MVC 框架

MVC 框架是一種常見於軟體開發中的設計模式，全名為「Model-View-Controller」(模型-視圖-控制器)。這種架構將應用程式劃分為三個主要部分，各自負責不同的職責，從而實現程式碼的解耦，提高維護性與擴展性。

1. Model (模型)

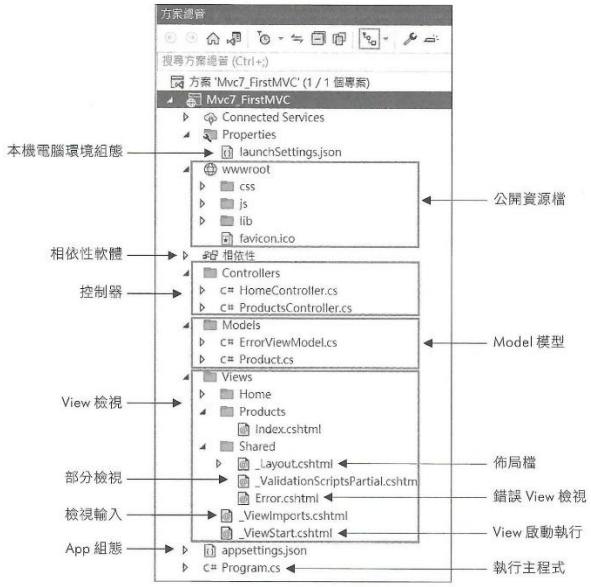
模型負責管理應用程式的數據和業務邏輯。舉例來說，在一個購物網站中，商品的資訊、庫存狀態、價格等都屬於模型的範疇。模型通常也會處理數據的存取與更新，並與資料庫或其他外部資源互動。

2. View (視圖)

視圖則專注於數據的顯示與呈現。也就是說，使用者在螢幕上看到的畫面 (如網頁、應用程式介面等)，都是由視圖負責根據模型的資料來顯示。視圖不直接處理數據，只負責如何將資訊以合適的方式展現給使用者。

3. Controller (控制器)

控制器扮演著協調者的角色，負責接收來自使用者的輸入 (例如點擊、填寫表單等)，並根據這些輸入去更新模型或改變視圖。控制器將使用者的操作轉化為對模型的指令，然後根據模型的變化更新視圖，從而實現互動。



範例 1：從一個簡單的程式開始-Hello world!

Controller: Controllers\HomeController.cs

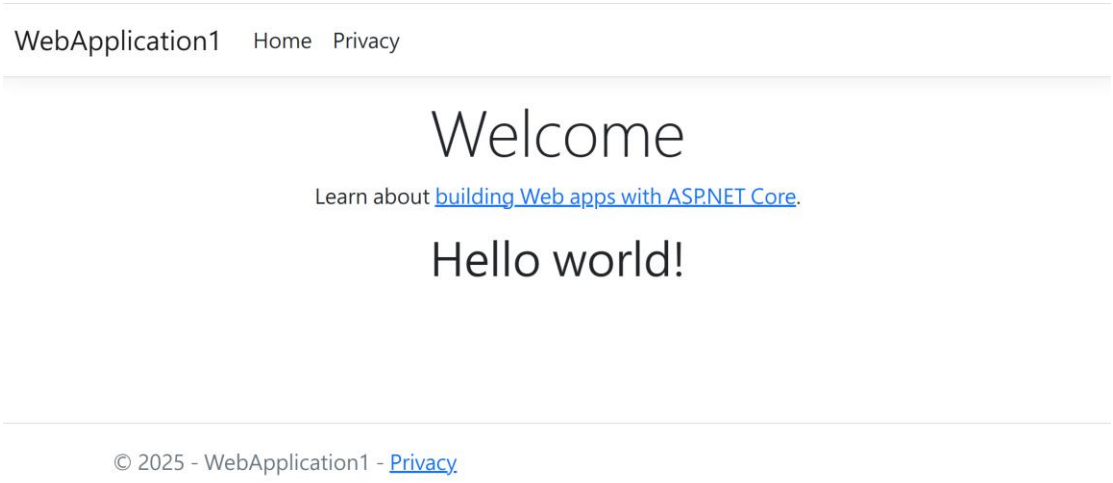
```
0 個參考
public IActionResult Index()
{
    ViewBag.Data = "Hello world!";
    return View();
}
```

