

# Jak usprawnić proces wytwarzania oprogramowania przy pomocy Roslyn.

Cezary Piątek



**Cezary Piątek** @cezary\_piątek · 2 godz.

The article "The reasons behind why I don't use #AutoMapper." was a great success with over 14k view. Maybe I should write the second part "The reasons behind why I don't use #MediatR" :)

Przetłumacz tweeta



**The reasons behind why I don't use AutoMapper.**

My list of AutoMapper disadvantages which you should consider before using it in your project.

[cezarypiatek.github.io](http://cezarypiatek.github.io)

4



3



**Jimmy Bogard** 🍷

@jbogard

Obserwuj

W odpowiedzi do @cezary\_piątek

**no one should use AutoMapper!**

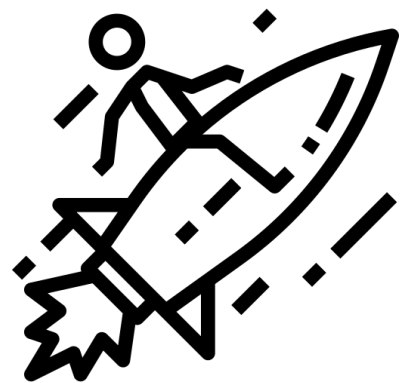
Przetłumacz tweeta

05:58 - 26 lut 2019

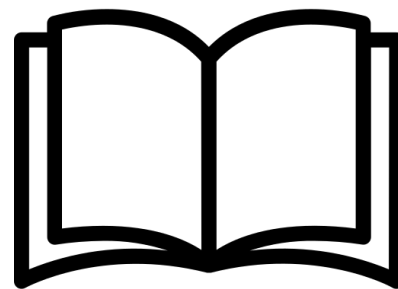


# Krótką historia Roslyn

2005



2011



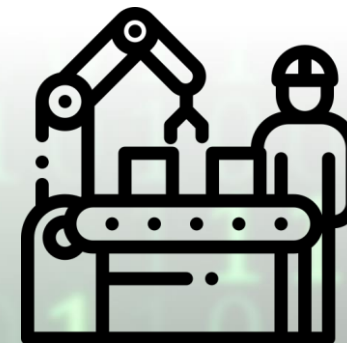
2015



2009



2014







Nowości w  
VisualStudio

---

# C# Interactive

```
C# Interactive (64-bit)
⏪ ⏩ ⏴ ⏵
Microsoft (R) Roslyn C# Compiler version 2.9.0.63208
Loading context from 'CSharpInteractive.rsp'.
Type "#help" for more information.
> #help
Keyboard shortcuts:
  Enter           If the current submission appears to be complete, evaluate it. Otherwise, insert a new line.
  Ctrl-Enter      Within the current submission, evaluate the current submission.
                  Within a previous submission, append the previous submission to the current submission.
  Shift-Enter     Insert a new line.
  Escape          Clear the current submission.
  Alt-UpArrow     Replace the current submission with a previous submission.
  Alt-DownArrow   Replace the current submission with a subsequent submission (after having previously navigated backwards).
  Ctrl-Alt-UpArrow Replace the current submission with a previous submission beginning with the same text.
  Ctrl-Alt-DownArrow Replace the current submission with a subsequent submission beginning with the same text (after having previously navigated backwards).
  Ctrl-K, Ctrl-Enter Paste the selection at the end of interactive buffer, leave caret at the end of input.
  Ctrl-E, Ctrl-Enter Paste and execute the selection before any pending input in the interactive buffer.
  Ctrl-A          First press, select the submission containing the cursor. Second press, select all text in the window.
REPL commands:
  #cls, #clear    Clears the contents of the editor window, leaving history and execution context intact.
  #help          Display help on specified command, or all available commands and key bindings if none specified.
  #reset         Reset the execution environment to the initial state, keep history.
Script directives:
  #r             Add a metadata reference to specified assembly and all its dependencies, e.g. #r "myLib.dll".
  #load         Load specified script file and execute it, e.g. #load "myScript.csx".
>
```

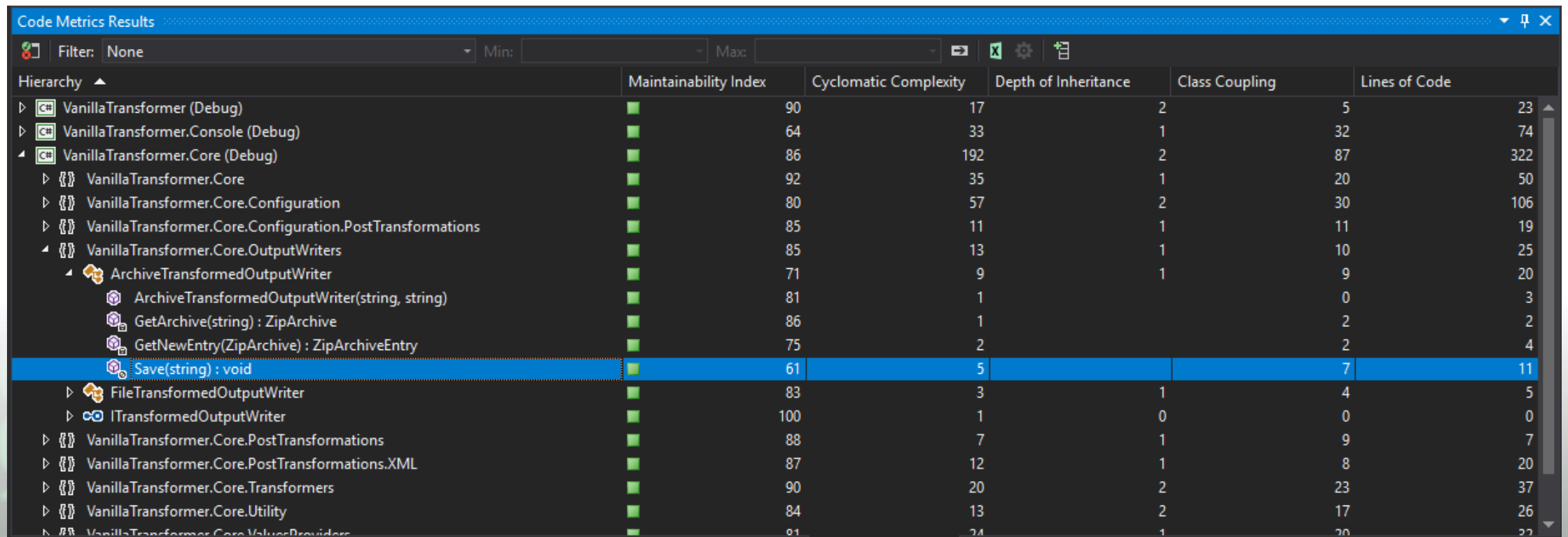
# Live Unit Testing

The screenshot displays the Visual Studio IDE. On the left, the Test Explorer shows a test suite for 'VanillaTransformer' with 32 tests, all of which are passing (indicated by green checkmarks). The test suite is expanded to show 'VanillaTransformer.Tests (32)' with a total duration of 236 ms. Underneath, 'VanillaTransformer.Tests.CoreTest (10)' is expanded to show 'TransformConfigurationReaderTests (10)' with a duration of 191 ms. The 'TransformConfigurationReaderTests' suite is further expanded to show 10 individual tests, all passing, with durations ranging from 1 ms to 178 ms. The main editor window shows the source code for 'TransformConfigurationReaderTests.cs' in the 'VanillaTransformer.Core' project. The code is a C# unit test for the 'CreateValuesProvider' method. The test method is annotated with [TestMethod] and [DataRow] attributes. The code includes a 'try-catch' block to handle 'InvalidOperationException' and a 'throw' statement for 'InvalidConfigurationException'. The test method is annotated with [ExpectedException(typeof(InvalidConfigurationException))]. The code also includes a 'private static' method 'CreateValuesProvider' that returns an 'IValuesProvider' object based on the 'ValuesSourceElementName' attribute of the 'y' parameter. The code is annotated with [ExpectedException(typeof(InvalidConfigurationException))]. The code is annotated with [ExpectedException(typeof(InvalidConfigurationException))].

```
129 return postTransformationsNode.Elements()
130     .Select(PostTransformationsConfigurationOperationFactory.Create)
131     .ToList();
132 }
133 catch (InvalidOperationException)
134 {
135     throw InvalidConfigurationException.BecauseDuplicatedPostTransformations(path);
136 }
137 }
138 }
139 private static IValuesProvider CreateValuesProvider(XElement y, string rootPath)
140 {
141     if (y.Attribute(ValuesSourceElementName) != null)
142     {
143         return new XmlFileConfigurationValuesProvider(UpdatePathWithRootPath(y.Attribute(ValuesSourceElementName).Value, rootPath));
144     }
145     if (y.Element(ValuesSourceElementName) != null)
146     {
147         return new XmlInlineConfigurationValuesProvider(y.Element(ValuesSourceElementName));
148     }
149     return null;
150 }
151 }
152 }
153 }
```



# Code Metrics Results



The screenshot shows a 'Code Metrics Results' window with a table of metrics. The table has columns for Hierarchy, Maintainability Index, Cyclomatic Complexity, Depth of Inheritance, Class Coupling, and Lines of Code. The 'Save(string) : void' method is highlighted in blue.

