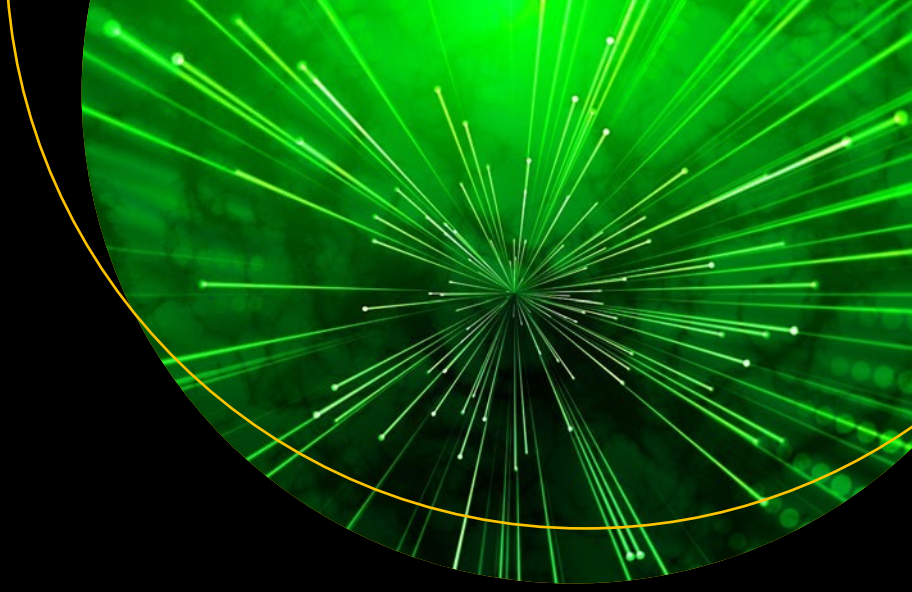

Exploring Blazor

Creating Server-side and Client-side Applications in .NET 7

Second Edition

Taurius Litvinavicius

Apress®

Exploring Blazor

Creating Server-side and Client-side Applications in .NET 7

Second Edition

Taurius Litvinavicius

Apress®

Exploring Blazor: Creating Server-side and Client-side Applications in .NET 7

Taurius Litvinavicius
Kaunas, Lithuania

ISBN-13 (pbk): 978-1-4842-8767-5 ISBN-13 (electronic): 978-1-4842-8768-2
https://doi.org/10.1007/978-1-4842-8768-2

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Smriti Srivastava
Development Editor: Laura Berendson
Coordinating Editor: Shrikant Vishwakarma
Copy Editor: Kim Wimpsett

Cover designed by eStudioCalamar

Cover image by Erik Witsoe on Unsplash (www.unsplash.com)

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 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). For more detailed information, please visit www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Author

Taurius Litvinavicius is a businessman and technology expert based in Lithuania who has worked with organizations in building and implementing various projects in software development, sales, and other fields of business. He currently works on modern financial applications and consults for companies on technology and cost-related issues. With most of his projects, he uses cutting-edge and cost-effective technologies, such as Blazor.

About the Technical Reviewer

Fabio Claudio Ferracchiati is a senior consultant and senior analyst/developer using Microsoft technologies. He works for BluArancio (www.bluarancio.com). He is a Microsoft Certified Solution Developer for .NET, a Microsoft Certified Application Developer for .NET, a Microsoft Certified Professional, and a prolific author and technical reviewer. Over the past ten years, he's written articles for Italian and international magazines and coauthored more than ten books on a variety of computer topics.

Introduction

For many years the web development community has been waiting for something new to escape the dreaded JavaScript monopoly. Finally, the time has arrived—first with the release of WebAssembly and now with the release of Blazor. This book will explore Blazor in depth, and with that, you will understand what role WebAssembly plays in this tool stack.

To start, you will learn what Blazor is, where it runs, and how to start using it. Blazor is convenient to code and also has tremendous business value. Although the technology is still relatively young, I have already managed and taken part in the development of a large-scale platform—`mashdrop.com`—and from that experience, can testify as to Blazor's efficiency and ease of use.

This book will focus on practicality and practice; you can expect lots of sample code and some exercises to complete. In fact, you will work through five exercises, covering all types of Blazor, and explore some use cases. I believe in experiential learning, which is why, from the early stages of the book, we will be exploring Blazor by looking at code samples and folder structures of projects. Since Blazor is not a stand-alone technology like a programming language, the best way to learn it is to interact with it, see what it looks like in the code, and uncover some similarities with technologies using the same programming language—in this case C#. You will see, you will do, and most importantly you will learn.

Source Code

All the source code used in this book is available for download at `https://github.com/apress/exploring-blazor-2e`.

CHAPTER 1

Introduction to Blazor

In this book, you will learn about Blazor, a modern framework for developing web applications using C#. You'll learn about all the features of Blazor, from the most basic to the more advanced. You will learn the fundamentals of Blazor syntax and project setup, as well as exciting modern features such as picking files and accessing them using C# in a web browser, accessing API data using JSON, and using many of the other latest features of Blazor. In addition, I will demonstrate what you can achieve in Blazor and provide a few tasks for you to practice yourself, along with the solutions I created for them.

