

Programación Web .NET Core

Módulo 3

Introducción a Razor

Motor de vistas Razor

Si bien ya sabemos que la lógica de negocios de nuestra aplicación y el acceso a datos (archivos de texto, base de datos etc.) se pueden codificar en el **modelo**, cuando tenemos que implementar la presentación de datos en las vistas, es necesario utilizar el motor de vistas **Razor**.

- Razor permite implementar código C# dentro de las vistas.
- Los bloques en Razor se encierran entre los caracteres `@{ }` Todo lo que codificamos dentro de las llaves, será ejecutado en el servidor y **no se mostrará en el navegador**.

1. Crearemos un proyecto. Dentro de él, agregaremos un controlador, y dentro del controlador una vista para el método **Index()**. En la vista, agregaremos **código Razor**.



Nota: Podemos crear más de un bloque de código Razor dentro de la página.

2. Se modifica la vista ***Index.cshtml*** con el código:

```
@{
    Layout = null;
}
@{
    string titulo = "Página principal";
    int dia = DateTime.Now.Day;
    int mes = DateTime.Now.Month;
    int anio = DateTime.Now.Year;
    string texto;
    if (dia > 15)
    {
        texto = "Estamos en la segunda mitad del mes";
    }
}
```



```
else
{
    texto = "Estamos en la primer mitad del mes";
}
}
```

```
<!DOCTYPEhtml>
```

```
<html>
```

```
<head>
```

```
<metaname="viewport"content="width=device-width"/>
```

```
<title>Index</title>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<h1>@titulo</h1>
```

```
<h3>Fecha:@dia/@mes/@anio</h3>
```

```
<h4>@texto</h4>
```

```
</div>
```

```
</body>
```

```
</html>
```



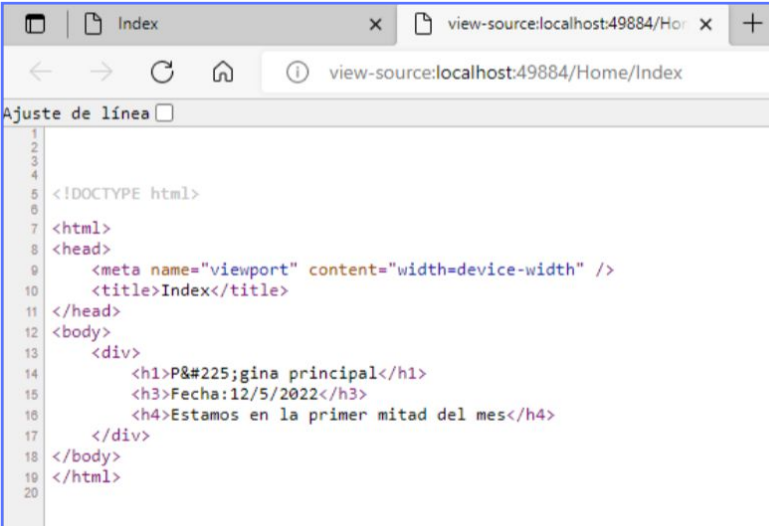
3. En el ejemplo, definimos un segundo bloque de código Razor donde se declaran cinco variables que se inicializan por asignación.
4. Para generar HTML dentro de la página, se debe definir fuera del código Razor la variable a mostrar, precedida por el **carácter @**.
5. Debemos tener en cuenta que **todos los bloques Razor “no se muestran” en el navegador web que hace la petición de la página.** Veamos la imagen de la derecha.



Si hacemos clic con el botón derecho del mouse dentro del navegador y seleccionamos la opción **Ver código fuente/origen de la página**, veremos lo siguiente:

Lo que sucede es que todos los bloques Razor que definimos, no se muestran en la página de vuelta al cliente y los elementos en los que agregamos las variables en Razor, por ejemplo **@titulo**, se reemplazaron por los valores contenidos en las variables.

Nota: Dentro de los bloques de código Razor no implementaremos código de acceso a bases de datos, archivos de texto, etc. ya que perdería sentido el patrón MVC.



```
1
2
3
4
5 <!DOCTYPE html>
6
7 <html>
8 <head>
9   <meta name="viewport" content="width=device-width" />
10  <title>Index</title>
11 </head>
12 <body>
13   <div>
14     <h1>P&#225;gina principal</h1>
15     <h3>Fecha:12/5/2022</h3>
16     <h4>Estamos en la primer mitad del mes</h4>
17   </div>
18 </body>
19 </html>
20
```



