Blazor 開發

分為幾種資料

1. Page

Add Razor Component

副檔名為 .razor

1. Model(自己建立資料夾)
2. Service(自己建立資料夾)

建立interface 與 service (interface檔名為service前面+i)

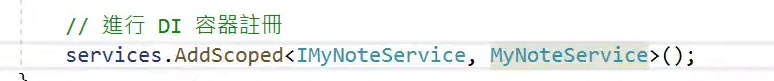
Page管理畫面

Model用於建立資料的物件

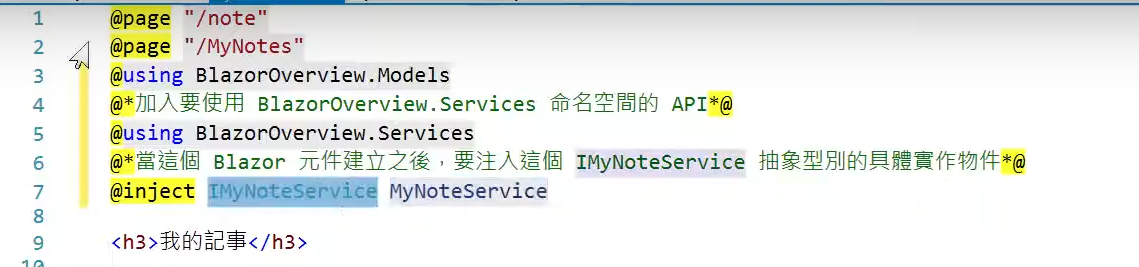
Service 用於建立 服務 與 interface

服務需要在startUp.cs 中註冊

於ConfigureService中如圖:



