
mundipagg

Thiago Barradas

Software Enginner | Mundipagg

[Web Applications] [ASP .NET]
[API RESTful] [Microsoft ❤ Linux]
[Elasticsearch] [Docker]
[DevOps] [Agile]

tbarradas@mundipagg.com
LinkedIn: thiagobarradas
(21) 99329-9143

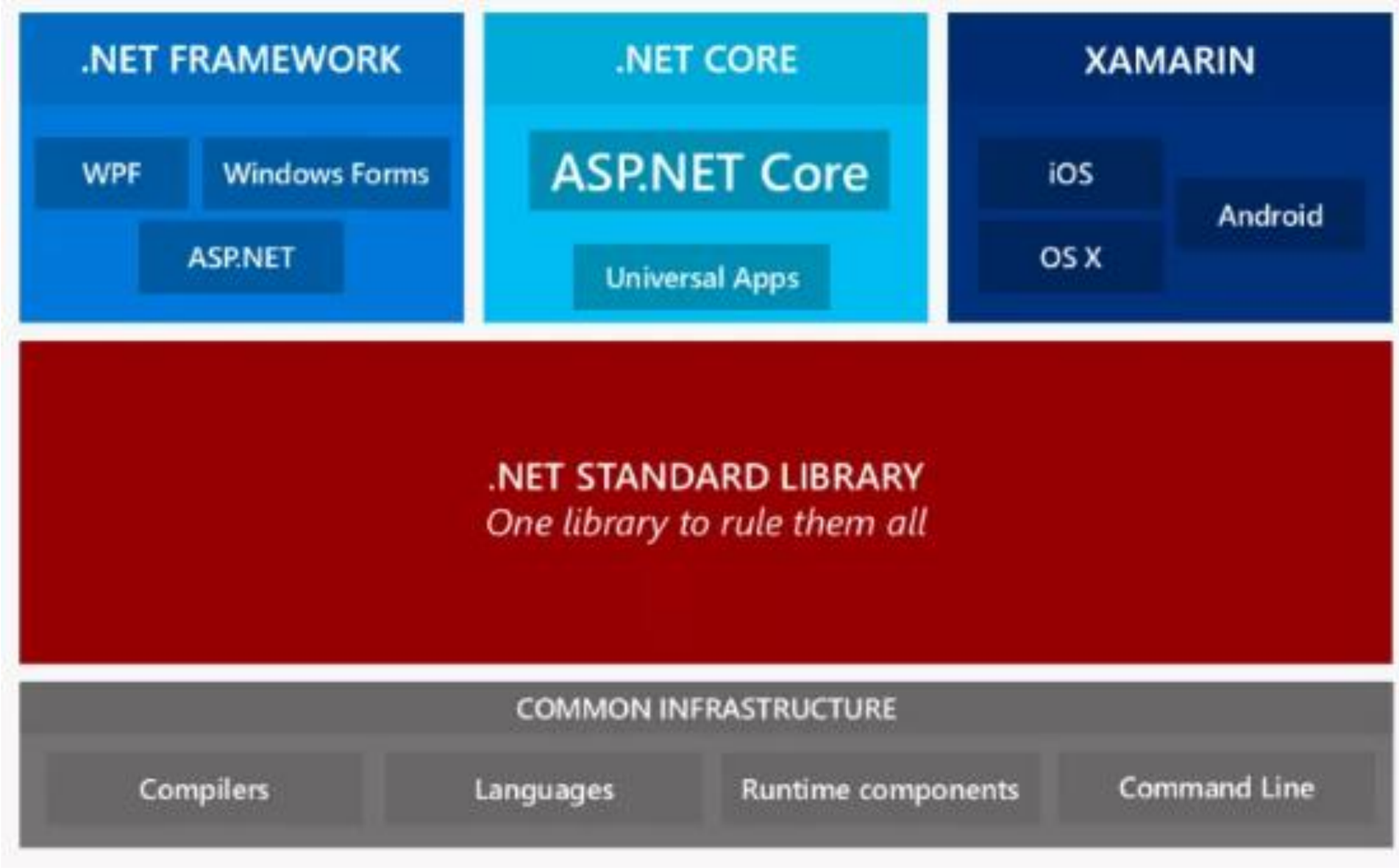
CRIANDO **COMPONENTES** E DISPONIBILIZANDO-O COMO **OPENSOURCE** EM 5 MINUTOS

POR QUÊ COMPONENTIZAR?

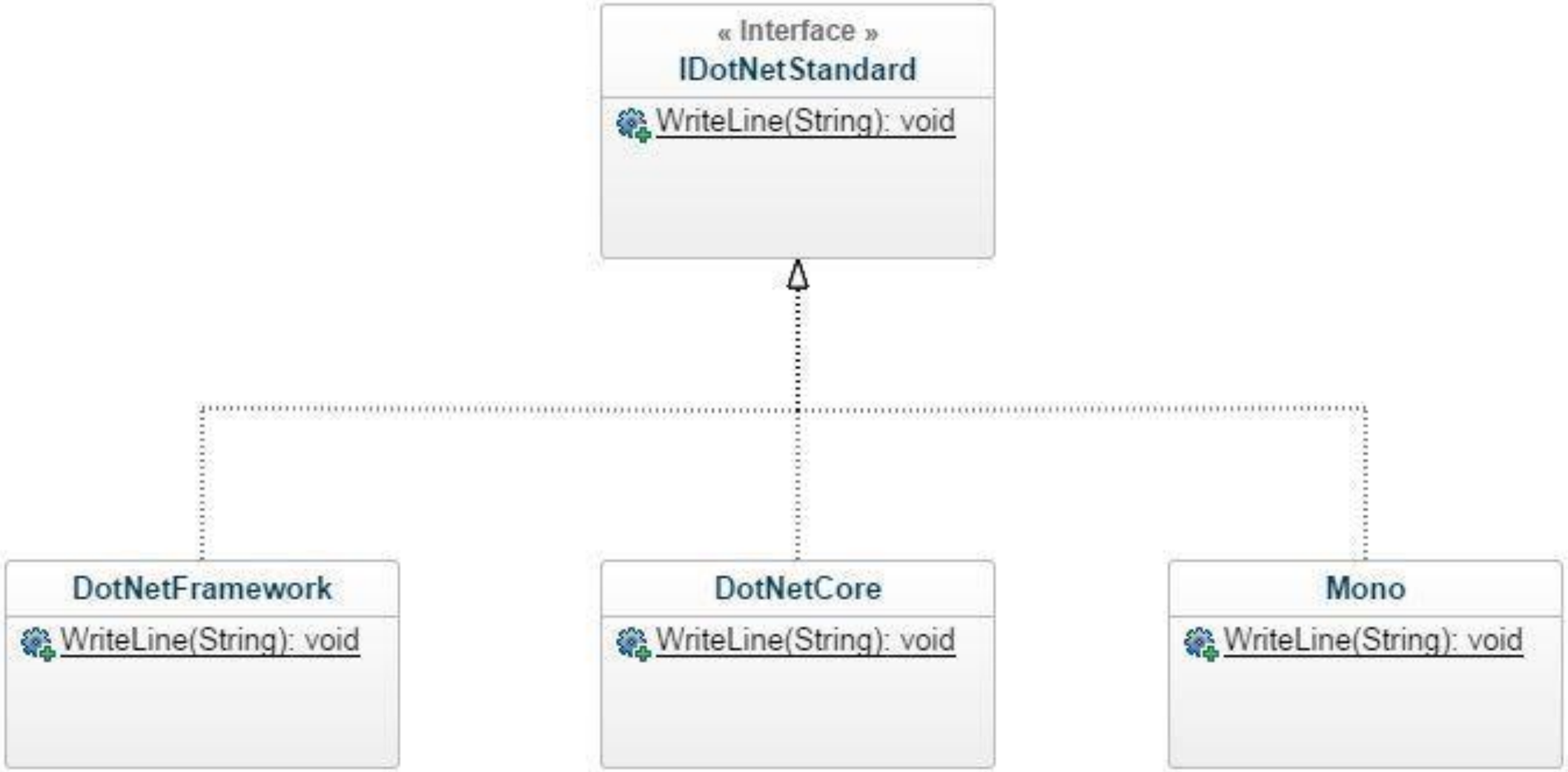
Por quê componentizar?

- Isola Comportamento / Lógica
- Evita Mutabilidade do Comportamento
- Evita Duplicação de Código
- Facilita o Teste Unitário
- Facilita o Compartilhamento

.NET STANDARD



.NET STANDARD



.NET STANDARD

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0	2.1
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	**1.0**	**2.0**	**3.0**
.NET Framework	4.5	**4.5**	**4.5.1**	**4.6**	4.6.1	4.6.1[1]	4.6.1[1]	**4.6.1**[1]	*N/A*[2]
Mono	4.6	4.6	4.6	4.6	4.6	4.6	**4.6**	**5.4**	**6.4**
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	**10.0**	**10.14**	**12.16**
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	**3.0**	**3.8**	**5.16**
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	**7.0**	**8.0**	**10.0**
Unity	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	2018.1	**2018.1**	*TBD*
Universal Windows Platform	8.0	**8.0**	**8.1**	10.0	**10.0**	10.0.16299	10.0.16299	**10.0.16299**	*TBD*

TFM

TFM (Target Framework Monikers)

netstandard1.0	netcoreapp1.0	net11	net451
netstandard1.1	netcoreapp1.1	net20	net452
netstandard1.2	netcoreapp2.0	net35	net46
netstandard1.3	netcoreapp2.1	net40	net461
netstandard1.4	netcoreapp2.2	net403	net462
netstandard1.5	netcoreapp3.0	net45	net47
netstandard1.6			net471
netstandard2.0			net472
netstandard2.1			net48

TFM

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFrameworks>netstandard1.4;net40;net45</TargetFrameworks>
  </PropertyGroup>