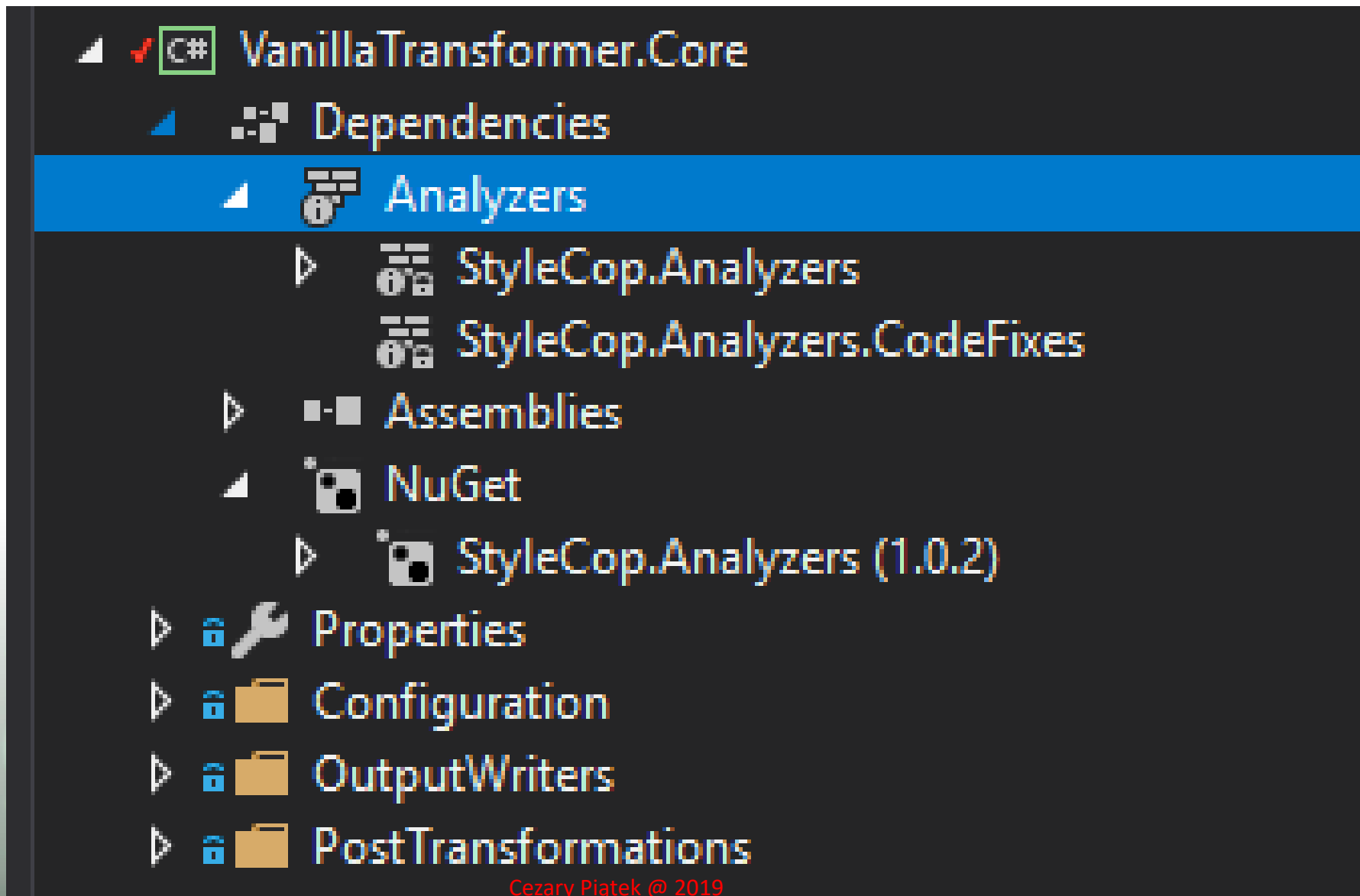
Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
VanillaTransformer (Debug)	90	17	2	5	23
VanillaTransformer.Console (Debug)	64	33	1	32	74
VanillaTransformer.Core (Debug)	86	192	2	87	322
VanillaTransformer.Core	92	35	1	20	50
VanillaTransformer.Core.Configuration	80	57	2	30	106
VanillaTransformer.Core.Configuration.PostTransformations	85	11	1	11	19
VanillaTransformer.Core.OutputWriters	85	13	1	10	25
ArchiveTransformedOutputWriter	71	9	1	9	20
ArchiveTransformedOutputWriter(string, string)	81	1		0	3
GetArchive(string) : ZipArchive	86	1		2	2
GetNewEntry(ZipArchive) : ZipArchiveEntry	75	2		2	4
Save(string) : void	61	5		7	11
FileTransformedOutputWriter	83	3	1	4	5
ITransformedOutputWriter	100	1	0	0	0
VanillaTransformer.Core.PostTransformations	88	7	1	9	7
VanillaTransformer.Core.PostTransformations.XML	87	12	1	8	20
VanillaTransformer.Core.Transformers	90	20	2	23	37
VanillaTransformer.Core.Utility	84	13	2	17	26
VanillaTransformer.Core.ValueProviders	81	24	1	20	22

A car is completely covered in a dark red, textured cloth. The car is viewed from the rear, and the cloth is draped over its roof, trunk, and rear fenders. The word "ANALIZATORY" is printed in large, white, sans-serif capital letters across the center of the rear of the car. The background is a dark, gradient grey.

