Agenda: Working with Tag Helpers

- About Tag Helpers
- Build-In Tag Helpers
- Anchor and Image Tag Helpers
- Form and other Input Tag Helpers
- Binding Tag Helpers to Model
- Custom Tag Helpers

About Tag Helpers

Tag Helpers enable **server-side code** to create, update and render HTML in the Razor view file. They are said to be an evaluation of Html Helpers. Tag Helpers are attached to HTML tags without effecting raw view.

How Tag Helpers are different from Html Helpers:

1. If you were creating a form and wanted to use the standard Bootstrap form-horizontal styling, you need to add a control-label class to each of your HTML label elements.

```
@Html.Label("Email", "Email address", new { @class = "control-label" })
```

This isn't exactly HTML friendly markup, because all of the parameters are strings, IntelliSense can't help, and we have to use an anonymous class to represent the attributes (remembering to use the @ prefix on class so C# interprets it as a property name and not the class keyword).

Using a Tag Helper, the same markup is

```
<label class="control-label" asp-for="Email"></label>
```

This looks like standard HTML markup, with a new attribute. This attribute is provided by a Tag Helper, in this case the **LabelTagHelper**, which knows how to turn it in to a properly formatted label tag.

2. Tag Helpers markup actually looks like HTML markup and is easier to read and maintain than the Html Helpers approach.

Managing Tag Helper Scope:

In Views/_ViewImports.cshtml:

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

The **@addTagHelper** directive makes Tag Helpers available to the view.

The first parameter specifies the Tag Helpers to load (we are using "*" for all Tag Helpers), and the second parameter "**Microsoft.AspNetCore.Mvc.TagHelpers**" specifies the assembly containing the Tag Helpers.

For custom tag helpers:

```
@addTagHelper MyDemoApp.TagHelpers.DemoTagHelper, MyDemoApp
```

OR

@addTagHelper MyDemoApp.TagHelpers.D*, MyDemoApp

To remove a Tag Helper that was previously added.:

@removeTagHelper MyDemoApp.TagHelpers.DemoTagHelper, MyDemoApp

For example, @removeTagHelper applied to a specific view removes the specified Tag Helper from the view.

Opting out of individual elements:

You can disable a Tag Helper at the element level with the Tag Helper opt-out character ("!"). For example, `Email` validation is disabled in the `<span>` with the Tag Helper opt-out character:

```
<!span asp-validation-for="Email" class="text-danger"></!span>
```

Using @tagHelperPrefix to make Tag Helper usage explicit:

This directive allows you to specify a tag prefix string to enable Tag Helper support and to make Tag Helper usage explicit.

For example, you could add the following markup to the *Views/_ViewImports.cshtml* file:

@tagHelperPrefix th:

```
<div class="form-group">
    <th:label asp-for="Password" class="col-md-2"></th:label>
    <div class="col-md-10">
        <th:input asp-for="Password" class="form-control" />
        <span asp-validation-for="Password" class="text-danger"></span>
    </div>
</div>
```

Following table shows the list of commonly used tag helpers:

asp-controller	It is used to specify controller to be used.
asp-action	It is used to specify action method to be used.
asp-area	It is used to specify the area to be used.
asp-for	It is used to specify the property to be used (only in Strongly Typed Views).
asp-route	It is used to specify named route instead of controller and action method.
asp-validation-for	It is used to specify the property to be validated.
asp-validation-summary	It is used to specify the mode of validation summary to be used.
asp-antiforgery	It is used to enable or disable anti-forgery.
asp-append-version	It is used to add a version number to the resource, so if there is any change in resource then version number is also changed. It ensures that in the case of any change one resource is not used from the cache.

