

.NET 5 & C# 9

A look at what's new

#DDD20

HELLO!

I am Stuart Lang

You can find me at

stu.dev



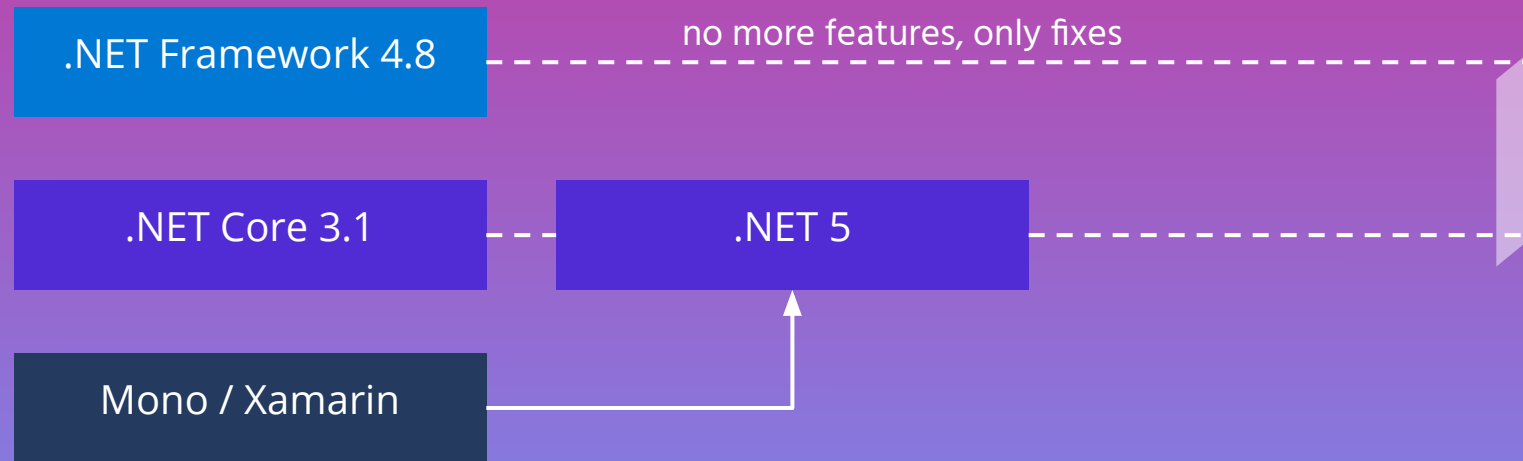
DMs open!



