**Develop ASP.Net Reusable UI Components**

**Author:** Sampath Mannevaru

**Why do we need a new Shared UI Project?**

The main intention of adding a new project is to make the UI components reusable for your project which supports asp.net mvc, Razor or Blazor projects.

**What does the Shared.UI project contain?**

Shared UI project contains all reusable UI components and is currently developed into 3 sections and can be extendable.

1. **Views folder →** this folder will have all the .cshtml files, we will add reusable .cshtml files just like we add the files under web projects.
2. **wwwroot folder →** this folder will hold all the static files. we would add reusable js or css files just like we add the files under web projects.
3. **SharedUI.cs file →** contains two classes, SharedUI\_ConfigureOptions & SharedUI\_ServiceCollectionExtensions

Code is as below:

internal class SharedUI\_ConfigureOptions : IPostConfigureOptions<StaticFileOptions>

{

private readonly IHostingEnvironment \_environment;

public SharedUI\_ConfigureOptions(IHostingEnvironment environment)

{

\_environment = environment;

}

public void PostConfigure(string name, StaticFileOptions options)

{

// Basic initialization in case the options weren't initialized by any other component

options.ContentTypeProvider = options.ContentTypeProvider ?? new FileExtensionContentTypeProvider();

if (options.FileProvider == null && \_environment.WebRootFileProvider == null)

{

throw new InvalidOperationException("Missing FileProvider.");

}

options.FileProvider = options.FileProvider ?? \_environment.WebRootFileProvider;

// Add our provider

var filesProvider = new ManifestEmbeddedFileProvider(GetType().Assembly, "wwwroot");

options.FileProvider = new CompositeFileProvider(options.FileProvider, filesProvider);

}

}

public static class SharedUI\_ServiceCollectionExtensions

{

public static void AddSharedUI(this IServiceCollection services)

{

services.ConfigureOptions(typeof(SharedUI\_ConfigureOptions));

}

}

**Note:** We can also add Folder Like Pages, for supporting Blazor or razor web applications.

**How to use Shared UI components in Asp.Net Web Projects?**

Steps:

1. Create a shared UI class library project.
2. Add shared UI class library project reference on your web application.

In my case:

<ItemGroup>

…

<ProjectReference Include="..\Shared.UI\Shared.UI.csproj" />

…

</ItemGroup>

1. Add shared UI service on your web project’s Startup.cs Configure service method.

* Import shared UI project in your current Asp.Net project.

using Shared.UI; //import project on top of the class.

* Add your shared ui project as a service to your project.

services.AddSharedUI(); //configureservices method.

Or if you don’t have Main method in you .Net Core, add you services to web builder

var builder = WebApplication.CreateBuilder(args);  
 builder.services.AddSharedUI(); //configureservices method.

1. If we want to Use

a. Views-> set the path on name attribute on partial in cshtml.

Eg:

<partial name="/Views/Shared/Modal/\_GlobalTemplates/\_ModalInvoker.cshtml"

model="

New

{

PageSpecificCssAdjustment,

ModalInvokerDescriptionText,

ModalInvokerButtonText,

ModalInvokerButtonCss

}" />

b. static files → are hosted at the root folder of your web project, As point 2.

add references of static files as below example in your cshtml files.

<script src="~/js/Shared/Modal/GlobalTemplates/modal-global-template-utility.js"></script>

**What makes the views & js/static files in shared UI reusable?**

1. Adding shared UI project reference on your web application makes views on Shared UI project reusable, the .NET/ .Net Core UI project will understand that adding Shared UI Project reference which is a Razor Class library type, is automatically reusable and treats as an extension on views/pages folder.

Adding Shared.UI project reference on your web project:

Eg:

<ItemGroup>

…

<ProjectReference Include="..\Shared.UI\Shared.UI.csproj" />

…

</ItemGroup>

1. Using the Shared UI Views on your web project’s cshtml file:

Eg Path:

<partial name="/Views/Shared/Modal/\_GlobalTemplates/\_ModalInvoker.cshtml"

model="

new

{

PageSpecificCssAdjustment,

ModalInvokerDescriptionText,

ModalInvokerButtonText,

ModalInvokerButtonCss

}" />

1. EmbeddedResource & SharedUI.cs (Post Configure method) class on Shared.UI will make Js or any static files reusable.

<Project Sdk="Microsoft.NET.Sdk.Razor">

<PropertyGroup>

<TargetFramework>netcoreapp{.net framework version}</TargetFramework>

<AddRazorSupportForMvc>true</AddRazorSupportForMvc>

<GenerateEmbeddedFilesManifest>true</GenerateEmbeddedFilesManifest>

</PropertyGroup>

<ItemGroup>

<EmbeddedResource Include="wwwroot\*" />

</ItemGroup>

<ItemGroup>

<FrameworkReference Include="Microsoft.AspNetCore.App" />

<None Include="wwwroot\js\Shared\Modal\GlobalTemplates\modal-global-template-utility.js" />

<PackageReference Include="Microsoft.AspNetCore.Mvc" Version="2.1.2" />

<PackageReference Include="Microsoft.Extensions.FileProviders.Embedded" Version="2.1.1" />

<PackageReference Include="Microsoft.AspNetCore.StaticFiles" Version="2.1.1" />

</ItemGroup>

</Project>

**Note:** We have to add AddSharedUI service on your startup/program.cs class on configure services method and add using statement on startup/program class.  
As we saw in “**How to use Shared UI components in Asp.Net Web Projects?”**

using Shared.UI; -> on top of the class

services.AddSharedUI(); -> configureservices method.

**Can Shared UI projects be extended for Blazor components?**

Yes, Shared UI project is of Razor Class Library Project Type and targets .NET Core, It supports Blazor/Razor Web projects.

**References:**

<https://www.learnrazorpages.com/advanced/razor-class-library>