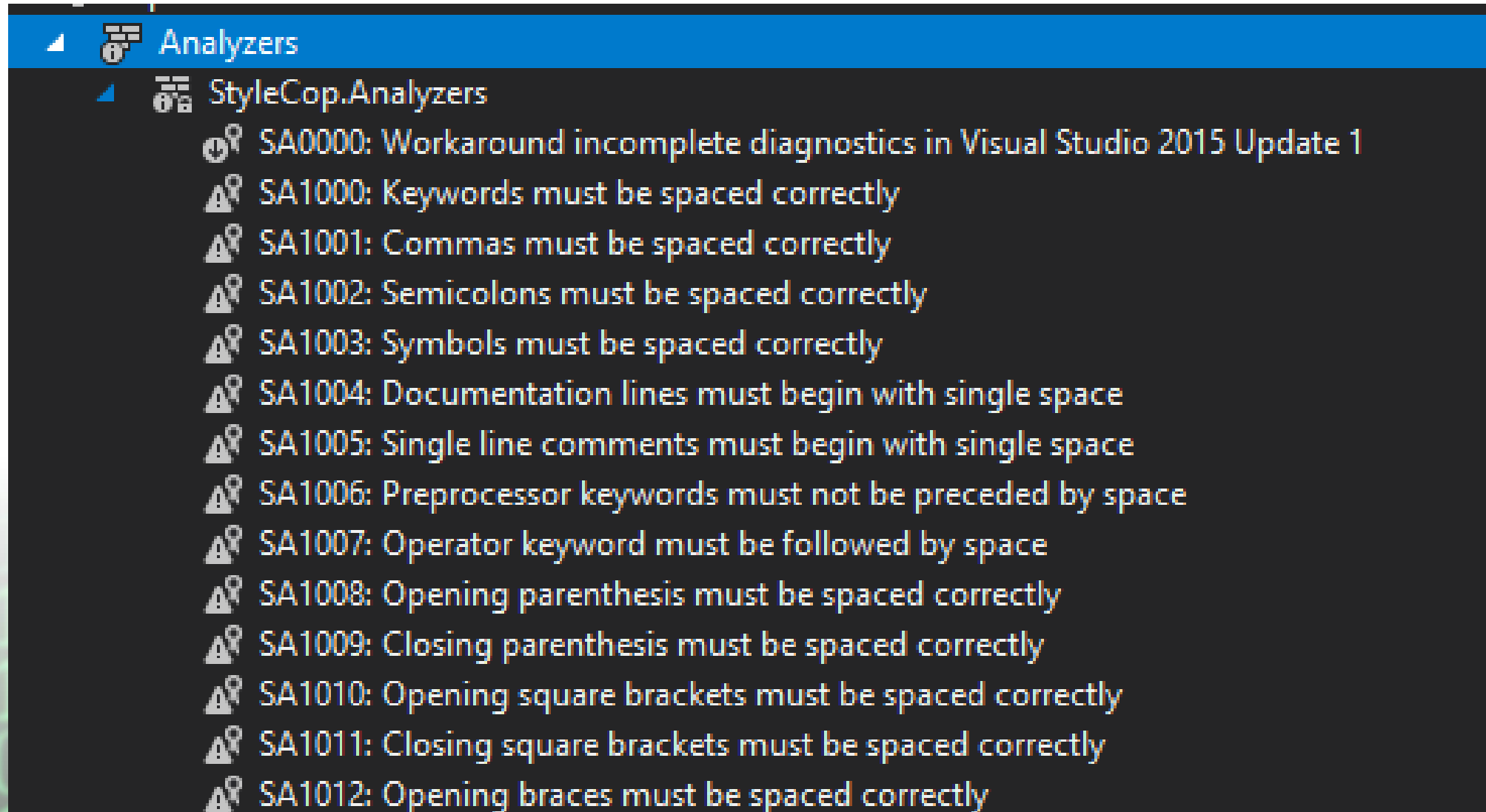
# ANALIZATORY



# Analizatory



# Analizatory



# Analizatory

```
13  
14 public void Save(string result)  
15 {  
    var fileInfo = new FileInfo(filePath);  
    Directory.CreateDirectory(filePath);  
    File.WriteAllText(filePath, result);  
}
```

Use explicit type instead of 'var'  
Prefix reference with 'this.'  
Suppress SA1101

```
public void Save(string result)  
{  
    var fileInfo = new FileInfo(filePath);  
    fileInfo.Directory.CreateDirectory(filePath);  
    File.WriteAllText(filePath, result);  
}
```

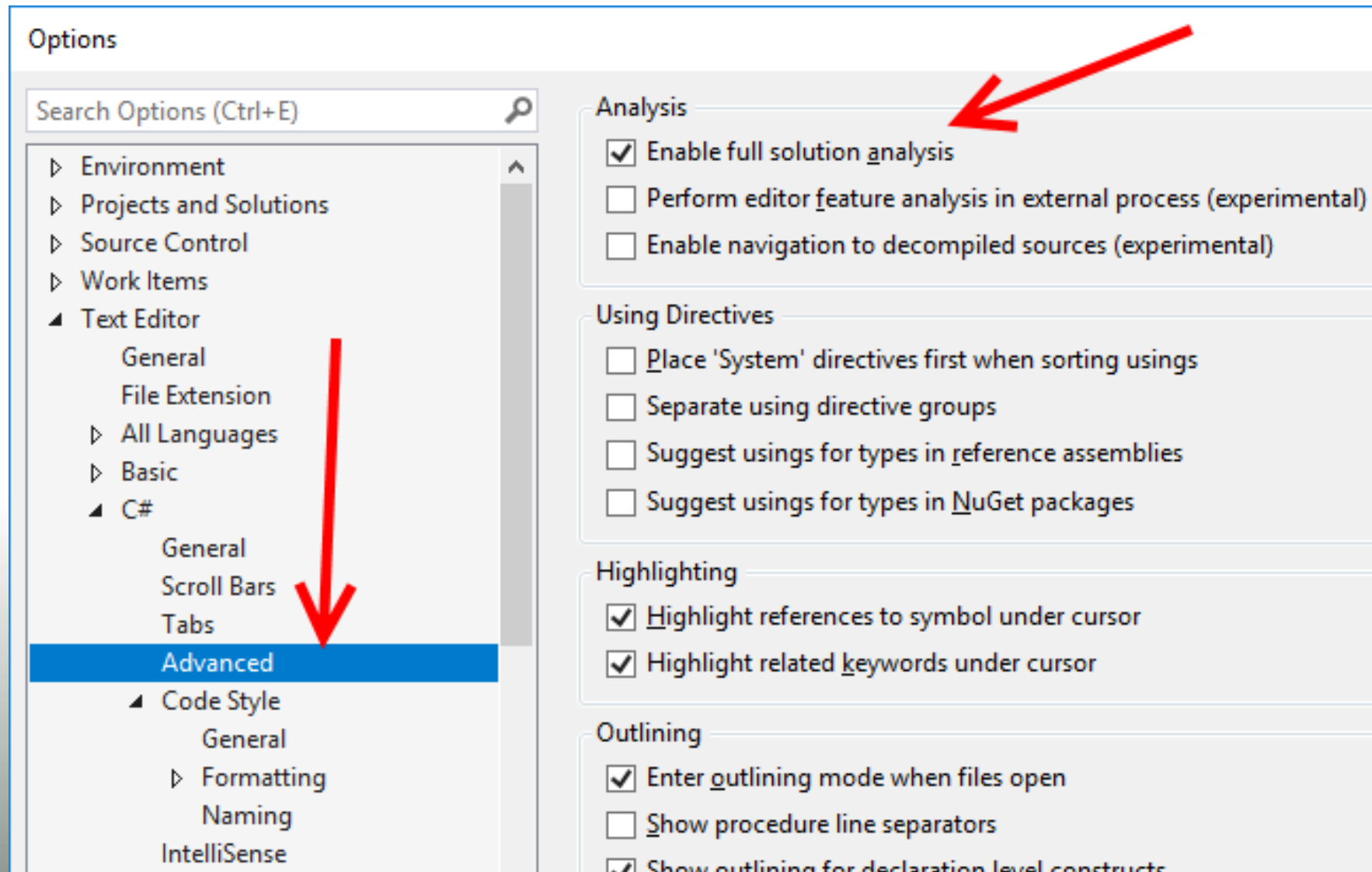
Prefix reference with 'this.'

SA1101 Prefix local calls with this

```
...  
{  
    var fileInfo = new FileInfo(filePath);  
    var fileInfo = new FileInfo(this.filePath);  
    fileInfo.Directory.CreateDirectory(filePath);  
    ...  
}
```

Preview changes  
Fix all occurrences in: Document | Project | Solution

# Analizatory





# Analizatory

Error List

Entire Solution 60 Errors 0 of 227 Warnings 0 of 6 Messages Build + IntelliSense

C.	Description	Project	File	Line	Suppression St...
SA1101	Prefix local calls with this	VanillaTransformer.Core	PostTransformationsConfigurationAddOperation...	17	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	PostTransformationsConfigurationRemoveOpera...	17	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfiguration.cs	20	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfiguration.cs	26	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfiguration.cs	28	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfigurationReader.cs	37	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfigurationReader.cs	39	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfigurationReader.cs	41	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfigurationReader.cs	43	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfigurationReader.cs	49	Active
SA1101	Prefix local calls with this	VanillaTransformer.Core	TransformConfigurationReader.cs	57	Active

# Analizatory

Application  
Build  
Build Events  
Debug  
Resources  
Services  
Settings  
Reference Paths  
Signing  
**Code Analysis**

Configuration: Active (Debug) Platform: Active (Any CPU)

Enable Code Analysis on Build  
 Suppress results from generated code (managed only)

Rule Set

Run this rule set:  
Rules for ClassLibrary1

Description:  
Code analysis rules for ClassLibrary1.csproj.

Path:  
C:\Users\cepi\source\repos\ClassLibrary1\ClassLibrary1\ClassLibrary1.ruleset

Open [Learn more about rule sets](#)

# Analizatory

Group by: Analyzer ID | Project: VanillaTransformer.Core | Search

ID	Name	Action
<input type="checkbox"/> Managed Binary Analysis		Multiple
<input checked="" type="checkbox"/> Microsoft.CodeAnalysis.CSharp		Multiple
<input checked="" type="checkbox"/> Microsoft.CodeAnalysis.CSharp.F...		Multiple
<input checked="" type="checkbox"/> Microsoft.CodeAnalysis.Features		Hidden
<input checked="" type="checkbox"/> StyleCop.Analyzers		Warning
<input checked="" type="checkbox"/> SA0000	Workaround incomplete diagnostics in Visual Studio 2015 Update 1	Warning
<input checked="" type="checkbox"/> SA1000	Keywords must be spaced correctly	Warning
<input checked="" type="checkbox"/> SA1001	Commas must be spaced correctly	Error
<input checked="" type="checkbox"/> SA1002	Semicolons must be spaced correctly	Info
<input checked="" type="checkbox"/> SA1003	Symbols must be spaced correctly	Hidden
<input checked="" type="checkbox"/> SA1004	Documentation lines must begin with single space	None
<input checked="" type="checkbox"/> SA1005	Single line comments must begin with single space	<Inherit>
<input checked="" type="checkbox"/> SA1006	Preprocessor keywords must not be preceded by space	<Inherit>

**Description:**

Visual Studio 2015 Update 1 contains a bug which can cause diagnostics to occasionally not display in the Errors window. When this occurs, it is impossible to use the code fixes to address style violations reported during a build. This analyzer works around the bug (dotnet/roslyn#7446).

