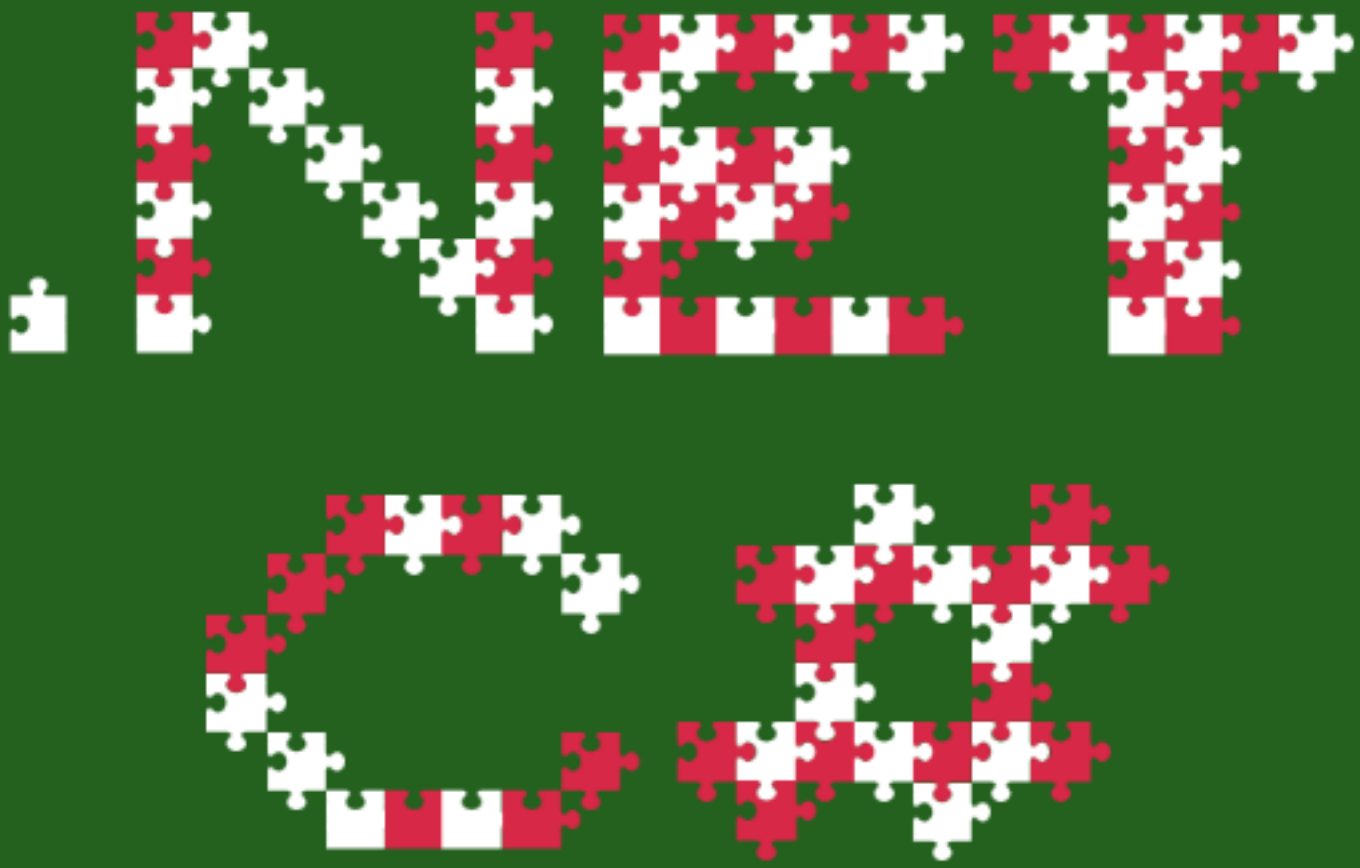
Blazor 快速體驗 HOL
Hands-on Lab

動手練習系列



Vulcan Lee 著

Blazor 快速體驗

Hands-On Lab 動手練習

Vulcan Lee

這本書的網址是 http://leanpub.com/Blazor-Quick-Overview

此版本發布於 2020-01-13

Contents

關於本書

這是一本帶領對於 Blazor 有興趣的新手開發者，可以快速體驗這個微軟最新的網頁開發框架技術的開發過程，在這本書中，將不會講解枯澀的相關技術內容，而是設計一個應用情境，也就是一般常用的 CRUD (新增 Create, 查詢 Retrive, 更新 Update, 刪除 Delete) 的記事應用程式需求開發，從無到有的帶領讀者透過 Visual Studio 2019 這個開發工具，設計出的 Blazor 伺服器端的這樣應用程式。

所以，購買本書的讀者，將會強烈建議讀者要跟著本書的內容，逐一進行專案的設計與練習，在每個練習階段，都會有不同要帶給讀者的學習方向；透過這樣的練習開發過程，就會明瞭 Blazor 這個開發框架技術究竟可以做到什麼樣的強大功能；不過，這本書中並不會詳盡說明各種 Blazor 開發技術內容，而是提供一個動手練習實作的說明操作步驟，體驗 Blazor 專案開發過程。因此，若您沒有這樣的興趣或者這樣的需求，請不要購買這本書。

因此，當讀者完成所有的專案練習開發，相信您對於 Blazor 這個優秀的開發框架必定有更清楚的認識，了解到不需要使用到繁雜的 JavaScript 程式語言，僅使用 Blazor 就可以做到那些網站設計上的功能。