View: Views\Home\Index.cshtml

```
1  @{
2      ViewData["Title"] = "Home Page";
3  }
4
5  <div class="text-center">
6      <h1 class="display-4">Welcome</h1>
7      <p>Learn about <a href="https://learn.microsoft.c
8      <h1>@ViewBag.Data</h1>
9  </div>
10
```

加入這一句

執行結果：



範例 2：Razor-if

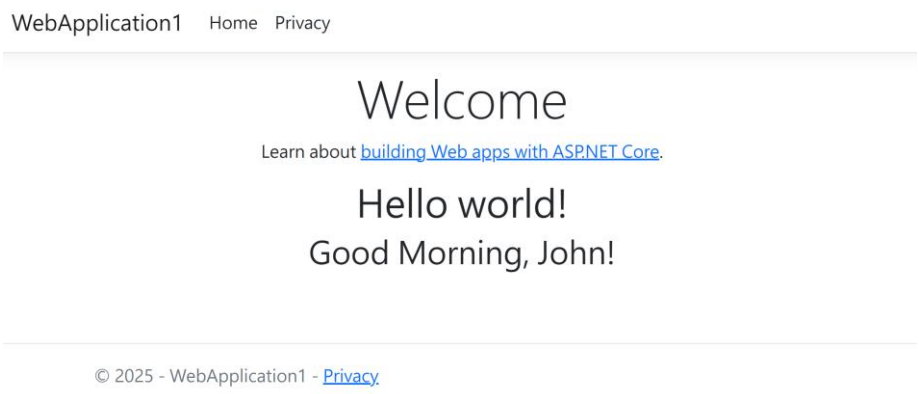
Controller: Controllers\HomeController.cs

```
public IActionResult Index()
{
    ViewBag.Data = "Hello world!";
    ViewBag.Name = "John";
    return View();
}
```

View: Views\Home\Index.cshtml

```
<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://learn.microsoft.com/a
    <h1>@ViewBag.Data</h1>
    @if (DateTime.Now.Hour < 12)
    {
        <h2>Good Morning, @ViewBag.Name!</h2>
    }
    else
    {
        <h2>Good Afternoon, @ViewBag.Name!</h2>
    }
</div>
```

執行結果：



範例 3：Razor-for

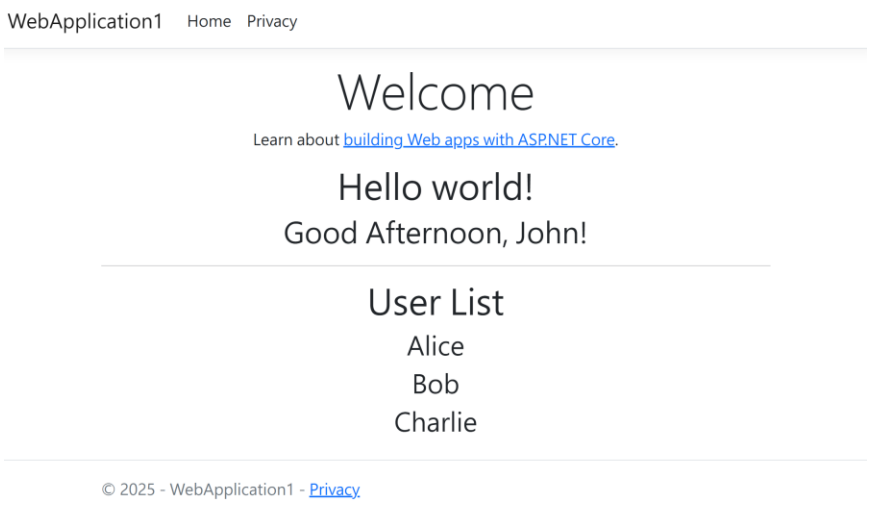
Controller: Controllers\HomeController.cs

```
0 個參考
public IActionResult Index()
{
    ViewBag.Data = "Hello world!";
    ViewBag.Name = "John";
    List<string> users = new List<string> { "Alice", "Bob", "Charlie" };
    ViewBag.Users = users;
    return View();
}
```

View: Views\Home\Index.cshtml

```
<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://learn.microsoft.com/as
    <h1>@ViewBag.Data</h1>
    @if (DateTime.Now.Hour < 12)
    {
        <h2>Good Morning, @ViewBag.Name!</h2>
    }
    else
    {
        <h2>Good Afternoon, @ViewBag.Name!</h2>
    }
    <hr />
    <h1>User List</h1>
    @foreach(var user in ViewBag.Users)
    {
        <h3>@user</h3>
    }
</div>
```

