HTML Helpers in ASP.NET MVC - Inline HTML Helper

HTML Helpers are methods that returns HTML strings. These are used in the view. In simple terms, these are C# methods that are used to return HTML. Using HTML helpers, you can render a text box, an area, image tag, etc. In MVC we have many built-in HTML helpers and we can create custom helpers too. Using HTML helpers, a view can show model properties and can generate HTML as per the types of properties.

Types of HTML Helpers:

1. Inline HTML Helper
2. Built-in HTML Helper
    - Standard HTML helper
    - Strongly Typed HTML helper
    - Templated HTMl Helper
3. Custom HTML Helper

Inline HTML Helpers
These are the type of helpers that are used on a single view and are used on the same page. inline HTML helpers can be created using **@helper tag.**

You can create your own HTML Helper with the following syntax.

```
@helper HelperName(parameters)

{

   // code

}
```

To use the above-created helper we use the following syntax

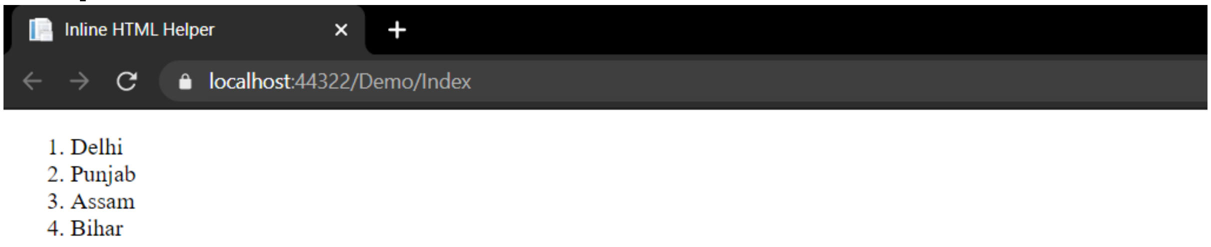
```
@HelperName(parameters)
```

Example:

```
@{
    Layout = null;
}


<!--created a inline HTMl Helper
    with a single string type parameter-->
@helper MyInlineHelper(string[] words)
{
    <ol>
        <!--Used a foreach loop inside HTML.
 similarly we can use any conditional statement
          or any logic like
            we use in normal C# code.-->
        @foreach (string word in words)
        {
            <li>@word</li>


        }
    </ol>
}
<!DOCTYPE html>

<html>
```

```
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Inline HTML Helper</title>
</head>
<body>
    <div>
        <!--called it inside this
            div and to get the output-->
        @MyInlineHelper(new string[] {
                  "Delhi", "Punjab", "Assam", "Bihar" })

    </div>
</body>
</html>
```

Output:



```
1. Delhi
2. Punjab
3. Assam
4. Bihar
```

Drawback of Inline HTML helpers

- These helpers can only be used with a single view only. You cannot use it with multiple views.

Different Types of HTML Helpers in ASP.NET MVC

ASP.NET provides a wide range of built-in HTML helpers that can be used as per the user's choice as there are multiple overrides available for them. There are three types of built-in HTML helpers offered by ASP.NET.

1. Standard HTML Helper

The HTML helpers that are mainly used to render HTML elements like text boxes, checkboxes, Radio Buttons, and Dropdown lists, etc. are called Standard HTML helpers.

