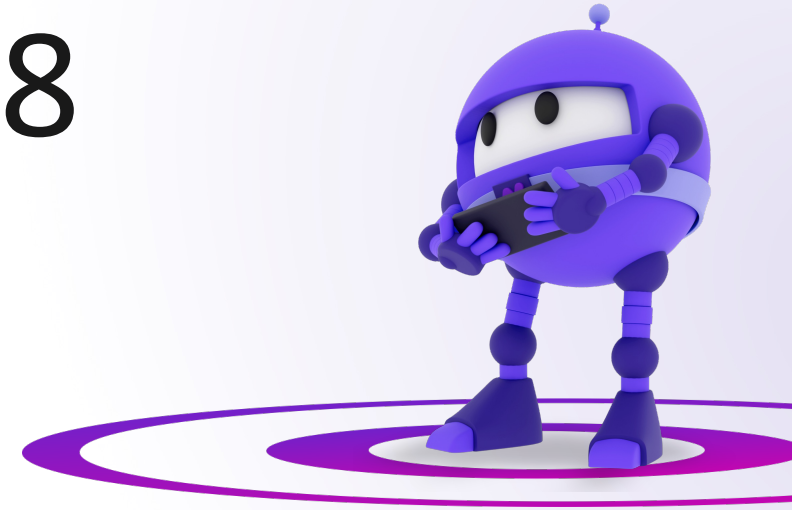
.NET Conf 2023

beanTech
IT moves your business

[stesi]
Powered by Innovation

Source Generator in .NET 8

Marco Bortolin



Source Generator

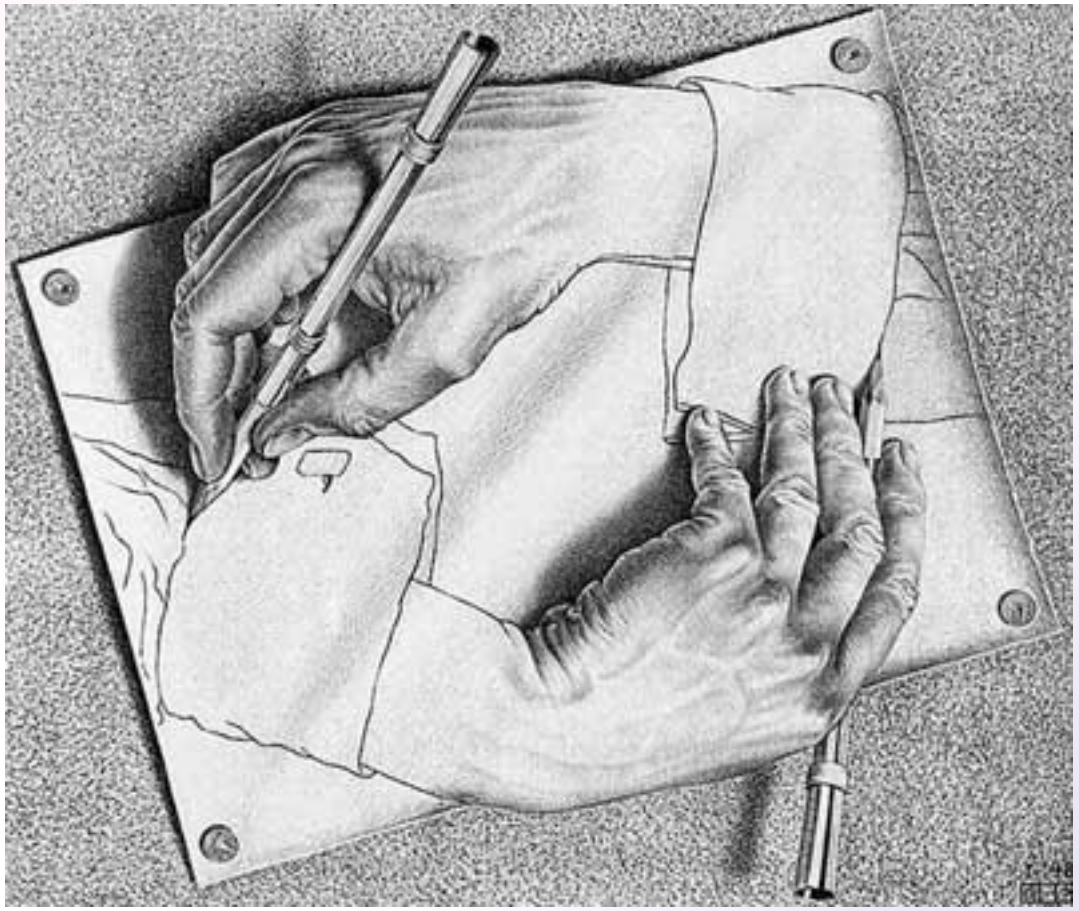
"A Source Generator is a piece of code that runs during compilation and can inspect your program to produce additional source files that are compiled together with the rest of your code"

