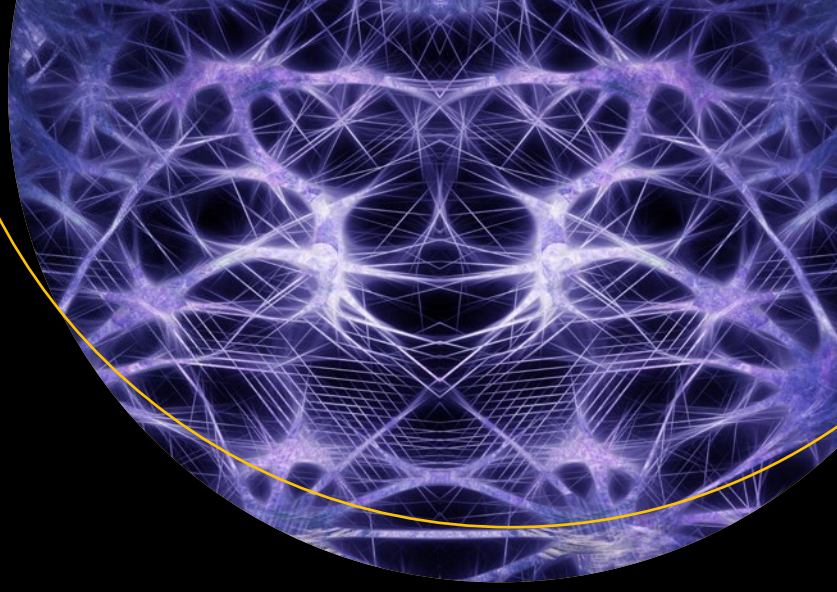


Pro .NET 5 Custom Libraries

Implementing Custom .NET Data Types

Roger Villela

Apress®

Pro .NET 5 Custom Libraries

Implementing Custom .NET Data Types

Roger Villela

Apress®

Pro .NET 5 Custom Libraries: Implementing Custom .NET Data Types

Roger Villela
Sao Paulo, São Paulo, Brazil

ISBN-13 (pbk): 978-1-4842-6390-7 ISBN-13 (electronic): 978-1-4842-6391-4
https://doi.org/10.1007/978-1-4842-6391-4

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Matthew Moodie
Coordinating Editor: Shrikant Vishwakarma

Cover designed by eStudioCalamar

Cover image designed by Pexels

Distributed to the book trade worldwide by Springer Science+Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, email orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC, and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at www.apress.com/bulk-sales.

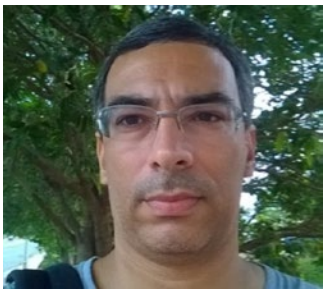
Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6390-7. For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

This book is dedicated to my mother, Marina Roel de Oliveira.
January 14, 1952 to March 17, 2017 (†)

Table of Contents

About the Author



Roger Villela is a software engineer and entrepreneur with almost 30 years of experience in the industry and works as an independent professional. Currently, he is focused on his work as a book author and technical educator and specializes in the inner workings of orthogonal features of the following Microsoft development platforms and specialized application programming interfaces (APIs):

- Microsoft Windows operating system base services
- Microsoft Windows APIs architecture and engineering
- Microsoft Universal Windows Platform (UWP)
- Microsoft WinRT platform
- Microsoft .NET Framework implementation of the runtime environment (Common Language Runtime [CLR])

His work is based on Microsoft Windows software development kit (SDK) tools and libraries, Microsoft Visual Studio, and platform foundational APIs, architectures, and engineering. He works with the Microsoft Windows operating system, incorporating the following programming languages, extensions, and projections:

- C/C++
- Assembly (Intel IA-32/Intel 64 [x64/amd64])
- Component extensions/projections for runtimes
- C++/CLI
- C++/CX
- C++/WinRT
- C#
- Common Intermediate Language (Microsoft Intermediate Language [MSIL]) implementation for CLR platforms

About the Technical Reviewer

Carsten Thomsen is a back-end developer primarily, but he works with smaller front-end bits as well. He has authored and reviewed a number of books, and created numerous Microsoft Learning courses, all to do with software development. He works as a freelancer/contractor in various countries in Europe, using Azure, Visual Studio, Azure DevOps, and GitHub as some of his tools. He is an exceptional troubleshooter, asking the right questions, including the less-logical ones (in a most-logical to least-logical fashion). He also enjoys working with architecture, research, analysis, development, testing, and bug fixing. Carsten is a very good communicator with great mentoring and team-lead skills, and he also excels at researching and presenting new material.

