**ASP.NET Core MVC**

**Authoring Custom Tag Helpers**

# Getting started with Tag Helpers

This tutorial provides an introduction to programming Tag Helpers.

A tag helper is any class that implements the ITagHelper interface. However, when you author a tag helper, you generally derive from TagHelper, doing so gives you access to the Processmethod. We will introduce the TagHelper methods and properties as we use them in this tutorial.

1. Create a new ASP.NET Core project called **AuthoringTagHelpers**. You won’t need authentication for this project.
2. Create a folder to hold the Tag Helpers called *TagHelpers*. The *TagHelpers* folder is *not*required, but it is a reasonable convention. Now let’s get started writing some simple tag helpers.

# Starting the email Tag Helper

In this section we will write a tag helper that updates an email tag. For example:

<email>Support</email>

The server will use our email tag helper to convert that markup into the following:

<a href**=**"mailto:Support@contoso.com">Support@contoso.com</a>

That is, an anchor tag that makes this an email link. You might want to do this if you are writing a blog engine and need it to send email for marketing, support, and other contacts, all to the same domain.

1. Add the following EmailTagHelper class to the *TagHelpers* folder.

**using** Microsoft.AspNetCore.Razor.TagHelpers;

**using** System.Threading.Tasks;

**namespace** AuthoringTagHelpers.TagHelpers

{

**public** **class** **EmailTagHelper** : TagHelper

{

**public** **override** **void** **Process**(TagHelperContext context, TagHelperOutput output)

{

output.TagName = "a"; *// Replaces <email> with <a> tag*

}

}

}

**Notes:**

* Tag helpers use a naming convention that targets elements of the root class name (minus the*TagHelper* portion of the class name). In this example, the root name of **Email**TagHelper is*email*, so the <email> tag will be targeted. This naming convention should work for most tag helpers, later on I’ll show how to override it.
* The EmailTagHelper class derives from TagHelper. The TagHelper class provides methods and properties for writing Tag Helpers.
* The overridden Process method controls what the tag helper does when executed. TheTagHelper class also provides an asynchronous version (ProcessAsync) with the same parameters.
* The context parameter to Process (and ProcessAsync) contains information associated with the execution of the current HTML tag.
* The output parameter to Process (and ProcessAsync) contains a stateful HTML element representative of the original source used to generate an HTML tag and content.
* Our class name has a suffix of **TagHelper**, which is *not* required, but it’s considered a best practice convention. You could declare the class as:

**public** **class** **Email** : TagHelper

1. To make the EmailTagHelper class available to all our Razor views, we will add the addTagHelperdirective to the *Views/\_ViewImports.cshtml* file:

@using AuthoringTagHelpers

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

@addTagHelper "AuthoringTagHelpers.TagHelpers.EmailTagHelper, AuthoringTagHelpers"

The code above uses the wildcard syntax to specify all the tag helpers in our assembly will be available. The first string after @addTagHelper specifies the tag helper to load (we are using “\*” for all tag helpers), and the second string “AuthoringTagHelpers” specifies the assembly the tag helper is in. Also, note that the second line brings in the ASP.NET Core MVC tag helpers using the wildcard syntax (those helpers are discussed in Introduction to Tag Helpers.) It’s the @addTagHelperdirective that makes the tag helper available to the Razor view. Alternatively, you can provide the fully qualified name (FQN) of a tag helper as shown below:

@using AuthoringTagHelpers

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

@addTagHelper "AuthoringTagHelpers.TagHelpers3.EmailTagHelper, AuthoringTagHelpers"

To add a tag helper to a view using a FQN, you first add the FQN (AuthoringTagHelpers.TagHelpers.EmailTagHelper), and then the assembly name (*AuthoringTagHelpers*). Most developers will prefer to use the wildcard syntax. Introduction to Tag Helpers goes into detail on tag helper adding, removing, hierarchy, and wildcard syntax.

1. Update the markup in the *Views/Home/Contact.cshtml* file with these changes:

@{

ViewData["Title"] = "Contact";

}

<h2>@ViewData["Title"].</h2>

<h3>@ViewData["Message"]</h3>

<address>

One Microsoft Way<br />

Redmond, WA 98052<br />

<abbr title**=**"Phone">P:</abbr>

425.555.0100

</address>

<address>

<strong>Support:</strong><email>Support</email><br />

<strong>Marketing:</strong><email>Marketing</email>

</address>

1. Run the app and use your favorite browser to view the HTML source so you can verify that the email tags are replaced with anchor markup (For example, <a>Support</a>). *Support* and*Marketing* are rendered as a links, but they don’t have an href attribute to make them functional. We’ll fix that in the next section.

