educación IT
CODE YOUR FUTURE

Programación Web .NET Core

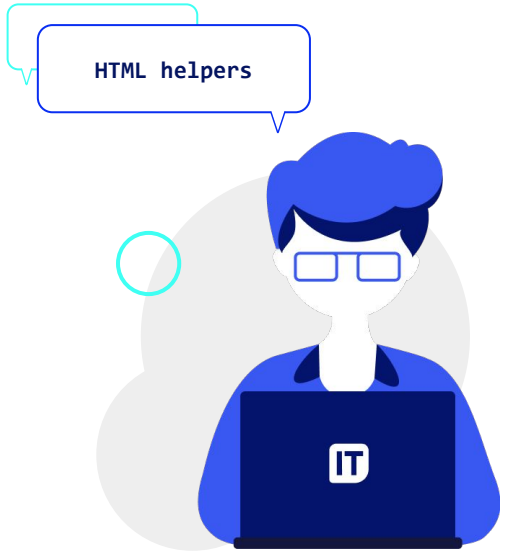
Módulo 6

Uso de HTML Helpers y Tag Helpers

Uso de HTML Helpers y Tag Helpers

HTML helpers

- Utiliza sintaxis de **Razor**.
- Facilita la identificación de áreas de código.
- No requiere la habilitación explícita.



Html.ActionLink()

```
@Html.ActionLink ("Clic aquí para ver la foto 1",
"Display",new{id=1})
```



```
<a href="/photo/display/1">Clic aquí para ver la
foto 1</a>
```

Url.Action()

```
<img alt ="Esta imagen proviene de una acción"
 src="@Url.Action("GetImage", new {id=1})"/>
```



```
<img
 alt="Esta imagen proviene de una acción"
 src="/photo/getimage/1"})"
 />
```

Tag Helpers

- Utiliza una sintaxis similar a HTML con **etiquetas como propiedades**.
- Requiere el uso explícito de una **directiva**.
- Crea código HTML legible.
- Son una **alternativa a los HTML Helpers**.
- Se ven como elementos HTML normales.

```
<form method="post"
      enctype="multipart/form-data"
      asp-action="Create">
```

El HTML Helper y el Tag Helper siguientes producen el mismo HTML:

HTML Helper:

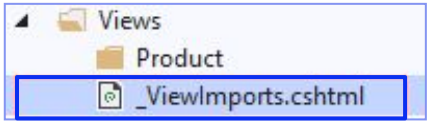
```
@Html.ActionLink("Presióname", "AnotherAction")
```

Tag Helper:

```
<a asp-action="AnotherAction">Presióname</a>
```

ASP.NET Core MVC incluye varios **Tag Helpers** predefinidos que se encuentran en el espacio de nombres cuya denominación es la siguiente: **`Microsoft.AspNetCore.Mvc.TagHelpers`**. Para usarlos en una vista, primero debes agregar la directiva **`@addTagHelper`**.

En forma alternativa, puedes agregar la directiva **`@addTagHelper`** a un archivo ***_ViewImports.cshtml*** que se encuentra a nivel de la carpeta ***Views***, de modo que esté disponible para todas las vistas.



Uno de los **Tag Helpers** predefinidos es **`Anchor`**. Está asociado con el elemento HTML **a**.

En el ejemplo que vimos en la pantalla anterior **`<a asp-action="AnotherAction"> Presióname </a>`**, la etiqueta **Tag Helper `Anchor`** es esencialmente una clase de **C#** que contiene varios atributos como **`asp-action`**, que indica el nombre de la acción, y **`asp-controller`**, que indica el nombre del controlador.

El ejemplo aquí debajo muestra cómo usar un **Tag Helper** y un **HTML Helper**, de modo que el mismo HTML es procesado en el navegador:

```
public class HomeController :Controller
{
   [Route("Home/Index")]
public IActionResult Index()
   {
return View();
   }
[Route("Home/AnotherAction")]
public IActionResult AnotherAction()
   {
return Content("AnotherActionresult");
   }
}
```

El siguiente ejemplo de código muestra el HTML **`HelperHtml.ActionLink`** seguido de Tag Helper:

```
@addTagHelper *,
Microsoft.AspNetCore.Mvc.TagHelpers
@Html.ActionLink("Presióname", "AnotherAction")
<a asp-action="AnotherAction">Presióname</a>
```

En el caso de que un usuario solicite la URL relativo **`/Home/Index`**, se enviará al navegador el HTML que vemos a continuación:

```
<a href="/Home/AnotherAction">Presióname</a>
<a href="/Home/AnotherAction">Presióname</a>
```

Observa que el HTML producido por el asistente de HTML y el asistente de etiquetas **son idénticos**.

Acciones de llamada en otros controladores

En el ejemplo anterior, vimos cómo se pueden usar el **HTML Helper** y el **Tag Helper** para llamar a una acción en el mismo controlador. También es posible usarlos para llamar a acciones en otros controladores. **El HTML producido para ambos *Helpers* es idéntico**.