from docs.microsoft.com

Un **Source Generator** è una componente del compilatore che ti consente di fare due cose principali:

- Analizzare il codice sorgente del tuo progetto in termini di sintassi e i modelli semantici in fase di compilazione, proprio come già fanno gli analizzatori di codice.

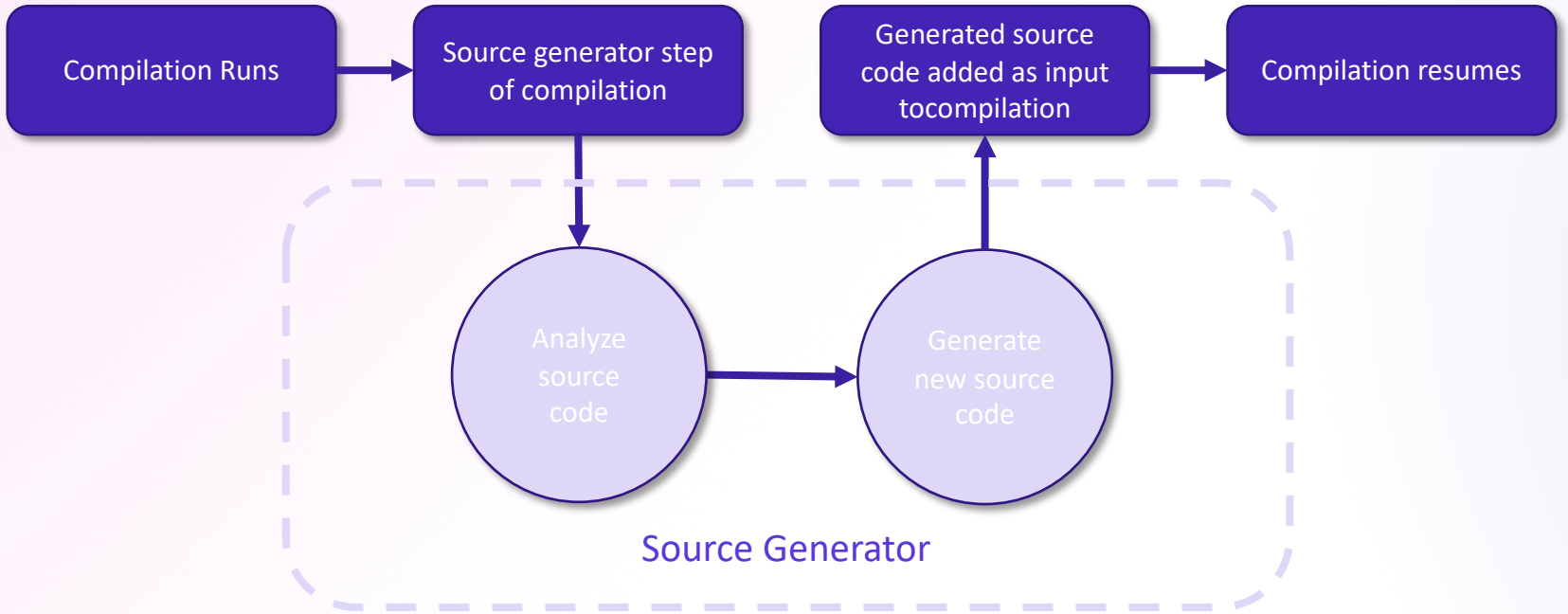
- Generare file sorgenti che possono essere aggiunti al contesto del compliatore durante la compilazione stessa. In altre parole, puoi fornire codice sorgente aggiuntivo come input per una compilazione mentre il codice viene compilato.