這本書能提供什麼

在這本書裡面，將會提供 8 章的內容，分別是

- 建立 Blazor Server Side 伺服器端的專案

  了解如何透過 Visual Studio 2019 開發工具，開始建立一個伺服器端的 Blazor 專案。

- 開始設計 CRUD 的新增與刪除功能

  首先，這裡將會透過剛剛建立的 Blazor Server Side 伺服器端專案，設計出一個使用記憶體作為儲存空間的 CRUD 應用程式，不過，在這裡將會先設計出新增與刪除的功能，在下一章內容才會介紹如何使用對話窗來強化新增與修改的功能；另外，在這兩章的內容裡，會將 CRUD 的各項程式碼，直接寫 Blazor Component 元件裡，還是那句話，這裡完全不會使用到任何 JavaScript 程式碼，僅僅使用到 C# 程式語言與 HTML 宣告式標記語言而已。
- 增加 CRUD 的新增與修改對話窗功能

  延續上一章的開發結果，將會繼續開發新增與修改功能，不過，對於要新增與修改的紀錄，可以透過 BootStrap 的對話窗功能來進行記錄輸入；不過，在這裡將不是透過呼叫 jQuery 的方式，而是透過 C# 資料綁定的處理機制來顯示這個對話窗。
- 建立記事服務並且使用相依性注入服務

  為了要能夠設計出一個容易維護的專案程式碼，因此需要套用 SRP 原則，因此，在這裡將會設計一個抽象介面與實作該介面的具體服務類別，將相關記事紀錄處理用到的程式碼分離到這個具體服務類別內，並且透過 ASP.NET Core 提供的相依性注入 Dependency Injection 服務，把所需要的具體服務類別執行個體，注入到 Blazor Component 元件內，如此，便可以在 Blazor 元件內使用這個具體服務類別物件，來進行 CRUD 的存取。透過這樣的設計技巧，可以讓這個專案具有高度鬆散耦合關係。
- 使用資料庫來儲存記事服務相關紀錄

  接下來將會使用 Entity Framework Core 套件，設計一個新的提供資料庫 (這裡將會使用 SQLite 資料庫做為操作範例) 具體服務類別；一旦完成這個資料庫存取的具體服務類別，只需要在 Startup 類別內 (在專案根目錄下，可以找到 Startup.cs 這個檔案)，註冊這個新建立的資料庫存取具體服務類別，這樣，在不修改原先 Blazor 元件的程式碼下，便可以直接讓這個原有 Blazor 專案，瞬間華麗變換成為不再使用記憶體作為存取目媒介，而是使用資料庫作為存取媒介的應用程式了。
- 在 Blazor 專案內呼叫 JavaScript 程式碼

在這一章中，將會展示如何透過 C# 來呼叫 JavaScript 的設計過程；這裡會先修改原先的 CRUD 元件中使用的對話窗功能，當要進行新增與修改的時候，將會透過 C# 來呼叫 jQuery 程式碼，讓這個對話窗顯示窗來；另外，將會使用已經包裝好的一個 Blazor NuGet 套件，該套件可以透過 C# 程式碼的呼叫，動態的顯示出一個對話窗，這裡將會針對刪除按鈕來進行重新設計，當使用點選刪除按鈕之後，將會透過 Blazored.Modal 元件來顯示一個對話窗，詢問使用者是否要刪除這筆紀錄，若回答是肯定的，將會立即刪除這筆紀錄。

- 在 Blazor 專案內設計 Web API 服務

  在這之前，已經完成了一個 CRUD 應用程式，並且會將資料記錄存取到資料庫內的功能，現在，將可以在 Blazor 專案內啟用 RESTful Web API 服務功能，讓這個網站也可以提供 Web API 的服務；所以，透過這章的開發練習，將會知道如何在 Blazor 專案內來設計出 RESTful Web API 的服務。

- 使用 Web API 來儲存記事服務相關紀錄

  最後，將會需要新設計一個存取 Web API 的具體服務類別，並且將這個服務註冊到 ASP.NET Core 內的 DI Container 容器中，當然，也無須修改原有 Blazor 專案內的元件設計程式碼，便可以讓這個 Blazor 專案，立即使用透過呼叫 Web API 服務，來進行記事紀錄之 CRUD 應用。

誰適合閱讀這本書

對於任何想要學習 Blazor 這個開發技術，可以透過本書的開發練習說明，逐步了解到如何真正的設計出一個 Blazor 實用專案。

讀者必須具備 .NET / C# 的開發經驗、HTML / CSS 的基本認識即可，對於開發工具而言，本書是在 Windows 10 作業系統下，使用任何 Visual Studio 2019 的版本，就可以進行開發。

練習專案原始碼

本電子書中的練習專案原始碼，可以從 https://github.com/vulcanlee/Blazor-Quick-Overview[1] 取得

[1] https://github.com/vulcanlee/Blazor-Quick-Overview

1. 建立 Blazor Server Side 伺服器端的專案

首先，先要來學習如何透過 Visual Studio 2019 開發工具來建立一個 Blazor 伺服器端的專案

> 這本書的所有內容將會是在 Windows 10 專業版 (版本 1909 / 組建 18363.535) 與 Visual Studio 2019 Enterprise (Version 16.4.2) 下所建立出來的，當然，任何 Visual Studio 2019 的版本皆可，不一定要使用 Enterprise 版本，Community / Professioal 版本也可以。
>
> 在安裝的時候，記得在 [工作負載] 頁次中，勾選 [ASP.NET 與網頁程式開發] 和 [.NET Core 跨平台開發]

