

Roslyn Analyzers

Gérald Barré

Website: <https://www.meziantou.net>

Bluesky: [@meziantou.net](https://bsky.app/profile/meziantou.net)

Mastodon: [@meziantou@hachyderm.io](https://hachyderm.io/@meziantou)



What's a Roslyn Analyzer?

The screenshot shows the Visual Studio IDE with a code editor and a warning tooltip. The code in the editor is `_ = new int[0];`. A tooltip is displayed over the code, showing a warning icon and the text: **CA1825** Avoid unnecessary zero-length array allocations. Use `Array.Empty<int>()` instead.

On the left side of the tooltip, there is a context menu with the following options:

- Use Array.Empty
- Convert to 'Program.Main' style program
- Suppress or configure issues

The main body of the tooltip shows a preview of the code changes. It displays three lines: `...`, `- _ = new int[0];`, and `+ _ = Array.Empty<int>();`. The `new` keyword and the `[0]` in the original code are highlighted with red boxes, and the `Array.Empty<int>()` in the new code is highlighted with green boxes.

At the bottom of the tooltip, there are links for [Preview changes](#), [Fix all occurrences in: Document](#), [Project](#), [Solution](#), [Containing Member](#), and [Containing Type](#).

Goals

- Understand what are Roslyn analyzers
- Be able to write your first analyzer

What is Roslyn?

- C# and VB.NET compilers
- First CTP in 2011, first RTM in VS2015
- Provide language services (completion, refactoring, formatter...)
- Enables building code analysis tools
- Open source
 - <https://github.com/dotnet/roslyn>
 - <https://github.com/dotnet/csharplang>
 - <https://github.com/dotnet/vblang>

Roslyn Analyzer

- Static analysis
- Detect patterns in code and optionally suggest fixes
- Can report suggestions, warnings, errors
- Integrated in build process
- Integrated in IDE
 - Detect issues when writing code => save time in code reviews

Examples

New language features

 `string[] strings = { "A", "B", "C" };`

Convert to interpolated string

Use collection expression ▶

Convert to 'Program.Main' style program

Use discard '_'

Suppress or configure issues ▶

 **IDE0300** Collection initialization can be simplified

...

- `string[] strings = { "A", "B", "C" };`

+ `string[] strings = ["A", "B", "C"];`

Preview changes

Fix all occurrences in: [Document](#) | [Project](#) | [Solution](#) | [Containing Member](#) | [Containing Type](#)

New .NET features

The screenshot shows a Visual Studio code editor with a C# snippet: `if (a.Count == 0)`. A lightbulb icon in the top left corner of the editor triggers a dropdown menu. This menu contains four options: 'Prefer IsEmpty over Count' (highlighted in blue), 'Wrap expression', 'Convert to 'Program.Main' style program', and 'Suppress or configure issues'. To the right of the code, a green warning banner displays the text: `CA1836: Prefer 'IsEmpty' over 'Count' to determine whether the object is empty`. Below this banner, a detailed tooltip is visible. It features a dropdown arrow, a warning icon, and the rule ID `CA1836`. The main text of the tooltip reads: 'Prefer 'IsEmpty' over 'Count' to determine whether the object is empty'. Below this text, a code diff is shown: the original line `- if (a.Count == 0)` is crossed out with a red background, and the suggested line `+ if (a.IsEmpty)` is highlighted with a green background. The tooltip also shows the opening curly brace of the if-statement and an ellipsis indicating the rest of the code block. At the bottom of the tooltip, there is a section titled 'Preview changes' with the text 'Fix all occurrences in:' followed by a series of blue links: 'Document', 'Project', 'Solution', 'Containing Member', and 'Containing Type'.

if (a.Count == 0)

CA1836: Prefer 'IsEmpty' over 'Count' to determine whether the object is empty

Prefer IsEmpty over Count

Wrap expression

Convert to 'Program.Main' style program

Suppress or configure issues

CA1836 Prefer 'IsEmpty' over 'Count' to determine whether the object is empty

...

- if (a.Count == 0)

+ if (a.IsEmpty)

{

...

Preview changes

Fix all occurrences in: [Document](#) | [Project](#) | [Solution](#) | [Containing Member](#) | [Containing Type](#)

Potential bugs

```
static Task Sample(Func<Task> action)
{
    using var scope = new Scope();
    return action();
}
```

MA0100: Await task before disposing of resources

Show potential fixes (Alt+Enter or Ctrl+.)