  <!-- Conditionally obtain references for the .NET Framework 4.0 target -->
  <ItemGroup Condition=" '$(TargetFramework)' == 'net40' ">
    <Reference Include="System.Net" />
  </ItemGroup>

  <!-- Conditionally obtain references for the .NET Framework 4.5 target -->
  <ItemGroup Condition=" '$(TargetFramework)' == 'net45' ">
    <Reference Include="System.Net.Http" />
    <Reference Include="System.Threading.Tasks" />
  </ItemGroup>

</Project>
```

PREPROCESSOR SYMBOLS

```
public class MyClass
{
    static void Main()
    {
#if NET40
        Console.WriteLine("Target framework: .NET Framework 4.0");
#elif NET45
        Console.WriteLine("Target framework: .NET Framework 4.5");
#else
        Console.WriteLine("Target framework: .NET Standard 1.4");
#endif
    }
}
```

PREPROCESSOR SYMBOLS

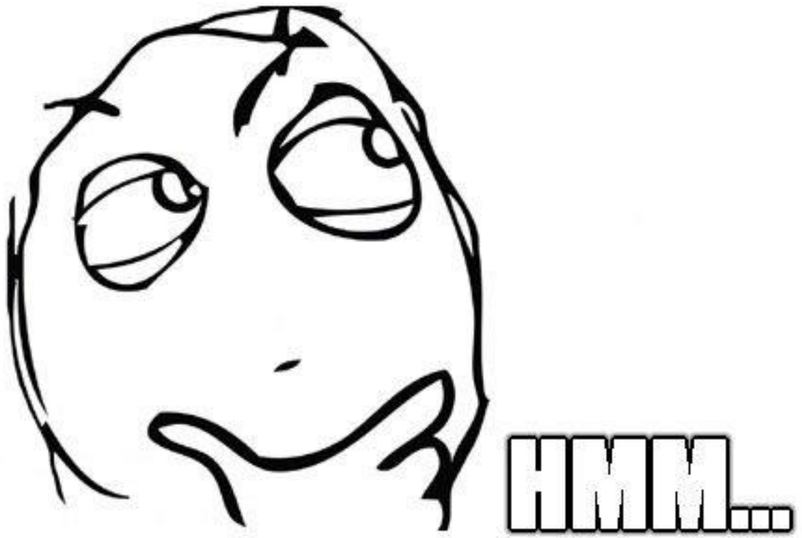
Target Frameworks	Symbols
.NET Framework	`NETFRAMEWORK`, `NET20`, `NET35`, `NET40`, `NET45`, `NET451`, `NET452`, `NET46`, `NET461`, `NET462`, `NET47`, `NET471`, `NET472`, `NET48`
.NET Standard	`NETSTANDARD`, `NETSTANDARD1_0`, `NETSTANDARD1_1`, `NETSTANDARD1_2`, `NETSTANDARD1_3`, `NETSTANDARD1_4`, `NETSTANDARD1_5`, `NETSTANDARD1_6`, `NETSTANDARD2_0`, `NETSTANDARD2_1`
.NET Core	`NETCOREAPP`, `NETCOREAPP1_0`, `NETCOREAPP1_1`, `NETCOREAPP2_0`, `NETCOREAPP2_1`, `NETCOREAPP2_2`, `NETCOREAPP3_0`

ENTÃO...

netstandard2.0

=

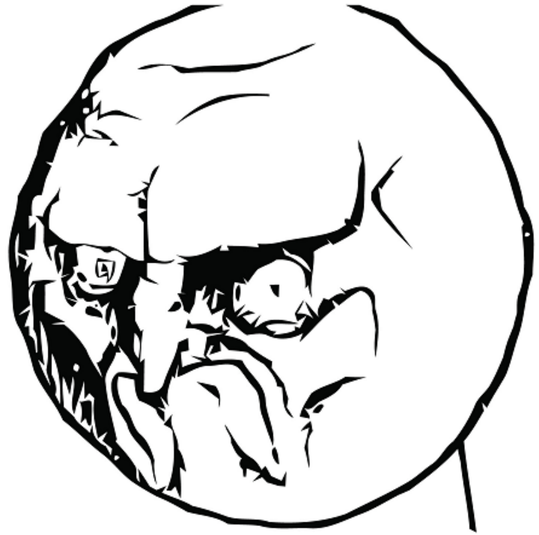
netcoreapp2.0;netcoreapp2.1;netcoreapp2.2;net461;net462;net47;net471;net472;net48



ENTÃO...

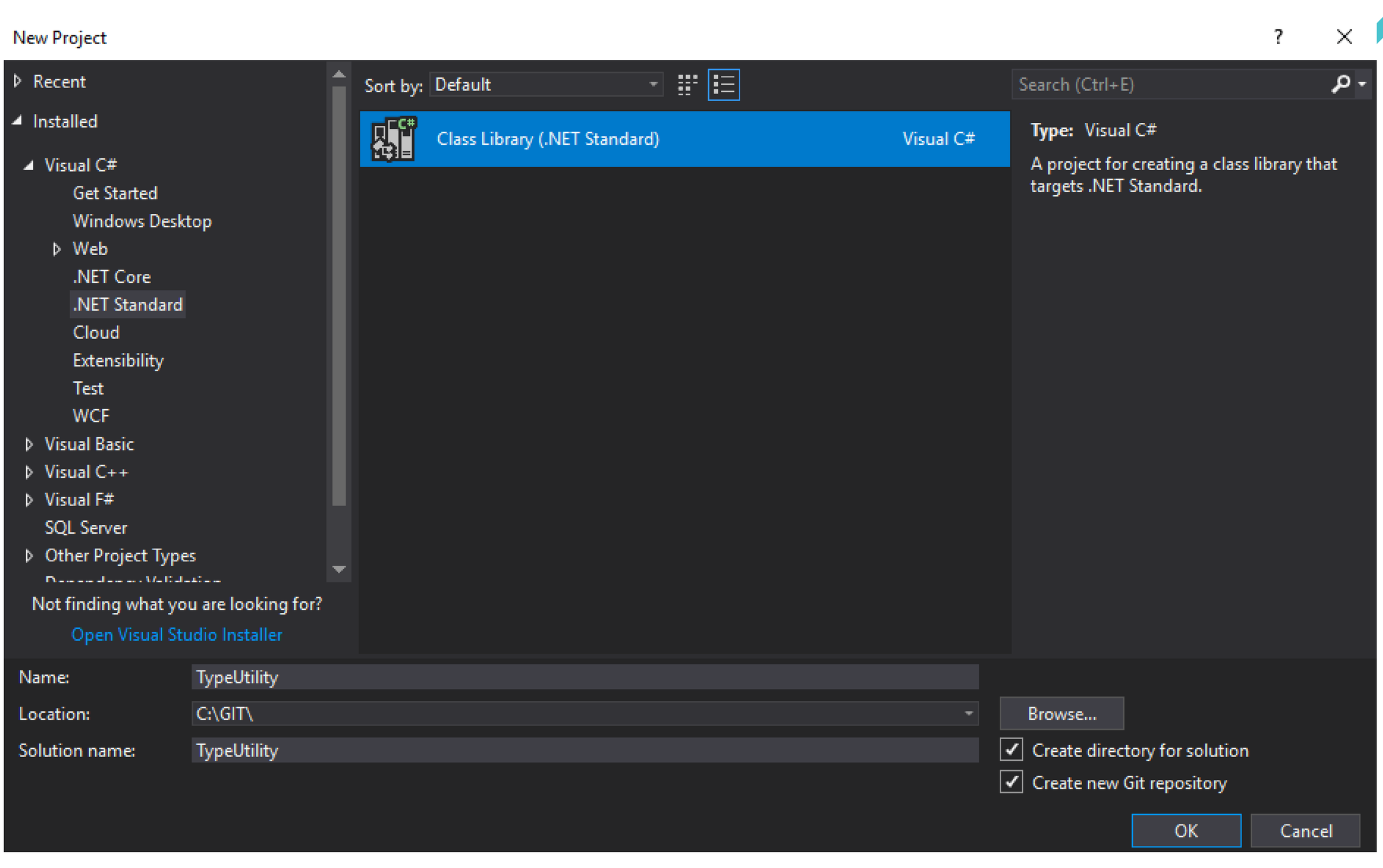
netstandard2.0

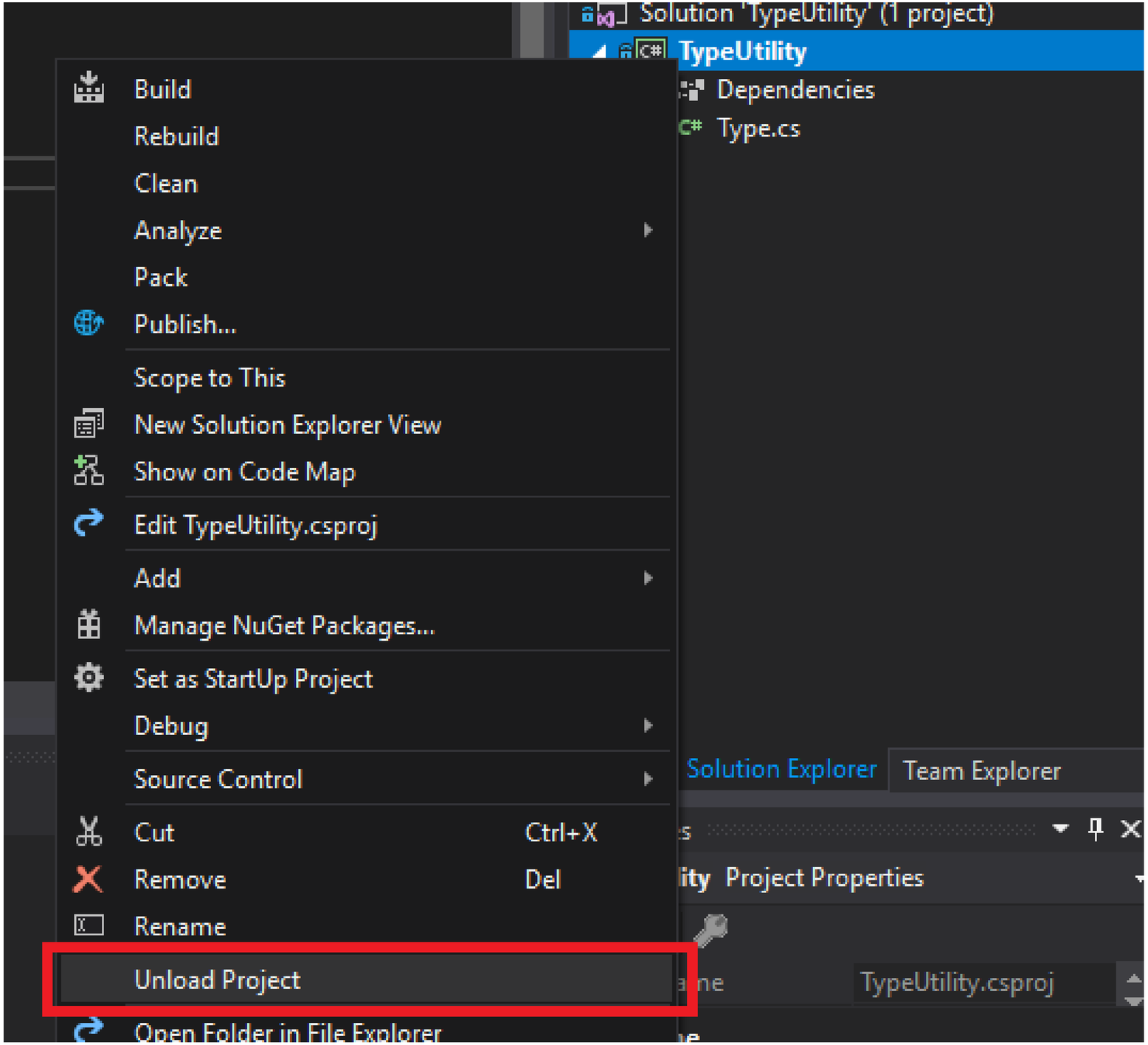
netcoreapp2.0;netcoreapp2.1;netcoreapp2.2;net461;net462;net47;net471;net472;net48

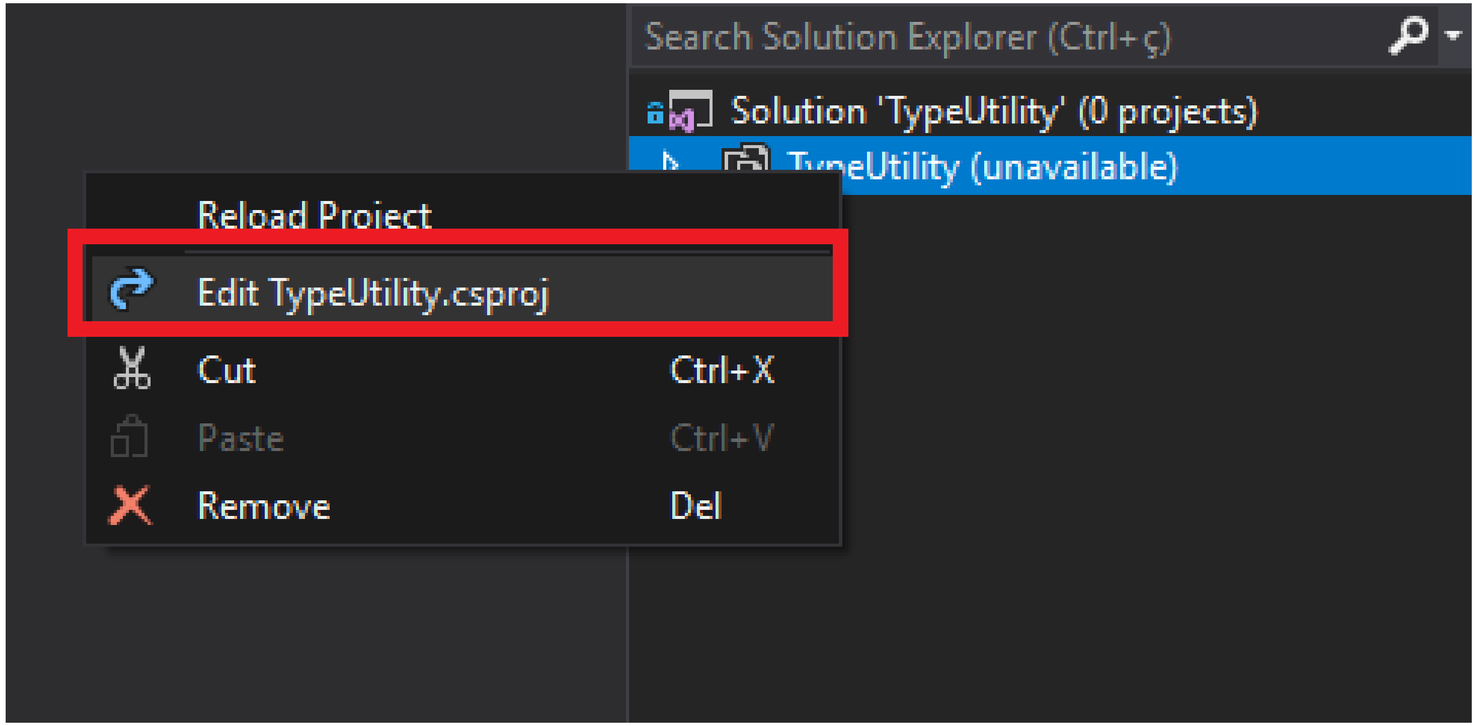


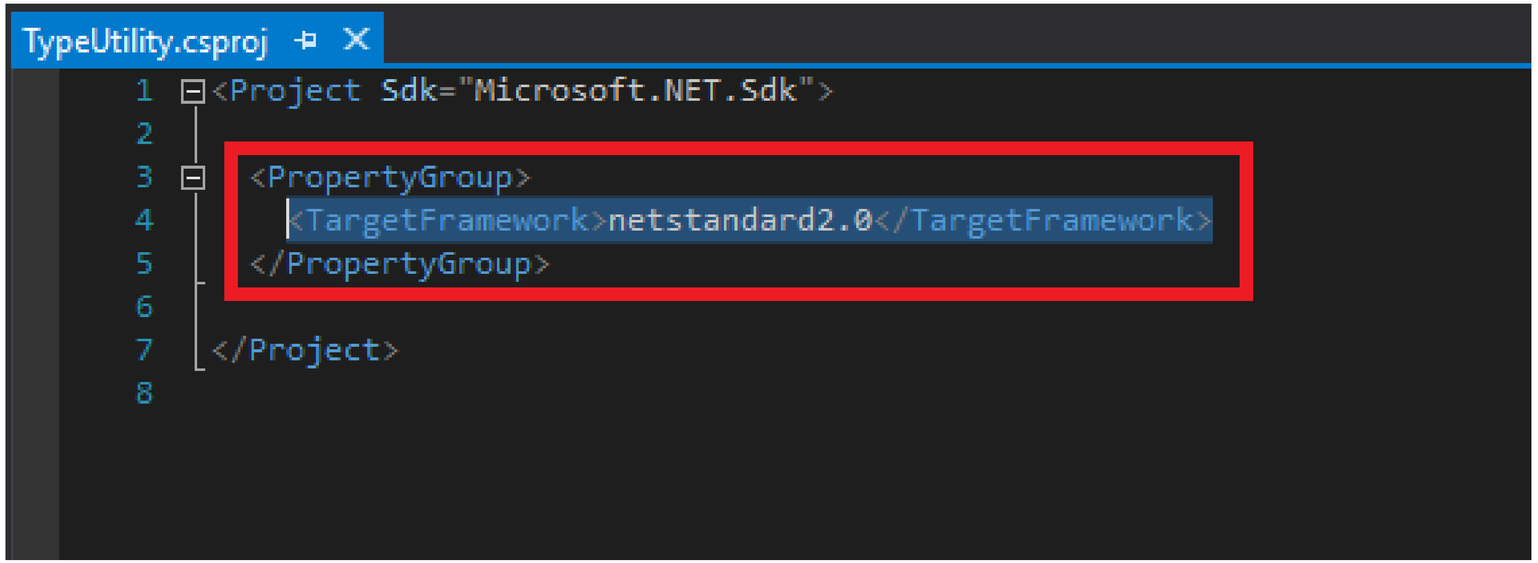
NO.

