C# 13 and .NET 9

Guide to C# 13, .NET 9, data management, and deployment strategies

Jefferson S. Motta

www.bpbonline.com

First Edition 2025

ISBN: 978-93-65898-521

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

To the sisters: M. A. A.

***Prof. Cássio Saldanha**, My Mentor*

Natali Fonseca

About the Author

Jefferson S. Motta is currently working as a consultant with over 25 years of experience in desktop, web, and cloud development. His professional journey began in 1994 as a programming intern at Janssen Consultoria, where he helped develop an MRP II System for the Olvebra Group.

In 1996, he founded Menphis - Intelligent Systems, focusing on IT solutions for Olvebra and new clients. His most notable achievement was the development of Advocati.NET, a management system for law firms that has earned a respectable place in Brazilian legal technology. The system has undergone continuous improvement for over 20 years and is built using C# 13, ASP.NET Web Forms, and WinForms .NET 9.0, incorporating elements of artificial intelligence and big data.

Jefferson has been a pioneer in technological innovation, developing applications with artificial intelligence as early as 2012, followed by implementing bigdata solutions and automated control systems for physical equipment. In 2005, he created a management system for Medical Clinics.

He has made significant contributions to the tech community as an editor at C# Corner, a respected platform for technical articles. As a four-time MVP, his articles in English have garnered over 850,000 views, elevating him to approximately the 90th position in the global ranking.

He is also a graduate with a software engineering degree and has a diverse educational background, including digital marketing analysis (SENAC), digital security (Sisnema), and various technical courses.

About the Reviewer

Rositsa is an experienced engineering manager with 20 years of experience in the software industry, with a significant portion of that time leading teams that developed Telerik UI components for .NET MAUI, and Telerik UI for WPF and WinForms.

Her focus is on building high-performing teams that enjoy crafting software that they are proud of. She has also been a Microsoft MVP for .NET for the last 5 years (2020-2025).

Acknowledgement

Hey there! We want to thank everyone from the bottom of our hearts who helped make this book happen!

First, a big hug to our family and friends who stuck by us throughout this journey. Your love kept us going when things got tough, you know?

We owe a big debt of gratitude to the team at BPB Publications, who gave us expert publishing support. You guys were amazing in helping us navigate this crazy world of book publishing!

We can't forget the reviewers, technical experts, and editors who gave us that honest feedback to improve our manuscript. Your suggestions made all the difference!

Last but not least, a special thanks to you, the reader of our book. Your support means everything to us!

Thanks to everyone who pitched in to make this book a reality. You rock!

Preface

If you are new to programming or an experienced professional looking to enhance your skills, these pages will provide you with the information and tips you need to succeed in the .NET ecosystem, which is constantly evolving.

From Chapter 1's Introducing to C# 13 and .NET 9 to Chapter 16's Packaging and Deployment, each chapter is carefully organized to build on what you already know while presenting new ideas at a pace that you can easily handle.

In this book, you will discover the remarkable flexibility of the C# language and the .NET system. It will guide you through object-oriented programming, working with data using Entity Framework Core, utilizing LINQ to modify data in a concise and elegant manner, and leveraging ASP.NET Core and Blazor for modern web development.

I hope that this book not only helps you learn how to code but also inspires you to do so, providing you with the tools to solve complex problems in innovative ways and adding your own unique perspective and will to the world of software development.

Chapter 1: Introducing to C# 13 and .NET 9 - This chapter guid you through the key stages of configuring your development environment and acquiring fundamental knowledge of C# programming.This first chapter establishes a solid foundation for your coding journey.

Here, you will find the step-by-step installation procedure for the .NET **software development kit** (**SDK**)—an essential tool for any developer—as well as guidance on how to select an appropriate **integrated development environment** (**IDE**).

Chapter 2: C# Fundamentals - Originally part of the .NET framework, C# has proven to be a top option in software development for Microsoft because of its scalability, dependability, and simple syntax. The first part looks at the basic structural guidelines and conventions needed to write C# programs. From here, the trip delves into an in-depth study of data types, revealing the nuances of integers, floats, strings, and Booleans—the fundamental building blocks of data storage and processing.

Chapter 3: Harnessing the Code - In this chapter, we will use iterative techniques to explore iteration and learn how to effectively navigate data. We will clarify type conversion and type casting as we proceed, thereby equipping you with the tools to properly manage data. Furthermore, we will cover the principles of robust error management and exception handling, enabling you to produce consistent and reliable code.

Chapter 4: Functions In-depth - In this chapter, we will find ways to improve your code's modularity and adaptability. Techniques for exception handling are discussed to ensure robustness and to safeguard your code from unexpected errors. The need for unit testing in C# is also addressed since it helps to preserve code quality and dependability.