Escher, Mani che disegnano

Source Generator

Si tratta di una feature nativa del compilatore Roslyn e fa parte della pipeline di compilazione.

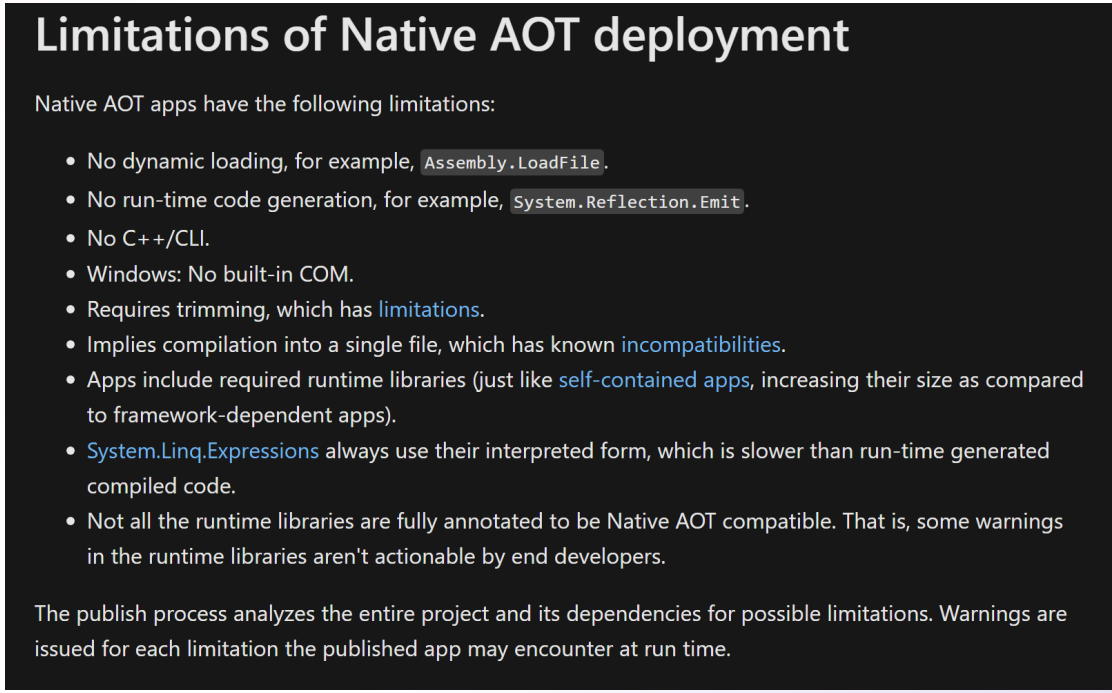


https://devblogs.microsoft.com/dotnet/introducing-c-source-generators/

Source Generator - Perchè sono usati

- Alternativa alla Reflection
  - Performance (Compile-time not runtime)
  - Risolvere problemi problemi di AOT Trimming
  - Maggiore chiarezza e facilità di debug

- Altri casi d'uso:
  - Dobbiamo costantemente scrivere codice standard e potremmo automatizzarlo utilizzando SG
  - Abbiamo alcuni file di dati e dobbiamo analizzarli per avere del codice fortemente tipizzato
  - Abbiamo un nostro metalinguaggio da eseguire o compilare
  - Abbiamo necessità di fare discovery di oggetti e assembly allo start-up
  - Abbiamo del codice basato sulla riflessione che potrebbe essere lento a runtime



Limitations of Native AOT deployment

Native AOT apps have the following limitations:

- No dynamic loading, for example, `Assembly.LoadFile`.
- No run-time code generation, for example, `System.Reflection.Emit`.
- No C++/CLI.
- Windows: No built-in COM.
- Requires trimming, which has limitations.
- Implies compilation into a single file, which has known incompatibilities.
- Apps include required runtime libraries (just like self-contained apps, increasing their size as compared to framework-dependent apps).
- System.Linq.Expressions always use their interpreted form, which is slower than run-time generated compiled code.
- Not all the runtime libraries are fully annotated to be Native AOT compatible. That is, some warnings in the runtime libraries aren't actionable by end developers.

The publish process analyzes the entire project and its dependencies for possible limitations. Warnings are issued for each limitation the published app may encounter at run time.

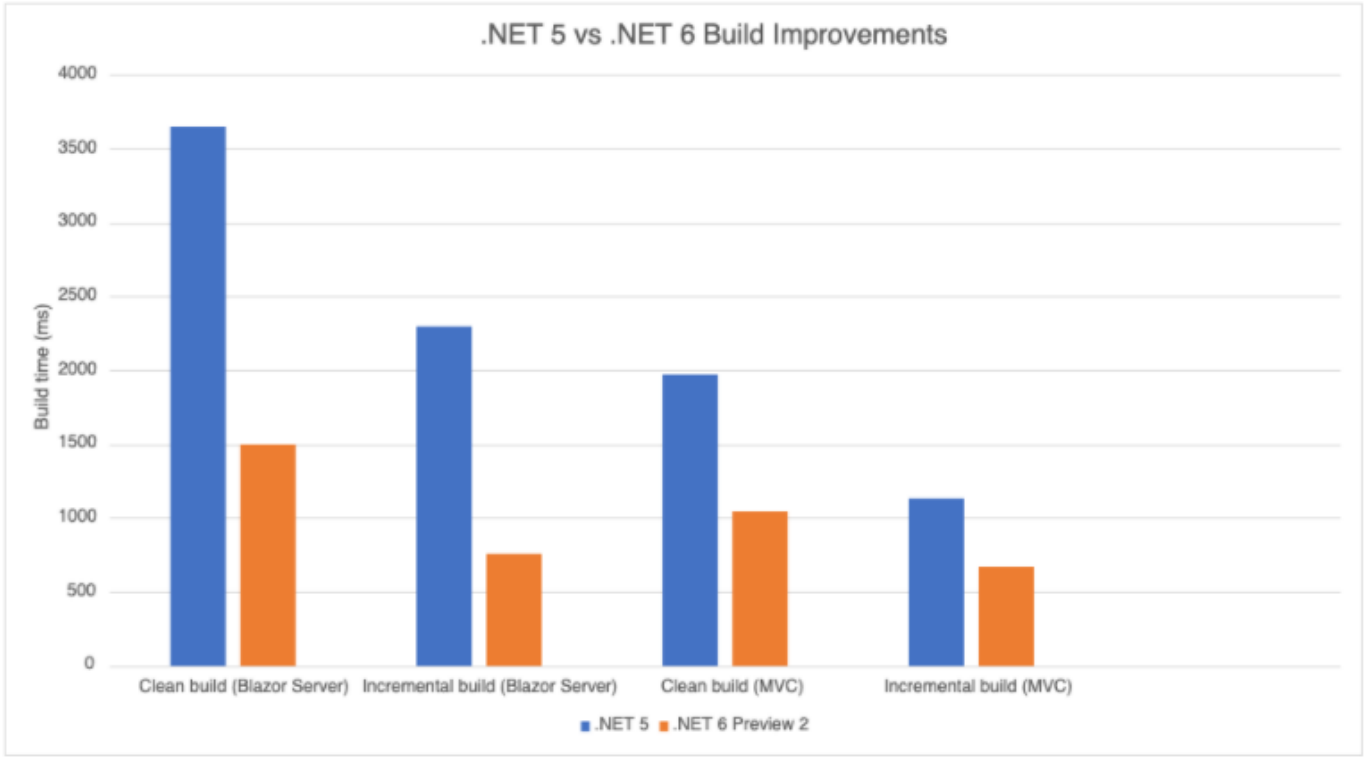
Demo

Razor & Source Generators

Da .NET 6 il compilatore Razor è stato implementato utilizzando generatori di sorgenti C#.

