

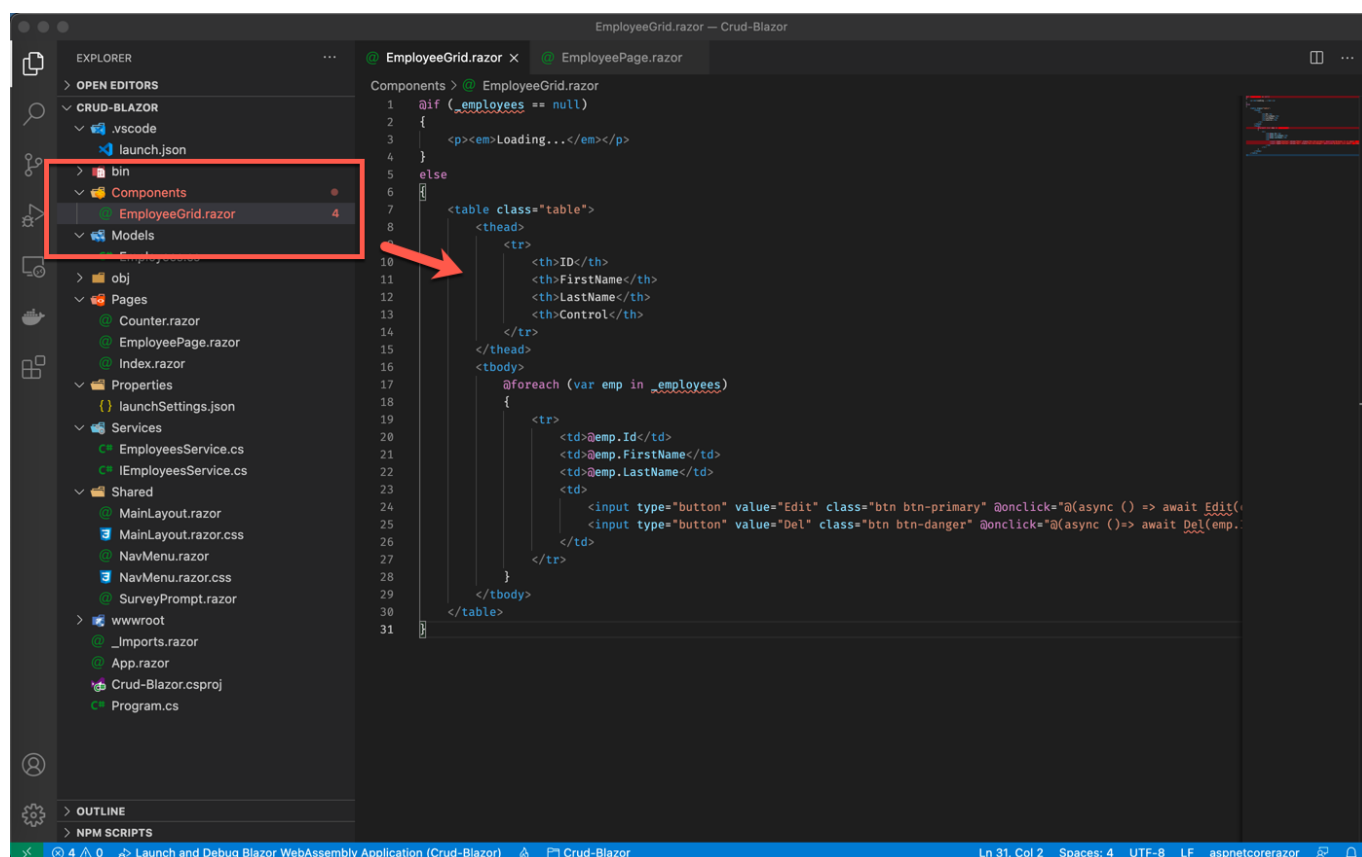
Component

這個章節，我們將學會如何自己寫 Component，並且在 Component 溝通。

建立 Component

這邊建立一個新的 Component 資料夾，並取名為 EmployeeGrid.razor，內容為剛剛的 Table 區段 (請剪下，貼上)。

如下圖。



然後，我們也要加一下 Code 如下

