

WEB DAY 2023

MILANO 16 MARZO



Blazor: sviluppo desktop, web e mobile cross-platform con un unico framework

Michele Aponte

CEO&CTO @ Ellycode / CEO @ Blexin / CTO @ Improve Skills

Founder @ Blazor Developer Italiani / MVP Developer Technologies & AI



apomic80



apomic80



apomic80



apomic80

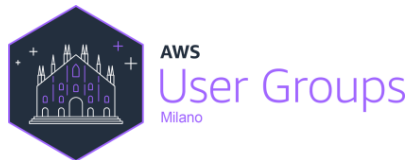


apomic80

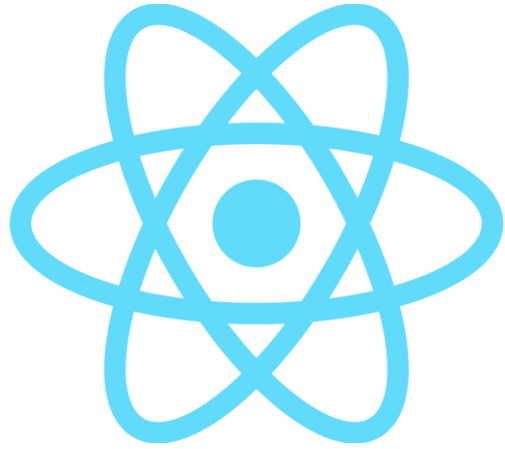
/* Sponsor */



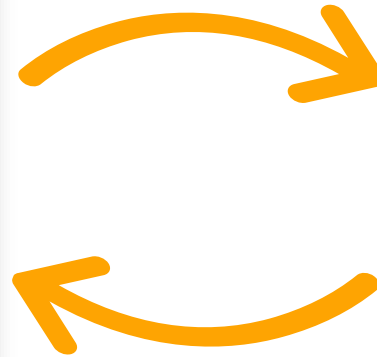
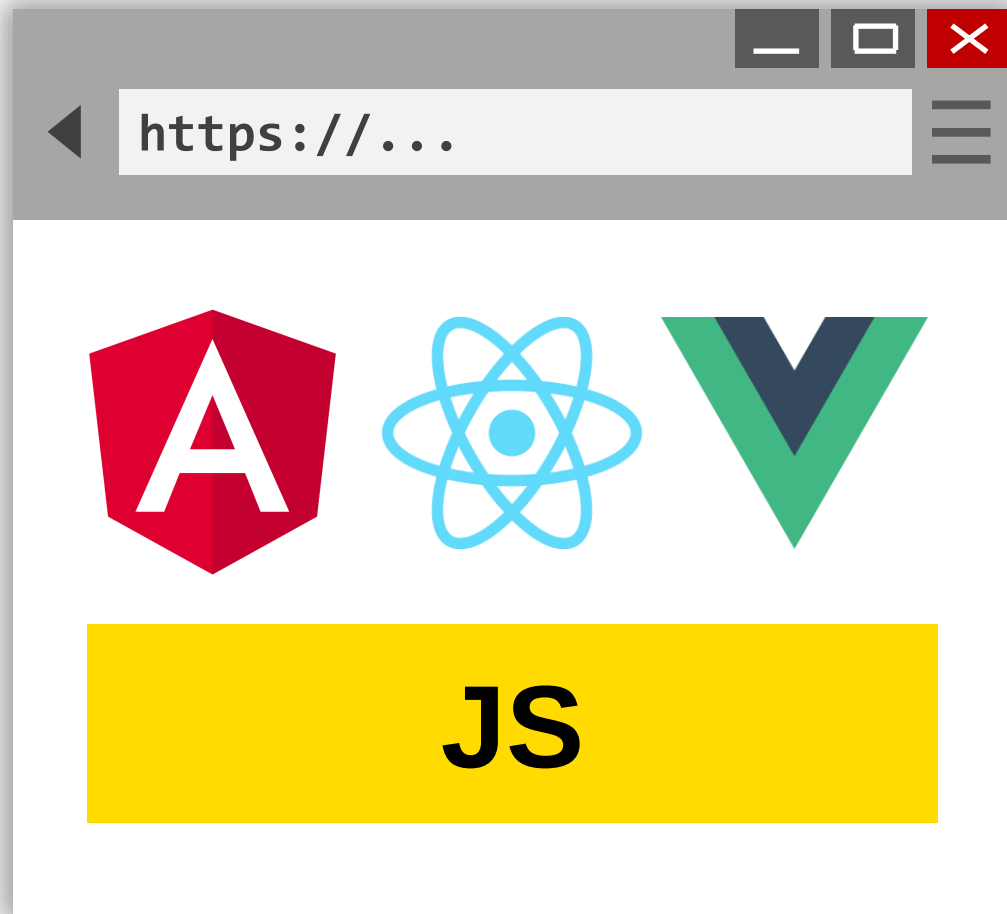
/* Partner */



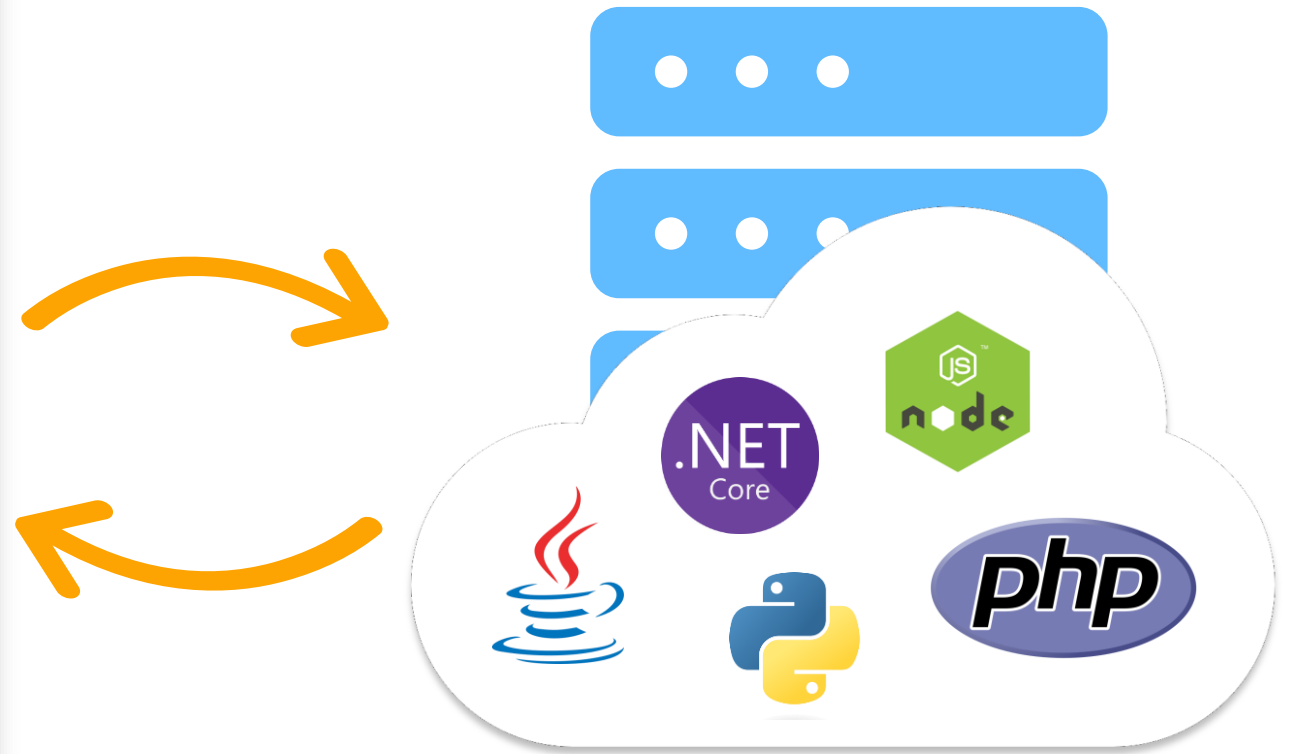
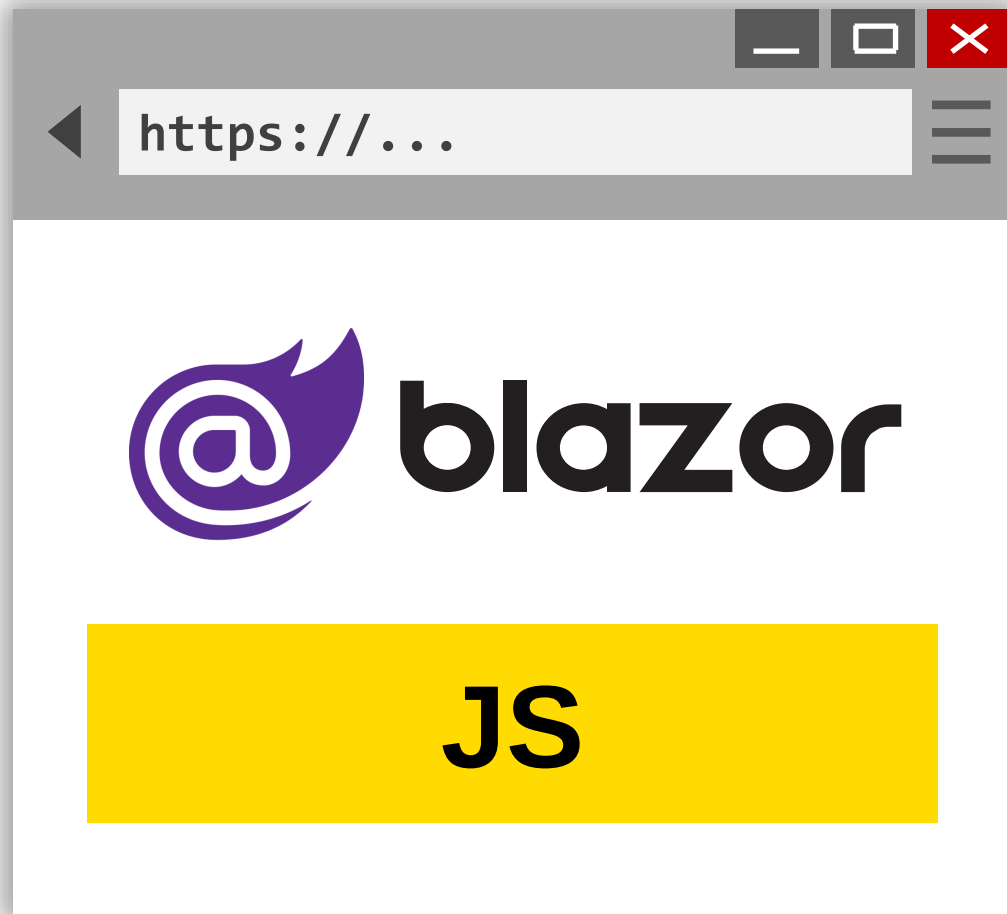
Perché Blazor?



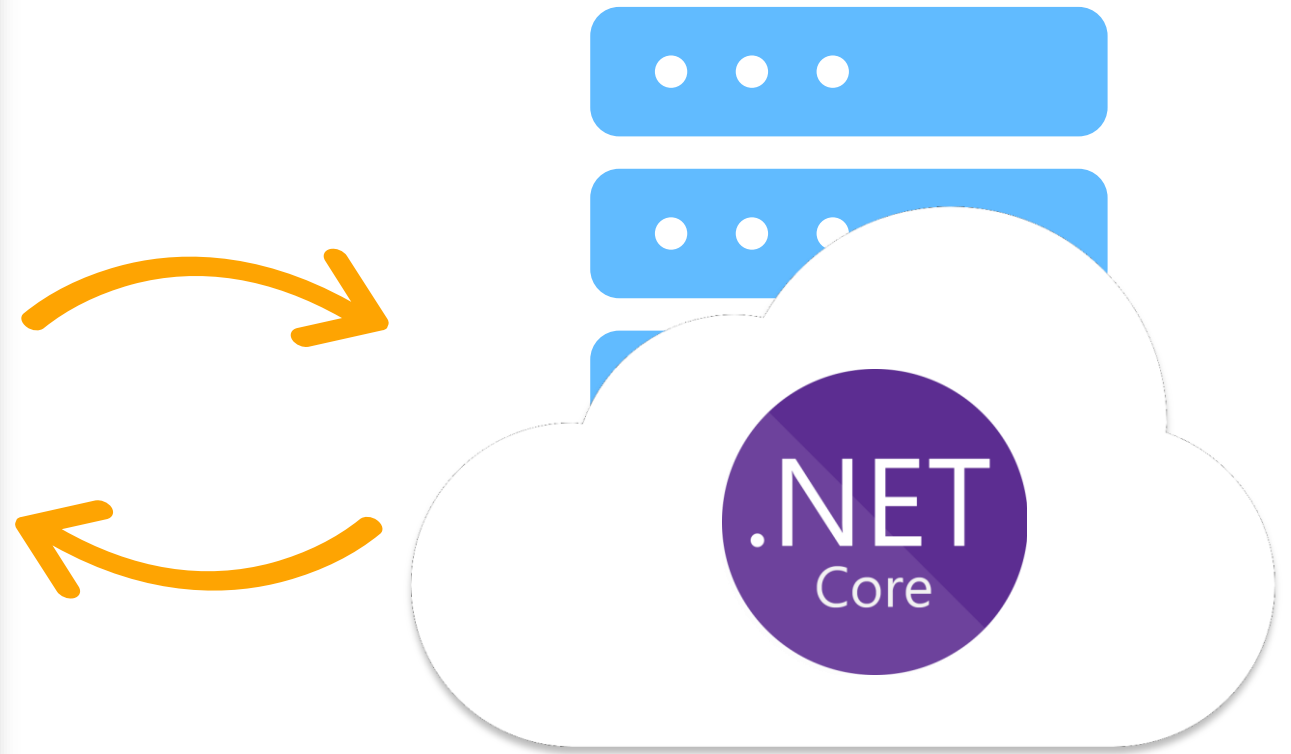
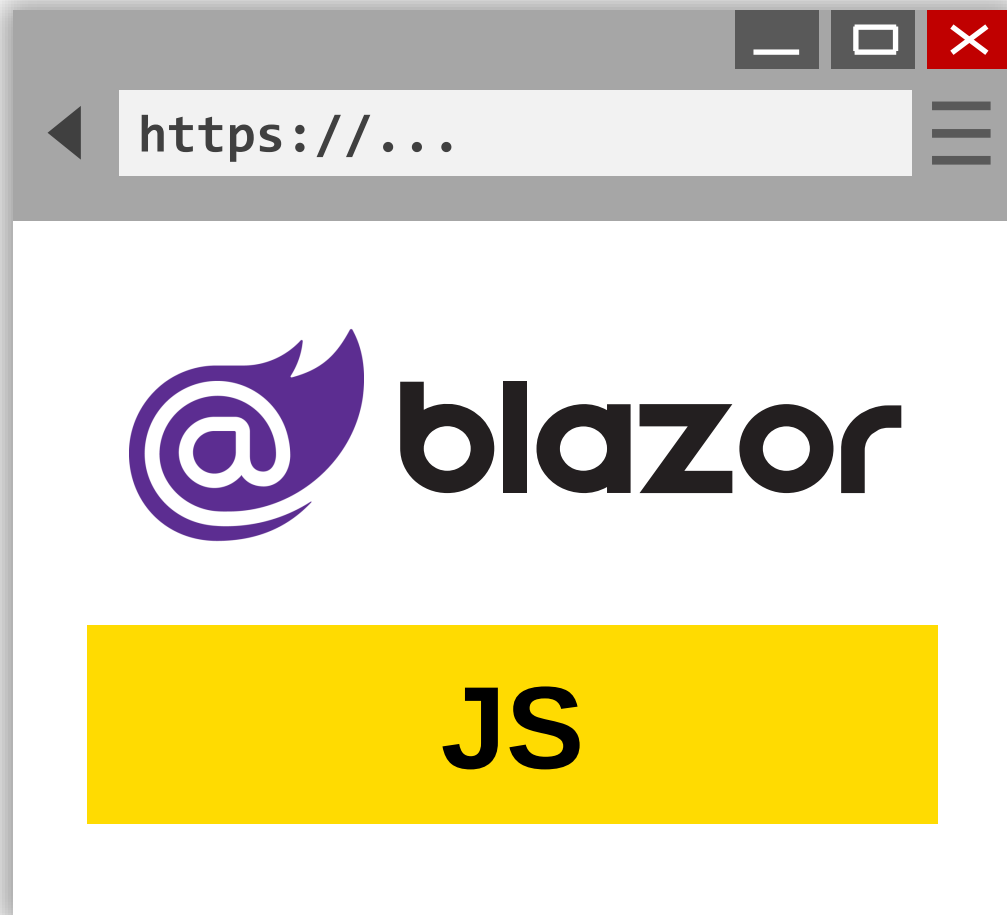
Perché Blazor?



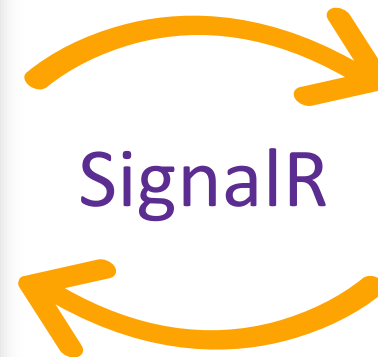
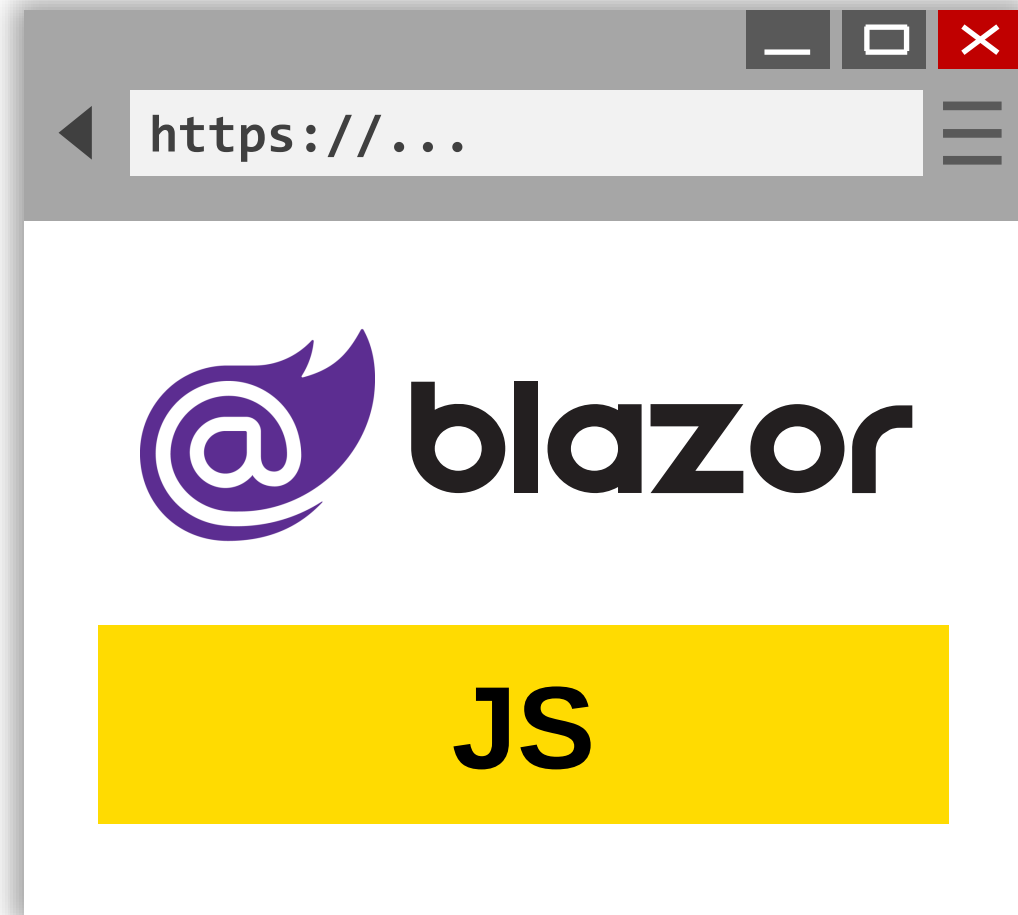
Perché Blazor?