ASP.NET Core provides a set of predefined Tag Helper classes in **Microsoft.AspNetCore.Mvc.TagHelpers** namespace

- AnchorTagHelper
- ImageTagHelper
- FormTagHelper
- InputTagHelper
- LabelTagHelper
- SelectTagHelper
- TextAreaTagHelper
- ValidationMessageTagHelper
- ValidationSummaryTagHelper
- EnvironmentTagHelper
- PartialTagHelper

AnchorTagHelper

The <a> taghelper defines a hyperlink, which is used to link from one page to another. It allows data element to be mapped a tag through asp-area, asp-controller, asp-action, asp-route to specify area, controller, action method and route. We can also pass parameters asp-route- construct.

```
<a asp-controller="Home" asp-action="index" asp-route-id="1" asp-protocol="https"
 asp-host="microsoft.com" asp-fragment="top" >Get Details</a>
to
https://microsoft.com/Home/index/1#top
```

Note: asp-route-**{parameter name}**:

asp-route:

```
[Route("/Speaker/Evaluations", Name = "speakerevals")]
public IActionResult Evaluations(){
 return View();
}

@{
var parms = new Dictionary<string, string>
        {
          { "speakerId", "11" },
          { "currentYear", "true" }
        };
}
```

```
<a asp-route="speakerevals" asp-all-route-data="parms">Speaker Evaluations</a>
Transforms to
<a href="/Speaker/EvaluationsCurrent?speakerId=11&currentYear=true">Speaker Evaluations</a>
```

Razor Pages related attributes:

```
public class AttendeeModel : PageModel
{
   public void OnGet(int attendeeid)
   {   }
   public void OnGetProfile(int attendeeid)
   {   }
}
<a asp-page="/Attendee"
  asp-page-handler="Profile"
  asp-route-attendeeid="12">Attendee Profile</a>
transforms to
<a href="/Attendee?attendeeid=12">Attendee Profile</a> => Maps to OnGet method
<a href="/Attendee?attendeeid=12@handler=Profile">Attendee Profile</a> => Maps to OnGetProfile method
```

ImageTagHelper

Cache Busting:

The <img> tag helper defines an image in an HTML page. It allows to specify asp-append-version, which is used to address image cache problem.

```
<img asp-append-version="true" src="~/images/logo.png" alt="Loading..." />
transfroms to
<img src="/images/logo.png?v=Kl_dqr9NVtnMdsM2MUg4qthUnWZm5T1fCEimBPWDNgM"/>
```

The value assigned to the parameter v is the hash value of the *logo.png* file on disk. If the web server is unable to obtain read access to the static file, no **v** parameter is added to the **src** attribute in the rendered markup.

Hash caching behavior:

The Image Tag Helper uses the cache provider on the local web server to store the calculated Sha512 hash of a given file. If the file is requested multiple times, the hash isn't recalculated. The cache is invalidated by a file watcher that's attached to the file when the file's Sha512 hash is calculated. When the file changes on disk, a new hash is calculated and cached.

FormTagHelper

The <form> tag helper is used to create an HTML form for user input. It allows to specify the controller, action method. It also allows to specify route name instead of controller and action. Furthermore, it provides services to handle cross-site request forgery. It is important to remember that on submit action form uses name fields of inner objects to create response body.

```
<form asp-controller="Home" asp-action="Index" asp-antiforgery="true" method="post">
      ...
</form>
```

InputTagHelper:

The <input> tag specifies an input field where the user can enter data. It allows data element to be mapped input tag through an **asp-for** attribute in **strongly typed view**. While input tag type is decided through data element type and data annotation. While data validation rules are applied through data annotation.

```
<input asp-for="EmailId" class="form-control" />
```

LabelTagHelper:

LabelTagHelper is used with a label tag. It allows data element to be mapped input tag through an asp-for attribute. It uses Display data annotation value of specified data element to display label. If Display attribute has not been applied then element name is used.

```
<label asp-for="EmailId" class="control-label">EmailId</label>
```

SelectTagHelper:

SelectTagHelper is used with select tag. It allows data element to be mapped select tag and option through asp-for and asp-items attributes, asp-for is used to specify selected value element and asp-items is used to specify list to be bounded with options. While data validation rules are applied through data annotation.

```
<select asp-for="Title" asp-items="Model.Titles"></select>
```

- **Title** is of type **String**
- **Titles** is of type **List<SelectedListItem>**

Multiple select: The Select Tag Helper will automatically generate the **multiple = "multiple"** attribute if the property specified in the **asp-for** attribute is an **IEnumerable**.

```
<select asp-for="Titles" asp-items="Model.Titles"></select>
```

TextAreaTagHelper:

TextAreaTagHelper is used with textarea tag. It allows data element to be mapped textarea tag through an asp-for attribute. While data validation rules are applied through data annotation.

```
<textarea asp-for="Description" rows="5" cols="30" ></textarea>
```

ValidationMessageTagHelper:

ValidationMessageTagHelper is used with a span tag. It allows validation messages mapped to span tag through an asp-validation-for attribute.

```
<span asp-validation-for="EmailId" class="text-danger" ></span>
```

```
@section Scripts {
  @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

ValidationSummaryTagHelper:

ValidationSummaryTagHelper is used with div tag. It allows all validation messages mapped to div tag filtered through an asp-validation-summary attribute.

Possible values for asp-validation-summary as **All, ModelOnly, None**.

```
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
```

Binding Tag Helpers to Model

Add the following code to the Model

```
public class Employee
{
  public int Id { get; set; }
  public string Name { get; set; }
  public decimal Salary { get; set; }
  public string Email { get; set; }
  public string Gender { get; set; }
  public DateTime DateOfJoin { get; set; }
  public bool IsActive { get; set; }
  public string Country { get; set; }
  public List<SelectListItem> Countries { get; } = new List<SelectListItem>
  {
    new SelectListItem { Value = "IN", Text = "India" },
    new SelectListItem { Value = "UK", Text = "United Kingdom" },
    new SelectListItem { Value = "US", Text = "USA"  },
  };
}
```

Add the following code to the Controller

```
public IActionResult Index()
{
  Employee emp = new Employee()
  {
    Id = 1,
    Name = "Sandeep",
    Salary = 20000,
    DateOfJoin = DateTime.Now,
    Email = "sandeep@deccansoft.com",
    IsActive = true,
    Country = "IN"
  };
  return View(emp);
}
```

Add the following code to the view

```
<form asp-action="Index" asp-controller="Home" method="post" class="form-horizontal" role="form">
  <div class="form-group">
    <label asp-for="Id" class="col-md-2 control-label"></label>
    <div class="col-md-10">
      <input asp-for="Id" class="form-control" />
    </div>
  </div>
  <div class="form-group">
    <label asp-for="Name" class="col-md-2 control-label"></label>
    <div class="col-md-10">
      <input asp-for="Name" class="form-control" />
    </div>
  </div>
  <div class="form-group">
    <label asp-for="Salary" class="col-md-2 control-label"></label>
    <div class="col-md-10">
```

```
        <input asp-for="Salary" class="form-control" />
    </div>
</div>
<div class="form-group">
    <label asp-for="Email" class="col-md-2 control-label"></label>
    <div class="col-md-10">
        <input asp-for="Email" class="form-control" />
    </div>
</div>
<div class="form-group">
    <label asp-for="Gender" class="col-md-2 control-label"></label>
    <div class="col-md-10">
        <input asp-for="Gender" type="radio" value="Male" /><label for="Gender">Male</label>
        <input asp-for="Gender" type="radio" value="Female" /><label for="Gender">Female</label>
    </div>
</div>
<div class="form-group">
    <label asp-for="DateOfJoin" class="col-md-2 control-label"></label>
    <div class="col-md-10">
        <input asp-for="DateOfJoin" class="form-control" />
    </div>
</div>
<div class="form-group">
    <label asp-for="IsActive" class="col-md-2 control-label"></label>
    <div class="col-md-10">
        <input asp-for="IsActive" />
    </div>
</div>
<div class="form-group">
    <div class="col-md-10">
        <div class="col-md-3"></div>
        <input type="submit" name="btnSubmit" value="Submit" class="btn-success" />
    </div>
</div>
<div class="form-group">
```

```
        <label asp-for="Country" class="col-md-2 control-label"></label>
        <div class="col-md-10">
            <select asp-for="Country"  asp-items="Model.Countries" class="form-control" />
        </div>
    </div>
</form>
<div>
    Id: @Model.Id              <br>
    Name: @Model.Name    <br>
    Salary: @Model.Salary    <br>
    Email:@Model.Email       <br>
    DateOfJoin:@Model.DateOfJoin <br>
    IsActive:@Model.IsActive <br>
</div>
```

EnvironmentTagHelper

Bundling and minification is an important optimization technique for managing scripts and stylesheets in any web application. ASP.NET Core MVC takes a different approach

