BLAZOR FOR BOSSES

BY BARBARA BONILLA

"This book contains everything you need to know...and more!"

QS2POINT
www.qs2point.com

INTRODUCTION TO BLAZOR

1. Introduction to Blazor

1.1 What is Blazor?

Blazor is a framework[1] that lets you build interactive web user interfaces using C#[2].

1.2 When was Blazor introduced?

The first release of Blazor was in September 2019 (server-side version). It was an experimental project from 2018 up to this point. Client-side Blazor was released in May 2020. In November 2020 it became part of .NET 5 so it is a core technology of .NET now.

1.3 The Basics (a.k.a. What The Boss Should Know)

- Blazor currently exists in two flavours (hosting models):
  client-side You can choose to run your application locally (meaning it will be downloaded to the client-side and run from there). In this case, we'll rely on WebAssembly[3].
  server-side Alternatively, Blazor apps can also run on the server (using SignalR connections between the browser and the server-side app).
  Both versions rely on .NET and you can even share all the code between client-side Blazor and server-side Blazor. All the goodness that .NET brings (performance and security features) also apply in full in Blazor since it sits on top of ASP.NET Core, therefore it has access to all the features of the platform.
  (Client-side Blazor is the most exciting one!)
- The building block of Blazor applications are the Components. Components are UI elements that contain Razor markup, C# code and possibly other nested components[4]. The components can be wrapped inside of a library and shared between projects.

[1] a feature of ASP.NET Core

[2] which previously was only used to write back-end code but now you can write client and server code in C# (plus HTML and CSS)

[3] an open standard that defines a portable binary code format for executable programs

[4] you can compose components in order to create more complex UI elements

- While Blazor already has quite a large number of features onboard might be needed to integrate with JavaScript to get access to some browser features. You can extend Blazor (bring in components) through JavaScript interop. This can be seen as a bridge that allows Blazor apps to invoke functionality otherwise unavailable to them out of the box.
- Blazor has a component model comparable to React or Angular.
- Blazor is built on top of open web standards and does not require installing any plugin in the browser. Blazor should run in all modern browsers, mobile or desktop without any trouble.
- Client-side Blazor projects can easily be converted to server-side by following some patterns.
- Blazor makes working with data easy. By bringing in support for the HTTP client we can easily access data living on a server[5]. Once you have the data in our application we can use the data binding engine that comes with Blazor to make it easy to display or edit data (and validate the data as well).

1.4 The Learning Curve

A shocking thing about Blazor is how genuinely simple it feels to use. Blazor combines the ease of the Razor markup syntax with other .NET concepts. It has borrowed the best patterns from popular JavaScript frameworks like Angular and React while leveraging Razor templates and providing parity with other .NET conventions.
The learning curve is very reduced for .NET developers.

1.5 Shortest Summary Ever

Blazor allows creating rich experiences in the browser using C# and HTML.

[5] you can interact with your own API or a third-party API

2. A note from the writer

Refer to the following resources[1] for a deeper understandment of Blazor or contact *bbonillasnchz* if further information is needed.

(eBook) Blazor, A Beginners Guide. A quick-start guide to productivity with Blazor by Ed Charbeneau

(PluralSight Course) Blazor: Getting Started by Gill Cleeren
Recommended aproach: Watch only the Introduction and Summary chapters of each module.

[1] These resources are the base of this information