```
@code {
    [Parameter]
    public List<Employee> Employees { get; set; }

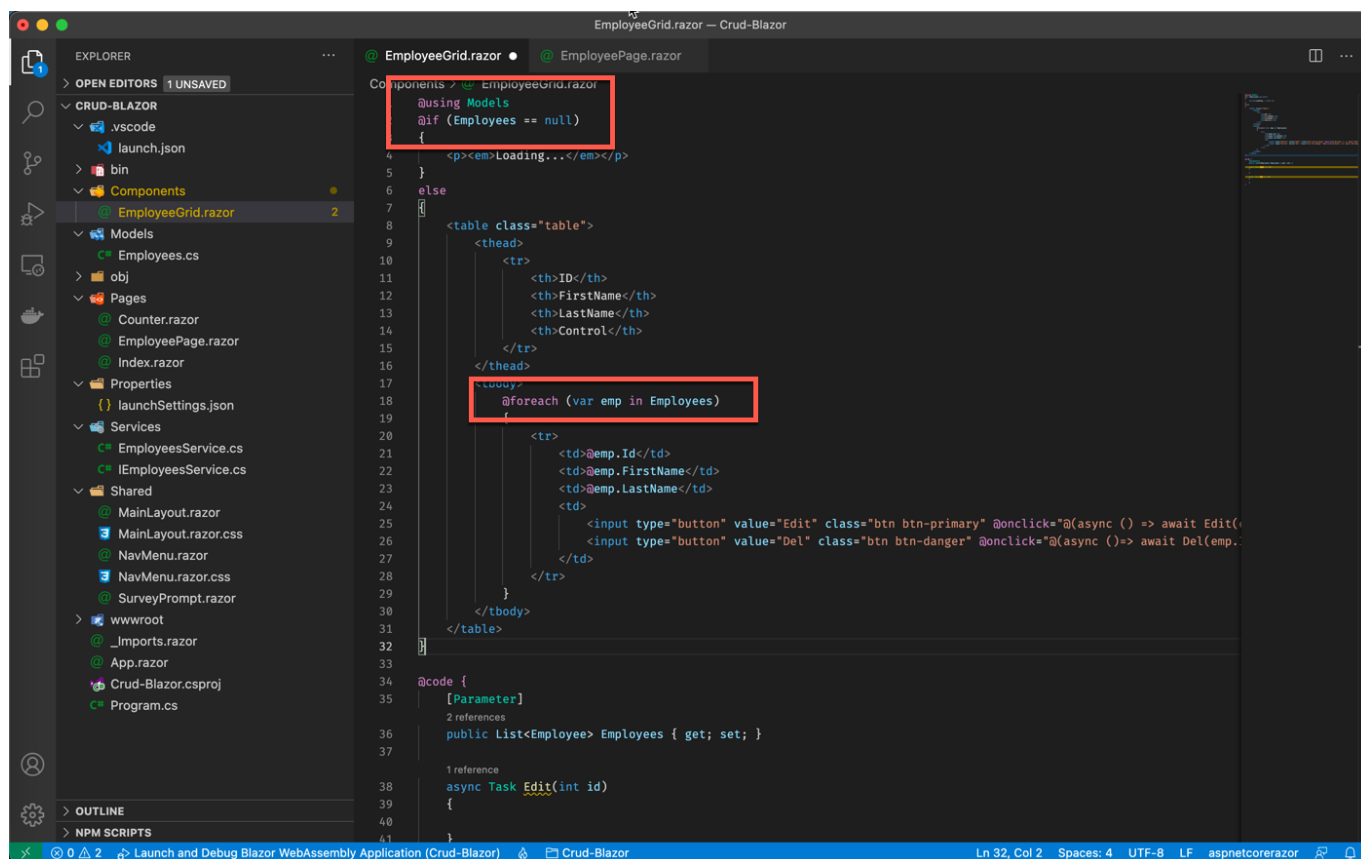
    async Task Edit(int id)
    {

    }

    async Task Del(int id)
    {

    }
}
```

