Blazor multiple cascading parameters

In this video we will discuss **multiple cascading parameters in Blazor**. This is continuation to our previous video [Part 48](https://youtu.be/ZmDMKp1Q8kA). Please watch [Part 48](https://youtu.be/ZmDMKp1Q8kA) from [Blazor tutorial](https://www.youtube.com/playlist?list=PL6n9fhu94yhVowClAs8-6nYnfsOTma14P) before proceeding.

Any of the descendant components in the component tree can access the cascading value. The descendant component, that wants to access the cascading value simply declares a property of the same type, decorated with the [CascadingParameter] attribute.

There are 2 ways to cascade values from the parent component to a child component (in fact any child component) in the component tree.

Cascading values by type

ParentComponent.razor

This parent component is passing 2 cascading values

* Style (Value is of type string)
* EmployeeAge (Value is of type int)

<CascadingValue Value="@Style">

<CascadingValue Value="@EmployeeAge">

<ChildComponent></ChildComponent>

</CascadingValue>

</CascadingValue>

@code {

public string Style { get; set; } = "color:red";

public int EmployeeAge { get; set; } = 25;

}

Copy

ChildComponent.razor

* Since the cascaded values are of different type, this child component has to simply create a property of the same type and decorate with [CascadingParameter] attribute.
* ElementStyle property is of type string so it receives the Style cascaded value from the parent component.
* Along the same lines, EmpAge property in the child component receives the EmployeeAge cascaded value from the parent component.

<h1 style="@ElementStyle">-Child Component Text. Employee Age = @EmpAge</h1>

@code {

[CascadingParameter]

public string ElementStyle { get; set; }

[CascadingParameter]

public int EmpAge { get; set; }

}

Copy

Multiple cascading values of the same type

What if we have multiple cascading values of the same type.

ParentComponent.razor

Both the cascading values (Style and BorderStyle) are of type string.

<CascadingValue Value="@Style">

<CascadingValue Value="@BorderStyle">

<ChildComponent></ChildComponent>

</CascadingValue>

</CascadingValue>

@code {

public string Style { get; set; } = "color:red";

public string BorderStyle { get; set; } = "border:1px solid red";

}

Copy

ChildComponent.razor

* ElementStyle property in this child component receives BorderStyle cascaded values from the parent component.
* This is because both Style and BorderStyle are of type string and it is the BorderStyle cascaded value that is nearest to the child component so it wins.

<h1 style="@ElementStyle">-Child Component Text.</h1>

@code {

[CascadingParameter]

public string ElementStyle { get; set; }

}

Copy

ChildComponent.razor

* Both the cascading parameters in this example receives the Style cascading value from the parent component and the reasoning is the same.
* If multiple cascaded values of the same type are passed by the parent component to the child component, by default the cascaded value that is nearest to the child component in the hierarchy wins.

<h1 style="@ElementStyle; @H1Border">-Child Component Text.</h1>

@code {

[CascadingParameter]

public string ElementStyle { get; set; }

[CascadingParameter]

public string H1Border { get; set; }

}

Copy

Cascading values by name

* Another way to access cascading values is by Name.
* Since a unique name is provided to each of the values, these values are now cascaded to the descendant components by Name.

ParentComponent.razor

<CascadingValue Value="@Style" Name="ColorStyle">

<CascadingValue Value="@BorderStyle" Name="BorderStyle">

<ChildComponent></ChildComponent>

</CascadingValue>

</CascadingValue>

@code {

public string Style { get; set; } = "color:red";

public string BorderStyle { get; set; } = "border:1px solid red";

}

Copy

ChildComponent.razor

* The child component can now decide which cascading value it wants by using the Name.
* ElementStyle cascading parameter receives the ColorStyle cascading value and
* H1Border cascading parameter receives the BorderStyle cascading value