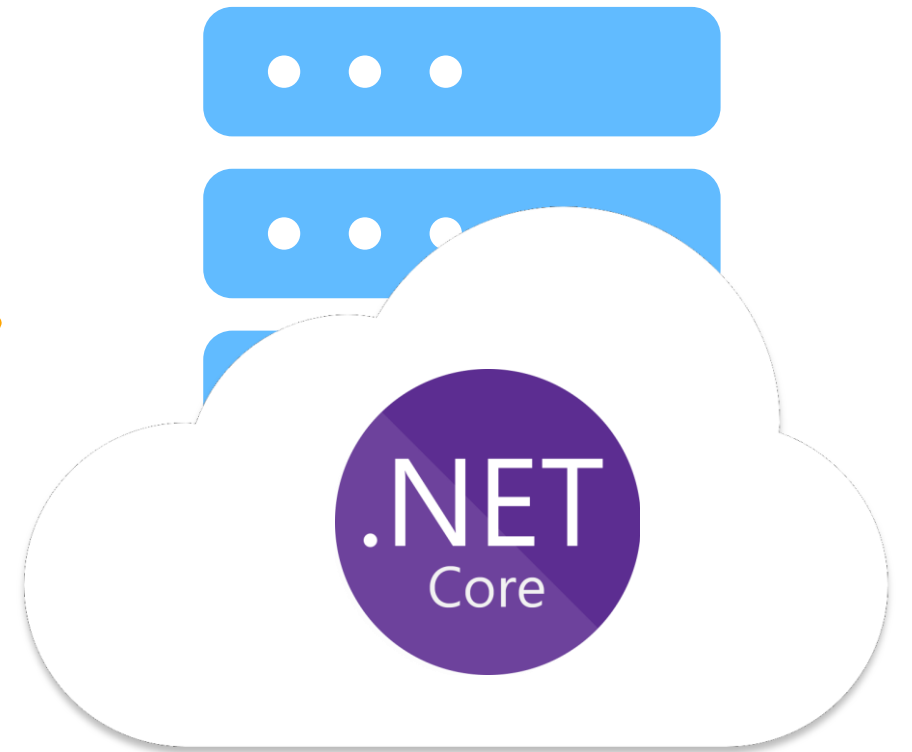
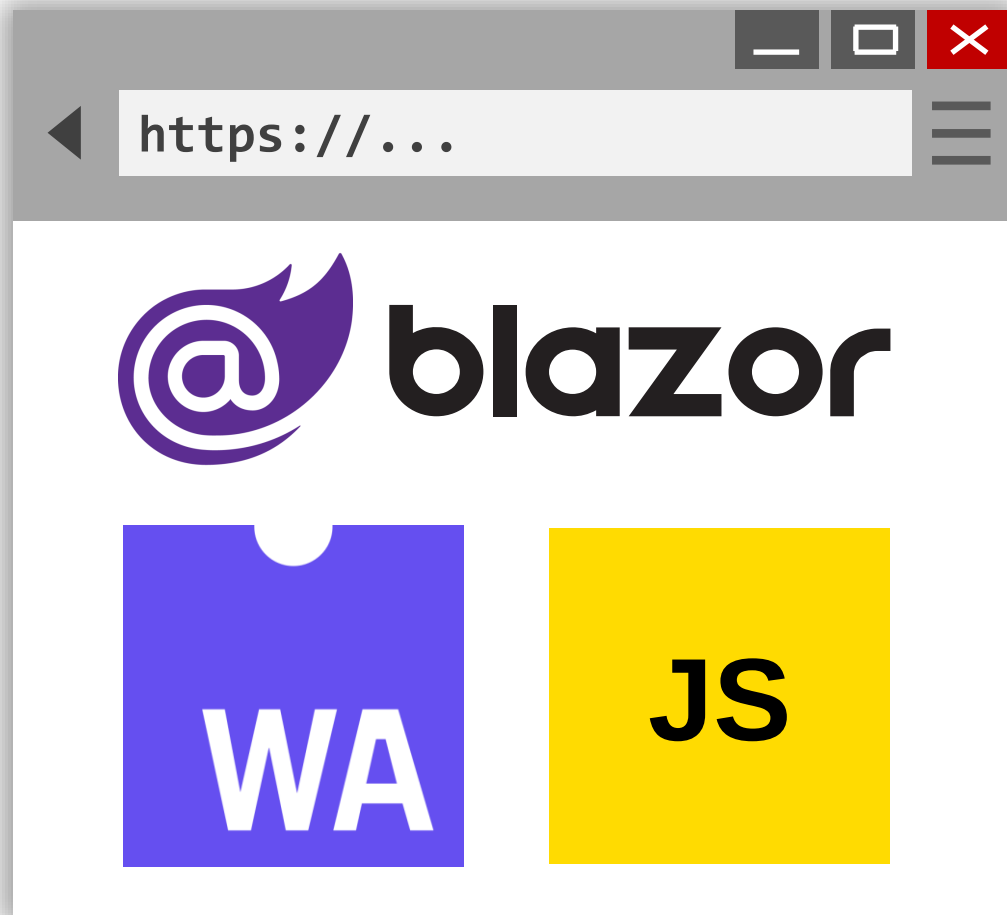
Perché Blazor?



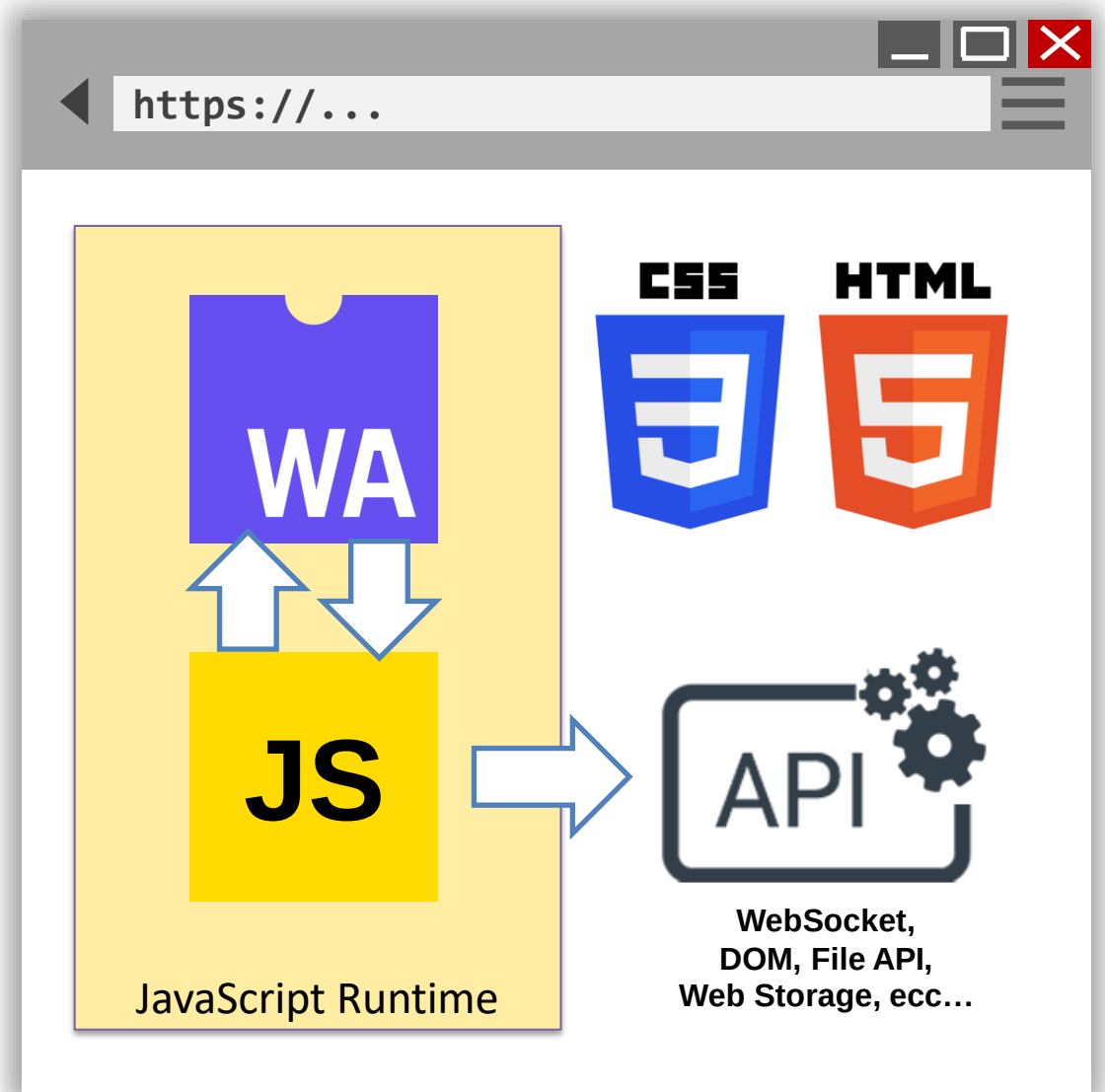
Blazor Server



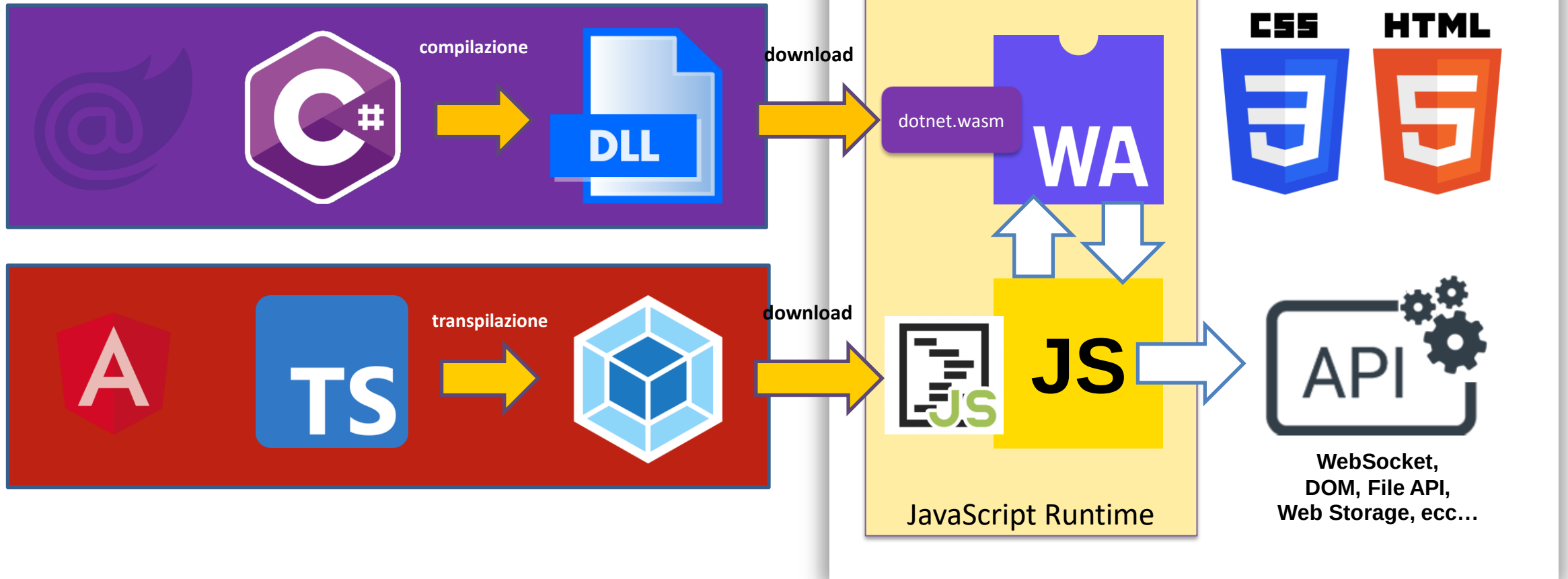
Blazor WebAssembly



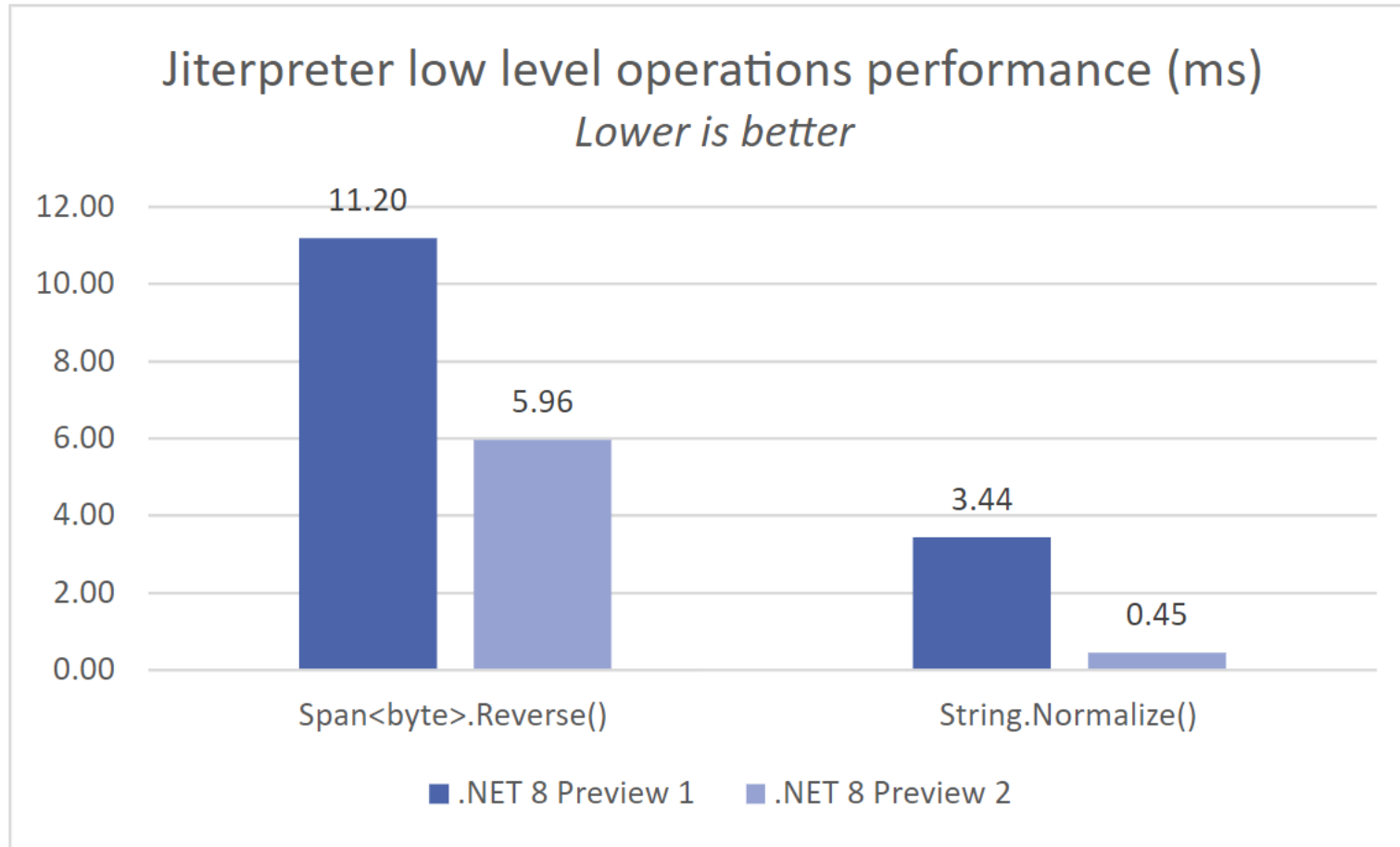
WebAssembly?



WebAssembly?



