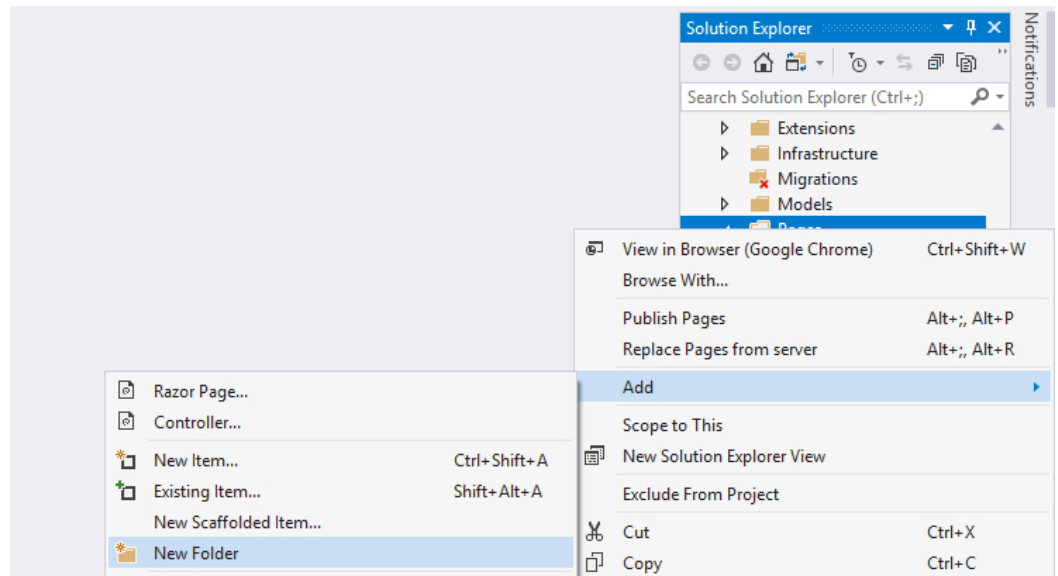


Easy CRUD Pages

While these Razor pages don't make use of MVC architecture and should not be used in a general MVC setting, they are a quick and easy way to generate lookup table maintenance screens. (See <https://stackoverflow.com/questions/50157934/what-is-difference-between-view-and-page-in-asp-net-core-2>)

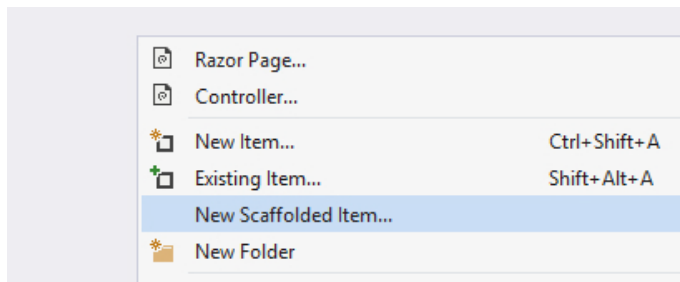
- 1) Choose a table to make CRUD pages for (example: PainScaleType).

- 2) Right-click on the Pages|PCA folder. Choose Add | New Folder. Name it the same as the table (ex:

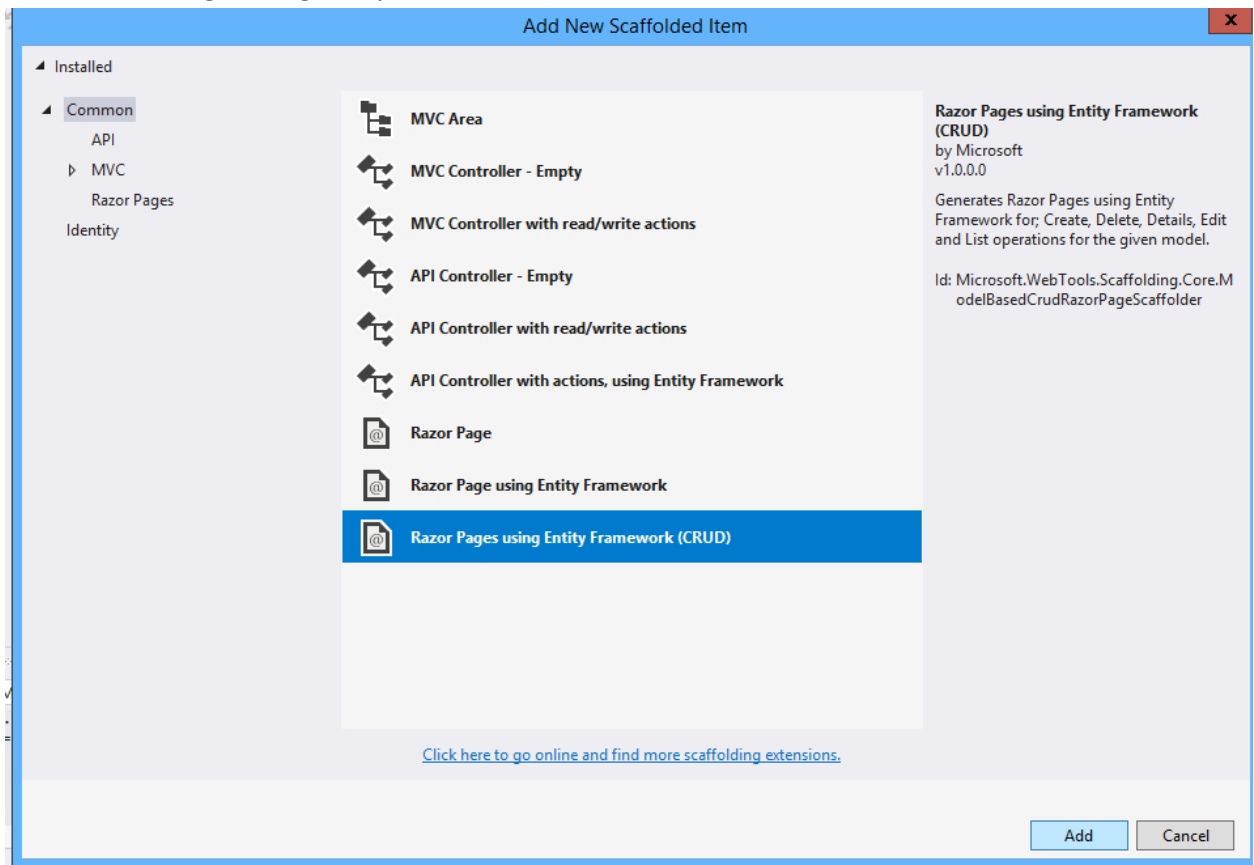


PainScaleType).

