**Automate using Roslyn**

[**https://www.youtube.com/watch?v=V4zqk4-LL1M&ab\_channel=NDCConferences**](https://www.youtube.com/watch?v=V4zqk4-LL1M&ab_channel=NDCConferences)

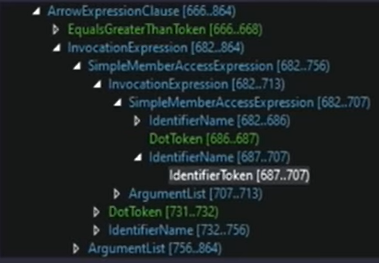


Every bit of the code is reprezented by a **syntax node**, and they have descendants nodes themselves forming a **syntax treee**, like a DOM/XML element

A node contains **Tokens** which are variables, identifier, semicolons

**SyntaxTrivia** are elements that are important but does not affect compilation, for example spacing

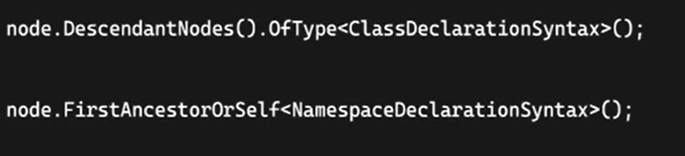
If installed code analyis tool -> view - >other windows -> syntax visualizer : Inspect syntax tree and understand it.



Topics

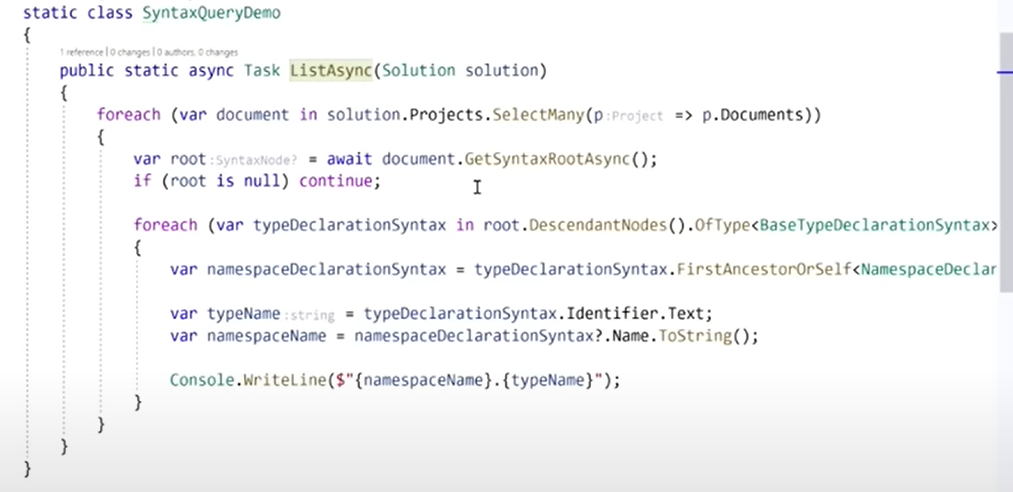
1. Navigating Syntax
2. CSharpSyntaxWalker
3. Semantic Model
4. Find all used types
5. Add ToLowerCase for every string
6. Roslyn Analyzer

**1.Navigating Syntax**



For specific node, gets all it's descentants that are of type ClassDelcaration syntax : 'public class Foo' -> ClassDeclarationSyntax, when you have this variable you cann access Acceser, IntetifierToken, class nametoken etc

FirstAncestorOrSelf -> going up on the tree, if you are on a node of class syntax, you can find it's namespace



searches thorugh every document in every project inside that solution

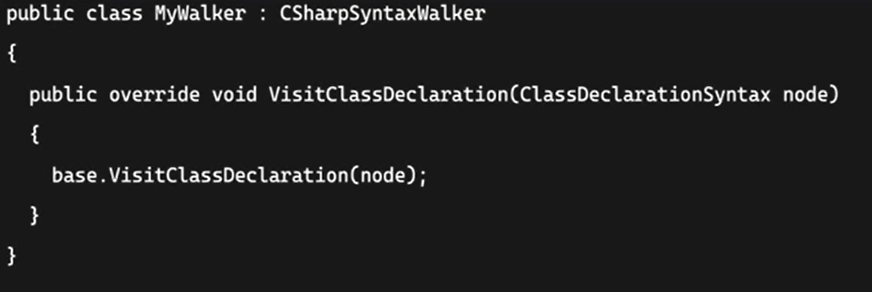
gets the syntax root for every document, it might return null if the docuemnt is not a syntaxy doc, appsetings, txt etc

