

Exploring Blazor

Creating Hosted, Server-side, and
Client-side Applications with C#

Taurius Litvinavicius

Apress®

Exploring Blazor

Creating Hosted, Server-side,
and Client-side Applications
with C#

Taurius Litvinavicius

Apress®

Exploring Blazor: Creating Hosted, Server-side, and Client-side Applications with C#

Taurius Litvinavicius
Jonava, Lithuania

ISBN-13 (pbk): 978-1-4842-5445-5
<https://doi.org/10.1007/978-1-4842-5446-2>

ISBN-13 (electronic): 978-1-4842-5446-2

Copyright © 2019 by Taurius Litvinavicius

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

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

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. 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 rights@apress.com, or visit <http://www.apress.com/rights-permissions>.

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 via the book's product page, located at www.apress.com/978-1-4842-5445-5. For more detailed information, please visit <http://www.apress.com/source-code>.

Printed on acid-free paper

Table of Contents

About the Author	vii
About the Technical Reviewer	ix
Introduction	xi
Chapter 1: Introduction.....	1
What is Blazor?	1
What is WebAssembly?	2
Blazor Types	3
Blazor Server-side	4
Blazor Client-side	4
Blazor Hosted	5
Summary.....	6
Chapter 2: Razor Syntax and Basics of Blazor	7
Differences between Razor and Blazor	7
Syntax	8
Comments	8
Sections.....	9
Blazor Binds.....	11
Bind to Element	12
code.....	13
Page Events.....	16
Summary.....	17

TABLE OF CONTENTS

Chapter 3: Blazor server-side	19
Default template overview	19
Startup	19
Injections	23
Navigation	26
Pages	27
Components	29
Parameters	31
Finished example project	33
Summary	42
Chapter 4: Blazor Client-side	43
Default Template Overview	43
Program and Startup	43
Clean Up the Template	47
Navigation	49
Components	53
Using Key to Preserve Components	56
Example	58
Summary	65
Chapter 5: Blazor hosted	67
Default template overview	67
General structure	67
Clean up the template	72
Navigation	75
API calls	77
JSONfull way	81

HTTP client manipulations.....	84
Example.....	87
Summary.....	111
Chapter 6: General Blazor.....	113
Interacting with JavaScript.....	113
Execute JavaScript Function	114
UI Events.....	116
UI Arguments	117
Local Storage	118
Where to Store?.....	119
Store Text.....	120
Store Other Types	122
Pick and Save Files.....	125
Pick File	129
Save File.....	130
Summary.....	131
Chapter 7: Practice Tasks for Server-side	133
Task 1.....	133
Description	133
Resources.....	134
Solution	135
Task 2.....	143
Description	143
Solution	144
Summary.....	151

TABLE OF CONTENTS

Chapter 8: Practice Tasks for Client-side153

Task 1 153

 Description 153

 Solution 156

Task 2..... 169

 Description 170

 Solution 170

Summary..... 178

Chapter 9: Practice Task for Blazor Hosted179

Task 1 179

 Description 180

 Resources..... 180

 Solution 184

Summary..... 193

Index.....195

About the Author



Taurius Litvinavicius is a businessman and technology expert based in Lithuania who has worked with various organizations in building and implementing various projects in software development, sales, and other fields of business. He works on a platform called MashDrop, which is a modern way to monetize the influence of an influencer. As with most of his projects, this one uses cutting-edge technologies such as Blazor. He is responsible for technological improvements, development of new features, and general management.

Taurius is also the director at the Conficiens solutio consulting agency, where he supervises development and maintenance of various projects and activities.

About the Technical Reviewer



Carsten Thomsen is a back-end developer primarily, but working 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 the tools he works with. Being 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 great skills researching and presenting new material.

Introduction

For many years the web development community has been waiting for something new, something to escape that dreaded JavaScript monopoly. Finally, the prayers have been answered – first with the release of WebAssembly and now with the release of Blazor. This book will explore Blazor in its full depth, and alongside that, you will understand what role WebAssembly plays in this whole arrangement. In fact, this is where we will begin; we will learn what Blazor is, where it runs, and how to start using it. Being a businessman with software development skills, the author has a unique view toward technologies and may base his predictions of the future for the technology not only on it being convenient to code but also on having tremendous business value. Although the technology is still young, the author has already managed and taken part in the development of a large-scale platform – mashdrop.com – and from that experience can tell you firsthand about the ease of use and efficiency of using Blazor for the project.