執行結果：



範例 4：回傳整個 Form

修改 **View: Views\shared_Layout.cshtml**

```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Employee">Employee</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </li>
    </ul>
</div>
```

Controller: Controllers\HomeController.cs

```
[HttpGet]
0 個參考
public IActionResult Employee()
{
    return View();
}
[HttpPost]
0 個參考
public IActionResult Employee(int id, string name, DateTime? birthdate, int salary)
{
    ViewBag.EmployeeData = new {
        Id = id,
        Name = name,
        Birthdate = birthdate,
        Salary = salary };
    return View("EmployeeResult");
}
```

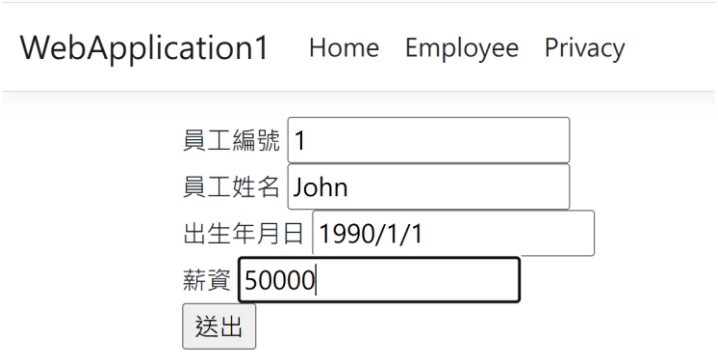
View: Views\Home\Employee.cshtml

```
@*
    For more information on enabling MVC for empty
*@
@{
}
<form method="post" asp-action="Employee">
    <label for="txtId">員工編號</label>
    <input id="txtId" name="id" />
    <br />
    <label for="txtName">員工姓名</label>
    <input id="txtName" name="name" />
    <br />
    <label for="txtBirthdate">出生年月日</label>
    <input id="txtBirthdate" name="birthdate" />
    <br />
    <label for="txtSalary">薪資</label>
    <input id="txtSalary" name="salary" />
    <br />
    <input type="submit" value="送出" />
</form>
```

```
@*
    For more information on enabling MVC for empty
*@
@{
    DateTime d = ViewBag.EmployeeData.Birthdate;
    string s = d.ToString("yyyy-MM-dd");
}
<h1>
    <h2>資料內容:</h2>
    <h3>員工編號:@ViewBag.EmployeeData.Id</h3>
    <h3>員工姓名:@ViewBag.EmployeeData.Name</h3>
    <h3>出生年月日:@s</h3>
    <h3>薪資:@ViewBag.EmployeeData.Salary</h3>
</h1>
```

View: Views\Home\EmployeeResult.cshtml

執行結果：



輸入各欄位資料 → 送出：

範例 5：合併成一個 View(EmployeeCombine.cshtml)

修改 Views:\shared_Layout.cshtml

```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Employee">Employee</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="EmployeeCombine">
                Employee(Combine)</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </li>
    </ul>
</div>
```

View: Views\Home\EmployeeCombine.cshtml

(注意: **asp-action="EmployeeCombine"**)

```
@*
    For more information on enabling MVC for empty projects, visit
*@

<form method="post" asp-action="EmployeeCombine">
    <label for="txtId">員工編號</label>
    <input id="txtId" name="id" />
    <br />
    <label for="txtName">員工姓名</label>
    <input id="txtName" name="name" />
    <br />
    <label for="txtBirthdate">出生年月日</label>
    <input id="txtBirthdate" name="birthdate" />
    <br />
    <label for="txtSalary">薪資</label>
    <input id="txtSalary" name="salary" />
    <br />
    <input type="submit" value="送出" />
</form>
@if(ViewBag.EmployeeData != null)
{
    <hr />

        DateTime d = ViewBag.EmployeeData.Birthdate;
        string s = d.ToString("yyyy-MM-dd");

    <h1>
        <h2>資料內容:</h2>
        <h3>員工編號:@ViewBag.EmployeeData.Id</h3>
        <h3>員工姓名:@ViewBag.EmployeeData.Name</h3>
        <h3>出生年月日:@s</h3>
        <h3>薪資:@ViewBag.EmployeeData.Salary</h3>
    </h1>
}
```

