.NET Core Web API Language Support

```
[Route("api/[controller]")]
     public class HomeController : Controller
     {
          private readonly IStringLocalizer<HomeController> _localizer;

          public HomeController(IStringLocalizer<HomeController> localizer)
          {
              _localizer = localizer;
          }
          [HttpGet("index/{id}")]
          public IActionResult Index()
          {
              return new ObjectResult(_localizer["Hello"]);

          }

     }
```

Add Resources folder then, add Controllers.HomeController.tr-TR.resx and Controllers.HomeController.en-EN.resx files in this Resources folder

Startup.cs file

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Localization;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using Microsoft.EntityFrameworkCore;
using TodoApi.Models;
using Microsoft.AspNetCore.Mvc.Razor;
using Microsoft.Extensions.Options;

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

          public IConfigurationRoot Configuration { get; }

          // This method gets called by the runtime. Use this method to add services to the
container.
          public void ConfigureServices(IServiceCollection services)
          {
              // Add framework services.
              services.AddDbContext<TodoContext>(opt => opt.UseInMemoryDatabase());
              services.AddLocalization(options => options.ResourcesPath= "Resources");
```

```
            services.AddMvc().AddViewLocalization(
            LanguageViewLocationExpanderFormat.Suffix,
            opts => { opts.ResourcesPath= "Resources"; })
            .AddDataAnnotationsLocalization();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();
            var supportedCultures = new[]
            {

                new CultureInfo("en"),
                new CultureInfo("en-US"),
                new CultureInfo("tr"),
                new CultureInfo("tr-TR"),

            };

            app.UseRequestLocalization(new RequestLocalizationOptions
            {
                DefaultRequestCulture = new RequestCulture("tr-TR"),
                // Formatting numbers, dates, etc.
                SupportedCultures = supportedCultures,
                // UI strings that we have localized.
                SupportedUICultures = supportedCultures
            });

            app.UseMvc(routes =>
            {
                routes.MapRoute(
                name: "default",
                template: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}
```

When requesting Accept-Language header value `en-EN,tr;q=0.8,en-US;q=0.5,en;q=0.3` value, result will be as follows:

```
{
    "name": "Hello",
    "value": "Hi",
    "resourceNotFound": false
}
```

If Accept-Language header value is `tr-TR,tr;q=0.8,en-US;q=0.5,en;q=0.3` result will be:

```
{
    "name": "Hello",
    "value": "Merhaba",
    "resourceNotFound": false
}
```