Before you start, you need to know and prepare a few things. This is not an introductory book to C# or .NET Core development, so you should already have good knowledge of C# and be able to build applications with it. It does not matter if you develop back-end applications, Windows applications, or mobile applications; as long as you use C#, you will find something familiar in Blazor. If you haven't already, you'll need to install Visual Studio 2022 and make sure that you have the .NET 7 SDK installed on your computer.

What Is Blazor?

Blazor is a web UI framework that allows you to use C# and .NET Core on the front end. It allows you to develop your front-end logic in a couple of different ways using the C# programming language, which is something that you will explore later in this chapter.

T. Litvinavicius, *Exploring Blazor*, https://doi.org/10.1007/978-1-4842-8768-2_1

Technical aspects aside, think of it this way: in any standard web development project, you would need to have two people, one for the JavaScript and the other for the back end. Sometimes you also need a designer to work with HTML elements and CSS and do other design-related tasks. The Blazor technology will not remove any dependency for a designer, but it will surely remove the dependency on JavaScript. (However, JavaScript can still be used with the Blazor technology.)

Blazor uses the Razor syntax (C# mixed with HTML), which will be covered in Chapter 2, so any familiarity with the Razor syntax will give you an edge when developing. There are some differences, though, as you will see shortly. Most important, your C# code in Razor (the `.cshtml` file) will execute only when the page is loaded, but in Blazor (the `.razor` file) the code will execute on the loaded page on various events, such as `onclick`, `onchange`, and others.

Blazor uses WebSocket to communicate with the server as well as work on the server side, or it uses the WebAssembly technology, which allows for C# to be built on the client side. This is where the different types of Blazor technology come into play.

What Is WebAssembly?

WebAssembly is a technology that allows you to compile languages such as C++ or C# in the browser, thus allowing Blazor to exist. It first appeared as a minimum viable product in early 2017, and while the technology is still in its early years, it is being co-developed by companies such as Microsoft, Google, Apple, and others. The technology has the support of most major browsers (`https://webassembly.org/roadmap/`)—Edge, Chrome, Firefox, Opera, and Maxthon (MX)—and the equivalent mobile versions. With its growth, we can expect the support to be there for a long time. In general, Blazor simply sends a source code file to the browser, and WebAssembly compiles it into a binary file.

WebAssembly gives you a safe, sandboxed environment, so it appears similarly as running JavaScript. Nothing is accessible from outside the specific browser tab the user is using.

Blazor Types

The *server-side* type of Blazor will run all the logic on the server side, mainly using WebSockets to accomplish tasks (Figure 1-1). Although it does give you an ability to use C# to write the front end, this may not be the most efficient option. You will eliminate the need for API calls with this option, as you will simply inject your libraries directly to the front-end part.

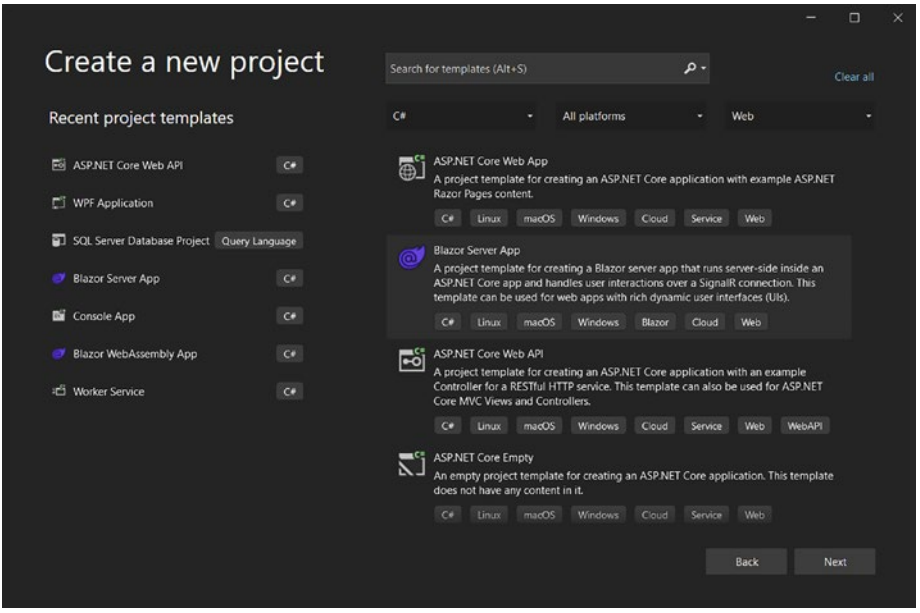

Figure 1-1. *Blazor Server App template in Visual Studio 2022*

The *client* type of Blazor runs completely on the client side, on the browser (Figure 1-2). You will have your pages on the server, but other than that, the client side handles everything. So, this is great for presentation

websites or websites that provide calculators and other such services. If you need database interactions or if you already have APIs and class libraries, this will not be your choice.

