I POCKET GUIDES

Task Programming in C# and .NET

Modern Day Foundation for Asynchronous Programming

Vaskaran Sarcar

Apress®

Apress Pocket Guides

Apress Pocket Guides present concise summaries of cutting-edge developments and working practices throughout the tech industry. Shorter in length, books in this series aims to deliver quick-to-read guides that are easy to absorb, perfect for the time-poor professional.

This series covers the full spectrum of topics relevant to the modern industry, from security, AI, machine learning, cloud computing, web development, product design, to programming techniques and business topics too.

Typical topics might include:

- A concise guide to a particular topic, method, function or framework
- Professional best practices and industry trends
- A snapshot of a hot or emerging topic
- Industry case studies
- Concise presentations of core concepts suited for students and those interested in entering the tech industry
- Short reference guides outlining 'need-to-know' concepts and practices.

More information about this series at https://link.springer.com/bookseries/17385.

Task Programming in C# and .NET

Modern Day Foundation for Asynchronous Programming

Vaskaran Sarcar

Apress®

Task Programming in C# and .NET: Modern Day Foundation for Asynchronous Programming

Vaskaran Sarcar
Kolkata, West Bengal, India

ISBN-13 (pbk): 979-8-8688-1278-1 ISBN-13 (electronic): 979-8-8688-1279-8
https://doi.org/10.1007/979-8-8688-1279-8

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Coordinating Editor: Kripa Joseph

Cover designed by eStudioCalamar

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail 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 http://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 (https://github.com/Apress/Task-Programming-in-C-and-.NET). For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

To my seniors, colleagues, and teachers including C# and .NET community members who directly and indirectly helped me to write better code. In fact, I am still learning from them.

Table of Contents

About the Author

Vaskaran Sarcar obtained his Master of Engineering degree in Software Engineering from Jadavpur University, Kolkata (India), and an MCA from Vidyasagar University, Midnapore (India). He was a National Gate Scholar (2007–2009) and has over 12 years of experience in education and the IT industry. He devoted his early years (2005–2007) to the teaching profession at various engineering colleges, and later, he joined HP India PPS R&D Hub in Bangalore. He worked there until August 2019 and became a Senior Software Engineer and Team Lead. After working for more than ten years at HP, he decided to follow his passion completely. He is now an independent full-time author.

You can refer to the link amazon.com/author/vaskaran_sarcar (or Appendix B) to find all his books. You can also find him on LinkedIn at `https://www.linkedin.com/in/vaskaransarcar`.

About the Technical Reviewer

Shekhar Kumar Maravi is a software architect – design and development, whose main interests are programming languages, Linux system programming, Linux kernel, algorithms, and data structures. He obtained his master's degree in Computer Science and Engineering from Indian Institute of Technology Bombay. After graduation, he joined Hewlett Packard's R&D Hub in India to work on printer firmware. Currently, he is a Product and Solution Development Team Lead for automated pathology lab diagnostic devices at Siemens Healthcare R&D division. He can be reached by email at shekhar.maravi@gmail.com or via LinkedIn at https://www.linkedin.com/in/shekharmaravi.

Acknowledgments

At first, I thank the Almighty. I sincerely believe that with His blessings only, I could complete this book. I extend my deepest gratitude and thanks to the following people:

Shekhar: Whenever I was in need, he provided support. He answered all my queries over phone calls, WhatsApp, and emails. Thank you one more time.

Smriti, Kripa, Celestin, and the Apress team: I sincerely thank each of you for giving me another opportunity to work with you and Apress.

Nirmal and the Production team: Thanks to each of you for your exceptional support in beautifying my work. Your efforts are extraordinary.

Finally, I thank those people from different online communities (particularly, the C# developer community, .NET developer community, and Stack Overflow community) who share their knowledge in various forms. In fact, I thank everyone who directly or indirectly contributed to this work.

Introduction

With the availability of multicore computers, asynchronous programming and parallel programming are becoming increasingly important. Why not? It is essential for building highly responsive software.

This is why playing with threads in a multithreaded environment is inevitable. Undoubtedly, it is hard, but in earlier days, it was harder. To simplify the overall coding experience, starting from the .NET Framework 4.0, Microsoft introduced Task Parallel Library (TPL) which was based on the concept of tasks. Later, in C#5, we saw the revolutionary introduction of the `async` and `await` keywords. Using them, we started passing the heavy work(s) to the compiler. However, you need to remember that a typical `async` method normally returns a task (in programming terms, a `Task` or a `Task<TResult>`). So, there is no wonder that task programming became the modern-day foundation for asynchronous programming. In addition, the patterns used earlier to deal with asynchronous and parallel programming are not recommended now.