**Note**

Like HTML tags and attributes, tags, class names and attributes in Razor, and C# are not case-sensitive.

# A working email Tag Helper

In this section, we’ll update the EmailTagHelper so that it will create a valid anchor tag for email. We’ll update it to take information from a Razor view (in the form of a mail-to attribute) and use that in generating the anchor.

Update the EmailTagHelper class with the following:

**public** **class** **EmailTagHelper** : TagHelper

{

**private** **const** **string** EmailDomain = "contoso.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);

}

}

**Notes:**

* Pascal-cased class and property names for tag helpers are translated into their lower kebab case. Therefore, to use the MailTo attribute, you’ll use <email mail-to="value"/> equivalent.
* The last line sets the completed content for our minimally functional tag helper.
* The highlighted line shows the syntax for adding attributes:

**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);

}

That approach works for the attribute “href” as long as it doesn’t currently exist in the attributes collection. You can also use the output.Attributes.Add method to add a tag helper attribute to the end of the collection of tag attributes.

1. Update the markup in the *Views/Home/Contact.cshtml* file with these changes:

@{

ViewData["Title"] = "Contact Copy";

}

<h2>@ViewData["Title"].</h2>

<h3>@ViewData["Message"]</h3>

<address>

One Microsoft Way Copy Version <br />

Redmond, WA 98052-6399<br />

<abbr title**=**"Phone">P:</abbr>

425.555.0100

</address>

<address>

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

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

</address>

1. Run the app and verify that it generates the correct links.

**Note:** If you were to write the email tag self-closing (<email mail-to="Rick" />), the final output would also be self-closing. To enable the ability to write the tag with only a start tag (<email mail-to="Rick">) you must decorate the class with the following:

[HtmlTargetElement("email", TagStructure = TagStructure.WithoutEndTag)]

**public** **class** **EmailVoidTagHelper** : TagHelper

{

**private** **const** **string** EmailDomain = "contoso.com";

*// Code removed for brevity*

With a self-closing email tag helper, the output would be <a href="mailto:Rick@contoso.com" />. Self-closing anchor tags are not valid HTML, so you wouldn’t want to create one, but you might want to create a tag helper that is self-closing. Tag helpers set the type of the TagMode property after reading a tag.

# An asynchronous email helper

In this section we’ll write an asynchronous email helper.

1. Replace the EmailTagHelper class with the following code:

**public** **class** **EmailTagHelper** : TagHelper

{

**private** **const** **string** EmailDomain = "contoso.com";

**public** **override** **async** Task **ProcessAsync**(TagHelperContext context, TagHelperOutput output)

{

output.TagName = "a"; *// Replaces <email> with <a> tag*

**var** content = **await** output.GetChildContentAsync();

**var** target = content.GetContent() + "@" + EmailDomain;

output.Attributes.SetAttribute("href", "mailto:" + target);

output.Content.SetContent(target);

}

}

**Notes:**

* This version uses the asynchronous ProcessAsync method. The asynchronousGetChildContentAsync returns a Task containing the TagHelperContent.
* We use the output parameter to get contents of the HTML element.

1. Make the following change to the *Views/Home/Contact.cshtml* file so the tag helper can get the target email.

@{

ViewData["Title"] = "Contact";

}

<h2>@ViewData["Title"].</h2>

<h3>@ViewData["Message"]</h3>

<address>

One Microsoft Way<br />

Redmond, WA 98052<br />

<abbr title**=**"Phone">P:</abbr>

425.555.0100

</address>

<address>

<strong>Support:</strong><email>Support</email><br />

<strong>Marketing:</strong><email>Marketing</email>

</address>

1. Run the app and verify that it generates valid email links.

EmailTagHelper.cs

using Microsoft.AspNetCore.Razor.TagHelpers;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Day4Demos.MyTagHelpers

{

public class EmailTagHelper : TagHelper

{

public string EmailDomain { get; set; }

public string MailTo { get; set; }

public override void Process(TagHelperContext context,

TagHelperOutput output)

{

output.TagName = "a";

var address = MailTo + "@" + EmailDomain;

output.Attributes.SetAttribute("href", "mailto:" + address);

output.Content.SetContent(address);

}

}

}

Copyright.cshtml

@using Day4Demos

@addTagHelper "Day4Demos.MyTagHelpers.EmailTagHelper,Day4Demos"

<h3><i>&copy;Srini Iyer, 2017</i></h3>

<**email** **mail-to**="Support" **email-domain**="mytest.com"></**email**>