The book will focus on practicality and practice; therefore, you can expect lots of sample code and some exercises to complete. In fact, we will have five exercises, covering all types of Blazor, and with that, we will explore some use cases. The author believes in experiential learning; that is why, from the early stages of the book, we will be exploring Blazor by looking at code samples or folder structures of projects. Since Blazor is not a stand-alone technology, such as 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.

CHAPTER 1

Introduction

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 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. You need to install some software on your system, starting with Visual Studio 2019, followed by the latest version of .NET Core 3.0.

What is Blazor?

Blazor is a web UI framework allowing 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, and that is something that we will explore later in this chapter.

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 the next chapter, 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 importantly, Razor only happens once and Blazor will happen over and over again, meaning that your C# part in Razor (.cshtml file) will only execute when the page is loaded, but in Blazor (.razor file) the code will execute on the loaded page on various events, such as onclick, onchange, and others.

It 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 like 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 like Microsoft, Google, Apple, and others. The technology already has the support of all major browsers (<https://webassembly.org/roadmap/>), and with its growth, we can expect the support to be there for a very long time. In general, Blazor simply sends a source code file to the browser and WebAssembly compiles it into a binary file. The technology is available in all major browsers – Edge, Chrome, Firefox, Opera, and Maxthon (MX) – and the equivalent mobile versions.

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

Server-side (see Figure 1-1) Blazor will run all the logic, mainly using WebSocket to accomplish the task. While it does give you an ability to use C# for writing front-end code, this may not be the most efficient option. You eliminate the need for API calls with this option, as you will simply inject your libraries directly into the front-end part.

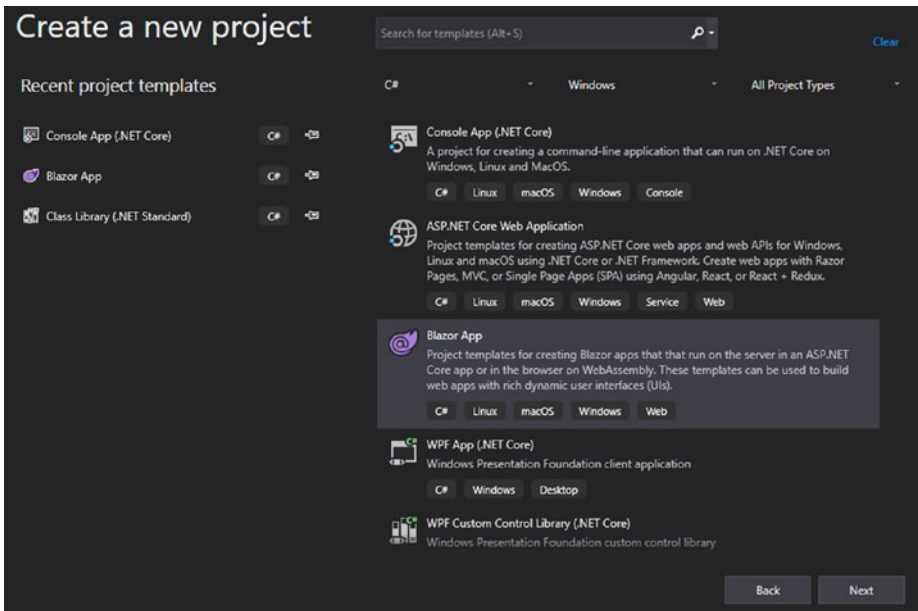


Figure 1-1. Blazor templates

All three types of Blazor have different templates in Visual Studio, and you should always use them for your Blazor projects no matter which type you choose. As shown in Figure 1-1, you will need to choose Blazor App project type and then choose the type of Blazor after you have picked your project location.

Blazor Server-side

While the server-side may come across as a convenience, you should still go with the client Blazor, that is, Blazor running in a browser. Server-side will use server resources, while the client will save you resources or at the very least will not waste them.

