# Gestione della culture in una WebApi Core

## Startup.cs ConfigureServices

// Globalization START

// Imposto il percorso dove sono contenuti i files Resources

// I files Resources, siccome sono utilizzati nei controller devono avere la seguente nomenclatura

// Controllers.<NomeController>.it.resx per esempio

// Controllers.TestController.it.resx

services.AddLocalization(options => options.ResourcesPath = "Resources");

// Globalization END

// Globalization START

// Indico l'elenco delle lingue gestite dall'applicazione

services.Configure<RequestLocalizationOptions>(options =>

{

var supportedCultures = new List<CultureInfo>

{

new CultureInfo("it"),

new CultureInfo("en")

};

options.DefaultRequestCulture = new RequestCulture("it");

options.SupportedCultures = supportedCultures;

options.SupportedUICultures = supportedCultures;

});

// Globalization END

## Startup.cs Configure

Come prima cosa indicare quanto segue :

var supportedCultures = new[]

{

new CultureInfo("it"),

new CultureInfo("en"),

};

app.UseRequestLocalization(new RequestLocalizationOptions

{

DefaultRequestCulture = new RequestCulture("it"),

// Formatting numbers, dates, etc.

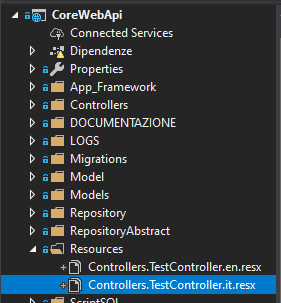
SupportedCultures = supportedCultures,

// UI strings that we have localized.

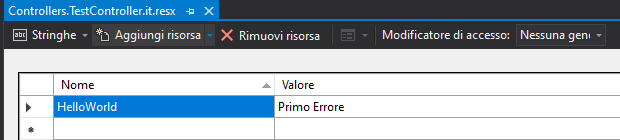
SupportedUICultures = supportedCultures

});

## Creare Una Cartella Resources



## Esempio di file Resource



## Controller TestController

private readonly IStringLocalizer<TestController> \_localizer;

public TestController(…, IStringLocalizer<TestController> localizer)

{

…

this.\_localizer = localizer;

}

## Richiamo di un valore da tradurre

[HttpGet]

[Route("TestLang")]

public void TestLang()

{

string a = \_localizer["HelloWorld"];

}