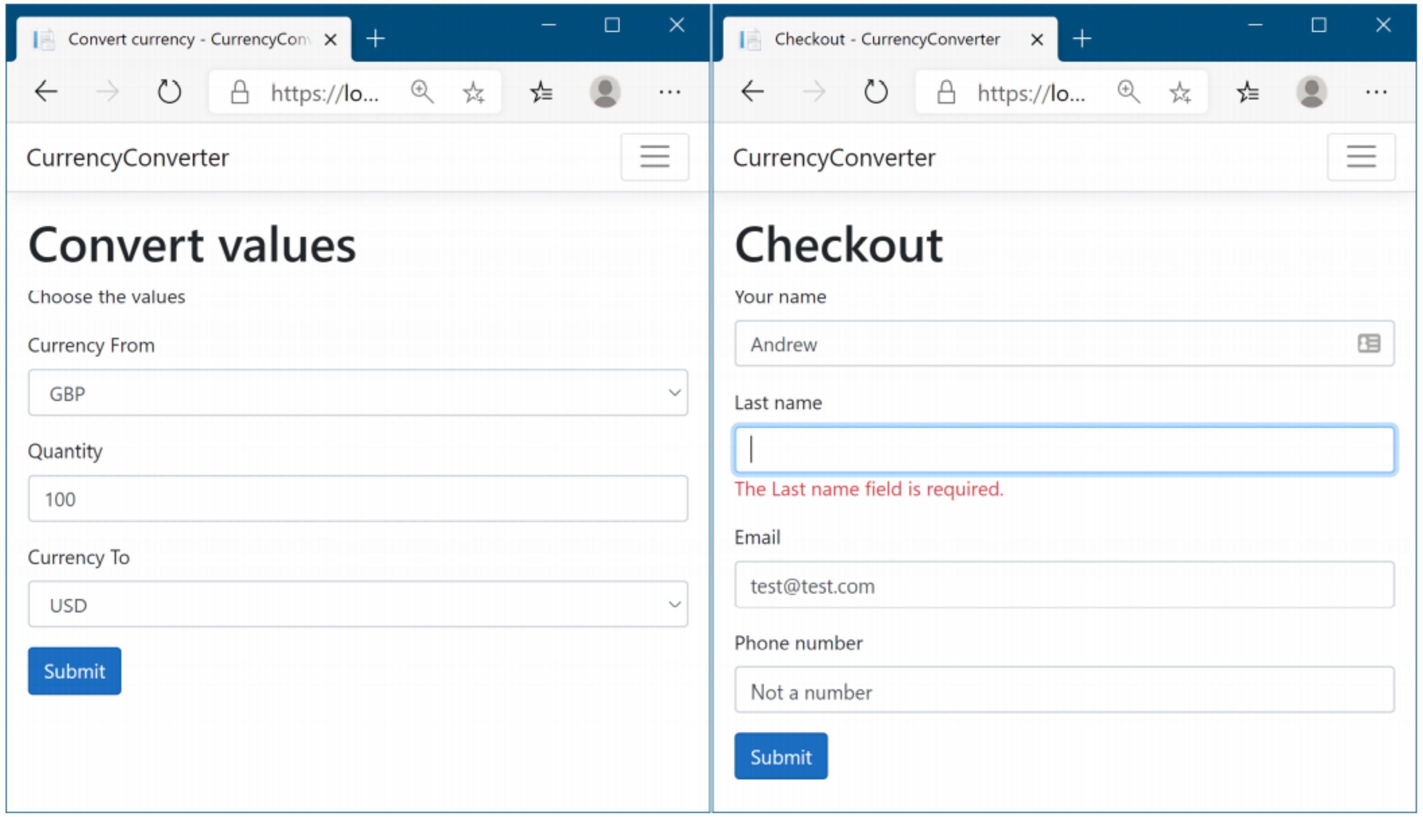
ASP.NET CORE –BUILDING FORMS WITH TAG HELPERS

Kamal Beydoun
Lebanese University – Faculty of Sciences I
Kamal.Beydoun@ul.edu.lb

INTRODUCTION

- Tag Helpers are new to ASP.NET Core and for **server-side** HTML rendering.
- They're **Razor components** that you can use to **customize** the **HTML** generated in your **templates**.
- Tag Helpers can be **added to a standard HTML element**, such as an <input>.
- They can also be **standalone elements** and can be used to generate completely customized HTML.

APPLICATION OF THE CHAPTER


Convert currency - CurrencyCon
https://lo...
CurrencyConverter

Convert values

Choose the values

Currency From
GBP

Quantity
100

Currency To
USD

Submit

Checkout - CurrencyConverter
https://lo...
CurrencyConverter

Checkout

Your name
Andrew

Last name

The Last name field is required.

