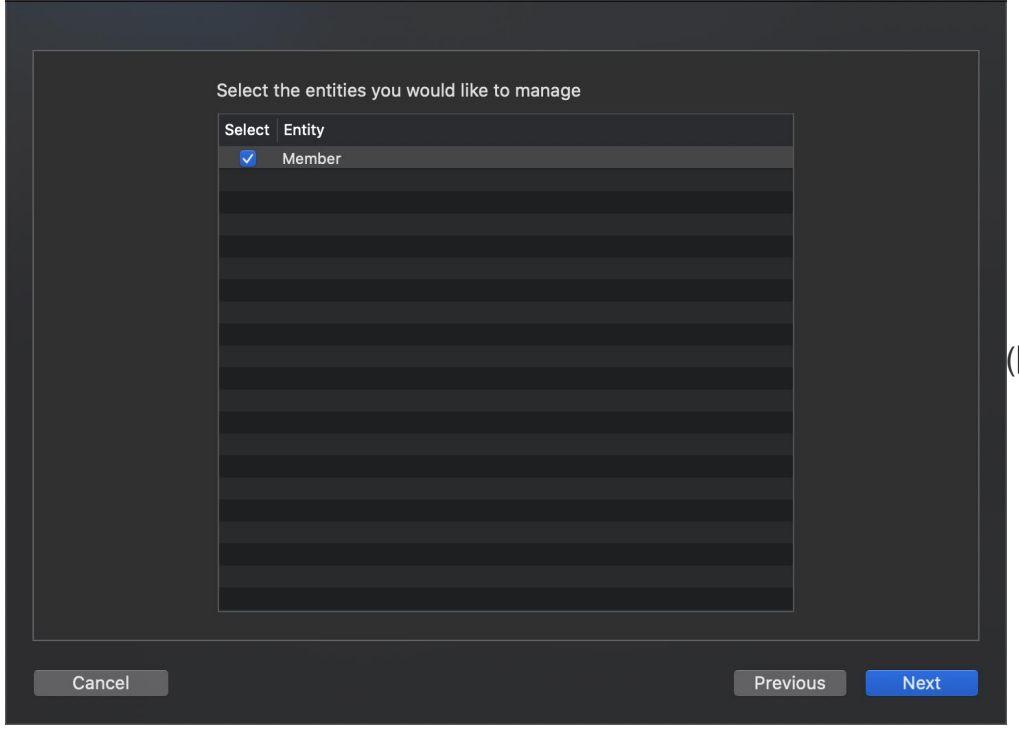
(圖7)

打勾之後點選 Next



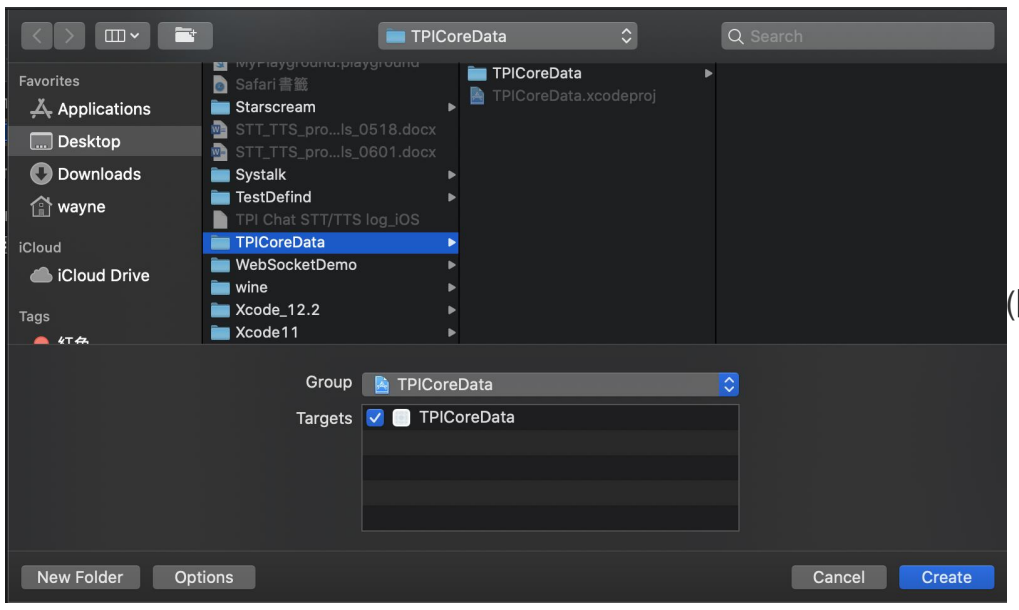
(圖8)