然後，因為我們的變數從 `_employees` 改成 `Employees`，所以底下要調整一下。

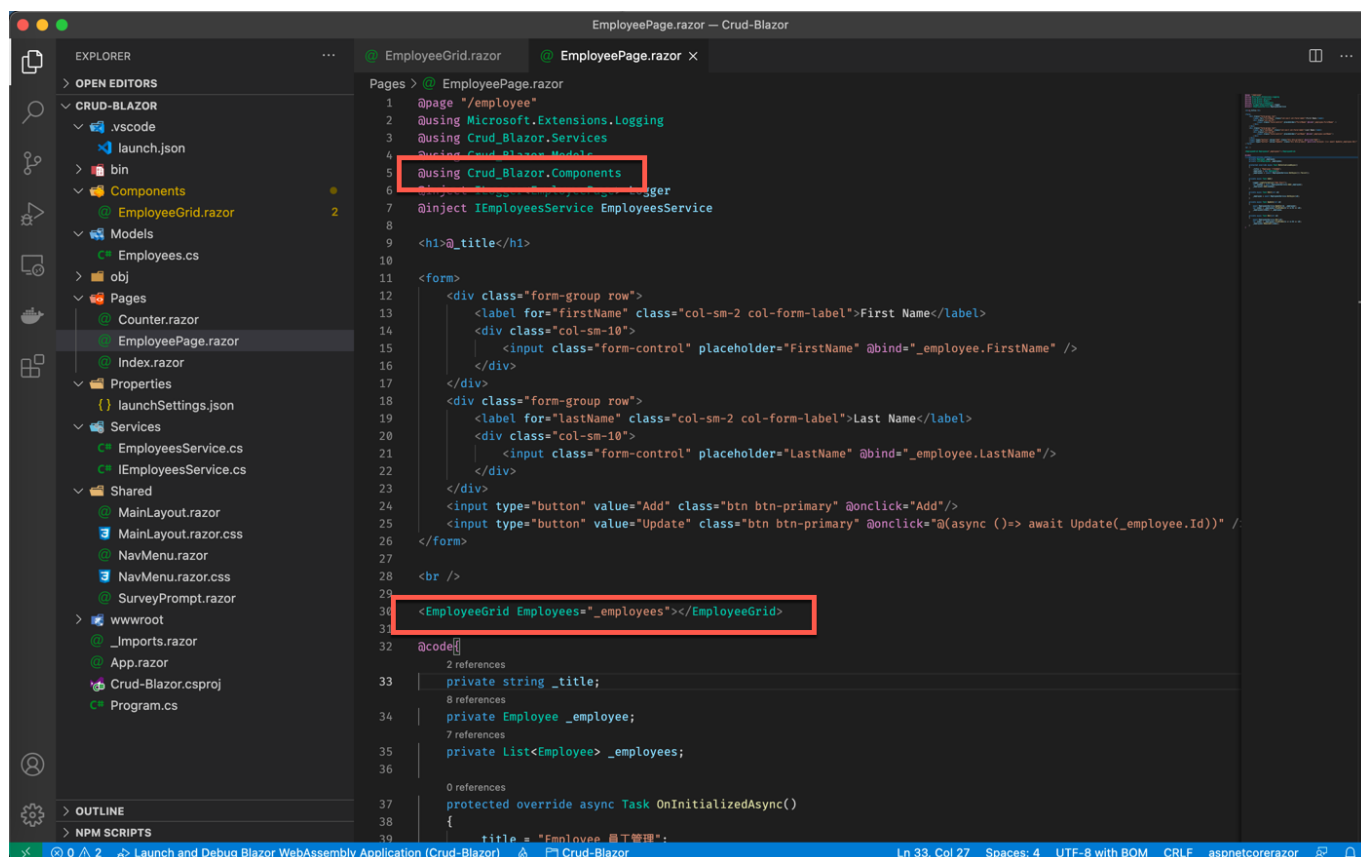


```
using Components;
@if (Employees == null)
{
    <p><em>Loading...</em></p>
}
else
{
    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>FirstName</th>
                <th>LastName</th>
                <th>Control</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var emp in Employees)
            {
                <tr>
                    <td>@emp.Id</td>
                    <td>@emp.FirstName</td>
                    <td>@emp.LastName</td>
                    <td>
                        <input type="button" value="Edit" class="btn btn-primary" @onclick="@() => await Edit(emp.Id)" />
                        <input type="button" value="Del" class="btn btn-danger" @onclick="@() => await Del(emp.Id)" />
                    </td>
                </tr>
            }
        </tbody>
    </table>
}

@code {
    [Parameter]
    public List<Employee> Employees { get; set; }

    async Task Edit(int id)
    {
    }
}
```