Chapter 5: Building Apps with OOPs- In this chapter,thorough investigations of key ideas such as encapsulation and abstraction will help to create strong and safe programs. Moving into advanced subjects, we explore inheritance and polymorphism, which enable dynamic behavior and code reuse. Along with static and instance members, you will investigate constructors, acquiring a clear knowledge of how objects are initialized and controlled.

Chapter 6: Mastering Interfaces and Inheriting Classes- This chapter explores polymorphism, which enables objects to take on multiple forms; examines inheritance as a foundational concept in object-oriented programming, facilitating code reuse; and discusses the value of interfaces as contracts for class behavior. Mastering these basic concepts will help you build a solid foundation to enhance your programming skills.

Chapter 7: .NET Toolbox - Forming one of the basic data types within the .NET framework, strings are crucial in many applications. They provide many features for text manipulation. Focusing on the string class and its closely related types—such as DateTime, TimeSpan, encoding, and regex—this chapter guides you on an in-depth exploration of the System namespace. We will demonstrate their practical applications in various situations and reveal the architectural ideas and design choices that underpin these classes

Chapter 8: Data in Motion- This chapter covers the fundamentals of file operations in .NET, including reading and writing text files, handling file and directory exceptions, and guaranteeing safe file operations. We also examine the key role of .NET streams in supporting data reading and writing across various sources, including files, networks, and memory. You will also learn about .NET serialization, a strong tool for transforming data structures or objects into digital forms for storage or transport.

Chapter 9: Data Handling with EF Core - This chapter explores **Entity Framework Core** (**EF Core**), a sophisticated **Object-Relational Mapping** (**ORM**) framework designed for .NET applications. EF Core simplifies processes and increases output by utilizing .NET objects instead of SQL queries, thereby enabling efficient data management.

Chapter 10: LINQ Unleashed- This chapter delves into the intricacies of **Language Integrated Query** (**LINQ**), a powerful C# feature that enables programmers to query and manipulate data seamlessly across multiple data sources. You will learn to use LINQ with various data types—including collections, databases, XML, and JSON—by means of

a coherent and expressive syntax. You will also investigate leveraging LINQ operators, lambda expressions, and deferred execution to create efficient designs and adaptable queries.

Chapter 11: ASP.NET Core the Future of Web Development- This chapter introduces a modern and flexible framework meant for creating web apps and services, ASP.NET Core. Among its cross-platform support, great performance, modular architecture, and strong security, the fundamental features and benefits of ASP.NET Core will be revealed. Through hands-on activities, you will acquire knowledge of building a simple web application, investigating its architecture, adding features, running tests, refining features, and efficiently deploying it.

Chapter 12: Building Powerful Websites with Razor- This chapter covers Razor Pages, a key feature of ASP.NET Core that utilizes a page-oriented approach to transform web development. Emphasizing obvious separation of concerns and effective data management, Razor Pages simplifies the process by combining C# and HTML into a single file. Using Razor syntax, Tag Helpers, layouts, and partials, you will learn how to create and control Razor Pages using the .NET Core framework to build dynamic and responsive online applications. Form management, user input processing, data validation, and fundamental **Create, Read, Update, Delete** (**CRUD**) activities using Razor Pages are all covered in the chapter.

Chapter 13: Website Development Using MVC Pattern- This chapter explores the **Model-View-Controller** (**MVC**) architecture within the ASP.NET Core framework, a powerful tool for creating modern web applications. You will learn to create views and controllers, set up MVC routing, use Entity Framework Core to handle data, apply strong form validation, and use ViewModels to simplify your development process.

Chapter 14: Mastering Web Services- This chapter provides a comprehensive overview of the various web service domains within the .NET environment, including an analysis of their potential as well as their challenges. Emphasizing strong security policies against current cyber threats, it unpacks the complexity of building and using web services from basic ideas like RESTful APIs in ASP.NET Core to advanced technologies like gRPC and GraphQL.

Chapter 15: Blazor for UI Development- This chapter explores Blazor, a state-of-the-art UI framework using C# for both client-side and server-side programming. Key subjects are thoroughly investigated in it, including Blazor component development, project organization, form handling and validation, JavaScript interoperability, CRUD operations

with Blazor and Entity Framework Core, differences between Server and WebAssembly models, routing, and application testing.

