Announcing .NET 5.0 RC 2

Richard
October 13th, 2020

Today, we are shipping [.NET 5.0 Release Candidate 2 (RC2)](). It is a near-final release of .NET 5.0, and the last of two RCs before the official release in November. RC2 is a "go live" release; you are supported using it in production. At this point, we're looking for reports of any remaining critical bugs that should be fixed before the final release.

We also released new versions of ASP.NET Core and EF Core today.

You can download .NET 5.0, for Windows, macOS, and Linux:

- Installers and binaries
- Container images
- Snap installer
- Release notes
- Known issues
- GitHub issue tracker

You need the latest preview version of Visual Studio (including Visual Studio for Mac) to use .NET 5.0.

.NET 5.0 includes many improvements, notably single file applications, smaller container images, more capable JsonSerializer APIs, a complete set of nullable reference type annotations, new target framework names, and support for Windows ARM64. Performance has been greatly improved, in the NET libraries, in the GC, and the JIT. ARM64 was a key focus for performance investment, resulting in much better throughput and smaller binaries. .NET 5.0 includes new language versions, C# 9 and F# 5.0. Check out some .NET 5.0 examples so you can try these features out for yourself.

Today is an auspicious day because we're kicking off the 2020 .NET@Microsoft internal conference. There will be many speakers from the .NET team, but also developers and architects from services teams that rely on .NET to power the Microsoft cloud, sharing their victories and also their challenges. I'm presenting (unsurprisingly) "What's new in .NET 5.0". My talk will be easy; I'll just read the .NET 5.0 blog posts, preview by preview! It will be a great talk. More seriously, the conference is our opportunity to make the case why Microsoft teams should adopt .NET 5.0 soon after it is available. At least one large team I know of is running on RC1 in production. The official .NET Microsoft site has been running on .NET 5.0 since Preview 1. It is now running RC2. The case we'll make to Microsoft teams this week is very similar to the case that I've intended to make to you across all of these .NET 5.0 blog posts. .NET 5.0 is a great release and will improve the fundamentals of your app.

Speaking of conferences, please save the date for .NET Conf 2020. This year, .NET 5.0 will launch at .NET Conf 2020! Come celebrate and learn about the new release. We're also celebrating our 10th anniversary and we're working on a few more surprises. You won't want to miss this one.

Just like I did for .NET 5.0 Preview 8 and .NET 5.0 RC1, I've chosen a selection of features to look at in more depth and to give you a sense of how you'll use them in real-world usage. This post is dedicated to C# 9 pattern matching, Windows ARM64, and

ClickOnce.

C# 9 Pattern Matching

Pattern matching is a language feature was first added in C# 7.0. It's best to let Mads reintroduce the concept. This is what he had to say when he originally introduced the feature.

> C# 7.0 introduces the notion of *patterns*, which, abstractly speaking, are syntactic elements that can test that a value has a certain "shape", and extract information from the value when it does.

That's a really great description, perfectly worded.

The C# team has added new patterns in each of the C# 7, C# 8, and C# 9 versions. In this post, you'll see patterns from each of those language versions, but we'll focus on the new patterns in C# 9.

The three new patterns in C# 9 are:

- Relational patterns, using relational operators such as `<` and `>=`.
- Logical patterns, using the keywords `and`, `or`, and `not`. The poster child example is `foo is not null`. This type of pattern is most useful when you want to compare multiple things in one pattern.
- Simple type patterns, using solely a type and no other syntax for matching.

I'm a big fan of the BBC Sherlock series. I've written a small app that determines if a given character should have access to a given piece of content within that series. Easy enough. The app is written with two constraints: stay true to the show timeline and characters, and be a great demonstration of patterns. If anything, I suspect I've failed most on the second constraint. You'll find a broader set of patterns and styles than one would expect in a given app (particularly such a small one).

When I'm using patterns, I sometimes want to do something subtly different than a pattern I'm familiar with achieves and am not sure how to extend that pattern to satisfy my goal. Given this sample, I'm hoping you'll discover more approaches than perhaps you were aware of before, and can extend your repertoire of familiar patterns.

There are two switch expressions within the app. Let's start with the smaller of the two.

```
public static bool IsAccessOKAskMycroft(Person person) => person switch
{
    // Type pattern
    OpenCaseFile f when f.Name == "Jim Moriarty"    => true,
    // Simple type pattern
    Mycroft                                         => true,
    _                                               => false,
};
```

The first two patterns are type patterns. The first pattern is supported with C# 8. The second one — `Mycroft` — is an example of the new simple type pattern. With C# 8, this pattern would require an identifier, much like the first pattern, or at the very least a discard such as `Mycroft _`. In C# 9, the identifier is no longer needed. Yes, `Mycroft` is a type in the app.

Let's keep to simple a little longer, before I show you the other switch expression. The following `if` statement demonstrates a logical pattern, preceded by two instances of a type pattern using `is`.

```
if (user is Mycroft m && m.CaresAbout is not object)
{
    Console.WriteLine("Mycroft dissapoints us again.");
}
```

The type isn't known, so the `user` variable is tested for the `Mycroft` type and is then assigned to `m` if that test passes. A property on the `Mycroft` object is tested to be `not` an object. A test for `null` would have also worked, but wouldn't have demonstrated a logical pattern.

The other switch expression is a lot more expansive.

```
    public static bool IsAccessOkOfficial(Person user, Content content, int
 season) => (user, content, season) switch
    {
        // Tuple + property patterns
        ({Type: Child}, {Type: ChildsPlay}, _)          => true,
        ({Type: Child}, _, _)                           => false,
        (_ , {Type: Public}, _)                         => true,
        ({Type: Monarch}, {Type: ForHerEyesOnly}, _)    => true,
        // Tuple + type patterns
        (OpenCaseFile f, {Type: ChildsPlay}, 4) when f.Name == "Sherlock
 Holmes"  => true,
        // Property and type patterns
        {Item1: OpenCaseFile {Type: var type}, Item2: {Name: var name}}
            when type == PoorlyDefined && name.Contains("Sherrinford") &&
 season >= 3 => true,
        // Tuple and type patterns
        (OpenCaseFile, var c, 4) when c.Name.Contains("Sherrinford")
 => true,
        // Tuple, Type, Property and logical patterns
        (OpenCaseFile {RiskLevel: >50 and <100 }, {Type: StateSecret}, 3) =>
 true,
        _                                               => false,
    };
```

