

# Asp.Net Core Localized Resource in Separate Assembly

Asked 4 years, 7 months ago   Active 8 months ago   Viewed 9k times



20



6



I am trying to use the new Localization features of .NET Core, but outside the simple sample Microsoft has provided here, <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/localization#resource-file-naming>.

I have my Controllers in a separate project, ProjectA.Controllers, while I have a shared resource class in a common project, ProjectB.Localization. I've configured my startup class as prescribed in the docs.

I am unclear on what to name my resource file and where exactly to put it. I've configured the option to store in the directory "Resources". Is that in the Web project or my ProjectB.Localization where my SharedResource class is? The docs say that if it's a separate assembly, the full namespace should be used. So I've named it, "WorldCart.Facilities.Localization.SharedResource.es.resx" and placed it in the resources folder of the website.

When I run the web app, and debug in the home controller, I do not get a translated string, I get the english version.

Any ideas?

[localization](#)   [asp.net-core](#)

Share Edit Follow

asked Feb 4 '17 at 19:10



[Joshua Belden](#)

9,367   5   35   53

I'm loading <localhost/?culture=es>. Es has been added to the supported cultures. – [Joshua Belden](#) Feb 4 '17 at 19:23

did you found something? I want to implement my own resource finder to be able to put the resource in the controller or view folder then fallback to a global resource if no string is found ... [stackoverflow.com/questions/44221309/...](https://stackoverflow.com/questions/44221309/...) – [Benoît](#) May 28 '17 at 5:01

1 I haven't found anything, but I moved on to other challenges and haven't looked back. Microsoft docs should be improved and may cover the issue better. – [Joshua Belden](#) May 30 '17 at 15:09

all i have found is using resourcepath to "". So resources are in the right dirrectory, no need to use full namespace. But i wish to find a way to

implement a resourceLocator or a resourceFinder. Fow now the resource work like this ... – Benoit Jun 6 '17 at 0:54

## 1 Answer

Active

Oldest

Votes



32



Very late answer, but it might help someone... I had a similar situation where I had to have the resource files in separate common assembly instead of having it inside the mail web/api project (Core 2.1). The reason being, I could be using the localized resources from other assemblies like Business or DAL layer for throwing warning/error/information messages. This is what I did:

Assume that my web project namespace is `MyApp.Web` and my resources are in separate class lib `MyApp.Resources`. In the resources library, create a folder (optional), say "Messages", and create a class `Messages.cs`. Create the resource files inside the same folder adhering to the naming conventions. For example, `Messages.fr.resx`.

In the `ConfigureServices` method of the main project, add the localization without any resource path\*:

```
services.AddLocalization();

services.Configure<RequestLocalizationOptions>(
    opts =>
    {
        /* your configurations*/
        var supportedCultures = new List<CultureInfo>
        {
            new CultureInfo("en"),
            new CultureInfo("fr")
        };

        opts.DefaultRequestCulture = new RequestCulture("en", "en");
        // Formatting numbers, dates, etc.
        opts.SupportedCultures = supportedCultures;
        // UI strings that we have localized.
        opts.SupportedUICultures = supportedCultures;
    });
```

And in the `Configure` method, add `app.UseRequestLocalization();`

In your controller, inject `IStringLocalizer<Messages> localizer`, where `Messages` is the class you created in the Resources library. All your localized resources will be available in the `localizer` object, i.e., `localizer["your key or default text"]`.

- The reason for not adding any `ResourcePath` in the `services.AddLocalization();` options is due to the reason that both the resource files (`Messages.fr.resx`) and the dummy class (`Messages.cs`) are in the same path. The framework will check for the resource file relative to the class which we have specified in `IStringLocalizer<>`. If the `Messages.cs` was in the root folder of `MyApp.Resources` lib and the resource files were inside folder "xyz", then the configuration should be `services.AddLocalization(opts => opts.ResourcesPath = "xyz");`