Controller: Controllers\HomeController.cs

```
[HttpGet]
0 個參考
public IActionResult EmployeeCombine()
{
    return View();
}
[HttpPost]
0 個參考
public IActionResult EmployeeCombine(int id, string name, DateTime? birthdate, int salary)
{
    ViewBag.EmployeeData = new
    {
        Id = id,
        Name = name,
        Birthdate = birthdate,
        Salary = salary
    };
    return View();
}
```

範例 6：ViewModel

Model: Models\EmployeeViewModel.cs

```
using System.ComponentModel.DataAnnotations;
namespace WebApplication1.Models;


3 個參考
public class EmployeeViewModel
{
    [Key]
    3 個參考
    public int Id { get; set; }

    [Required]
    [MaxLength(20)]
    3 個參考
    public string Name { get; set; }
    [Required]
    3 個參考
    public DateTime? Birthdate { get; set; }
    [Required]
    [Range(0, 200000)]
    3 個參考
    public int Salary { get; set; }
}
```

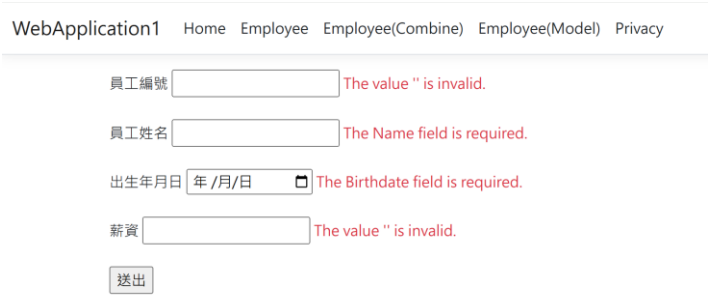
Controller: Controllers\HomeController.cs

```
[HttpGet]
0 個參考
public IActionResult EmployeeModel()
{
    return View();
}

[HttpPost]
0 個參考
public IActionResult EmployeeModel(EmployeeViewModel model)
{
    if (ModelState.IsValid)
        ViewBag.EmployeeData = model;
    else
        ViewBag.EmployeeData = null;

    return View();
}
```

View: Views\Home\EmployeeModel.cshtml

```
@model WebApplication1.Models.EmployeeViewModel

<form method="post" asp-action="EmployeeModel">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>

    <div>
        <label asp-for="Id">員工編號</label>
        <input asp-for="Id" />
        <span asp-validation-for="Id" class="text-danger"></span>
    </div>
    <br />

    <div>
        <label asp-for="Name">員工姓名</label>
        <input asp-for="Name" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <br />

    <div>
        <label asp-for="Birthdate">出生年月日</label>
        <input asp-for="Birthdate" type="date" />
        <span asp-validation-for="Birthdate" class="text-danger"></span>
    </div>
    <br />

    <div>
        <label asp-for="Salary">薪資</label>
        <input asp-for="Salary" />
        <span asp-validation-for="Salary" class="text-danger"></span>
    </div>
    <br />

    <input type="submit" value="送出" />
</form>

@if (ViewBag.EmployeeData != null)
{
    <hr />

    DateTime d = ViewBag.EmployeeData.Birthdate;
    string s = d.ToString("yyyy-MM-dd");

    <h1>
        <h2>資料內容:</h2>
        <h3>員工編號:@ViewBag.EmployeeData.Id</h3>
        <h3>員工姓名:@ViewBag.EmployeeData.Name</h3>
        <h3>出生年月日:@s</h3>
        <h3>薪資:@ViewBag.EmployeeData.Salary</h3>
    </h1>
}
```

執行結果：


WebApplication1
Home
Employee
Employee(Combine)
Employee(Model)
Privacy
員工編號
The value '' is invalid.
員工姓名
The Name field is required.
出生年月日
年 /月/日
The Birthdate field is required.
薪資
The value '' is invalid.
送出

檢查必填：


WebApplication1
Home
Employee
Employee(Combine)
Employee(Model)
Privacy
員工編號
1
員工姓名
John
出生年月日
1990/01/01
薪資
50000
送出
資料內容:
員工編號:1
員工姓名:John
出生年月日:1990-01-01
薪資:50000

