

 무료 전자 책

배우기

roslyn

Free unaffiliated eBook created from
Stack Overflow contributors.

#roslyn

.....	1
1:	2
.....	2
Examples.....	2
.....	2
.....	2
2: Roslyn	3
.....	3
.....	3
Examples.....	3
C #	3
SyntaxRewriter C #	4
3: Roslyn	6
Examples.....	6
C #	6
C # "Hello World"	6
VB.NET "Hello World"	7
C #	8
'var'	8
4:	10
.....	10
.....	10
Examples.....	10
.....	10
LINQ	10
CSharpSyntaxWalker	10
5:	12
.....	12
.....	12
Examples.....	12
.....	12
.....	12

6:	13
.....	13
.....	13
Examples.....	13
AdhocWorkspace	13
MSBuildWorspace ,	13
Visual Studio Extension VisualStudioWorkspace	13
.....	15

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [roslyn](#)

It is an unofficial and free roslyn ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official roslyn.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

1:

Roslyn .

- [API](#)
- [API](#)
- .

Examples

Roslyn NuGet .

- **C # VB** - `Microsoft.Net.Compilers` .

```
nuget install Microsoft.Net.Compilers
```
- **API** - `Microsoft.CodeAnalysis` .

```
nuget install Microsoft.CodeAnalysis
```

.NET SDK . :

- , **CodeFixes** C # Visual Basic
- **Visual Studio** (View -> Other Windows -> Syntax Visualizer) .
- Roslyn Quoter

C # API . .

-

Roslyn .

: <https://riptutorial.com/ko/roslyn/topic/2905/-->

2: Roslyn

Roslyn .

- Roslyn . ReplaceNodes . . .

Examples

C

ReplacementAttribute Attribute PreviousAttribute . . .

```
static async Task<bool> ModifySolution(string solutionPath)
{
    using (var workspace = MSBuildWorkspace.Create())
    {
        // Selects a Solution File
        var solution = await workspace.OpenSolutionAsync(solutionPath);
        // Iterates through every project
        foreach (var project in solution.Projects)
        {
            // Iterates through every file
            foreach (var document in project.Documents)
            {
                // Selects the syntax tree
                var syntaxTree = await document.GetSyntaxTreeAsync();
                var root = syntaxTree.GetRoot();
                // Finds all Attribute Declarations in the Document
                var existingAttributesList =
root.DescendantNodes().OfType<AttributeListSyntax>()
                // Where the Attribute is declared on a method
                .Where(curr => curr.Parent is MethodDeclarationSyntax)
                // And the attribute is named "PreviousAttribute"
                .Where(curr => curr.Attributes.Any(currentAttribute =>
currentAttribute.Name.GetText().ToString() == "PreviousAttribute"))
                .ToList();
                if (existingAttributesList.Any())
                {
                    // Generates a replacement for every attribute
                    var replacementAttribute = SyntaxFactory.AttributeList(
                        SyntaxFactory.SingletonSeparatedList(
SyntaxFactory.Attribute(SyntaxFactory.IdentifierName("ReplacementAttribute"),
                            SyntaxFactory.AttributeArgumentList(
                                SyntaxFactory.SeparatedList(new[]
                                {
                                    SyntaxFactory.AttributeArgument(
                                        SyntaxFactory.LiteralExpression(
                                            SyntaxKind.StringLiteralExpression,
SyntaxFactory.Literal(@"Sample"))
                                    )
                                }
                                )))))));
                    // Replaces all attributes at once.
                    // Note that you should not use root.ReplaceNode
                    // since it would only replace the first note
                }
            }
        }
    }
}
```

```

        root = root.ReplaceNodes(existingAttributesList, (node, n2) =>
replacementAttribute);
        // Exchanges the document in the solution by the newly generated
document
        solution = solution.WithDocumentSyntaxRoot(document.Id, root);
    }
}
// applies the changes to the solution
var result = workspace.TryApplyChanges(solution);
return result;
}
}

```

```

public class Program
{
    [PreviousAttribute()]
    static void Main(string[] args)
    {
    }
}

```

Method root.ReplaceNode

```

foreach(var node in existingAttributesList){
    root = root.ReplaceNode(node, replacementAttribute);
}

```

ReplaceNode . existingAttributesList ()

