

Comment créer des HTML Helpers

par Matt-k (Traduction)


Date de publication : mai 2009

Dernière mise à jour :

Le but de ce tutoriel est de montrer comment concevoir des HTML Helpers personnalisés et utilisables dans vos vues MVC. En exploitant les HTML Helpers, vous pouvez réduire le nombre de tags HTML à écrire pour créer une page au standard HTML.

Traduction.....	3
Introduction.....	3
Comprendre les HTML Helpers.....	3
Développer des HTML Helpers avec des Méthodes Statiques.....	5
Développer des HTML Helpers par des Méthodes d'Extension.....	6
Conclusion.....	8

Traduction

Cet article est la traduction la plus fidèle possible de l'article original :  **Creating Custom HTML Helpers**

Introduction

Dans la première partie de ce tutoriel, je décrirai quelques HTML Helpers inclus dans le Framework ASP.NET MVC. Puis, je présenterai deux méthodes pour créer ses HTML Helpers personnalisés : en utilisant une méthode statique et en créant une méthode d'extension.

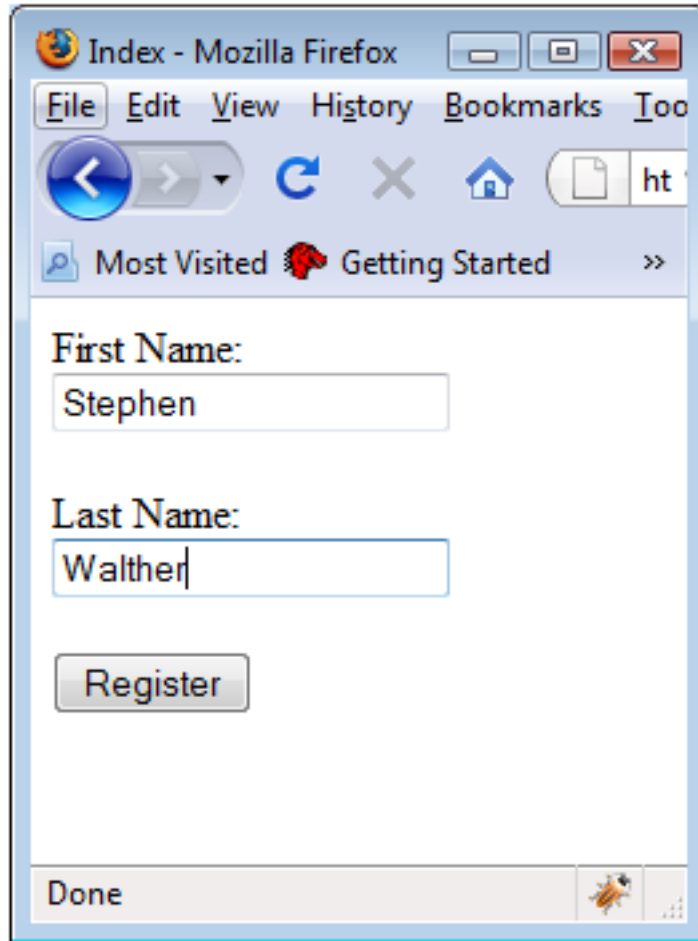
Comprendre les HTML Helpers

Un HTML Helper est juste une méthode qui retourne un string. La chaîne de caractère peut représenter tout type contenant ce que vous voulez. Par exemple, vous pouvez utiliser les HTML Helpers pour générer des tags HTML tels que `<input>` et ``. Vous pouvez aussi utiliser les HTML Helpers pour générer du contenu plus complexe comme des onglets ou un tableau HTML contenant vos données issues d'une base de données.

Le Framework ASP.NET MVC contient une série d'HTML Helpers (cette liste n'est pas exhaustive):

- `Html.ActionLink()`
- `Html.BeginForm()`
- `Html.CheckBox()`
- `Html.DropDownList()`
- `Html.EndForm()`
- `Html.Hidden()`
- `Html.ListBox()`
- `Html.Password()`
- `Html.RadioButton()`
- `Html.TextArea()`
- `Html.TextBox()`

Par exemple, considérons le formulaire ci-dessous.