Potential bugs

  `if (value.StartsWith("."))`

Add StringComparison.Ordinal

Add StringComparison.OrdinalIgnoreCase

Convert to interpolated string

Introduce local

Convert to 'Program.Main' style program

 Fix with Copilot

Suppress or configure issues

 **MA0074** Use an overload of 'StartsWith' that has a StringComparison parameter

...

- `if (value.StartsWith("."))`

+ `if (value.StartsWith(".", StringComparison.Ordinal))`

{

...

Preview changes

Fix all occurrences in: [Document](#) | [Project](#) | [Solution](#) | [Containing Member](#) | [Containing Type](#)

Security

```
ZipArchiveEntry zipArchiveEntry = default!;
```

```
zipArchiveEntry.ExtractToFile(zipArchiveEntry.FullName);
```

 (local variable) ZipArchiveEntry zipArchiveEntry

'zipArchiveEntry' is not null here.

CA5389: When creating path for 'void ZipFileExtensions.ExtractToFile(ZipArchiveEntry source, string destinationFileName) in method <top-level-statements-entry-point>' from relative archive item path to extract file and the source is an untrusted zip archive, make sure to sanitize relative archive item path 'string ZipArchiveEntry.FullName in method <top-level-statements-entry-point>'

Usage of libraries

`Assert.True(collection.Contains("sample"));`

Use Assert.Contains ▶

Add StringComparer.Ordinal

Add StringComparer.OrdinalIgnoreCase

Convert to interpolated string

Convert to 'Program.Main' style program

Suppress or configure issues ▶

 **xUnit2017** Do not use Assert.True() to check if a value exists in a collection. Use Assert.Contains instead.

...

- `Assert.True(collection.Contains("sample"));`

+ `Assert.Contains("sample", collection);`

Preview changes

Fix all occurrences in: [Document](#) | [Project](#) | [Solution](#) | [Containing Member](#) | [Containing Type](#)

Usage of libraries

```
[Theory]
[InlineData("test")]
public void Foo(int value)
{
    Assert.Equal(1, value);
}
```

Usage of libraries

 `Console.WriteLine("");`

Convert static call to AnsiConsole to Spectre.Console.AnsiConsole ▶

Convert to interpolated string

Convert to 'Program.Main' style program

Suppress or configure issues ▶

⚠ Spectre1000: Use AnsiConsole instead of System.Console

⚠ Spectre1000 Use AnsiConsole instead of System.Console

...

- `Console.WriteLine("");`

+ `AnsiConsole.WriteLine("");`

Preview changes

Fix all occurrences in: [Document](#) | [Project](#) | [Solution](#) | [Containing Member](#) | [Containing Type](#)

Ban symbols

```
_ = DateTime.Now;
```

 `readonly struct System.DateTime`

Represents an instant in time, typically expressed as a date and time of day.

RS0030: The symbol 'DateTime.Now' is banned in this project: Use System.DateTime.UtcNow instead

Some Roslyn analyzers

- Microsoft.CodeAnalysis.NetAnalyzers
- Microsoft.CodeAnalysis.BannedApiAnalyzers
- xunit.analyzers
- FakeItEasy.Analyzer
- Spectre.Console.Analyzer
- Roslynator.Analyzers
- Meziantou.Analyzer 😊

Roslyn concepts (for analyzers)

Compilation

- All information needed to compile a project
 - Compiler options
 - Source files
 - Referenced assemblies
 - ❌ No concept of TFM or NuGet packages
 - ❌ *Not aware of other projects in the solution*
- Useful to access available types
 - GetTypeByMetadataName("System.Action`1")
 - GetSpecialType(SpecialType.System_String)