Acknowledgments

I want to thank to the Apress team who worked with me on this book: Smriti Srivastava (Acquisitions Editor), Shrikant Vishwakarma (Coordinating Editor), Matthew Moodie (Development Editor), Welmoed Spahr (Managing Director), and Carsten Thomsen (Technical Reviewer). It was a pleasure and an honor to work with such a highly professional team.

I also want to thank my parents, with a special nod to my dad (Gilberto), my two brothers (Eder and Marlos and his wife Janaína), my nephew Gabriel, my nieces Lívia and Rafaela, and my cousin Ariadne Villela.

I must also express special thanks to my friends Paula Carolina Damasio, Alessandro Augusto de Jesus, and Neide Pimenta. I also want to acknowledge and thank all the people who work really hard on team Praxio Tecnologia developing one of the greatest specialized enterprise resource planning (ERP) products on the market; congratulations to all of you for your efforts.

I also want to thank my professional colleagues and friends who have worked with me throughout the years.

Introduction

This book covers programming with .NET 5 to develop custom data types and custom libraries for use on Microsoft Windows, Linux, and Apple macOS. These custom libraries can be used in different operating system platforms because they are written using .NET 5 (a cross-platform implementation of the ECMA-335 specification) and because all source code is written in the C# programming language and uses only cross-platform Base Class Library (BCL) types.

This book focuses on how to best exploit the .NET 5 custom data types for software libraries so that companies and software engineers can design and implement internal/commercial tools for various scenarios on myriad target platforms. Contextual modeling and planning is difficult without a fundamental understanding of the .NET 5 platform, which this book seeks to provide. The book also covers internal aspects of the BCL .NET types and APIs, with walkthroughs covering the implementation process of custom .NET data types and .NET custom libraries.

You will also learn about .NET assembly and .NET module structures, the inner workings of the BCL implementation on the .NET platform, custom data types available through the .NET platform, and how to write a custom library that incorporates .NET APIs available through the .NET BCL.

CHAPTER 1

.NET Platform

This chapter provides an overview of .NET 5 (previously .NET Core) and describes the fundamental architectural and the engineering features that you should expect in any implementation of .NET 5 (regardless of hardware, operating system, or execution system).

Acronyms

The following acronyms are introduced in this chapter:

- Base Class Library (BCL)
- Common Intermediate Language (CIL)
- Common Language Infrastructure (CLI)
- Common Language Runtime (CLR)
- Common Type System (CTS)
- Framework Class Library (FCL) (Although not specific to the .NET Framework implementation, the term is used for the full range of .NET types available in an official distribution of .NET.)
- Intermediate Language (IL)
- Microsoft Intermediate Language (MSIL)
- Virtual Execution System (VES)
- Windows Presentation Foundation (WPF) (a.k.a. *execution engine*)

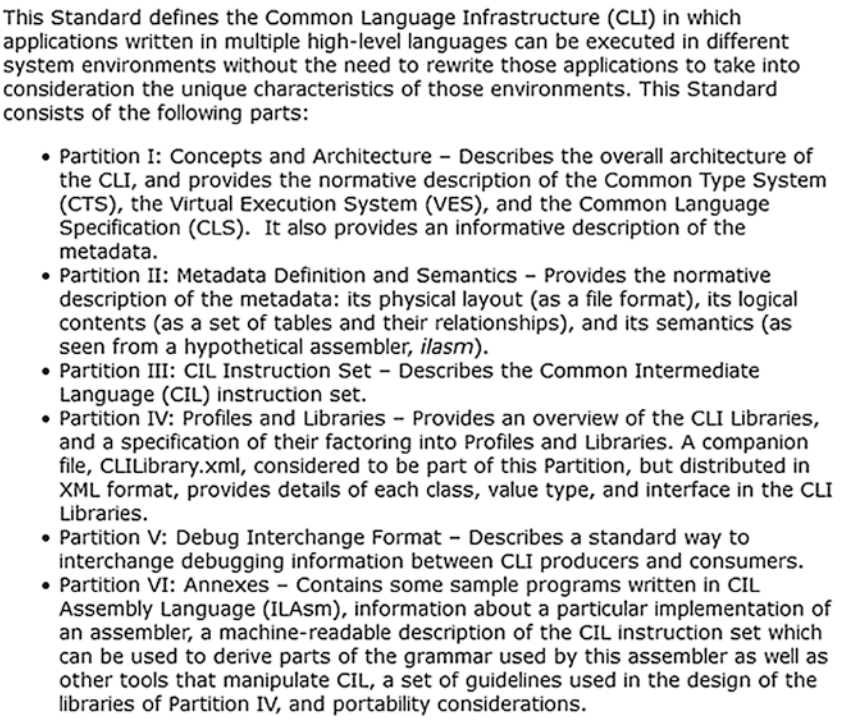
R. Villela, *Pro .NET 5 Custom Libraries*, https://doi.org/10.1007/978-1-4842-6391-4_1