Email
test@test.com

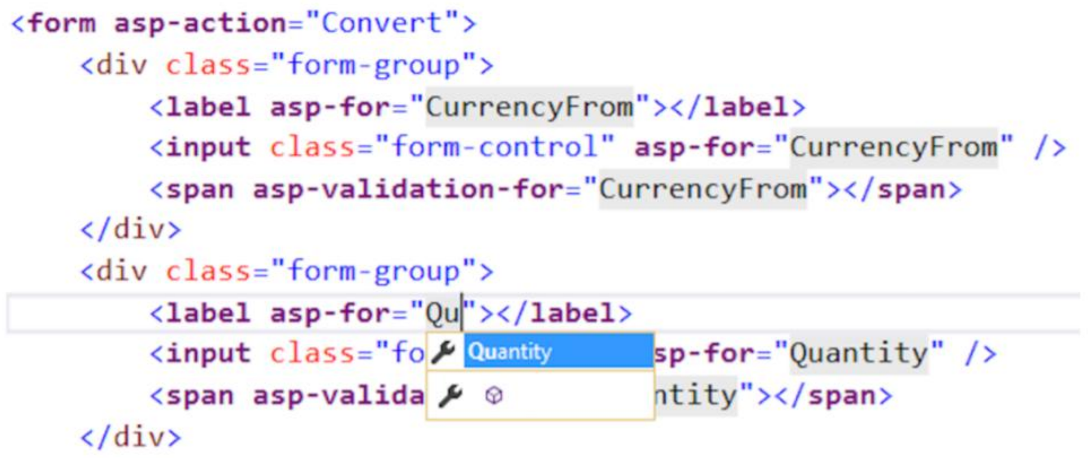
Phone number
Not a number

Submit

```
@page                                                                  #A
@model ConvertModel                                                    #A
<form method="post">
    <div class="form-group">
        <label asp-for="CurrencyFrom"></label>                         #B
        <input class="form-control" asp-for="CurrencyFrom" />          #C
        <span asp-validation-for="CurrencyFrom"></span>                #D
    </div>
    <div class="form-group">
        <label asp-for="Quantity"></label>                             #B
        <input class="form-control" asp-for="Quantity" />              #C
        <span asp-validation-for="Quantity"></span>                    #D
    </div>
    <div class="form-group">
        <label asp-for="CurrencyTo"></label>                           #B
        <input class="form-control" asp-for="CurrencyTo" />            #C
        <span asp-validation-for="CurrencyTo"></span>                  #D
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

#A This is the view for the Razor Page Convert.cshtml. The Model type is ConvertModel.
#B asp-for on Labels generates the caption for labels based on the view model.
#C asp-for on Inputs generate the correct type, value, name, and validation attributes for the model.
#D Validation messages are written to a span using Tag Helpers.

TAG HELPERS VS HTML ELEMENTS



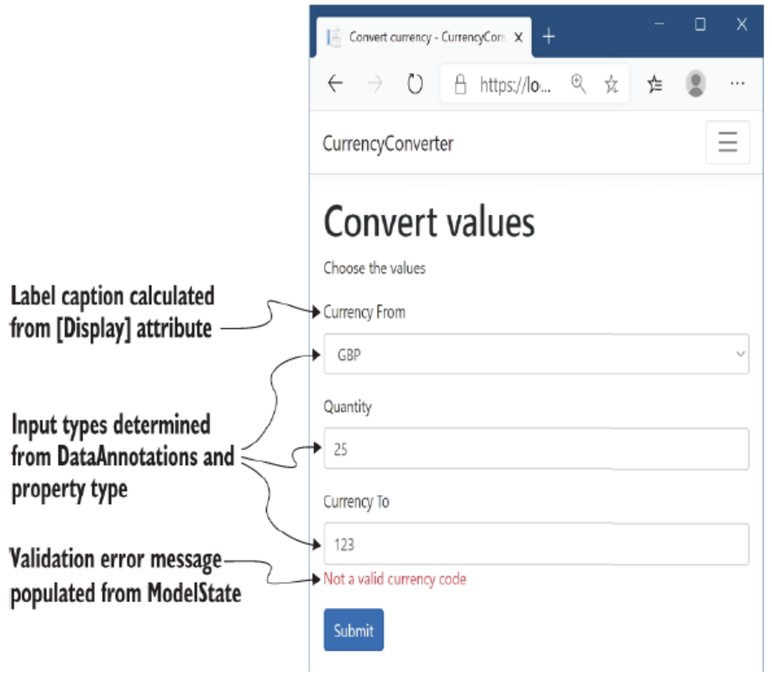
```
<form asp-action="Convert">
    <div class="form-group">
        <label asp-for="CurrencyFrom"></label>
        <input class="form-control" asp-for="CurrencyFrom" />
        <span asp-validation-for="CurrencyFrom"></span>
    </div>
    <div class="form-group">
        <label asp-for="Qu"></label>
        <input class="fo                 sp-for="Quantity" />
        <span asp-valida                 ntity"></span>
    </div>
```

In Visual Studio, Tag Helpers are distinguishable from normal elements by being **bold** and a **different color**, C# is shaded, and IntelliSense is available.

Tag Helpers are **extra attributes** on standard HTML elements (**or new elements entirely**) that work by modifying the HTML element they're attached to.

WHY TAG HELPERS ?

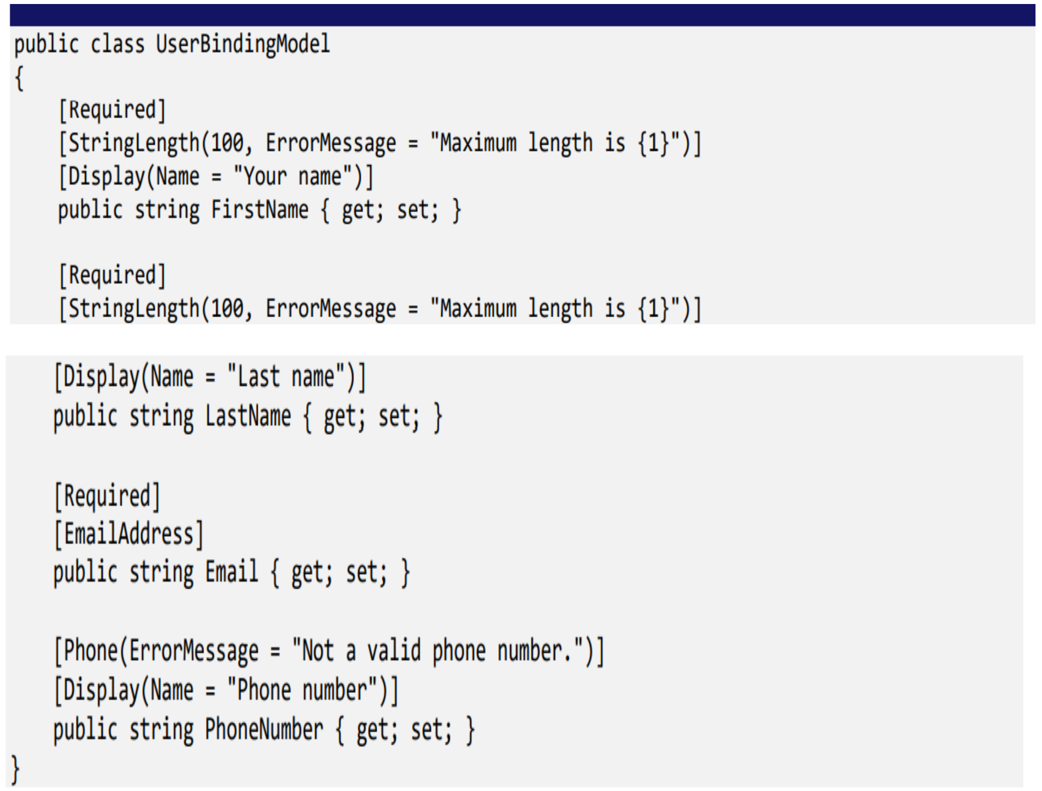
- Let us easily **integrate** server-side values, such as those in your **PageModel**, **with the generated HTML**.
- Tag Helpers are used to
  - Automatically **populate** the values from the PageModel property.
  - Choose the correct **id and name**, so that when the form is POSTed back to the Razor Page, the property will be model bound correctly.
  - Choose the **correct input type** to display.
  - Display any **validation** errors.


Convert currency - CurrencyCon
https://lo...
CurrencyConverter
Convert values
Choose the values
Label caption calculated from [Display] attribute
Currency From
GBP
Input types determined from DataAnnotations and property type
Quantity
25
Currency To
123
Validation error message populated from ModelState
Not a valid currency code
Submit

CREATING FORMS USING TAG HELPERS

- You can use Tag Helpers
  - to **generate HTML markup** based on **properties** of your PageModel,
  - creating the correct **id** and **name** attributes of the element
  - **setting the value** of the element to the model property's value

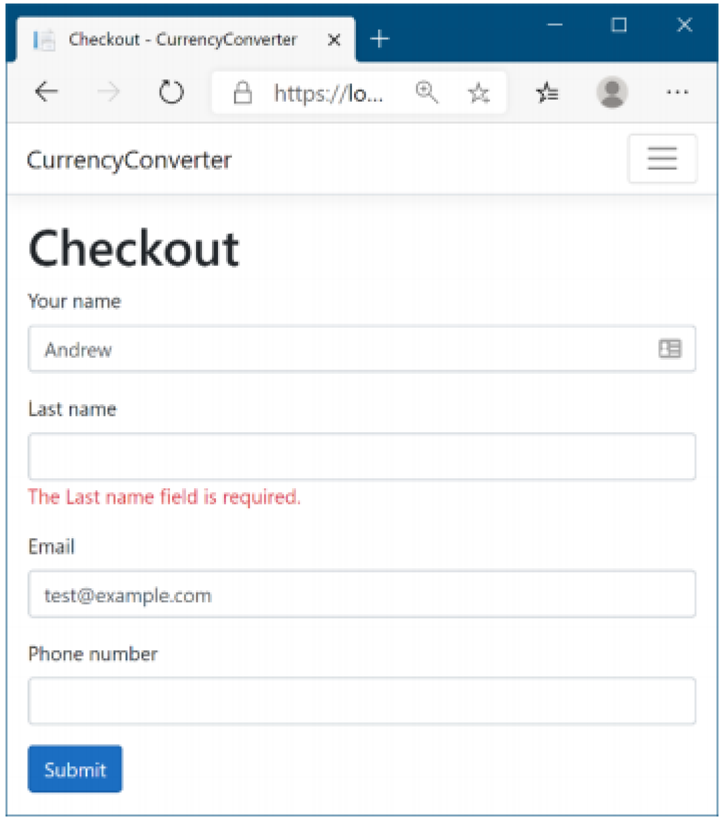
DATAANNOTATIONS ATTRIBUTES



