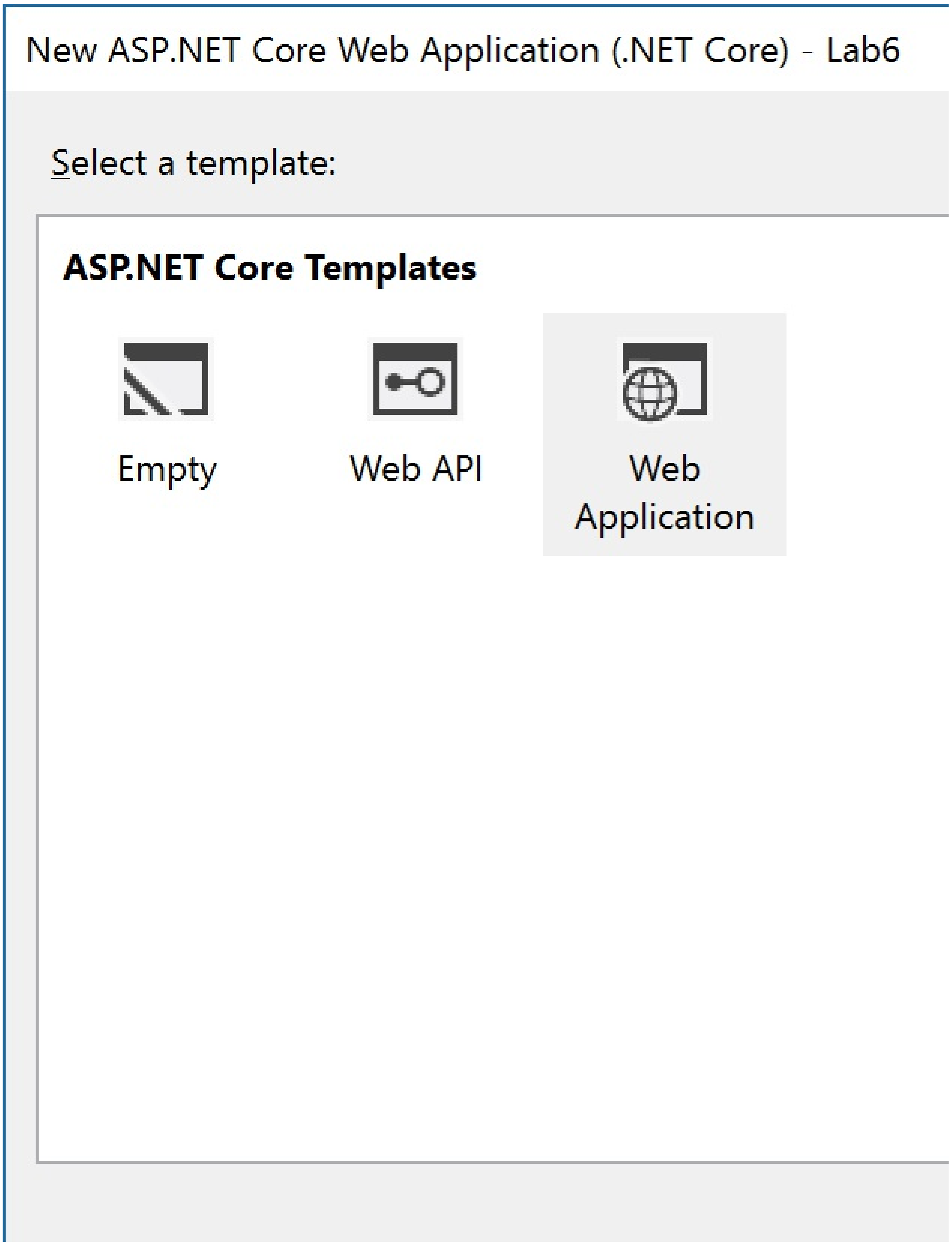
Working with Razor Tag Helpers

Look at Tag Helpers in the project template

1. Create a new ASP.NET Core project using the "Web Application" template with "Individual User Accounts" selected.



New ASP.NET Core Web Application (.NET Core) - Lab6

Select a template:

ASP.NET Core Templates

2. Open the view `Views/Account/Register.cshtml`

3. Look at the Tag Helpers being used in this view (they're colored purple and in bold in Visual Studio, make sure you build the project) and play around with setting their attributes and exploring the IntelliSense offered for the different attribute types

4. Run the application and see the HTML output by the Tag Helpers

5. Look at the other views in `Views/Account/` folder to see how they use Tag Helpers

6. Open the file `Views/Shared/_Layout.cshtml`

7. Look at the Tag Helpers being used in the `<head>` element and at the end of the page to render CSS stylesheets and JavaScript files and compare it to the generated HTML output

Create a custom Tag Helper

1. Create a new class in the application you created above, `public class RepeatTagHelper : TagHelper` and resolve any required namespaces

2. Add a property `public int Count { get; set; }`

3. Add an override for the TagHelper::ProcessAsync() method which repeats the content via the `output` parameter in a loop for `Count` iterations, e.g.:

```
public async override Task ProcessAsync(TagHelperContext context,
  TagHelperOutput output)
{
    for (int i = 0; i < Count; i++)
    {
        output.Content.AppendHtml(
            await output.GetChildContentAsync(useCachedResult: false));
        }
    }
```

4. Open the `_ViewImports.cshtml` file and add line to register your Tag Helper: `@addTagHelper "*, WebApplication49"` (adjust WebApplication49 to your assembly name)
5. Open `Views/Home/Index.cshtml` and use your Tag Helper, e.g.:

   ```
   <repeat count="5">
     <h1>I'll be repeated!! @DateTime.UtcNow.Ticks</h1>
   </repeat>
   ```

6. Run the application again and ensure your Tag Helper is working
7. Note that the Tag Helper is rendering itself as a `<repeat>` tag (see it in the rendered HTML). We'll fix that now so that only the contents are rendered.
8. Open the `RepeatTagHelper` again and in the `ProcessAsync` method add a line to null out the tag name: `output.TagName = null;`
9. Run the application again and see that the outer tag is no longer rendered

Extra if you have time

1. Experiment with decorating your `RepeatTagHelper` class with the `[HtmlTargetElement()]` attribute to change which HTML tag and/or attribute names it will attach itself to