

HTML Helper Methods i Asp.NET MVC

Emner

- Form Helper Methods (to versioner)
 - SelectList, SelectListItem
- Templated Helper Methods
 - Model Metadata

Html.BeginForm

Action Method

Controller

HTTP Method

```
@{Html.BeginForm("Search", "Home", FormMethod.Get);}  
    <input type="text" name="q" />  
    <input type="submit" value="Search" />  
@{Html.EndForm();}
```

Html.BeginForm (self closing form)

Action Method

Controller

HTTP Method

```
@using (Html.BeginForm("Search", "Home", FormMethod.Get))  
{  
    <input type="text" name="q" />  
    <input type="submit" value="Search" />  
}
```

The HTML **BeginForm** method returns an object that implements **IDisposable** interface with the **Dispose** method

[https://msdn.microsoft.com/en-us/library/system.web.mvc.html.formextensions.beginform\(v=vs.118\).aspx](https://msdn.microsoft.com/en-us/library/system.web.mvc.html.formextensions.beginform(v=vs.118).aspx)

GET eller POST?

- GET request repræsenterer en read-only operation .
- Du kan sende GET request til en server flere gange uden skadelige sideeffekter, fordi GET ikke ændrer (eller burde ikke ændre – det er jo altid muligt at lave dårlig kode) på state på serveren.
- En POST request vil generelt modificere state på en server
- Repeating POST requests kan give skadelige sideeffekter (f.eks. at man betaler 2 gange for den samme varer, hvis man trykker på buy-knappen). Typisk skal man tjekke state på server side for at undgå dette. Mange browser kommer også med en warning

Default metode for forms, hvis man ikke angiver noget (altså ikke angiver post eller get ved formen):

- ASP.NET MVC:
 - POST
- Standard HTML form:
 - GET

Udvalgte Overloads af BeginForm Helper Method

Overload	Description
<code>BeginForm()</code>	Creates a form which posts back to the action method it originated from
<code>BeginForm(action, controller)</code>	Creates a form which posts back to the action method and controller, specified as strings
<code>BeginForm(action, controller, method)</code>	As for the previous overload, but allows you to specify the value for the method attribute using a value from the <code>System.Web.Mvc.FormMethod</code> enumeration
<code>BeginForm(action, controller, method, attributes)</code>	As for the previous overload, but allows you to specify attributes for the form element an object whose properties are used as the attribute names
<code>BeginForm(action, controller, routeValues, method, attributes)</code>	As for the previous overload, but allows you to specify values for the variable route segments in your application routing configuration as an object whose properties correspond to the routing variables

Input HTML Helpers

Gør livet nemmere for os.

HTML Element	Example
Check box	<pre>Html.CheckBox("myCheckbox", false)</pre> <p>Output:</p> <pre><input id="myCheckbox" name="myCheckbox" type="checkbox" value="true" /> <input name="myCheckbox" type="hidden" value="false" /></pre>
Radio button	<pre>Html.RadioButton("myRadiobutton", "val", true)</pre> <p>Output:</p> <pre><input checked="checked" id="myRadiobutton" name="myRadiobutton" type="radio" value="val" /></pre>
Hidden field	<pre>Html.Hidden("myHidden", "val")</pre> <p>Output:</p> <pre><input id="myHidden" name="myHidden" type="hidden" value="val" /></pre>
Password	<pre>Html.Password("myPassword", "val")</pre> <p>Output:</p> <pre><input id="myPassword" name="myPassword" type="password" value="val" /></pre>

HTML Element	Example
Text area	<pre>Html.TextArea("myTextarea", "val", 5, 20, null)</pre> <p>Output:</p> <pre><textarea cols="20" id="myTextarea" name="myTextarea" rows="5">val</textarea></pre>
Text box	<pre>Html.TextBox("myTextbox", "val")</pre> <p>Output:</p> <pre><input id="myTextbox" name="myTextbox" type="text" value="val" /></pre>

Eksempler på brug – strongly typed view

```
@model Models.Person
@using(Html.BeginForm()) {
    <label>PersonId</label>
    @Html.TextBox("personId", @Model.PersonId) <br/>
    <label>First Name</label>
    @Html.TextBox("firstName", @Model.FirstName)
    <br/>
    <label>Last Name</label>
    @Html.TextBox("lastName", @Model.LastName) <br/>
    <input type="submit" value="Submit" />
}
```

Eksempler:

En overload som tager et enkelt string argument

```
@model Models.Person
@using(Html.BeginForm()) {
    <label>PersonId</label>
    @Html.TextBox("personId") <br/>
    <label>First Name</label>
    @Html.TextBox("firstName") <br/>
    <label>Last Name</label>
    @Html.TextBox("lastName") <br/>
    <input type="submit" value="Submit" />
}
```

