.NET Development Using the Compiler API

Jason Bock

Apress®

.NET Development Using the Compiler API

Jason Bock

Apress®

.NET Development Using the Compiler API

Jason Bock
Shakopee
Minnesota, USA

ISBN-13 (pbk): 978-1-4842-2110-5 ISBN-13 (electronic): 978-1-4842-2111-2
DOI 10.1007/978-1-4842-2111-2

Library of Congress Control Number: 2016945755

Contents at a Glance

Contents

About the Author

Jason Bock is a Practice Lead for Magenic (`http://www.magenic.com`) and a Microsoft MVP (C#). He has 20 years of experience working on a number of business applications using a diverse set of frameworks and languages such as C#, .NET, and JavaScript. He is the author of *Metaprogramming in .NET, Applied .NET Attributes,* and *CIL Programming: Under the Hood of .NET*. He has written numerous articles on software development issues and has presented at a number of conferences and user groups. He is a leader of the Twin Cities Code Camp (`http://www.twincitiescodecamp.com`). Jason holds a Master's degree in electrical engineering from Marquette University. Visit his website at `http://www.jasonbock.net`.

About the Technical Reviewer

A prolific writer on cutting-edge technologies, **Fabio Claudio Ferracchiati** has contributed to more than a dozen books on .NET, C#, Visual Basic, and ASP.NET. He is a .NET Microsoft Certified Solution Developer and lives in Milan, Italy. You can read his blog at `http://www.Ferracchiati.com`.

Acknowledgments

I'd like to thank Apress for contacting me at VSLive in 2015 and asking me if I'd be interested in writing a book on the Compiler API. Getting the opportunity to write on a topic that I've been deeply interested in since I heard about it eight years ago was something I just couldn't pass up. Specifically, I'd like to thank Anne Marie Walker, James DeWolf, Mark Powers, Melissa Maldonado, and Fabio Claudio Ferracchiati for their assistance, guidance, and editing prowess—they made the book far better than it would have been if I did it on my own.

Thanks also to Magenic, especially Greg Frankenfield and Paul Fridman, for creating and growing a great place to work. I've been with Magenic for 15 years, and I feel fortunate to work for a dynamic and innovative company where I fit in. Here's to another 15 (or more!) years. I'd also like to thank Jeff Ferguson for providing a couple of figures for me that are used in the book.

Finally, I'd like to thank my family for their support and encouragement: my wife Liz and my sons Hayden and Ryan. I am grateful to have found someone like Liz and that we've been able to have two awesome sons.

Introduction

Most developers I know typically view coding as a means to an end. That is, they write the code to satisfy the requirements set forth by the business. The code is interpreted or compiled, but either way, the final result is machine code that executes and (hopefully) does the right thing.

However, there's more to software development than just that. I'm not talking about process or patterns per se; what I'm getting at is for developers to view their code in a more analytical way. Throughout my career, I've run into numerous cases in which I would've loved to have the ability to analyze my code so I could find errors quickly. I've also wanted to be able to extend and augment languages in certain ways so I didn't have to write the same code over and over again. The primary language that I've used throughout my career has been C#, and although C# is a fine language to develop in, it seemed to lack these dynamic, analytical capabilities.

That's no longer the case. Microsoft has provided public, open-source components in its Compiler API that allows developers to create analyzers that will help them detect problematic issues. This API also empowers developers to build code at runtime to create amazing, dynamic applications and libraries. Because all of this code is open source, it's available to read and contribute to. Enabling .NET developers to shape and mold the future of the .NET compilation system is a wonderful thing to behold, and it's exciting to see the development community embrace this model.

I wrote this book to help you navigate this new open-source API world. In it, I demonstrate how to use the Compiler API to write custom analyzers and refactorings to improve your code base. I show you how to use the Scripting API (part of the Compiler API) to use C# as a scripting language, a feature that was essentially unavailable to C# developers. I also illustrate how to use the Compiler API in innovative ways that go beyond these typical scenarios. My hope is that when you've finished this book, you'll view C# and the ecosystem that supports it in a fundamentally different (and hopefully positive!) way—as a language that is open in terms of its implementation and its community involvement.

Who This Book Is For

This book is for architects and developers who have experience with C# and want to dive deeper into how code is compiled and executed. There's no expectation that the reader has any experience with compilers, but I do assume that the reader has foundational knowledge of C#.

Chapter Contents

To give you a feel for the content in the book, here's a brief synopsis of each chapter.

- Chapter 1—You'll get an introduction to the Compiler API and its constituent parts: syntax trees, semantic models, and formatters.
- Chapter 2—This chapter covers diagnostics. You'll learn how to write analyzers and build code fixes to automate code corrections.
- Chapter 3—Refactoring code is a primary tenant for developers. This chapter shows you how to write refactorings to clean up your code base.
- Chapter 4—C# is now a scripting language! In this chapter, you'll see how the Scripting API works.
- Chapter 5—You'll discover how developers are using the Compiler API to empower their own components and get a preview of a future C# feature based on the Compiler API that could fundamentally change how you write code in C#.

Code Samples

Throughout the book I show code snippets to illustrate various aspects of the Compiler API. You'll find all of the code at `https://github.com/jasonbock/compilerAPIBook`. The folder structure is set up to map the code content to each chapter of the book.

Errata

The author, the technical reviewers, and many Apress staff have made every effort to find and eliminate all errors from this book's text and code. Even so, there are bound to be one or two glitches left. To keep you informed, there's an Errata tab on the Apress book page (`www.apress.com/9781484221105`). If you find any errors that haven't already been reported, such as misspellings or faulty code, please let us know by e-mailing `support@apress.com`.

Customer Support

Apress wants to hear what you think—what you liked, what you didn't like, and what you think could be done better next time. You can send comments to `feedback@apress.com`. Be sure to mention the book title in your message.

Contacting the Author

Feel free to follow me on Twitter at @jasonbock. My web site is `http://www.jasonbock.net`. You can also contact me via e-mail at `jasonb@magenic.com`.

CHAPTER 1

An Overview of the Compiler API

This chapter covers the basics of the Compiler API, including the essentials of a compiler and their history in the .NET world. You'll learn how to compile code and the trees that constitute the fundamental API data structure. You'll discover how to build your own trees from scratch and navigate their content. Finally, we'll explore annotating and formatting trees.

From Closed to Open

Compilers are used more than any other tool by a developer. Every time you tell Visual Studio to build your code, you're invoking csc.exe, which is the C# compiler. Without compilers, your C# code would be worthless. In this section, you'll gain an understanding of what compilers do, how they've been designed in the .NET world, and how they have changed in .NET 4.6.

Note You can invoke csc.exe directly from the command line, but generally most .NET developers will use it indirectly through Visual Studio or some other IDE.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-2111-2_1) contains supplementary material, which is available to authorized users.