Blazor Attribute Splatting

In this article we will discuss, what is Attribute Splatting in Blazor with an example.

Passing data from parent component to child component

Child Component (ChildComponent.razor)

<input id="firstName" placeholder="Child Component Placeholder"/>

Copy

Parent Component (ParentComponent.razor)

<ChildComponent placeholder="Parent Component Placeholder"></ChildComponent>

Copy

This code throws the following InvalidOperationException at runtime.

Object of type 'ChildComponent' does not have a property matching the name 'placeholder'.

Copy

To be able to pass data from Parent Component to Child Component we use component parameters. We discussed component parameters in [Part 27](https://youtu.be/Ii7QwHdD0oU) of [Blazor tutorial](https://www.pragimtech.com/courses/blazor-tutorial-for-beginners/).

To make the above example work modify the code in the child component

<input id="firstName" placeholder="@Placeholder"/>

@code {

[Parameter]

public string Placeholder { get; set; } = "Child Component Placeholder";

}

Copy

* We created Placeholder parameter.
* The <input> element binds to this Placeholder parameter.
* The parent component can now pass a value for this Placeholder parameter.

<ChildComponent placeholder="Parent Component Placeholder"></ChildComponent>

Copy

Example without using attribute splatting

Child Component

This child component has several parameters. It is tedious to declare these parameters in code and then assign values in the HTML, especially if you have lot of parameters. Instead we could we use attribute splatting. In both the cases the generated HTML is the same. The benefit of attribute splatting is we write a lot less code.

<input id="firstName" required="@Required" maxlength="@Maxlength" size="@Size"

placeholder="@Placeholder" />

@code {

[Parameter]

public string Required { get; set; } = "required";

[Parameter]

public string Maxlength { get; set; } = "15";

[Parameter]

public string Size { get; set; } = "25";

[Parameter]

public string Placeholder { get; set; } = "Child Component Placeholder";

}

Copy

Parent Component

<ChildComponent required="required" maxlength="100" size="50"

placeholder="Parent Component Placeholder">

</ChildComponent>

Copy

Example with attribute splatting

Child Component

* With Attribute Splatting, a dictionary is used to capture attributes.
* The dictionary key is of type string and value is of type object.
* These attributes are then splatted onto an HTML element using  @attributes Razor directive.

<input id="firstName" @attributes="InputAttributes" />

@code {

[Parameter]

public Dictionary<string, object> InputAttributes { get; set; } =

new Dictionary<string, object>()

{

{ "required", "required" },

{ "placeholder", "Child Component Placeholder" },

{ "size", "100" },

{ "maxlength", "15" }

};

}

Copy

Parent Component

<ChildComponent InputAttributes="attributesFromParent">

</ChildComponent>

@code {

public Dictionary<string, object> attributesFromParent { get; set; } =

new Dictionary<string, object>()

{

{ "required", "required" },

{ "placeholder", "Parent Component Placeholder" },

{ "size", "50" },

{ "maxlength", "5" }

};

}

Copy

Declared Properties and Attribute splatting

* A component can have both explicitly declared parameters and attribute splatting.
* In general, attribute splatting is used to capture and render additional attributes in addition to the component's explicitly declared parameters.
* In the following example, Value is explicitly declared parameter.

Child Component

<input id="firstName" @attributes="InputAttributes" value="@Value"/>

@code {

[Parameter]

public Dictionary<string, object> InputAttributes { get; set; } =

new Dictionary<string, object>()

{

{ "required", "required" },

{ "placeholder", "Child Component Placeholder" },

{ "size", "100" },

{ "maxlength", "15" }

};

[Parameter]

public string Value { get; set; } = "Child Value";

}

Copy

Parent Component

The parent component passes the value (Parent Value) to the child component's explicitly declared Value parameter.

<ChildComponent InputAttributes="attributesFromParent" Value="Parent Value">

</ChildComponent>

@code {

[Parameter]

public Dictionary<string, object> attributesFromParent { get; set; } =

new Dictionary<string, object>()

{

{ "required", "required" },

{ "placeholder", "Parent Component Placeholder" },

{ "size", "50" },

{ "maxlength", "5" }

};

}