### Startup.cs

```
// This method gets called by the runtime. Use this method to add services to the container.
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddLocalization();

    services.Configure<RequestLocalizationOptions>(
        opts =>
        {
            var supportedCultures = new List<CultureInfo>
            {
                new CultureInfo("en"),
                new CultureInfo("fr")
            };

            opts.DefaultRequestCulture = new RequestCulture("en", "en");
            // Formatting numbers, dates, etc.
            opts.SupportedCultures = supportedCultures;
            // UI strings that we have localized.
            opts.SupportedUICultures = supportedCultures;
        });

    services.AddMvc()
        .SetCompatibilityVersion(CompatibilityVersion.Version_2_1);

    IoC.Register(services, Configuration);
}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsEnvironment("Debug"))
    {
        app.UseDeveloperExceptionPage();
    }

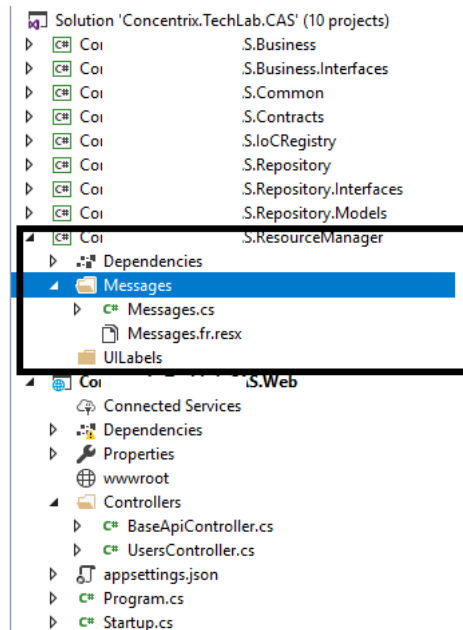
    app.UseRequestLocalization();

    app.UseMvc();
}
```

### Controller

```
[Route("api/[controller]")]
[ApiController]
1 reference | 0 requests
public class UsersController : ControllerBase
{
    readonly IConfiguration configuration;
    IUserBusiness userBusiness;
```

### Project Structure



```

IStringLocalizer<Messages> localizer;
0 references | 0 exceptions
public UsersController(IConfiguration configuration, IUserBusiness userBusiness, IStringLocalizer<Messages> localizer)
{
    this.configuration = configuration;
    this.userBusiness = userBusiness;
    this.localizer = localizer;
}

[HttpGet]
0 references | 0 requests | 0 exceptions
public async Task<ActionResult<IEnumerable<UserDetails>>> GetAsync()
{
    var test = localizer["This field is required"];
    var result = await userBusiness.GetUsersAsync();

    return Ok(result);
}

```

**UPDATE** - Responding to the queries in the comments:

**MVC Views** In MVC Views, the documented approach uses `IViewLocalizer`, but does not support resource sharing. So you can inject `IStringLocalizer<>` in the view for using the shared resources. For example:

```

@inject IStringLocalizer<Messages> localizer
<h2>Information - @localizer["Shared resource access in MVC Views"]</h2>

```

**Data Annotations** In order to use shared resources in the data annotations, you can use the factory method in the service:

```

services.AddMvc()
    .SetCompatibilityVersion(CompatibilityVersion.Version_2_1)
    .AddDataAnnotationsLocalization(options => {
        options.DataAnnotationLocalizerProvider = (type, factory) =>
            factory.Create(typeof(Messages));
    });

```

where the `Messages` in the `typeof(Messages)` is your shared resource dummy class.

Share Edit Follow

edited Jan 11 at 22:34



Grigory Zhadko

1,112 1 14 22

answered Oct 12 '18 at 8:09



Developer


5,675 2 16 24

Do you have any resource (official or not) for this approach? I wish you would documented it with a better one. – [ibubi](#) Oct 16 '18 at 8:07


@ibubi - I couldn't find any resource or documentation for the same. Ended up with similar requirement and figured it out myself. There is one

another blog with a different approach ([blog.mohnady.com/2017/05/...](http://blog.mohnady.com/2017/05/...)) which I kind of didn't like. – Developer Oct 16 '18 at 9:49


---

I see that you are implementing that in an api project, so data annotations on viewmodels might not be in your scope, however do you know this approach covers model localization via data annotations? – ibubi Oct 17 '18 at 12:46 

---

For **mvc views**, `IViewLocalizer` can be implemented if it is intended to go on with conventional approach rather than a shared resource, if needed a sample, ping me, I can update answer for this. – ibubi Nov 23 '18 at 13:41 

---

- 3 Thank you so much @Developer!! I feel the .net core's documentation is miserable, it only scratches the surface of any kind of real world scenarios. Kudos to people like you for actually making "docs" for scenarios that matter!! – viper Apr 30 '19 at 12:45 
-