When this analyzer is enabled, all diagnostics will eventually be reported in the Error window, but the performance of the analyzers is reduced. The rule is disabled for maximum performance, but can be enabled if users notice errors appearing during a build but not while editing, and they wish to use the code fixes to correct them.

Note that some situations are not affected by the bug:

- \* When building a project, all relevant warnings are reported even if this rule is disabled.
- \* The various Fix All operations work properly for the selected scope, even if only a subset of the violations are appearing in the Errors window.

# Analizatory



## .NET Analyzers

Report abuse

An organization for the development of analyzers (diagnostics, code fixes, and refactorings) using the .NET Compiler Platform

Repositories 17

People 9

Projects 0

### Pinned repositories

#### StyleCopAnalyzers

An implementation of StyleCop rules using the .NET Compiler Platform

C# ★ 1.1k 🍴 238

#### AsyncUsageAnalyzers

Analyzers for asynchronous .NET code

C# ★ 92 🍴 17

#### WpfAnalyzers

C# ★ 48 🍴 8



# Analizatory



- A collection of 500+ analyzers, refactorings and fixes for C#, powered by [Roslyn](#).



## Refactoring Essentials

The premier free Visual Studio extension for C# and VB.NET refactorings, including code best practice analyzers/fixes to improve your projects.

Install in 2015 »

Install in 2017 »

Star 576

Fork 118

Tweet



## Code Cracker

Holding the Code Cracker Project.

Brazil

Repositories 7

People 5

Projects 0

Cezary Piątek @ 2019

# Analizatory

## Roslyn Clr Heap Allocation Analyzer

---

[gitter](#) [join chat](#)

# Toolkit

Asyncifier & AsyncFixer

## AsyncFixer



SemihOkur | 11 062 installs | ★★★★★ (13)

Advanced Async/Await Diagnostics and CodeFixes for C#.

Download

We analyzed:

**1378** Windows Phone Apps

comprising

**12M** Source Lines of Code

produced by

**3376** Developers

<http://www.learnasync.net/>

Cezary Piątek @ 2019



# Security Code Scan - static code analyzer for .NET

Download:

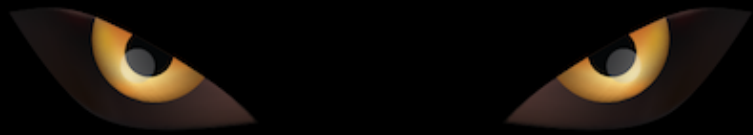
[NuGet package](#)

[Visual Studio extension](#)

- SCS0001: Command injection possible in {1} argument passed to '{0}'
- SCS0002: LINQ API: SQL injection possible in {1} argument passed to '{0}'
- SCS0003: XPath injection possible in {1} argument passed to '{0}'
- SCS0004: Certificate Validation has been disabled
- SCS0005: Weak random generator
- SCS0006: Weak hashing function
- SCS0007: XML parsing vulnerable to XXE
- SCS0008: The cookie is missing security flag Secure
- SCS0009: The cookie is missing security flag HttpOnly
- SCS0010: Weak cipher algorithm
- SCS0011: CBC mode is weak
- SCS0012: ECB mode is weak
- SCS0013: Weak cipher mode
- SCS0014: Possible SQL injection in {1} argument passed to '{0}'
- SCS0015: Hardcoded password
- SCS0016: Controller method is vulnerable to CSRF
- SCS0017: Request validation disabled in base class

- SCS0018: Path traversal: injection possible in {1} argument passed to '{0}'
- SCS0019: OutputCache annotation is disabling authorization checks
- SCS0020: OleDb API: SQL injection possible in {1} argument passed to '{0}'
- SCS0021: Request validation has been disabled in {0}({1}): {2}
- SCS0022: Event validation is disabled in {0}({1}): {2}
- SCS0023: View state is not encrypted in {0}({1}): {2}
- SCS0024: View state mac is disabled in {0}({1}): {2}
- SCS0025: Odbc API: SQL injection possible in {1} argument passed to '{0}'
- SCS0026: MsSQL Data Provider: SQL injection possible in {1} argument passed to '{0}'
- SCS0027: Open redirect: possibly unvalidated input in {1} argument passed to '{0}'
- SCS0028: Type information used to serialize and deserialize objects
- SCS0029: Potential XSS vulnerability
- SCS0030: Request validation is enabled only for pages, not for all HTTP requests. {0}({1}): {2}
- SCS0032: The RequiredLength property of PasswordValidator should be set to at least {0}
- SCS0033: Less than {0} properties set in PasswordValidator declaration
- SCS0034: The {0} property must be set





# PUMASCAN

- ⊗ SEC0001: Debug Build Enabled
- ⊗ SEC0002: Custom Errors Disabled
- ⚠ SEC0003: Forms Authentication Secure Cookie Disabled
- ⚠ SEC0004: Forms Authentication Cookieless Session Enabled
- ⚠ SEC0005: Forms Authentication CrossAppRedirects Enabled
- ⚠ SEC0006: Forms Authentication Weak Cookie Protection
- ⚠ SEC0007: Forms Authentication Weak Timeout
- ⚠ SEC0008: HTTP Header Checking Disabled
- ⚠ SEC0009: Version HTTP Response Header Enabled
- ⚠ SEC0010: Pages EventValidation Disabled
- ⚠ SEC0011: Pages ViewStateMac Disabled
- ⚠ SEC0012: Pages ValidateRequest Disabled
- ⚠ SEC0013: Pages ViewStateEncryptionMode Disabled
- ⚠ SEC0014: Insecure HTTP Cookie Transport
- ⊗ SEC0015: Cookie Accessible via Script
- ⚠ SEC0016: Cleartext Machine Key
- ⚠ SEC0017: Weak Password Complexity
- ⚠ SEC0018: Identity Password Lockout Disabled
- ⚠ SEC0019: Missing AntiForgeryToken Attribute
- ⚠ SEC0020: Weak Session Timeout
- ⚠ SEC0021: Session State Server Mode
- ⚠ SEC0022: Model Request Validation Disabled
- ⚠ SEC0023: Action Request Validation Disabled
- ⚠ SEC0024: Unencoded Response Write
- ⚠ SEC0025: Weak Cryptography Algorithm (DES)

- ⚠ SEC0026: Insecure Cipher Mode - Electronic Codebook (ECB)
- ⚠ SEC0027: Weak Cryptography Algorithm (MD5)
- ⚠ SEC0028: Weak Cryptography Algorithm (SHA1)
- ⚠ SEC0029: Insecure Deserialization - BinaryFormatter
- ⚠ SEC0030: Insecure Deserialization - Newtonsoft JSON
- ⚠ SEC0031: Command Injection Process Start
- ⚠ SEC0032: Command Injection Process Start Info
- ⚠ SEC0100: Raw Inline Expression
- ⚠ SEC0101: Raw Binding Expression
- ⚠ SEC0102: Raw Razor Method
- ⚠ SEC0103: Raw WriteLiteral Method
- ⚠ SEC0104: Unencoded Literal Text
- ⚠ SEC0105: Unencoded Label Text
- ⚠ SEC0106: SQL Injection Dynamic LINQ Query
- ⚠ SEC0107: SQL Injection ADO .NET
- ⚠ SEC0108: SQL Injection Dynamic EF Query
- ⚠ SEC0109: Unvalidated MVC Redirect
- ⚠ SEC0110: Unvalidated WebForms Redirect
- ⚠ SEC0111: Path Tampering File Path Result
- ⚠ SEC0112: Path Tampering Unvalidated File Path
- ⚠ SEC0113: Certificate Validation Disabled
- ⚠ SEC0114: LDAP Injection Directory Entry
- ⚠ SEC0115: Insecure Random Number Generator
- ⚠ SEC0116: Path Tampering Unvalidated File Path
- ⚠ SEC0117: LDAP Injection Path Assignment
- ⚠ SEC0118: LDAP Injection Directory Searcher
- ⚠ SEC0119: LDAP Injection Filter Assignment