Ce formulaire a été créé grâce à deux HTML Helpers standards. Il utilise les méthodes des Helpers **Html.BeginForm()** et **Html.TextBox()** pour générer un simple formulaire HTML.

Views\Home\Index.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Index.aspx.cs"
    Inherits="MvcApplication1.Views.Home.Index"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Index</title>
</head>
<body>
    <div>
        <% using (Html.BeginForm())
        { %>
            <label for="firstName">First Name:</label>
            <br />
            <%= Html.TextBox("firstName") %>
            <br /><br />
            <label for="lastName">Last Name:</label>
            <br />
            <%= Html.TextBox("lastName") %>
            <br /><br />
            <input type="submit" value="Register" />
        <% } %>
    </div>
</body>
</html>
```

La méthode de l'Helper **Html.BeginForm()** est utilisée pour ouvrir et fermer le tag HTML **<form>**. Noté que la méthode **Html.BeginForm()** est appelée au sein de la déclaration *using*. La déclaration *using* assure que le tag **<form>** sera fermé à la fin du bloc.

Si vous préférez, à la place de déclarer un bloc *using*, vous pouvez appeler la méthode de l'Helper **Html.EndForm()** pour fermer le tag **<form>**. Utilisez l'approche qui vous semble la plus intuitive pour ouvrir et fermer le tag **<form>**.

Les Helpers **Html.TextBox()** sont utilisés pour générer les tags HTML **<input>**. Si vous regardez le code html grâce à votre navigateur web, vous pourrez voir le code ci-dessous. Notez que le code source ne contient que des tags HTML standards.

⚠ Important : notez que l'HTML Helper **Html.TextBox()** est généré avec les tags **<%= %>** et non **<% %>**. La présence du signe = est primordiale, son oubli ne provoquera aucune génération code HTML.

Le Framework ASP.NET MVC contient un petit ensemble de Helpers. Le plus souvent, vous serez amenés à étendre le framework MVC avec des HTML Helpers personnalisés. Dans la suite de ce tutoriel, vous apprendrez deux méthodes pour créer vos HTML Helpers.

Code source de Index.aspx

```
<%@ Page Language="C#" AutoEventWireup="false" CodeBehind="Index.aspx.cs" Inherits="MvcApplication1.Index" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Index</title>
</head>
<body>
<div>
<form action="http://localhost:33102/" method="post">
<label for="firstName">First Name:</label>
<br />
<input id="firstName" name="firstName" type="text" value="" />
<br /><br />
<label for="lastName">Last Name:</label>
<br />
<input id="lastName" name="lastName" type="text" value="" />
<br /><br />
<input id="btnRegister" name="btnRegister" type="submit" value="Register" />
</form>
</div>
</body>
</html>
```

Développer des HTML Helpers avec des Méthodes Statiques

La méthode la plus simple pour développer un HTML Helper est de créer une méthode statique qui retourne un string. Imaginons, par exemple, que vous décidez de créer un nouvel HTML Helper qui générera le tag HTML **<label>**. Vous pouvez utiliser la classe présentée dans le code ci-dessous pour générer un **<label>**.

Helpers\LabelHelper.cs

```
using System;
namespace MvcApplication1.Helpers
{
public class LabelHelper
{
public static string Label(string target, string text)
{
return String.Format("<label for='{0}'>{1}</label>", target, text);
}
}
```

Helpers\LabelHelper.cs