Visual Studio 2019 Installer 之工作負載選擇

- 打開 Visual Studio 2019 開發工具
- 當 [Visual Studio 2019] 對話窗出現之後，點選右下方的 [建立新的專案] 按鈕

Visual Studio 建立新的專案按鈕

- 在 [建立新專案] 對話窗內，請找出 [Blazor 應用程式] 這個專案開發範本，並且點選這個專案開發範本
- 請點選右下角 [下一步] 按鈕

選擇使用 Blazor 應用程式專案範本

- 出現 [設定新的專案] 對話窗，輸入適當的 [專案名稱] 、[位置] ，完成後，請點選右下角 [建立] 按鈕

Blazor 應用程式設定新的專案

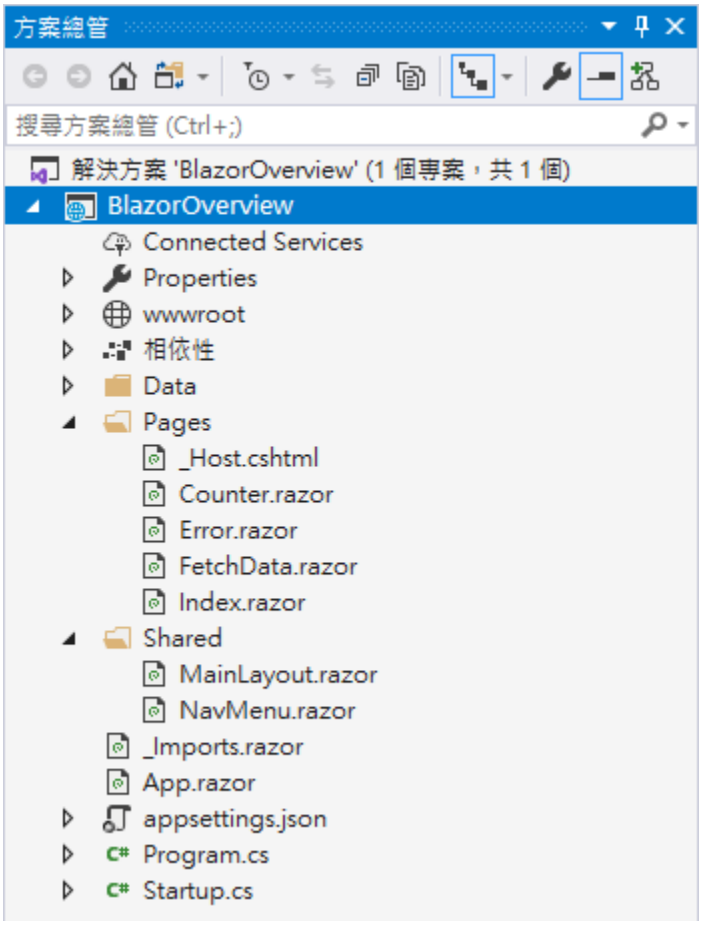
說明在這個範例程式碼中，將會建立一個 BlazorOverview 專案名稱

- 接下來將會看到 [建立新的 Blazor 應用程式] 對話窗，這裡可以根據當時開發專案需要，自行決定是否有調整 Blazor 專案的其他特性，不過，在這裡將不需要做任何額外的設定，請點選右下角的 [建立] 按鈕

Blazor 應用程式建立新的 Blazor 應用程式

注意事項在建立練習專案的時候，無須在 [建立新的 Blazor 應用程式] 對話窗下，點選該對話窗右上方的 [驗證] 設定選項，也就是，維持 [驗證] 選項為 [無驗證]

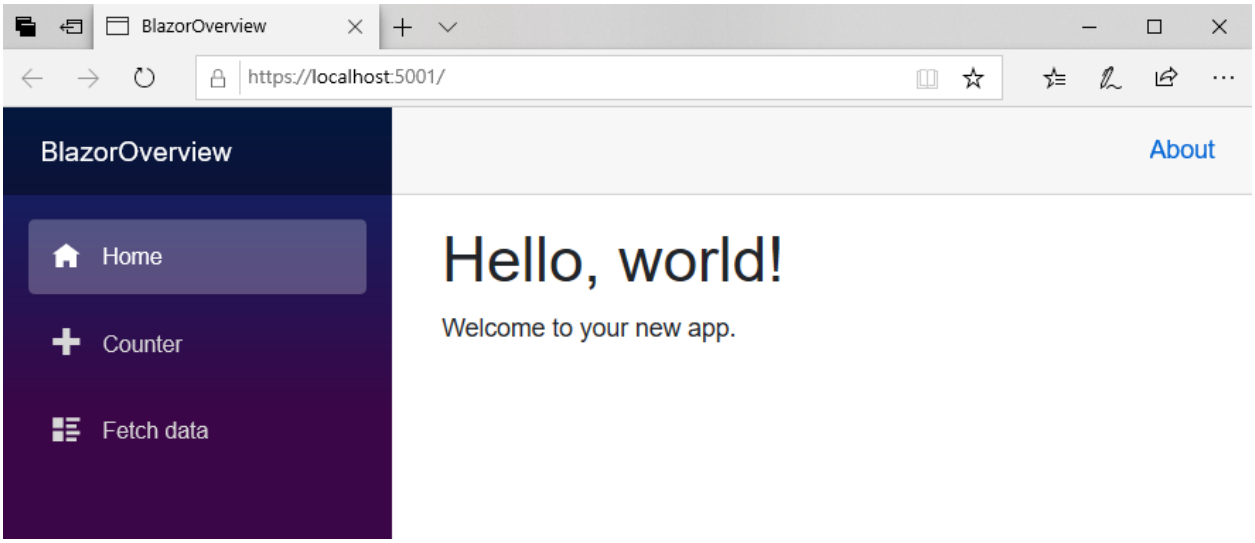
- 現在，這個 Blazor 專案已經建立完成
- 完成後的 Blazor 專案，將會有底下的方案結構



