# Blazor

## Overview

For building interactive web UI. Use C# instead of JS.

## Components

Define Flexible UI rendering logic

Handle user events

Nested and reused

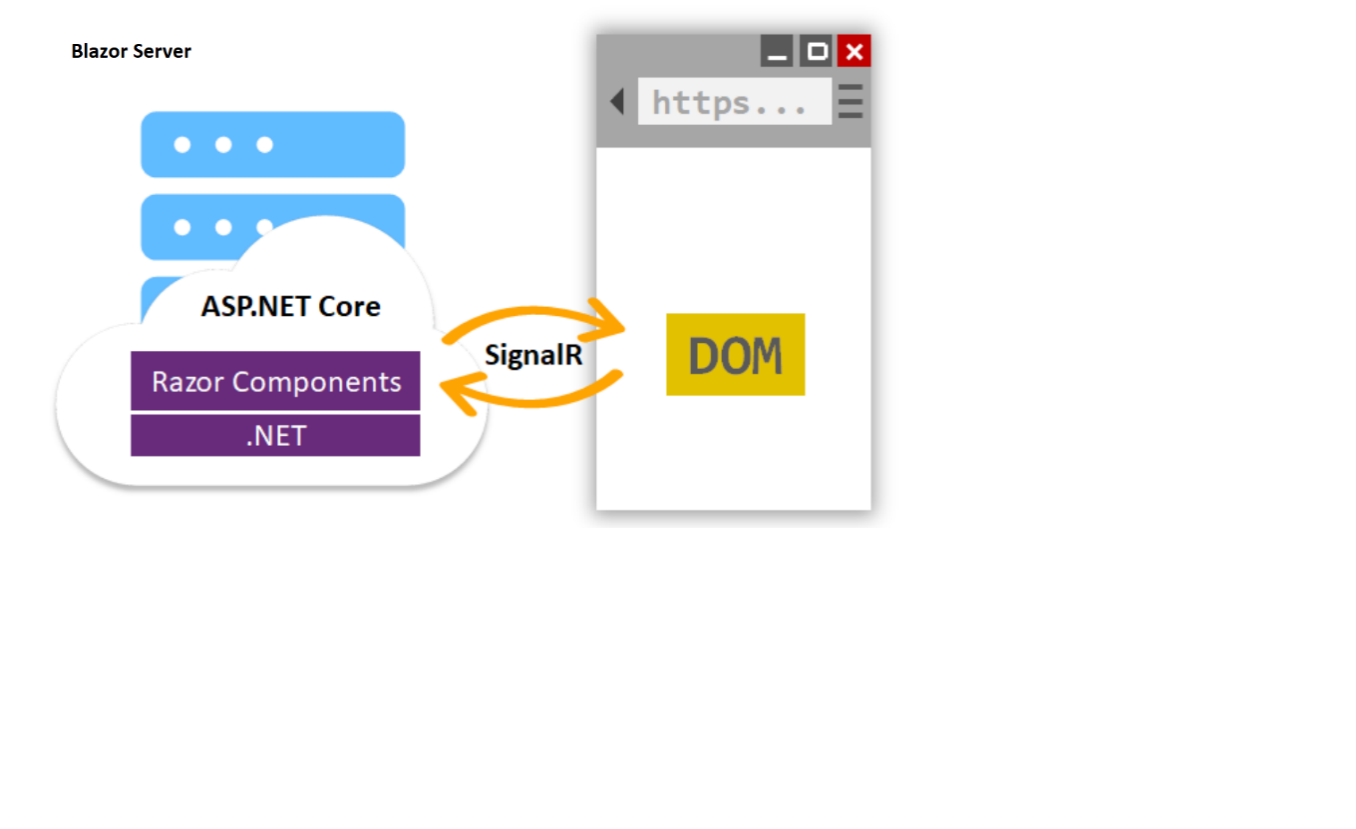
Shared and distributed as Razor class lib or nuget packages