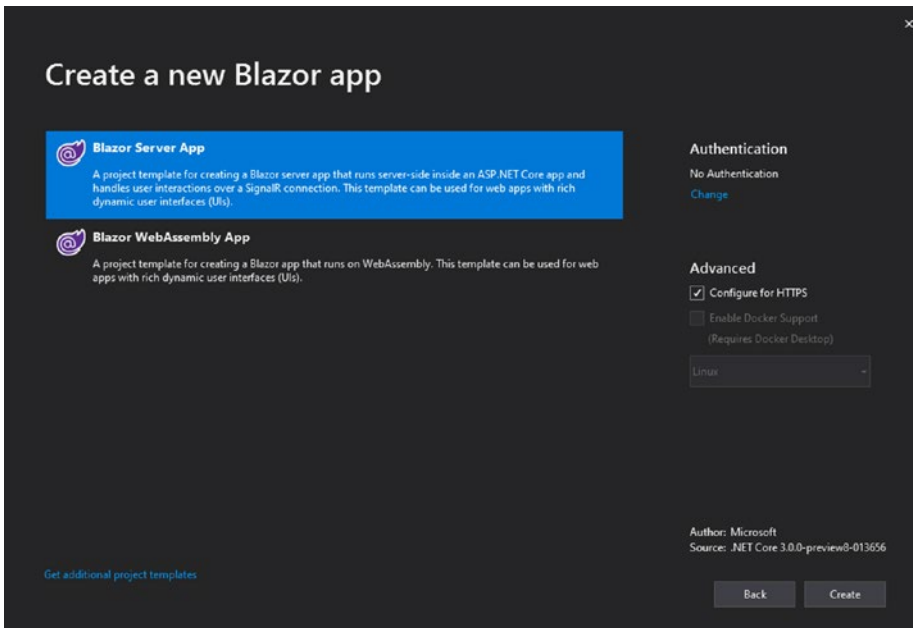


Figure 1-2. Server-side Blazor template selection

Once you get to picking the Blazor type in Visual Studio, you need to select “Blazor Server App” as shown in Figure 1-2.

Blazor Client-side

Client-side (see Figure 1-2) Blazor runs entirely on client-side in the browser. Your pages reside on the server, but it is all for client-side to handle. This is good for a presentation web site or web sites that

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

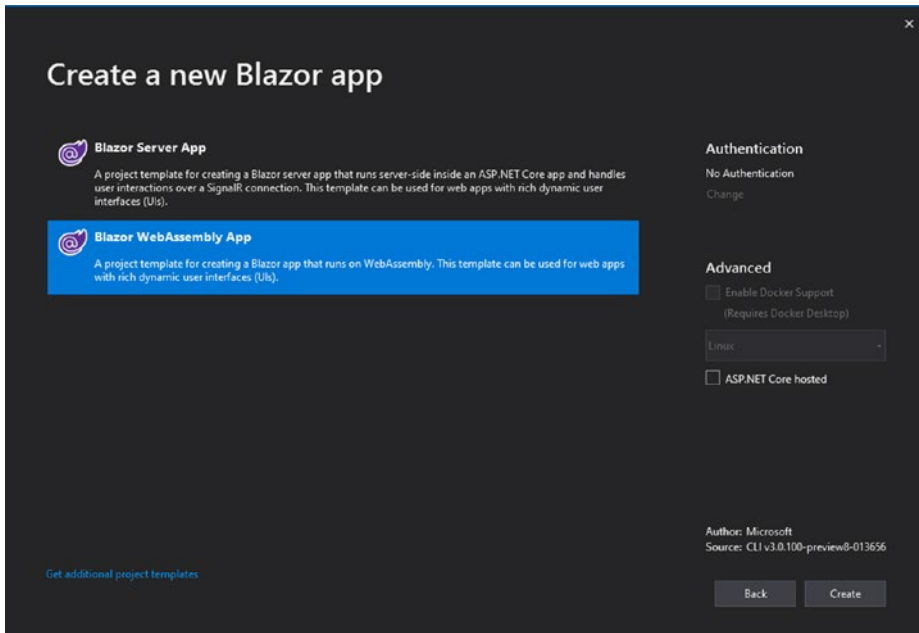


Figure 1-3. Client-side Blazor template selection

For the client-side project, you need to pick the template “Blazor WebAssembly App”. With that, the checkbox on the right side “ASP.NET Core hosted” needs to be unchecked.

