Blazor Web Apps

Rajesh Kolla

Full-stack development ,Azure Architect

Email: rajesh.kolla01@outlook.com

Twitter: @RajeshKolla18

LinkedIn: https://be.linkedin.com/in/razeshkolla

Agenda

- Overview
- What is Blazor?
- Overview of Blazor Hosting Model, Components , Bindings , Parameters
- Dependency Injection
- JS Interoperability
- Demos
- Wrap up
- Q&A

What is Blazor? Why do need Blazor

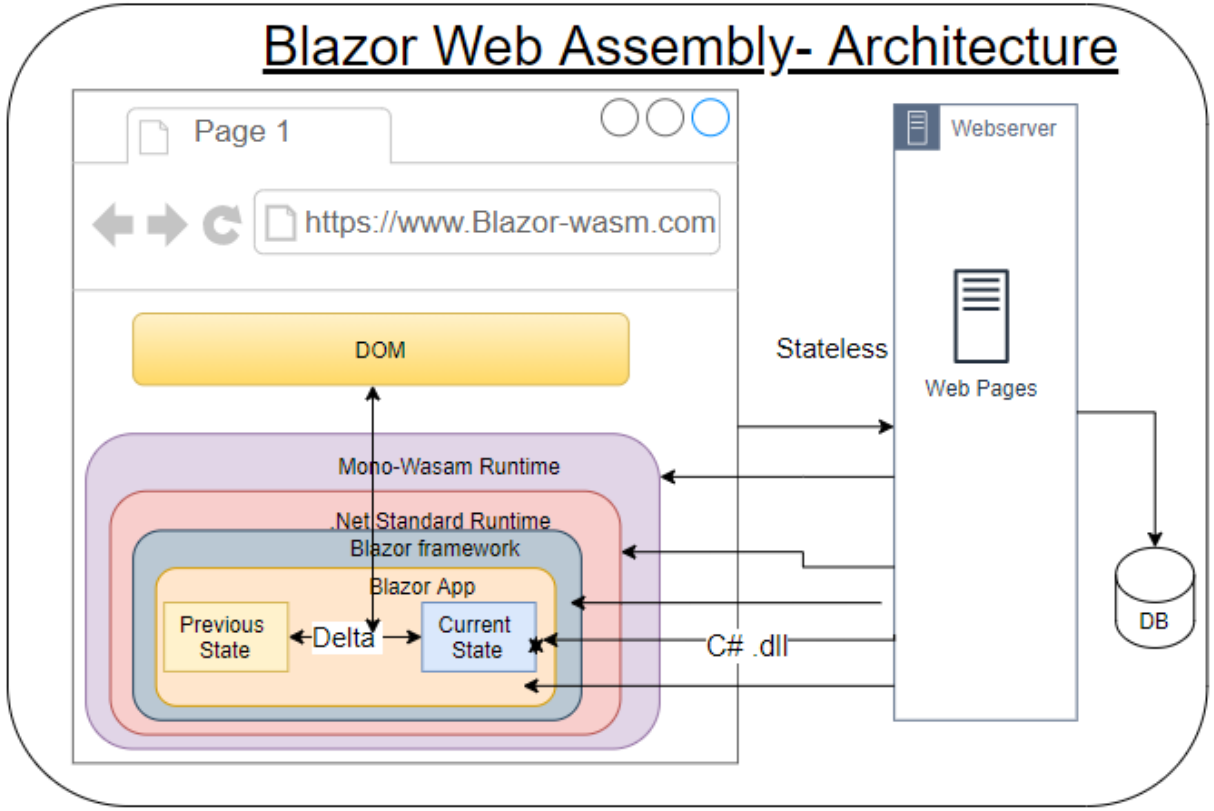
- open-source Single Page Application development framework for building interactive client-side web UI with .NET
- Build rich UI using C# instead of Java script framework
- Capable of executing views on the client and server as well.
- Share server side and client-side app logic in .NET
- No additional plug-in installed on Browser to run Blazor App.
- Blazor is capability of running in server and client side.
- Blazor mobile bindings to build native mobile apps
- C# developer become a Full stack development

Blazor Hosting model

What is Web Assembly:-

Web Assembly is an instruction set (CIL) that is formatted in specific binary format to run on any host.

- doesn’t require .NET to be installed on client to run through web assembly.
- supported by all latest browsers.



Web Assembly runs on the client in side browser and download files as static files . it will not work on local file system due to security reason.