Chapter 16: Packaging and Deployment- This chapter explores the complexities of packaging and distributing .NET applications, including a thorough guide to components and libraries, NuGet's important role, and the process of publishing code for deployment. It highlights the importance of .NET Standard and .NET Core libraries, assembly versioning, the **Global Assembly Cache** (**GAC**), and strongly named assemblies.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/i244egu

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/C-Sharp-13-and-.NET-9.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Introduction to C# 13 and .NET 9

Introduction

Welcome to our journey into the world of C# (pronounced C-Sharp) and .NET (pronounced dot-Net) programming. This opening chapter aims to set the stage for our coding study, helping you navigate the initial, crucial steps of setting up our development environment and knowledge of C# programming. We start exploring the process of selecting an appropriate **integrated development environment** (**IDE**) and guide you through installing the .NET **software development kit** (**SDK**), an essential tool in our C# toolkit.

We will also understand the fundamental understanding of the .NET platform, a versatile framework central to C# programming. Moreover, we will delve into the diverse types of applications that can be crafted using C#, showcasing the extensive adaptability of this programming language.

Structure

This chapter covers the following topics:

- Introduction to C# and .NET
- Choosing the right tools for C# programming
- Understanding application types in C#
- Deploying cross-platform applications

- Role of integrated development environments
- .NET Framework vs. .NET Core
- Overview of C# and .NET ecosystem
- Installing and configuring Visual Studio
- Exploring Visual Studio Code for C# development

Objectives

In this chapter, we aim to provide you with a comprehensive understanding of C# and .NET programming. Throughout this section, you will gain familiarity with the C# language and the .NET platform, learning about their functionalities, advantages, and their place in the broader programming landscape. We will guide you through setting up your development environment, including the installation and configuration of essential tools like Visual Studio, empowering you to select the right tools for various C# programming tasks, thereby enhancing your productivity and efficiency. Additionally, you will explore the diverse range of applications that can be developed using C#, from desktop and web applications to mobile and cloud-based solutions. Another crucial aspect will be to understand the intricacies involved in deploying cross-platform applications, along with platform-specific configurations and deployment techniques. By comprehending the role and utility of IDEs in streamlining the software development process, you will be better equipped to make informed decisions.

Furthermore, you will gain insights into the key differences between the .NET Framework and .NET Core, aiding you in choosing the appropriate platform for your specific development needs. We will also provide an overview of the C# and .NET ecosystem, acquainting you with related frameworks, tools, and libraries such as **.NET Multi-platform App UI** (**.NET MAUI**), Unity, and ASP.NET. Finally, you will receive hands-on guidance for installing and configuring Visual Studio, preparing you to embark on your development journey. By the end of this chapter, you will have established a solid foundation in C# and .NET programming, empowering you to create and deploy your own applications confidently.

Introduction to C# and .NET

C# is a modern, in constant evolution, open-sourced, object-oriented programming language developed by *Microsoft*, envisioned as a part of their **.NET initiative**. The language was designed to offer a mix of the best features from languages like Java™ and C++ while incorporating unique elements that deliver power and development flexibility.

Much like Java™, C# is a statically typed, object-oriented language, emphasizing safety and simplicity in its syntax. Both languages share similarities like garbage collection, exceptions, and the extensive use of libraries for different functionalities. They also provide capabilities for threading and synchronization, which are vital for modern, multi-threaded software.

However, there are differences, too. One significant distinction is that while Java™ was built with a *write once, run anywhere* philosophy and is thus platform-independent, C# was primarily designed for Windows development. But with the introduction of .NET Core and its evolution into .NET 5 and beyond, the latter has become increasingly cross-platform, supporting Android, Linux, iOS, and macOS.

The .NET, or Microsoft .NET initiative, is a comprehensive software development framework introduced by Microsoft in the late 1990s and officially launched in 2002. It represented a strategic initiative to provide developers with a unified, cohesive environment for developing applications that could run on Windows platforms and beyond.

At the core of the .NET initiative was the idea of language interoperability, which would allow different high-level languages to communicate seamlessly with one another. Using the **Common Language Runtime** (**CLR**) and the .NET **Framework Class Library** (**FCL**), developers could write applications in various .NET compatible languages, such as C#, Visual Basic .NET, and F#.

Tip: Interoperability refers to the ability of different systems, devices, or applications to connect and communicate in a coordinated way without any effort from the end user.

Another significant aspect of the .NET initiative was the concept of managed code. Managed code is executed by the .NET CLR, which offers services like garbage collection, exception handling, and security, simplifying application development by automating routine or complex tasks.

The .NET Framework also introduced the idea of **software as a service** (**SaaS**), highlighting the web services aspect of development. It is a model where the software is provided as a service over the Internet, which can be used by other software applications, irrespective of platform or language.

The .NET Framework, from 1.0 to 4.8, is closely tied to the Windows operating system, forming the backbone of many applications from Microsoft and other enterprises. Microsoft continues to maintain .NET Framework 4.8 despite the arrival of its new brother, .NET Core.