回到 `EmployeePage.razor`，並修改底下程式碼，我們將使用剛剛產生的元件。



```
@page "/employee"
@using Microsoft.Extensions.Logging
@using Crud.Blazor.Services
@using Crud.Blazor.Components
@inject ILogger<EmployeePage> Logger
@inject IEmployeeService EmployeesService

<h1>@_title</h1>

<form>
    <div class="form-group row">
        <label for="firstName" class="col-sm-2 col-form-label">First Name</label>
        <div class="col-sm-10">
            <input class="form-control" placeholder="First Name" @bind="_employee.FirstName" />
        </div>
    </div>
    <div class="form-group row">
        <label for="lastName" class="col-sm-2 col-form-label">Last Name</label>
        <div class="col-sm-10">
            <input class="form-control" placeholder="Last Name" @bind="_employee.LastName" />
        </div>
    </div>
    <input type="button" value="Add" class="btn btn-primary" @onclick="Add" />
    <input type="button" value="Update" class="btn btn-primary" @onclick="@() => await Update(_employee.Id)" />
</form>

<br />

<EmployeeGrid Employees="_employees"></EmployeeGrid>

@code {
    private string _title;
    private Employee _employee;
    private List<Employee> _employees;

    protected override async Task OnInitializedAsync()
    {
        _title = "Employee 員工管理";
    }
}
```

完成後，可以回到頁面，確定沒什麼問題。

The screenshot shows a web application titled "Employee 員工管理". On the left is a dark blue sidebar with the text "Crud-Blazor" at the top and navigation links for "Home", "Counter", and "Employee" (which is highlighted). The main content area has a light gray header with an "About" link. Below the header, there are two input fields: "First Name" (containing "FirstName") and "Last Name" (containing "LastName"). Below these are "Add" and "Update" buttons. A table below shows a list of employees with columns "ID", "FirstName", "LastName", and "Control". The first row has ID "3", FirstName "炭治郎", LastName "竈門", and two buttons "Edit" and "Del".

ID	FirstName	LastName	Control
3	炭治郎	竈門	<button>Edit</button> <button>Del</button>

增加 CallBack

原則上，我們不希望方法散落在各地，所以通常會在最外層實作真正的方法，而內層則是透過委派 (Callback) 的方式，呼叫外層的方法，所以，我們要在內層，也就是 EmployeeGrid.razor 準備 EventCallback 變數，來提供上層將方法傳遞進來。