範例 7：寫入檔案

步驟一、_layout.cshtml 上新增一個

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home"
            asp-action="EmployeeSave">Employee(Save) </a>
</li>
```

步驟二、HomeController.cs 新增/Home/EmployeeSave 這個 Action ([HttpGet], [HttpPost]兩個 Action(從 EmployeeModal 複製、貼上)

```
[HttpGet]
public IActionResult EmployeeSave()
{
    return View();
}
[HttpPost]
public IActionResult EmployeeSave(EmployeeViewModel model)
{
    if (ModelState.IsValid)
        ViewBag.EmployeeData = model;
    else
        ViewBag.EmployeeData = null;
    return View();
}
```

步驟三、新增 Views\Home\EmployeeSave.cshtml 檔案

步驟四、Copilot Chat 中輸入: 在 EmployeeSave 中將畫面上的文字 Id, Name, Birthdate, Salary 存入目前專案的 wwwroot\uploads\Employee.txt 檔案中, 如果檔案不存在, 建立新檔, 如果檔案已存在, 將檔案覆蓋掉

步驟五、Copilot Chat 中輸入: EmployeeSave 中, 從目前專案的 wwwroot\uploads\Employee.txt 中將資料讀到畫面上

步驟六、Copilot Chat 中輸入: 將資料直接寫到<input>中

Model: Models\EmployeeViewModel.cs(未改變)

```
using System.ComponentModel.DataAnnotations;
namespace WebApplication1.Models;


3 個參考
public class EmployeeViewModel
{
    [Key]
    3 個參考
    public int Id { get; set; }

    [Required]
    [MaxLength(20)]
    3 個參考
    public string Name { get; set; }
    [Required]
    3 個參考
    public DateTime? Birthdate { get; set; }
    [Required]
    [Range(0, 200000)]
    3 個參考
    public int Salary { get; set; }
}
```

Controller: Controllers\HomeController.cs

```
[HttpGet]
0 個參考
public IActionResult EmployeeSave()
{
    EmployeeViewModel model = new EmployeeViewModel();

    try
    {
        string uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "uploads");
        string filePath = uploadsFolder +  "\\Employee.txt";

        if (System.IO.File.Exists(filePath))
        {
            using (StreamReader rd = new StreamReader(filePath, System.Text.Encoding.UTF8))
            {
                model.Id = int.Parse(rd.ReadLine());
                model.Name = rd.ReadLine();
                model.Birthdate = DateTime.Parse(rd.ReadLine());
                model.Salary = int.Parse(rd.ReadLine());
                rd.Close();
            }
            ViewBag.ReadMessage = "已成功讀取檔案資料並載入表單";
        }
        else
        {
            ViewBag.FileContent = null;
            ViewBag.ReadMessage = "檔案尚未建立, 請先儲存資料";
        }
    }
    catch (Exception ex)
    {
        ViewBag.ReadMessage = $"讀取失敗: {ex.Message}";
        ViewBag.FileContent = null;
    }

    return View(model);
}
```

(續下頁)

```
[HttpPost]
0 個參考
public IActionResult EmployeeSave(EmployeeViewModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
            string uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "uploads");
            if (!Directory.Exists(uploadsFolder))
            {
                Directory.CreateDirectory(uploadsFolder);
            }
            string filePath = uploadsFolder +  "\\Employee.txt";
            StreamWriter wr = new StreamWriter(filePath, false, System.Text.Encoding.UTF8);
            wr.WriteLine(model.Id);
            wr.WriteLine(model.Name);
            wr.WriteLine(model.Birthdate);
            wr.WriteLine(model.Salary);
            wr.Close();
            ViewBag.EmployeeData = model;
            ViewBag.SaveMessage = "資料已成功儲存！";
        }
        catch (Exception ex)
        {
            ViewBag.SaveMessage = $"儲存失敗: {ex.Message}";
            ViewBag.EmployeeData = model;
        }
    }
    else
    {
        ViewBag.EmployeeData = null;
    }

    return View();
}
```

View: Views\Home\EmployeeSave.cshtml

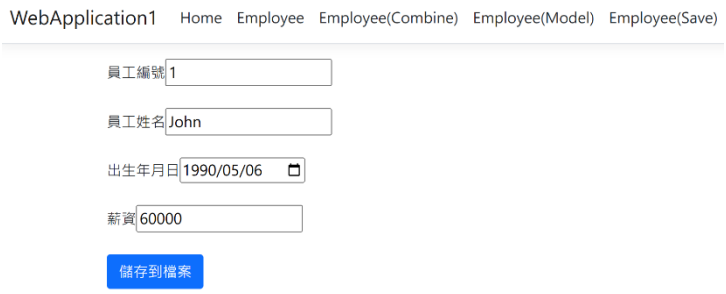
```
@model WebApplication1.Models.EmployeeViewModel
@{
    ViewData["Title"] = "員工資料儲存";
}
<form method="post" asp-action="EmployeeSave">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>

