

Roslyn Analyzers

Gérald Barré

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

Bluesky: [@meziantou.net](#)

Mastodon: [@meziantou@hachyderm.io](#)



What's a Roslyn Analyzer?

5 `_ = new int[0];`

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

i CA1825 Avoid unnecessary zero-length array allocations. Use `Array.Empty<int>()` instead.

...

```
- _ = new int[0];  
+ _ = Array.Empty<int>();
```

Preview changes

Fix all occurrences in: Document | Project | Solution | Containing Member | 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/csharpplang>
 - <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

i 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

`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

The screenshot shows an IDE interface with a code editor and a context menu. The code in the editor is `if (value.StartsWith("."))`. A warning icon (lightbulb) is visible in the top left corner of the editor. A warning message is displayed in the top right: `MA0074: Use an overload of 'StartsWith' that has a Str`. A context menu is open over the code, listing several options: `Add StringComparison.Ordinal` (highlighted), `Add StringComparison.OrdinalIgnoreCase`, `Convert to interpolated string`, `Introduce local`, `Convert to 'Program.Main' style program`, `Fix with Copilot`, and `Suppress or configure issues`. To the right of the context menu, a detailed warning message is shown: `MA0074 Use an overload of 'StartsWith' that has a StringComparison parameter`. Below this message, a preview of the code change is displayed: `- if (value.StartsWith("."))` is crossed out, and `+ if (value.StartsWith(".", StringComparison.Ordinal))` is added. The new code is highlighted in green. Below the preview, there are links for `Preview changes`, `Fix all occurrences in: Document | Project | Solution | Containing Member | Containing Type`.

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

MA0074: Use an overload of 'StartsWith' that has a Str

- 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

The screenshot shows a code editor with the following elements:

- Code Editor:** `Console.WriteLine("");` with a lightbulb icon on the left and a warning icon on the right.
- Code Suggestion Panel (Left):**
 - Convert static call to AnsiConsole to Spectre.Console.AnsiConsole
 - Convert to interpolated string
 - Convert to 'Program.Main' style program
 - Suppress or configure issues
- Warning Panel (Right):**
 - Spectre1000:** Use AnsiConsole instead of System.Console
 - ... (ellipsis)
 - Console.WriteLine("");** (line to be removed)
 - + AnsiConsole.WriteLine("");** (line to be added)
 - 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]
  └─ PublicKeyword [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]
    └─ PublicKeyword [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`


Visual Studio Installer

Modifying — Visual Studio Enterprise 2022 Preview — 17.13.0 Preview 3.0

- 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

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](#)

All languages

All platforms

All project types



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();
    }
}

```



```
[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;
```

```
            }
```

```
        }
```

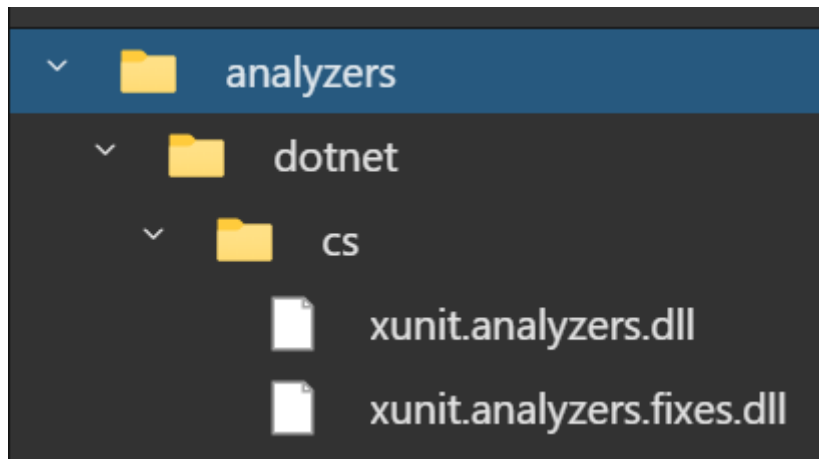
```
        """;
```

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