Pros

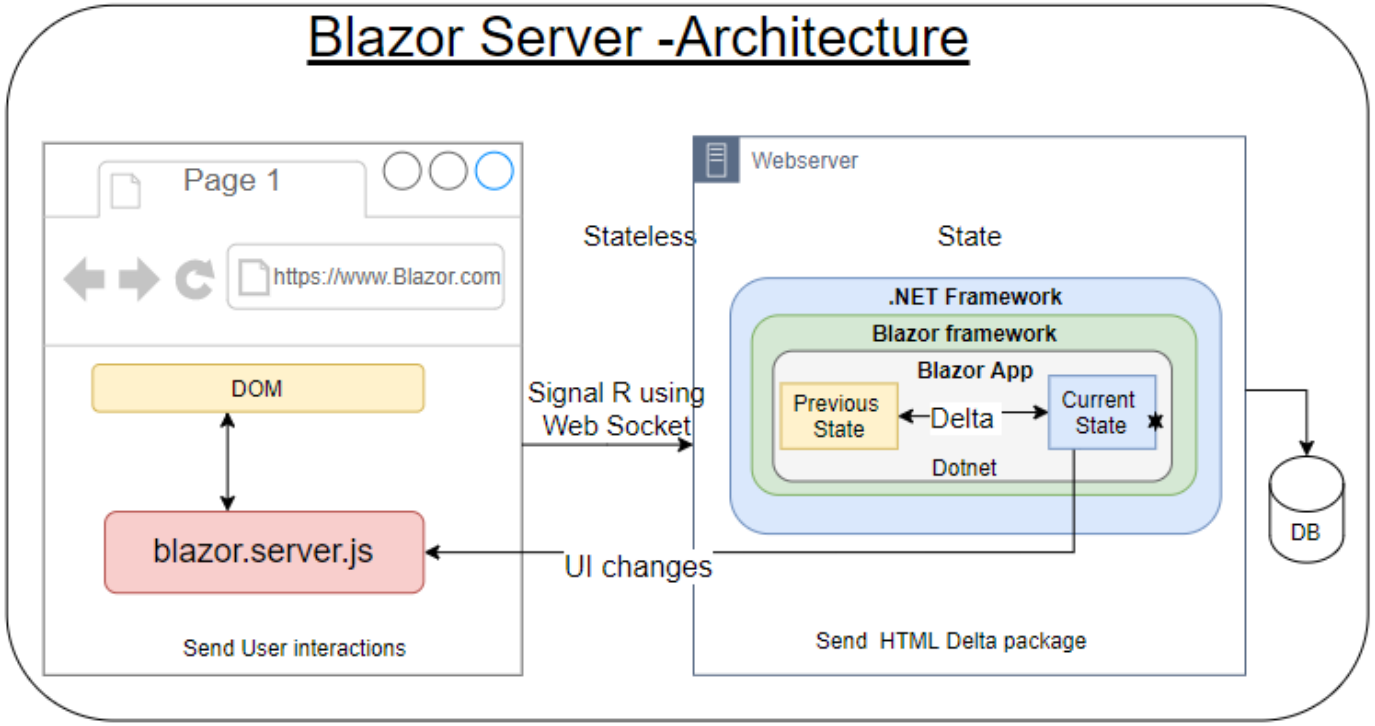
- Fully utilize the Client Resources
- latest version
- Suitable for off-line work loads
- Native speed
- No server side latency
- It can also quite easily run as a Progressive Web App,

Cons

- Limited to Modern browsers
- Initial respond payload size is high
- support multi threading
- all processing happened on UI thread asynchronously so doesn't block UI responsiveness as

Blazor Hosting model

2. Blazor –Server



Pros

- Initial request payload size is small compare to Blazer-Wasm
- Fast development
- Support all Browsers
- Large Security sandbox