# Analizatory

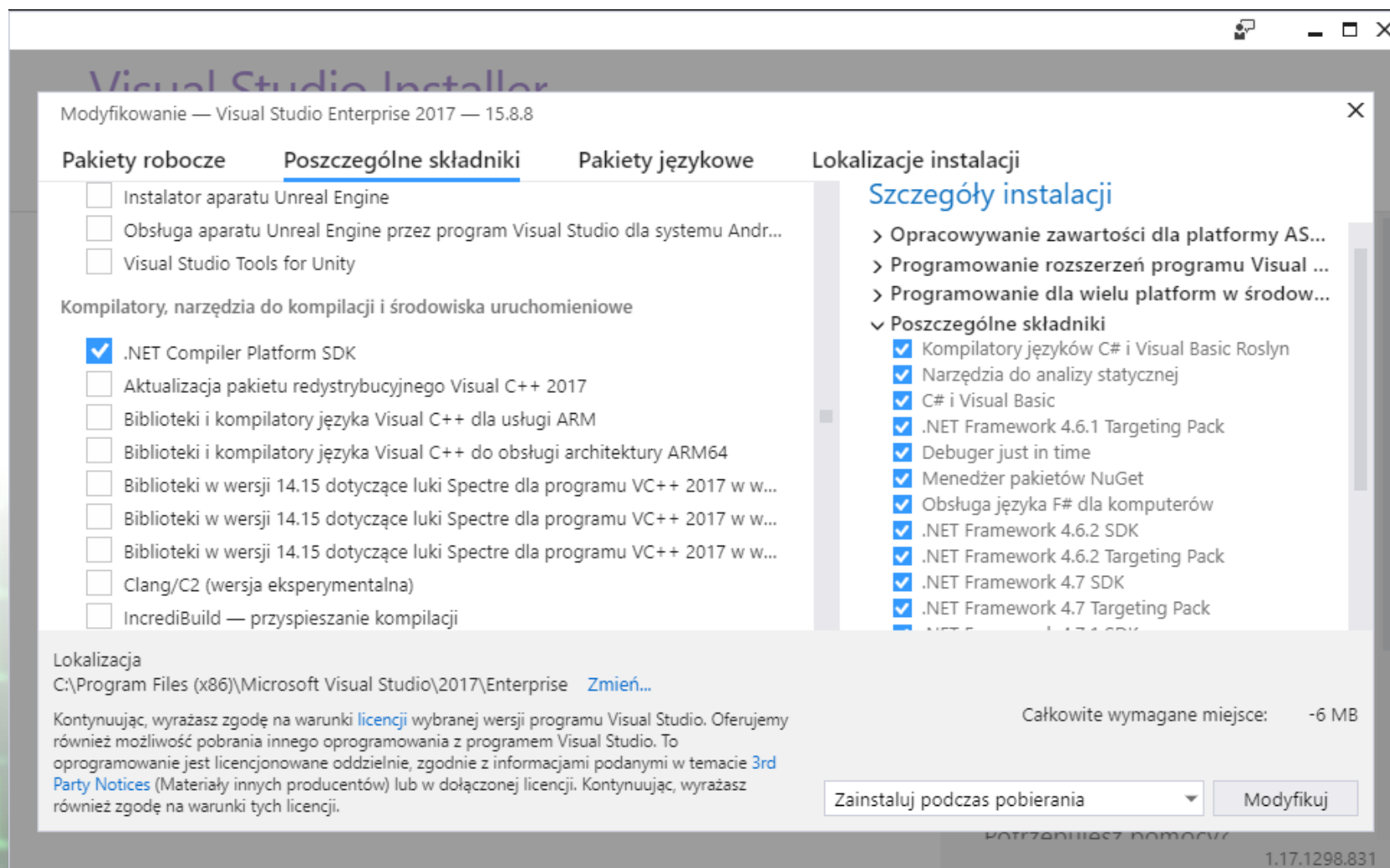
## AnyConstraint.Analyzer

---

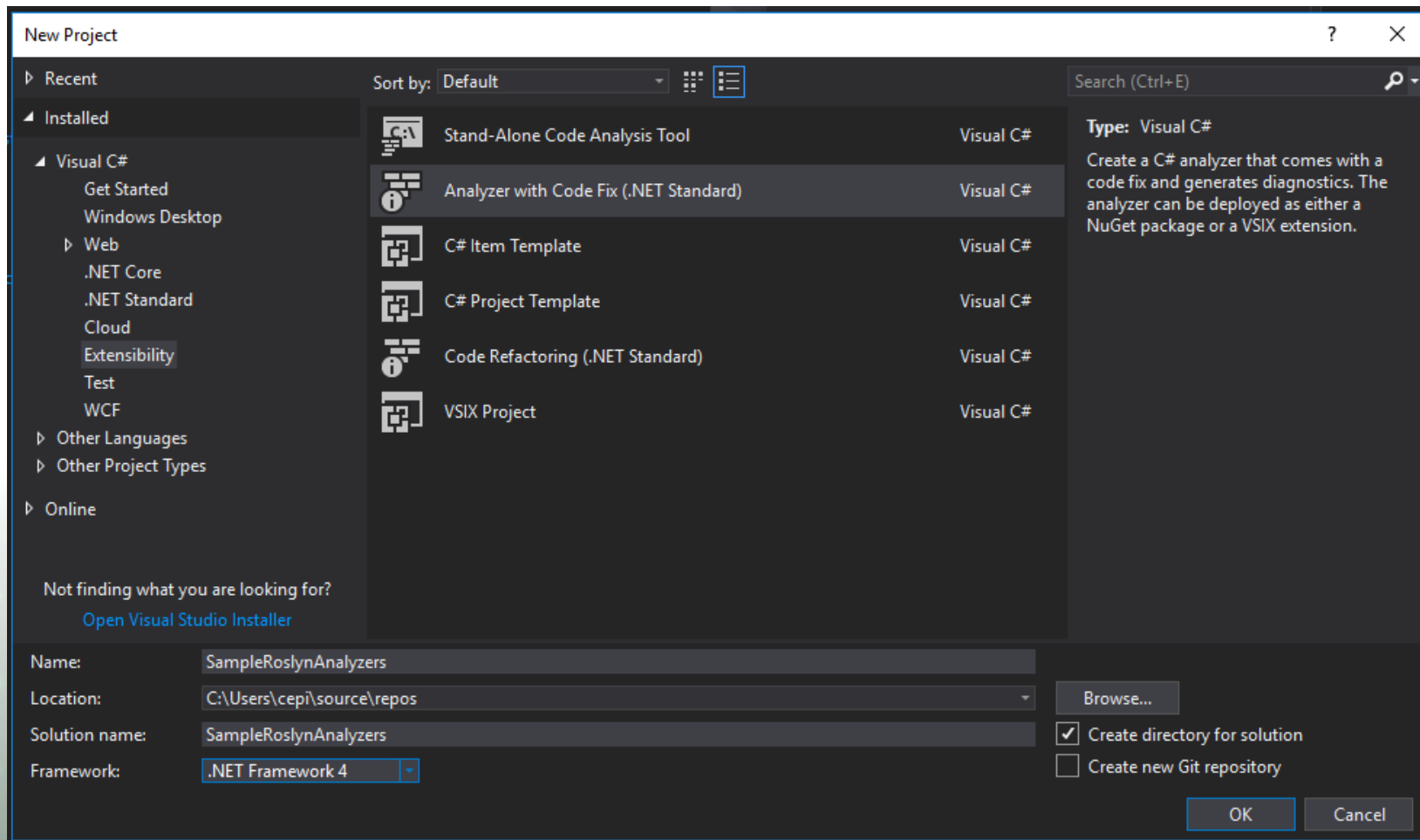
A simple C# analyzer that suppresses the CS0702: "Constraint cannot be special class 'Delegate' / 'Enum' / '...' " error.

```
--public static void DoSth<T>(T aa) where T:Enum  
--{  
--    throw new NotImplementedException();  
--}
```

