RAZOR

What is Razor?

Razor is a **server side** markup language.

RAZOR or HTML?

HTML is the standard markup language for creating Web pages.

HTML is used to give websites structure with text, links, images, and other fundamental elements.

Razor is a **server side** markup language. It does everything HTML does but also allows to use **server side** code. Server generates HTML from Razor code.

Why Razor?

Razor is a markup syntax that lets you embed **server-based code** (C#) into web pages.

That means you can use variables and objects from C#, loops, conditionals, e.t.c.

Why Razor? #2

Server-based code can create dynamic web content on the fly.

When a web page is called, the server executes the server-based code inside the page before it returns the page to the browser.

By running on the server, the code can perform complex tasks, like accessing databases.

RAZOR Syntax

Any HTML code is a valid Razor syntax.

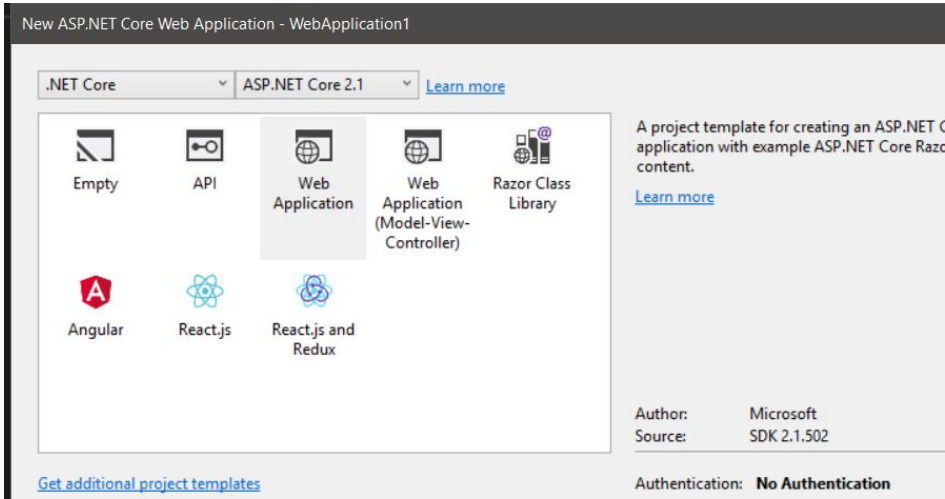
However Razor adds special “Razor block” which will execute some kind of C# code.