The only really interesting pattern is the very last one (before the discard: `-`), which tests for a `Risklevel` that is `>50 and <100`. There are many times I've wanted to write an `if` statement with that form of logical pattern syntax without needing to repeat a variable name. This logical pattern could also have been written in the following way instead and would have more closely matched the syntax demonstrated in the C# 9 blog post. They are equivalent.

```
        (OpenCaseFile {RiskLevel: var riskLevel}, {Type: StateSecret}, 3) when
 riskLevel switch
            {
                >50 and <100        => true,
                _                   => false
            }                                           => true,
```

I'm far from a language expert. Jared Parsons and Andy Gocke gave me a lot of help with this section of the post. Thanks! The key stumbling block I had was with a switch on a tuple. At times, the positional pattern is inconvenient, and you only want to address one part of the tuple. That's where the property pattern comes in, as you can see in the following code.

```
 {Item1: OpenCaseFile {Type: var type}, Item2: {Name: var name}}
            when type == PoorlyDefined && name.Contains("Sherrinford") &&
 season >= 3 => true,
```

There is a fair bit going on there. The key point is that the tuple properties are being tested, as opposed to matching a tuple positionally. That approaches provides a lot more flexibility. You are free to intermix these approaches within a given switch expression. Hopefully that helps someone. It helped me.

If you are curious about what the app does, I've saved the output of the program in the app gist. You can also run the app for yourself. I believe it requires .NET 5.0 RC2 to run.

If there has been a pattern with the last three C# (major) versions, it has been patterns. I certainly hope the C# team matches that pattern going forward. I imagine it is the shape of things, and there are certainly more values to extract.

ClickOnce

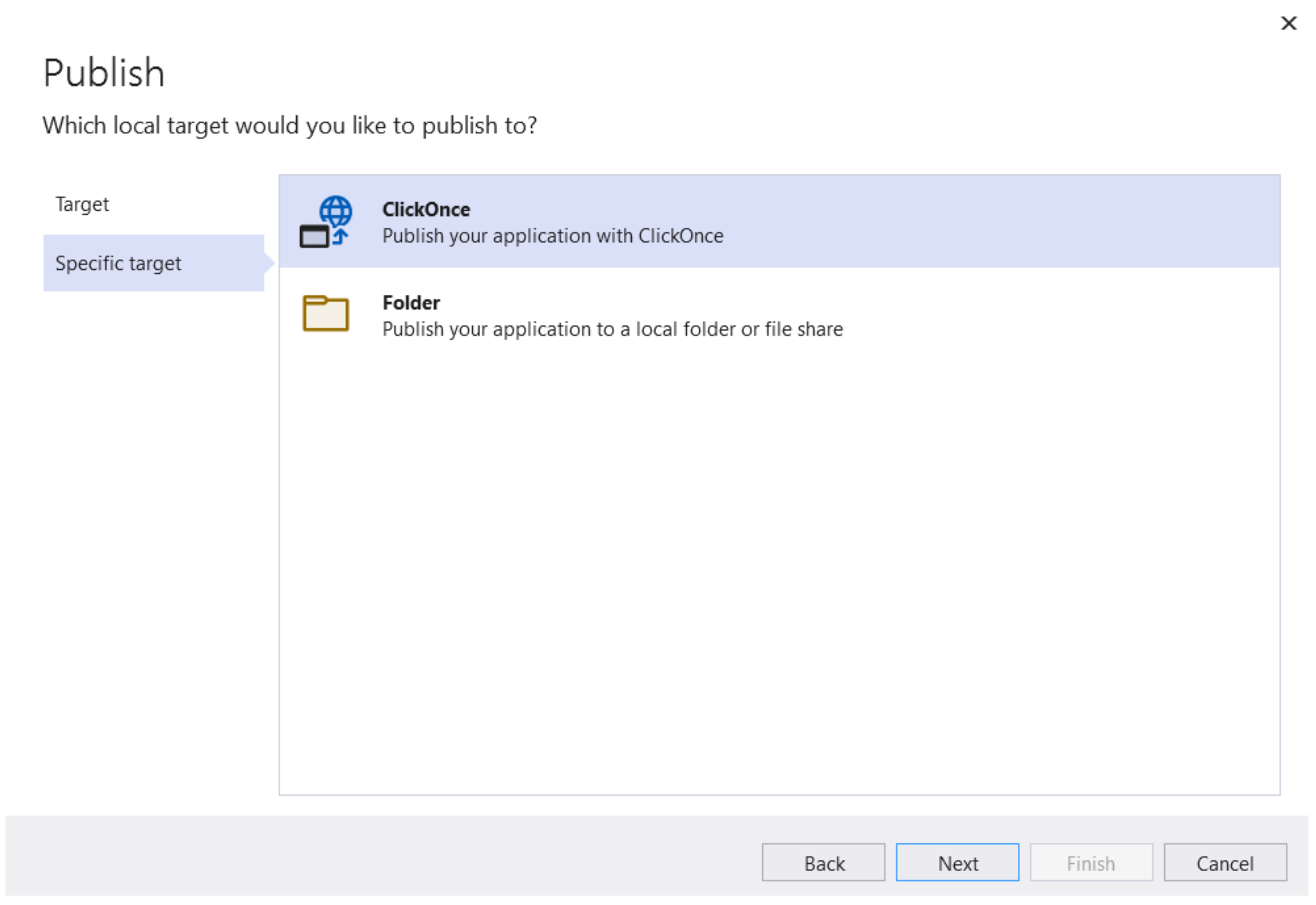
ClickOnce has been a popular .NET deployment option for many years. It's now supported for .NET Core 3.1 and .NET 5.0 Windows apps. We knew that many people would want to use ClickOnce for application deployment when we added Windows Forms and WPF support to .NET Core 3.0. In the past year, the .NET and Visual Studio teams worked together to enable ClickOnce publishing, both at the command line and in Visual Studio.

We had two goals from the start of the project:

- Enable a familiar experience for ClickOnce in Visual Studio.
- Enable a modern CI/CD for ClickOnce publishing with command-line flows, with either MSBuild or the Mage tool.

It's easiest to show you the experience in pictures.

Let's start with the Visual Studio experience, which is centered around project publishing.