```
public class UserBindingModel
{
    [Required]
    [StringLength(100, ErrorMessage = "Maximum length is {1}")]
    [Display(Name = "Your name")]
    public string FirstName { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "Maximum length is {1}")]
    [Display(Name = "Last name")]
    public string LastName { get; set; }

    [Required]
    [EmailAddress]
    public string Email { get; set; }

    [Phone(ErrorMessage = "Not a valid phone number.")]
    [Display(Name = "Phone number")]
    public string PhoneNumber { get; set; }
}
```

These attributes are also used by the Razor templating language to provide the **metadata required** to **generate the correct HTML** when you use Tag Helpers.

For simplicity, We are using the same object for both binding model and view model, but in practice you should use two separate objects.



Checkout

Your name

Andrew

Last name

The Last name field is required.

Email

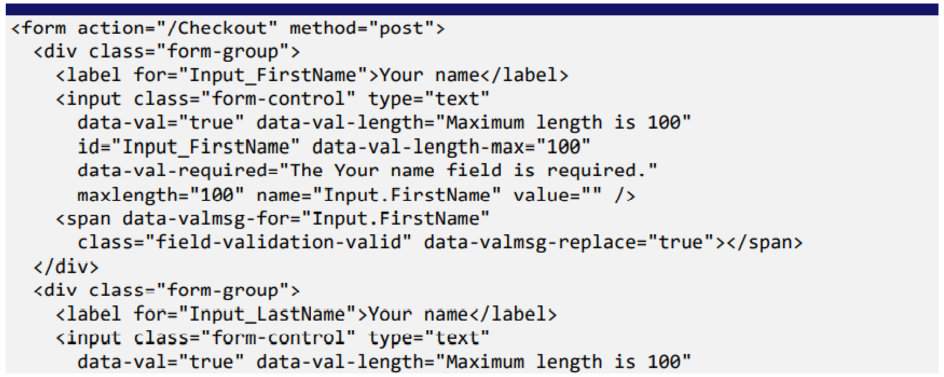
test@example.com

Phone number

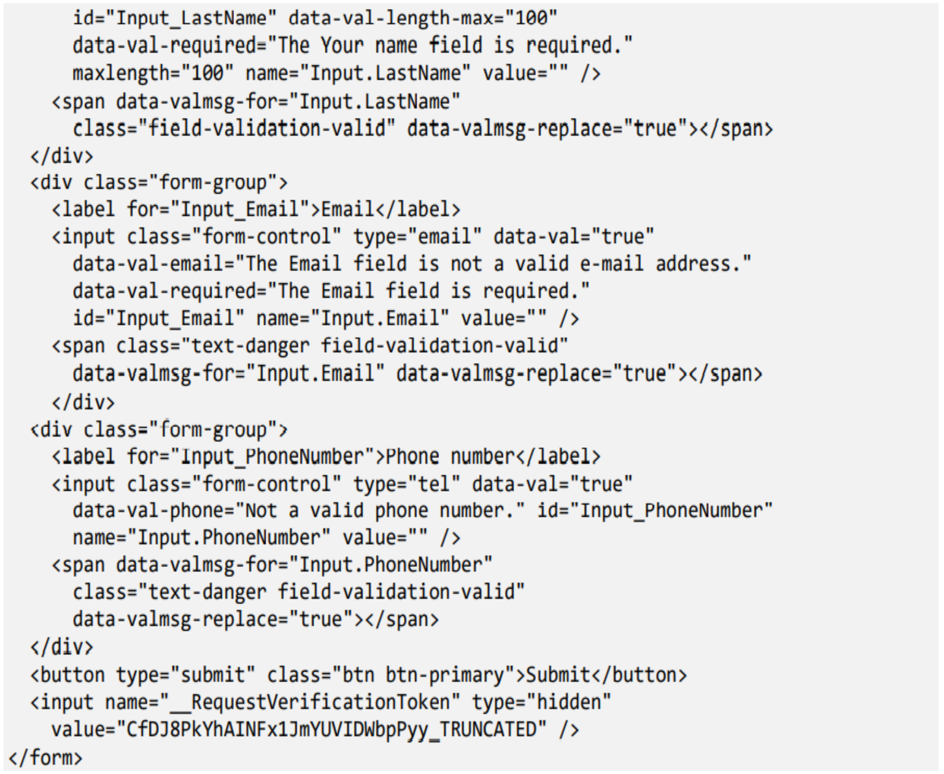
Submit

```
@page
@model CheckoutModel                                              #A
@{
    ViewData["Title"] = "Checkout";
}
<h1>@ViewData["Title"]</h1>
<form asp-page="Checkout">                                        #B
    <div class="form-group">
        <label asp-for="Input.FirstName"></label>                 #C
        <input class="form-control" asp-for="Input.FirstName" />
        <span asp-validation-for="Input.FirstName"></span>
    </div>
    <div class="form-group">
        <label asp-for="Input.LastName"></label>
        <input class="form-control" asp-for="Input.LastName" />
        <span asp-validation-for="Input.LastName"></span>
    </div>
    <div class="form-group">
        <label asp-for="Input.Email"></label>
        <input class="form-control" asp-for="Input.Email" />          #D
        <span asp-validation-for="Input.Email"></span>
    </div>
    <div class="form-group">
        <label asp-for="Input.PhoneNumber"></label>
        <input class="form-control" asp-for="Input.PhoneNumber" />
        <span asp-validation-for="Input.PhoneNumber"></span>          #E
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

#A The CheckoutModel is the PageModel, which exposes a UserBindingModel on the Input property
#B Form Tag Helpers use routing to determine the URL the form will be posted to.
#C The Label Tag Helper uses DataAnnotations on a property to determine the caption to display.
#D The Input Tag Helper uses DataAnnotations to determine the type of input to generate.
#E The Validation Tag Helper displays error messages associated with the given property.



```
<form action="/Checkout" method="post">
  <div class="form-group">
    <label for="Input_FirstName">Your name</label>
    <input class="form-control" type="text"
      data-val="true" data-val-length="Maximum length is 100"
      id="Input_FirstName" data-val-length-max="100"
      data-val-required="The Your name field is required."
      maxlength="100" name="Input.FirstName" value="" />
    <span data-valmsg-for="Input.FirstName"
      class="field-validation-valid" data-valmsg-replace="true"></span>
  </div>
  <div class="form-group">
    <label for="Input_LastName">Your name</label>
    <input class="form-control" type="text"
      data-val="true" data-val-length="Maximum length is 100"
```



```
      id="Input_LastName" data-val-length-max="100"
      data-val-required="The Your name field is required."
      maxlength="100" name="Input.LastName" value="" />
    <span data-valmsg-for="Input.LastName"
      class="field-validation-valid" data-valmsg-replace="true"></span>
  </div>
  <div class="form-group">
    <label for="Input_Email">Email</label>
    <input class="form-control" type="email" data-val="true"
      data-val-email="The Email field is not a valid e-mail address."
      data-val-required="The Email field is required."
      id="Input_Email" name="Input.Email" value="" />
    <span class="text-danger field-validation-valid"
      data-valmsg-for="Input.Email" data-valmsg-replace="true"></span>
    </div>
  <div class="form-group">
    <label for="Input_PhoneNumber">Phone number</label>
    <input class="form-control" type="tel" data-val="true"
      data-val-phone="Not a valid phone number." id="Input_PhoneNumber"
      name="Input.PhoneNumber" value="" />
    <span data-valmsg-for="Input.PhoneNumber"
      class="text-danger field-validation-valid"
      data-valmsg-replace="true"></span>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
  <input name="__RequestVerificationToken" type="hidden"
    value="CfDJ8PkYhAINFx1JmYUVIDWbpPyy_TRUNCATED" />
</form>
```