```
<environment names="Development">
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
</environment>
<environment names="Staging,Production">
    <link rel="stylesheet" href="~/css/site.min.css" asp-file-version="true" />
</environment>
```

In Development, the HTML is as follows:

```
<link rel="stylesheet" href="/css/site.css" />
```

ScriptTagHelper

You are only likely to use this helper if you use a CDN version of a script file and your site will be unusable in the event that it is not available. The test expression that you specify should resolve to true if the CDN version of the script is available.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"
        asp-fallback-src="~/lib/jquery/dist/jquery.min.js"
        asp-fallback-test="window.jQuery">
```

```
</script>
```

Globbing: is a way to specify sets of filename using wildcard patterns

We could include all the individual script files above using a single script tag helper:

```
<script asp-src-include="~/app/**/*.js"></script>
```

This is generate <script> tag for every file qualifying that condition

Cache Busting: Cache busting is the process of appending some form of file version hash to the filename of resources like JavaScript and CSS files.

```
<script asp-append-version="true"> </script>
Renders as
<script src="~/lib/jquery/dist/jquery.min.js?v=RFDfEE3j1E3DgwL6NiDG"></script>
```

Partial Tag Helper

From ASP.NET Core 2.1 onwards, the Partial Tag Helper is used for rendering a partial view in Razor Pages and MVC apps. It renders the partial view asynchronously.

```
<partial name="Shared/_ProductPartial.cshtml" asp-for="Product" />
```

Note: asp-for="Product" can be used instead of asp-for="@Model.Product"

Custom Tag Helpers

Tag Helpers provide a clean way to encapsulate view code so you can keep your views simple and markup focused. There are lots of built in Tag Helpers in the **Microsoft.AspNetCore.Mvc.TagHelpers** class, but one of the best features is that you can easily create your own helpers.

Add Class:

```
namespace HtmlHelpersDemoApp
{
  public class AppreciateTagHelper: TagHelper
  {
    private const string appreciateText = "Great Work!";
    public string PersonNameForAppreciation { get; set; }
    public override void Process(TagHelperContext context, TagHelperOutput output)
    {
      output.TagName = "span";
      var appreciateMsg = appreciateText + ", " + PersonNameForAppreciation;
```

```
            output.Content.SetContent(appreciateMsg);
        }
    }
}
```

Note: The Process() method gives us two very important parameters for creating output.

- The first parameter, **context**, gives us access to the control and any contained elements and attributes. This allows us to programmatically manipulate it by adding, modifying, or removing child elements.
- The second parameter, **output**, is used to write content back into the webpage.

In View:

```
<appreciate person-name-for-appreciation="Sandeep Soni"></appreciate>
```

We can do this in individual Razor files or we can add it to **_ViewImports.cshtml** file so it is applied everywhere:

```
@addTagHelper HtmlHelpersDemoApp.AppreciateTagHelper, HtmlHelpersDemoApp
Or
@addTagHelper *,HtmlHelpersDemoApp
```

Pascal-cased class and property names for tag helpers are translated into their lower kebab case. Therefore, to use the **PersonNameForAppreciation** attribute, you'll use **person-name-for-appreciation** attribute.

Example2:

Add the following class to the project.

```
[HtmlTargetElement("email", TagStructure = TagStructure.WithoutEndTag)]
public class EmailTagHelper : TagHelper
{
    private const string EmailDomain = "deccansoft.com";
    // Can be passed via <email mail-to="..." />.
    // Pascal case gets translated into lower-kebab-case.
    public string MailTo { get; set; }
    public override void Process(TagHelperContext context, TagHelperOutput output)
    {
        output.TagName = "a";    // Replaces <email> with <a> tag
        var address = MailTo + "@" + EmailDomain;
        output.Attributes.SetAttribute("href", "mailto:" + address);
        output.Content.SetContent(address);
    }
}
```

In .cshtml

```
@addTagHelper WebApplication1.EmailTagHelper, WebApplication1


<strong>Support:</strong><email mail-to="Support"></email><br />

<strong>Marketing:</strong><email mail-to="Marketing"></email>
```

- This version uses the asynchronous **ProcessAsync** method. The asynchronous **GetChildContentAsync** returns a Task containing the TagHelperContent.