我們增加底下的 Code。

```
@code {
    [Parameter]
    public List<Employee> Employees { get; set; }

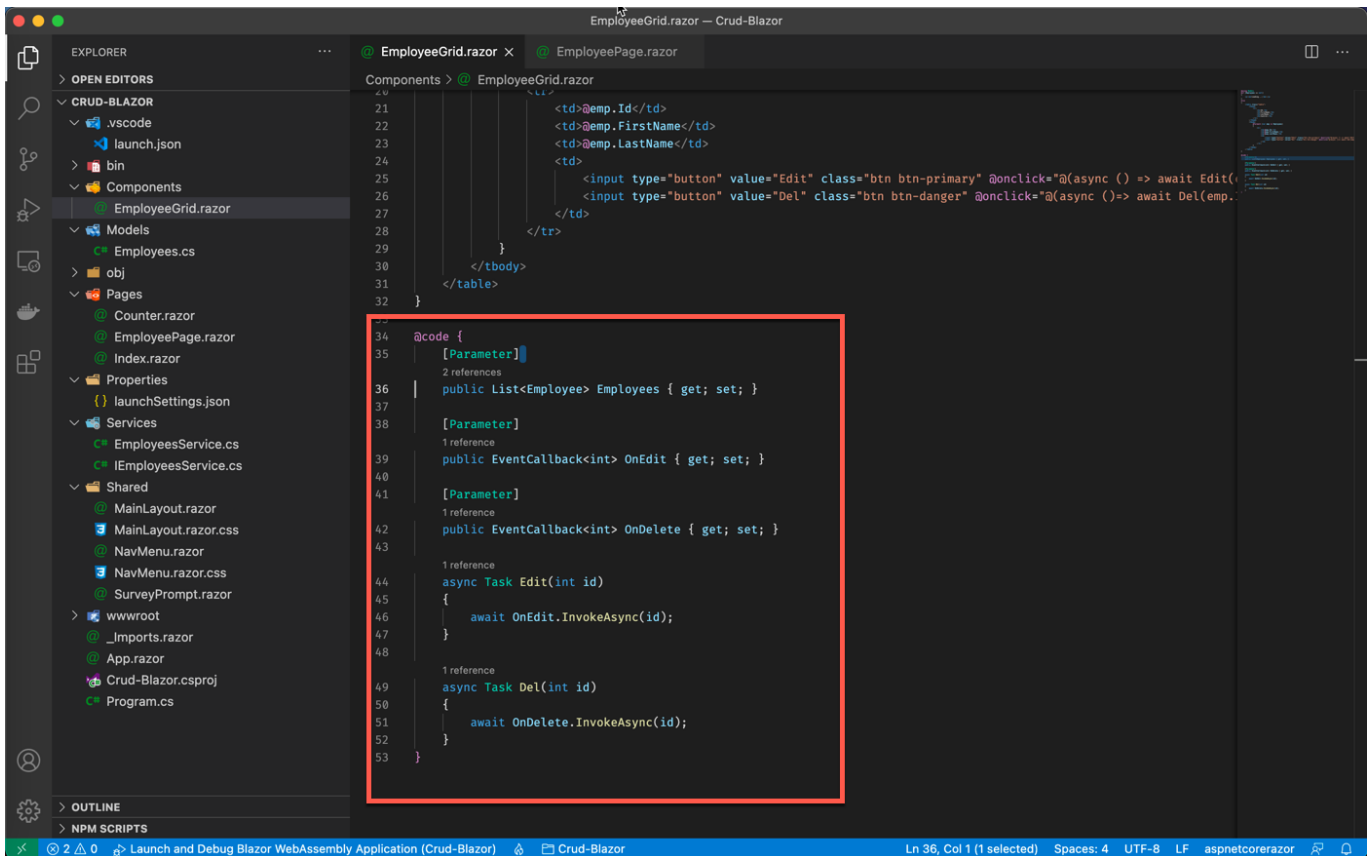
    [Parameter]
    public EventCallback<int> OnEdit { get; set; }

    [Parameter]
