

C#/.NET Source Generators

Writing code, to write code.

Peter "Shawty" Shaw/Digital Solutions UK



Introductions

- Many of you will already know me, I've been a .NET developer here in the north east for a long time now.
- I generally go by the name of "Shawty", but I answer to just about anything especially in a beer induced environment 😊
- Please note:
 - **I am Clinically Deaf**, and wear hearing aid's in order to hear, these don't always have the effect they should, especially when there's a lot of background noise, if you want to ask me questions during this session, you might have to raise your hand, or jump about like a lunatic to attract my attention.

Introductions

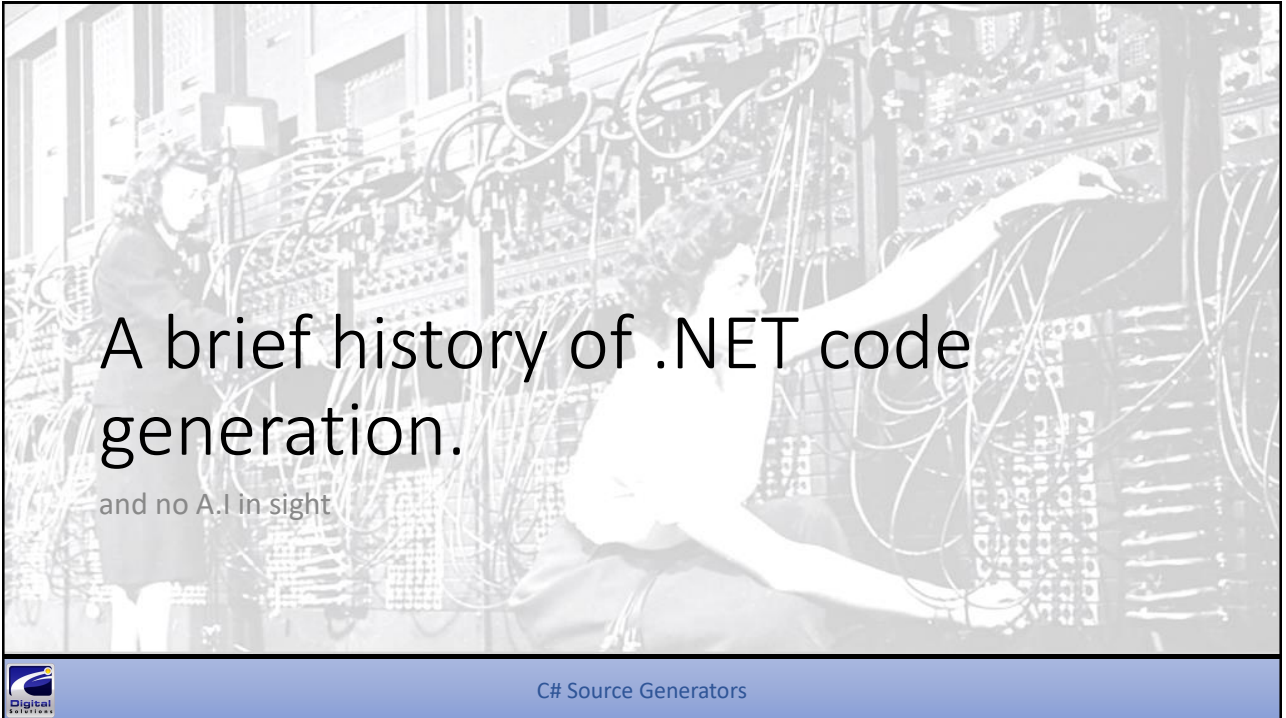
- If you've not met me before, then you can stalk me via the following means:
 - Email – shawty.d.ds@gmail.com
 - Twitter - @shawty_ds
 - Linked-in – “Peter Shaw”/”LiDNUG” (The online .NET user group I help run)
- The code for this session is available from my Github repo at:
 - <https://github.com/shawty/codegeneratorshainton2022>



Tonight's Agenda

- As the title suggests we'll be taking a look at the new source generation features in the latest .NET builds, covering the following topics:
 - Brief history of .NET code generation
 - What source generation options are currently in use in .NET
 - Why do we need another source generation option?
 - Demos of the new features
 - Where are we currently with the new offering
 - What has the .NET community already done with the new features
 - Summary





A brief history

- The first properly known code generation tool for any platform called “Matlab” was released in 1984, and is still in use to day in a surprising number of arenas.
- Matlab generates code for everything from “software designed radio” to huge scale “Industrial robotics”
- It’s a proprietary tool, that costs a lot to license, but has been customised and released as a “tool to control” a lot of different systems.