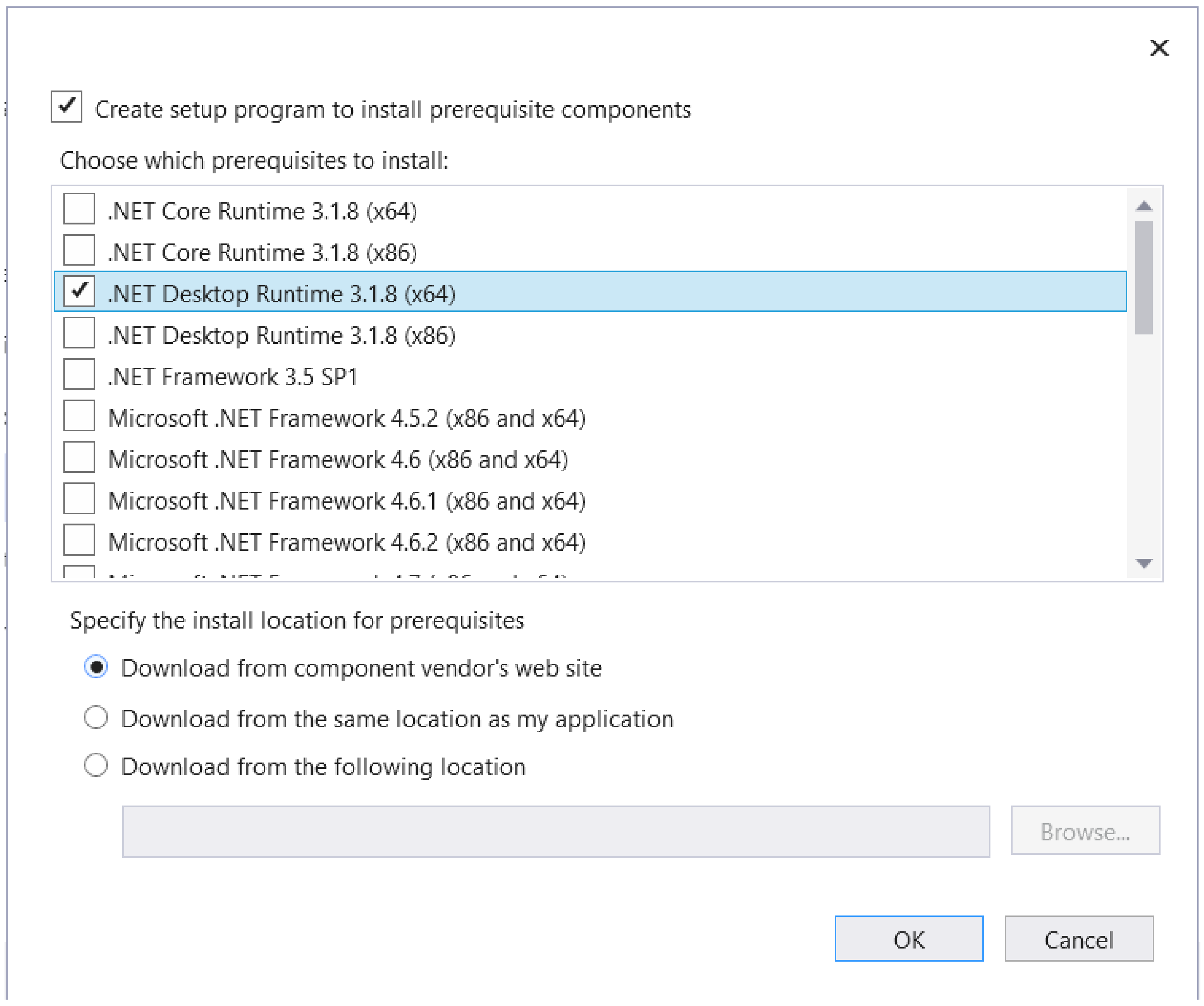

Publish

Which local target would you like to publish to?

ClickOnce
Publish your application with ClickOnce

Folder
Publish your application to a local folder or file share

The primary deployment model we're currently supporting is framework dependent apps. It is easy to take a dependency on the .NET Desktop Runtime (that's the one that contains WPF and Windows Forms). Your ClickOnce installer will install the .NET runtime on user machines if it is needed. We also intend to support self-contained and single file apps.



[✓] Create setup program to install prerequisite components

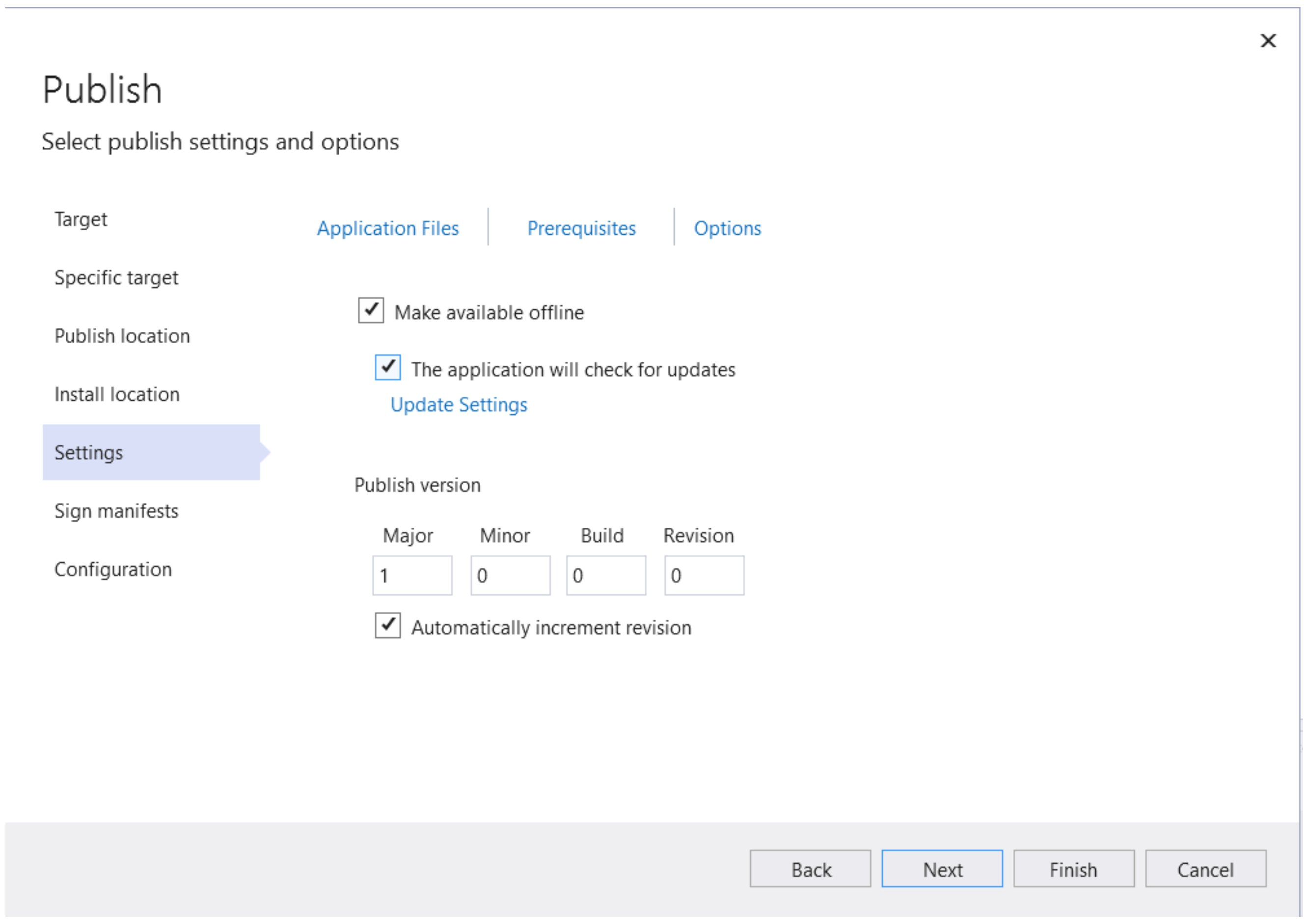
Choose which prerequisites to install:

- [ ] .NET Core Runtime 3.1.8 (x64)
- [ ] .NET Core Runtime 3.1.8 (x86)
- [✓] .NET Desktop Runtime 3.1.8 (x64)
- [ ] .NET Desktop Runtime 3.1.8 (x86)
- [ ] .NET Framework 3.5 SP1
- [ ] Microsoft .NET Framework 4.5.2 (x86 and x64)
- [ ] Microsoft .NET Framework 4.6 (x86 and x64)
- [ ] Microsoft .NET Framework 4.6.1 (x86 and x64)
- [ ] Microsoft .NET Framework 4.6.2 (x86 and x64)

Specify the install location for prerequisites

- ● Download from component vendor's web site
- ○ Download from the same location as my application
- ○ Download from the following location

You might wonder if you can still be able to take advantage of ClickOnce offline and updating features. Yes, you can.



Publish

Select publish settings and options

[✓] Make available offline

[✓] The application will check for updates

Publish version

Major	Minor	Build	Revision
1	0	0	0

[✓] Automatically increment revision

The same install locations and manifest signing features are included. If you have strict signing requirements, you will be covered with this new experience.

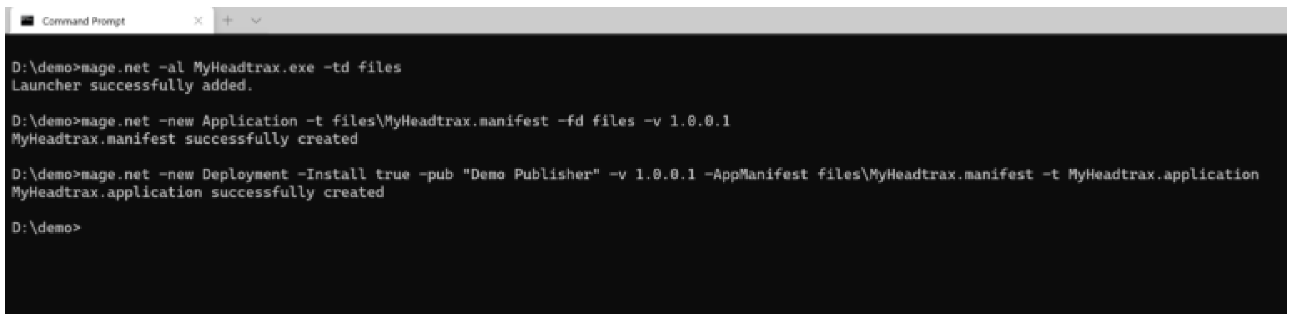
Now, let’s switch to the command line Mage experience.

The big change with Mage is that it is now a .NET tool, distributed on NuGet. That means you don’t need to install anything special on your machine. You just need the .NET 5.0 SDK and then you can install Mage as a .NET tool. You can use it to publish .NET Framework apps as well, however, SHA1 signing and partial trust support have been removed.

The Mage installation command follows:

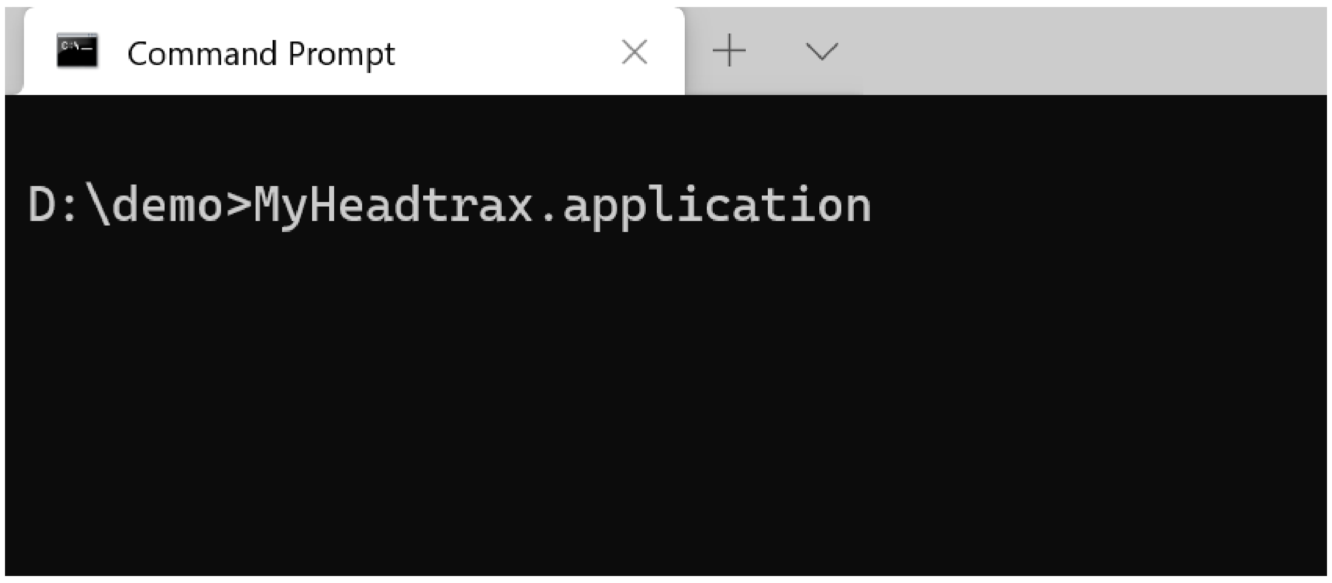
```
dotnet tool install -g Microsoft.DotNet.Mage
```

The following commands configure and publish a sample application.