# Własne rozszerzenie Roslyn

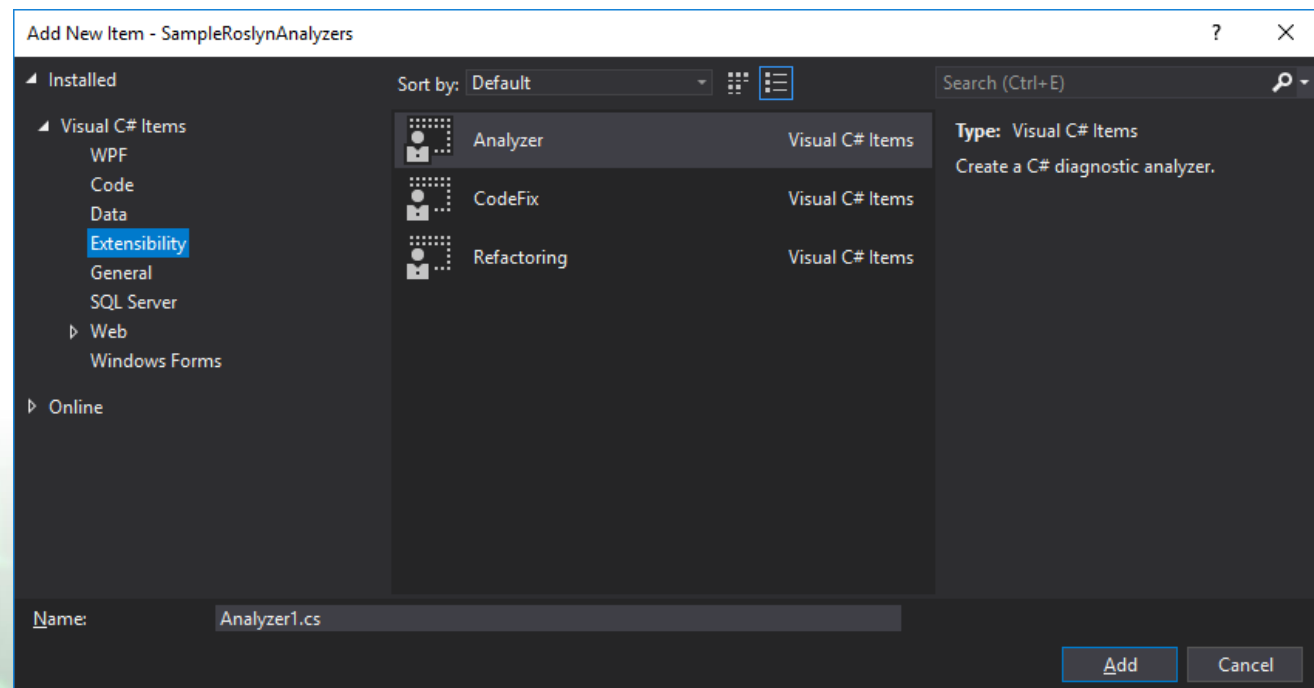
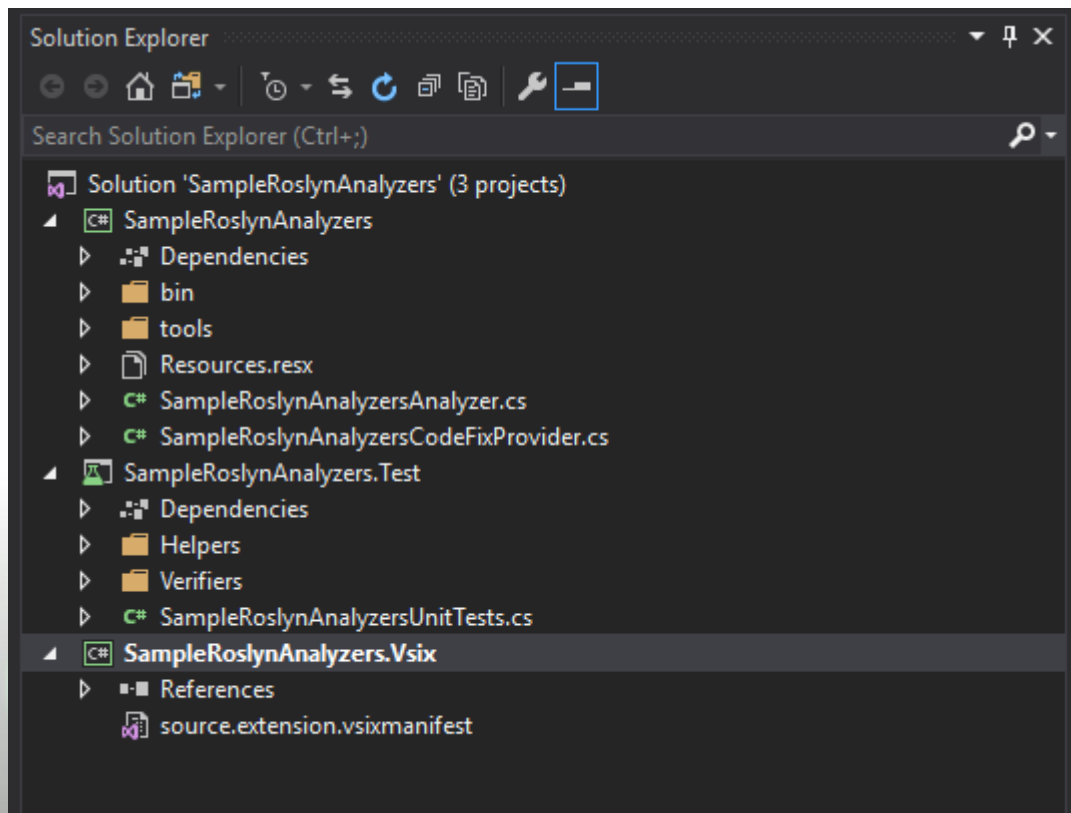


# Własne rozszerzenie Roslyn





# Własne rozszerzenie Roslyn



# Własny Diagnostic Analyzer

```
[DiagnosticAnalyzer(LanguageNames.CSharp)]
2 references
public class SampleRoslynAnalyzersAnalyzer : DiagnosticAnalyzer
{
    const

    private static DiagnosticDescriptor Rule = new DiagnosticDescriptor(DiagnosticId, Title, MessageFormat, Category, DiagnosticSeverity.Warning, isEnabledByDefault: true, description: Description);












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

    0 references
    public override void Initialize(AnalysisContext context)
    {
        context.RegisterSymbolAction(AnalyzeSymbol, SymbolKind.NamedType);
    }

    /// <summary>
    /// A symbol action is invoked once per complete semantic processing of a declaration of a type or type member,
    /// provided that symbol has a kind matching one of the kinds supplied when the action was registered.
    /// </summary>
    1 reference
    private static void AnalyzeSymbol(SymbolAnalysisContext context)
    {
        var namedTypeSymbol = (INamedTypeSymbol)context.Symbol;

        if (namedTypeSymbol.Name.ToCharArray().Any(char.IsLower))
        {
            var diagnostic = Diagnostic.Create(Rule, namedTypeSymbol.Locations[0], namedTypeSymbol.Name);
            context.ReportDiagnostic(diagnostic);
        }
    }
}
```

# Typy analizatorów

 RegisterCodeBlockAction	void
 RegisterCodeBlockStartAction	void
 RegisterCompilationAction	void
 RegisterCompilationStartAction	void
 RegisterOperationAction	void
 RegisterOperationBlockAction	void
 RegisterOperationBlockStartAction	void
 RegisterSemanticModelAction	void
 RegisterSymbolAction	void
 RegisterSyntaxNodeAction	void
 RegisterSyntaxTreeAction	void

# Własny CodeFix

```
[ExportCodeFixProvider(LanguageNames.CSharp, Name = nameof(ExplicitConversionCodeFixProvider)), Shared]
```

```
6 references | Cezary Piątek, 36 days ago | 2 authors, 11 changes
```

```
public class ExplicitConversionCodeFixProvider : CodeFixProvider
```

```
{
```

```
    2 references | Cezary Piątek, 120 days ago | 1 author, 3 changes
```

```
    public sealed override ImmutableArray<string> FixableDiagnosticIds
```

```
    => ImmutableArray.Create(CS0029, CS0266);
```

```
    2 references | Cezary Piątek, 246 days ago | 1 author, 4 changes
```

```
    public sealed override async Task RegisterCodeFixesAsync(CodeFixContext context)
```

```
    {
```

```
        var root = await context.Document.GetSyntaxRootAsync(context.CancellationToken).ConfigureAwait(false);
```

```
        var diagnostic = context.Diagnostics.First();
```

```
        var token = root.FindToken(diagnostic.Location.SourceSpan.Start);
```

```
        var statement = FindStatementToReplace(token.Parent);
```

```
    }
```

```
        switch (statement)
```

```
        {
```

```
            case AssignmentExpressionSyntax assignmentExpression:
```

```
                context.RegisterCodeFix(CodeAction.Create(title: title, createChangedDocument: c => GenerateExplicitConversion(context.Document, assignmentExpression, c), equivalenceKey: title), diagnostic);
```

```
                break;
```

```
            case ReturnStatementSyntax returnStatement:
```

```
                context.RegisterCodeFix(CodeAction.Create(title: title, createChangedDocument: c => GenerateExplicitConversion(context.Document, returnStatement, c), equivalenceKey: title), diagnostic);
```

```
                break;
```

```
            case YieldStatementSyntax yieldStatement:
```

```
                context.RegisterCodeFix(CodeAction.Create(title: title, createChangedDocument: c => GenerateExplicitConversion(context.Document, yieldStatement, c), equivalenceKey: title), diagnostic);
```

```
                break;
```

```
        }
```

```
    }
```

```
    1 reference | Cezary Piątek, 36 days ago | 2 authors, 7 changes
```

```
    private async Task<Document> GenerateExplicitConversion(Document document, AssignmentExpressionSyntax assignmentExpression, CancellationToken cancellationToken)...
```