Blazor 方案結構

1.1 執行這個專案

- 請點選工具列上方的綠色三角形，或者按下 F5 ，開始執行這個 Blazor 專案
- 此時，將會在瀏覽器上出現底下畫面



Blazor 專案第一次執行結果

1.2 結論

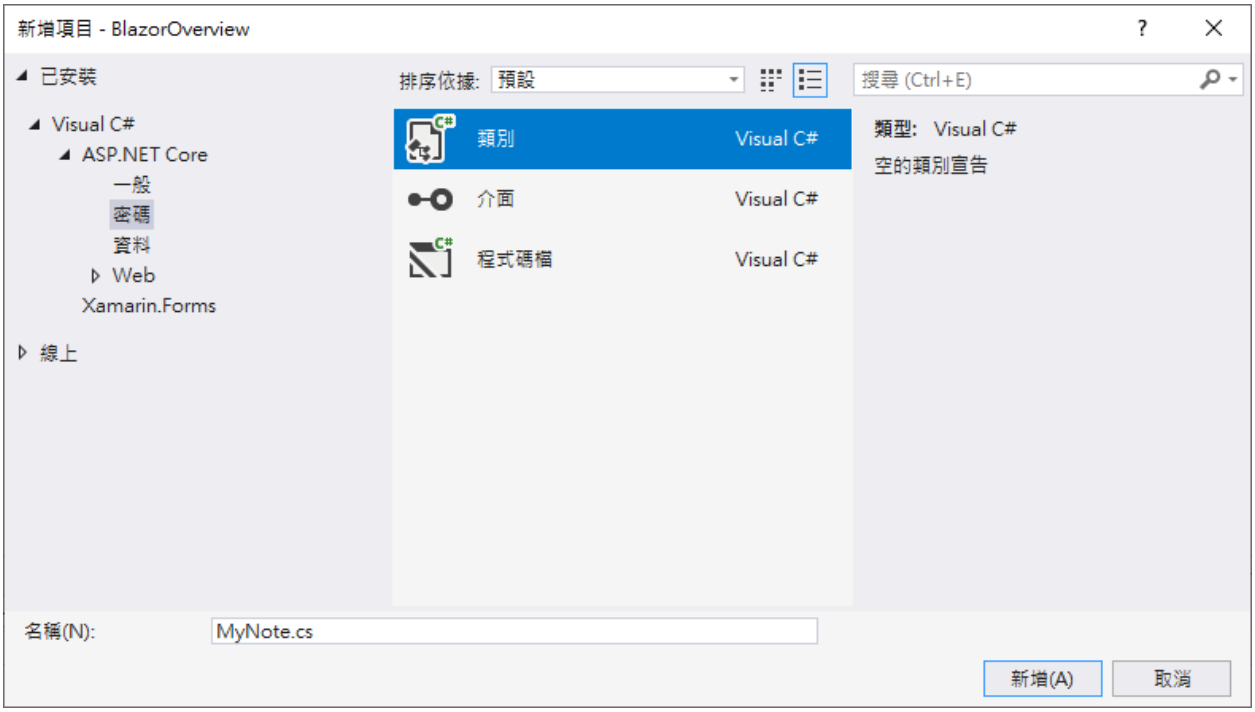
現在已經完成建立一個 Blazor 開發用的專案了

2. 開始設計 CRUD 的新增與刪除功能

現在要來先建立一個 Blazor 專案的 CRUD 之新增與修改功能，不過，在這裡的新增功能將不會有記事紀錄輸入畫面，而是自動產生一筆隨機記事紀錄。

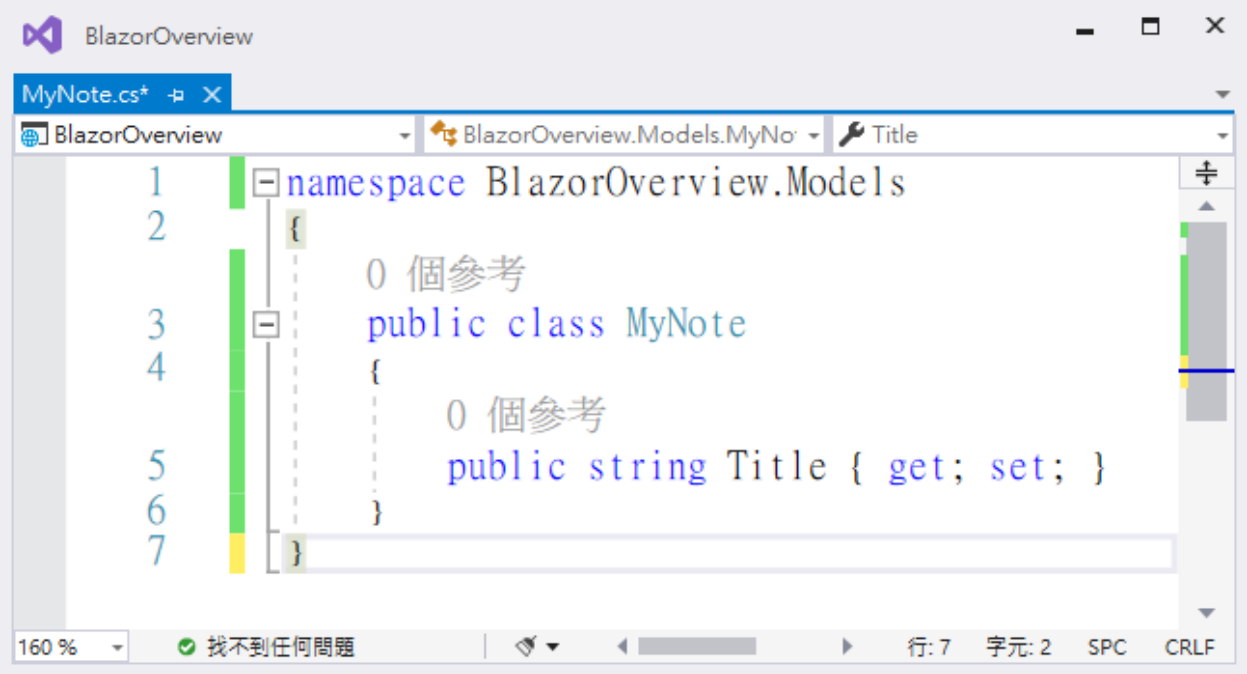
2.1 建立專案會用到的資料模型

- 滑鼠右擊專案節點
- 在彈出功能表點選 [加入] > [新增資料夾]
- 使用 `Models` 作為該新資料夾的名稱
- 滑鼠右擊 [Models] 資料夾節點
- 在彈出功能表點選 [加入] > [類別]
- 出現 [新增項目 - BlazorOverview] 對話窗
- 請在 [名稱] 欄位，輸入 `MyNote.cs`
- 最後，請點選 [新增] 按鈕