BOM, VAMOS LÁ!


New Project
Recent
Installed
Visual C#
Get Started
Windows Desktop
Web
.NET Core
.NET Standard
Cloud
Extensibility
Test
WCF
Visual Basic
Visual C++
Visual F#
SQL Server
Other Project Types
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Class Library (.NET Standard)
Visual C#
Search (Ctrl+E)
Type: Visual C#
A project for creating a class library that targets .NET Standard.
Name: TypeUtility
Location: C:\GIT\
Solution name: TypeUtility
Browse...
Create directory for solution
Create new Git repository
OK
Cancel

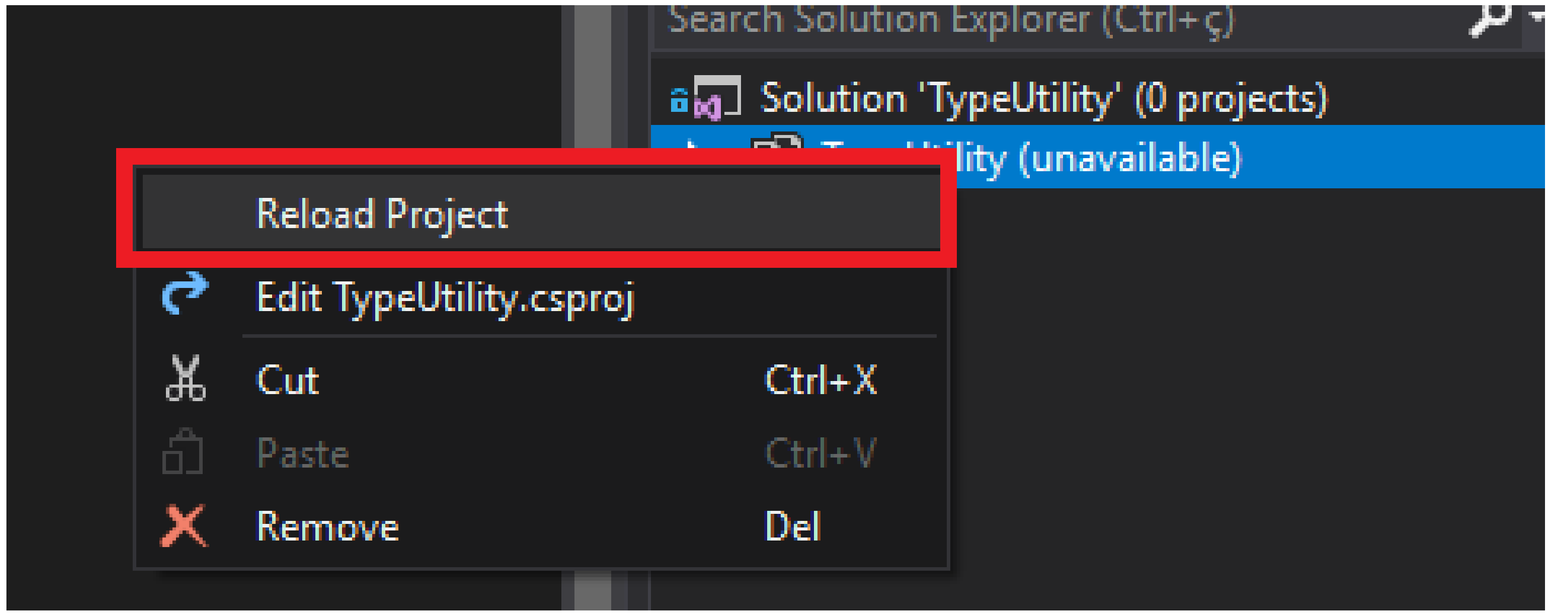

Solution 'TypeUtility' (1 project)
TypeUtility
Dependencies
Type.cs
Build
Rebuild
Clean
Analyze
Pack
Publish...
Scope to This
New Solution Explorer View
Show on Code Map
Edit TypeUtility.csproj
Add
Manage NuGet Packages...
Set as StartUp Project
Debug
Source Control
Cut Ctrl+X
Remove Del
Rename
Unload Project
Open Folder in File Explorer
Solution Explorer
Team Explorer
Project Properties
TypeUtility.csproj

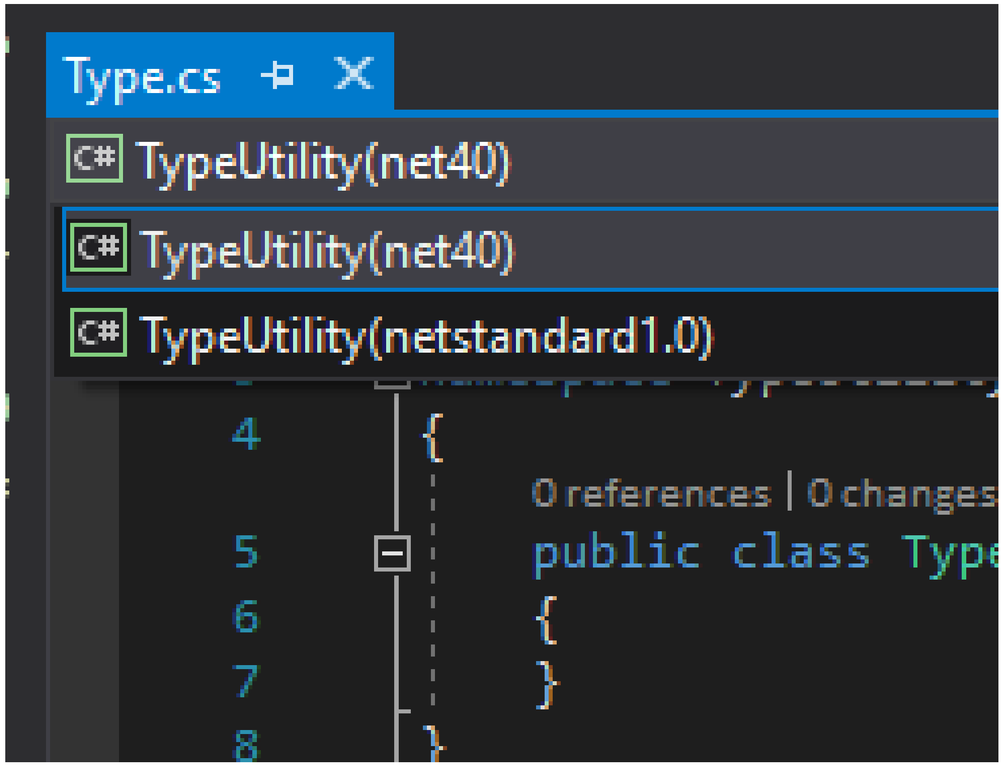

Search Solution Explorer (Ctrl+ç)
Solution 'TypeUtility' (0 projects)
TypeUtility (unavailable)
Reload Project
Edit TypeUtility.csproj
Cut Ctrl+X
Paste Ctrl+V
Remove Del

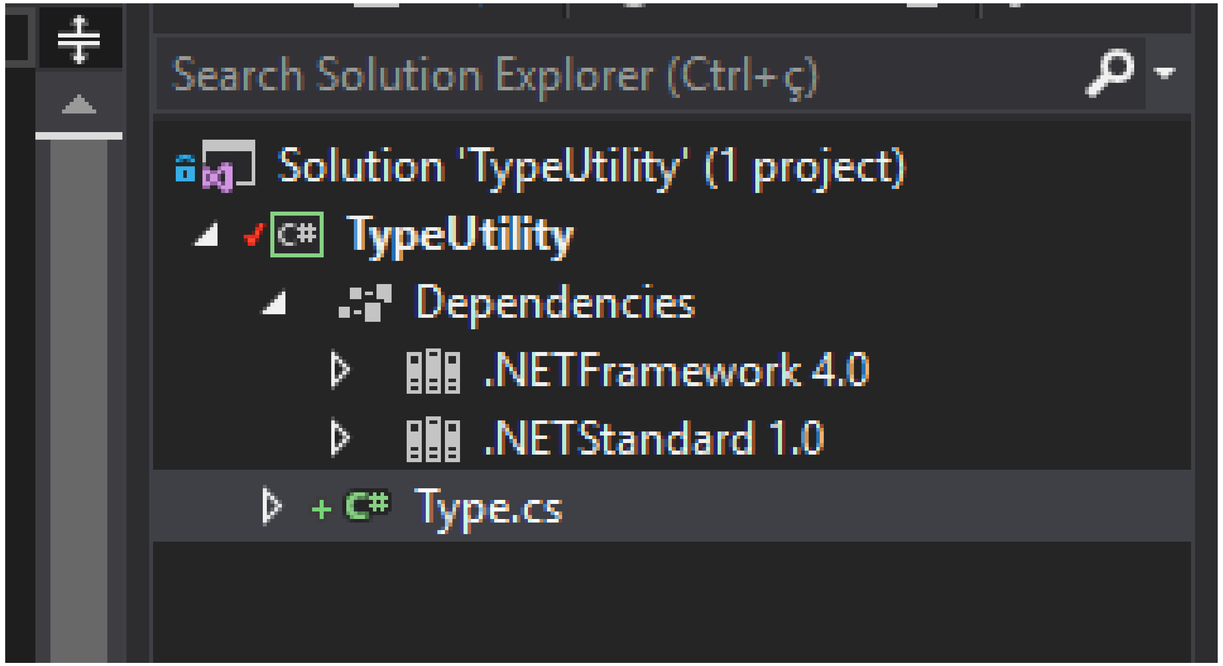

TypeUtility.csproj
1 <Project Sdk="Microsoft.NET.Sdk">
2
3 <PropertyGroup>
4 <TargetFramework>netstandard2.0</TargetFramework>
5 </PropertyGroup>
6
7 </Project>
8

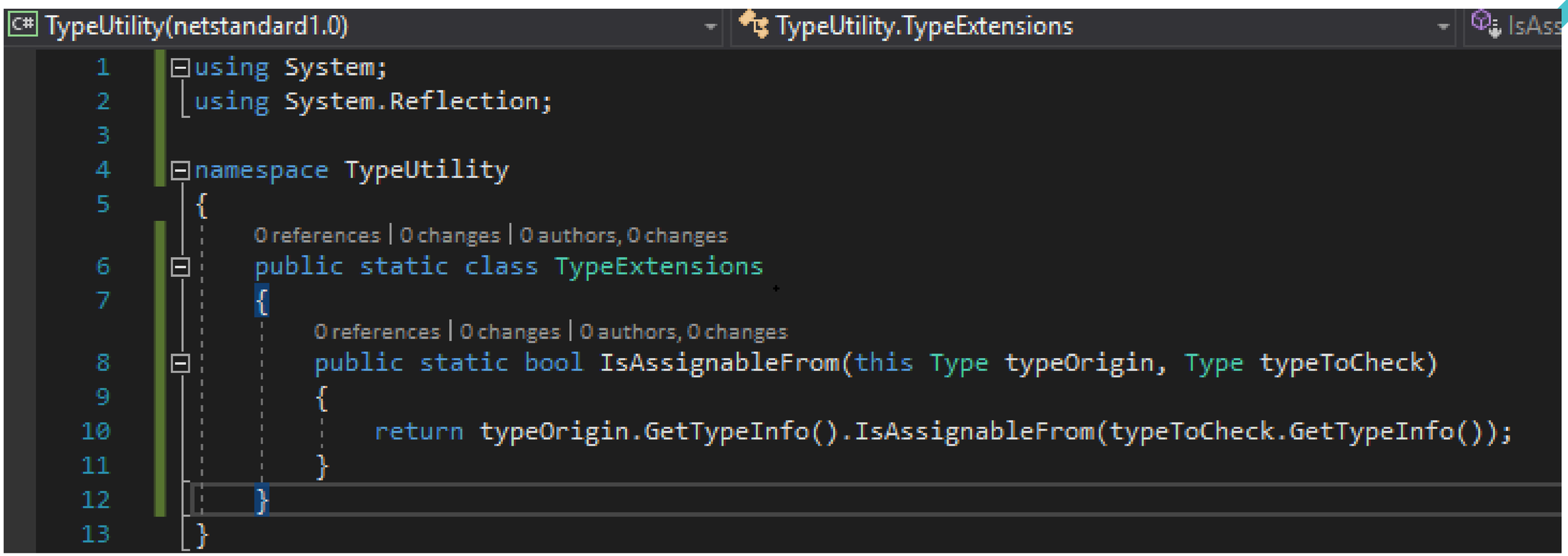
TypeUtility.csproj

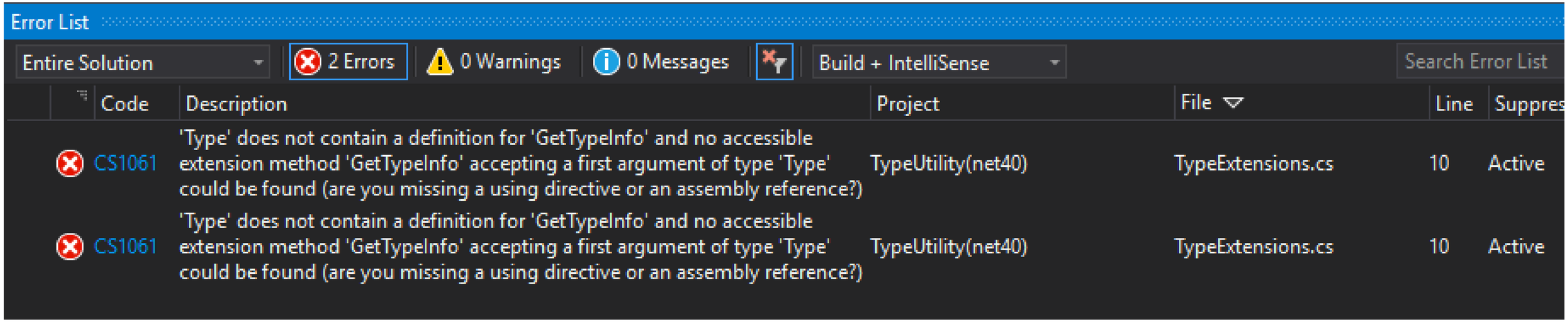
```
1  <Project Sdk="Microsoft.NET.Sdk">
2
3    <PropertyGroup>
4      <TargetFrameworks>net40;netstandard1.0</TargetFrameworks>
5    </PropertyGroup>
6
7  </Project>
8
```

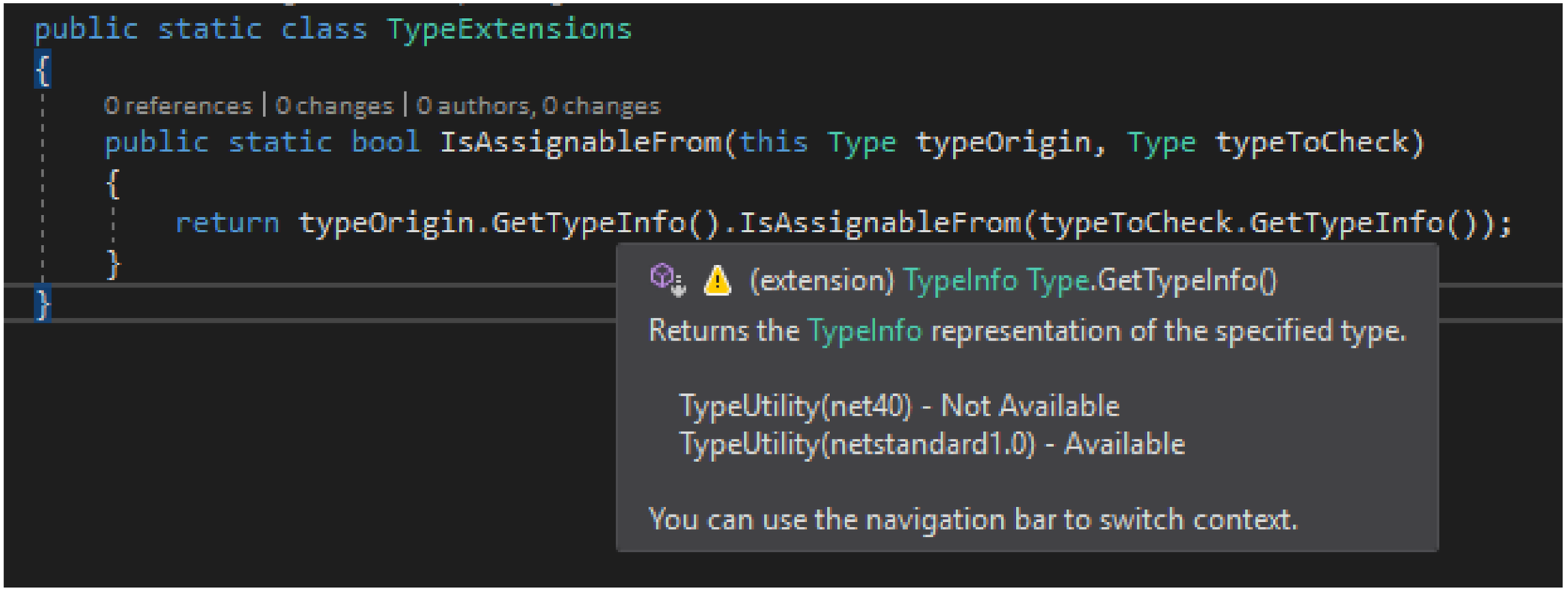



Type.cs
TypeUtility(net40)
TypeUtility(net40)