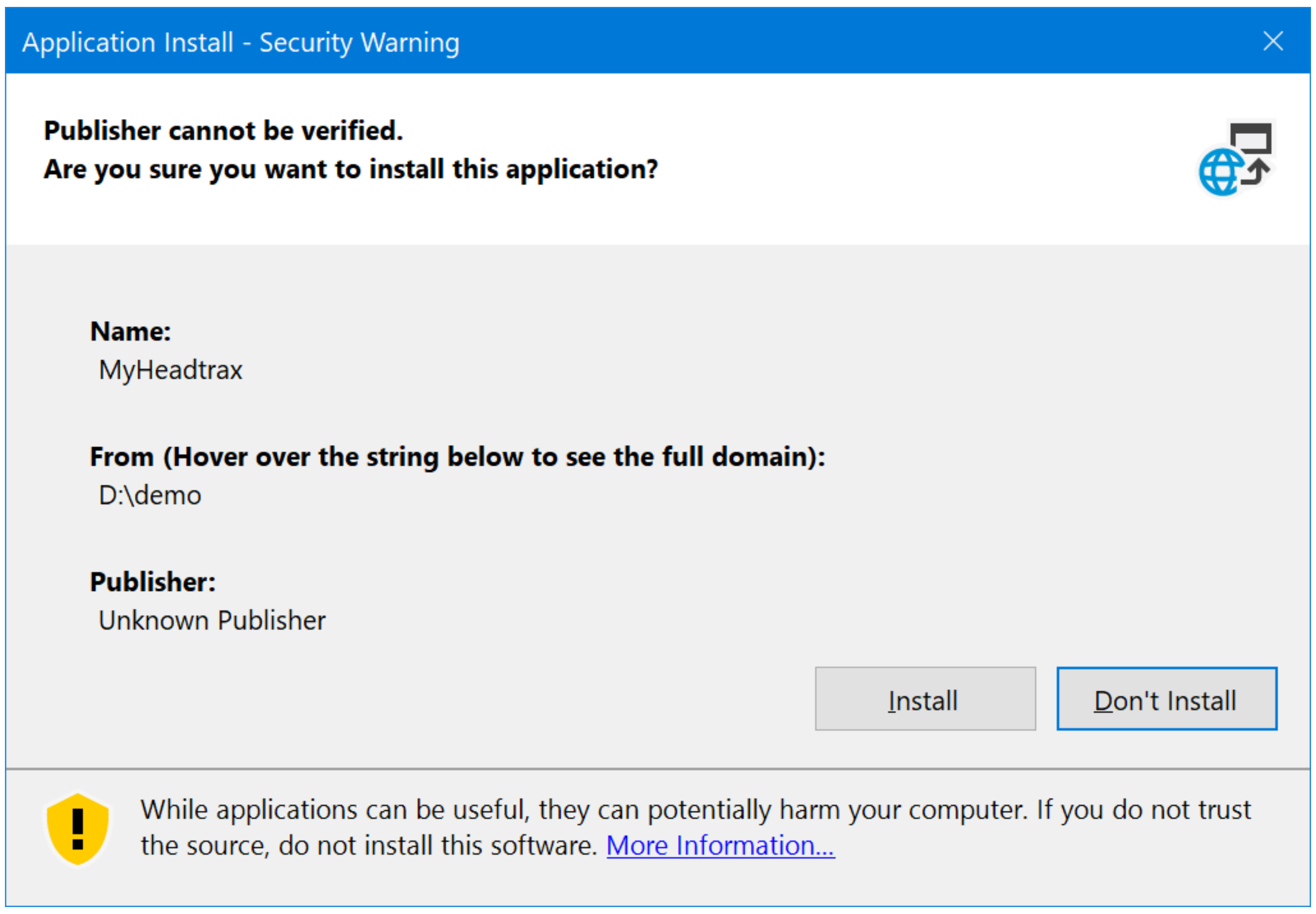
```
D:\demo>mage.net -al MyHeadtrax.exe -td files
Launcher successfully added.

D:\demo>mage.net -new Application -t files\MyHeadtrax.manifest -fd files -v 1.0.0.1
MyHeadtrax.manifest successfully created

D:\demo>mage.net -new Deployment -Install true -pub "Demo Publisher" -v 1.0.0.1 -AppManifest files\MyHeadtrax.manifest -t MyHeadtrax.application
MyHeadtrax.application successfully created

D:\demo>
```

The next command launches the ClickOnce application.



```
D:\demo>MyHeadtrax.application
```

And then the familiar ClickOnce installation dialog appears.



Publisher cannot be verified.
Are you sure you want to install this application?