Syntax Tree

- Represent the file parsed by the compiler
- Every bit of information in a code file is represented in the tree
- Syntax tree is immutable
- May represent code that is not valid
 - ⚠ It's more common than valid code
- Different syntax nodes for C# and VB.NET

```
└─ ClassDeclaration [391..2117]
  │   └─ AttributeList [391..433]
  │   └─ PublicKeyKeyword [435..441]
  │   └─ SealedKeyword [442..448]
  │   └─ ClassKeyword [449..454]
  │   └─ IdentifierToken [455..480]
  │   └─ BaseList [481..501]
  │   └─ OpenBraceToken [503..504]
  │   └─ FieldDeclaration [510..802]
  │   └─ PropertyDeclaration [810..915]
  │   └─ MethodDeclaration [923..1500]
  │       └─ MethodBodyOperation [923..1500]
  │       └─ PublicKeyKeyword [923..929]
  │       └─ OverrideKeyword [930..938]
  │       └─ PredefinedType [939..943]
  │       └─ IdentifierToken [944..954]
  │       └─ ParameterList [954..979]
  │       └─ Block [985..1500]
  │           └─ OpenBraceToken [985..986]
  │           └─ ExpressionStatement [996..1032]
  │           └─ ExpressionStatement [1042..1114]
  │               └─ InvocationExpression [1042..1113]
  │                   └─ SimpleMemberAccessExpression [1042..1080]
  │                   └─ ArgumentList [1080..1113]
  │                   └─ SemicolonToken [1113..1114]
  │               └─ ExpressionStatement [1126..1493]
  │               └─ CloseBraceToken [1499..1500]
  │       └─ MethodDeclaration [1508..2114]
```

Semantic Model

- Allow to get information about nodes in the syntax tree
 - What's the type of a variable?
 - Which method is called?
 - What's the symbol created by a syntax node?
 - Is this value constant?
 - May a value be null? (Nullable Reference Types)
 - Is a type/method/variable accessible from a location?

ISymbol

- Represents a symbol (namespace, class, method, parameter, variable, assembly, etc.) exposed by the compiler
- A symbol can be from your code or references
- Signature only
- To Compare symbols
 - `SymbolEqualityComparer.Default`
 - `SymbolEqualityComparer.IncludeNullability`

Operation

- Abstraction from the syntax
- Common types for C# and VB.NET
- Mix of Syntax tree and Semantic model
- Can only represent the method body and attributes
- C# has many ways to do the same thing
 - `new int[0]`
 - `new int[] {}`
 - `int[] sample = { };`
 - `const int length = 0; new int[length]`

=> `IArrayCreationOperation` with 1 dimension of constant length 0

DiagnosticDescriptor

```
private static readonly DiagnosticDescriptor Rule = new(  
    id: "Sample001",  
    title: "Replace new System.EventArgs",  
    messageFormat: "Replace new System.EventArgs",  
    description: "Replace new System.EventArgs with EventArgs.Empty for better performance",  
    category: "Performance",  
    defaultSeverity: DiagnosticSeverity.Info,  
    isEnabledByDefault: true,  
    helpLinkUri: "https://www.meziantou.net/");
```

DiagnosticAnalyzer / CodeFixProvider

- DiagnosticAnalyzer
 - Register callbacks to indicate Roslyn what to analyze (SyntaxNode, Operation, Symbol)
 - Report diagnostic in the code (Rule descriptor + location)
- CodeFixProvider
 - Indicates which rule ids are supported
 - Transform a solution/document to fix a diagnostic

Let's create an analyzer

Requirements

- Detect: `new System.EventArgs()`
- Fix: `System.EventArgs.Empty`

Workloads Individual components Language packs Installation locations**Desktop development with C++** ☒

Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.

**WinUI application development** ☐

Build applications for the Windows platform using WinUI with C# or optionally C++.

Gaming (2)**Game development with Unity** ☐

Create 2D and 3D games with Unity, a powerful cross-platform development environment.

**Game development with C++** ☐

Use the full power of C++ to build professional games powered by Unreal or Cocos2d.

Other Toolsets (2)**Linux and embedded development with C++** ☐

Create and debug applications running in a Linux environment or on an embedded device.

**Visual Studio extension development** ☒

Create add-ons and extensions for Visual Studio, including new commands, code analyzers and tool windows.

Installation details▸ **Desktop development with C++**▾ **Visual Studio extension development**

▾ Included

- ✓ Visual Studio SDK
- ✓ Visual Studio extension development prer...