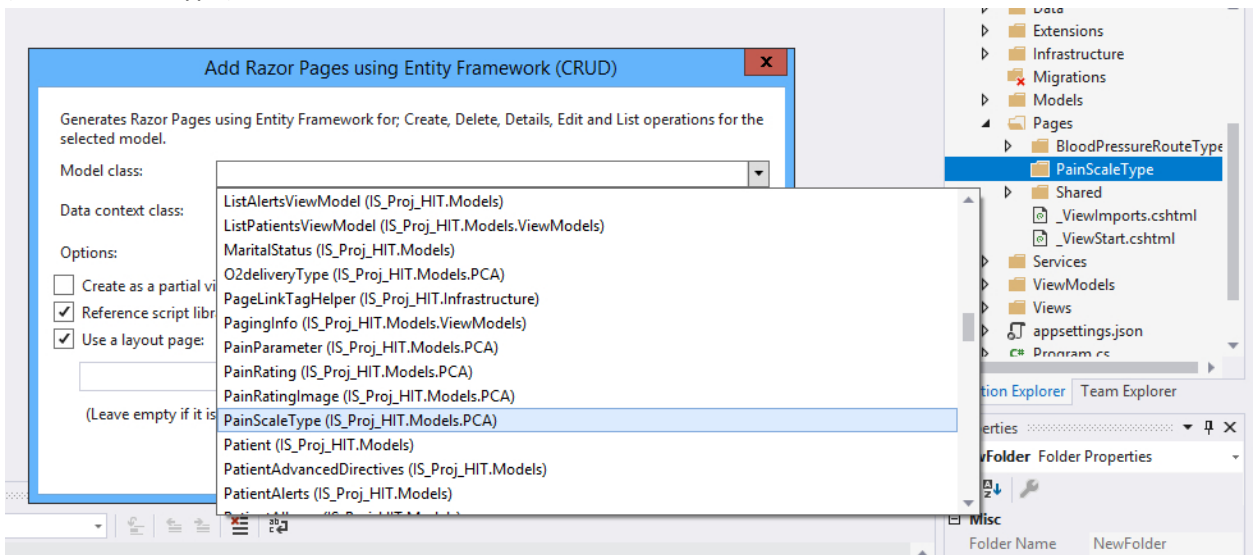
- 3) Right-click on the subfolder. Choose Add | New Scaffolded Item.



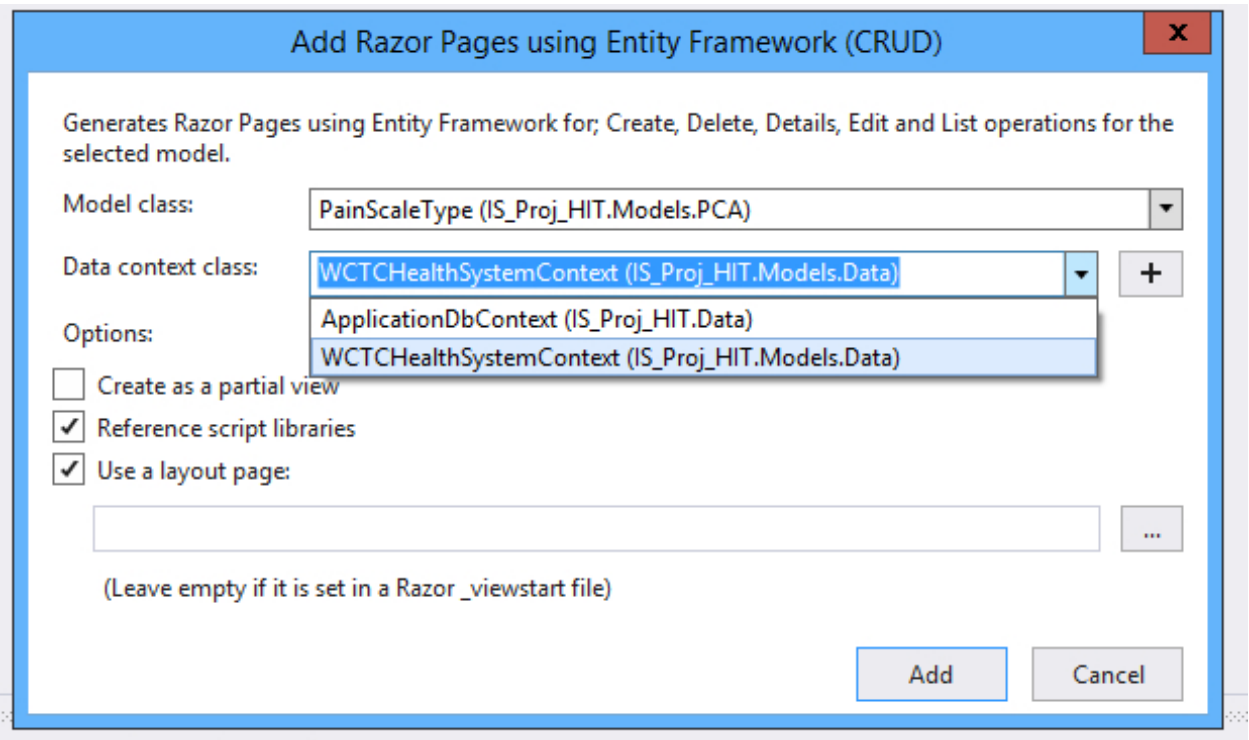
- 4) Choose Razor Pages using Entity Framework (CRUD) and click Add.