TypeUtility(netstandard1.0)
4
{
0 references | 0 changes
5
public class Type
6
{
7
}
8
}


Search Solution Explorer (Ctrl+ç)
Solution 'TypeUtility' (1 project)
TypeUtility
Dependencies
.NETFramework 4.0
.NETStandard 1.0
Type.cs


TypeUtility(netstandard1.0)
TypeUtility.TypeExtensions
using System;
using System.Reflection;
namespace TypeUtility
{
0 references | 0 changes | 0 authors, 0 changes
public static class TypeExtensions
{
0 references | 0 changes | 0 authors, 0 changes
public static bool IsAssignableFrom(this Type typeOrigin, Type typeToCheck)
{
return typeOrigin.GetTypeInfo().IsAssignableFrom(typeToCheck.GetTypeInfo());
}
}
}


Error List
Entire Solution
2 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Code
Description
Project
File
Line
Suppres
CS1061
'Type' does not contain a definition for 'GetTypeInfo' and no accessible extension method 'GetTypeInfo' accepting a first argument of type 'Type' could be found (are you missing a using directive or an assembly reference?)
TypeUtility(net40)
TypeExtensions.cs
10
Active
CS1061
'Type' does not contain a definition for 'GetTypeInfo' and no accessible extension method 'GetTypeInfo' accepting a first argument of type 'Type' could be found (are you missing a using directive or an assembly reference?)
TypeUtility(net40)
TypeExtensions.cs
10
Active


public static class TypeExtensions
{
0 references | 0 changes | 0 authors, 0 changes
public static bool IsAssignableFrom(this Type typeOrigin, Type typeToCheck)
{
return typeOrigin.GetTypeInfo().IsAssignableFrom(typeToCheck.GetTypeInfo());
}
}
(extension) TypeInfo Type.GetTypeInfo()
Returns the TypeInfo representation of the specified type.
TypeUtility(net40) - Not Available
TypeUtility(netstandard1.0) - Available
You can use the navigation bar to switch context.