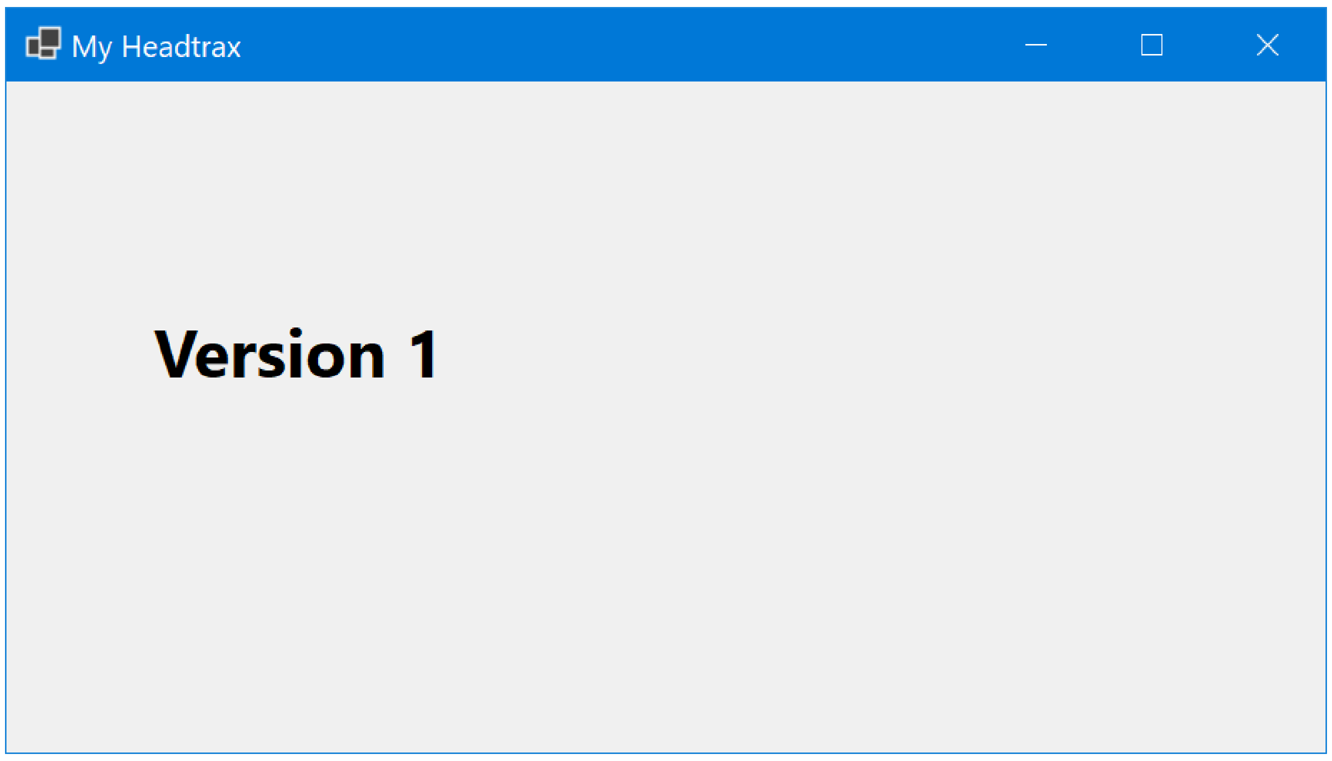
Name:
MyHeadtrax

From (Hover over the string below to see the full domain):
D:\demo

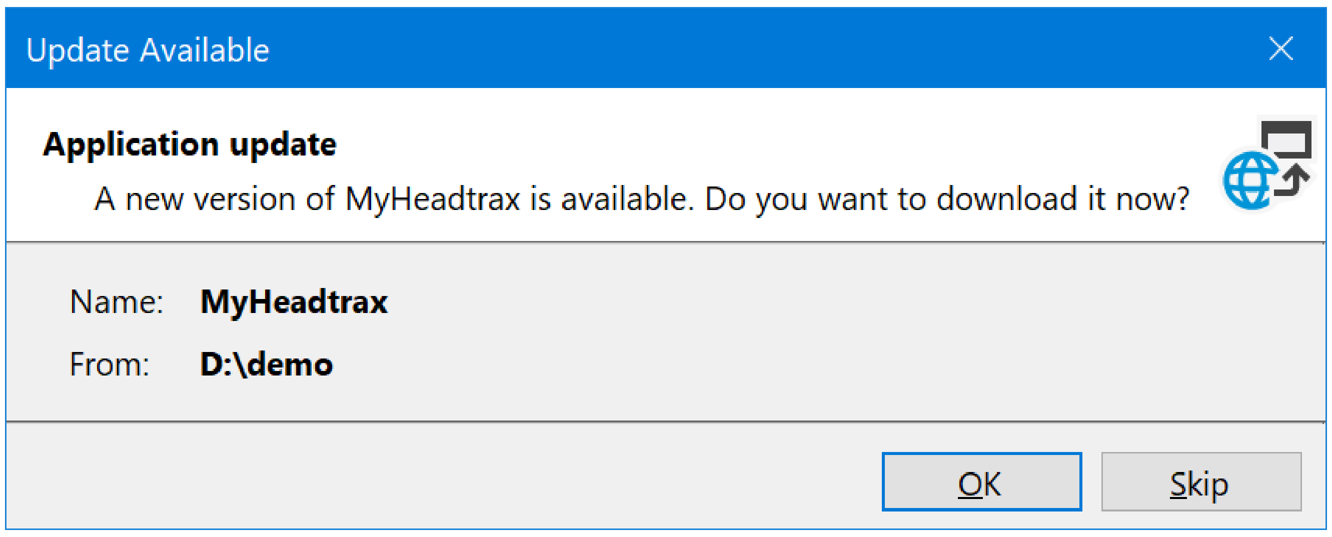
Publisher:
Unknown Publisher

While applications can be useful, they can potentially harm your computer. If you do not trust the source, do not install this software. More Information...

After installing the application, the app will be launched.



After re-building and re-publishing the application, users will see an update dialog.


Update Available

Application update

A new version of MyHeadtrax is available. Do you want to download it now?

Name: **MyHeadtrax**

From: **D:\demo**

And from there, the updated app will be launched.

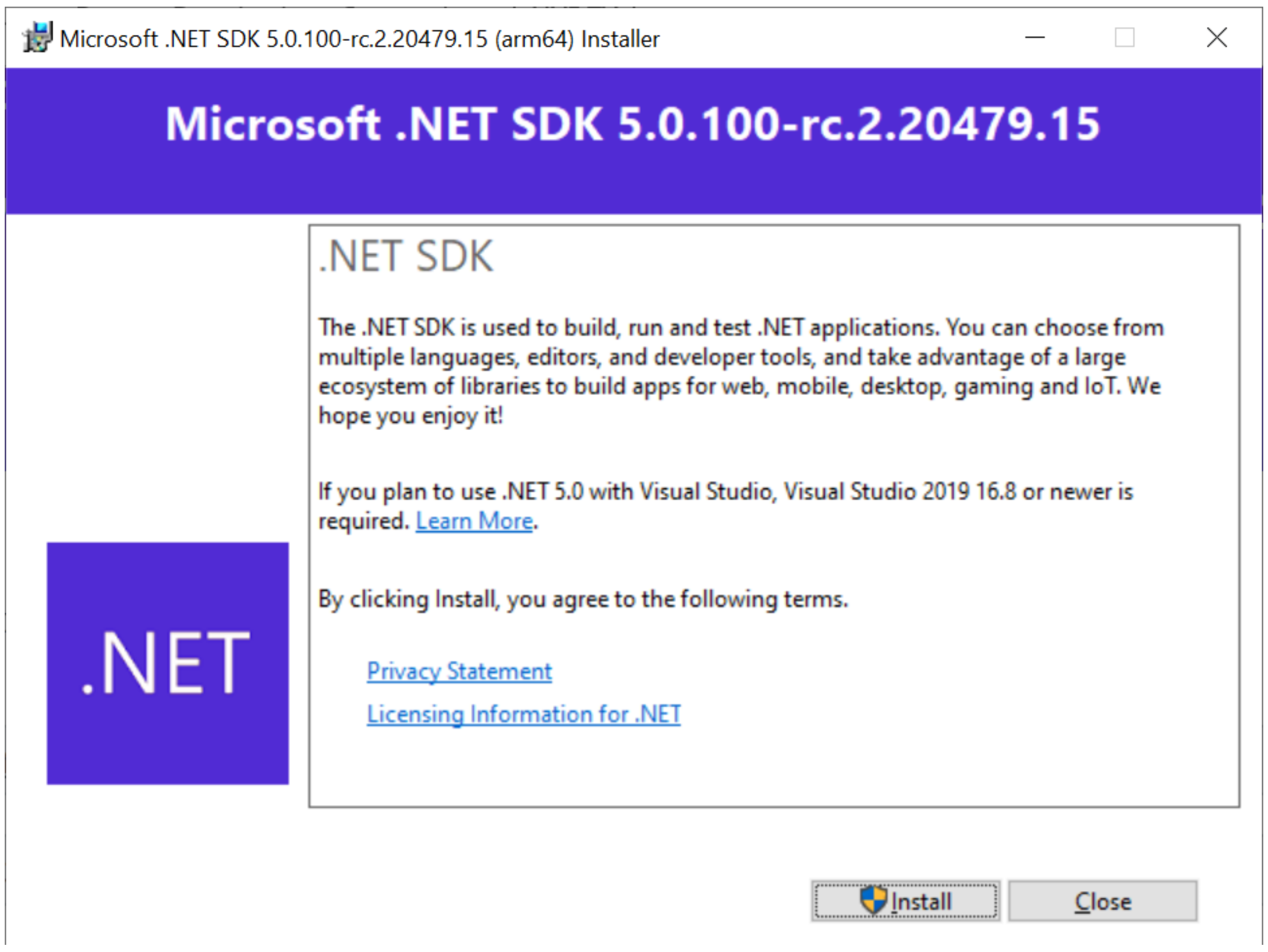
Note: The name of the Mage .NET tool will change from `mage.net` to `dotnet-mage` for the final release. The NuGet package name will remain the same.