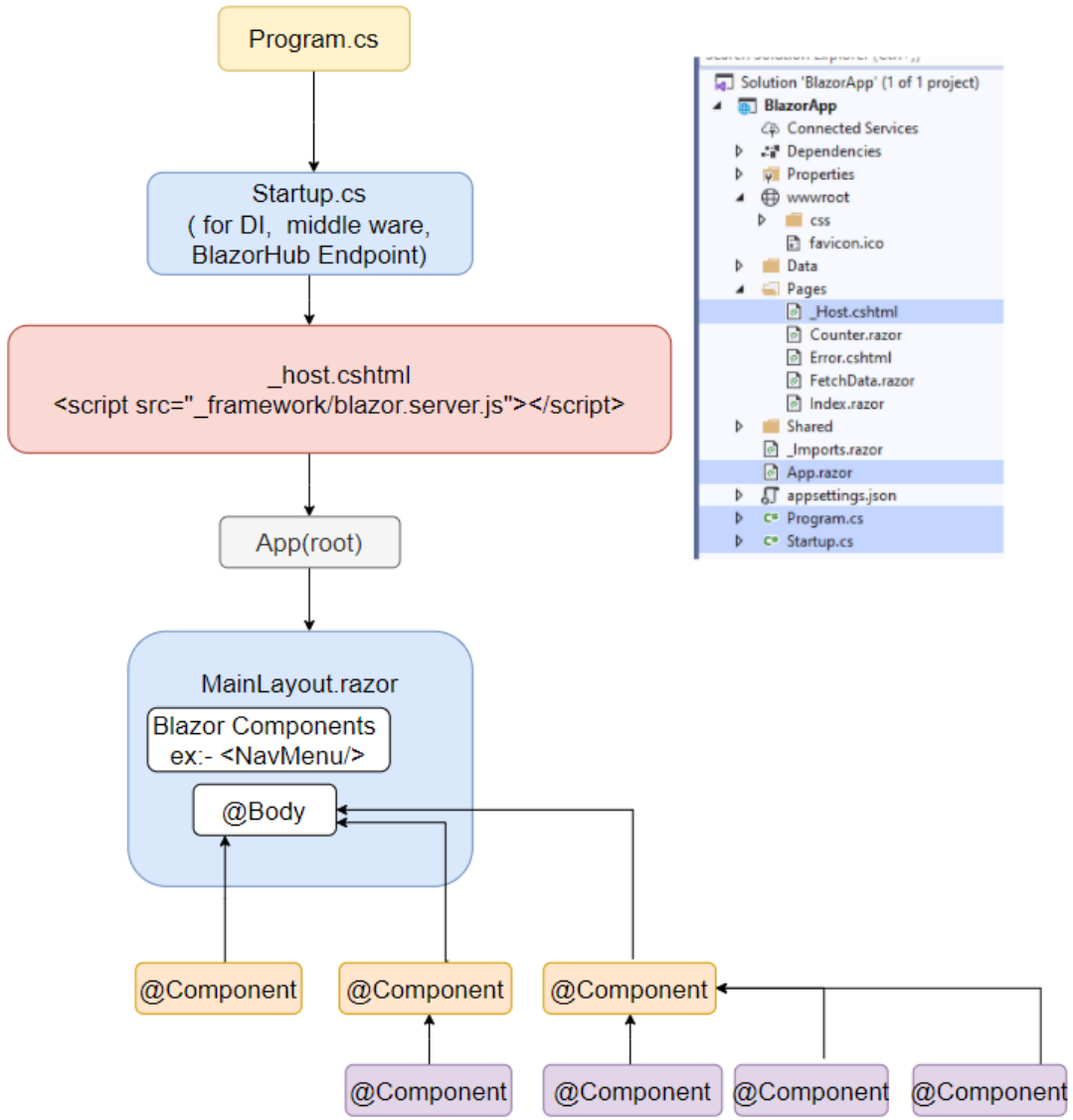
Cons

- High usage of resources on the server if more no of users are connected
- Doesn't work if signal-R disconnected
- Not suitable for offline workloads
- Latency on each event
- Maintain client state on Server

History

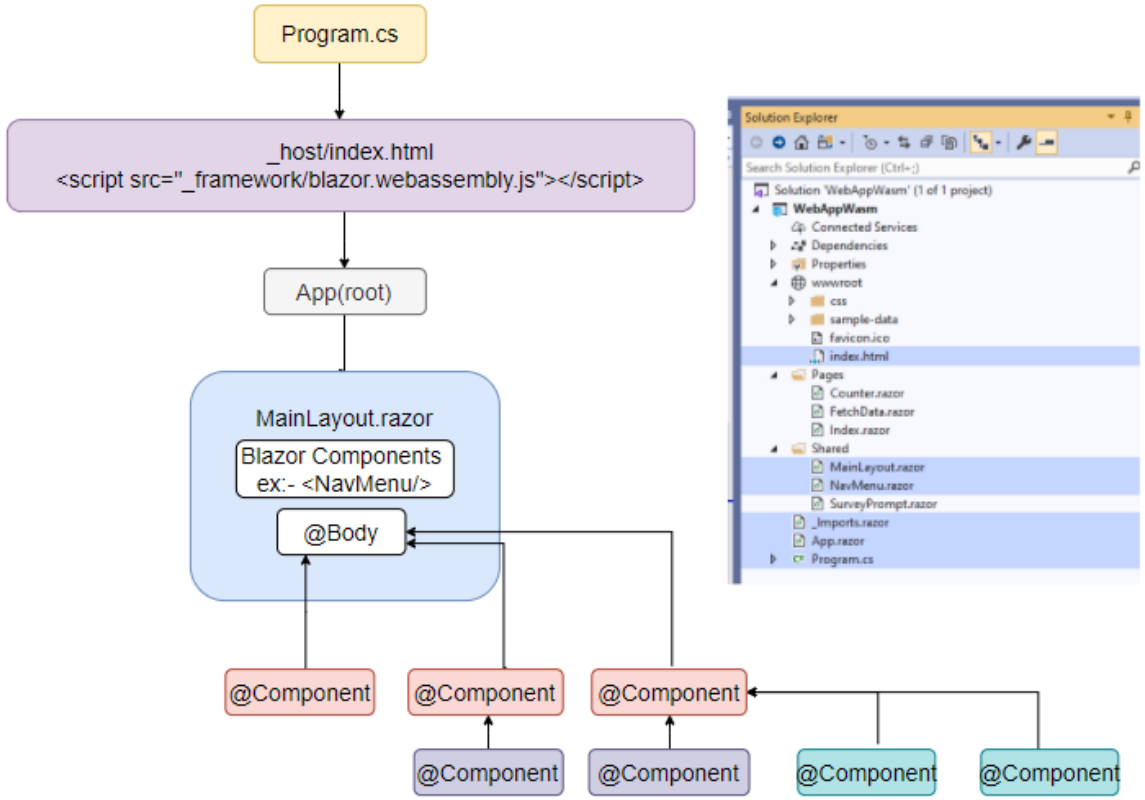
2002 –.NET Framework 1.0
2006 – Silverlight , Jquery
2007-2008 – Andriod & IPhones, Html5
2010 – Angular JS
2011 – Silverlight – last version
2013 – React 0.3.0, asm.js
2014 – port un-real Engine 4 to asm.js
2016 – .NET Core
2017 – Web Assembly
2017 – Blazor Announced – (Steve Sanderson)
2019 – Blazor Preview – (Daniel Roth) (April)
2020 – Blazor Server -.NET Core 3.0 (Sept)
2020 – Blazor WebAssembly - .NET Core 3.1 (May)
2021 – Blazor LTS