Built as razor markup page in .razor file extension

### Sample component

<div class="card" style="width:22rem">  
<div class="card-body">  
<h3 class="card-title">@Title</h3>  
<p class="card-text">@ChildContent</p>  
<button @onclick="OnYes">Yes!</button>  
</div>  
</div>  
@code {  
[Parameter]  
public RenderFragment? ChildContent { get; set; }  
[Parameter]  
public string? Title { get; set; }  
private void OnYes()  
{  
Console.WriteLine("Write to the console in C#! 'Yes' button  
selected.");  
}  
}

## Blazor Server



* Hosting Razor components on the server
* UI Updates are handled over signalR connection
* Handles javascript interop
* Rendering the markup in UI varies for blazor server app and asp.net core apps using razor views or pages. Both models uses razor language.
* View to HTML text. Once its done server disposes view or page. For every request the page is re-rendered to HTML again and send to the client
* Graph of components is equivalent for DOM in HTML or XML
* Component graph includes state held in properties and fields.
* Binary form of component graph is sent between server and client
* After the connection, the components static prerendered elements are replaced with interactive elements.
* A component is disposed after the user navigates away from the component

### Basics

#### Project type in visual studio

Blazor server app. Console application is the output type for this project.

App.Razor: To take the decision when routedata is present/absent

**app.MapFallbackToPage("/\_Host"); -> \_Host.cshtml -> App.Razor**

#### Web server

A web server is a system that hosts web applications and serves content to web browsers through the HTTP protocol. It listens to requests from browsers and responds with the requested content (such as HTML files, images, etc.).

IIS and Kestrel server.

Kestrel is only for development in visual studio.

IIS express can host multiple websites.

#### Web Browser

A web browser like Google Chrome is a software application used to access information on the World Wide Web. It interprets and renders HTML, CSS, and JavaScript to display web pages to users. Browsers also manage client-side interactions and communicate with web servers to request and receive content.

#### Web Application

A web application is a client-server software application that runs in a web browser. It involves both server-side logic (running on a web server) and client-side logic (running in the browser).

#### Routing

Identifier for each razor page. Using **@Page** to define the routing. We can define more than one route to a razor component. Path next to @Page directive is irrespective of razor component name.

#### Development Hosting

IIS allows in process and out of process hosting. In case of out of process web application and web server will be running in two different process.

#### Launchsettings.json

All launch profiles information, available profiles, URL, environment variables, etc.

### Included systems

#### Dependency Injection

AddSingleton(one instance throughout the life span of application), AddScoped (one instance per session), AddTransient (one instance per call. So, In same session it will be instantiated multiple times). Use the appropriate injection based on use case.

Use **@inject** to use the service in razor page

For class,

In constructor, add parameter for the type and use it inside the required class.

#### Logging

Be default log to console and debug window will happen. Additional logging settings will be present in **appsettings.json.** If the environment is set to development then appsettings.json will be overridden by appsettings.Development.json. In case of production appsettings.json will take the benefits.

|  |
| --- |
| **Trace** |
| **Debug** |
| **Information** |
| **Warning** |
| **Error** |
| **Critical - Highest level** |

#### Configuration

**Order of precedence of several configuration providers**

* ***JSON Files***
  + ***Appsettings.json***
  + ***Appsettings.<Environment>.json***
* ***Secrets.json - stored in local machine and will not be present in code location***
* XML Files
* INI Files
* Environment variables
* Command-line arguments
* In-memory .NET objects
* Secret Manager storage
* Encrypted in Azure Key Vault - Highest

Property will be overridden by highest precedence provider

Note: Italic represents frequently used ones

Refer: https://devblogs.microsoft.com/premier-developer/order-of-precedence-when-configuring-asp-net-core/

##### Secrets.json

File location: C:\Users\balaj\AppData\Roaming\Microsoft\UserSecrets\

Inside this location a new folder(unique GUID) will be created for each project. Secrets.json will be present inside this folder.

## Blazor Webassembly

* Run dotnet code in web browser
* Bytecode format
* Open web standard, so no plugins required
* Will access browser functionality using JS interop
* Runs in JS sandbox with protection against malicious actions on the client machine