```
}
```

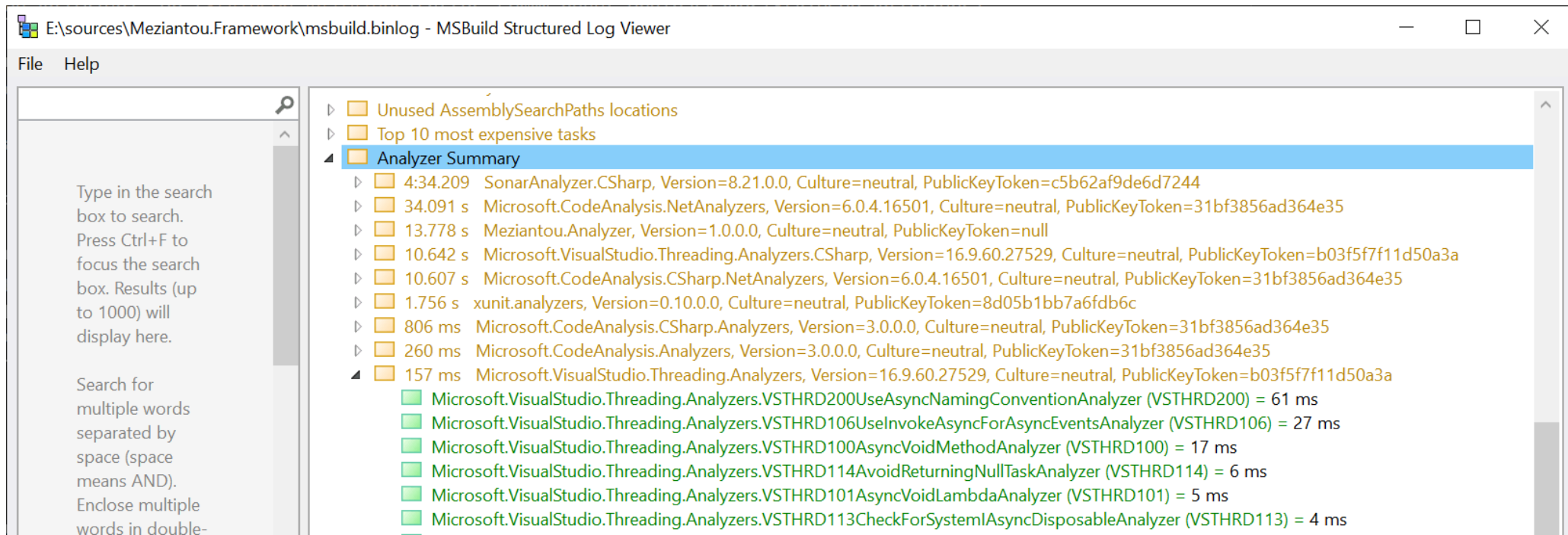
Package a Roslyn analyzer

- 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`



The screenshot shows the MSBuild Structured Log Viewer window. The main pane displays a tree view of performance analysis results. The 'Analyzer Summary' section is expanded, showing a list of analyzers and their execution times. The 'Unused AssemblySearchPaths locations' and 'Top 10 most expensive tasks' sections are also visible.

Time	Analyzer Name	Duration
4:34.209	SonarAnalyzer.CSharp, Version=8.21.0.0, Culture=neutral, PublicKeyToken=c5b62af9de6d7244	
34.091 s	Microsoft.CodeAnalysis.NetAnalyzers, Version=6.0.4.16501, Culture=neutral, PublicKeyToken=31bf3856ad364e35	
13.778 s	Meziantou.Analyzer, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null	
10.642 s	Microsoft.VisualStudio.Threading.Analyzers.CSharp, Version=16.9.60.27529, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a	
10.607 s	Microsoft.CodeAnalysis.CSharp.NetAnalyzers, Version=6.0.4.16501, Culture=neutral, PublicKeyToken=31bf3856ad364e35	
1.756 s	xunit.analyzers, Version=0.10.0.0, Culture=neutral, PublicKeyToken=8d05b1bb7a6fdb6c	
806 ms	Microsoft.CodeAnalysis.CSharp.Analyzers, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35	
260 ms	Microsoft.CodeAnalysis.Analyzers, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35	
157 ms	Microsoft.VisualStudio.Threading.Analyzers, Version=16.9.60.27529, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a	
	Microsoft.VisualStudio.Threading.Analyzers.VSTHRD200UseAsyncNamingConventionAnalyzer (VSTHRD200)	61 ms
	Microsoft.VisualStudio.Threading.Analyzers.VSTHRD106UseInvokeAsyncForAsyncEventsAnalyzer (VSTHRD106)	27 ms
	Microsoft.VisualStudio.Threading.Analyzers.VSTHRD100AsyncVoidMethodAnalyzer (VSTHRD100)	17 ms
	Microsoft.VisualStudio.Threading.Analyzers.VSTHRD114AvoidReturningNullTaskAnalyzer (VSTHRD114)	6 ms
	Microsoft.VisualStudio.Threading.Analyzers.VSTHRD101AsyncVoidLambdaAnalyzer (VSTHRD101)	5 ms
	Microsoft.VisualStudio.Threading.Analyzers.VSTHRD113CheckForSystemIAsyncDisposableAnalyzer (VSTHRD113)	4 ms

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