# Własny Code Refactoring

```
[ExportCodeRefactoringProvider(LanguageNames.CSharp, Name = nameof(ImplementCloneMethodRefactoring)), Shared]
2 references | Cezary Piatek, 47 days ago | 1 author, 4 changes
public class ImplementCloneMethodRefactoring : CodeRefactoringProvider
{
    public const string Title = "Implement clone method";

    2 references | Cezary Piatek, 47 days ago | 1 author, 2 changes
    public sealed override async Task ComputeRefactoringsAsync(CodeRefactoringContext context)
    {
        var root = await context.Document.GetSyntaxRootAsync(context.CancellationToken).ConfigureAwait(false);
        var node = root.FindNode(context.Span);
        if (node is ClassDeclarationSyntax || node is StructDeclarationSyntax)
        {
            var typeDeclarationSyntax = node as TypeDeclarationSyntax;
            if (typeDeclarationSyntax.Modifiers.Any(SyntaxKind.StaticKeyword))
            {
                return;
            }
            context.RegisterRefactoring(CodeAction.Create(title: Title, createChangedDocument: c => AddCloneImplementation(context.Document, typeDeclarationSyntax, c), equivalenceKey: Title));
        }

        if (node is MethodDeclarationSyntax md && IsCandidateForCloneMethod(md))
        {
            context.RegisterRefactoring(CodeAction.Create(title: Title, createChangedDocument: c => ImplementCloneMethodBody(context.Document, md, c), equivalenceKey: Title));
        }
    }

    1 reference | Cezary Piatek, 47 days ago | 1 author, 1 change
    private async Task<Document> ImplementCloneMethodBody(Document document, MethodDeclarationSyntax methodDeclaration, CancellationToken cancellationToken)
    {
        var generator = SyntaxGenerator.GetGenerator(document);
        var semanticModel = await document.GetSemanticModelAsync(cancellationToken);
        var methodSymbol = semanticModel.GetDeclaredSymbol(methodDeclaration);
        var cloneExpression = CreateCloneExpression(generator, semanticModel, methodSymbol.ReturnType as INamedTypeSymbol);
        return await document.ReplaceNodes(methodDeclaration.Body, ((BaseMethodDeclarationSyntax) generator.MethodDeclaration(methodSymbol, cloneExpression)).Body, cancellationToken);
    }
}
```



# Syntax Visualizer

The screenshot shows the Visual Studio IDE with the Syntax Visualizer tool open. The main editor displays the following C# code:

```
17 public void Save(string result)
18 {
19     var archiveFileInfo = new FileInfo(archivePath);
20     archiveFileInfo.Directory?.Create();
21
22     using (var archive = GetArchive(archivePath))
23     {
24         var archiveFile = GetNewEntry(archive);
25         using (var s = archiveFile.Open())
26         {
27             using (var w = new StreamWriter(s))
28             {
29                 w.Write(result);
30             }
31         }
32     }
33 }
34
35 private ZipArchiveEntry GetNewEntry(ZipArchive archive)
36 {
37     var archiveFile = archive.GetEntry(filePath);
38     archiveFile?.Delete();
39     return archive.CreateEntry(filePath);
40 }
41
42 private static ZipArchive GetArchive(string archivePath)
43 {
44     return ZipFile.Open(archivePath, ZipArchiveMode.Update);
```

The Syntax Visualizer window shows the following Syntax Tree:

- ConstructorDeclaration [284..464]
- MethodDeclaration [476..1024]
  - PublicKeyword [476..482]
  - PredefinedType [483..487]
  - IdentifierToken [488..492]
  - ParameterList [492..507]
  - Block [517..1024]
    - OpenBraceToken [517..518]
    - LocalDeclarationStatement [532..580]
    - ExpressionStatement [594..630]
    - UsingStatement [646..1013]
      - UsingKeyword [646..651]
      - OpenParenToken [652..653]
      - VariableDeclaration [653..690]
      - CloseParenToken [690..691]
      - Block [705..1013]
        - OpenBraceToken [705..706]
        - LocalDeclarationStatement [724..763]
          - VariableDeclaration [724..763]

The Properties window shows the following details for the selected MethodDeclarationSyntax:

Property	Value
Type	MethodDeclarationSyntax
Kind	MethodDeclaration
ContainsDiagnostics	False
ContainsDirectives	False
ContainsSkippedText	False
ExplicitInterfaceSpecifier	
ExpressionBody	
FullSpan	[466..1026]

# Roslyn Quoter

Open-source at <https://github.com/KirillOsenkov/RoslynQuoter>

Fork me on GitHub

```
public class UserDTO
{
    public string FirstName {get; set;}
}
```

Parse as:

- 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;'

```
CompilationUnit()
.WithMembers(
    SingletonList<MemberDeclarationSyntax>(
        ClassDeclaration("UserDTO")
        .WithModifiers(
            TokenList(
                Token(SyntaxKind.PublicKeyword)))
        .WithMembers(
            SingletonList<MemberDeclarationSyntax>(
                PropertyDeclaration(
                    PredefinedType(
                        Token(SyntaxKind.StringKeyword)),
                    Identifier("FirstName"))
                .WithModifiers(
                    TokenList(
                        Token(SyntaxKind.PublicKeyword)))
                .WithAccessorList(
                    AccessorList(
                        List<AccessorDeclarationSyntax>(
```


# Testowanie narzędzi Roslyn

```
0 references | Cezary Piatek, 54 days ago | 1 author, 3 changes
public class ImplementCloneMethodTests : CodeRefactoringTestFixture
{
    [Test]
    0 references | Cezary Piatek, 66 days ago | 1 author, 1 change
    public void should_be_able_to_generate_deep_clone_method()
    {
        var test = ImplementCloneMethodTestCases._001_DeepClone;
        var fixedCode = ImplementCloneMethodTestCases._001_DeepClone_FIXED;
        TestCodeRefactoring(test, fixedCode);
    }

    2 references | Cezary Piatek, 66 days ago | 1 author, 1 change
    protected override string LanguageName => LanguageNames.CSharp;
    2 references | Cezary Piatek, 66 days ago | 1 author, 1 change
    protected override CodeRefactoringProvider CreateProvider()
    {
        return new ImplementCloneMethodRefactoring();
    }
}
```



## Mapping Generator | Reports | Manage

Cezary Piątek |  1 670 clicks | ★★★★★ (4)

"AutoMapper" like, Roslyn based, code fix provider that allows to generate mapping code in design time.