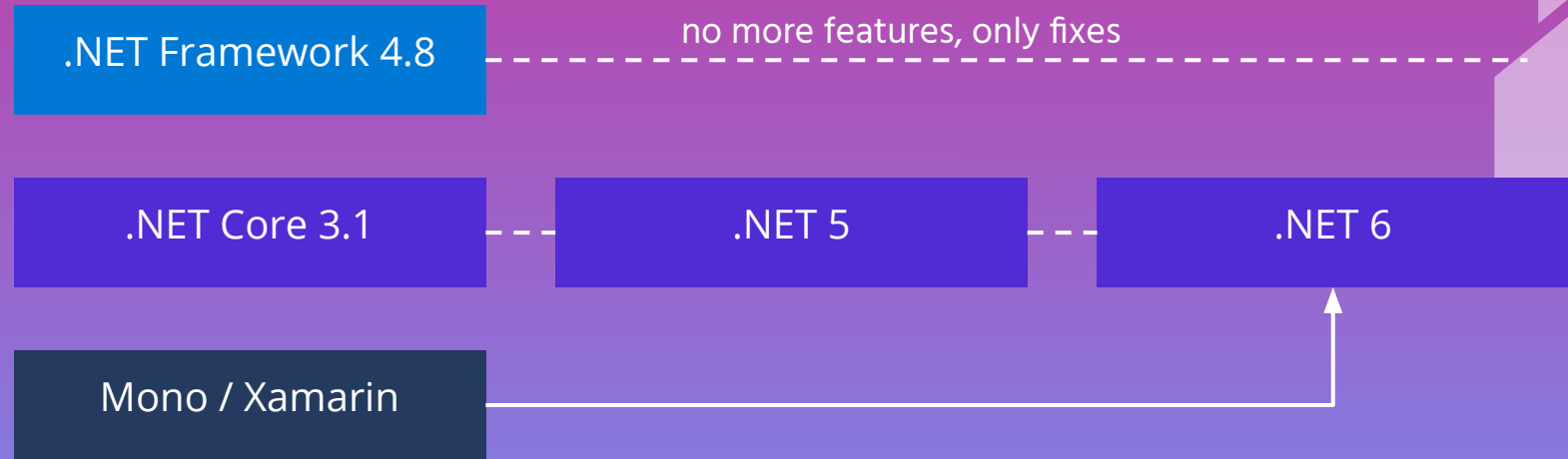
Agenda

- › .NET 5
 - › What is it
 - › Support
 - › What's new
- › C# 9
 - › Top-level programs
 - › Patterns
 - › Records
 - › Source Generators