Even better than this, you can also set attributes that are normally generated by a Tag Helper, like the type attribute on an <input> element.

ASP-PAGE

```
<form asp-page="Checkout">
```

This resulted in the addition of **action** and **method** attributes to the final HTML, indicating the **URL that the form should be sent** to when submitted:

```
<form action="/Checkout" method="post">
```

If you omit the asp-page attribute, the form will post back to the same URL address it was served from. This is very common with Razor Pages.

ASP-ROUTE-*

➢ Used to **set arbitrary route parameters.**

```
<form asp-page ="Product" asp-route-id="5">
```

Will generate:

```
<form action="/Product/5" method="post">
```

- You can add **as many** asp-route-* attributes as necessary to your <form> to generate the correct action URL.
- You can also set the Razor Page **handler** to use using the **asp-page-handler** attribute.

THE LABEL TAG HELPER

- Used to generate the **caption** (the visible text) and the **for attribute** for a **<label>** element, based on the properties in the view model.
- [**Display**] DataAnnotations attribute determines the appropriate value to display.

```
public class UserBindingModel
    {
     [Display(Name = "Your name")]
    public string FirstName { get; set; }
    public string Email { get; set; }
    }
```

The following Razor:

```
<label asp-for="FirstName"></label>
<label asp-for="Email"></label>
```

would generate the HTML

```
<label for="FirstName">Your Name</label>
<label for="Email">Email</label>
```

THE LABEL TAG HELPER

As well as properties on the PageModel, you can also reference sub-properties on child objects.

```
public class CheckoutModel: PageModel
{
    [BindProperty]
    public UserBindingModel Input { get; set; }
}
```

```
<label asp-for="Input.FirstName"></label>
<label asp-for="Input.Email"></label>
```

THE LABEL TAG HELPER

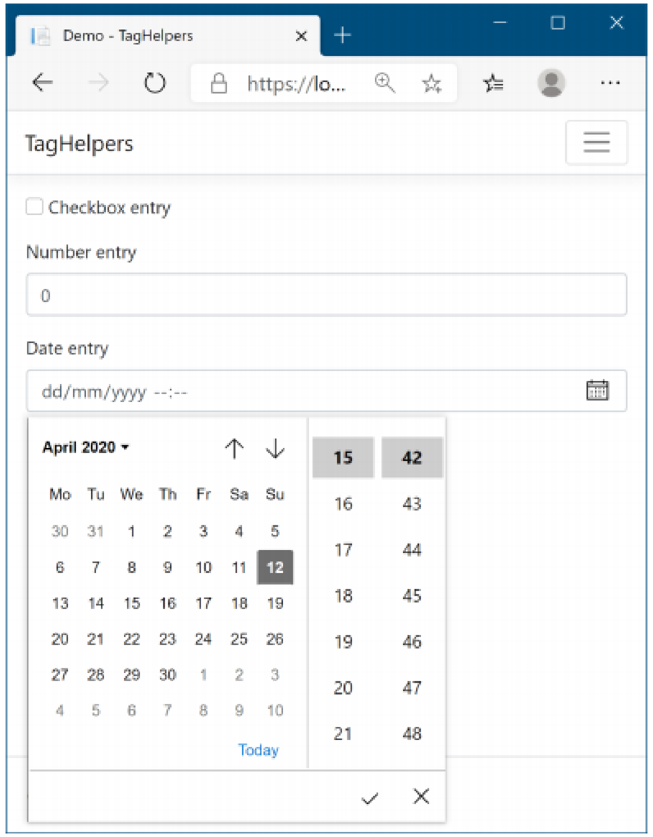
- If you **don’t want to use the caption generated by the helper**, you could insert your own manually.

```
<label asp-for="Email">Please enter your Email</label>
```

would generate the HTML

```
<label for="Email">Please enter your Email</label>
```

THE INPUT AND TEXTAREA TAG HELPERS



THE INPUT AND TEXTAREA TAG HELPERS

- To determine the **type of the input** element to generate, Tag Helpers uses information based on:
  - the **type of the property** (bool, string, int, and so on)
  - **DataAnnotations attributes** applied to the property
- **DataAnnotations** are also used to add data-val-* **client-side validation** attributes to the generated HTML.

EXAMPLE

```
        <input asp-for="“Input.Email" />
```

Will generate

```
<input type="email" id="“Input_Email" name="“Input.Email"
value="test@example.com" data-val="true“
data-val-email="The Email Address field is not a valid e-mail address."
 data-val-required="The Email Address field is required."
 />
```

Perhaps the most striking addition is the swath of data-val-* attributes. These can be used by client-side JavaScript libraries such as jQuery to provide client-side validation of your DataAnnotations constraints.

Client-side validation

In order to enable client-side validation in your application, you need to add some jQuery libraries to your HTML pages. In particular, you need to include the jQuery, jQuery-validation, and jQuery-validation-unobtrusive JavaScript libraries. You can do this in a number of ways, but the simplest is to include the script files at the bottom of your view using

