Roslyn

Rozszerzanie możliwości kompilatora C#

Grzegorz Wodniczak

Plan prezentacji

- Co to jest *Roslyn*?
- *Syntax Tree*
- *Analyzer with Code Fix*
- Podsumowanie

Co to jest *Roslyn*? / Wprowadzenie

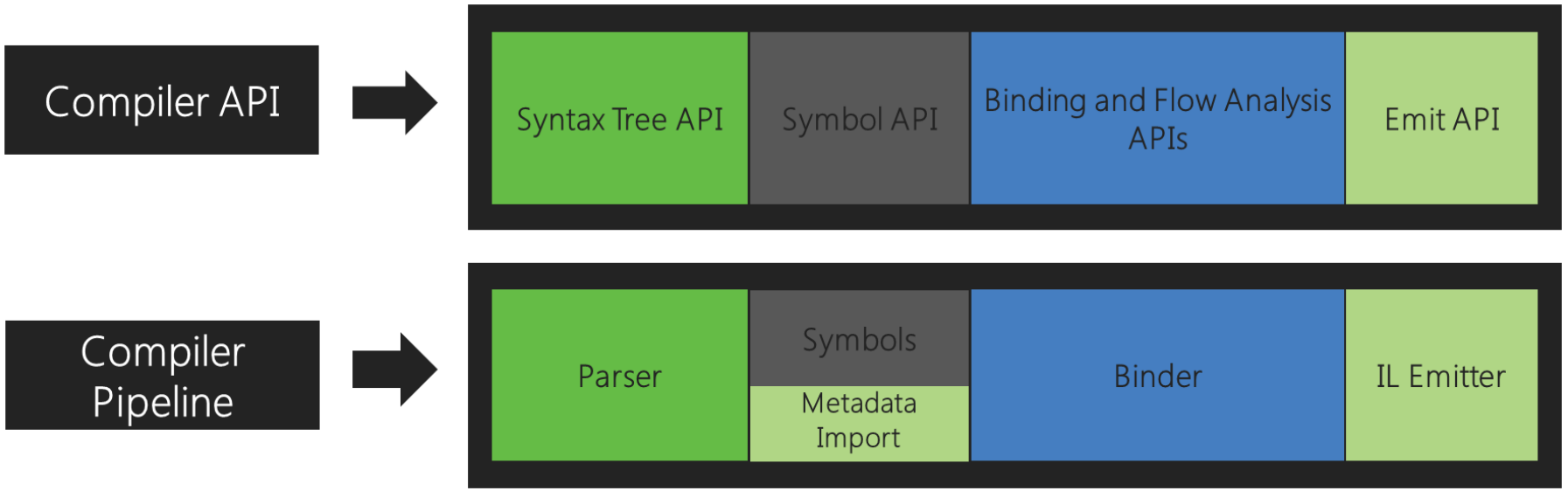
- Oficjalna nazwa: *.NET Compiler Platform*
- Nowy kompilator języków:
  - C# - `csc.exe`
  - VB (*Visual Basic*) - `vbc.exe`
- Wykorzystywany w *Visual Studio 2015* (wersja 14.0)
- Licencja: *Apache License 2.0*

Co to jest *Roslyn*? / Przyczyny powstania

- Potrzeba napisania kompilatora w języku C# zamiast C++
- Potrzeba scalenia różnych wersji kompilatora
- *„Opening up the black boxes and allowing tools and end users to share in the wealth of information compilers have about our code. Instead of being opaque source-code-in and object-code-out translators, through the .NET Compiler Platform (Roslyn), compilers become platforms—APIs that you can use for code related tasks in your tools and applications.”*