ECMA-335 and .NET

ECMA-335

The ECMA-335 standard specification defines the Common Language Infrastructure (CLI), which includes a set of conceptual definitions and rules to be followed and engineering mechanisms to be implemented, independent of the target operating system and hardware platforms. The CLI ensures that applications, components, and libraries can be written in multiple high-level languages and can be executed in different target system environments without needing to be rewritten.

We can access the ECMA-335 specification at www.ecma-international.org/publications/standards/Ecma-335.htm. Figure 1-1 shows an excerpt. The download link is www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf, and the XML specification download link is www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.zip.



This Standard defines the Common Language Infrastructure (CLI) in which applications written in multiple high-level languages can be executed in different system environments without the need to rewrite those applications to take into consideration the unique characteristics of those environments. This Standard consists of the following parts:

- Partition I: Concepts and Architecture – Describes the overall architecture of the CLI, and provides the normative description of the Common Type System (CTS), the Virtual Execution System (VES), and the Common Language Specification (CLS). It also provides an informative description of the metadata.
- Partition II: Metadata Definition and Semantics – Provides the normative description of the metadata: its physical layout (as a file format), its logical contents (as a set of tables and their relationships), and its semantics (as seen from a hypothetical assembler, *ilasm*).
- Partition III: CIL Instruction Set – Describes the Common Intermediate Language (CIL) instruction set.
- Partition IV: Profiles and Libraries – Provides an overview of the CLI Libraries, and a specification of their factoring into Profiles and Libraries. A companion file, CLILibrary.xml, considered to be part of this Partition, but distributed in XML format, provides details of each class, value type, and interface in the CLI Libraries.
- Partition V: Debug Interchange Format – Describes a standard way to interchange debugging information between CLI producers and consumers.
- Partition VI: Annexes – Contains some sample programs written in CIL Assembly Language (ILAsm), information about a particular implementation of an assembler, a machine-readable description of the CIL instruction set which can be used to derive parts of the grammar used by this assembler as well as other tools that manipulate CIL, a set of guidelines used in the design of the libraries of Partition IV, and portability considerations.

Figure 1-1. *Excerpt of web page with information about the ECMA-335 standard specification*

More objectively, the CLI is an open specification that describes executable code and an execution environment that enables multiple high-level languages to be used on different architectural platforms without being rewritten.

This execution environment must follow the architectural infrastructure described by the following:

- *Base Class Library (BCL)*: Foundational library defined by and part of the CLI standard specification. It is implemented by .NET Framework, .NET Core, .NET 5, and .NET 6 (early stages, available on Github.com), and is the main reason for the existence of the .NET standard.
- *Common Language Specification (CLS)*: Rules (restrictions and models) required for language interoperability. The detailed information on the CLS group is a subset of what is in the CTS, but the content is primarily for language designers and class library designers (frameworks). So, learning about CTS will offer a great base of knowledge for you and your team for when we start working with the rules in the CLS.
- *Common Type System (CTS)*: The CTS is a set of data types and operations that are shared by all languages that support the CTS, and learning about the CTS will offer a great base of knowledge to you and your team when we start working with the rules in the CLS.
- *Metadata*: The metadata describes the program structure, enabling languages and tools to work together. Detailed understanding of the metadata group is not a requisite for a component developer or application developer. Instead, detailed information about such is primarily for tool builders and compiler writers.
- *Virtual Execution Engine (VES)*: How code is executed (and how types are instantiated), interacts, and dies. More abstractly, it is also known as an execution engine or execution environment. This execution system is responsible for loading, instantiating, executing, and ensuring the cohesiveness of the interactions between the instances. In brief, it offers entire lifecycle support for the instance of