- 5) For Model Class open the dropdown. Choose the table that's the same name as the subfolder (ex: PainScaleType).

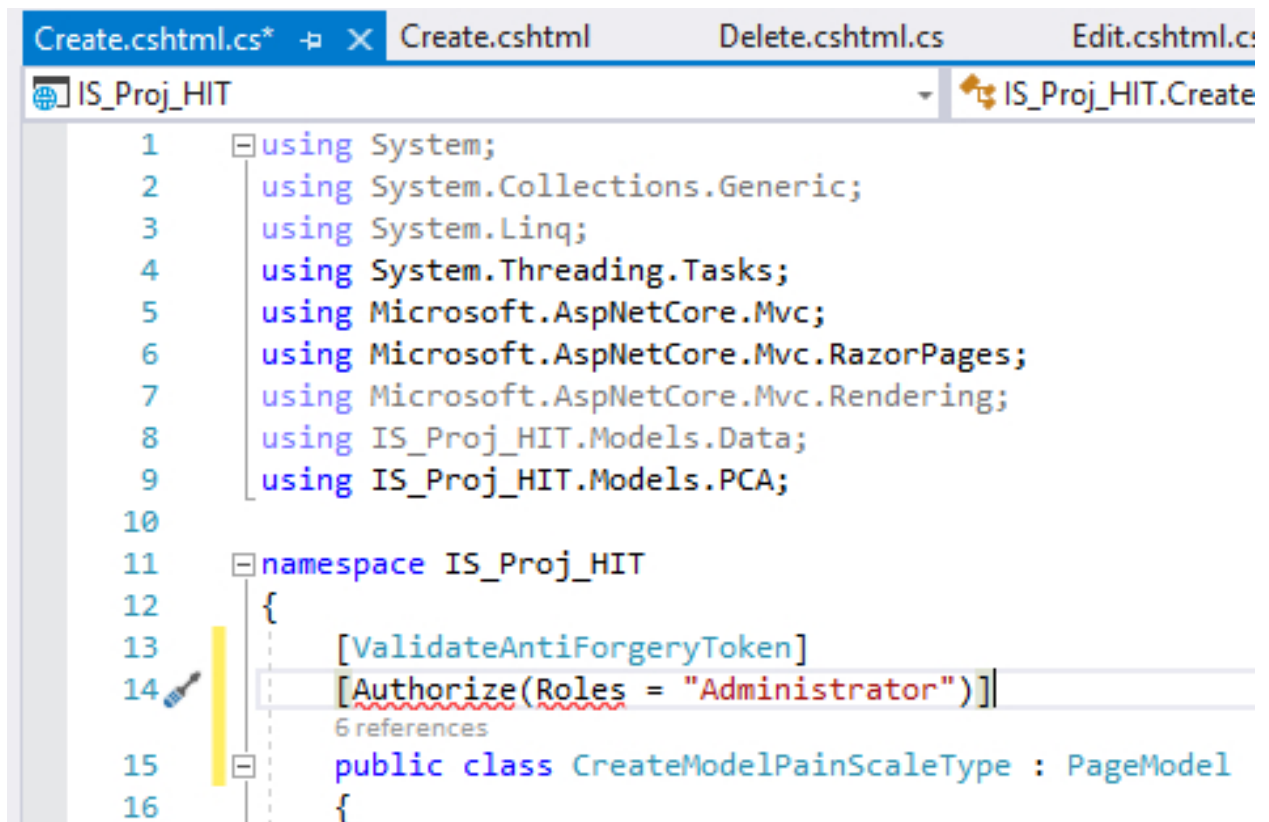


- 6) For Data Context Class choose WCTCHealthSystemContext.



