HTML Helpers in ASP.NET MVC

An HTML Helper is just a method that returns a HTML string. The string can represent any type of content that you want. For example, you can use HTML Helpers to render standard HTML tags like HTML <input>, <button> and <img> tags etc.

You can also create your own HTML Helpers to render more complex content such as a menu strip or an HTML table for displaying database data.

Different types of HTML Helpers

There are three types of HTML helpers as given below:

Inline Html Helpers

> These are create in the same view by using the Razor @helper tag. These helpers can be reused only on the same view.

```
1. @helper ListingItems(string[] items)
2.  {
3.  <ol>
4.  @foreach (string item in items)
5.  {
6.  <li>@item</li>
7.  }
8.  </ol>
9.  }
10.
11.   <h3>Programming Languages:</h3>
12.
13.   @ListingItems(new string[] { "C", "C++", "C#" })
14.
15.   <h3>Book List:</h3>
16.
17.   @ListingItems(new string[] { "How to C", "how to C++", "how to C#" })
```

Built-In Html Helpers

Built-In Html Helpers are extension methods on the HtmlHelper class. The Built-In Html helpers can be divided into three categories.

Standard Html Helpers

These helpers are used to render the most common types of HTML elements like as HTML text boxes, checkboxes etc. A list of most common standard html helpers is given below:

HTML Element

TextBox
@Html.TextBox("Textbox1", "val")
`Output`: <input id="Textbox1" name="Textbox1" type="text" value="val" />
TextArea
@Html.TextArea("Textarea1", "val", 5, 15, null)
`Output`: <textarea cols="15" id="Textarea1" name="Textarea1" rows="5">val</textarea>
Password
@Html.Password("Password1", "val")
`Output`: <input id="Password1" name="Password1" type="password" value="val" />
Hidden Field
@Html.Hidden("Hidden1", "val")
`Output`: <input id="Hidden1" name="Hidden1" type="hidden" value="val" />
CheckBox
@Html.CheckBox("Checkbox1", false)
`Output`: <input id="Checkbox1" name="Checkbox1" type="checkbox" value="true" /> <input name="myCheckbox" type="hidden" value="false" />
RadioButton
@Html.RadioButton("Radiobutton1", "val", true)
`Output`: <input checked="checked" id="Radiobutton1" name="Radiobutton1" type="radio" value="val" />
Drop-down list
@Html.DropDownList ("DropDownList1", new SelectList(new [] {"Male", "Female"}))
`Output`: <select id="DropDownList1" name="DropDownList1"> <option>M</option> <option>F</option> </select>
Multiple-select
Html.ListBox("ListBox1", new MultiSelectList(new [] {"Cricket", "Chess"}))
`Output`: <select id="ListBox1" multiple="multiple" name="ListBox1">
<option>Cricket</option> <option>Chess</option> </select>

Strongly Typed HTML Helpers

These helpers are used to render the most common types of HTML elements in strongly typed view like as HTML text boxes, checkboxes etc. The HTML elements are created based on model properties.

The strongly typed HTML helpers work on lambda expression. The model object is passed as a value to lambda expression, and you can select the field or property from model object to be used to set the id, name and value attributes of the HTML helper. A list of most common strongly-typed html helpers is given below:

HTML Element

TextBox
@Html.TextBoxFor(m=>m.Name)
Output: <input id="Name" name="Name" type="text" value="Name-val" />
TextArea
@Html.TextArea(m=>m.Address , 5, 15, new{}))
Output: <textarea cols="15" id="Address" name=" Address "
rows="5">Addressvalue</textarea>
Password
@Html.PasswordFor(m=>m.Password)
Output: <input id="Password" name="Password" type="password"/>
Hidden Field
@Html.HiddenFor(m=>m.UserId)
Output: <input id=" UserId" name=" UserId" type="hidden" value="UserId-val" />
CheckBox
@Html.CheckBoxFor(m=>m.IsApproved)
Output: <input id="Checkbox1" name="Checkbox1" type="checkbox" value="true" /> <input
name="myCheckbox" type="hidden" value="false" />
RadioButton
@Html.RadioButtonFor(m=>m.IsApproved, "val")
Output: <input checked="checked" id="Radiobutton1" name="Radiobutton1" type="radio"
value="val" />
Drop-down list
@Html.DropDownListFor(m => m.Gender, new SelectList(new [] {"Male", "Female"}))
Output: <select id="Gender" name="Gender"> <option>Male</option>
<option>Female</option> </select>
Multiple-select
Html.ListBoxFor(m => m.Hobbies, new MultiSelectList(new [] {"Cricket", "Chess"}))
Output: <select id="Hobbies" multiple="multiple" name="Hobbies">
<option>Cricket</option> <option>Chess</option> </select>

Templated HTML Helpers

These helpers figure out what HTML elements are required to render based on properties of your model class. This is a very flexible approach for displaying data to the user, although it requires some initial care and attention to set up. To setup proper HTML element with Templated HTML Helper, make use of DataType attribute of DataAnnitation class.

For example, when you use DataType as Password, A templated helper automatically render Password type HTML input element.

Templated Helper

Display
Renders a read-only view of the specified model property and selects an appropriate HTML element based on property's data type and metadata.
Html.Display("Name")
DisplayFor
Strongly typed version of the previous helper
Html.DisplayFor(m => m. Name)
Editor
Renders an editor for the specified model property and selects an appropriate HTML element based on property's data type and metadata.
Html.Editor("Name")
EditorFor
Strongly typed version of the previous helper
Html.EditorFor(m => m. Name)

Custom Html Helpers

You can also create your own custom helper methods by creating an extension method on the HtmlHelper class or by creating static methods with in a utility class.