建立 MyNote 資料模型類別

- 使用底下程式碼來替換到這個檔案內的所有內容



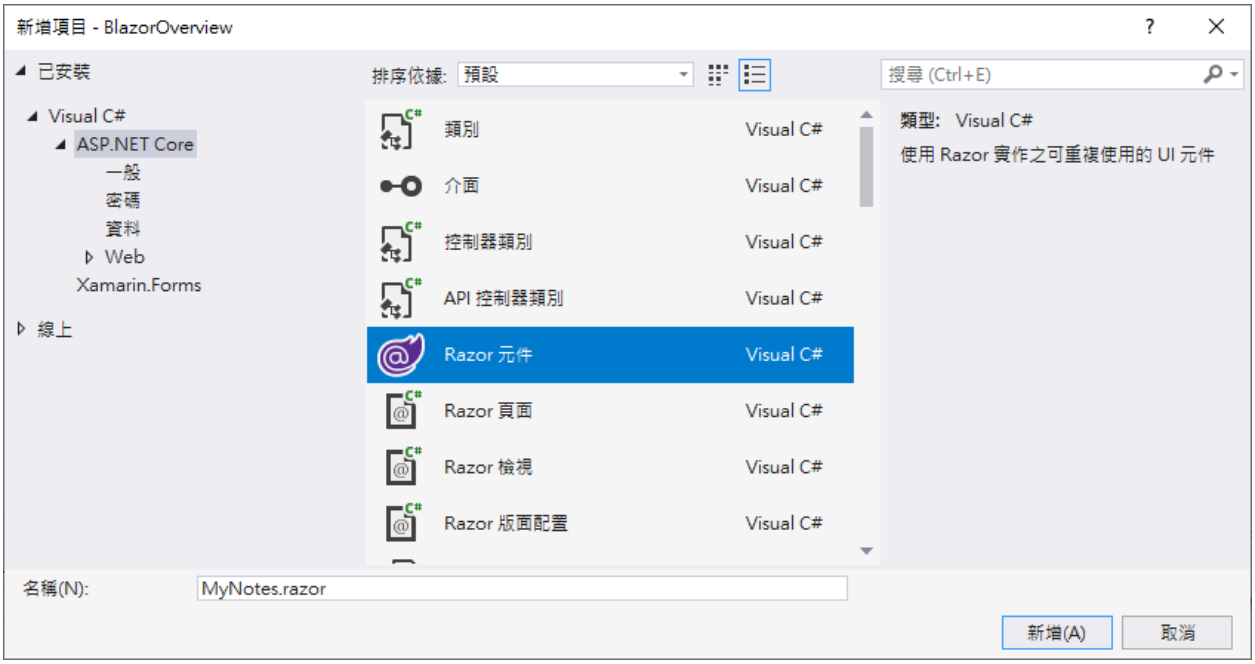
設計 MyNote 類別成員

```
namespace BlazorOverview.Models
{
   public class MyNote
   {
      public string Title { get; set; }
   }
}
```

2.2 建立與設計 Blazor 元件 - 顯示所有記事紀錄

- 滑鼠右擊 [Pages] 資料夾節點
- 在彈出功能表點選 [加入] > [新增項目]
- 出現 [新增項目 - BlazorOverview] 對話窗

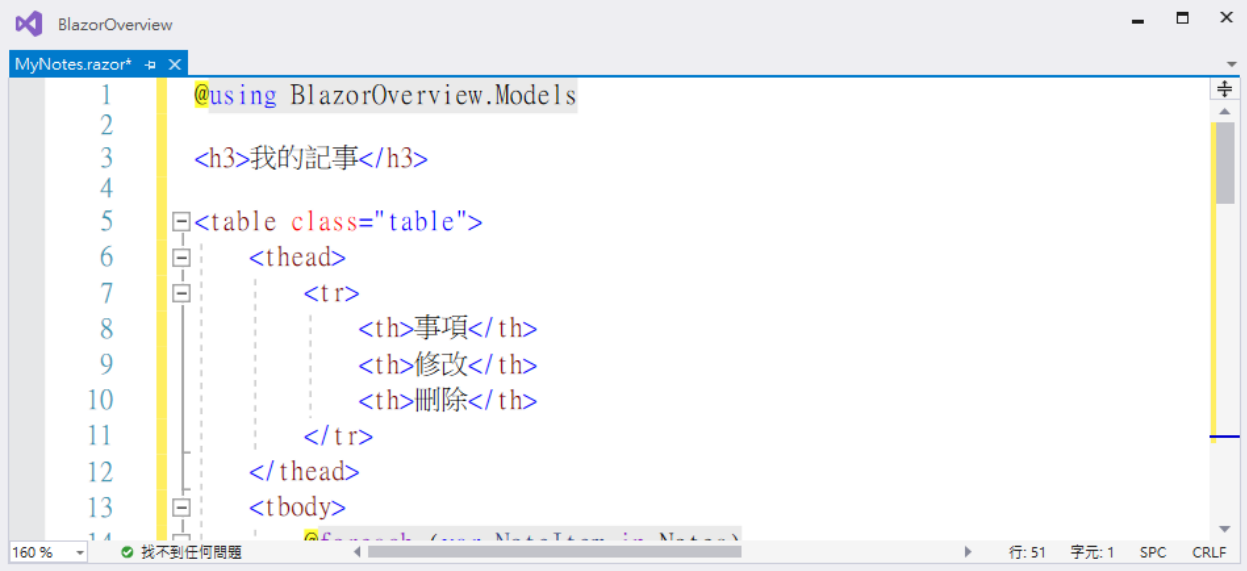
- 請確認該對話窗左方的清單位於 [已安裝] > [Visual C#] > [ASP.NET Core] 節點上
- 在該對話窗的中間區域，點選 [Blazor 元件] 這個選項
- 請在 [名稱] 欄位，輸入 `MyNotes.razor`
- 最後，請點選 [新增] 按鈕



