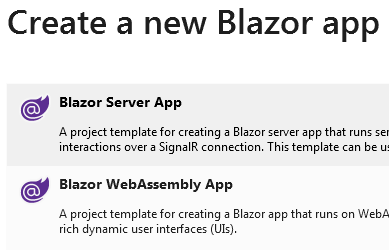
Blazor hosting models

* We will create our first Blazor project. In fact, we will create 2 blazor projects. One with server-side hosting and the other with client-side hosting. We will then run both the projects and see the difference in where and how they execute.
* Along the way, we will understand the difference between the two hosting models and the pros and cons of each

Blazor WebAssembly app template missing

Blazor WebAssembly (also called client-side blazor) is still in preview. So, if you try to create a Blazor project using Visual Studio 2019, you will only find **Blazor Server App** template and not **Blazor WebAssembly App** template.



Execute the following command to install the **Blazor WebAssembly App** template

dotnet new --install Microsoft.AspNetCore.Blazor.Templates::3.1.0-preview4.19579.2

Copy

You can execute this command either from the Package Manager Console in Visual Studio or from the command prompt.

[Click here to get the latest Blazor WebAssembly App template](https://www.nuget.org/packages/Microsoft.AspNetCore.Blazor.Templates)

Blazor hosting models

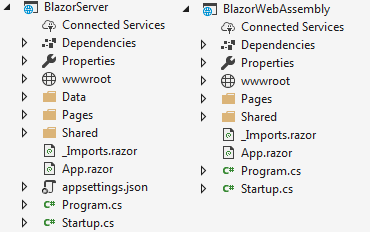
With Blazor we have two hosting models, **Blazor WebAssembly** (Client-side hosting model) and **Blazor Server** (Server-side hosting model)



As the template names imply, use the Blazor Server App template to create a blazor application with server hosting model and Blazor WebAssembly template to create a blazor application with client hosting model.

Blazor Server vs Blazor WebAssembly

The project structure and layout is not that different between the 2 project types.

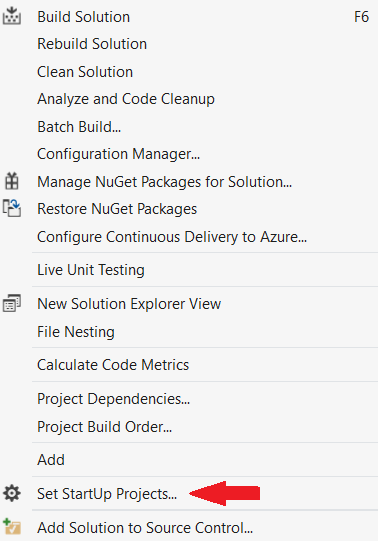


We will discuss the purpose of these different files and folders in detail in our next video.

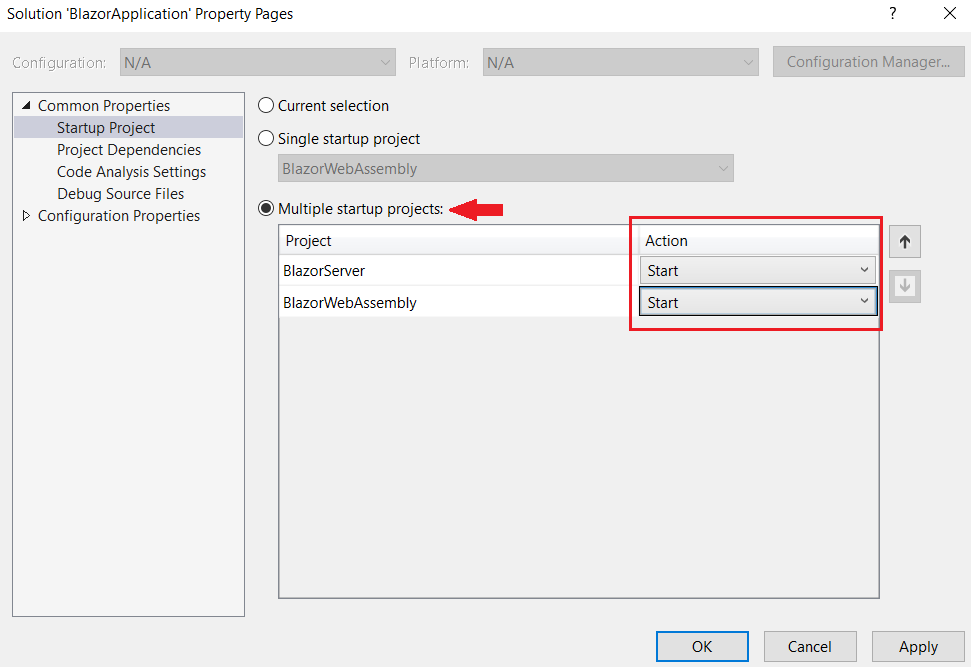
Configure multiple StartUp projects in Visual Studio

We can configure more than one project as a **StartUp project** in Visual Studio. The following are the steps.

