Masaryk University
Faculty of Informatics

Custom Roslyn Tool for Real-Time Static Code Analysis

Master's Thesis

Zuzana Dankovčíková

Brno, Spring 2017

Masaryk University
Faculty of Informatics

Custom Roslyn Tool for Real-Time Static Code Analysis

Master's Thesis

Zuzana Dankovčíková

Brno, Spring 2017

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Zuzana Dankovčíková

Advisor: Bruno Rossi PhD

Acknowledgement

TODO: This is the acknowledgement…

Abstract

TODO: This is the abstract ...

Keywords

roslyn, C#, compilers, code review, .NET compiler platform, Kentico, analyzer, code fix..., …

Contents

List of Tables

List of Figures

1 Introduction

TODO...

Ideas:

What is code quality, why is it important, tool that support it.. compilers, diversion ... aaaand here comes Roslyn which provides compiler as a platform.

In the .NET world, the compiler used to be a black box that given the file paths to the source text, produced an executable (see Figure 3.1). In order to do that, compiler has to collect large amount of information about the code it is processing. This knowledge, however, was unavailable to anyone but the compiler itself and it was immediately forgotten once the translated output was produced [**roslyn-overview-github**].

Why is this an issue when for decades this black-boxes served us well? Programmers are increasingly becoming reliant upon the powerful integrated development environments (IDEs). Features like IntelliSense, intelligent rename, refactoring or "Find all references" are key to developers' productivity; and even more so in an enterprise-size systems.

This gave a rise to number of tools that analyze the code for common issues and are able to suggest a refactoring. The problem is that that such tool needs to parse the code first in order to be able to understand and analyze it. As a result companies need to invest fair amount of resources to duplicate the logic that the .NET compiler already possesses. Not only is it possible that the compiler and the tool may disagree on some specific piece of code, but with every new version of C# the tool needs to be updated to handle new language features[**dot-net-development-using-the-compiler-api**].

With roslyn.. etc. etc. .. API for analysis.. use in companies for custom analyzers... etc. etc.... https://github.com/dotnet/roslyn/wiki/Roslyn Overview – motivation Make sure to stress out that ".NET Compiler Platform" and "Roslyn" names will be used interchangably as it is in Roslyn Succinctly on page 11.

2 Code Quality and Static Code Analysis

TODO

3 .NET Compiler Platform

As per [**dragon-book**], compiler is a program that can read a program in a *source* language and translate it into an equivalent program in *target* language while reporting any errors detected in the translation process.

In the .NET world, the compiler used to be a black box that given the file paths to the source text, produced an executable (see Figure 3.1). This perception was changed in 2015 when Microsoft introduced the .NET Compiler Platform (commonly referred to as "Project Roslyn").

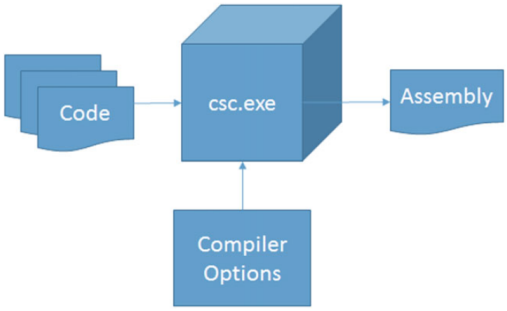

Figure 3.1: Compiler as a black box[**dot-net-development-using-the-compiler-api**]

Not only have been compilers for both VisualBasic and C# rewritten into entirely managed code, they also expose the internals of the compiler pipeline via a public .NET API [1]. This makes them a platform (also known as *compiler-as-as-service*) with rich code analysis APIs that can be leveraged by developers to perform analysis, code generation or dynamic compilation in their own programs [**roslyn-succinctly**]. Those can be later easily integrated into VisualStudio all without the hard work of duplicating compilers' parsing logic.

This chapter will take a look at how the Roslyn API layers are structured, how the original source code is represented by the compiler and how developers can build tools upon the compiler's API. Lastly it will provide a short overview and evaluation of other tools that

1. Application Programming Interface

were used before Roslyn or have emerged thanks to .NET Compiler Platform.

Note that although Roslyn provides equivalent APIs for both VisualBasic and C#, this thesis will only focus on the C# part, since only that one is relevant for the practical part of the thesis.

3.1 Compiler APIs

3.2 Syntax Tree API

3.3 Semantics API

3.4 Workspaces API

3.5 Other Tools

A Upgrades and Versioning

TODO...