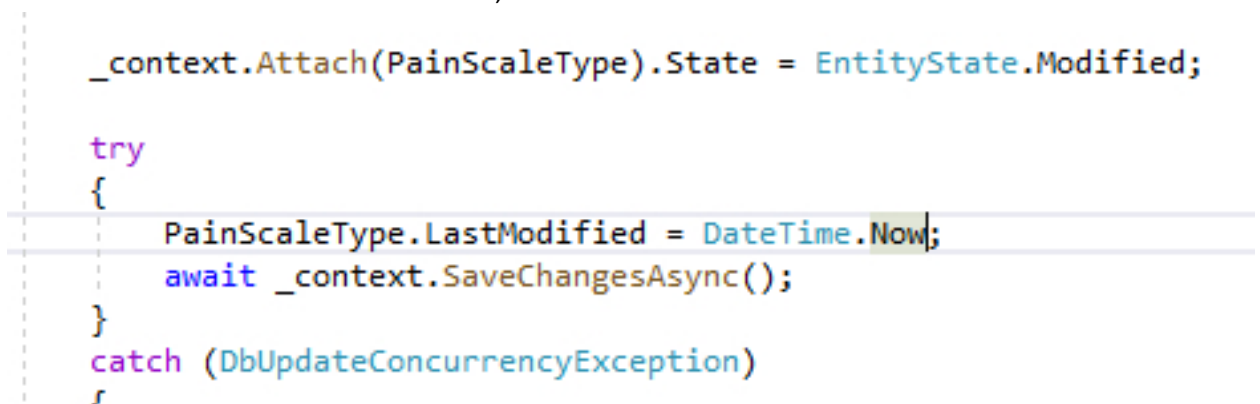
- 7) Click Add.
- 8) It takes a while for Visual Studio to generate the code.
- 9) This is the fiddly part. To prevent name collisions, go into each of the cshtml.cs files and add the table name to the class and the constructor. Then go into each of the cshtml files and add the table name to the @model IS_Proj_HIT.IndexModel .
- a. Index.cshtml.cs (ex: PainScaleType)
 - i. public class IndexModel : PageModel --> public class IndexModelPainScaleType : PageModel
 - ii. public IndexModel(IS_Proj_HIT.Models.Data.WCTCHealthSystemContext context) --> public IndexModelPainScaleType(IS_Proj_HIT.Models.Data.WCTCHealthSystemContext context)
 - b. Index.cshtml
 - i. @model IS_Proj_HIT.IndexModel --> @model IS_Proj_HIT.IndexModelPainScaleType
 - c. Do this for each of
 - i. Create
 - ii. Delete
 - iii. Details
 - iv. Edit
 - v. Index

10) Restrict to Administrator access for create, delete, edit.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Mvc;
6 using Microsoft.AspNetCore.Mvc.RazorPages;
7 using Microsoft.AspNetCore.Mvc.Rendering;
8 using IS_Proj_HIT.Models.Data;
9 using IS_Proj_HIT.Models.PCA;
10
11 namespace IS_Proj_HIT
12 {
13     [ValidateAntiForgeryToken]
14     [Authorize(Roles = "Administrator")]
15     public class CreateModelPainScaleType : PageModel
16     {
```

- a. Go into Create.cshtml.cs
 - i. Over the class declaration, add:
[ValidateAntiForgeryToken]
[Authorize(Roles = "Administrator")]
 - b. Right click on the line and choose Quick Actions and Refactorings. Add using Microsoft.AspNetCore.Authorization;
 - c. Repeat steps b. and c. for Delete.cshtml.cs
 - d. Repeat steps b. and c. for Edit.cshtml.cs
- 11) In Edit.cshtml -- Change LastModified to read-only by adding readonly to the <input> element.
- 12) In Edit.cshtml.cs, before the save, add (ex: PainScaleType.LastModified = DateTime.Now;)
tableName.LastModified=DateTime.Now;