searches through every baseDeclarationSyntax inside that document

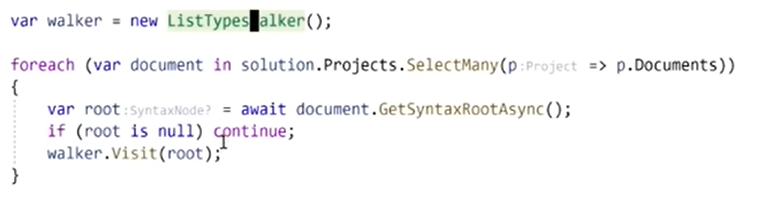
for every class/interface, it gets it's namespace declaration, typeName = name of the class found

//BaseTypeDelcarationSyntax is the base for class/interface declaration

**2.CSharpSyntaxWalker**

Used when you need to find a certain syntax inside the document   


//There is also **CSharpSyntaxVisitor**,which stops at the first found element, **CSharpSyntaxWalker** goes thorugh all the descendats until it finishes

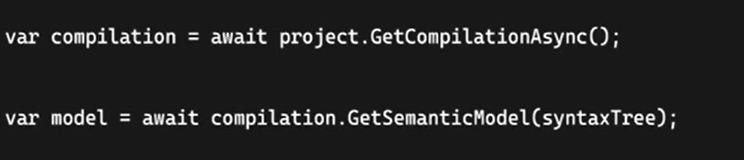




Which does the same thing as example 1

**3.Semantic Model**

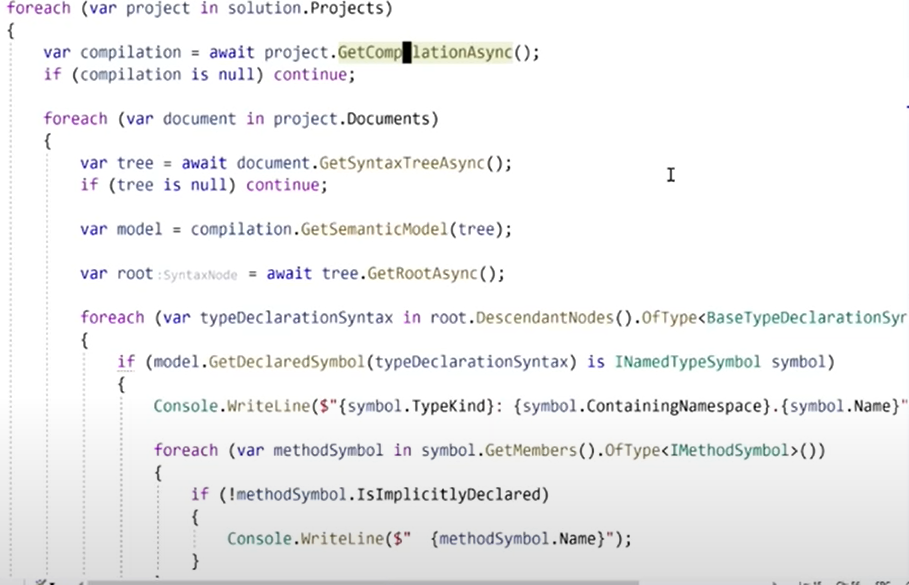
Transforming a syntax tree in a semantic model, making it easier to query working like Reflection



**Symbols**

Syntax tree has syntax nodes

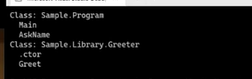
Semantic model has symbols : MethodSymbol,NamedTypeSymbol, like method info, type info from reflection