List of Standard HTML Helpers

@Html.ActionLink() - Used to create link on html page

@Html.TextBox() - Used to create text box

@Html.CheckBox() - Used to create check box

@Html.RadioButton() - Used to create Radio Button

@Html.BeginFrom() - Used to start a form

@Html.EndFrom() - Used to end a form

@Html.DropDownList() - Used to create drop down list

@Html.Hidden() - Used to create hidden fields

@Html.label() - Used for creating HTML label is on the browser

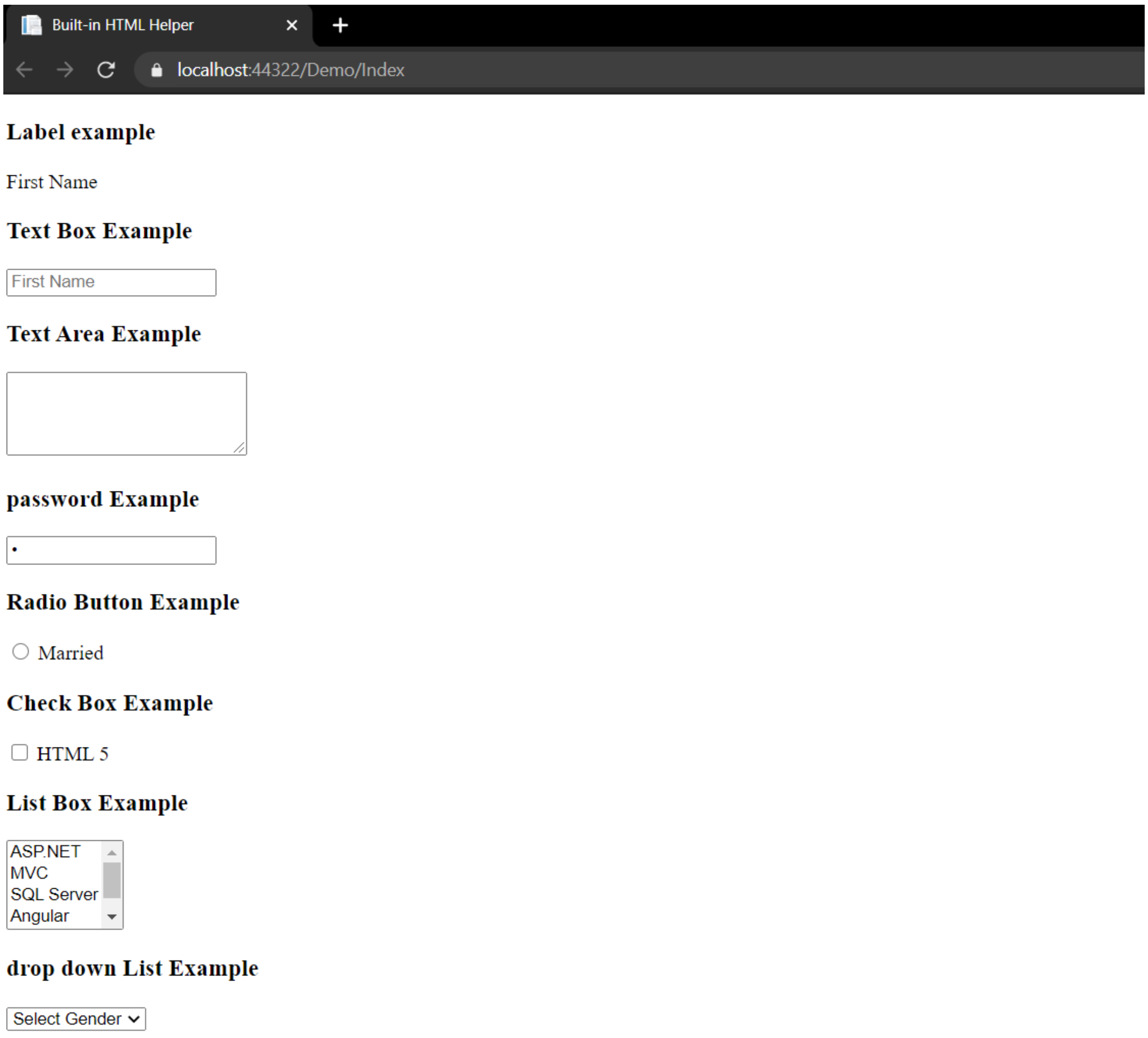
@Html.TextArea() - The TextArea Method renders textarea element on browser

@Html.Password() - This method is responsible for creating password input field on browser

@Html.ListBox() - The ListBox helper method creates html ListBox with scrollbar on browser

HTML

```
@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Built-in HTML Helper</title>
</head>
<body>
    <div>
        <h3>Label example</h3>
        @Html.Label("firstName", "First Name")

        <h3>Text Box Example</h3>
        @Html.TextBox("txtFirstName", "", new { @class =
"form-control", placeholder = "First Name" })

        <h3>Text Area Example</h3>
        @Html.TextArea("address", new { @class = "form-
control", rows = "5" })

        <h3>password Example</h3>
        @Html.Password("password", " ", new { @class =
"form-control" })

        <h3>Radio Button Example</h3>
        @Html.RadioButton("MaritalStatus", "Married", new { id
= "IsMarried" }) Married

        <h3>Check Box Example</h3>
        @Html.CheckBox("htmlSkill") HTML 5

        <h3>List Box Example</h3>
        @Html.ListBox("Skills", new List<SelectListItem> {
            new SelectListItem { Text="ASP.NET",Value="1"},
            new SelectListItem { Text="MVC",Value="2"},
            new SelectListItem { Text="SQL Server",Value="3"},
            new SelectListItem { Text="Angular",Value="4"},
            new SelectListItem { Text="Web API",Value="5"}
        }, new { @class = "form-control" })

        <h3>drop down List Example</h3>
        @Html.DropDownList("Gender", new
List<SelectListItem> {
                    new SelectListItem {Text="Select
Gender",Value="-1" },
                    new SelectListItem {Text="Male",Value="1" },
                    new SelectListItem {Text="Female", Value="2" }
                    }, new { @class = "custom-select" })

    </div>
</body>
</html>
```