Blazor Server Request Flow &Project Structure

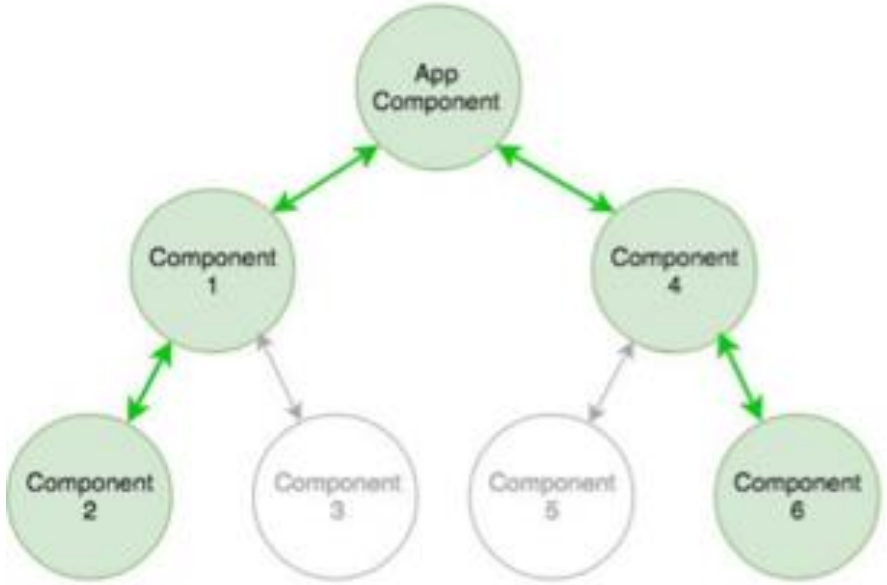


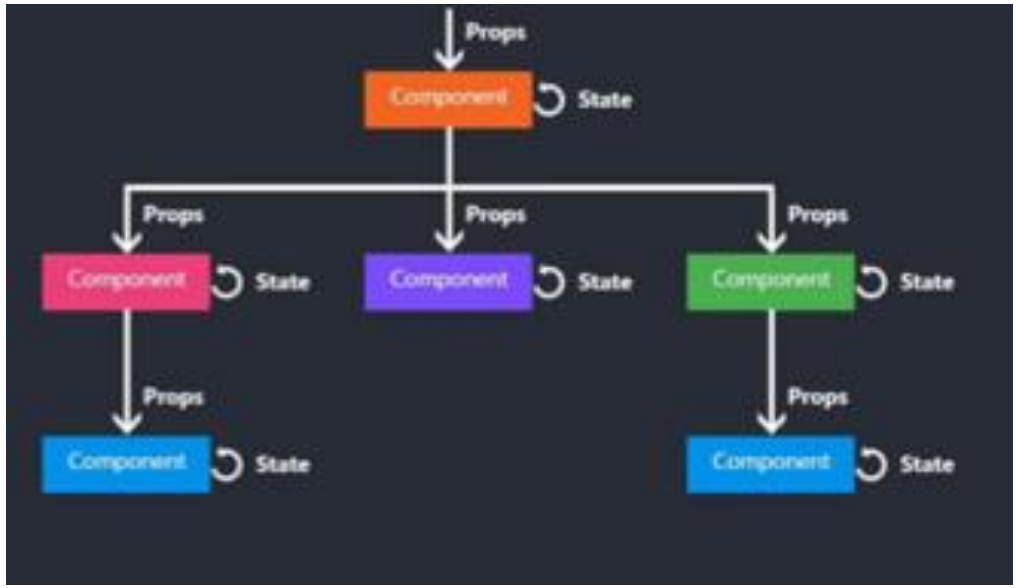
dotnet new blazorserver -o BlazorApp

Blazor WASM request Flow & Project Structure



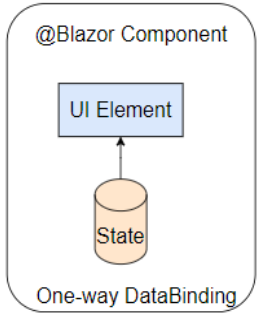
dotnet new blazorwasm –o BlazorWasmApp

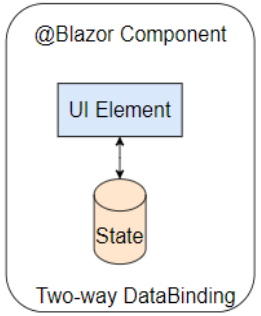




DataBinding

1. One Databinding is used to display data on UI
2. Two Databinding is used for manipulate data






```
<tr>
    <td>
        <NavLink href="@($"product/{Product.Id}")">
            <span>  @Product.Title</span>
        </NavLink>
    </td>
    <td>@Product.Maker</td>
</tr>

@code {
    [Parameter]
    public Product Product { get; set; }

    }
```

```
<div>
    <label for="searchCriterion">Search:  </label>
    <input type="text" placeholder="Search Products"
           @bind-value="searchCriterion"
           @bind-value:event="oninput"/>
    <span>@($"Given search criterion : {searchCriterion}")
     </span>
</div>
@code {

    private string searchCriterion;
}
```

Life Cycle events of Components :- Every component has series of events to render Blazor Component

```
/* order of life cycle events for Blazor component*/
protected override void OnParametersSet()
{
    base.OnParametersSet();
}

protected override void OnInitialized()
{
    base.OnInitialized();
}

protected override bool ShouldRender()
{
    return base.ShouldRender();
}
protected override void OnAfterRender(bool firstRender)
{
    base.OnAfterRender(firstRender);
}
```

```
/* order of life cycle async events for Blazor component*/

protected override async Task OnParametersSetAsync()
{
    await base.OnParametersSetAsync();
}

protected override async Task OnInitializedAsync()
{
    await base.OnInitializedAsync();
}
protected override bool ShouldRender()
{
    return base.ShouldRender();
}
protected override async Task OnAfterRenderAsync(bool firstRender)
{
    await base.OnAfterRenderAsync(firstRender);
}
```

Layout Component

Component to define common static content (Header , menu , Footer ,etc.) by using @LayoutComponentBase

Page Component

decorated with @page directive to define route for page.

@page "/customer"

Non-Page Component

Blazor components are defined with out @page directive.

Custom component

Custom components by using ComponentBase class in C# without defining .razor file