▾ Optional

- ☒ .NET profiling tools
- ☒ IntelliCode
- ☒ IntelliTrace
- ☒ Text Template Transformation
- ☒ .NET Compiler Platform SDK
- ☒ GitHub Copilot
- ☒ Developer Analytics tools
- ☐ Modeling SDK

▸ **Individual components****Location**

C:\Program Files\Microsoft Visual Studio\2022\Preview

Remove out-of-support components

By continuing, you agree to the [license](#) for the product edition you selected. We also offer the ability to download other software. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

Total space required 0 B

Install while downloading ▾

Close

Create a new project

Recent project templates

- | | |
|--|----|
|  Console App | C# |
|  .NET Aspire Starter App | C# |
|  xUnit.net v3 Test Project (xUnit.net Team) | C# |
|  ASP.NET Core Web API | C# |
|  Worker Service | C# |
|  Class Library | C# |
|  Analyzer with Code Fix (.NET Standard) | C# |
|  MSTest Test Project | C# |
|  xUnit Test Project | C# |
|  WPF Application | C# |

[Clear all](#)

Analyzer with Code Fix (.NET Standard)

Create a C# analyzer that comes with a code fix and generates diagnostics. The analyzer can be deployed as either a NuGet package or a VSIX extension.

[C#](#)[Windows](#)[Linux](#)[macOS](#)[Roslyn](#)[Extensions](#)

Analyzer with Code Fix (.NET Standard)

Create a Visual Basic analyzer that comes with a code fix and generates diagnostics. The analyzer can be deployed as either a NuGet package or a VSIX extension.

[Visual Basic](#)[Windows](#)[Linux](#)[macOS](#)[Roslyn](#)[Extensions](#)

Not finding what you're looking for?

[Install more tools and features](#)[Back](#)[Next](#)



Solution 'SampleAnalyzer' (5 of 5 projects)

- ▲  **SampleAnalyzer**
 - ▷  Dependencies
 - ▷  Resources.resx
 - ▷  SampleAnalyzerAnalyzer.cs
- ▲  SampleAnalyzer.CodeFixes
 - ▷  Dependencies
 - ▷  CodeFixResources.resx
 - ▷  SampleAnalyzerCodeFixProvider.cs
- ▷  SampleAnalyzer.Package
- ▲  SampleAnalyzer.Test
 - ▷  Dependencies
 - ▷  Verifiers
 - ▷  SampleAnalyzerUnitTests.cs
- ▷  SampleAnalyzer.Vsix

Sharplab.io

```
using System;  
  
_ = new EventArgs();
```

-  CompilationUnit
 - ⦿ *Operation: MethodBodyOperation*
- +  UsingDirective
-  GlobalStatement
 -  Statement: ExpressionStatement
 - ⦿ *Operation: ExpressionStatement*
 -  Expression: SimpleAssignmentExpression
 - + ⦿ *Operation: SimpleAssignment*
 - +  Left: IdentifierName
 - +  OperatorToken: EqualsToken
 -  Right: ObjectCreationExpression
 - ⦿ *Operation: ObjectCreation*
 - Constructor: System.EventArgs.EventArgs()
 - Type: System.EventArgs
 - +  NewKeyword: NewKeyword
 -  Type: IdentifierName
 -  Identifier: EventArgs IdentifierToken
 - +  ArgumentList: ArgumentList
 - +  SemicolonToken: ; SemicolonToken
 - +  EndOfFileToken: EndOfFileToken

```
[DiagnosticAnalyzer(LanguageNames.CSharp)]
public class SampleAnalyzer : DiagnosticAnalyzer
{
    private static readonly DiagnosticDescriptor Rule = new ...;

    public override ImmutableArray<DiagnosticDescriptor> SupportedDiagnostics => ImmutableArray.Create(Rule);

    public override void Initialize(AnalysisContext context)
    {
        context.ConfigureGeneratedCodeAnalysis(GeneratedCodeAnalysisFlags.None);
        context.EnableConcurrentExecution();

        context.RegisterOperationAction(AnalyzeObjectCreation, OperationKind.ObjectCreation);

        static void AnalyzeObjectCreation(OperationAnalysisContext context)
        {
            var eventArgsSymbol = context.Compilation.GetTypeByMetadataName("System.EventArgs");
            if (eventArgsSymbol == null)
                return;

            var objectCreation = (IObjectCreationOperation)context.Operation;
            if (objectCreation.Arguments.Length > 0)
                return;

            if (SymbolEqualityComparer.Default.Equals(objectCreation.Type, eventArgsSymbol))
            {
                context.ReportDiagnostic(Diagnostic.Create(Rule, objectCreation.Syntax.GetLocation()));
            }
        }
    }
}
```

```

[ExportCodeFixProvider(LanguageNames.CSharp, Name = nameof(MsDevMtlSampleCodeFixProvider)), Shared]
public class MsDevMtlSampleCodeFixProvider : CodeFixProvider
{
    public sealed override ImmutableArray<string> FixableDiagnosticIds
        => ImmutableArray.Create("Sample001");

    public sealed override async Task RegisterCodeFixesAsync(CodeFixContext context)
    {
        var root = await context.Document.GetSyntaxRootAsync(context.CancellationToken).ConfigureAwait(false);
        var diagnostic = context.Diagnostics.First();
        var diagnosticSpan = diagnostic.Location.SourceSpan;

        var nodeToFix = root.FindNode(diagnosticSpan, getInnermostNodeForTie: true);

        context.RegisterCodeFix(
            CodeAction.Create(
                title: "Replace with EventArgs.Empty",
                createChangedDocument: cancellationToken => Fix(context, nodeToFix, cancellationToken),
                equivalenceKey: "CodeFixTitle"),
            diagnostic);
    }

    private async Task<Document> Fix(CodeFixContext context, SyntaxNode nodeToFix, CancellationToken cancellationToken)
    {
        var editor = await DocumentEditor.CreateAsync(context.Document, cancellationToken).ConfigureAwait(false);
        var generator = editor.Generator;

        var typeSymbol = editor.SemanticModel.Compilation.GetTypeByMetadataName("System.EventArgs");
        if (typeSymbol is null)
            return context.Document;

        var newExpression = generator.MemberAccessExpression(generator.TypeExpression(typeSymbol), "Empty");
        editor.ReplaceNode(nodeToFix, newExpression);
        return editor.GetChangedDocument();
    }
}