Źródło: https://github.com/dotnet/roslyn/wiki

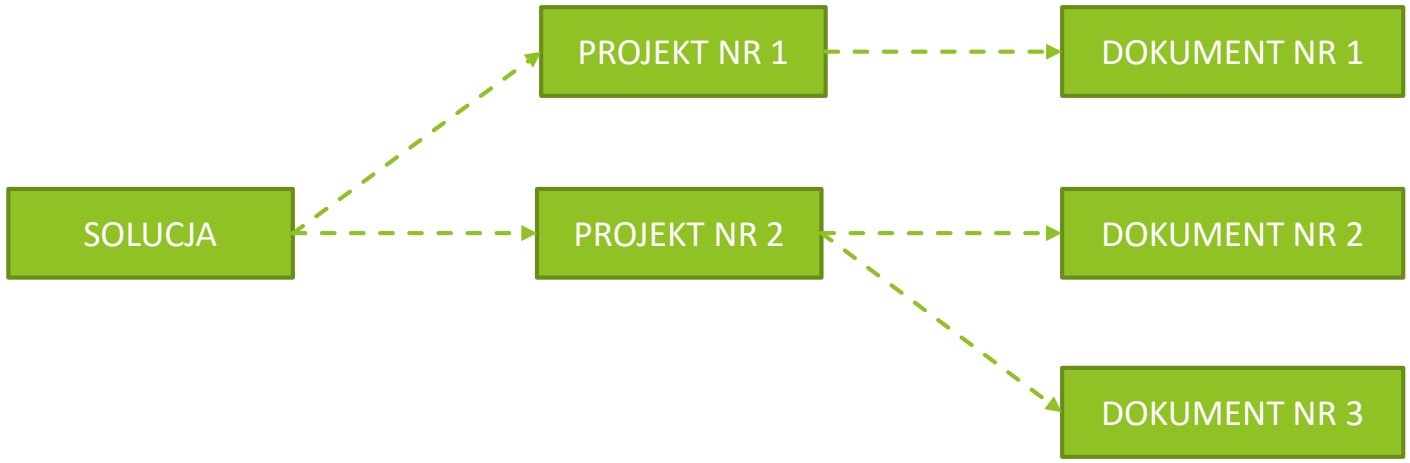
Co to jest *Roslyn*? / Compiler APIs



Źródło: https://github.com/dotnet/roslyn/wiki

Co to jest *Roslyn*? / Workspaces APIs

- Zarządzanie obszarem roboczym, tzn. solucją i jej składowymi



- Przykładowe usługi:
  - Wyszukiwanie (klasa *SymbolFinder*)
  - Formatowanie (klasa *Formatter*)
  - Przemianowanie (klasa *Renamer*)

Syntax Tree / Definicja

- *Syntax Tree* – reprezentacja kodu źródłowego w postaci drzewa składniowego
- Elementy:
  - *Syntax Node*

    Nieatomowy, istotny element kodu (m.in. deklaracja, wyrażenie, lista)
  - *Syntax Token*

    Atomowy, istotny element kodu (m.in. słowo kluczowe, operator, identyfikator)
  - *Syntax Trivia*

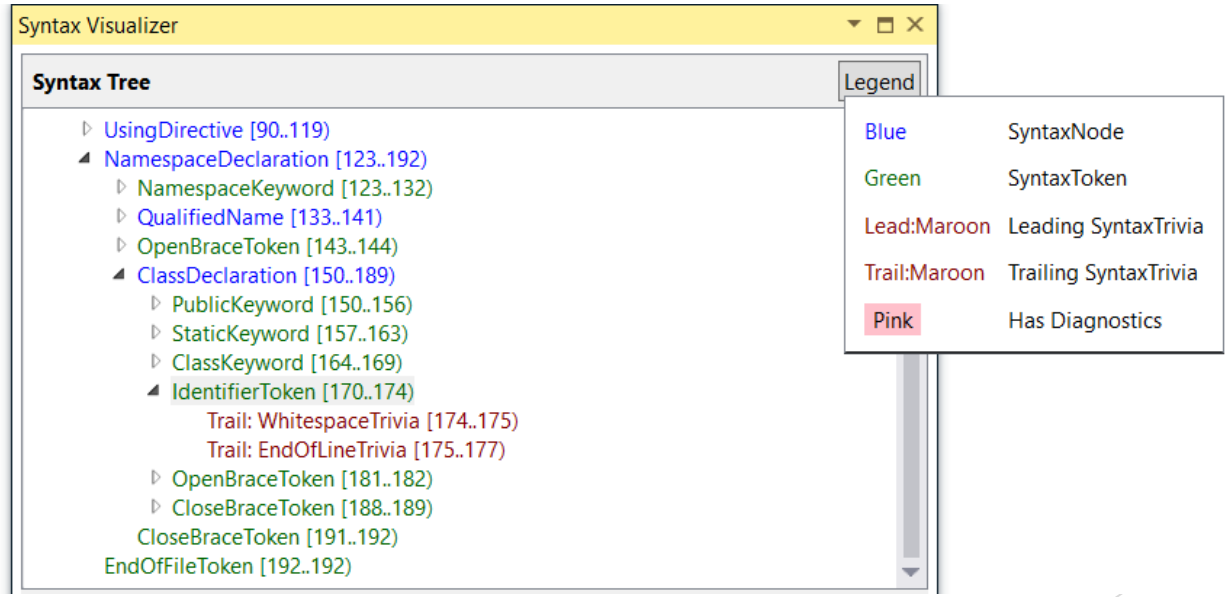
    Atomowy, nieistotny element kodu (m.in. biały znak, komentarz)