Component Inheritance

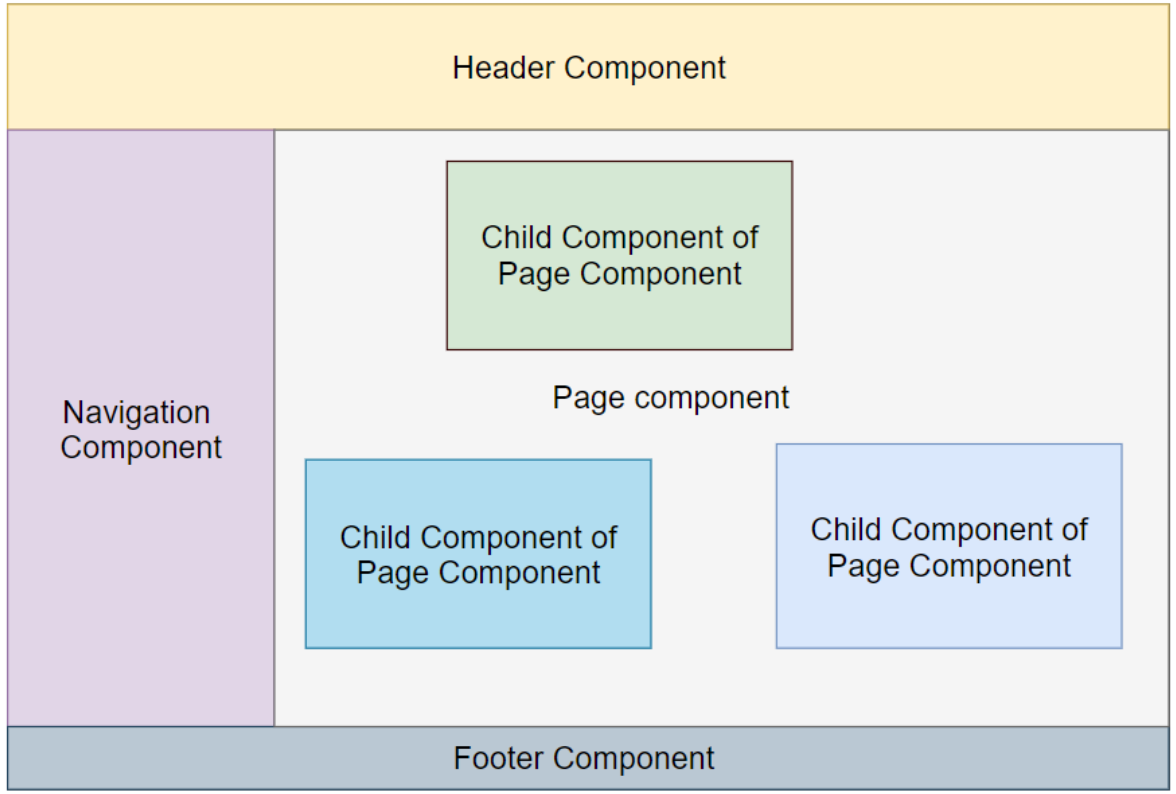
Can be inherit component by using @inherits directive .

Code block in Blazor Components

Used to write C# code for following things by using @code

- UI logic
- State
- Route parameters
- Component life cycle events,
- custom events in C#

Component Structure of Webpage



Components Overview & Inheritance

```
public class EmployeComponent :ComponentBase
{
    1 reference
    public int id { get; set; }
    2 references
    public string Name { get; set; }
    1 reference
    public string Dept { get; set; }
    1 reference
    public string Address { get; set; }

    0 references
    protected override void OnInitialized()
    {
        base.OnInitialized();
        var employee = EmployeService.GetEmployee();
        id = employee.id;
        Name = employee.Name;
        Dept = employee.Dept;
        Address = employee.Address;
    }
```

```
1  @page "/counter"
2  @inherits EmployeComponent
3
4  <span>@id</span>
5  <span>@Name</span>
6  <span>@Dept</span>
7  <span>@Address</span>
```

Communication between components & within component

Parameters

Properties

Parameters

Routing Parameters

Cascading Parameters

Used to maintain state

Events & Event-Callback

Used for actions on components

Communication between components & within component

Properties

```
<div class="card" style="width: 28rem;">
    <div class="card-body">
        <img src="@product.img" class="card-img-top" />
        <h5 class="card-title">@product.Title</h5>
        <h6 class="card-subtitle mb-2 text-muted">@product.Maker</h6>
        <hr />
        <p class="card-text">@product.Description</p>
    </div>
</div>
@code {
    private Product product;
}
```

Parameters

SearchProduct.razor* | ProductCard.razor*

```
1  <div class="card" style="width: 28rem;">
2      <div class="card-body">
3          <img src="@product.img" class="card-img-top" />
4          <h5 class="card-title">@product.Title</h5>
5          <h6 class="card-subtitle mb-2 text-muted">@product.Maker</h6>
6          <hr />
7          <p class="card-text">@product.Description</p>
8      </div>
9  </div>
10 @code {
11     private Product product;
12
13     [Parameter]
14     public int Id { get; set; }
```

SearchProduct.razor* | ProductCard.razor

```
22                                  {
23
24                                      <ProductCard Id =@id/>
25
```

Communication between components & within component

Routing Parameters:-

Routing Parameter used to pass data to component through route data

```
@page "/productcard/{Id:int}"
<div class="card" style="width: 28rem;">
    <div class="card-body">
        <img src="@product.img" class="card-img-top" />
        <h5 class="card-title">@product.Title</h5>
        <h6 class="card-subtitle mb-2 text-muted">@product.Maker</h6>
        <hr />
        <p class="card-text">@product.Description</p>
    </div>
</div>
@code {
    private Product product;

    [Parameter]
    public int Id { get; set; }
    protected override async Task OnParametersSetAsync()
    {
        await base.OnParametersSetAsync();

        product = await viewProduct.Execute(Id);
    }
}
```

