What are Blazor Partial Classes?

So far, we’ve learned how to separate our HTML and C# logic inside a single file. We’ve used the @code directive to do that, but ideally, we don’t want to mix these things together. As our projects grow, we would end up with more and more HTML and C# logic and that can become hard to read and maintain. In order to keep our codebase clean, we want to separate HTML markup and C# logic into separate files. That’s where partial classes come in. With the usage of partial classes, we can separate these into two different, but still connected files. Let’s see how we can do that. Let’s use the Home.razor file for this example.
If we inspect this file, we can see the C# logic in the @code part. This is a small file and having both parts in the same file is not an issue at all. But as our projects grow, we are ending up with more and more HTML and C# logic and that can become hard to read and maintain. So, it is good practice to separate these two parts. To do that, we need to create a new class in the Component folder and name it Home.razor.cs. So, let’s do that... As we can see, the Visual Studio recognizes this file and attaches it to the Home.razor file. But, if we inspect our new class, we are going to see an error. That’s because this new class has to be a partial one. So, let’s add a partial keyword. Now, we can open the Home component file... Cut all the code from the @code section... and remove the @code directive. After that, we are going to paste that code into our newly created **Home** partial class. Of course, we have to import the Microsoft.AspNetCore.Components library to be able to use **[Parameter]** attributes. Excellent. If we build and run our application now, everything should work as it supposed to, without a single problem... And one more thing. We can use multiple partial classes if we need them. The important part is to give them different names, for example, Home.razor.data.cs, Home.razor.internal.cs etc. That’s it for our partial class implementation. The concept is very easy to implement and use and it offers a clean approach for web development, which separates C# and HTML into different files. Let’s move on to RenderFragment Parameters.