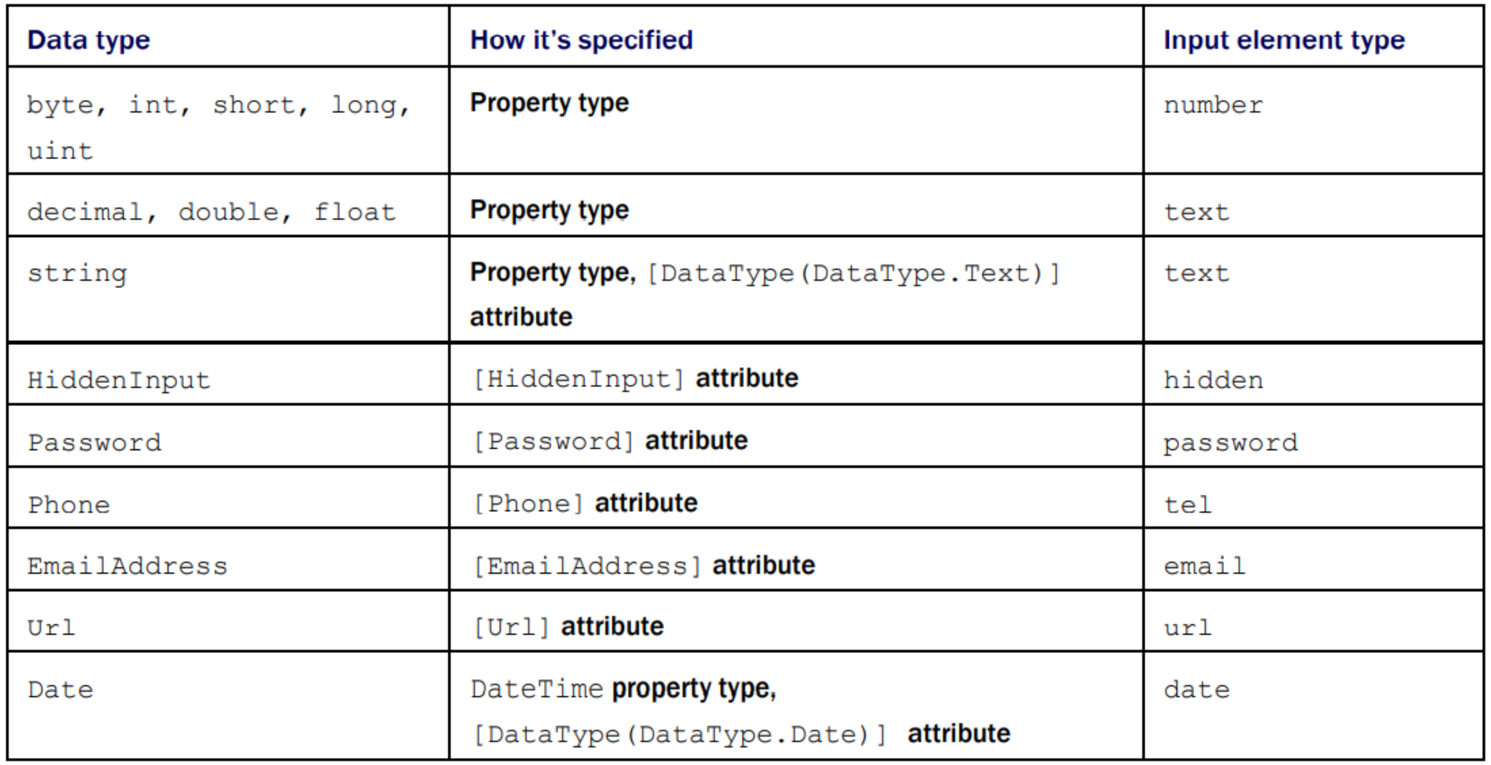
```
<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-
unobtrusive/jquery.validate.unobtrusive.min.js"></script>
```

The default templates include these scripts for you, in a handy partial template that you can add to your page in a `Scripts` section. If you're using the default layout and need to add client-side validation to your view, add the following section somewhere on your view:

```
@section Scripts{
    @Html.Partial("_ValidationScriptsPartial")
}
```

This partial view references files in your wwwroot folder. The default _layout template includes jQuery itself, as that's required by the front-end component library Bootstrap.[35]

INPUT TAG HELPERS



Data type	How it's specified	Input element type
byte, int, short, long, uint	**Property type**	number
decimal, double, float	**Property type**	text
string	**Property type,** [DataType(DataType.Text)] **attribute**	text
HiddenInput	[HiddenInput] **attribute**	hidden
Password	[Password] **attribute**	password
Phone	[Phone] **attribute**	tel
EmailAddress	[EmailAddress] **attribute**	email
Url	[Url] **attribute**	url
Date	DateTime **property type,** [DataType(DataType.Date)] **attribute**	date

As always, you can override the generated type by adding your own type attribute to the Razor.

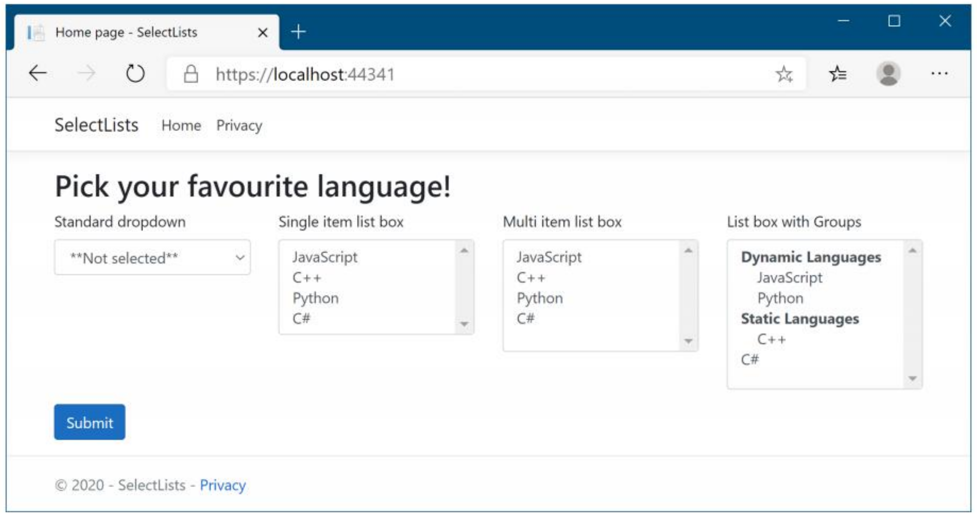
TEXT AREA TAG HELPER

```
<textarea asp-for="Multiline"></textarea>
```

generates

```
<textarea data-val="true" id="Multiline" name="Multiline"
data-val-length="Maximum length 200." data-val-length-max="200"
data-val-required="The Multiline field is required." >This is some text,
I'm going to display it in a text area</textarea>
```

THE SELECT TAG HELPER



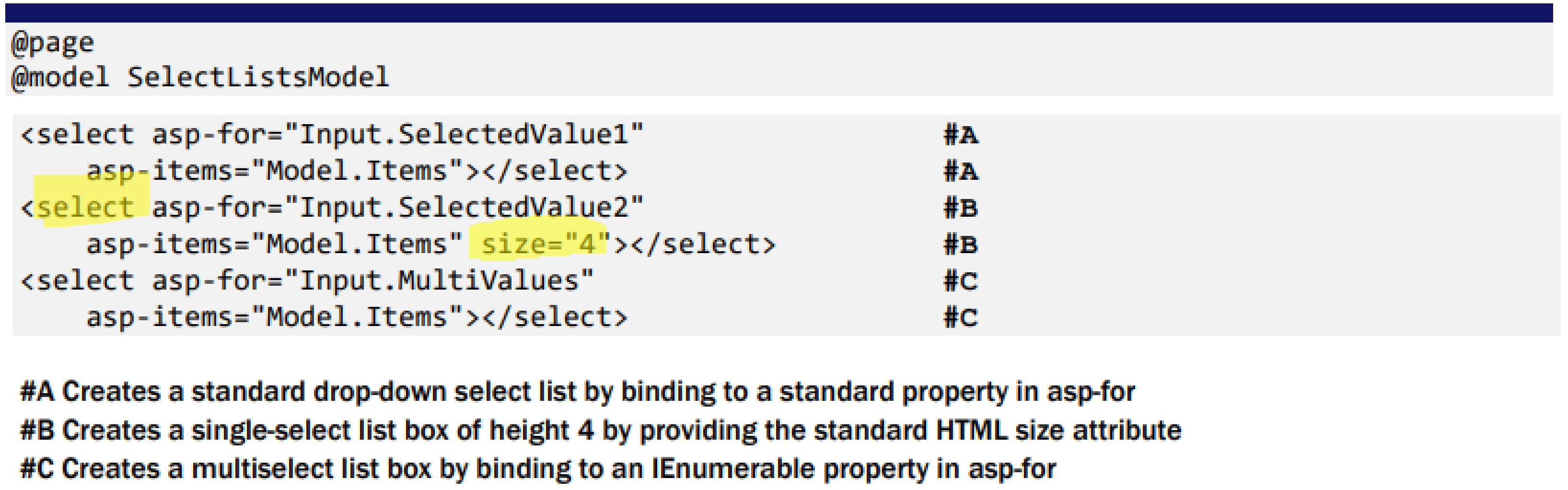
```
public class SelectListsModel: PageModel
{
    [BindProperty]                                                #A
    public class InputModel Input { get; set; }                   #A

    public IEnumerable<SelectListItem> Items { get; set; }        #B
        = new List<SelectListItem>                                #B
    {                                                             #B
        new SelectListItem{Value= "csharp", Text="C#"},           #B
        new SelectListItem{Value= "python", Text= "Python"},      #B
        new SelectListItem{Value= "cpp", Text="C++"},             #B
        new SelectListItem{Value= "java", Text="Java"},           #B
        new SelectListItem{Value= "js", Text="JavaScript"},       #B
        new SelectListItem{Value= "ruby", Text="Ruby"},           #B
    };                                                            #B

    public class InputModel
    {
        public string SelectedValue1 { get; set; }                #C
        public string SelectedValue2 { get; set; }                #C
        public IEnumerable<string> MultiValues { get; set; }      #D
    }
}
```