    public EventCallback<int> OnDelete { get; set; }

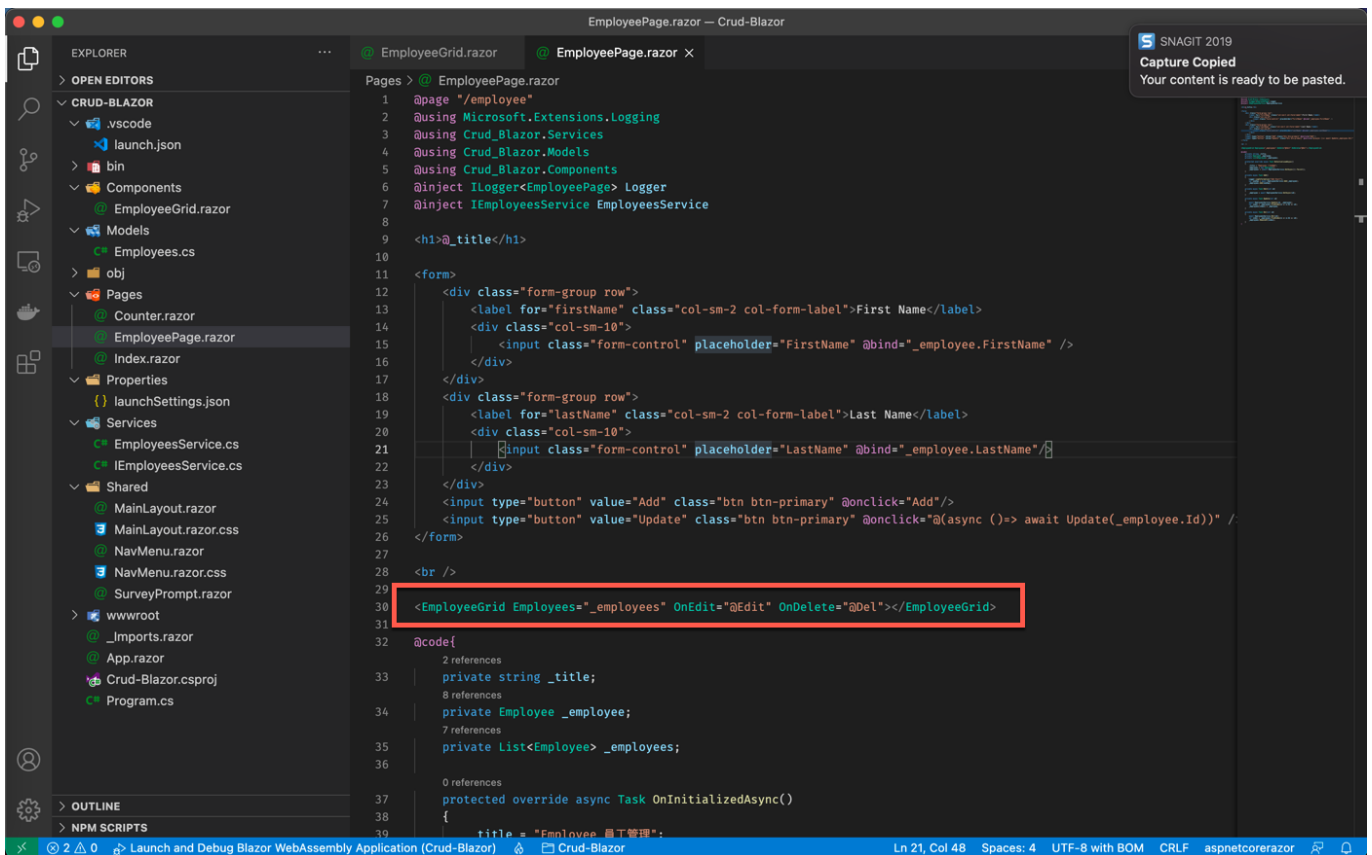
    async Task Edit(int id)
    {
        await OnEdit.InvokeAsync(id);
    }

    async Task Del(int id)
    {
        await OnDelete.InvokeAsync(id);
    }
}
```

