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

Validation

Model Attributes

- [CreditCard]: Validates that the property has a credit card format. Requires jQuery Validation Additional Methods.
- [Compare]: Validates that two properties in a model match.
- [EmailAddress]: Validates that the property has an email format.
- [Phone]: Validates that the property has a telephone number format.
- [Range]: Validates that the property value falls within a specified range.
- [Required]: Validates that the field isn't null. See [Required] attribute for details about this attribute's behavior.
- [StringLength]: Validates that a string property value doesn't exceed a specified length limit.
- [Url]: Validates that the property has a URL format.

Validation – Server Side

The ModelState Object

- You can check the state of your Model using the IsValid property

Custom Validations

- E.g., `ModelState.AddModelError("CustomError", "Some message");`
- Or, `ModelState.AddModelError(“Name", "Some message");`

Validation - Client Side

The `asp-validation-for` attribute is used to display model validation errors on the client side

- E.g., `<span asp-validation-for="RollNo" class="text-danger"></span>`

The `asp-validation-summary` attribute is used to display a summary of validation errors in the Model

- E.g., `<div asp-validation-summary="All"></div>`

Enable client side validation

- Include the Validation js files.
- These are already included by MVC in partial view called _ValidationScriptsPartial.cshtml
- You can include it in any view as below

```
@section Scripts
{
    <partial name="_ValidationScriptsPartial" />
}
```

IActionResult

The `IActionResult` return type is appropriate when multiple `ActionResult` return types are possible

`IActionResult` is an interface
- `ActionResult` is an abstract class which implements `IActionResult`
- Other action results inherit from `ActionResult`

Some of the class which inherit from ActionResult are

Class	MVC Usage example
`ViewResult`	`return View(student)`
`RedirectResult`	`return Redirect("http://google.com")`
`RedirectToActionResult`	`return RedirectToAction("Index")`
`NotFoundResult`	`return NotFound()`
`BadRequestResult`	`return BadRequest()`
`OkResult`	`return Ok()`

TempData

TempData is a dictionary object to store data temporarily.

It is alive from the time it is created in a http request till

- the subsequent http request completes
- The value is retrieved

TempData is used to transfer data

- from the view to the controller,
- the controller to the view, or
- from an action method to another action method of the same or a different controller.

ViewState Sample

Example

Add ViewState to display success/error messages in a CRUD Module

Partial Views

A partial view is a Razor markup file (. cshtml) without an @page directive

It is used to render HTML output within another markup file's output.

It is similar to UserControls in Asp.NET Framework Web Forms

Partial Views are usually names as _<Name>Partial.cshtml

- E.g., _ValidationScriptsPartial.cshtml

Anti-forgery Tokens

Cross-Site Request Forgery (CSRF) is an attack where a malicious site sends a request to a vulnerable site where the user is currently logged in.

Here is an example of a CSRF attack

- A user logs into his bank site and the bank session is active on the browser
- Without logging out, the user visits a malicious website which has html as follows

```
<h1>You Are a Winner!</h1>
<p>Click on the button below to claim your prize</p>
<form action="http://icicibank.com/api/account" method="post">
<input type="hidden" name="Transaction" value="withdraw" />
<input type="hidden" name="Amount" value="1000000" />
<input type="submit" value="Click Me"/> </form>
```

- Note that the form action posts to the vulnerable site i.e., Cross-Site
- When the user clicks on submit button, the browser will load the bank site as the user is still logged on.

Anti-forgery Tokens

To help prevent CSRF attacks, ASP.NET MVC uses anti-forgery tokens, also called request verification tokens.

- The client requests an HTML page that contains a form.
- The server includes two tokens in the response.
  - The tokens are generated randomly so that an adversary cannot guess the values.
- When the client submits the form, it must send both tokens back to the server.
  - Browser client automatically does this when the user submits the form
- If a request does not include both tokens, the server disallows the request.

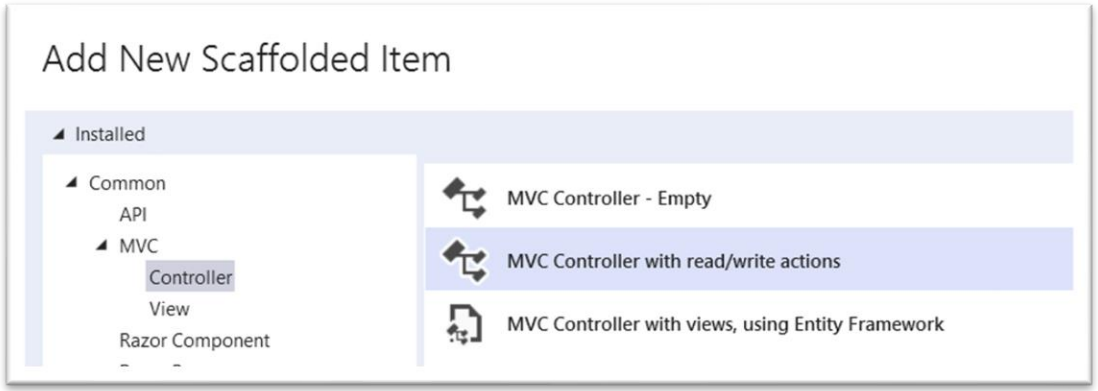
In a MVC controller, you can add anti-forgery token validation using a method/class attribute

- `[ValidateAntiForgeryToken]`

Scaffolding

Scaffolding

- is a technique used by many MVC frameworks like ASP.NET MVC, Ruby on Rails, Cake PHP and Node.JS etc.,
- It generates code for common application scenarios. For e.g.,
  - basic CRUD (create, read, update, and delete)
  - Identity (Login, Register, Logout, Forgot Password)
- Further you can edit or customize this auto generated code according to your need.



_layout

Any view that you load in the browser follows the following steps

- Load the content from _layout.cshtml
- In _layout.cshtml, at @RenderBody merge content from [View].cshtml
- In _layout.cshtml, at @RenderSectionAsync merge the content from the section inside [View].cshtml
  - If required: true, then every view should have the specified section
  - Else, the section is optional

Logging