打勾之後點選 Next



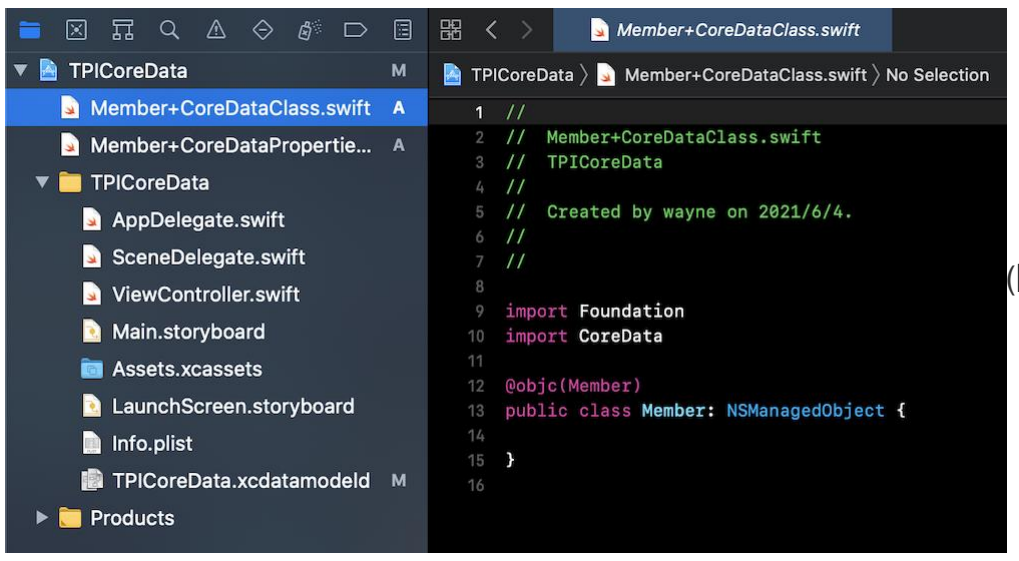
(圖9)

選擇檔案路徑後點選Create

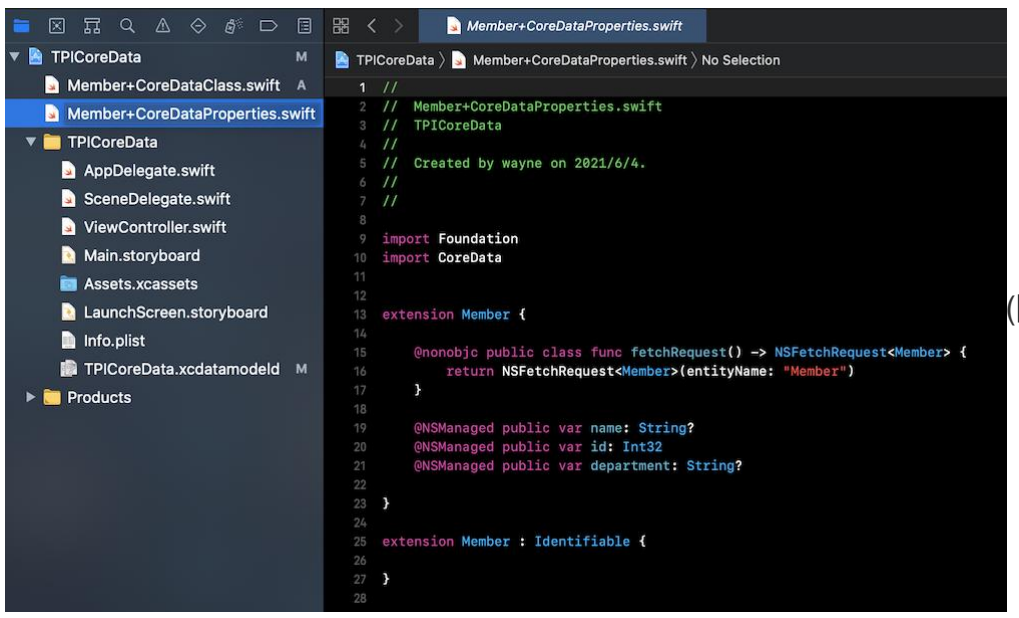


(圖10)

就會得到兩個檔案：Member+CoreDataClass.swift
和 Member+CoreDataProperties.swift



(圖11)

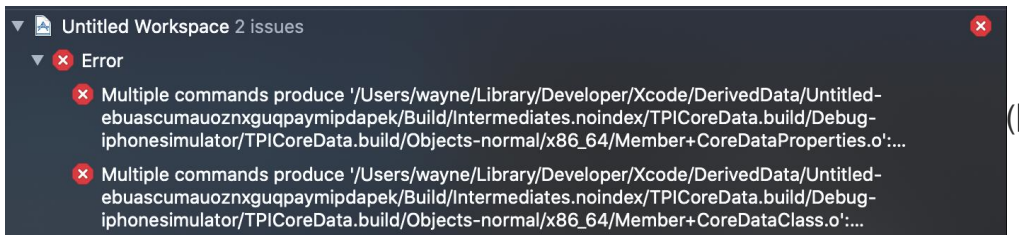


(圖12)