```
public static class TypeExtensions
{
    0 references | 0 changes | 0 authors, 0 changes
    public static bool IsAssignableFrom(this Type typeOrigin, Type typeToCheck)
    {
        return typeOrigin.GetTypeInfo().IsAssignableFrom(typeToCheck.GetTypeInfo());
    }
}
```

(extension) TypeInfo Type.GetTypeInfo()

Returns the TypeInfo representation of the specified type.

TypeUtility(net40) - Not Available
TypeUtility(netstandard1.0) - Available

TypeUtility(net40) | TypeUtility.TypeExtensions | IsAssig

```
1   using System;
2   using System.Reflection;
3
4   namespace TypeUtility
5   {
        0 references | 0 changes | 0 authors, 0 changes
6       public static class TypeExtensions
7       {
            0 references | 0 changes | 0 authors, 0 changes
8           public static bool IsAssignableFrom(this Type typeOrigin, Type typeToCheck)
9           {
10              return typeOrigin.GetTypeInfo().IsAssignableFrom(typeToCheck.GetTypeInfo());
11          }
12      }
13  }
```

```
namespace TypeUtility
{
    0 references | 0 changes | 0 authors, 0 changes
    public static class TypeExtensions
    {
        0 references | 0 changes | 0 authors, 0 changes
        public static bool IsAssignableFrom(this Type typeOrigin, Type typeToCheck)
        {
#if NET40
            return typeOrigin.IsAssignableFrom(typeToCheck);
#else
            return typeOrigin.GetTypeInfo().IsAssignableFrom(typeToCheck.GetTypeInfo());
#endif
        }
    }
}
```

Output

Show output from: Build