#A The InputModel for binding the user's selections to the select boxes
#B The list of items to display in the select boxes
#C These properties will hold the values selected by the single-selection select boxes.
#D To create a multiselect list box, use an IEnumerable<>.

The Select Tag Helper only works with SelectListItem elements.

To use <select> elements in your Razor code, you'll need to include two properties in your PageModel: one property for the list of options to display and one to hold the value (or values) selected.

THE SELECT TAG HELPER



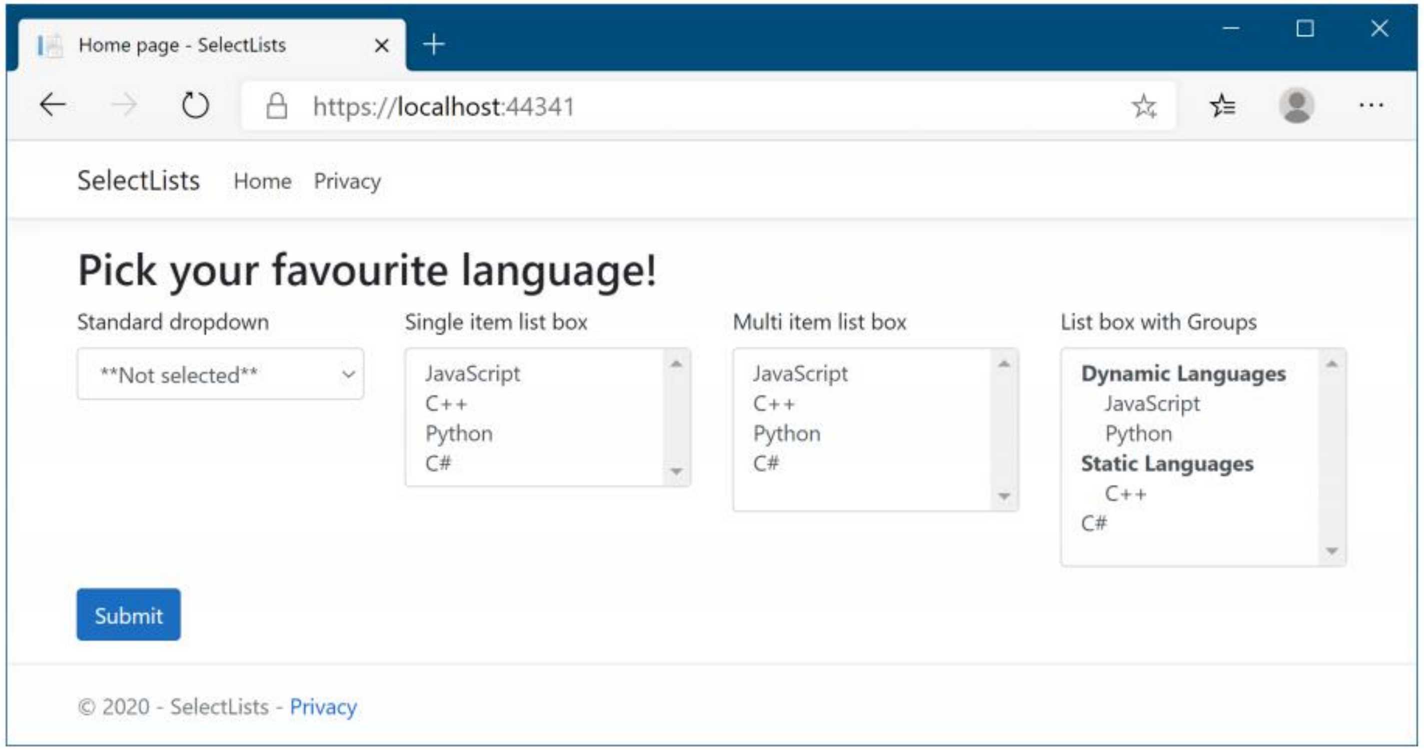
```
@page
@model SelectListsModel
```

```
<select asp-for="Input.SelectedValue1"                        #A
     asp-items="Model.Items"></select>                        #A
<select asp-for="Input.SelectedValue2"                        #B
     asp-items="Model.Items" size="4"></select>               #B
<select asp-for="Input.MultiValues"                           #C
     asp-items="Model.Items"></select>                        #C
```

#A Creates a standard drop-down select list by binding to a standard property in asp-for
#B Creates a single-select list box of height 4 by providing the standard HTML size attribute
#C Creates a multiselect list box by binding to an IEnumerable property in asp-for

- **asp-for** attribute specifies the **property** in your view model to bind to.
- **asp-items** attribute is provided the **IEnumerable<SelectListItem>** to display the available <option>.

DISPLAY GROUPS IN YOUR LIST BOXES



Pick your favourite language!

Standard dropdown	Single item list box	Multi item list box	List box with Groups
Not selected	JavaScript	JavaScript	**Dynamic Languages**
	C++	C++	JavaScript
	Python	Python	Python
	C#	C#	**Static Languages**
			C++
			C#

© 2020 - SelectLists - Privacy

```
public class SelectListsModel: PageModel
{
    [BindProperty]
    public IEnumerable<string> SelectedValues { get; set; }
    public IEnumerable<SelectListItem> Items { get; set; }

    public SelectListsModel()                                      #A
    {
        var dynamic = new SelectListGroup { Name = "Dynamic" };     #B
        var stat = new SelectListGroup { Name = "Static" };         #B
        Items = new List<SelectListItem>
        {
            new SelectListItem {
                Value= "js",
                Text="Javascript",
                Group = dynamic                                     #C
            },
            new SelectListItem {
                Value= "cpp",
                Text="C++",
                Group = stat                                        #C
            },
            new SelectListItem {
                Value= "python",
                Text="Python",
                Group = dynamic                                     #C
            },
            new SelectListItem {                                    #D
                Value= "csharp",                                    #D
                Text="C#",                                          #D
            }
        };
    }
}
```