Strongly Typed Input HTML Helpers

Med Lambda Expressions

Eksempler på Strongly Typed Input HTML Helpers

– med Lambda Expressions

```
@model Person // KEY LINE
@using(Html.BeginForm()) {
    <label>PersonId</label>
    @Html.TextBoxFor(m => m.PersonId) <br/>
    <label>First Name</label>
    @Html.TextBoxFor(m => m.FirstName) <br/>
    <label>Last Name</label>
    @Html.TextBoxFor(m => m.FirstName) <br/>
    <input type="submit" value="Submit" />
}
```

Strongly Typed Input HTML Helpers

HTML Element	Example
Check box	<pre>Html.CheckBoxFor(x => x.IsApproved)</pre> <p>Output:</p> <pre><input id="IsApproved" name="IsApproved" type="checkbox" value="true" /> <input name="IsApproved" type="hidden" value="false" /></pre>
Radio button	<pre>Html.RadioButtonFor(x => x.IsApproved, "val")</pre> <p>Output:</p> <pre><input id="IsApproved" name="IsApproved" type="radio" value="val" /></pre>
Hidden field	<pre>Html.HiddenFor(x => x.PersonId)</pre> <p>Output:</p> <pre><input id="x.PersonId" name="x.PersonId" type="hidden" value="" /></pre>

HTML Element	Example
Password	<pre>Html.PasswordFor(x => x.Password)</pre> <p>Output:</p> <pre><input id="Password" name="Password" type="password" /></pre>
Text area	<pre>Html.TextAreaFor(x => x.Bio, 5, 20)</pre> <p>Output:</p> <pre><textarea cols="20" id="Bio" name="Bio" rows="5"> Bio value</textarea></pre>
Text box	<pre>Html.TextBoxFor(x => x.FirstName)</pre> <p>Output:</p> <pre><input id="FirstName" name="FirstName" type="text" value="" /></pre>

Fordele ved at bruge strongly typed Helpers

Full IntelliSense support og kode tjek på **compile time**.

Man undgår stavefejl i parameter navnene, som kan være svære at debugge.

Tilføje CSS parametre til HTML elements?

- Overfør et anonymously typed object

```
Html.TextBoxFor(m => m.FirstName,  
    new {style = "background-color:lightyellow", size="2"})
```

```
Html.TextBox("LastName", "",  
    new {@class = "active", maxlength="40"})
```

```
Html.EditorFor(x => x.Age, new { htmlAttributes = new {  
    @class = "myclass" } })
```

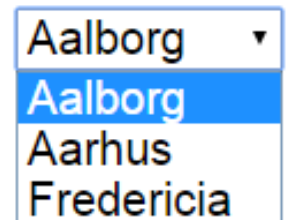
Lave drop downs:

```
List<SelectListItem> postalDistricts = new List<SelectListItem>();  
postalDistricts.Add(  
    new SelectListItem { Text = "Aalborg", Value = "9000" }  
);  
  
ViewBag.PostalDistrict = postalDistricts;  
return View();
```

Controller

```
<div>  
    @Html.DropDownList("PostalDistrict")  
</div>
```

View



Lave drop downs:

```
List<SelectListItem> postalDistricts = new List<SelectListItem>();  
postalDistricts.Add(  
    new SelectListItem { Text = "Aalborg", Value = "9000" }  
);  
  
ViewBag.PostalDistrict = postalDistricts;  
return View();
```

Controller

Eller måske er nedenstående pænere:

```
<div>  
    @Html.DropDownList("PostalDistrict" ViewBag.PostalDistrict)  
</div>
```

Lav Html helper

```
public static MvcHtmlString ListArray
    (this HtmlHelper html, string[] list)
{
    TagBuilder tagHeadline = new TagBuilder("h1");
    tagHeadline.InnerHtml += "My list from C# code";

    TagBuilder tag = new TagBuilder("ul");
    foreach(string str in list)
    {
        TagBuilder item = new TagBuilder("li");

        item.SetInnerText(str);
        tag.InnerHtml += item.ToString();
    }
    return new MvcHtmlString(tagHeadline.ToString()+tag.ToString());
}
```