El ejemplo a continuación muestra dos controladores. Cada controlador contiene una acción. La acción del primer controlador devuelve un objeto **`ViewResult`** y la acción del segundo devuelve un objeto **`ContentResult`**.

```
public class FirstController : Controller
{
   [Route("First/Some")]
   public IActionResult Some()
   {
        return View();
   }
}
```

```
public class SecondController : Controller
{
    [Route("Second/Index")]
    public IActionResult MyAction()
    {
          return Content("Second Controller");
    }
}
```

En el siguiente código se muestra el **HTML Helper** **`Html.ActionLink`** seguido de un **Tag Helper** para realizar acciones en otros controladores:

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@Html.ActionLink("Presiona el link", "MyAction", "Second")
<a asp-action="MyAction" asp-controller="Second">Presiona el
link</a>
```

Si un usuario solicita la URL relativa, **`/First/Some`**, se enviará al navegador el siguiente HTML:

```
<a href="/Second/Index">Presiona el link</a>
<a href="/Second/Index">Presiona el link</a>
```

Observa que el HTML producido por HTML Helper y el Tag Helper **son idénticos**.

Pasar parámetros a una acción

En los ejemplos anteriores, vimos cómo se usa el Tag Helper **Anchor** para llamar a una acción. En caso de que desees pasar parámetros a una acción, puedes recurrir al atributo cuyo nombre es **asp-route-{value}**.

El marcador de posición debe ser idéntico al nombre del parámetro de ruta.

El siguiente ejemplo de código muestra una clase de controlador con dos métodos de acción. El segundo método de acción, **Display**, obtiene un parámetro llamado **id** de tipo **int**:

```
public class PhotoController :Controller
{
   [Route("Photo/Choose")]
public IActionResultChoose()
   {
return View();
   }
   [Route("Photo/Display/{id}")]
public IActionResult Display(int id)
   {
string res = $"Photonumber {id} fue elegida”;
return Content(res);
    }
}
```

En el ejemplo debajo el Tag Helper **Anchor** se utiliza con un atributo **asp-route- {id}**. Si el usuario hace clic en el enlace, se llamará a la acción **Display** en el controlador **PhotoController**, y se pasará un valor entero al parámetro **id**:

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
<a asp-action="Display" asp-route-id="1"> Clic aquí para ver la foto 1</a>
<a asp-action="Display" asp-route-id="2"> Clic aquí para ver la foto 2</a>
```

Si un usuario solicita la URL relativa **`/Photo/Display`**, se enviará al navegador el siguiente HTML:

```
<a href="/Photo/Display/1">Clic aquí par aver la foto 1</a>
<a href="/Photo/Display/2">Clic aquí par aver la foto 2</a>
```

Si el usuario hace clic en el enlace **`Clic aquí para ver la foto 1`**, aparecerá el texto ***Se eligió la foto número 1*** en el navegador. Si el usuario hace clic en el enlace **`Clic aquí para ver la foto 2`**, el texto ***Se eligió la foto número 2*** aparece en el navegador.

Uso del archivo *_ViewImports.cshtml*

Los Tag Helpers solo estarán disponibles en la vista en la que agregues la directiva llamada **`@addTagHelper`**. Si deseas que estén disponibles para todas las vistas en la carpeta ***Views*** y sus subdirectorios, debes agregar la directiva cuyo nombre es **`@addTagHelper`** a un archivo ***_ViewImports.cshtml***.

El archivo ***_ViewImports.cshtml*** debe estar ubicado en la carpeta ***Views***.

Por ejemplo, en lugar de agregar la directiva **`@addTagHelper`** al archivo ***Index.cshtml***:

```
@addTagHelper *,
Microsoft.AspNetCore.Mvc.TagHelpers
<a asp-action="AnotherAction">Presióname</a>
```

Puedes agregarla al archivo ***_ViewImports.cshtml*** en la carpeta ***Views***:

El archivo ***Views /_ViewImports.cshtml***

```
@addTagHelper *,
Microsoft.AspNetCore.Mvc.TagHelpers
```

En este caso, no necesitas más la directiva **`@addTagHelper`** en el archivo ***Index.cshtml***. Se puede eliminar:

```
<a asp-action="AnotherAction"> Presióname </a>
```

`TagHelper Anchor` es uno de los Tag Helpers integrados. Hay otros Tag Helpers integrados, como **`TagHelper Input`**, que está asociado con el elemento HTML de entrada, y **`TagHelper ValidationMessage`**, que se utiliza para mostrar mensajes de error de validación. También se pueden crear **Tag Helpers personalizados**. Todos los Tag Helpers implementan la interfaz **`ITagHelper`**, y muchos Tag Helpers heredan de la clase **`TagHelper`**.

Uso de la directiva @addTagHelper

Para usar tag helper, debes agregar la directiva **`@addTagHelper`** a una vista:

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Para que el tag helper esté disponible para todas las vistas, **agrega la directiva `@addTagHelper` al archivo *_ViewImports.cshtml***.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!