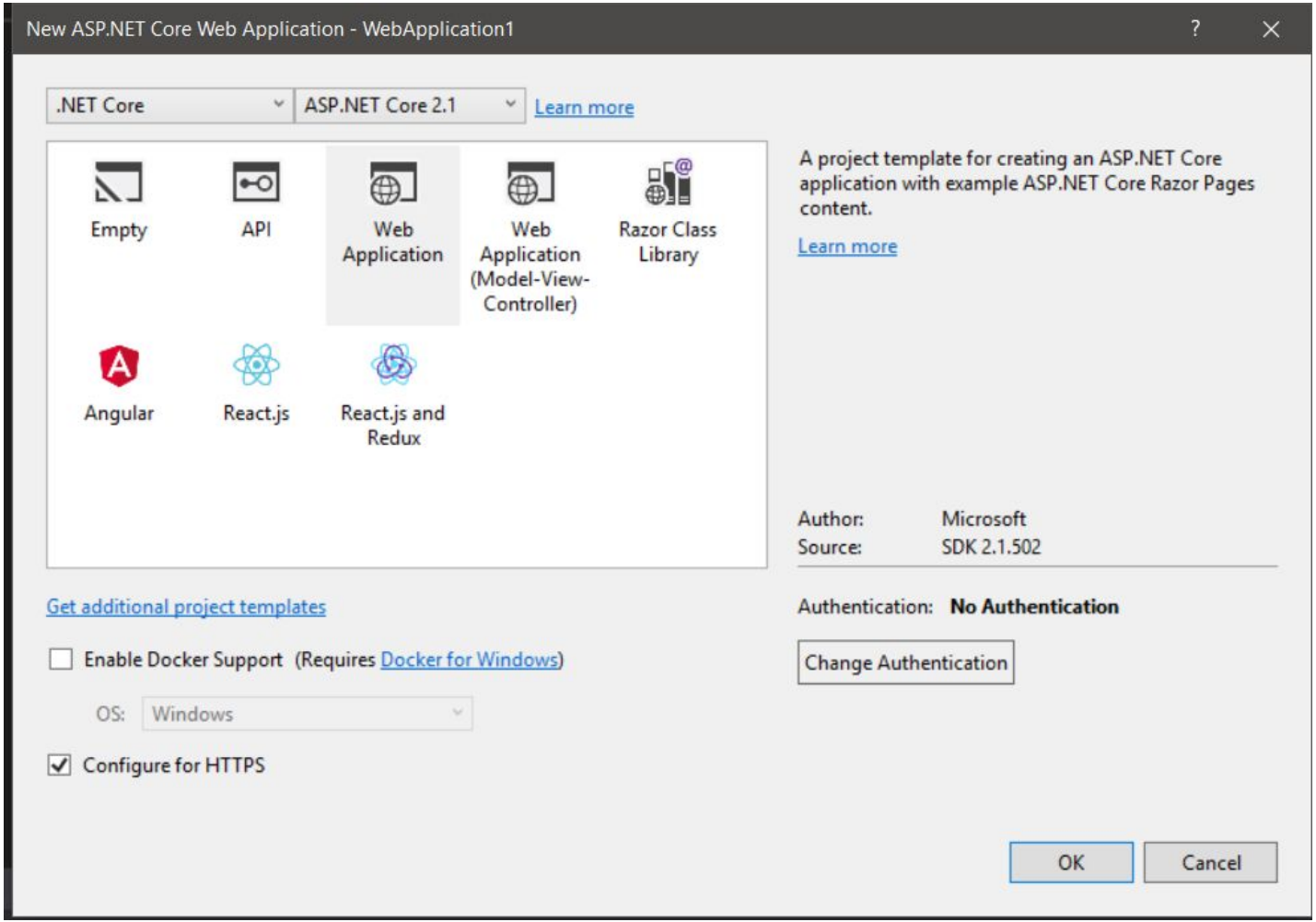
Razor block are enclosed in **@{ ... }**, variables are accessible with **@**

Razor files have **.cshtml** file extension

Creating Razor project in VS

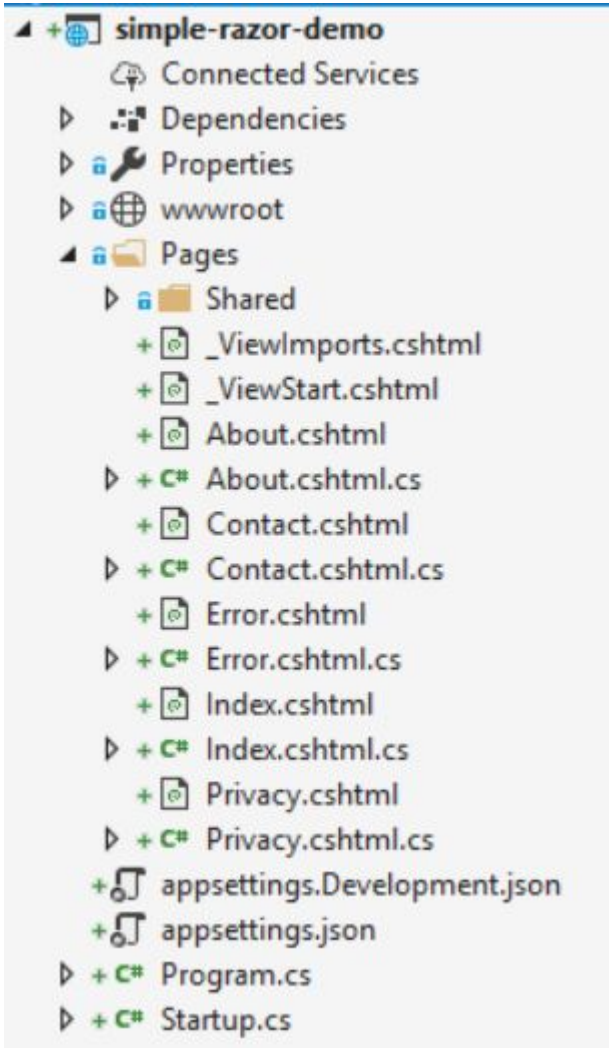
1) Open Visual Studio
2) File -> New -> Project
3) In “Visual C#” find “.NET Core” and choose “ASP.NET Core Web Application”
4) As a template choose “Web Application” and click ok