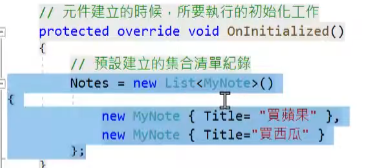
要使用Service page 或 cs 檔需用inject



為了防止後續需要非同步因為function都會以Task 型別為主

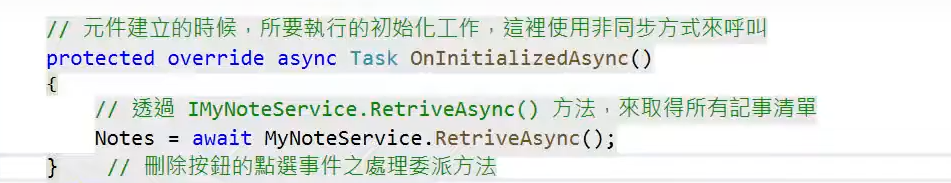
原先若使用Model的物件方法

會使用



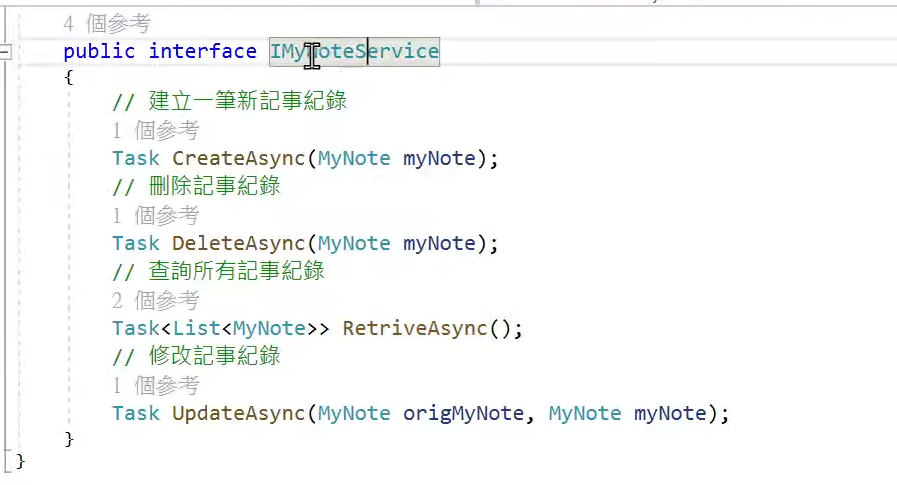
進行初始化

由於Service的方法會是task故 改為

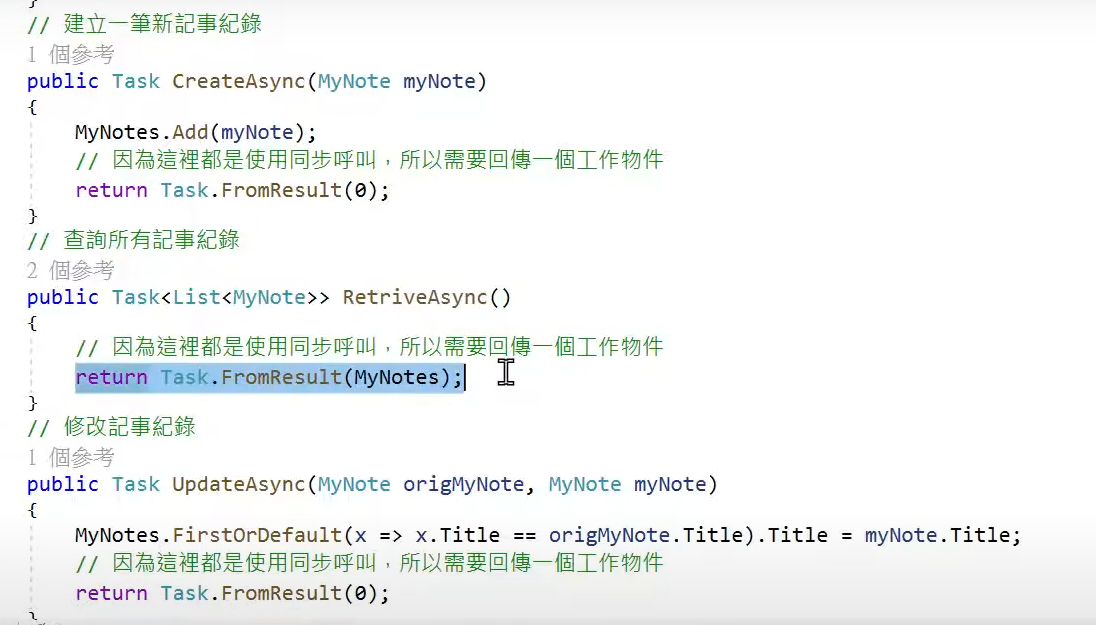


RetriveAsync 使用Service 的建構資料方法來準備資料

Interface 如下



Service的function大致如下



Task.FromResult();