我們建立的 Entity類別是繼承自NSManagedObject，透過剛剛的操作讓Xcode這邊幫我們建立了property

注意：這邊可能會遇到一個問題

透過剛剛的方式生成檔案可能會出現以下錯誤

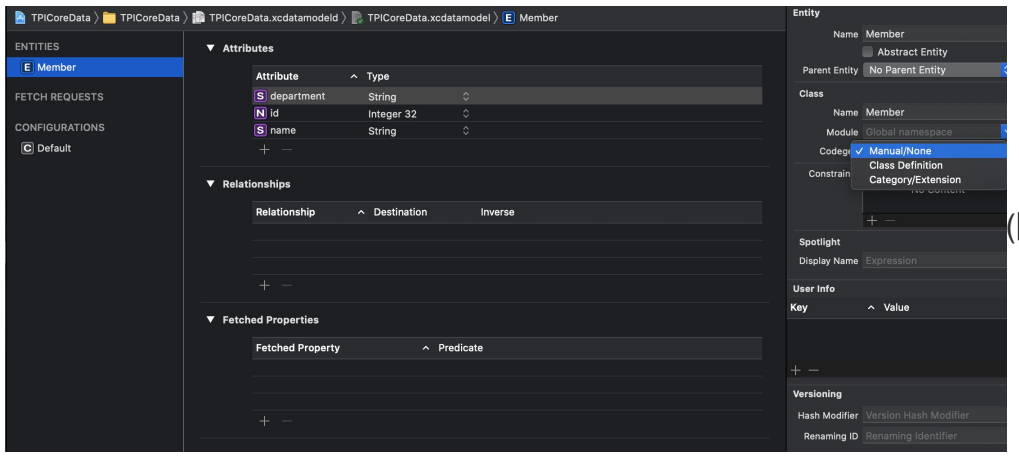


(圖14)

其實Xcode在我們建立Entity的時候，就已經幫我們自動建立好Entity的類別了

如果要解決這個錯誤，就需要調整Entity的class設定，將Codegen設定為Manual/None

這樣就Xcode就不會自動生成Entity的類別了



(圖15)

實作Core Data：

剛剛新增的Entity和Attribute是以部門人員為範例

department：部門

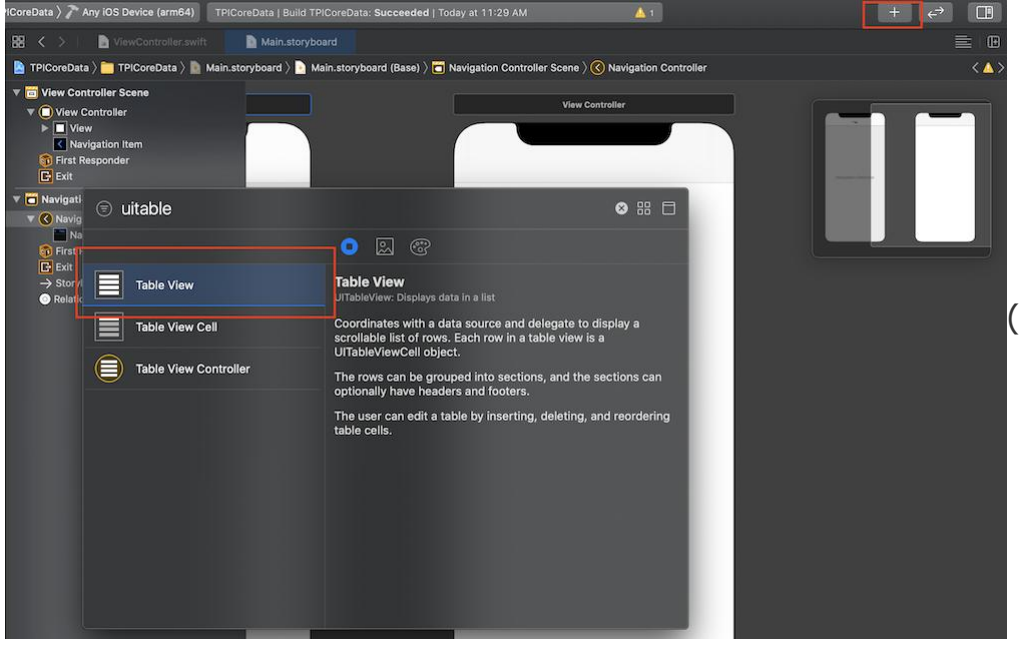
id：編號

name：姓名

```
import Foundation
import CoreData


extension Member {

    @nonobjc public class func fetchRequest() -> NSFetchRequest<Member> {
        return NSFetchRequest<Member>(entityName: "Member")
    }

    @NSManaged public var name: String?         //姓名
    @NSManaged public var id: Int32             //編號
    @NSManaged public var department: String?   //部門

}

extension Member : Identifiable {

}
```
(圖13)