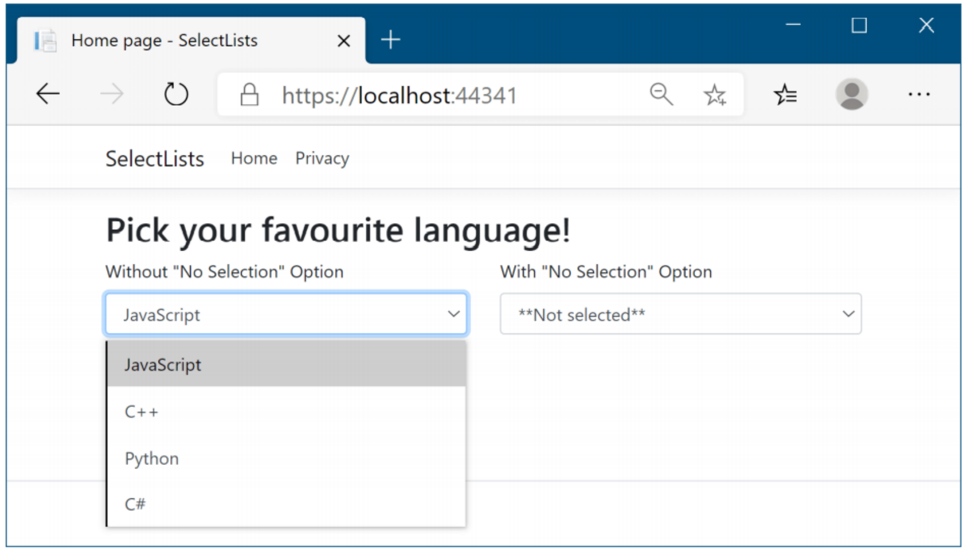
#A Initializes the list items in the constructor
#B Creates single instance of each group to pass to SelectListItems
#C Sets the appropriate group for each SelectListItem
#D If a SelectListItem doesn't have a Group, it won't be added to an <optgroup>.

```
@page
@model SelectListsModel
<select asp-for="SelectedValues" asp-items="Model.Items"></select>
```

would be rendered to HTML as:

```
<select id="SelectedValues" name="SelectedValues" multiple="multiple">
    <optgroup label="Dynamic">
        <option value="js">JavaScript</option>
        <option value="python">Python</option>
    </optgroup>
    <optgroup label="Static Languages">
        <option value="cpp">C++</option>
    </optgroup>
    <option value="csharp">C#</option>
</select>
```

SELECT WITH NO VALUES



Pick your favourite language!

Without "No Selection" Option

JavaScript

JavaScript
C++
Python
C#

With "No Selection" Option

Not selected

Figure 8.7 Without a “no selection” option, the `<select>` element will always have a value. This may not be the behavior you desire if you don’t want an `<option>` to be selected by default.

You can achieve this in one of two ways: you could either add the “not selected” option to the available `SelectListItem`s, or you could manually add the option to the Razor, for example by using

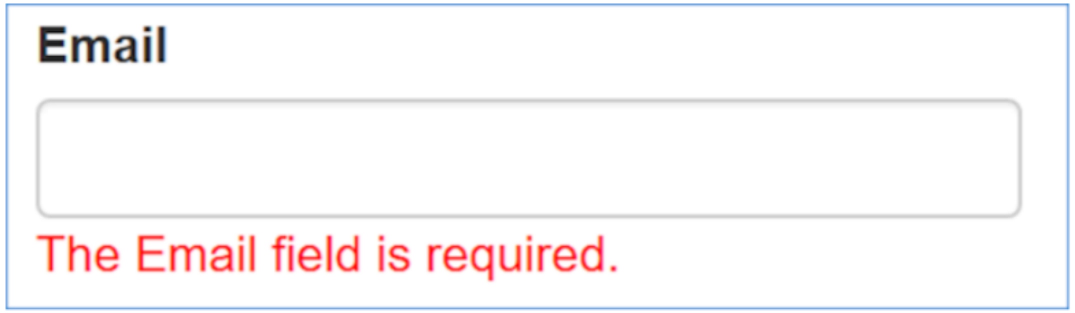
```
<select asp-for="SelectedValue" asp-items="Model.Items">
    <option Value = "">**Not selected**</option>
</select>
```

This will add an extra `<option>` at the top of your `<select>` element, with a blank `Value` attribute, allowing you to provide a “no selection” option for the user.

THE VALIDATION MESSAGE TAG HELPER

- This can be achieved for **each property** in your view model using the Validation Message Tag Helper applied to a <span> by using the **asp-validation-for** attribute:

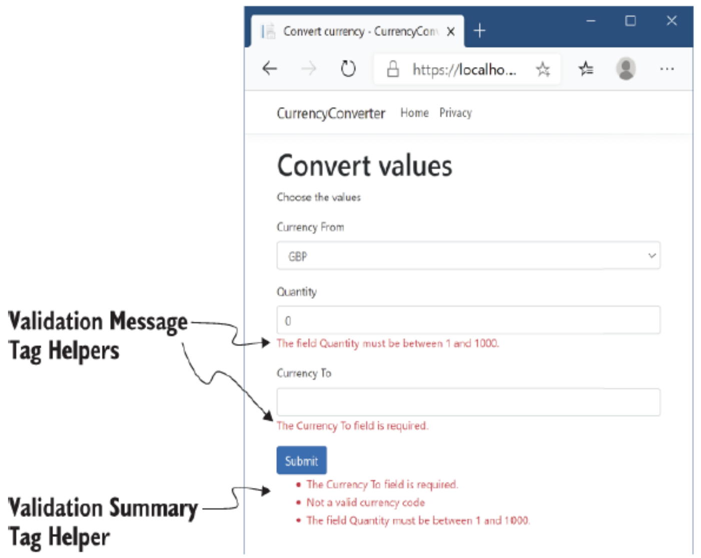
<span asp-validation-for="Email"></span>


Email

The Email field is required.

When an error occurs during **client-side validation** or **server-side validation** , the appropriate error message for the referenced property will be displayed in the <span>.

VALIDATION SUMMARY TAG HELPER


Convert currency - CurrencyCon
https://localho...
CurrencyConverter
Home
Privacy
Submit
Validation Message Tag Helpers
Validation Summary Tag Helper

Convert values

Choose the values

Currency From

GBP

Quantity

0

The field Quantity must be between 1 and 1000.

Currency To

The Currency To field is required.

- The Currency To field is required.
- Not a valid currency code
- The field Quantity must be between 1 and 1000.