主要用於需要回傳一個Task 回去 因此隨便丟了一個0回去

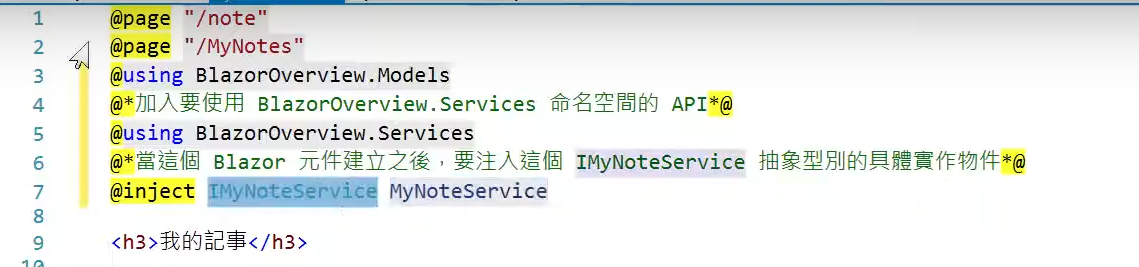
因Add MyNote時 referenvce 概念下 已不需回傳

＊依照公司專案由於Page 中 code 內容與 html 內容分開

因此

為page 與 同名.cs掛於同一集合內

因此 inject 要分別引入



Inject IMyNotesService MyNoteService

然後使用CRUD 方式操控資料

若沒分Service 原先function 如下



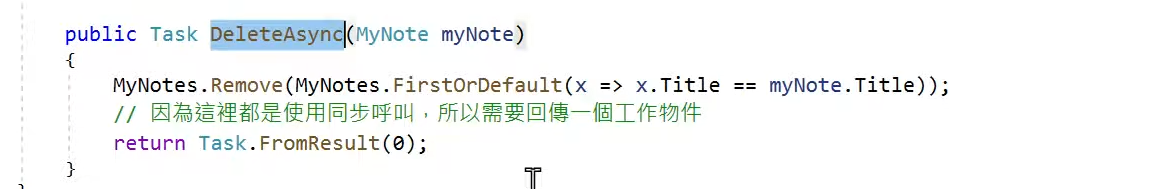
加入service 概念



因此我還是會建立方法名稱但實際的商業邏輯改在service 上實作

我想是變成實際CRUD部分上用Service 物件方法操作

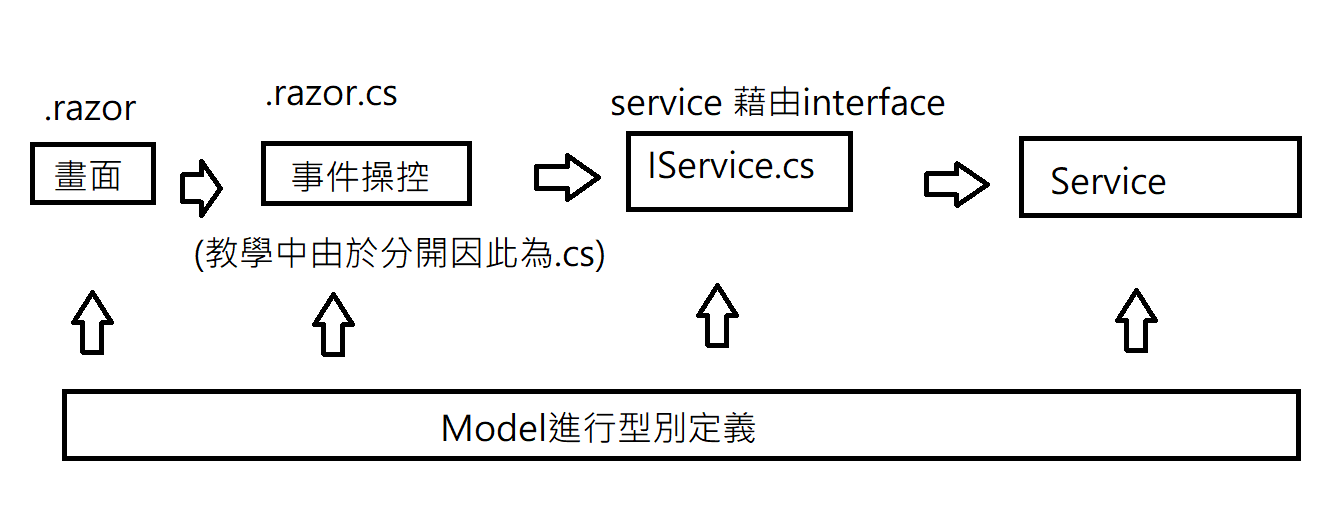
如下:



而原來function 則是討論刪除之後是否開啟某個畫面上按鈕或者彈窗部分

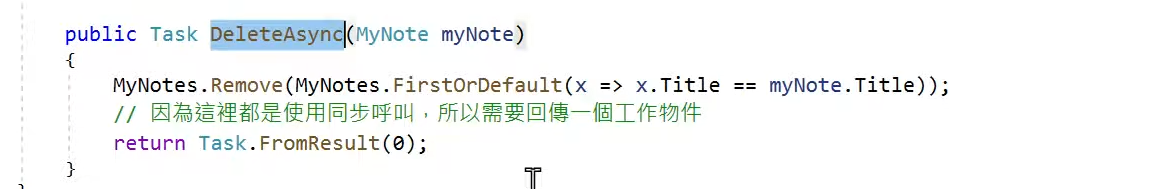
如開啟boolean 讓某個按鈕顯示等等

因此整體概念為



特別注意的是

由於使用依賴性注入



可以看到delete語法中

是丟入項目

將建構的子集合(MyNotes)中 將 myNote 給 remove

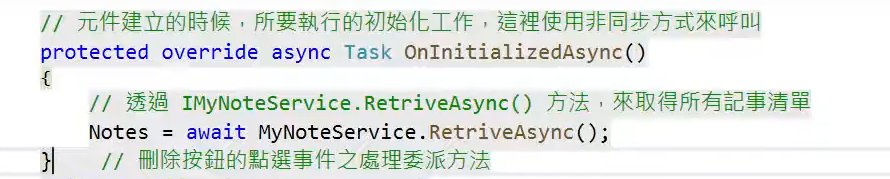
而MyNote的List物件正是在StartUp時 由於進行註冊在程式運行後會執行該Service建構方法先建立好了資料內容