```
1>------ Rebuild All started: Project: TypeUtility, Configuration: Debug Any CPU ------
1>TypeUtility -> C:\GIT\TypeUtility\TypeUtility\bin\Debug\net40\TypeUtility.dll
1>TypeUtility -> C:\GIT\TypeUtility\TypeUtility\bin\Debug\netstandard1.0\TypeUtility.dll
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

```
namespace TypeUtility
{
    0 references | 0 changes | 0 authors, 0 changes
    public static class TypeExtensions
    {
        0 references | 0 changes | 0 authors, 0 changes
        public static bool IsAssignableFrom(this Type typeOrigin, Type typeToCheck)
        {
#if NET40
            return typeOrigin.IsAssignableFrom(typeToCheck);
#else
            return typeOrigin.GetTypeInfo().IsAssignableFrom(typeToCheck.GetTypeInfo());
#endif
        }
    }
}
```

Output

Show output from: Build

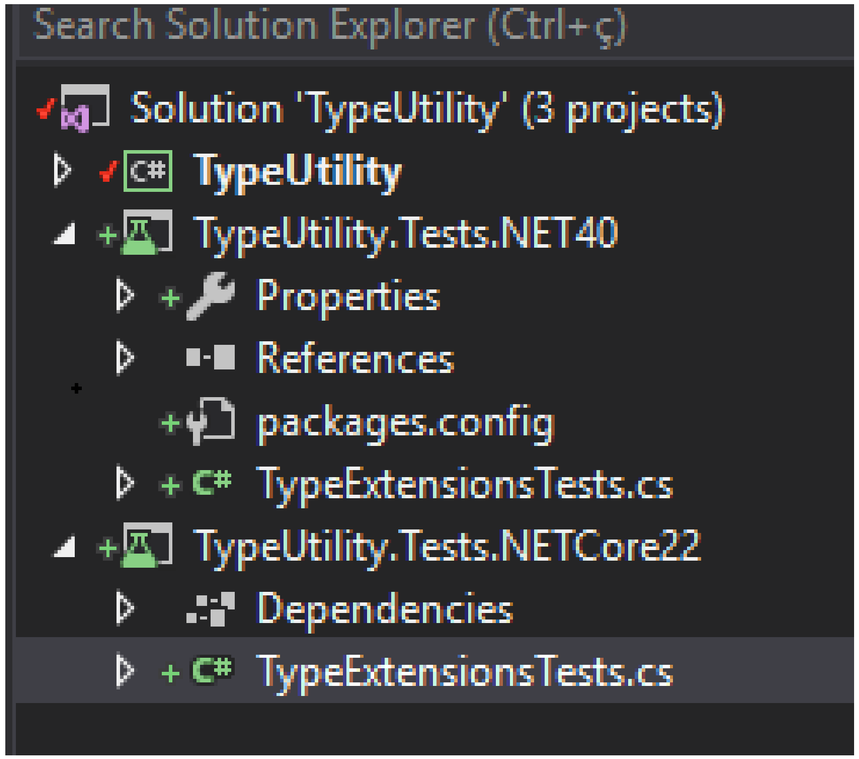
```
1>------ Rebuild All started: Project: TypeUtility, Configuration: Debug Any CPU ------
1>TypeUtility -> C:\GIT\TypeUtility\TypeUtility\bin\Debug\net40\TypeUtility.dll
1>TypeUtility -> C:\GIT\TypeUtility\TypeUtility\bin\Debug\netstandard1.0\TypeUtility.dll
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

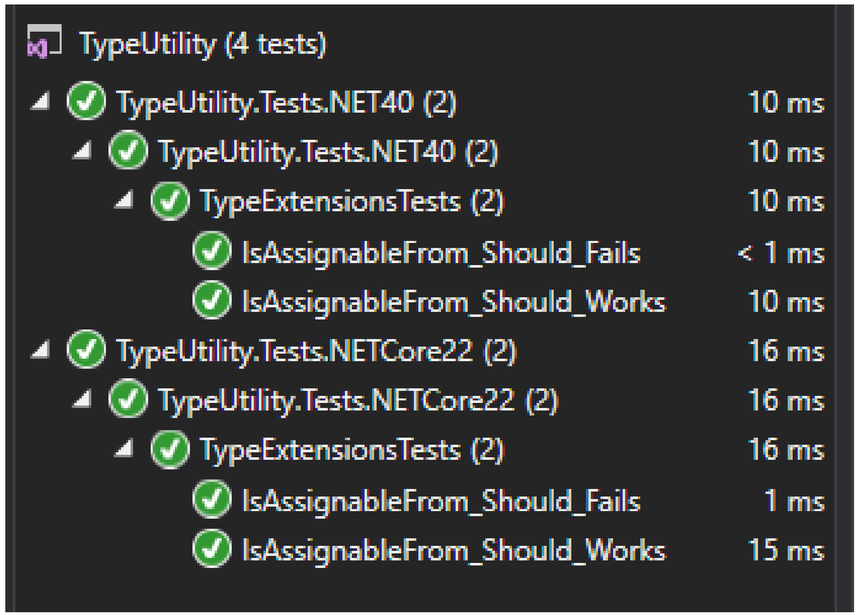
```
namespace TypeUtility
{
    0 references | 0 changes | 0 authors, 0 changes
    public static class TypeExtensions
    {
        0 references | 0 changes | 0 authors, 0 changes
        public static bool IsAssignableFrom(this Type typeOrigin, Type typeToCheck)
        {
#if NET40
            return typeOrigin.IsAssignableFrom(typeToCheck);
#else
            return typeOrigin.GetTypeInfo().IsAssignableFrom(typeToCheck.GetTypeInfo());
#endif
        }
    }
}
```

Output

Show output from: Build

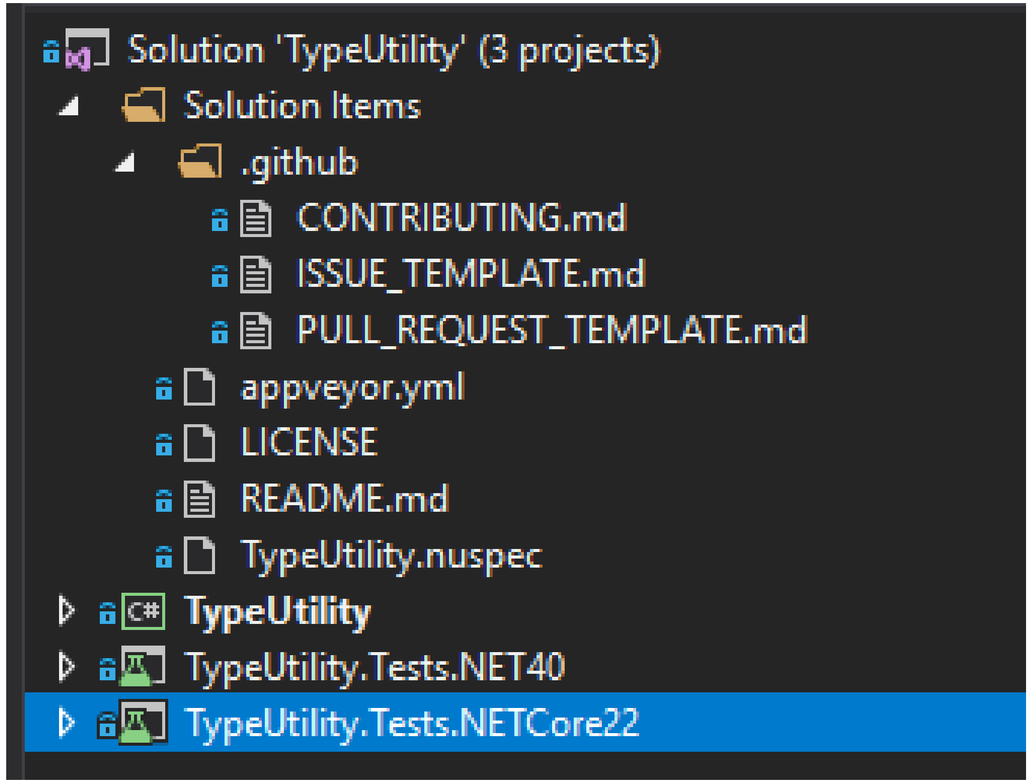
```
1>------ Rebuild All started: Project: TypeUtility, Configuration: Debug Any CPU ------
1>TypeUtility -> C:\GIT\TypeUtility\TypeUtility\bin\Debug\net40\TypeUtility.dll
1>TypeUtility -> C:\GIT\TypeUtility\TypeUtility\bin\Debug\netstandard1.0\TypeUtility.dll
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