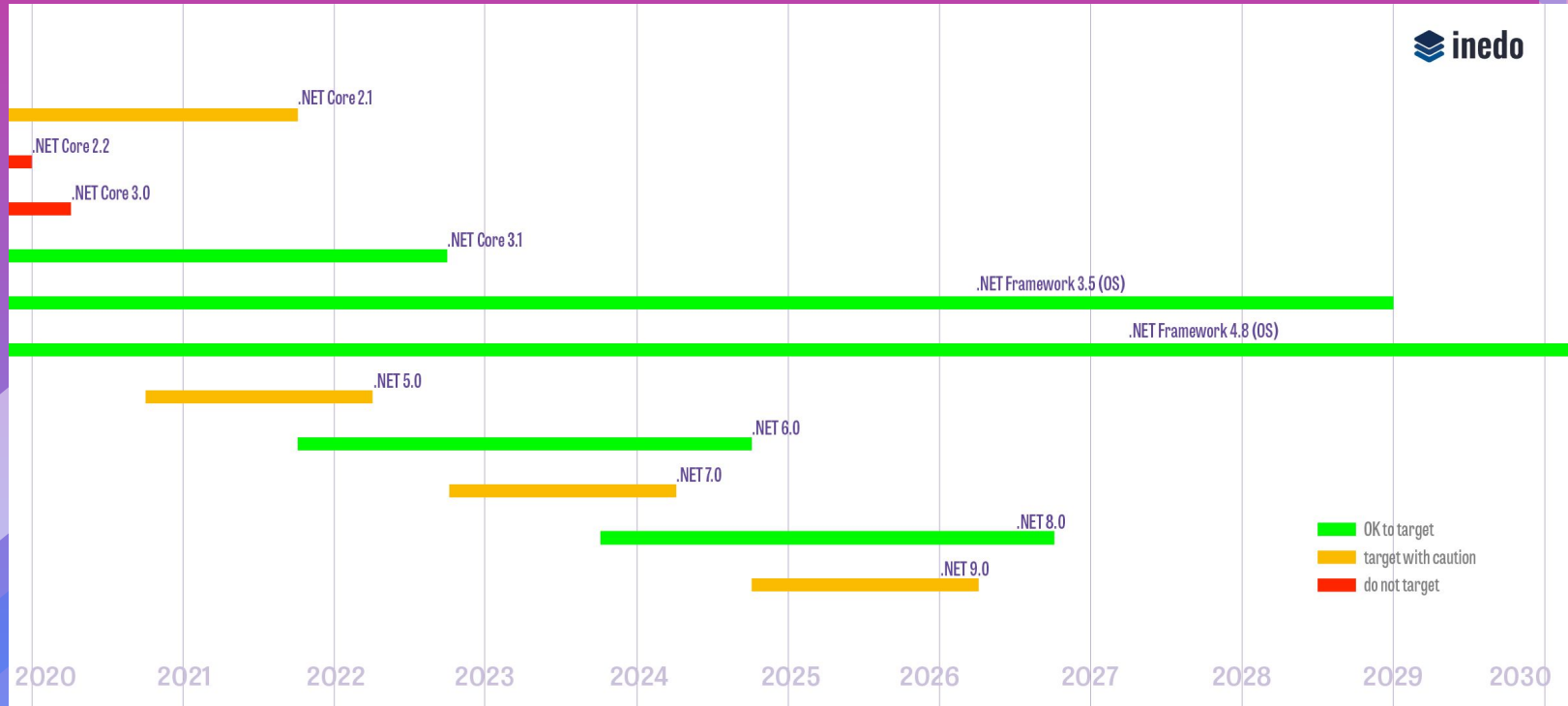
What's .NET 5



What's .NET 5



Release Plan



<https://blog.inedo.com/demystifying-net-lts>

Consider Ecosystem Support

 **Marc Gravell**
@marcgravell

I don't speak for anyone except myself, but in all honesty, I'm increasingly of the opinion that whatever is in my libs the day .NET 6 ships: that's what anything netfx stays with, unless someone is paying rate for my time. I am so ***done*** with netfx.

 **Marc Gravell** @marcgravell · Nov 20

Clarification: Microsoft may need to support .NET 3.5 and 4.8 for these timelines. If you're thinking that your familiar OSS tools are going to do likewise, it may be time to make sure you have a support license! Because: hell no.



Version	Support Status
.NET Framework 3.5 (OS)	OK to target
.NET Framework 4.8 (OS)	OK to target
.NET 8.0	OK to target
.NET 9.0	target with caution

11:25 PM · Nov 20, 2020 · Twitter for Android

Porting to .NET 5

try-convert: <https://github.com/dotnet/try-convert>

.NET Conf 2020
Ask questions on twitter at #dotNETConf

Porting to .NET 5

1. Understand goals
2. Inventory
3. Convert Project
4. Move to .NET 5
5. Move to other OS

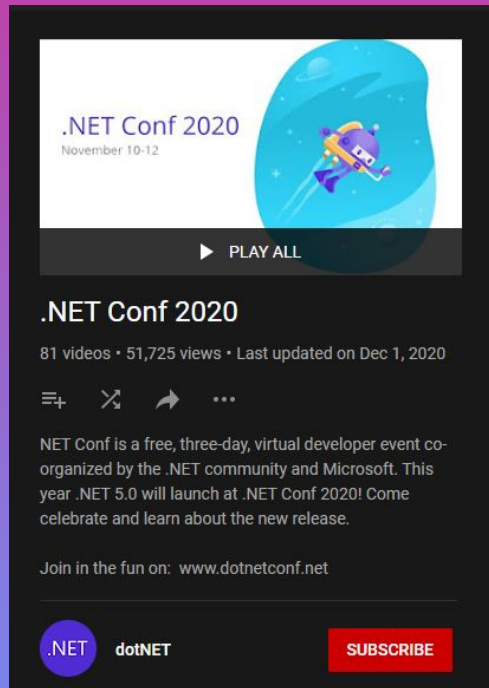
- **Understand if you need to port at all**
 - Only port projects that you need to innovate in
 - Maintenance-only projects can safely remain on .NET Framework
- **Make note of your desired target:**
 - .NET Core 3.1 vs. .NET 5
 - Windows vs. MacOS vs. Linux
- **Plan & ensure you have click stops**
 - So that you can keep shipping working software

Porting Projects to .NET 5
10,409 Views · Nov 12, 2020

188 2 SHARE SAVE ...

.NET Conf 2020
dotNET - 12 / 81

.NET Conf 2020



The image shows a YouTube video player interface for a playlist titled ".NET Conf 2020". The video thumbnail features a blue space-themed background with a rocket ship and the text ".NET Conf 2020" and "November 10-12". Below the thumbnail is a "PLAY ALL" button. The video title ".NET Conf 2020" is displayed in bold, followed by the statistics "81 videos • 51,725 views • Last updated on Dec 1, 2020". Below this are icons for playlist, share, and more options. A description follows: "NET Conf is a free, three-day, virtual developer event co-organized by the .NET community and Microsoft. This year .NET 5.0 will launch at .NET Conf 2020! Come celebrate and learn about the new release." Below the description is the URL "www.dotnetconf.net". At the bottom left is the ".NET dotNET" logo, and at the bottom right is a red "SUBSCRIBE" button.

