



ROSLYN SOURCE GENERATORS

In C# and Unity Development



Yuri Sokolov

Professional Developer since **2012**

Game Developer since **2017**

Gamedev Podcast Co-Host since **2022**



MOONACTIVE



PAPAYA

00

DEMONSTRATION

A quick look into what source generators can do






01

ROSLYN

.NET Compiler Platform

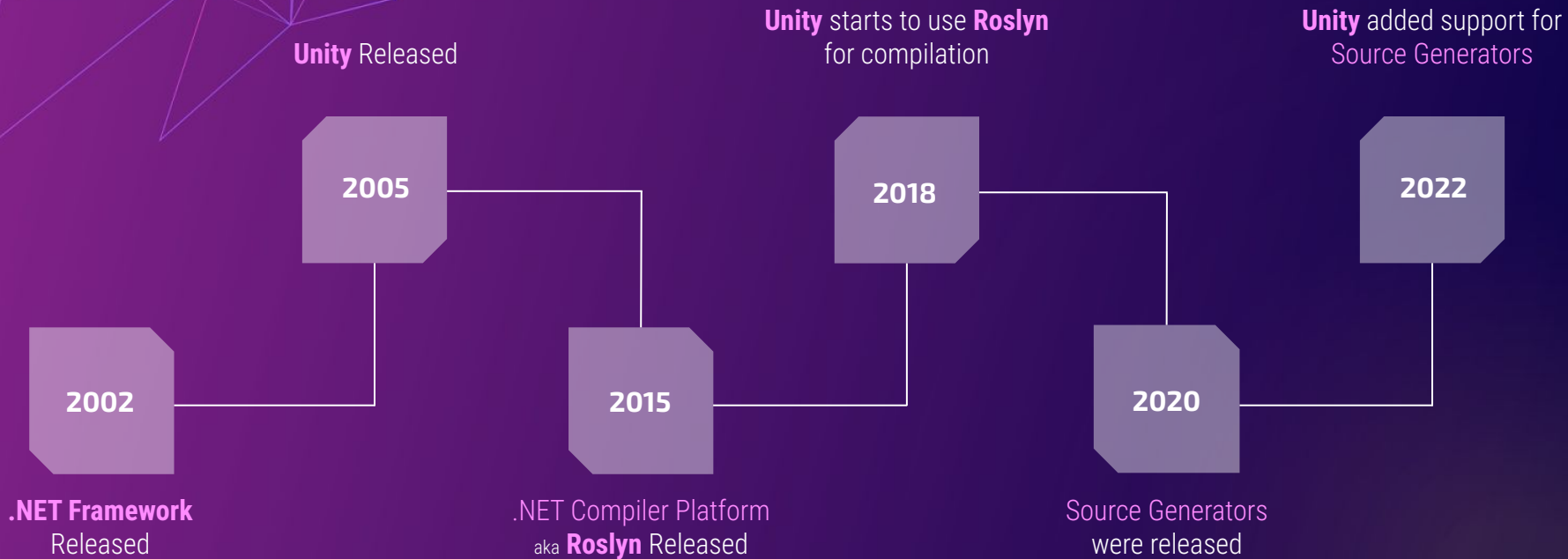


“Roslyn is an open-source .NET Compiler Platform that allows developers to analyze, generate, and manipulate code, effectively turning the C# and Visual Basic compilers into services that can be leveraged during the development process.”

—ChatGPT



A BIT of history





02

SOURCE GENERATORS

How it works?

Concepts



Code Analyzer

Introduced in 2015 along with **Roslyn** and allowed to analyze the code during a compilation.