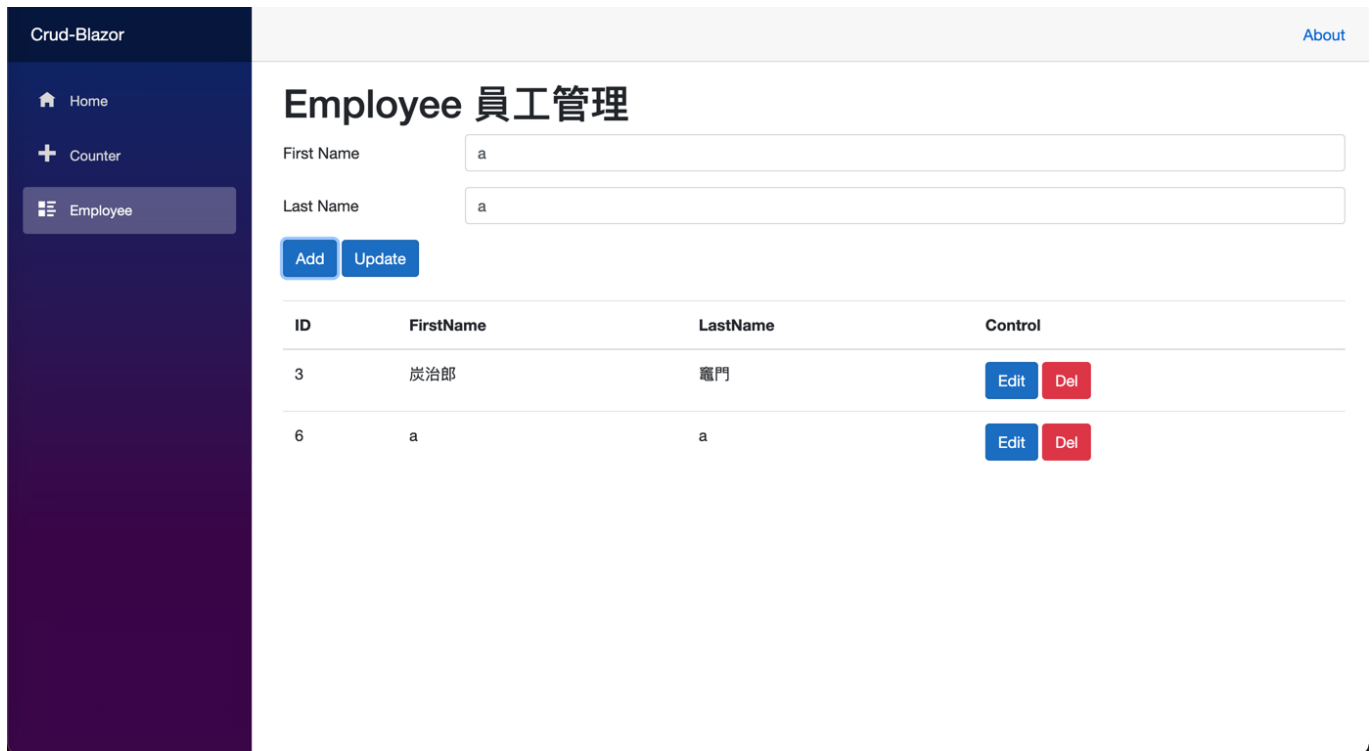
完成如下。



接著，回來修改 EmployeePage.razor，將原本的 edit 和 del 方法綁定進去，如下。



完成後測試。

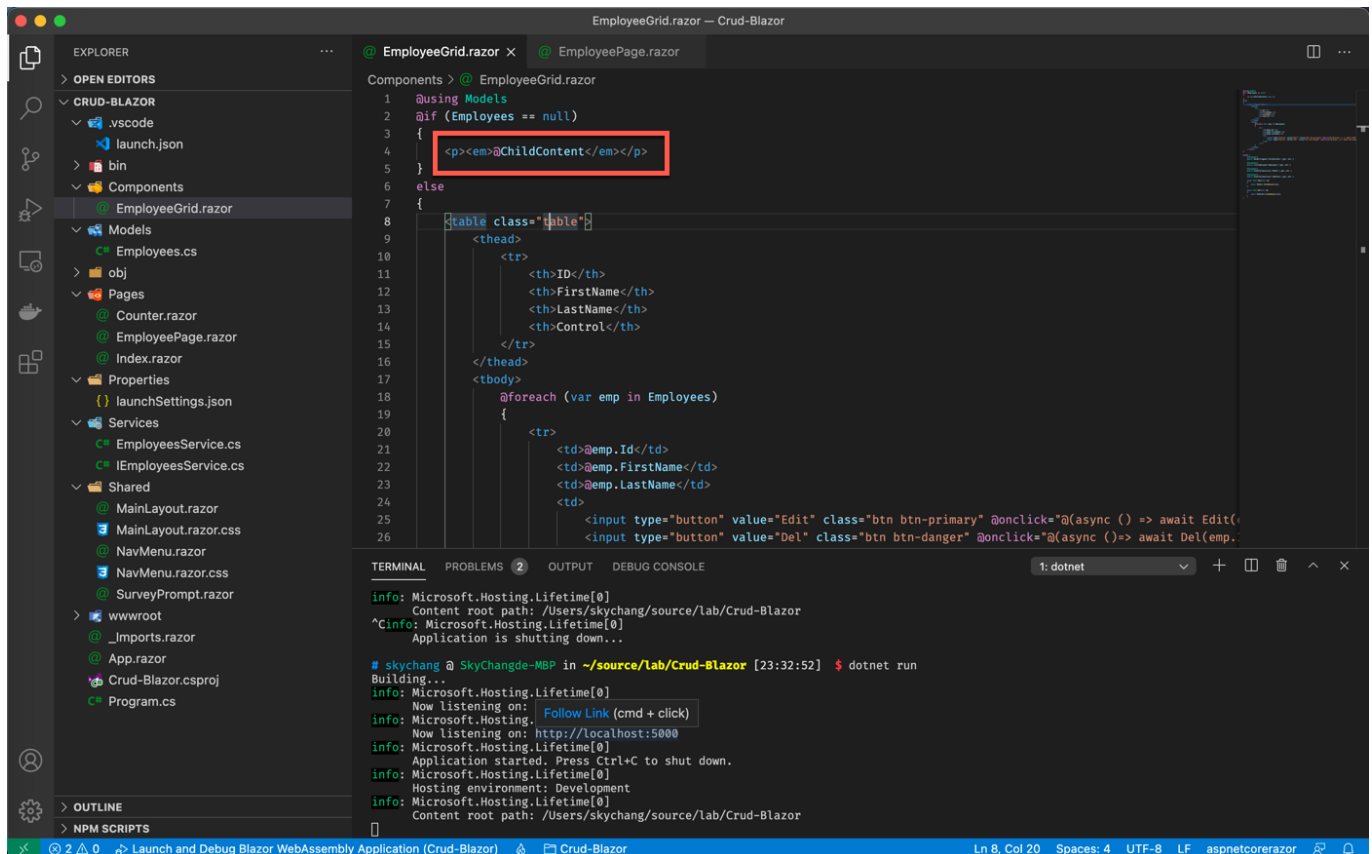


從父元件提供內容給子元件

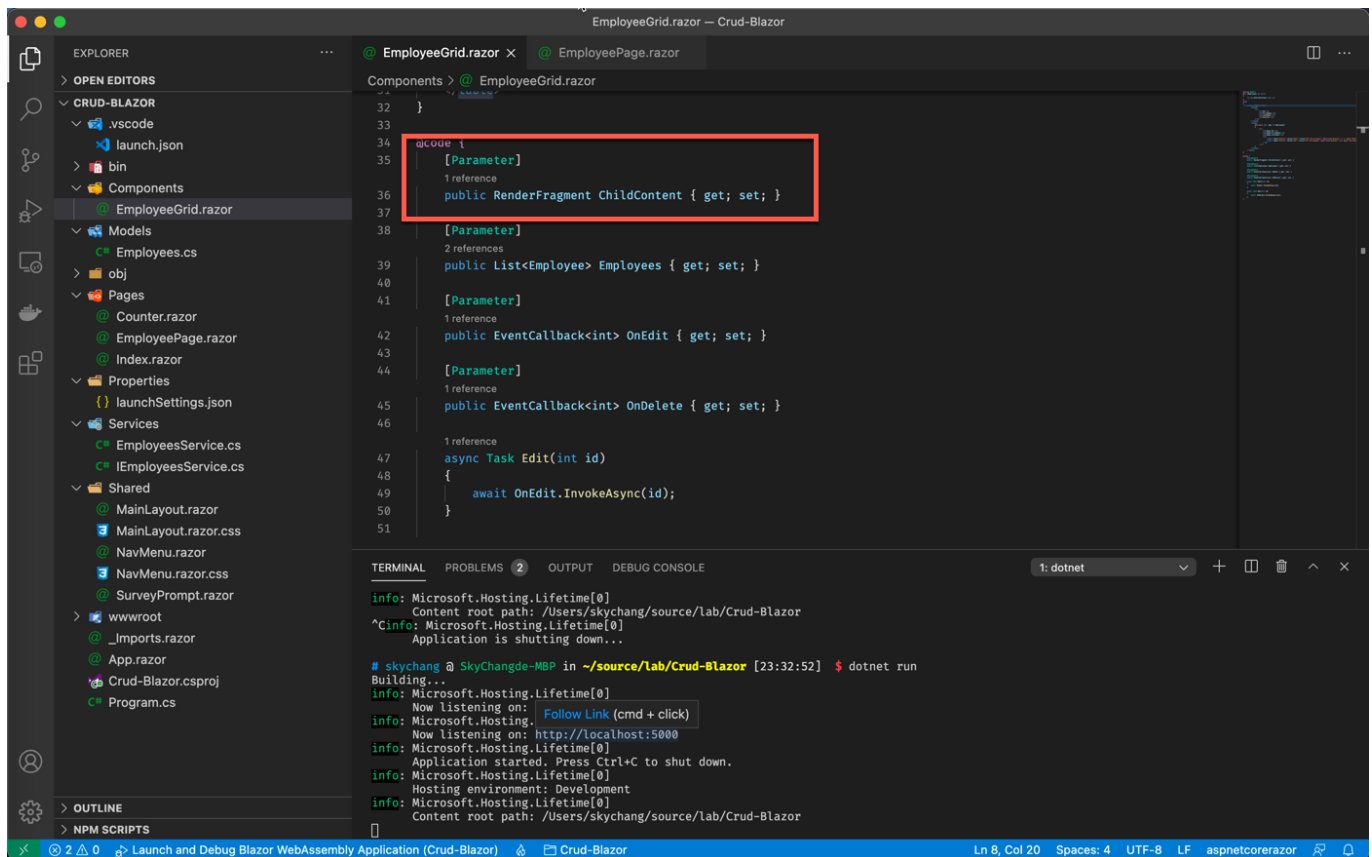
最後，我們希望能將子元件的內容，由父元件提供，例如，我們希望 Loading... 改成 Loading now... 而且這個 Loading now ... 是由父元件提供。

修改 EmployeeGrid.razor，將原本的 Loading...改成如下。

Note：請注意，變數名稱請使用此名稱。



因為 ChildContent 為變數，所以底下也要加上此變數



接著，回到 EmployeePage.razor，就可以調整成底下，將字串傳入進去。