.NET Conf 2020
November 10-12

▶ PLAY ALL

.NET Conf 2020
81 videos • 51,725 views • Last updated on Dec 1, 2020

≡+ ✂ ↗ ⋮

NET Conf is a free, three-day, virtual developer event co-organized by the .NET community and Microsoft. This year .NET 5.0 will launch at .NET Conf 2020! Come celebrate and learn about the new release.

Join in the fun on: www.dotnetconf.net

.NET dotNET **SUBSCRIBE**

.NET Conf 2020

.NET 5 Features

- › C# 9
- › Performance ⚡ [Link](#)
- › Single file apps
- › App trimming
- › System.Text.Json new features
- › F# 5
- › Add more

C# 9

Top-level programs

* demo *

Patterns

Patterns - Switch Expressions

```
var first = pattern[0];
var last = pattern[pattern.Length - 1];

if (first == '%' && last == '%')
{
    return src.ToLower().Contains(pattern.Substring(1, pattern.Length - 2).ToLower());
}

if (first == '%')
{
    return src.ToLower().EndsWith(pattern.Substring(1).ToLower());
}

if (last == '%')
{
    return src.ToLower().StartsWith(pattern.Substring(0, pattern.Length - 1).ToLower());
}

return src.ToLower().Equals(pattern.ToLower());
```

Patterns - Switch Expressions

```
var first = pattern[0];  
var last = pattern[^1];  
  
return (first, last) switch {  
    ('%', '%') => src.ToLower().Contains(pattern[1..^1].ToLower()),  
    ('%', _ ) => src.ToLower().EndsWith(pattern[1..].ToLower()),  
    ( _ , '%' ) => src.ToLower().StartsWith(pattern[..^1].ToLower()),  
    ( _ , _ ) => src.ToLower().Equals(pattern.ToLower())  
};
```

Patterns - Switch Expressions

```
static double ComputeArea(object shape)
{
    switch (shape)
    {
        case Square s when s.Side == 0:
        case Circle c when c.Radius == 0:
        case Triangle t when t.Base == 0 || t.Height == 0:
        case Rectangle r when r.Length == 0 || r.Height == 0:
            return 0;

        case Square s:
            return s.Side * s.Side;
        case Circle c:
            return c.Radius * c.Radius * Math.PI;
        case Triangle t:
            return t.Base * t.Height / 2;
        case Rectangle r:
            return r.Length * r.Height;
        default:
            throw new ArgumentException("shape is not a recognized shape",
                nameof(shape));
    }
}
```


Patterns - Switch Expressions

```
static double ComputeArea(object shape)
{
    return shape switch
    {
        Square {Side: 0} => 0,
        Circle {Radius: 0} => 0,
        Triangle {Base: 0} or Triangle {Height: 0} => 0,
        Rectangle {Length: 0} or Rectangle {Height: 0} => 0,
        Square s => s.Side * s.Side,
        Circle c => c.Radius * c.Radius * Math.PI,
        Triangle t => t.Base * t.Height / 2,
        Rectangle r => r.Length * r.Height,
        _ => throw new ArgumentException("shape is not a recognized shape", nameof(shape))
    };
}
```

Patterns - Examples



Joseph N. Musser II
@jnm236



Replying to @stuartblang

There's a few good ones in
[github.com/shouldly/shoul...](https://github.com/shouldly/shouldly) :D

E.g.:

```
return context.Expected is IEnumerable
    && !(context.Expected is string)
    && context.Actual is IEnumerable
    && !(context.Actual is string);

return context.Expected is IEnumerable and not string
    && context.Actual is IEnumerable and not string;
```

Patterns - Examples



Joseph N. Musser II
@jnm236



Replying to @jnm236 and @stuartblang

Another one in my Shouldly PR:

```
public override bool CanProcess(IShouldlyAssertionContext context)
{
-     return context.ShouldMethod.Equals("ShouldBe") &&
-         !(context.Expected is Expression) &&
-         context.Actual is IEnumerable<string>;
+     return context is
+     {
+         ShouldMethod: "ShouldBe",
+         Expected: not Expression,
+         Actual: IEnumerable<string>,
+     };
}
```