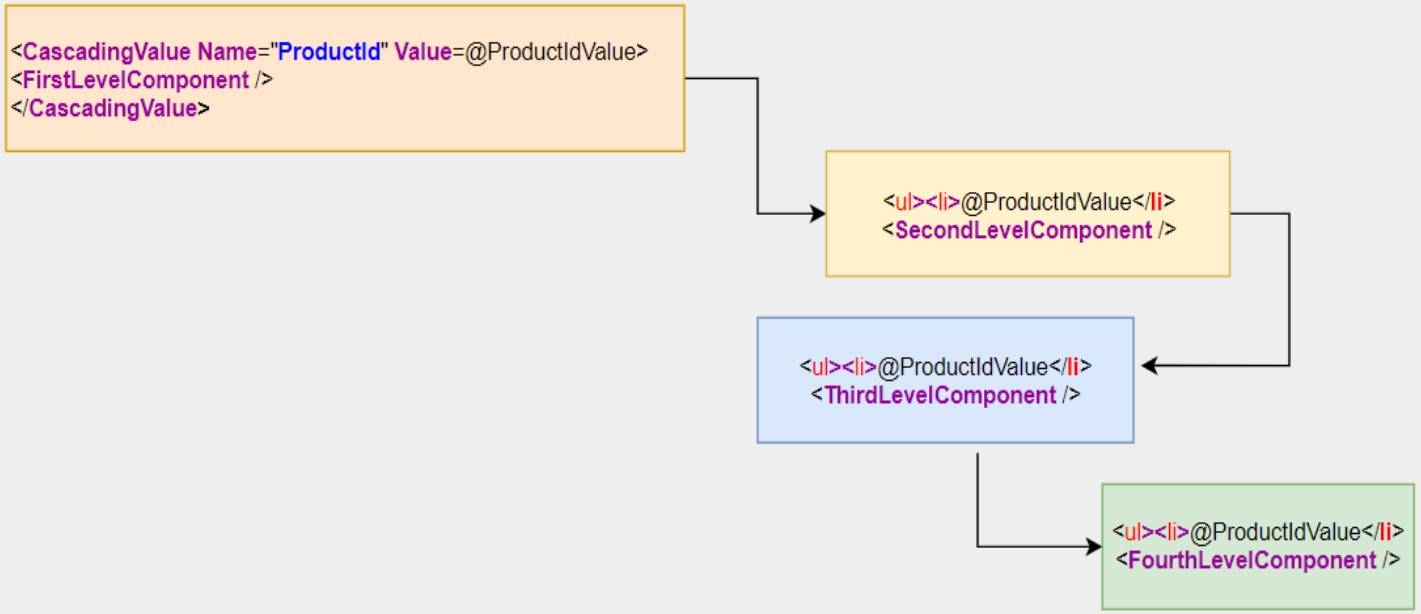
Routing Parameter with constraints used to restrict to datatype of route data

```
@page "/ComponentName/{parameterName:DataType}"
```

Communication between components & within component

Cascading Parameters:-

Cascading Parameters Cascading values and cascading parameters allow their values to cascade down the render tree without being passed explicitly from parent to child.



Communication between components & within component

Event Handlers:-

```
1   <form class="form-inline" >
2       <div class="form-group mx-sm-3 mb-2">
3       <label for="searchCriterion">Search:  </label>
4       <input type="text" class="form-control" id="filter"
5               placeholder="@placeholder" @bind-value="searchCriterion">
6       </div>
7       <button type="button" class="btn btn-primary mb-2"  @onclick="HandleSearch">
8       Search</button>
9   </form>
10  @code {
11      private string searchCriterion;
12      private string placeholder = "Search Products";
13
14      private void HandleSearch()
15      {
16          Console.WriteLine($" Handled search event");
17      }
18
19      }
```

Navigation

- NavLink component is used to navigate to another page component
- NavigationManage Class used to navigate to another page component

ProductItem.razor

```
1   <tr>
2       <td>
3           <NavLink href="@($"product/{Product.Id}")">
4           <span>  @Product.Title</span>
5           </NavLink>
6       </td>
7       <td>@Product.Maker </td>
8       <td>@AverageRating(Product)</td>
9   </tr>
10
```

Dependency Injection In Blazor

- Registering dependable objects in Blazor App withe the help in the configuring services
- .Net Core provide Dependency Injection (IoC) Container service to maintain singleton , transient & scoped objects

```
builder.Services.AddSingleton<IProductRepositoryAsync, ProductRepositoryAsync>();
builder.Services.AddTransient<ISearchProductAsync, SearchProductAsync>();
builder.Services.AddTransient<IViewProductAsync, ViewProductASync>();
builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new Uri(builder.HostEnvironment.BaseAddress)
```