```


Roslyn quoter

<https://github.com/KirillOsenkov/RoslynQuoter>

```
class C
{
    public void Toto()
    {
    }
}
```

Parse as: File 

- ☐ Open parenthesis on a new line
- ☐ Closing parenthesis on a new line
- ☐ Preserve original whitespace
- ☐ Keep redundant API calls
- ☐ Do not require 'using static Microsoft.CodeAnalysis.CSharp.SyntaxFactory;'

Get Roslyn API calls to generate this code!

Get Roslyn API calls to generate this code as LINQPad file!

```
CompilationUnit()  
.WithMembers(  
    SingletonList<MemberDeclarationSyntax>(  
        ClassDeclaration("C")  
        .WithMembers(  
            SingletonList<MemberDeclarationSyntax>(  
                MethodDeclaration(  
                    PredefinedType(  
                        Token(SyntaxKind.VoidKeyword)),  
                        Identifier("Toto"))  
                .WithModifiers(  
                    TokenList(  
                        Token(SyntaxKind.PublicKeyword)))  
                .WithBody(  
                    Block()))))  
        .NormalizeWhitespace())  
    )  
)
```

```
[TestMethod]
```

```
public async Task TestMethod3()
```

```
{
```

```
    var test = new CSharpCodeFixTest<MsDevMtlSampleAnalyzer, MsDevMtlSampleCodeFixProvider, DefaultVerifier>();
```

```
    test.ReferenceAssemblies = ReferenceAssemblies.NetStandard.NetStandard20;
```

```
    test.TestCode = /*lang=c#-test*/ """
```

```
        using System;
```

```
        class Sample
```

```
        {
```

```
            void A()
```

```
            {
```

```
                _ = [|new EventArgs()|];
```

```
            }
```

```
        }
```

```
        """;
```

```
    test.FixedCode = /*lang=c#-test*/ """
```

```
        using System;
```

```
        class Sample
```

```
        {
```

```
            void A()
```

```
            {
```

```
                _ = EventArgs.Empty;
```

```
            }
```

```
        }
```