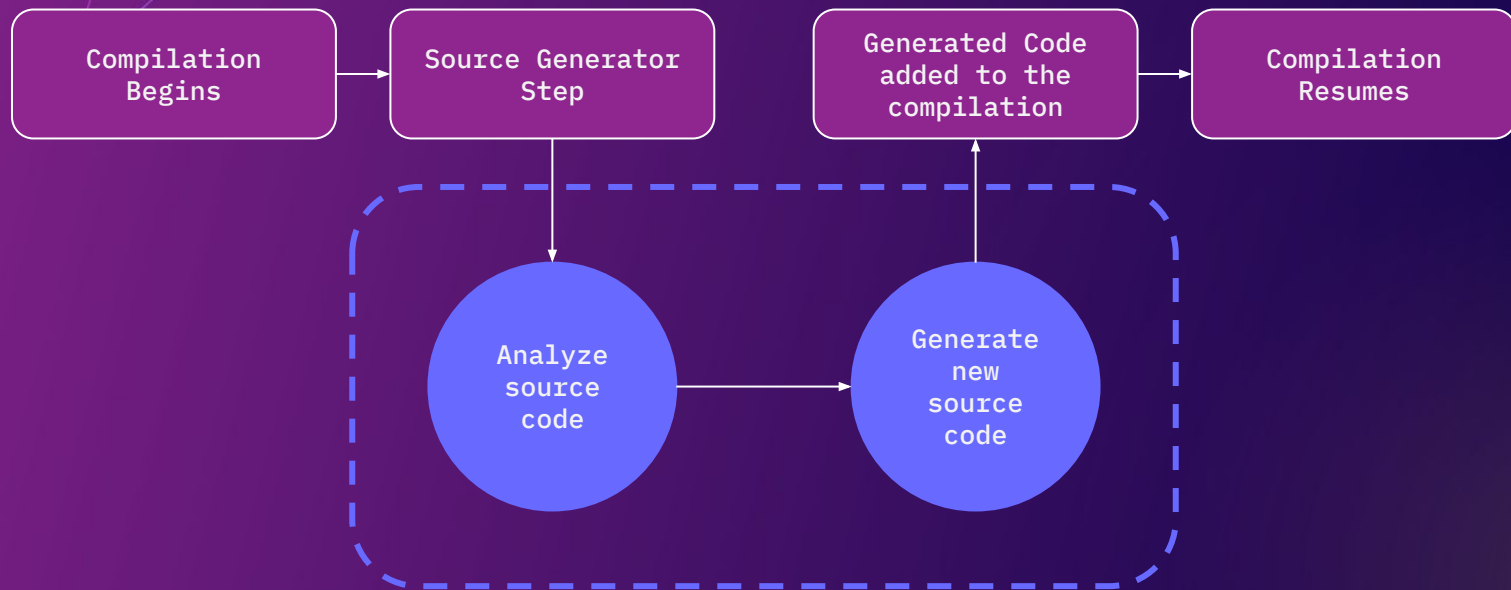
Source Generators

Built on top of **Code Analyzers** and using the same API to integrate generated code into compilation.

Regular Compilation Process



Roslyn Compilation Process



Q&A



Is this similar to reflection?

No. Unlike reflection, which occurs at runtime and deals with types, this process happens during compilation and works with actual text

Is this similar to Code Weaving?

Similar, but different. Code weaving occurs after compilation, meaning you don't have access to the weaved changes during the development process

What are the drawbacks?

A steep learning curve, increased build sizes, and the necessity to develop the generators outside of the solution generated by Unity

Key Concepts



SYNTAX TREES

Detailed representation of the code including every token and piece of trivia (whitespace, comments)

SYMBOLS



These represent elements in the code such as namespaces, types, methods, properties



SEMANTIC MODELS

They help you to understand the types of expressions, resolve method overloads, find implicit conversions, etc.

SYNTAX NODES



Every construct in the code (like declarations, statements, expressions) is a Syntax Node.
The Syntax Tree is a tree of Syntax Nodes



03

Usages

Why do we need Source Generators?

Usages



PRODUCTIVITY

Automating as much boilerplate code as possible

PERFORMANCE

Improving runtime performance with compile time optimizations



ERRORS PREVENTION

Reducing the chances for a potential human errors

CLEANER CODE

Sweeping all of the fugly* code under the rug



Other Use Cases

1. **[VContainer]** (<https://vcontainer.hadashikick.jp/>) - ultra fast, DI framework with Source Generation support for better results
2. **[Cloneable]** (<https://github.com/mostmand/Cloneable>) - auto-generated Clone() method.
3. **[Credfeto.Enumeration.Source.Generation]** (<https://github.com/credfeto/credfeto-enum-source-generation>) - Enum to text generator for enums - generates strongly typed enums for all enums in the assembly, and using EnumText attribute for third party enums. Also includes an analyzer to ensure that all enum usages use the .GetName extension method rather than .ToString.
4. **[Data Builder Generator]** (<https://github.com/dasMulli/data-builder-generator>) - Generate data builder patterns for your model classes.
5. **[Generator.Equals]** (<https://github.com/diegofrata/Generator.Equals>) - generates equality and hashing for classes and records, supports a number of strategies for comparing collections and properties.
6. **[FastGenericNew]** (<https://github.com/Nyrest/FastGenericNew>) - The ultimate fast alternative to Activator.CreateInstance<T> / new T(). Built on SourceGenerator V2 (Incremental Generator).
7. **[LinqGen]** (<https://github.com/cathei/LinqGen>) - Alloc-free and fast replacement for Linq, with code generation.
8. **[Mapperly]** (<https://github.com/riok/mapperly>) - A source generator for generating object mappings.
9. **[ToString]** (<https://github.com/Burgyn/MMLib.ToString>) - C# source generator for implementing ToString override like record type.
10. And many more!

04

DEMONSTRATION

More code, yay!



THANKS

Repo & Presentation