    <div>
        <label asp-for="Id">員工編號</label><input asp-for="Id" />
        <span asp-validation-for="Id" class="text-danger"></span>
    </div>
    <br />
    <div>
        <label asp-for="Name">員工姓名</label><input asp-for="Name" />
        <span asp-validation-for="Name" class="text-danger"></span>
    </div>
    <br />
    <div>
        <label asp-for="Birthdate">出生年月日</label><input asp-for="Birthdate" type="date" />
        <span asp-validation-for="Birthdate" class="text-danger"></span>
    </div>
    <br />

    <div>
        <label asp-for="Salary">薪資</label><input asp-for="Salary" />
        <span asp-validation-for="Salary" class="text-danger"></span>
    </div>
    <br />

    <input type="submit" value="儲存到檔案" class="btn btn-primary" />
</form>
```

(續下頁)

```razor
@if (ViewBag.EmployeeData != null)
{
    <hr />
    <h2>已儲存的資料內容:</h2>
    <h3>員工編號: @ViewBag.EmployeeData.Id</h3>
    <h3>員工姓名: @ViewBag.EmployeeData.Name</h3>
    <h3>出生年月日: @ViewBag.EmployeeData.Birthdate?.ToString("yyyy-MM-dd")</h3>
    <h3>薪資: @ViewBag.EmployeeData.Salary</h3>
    <p class="text-success">檔案位置: wwwroot\uploads\Employee.txt</p>
}

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

dynamic Microsoft.AspNetCore.Mvc.Razor.RazorPageBase.ViewBag { get; }
Gets the dynamic view data dictionary.

執行結果：
先從 uploads\Employee.txt 將資料載到畫面上



修改文字後按「儲存到檔案」會將畫面資訊寫到 uploads\Employee.txt

範例 8：利用 chart.js 繪製折線圖

步驟一、_layout.cshtml 上新增一個

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home"
            asp-action="Chart">Chart</a>
</li>
```

步驟二、HomeController.cs 新增/Home/Chart 這個 Action

```
[HttpGet]
public IActionResult Chart()
{
    return View();
}
```

步驟三、新增 Views\Home\Chart.cshtml 檔案

步驟四、Copilot Chat 中輸入: 在 Chart.cshtml 上, 利用 chart.js 產生台北市 2024 年 1~12 月的雨量折線圖

步驟五、Copilot Chat 中輸入: 雨量的資料從後端將資料傳到 Chart.cshtml

Controller: Controllers\HomeController.cs

```
[HttpGet]
0 個參考
public IActionResult Chart()
{
    // 後端準備資料
    var months = new[] { "1月", "2月", "3月", "4月", "5月", "6月", "7月", "8月", "9月", "10月", "11月", "12月" };
    var rainfallMm = new[] { 83.5, 170.0, 180.0, 178.8, 234.6, 325.3, 247.1, 352.5, 284.4, 112.0, 84.8, 136.3 };

    // 序列化給前端使用
    ViewBag.MonthsJson = JsonSerializer.Serialize(months);
    ViewBag.RainfallJson = JsonSerializer.Serialize(rainfallMm);

    return View();
}
```

View: Views\Home\Chart.cshtml

```
@{
    ViewData["Title"] = "台北市2004年雨量統計";
}
<style>
    .chart-container {
        width: 90%;
        margin: 0 auto;
        padding: 20px;
        position: relative;
    }
    .chart-wrapper {
        position: relative;
        height: 60vh; /* 讓圖表在高度上也具響應式 */
        width: 100%;
    }
</style>
<div class="container-fluid mt-4">
    <h2 class="text-center mb-4">台北市 2004 年月雨量統計圖</h2>

