



Blazor WebAssembly by example : a project-based guide to building web apps with .NET, Blazor WebAssembly, and C# /

Wright, Toi B.,
author

Monografía

Discover blueprints that explore various components of Blazor, C#, and .NET to help you build web apps without learning JavaScript. Explore complete, easy-to-follow web projects using Blazor. Build projects such as a weather app, expense tracker, and Kanban board with real-world applications. Understand and work with Blazor WebAssembly effectively without spending too much time focusing on the theory. Book Description: Blazor WebAssembly makes it possible to run C# code on the browser instead of having to use JavaScript, and does not rely on plugins or add-ons. The only technical requirement for using Blazor WebAssembly is a browser that supports WebAssembly, which, as of today, all modern browsers do. Blazor WebAssembly by Example is a project-based guide for learning how to build single-page web applications using the Blazor WebAssembly framework. This book emphasizes the practical over the theoretical by providing detailed step-by-step instructions for each project. You'll start by building simple standalone web applications and progress to developing more advanced hosted web applications with SQL Server backends. Each project covers a different aspect of the Blazor WebAssembly ecosystem, such as Razor components, JavaScript interop, event handling, application state, and dependency injection. The book is designed in such a way that you can complete the projects in any order. By the end of this book, you will have experience building a wide variety of single-page web applications with .NET, Blazor WebAssembly, and C#. What you will learn: Discover the power of the C# language for both server-side and client-side web development. Use the Blazor WebAssembly App project template to build your first Blazor WebAssembly application. Use templated components and the Razor class library to build and share a modal dialog box. Understand how to use JavaScript with Blazor WebAssembly. Build a progressive web app (PWA) to enable native app-like performance and speed. Understand dependency injection (DI) in .NET to build a shopping cart app. Get to grips with .NET Web APIs by building a task manager app. Who this book is for: This book is for .NET web developers who are tired of constantly learning new JavaScript frameworks and wish to write web applications using Blazor WebAssembly, leveraging the power of .NET and C#. The book assumes beginner-level knowledge of the C# language, .NET framework, Microsoft Visual Studio, and web development concepts.

Título: Blazor WebAssembly by example a project-based guide to building web apps with .NET, Blazor WebAssembly, and C# Toi B. Wright

Editorial: Birmingham, England Mumbai Packt [2021] 2021

Descripción física: 1 online resource (267 pages)