建立一個 MyNotes 之 Blazor 元件

說明每一個 Blazor 元件都可以成為具有路由的實際網頁 URL，不過，在這裡的練習過程中，將會簡化這些設計內容，將整個 CRUD 應用程式，濃縮在一個 Blazor Component 元件上，也讓讀者可以體會到 Blazor 的一個很重要的特色，那就是 Blazor 的元件是可以做到重複使用的目的。

- 使用底下 Razor 程式碼來替換到這個檔案內的所有內容



MyNotes 元件的設計結果

```
@using BlazorOverview.Models

<h3> 我的記事 </h3>

@* 這裡是 HTML 的標記宣告 *@
<table class="table">
  <thead>
    <tr>
      <th> 事項 </th>
      <th> 修改 </th>
      <th> 刪除 </th>
    </tr>
  </thead>
  <tbody>
    @* 列出集合清單中的每一筆紀錄到 HTML Table 內 *@
    @foreach (var NoteItem in Notes)
    {
      <tr>
        @* 透過資料綁定，把集合清單內的紀錄屬性，顯示在網頁上 *@
        <td>@NoteItem.Title</td>
        @* 這個按鈕尚未進行任何設計 *@
        <td><input type="button" class="btn btn-primary" value=" 修改" /></td>
```

```
            <td>
                @* 透過 Blazor 的資料綁定，將刪除按鈕的點選事件，綁定到 C# 的委派處理方法 *@
                <input type="button" class="btn btn-danger" value=" 刪除"
                    @onclick="()=>Delete(NoteItem)" />
            </td>
        </tr>
    }
  </tbody>
</table>
<div>
  @* 透過 Blazor 的資料綁定，將新增按鈕的點選事件，*@
  @* 綁定到 C# 的委派處理方法 *@
  <input type="button" class="btn btn-primary" @onclick="Add" value=" 新增" />
</div>

@code {
  // 儲存要顯示的集合清單內的所有紀錄
  public List<MyNote> Notes { get; set; } = new List<MyNote>();

  // 元件建立的時候，所要執行的初始化工作
  protected override void OnInitialized()
  {
    // 預設建立的集合清單紀錄
    Notes = new List<MyNote>()
    {
      new MyNote { Title= " 買蘋果" },
      new MyNote { Title=" 買西瓜" }
    };
  }
  // 新增按鈕的點選事件之處理委派方法
  private void Add()
  {
    //加入一筆紀錄到集合清單內
    Notes.Add(new MyNote { Title = $" 新事項 {DateTime.Now.ToString()}" });
  }
  // 刪除按鈕的點選事件之處理委派方法
  private void Delete(MyNote note)
  {
    //從集合清單中刪除所選擇的紀錄
    Notes.Remove(note);
  }
}
```

提示

對於上述的程式碼與 HTML 標記，代表什麼意思，可以參考上面相關註解文字內容。

說明

每個 Blazor 元件就是一個 Razor 元件檔案，附檔案名稱為 .razor，其是由 HTML 標記與 C# 程式語言所組成；在上面的 MyNotes.razor 檔案，`@code{...}` 區塊為要設計的 C# 程式語言，其他的部分則是 HTML 宣告標記語言，不過，可以使用 `@` 符號，讓 HTML 標記宣告語言參雜 C# 程式語言在其中。

2.3 在 Blazor 專案首頁，加入此元件

- 在 [Pages] 資料夾中找到 [Index.razor] 這個檔案
- 打開這個檔案
- 使用底下 Razor 程式碼來替換到這個檔案內的所有內容

說明想要在 Blazor 專案內的元件，使用該專案內的其他元件，可以把每個元件都是為一個 HTML 標籤，在這裡僅加入 `<MyNotes />` 這個元件參考，讓這個剛剛設計的新 Blazor 元件，可以顯示在首頁上

```
@page "/"

<h1>Hello, world!</h1>

Welcome to your new app.

@* 使用 HTML Tag 標籤宣告方式，宣告這裡要顯示 MyNotes 這個 Blazor 元件 *@
<MyNotes />
```