This quick lap around ClickOnce publishing and installation should give you a good idea of how you might use ClickOnce. Our intention has been to enable a parity experience with the existing ClickOnce support for .NET Framework. If you find that we haven't lived up to that goal, please tell us.

ClickOnce browser integration is the same as with .NET Framework, supported in Edge and Internet Explorer. Please tell us how important it is to support the other browsers for your users.

Windows Arm64

MSI installers are now available for Windows Arm64, as you can see in the following image of the .NET 5.0 SDK installer.


Microsoft .NET SDK 5.0.100-rc.2.20479.15 (arm64) Installer

Microsoft .NET SDK 5.0.100-rc.2.20479.15

.NET SDK

The .NET SDK is used to build, run and test .NET applications. You can choose from multiple languages, editors, and developer tools, and take advantage of a large ecosystem of libraries to build apps for web, mobile, desktop, gaming and IoT. We hope you enjoy it!

If you plan to use .NET 5.0 with Visual Studio, Visual Studio 2019 16.8 or newer is required. Learn More.

By clicking Install, you agree to the following terms.

Privacy Statement
Licensing Information for .NET

To further prove the point, I ran the [dotnet-runtimeinfo](#) tool on my Arm64 machine to demonstrate the configuration.

```
C:\Users\rich>dotnet tool install -g dotnet-runtimeinfo
You can invoke the tool using the following command: dotnet-runtimeinfo
Tool 'dotnet-runtimeinfo' (version '1.0.2') was successfully installed.

C:\Users\rich>dotnet-runtimeinfo
**.NET information
Version: 5.0.0
FrameworkDescription: .NET 5.0.0-rc.2.20475.5
Libraries version: 5.0.0-rc.2.20475.5
Libraries hash: c5a3f49c88d3d907a56ec8d18f783426de5144e9

**Environment information
OSDescription: Microsoft Windows 10.0.18362
OSVersion: Microsoft Windows NT 10.0.18362.0
OSArchitecture: Arm64
ProcessorCount: 8
```

The .NET 5.0 SDK does not currently contain the Windows Desktop components — Windows Forms and WPF — on Windows Arm64. This late change was initially shared in the .NET 5.0 Preview 8 post. We are hoping to add the Windows desktop pack for Windows Arm64 in a 5.0 servicing update. We don't currently have a date to share. For now, the SDK, console and ASP.NET Core applications are supported on Windows Arm64.

Closing

We're now so close to finishing off this release, and sending it out for broad production use. We believe it is ready. The production use that it is already getting at Microsoft brings us a lot of confidence. We're looking forward to you getting the chance to really take advantage of .NET 5.0 in your own environment.

It's been a long time since we've shared our social media pages. If you are on social media, check out the dotnet pages we maintain:

- Twitter
- Facebook

Richard Lander
Program Manager, .NET Team
Follow

Posted in .NET .NET Core

Read next

Announcing Entity Framework Core (EF Core) 5 RC2

Entity Framework Core EF Core 5.0 gets closer to a final release. RC2 is now available with bug fixes and a go-live license.

Jeremy Likness October 13, 2020

0 comment

.NET Core October 2020 Updates – 2.1.23 and 3.1.9

Today, we are releasing the .NET Core October 2020 Update. These updates contains reliability and other non-security fixes. See the individual release notes for details ...

Rahul Bhandari (MSFT) October 13, 2020

0 comment

24 comments

Log in to join the discussion.

Newer Comments →

Tony Henrique October 13, 2020 6:48 pm

The option to have ClickOnce is great.

Log in to Reply

Jefferson Motta October 13, 2020 6:51 pm

Thank God ClickOnce is back, I hope it could be used without a pfx on Windows 10. The last ClickOnce for net 4.8 is damaged on Windows 10.

Log in to Reply

Somnath Shukla October 13, 2020 8:15 pm

does .Net 5 has fix for this. Its show stopper for .Net 3.1 . i can not use WindowsIdentity.RunImpersonated from IIS.
https://github.com/dotnet/runtime/issues/29935

Log in to Reply

VB User1 October 13, 2020 8:49 pm

System.Windows.Forms.OpenFileDialog

Setting a value for the "InitialDirectory" property has no effect when running ShowDialog.

Works fine with .NET Core 3.1.

```
Dim d As System.Windows.Forms.OpenFileDialog
d = New System.Windows.Forms.OpenFileDialog
d.InitialDirectory = "X:\WORK"
d.ShowDialog() 'The other directory has opened.
d.Dispose()
```

Log in to Reply

Kirsan October 14, 2020 1:55 am

Nice catch. https://github.com/dotnet/winforms/issues/3513

Log in to Reply

Merrie McGaw October 14, 2020 10:24 am

Thanks! We're taking a look.

