

# How to implement resource files localization ASP-NET.core?

Asked 1 year, 3 months ago   Active 1 year ago   Viewed 350 times



I am asked to implement localization in a `net.core` solution as following:

0



- Resource files must be in a separated project (like "MyProject.Common") to be used in all other projects in the solution,
- Resources must be split by "sections", for example, we have 3 areas in the Web project like following :
  - Users ,
  - Content ,
  - Administration ,

So I'm asked to have something like:

- UsersResources.fr-CA.resx,
- UsersResources.en-CA.resx,
- ContentResources.fr-Ca.resx,
- ...

I started to read documentation for Localization is AP-NET core and I'm a bit confused on how itr works. Doesn't seem like what i'm told to do is possible.

Thing is, I may need to use resources in Business, Views and Controllers so I'm searching for a way to implement it so the team could use the old way, by calling `ContentResources.MyCustomResource` .

Is there a way to get close from that?

I found a post where someone was mentioning <https://www.nuget.org/packages/ResXResourceReader.NetStandard>.

But I don't know if it will fit my needs...

#EDIT : So, trying to implement Laz's solution for shared resources.

So far, in startup I have this : in `ConfigureServices` :

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

services.AddLocalization();

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

and in `Configure` :

```
app.UseRequestLocalization();
// used to force culture to fr but doesn't seem to work
var cultureInfo = new CultureInfo("fr");
CultureInfo.DefaultThreadCurrentCulture = cultureInfo;
CultureInfo.DefaultThreadCurrentUICulture = cultureInfo;
```

in `MyProject.Common`, I have this structure:

```
MyProject.Common
-Resources
--SharedResources.cs
---SharedResources.fr.resx
```

```

---SharedResources.en.resx
--UsersResources.cs
---UsersResources.fr.resx
---UsersResources.en.resx

```

Let's say I want to use `SharedResources`.

In `SharedResources.en.resx` I added resources:

Strings ▾ * Add Resource ▾ ✕ Remove Resource ▾ Access Modifier: No code gen ▾		
	Name ▲	Value
▶	Test	some test here
	Test2	some other test here

In `SharedResources.fr.resx` I added resources:

Strings ▾ * Add Resource ▾ ✕ Remove Resource ▾ Access Modifier: No code gen ▾		
	Name ▲	Value
▶	Test	un test ici
	Test2	un autre test ici

Now in my `UserService`, in `Business` layer, I did this:

```

private readonly IStringLocalizer Localizer;

public UserService(IStringLocalizerFactory factory)
{
    var type = typeof(SharedResources);
    var assemblyName = new AssemblyName(type.GetTypeInfo().Assembly.FullName);
    _localizer = factory.Create(type);
}

public void Test()
{
    var test = Localizer["Test"]; //using the key of resources file i want
}

```

but all I get as a result in `test` variable is "Test", which is the `key` of the resource, and not the `value`.

[asp.net-core](#) [localization](#) [resources](#)

Share Edit Follow

edited Sep 7 '20 at 19:01



Jason Aller

3,427 28 37 36

asked Jun 2 '20 at 15:30



j0w

425 1 8 26

1 You can refer to this SO thread that discussed similar requirement : [stackoverflow.com/a/52775064/6751634](https://stackoverflow.com/a/52775064/6751634) – Fei Han Jun 3 '20 at 7:27

@FeiHan Thanks for your time. Also tried the solution you linked but ended as the same result as Laz Ziya's solution below : all i get is the `key` of the resource – j0w Jun 3 '20 at 18:21

## 1 Answer

Active

Oldest

Votes



The default localization setup of .net core can be based on [one shared resource file](#) or [based on the class name](#).

1



In your case, you can use the shared resource approach, but you have to create the customized localizer in each controller/class by providing the factory with the desired resource type.