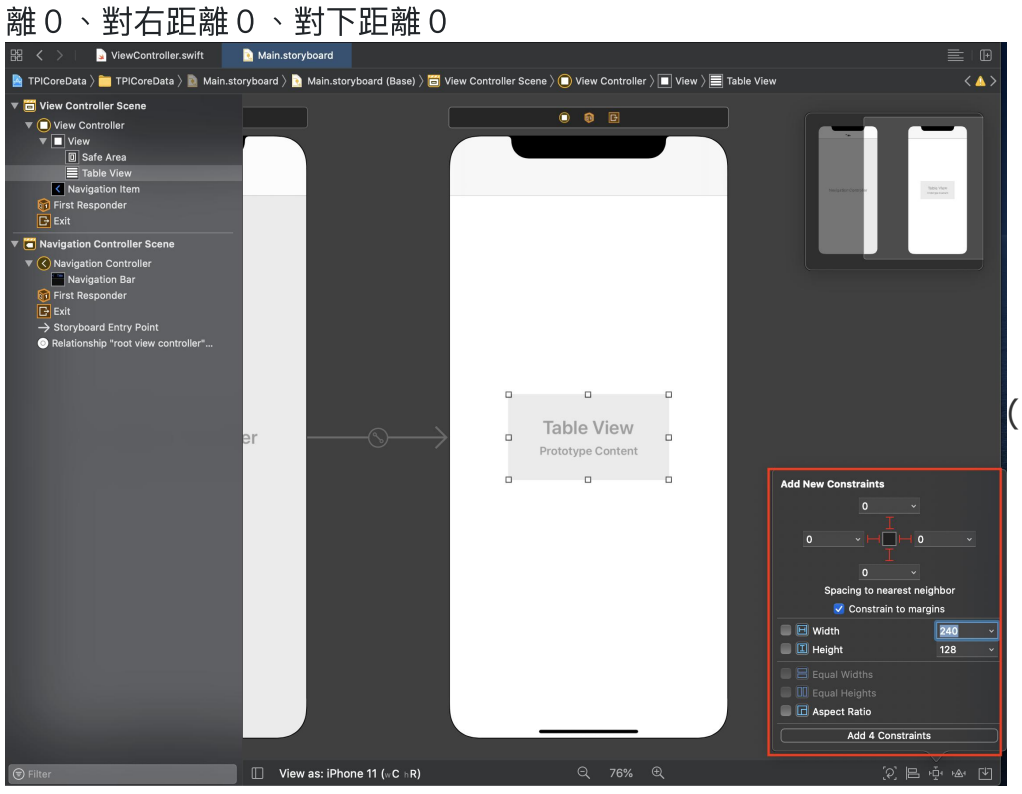
在實作Core Data之前，我們先對畫面(ViewController)做一些設定

點選 Main.Storyboard -> ViewCotroller -> 點選＋號 -> 選取TableView -> 將TableView拖拉到ViewController中



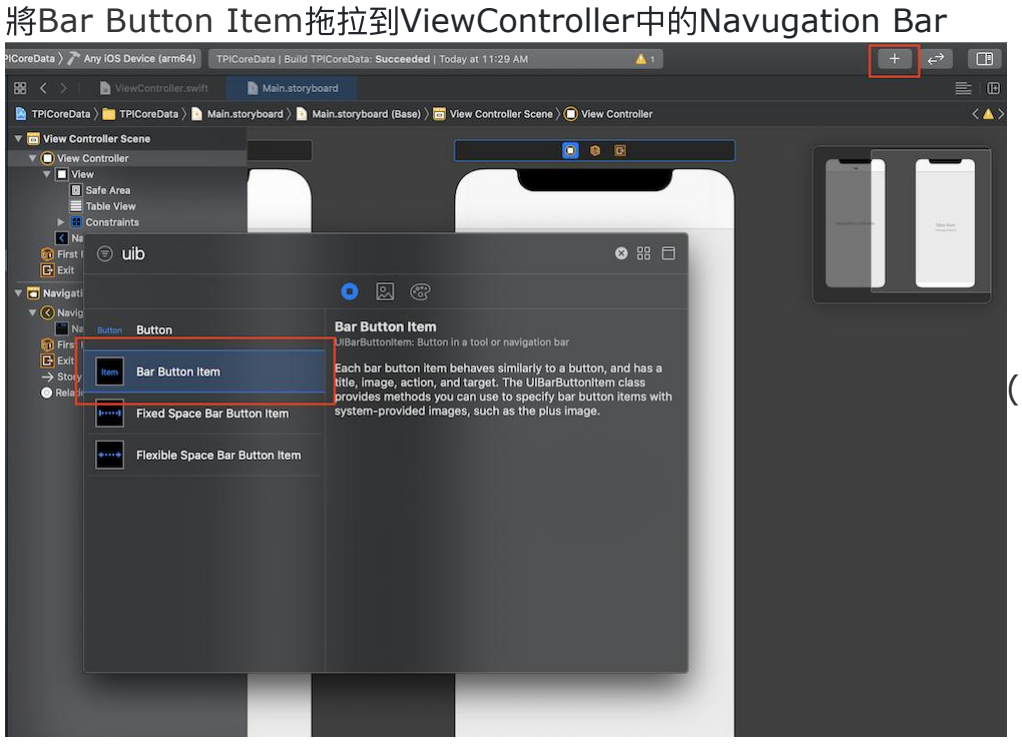
(圖16)

將剛剛加入的TableView配置適當的Constraints，我這裡設定的是：對上距離 0 、對左距離 0 、對右距離 0 、對下距離 0



(圖17)

加入一個Bar Button Item
點選 Main.Storyboard -> ViewCotroller -> 點選＋號 -> 選取Bar Button Item -> 將Bar Button Item拖拉到ViewController中的Navugation Bar