A brief history

- The FIRST code generation platform specifically for .NET was called “outsystems” and was released in 2001
- Unfortunately it’s a very closed platform, and not much information is available about it
- These days OUTSYSTEMS are known for “low code” and “no code” tools and are backed heavily by investors in the finance industry.



A brief history

- The most well known historical 3rd party code generation tool that worked for .NET is “Codesmith Generator”
- Codesmith is still available today, but it is now showing it’s age, it still works with the old .NET framework, but can generate any code you want.
- It uses a template system that looks very much like classic ASP web pages (Pre ASP.NET)
- Having used Codesmith myself, it’s a stable application, but hard to control in modern day CI/CD pipeline.



A brief history

- The first actual built in code generation system, provided by Microsoft in their dev tools, was “T4 Templates” released in 2005.
- The T4 system is now open source and available under an MIT license, but no one has taken up the challenge to improve it for the modern day tooling.
- T4 is still available even in Visual Studio 2022, but there is no syntax highlighting, intelli-sense or code completion and is quite cumbersome to use.
- T4 up until now however, remained the best built in tool in the VS ecosystem for code generation.



A brief history

- Notable mentions
 - Accelerator by Surround Technologies, released 2007
 - Code4Green by Code 4 Green Ltd, released 2009
 - Reegenerator by Kodeo Ltd, released 2010
 - CodeBaght by CodeBaght LLC, released 2014

From 2014 onwards, the landscape for code generators has been very quiet, Microsoft released things like Linq to SQL, but they were largely very specialised and only generated one thing.



```
namespace Meta
{
    class MailMessage
    {
        Person From;
        List<Person> To;

        string Subject;
        string Message;

        string FullName;
        string Email;
    }
}
```

What source generation options are currently in use in .NET?

Many things, that make many things, that make Nothing?



C# Source Generators

What's currently in use?

- The two main contenders for 3rd party tools are
 - Codesmith generator
 - T4 Templates
- Many developers have resorted to homebrew tools, code project and git-hub are littered with tools to generate various data abstraction layers from various database technologies.
- A number of ORM systems have CLI based tooling that can be called inside a CI/CD pipeline to perform various tasks.



C# Source Generators

What's currently in use?

- JavaScript environments have a stupid amount of tools for generating code, just to mention a few
 - Parcel
 - Webpack
 - Yeoman
 - Most of the CLI build tools for things like Angular, Aurelia and others
 - Even NPM can generate code



What's currently in use?

- In C#/.NET itself, the choices are very limited unless you want to use T4 or pay a lot of money.
- There are many open source tools, but you take what you get using this route.
- Reflection inside the runtime of your app is possible, as is the Roslyn compiler.
- Most of the well known NuGet packages for code generation, are VERY specific (EG: Swagger for API gen)
- Systems like Linq2Sql and EF Migrations are still a thing, but can be cumbersome to get right.





Revenue growth divisions.

TVU division

FRT division

Projected sales of main products

Why do we need another source generation option?

because the man said we do....

	TVU division			FRT division		
GHT	254	550	254	274	154	415
RCW	650	330	754	273	825	154
TRG	341	450	144	344	954	174
RTG	254	650	874	657	125	274
WER	784	145	124	752	741	750

Current market share

Market share

150

50

0

100

200

300

400

500

600

700

800

900

1000

1100

1200

1300

1400

1500

1600

1700

1800

1900

2000

2100

2200

2300

2400

2500

2600

2700

2800

2900

3000

3100

3200

3300

3400

3500

3600

3700

3800

3900

4000

4100

4200

4300

4400

4500

4600

4700

4800

4900

5000

5100

5200

5300

5400

5500

5600

5700

5800

5900

6000

6100

6200

6300

6400

6500

6600

6700

6800

6900

7000

7100

7200

7300

7400

7500

7600

7700

7800

7900

8000

8100

8200

8300

8400

8500

8600

8700

8800

8900

9000

9100

9200

9300

9400

9500

9600

9700

9800

9900

10000

Digital Solutions

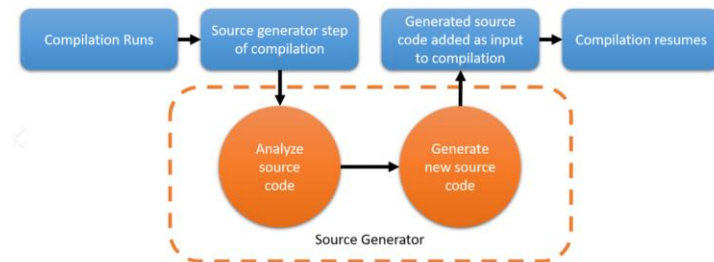
C# Source Generators

Why do we need another option?

