



# Tag Helpers

---

.NET CORE

***Tag Helpers*** enable server-side code to participate in creating and rendering HTML elements in Razor files.

[HTTPS://DOCS.MICROSOFT.COM/EN-US/ASPNET/CORE/MVC/VIEWS/TAG-HELPERS](https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers)

# Tag Helpers vs HTML Helpers

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/intro?view=aspnetcore-3.1#what-are-tag-helpers>  
<https://www.tutorialrepublic.com/html-tutorial/html-elements.php>

---

**Tag Helpers** are preferable to HTML Helpers for a variety of reasons. They:

- are written in C#
- target HTML *elements* based on *element* name, *attribute* name, or parent *tag*
- reduce explicit transitions between HTML and C#
- provide a HTML-friendly environment because the syntax is still familiar to front-end designers.
- provide *IntelliSense*.
- target standard HTML *elements*
- provide *attributes* created server-side for the *element*.

**HTML Helpers** can be used when a specific **Tag Helper** is unavailable. **Tag Helpers** don't replace **HTML Helpers** and there's not a **Tag Helper** equivalent for every **HTML Helper**.

TagHelper:

```
<label asp-for="Movie.Title"></label>
```

HTML Helper:

```
<label for="Movie_Title">Title</label>
```

# <Form> Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-3.1#the-form-tag-helper>

---

- Generates a hidden Request Verification Token to prevent ***cross-site request forgery (CSRF)*** when used with the ***[ValidateAntiForgeryToken]*** attribute in the HTTP Post ***action*** method.
- The Alternative, ***Html.BeginForm***, doesn't automatically include anti-forgery token. It's used with a ***using*** statement to wrap the form

```
<form asp-controller="Demo" asp-action="Register" method="post">  
    <!-- Input and Submit elements -->  
</form>
```

# <label> Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-3.1#the-label-tag-helper>

---

The `<label>` tag helper `<label asp-for="Name">`, gives the label for a model value. It's html helper equivalent is *Html.LabelFor*.

The `<label>` Tag Helper provides the following benefits over a pure HTML label element:

- Get the descriptive label value from the [Display] attribute in the Model.
- Less markup in source code
- Strong typing with the Model property.

```
using System.ComponentModel.DataAnnotations;

namespace FormsTagHelper.ViewModels
{
    public class SimpleViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email Address")]
        public string Email { get; set; }
    }
}
```

```
@model SimpleViewModel

<form asp-controller="Demo" asp-action="RegisterLabel" method="post">
    <label asp-for="Email"></label>
    <input asp-for="Email" /> <br />
</form>
```

# <Input> Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-3.1#the-input-tag-helper>

- The **<input>** Tag Helper binds an HTML **<input>** element to a model expression.
- **<input asp-for="Name" />** is equal to **@Html.EditorFor(m => m.Name)**
- If a **model** is passed to the **view**, the form control will begin already populated with the model's values.
- Generates HTML5 validation attributes from **data annotation attributes** applied to the **models** properties.

```
using System.ComponentModel.DataAnnotations;

namespace FormsTagHelper.ViewModels
{
    public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email Address")]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }
    }
}
```

```
@model RegisterViewModel

<form asp-controller="Demo" asp-action="RegisterInput" method="post">
    Email: <input asp-for="Email" /> <br />
    Password: <input asp-for="Password" /><br />
    <button type="submit">Register</button>
</form>
```

# Validation Message Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-3.1#the-validation-message-tag-helper>

---

- Used for client-side validation and the results of server-side validation when form is rejected.
- `<span asp-validation-for="">` is equal to html helper *Html.ValidationMessageFor*
- Leverages *model data attributes* and jQuery to display validation error messages in the body of the `<span>` element and prevent the form from being submitted.
- It will also display server-side *ModelState* errors.

```
<span asp-validation-for="Email"></span>
```



# Validation Summary Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-3.1#the-validation-summary-tag-helper>

asp-validation	Validation messages
= All	Property and model level
= ModelOnly	Model
None	None

- *asp-validation-summary* is equal to html helper *Html.ValidationSummary*.
- also displays server-side *ModelState* errors.
- used to display a summary of validation messages.

```
using System.ComponentModel.DataAnnotations;

namespace FormsTagHelper.ViewModels
{
    public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Email Address")]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }
    }
}
```

```
@model RegisterViewModel

<form asp-controller="Demo" asp-action="RegisterValidation" method="post">
    <div asp-validation-summary="ModelOnly"></div>
    Email: <input asp-for="Email" /> <br />
    <span asp-validation-for="Email"></span><br />
    Password: <input asp-for="Password" /><br />
    <span asp-validation-for="Password"></span><br />
    <button type="submit">Register</button>
</form>
```



# <select> Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-3.1#the-select-tag-helper>

---

The **<select>** tag helper is equal to **Html.DropDownListFor**

You give it the list of items and the property to put the selected item into.

The list should be some **IEnumerable<SelectListItem>** (e.g. **SelectList**) property on the *model*.

```
using Microsoft.AspNetCore.Mvc.Rendering;
using System.Collections.Generic;

namespace FormsTagHelper.ViewModels
{
    public class CountryViewModel
    {
        public string Country { get; set; }

        public List<SelectListItem> Countries { get; } = new List<SelectListItem>
        {
            new SelectListItem { Value = "MX", Text = "Mexico" },
            new SelectListItem { Value = "CA", Text = "Canada" },
            new SelectListItem { Value = "US", Text = "USA" },
        };
    }
}
```

```
@model CountryViewModel

<form asp-controller="Home" asp-action="Index" method="post">
    <select asp-for="Country" asp-items="Model.Countries"></select>
    <br /><button type="submit">Register</button>
</form>
```

# <a> (anchor) Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/built-in/anchor-tag-helper?view=aspnetcore-3.1#anchor-tag-helper-attributes>

---

- The <a> Tag Helper is equal to *Html.ActionLink*. It is used to create a link to specify a **controller/action** method.
- If the **asp-controller** attribute is specified and **asp-action** isn't, the default **asp-action** value is the **action** associated with the currently executing **view**.

```
<a asp-controller="Speaker"
    asp-action="Index">All Speakers</a>
```

```
<a asp-controller="Speaker"
    asp-action="Evaluations">Speaker Evaluations</a>
```

# <a> (anchor) Tag Helper

<https://docs.microsoft.com/en-us/aspnet/core/mvc/views/tag-helpers/built-in/anchor-tag-helper?view=aspnetcore-3.1#anchor-tag-helper-attributes>

---

- asp-route-id enables a wildcard route prefix. Any value occupying the {value} placeholder in the URL is interpreted as a potential route parameter. If a default route isn't found, this route prefix is appended to the generated href attribute as a request parameter and value.

```
@model Speaker
<!DOCTYPE html>
<html>
<body>
    <a asp-controller="Speaker"
        asp-action="Detail"
        asp-route-id="@Model.SpeakerId">SpeakerId: @Model.SpeakerId</a>
</body>
</html>
```