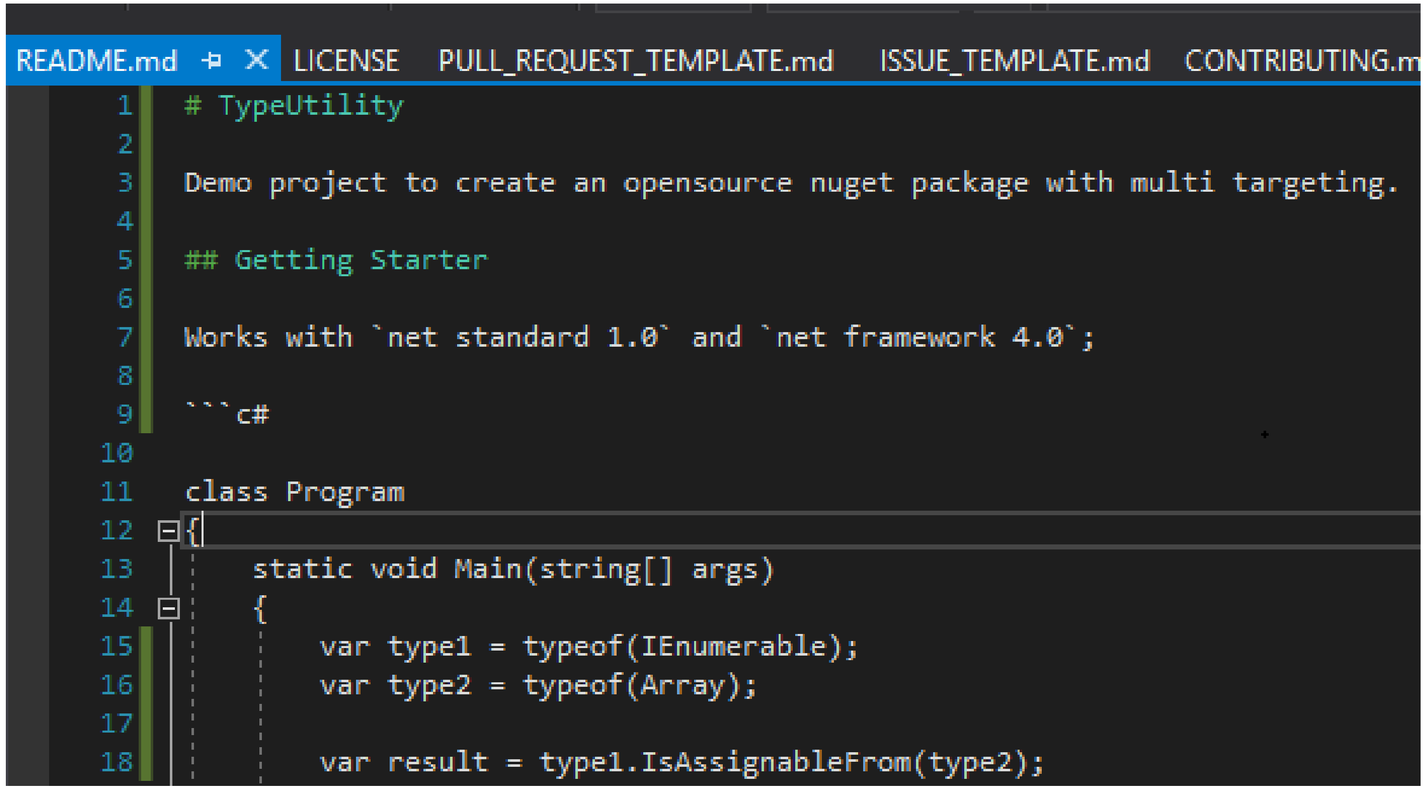





```
[Fact]
0 references | 0 changes | 0 authors, 0 changes
public void IsAssignableFrom_Should_Works()
{
    // arrange
    var type1 = typeof(IEnumerable);
    var type2 = typeof(Array);

    // act
    var result = type1.IsAssignableFrom(type2);

    // assert
    Assert.True(result);
}
```

```
[TestMethod]
0 references | 0 changes | 0 authors, 0 changes
public void IsAssignableFrom_Should_Works()
{
    // arrange
    var type1 = typeof(IEnumerable);
    var type2 = typeof(Array);

    // act
    var result = type1.IsAssignableFrom(type2);

    // assert
    Assert.IsTrue(result);
}
```


Solution 'TypeUtility' (3 projects)
Solution Items
.github
CONTRIBUTING.md
ISSUE_TEMPLATE.md
PULL_REQUEST_TEMPLATE.md
appveyor.yml
LICENSE
README.md
TypeUtility.nuspec
TypeUtility
TypeUtility.Tests.NET40
TypeUtility.Tests.NETCore22


README.md
LICENSE
PULL_REQUEST_TEMPLATE.md
ISSUE_TEMPLATE.md
CONTRIBUTING.m
TypeUtility
Demo project to create an opensource nuget package with multi targeting.
Getting Starter
Works with `net standard 1.0` and `net framework 4.0`;
```c#
class Program
{
static void Main(string[] args)
{
var type1 = typeof(IEnumerable);
var type2 = typeof(Array);
var result = type1.IsAssignableFrom(type2);
```

TypeUtility.nuspec

```
<?xml version="1.0"?>
<package >
  <metadata>
    <id>TypeUtility</id>
    <version>__version__</version>
    <title>TypeUtility</title>
    <authors>Thiago Barradas</authors>
    <owners>Thiago Barradas</owners>
    <projectUrl>https://github.com/ThiagoBarradas/type-extensions</projectUrl>
    <licenseUrl>https://github.com/ThiagoBarradas/type-extensions/blob/master/LICENSE</licenseUrl>
    <iconUrl>https://i.imgur.com/P7aDP0E.png</iconUrl>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <description>Extensions for .NET "Type";</description>
    <releaseNotes>
      See changelog in https://github.com/thiagobarradas/type-extensions/releases/tag/__version__
    </releaseNotes>
    <copyright>2019</copyright>
    <tags>type, extension, demo, talk, sample</tags>
  </metadata>
  <files>
    <file src="TypeUtility\bin\Release\net40\TypeUtility.dll" target="lib\net40" />
    <file src="TypeUtility\bin\Release\netstandard1.0\TypeUtility.dll" target="lib\netstandard1.0" />
  </files>
</package>
```

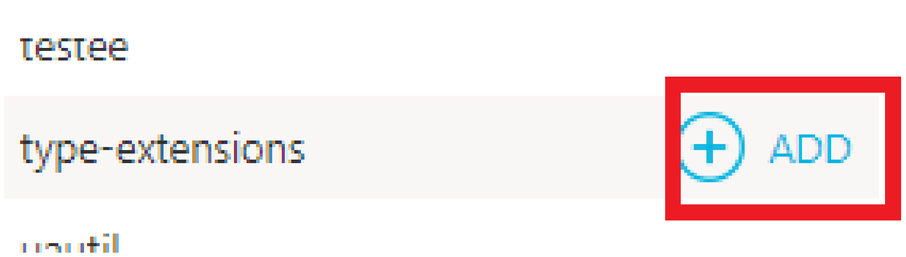
TypeUtility.nuspec

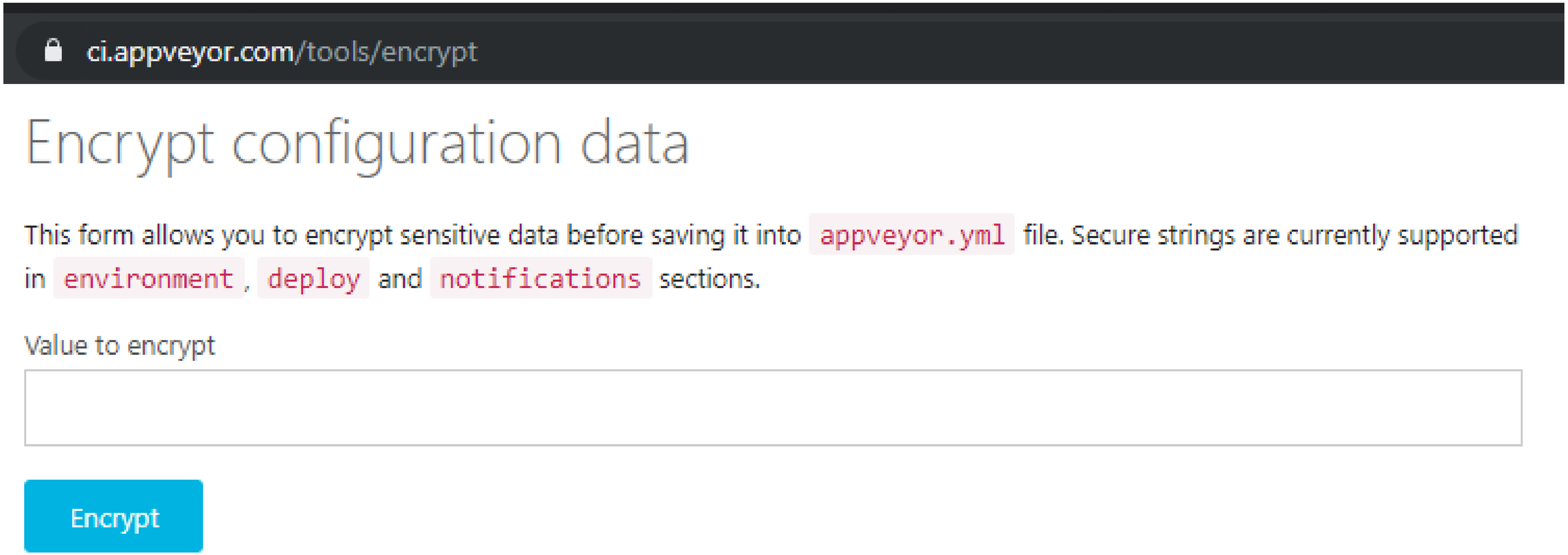
```
<?xml version="1.0"?>
<package >
  <metadata>
    <id>TypeUtility</id>
    <version>__version__</version>
    <title>TypeUtility</title>
    <authors>Thiago Barradas</authors>
    <owners>Thiago Barradas</owners>
    <projectUrl>https://github.com/ThiagoBarradas/type-extensions</projectUrl>
    <licenseUrl>https://github.com/ThiagoBarradas/type-extensions/blob/master/LICENSE</licenseUrl>
    <iconUrl>https://i.imgur.com/P7aDP0E.png</iconUrl>
    <requireLicenseAcceptance>false</requireLicenseAcceptance>
    <description>Extensions for .NET "Type";</description>
    <releaseNotes>
      See changelog in https://github.com/thiagobarradas/type-extensions/releases/tag/__version__
    </releaseNotes>
    <copyright>2019</copyright>
    <tags>type, extension, demo, talk, sample</tags>
  </metadata>
  <files>
    <file src="TypeUtility\bin\Release\net40\TypeUtility.dll" target="lib\net40" />
    <file src="TypeUtility\bin\Release\netstandard1.0\TypeUtility.dll" target="lib\netstandard1.0" />
  </files>
</package>
```

appveyor.yml

```yaml
image: Visual Studio 2017
platform: Any CPU

environment:
  version: $(APPVEYOR_BUILD_VERSION)

configuration:
- Release

before_build:
  - nuget restore

build:
  project: TypeUtility.sln

after_test:
  - ps: (Get-Content TypeUtility.nuspec) | ForEach-Object { $_ -replace "__version__", "$env:version" } | Set-Content Typ
  - nuget pack TypeUtility.compiled.nuspec -Prop Configuration=Release

artifacts:
  - path: TypeUtility\bin\Release\netstandard1.0\TypeUtility.dll
  - path: TypeUtility\bin\Release\net40\TypeUtility.dll
  - path: TypeUtility.%version%.nupkg

deploy:
```