SyntaxRewriter C

"ReplacementAttribute" "PreviousAttribute" . SyntaxRewriter .

```

/// <summary>
/// The CSharpSyntaxRewriter allows to rewrite the Syntax of a node
/// </summary>
public class AttributeStatementChanger : CSharpSyntaxRewriter
{
    /// Visited for all AttributeListSyntax nodes
    /// The method replaces all PreviousAttribute attributes annotating a method by
ReplacementAttribute attributes
    public override SyntaxNode VisitAttributeList(AttributeListSyntax node)
    {
        // If the parent is a MethodDeclaration (= the attribute annotates a method)
        if (node.Parent is MethodDeclarationSyntax &&
            // and if the attribute name is PreviousAttribute
            node.Attributes.Any(
                currentAttribute => currentAttribute.Name.GetText().ToString() ==
"PreviousAttribute"))
        {
            // Return an alternate node that is injected instead of the current node
            return SyntaxFactory.AttributeList(
                SyntaxFactory.SingletonSeparatedList(

```

```

SyntaxFactory.Attribute(SyntaxFactory.IdentifierName("ReplacementAttribute"),
                        SyntaxFactory.AttributeArgumentList(
                            SyntaxFactory.SeparatedList(new[]
                                {
                                    SyntaxFactory.AttributeArgument(
                                        SyntaxFactory.LiteralExpression(
                                            SyntaxKind.StringLiteralExpression,
SyntaxFactory.Literal(@"Sample"))
                                )
                            }
                        ))));
    }
    // Otherwise the node is left untouched
    return base.VisitAttributeList(node);
}
}

/// The method calling the Syntax Rewriter
private static async Task<bool> ModifySolutionUsingSyntaxRewriter(string solutionPath)
{
    using (var workspace = MSBuildWorkspace.Create())
    {
        // Selects a Solution File
        var solution = await workspace.OpenSolutionAsync(solutionPath);
        // Iterates through every project
        foreach (var project in solution.Projects)
        {
            // Iterates through every file
            foreach (var document in project.Documents)
            {
                // Selects the syntax tree
                var syntaxTree = await document.GetSyntaxTreeAsync();
                var root = syntaxTree.GetRoot();

                // Generates the syntax rewriter
                var rewriter = new AttributeStatementChanger();
                root = rewriter.Visit(root);

                // Exchanges the document in the solution by the newly generated document
                solution = solution.WithDocumentSyntaxRoot(document.Id, root);
            }
        }
        // applies the changes to the solution
        var result = workspace.TryApplyChanges(solution);
        return result;
    }
}
}

```

```

public class Program
{
    [PreviousAttribute()]
    static void Main(string[] args)
    {
    }
}

```

Roslyn : <https://riptutorial.com/ko/roslyn/topic/5221/roslyn--->

3: Roslyn

Examples

C

- 1.
2. **NuGet** Microsoft.CodeAnalysis .
3. Microsoft.CodeAnalysis.MSBuild , System.Linq Microsoft.CodeAnalysis.CSharp.Syntax
4. Main .

```
// Declaring a variable with the current project file path.
const string projectPath = @"C:\<your path to the project\<project file name>.csproj";

// Creating a build workspace.
var workspace = MSBuildWorkspace.Create();

// Opening this project.
var project = workspace.OpenProjectAsync(projectPath).Result;

// Getting the compilation.
var compilation = project.GetCompilationAsync().Result;

// As this is a simple single file program, the first syntax tree will be the current file.
var syntaxTree = compilation.SyntaxTrees.First();

// Getting the root node of the file.
var rootSyntaxNode = syntaxTree.GetRootAsync().Result;

// Finding all the local variable declarations in this file and picking the first one.
var firstLocalVariablesDeclaration =
rootSyntaxNode.DescendantNodesAndSelf().OfType<LocalDeclarationStatementSyntax>().First();

// Getting the first declared variable in the declaration syntax.
var firstVariable = firstLocalVariablesDeclaration.Declaration.Variables.First();

// Getting the text of the initialized value.
var variableInitializer = firstVariable.Initializer.Value.GetFirstToken().ValueText;

// This will print to screen the value assigned to the projectPath variable.
Console.WriteLine(variableInitializer);

Console.ReadKey();
```