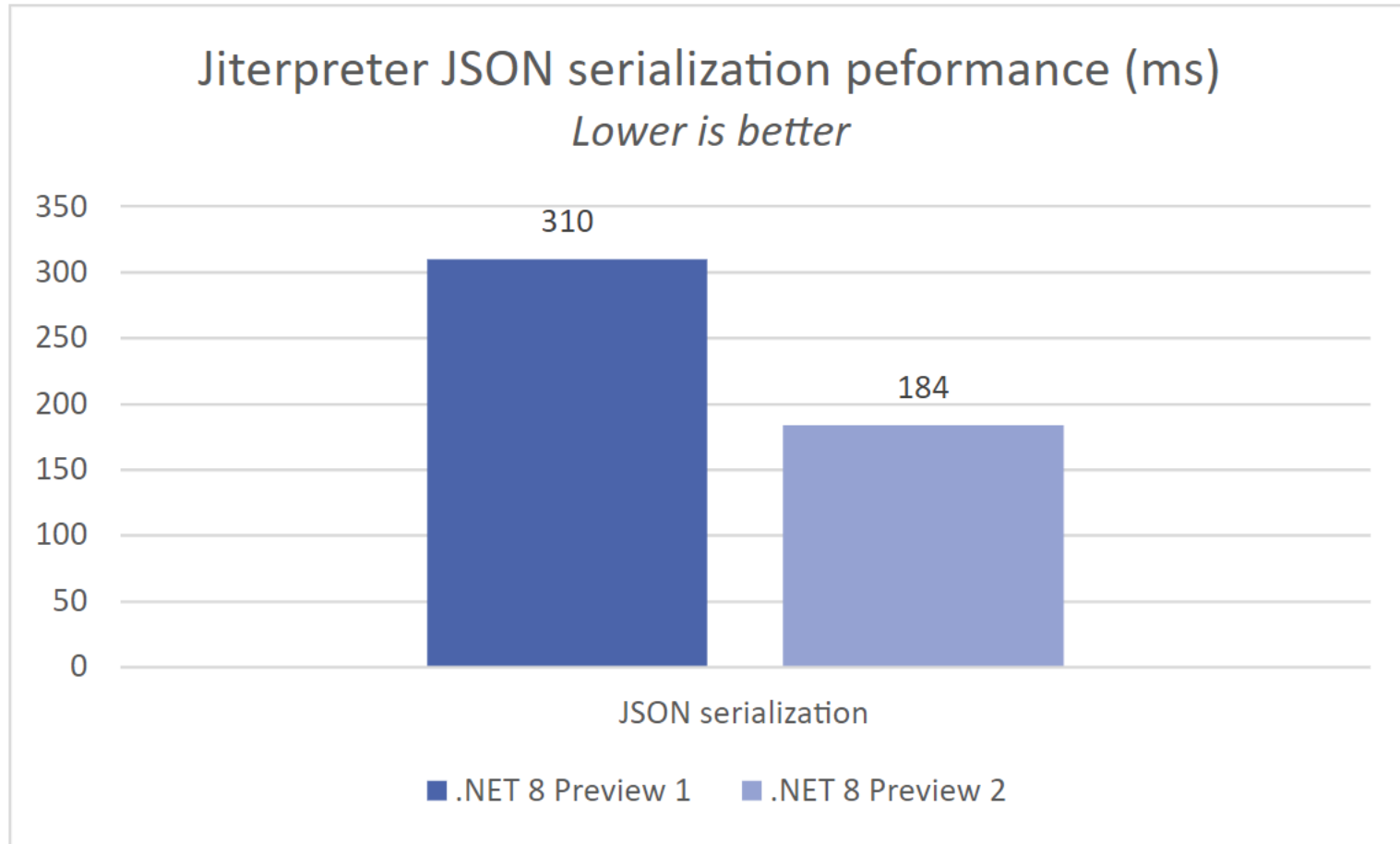
Performance Improved



Il nuovo **jiterpreter** ottimizza l'esecuzione dell'interprete del bytecodes sostituendolo con piccoli blob di codice WebAssembly

<https://devblogs.microsoft.com/dotnet/asp-net-core-updates-in-dotnet-8-preview-2/>

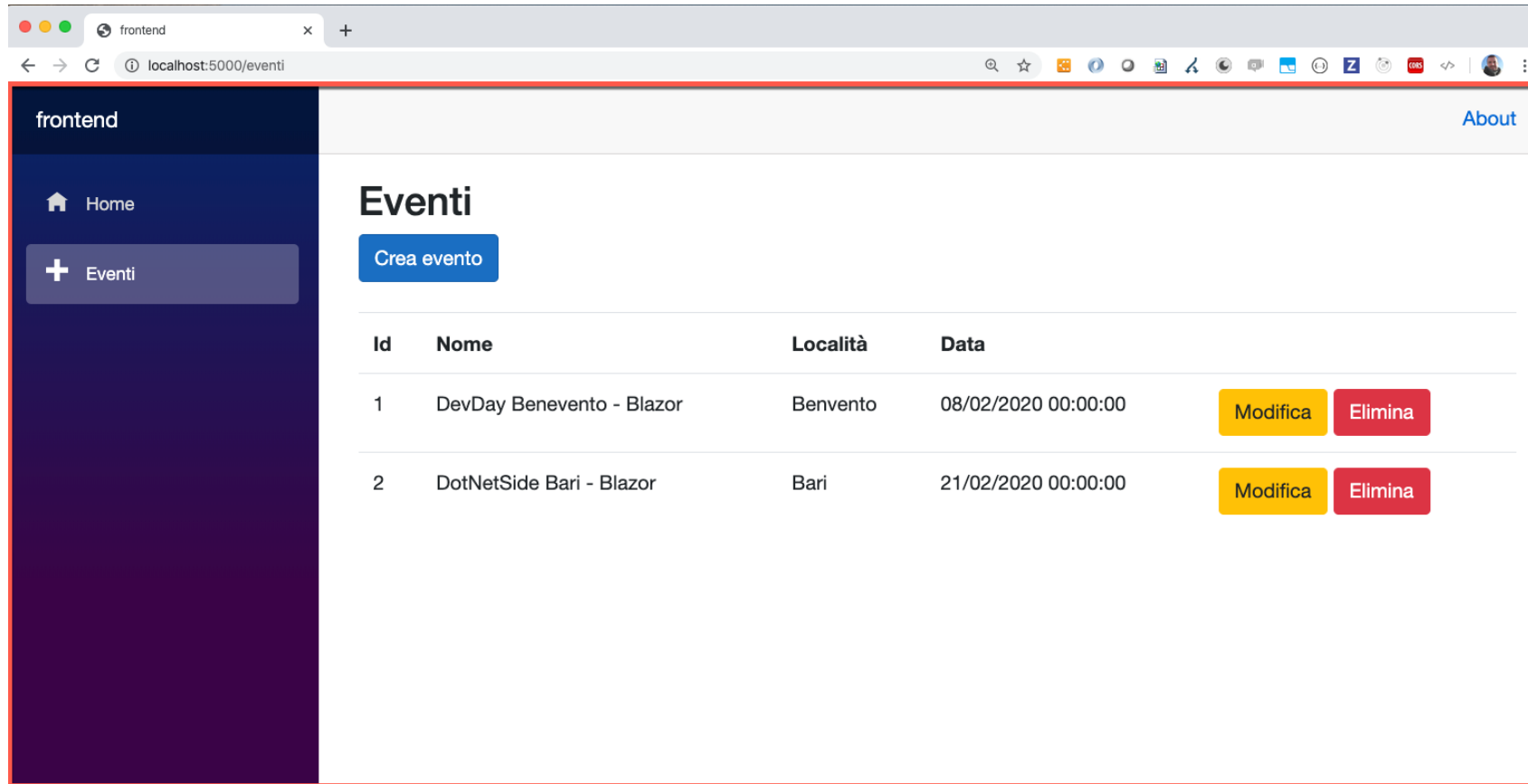
Performance Improved



Il nuovo **jiterpreter** ottimizza l'esecuzione dell'interprete del bytecodes sostituendolo con piccoli blob di codice WebAssembly

<https://devblogs.microsoft.com/dotnet/asp-net-core-updates-in-dotnet-8-preview-2/>

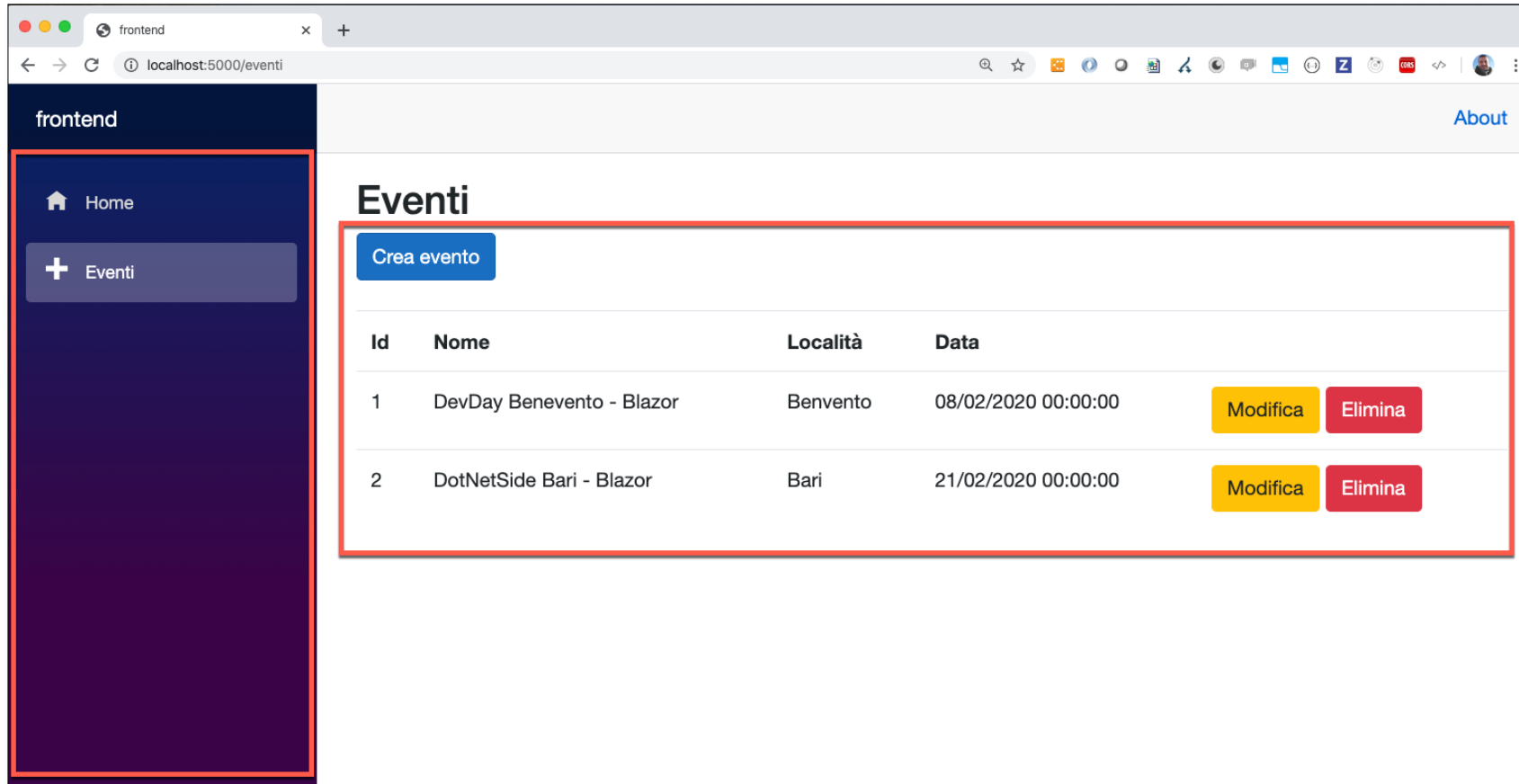
Blazor spiegato a mia nonna (cit.)



- Tutta l'applicazione viene **costruita dinamicamente** a partire da **un'unica pagina**

- La **navigazione** viene **gestita** da un apposito componente

Blazor Components



I singoli elementi della pagina sono dei **componenti**:

- formano un **albero** di componenti
- della giusta **dimensione (SoC)**
- **riutilizzabili** e **NON accoppiati**

Blazor Components

The diagram illustrates the Blazor component lifecycle. On the left, a browser window shows a web application with a sidebar navigation menu. A red box highlights the sidebar, and a purple arrow points from it to the central code block. The central code block, titled 'NavMenu.razor', contains the Razor code for the component. It is divided into two sections: the top section shows the HTML markup for the navigation menu, and the bottom section shows the C# code-behind logic. The right side of the diagram shows a partial view of the web application, including a table of events and buttons for editing and deleting items.

NavMenu.razor

```
<div class="top-row pl-4 navbar navbar-dark">
  <a class="navbar-brand" href="">frontend</a>
  <button class="navbar-toggler" @onclick="ToggleNavMenu">
    <span class="navbar-toggler-icon"></span>
  </button>
</div>

<div class="@NavMenuCssClass" @onclick="ToggleNavMenu">
  <ul class="nav flex-column">
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
        <span class="oi oi-home" aria-hidden="true"></span> Home
      </NavLink>
    </li>
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="eventi">
        <span class="oi oi-plus" aria-hidden="true"></span> Eventi
      </NavLink>
    </li>
  </ul>
</div>

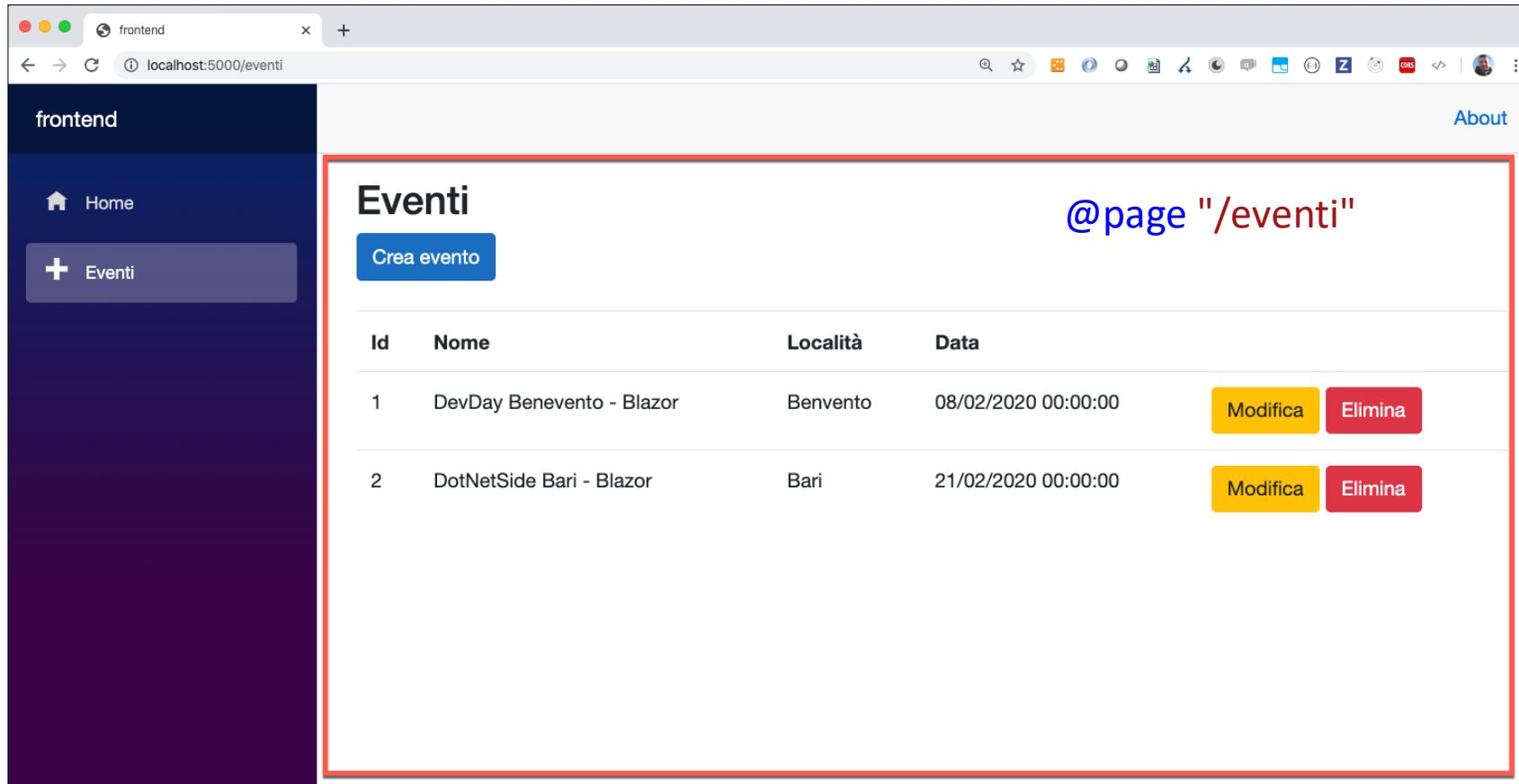
@code {
  2 references
  private bool collapseNavMenu = true;

  1 reference
  private string NavMenuCssClass => collapseNavMenu ? "collapse" : null;

  2 references
  private void ToggleNavMenu()
  {
    collapseNavMenu = !collapseNavMenu;
  }
}
```

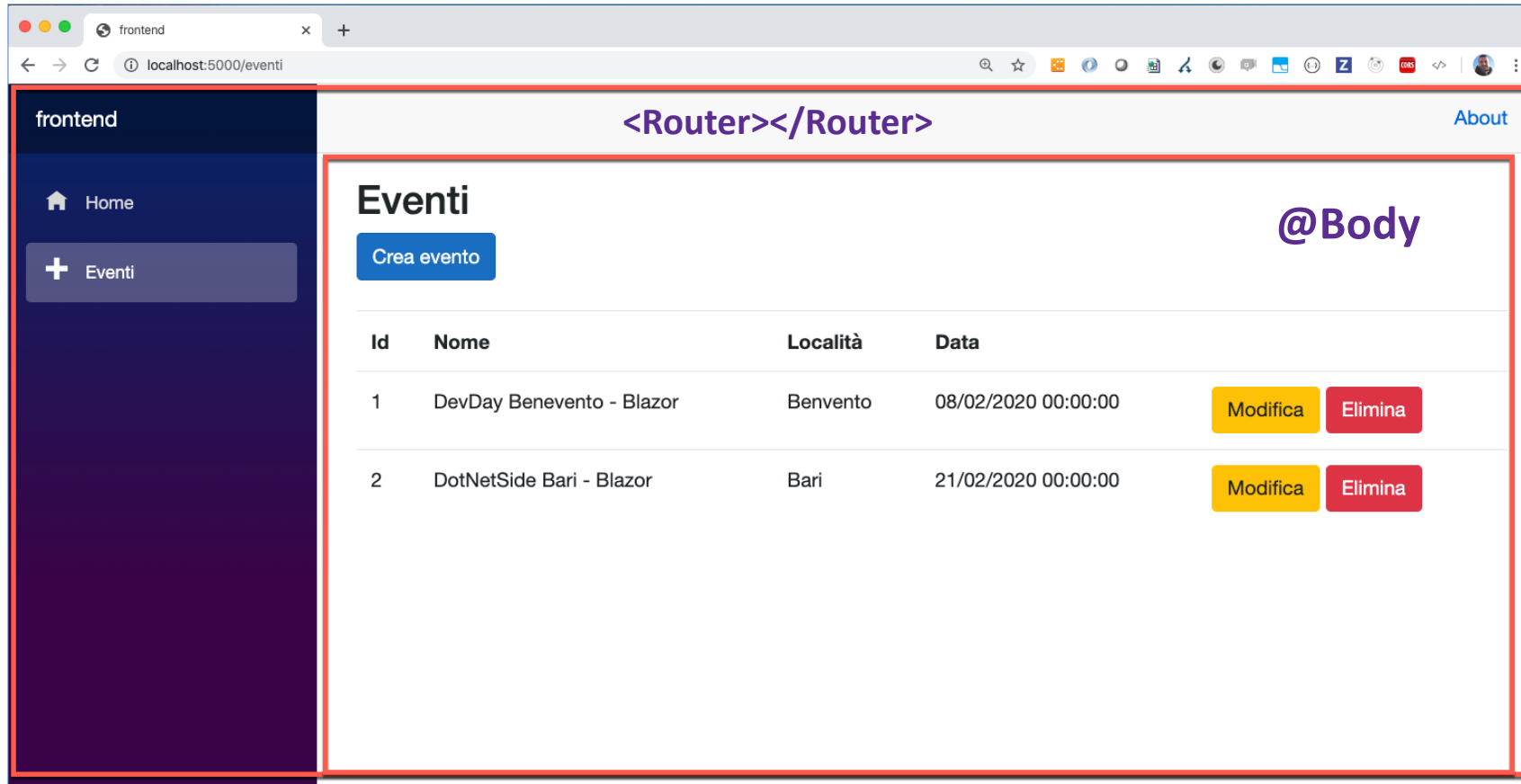
- Tecnicamente sono dei file **.razor** che contengono Markup e codice
- Il **nome del file** coincide con il **TAG** del componente