Acceso a estructuras de datos

Como vemos, podemos acceder a variables de tipos nativos (**int**, **string**, **float**, etc.), pero también a estructuras de datos, y definir estructuras del tipo **for**, **while**, **if**, etc. fuera del bloque Razor:

```
@{
    Layout = null;
}
@{
    string titulo = "Página principal";
    int dia = DateTime.Now.Day;
    int mes = DateTime.Now.Month;
    int año = DateTime.Now.Year;
    string texto;
    if (dia > 15)
    {
        texto = "Estamos en la segunda mitad del mes";
    }
}
```



```
else
{
    texto = "Estamos en la primer mitad del mes";
}
}
@{
    string[] cursos = { "C# para no programadores", "Introducción al Paradigma de Objetos",
        "HTML5: Fundamentos web", "Programación ASP .Net MVC" };
}
```



```
<!DOCTYPEhtml>

<html>
<head>
<metaname="viewport"content="width=device-width"/>
<title>Index</title>
</head>
<body>
<div>
<h1>@titulo</h1>
<h3>Fecha:@dia/@mes/@anio</h3>
<h4>@texto</h4>
</div>
```





```
<div>
<h1>Lista de Cursos:</h1>
<ul>
@foreach (string curso in cursos)
{
<p> - @curso</p>
}
</ul>
</div>
</body>
</html>
```

En este ejemplo, se crea un nuevo bloque Razor en el que se declara e instancia un **array** de tipo **string**.

6. Los elementos del array se muestran definiendo una instrucción **foreach** fuera del bloque. Para poder hacer esto, se debe anteponer el carácter **@** a la palabra clave **foreach**.

El motor Razor puede identificar cuál es el código HTML que debe generar y cuál es la variable a mostrar, ya que antepusimos el carácter **@** a la variable **curso**.



Estructuras iterativas en Razor

Como hemos visto, anteponiendo el carácter @ podemos definir cualquier estructura iterativa o condicional en la página HTML.

Para probar la facilidad de mezclar HTML y Razor, codificaremos la tabla de multiplicar de los números del 1 al 10, como se muestra en la siguiente pantalla.



```
<div>
<tablestyle="border: 1pxsolidblue; border-collapse: collapse">
@for (int fila = 1; fila <= 10; fila++)
    {
<tr>
@for (int columna = 1; columna <= 10; columna++)
    {
<tdstyle="border: 1pxsolidblue; border-collapse: collapse">@fila * @columna =
@ (fila * columna)</td>
    }
</tr>
    }
</table>
</div>
```

Página principal

Fecha:12/5/2022

Estamos en la primer mitad del mes

Lista de Cursos:

- C# para no programadores
- Introducción al Paradigma de Objetos
- HTML5: Fundamentos web
- Programación ASP .Net MVC

1 * 1 = 1	1 * 2 = 2	1 * 3 = 3	1 * 4 = 4	1 * 5 = 5	1 * 6 = 6	1 * 7 = 7	1 * 8 = 8	1 * 9 = 9	1 * 10 = 10
2 * 1 = 2	2 * 2 = 4	2 * 3 = 6	2 * 4 = 8	2 * 5 = 10	2 * 6 = 12	2 * 7 = 14	2 * 8 = 16	2 * 9 = 18	2 * 10 = 20
3 * 1 = 3	3 * 2 = 6	3 * 3 = 9	3 * 4 = 12	3 * 5 = 15	3 * 6 = 18	3 * 7 = 21	3 * 8 = 24	3 * 9 = 27	3 * 10 = 30
4 * 1 = 4	4 * 2 = 8	4 * 3 = 12	4 * 4 = 16	4 * 5 = 20	4 * 6 = 24	4 * 7 = 28	4 * 8 = 32	4 * 9 = 36	4 * 10 = 40
5 * 1 = 5	5 * 2 = 10	5 * 3 = 15	5 * 4 = 20	5 * 5 = 25	5 * 6 = 30	5 * 7 = 35	5 * 8 = 40	5 * 9 = 45	5 * 10 = 50
6 * 1 = 6	6 * 2 = 12	6 * 3 = 18	6 * 4 = 24	6 * 5 = 30	6 * 6 = 36	6 * 7 = 42	6 * 8 = 48	6 * 9 = 54	6 * 10 = 60
7 * 1 = 7	7 * 2 = 14	7 * 3 = 21	7 * 4 = 28	7 * 5 = 35	7 * 6 = 42	7 * 7 = 49	7 * 8 = 56	7 * 9 = 63	7 * 10 = 70
8 * 1 = 8	8 * 2 = 16	8 * 3 = 24	8 * 4 = 32	8 * 5 = 40	8 * 6 = 48	8 * 7 = 56	8 * 8 = 64	8 * 9 = 72	8 * 10 = 80
9 * 1 = 9	9 * 2 = 18	9 * 3 = 27	9 * 4 = 36	9 * 5 = 45	9 * 6 = 54	9 * 7 = 63	9 * 8 = 72	9 * 9 = 81	9 * 10 = 90
10 * 1 = 10	10 * 2 = 20	10 * 3 = 30	10 * 4 = 40	10 * 5 = 50	10 * 6 = 60	10 * 7 = 70	10 * 8 = 80	10 * 9 = 90	10 * 10 = 100

En este ejemplo, tenemos **dos estructuras iterativas anidadas**: dentro del **for** interno mostramos los contenidos de las dos variables de los **for** y para poder mostrar la multiplicación, debemos encerrarlas entre paréntesis.

Razor puede identificar dentro de la estructura repetitiva cuándo se trata de un elemento HTML, por ejemplo **<td>** (**celda**)



**¡Sigamos
trabajando!**