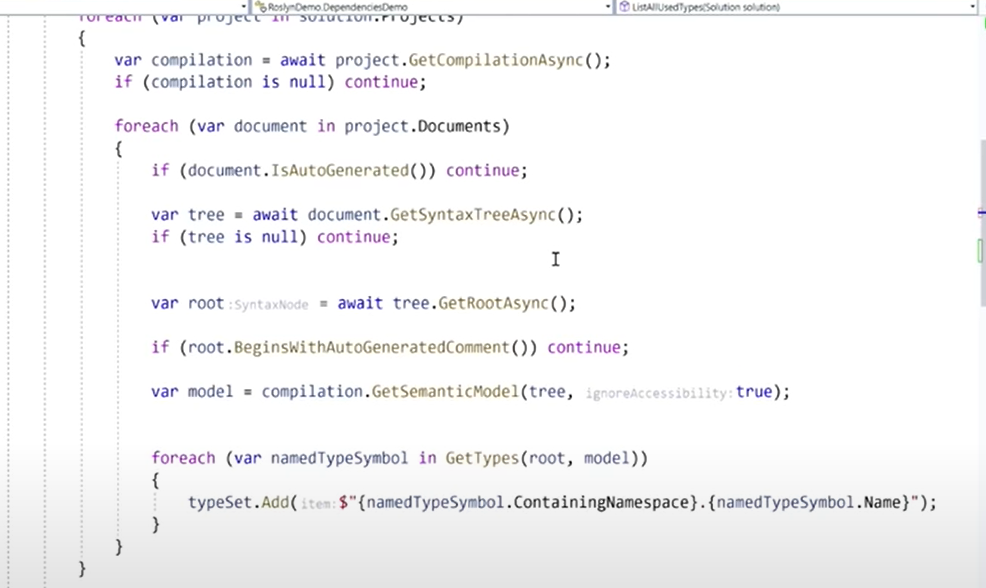
symbol.TypeKind -> class,interface,struct etc

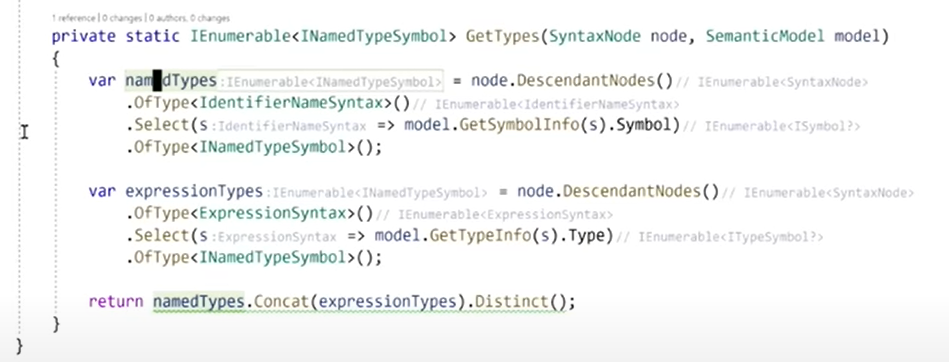
IsImplictlyDeclared -> default emtpy constructor, getters and seters created by properity sugar syntax

For every class/interface, it writes the type, it's namespace and the type's name and every method inside it



**4.Find all used types**

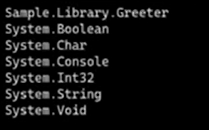




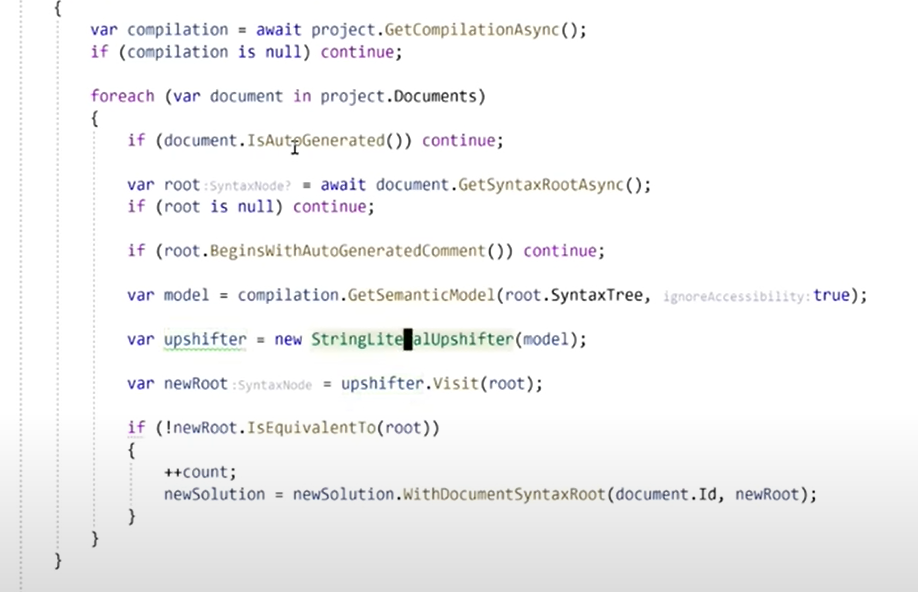
IdentifierNameSyntax : variables,fields,properties

ExpresionSyntax : x = x+1 ,x -> int => expresion is int, or function types

Find things that are NamedType, because some are ErrorType and the compiler must run even tho the code is broken



**5. Add ToUpper on every string**

****

**6.Roslyn Analyzers**