L'uso dei generatori di sorgenti semplifica il compilatore Razor e accelera notevolmente i tempi di compilazione.

The following graph shows the build time improvements when using the new Razor compiler to build the default Blazor Server and MVC templates:



System.Text.Json & Source Generators

Da .NET 6 è possibile un nuovo approccio alla serializzazione JSON per che consiste nel spostare a compile-time l'ispezione dei tipi serializzabili JSON, tramite Source Generator.

Serialization

	Elapsed time (ms)	Allocated (KB)
Serializer	28.25	1110.00
SrcGenSerializer	12.75	563.00

Deserialization

	Elapsed time (ms)	Allocated (KB)
Serializer	24.75	1457.00
SrcGenSerializer	15.50	1025.00

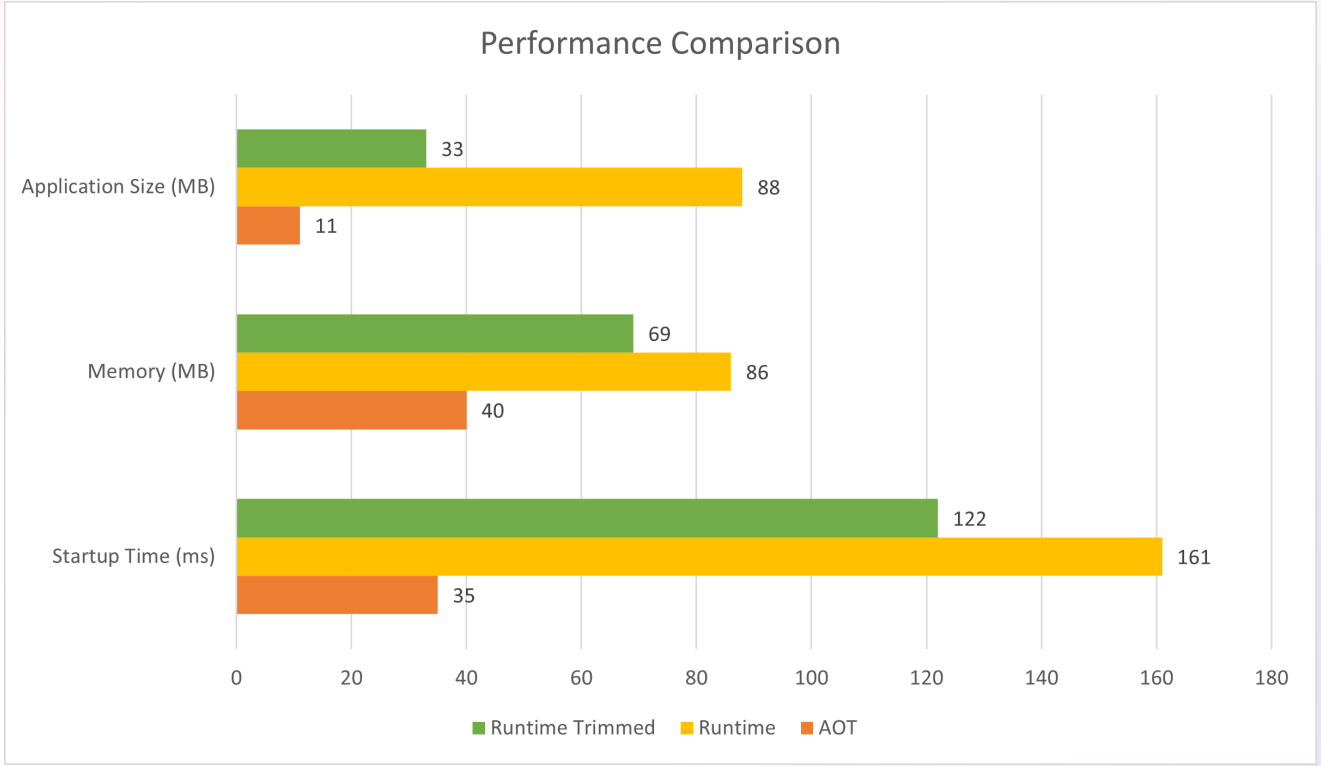
	Requests/sec	Requests
net5.0	243,000	3,669,151
net6.0	260,928	3,939,804
net6.0 + JSON source gen	364,224	5,499,468