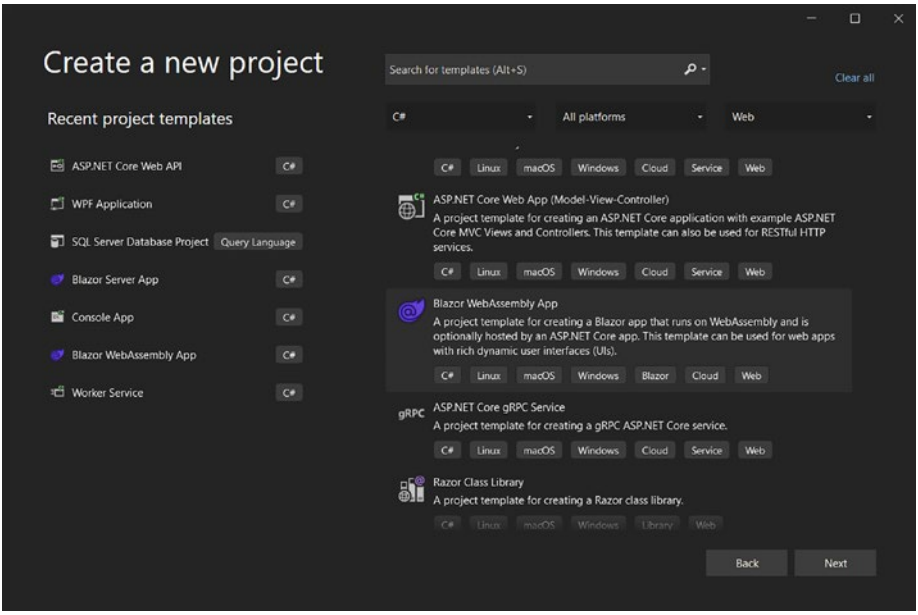

Figure 1-2. *Blazor WebAssembly App template in Visual Studio 2022*

There are also other possible variations of these two types. One of them is Blazor hosted; this project is client-side Blazor (Blazor WebAssembly) interconnected with the web API project. The client Blazor and API program run separately, but development-wise they will be able to share a common code library, mostly for data models. There is also a progressive web application (PWA) option, which allows the Blazor client to run offline. Finally, you can add API capabilities (controllers and such) to a Blazor server project.

Along with these main projects, you will also find a Razor Class Library project (Figure 1-3). This allows you to create Blazor components with all the Blazor features in a code library and if needed publish that to NuGet.

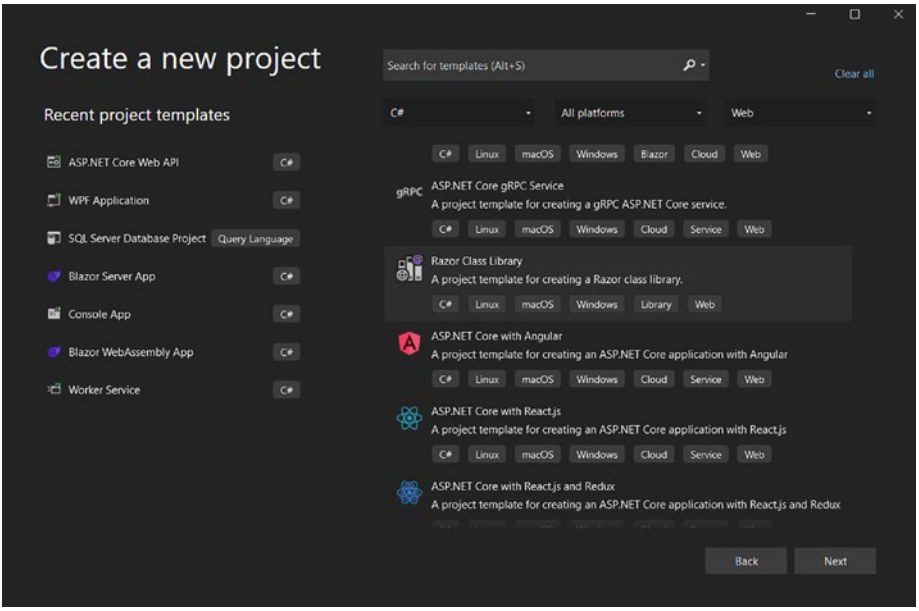

Figure 1-3. *Razor Class Library template in Visual Studio 2022*

Summary

There is no best type of Blazor; as you have seen throughout this chapter, every option has its own use case. Everything depends on what your project needs right now and, more important, what it will need in the future. If you are not sure, simply go with the client-side version, as it will be the most diverse option. In the next chapter, we will dive deeper into Blazor and explore the syntax and some other topics. You will see that while the structure may be different, for the most part, coding happens in the same way for all types of Blazor.