ThiagoBarradas
Projects
Environments
Self-host
Projects
NEW PROJECT


testee
type-extensions
ADD
uutil


ci.appveyor.com/tools/encrypt
Encrypt configuration data
This form allows you to encrypt sensitive data before saving it into appveyor.yml file. Secure strings are currently supported in environment, deploy and notifications sections.
Value to encrypt
Encrypt

ThiagoBarradas Projects Environments Self-hosted jobs NEW Account Support Thiago Barradas

TypeExtensions Library UTILS

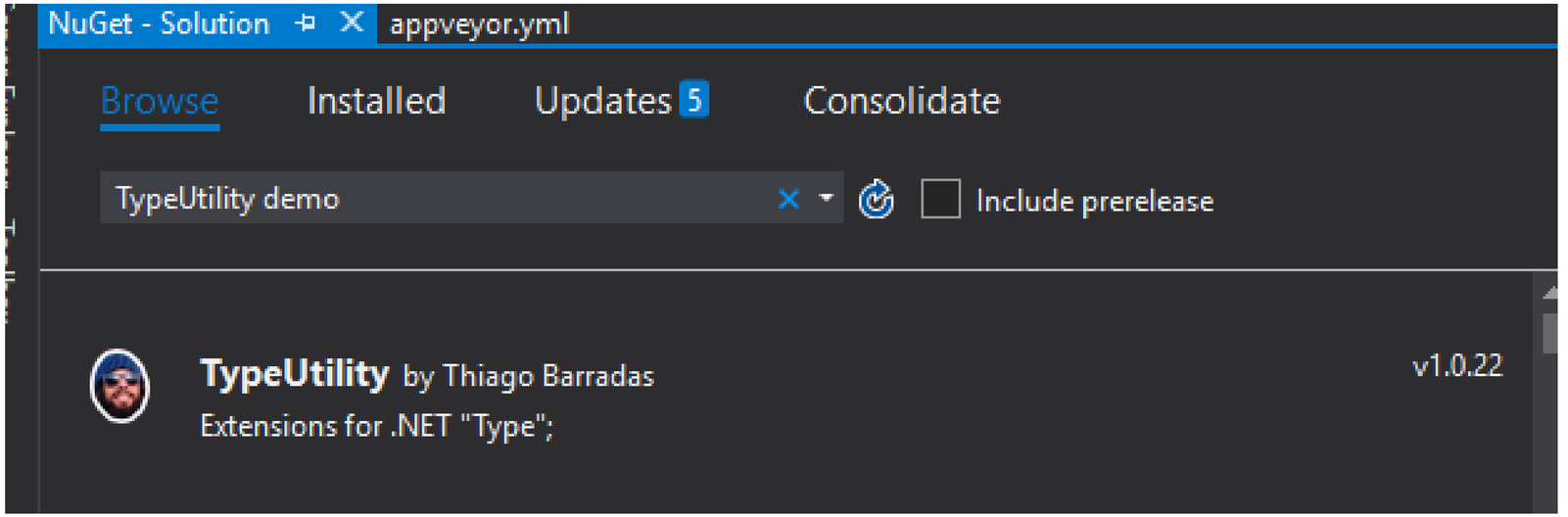
Current build History Deployments Events Settings NEW BUILD RE-BUILD COMMIT DEPLOY LO

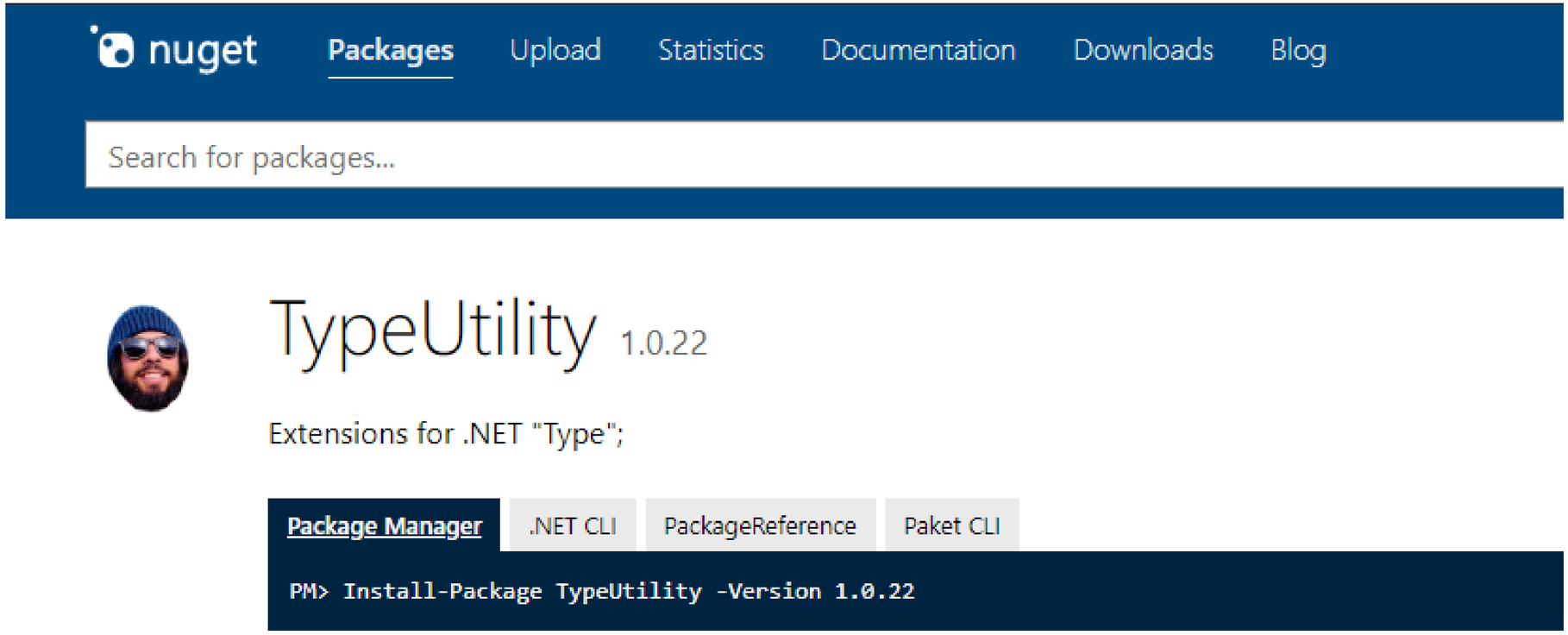
first commit 1.0.2

36 minutes ago by Thiago Barradas master 6bb6d12d 35 minutes ago in 32 s

Console Messages Tests 4 Artifacts 3 Events

```
1  Build started
2  git clone -q --branch=master https://github.com/ThiagoBarradas/type-extensions.git C:\projects\type-extensions
3  git checkout -qf 6bb6d12d52682a766f61cdcc8e29e3d60ceabbf2
4  nuget restore
5  MSBuild auto-detection: using msbuild version '15.9.21.664' from 'C:\Program Files (x86)\Microsoft Visual
   Studio\2017\Community\MSBuild\15.0\bin'.
6  Restoring NuGet package MSTest.TestFramework.1.3.2.
7  Restoring NuGet package MSTest.TestAdapter.1.3.2.
8  Adding package 'MSTest.TestFramework.1.3.2' to folder 'C:\projects\type-extensions\packages'
9  Adding package 'MSTest.TestAdapter.1.3.2' to folder 'C:\projects\type-extensions\packages'
10 Added package 'MSTest.TestAdapter.1.3.2' to folder 'C:\projects\type-extensions\packages'
```


NuGet - Solution
appveyor.yml
Browse
Installed
Updates 5
Consolidate
TypeUtility demo
Include prerelease
TypeUtility by Thiago Barradas
Extensions for .NET "Type";
v1.0.22


nuget
Packages
Upload
Statistics
Documentation
Downloads
Blog
Search for packages...
TypeUtility 1.0.22
Extensions for .NET "Type";
Package Manager
.NET CLI
PackageReference
Paket CLI
PM> Install-Package TypeUtility -Version 1.0.22

README.md

build passing

TypeUtility

Demo project to create an opensource nuget package with multi targeting.

Getting Starter

Works with `net standard 1.0` and `net framework 4.0`;

```
class Program
{
    static void Main(string[] args)
    {
        var type1 = typeof(IEnumerable);
        var type2 = typeof(Array);
```

Links:

https://github.com/ThiagoBarradas/type-extensions

https://github.com/ThiagoBarradas/jsonmasking

https://tinyurl.com/artigo-appveyor

https://docs.microsoft.com/pt-br/dotnet/standard/frameworks

https://docs.microsoft.com/pt-br/dotnet/standard/net-standard

Thiago Barradas

tbarradas@mundipagg.com

+55 (21) 99329-9143

Linkedin: **thiagobarradas**

Obrigado!