Log in to Reply

John King October 13, 2020 9:04 pm

I sorry that I hate ClickOnce , because I never use a app that actually use ClickOnce !!!!!!
Visual Studio has it's own installer and it is writing use js/electron .
Skype for win32 has it's own installer.
QQ/Wechat has it's own installer.
Finddler has it's own installer.
VsCode has it's own installer.
Even .Net Core has it's own installer !!!!

so microsoft make ClickOnce for what ? just be different than others?
ClickOnce is the wrong action.
the right action is to improve WXI , which dotnet core in use.

app

1. wxi globaltool
2. vs integrated
3. vs code integrated
4. standalone binary / installer
5. msbuild/dotnet cli (`dotnet publish`)

feature

1. simple package
   - choose location
   - progress bar
   - update tool intergrated (nuget package that can be use in update tool app, we can add owe own url to our own update serve)
   - [update server]update serve protocol : maybe an open source asp.netcore app that can publish new version.
   - knwn appdata location and asking to remove it when uninstall.
   - launch app at OS start.
   - register services
   - bundle dependency(.net core runtime, or other software)
2. custom ui
   - chose backgound-color
   - chose backgound-image / slide
3. custom action
   - add custom script/dll /UI-script to build process

Log in to Reply

Richard Lander October 13, 2020 9:21 pm

Lots of choices! Visual Studio isn't going to use ClickOnce. Visual Studio about 1000x too complicated. Also, ClickOnce is for apps not platforms (like .NET Core). Those are not great candidates. ClickOnce is best for enterprise apps with limited installation requirements. People like ClickOnce because it's straightforward and has an update mechanism built-in. Very few other installers have this feature. It also supports HMAC signing for manifests. There are customers that require this signing feature.

Log in to Reply

Reilly Wood October 13, 2020 11:18 pm

Notably, ClickOnce is still used by VSTO Office addins. I really, really hope this lets the Office team finally move past the .NET Framework requirement for .NET Office addins – it's the only thing keeping many enterprise devs on Framework.

Log in to Reply

John King October 14, 2020 12:02 am

if you are in the team, then please consider my suggestion, make Wix the first choice for installer , by provide UI in visual studio, and .net core global tool, and use a `Target` to to opt-in the bundler process. read my comment above!

not only that, like I suggestion, we should provide more nuget package to support this tool chain, like the update tool, we can write all the update logic in the nuget, and provide a webapi service to know should it be update, the server url can be easily configured.

Log in to Reply

dexi lin October 14, 2020 12:40 am

@John King I quite agree with you.

Log in to Reply

Richard Lander October 14, 2020 9:01 am

I better understand your viewpoint now. We talked about this general idea before taking on the ClickOnce project. We knew that there were a lot of people that specifically wanted ClickOnce. As a result, we decided to add support for that. At this point, we'll be looking for the largest clusters of feedback: improve ClickOnce further, create an option along the lines of what you've described or something else.

Thanks for the feedback. Yes, I'm on the team.

Log in to Reply

Dat Minh October 13, 2020 9:20 pm

Sorry sir, but i like use != more than "is not", it short and clearly. I wanna my code can readable and maintainable.

Log in to Reply

Richard Lander October 13, 2020 9:26 pm

I understand. That said, they don't actually do the same thing. Context: https://github.com/dotnet/roslyn/issues/39253

Log in to Reply

Myo Zaw Latt October 13, 2020 9:38 pm

I am curious about Webform and MVC in coming .NET 5.
Are they still supported in .NET 5 and which versions of them are shipped with it?

Log in to Reply

Richard Lander October 14, 2020 9:04 am

They are not coming to .NET 5.0+. For web sites, ASP.NET Core and Blazor are the two options we offer on .NET 5.0+.

Log in to Reply

Myo Zaw Latt October 14, 2020 11:22 am

Is that mean we must migrate our existing Web Form and MVC applications to ASP.NET Core to get future support with .NET 5+?
May I know the future of Web Form and MVC, are they dead?

Log in to Reply

Martin Weeks October 13, 2020 10:22 pm

Love the idea of ClickOnce in .NET 5, to be honest that's only thing holding me back from migrating our company's internal apps to .NET Core/.NET 5.
Appreciate that MSIX is a bit more robust, but for quickly launching a basic app to a bunch of colleagues you can't beat ClickOnce.

It appears that it only works for .NET Core 3.1 right now, is that the case? I tried both .NET 5 and .NET Framework 4.8, and both wouldn't let me past the 'Finish' button. It briefly flashes 'Waiting…' but does nothing.

Log in to Reply

Richard Lander October 14, 2020 9:09 am

I heard from the Visual Studio team that .NET Core 3.1 has the best ClickOnce experience currently. Sounds like there are some remaining bugs to fix. Next update is November 10th.

Log in to Reply

A L October 14, 2020 12:06 am

During internal presentations you really should promote Blazor for Office teams! The new Office add-in platform is based on an embedded web site, interaction with client app and documents is accomplished using Office.js JavaScript APIs. The Office add-ins run cross platform, but currently can only be developed using JavaScript. Blazor could enable C#-based Office add-in development, also maybe provide a smoother upgrade path for current C#-based VSTO add-ins.

Log in to Reply

Holger Schwichtenberg October 14, 2020 2:01 am

The option to use ClickOnce in .NET is great!

However, I cannot see the target "ClickOnce" in Visual Studio 2019 v16.8 Preview 4 (latest version, published yesterday) when I select "Build/Publish…"

How can I activate this feature?

If it is not available in the current Visual Studio release, it should be mentioned in the blog post 😉

Log in to Reply

Ian Marteens October 14, 2020 7:38 am

You must create a new publishing Profile. Check Folder in the first page of the assistant, and then in the second page, you'll find ClickOnce.

Log in to Reply

Newer Comments →

Relevant Links

.NET Download
.NET Hello World
.NET Meetup Events
.NET Documentation
.NET API Browser
.NET SDKs

.NET Application Architecture Guides

Web apps with ASP.NET Core
Mobile apps with Xamarin.Forms
Microservices with Docker Containers
Modernizing existing .NET apps to the cloud

Archive

October 2020
September 2020
August 2020
July 2020
June 2020
May 2020
April 2020
March 2020
February 2020
January 2020
December 2019

Topics

Dot.Net
.NET
.NET Core
.NET Framework
Entity Framework
C#
ML.NET
Visual Studio
F#
WPF
Machine Learning

Stay informed