[Get Started](#)



## DDToolbox | Reports | Manage

Cezary Piątek |  53 clicks | ★★★★★ (0)

A set of Roslyn refactorings supporting DDD design

[Get Started](#)



# MappingGenerator

```
7 public static class Mapper
8 {
9     public static UserDTO Map(UserEntity entity)
10    {
11        throw new NotImplementedException();
12    }
13 }
14 }
```

Cezary Piątek @ 2019





# MappingGenerator

```
6 namespace TestAutoMapper
7 {
8     References
9     public class MappingTest
10    {
11        References
12        public void Map(UserEntity source, UserDTO target)
13        {
14            target.FirstName = source.FirstName;
15            target.ImageData = source.ImageData.ToArray();
16            target.LastName = source.LastName;
17            target.LuckyNumbers = source.LuckyNumbers.ToList();
18            target.MainAddress = source.MainAddress;
19        }
20    }
21 }
22 }
23 }
24 }
```

Generate explicit conversion  **CS0029** Cannot implicitly convert type 'TestAutoMapper.AddressEntity' to 'TestAutoMapper.AddressDTO'

```
...
target.LastName = source.LastName;
target.LuckyNumbers = source.LuckyNumbers.ToList();
target.MainAddress = source.MainAddress;
var targetMainAddress = new AddressDTO();
targetMainAddress.BuildingNo = source.MainAddress.BuildingNo;
targetMainAddress.City = source.MainAddress.City;
targetMainAddress.FlatNo = source.MainAddress.FlatNo;
targetMainAddress.Street = source.MainAddress.Street;
targetMainAddress.ZipCode = source.MainAddress.ZipCode;
target.MainAddress = targetMainAddress;
target.UnitId = source.Unit.Id;
...
```

[Preview changes](#)  
Fix all occurrences in: [Document](#) | [Project](#) | [Solution](#)



# MappingGenerator

```
5
6 namespace TestAutoMapper
7 {
8     public class MappingTest
9     {
10         ReadOnlyCollection<AddressDTO> Map(IList<AddressEntity> addresses)
11         {
12             return addresses;
13         }
14     }
15 }
16
```

Generate explicit conversion   CS0266 Cannot implicitly convert type 'System.Collections.Generic.IList<TestAutoMapper.AddressEntity>' to 'System.Collections.ObjectModel.ReadOnlyCollection<TestAutoMapper....

Use expression body for methods

```
...
{
    return addresses;
    return addresses.Select(addressEntity =>
    {
        var addressDTO = new AddressDTO();
        addressDTO.BuildingNo = addressEntity.BuildingNo;
        addressDTO.City = addressEntity.City;
        addressDTO.FlatNo = addressEntity.FlatNo;
        addressDTO.Street = addressEntity.Street;
        addressDTO.ZipCode = addressEntity.ZipCode;
        return addressDTO;
    }).ToList().AsReadOnly();
}
...
Preview changes
Fix all occurrences in: Document | Project | Solution
```



# MappingGenerator

```
7 | {
8 |   0 references
9 |   public class MappingTest
10 |   {
11 |     0 references
12 |     public void Update(UserDTO source)
13 |     {
14 |       Execute(source);
15 |     }
16 |     1 reference
17 |     public void Execute(string login, string firstName, string lastName, int age)
18 |     {
19 |     }
```

```
67 | public void Execute( string firstName, string lastName, int age, string unitName, int unitId)
68 | {
69 |   var userDTO = new UserDTO();
70 | }
71 |
72 |
73 |
74 |
75 |
76 |
77 |
```



# MappingGenerator

```
89  
90 public UserEntity Create(string firstName, string lastName, int age)  
91 {  
92     return new UserEntity()  
93     {  
94     };  
95 }  
96 }  
97 }  
98
```



# MappingGenerator

```
11 public static class SampleFactory
12 {
13     // 0 references
14     public static SampleletObject New()
15     {
16         // 0 references
17         return new SampleletObject
18         {
19             // 0 references
20             // 0 references
21         };
22     }
23 }
24
25 // 2 references
26 public class SampleletObject
27 {
28     // 0 references
29     public string StringProperty { get; set; }
30
31     // 0 references
32     public int IntProperty { get; set; }
33
34     // 0 references
35     public float FloatProperty { get; set; }
36
37     // 0 references
38     public bool BoolProperty { get; set; }
39 }
```





# DDDDToolbox

```
6 namespace DDDToolboxTest
7 {
8     Oreferences
9     public class User
10    {
11        Oreferences
12        public string FirstName { get; set; }
13        Oreferences
14        public string LastName { get; set; }
15        Oreferences
16        public int Age { get; set; }
17    }
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
```



# DDDDToolbox

```
4
5 namespace Domain
6 {
7     public class Age
8     {
9         public int NumberOfYears { get; set; }
10    }
11 }
12
```

# Source Generators

```
// original.cs:  
partial class C  
{  
    void F() { }  
    int P { get; set; }  
    object this[int index] { get { return null; } }  
    event EventHandler E;  
}
```



```
// replace.cs:  
partial class C  
{  
    replace void F() { original(); }  
    replace int P  
    {  
        get { return original; }  
        set { original += value; } // P.get and P.set  
    }  
    replace object this[int index]  
    {  
        get { return original[index]; }  
    }  
    replace event EventHandler E  
    {  
        add { original += value; }  
        remove { original -= value; }  
    }  
}
```

# Sharpen





## Sharpen Results

- C# 5.0 (49 items)
  - Async and await (49 items)
    - Await equivalent asynchronous method (17 items)
    - Await task instead of calling Task.Result (3 items)
    - Await task instead of calling Task.Wait() (3 items)
    - Await Task.Delay() instead of calling Thread.Sleep() (2 items)
    - Await Task.WhenAll() instead of calling Task.WaitAll() (3 items)
    - Await Task.WhenAny() instead of calling Task.WaitAny() (4 items)
    - Consider awaiting equivalent asynchronous method (17 items)
- C# 6.0 (532 items)
  - Expression-bodied members (83 items)
  - Nameof expressions (449 items)
- C# 7.0 (252 items)
  - Expression-bodied members (84 items)
  - Out variables (168 items)
    - Discard out variables in method invocations (42 items)
    - Discard out variables in object creations (42 items)
    - Use out variables in method invocations (42 items)
    - Use out variables in object creations (42 items)
- C# 7.1 (78 items)
  - Default expressions (78 items)
    - Use default expression in optional constructor parameters (13 items)
    - Use default expression in optional method parameters (13 items)
    - Use default expression in return statements (52 items)

## Sharpen Results

- ▶ C# 6.0 (4962 items)
- ▶ C# 7.0 (6548 items)
- ▲ C# 7.1 (495 items)
  - ▶ Use default expression in optional method parameters (488 items)
  - ▲ Use default expression in return statements (7 items)
    - ▶ <NHibernate>\Collection\Generic\Persis
    - ▶ <NHibernate>\Impl\AbstractQueryImpl
    - ▶ <NHibernate>\Impl\CriterialImpl.cs (1 ite
    - ▶ <NHibernate>\Linq\ExpressionToHqlTranslationResults.cs (1 item)
    - ▶ <NHibernate>\Util\CollectionHelper.cs (1 item)
    - ▶ <NHibernate.Test>\NHSpecificTest\NH1136\MilestoneCollection.cs (1 item)

 Analyze Again

 Apply Suggestions



# Rider

The screenshot shows the JetBrains Rider IDE interface. The Explorer on the left shows a project structure with a folder named 'Analyzers' containing several analyzers. A red arrow labeled '1' points to this folder. The Settings dialog is open, showing the 'Roslyn Analyzers' section under 'Editor > Inspection Settings'. A red arrow labeled '3' points to the 'Roslyn Analyzers' option in the left-hand tree of the settings dialog. The right-hand pane of the settings dialog shows the configuration for 'Context Actions', with a red arrow labeled '2' pointing to the 'Context Actions' section. The status bar at the bottom indicates '51 errors in 7 files'.

1

2

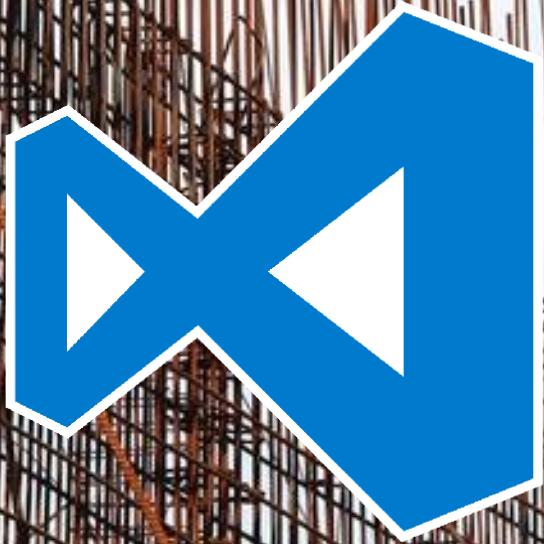
3

```
public byte[] ImageData { get; set; }
public List<int> LuckyNumbers { get; set; }
public int Total { get; set; }
public AddressDTO MainAddress { get; set; }
public ReadOnlyCollection<AddressDTO> Addresses { get; set; }
public int UnitId { get; set; }
```

6: TODO NuGet Unit Tests Terminal

51 errors in 7 files







# Linki



<https://github.com/cezarypiatek/Presentations/blob/master/RoslynLinks.md>



@cezary\_piatek



<https://github.com/cezarypiatek/>



<https://cezarypiatek.github.io/>