Nota general: Includes index

Contenido: Cover -- Title Page -- Copyright and Credits -- Contributors -- Table of Contents -- Preface -- Chapter 1: Introduction to Blazor WebAssembly -- Benefits of using the Blazor framework -- .NET Framework -- SPA framework -- Razor syntax -- Awesome tooling -- Hosting models -- Blazor Server -- Blazor WebAssembly -- What is WebAssembly? -- WebAssembly goals -- WebAssembly support -- Setting up your PC -- Installing Visual Studio Community Edition -- Installing .NET 5.0 -- Installing SQL Server Express -- Summary -- Questions -- Further reading -- Chapter 2: Building Your First Blazor WebAssembly Application -- Technical requirements -- Razor components -- Using components -- Parameters -- Naming components -- Component life cycle -- Component structure -- Routing in Blazor WebAssembly -- Route parameters -- Catch-all route parameters -- Route constraints -- Razor syntax -- Inline expressions -- Control structures -- Project overview -- Creating the Demo Blazor WebAssembly project -- Creating the Demo project -- Running the Demo project -- Examining the Demo project's structure -- Examining the shared Razor components -- Examining the routable Razor components -- Using a component -- Adding a parameter to a component -- Using a parameter with an attribute -- Adding a route parameter -- Using partial classes to separate markup from code -- Creating a custom Blazor WebAssembly project template -- Creating an empty Blazor project -- Creating a project template -- Updating a custom project template -- Using a custom project template -- Summary -- Questions -- Further reading -- Chapter 3: Building a Modal Dialog Using Templated Components -- Technical requirements -- RenderFragment parameters -- EventCallback parameters -- CSS isolation -- Enabling CSS isolation -- Supporting child components -- Project overview -- Creating the modal dialog project Getting started with the project -- Adding the Dialog component -- Adding a CSS -- Testing the Dialog component -- Adding EventCallback parameters -- Adding RenderFragment parameters -- Creating a Razor class library -- Testing the Razor class library -- Adding a component to the Razor class library -- Summary -- Questions -- Further reading -- Chapter 4: Building a Local Storage Service Using JavaScript Interoperability (JS Interop) -- Technical requirements -- Why use JavaScript? -- Exploring JS interop -- InvokeVoidAsync -- InvokeAsync -- Invoking JavaScript from .NET synchronously -- Invoking .NET from JavaScript -- Understanding local storage -- Project overview -- Creating the local storage service -- Creating the local storage service project -- Writing JavaScript to access localStorage -- Adding the ILocalStorageService interface -- Creating the LocalStorageService class -- Writing to localStorage -- Reading from localStorage -- Summary -- Questions -- Further reading -- Chapter 5: Building a Weather App as a Progressive Web App (PWA) -- Technical requirements -- Understanding PWAs -- HTTPS -- Manifest files -- Service workers -- Working with manifest files -- Working with service workers -- Service worker life cycle -- Updating a service worker -- Types of service workers -- Using the CacheStorage API -- Using the Geolocation API -- Using the OpenWeather One Call API -- Project overview -- Creating a PWA -- Getting started with the project -- Adding a JavaScript function -- Using the Geolocation API -- Adding a Forecast class -- Adding a DailyForecast component -- Using the OpenWeather One Call API -- Displaying the forecast -- Adding the logo -- Adding a manifest file -- Adding a simple service worker -- Testing the service worker -- Installing the PWA -- Uninstalling the PWA -- Summary -- Questions -- Further reading Chapter 6: Building a Shopping Cart Using Application State -- Technical requirements -- Application state -- Understanding DI -- DI container -- Service lifetime -- Project overview -- Creating the shopping cart project -- Getting started with the project -- Adding the Product class -- Adding the Store page -- Demonstrating that application state is lost -- Creating the ICartService interface -- Creating the CartService class -- Registering CartService in the DI container -- Injecting CartService -- Adding the cart total to all of the pages -- Using the OnChange method -- Summary -- Questions -- Further reading -- Chapter 7: Building a Kanban Board Using Events -- Technical requirements -- Event handling -- Lambda expressions -- Preventing default actions -- Attribute splatting -- Arbitrary parameters -- Project overview -- Creating the Kanban board project -- Getting started with the project -- Adding the classes -- Creating the Dropzone component -- Adding a style sheet -- Creating the Kanban board -- Creating the NewTask component -- Using the NewTask component -- Summary -- Questions -- Further reading -- Chapter 8: Building a Task Manager Using ASP.NET Web API -- Technical requirements -- Understanding hosted applications -- Client project -- Server project -- Shared project -- Using the

HttpClient service -- Using JSON helper methods -- GetFromJsonAsync -- PostAsJsonAsync -- PutAsJsonAsync --
HttpClient.DeleteAsync -- Project overview -- Creating the TaskManager project -- Getting started with the project
-- Examining the hosted Blazor WebAssembly app -- Emptying the solution -- Adding the TaskItem class --
Adding the TaskItem API controller -- Setting up SQL Server -- Displaying the tasks -- Completing the tasks --
Deleting the tasks -- Adding new tasks -- Summary -- Questions -- Further reading Chapter 9: Building an Expense
Tracker Using the EditForm Component -- Technical requirements -- Overview of the EditForm component --
Using the built-in input components -- Using the validation components -- Project overview -- Creating the
ExpenseTracker project -- Getting started with the project -- Removing the demo project -- Adding the classes --
Adding the API controllers -- Creating the SQL Server database -- Viewing the expenses -- Adding the
ExpenseEdit component -- Adding the input components -- Summary -- Questions -- Further reading -- Why
subscribe? -- About Packt -- Other Books You May Enjoy -- Index

ISBN: 1-80056-393-0

Material: Web applications

Baratz Innovación Documental

- Gran Vía, 59 28013 Madrid
- (+34) 91 456 03 60
- informa@baratz.es