Hence, the .NET initiative represented a significant shift in the approach to software development by Microsoft, emphasizing interoperability, service-oriented architecture, and managed code, making application development more streamlined and efficient.

C# programming and the .NET platform form a potent combination. They have continued to evolve from their beginnings to the present day with a new **long-term support** (**LTS**) version every two years, offering programmers a robust, versatile, and efficient environment for creating powerful, modern applications. The LTS means it gets updated and receives security fixes for three years after the official launch. A non-LTS version is released every two years, with one year of support. So, we have one new version of .NET every year. Despite sharing roots and similarities with Java™, the pair has charted its

unique path in the programming world. With their recent cross-platform capabilities, they are showing no signs of slowing down.

What is new in C# 13

In this chapter, we will see the innovative elements that C# 13 introduces.

Let us briefly describe how it has grown since its first release. Generics were introduced with C# 2.0, allowing us to design classes, interfaces, and functions with placeholders for the data type they store or use. It enabled more type-safe collections and increased performance by eliminating boxing and unboxing requirements. Iterators and partial types were also implemented, improving how we wrote code and organized our projects. C# 3.0 introduced many new improvements, the most notable of which was **Language-Integrated Query** (**LINQ**), which provided an SQL-like syntax for querying data sets directly from C#.

C# 4.0 improved interoperability with other languages and systems, making it more dynamic and versatile. Key features include the dynamic type, which allows operations to avoid compile-time type verification and defer binding until runtime. It made coding with COM objects, dynamic programming languages, and reflection easier and less error-prone.

C# 5.0 added the **`async`** and **`await`** keywords, significantly simplifying asynchronous programming. These keywords enabled developers to construct asynchronous code that is easy to read and maintain while approximating synchronous code in structure. C# 6.0 and 7.0 included syntactic sugar and new capabilities such as expression-bodied members, pattern matching, and tuples, reducing code complexity and boosting readability and conciseness. C# 8.0 included nullable reference types and more powerful pattern-matching capabilities, which improved the language's type safety and control flow. C# 9.0 and 10.0 were designed to make the language more concise and expressive by introducing records for immutable data structures, init-only properties, and pattern-matching improvements. C# 11, which builds on these advancements and becomes the language to the language of the year 2023.

C# 13 introduces several new features and changes to improve the programming experience, streamline syntax, and give additional functionality.

The primary constructors' feature, which has been expanded to be usable in any class and struct, is a significant addition. Unlike prior versions, which restricted primary constructors to record types, developers can now use this feature more liberally, allowing for more straightforward and concise class and struct definitions.

Collection expressions are another novel feature in C# 13. This feature reveals a more concise syntax that facilitates the generation of common collection values, encouraging code brevity and enhancing readability. This improved syntax is intended to give developers a more efficient way of working with collections.

C# 13 improves lambda expressions with default lambda parameters. This enables developers to assign default values to lambda expression parameters. This enhancement offers flexibility and simplicity to the definition and use of lambda expressions in diverse circumstances.

Furthermore, an alias of any type is a new feature introduced with Visual Studio 17.6 Preview 3. This feature increases the language's adaptability by enabling more robust and flexible type aliasing.

Inline arrays were also implemented to the C# 13 runtime as part of the previous .NET 9 preview. This enhancement enhances the language's array handling features, resulting in a more integrated and fluid array utilization experience.

Finally, interceptors are introduced as an experimental feature, providing new code generation and modification opportunities. Interceptors allow generators to reroute code, allowing for context-specific optimization and improving generated code functionality and efficiency. This broadens the scope of code modification and optimization options in C# programming.

The primary constructors extend to all classes and structs, letting us add parameters directly to class declarations for property initialization and method/property accessor usage.

Lambda expressions now support default parameter values, streamlining methods, and local function argument handling. The using alias directive no longer limits to named types. Creating semantic aliases for tuples, arrays, pointers, and other types is now possible.

Choosing the right tools for C# programming

Choosing the right tools can drastically impact our efficiency, code quality, and work. Each tool serves a specific purpose and choosing which tool to use can mean the difference between a smooth workflow and a choppy one. Let us have a detailed look at the essential pillars of C# programming tools.

Integrated development environment

The IDE is the platform where we write our code. Microsoft's Visual Studio is an excellent IDE for C# programming. It is a fully featured tool that offers an integrated debugger, a code editor with IntelliSense (an auto-completion tool to increase speed and accuracy), support for various project configurations, and much more. However, it is possible to use a lighter solution using Visual Studio Code. It is a more streamlined editor that still supports a wide range of languages and has robust community support for extensions.