Blazor data binding

In this video we will learn

1. One way data binding in Blazor
2. Two way data binding in Blazor

One way data binding in Blazor

Binding to a property

In the following example, we have one way data binding i.e from the component class to the component view. The view binds to the Name property in the component class. Whenever the Name property value changes, the UI is automatically updated to reflect the change.

Component class

protected string Name { get; set; } = "Tom";

Copy

Component view

<b>Name :</b> @Name

Copy

Binding to razor expression

The following example display employee name with salutation depending on their gender.

Component class

protected string Name { get; set; } = "Tom";

protected string Gender { get; set; } = "Male";

Copy

Component view

<b>Name with salutation:</b> @(Gender == "Male" ? $"Mr. {Name}" : $"Miss. {Name}")

Copy

Binding to input element

In the following example, the html input element (text box) is bound to Name property. This is still one way binding i.e from the component class to the view. Whenever the Name property value in the class changes, the UI is automatically updated to reflect the change. However, when we change the value in the text box, the Name property in the component class is not changed. Data is flowing only in one direction i.e from the component class to the UI but not in the other direction i.e from the UI to the class.

In this example we are binding to an input element. In a similar way, we can bind to any HTML element, like radio button, checkbox etc.

Component class

protected string Name { get; set; } = "Tom";

Copy

Component view

<b>Name :</b> <input value="@Name" />

Copy

Two way data binding in Blazor

* With two way data binding, data moves in both the directions i.e from the component class to the UI and from the UI to the component class.
* The following is one way to implement two way data binding in Blazor. The value attribute is bound to Name property, so the data flows from the component class to the UI.
* We are also binding to onchange event. The value for onchange attribute is a lambda expression which updates the Name property with the value in the text box. So with this event binding, the data flows from the UI to the component class property.
* Please note : We are using a lambda expression here, but you can also create a separate named method in the component class and then specify the name of that method as the value for onchange event.

Component class

protected string Name { get; set; } = "Tom";

Copy

Component view

<b>Name :</b> <input value="@Name" @onchange="@(e => { Name = e.Value.ToString(); })" />

Copy

Making two way data binding easier

The above example can be rewritten using the bind attribute as shown below. It simplifies two way data binding.

Component class

protected string Name { get; set; } = "Tom";

Copy

Component view

<b>Name :</b> <input @bind="Name" />

Copy

The change event is triggered, only when the input element loses focus and that's when the property in the component class is updated with the data in the input element.

Event parameter with two way data binding

* What if we do not want to wait, until the input element loses focus. As the value in the input element is being changed, we want the component class property to be updated.
* This can be achieved by specifying the event name in the two way data binding. To specify the event name we use the event parameter as shown in the example below. In this case we are binding to input event. This is raised as the value is being changed.

<b>Name :</b> <input @bind="Name" @bind:event="oninput" />

Copy

Binding to element attributes other than value attribute

@bind-{ATTRIBUTE} along with @bind-{ATTRIBUTE}:event syntax can be used to bind element attributes other than the value attribute. In the example below, the <div> element style is updated when the Colour property value changes:

Component class

protected string Colour { get; set; } = "background-color:white";

Copy

Component view

<b>Colour :</b> <input @bind="Colour" />

<div @bind-style="Colour" @bind-style:event="onchange">

<h3>The background colour of this text changes</h3>

</div>