First create a class library with the desired resources, create a public dummy class for each resource type you want, so the class library structure can be like below:



```
// SharedResourcesLibrary.csproj
```

```
- UsersResources.cs
- UsersResources.fr-ca.resx
- UsersResources.en-ca.resx

- ContentResources.cs
- ContentResources.fr-ca.resx
- ContentResources.en-ca.resx
```

...

The dummy classes are empty, they are used just as a type to call the relevant resx file.

```
// Dummy users resources class
public class UsersResources { }
```

Then after referencing the ResourcesLibrary project into other projects, you can use the resources by calling the relevant resource type (the dummy class):

```
using SharedResourcesLibrary;

public class UsersController : Controller
{
    private readonly IStringLocalizer _localizer;

    public UsersController(IStringLocalizerFactory factory)
    {
        var type = typeof(UsersResources);
        _localizer = factory.Create(type);
    }

    public IActionResult About()
    {
        ViewData["Message"] = _localizer["Welcome."];
    }
}
```

To use other resources just create the localizer using the relevant resource type.

Another approach can be done by creating custom multiple `IStringLocalizer` s according to your areas, then inject them in the controllers.

```
// Create one localizer for each area
public class UsersLocalizer : IStringLocalizer
{
    private readonly IStringLocalizer _localizer;

    public UsersLocalizer(IStringLocalizerFactory factory)
    {
```

```

        var type = typeof(UsersResources);
        var assemblyName = new AssemblyName(type.GetTypeInfo().Assembly.FullName);
        _localizer = factory.Create(type);
    }

    public LocalizedString this[string name] => _localizer[name];

    public LocalizedString this[string name, params object[] arguments] =>
        _localizer[name, arguments];

    // ...
}

```

Similarly you can create localizers for other areas... then register in startup:

```

services.AddTransient<IStringLocalizer, UsersLocalizer>();
services.AddTransient<IStringLocalizer, AdminsLocalizer>();
services.AddTransient<IStringLocalizer, ContentLocalizer>();
// ...

```

This way all localizers will be registered, and if you simply inject `IStringLocalizer` it will get the last registered one, because all localizers are implementing the same `IStringLocalizer` interface.

So you have to do type selection for injecting the correct localizer:

```

public UsersController : Controller
{
    private readonly IStringLocalizer _localizer;

    public UsersController(IEnumerable<IStringLocalizer> localizers)
    {
        _localizer = localizers.FirstOrDefault(x => x.GetType() ==
            typeof(UsersLocalizer));
    }

    public IActionResult About()
    {
        ViewData["Message"] = _localizer["Welcome."];
    }
}

```

You can refer to this article for different ways of [Registering multiple implementation with the same interface in Asp.Net Core](https://stackoverflow.com/questions/62155569/how-to-implement-resource-files-localization-asp-net-core)

[Share](#) [Edit](#) [Follow](#)[edited Jun 4 '20 at 5:57](#)

answered Jun 3 '20 at 5:42

[LazZiya](#)**4,004**

2

16

30

---

Thank you for the detailed answer. Currently trying the first approach (the `factory.Create(type) one`), and all i have as a result when I use `Localizer["Test"]` in my business layer is the key `Test` itself, and not the value `Here is a test`. Any idea what i could have missed here?

– [j0w](#) Jun 3 '20 at 15:50

---

Do you get the same result when the resource files in the same project? – [LazZiya](#) Jun 4 '20 at 5:29

---

And how do you change the culture? See the output of `@CultureInfo.CurrentCulture.Name` in any view to ensure you are at the right culture

– [LazZiya](#) Jun 4 '20 at 5:34

---

I just created a test sample with the first approach and your startup settings, everything worked fine *can share if you want* :) So, I assume your issue is with the culture setup and current culture. – [LazZiya](#) Jun 4 '20 at 6:01

---

1 [@j0w](#) glad to know it worked, and for the enums that would be another question, but [this](#) might help :) – [LazZiya](#) Jun 4 '20 at 13:59

---