ASP.NET Core support for native AOT

Grazie all'utilizzo dei Source Generator, con .NET 8 alcune features di ASP.NET Core sono adesso compatibili con la compilazione AOT:

- gRpc
- Minimal APIs (partial)

...



https://learn.microsoft.com/en-us/aspnet/core/fundamentals/native-aot?view=aspnetcore-8.0

https://andrewlock.net/exploring-the-dotnet-8-preview-exploring-the-new-minimal-api-source-generator/

Source Generator tips

Prerequisiti:
- C# 9.0+ (SDK 5.0.100+)
- Microsoft Visual Studio 16.8.0+

Limiti:
- I source generator non consentono di riscrivere/modificare il codice sorgente dell'utente.
  (È possibile solo aggiungere nuovo sorgente C# anche sfruttando le «partial class» e i «partial method»).
- Esecuzione non ordinata, ogni generatore vedrà la stessa compilazione di input, senza accesso ai file creati da altri generatori di origine.

Deploy:
- E' possibile distribuire il nostro SG tramite NuGet

```
<PropertyGroup>
  <GeneratePackageOnBuild>true</GeneratePackageOnBuild> <!-- Generates a package at build -->
  <IncludeBuildOutput>false</IncludeBuildOutput> <!-- Do not include the generator as a lib dependency -->
</PropertyGroup>

<ItemGroup>
  <!-- Package the generator in the analyzer directory of the nuget package -->
  <None Include="$(OutputPath)\$(AssemblyName).dll" Pack="true" PackagePath="analyzers/dotnet/cs" Visible="false" />
</ItemGroup>
```

Debug:
- E' possibile usare il debugger utilizzando il metodo Debugger.Launch(). (Viene aperta un'altra istanza di VS)
- Con VS 2022 e il setting a livello di progetto
  <IsRoslynComponent>true</IsRoslynComponent>
  si può creare un profilo di debug specifico per gli Analyzer ed effettuare il debug del progetto SG nella stessa istanza
- E' possibile forzare la generazione effettiva dei file sorgente e la cartella di destinazione

```
<PropertyGroup>
  <EmitCompilerGeneratedFiles>true</EmitCompilerGeneratedFiles>
  <CompilerGeneratedFilesOutputPath>GeneratedFiles</CompilerGeneratedFilesOutputPath>
</PropertyGroup>
```

```
<ItemGroup>
    <AdditionalFiles Include="People.csv" CsvLoadType="Startup" />
    <AdditionalFiles Include="Cars.csv" CsvLoadType="OnDemand" CacheObjects="true" />
</ItemGroup>
```

```
// find anything that matches our files
var myFiles = context.AnalyzerOptions.AdditionalFiles.Where(at => at.Path.EndsWith(".xml"));
```

Resources

- Source Generators (Official)
  https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/source-generators-overview

  roslyn/docs/features/source-generators.cookbook.md at main · dotnet/roslyn (github.com)

- A list of C# Source Generators (Thanks to Amadeusz Sadowski)
  https://github.com/amis92/csharp-source-generators

- Demo Project Source Generator
  https://github.com/bortolin/NetConfSG.Example

- Example without Incremental Source
  https://github.com/bortolin/XeDotNet.SourceGeneratorExample

Marco Bortolin

email: m.bortolin@hunext.com
twitter: @marcobortolin

https://github.com/bortolin
https://www.linkedin.com/in/marcobortolin