.NET Core - Globalization and Localization - Class library

Asked 4 years, 6 months ago Active 4 years ago Viewed 3k times

4

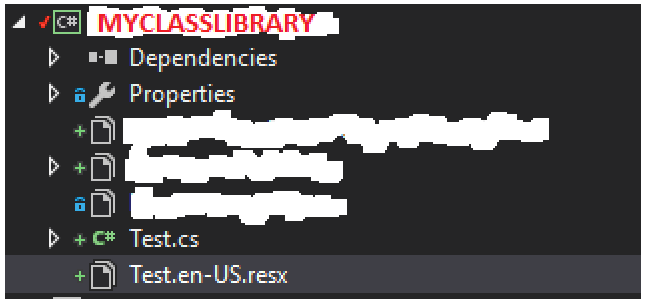
1

Following this documentation on how to implement globalization and localization using .NET Core, my goal is to store all my resources in a single global resource file located in a different project, (class library).

Project 1 - Startup.cs

```
public class Startup
{
    public Startup(IHostingEnvironment env)
    {
        var builder = new ConfigurationBuilder()
            .SetBasePath(env.ContentRootPath)
            .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
            .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true);

        if (env.IsEnvironment("Development"))
        {
            builder.AddApplicationInsightsSettings(developerMode: true);
        }

        builder.AddEnvironmentVariables();
        Configuration = builder.Build();
    }

    public IConfigurationRoot Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
    {
        services.AddApplicationInsightsTelemetry(Configuration);

        services.AddLocalization(lo => lo.ResourcesPath = "/MYCLASSLIBRARY");
//External project ~ How?

        services.AddMvc(config =>
        {
            var policy = new AuthorizationPolicyBuilder()
                             .RequireAuthenticatedUser()
                             .Build();
            config.Filters.Add(new AuthorizeFilter(policy));
        });
        services.Configure<RequestLocalizationOptions>(
            opts =>
            {
                var supportedCultures = new List<CultureInfo>
                {
                    new CultureInfo("en-US"),
                    new CultureInfo("sv-SE")
                };

                opts.DefaultRequestCulture = new RequestCulture(culture: "en-US",
uiCulture: "en-US");
                opts.SupportedCultures = supportedCultures;
                opts.SupportedUICultures = supportedCultures;
            });
    }

    public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerFactory)
    {
        loggerFactory.AddConsole();

        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }


app.UseRequestLocalization(app.ApplicationServices.GetService<IOptions<RequestLocalizatio
().Value);

        app.UseMvc();

        app.UseDefaultFiles();

        app.UseStaticFiles();
    }
}
```

Project 2 - Class library



Project 1 - Controller

```
using MYCLASSLIBRARY; //External project

[Route("api/[controller]")]
public class HelloController : Controller
{
    private readonly IStringLocalizer<Test> _localizer; //External project class

    public OrganisationController(IStringLocalizer<Test> localizer)
    {
        _mapper = mapper;

        _localizer = localizer;
    }

    [HttpGet("GetResource")]
    public string GetResource()
    {
        return _localizer["Help"];
    }
}
```

How can I reference an external project when setting ResourcesPath?

```
services.AddLocalization(lo => lo.ResourcesPath = "/MYCLASSLIBRARY");
```

c# .net localization asp.net-core .net-core

Share Edit Follow

asked Mar 24 '17 at 22:37

Reft
2,023 27 60

I am assuming you are working with .resx files. Here is a link for building them programmatically. The crux of .resx files is that they are embedded. Another answer tries to explain this but is sadly downvoted. Not sure why localization would need to be in a separate assembly. Perhaps, I am missing something. – trevorc Mar 24 '17 at 23:29

I'm not after building the resources programtically, I just want to register the external resource in my startup file:) The reason to using separate assembly is because I want to be able to use the same resource from multiple projects, for example retrieve resource from within Web.WebApi or my middle layer, Business.WebService, (two different projects). Both these projects will have a reference to the resouce class library. – Reft Mar 24 '17 at 23:45

Your "Another answer" is a 5 year old question. I didn't have any problems implementing external resources with mvc 5 and below. "While the concept of a .resx file per culture remains in ASP.NET Core, the way resources are used has changed quite significantly. In the previous version, when you added a .resx file to your solution, a designer file would be created, providing static strongly typed access to your resources through calls such as Resources.MyTitleString." – Reft Mar 24 '17 at 23:51

Ok. I understand now. However, I would question whether your DAL needs to handle that responsibility. At the end of the day your UI/WebApi layer is probably where the presentation should take place. I get where you are going but you could simplify things by moving all localization/presentation to the layer where it belongs. – trevorc Mar 24 '17 at 23:53

Alright I hear you, but imagine if I have an email service in my DAL with 100~ SendEmail functions. If I were to move the resource handling to my WebLayer it would be too much clutter. I would have to build some sort of extra EmailService in my WebLayer to create different email formats and then pass it to the DAL.. I don't know maybe thats better – Reft Mar 25 '17 at 0:06

1 Answer

Active Oldest Votes

2

Not sure you have already figured it out, just in case if you did not, here is the simple thing you could do in your current setup. Just replace the line below

`services.AddLocalization(lo => lo.ResourcesPath = "/MYCLASSLIBRARY"); //External project ~ How?` with

```
services.AddLocalization(); //Removing the root folder in the target assembly hence it will look for the file in the root of the
assembly of your MYCLASSLIBRARY
```

OR

Move your resource files under the folder "MYCLASSLIBRARY". Just make sure you don't remove the leading '/' when you define the ResourcesPath.

I hope it helps.

Share Edit Follow

answered Aug 24 '17 at 1:16

slolam

49 4

Thank you, it helped me. – graycrow Apr 6 '18 at 8:50