Templated Helper Methods

The MVC Templated HTML Helpers

Helper	Example	Description
Display	<code>Html.Display("FirstName")</code>	Renders a read-only view of the specified model property, choosing an HTML element according to the property's type and metadata
DisplayFor	<code>Html.DisplayFor(x => x.FirstName)</code>	Strongly typed version of the previous helper
Editor	<code>Html.Editor("FirstName")</code>	Renders an editor for the specified model property, choosing an HTML element according to the property's type and metadata
EditorFor	<code>Html.EditorFor(x => x.FirstName)</code>	Strongly typed version of the previous helper
Label	<code>Html.Label("FirstName")</code>	Renders an HTML <code><label></code> element referring to the specified model property
LabelFor	<code>Html.LabelFor(x => x.FirstName)</code>	Strongly typed version of the previous helper

Modellen – et eksempel



```
public class Person
{
    public int PersonId { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    [DataType(DataType.Date)]
    public DateTime BirthDate { get; set; }
}
```



The View – razor

```
<h2>Create Person</h2>
<div class="dataElem">
    @Html.Label("PersonId")
    @Html.Editor("PersonId")
</div>
<div class="dataElem">
    @Html.Label("FirstName")
    @Html.Editor("FirstName")
</div>
<div class="dataElem">
    @Html.LabelFor(m => m.LastName)
    @Html.EditorFor(m => m.LastName)
</div>
<div class="dataElem">
    @Html.LabelFor(m => m.BirthDate)
    @Html.EditorFor(m => m.BirthDate)
</div>
```

Output

Create Person

PersonId	<input type="text" value="9"/>
FirstName	<input type="text"/>
LastName	<input type="text"/>
BirthDate	<input type="text" value="09-09-2014"/>   

september 2014 ▼   

ma	ti	on	to	fr	lø	sø
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

HTML Source Coden som genereres.

```
<h2>Create Person</h2>
<div class="dataElem">
  <label for="PersonId">PersonId</label>
  <input class="text-box single-line" id="PersonId" name="PersonId" type="number" value="" />
</div>
<div class="dataElem">
  <label for="FirstName">FirstName</label>
  <input class="text-box single-line" id="FirstName" name="FirstName" type="text" value="" />
</div>
<div class="dataElem">
  <label for="LastName">LastName</label>
  <input class="text-box single-line" id="LastName" name="LastName" type="text" value="" />
</div>
<div class="dataElem">
  <label for="BirthDate">BirthDate</label>
  <input class="text-box single-line" id="BirthDate" name="BirthDate" type="date" value="" />
</div>
```

Model Metadata

Brug **Metadata** til at kontrollere display, visibility of type

```
public class Person {  
    → [HiddenInput(DisplayValue = false)]  
    public int PersonId { get; set; }  
    → [Display(Name="First")]  
    public string FirstName { get; set; }  
}  
    → [Display(Name="Last")]  
    public string LastName { get; set; }  
    → [Display(Name = "Birth Date")]  
    → [DataType(DataType.Date)]  
    public string BirthDay { get; set; }  
}
```

Create Person

First	<input type="text"/>
Last	<input type="text"/>
Birth Date	<input type="text" value="dd-mm-åååå"/>
<input type="button" value="Create"/>	

Built-In MVC Framework View Templates

Boolean	Date	MultilineText	String	Url
Collection	EmailAddress	Number	Text	
Decimal	HiddenInput	Object	Tel	
DateTime	Html	Password	Time	

```
public class Person
{
    ...
    [UIHint("MultilineText")]
    public string Comment { get; set; }
}
```

Comment

Create

Whole-Model Templated Helpers (scaffolding)

Bemærk: @model skal angives i viewet (ikke vist her)

```
public class Person {  
    [HiddenInput(DisplayValue = false)]  
    public int PersonId { get; set; }  
    [Display(Name="First")]  
    public string FirstName { get; set; }  
    [Display(Name="Last")]  
    public string LastName { get; set; }  
    [Display(Name = "Birth Date")]  
    [DataType(DataType.Date)]  
    public DateTime BirthDate { get; set; }  
    [DataType(DataType.PhoneNumber)]  
    public string Phone { get; set; }  
    [DataType(DataType.EmailAddress)]  
    public string Email { get; set; }  
}
```

```
<h2>Create Person</h2>  
  
@using (Html.BeginForm()) {  
    @Html.EditorForModel()  
    <br />  
    <input type="button"  
        value="Create">  
}
```

Scaffolding View (@Html.EditorForModel)

Create Person

First

Last

Birth Date

dd - mm - åååå

Phone

Email

Create

Helper	Example	Description
DisplayForModel	Html.DisplayForModel()	Renders a read-only view
EditorForModel	Html.EditorForModel()	Renders editor elements
LabelForModel	Html.LabelForModel()	Renders an HTML <label> element object

Begrænsinger ved EditorForModel

Ved ikke, hvordan man får den til at genere dropdowns

Opgave 3