```
_context.Attach(PainScaleType).State = EntityState.Modified;

try
{
    PainScaleType.LastModified = DateTime.Now;
    await _context.SaveChangesAsync();
}
catch (DbUpdateConcurrencyException)
{
```

13) If the table is dependent on another table (has a foreign key) the LastModified must also be explicitly set in Create. For example, in Create.cshtml.cs for CareSystemParameter, before the add, enter the line CareSystemParameter.LastModified = DateTime.Now;

14) Add Length Limits to character fields if any character fields are < varchar(max)

- a. Go into Create.cshtml and Edit.cshtml
- b. Add to the <input> element: `maxlength="200"` (or whatever the length is)

15) If you need to add delete confirmation, here is the code block:

```
<form method="post">
    <input type="hidden" asp-for="PulseRouteType.PulseRouteTypeId" />
    <input type="submit" value="Delete" class="btn btn-danger"
        onclick="if (!confirm('Are you sure you want to delete this record?'))
            return false;" /> |
    <a asp-page="./Index">Back to List</a>
</form>
```

16) If you have tables that require cascading delete (example CareSystemType delete requires all its CareSystemParameters be deleted first):

- a. Add the detail rows as a table in the master's delete.cshtml
 - i. `@foreach (var item in Model.CareSystemType.CareSystemParameters)`
 - ii. `@Html.DisplayFor(modelItem => item.Name)`
- b. Include the detail in the master's delete.cshtml.cs
 - i. `CareSystemType = await _context.CareSystemType.Include(csp=>csp.CareSystemParameters).FirstOrDefaultAsync(m => m.CareSystemTypeId == id);`
- c. Code the delete in the delete.cshtml.cs
 - i. Add using System.Transactions; to top
 - ii. Replace the correct parts in the public async Task<IActionResult> OnPostAsync(short? id)
 1. Make sure to remove all ASYNC within the transaction
 2. Add the include to the row get: `CareSystemType = await _context.CareSystemType.Include(csp => csp.CareSystemParameters).FirstOrDefaultAsync(m => m.CareSystemTypeId == id);`
 3. Make the delete/save changes transactional: `using (var tran = new TransactionScope()) { tran.Complete(); }`
 4. Within the transaction, add the child(ren) delete: `foreach (CareSystemParameter csp in CareSystemType.CareSystemParameters) { _context.CareSystemParameter.Remove(csp); }`
 5. Only after the child(ren) delete(s) do the master: `_context.CareSystemType.Remove(CareSystemType);`
 6. Save changes (not async): `_context.SaveChanges();`

17) General delete pages

- a. In delete.cshtml REMOVE
`<h3>Are you sure you want to delete this?</h3>`
and ADD
`<h3>@ViewData["RegularMessage"]</h3>`
`<h3 class="error">@ViewData["ErrorMessage"]</h3>`
- b. In delete.cshtml.cs add to the on get
`ViewData["RegularMessage"] = "Are you sure you want to delete this?";`
`ViewData["ErrorMessage"] = "";`
- c. In the same file add a check to the on post for existing system records using this lookup (sample:)
`// See if any PCA records exist with this type`
`bool usingExists = _context.Pcarecord.Any(p => p.TempRouteTypeId == TempRouteType.TempRouteTypeId);`

```
if (usingExists)
{ Console.WriteLine("PCA records exist using this record.");
  ViewData["RegularMessage"] = "";
  ViewData["ErrorMessage"] = "PCA records exist using this record. Delete not available.";
  return Page(); }
```

18) Update Admin for new CRUD pages

- a. Update the Administration Controller per the similar
- b. Update the Views | Administration | Index with the new pages set
- c. Create a View | Administration | DataEntry.cshtml per the similar