Patterns - Examples



Joseph N. Musser II

@jnm236



Replying to @jnm236 and @stuartblang

Another one:

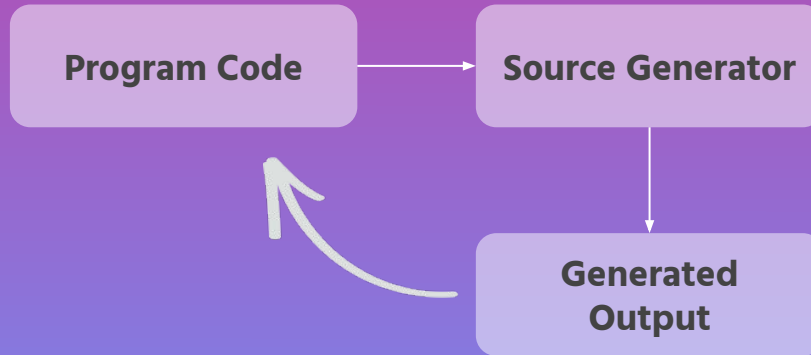
```
public override bool CanProcess(IShouldlyAssertionContext context)
{
-     return context.ShouldMethod == "ShouldBeginWith" ||
-     context.ShouldMethod == "ShouldNotBeginWith" ||
-     context.ShouldMethod == "ShouldEndWith" ||
-     context.ShouldMethod == "ShouldNotEndWith";
+     return context.ShouldMethod is
+         "ShouldBeginWith" or
+         "ShouldNotBeginWith" or
+         "ShouldEndWith" or
+         "ShouldNotEndWith";
}
```

Records

* demo *

Source Generators

Source Generators



Source Generators

Source Generator Playground - 1.0.22+b848601427 Load sample: **Hello World** by @davidwengier - GitHub

Program Code Load **Source Generator** Load

```
1
2 namespace MyApp
3 {
4     class Program
5     {
6         static void Main()
7         {
8             HelloFromGeneratedCode();
9         }
10    }
11 }
12
```

Source Generator

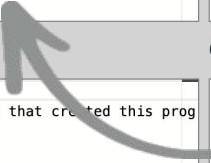
```
1 using System;
2 using System.Collections.Generic;
3 using System.Text;
4
5 using Microsoft.CodeAnalysis;
6 using Microsoft.CodeAnalysis.Text;
7
8 namespace SourceGenerator
9 {
10    [Generator]
11    public class HelloWorldGenerator : ISourceGenerator
12    {
13        public void Execute(GeneratorExecutionContext context)
14        {
15            // begin creating the source we'll inject into the users code
16        }
17    }
18 }
```

Output **Generated Output** Auto Refresh

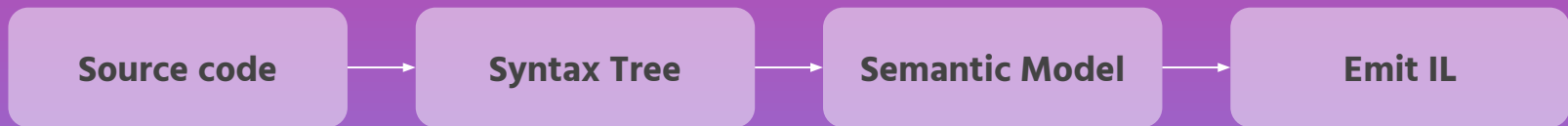
```
Hello from generated code!
The following syntax trees existed in the compilation that created this prog
- Program.cs
```

Generated Output

```
1 using System;
2
3 namespace HelloWorldGenerated
4 {
5     public static class HelloWorld
6     {
7         public static void SayHello()
8         {
9             Console.WriteLine("Hello from generated code!");
10            Console.WriteLine("The following syntax trees existed in the
11            Console.WriteLine(@" - Program.cs");
12        }
13    }
14 }
15
```



Source Generators - Roslyn



Source Generators

* demo *



THANKS!

Any questions?

You can find me at:

@stuartblang · stu.dev