- @inject directive helps to inject dependency objects into Razor component
- Inject attribute helps to inject dependency object in Razor component backend class

ProductCard.razor

```
1  @page "/product/{id}"
2  @inject IViewProductAsync viewProduct
3  @inject NavigationManager navigationManager
4
5  <h3> Product Details </h3>
```

JS- Interoperability

- Call JavaScript from Blazor using JSRuntime object

```
jsText = await JSRuntime.InvokeAsync<string>("JSFromBlazor.ReplayToBlazor");
```

```
<script src="scripts/JSFromBlazor.js"></script>
```

```
JSFromBlazor.js*
eShop.BlazorWasm.UI JavaScript Content Files        ReplayToBlazor
var JSFromBlazor = JSFromBlazor || {};
JSFromBlazor.ReplayToBlazor = function () {
    return 'Hello Blazor ,You made debugging C# on Browser with Wasm.'
};
```

- Call C# function from JS in Blazor using JSInvokable Attribute, DotNetObjectReference

```
var dotNetReference = DotNetObjectReference.Create(this);
JSRuntime.InvokeVoidAsync("BlazorFromJS.startRandomGenerator", dotNetReference);
```

```
[JSInvokable("AddText")]
public void AddTextToTextHistory(string text)
{

    System.Diagnostics.Debug.WriteLine("DotNet: Received " + text);
}
```

```
BlazorFromJS.js
eShop.BlazorWasm.UI JavaScript Content Files        setInterval() callback        text
var BlazorFromJS = BlazorFromJS || {};
BlazorFromJS.startRandomGenerator = function (dotNetObject) {
    setInterval(function () {
        let text = Math.random() * 1000;
        console.log("JS: Generated " + text);
        dotNetObject.invokeMethodAsync('AddText', text.toString());
    }, 1000);
};
```

```
<script src="scripts/JSFromBlazor.js"></script>
```

EditorForms & Validation

ProductViewModel.cs | ProductRepositoryAsync.cs | SearchProduct.razor | JSInterOp.razor

eShop.BlazorWasm.UI | eShop.BlazorWasm.UI.ViewModel.ProductViewModel

```
using FluentValidation;
//using System;
//using System.Collections.Generic;
//using System.ComponentModel.DataAnnotations;
//using System.Text;

namespace eShop.BlazorWasm.UI.ViewModel
{
    2 references
    public class ProductViewModel
    {
        0 references
        public int Id { get; set; }
        //[Required]
        //[StringLength(12,ErrorMessage = "Maker Name is too long (12 characters limit)")]
        5 references
        public string Maker { get; set; }
        //[Required]
        //[StringLength(15, ErrorMessage = "Title is too long (15 characters limit)")]
        5 references
        public string Title { get; set; }

        //[StringLength(50, ErrorMessage = "Description Name is too long (50 characters limit)")]
        //[Required]
        5 references
        public string Description { get; set; }
    }
    1 reference
    public class ProductViewModelValidator:AbstractValidator<ProductViewModel>
    {
        0 references
        public ProductViewModelValidator()
        {
            RuleFor(x => x.Maker).NotEmpty().Length(5,12);
            RuleFor(x => x.Title).NotEmpty().Length(5, 15); ;
            RuleFor(x => x.Description).NotEmpty().MaximumLength(50);

        }

    }
}
```

EditProduct.razor

```
@page "/addproduct"
<h3>Edit Product</h3>
<EditForm Model="ProductModel"
          OnValidSubmit="HandleSumitForm">
    @*<DataAnnotationsValidator />*@
    <FluentValidationValidator/>


    <div class="form-group mx-sm-3 mb-2">
        <label for="Maker">Maker:   </label>
        <InputText id="Maker" class="form-control"
                   placeholder="Maker Name" @bind-Value="ProductModel.Maker">Maker</InputText>
    </div>
    <div class="form-group mx-sm-3 mb-2">
        <label for="Title">Title :   </label>
        <InputText id="Title" class="form-control"
                   placeholder="Title Name" @bind-Value="ProductModel.Title">Title</InputText>
    </div>
    <div class="form-group mx-sm-3 mb-2">
        <label for="Description">Description :   </label>
        <InputText id="Description" class="form-control"
                   placeholder="Product Description" @bind-Value="ProductModel.Description">Title</InputText>
    </div>

    <ValidationSummary />
    <span class="bg-success">@Message </span>


    <button type="submit" class="btn btn-primary mb-2">
        Add Product
    </button>

</EditForm>
@code {

    private ProductViewModel ProductModel;
    private string Message;
```

dotnet add package Blazored.FluentValidation

Demos

Q&A

Thank you!

Email: rajesh.kolla01@outlook.com
Twitter: @RajeshKolla18
LinkedIn: https://be.linkedin.com/in/razeshkolla