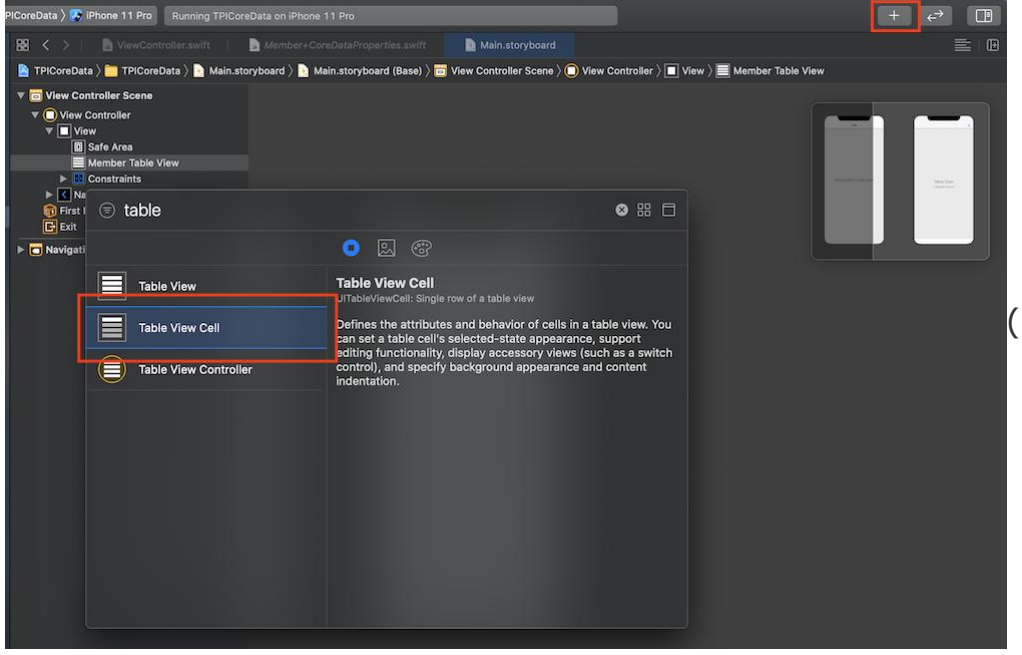


(圖18)

並將Bar Button Item中System Item設定為Add

(圖19)

加入一個TableViewCell
點選 Main.Storyboard -> ViewCotroller -> 點選＋號 -> 選取TableViewCell -> 將TableViewCell拖拉到TableView中

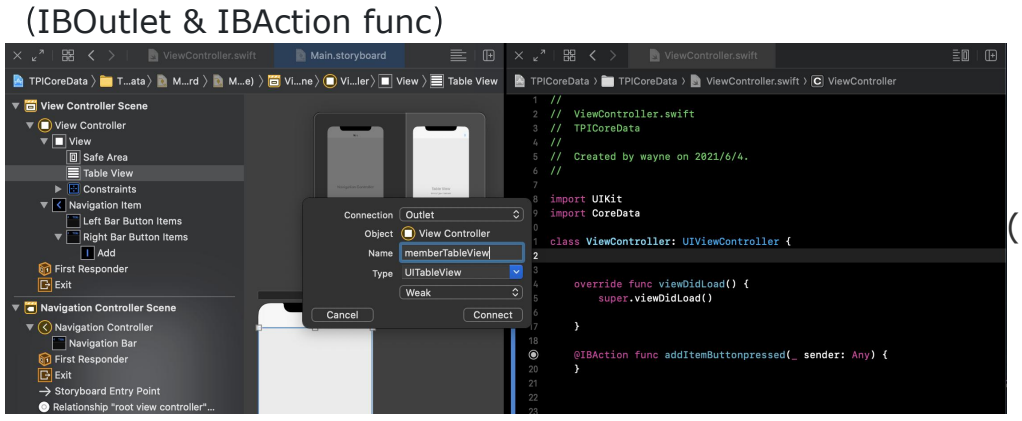


(圖22)

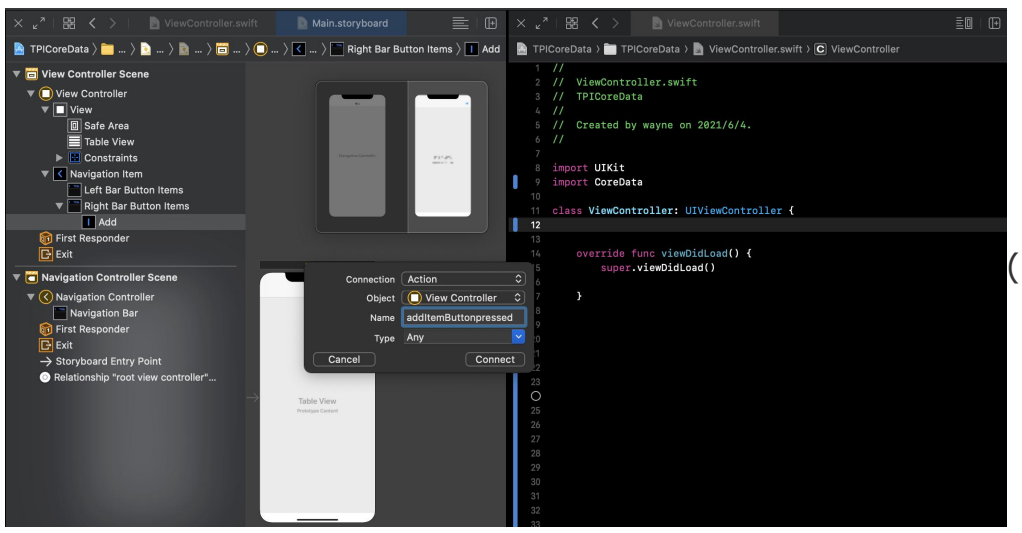
並將TableViewCell的Identifier設定為"Cell"

