Coding C# with C#

with AL Rodriguez

Duende

About Me

- @ProgrammerAL
- https://ProgrammerAL.com
- Customer Success Engineer at Duende

Duende

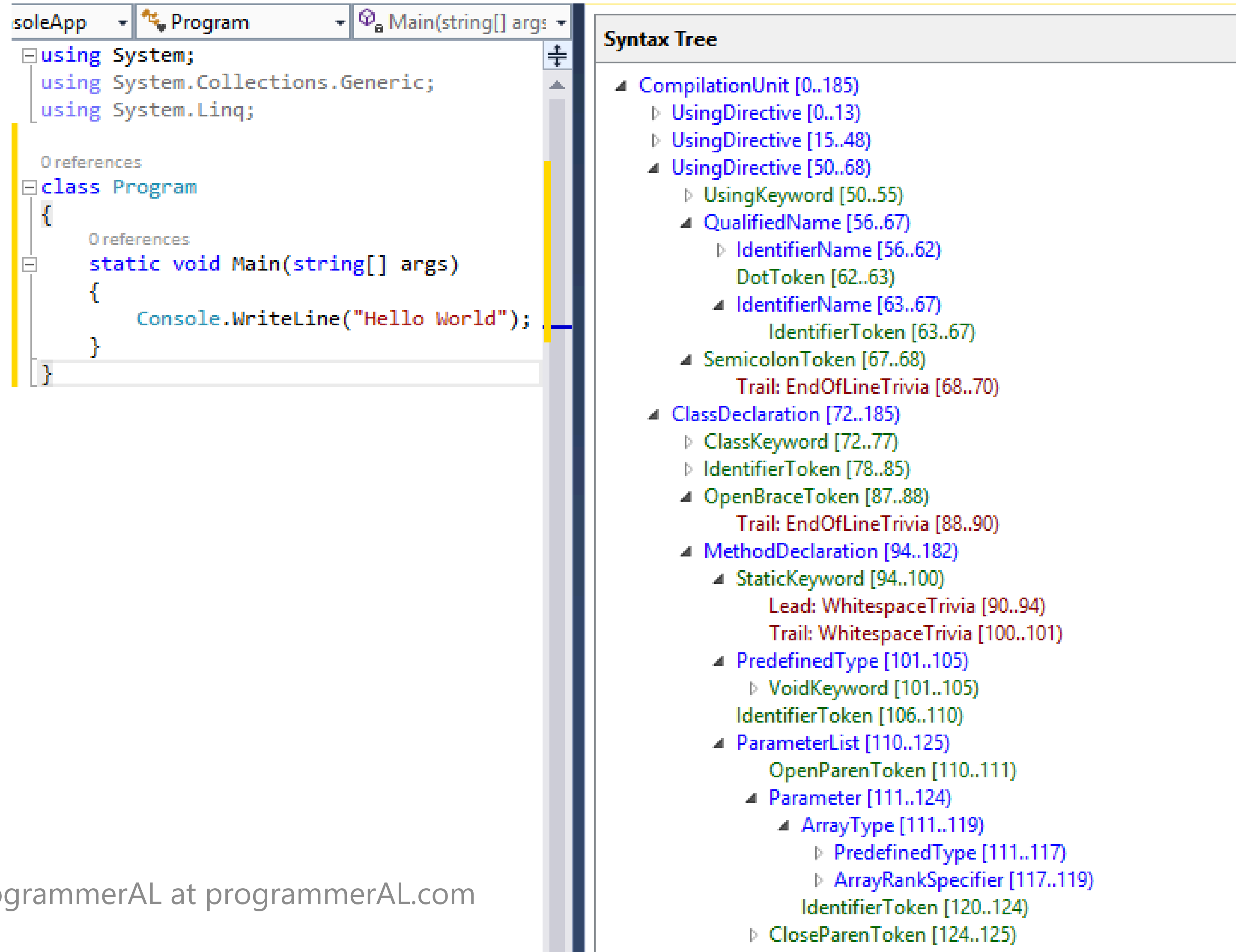
Why are we here? To Review!