C # "Hello World"

```
Main : Console.WriteLine("Hello World")
```

```
.csproj .csproj .
```

```
Microsoft.CodeAnalysis NuGet .
```

```

const string projectPath = @"C:\HelloWorldApplication\HelloWorldProject.csproj";

// Creating a build workspace.
var workspace = MSBuildWorkspace.Create();

// Opening the Hello World project.
var project = workspace.OpenProjectAsync(projectPath).Result;

// Getting the compilation.
var compilation = project.GetCompilationAsync().Result;

foreach (var tree in compilation.SyntaxTrees)
{
    Console.WriteLine(tree.FilePath);

    var rootSyntaxNode = tree.GetRootAsync().Result;

    foreach (var node in rootSyntaxNode.DescendantNodes())
    {
        Console.WriteLine($" *** {node.Kind()}");
        Console.WriteLine($"      {node}");
    }
}

Console.ReadKey();

```

Hello World

VB.NET "Hello World"

```
Main : Console.WriteLine("Hello World")
```

```
.vbproj .vbproj.
```

Microsoft.CodeAnalysis NuGet

```

Const projectPath = "C:\HelloWorldApplication\HelloWorldProject.vbproj"

' Creating a build workspace.
Dim workspace = MSBuildWorkspace.Create()

' Opening the Hello World project.
Dim project = workspace.OpenProjectAsync(projectPath).Result

' Getting the compilation.
Dim compilation = project.GetCompilationAsync().Result

For Each tree In compilation.SyntaxTrees

    Console.WriteLine(tree.FilePath)

    Dim rootSyntaxNode = tree.GetRootAsync().Result

    For Each node In rootSyntaxNode.DescendantNodes()

        Console.WriteLine($" *** {node.Kind()}")
        Console.WriteLine($"      {node}")
    Next

```

Next

```
Console.ReadKey()
```

Hello World

C

```
var syntaxTree = CSharpSyntaxTree.ParseText(
@"using System;
using System.Collections;
using System.Linq;
using System.Text;

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

var root = syntaxTree.GetRoot() as CompilationUnitSyntax;

var namespaceSyntax = root.Members.OfType<NamespaceDeclarationSyntax>().First();

var programClassSyntax = namespaceSyntax.Members.OfType<ClassDeclarationSyntax>().First();

var mainMethodSyntax = programClassSyntax.Members.OfType<MethodDeclarationSyntax>().First();

Console.WriteLine(mainMethodSyntax.ToString());

Console.ReadKey();
```

Main .

'var'

```
var SemanticModel GetSymbolInfo() . MSBuildWorkspace MSBuildWorkspace . SyntaxRoot
SemanticModel VariableDeclarations Type .
```

```
var workspace = MSBuildWorkspace.Create();
var solution = workspace.OpenSolutionAsync("c:\\path\\to\\solution.sln").Result;

foreach (var document in solution.Projects.SelectMany(project => project.Documents))
{
var rootNode = document.GetSyntaxRootAsync().Result;
var semanticModel = document.GetSemanticModelAsync().Result;

var variableDeclarations = rootNode
.DescendantNodes()
.OfType<LocalDeclarationStatementSyntax>();
foreach (var varDeclaration in variableDeclarations)
```

```
{
    var symbolInfo = semanticModel.GetSymbolInfo(varDeclaration.Declaration.Type);
    var typeSymbol = symbolInfo.Symbol; // the type symbol for the variable..
}
```

Roslyn : <https://riptutorial.com/ko/roslyn/topic/4712/roslyn--->

4:

Roslyn Syntax API. Visual Basic C# .

- Roslyn .

Examples

Document () .

```
Document document = ... // Get document from workspace or other source
var syntaxRoot = await document.GetSyntaxRootAsync();
```

LINQ

LINQ . A ClassDeclarationSyntax () .

```
var allClassesWithNameStartingWithA = syntaxRoot.DescendantNodes()
    .OfType<ClassDeclarationSyntax>()
    .Where(x => x.Identifier.ToString().StartsWith("A"));
```

```
var allClassesWithAttributes = syntaxRoot.DescendantNodes()
    .OfType<ClassDeclarationSyntax>()
    .Where(x => x.AttributeLists.Any(y => y.Attributes.Any()));
```