Right click on the Solution Name in the Solution Explorer and select **Set StartUp Project**s option

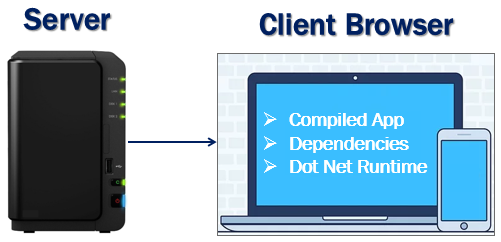


Select **Multiple startup projects** radio button. Select **Start** from the **Action** dropdownlist against each project that you want to setup as a startup project.



Blazor WebAssembly hosting model

With this hosting model, the application runs directly in the browser on WebAssembly. So, everything the application needs i.e the compiled application, it's dependencies and the .NET runtime are downloaded to the client browser from the server. A Blazor WebAssembly app can run entirely on the client without a connection to the server or we can optionally configure it to interact with the server using web API calls or SignalR.



Blazor WebAssembly hosting model benefits :

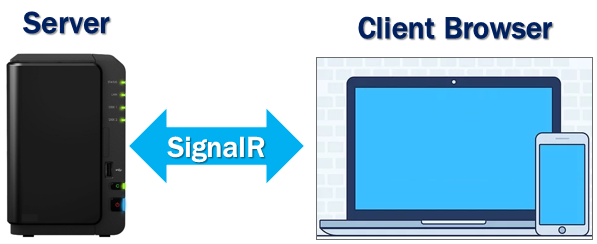
* A Blazor WebAssembly app can run entirely on the client machine. So, after the application is downloaded, a connection to the server is not required. This means there is no need for your server to be up and running 24X7.
* Work is offloaded from the server to the client. It is the client resources and capabilities that are being used.
* We do not need a full-blown ASP.NET Core web server to host the application. We just need a server somewhere, that can deliver the application to the client browser. This means we can host the application on our own server on the internet somewhere, in the cloud, on Azure as a static website or even on a CDN Content Delivery Network.

Downsides of Blazor WebAssembly hosting :

* The very first request usually takes longer as the entire app, it's dependencies and the .NET runtime must be downloaded to the client browser. Keep in mind, it's only the first request that takes longer than usual. If that same client visits the application later, it usually starts fast because the browser caches the files.
* Since the app runs entirely on the client browser, it is restricted to the capabilities of the browser.
* Depending on the nature of the application, capable client hardware and software is required. From software standpoint, for example, at least a browser with WebAssembly support is required.

Blazor Server hosting model

With this hosting model, the application is executed on the server. Between the client and the server a SignalR connection is established. When an event occurs on the client such as a button click for example, the information about the event is sent to the server over the SignalR connection. The server handles the event and for the generated HTML a diff (difference) is calculated. The entire HTML is not sent again to the client, it's only the diff that is sent to the client over the SignalR connection. The browser then updates the UI. Since only the diff is applied to update the UI, the application feels faster and more responsive to the user.



Blazor Server hosting model benefits :

* The app loads much faster as the download size is significantly smaller than a Blazor WebAssembly app
* Since the app runs on the server, it can take full advantage of server capabilities, including use of any .NET Core compatible APIs.
* All the client needs, to use the app is a browser. Even browsers that don't support WebAssembly can be used.
* More secure because the app's .NET/C# code isn't served to clients.

Downsides of Blazor Server hosting :

* A full-blown ASP.NET Core server is required to host the application. Serverless deployment scenarios such as serving the app from a CDN aren't possible.
* An active connection to the server is always required. This means there is a need to keep the server up and running 24X7. If the server is down, the application stops working.
* As every user interaction involves a round trip to the server a higher latency usually exists when compared with Blazor WebAssembly hosting.
* Scalability can be challenging especially for the apps that have many users as the server must manage multiple client connections and handle client state. However, we can overcome this scalability issue, by using Azure SignalR Service with a Blazor Server app. This service allows a Blazor Server app to scale really well by supporting a large number of concurrent SignalR connections.

Slides