New ASP.NET Core Web Application - WebApplication1
.NET Core
ASP.NET Core 2.1
Learn more
Empty
API
Web Application
Web Application (Model-View-Controller)
Razor Class Library
Angular
React.js
React.js and Redux
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content.
Learn more
Author: Microsoft
Source: SDK 2.1.502
Get additional project templates
Authentication: No Authentication
Enable Docker Support (Requires Docker for Windows)
Change Authentication
OS: Windows
Configure for HTTPS
OK
Cancel

A new project will be created

1) Startup.cs and Program.cs are needed to setup and run web-application and contain application entrypoint (*void Main()*)
2) “Pages” folder contain all the Razor files (.cshtml) and associated C# code with them (.cshtml.cs)



.cshtml.cs structure

Open **Pages/About.cshtml.cs**

This C# code defines page model.

Properties of this class will be available in corresponding .cshtml file

Function “OnGet()” is executed when user opens this page in browser.

.cshtml.cs structure #2

Add a new public property “Name” which is a string.

In method OnGet set it to be your name.

```
0 references | 0 changes | 0 authors, 0 changes
public class AboutModel : PageModel
{
    1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Message { get; set; }

    1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Name { get; set; }

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public void OnGet()
    {
        Name = "Dmitrijs";

        Message = "Your application description page.";
    }
}
```

.cshtml structure

Open **Pages/About.cshtml**

This is a Razor code which will generate HTML for “About” page.

```
@page
@model AboutModel
@{
    ViewData["Title"] = "About";
}
<h2>@ViewData["Title"]</h2>
<h3>@Model.Message</h3>

<p>Use this area to provide additional information.</p>
```

@page - directive to tell that this is a Razor page
@model - directive to set corresponding C# model
@{...} - code block
@Model.Message - syntax to access Message property of model class of this page

.cshtml structure #2

Change paragraph text
to display your name.

Congratulations. You just wrote your first Razor code.

```
@page
@model AboutModel
@{
    ViewData["Title"] = "About";
}
<h2>@ViewData["Title"]</h2>
<h3>@Model.Message</h3>

<p>Hello! My name is @Model.Name</p>
```

Run the project

A new browser window will open. Navigate to “About” page. Enjoy your own changes :)

Index page

“Index” page is web-application entry point. When you open any website “index.html” file is loaded from the server.

In Razor Index is default page.

Page “Index” under folder “Visma” corresponds to “{url}/Visma” page

Page “Contacts” under folder “Visma” corresponds to “{url}/Visma/Contacts” page

Let’s create a new page

1) Create a new folder under “Pages” called “Visma”
2) Right click -> Add -> Razor Page
3) Razor Page name: Index, **uncheck “use layout page” !!!**

A new .cshtml page “Index” is created with corresponding C# model.