而畫面上的資料內容則是使用Retrive 取得 startUp時 經由建構方法產生出來的資料集合引導至畫面上



可以看到直接丟回去完全不用寫,執行時機為StartUp時

而畫面則是為init時 藉由Retrive 進行取得



＊記憶點service 沒有init,init是畫面用的

Service 由於為class 討論的 是建構方法

Page(.razor)才討論init

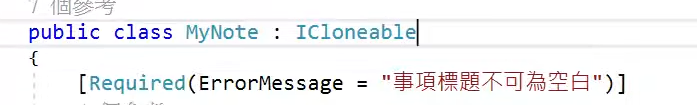
＊要特別注意的事情是

Update來說 Service 上的update 為 進行資料內容上真正的改變

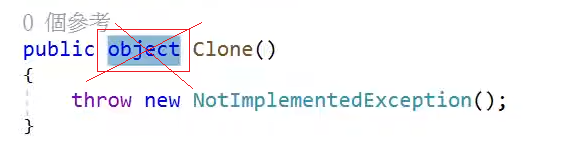
而razor.cs 上的update 反而比較偏向於畫面上的操作

＊Edit 會用到Clone 概念理由是有可能放棄變更,但若直接使用畫面上的obj還要寫還原邏輯因此會先複製一份

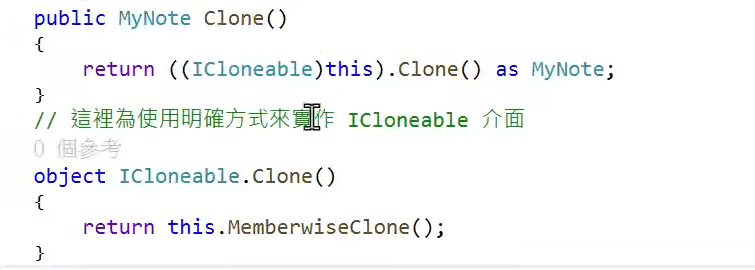
Model繼承IClonable介面



不要直接實作介面因為會是弱型別

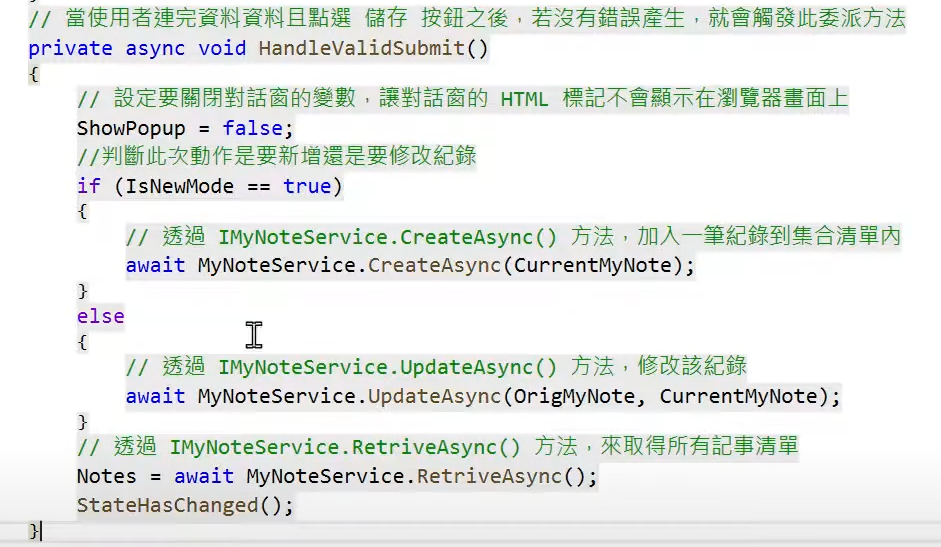


使用依明確方法進行實作



MemberwiseClone是.NET 物件保留方法用來Clone 物件

而實際的變更確實使用submit 事件時 才會使用service的update

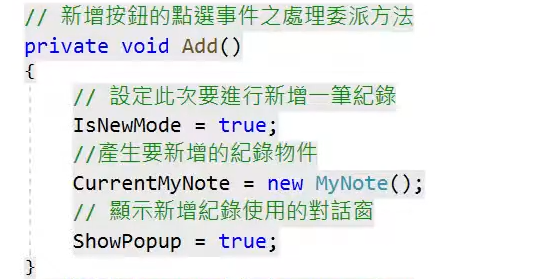


與page 內容作呼應



可以看到page 中格式綁定了submit 後使用 HandleValidSubmit事件

而CurrentMyNote 是在按下新增 或編輯按鈕時定義的



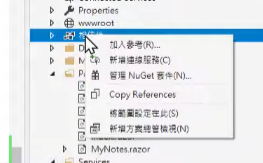
其中StateHasChange();

