Razor Asp.NET Core

ARCTECH INFO PRIVATE LIMITED

What is Razor

Razor is one of the view engines supported in ASP.NET MVC

Allows you to write a mix of HTML and server-side code using C# or Visual Basic

Razor syntax has the following Characteristics:

- Compact: Razor syntax is compact, enabling you to minimize the number of characters and keystrokes required to write code.
- Easy to Learn: Razor syntax is easy to learn where you can use your familiar language C# or Visual Basic.
- IntelliSense: Razor syntax supports statement completion within Visual Studio.

Some Razor features

Inline expression

- Start with @ symbol to write server-side C# within HTML code.
- For example, @Variable_Name display the value of a server-side variable
- E.g., Write @DateTime.Now to display the current date and time, as shown below.

```
<h1>Razor syntax demo</h1>
<h2>@DateTime.Now.ToShortDateString()</h2>
```

- Output

```
Razor syntax demo
23-02-2022
```

- A single line expression does not require a semicolon at the end of the expression.

Multiline Code Block

You can write multiple lines of server-side code enclosed in braces @{ ... }.

Each line must end with a semicolon the same as C#.

```
@{
    var date = DateTime.Now.ToShortDateString();
    var message = "Hello World";
}

<h2>Today's date is: @date </h2>
<h3>@message</h3>
```

Output

```
Today's date is: 08-09-2014
Hello World!
```

Each line must end with a semicolon the same as C#.

If-else condition

Write if-else condition starting with @ symbol.

The if-else code block must be enclosed in braces { }, even for a single statement.

```
@if(DateTime.IsLeapYear(DateTime.Now.Year) )
{
    <span>@DateTime.Now.Year is a leap year</span>
}
else
{
    <span>@DateTime.Now.Year is not a leap year.</span>
}
```

Output

```
2022 is not a leap year.
```

for loop

Code

```
@for (int i = 0; i < 5; i++)
{
    @i.ToString() <br />
}
```

Output

```
0
1
2
3
4
```

Model

Use @model to use model object anywhere in the view.

```
@model Student

<h2>Student Detail:</h2>
<ul>
    <li>Student Id: @Model.StudentId</li>
    <li>Student Name: @Model.StudentName</li>
    <li>Age: @Model.Age</li>
</ul>
```

Output

```
Student Detail:

- Student Id: 1

- Student Name: John

- Age: 18
```

Declare Variables

Declare a variable in a code block enclosed in brackets and then use those variables inside HTML with @ symbol.

```
@{
    var str = "";
    var num = 10;
    if(num > 0)
    {
        str = "Hello World!";
    }
}
<p>@str</p>
```

Output

```
Hello World!
```

Hands On

Create a Student Model class
- Place it in the Models Folder

Create a StudentsService class
- Place it in a new folder called Services
- Add Service methods like GetAllData, Create
- Write ADO.Net code to select, insert, update and delete

Create a StudentController class
- Provide Actions Index and Create
- In the Index action, create a List of Students and sent it to the View

Create Views Index and Create
- In the Index View, configure the @model and then iterate/display all students
- Use BootStrap classes to beautify the student listing

Hands On - Create

Add Create action in StudentController

- This is the action when user visits url https://.../Students/Create

Add Create.cshtml view and add the fields and labels for the model

- Specify @model Student (Note: model is Student and not List<Student> as we are creating a single record)
- Use Tag Helper asp-for
- Ensure form action="post" and submit button are there

Add Create(Student student) action in StudentController

- This is an overloaded Create action
- The overloaded Create action will be called when the user clicks [Create] button after entering all the data.
- The default Create action will be called when user visits the page initially
- Note: The overloaded Create action needs following attributes
  - `[HttpPost]`
  - `[ValidateAntiForgeryToken]`

Tag Helpers

Allows server-side code to participate in creating and rendering HTML elements in Razor files

You can build an entire ASP.NET Core MVC application without using a single HTML helper.

However, HTML helpers make your life as a developer easier.

By taking advantage of helpers, you can build your views with far less work.

There are many built-in Tag Helpers for common tasks

- such as creating forms, links, loading assets and more
- and even more available in public GitHub repositories and as NuGet packages

HTML Helpers & Tag Helpers

Tag Helpers are the modern version of the old HTML Helpers in ASP.NET Core

HTML Tags with Tag Helpers are the modern .NET Core version (.NET Core 2+, .NET 5+)

- Server Controls in ASP.NET WebForms
- HTMLHelpers in older version of .NET Core

Tag Helpers look like Html attributes and have excellent IntelliSense support

- `<a asp-controller="Student" asp-action=“Create">Create New</a>`
- `<label asp-for="FirstName" class="form-control"></label>`
- `<input asp-for="FirstName" class="bright big-box" />`

Old HTML Helpers looked like C# code

- `@Html.ActionLink("Create New", "Create", "Student")`
- `@Html.LabelFor(student => student.FirstName, new {@class = “bright”})`
- `@Html.TextBoxFor(student => student.FirstName, new {@class = “bright big-box”})`

Benefits of Tag Helpers

An HTML-friendly development experience

- Tag Helpers looks like standard HTML.
- Front-end designers conversant with HTML/CSS/JavaScript can edit Razor without learning C#.

Rich IntelliSense for HTML and Razor markup

- HTML Helpers have limited IntelliSense

The markup is much cleaner and easier to read, edit, and maintain than the HTML Helpers approach.

It will make you more productive

How to use TagHelpers

Using the `@addTagHelper` directive

- Tag Helpers are optional features and not available in Razor Views by default
- You can enable it for any view by using the `@addTagHelper` directive at the top of the cshtml file
  - `@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers`
  - 1st parameter : * - Include all Tags in the namespace
  - 2nd parameter: The namespace of the
- You can enable it for all views by using the directive in the _ViewImports.cshtml file
- The common MVC Tag Helpers are found in the `Microsoft.AspNetCore.Mvc.TagHelpers` namespace.
- This directive is included in the project by default when you create a new project
- You can add additional Tag Helpers from public github or 3rd party nuget packages