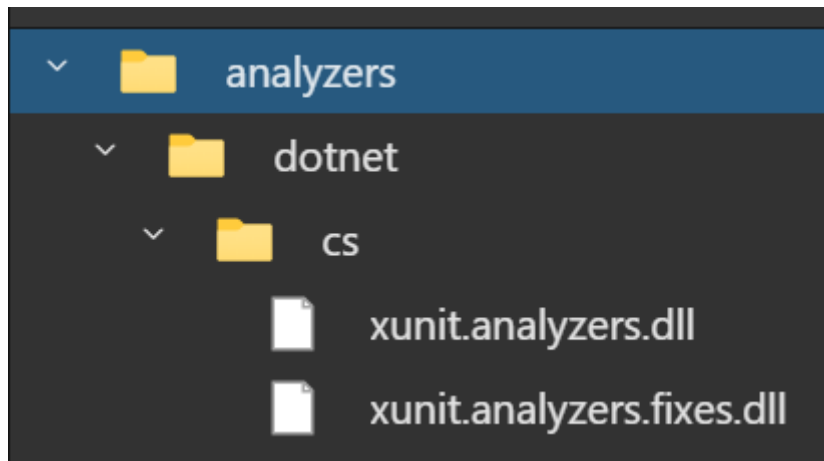
```
        """;
```

```
    await test.RunAsync();
```

```
}
```

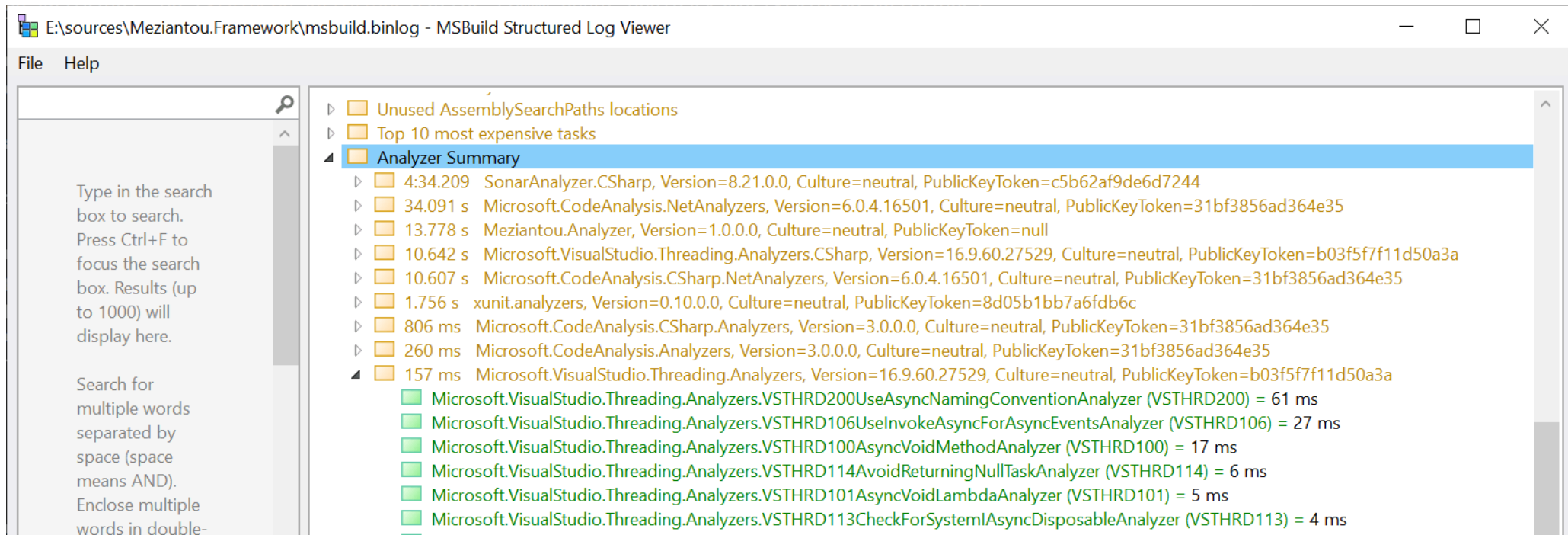
Package a Roslyn analyzer

- DLL must target .NET Standard 2.0
- NuGet package with the following structure



Performance

- Be careful with performance
 - Can increase build time
 - Can slow down IDE
 - `<ReportAnalyzer>true</ReportAnalyzer>` + `dotnet build /bl`



Roslyn version

- Select the right version of Roslyn
 - 4.12.0 - VS2022 (17.12)
 - 3.11.0 - VS2019 (16.11)
 - See <https://learn.microsoft.com/en-us/visualstudio/extensibility/roslyn-version-support>

Configuration

[*.cs]

dotnet_diagnostic.Sample0001.severity = error