Syntax Tree / Wtyczka *Syntax Visualizer*

- *Syntax Visualizer* – wtyczka do *Visual Studio 2015* ułatwiająca pracę z drzewami składniowymi

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace App.Core
{
    public static class Test
    {
    }
}
```

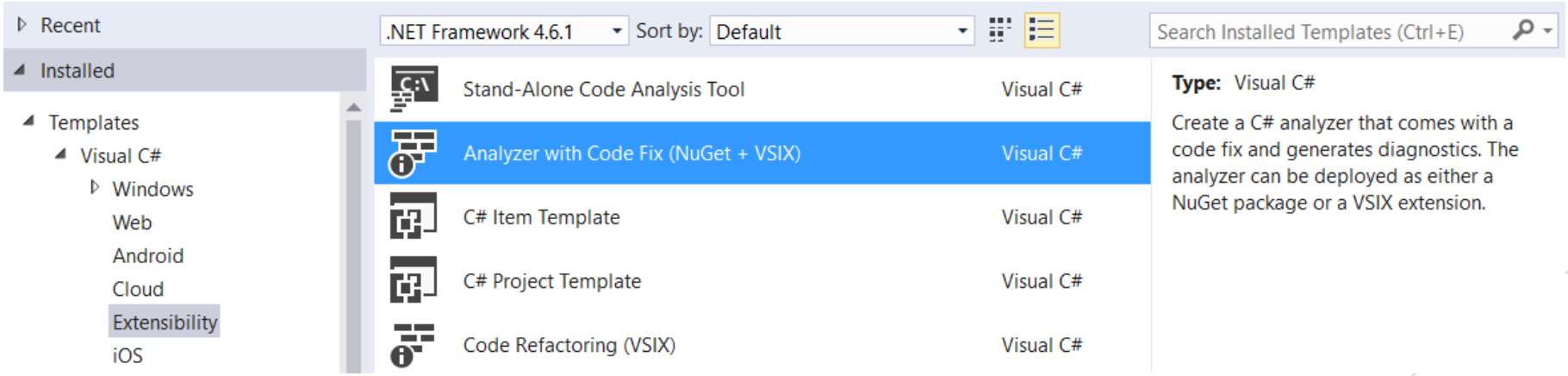


Analyzer with Code Fix / Wprowadzenie

- *Analyzer* – wykrywa nieprawidłowości w drzewie składniowym
- *Code Fix* – automatycznie poprawia wykryte nieprawidłowości
- Praktyczne zastosowanie:
  - Narzucenie konwencji kodowania (w zespole lub w konkretnym projekcie)
  - Wsparcie osób korzystających z zaprojekowanych przez nas bibliotek

Analyzer with Code Fix / Narzędzia

- *Visual Studio 2015*
  - Rozszerzenie: *Visual Studio Extensibility Tools*
  - Rozszerzenie: *.NET Compiler Platform SDK*
    - Pasek narzędzi: *Syntax Visualizer*
    - Szablon: *Analyzer with Code Fix (NuGet + VSIX)*



Analyzer with Code Fix / Demo nr 1

- Opis:

  Analizator sprawdzający, czy deklaracja interfejsu zawiera publiczny modyfikator dostępu

- Kod źródłowy:

  https://github.com/vebaspect/roslyn-analyzers

  Solucja: *InterfaceMustBePublic*

Analyzer with Code Fix / Demo nr 2

- Opis:

  Analizator sprawdzający, czy nazwa pliku źródłowego jest tożsama z nazwą zadeklarowanej w nim klasy

- Kod źródłowy:

  https://github.com/vebaspect/roslyn-analyzers

  Solucja: *FileNameMustBeTheSameAsClassName*

Podsumowanie / Przydatne adresy

- Oficjalna strona + dokumentacja + repozytorium kodu:

  https://github.com/dotnet/roslyn

- **Channel 9** / Tomas Herceg: *Roslyn – why should you care?*

  https://channel9.msdn.com/Series/NET-DeveloperDays-2015-on-demand/Roslyn--why-should-you-care-

- **Pluralsight** / Bart De Smet: *Introduction to the .NET Compiler Platform*

  https://app.pluralsight.com/library/courses/dotnet-compiler-platform-introduction

- **Syncfusion** / Alessandro Del Sole: *Roslyn Succinctly*

  https://www.syncfusion.com/resources/techportal/details/ebooks/roslyn