    <div class="chart-container">
        <div class="chart-wrapper">
            <canvas id="rainfallChart"></canvas>
        </div>
    </div>
</div>
@section Scripts {
    <script src="https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.umd.min.js"></script>
    <script>
        // 從後端取得的資料
        const labels = @Html.Raw(ViewBag.MonthsJson ?? "[]");
        const rainfallData = @Html.Raw(ViewBag.RainfallJson ?? "[]");
        const ctx = document.getElementById('rainfallChart').getContext('2d');
        const rainfallChart = new Chart(ctx, {
            type: 'line',
            data: {
                labels: labels,
                datasets: [{
                    label: '雨量 (mm)',
                    data: rainfallData,
                    borderColor: 'rgb(75, 192, 192)',
                    backgroundColor: 'rgba(75, 192, 192, 0.2)',
                    borderWidth: 3,
                    tension: 0.4,
                    fill: true,
                    pointRadius: 5,
                    pointBackgroundColor: 'rgb(75, 192, 192)',
                    pointBorderColor: '#fff',
                    pointBorderWidth: 2,
                    pointHoverRadius: 7
                }]
            },
```

(續下頁)

```
            options: {
                responsive: true,
                maintainAspectRatio: false, // 使其隨容器寬高調整
                plugins: {
                    title: {
                        display: true,
                        text: '台北市 2004 年各月份雨量分布',
                        font: { size: 18, weight: 'bold' },
                        padding: 20
                    },
                    legend: {
                        display: true,
                        position: 'top',
                        labels: { font: { size: 14 } }
                    },
                    tooltip: {
                        backgroundColor: 'rgba(0, 0, 0, 0.8)',
                        padding: 12,
                        titleFont: { size: 14 },
                        bodyFont: { size: 13 },
                        callbacks: {
                            label: (ctx) => `${ctx.dataset.label}: ${ctx.parsed.y} mm`
                        }
                    }
                },
                scales: {
                    y: {
                        beginAtZero: true,
                        title: { display: true, text: '雨量 (mm)', font: { size: 14, weight: 'bold' } },
                        ticks: { font: { size: 12 } },
                        grid: { color: 'rgba(0, 0, 0, 0.1)' }
                    },
                    x: {
                        title: { display: true, text: '月份', font: { size: 14, weight: 'bold' } },
                        ticks: { font: { size: 12 } },
                        grid: { color: 'rgba(0, 0, 0, 0.1)' }
                    }
                }
            }
        });
        // 視窗大小變化時自動調整
        window.addEventListener('resize', () => rainfallChart.resize());
    </script>
}
```

執行結果：

範例 9：利用 chart.js 繪製折線圖(從檔案讀取)

步驟一、_layout.cshtml 上新增一個

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home"
            asp-action="ChartSave">ChartSave</a>
</li>
```

步驟二、HomeController.cs 新增/Home/ChartSave 這個 Action

```
[HttpGet]
public IActionResult ChartSave()
{
    return View();
}
```

步驟三、新增 Views\Home\ChartSave.cshtml 檔案

步驟四、將 Chart.cshtml 內容複製到 ChartSave.cshtml，將

```
public IActionResult Chart()
{
.....
}
```

內容複製到 public IActionResult ChartSave()

步驟五、Copilot Chat 中輸入：建立 RainDataViewModel，產生 1-12 月雨量的屬性，在 ChartSave.cshtml 建立對應的<input>提供輸入，根據輸入的值，存檔到專案路徑下 wwwroot\uploads，並且變更 ChartSave 畫面上的折線圖

執行結果：


Vlodel) Employee(Save) Chart ChartSave Privacy
輸入 1–12 月雨量（mm）並即時更新折線圖
1月
45
2月
180
3月
180
4月
178.8
5月
234.6
6月
255
7月
247.1
8月
352.5
9月
125
10月
112
11月
84.8
12月
136.3
儲存並更新圖表
台北市各月份雨量分布
雨量 (mm)
400
350
300
250
200
150
100
50
0
雨量 (mm)
1月
2月
3月
4月
5月
6月
7月
8月
9月
10月
11月
12月
月份

範例 10：從資料庫讀取資料到畫面上

步驟一、_layout.cshtml 上新增一個

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home"
            asp-action="EmployeeADO">Employee(ADO.Net)</a>
</li>
```

步驟二、HomeController.cs 新增/Home/EmployeeADO 這個 Action

```
[HttpGet]
public IActionResult EmployeeADO()
{
    return View();
}
```

步驟三、新增 Views\Home\EmployeeADO.cshtml 檔案

步驟四、進入 SQL Server

步驟五、建立資料庫: db01

T-SQL:

```
USE master;

IF EXISTS(SELECT * FROM sys.databases WHERE name='db01')
BEGIN
    ALTER DATABASE db01 SET SINGLE_USER WITH ROLLBACK IMMEDIATE
    DROP DATABASE db01;
END;
GO

CREATE DATABASE db01
GO

USE db01;

CREATE TABLE dbo.Employee
(
emp_id int IDENTITY(1, 1) PRIMARY KEY,
emp_name nvarchar(20),
birthdate date,
salary int
);
```

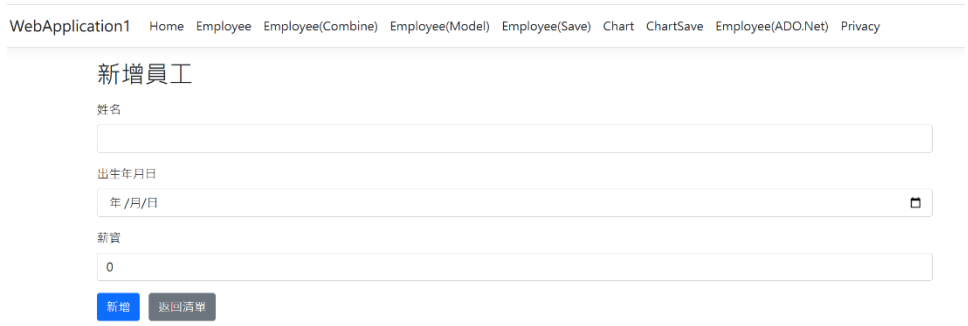
步驟六、建立連線字串，並存於 appsettings.Development.json
Copilot Chat 中輸入: 建立 ADO.Net 連線字串，連線到 SQL Server 2025, Server=.\SQL2025, Database=db01，信任連線，將連線字串存於 appsettings.Development.json

步驟七、EmployeeADO.cshtml 生成列表頁
Copilot Chat 中輸入: EmployeeADO.cshtml 生成畫面，並且由連線字串的 dbo.Employee 資料表中帶入資料，並且放置新增按鈕，在每一列上放置編輯、刪除按鈕，使用 EmployeeViewModel，資料庫 db01 的建置語法:

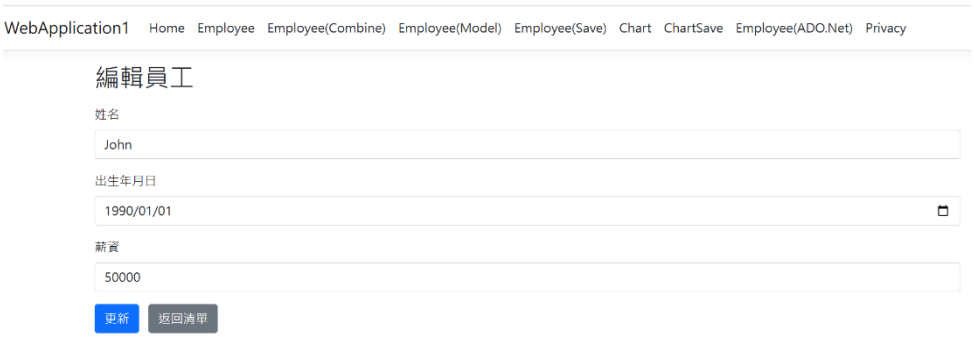
```
CREATE TABLE dbo.Employee
(
emp_id int IDENTITY(1, 1) PRIMARY KEY,
emp_name nvarchar(20),
birthdate date,
salary int
);
```

執行結果：

按下新增員工



編輯



範例 11：轉出 PDF 檔

Copilot Chat 中輸入: 在 EmployeeADO.cshtml 建立按鈕，轉出 PDF，將員工資料存成 PDF 檔，檔名為 Employee.pdf，存入目前專案的 wwwroot\uploads 資料夾，並且提供連結讓使用者下載

使用

```
using iTextSharp.text;
using iTextSharp.text.pdf;
```

範例 12：轉出 Excel 檔

Copilot Chat 中輸入: 在 EmployeeADO.cshtml 建立按鈕，轉出 Excel，將員工資料存成 PDF 檔，檔名為 Employee.xlsx，存入目前專案的 wwwroot\uploads 資料夾，並且提供連結讓使用者下載