Validating nested complex models and collection types in Blazor

In this video we will discuss, **validating nested complex models and collection types in Blazor**.

Validating top-level model properties in Blazor

public class Employee

{

[Required]

public string FirstName { get; set; }

[Required]

public string LastName { get; set; }

}

Copy

To validate the top-level properties (FirstName and LastName) of Employee class, we use DataAnnotationsValidator component.

<EditForm Model="@Employee">

<DataAnnotationsValidator />

<InputText id="firstName" @bind-Value="EditEmployeeModel.FirstName" />

<ValidationMessage For="@(() => EditEmployeeModel.FirstName)" />

</EditForm>

Copy

However, DataAnnotationsValidator component cannot be used to validate nested complex models and collection types.

public class Employee

{

[Required]

public string FirstName { get; set; }

[Required]

public string LastName { get; set; }

[ValidateComplexType]

public Department Department { get; set; }

}

public class Department

{

[Required]

public string DepartmentName { get; set; }

}

Copy

Department property in the Employee class is a complex type. To validate this use ObjectGraphDataAnnotationsValidator instead of DataAnnotationsValidator component. As the name implies, the ObjectGraphDataAnnotationsValidator validates the entire object graph, including collection and complex-type properties.

<EditForm Model="@Employee">

<ObjectGraphDataAnnotationsValidator />

<InputText id="firstName" @bind-Value="EditEmployeeModel.FirstName" />

<ValidationMessage For="@(() => EditEmployeeModel.FirstName)" />

</EditForm>

Copy

The complex types in the model should be decorated with [ValidateComplexType] attribute. In our example, Department property in the Employee class is decorated with this attribute.

Both ValidateComplexType and ObjectGraphDataAnnotationsValidator are in the nuget package Microsoft.AspNetCore.Components.DataAnnotations.Validation