(圖23)

將需要使用的元件新增且把樣式設定完畢後，再來就是要加入元件和程式的關聯了（IBOutlet & IBAction func）



(圖20)



(圖21)

完成元件與程式的關聯，再來就是程式撰寫

分析一下需要實作的功能

1.設定TableViewDelegate、TableViewDataSource和相關參數
2.資料的新增，點擊畫面右上角＋號，新增一筆資料
3.資料的查詢，新增的資料透過Tableview的特性，以List的方式顯示
4.資料的修改，點擊TablewView的item，修改資料
5.資料的刪除，點擊TablewView的item，刪除資料

將上述功能一一實作就能掌握Coer Data的基本功能了

1.設定TableViewDelegate、TableViewDataSource和相關參數

```
    //宣告Core Data 常數
    let context = (UIApplication.shared.delegate as!
AppDelegate).persistentContainer.viewContext

    //宣告一個Array，紀錄資料庫查詢出來的結果
    var memberList:[Member] = []

    override func viewDidLoad() {
        super.viewDidLoad()
        //將TablewView的delegate和dataSource指定給Self(ViewCotroller)
        self.memberTableView.delegate = self
        self.memberTableView.dataSource = self
    }
```

```
//實作UITableViewDelegate,UITableViewDataSource
extension ViewController:UITableViewDelegate,UITableViewDataSource
{

    func tableView(_ tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {
        // Core Data 資料長度
        return self.memberList.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell {
        //顯示 Core Data 資料
        let cell = tableView.dequeueReusableCell(withIdentifier:
"Cell", for: indexPath) as UITableViewCell
        cell.textLabel?.text = "編號：" +
String(self.memberList[indexPath.row].id) + " , 部門：" +
String(self.memberList[indexPath.row].department ?? "") + " , 姓
名：" + String(self.memberList[indexPath.row].name ?? "")

        return cell
    }

    func tableView(_ tableView: UITableView, didSelectRowAt
indexPath: IndexPath) {
        //點擊cell 讓使用者選擇刪除或編輯資料
        let alert = UIAlertController.init(
            title:          "更新或刪除一筆資料",
            message:        "",
            preferredStyle: .alert
        )
        let okAction = UIAlertAction.init(title: "更新", style:
.default) { _ in
            DispatchQueue.main.async {
                self.updateObject(indexPath: indexPath)
            }
        }
        let cancelAction = UIAlertAction.init(title: "刪除", style:
.default) { _ in
            DispatchQueue.main.async {
                self.deleteObject(indexPath: indexPath)
            }
        }
        alert.addAction(okAction)
        alert.addAction(cancelAction)
        self.present(alert, animated: true, completion: nil)
    }


}
```

2.資料的新增，點擊畫面右上角＋號，新增一筆資料

```swift
    @IBAction func addItemButtonpressed(_ sender: Any) {

        self.showAlert(type:"新增", alertTitle: "新增人員資料",
actionHandler: { (textFields: [UITextField]?) in
            DispatchQueue.main.async {
                self.insertObject(department: String(textFields?
[0].text ?? ""), id: Int(textFields?[1].text ?? "") ?? 0, name:
String(textFields?[2].text ?? ""))
            }
        })
    }

    //新增資料
    func insertObject(department: String, id: Int, name: String) {
        let member =
NSEntityDescription.insertNewObject(forEntityName: "Member", into:
self.context)as! Member
        member.id = Int32(id)
        member.name = name
        member.department = department
        do {
            try self.context.save()
        } catch {
            fatalError("\(error)")
        }
        //新增完畢後查詢資料庫資料並將資料庫資料顯示在TableView上
        self.memberList = self.selectObject()
        DispatchQueue.main.async {
            self.memberTableView.reloadData()
        }
    }
```

```swift
    //實作一個Alert 讓user輸入新增或編輯資料
    func showAlert(type:String, alertTitle: String, actionHandler:
((_ textFields: [UITextField]?) -> Void)? = nil) {
        let alert = UIAlertController.init(
            title:          alertTitle,
            message:        "",
            preferredStyle: .alert
        )
        //新增三個輸入框分別讓使用者輸入部門、編號、姓名
        for index in 0...2 {
            alert.addTextField { (textField:UITextField) in
                if index == 0 {
                    textField.placeholder = type + "部門"
                }else if index == 1 {
                    textField.placeholder = type + "編號"
                }else if index == 2 {
                    textField.placeholder = type + "姓名"
                }
            }
        }
        alert.addAction (UIAlertAction.init(title: "確定", style:
.default, handler: { (action:UIAlertAction) in
            DispatchQueue.main.async {
                actionHandler?(alert.textFields)
            }
        }))

        let cancelAction = UIAlertAction.init(title: "取消", style:
.cancel) { _ in
            DispatchQueue.main.async {

            }
        }
        alert.addAction(cancelAction)

        self.present(alert, animated: true, completion: nil)
    }
```