提示

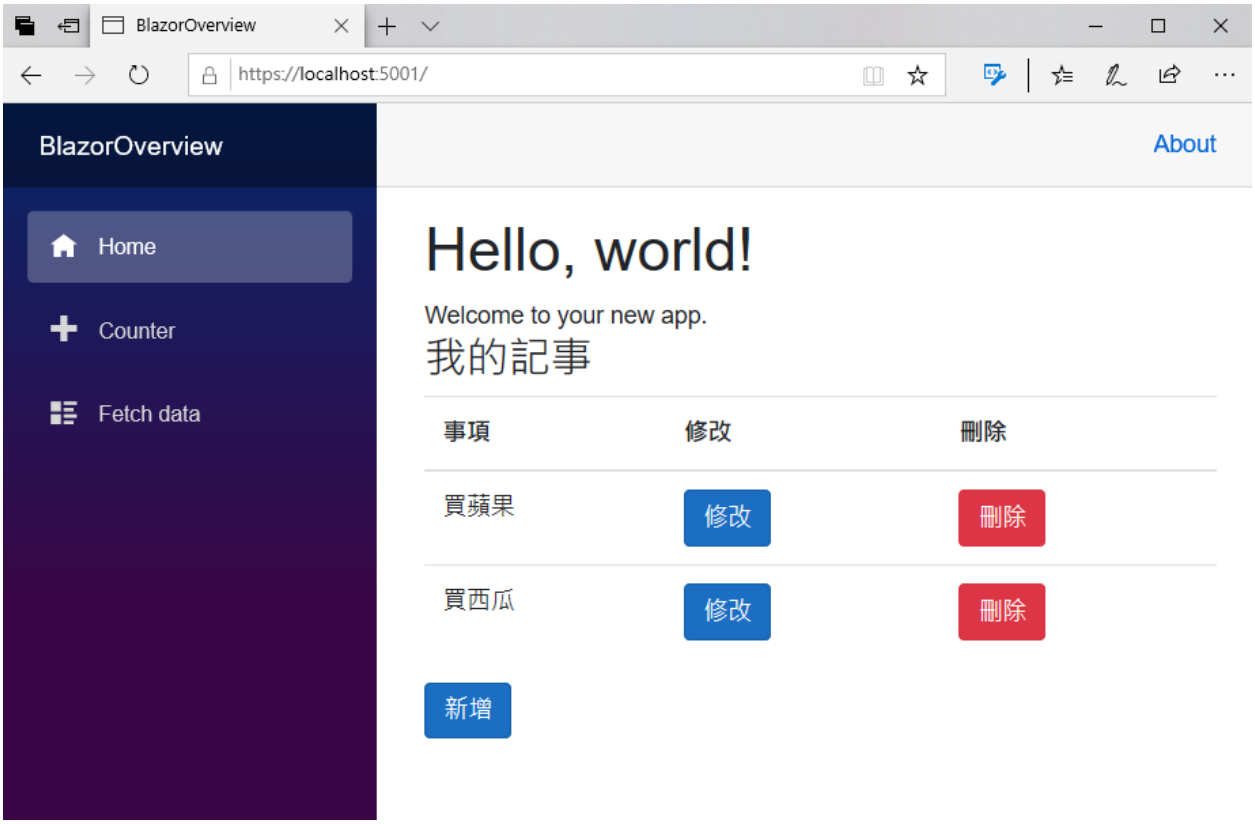
每個 Blazor 元件於設計完成之後，只要相對應的命名空間有宣告，便可以在 HTML 標記文件中，直接使用這個元件名稱作為一個 HTML 標記來使用，可謂相當的直覺與容易使用。

說明

透過這樣的練習設計，簡化了每次執行專案的時候，就可以立即在首頁上看到這次設計的 CRUD 應用的元件了。

2.4 執行這個專案

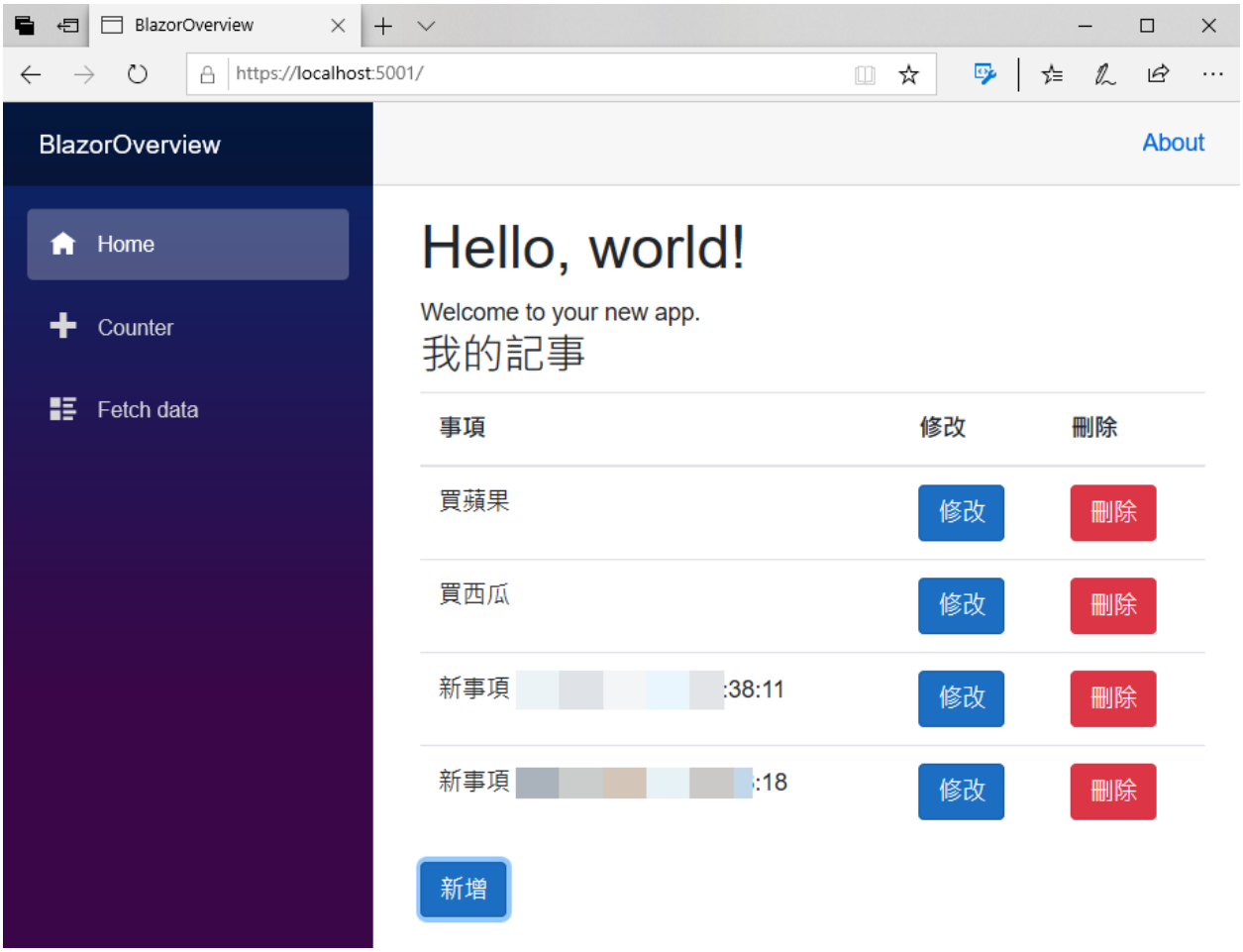
- 請點選工具列上方的綠色三角形，或者按下 F5 ，開始執行這個 Blazor 專案
- 此時，將會在瀏覽器上出現底下畫面