Blazor Pages



- Una pagina è un **componente** che definisce una destinazione per una **rotta**
- Contiene la direttiva **@page**

Blazor Routing



- Tutte le **richieste di navigazione** vengono gestite dal componente **Router**
- Ogni volta che viene richiesta una pagina il router la renderizza nel **@Body**

Gestione delle Form in Blazor

```
<EditForm Model="@DettaglioCorrente"
  OnValidSubmit="@e => OnSalva.InvokeAsync(DettaglioCorrente)">
  <DataAnnotationsValidator />

  <div class="form-group">
    <label for="nome">Nome:</label>
    <InputText id="nome" @bind-Value="DettaglioCorrente.Nome" class="form-control" />
    <ValidationMessage For="@(() => DettaglioCorrente.Nome)" />
  </div>

  <div class="form-group"> ...
  </div>

  <div class="form-group"> ...
  </div>

  <div class="form-group"> ...
  </div>

  <div class="form-group"> ...
  </div>

  <button type="submit" class="btn btn-primary">Salva</button>
  <button type="button" class="btn btn-warning" @onclick="OnAnnulla">Annulla</button>
</EditForm>
```

frontend

Home

+ Eventi

About

Evento

Nome:

Localita:

Nome:

Descrizione:

Note:

Salva Annulla

Gestione delle Form in Blazor

```
<EditForm Model="@DettaglioCorrente"
    OnValidSubmit="@(() => OnSalva.InvokeAsync(DettaglioCorrente))">
    <DataAnnotationsValidator />

    <div class="form-group">
        <label for="nome">Nome:</label>
        <InputText id="nome" @bind-Value="DettaglioCorrente.Nome" class="form-control" />
        <ValidationMessage For="@(() => DettaglioCorrente.Nome)" />
    </div>

    <div class="form-group"> ...
    </div>

    <div class="form-group"> ...
    </div>

    <div class="form-group"> ...
    </div>

    <div class="form-group"> ...
    </div>

    <button type="submit" class="btn btn-primary">Salva</button>
    <button type="button" class="btn btn-warning" @onclick="OnAnnulla">Annulla</button>
</EditForm>
```

frontend

Home

+ Eventi

Evento

Nome:

Il nome è obbligatorio!

Localita:

Benvento

Nome:

08/02/2020

Descrizione:

Note:

Salva Annulla

Gestione delle Form in Blazor

```
<EditForm Model="@DettaglioCorrente"
    OnValidSubmit="@(() => OnSalva InvokeAsync(DettaglioCorrente))">
    <DataAnnotationsValidator />

    <div class="form-group">
        <label for="nome">Nome:</label>
        <InputText id="nome" @bind-Value="DettaglioCorrente.Nome" class="form-control" />
        <ValidationMessage For="@(() => DettaglioCorrente.Nome)" />
    </div>

    <div class="form-group">...
    </div>

    <div class="form-group">...
    </div>

    <div class="form-group">...
    </div>

    <div class="form-group">...
    </div>

    <button type="submit" class="btn btn-prim
    <button type="button" class="btn btn-warn
</EditForm>
```

```
public class DettaglioEvento
{
    3 references
    public int Id { get; set; }

    [Required(ErrorMessage="Il nome è obbligatorio!")]
    9 references
    public string Nome { get; set; }

    [Required(ErrorMessage="La località è obbligatoria!")]
    8 references
    public string Localita { get; set; }

    [Required(ErrorMessage="La data è obbligatorio!")]
    8 references
    public DateTime Data { get; set; }

    5 references
    public string Descrizione { get; set; }

    5 references
    public string Note { get; set; }
}
```

frontend

Home

+ Eventi

About

Evento

Nome:

Il nome è obbligatorio!

Localita:

Benvento

Nome:

08/02/2020

Descrizione:

Note:

Salva Annulla

Da .NET a JavaScript

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
    ...
    <script src="js/jquery.min.js"></script>
    <script src="js/popper.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script>
        window.mostraConferma = (id) => {
            $('#'+id).modal('show');
        };
        window.nascondiConferma = (id) => {
            $('#'+id).modal('hide');
        };
    </script>
</body>
</html>
```

```
@inject IJSRuntime JSRuntime;

<h2>@Titolo</h2>
<button class="btn btn-primary mb-4" @onclick="OnCrea">Crea Evento</button>
<table class="table">
    ...
</table>
```

```
@code {
    ...

    private ElementoListaEventi elementoDaEliminare;

    private async Task MostraConferma(ElementoListaEventi elemento)
    {
        this.elementoDaEliminare = elemento;
        await JSRuntime.InvokeVoidAsync("mostraConferma", "eliminaEvento");
    }

    private async Task ConfermaEliminazione()
    {
        await OnElimina.InvokeAsync(this.elementoDaEliminare);
        await JSRuntime.InvokeVoidAsync("nascondiConferma", "eliminaEvento");
    }
}
```



Da JavaScript a .NET

```
public class MiaClasse
{
    [JSInvokable]
    public static string[] RecuperaListaStringhe()
    {
        return new string[] { "elemento1", "elemento2", "elemento3" };
    }

    [JSInvokable]
    public static Task<string[]> RecuperaListaStringheAsync()
    {
        return Task.FromResult(new string[] { "elemento1", "elemento2", "elemento3" } );
    }
}
```

```
window.chiamaDotNet = () => {
    var elementi = DotNet.invokeMethod('MyBlazorApp', 'RecuperaListaStringhe');

    DotNet.invokeMethodAsync('MyBlazorApp', 'RecuperaListaStringheAsync')
        .then(listastringhe => console.log(listastringhe));
}
```



Da JavaScript a .NET

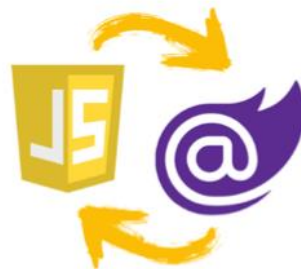
```
public class MiaClasse
{
    public async Task ChiamaJavaScript()
    {
        await JSRuntime.InvokeVoidAsync(
            "chiamaJavaScript",
            DotNetObjectReference.Create(this));
    }

    [JSInvokable]
    public string[] RecuperaListaStringhe()
    {
        return new string[] { "elemento1", "elemento2", "elemento3" };
    }

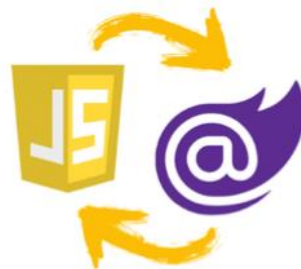
    [JSInvokable]
    public Task<string[]> RecuperaListaStringheAsync()
    {
        return Task.FromResult(new string[] { "elemento1", "elemento2", "elemento3" } );
    }
}
```

```
window.chiamaJavaScript = (oggetto) => {
    var elementi = oggetto.invokeMethod('RecuperaListaStringhe');

    oggetto.invokeMethodAsync('RecuperaListaStringheAsync')
        .then(listastringhe => console.log(listastringhe));
}
```



JavaScript Isolation



```
export function showPrompt(message) {  
    return prompt(message, 'Type anything here');  
}
```

```
@page "/call-js-example-6"
```

```
@implements IAsyncDisposable
```

```
@inject IJSRuntime JS
```

```
<h1>Call JS Example 6</h1>
```

```
<p>
```

```
    <button @onclick="TriggerPrompt">Trigger browser window prompt</button>
```

```
</p>
```

```
<p>
```

```
    @result
```

```
</p>
```

```
@code {
```

```
    private IJSObjectReference module;
```

```
    private string result;
```

```
    protected override async Task OnAfterRenderAsync(bool firstRender)
```

```
    {
```

```
        if (firstRender)
```

```
        {
```

```
            module = await JS.InvokeAsync<IJSObjectReference>("import",  
                "./scripts.js");
```

```
        }
```

```
    }
```

```
    private async Task TriggerPrompt()
```

```
    {
```

```
        result = await Prompt("Provide some text");
```

```
    }
```

```
    public async ValueTask<string> Prompt(string message)
```

```
    {
```

```
        return await module.InvokeAsync<string>("showPrompt", message);
```

```
    }
```

```
    async ValueTask IAsyncDisposable.DisposeAsync()
```

```
    {
```

```
        if (module is not null)
```

```
        {
```

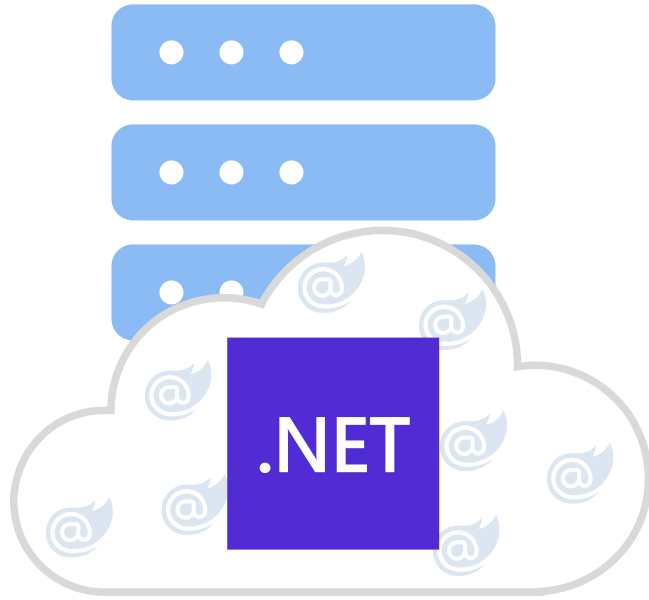
```
            await module.DisposeAsync();
```

```
        }
```

```
    }
```

```
}
```

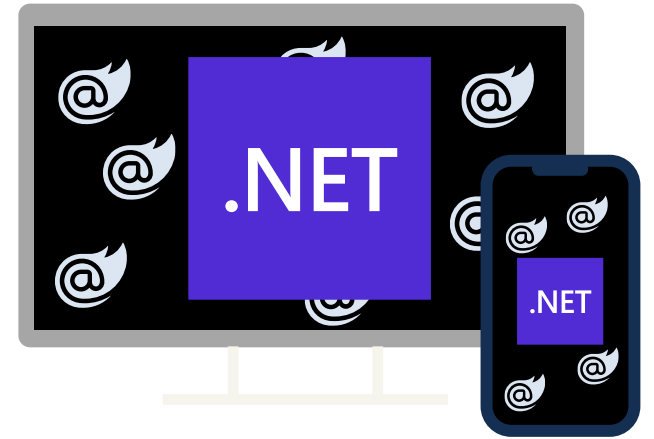
Blazor Hybrid (.NET 6)



**Blazor
Server**

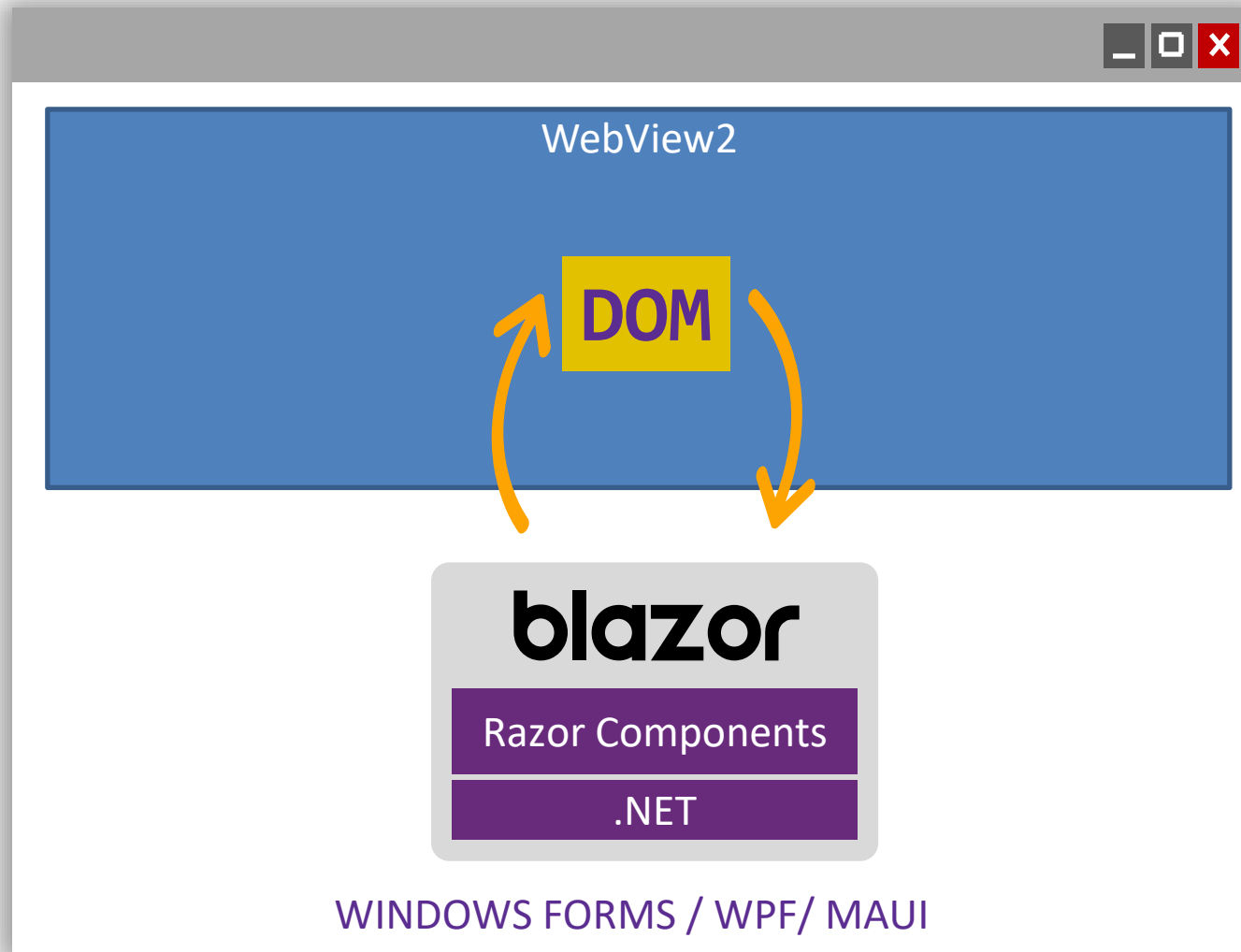


**Blazor
WebAssembly**

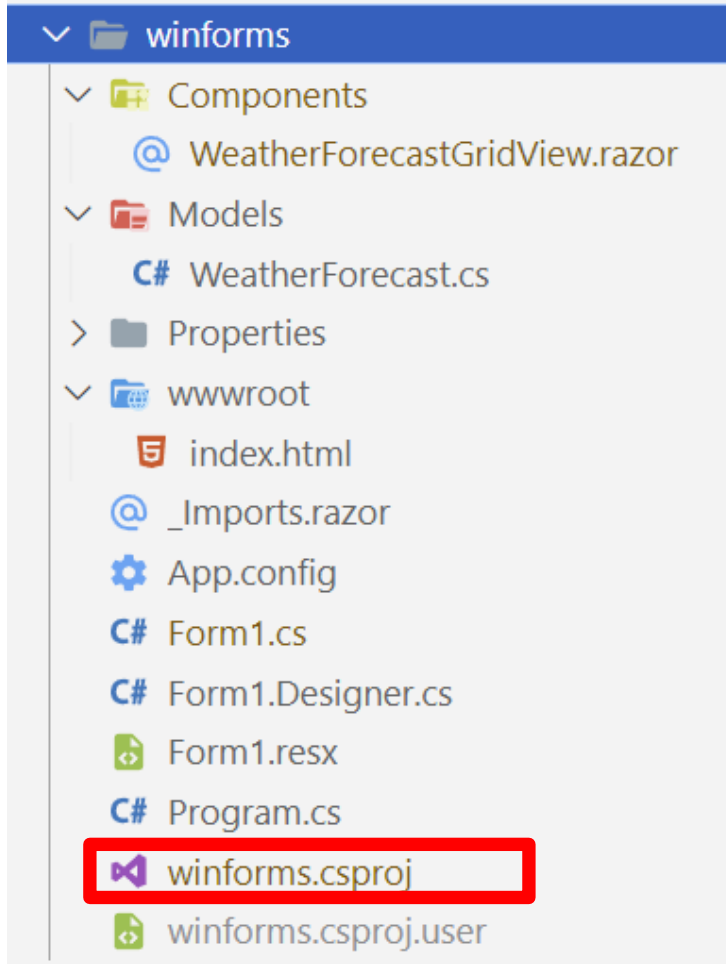


**Blazor
Hybrid**

Blazor Hybrid



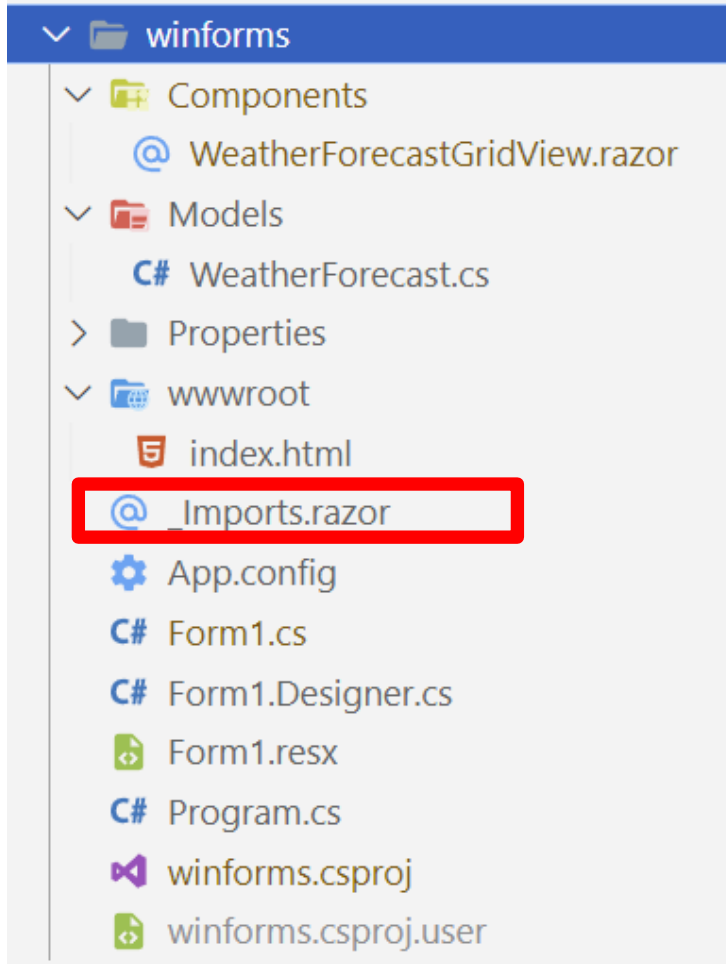
Blazor Hybrid in Windows Forms



Blazor Hybrid in Windows Forms

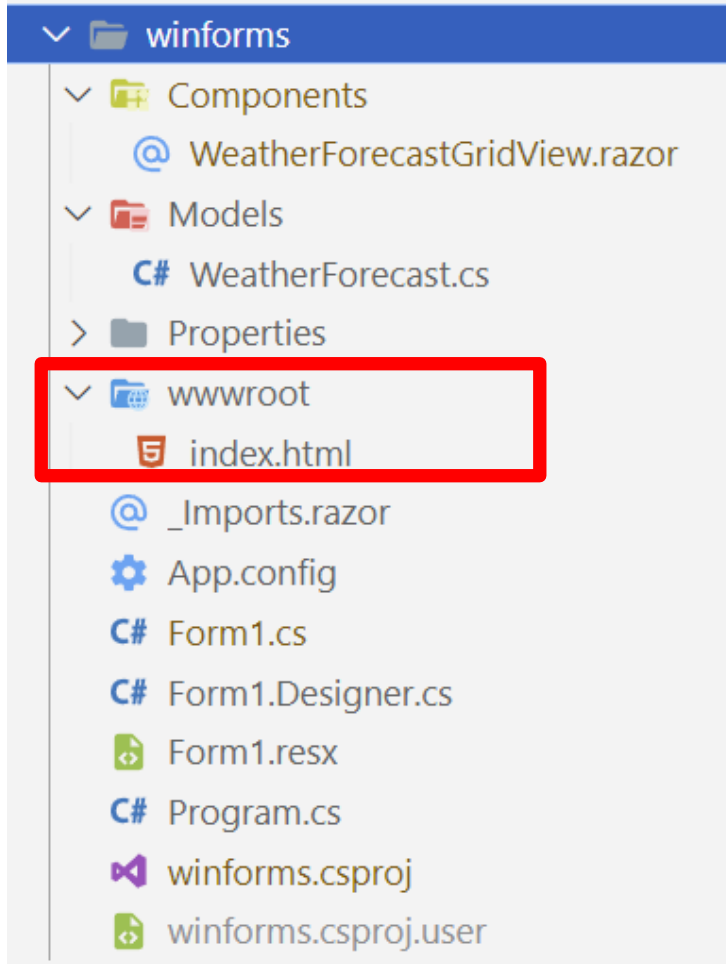
```
<Project Sdk="Microsoft.NET.Sdk.Razor">
  <PropertyGroup>
    <TargetFramework>net7.0-windows</TargetFramework>
    <OutputType>WinExe</OutputType>
    <GenerateAssemblyInfo>>false</GenerateAssemblyInfo>
    <UseWindowsForms>true</UseWindowsForms>
    <ImportWindowsDesktopTargets>true</ImportWindowsDesktopTargets>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Components.WebView.WindowsForms" Version="7.0.49" />
    <PackageReference Include="Microsoft.CSharp" Version="4.7.0" />
    <PackageReference Include="System.Data.DataSetExtensions" Version="4.5.0" />
    <PackageReference Include="Microsoft.DotNet.UpgradeAssistant.Extensions.Default.Analyzers" Version="0.4.355802">
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.Windows.Compatibility" Version="7.0.0" />
  </ItemGroup>
</Project>
```

Blazor Hybrid in Windows Forms



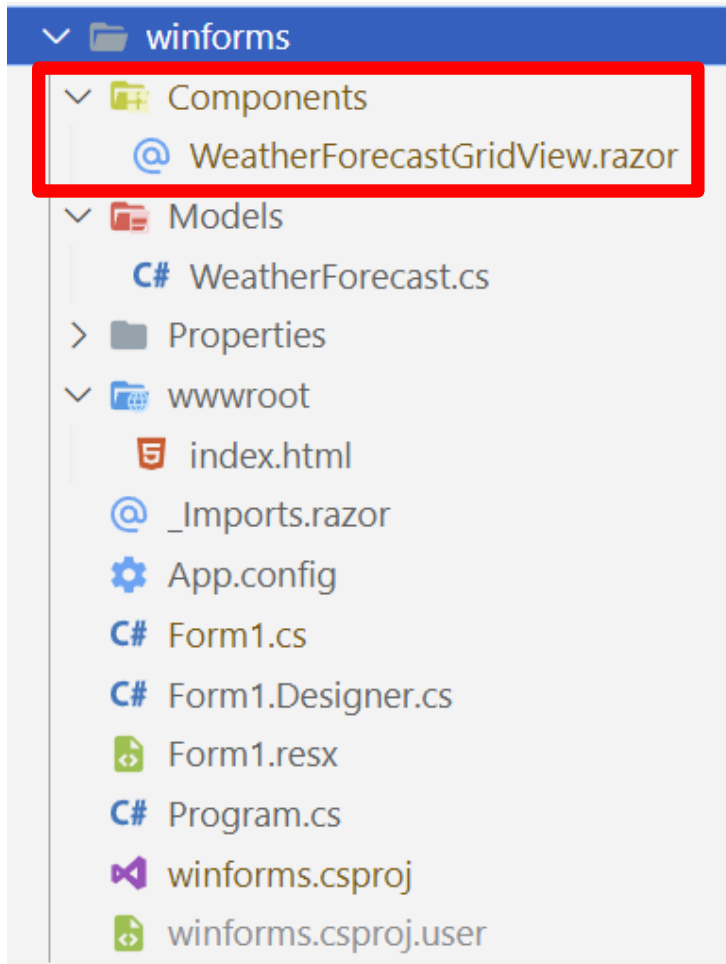
```
@using Microsoft.AspNetCore.Components.Web
```

Blazor Hybrid in Windows Forms



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>DotNetConf 2022</title>
</head>
<body>
  <div id="app"></div>
  <script src="_framework/blazor.webview.js"></script>
</body>
</html>
```