3.資料的查詢，新增的資料透過Tableview的特性，以List的方式顯示

```
    //查詢資料
    func selectObject() -> Array<Member> {
        var array:[Member] = []
        let request = NSFetchRequest<Member>(entityName: "Member")
        do {
            let results = try self.context.fetch(request)
            for result in results {
                array.append(result)
            }
        }catch{
            fatalError("Failed to fetch data: \(error)")
        }
        return array

    }
```

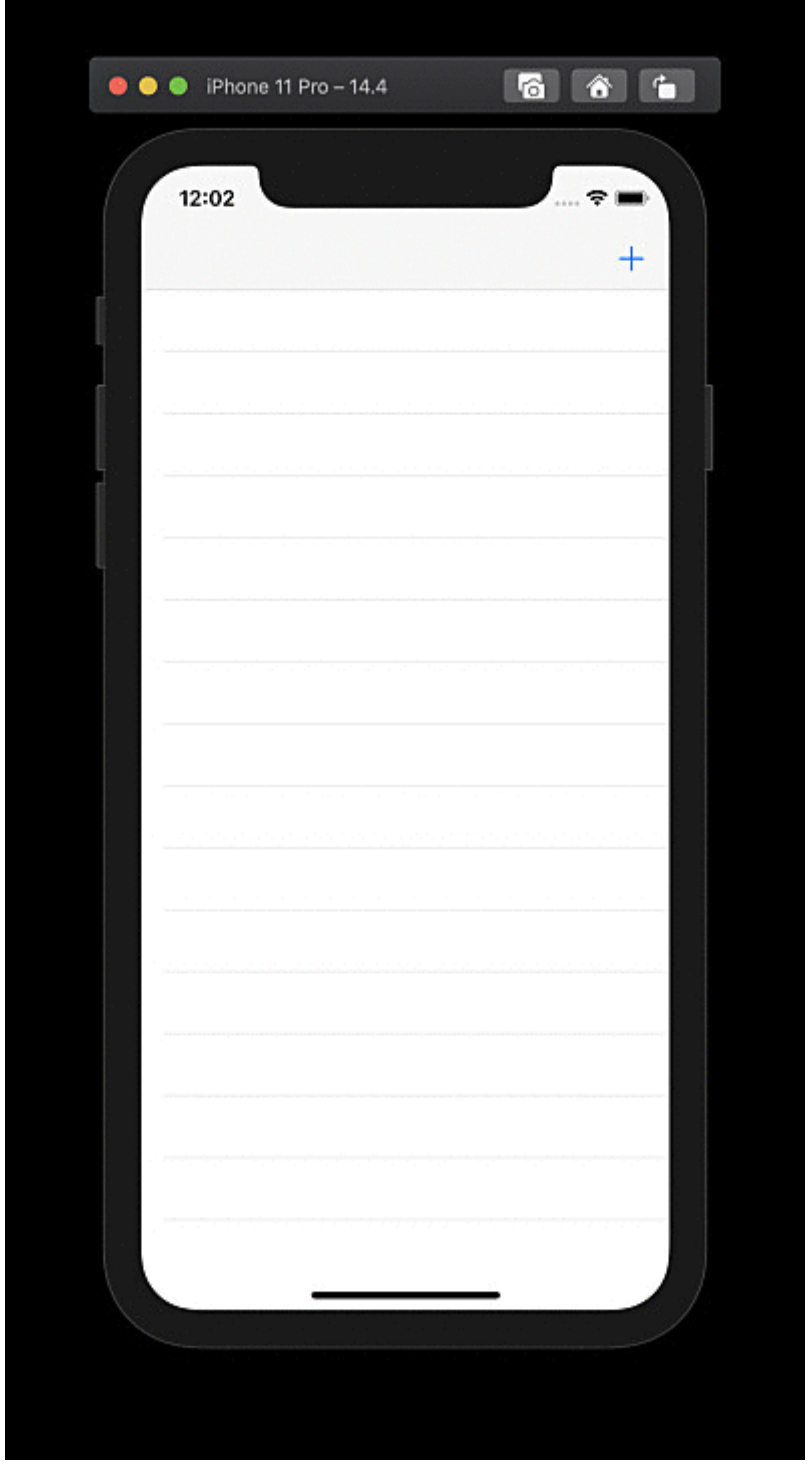
4.資料的修改，點擊TablewView的item，修改資料

```
    //更新資料
    func updateObject(indexPath:IndexPath) {
        self.showAlert(type: "修改", alertTitle: "修改人員資料") {
(textFields: [UITextField]?) in
            //更新:將查詢到的結果更新後，再呼叫context.save()儲存
            let request = NSFetchRequest<Member>(entityName:
"Member")
            do {
                let results = try self.context.fetch(request)
                for item in results {
                    if item.id == self.memberList[indexPath.row].id
&&
                        item.name ==
self.memberList[indexPath.row].name &&
                        item.department ==
self.memberList[indexPath.row].department {

                        item.department = textFields?[0].text
                        item.id = Int32(Int(textFields?[1].text ??
"") ?? 0)
                        item.name = textFields?[2].text
                    }
                }
                try self.context.save()
            }catch{
                fatalError("Failed to fetch data: \(error)")
            }
            //新增完畢後查詢資料庫資料並將資料庫資料顯示在TableView上
            self.memberList = self.selectObject()
            DispatchQueue.main.async {
                self.memberTableView.reloadData()
            }
        }
    }
```