Index.cshtml contains basic HTML, **Index.cshtml.cs** contains page model.

```razor
@page
@model simple_razor_demo.Pages.Visma.IndexModel
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
</body>
</html>
```

```
0 references | 0 changes | 0 authors, 0 changes
public class IndexModel : PageModel
{
    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public void OnGet()
    {

    }
}
```

Add the “Hello world” to body

Run the project

Navigate to “/Visma”

```
@page
@model simple_razor_demo.Pages.Visma.IndexModel
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Index</title>
</head>
<body>
    <h1>Hello world!</h1>
</body>
</html>
```

Razor loops

It is possible to run loops on Razor pages.

```
<ul>
    @{
        for (var i = 0; i < 10; i++)
        {
            <li>@i</li>
        }
    }
</ul>
```

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

Razor foreach loop

Add a new array property to page model.

```
public class IndexModel : PageModel
{
    1 reference | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string[] Names { get; set; }

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public void OnGet()
    {
        Names = new[] { "Dmitrijs", "Jurijs", "Aleks" };
    }
}
```

Razor foreach loop #2

Add a new list on page with foreach loop inside it

```
<ol>
    @{
        foreach(var name in Model.Names)
        {
            <li>Name: @name</li>
        }
    }
</ol>
```

1. Name: Dmitrijs
2. Name: Jurijs
3. Name: Aleks

Razor inline variables

It is possible to create variables which are local just for Razor.

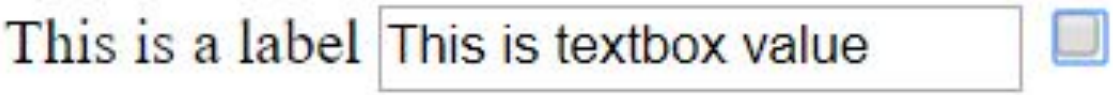
```
@{
    var something = "hello";
}

<p>Razor variable "something" is "@something"</p>
```

Razor variable "something" is "hello"

Razor built-in helpers

```
@Html.Label("This is a label")
@Html.TextBox("This is a textbox", "This is textbox value")
@Html.CheckBox("This is a checkbox")
```



Razor forms

To pass data from HTML to C# code we need to use HTML Forms.

Razor have convenient way to do it: “Html.BeginForm()” helper!

Make a new Razor Form

Create a new page in Visma folder, called **Contact.** (Right click -> Add -> Razor Page, **uncheck “use layout page” !!!**)

Modify “Contact” C# model:

1. Add 3 properties - Name, Email and a Message
2. Add **[BindProperty]** annotation to Name and Email.
3. Add a new method **OnPost.** In this method set the message to contain name and email from model.

```
public class ContactModel : PageModel
{
    [BindProperty]
    2 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Name { get; set; }
    [BindProperty]
    2 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Email { get; set; }

    2 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public string Message { get; set; }

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public void OnGet()
    {

    }

    0 references | 0 changes | 0 authors, 0 changes | 0 exceptions
    public void OnPost()
    {
        Message = $"{Name} will be contacted using {Email} email";
    }
}
```

Make a new Razor Form #2

Modify “Contact” Razor page:

1. Add “@using (Html.BeginForm())” Razor block
2. Inside it add 2 input fields for “Model.Name” and “Model.Email”
3. Inside it add “submit” button
4. Add a message on page

```
<body>

<h3>If you want to be contacted leave your name and email below</h3>

@using (Html.BeginForm())
{
    <label>Your name:</label>
    @Html.TextBoxFor(model => model.Name)
    <label>Your email:</label>
    @Html.TextBoxFor(model => model.Email)

    <button type="submit">Submit</button>
}

<strong>@Model.Message</strong>

</body>
```

Make a new Razor Form #3

Now run solution, navigate to “/Visma/Contact” and test it!

Small task!

Now create webpage using Razor and C#.
On this page user can calculate right angled triangle hypotenuse by giving length of 2 legs.

Only if you are brave: show calculation history (use array and add calculations to it :))

$$c = \sqrt{a^2 + b^2}$$

a Leg Enter value

b Leg Enter value

Calculate