方法便是由於使用不同task去變更原先物件的值卻沒有偵測到物件值有變更

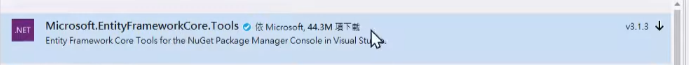
便使用此方法進行檢查並刷新

若要連SQL

相依性>加入參考



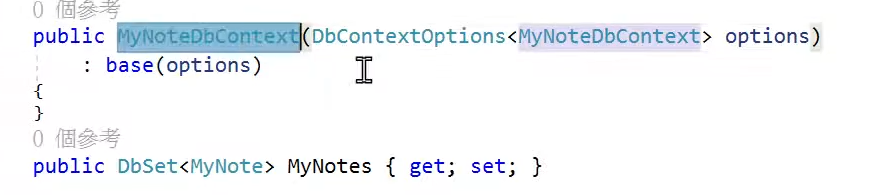




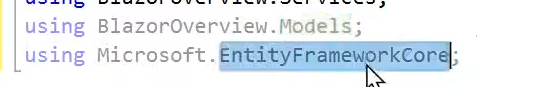
Model 中建立一個類別

依此範例名為MuNotesDBContext繼承DbContext

設定建構式



StartUp導入EntityFrankworkCore與Model



使用Entityfrankwork 要於ConfigureService 中宣告



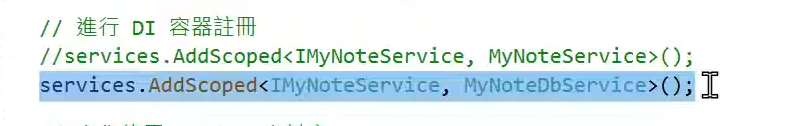
不過教學中 最後需要做

Add-Migration init

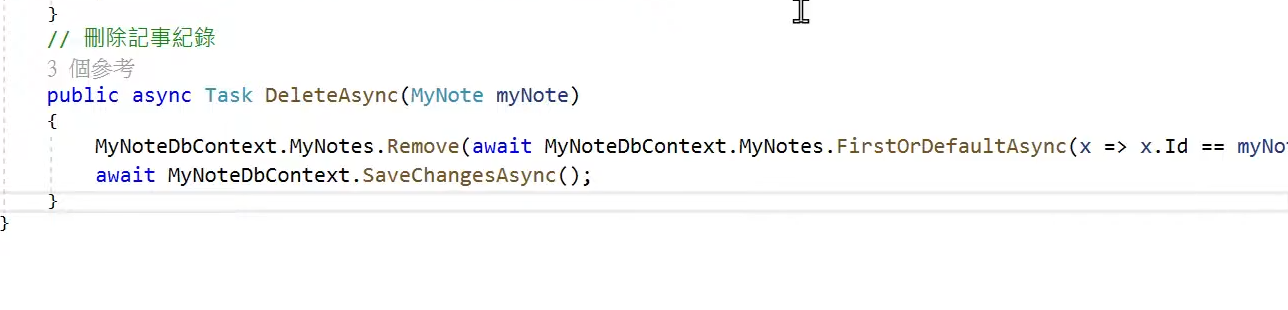
與

Updata database 不太適合公司目前做法暫不討論

StartUp.cs 改



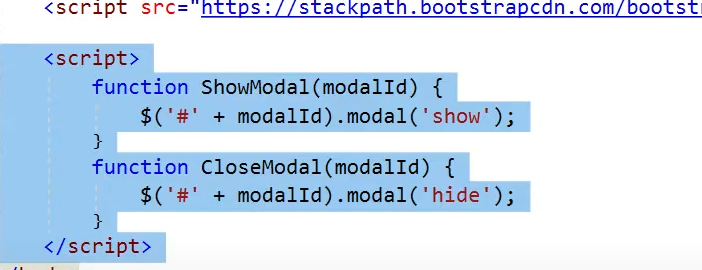
大概CRUD(依此類推)



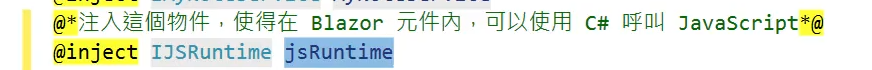
呼叫javascript

於page/\_host.cshtml

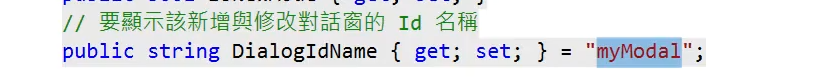
撰寫一段js



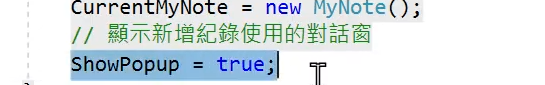
於.razor 或.cs 需要inject



用C# 定義JS ID



原先邏輯是使用



來決定是否顯示modal選單



並且由於為await 方法

故若原先不是Task 的funtion 就補上 asycn