Blazor 專案具有新增與刪除執行結果

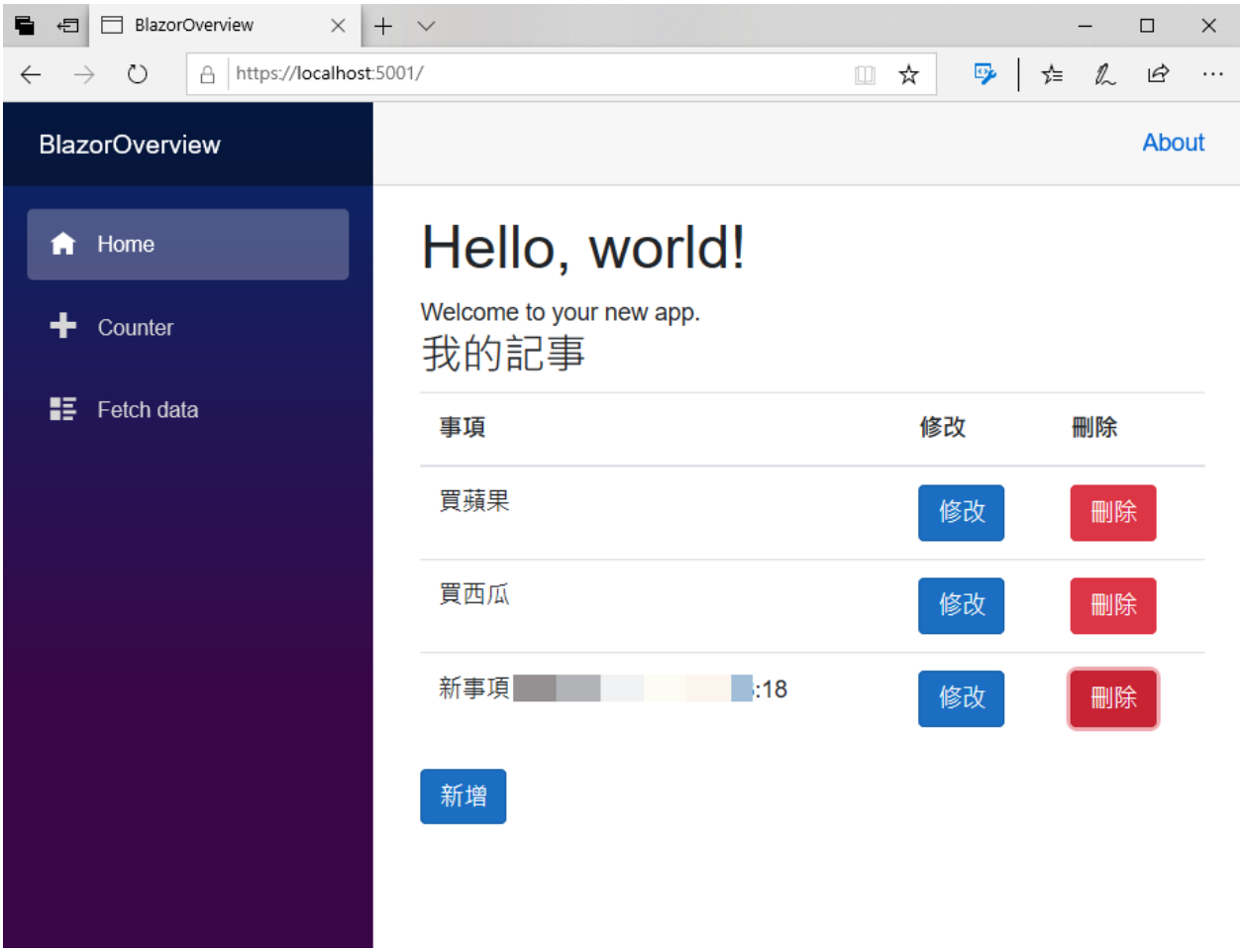
說明這裡會預先看到兩筆記事紀錄，這是因為在該 MyNotes.razor 元件建立的時候，透過 `OnInitialized` 方法呼叫，就已經預先產生兩筆記事紀錄了，不過，由於這些記事紀錄都是使用電腦記憶體作為儲存之用，因此，每次重啟這個專案的時候，這些記事紀錄都會消失不見。

- 點選 [新增] 按鈕兩次
- 此時將會自動加入兩筆紀錄到集合清單內
- 這裡是透過 Blazor 內建的資料綁定 Data Binding 機制，將會顯示在瀏覽器網頁上，如下面螢幕截圖



新增兩筆紀錄

- 點選到數第二筆紀錄的紅色 [刪除] 按鈕
- 此時，該筆紀錄將會從集合清單內刪除掉，並且即時更新在網頁上，如下面螢幕截圖



刪除其中一筆紀錄的執行結果

2.5 結論

現在已經完成具有新增與刪除 Blazor 專案了，不過，所有集合清單資料，都是儲存在記憶體內，一旦專案關閉結束並且重新執行，這些集合清單資料將會消失不見，這個問題將會在稍後，使用資料庫的方式來解決。

版權頁

Blazor Quick Overview Hands-On Lab 動手練習

檔案格式：EPUB3、PDF、MOBI

版本： 1.0
日期： 2020.01

作者： Vulcan Lee 李進興