CSharpSyntaxWalker

CSharpSyntaxWalker Visitor , Syntax Tree . A struct Syntax Walker .

```
public class StructCollector : CSharpSyntaxWalker
{
    public StructCollector()
    {
        this.Structs = new List<StructDeclarationSyntax>();
    }

    public IList<StructDeclarationSyntax> Structs { get; }

    public override void VisitStructDeclaration(StructDeclarationSyntax node)
    {
        if (node.Identifier.ToString().StartsWith("A"))
        {
            this.Structs.Add(node);
        }
    }
}
```

```
    }  
}
```

SyntaxWalker .

```
var structCollector = new StructCollector();  
structCollector.Visit(syntaxRoot); // Or any other syntax node  
Console.WriteLine($"The number of structs that have a name starting with the letter 'A' is  
{structCollector.Structs.Count}");
```

: <https://riptutorial.com/ko/roslyn/topic/9765/>-

5:

API `"" " ?"," ?"," ?"," / ?"`

- , .

Examples

semantic .

- Document

```
Document document = ...;  
SemanticModel semanticModel = await document.GetSemanticModelAsync();
```

- Compilation

```
CSharpCompilation compilation = ...;  
var semanticModel = await compilation.GetSemanticModel(syntaxTree);
```

- AnalysisContext . DiagnosticAnalyzer :

```
public override void Initialize(AnalysisContext context)  
{  
    context.RegisterSemanticModelAction(x =>  
    {  
        var semanticModel = x.SemanticModel;  
        // Do magical magic here.  
    });  
}
```

```
var syntaxRoot = await document.GetSyntaxRootAsync();  
  
var semanticModel = await document.GetSemanticModelAsync();  
var sampleMethodInvocation = syntaxRoot  
    .DescendantNodes()  
    .OfType<InvocationExpressionSyntax>()  
    .First();  
  
var sampleMethodSymbol = semanticModel.GetSymbolInfo(sampleMethodInvocation).Symbol;  
var referencesToSampleMethod = await SymbolFinder.FindReferencesAsync(sampleMethodSymbol,  
    document.Project.Solution);
```

: <https://riptutorial.com/ko/roslyn/topic/9772/>

6:

, C# .

- .NET Standard MSBuild . . .

Examples

AdhocWorkspace .

AdhocWorkspace AdhocWorkspace .

```
var workspace = new AdhocWorkspace();

string projectName = "HelloWorldProject";
ProjectId projectId = ProjectId.CreateNewId();
VersionStamp versionStamp = VersionStamp.Create();
ProjectInfo helloWorldProject = ProjectInfo.Create(projectId, versionStamp, projectName,
projectName, LanguageNames.CSharp);
SourceText sourceText = SourceText.From("class Program { static void Main() {
System.Console.WriteLine(\"HelloWorld\"); } }");

Project newProject = workspace.AddProject(helloWorldProject);
Document newDocument = workspace.AddDocument(newProject.Id, "Program.cs", sourceText);
```

MSBuildWorspace ,

MSBuildWorspace MSBuild (.sln) (.csproj, .vbproj) . . .

```
string solutionPath = @"C:\Path\To\Solution\Sample.sln";

MSBuildWorkspace workspace = MSBuildWorkspace.Create();
Solution solution = await workspace.OpenSolutionAsync(nancyApp);

var allDocumentsInSolution = solution.Projects.SelectMany(x => x.Documents);
```

Visual Studio Extension VisualStudioWorkspace

VisualStudioWorkspace . Visual Studio Extension .

[YourVSPackage]Package.cs . . .

```
protected override void Initialize()
{
    // Additional code...

    var componentModel = (IComponentModel)this.GetService(typeof(SComponentModel));
    var workspace = componentModel.GetService<VisualStudioWorkspace>();
}
```


MEF :

```
[Import(typeof(VisualStudioWorkspace))]  
public VisualStudioWorkspace ImportedWorkspace { get; set; }
```

VisualStudioWorkspace .

: <https://riptutorial.com/ko/roslyn/topic/9755/-->

S. No		Contributors
1		Community, Teodor Kurtev
2	Roslyn	SJP, Teodor Kurtev
3	Roslyn	andyp, Michael Rätzel, SJP, Stefano d'Antonio
4		Teodor Kurtev
5		Teodor Kurtev
6		Teodor Kurtev