Built-in HTML Helper

localhost:44322/Demo/Index

Label example

First Name

Text Box Example

First Name

Text Area Example

password Example

•

Radio Button Example

○ Married

Check Box Example

☐ HTML 5

List Box Example

ASP.NET
MVC
SQL Server
Angular

drop down List Example

Select Gender

2. Strongly-Typed HTML Helper

The Strongly-Typed HTML helper takes a lambda as a parameter that tells the helper, which element of the model to be utilized in the typed view. The Strongly typed views are used for rendering specific sorts of model objects, rather than using the overall View-Data structure.

List of strongly-Typed HTML Helper

@Html.HiddenFor()

@Html.LabelFor()

@Html.TextBoxFor()

@Html.RadioButtonFor()

@Html.DropDownListFor()

@Html.CheckBoxFor()

@Html.TextAreaFor()

@Html.PasswordFor()

@Html.ListBoxFor()

The functionality of all of these are the same as above but they are used with modal classes. Now, as we know we need a model class to use strongly typed HTML. So firstly we will add a model class as follows

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace HTML_Helper_Demo.Models
{
    public class Employee
    {
        public int EmpId { get; set; }
        public string Name { get; set; }
        public string Gender { get; set; }
        public city city { get; set; }
        public skills skills { get; set; }
        public string Address { get; set; }
        public string Password { get; set; }
        public bool AgreeTerm { get; set; }
    }
}
public enum city
{
    Dehli,
    Mumbai,
    Kolkata,
    Channai,
    Bangalore
}
public enum skills
{
    HTML5,
    CSS3,
    Bootstrap,
    JavaScript,
    JQuery,
    Angular,
    MVC,
    WebAPI
}
```

Now write the following code in the controller. And then add a view with the default properties.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using HTML_Helper_Demo.Models;

namespace HTML_Helper_Demo.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Index(Employee emp)
        {
            return View();
        }
    }
}
```

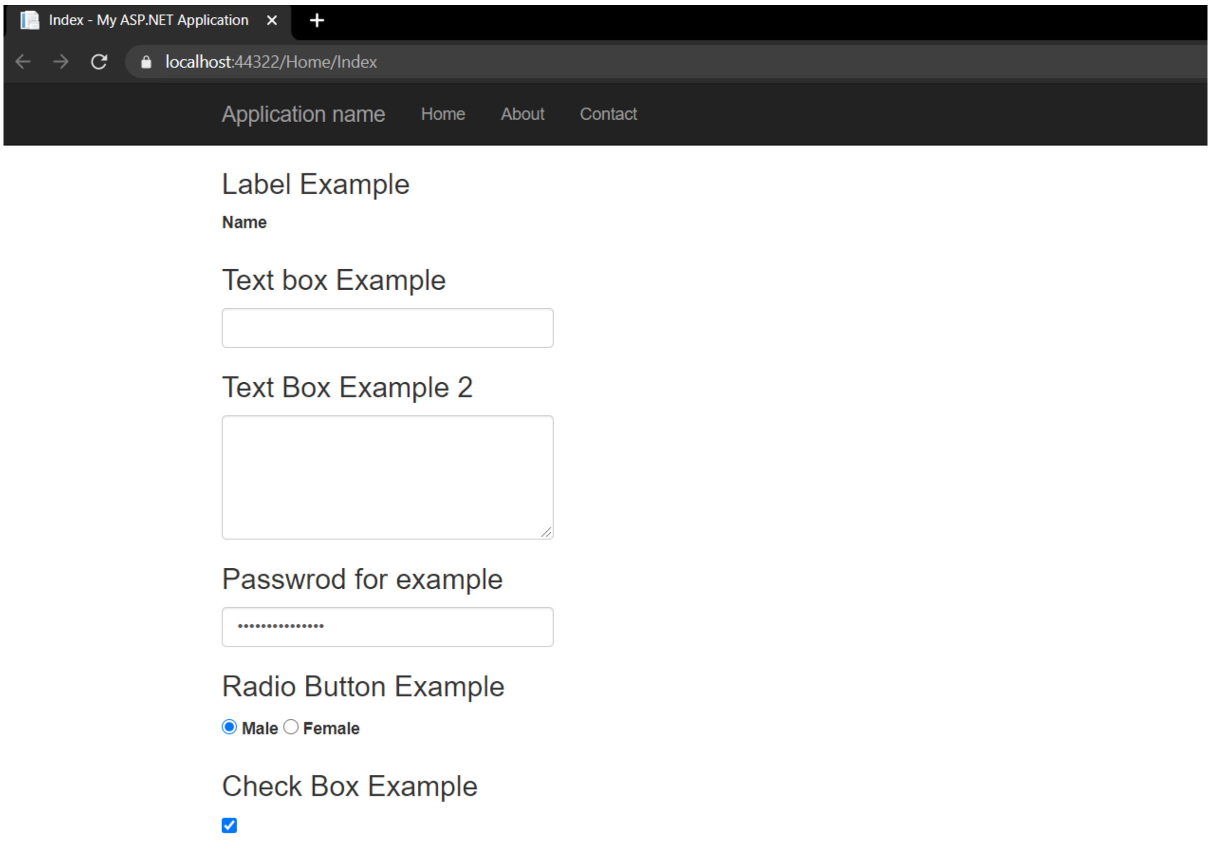
Now write the HTML as follows:

HTML

```
@using HTML_Helper_Demo.Models
@model Employee
@{
    ViewBag.Title = "Index";
}


<div>
    <h3>Label  Example</h3>
    @Html.LabelFor(model => model.Name, new { @class =
"label-control" })
```

```
    <h3>Text box Example</h3>

    @Html.TextBoxFor(model => model.Name, new { @class = "form-control" })


    <h3>Text Box Example 2</h3>

    @Html.TextAreaFor(model => model.Address, new { @class = "form-control", rows = "5" })


    <h3>Password for example</h3>

    @Html.PasswordFor(model => model.Password, new { @class = "form-control" })


    <h3>Radio Button Example</h3>

    @Html.RadioButtonFor(model => model.Gender, true, new { id = "male-true" })

    @Html.Label("male-true", "Male")

    @Html.RadioButtonFor(model => model.Gender, false, new { id = "female-true" })

    @Html.Label("female-true", "Female")


    <h3>Check Box Example</h3>

    @Html.CheckBoxFor(model => model.AgreeTerm)


    <h3>List Box Example</h3>
    @Html.ListBoxFor(model => model.skills, new
SelectList(Enum.GetValues(typeof(skills))),
                                            new { @class = "form-control" })



    <h3>Drop Down List Example</h3>
    @Html.DropDownListFor(model => model.city, new
SelectList(Enum.GetValues(typeof(city))),
                                             "Select City", new { @class =
"form-control" })


</div>
```

The output will be as follows:



Index - My ASP.NET Application

localhost:44322/Home/Index

Application name Home About Contact

Label Example

Name

Text box Example

Text Box Example 2

Passwrod for example

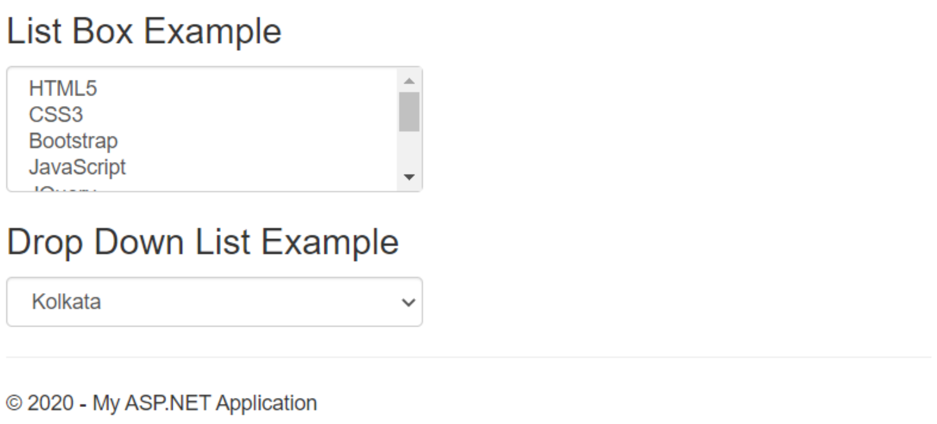
•••••••••••••••

Radio Button Example

[✓ Male] Female

Check Box Example

[✓]



List Box Example

HTML5
CSS3
Bootstrap
JavaScript

Drop Down List Example

Kolkata

© 2020 - My ASP.NET Application

3. Templated HTML Helper

The templated HTML Helper is used for data display and input. It generates HTML automatically as per model property and it can generate HTML for a complete model with a single tag. These are divided into two categories