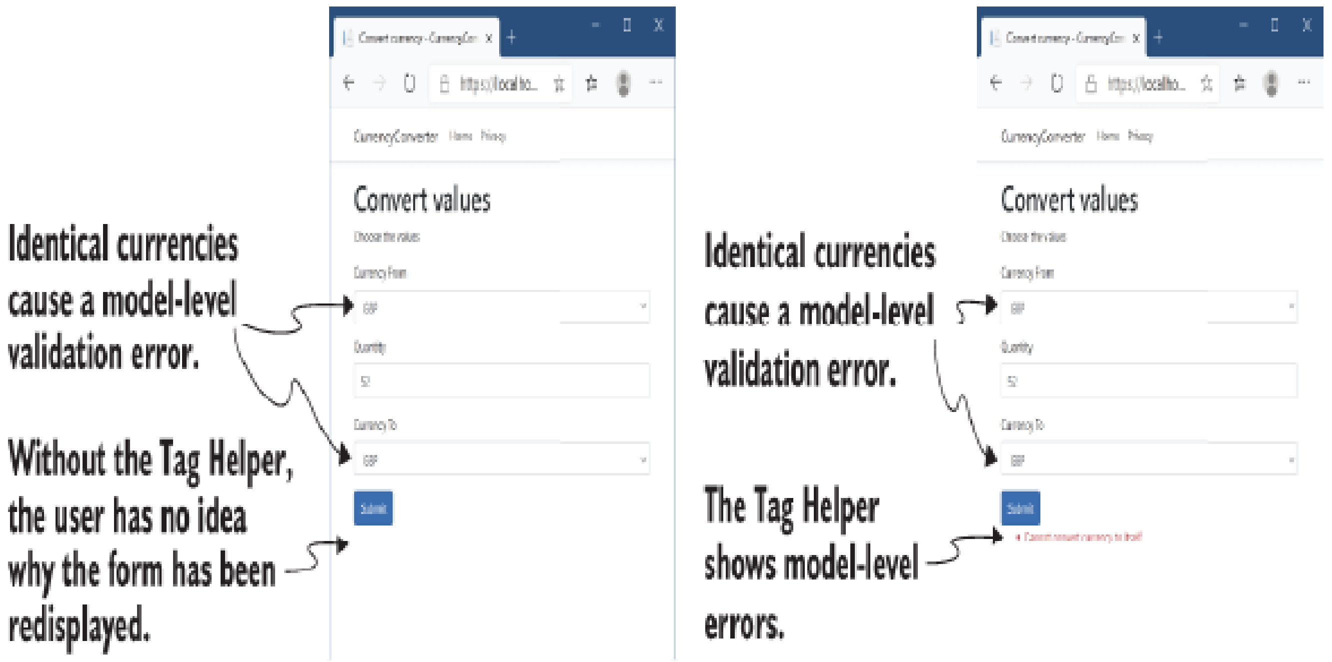
Applied to a <div> using the **asp-validation-summary** attribute and providing a ValidationSummary enum value, such as:

<div asp-validation-summary="All"></div>

The ValidationSummary enum controls which values are displayed, and has three possible values:

- **None**—Don't display a summary.
- **ModelOnly**—Only display errors that are not associated with a property.
- **All**—Display errors either associated with a property or with the model.

```
public class ConvertModel : PageModel
{
    [BindProperty]
    public InputModel Input { get; set; }

    [HttpPost]
    public IActionResult OnPost()
    {
        if(Input.CurrencyFrom == Input.CurrencyTo)          #A
        {
            ModelState.AddModelError(                       #B
                string.Empty,                               #B
                "Cannot convert currency to itself");       #B
        }
        if (!ModelState.IsValid)                            #C
        {                                                   #C
            return Page();                                  #C
        }                                                   #C

        //store the valid values somewhere etc
        return RedirectToPage("Checkout");
    }
}
```


Identical currencies cause a model-level validation error.
Without the Tag Helper, the user has no idea why the form has been redisplayed.
Convert values
Submit
Identical currencies cause a model-level validation error.
The Tag Helper shows model-level errors.
Convert values
Submit

#A Can't convert currency to itself
#B Adds model-level error, not tied to a specific property, by using empty key
#C If there are any property-level or model-level errors, display them.

GENERATING LINKS WITH THE ANCHOR TAG HELPER

- The Anchor Tag Helper can be used to generate the URL for a given page handler using **routing**.
  - you provide **asp-page** and **asp-page-handler** attributes, along with **asp-route-*** attributes as necessary.

```
<ul class="navbar-nav flex-grow-1">
```

```
    <li class="nav-item">
        <a class="nav-link text-dark"
            asp-area="" asp-page="/Index">Home</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark"
            asp-area="" asp-page="/Privacy">Privacy</a>
    </li>
</ul>
```

```
<ul class="nav navbar-nav">
    <li class="nav-item">
        <a class="nav-link text-dark" href="/">Home</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" href="/Privacy">Privacy</a>
    </li>t
</ul>
```

Handler names are case-sensitive in URLs or form attributes !

ANCHOR TAG HELPER PROPERTIES

- **asp-page**—Sets the Razor Page to execute.
- **asp-page-handler**—Sets the Razor Page handler to execute.
- **asp-area**—Sets the area route parameter to use. Areas can be used to provide an additional layer of organization to your application.
- **asp-host**—If set, the link will point to the provided host and will generate an absolute URL instead of a relative URL.
- **asp-protocol**—Sets whether to generate an http or https link. If set, it will generate an absolute URL instead of a relative URL.
- **asp-route**—Uses the named route to generate the URL.
- **asp-route-***—Sets the route parameters to use during generation. Can be added multiple times for different route parameters.

CACHE-BUSTING WITH THE APPEND VERSION TAG HELPER

- For performance reasons, browsers often cache files locally and reuse them for subsequent requests.
- A cache-busting query string adds a query parameter to a URL, such as ?v=1. Browsers will cache the response and use it for subsequent requests to the URL.
- When the **resource changes**, the query string is also changed, for example to ?v=2.
- Browsers will see this is a request for a new resource, and will make a fresh request.

When an application goes into production, is ensuring that browsers are all using the latest files.

APPEND TAG HELPER

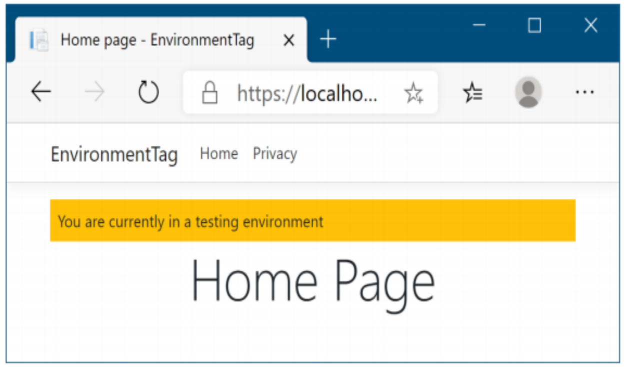
<script src="~/js/site.js" **asp-append-version="true"**></script>

The **asp-append-version** attribute will load the file being referenced and generate a unique hash based on its contents. This is then appended as a unique query string to the resource URL:

<script
src="/js/site.js?v=EWaMeWsJBYWmL2g_KkgXZQ5nPe"></script>

USING CONDITIONAL MARKUP WITH THE ENVIRONMENT TAG HELPER

- In many cases, you want to render different HTML in your Razor templates depending if your website is running in a **development** or **production** environment.



You are currently in a testing environment

Home Page

```
@if(env == "Testing" || env == "Staging")
{
 <div class="warning">You are currently on a testing
environment</div>
}
```

<environment include="Testing,Staging">
<div class="warning">You are currently on a testing environment</div>
</environment>

VALIDATEANTIFORGERYTOKEN

- security data annotation used in ASP.NET to prevent Cross-Site Request Forgery (CSRF) attacks.
- [**ValidateAntiForgeryToken**] ensures that the form request came from the legitimate site.
- It works together with the token generated by @Html.AntiForgeryToken() in the HTML form.
- <form> Tag Helper — It Adds the Anti-Forgery Token Automatically
  - When you use the <form> tag with asp-page, asp-action, or asp-controller and the method is post, ASP.NET Core automatically adds the anti-forgery token for you.