5.資料的刪除，點擊TablewView的item，刪除資料

```swift
    //刪除資料
    func deleteObject(indexPath:IndexPath) {
        //刪除:將查詢到的結果刪除後，再呼叫context.save()儲存
        let request = NSFetchRequest<Member>(entityName: "Member")
        do {
            let results = try self.context.fetch(request)
            for item in results {
                if item.id == self.memberList[indexPath.row].id &&
                    item.name ==
self.memberList[indexPath.row].name &&
                    item.department ==
self.memberList[indexPath.row].department {
                    context.delete(item)
                }
            }
            try self.context.save()
        }catch{
            fatalError("Failed to fetch data: \(error)")
        }
        let alert = UIAlertController.init(
            title:          "已刪除",
            message:        "",
            preferredStyle: .alert
        )

        let okAction = UIAlertAction.init(title: "OK", style:
.default)
        alert.addAction(okAction)

        self.present(alert, animated: true, completion: {
            //新增完畢後查詢資料庫資料並將資料庫資料顯示在TableView上
            self.memberList = self.selectObject()
            DispatchQueue.main.async {
                self.memberTableView.reloadData()
            }
        })
    }
```

結果：



結論：

在iOS的開發中一定還有很多資料庫的應用，這邊只是針對Core Data實作一些基本的應用

不論是哪種實作方式，只要能夠完成功能，都是好方法

就讓我們繼續寫Code吧！

參考資料：

https://developer.apple.com/documentation/coredata

https://www.reddit.com/r/swift/comments/9t3vj0/serious_problem_with_coredata_and_nsmanagedobject/

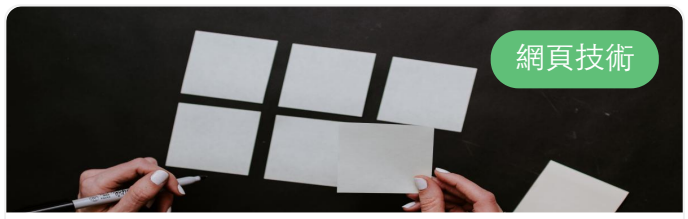
https://www.hackingwithswift.com/read/38/4/creating-an-nsmanagedobject-subclass-with-xcode

0

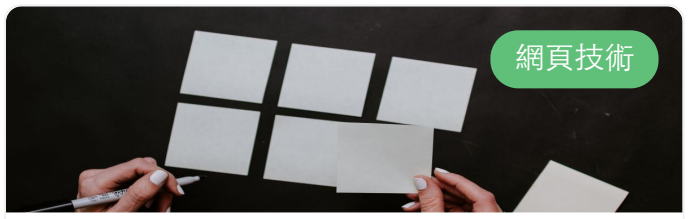
林哲緯

請先登入，再輸入您的回覆...

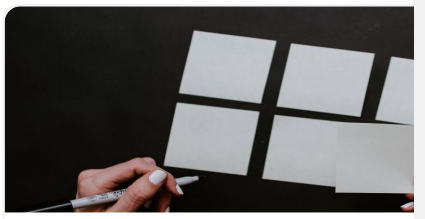
最熱門文章



Angular Custom Modal - 共用元件實作 - 彈窗 (1)



Angular Custom Modal -共用元件實作 - 彈窗 (3)



Angular Custom Moda
作 - 彈窗 (2)

線上人數：634

訂閱昕力大學

關於昕力大學

「昕力大學」（TPI University）是昕力資訊於 2015 年成立的技術共享專業論壇，由昕力資訊員工以 Xamarin、網頁技術、Java、.NET、企業系統軟體、資料庫技術、專案實務為主題，分享在工作中所學習到的專案經驗、新技術研究心得、從錯誤中學到的寶貴經驗，及其他相關的技術知識等，撰寫成文章，成為一個資訊共享及知識交流平台，期望能夠成為台灣最專業的技術論壇。

據點

台北總部

新竹專案管理中心

台中專案管理中心

台南開發中心

高雄開發中心

上海

新加坡

越南

電子報

2020

2019

2018

訂閱電子報

相關網站

昕力官網

產品中心

電話 : +886 2 8751 1610

Email : service@tpisoftware.com

昕力資訊