Blazor Hybrid in Windows Forms

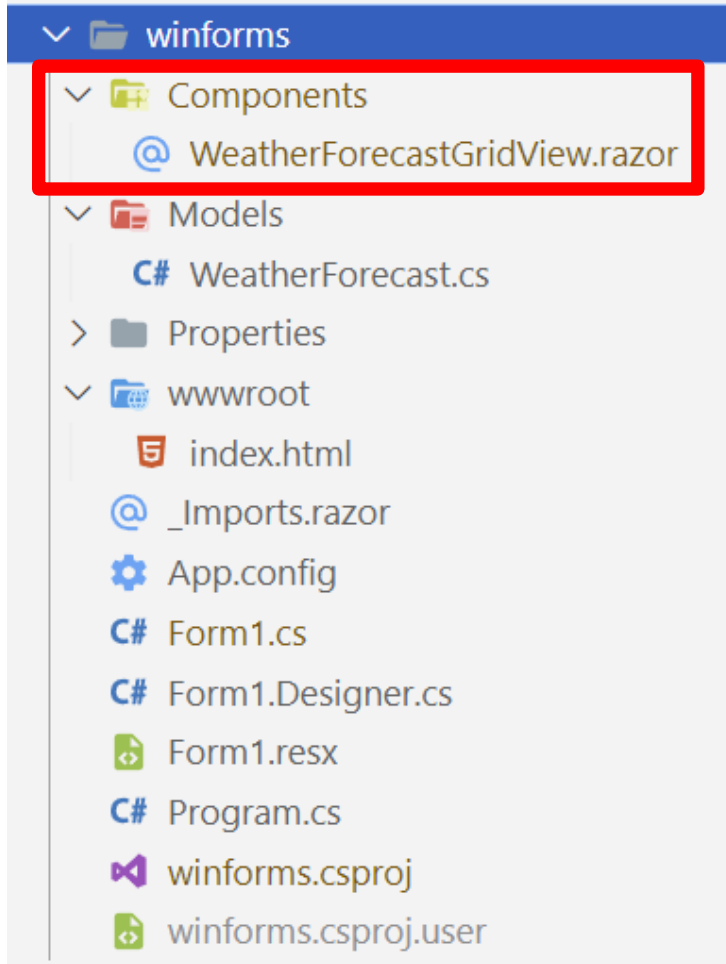


```
@using System.Windows.Forms
@using System.IO
@using System.Text
@using System.Runtime.Serialization.Json
@using winforms.Models

<button @onclick="LoadData">Load Data File</button>

@if (WeatherForecasts?.Count > 0)
{
    var props = WeatherForecasts[0].GetType().GetProperties();
    <table>
        <thead>
            <tr>
                @foreach (var prop in props)
                {
                    <th>@prop.Name</th>
                }
            </tr>
        </thead>
        @foreach (var item in WeatherForecasts)
        {
            <tr>
                @foreach (var prop in props)
                {
                    <td>@prop.GetValue(item)</td>
                }
            </tr>
        }
    </table>
}
else
{
    <span>No Data available</span>
}
```

Blazor Hybrid in Windows Forms

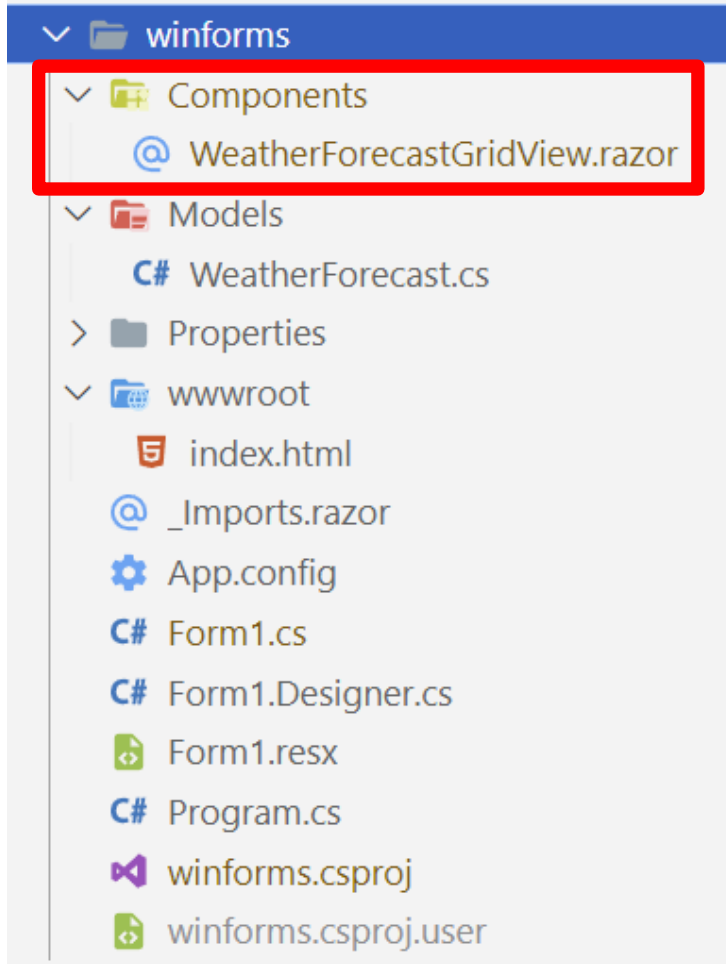


```
@using System.Windows.Forms
@using System.IO
@using System.Text
@using System.Runtime.Serialization.Json
@using winforms.Models

<button @onclick="LoadData">Load Data File</button>

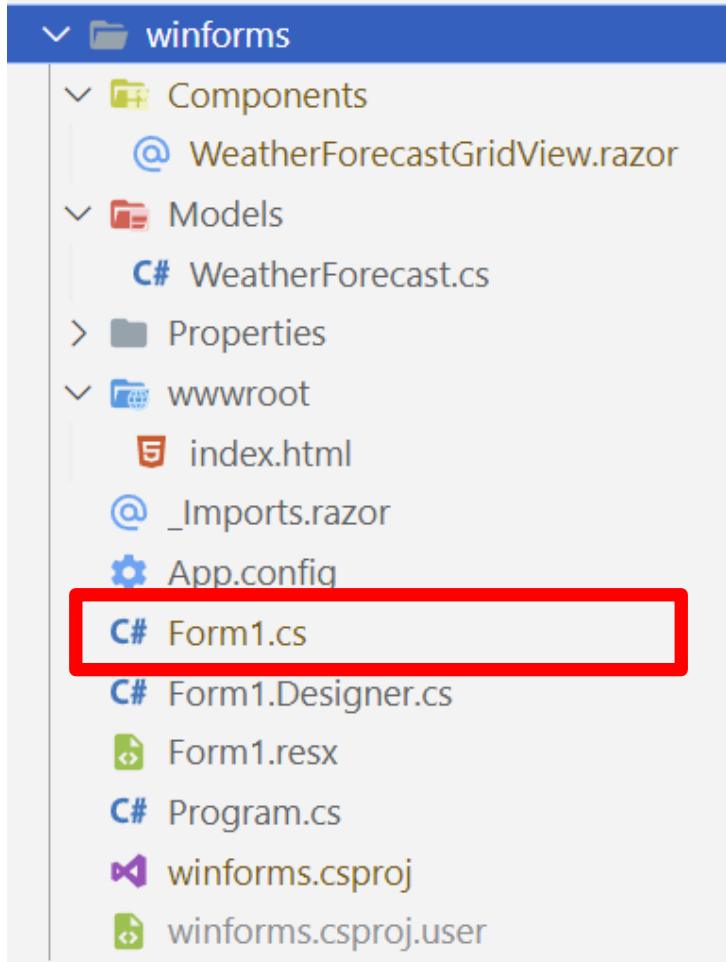
@if (WeatherForecasts?.Count > 0)
{
    var props = WeatherForecasts[0].GetType().GetProperties()
    <table>
        <thead>
            <tr>
                @foreach (var prop in props)
                {
```

Blazor Hybrid in Windows Forms



```
@code {  
  
    private List<WeatherForecast> WeatherForecasts { get; set; }  
  
    private void LoadData()  
    {  
        using (OpenFileDialog openFileDialog = new OpenFileDialog())  
        {  
            if (openFileDialog.ShowDialog() == DialogResult.OK)  
            {  
                string json = File.ReadAllText(openFileDialog.FileName);  
                using (var ms = new MemoryStream(Encoding.UTF8.GetBytes(json)))  
                {  
                    var ser = new DataContractJsonSerializer(typeof(List<WeatherForecast>));  
                    var data = ser.ReadObject(ms) as List<WeatherForecast>;  
                    WeatherForecasts = data;  
                    ms.Close();  
                }  
            }  
        }  
    }  
}
```

Blazor Hybrid in Windows Forms

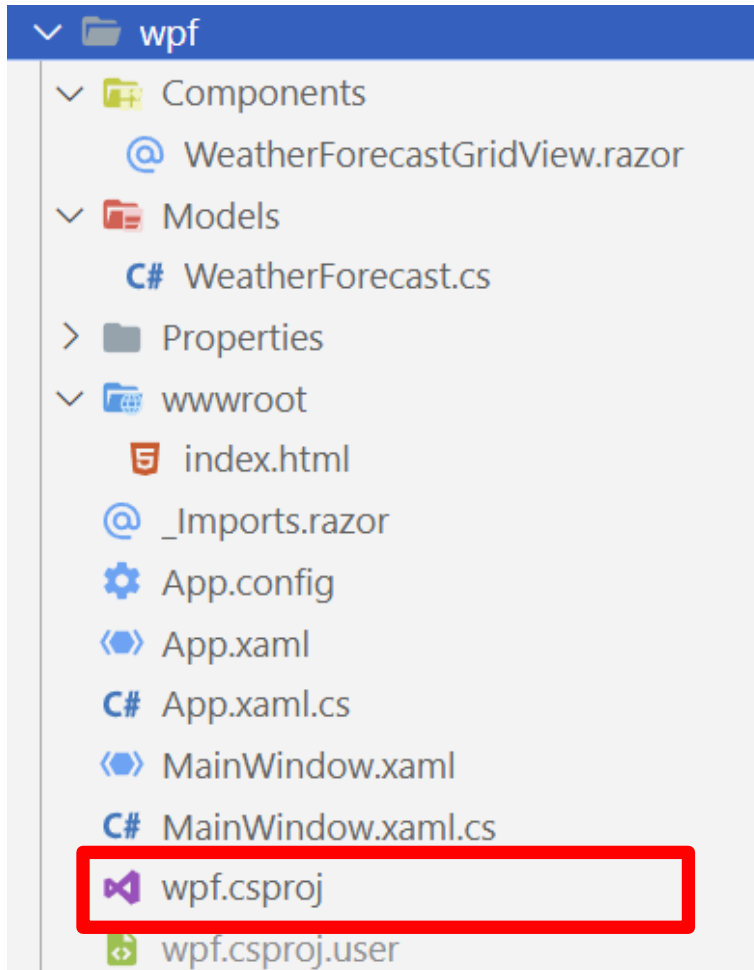


```
using System.Windows.Forms;
using winforms.Components;

namespace winforms
{
    public partial class frmDataVisualizer : Form
    {
        public frmDataVisualizer()
        {
            InitializeComponent();

            var serviceCollection = new ServiceCollection();
            serviceCollection.AddWindowsFormsBlazorWebView();
            blazorWebView1.HostPage = "wwwroot/index.html";
            blazorWebView1.Services = serviceCollection.BuildServiceProvider();
            blazorWebView1.RootComponents.Add<WeatherForecastGridView>("#app");
        }
    }
}
```