This is why I decided to write a pocketbook series on asynchronous and parallel programming. This pocketbook series will try to simplify the concept using the modern C# features and libraries that Microsoft recommends. *Task Programming in C# and .NET: Modern Day Foundation for Asynchronous Programming* is the first book in this series. It focuses on task programming without using the `async` and `await` keywords.

How Is the Book Organized?

This book helps you to understand task programming using six chapters with many Q&A sessions and exercises. To give you an idea about the organization of the chapters and the contents of this book, let me summarize the following points:

- Chapter 1 introduces asynchronous programming with useful scenarios. It also provides an overview of Task Parallel Library (TPL) and discusses tasks. These are the foundation for the next chapters.
- Chapter 2 discusses the creation and execution of tasks. Once you execute a task, most probably, you'd like to see the result of the execution. It means that you need to wait for the task to finish its execution. Implementing a correct waiting mechanism in a multithreaded environment is extremely important. This chapter discusses different types of waiting mechanisms as well.
- Chapter 3 talks about task continuation scenarios. It also discusses nested tasks.
- Exception handling is an essential part of programming. Chapter 4 covers this topic and shows you different ways of exception handling mechanisms in task programming.
- Normally, we do not like to wait for long-running tasks. Also, if you identify a mistake early, you may not continue running the tasks. So, task cancellations are also common when you play with tasks. Chapter 5 covers this topic.

- In Chapter 6, you'll find some extra materials that were not discussed in the previous chapters.
- You can enjoy learning when you analyze case studies, ask questions (about the doubts), and do some exercises. So, throughout this book, you will see interesting program segments, "Q&A Sessions", and exercises. By analyzing these Q&As and doing the exercises, you can verify your progress. As said before, these are presented to make your future learning easier and enjoyable, but most importantly, they make you confident as a developer.
- Each question in these Q&A sessions is marked with **<chapter_no>.<Question_no>**. For example, Q2.1 means question number 1 from Chapter 2. At the end of the chapter, you'll see some exercises. You can use them to evaluate your progress. Each question in these exercises is marked with **E<chapter_no>.<Question_no>**. For example, E5.3 means exercise number 3 from Chapter 5.
- You can download all the source codes of the book from the publisher's website (`https://github.com/Apress/Task-Programming-in-C-and-.NET`).

Prerequisite Knowledge

I expect you to be very much familiar with C#. In fact, knowing about some of the advanced concepts like delegates and lambda expressions can accelerate your learning. I assume that you know how to compile or run a C# application in Visual Studio. This book does not invest time in easily available topics, such as how to install Visual Studio on your system

or how to write a “Hello World” program in C#. In short, the target readers of this book are those who want to make the most out of C# by harnessing the power of both object-oriented programming (OOP) and functional programming (FP).

Who This Book Is For

You can pick the book if the answer is “yes” to the following questions:

- Are you familiar with .NET, C#, and basic object-oriented concepts like polymorphism, inheritance, abstraction, and encapsulation?
- Are you familiar with some of the advanced concepts in C# such as delegates, lambda expressions, and generics?
- Do you know how to set up your coding environment?
- Are you interested to know how the modern-day constructs of C# can help you in asynchronous and parallel programming?

Probably you shouldn’t pick this book if the answer is "yes" to any of the following questions:

- Are you looking for a C# tutorial or reference book?
- Are you not ready to experiment with asynchronous programming using C# and .NET?
- “I do not like Windows, Visual Studio, and/or .NET. I want to learn asynchronous and parallel programming without them.” Is this statement true for you?

Useful Software

These are the important software/tools that I used for this book:

- All the programs were tested with C# 13 and .NET 9. In this context, it is useful to know that nowadays the C# language version is automatically selected based on your project's target framework(s) so that you can always get the highest compatible version by default. In the latest versions, Visual Studio doesn't support the UI to change the value, but you can change it by editing the `csproj` file. If you are interested more in the C# language versioning, you can follow the link https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/configure-language-version.
- During the development of this book, software updates kept coming, and I also kept updating. When I finished my initial draft, I had the latest edition of Microsoft Visual Studio Community 2022 (64-bit) – Preview Version 17.12.0 Preview 3.0. When I submitted the final draft, I had Microsoft Visual Studio Community 2022 (64-bit)-17.12.4.
- The good news for you is that this community edition is free of cost. If you do not use the Windows operating system, you can also use Visual Studio Code which is also a source code editor developed by Microsoft that runs on Windows or macOS and Linux operating systems. This multiplatform IDE is also free. However, I recommend that you check the license and privacy statement as well. It is because this statement may change in the future.