- Display Template
- Editor Template

List of Templated HTML Helpers

Display

@Html.Display()

@Html.DisplayFor()

@Html.DisplayName()

@Html.DisplayNameFor()

@Html.DisplayText()

@Html.DisplayTextFor()

@Html.DisplayModelFor()

Edit / Input

@Html.Editor()

@Html.EditorFor()

@Html.EditorForModel()

Now, here we can use the previously created model class and then we should write this code in the controller

C#

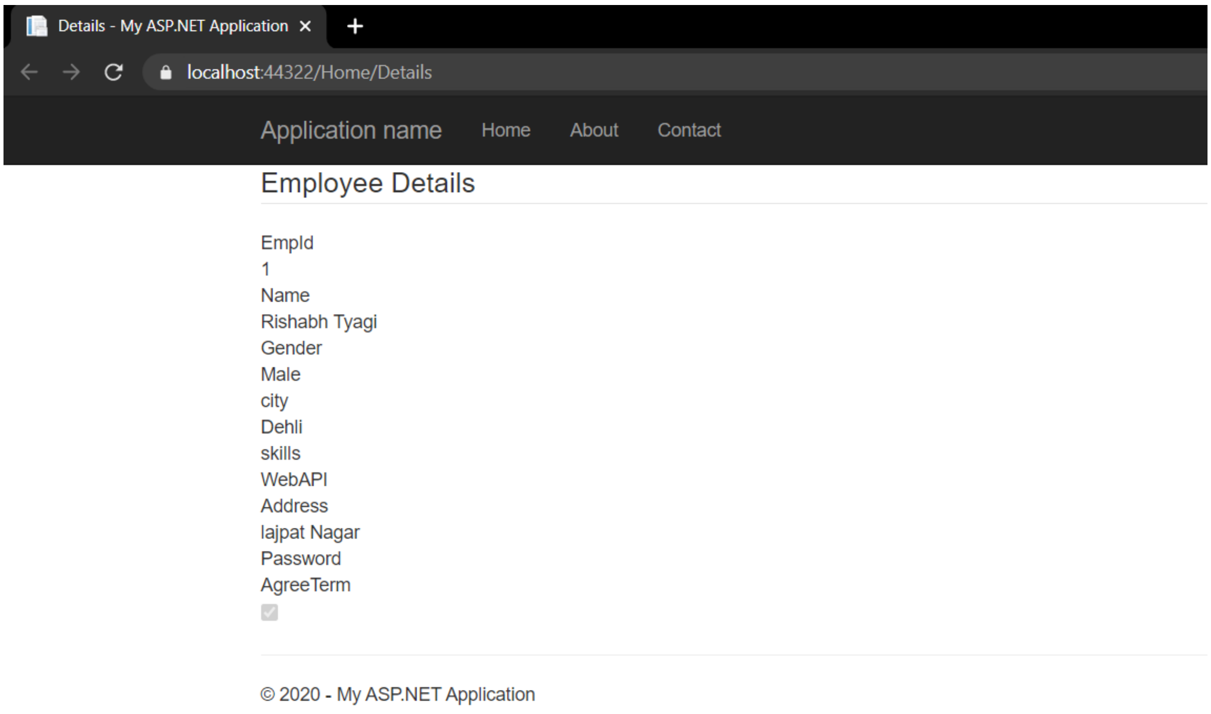
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using HTML_Helper_Demo.Models;

namespace HTML_Helper_Demo.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Details()
        {
            //Here we are hardcoded the Employee Details
            //In Realtime you will get the data from any data
source
            Employee employee = new Employee()
            {
                EmpId = 1,
                Name = "Rishabh Tyagi",
                Gender = "Male",
                city = city.Dehli,
                skills = skills.WebAPI,
                Address = "lajpat Nagar",
                AgreeTerm = true
            };
            ViewData["EmployeeData"] = employee;
            return View();
        }
    }
}
```

Now add the view with all the default properties and write the following code

HTML

```
@{
    ViewBag.Title = "Details";
}
<fieldset>
    <legend>Employee Details</legend>
    @Html.Display("EmployeeData")
</fieldset>
```

The output will be as follows


Details - My ASP.NET Application
localhost:44322/Home/Details
Application name
Home
About
Contact

Employee Details

```
EmpId
1
Name
Rishabh Tyagi
Gender
Male
city
Dehli
skills
WebAPI
Address
lajpat Nagar
Password
AgreeTerm
✓
```

© 2020 - My ASP.NET Application