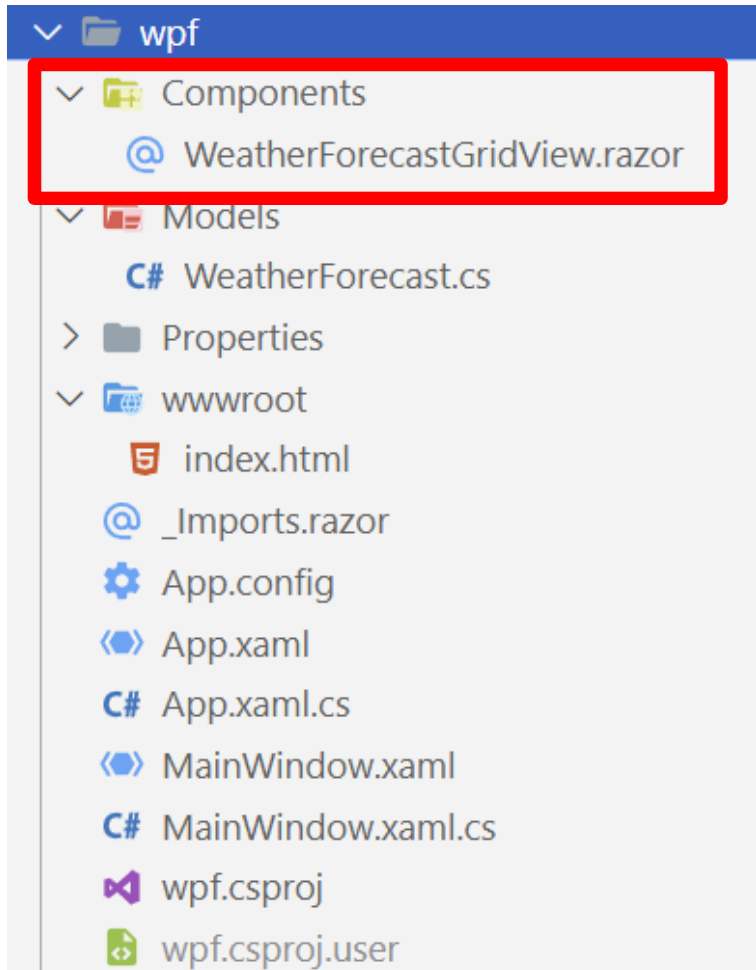
Blazor Hybrid in WPF



Blazor Hybrid in WPF

```
<Project Sdk="Microsoft.NET.Sdk.Razor">
  <PropertyGroup>
    <TargetFramework>net7.0-windows</TargetFramework>
    <OutputType>WinExe</OutputType>
    <GenerateAssemblyInfo>>false</GenerateAssemblyInfo>
    <UseWPF>true</UseWPF>
    <ImportWindowsDesktopTargets>true</ImportWindowsDesktopTargets>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Components.WebView.Wpf" Version="7.0.49" />
    <PackageReference Include="Microsoft.CSharp" Version="4.7.0" />
    <PackageReference Include="System.Data.DataSetExtensions" Version="4.5.0" />
    <PackageReference Include="Microsoft.DotNet.UpgradeAssistant.Extensions.Default.Analyzers" Version="0.4.355802">
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.Windows.Compatibility" Version="7.0.0" />
  </ItemGroup>
</Project>
```

Blazor Hybrid in WPF



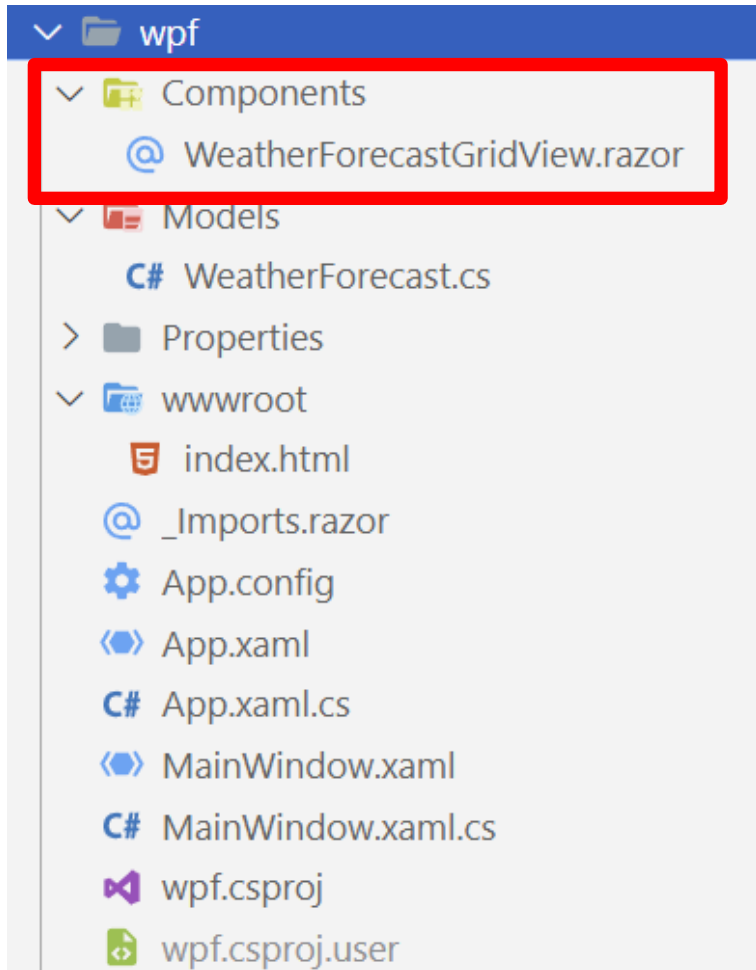
```
@using System.IO
@using System.Text
@using System.Runtime.Serialization.Json
@using Microsoft.Win32
@using wpf.Models

<button @onclick="LoadData">Load Data File</button>

@if (WeatherForecasts?.Count > 0)
{
    var props = WeatherForecasts[0].GetType().GetProperties();

    <table>
        <thead>
            <tr>
                @foreach (var prop in props)
                {
                    <th>@prop.Name</th>
                }
            </tr>
        </thead>
        @foreach (var item in WeatherForecasts)
        {
            <tr>
                @foreach (var prop in props)
                {
                    <td>@prop.GetValue(item)</td>
                }
            </tr>
        }
    </table>
}
else
{
    <span>No Data available</span>
}
```

Blazor Hybrid in WPF



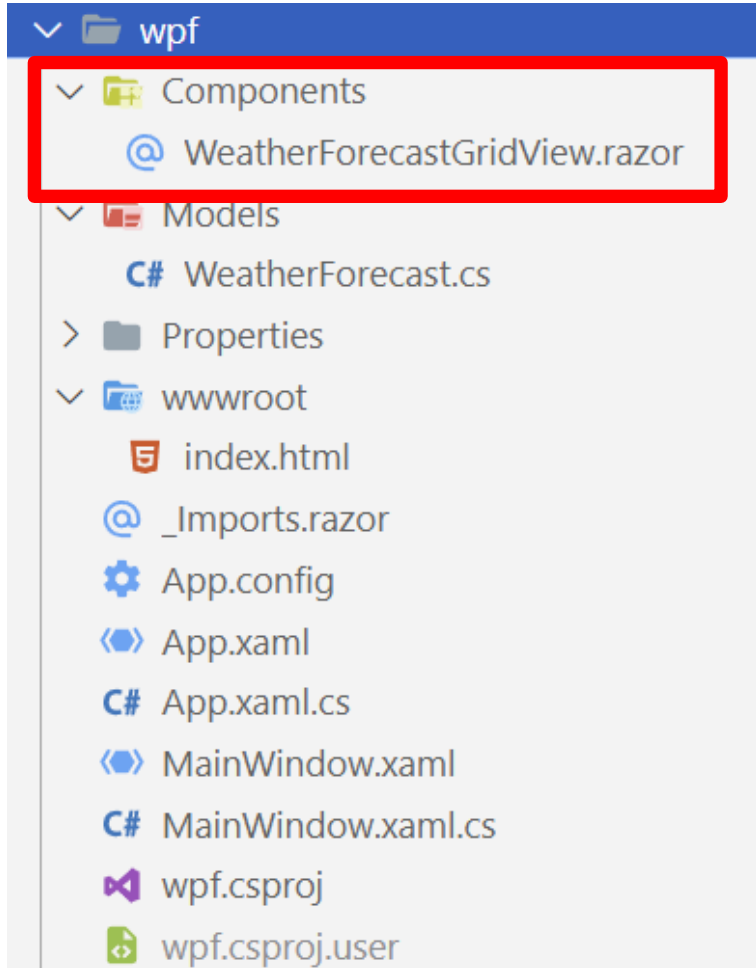
```
@using System.IO
@using System.Text
@using System.Runtime.Serialization.Json
@using Microsoft.Win32
@using wpf.Models

<button @onclick="LoadData">Load Data File</button>

@if (WeatherForecasts?.Count > 0)
{
    var props = WeatherForecasts[0].GetType().GetProperties()

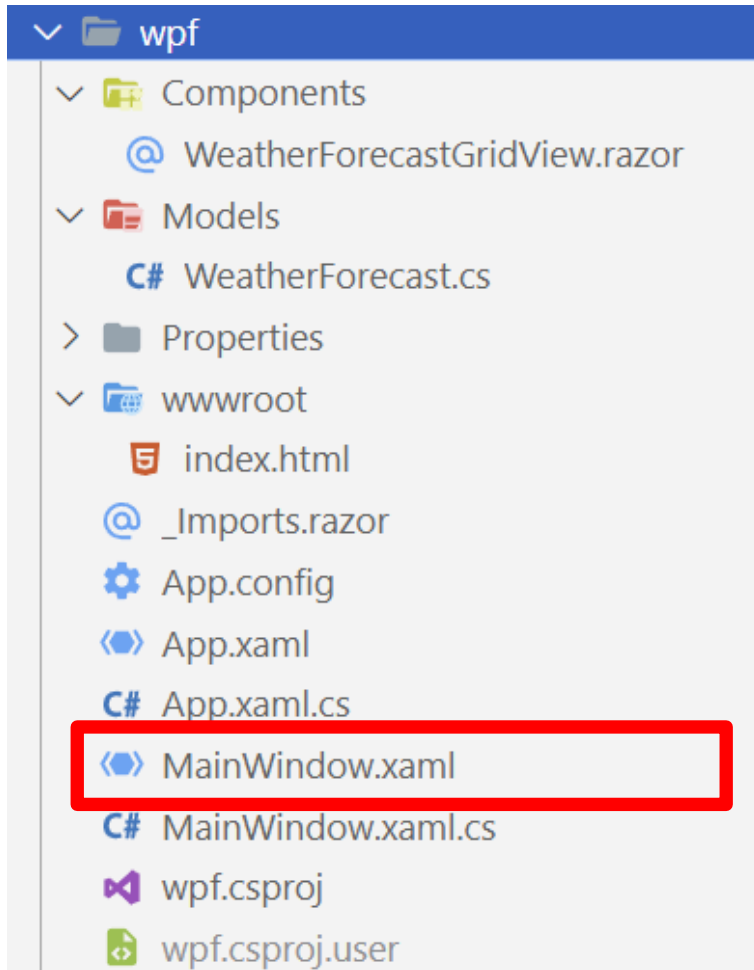
    <table>
        <thead>
            <tr>
                @foreach (var prop in props)
```

Blazor Hybrid in WPF



```
@code {  
  
    private List<WeatherForecast> WeatherForecasts { get; set; }  
  
    private void LoadData()  
    {  
        OpenFileDialog openFileDialog = new OpenFileDialog();  
        if (openFileDialog.ShowDialog() == true)  
        {  
            string json = File.ReadAllText(openFileDialog.FileName);  
            using (var ms = new MemoryStream(Encoding.UTF8.GetBytes(json)))  
            {  
                var ser = new DataContractJsonSerializer(typeof(List<WeatherForecast>));  
                var data = ser.ReadObject(ms) as List<WeatherForecast>;  
                WeatherForecasts = data;  
                ms.Close();  
            }  
        }  
    }  
}
```

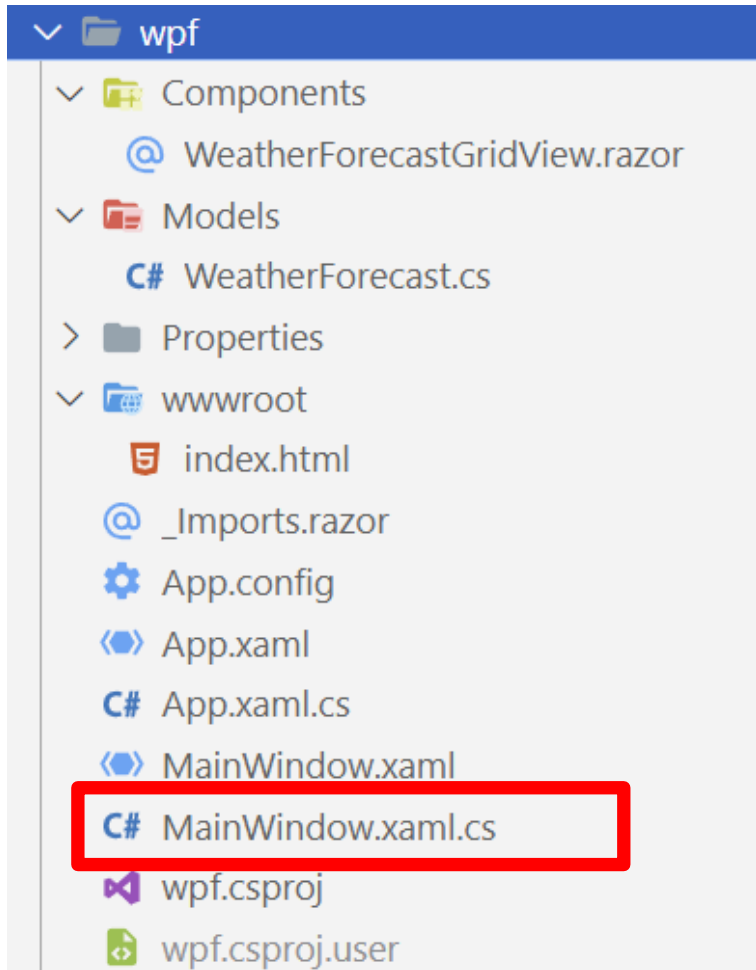
Blazor Hybrid in WPF



Blazor Hybrid in WPF

```
<Window x:Class="wpf.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local="clr-namespace:wpf"
  xmlns:components="clr-namespace:wpf.Components"
  xmlns:blazor="clr-namespace:Microsoft.AspNetCore.Components.WebView.Wpf;assembly=Microsoft.AspNetCore.Components.WebView.Wpf"
  mc:Ignorable="d"
  Title="Data Visualizer" Height="450" Width="800">
  <StackPanel>
    <blazor:BlazorWebView HostPage="wwwroot/index.html" Services="{StaticResource services}" MinHeight="200">
      <blazor:BlazorWebView.RootComponents>
        <blazor:RootComponent Selector="#app" ComponentType="{x:Type components:WeatherForecastGridView}" />
      </blazor:BlazorWebView.RootComponents>
    </blazor:BlazorWebView>
  </StackPanel>
</Window>
```

Blazor Hybrid in WPF

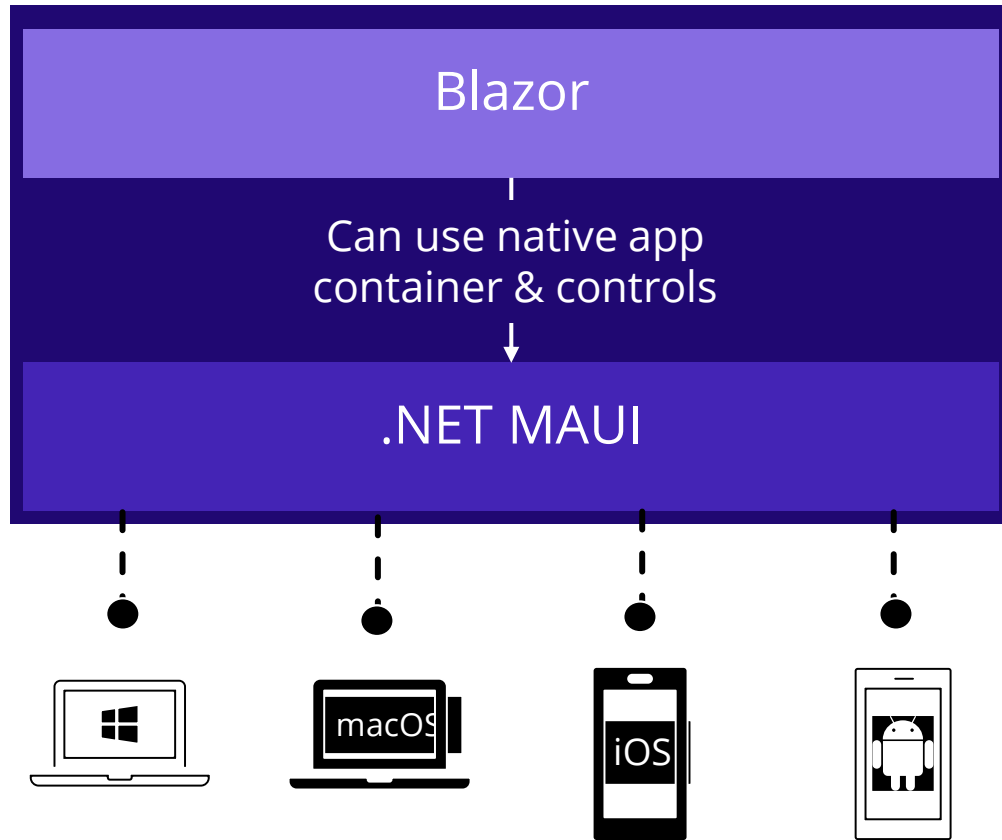


```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization.Json;
using System.Threading.Tasks;
using System.Windows;

namespace wpf
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            var serviceCollection = new ServiceCollection();
            serviceCollection.AddWpfBlazorWebView();
            Resources.Add("services", serviceCollection.BuildServiceProvider());

            InitializeComponent();
        }
    }
}
```

Blazor Hybrid for MAUI applications



Reuse UI components across native and web

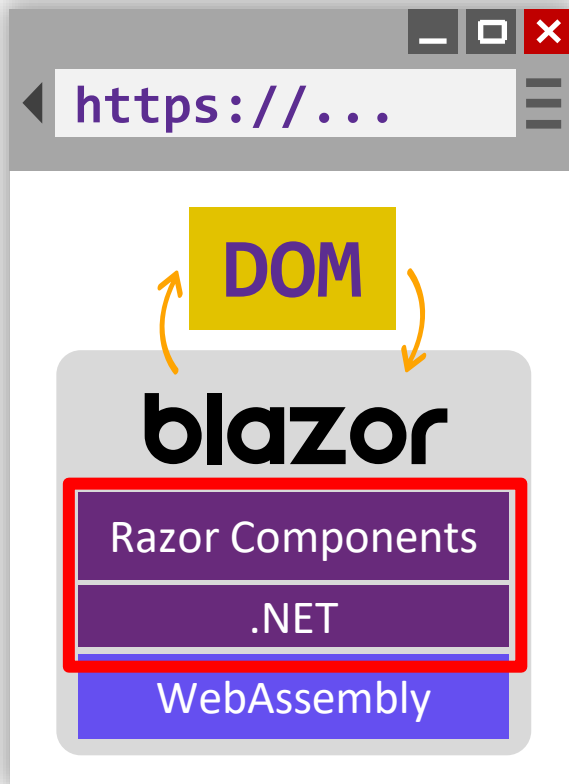
Built on top of .NET Multi-platform App UI

Native app container & embedded controls

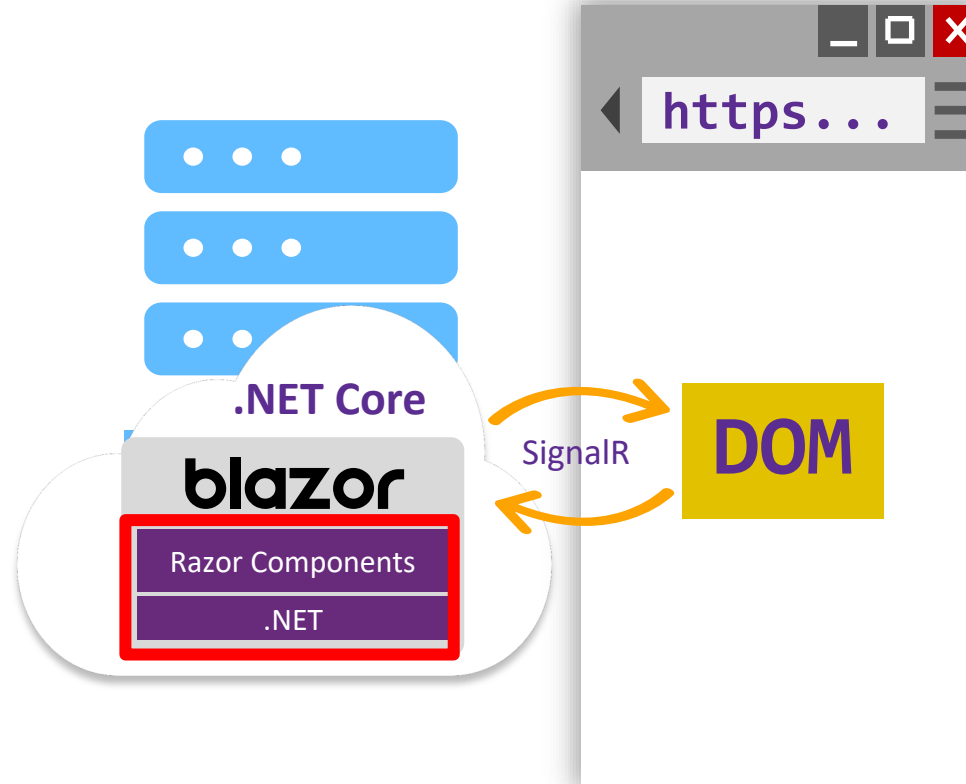
Generally available

Cosa hanno in comune?