Blazor Hosted

Blazor hosted (see Figure 1-3), this is probably the best type to go with as your logic will run on the browser saving those precious server resources. Basically, there are two parts, the client Blazor project and an API project. They are connected in a unique way, so you will not need to treat them as separate projects.

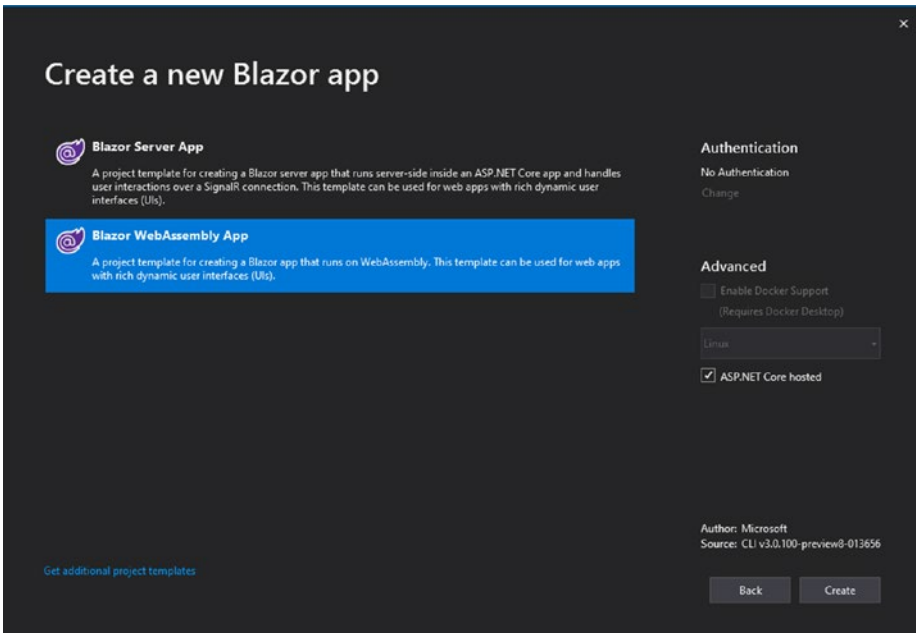


Figure 1-4. Blazor hosted template selection

If you wish to go with our final type of Blazor, you will need to select the template “Blazor WebAssembly App” (see Figure 1-4) just like you did for the client-side, but in this case you do need to check the checkbox on the right side “ASP.NET Core hosted”.

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 importantly what it will need in the future. If you are not, simply go with the client-side, 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 things. 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.

CHAPTER 2

Razor Syntax and Basics of Blazor

This chapter will get you started with Blazor, as mentioned in the last chapter – all three types of Blazor have a lot in common and this is what this chapter is all about. Before we can go any further, we will need to look at the syntax and see how it works. Then we will get to the essentials of Blazor, such as bindings and method execution; all that will be used later in the book.

In this chapter, you will learn

- Syntax
- Element and variable bindings
- Method executions
- Use of general page events

Differences between Razor and Blazor

Simplistically speaking, the difference is that Razor will happen once on “page launch,” while the Blazor will work all the time. The loops and logic statements will get re-evaluated in Blazor, while with Razor it will only happen once.

Syntax

As mentioned previously, if you know the Razor syntax, you will know Blazor syntax. However, if you do not know the Razor syntax, this is the part for you. Blazor syntax goes into a markup file named `.razor`, which contains HTML, as well as C# code.

Comments

Even though we use HTML syntax in a Blazor file, we do not use html comments. Instead we use Razor syntax for that and get beautiful and efficient commenting system, with no comments left on a generated page.