- C# analyzers
- C# source code generation
- How these are already built into C#
- Note: Everything Mentioned is Free

History Recap: C# Compiler

- 2000: Compiler written in C++
  - Big-Bang features added each update
  - Tech debt added over time
- 2011: Roslyn Compiler released
  - Full rewrite in C#
    - With knowledge of how C# is used
  - Added hooks into the compilation process

Roslyn Syntax Tree



```
using System;
using System.Collections.Generic;
using System.Linq;

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World");
    }
}
```

Syntax Tree

```
CompilationUnit [0..185)
    UsingDirective [0..13)
    UsingDirective [15..48)
    UsingDirective [50..68)
        UsingKeyword [50..55)
        QualifiedName [56..67)
            IdentifierName [56..62)
            DotToken [62..63)
            IdentifierName [63..67)
                IdentifierToken [63..67)
        SemicolonToken [67..68)
            Trail: EndOfLineTrivia [68..70)
    ClassDeclaration [72..185)
        ClassKeyword [72..77)
        IdentifierToken [78..85)
        OpenBraceToken [87..88)
            Trail: EndOfLineTrivia [88..90)
        MethodDeclaration [94..182)
            StaticKeyword [94..100)
                Lead: WhitespaceTrivia [90..94)
                Trail: WhitespaceTrivia [100..101)
            PredefinedType [101..105)
                VoidKeyword [101..105)
            IdentifierToken [106..110)
            ParameterList [110..125)
                OpenParenToken [110..111)
                Parameter [111..124)
                    ArrayType [111..119)
                        PredefinedType [111..117)
                        ArrayRankSpecifier [117..119)
                    IdentifierToken [120..124)
                CloseParenToken [124..125)
```

Duende

Helpful Tools: Roslyn Syntax Visualization

- https://sharplab.io
- Built into Rider: Syntax Tree Visualizer
- Extension in Visual Studio: https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/syntax-visualizer

Rosyln Analyzer

- Keyword: Analyzer
- Checks code for rules
    - Errors, Warnings, Suggestions, etc
- Reads code syntax using syntax tree

Are they used often? Yes!

- Code analysis built-in is all Roslyn Analyzers
  - https://learn.microsoft.com/en-us/dotnet/fundamentals/code-analysis/overview
- Many 3rd Party Analyzer NuGet packages
  - SonarAnalyzer.CSharp
  - Roslynator.Analyzers
  - StyleCop.Analyzers
  - SerilogAnalyzer
  - xunit.analyzers
  - MongoDB.Analyzer

Demo Time

- Existing Analyzer:
  - https://github.com/ProgrammerAL/required-auth.analyzer
- Scenario:
  - Require `[Authorize]` / `[Anonymous]` attribute in controller files

Extra Credit: Roslyn Analyzer Code Fix

- Analyzers can edit code to comply with the rule
- Analyzer generates the code change, user approves it

C# Source Generator

- Code created in-memory at compile time
    - Can write to files if flag enabled in project, all or nothing
- Written using same Roslyn Syntax Tree API as Analyzers
- Additive only, cannot modify code

Are they used often? Yes!

- .NET Team adding them for AoT support, or to remove reflection, for performance
- Public List:
  - https://github.com/amis92/csharp-source-generators

Useful Source Generator Links

- Cookbook: https://github.com/dotnet/roslyn/blob/main/docs/features/incremental-generators.cookbook.md
- Andrew Lock's blog series: https://andrewlock.net/series/creating-a-source-generator

2 Types of Source Generators

- Incremental
  - Always do this one
  - Added in .NET 6
  - v2 of the API
- Non-Incremental
  - Added in .NET 5
  - v1 of the API

Demo Time

- Existing Generator:
  - https://github.com/ProgrammerAL/public-interface-generator
- Scenario:
  - Generate interface code from a class
  - Only use it for internal interfaces needed for unit tests

Duende

Review

- Add custom hooks to compiler
    - Roslyn Analyzers to check code
    - Source Generators to add code
- API is specific to parsing code tree

Content

Feedback

Duende