- In all honesty, we don't actually, what we need is an option that runs where we need it too.
- The existing systems all run:
 - Pre build
 - At specific points in the build beyond our control
 - For specific scenarios only
 - Post Build

Why do we need another option?

- The new source generation features run during compilation of your regular source files.
- They generate NEW source code while the compile is currently in progress.



C# Source Generators

Why do we need another option?

- Source generators are built on the back of the Roslyn Compiler technologies and libraries.
- They use the same frameworks as Roslyn Code Analysers.
- Source generators CAN ONLY GENERATE C# Source code, they CANNOT create HTML, CSS, XML, Json or anything that is non compliable.



C# Source Generators

Code:

<https://github.com/shawty/codegeneratorshainton2022>

Demos

with hopefully no self exploding code



C# Source Generators

A grayscale illustration of a complex maze. Several stylized human figures are shown navigating through the maze. One figure is carrying a basket, another is using a cane, and others are walking or standing. The maze is composed of white walls on a light gray background.

Where are we currently with the new offering?

It's always harder at first



C# Source Generators

Where are we currently?

- The technology is very simple to use
- It works very well
- Adding it to a project is still somewhat manual
- You can use NuGet to deploy generators but you have to tell NuGet it's a Roslyn analyser.
- Debugging could be better



Where are we currently?

- Microsoft ARE putting their weight behind this, they've already announced a slew of new features around the technology.
- One really interesting idea is "Regex Source Generators"
- Another idea that has been touted as "Very Useful" is auto generation of "Notify Properties" from "Normal Properties" in WPF/MVVM models



What has the .NET community already done with the new features?

a good community, with an exceptional track record, makes a big difference.



C# Source Generators

What has the community done?

- As is usually the case, the .NET community have been feverously busy putting the new technology through it's paces.
- There are plenty of blog posts
 - <https://medium.com/rocket-mortgage-technology-blog/generating-code-in-c-1868ebbe52c5>
 - <https://nicksnettravels.builttoroam.com/debug-code-gen/>
 - <https://blog.jetbrains.com/dotnet/2020/11/12/source-generators-in-net-5-with-resharper/>
 - <https://devblogs.microsoft.com/dotnet/using-c-source-generators-to-create-an-external-dsl/>



C# Source Generators

What has the community done?

- As is usually the case, the .NET community have been feverously busy putting the new technology through it's paces.
- We already have the makings of an "Awesome xxxxx" page for it:
 - <https://github.com/amis92/csharp-source-generators>
- And the obligatory Microsoft samples project for it too:
 - <https://devblogs.microsoft.com/dotnet/new-c-source-generator-samples/>
- Microsoft also publish the "Source Generators Cookbook"
 - <https://github.com/dotnet/roslyn/blob/main/docs/features/source-generators.cookbook.md>

A background image showing a pair of hands pointing at a document on a desk. There is a small orange clock and some papers on the desk. The image is slightly blurred and has a warm, yellowish tint.

Summary

aka – the post chat, about the chat we just chatted about.



Summary

- Source generators fill in a missing gap in the .NET build system that until now has been difficult to get at.
- It's still an early technology with a few rough edges, but it has great promise.
- Yes debugging is cumbersome, but there are ways to deal with that easily until things improve.
- You can generate pretty much ANY extra source you want and it can be as dynamic as you want, your imagination is the only restriction.
- For everything else T4 is still a great option.
- The .NET community as expected has stepped up and taken the bull by the horns with some fantastic results.



C# Source Generators

I HAVE A VERY PARTICULAR SET OF SKILLS.

Code & Slides:

<https://github.com/shawty/codegeneratorshainton2022>

Contacting me:

Twitter: @shawty_ds

Email: shawty.d.ds@gmail.com

Search: "shawty/ds", "shawty/Lidnug"

Thanks for listening

I WILL FIND YOUR QUESTIONS,
AND I WILL ANSWER THEM.



C# Source Generators