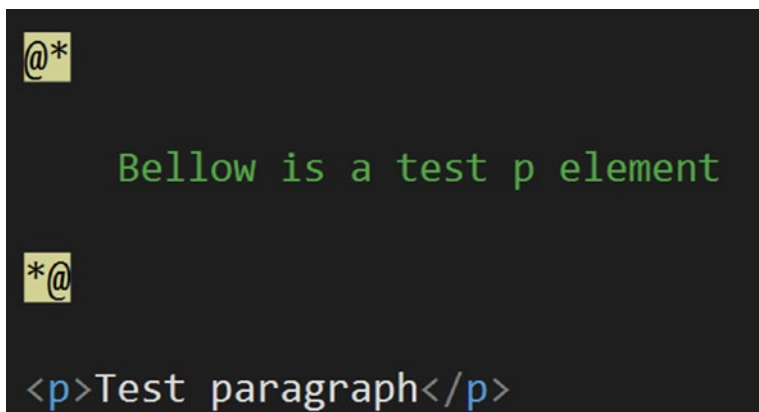
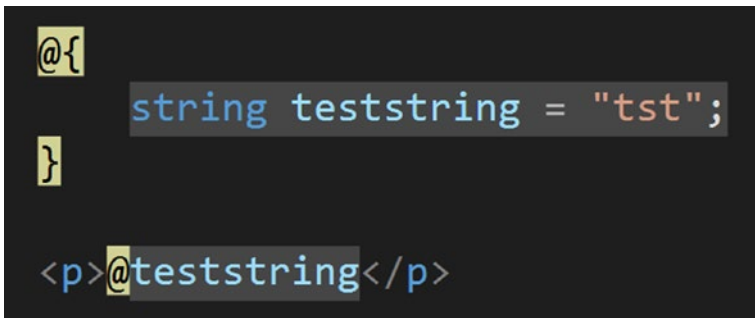


Figure 2-1. *Razor/Blazor comment syntax*

As shown in Figure 2-1, you simply start a comment section with `@*` and then end with `*@`. You can also use the standard HTML comments, but using the Razor/Blazor syntax will appear clearer on the code. The Razor/Blazor comment syntax characters get highlighted and the actual comment is displayed in a green font. The comments are compiled into the code; thus, they will not appear in the developer tools in the browser.

Sections

Razor syntax is basically C# and html code in one file, and while they do interact, you still need some things to be clearer than the others and you need some higher contrast between the two languages. That is where all the different sections come in; as you will see, they are C# dominant and they are used to highlight the C# parts of the code.



```
@{  
    string teststring = "tst";  
}  
  
<p>@teststring</p>
```

Figure 2-2. *Basic Blazor Sections*

In Figure 2-2, a variable is being declared and then it is displayed directly in a paragraph using C# code. So, for a single variable and construction of classes, you can simply use the @ sign and write everything on a single line, but if you want to do more than one line for a variable declaration, you need to create a section using @{ ... }.

At this point, this may look very simple, so let's dive into a few more examples:

```
@{
    int testint = 0;
}

@if (testint != 0)
{
    <p>Is not equal to zero</p>
}
```

Figure 2-3. *if statement syntax*

In Figure 2-3, the `testint` variable is declared and set to the value 0, followed by an if statement checking if the value of `testint` is not 0. Since the statement criteria is not satisfied, whatever is inside the if statement is not displayed. If the `testint` variable is set to any other value than 0, say 1, the HTML paragraph tag and value would be displayed. The C# code in the `@{ }` section is highlighted, and it requires no `@` sign for each line. The if statement part starts with `@` sign and creates a section similar to the previous example. This means the HTML code in the if statement section is not highlighted in any way.

```
@for (int i = 0; i < 5; i++)
{
    <p>@i</p>
}
```

Figure 2-4. *<Caption>syntax coloring*

In Figure 2-4, a for loop has been created, looping five times. Each loop creates a new paragraph tag containing the value for the current iteration, *i*. The for loop part is highlighted in a slight shade of gray, while the @ signs are highlighted in a yellow color. The HTML part inside the loop is not highlighted but the C# code is; that is how you can tell the difference between HTML markup and C# code.

Listing 2-1. Code section

```
<p>test</p>

@code {
    int a;
    double b = 2.5;

    void testmethod() {

    }
}
```

Finally, there's the code section (see Listing 2-1), where all the methods should be declared. The binding variables should also be added to the code section, which we will explain in a later chapter (Blazor binds).

Blazor Binds

Blazor allows you to bind an HTML input value to a variable and vice versa. Therefore, for the most part, we can call all bindings two-way. If you bind a text box (input type text) to a string variable, the displayed value will be the value of that string. Different elements will work differently and there are many use cases for this.