```
}
}
```

Il n'y a donc rien de particulier dans cette classe. La méthode **Label()** retourne simplement une chaîne de caractères.

Dans le code suivant, nous utilisons notre **LabelHelper** pour générer le tag HTML **<label>** dans notre vue Index. Notez que la vue inclut la directive `<%@ imports %>` pour importer notre namespace **Application1.Helpers**.

Code de la vue Index

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Index2.aspx.cs"
    Inherits="MvcApplication1.Views.Home.Index2"%>
<%@ Import Namespace="MvcApplication1.Helpers" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Index2</title>
</head>
<body>
    <div>
        <% using (Html.BeginForm())
        { %>
            <%= LabelHelper.Label("firstName", "First Name:") %>
            <br />
            <%= Html.TextBox("firstName") %>
            <br /><br />
            <%= LabelHelper.Label("lastName", "Last Name:") %>
            <br />
            <%= Html.TextBox("lastName") %>
            <br /><br />
            <input type="submit" value="Register" />
        <% } %>
    </div>
</body>
</html>
```

Développer des HTML Helpers par des Méthodes d'Extension

Si vous voulez développer des HTML Helpers qui fonctionnent comme les HTML Helpers standards inclus dans le Framework ASP.NET MVC vous devez créer une méthode d'extension. Les méthodes d'extensions vous permettent d'ajouter de nouvelles méthodes à une classe existante. Lors de la création de la méthode de l'HTML Helper, vous ajoutez de nouvelles méthodes à la classe **HtmlHelper** représentée par la propriété **Html** de la vue.

La code ci-dessous ajoute une méthode d'extension **Label()** à la classe **HtmlHelper**. Vous devez remarquer quelques petits détails sur cette classe. Premièrement, notez que cette classe est static. Vous devez définir une méthode d'extension dans une classe static.

Deuxièmement, le première paramètre de la méthode **Label()** est précédée du mot clé **this**. Le premier paramètre d'une méthode d'extension indique la classe que la méthode d'extension étends.

Helpers\LabelExtensions.cs

```
using System;
using System.Web.Mvc;

namespace MvcApplication1.Helpers
{
    public static class LabelExtensions
    {
        public static string Label(this HtmlHelper helper, string target, string text)
        {

```

Helpers\LabelExtensions.cs

```
return String.Format("<label for='{0}'>{1}</label>", target, text);
}
}
```

Après avoir créé votre méthode d'extension, et compilé votre application, la méthode apparaît dans l'Intellisense de Visual Studio comme toutes autres méthodes d'une classe (voir figure ci-dessous). La seule différence est que ce type de méthodes apparaît avec une icône "spéciale" (une flèche qui pointe vers le bas).

```
<% using (Html.Form())
{ %>

<%= Html.Label | %>
<br />
<%= Ht
<br />
<%= Ht
<br />
<%= Ht
<br />
<%= Ht
<br />
<%= } %>

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

Le code ci-dessous utilise notre méthode d'extension **Html.Label()** pour générer les tags **<label>** dans notre vue Index.

Views\Home\Index3.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Index3.aspx.cs"
Inherits="MvcApplication1.Views.Home.Index3" %>
<%@ Import Namespace="MvcApplication1.Helpers" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional"
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
<title>Index3</title>
</head>
<body>
<div>
<% using (Html.BeginForm())
{ %>
<%= Html.Label("firstName", "First Name:") %>
<br />
```

Views\Home\Index3.aspx

```
<%= Html.TextBox("firstName") %>
<br /><br />
<%= Html.Label("lastName", "Last Name:") %>
<br />
<%= Html.TextBox("lastName") %>
<br /><br />
<input type="submit" value="Register" />
<% } %>
</div>
</body>
</html>
```

Conclusion

Dans ce tutoriel, vous avez appris deux techniques pour créer vos propres HTML Helpers. Premièrement, vous avez vu comment créer votre HTML Helper **Label()** en utilisant une méthode statique retournant une chaîne de caractères. Puis, vous avez vu comment développer la méthode de l'HTML Helper **Label()** en exploitant les méthodes d'extensions de la classe **HtmlHelper**.

Durant ce tutoriel, je me suis concentré sur la création d'une méthode d'un HTML Helper extrêmement basique. Vous devez néanmoins réaliser que vous pouvez créer des HTML Helpers aussi compliqués que vous le désirez. Ainsi vous pouvez développer des HTML Helpers qui génèrent des contenus riches tels que des TreeView, Menus, ou des tableaux de données.