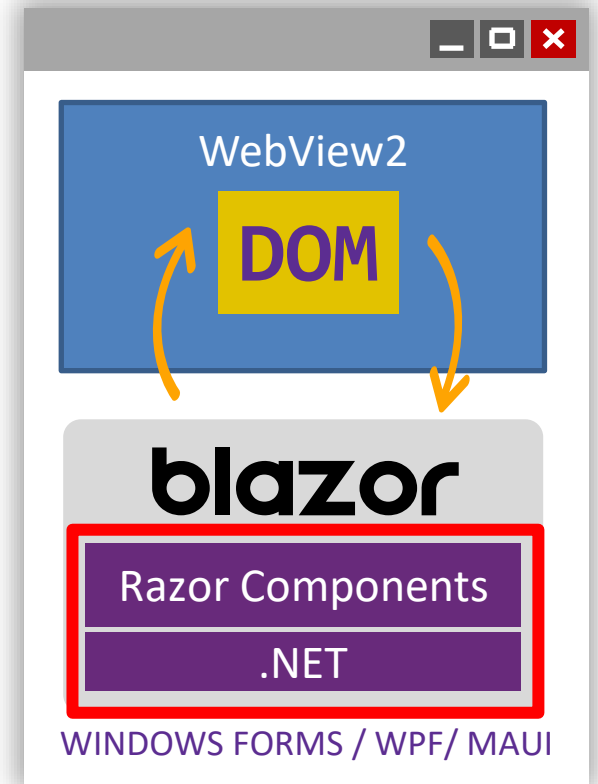
WebAssembly



Server



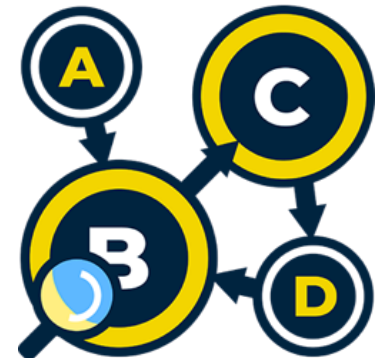
Hybrid



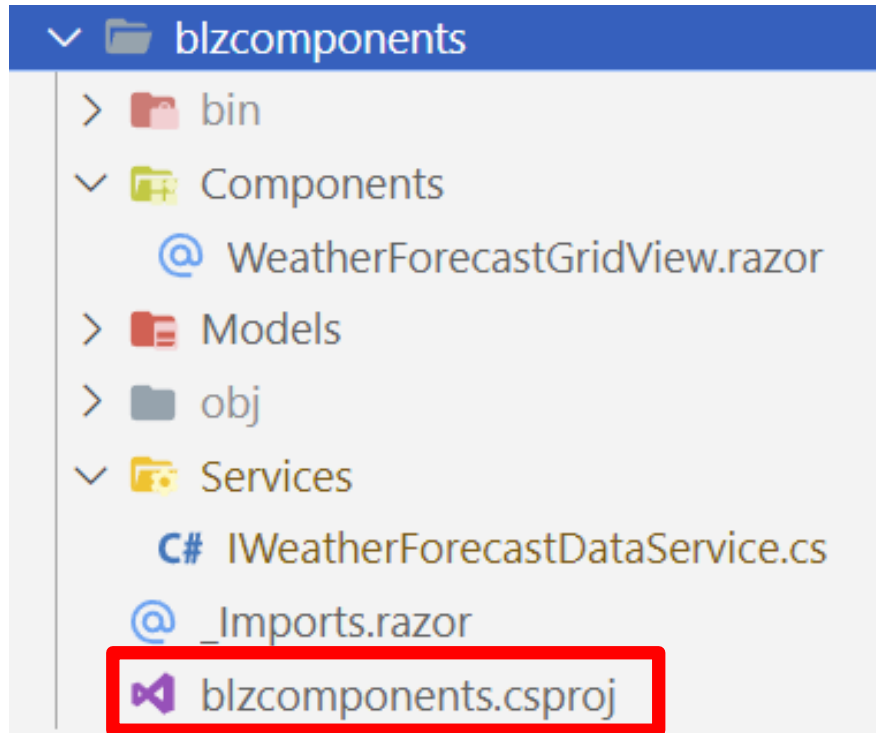
Dependency Injection in Blazor

In .NET Core viene fornito un **motore pluggabile** di DI, con una **implementazione di default**:

- Possiamo usarla per **disaccoppiare** i componenti dalle classi di servizio
- Possiamo iniettare una dipendenza con **@inject**
- Possiamo utilizzare un **singleton/scoped** come sessione applicativa (in-memory) o per far comunicare componenti



Blazor Hybrid con libreria condivisa



Blazor Hybrid con libreria condivisa

```
<Project Sdk="Microsoft.NET.Sdk.Razor">

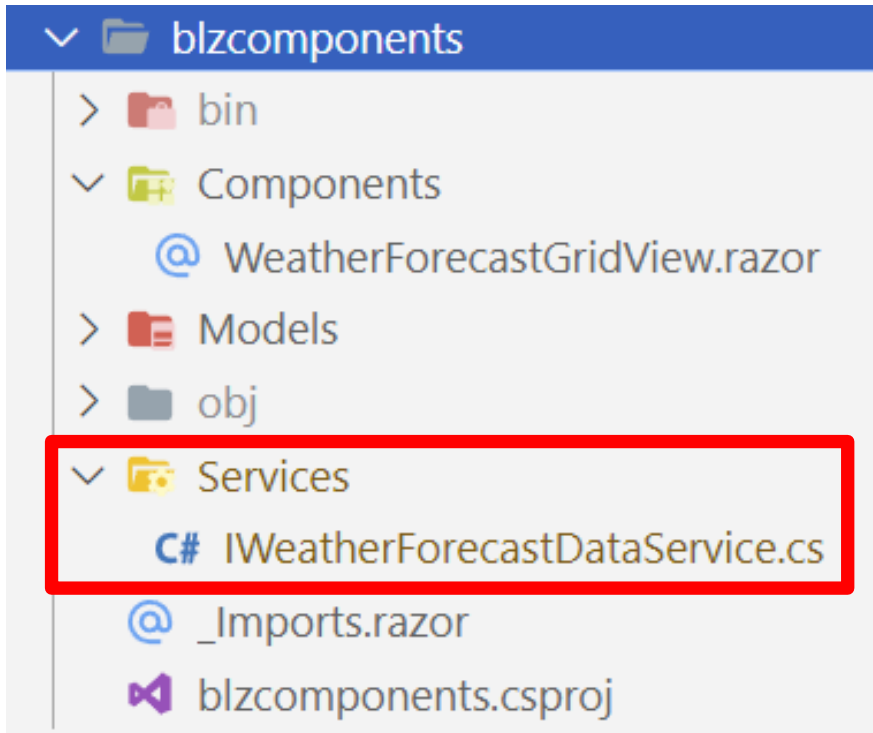
  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>disable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <SupportedPlatform Include="browser" />
  </ItemGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Components.Web" Version="7.0.0" />
  </ItemGroup>

</Project>
```

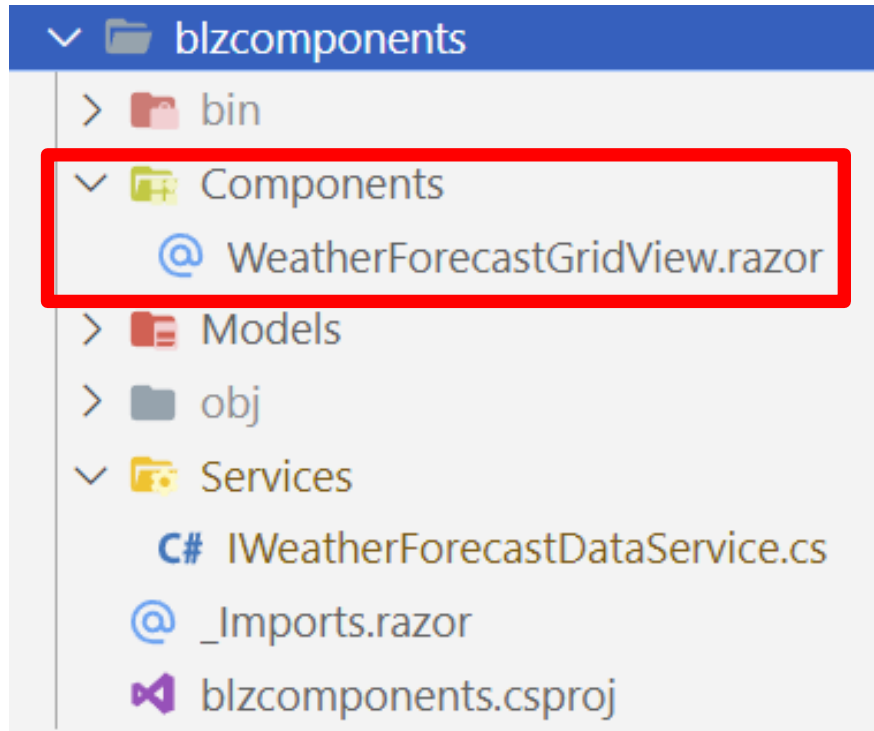
Blazor Hybrid con libreria condivisa



```
using blzcomponents.Models;

namespace blzcomponents.Services
{
    public interface IWeatherForecastDataService
    {
        Task<List<WeatherForecast>> GetWeatherForecastsAsync();
    }
}
```

Blazor Hybrid con libreria condivisa



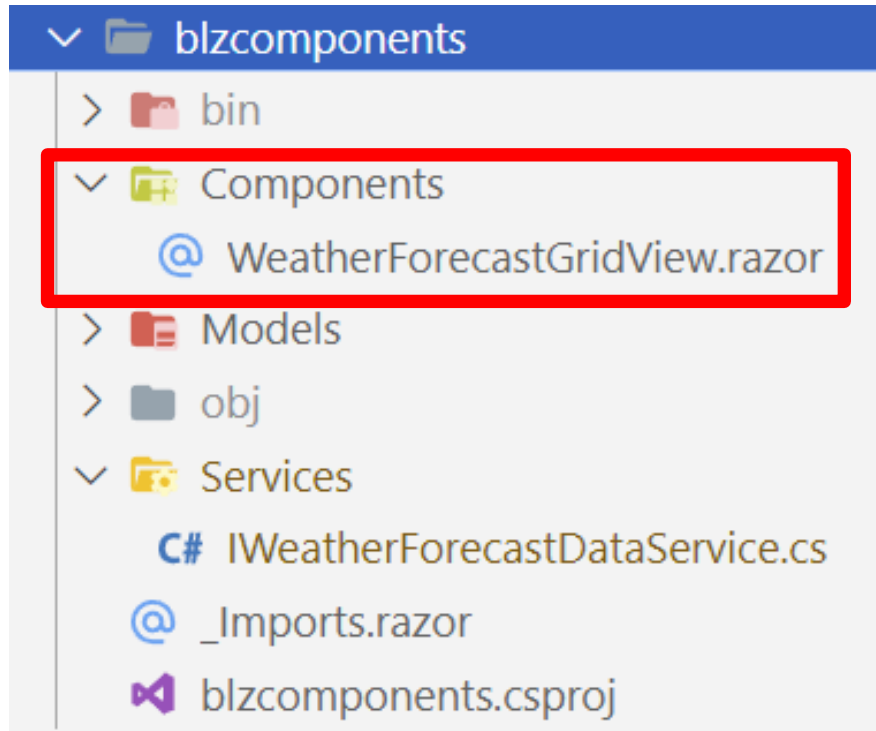
```
@using blzcomponents.Models
@using blzcomponents.Services
@Inject IWeatherForecastDataService dataService

<button @onclick="LoadData">Load Data File</button>

@if (WeatherForecasts?.Count > 0)
{
    var props = WeatherForecasts[0].GetType().GetProperties();

    <table>
        <thead>
            <tr>
                @foreach (var prop in props)
                {
                    <th>@prop.Name</th>
                }
            </tr>
        </thead>
        @foreach (var item in WeatherForecasts)
        {
            <tr>
                @foreach (var prop in props)
                {
                    <td>@prop.GetValue(item)</td>
                }
            </tr>
        }
    </table>
}
else
{
    <span>No Data available</span>
}
```

Blazor Hybrid con libreria condivisa



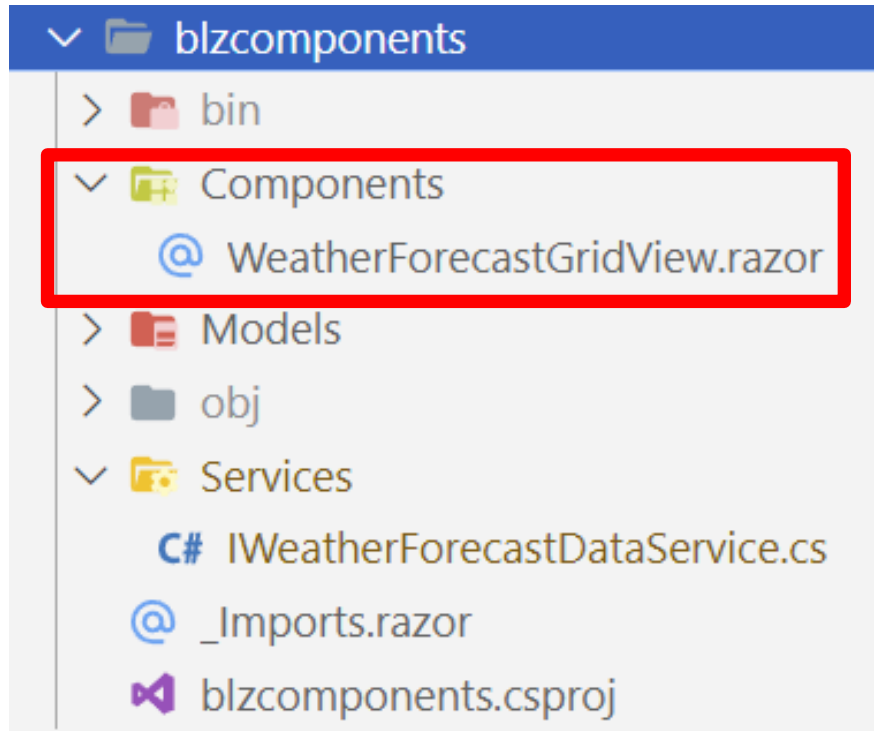
```
@using blzcomponents.Models
@using blzcomponents.Services
@Inject IWeatherForecastDataService dataService

<button @onclick="LoadData">Load Data File</button>

@if (WeatherForecasts?.Count > 0)
{
    var props = WeatherForecasts[0].GetType().GetProperties()

    <table>
        <thead>
            <tr>
                @foreach (var prop in props)
                {
                    <th>@prop.Name</th>
                }
            }
        }
    }
}
```

Blazor Hybrid con libreria condivisa



```
@code {  
  
    private List<WeatherForecast> WeatherForecasts { get; set; }  
  
    private async Task LoadData()  
    {  
        WeatherForecasts = await dataService.GetWeatherForecastsAsync();  
    }  
}
```

Blazor Hybrid con libreria condivisa

```
using bl
using bl
using Sy

namespace winfo
{
    public class
    {
        public
        {
            usi
            {
                publ
                {
                    }
                }
            }
        }
    }
}

namespace wpf.Services
{
    public class WeatherForecastDataService : IWeatherForecastDataService
    {
        public Task<List<WeatherForecast>> GetWeatherForecastsAsync()
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            if (openFileDialog.ShowDialog() == true)
            {
                string json = File.ReadAllText(openFileDialog.FileName);
                using (var ms = new MemoryStream(Encoding.UTF8.GetBytes(json)))
                {
                    var ser = new DataContractJsonSerializer(typeof(List<WeatherForecast>));
                    var data = ser.ReadObject(ms) as List<WeatherForecast>;
                    ms.Close();
                    return Task.FromResult(data);
                }
            }
            else return Task.FromResult(new List<WeatherForecast>());
        }
    }
}
```

Blazor Hybrid con libreria condivisa

```
public partial class frmDataVisualizer : Form
{
    public frmDataVisualizer()
    {
        InitializeComponent();

        var serviceCollection = new ServiceCollection();
        serviceCollection.AddWindowsFormsBlazorWebView();
        serviceCollection.AddSingleton<IWeatherForecastDataService, WeatherForecastDataService>();
        blazorWebView1.HostPage = "wwwroot/index.html";
        blazorWebView1.Services = serviceCollection.BuildServiceProvider();
        blazorWebView1.RootComponents.Add<WeatherForecastsGridView>("#app");
    }
}
```

Condivisione di oggetti tra Blazor e Progetto Host

- Quando indichiamo il componente da renderizzare, possiamo passare anche i suoi parametri
 - `blazorWebView1.RootComponents.Add<WeatherForecasts GridView>("#app", new Dictionary<string, object> { { "WeatherForecasts", _weatherForecasts } });`
- Ma se la collection cambia Blazor non se ne accorge, dobbiamo, al momento giusto, invocare esplicitamente l'aggiornamento della UI:
 - `StateHasChanged();`
- Come possiamo farlo?

