Roslyn

Compiler as a service

DEVELOPMENTOR
DEVELOPING PEOPLE WHO DEVELOP SOFTWARE

Agenda

- What is Roslyn
- Exploring the Compiler pipeline
- Scripting
- Code Issues and Actions

Roslyn

- Managed compilers
- Code analysis APIs
- Language service extensibility
- Read-Eval-Print-Loop (REPL)
- What it is not
  - Its not a generic compiler tool set

What will it mean for the IDE

- REPL
- Replaces rendering of text in the IDE
  - Formatting
  - Intelli sense
  - Refactorings
- More ReSharper like functionality
- Perhaps one day better than ReSharper?
  - Community based refactorings

What it could mean for you

- Code generators
- Extend Visual Studio IDE
  - Build your own code smells plugins
  - Make code comply with coding standards
- Add C# or VB scripting extension points to your application

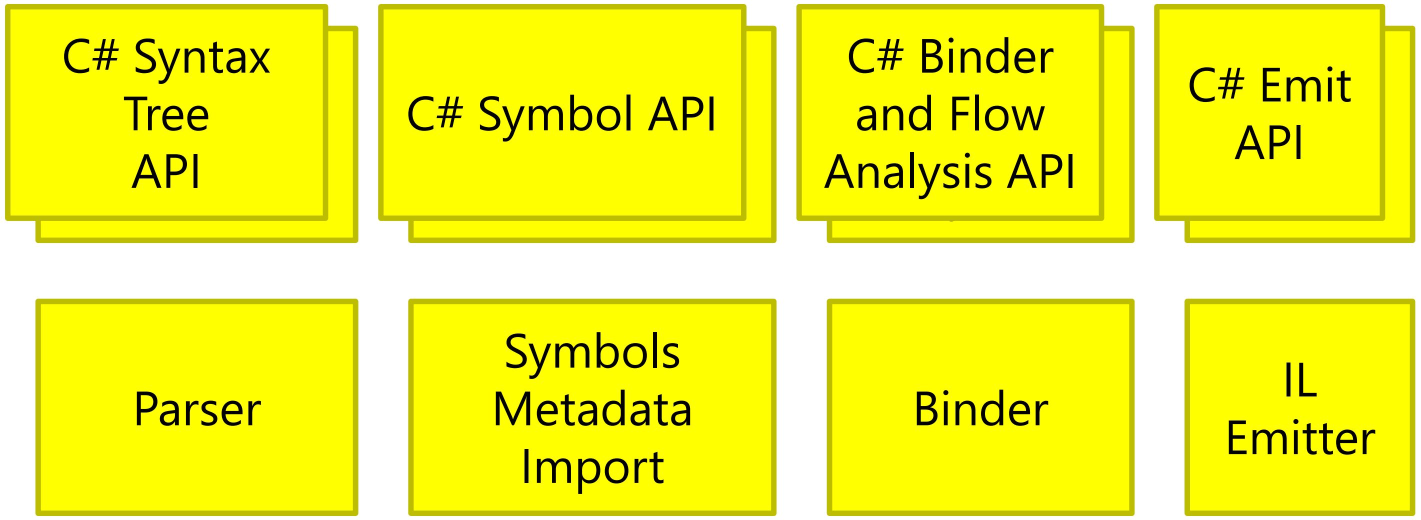
Parts of a compiler

- Pipeline

Parser	Tokenises produces and matches tokens against language grammar
Symbols Metadata Import	Identify all the declarations and import any Additional data, to form a set of symbols
Binder	Wire up method calls, gains understanding of the code, loads meta data
IL Emitter	Performs the code generation

Compiler API

- A different API for each language



Syntax API

- Walk the source code in terms of either
  - Tokens (terminals)
  - Syntax nodes (non terminals)
- Trivia
  - Comments, white space preserved
- Navigate tree using Linq or Visitor pattern
- Highly fault tolerant
  - Does its best, expected to work with partially completed code

Semantic model, Syntax parsing only takes you so far

- Parsing only gets you so far
  - Validates the source is grammatically correct
- Compilation (Binding)
  - Ability to ask questions
    - What symbols have been declared globally
    - What symbols have been declared for a given block of code
    - What are the exit points for a given block of code

Modifying the code

- Syntax Trees are immutable
- Replacing a SyntaxNode produces a “new” tree
- Don’t PANIC its not a complete new copy, it keeps track of the changes.

Emitting IL

- Compiling just analyses the code, ready for emitting IL
- Explicitly call Emit to produce IL
  - Compile to file
  - Compile to ReflectionEmit API
    - Dynamically created assembly available to use

Code Generation, Is Roslyn the right tool ?

- Build ASTs by hand
  - Cumbersome
- Supports more language features than the Code Dom,
- Possibly good for
  - Tweak existing code structure
  - Translate “other” language to C# and VB
  - DSL built upon C# or VB
- T4 templates still very attractive

Workspace, processing more than one file

- Visual studio without the UI
  - Understands a solution
  - Allows compilation of whole projects
- More convenient entry point, assembly references etc all done.
- Modify Solutions and Projects, code analyses across entire solution

Scripting

- Allow sophisticated extension points in application
- Advanced users extend functionality
- Possible DANGER
  - Consider sandboxing via CAS

Visual Studio Plugins

- Roslyn takes over code rendering in Visual Studio
- Offers own plugin framework
  - Code Issues
  - Code Actions “Quick Fix”
  - Outliners
  - Refactorings
  - Completions
- Use Roslyn API’s to query,modify the AST

Additional benefits

- Built around immutable data structures
  - Parallel compile just falls out
- ASP.NET can now host compiler in process
  - Faster page compilations

How do you get it

- Available as a CTP
- http://msdn.microsoft.com/en-us/vstudio/roslyn.aspx
  - Partally available via NuGet
- When will it be released ?
  - Who knows ….

Summary

- Compiler is no longer a black box
  - Makes it **easier** to build productivity tool
- Adding scripting to your application is now low cost possibility, but consider sandboxing
- Provides a great foundation for more sophisticated refactoring support
- Is the average team going to build refactoring tools ?