Condivisione di oggetti tra Blazor e Progetto Host: soluzione 1

Creare un message broker con cui comunicare il cambiamento:

```
public class PropertyChangedNotifier
{
    public event EventHandler<string> PropertyChanged;

    public void NotifyPropertyChange(string property)
    {
        PropertyChanged?.Invoke(this, property);
    }
}
```

Condivisione di oggetti tra Blazor e Progetto Host: soluzione 1

```
@using winforms.Services;
@Inject PropertyChangedNotifier PropertyChangedNotifier

@if (WeatherForecasts?.Count > 0)
{ ...
}
else
{ ...
}

@code {
    [Parameter]
    public List<WeatherForecast> WeatherForecasts { get; set; }

    protected override void OnInitialized()
    {
        PropertyChangedNotifier.PropertyChanged += (s, p) =>
        {
            if (p == "WeatherForecasts") StateHasChanged();
        };
    }
}
```

Condivisione di oggetti tra Blazor e Progetto Host: soluzione 2

Condividere lo stato applicativo contenente le proprietà che cambiano:

```
public class AppState
{
    public event EventHandler AppStateChanged;

    public List<WeatherForecast> WeatherForecasts { get; private set; }

    public void SetWeatherForecasts(List<WeatherForecast> weatherForecasts)
    {
        WeatherForecasts = weatherForecasts;
        AppStateChanged?.Invoke(this, null);
    }
}
```

Condivisione di oggetti tra Blazor e Progetto Host: soluzione 2

```
@inject AppState AppState
```

```
@if (AppState.WeatherForecasts?.Count > 0)  
> { ...  
}  
else  
> { ...  
}
```

```
@code {
```

```
protected override void OnInitialized()
```

```
{
```

```
AppState.AppStateChanged += (s, e) => StateHasChanged();
```

```
}
```

```
}
```

Blazor in .NET 7

- .NET Hot reload (improvements)
- WebAssembly AOT (improvements)
- Web crypto
- Blazor Custom Elements
- Bind get/set/after modifier
- Virtualization (improvements)
- Hystory entry state
- Loading Page
- Empty Template
- Authentication (improvements)
- JS Interop (improvements)
- Cancel navigation events
- Image component
- CSS Hot Reload in Blazor Hybrid
- Multithreading (preview)
- .NET WASM without UI (preview)

Blazor in .NET 7

- .NET Hot reload (improvements)
- WebAssembly AOT (improvements)
- Web crypto
- **Blazor Custom Elements**
- Bind get/set/after modifier
- Virtualization (improvements)
- Hystory entry state
- Loading Page
- Empty Template
- Authentication (improvements)
- JS Interop (improvements)
- Cancel navigation events
- Image component
- CSS Hot Reload in Blazor Hybrid
- **Multithreading (preview)**
- **.NET WASM without UI (preview)**

Blazor in .NET 8

- Authentication (improvements)
- JS Interop (improvements)
- .NET Hot reload (improvements)
- **WebAssembly AOT mixed mode**
- **Multithreading**
- QuickGrid
- **Blazor United**

Approfondimenti:

- Blazor .NET 8 Planning: <https://www.youtube.com/watch?v=QVxusemLoo>
- Exploring new and experimental features in Blazor: https://www.youtube.com/watch?v=fQ1Bg-u_YLw
- Blazor United in .NET 8: https://www.youtube.com/watch?v=klH_Py8ytIY
- Blazor United prototype: https://www.youtube.com/watch?v=48G_CEGXZZM

Takeaways

- Ha senso utilizzare **Blazor** se già conosco **.NET framework**
- Possiamo **condividere componenti di UI tra diversi client**, usando **Blazor Hybrid** e la potenza della **Dependency Injection**
- Blazor non è **Silverlight**, Microsoft ci sta investendo tantissimo e vale la pena dargli una possibilità

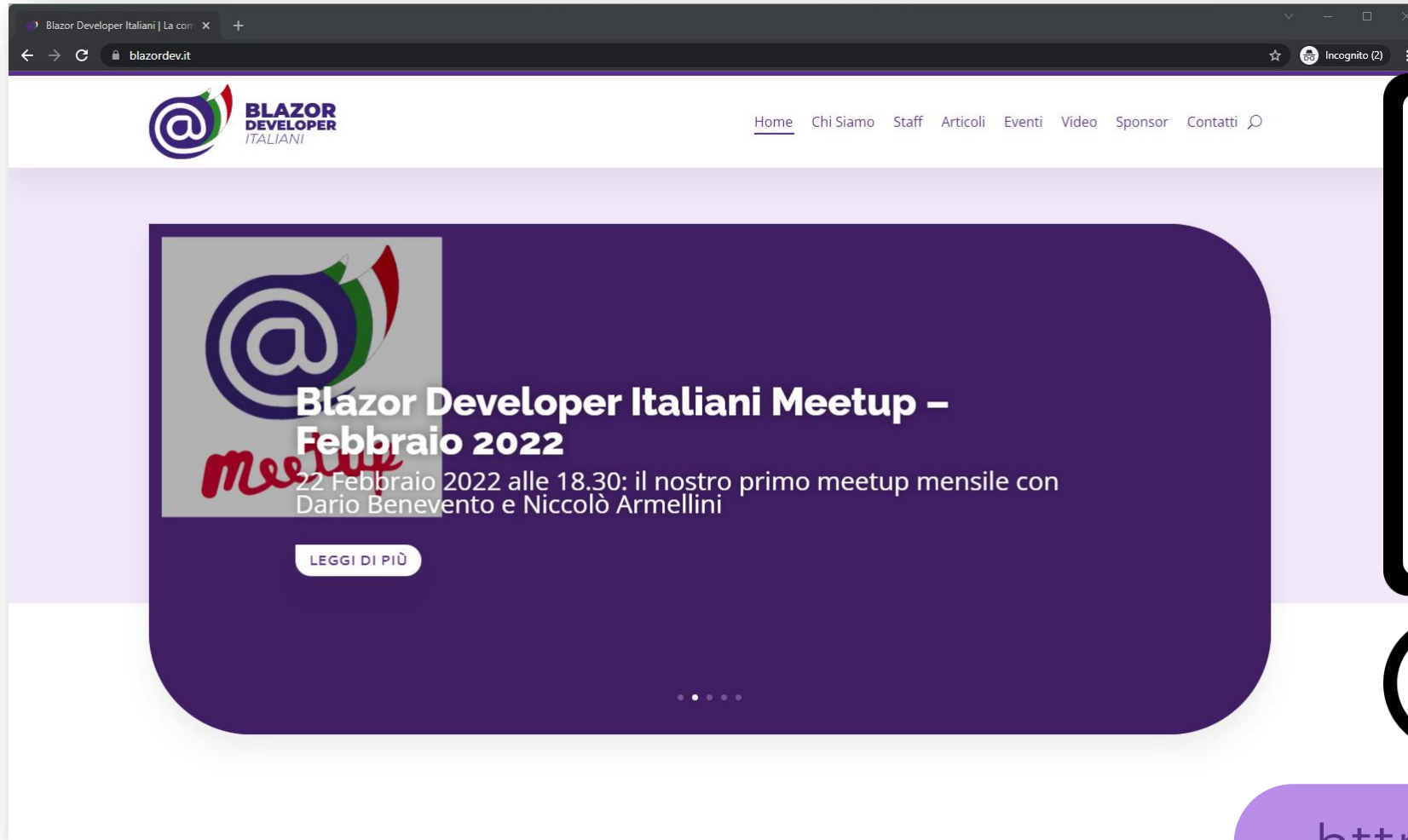


Interessati a Blazor?



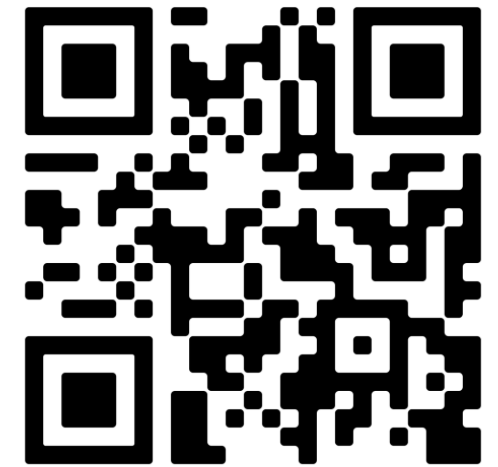
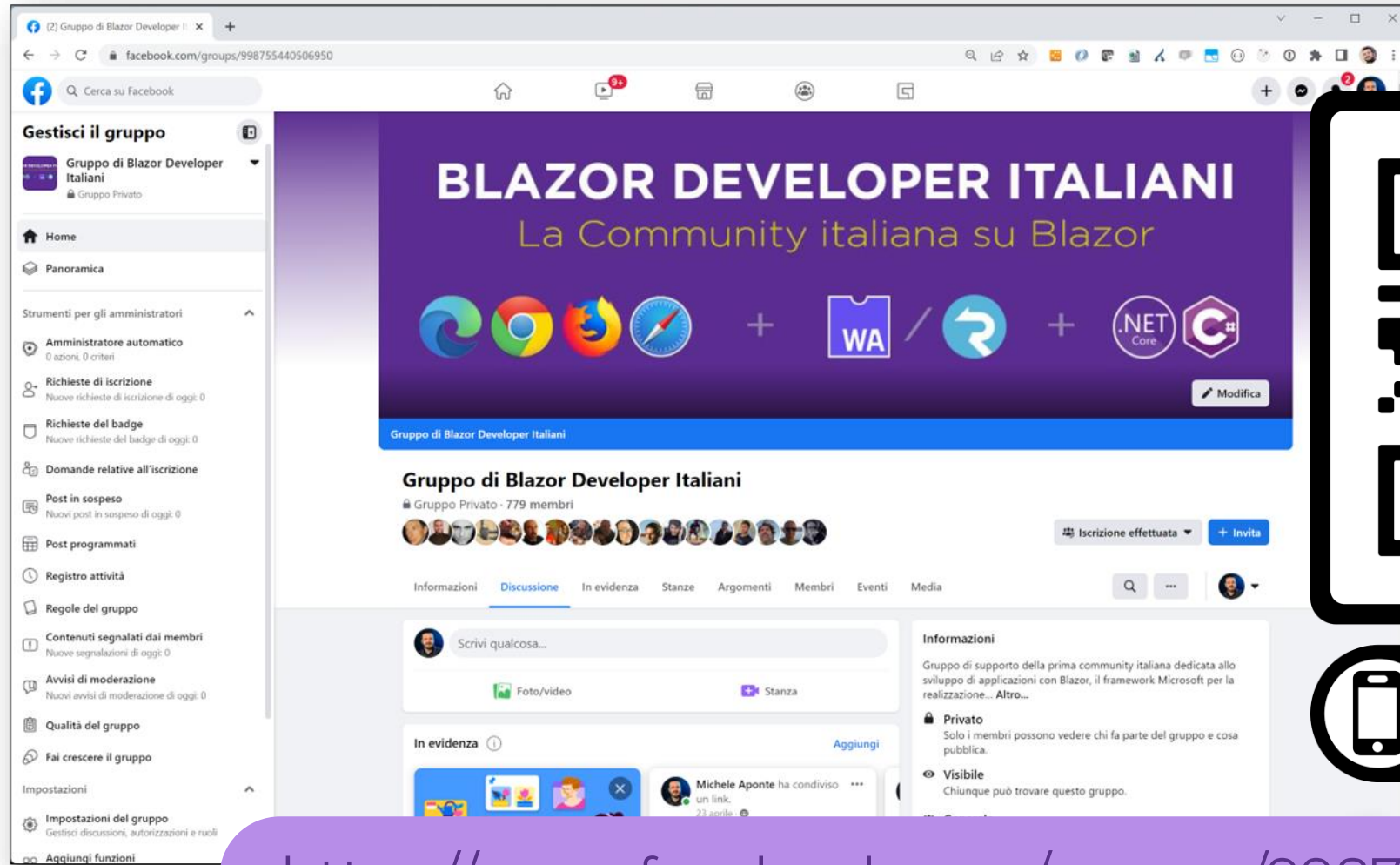
Blazor Developer Italiani è la prima **community italiana** dedicata allo sviluppo di applicazioni con **Blazor**

Dove ci incontriamo?



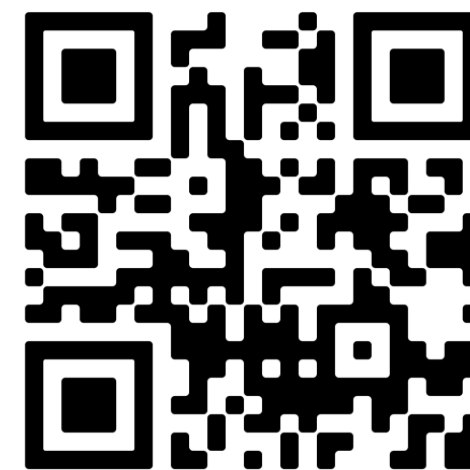
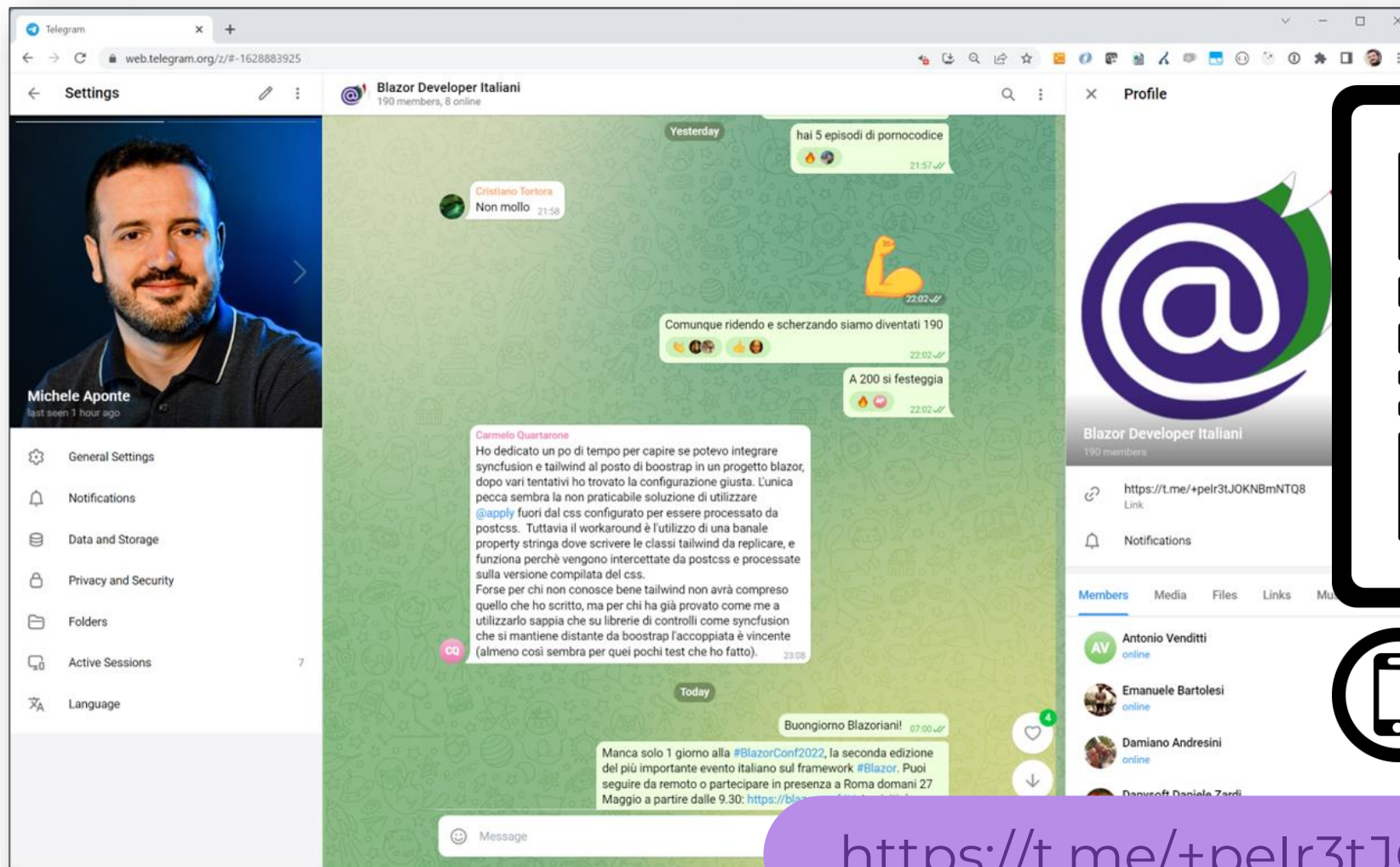
<https://blazordev.it>

Dove ci incontriamo?



<https://www.facebook.com/groups/998755440506950>

Dove ci incontriamo?



<https://t.me/+peIr3tJOKNBmNTQ8>

Dove seguirci?

<https://twitter.com/blazordevita>



<https://fb.me/blazordeveloperitaliani>



<https://www.linkedin.com/company/blazor-developer-italiani>



<https://youtu.be/c/BlazorDeveloperItaliani>



<https://www.twitch.tv/blazordevita>



<https://github.com/blazordevita>





More Info
www.blazorconf.it



Blazor Conf 2023



26 Maggio 2023



09:00



SCAN ME

Blazor: dalla teoria alla pratica

<https://www.improove.tech>



 SCAN ME



Michele Aponte



improove

Domande? Dubbi? Perplexità?



Michele Aponte

CEO&CTO @ Ellycode / CEO @ Blexin / CTO @ Improove Skills

Founder @ Blazor Developer Italiani / MVP Developer Technologies & AI



apomic80



apomic80



apomic80



apomic80



apomic80

WEB  DAY 2023

MILANO 16 MARZO

GRAZIE