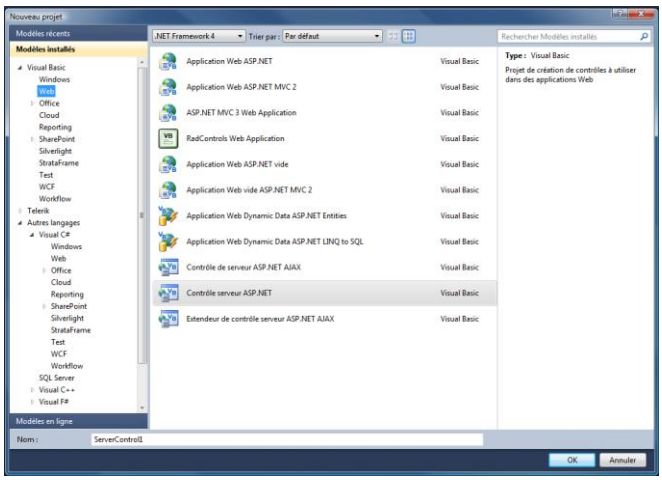
Créer une bibliothèque de classes pour le Web

Dans ce document, nous allons suivre pas à pas toutes les étapes permettant de créer une bibliothèque de contrôles pour le Web, l'ajouter dans la boite à outil, utiliser ces contrôles sur des pages Aspx.

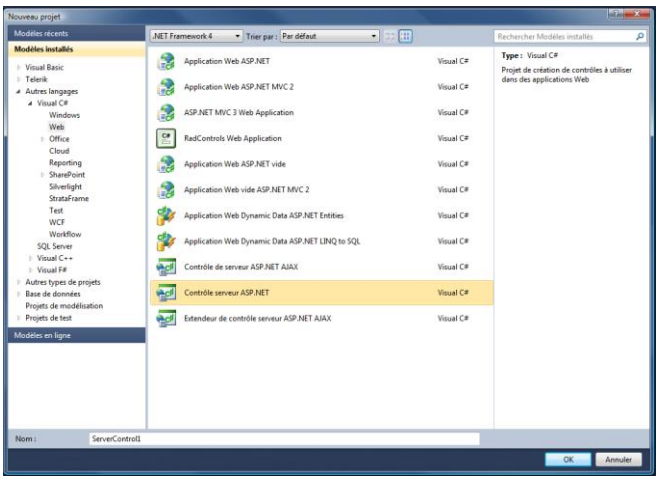
1. Création du projet

a. Dans Visual Studio, il faut créer un Nouveau Projet « Contrôle Serveur ASP.NET », dans les projets Web

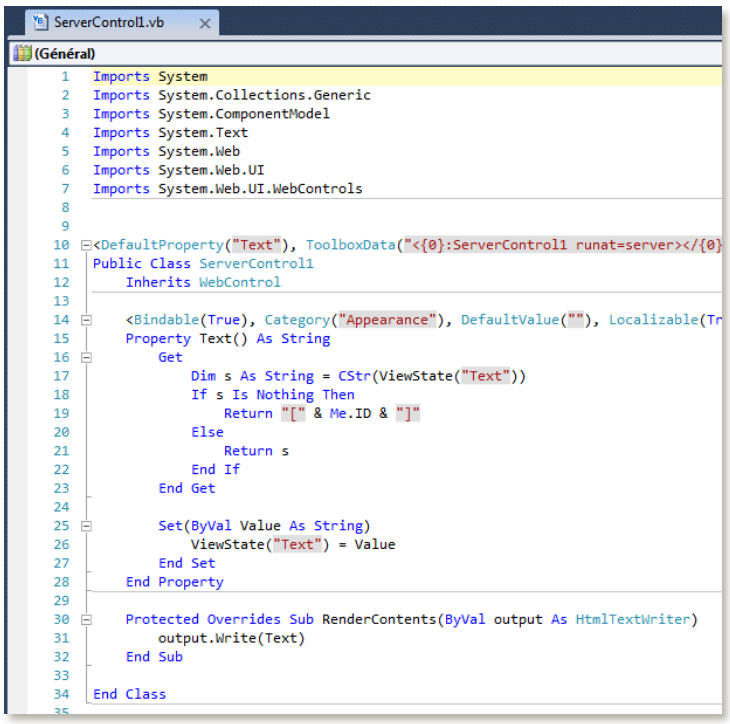
VB

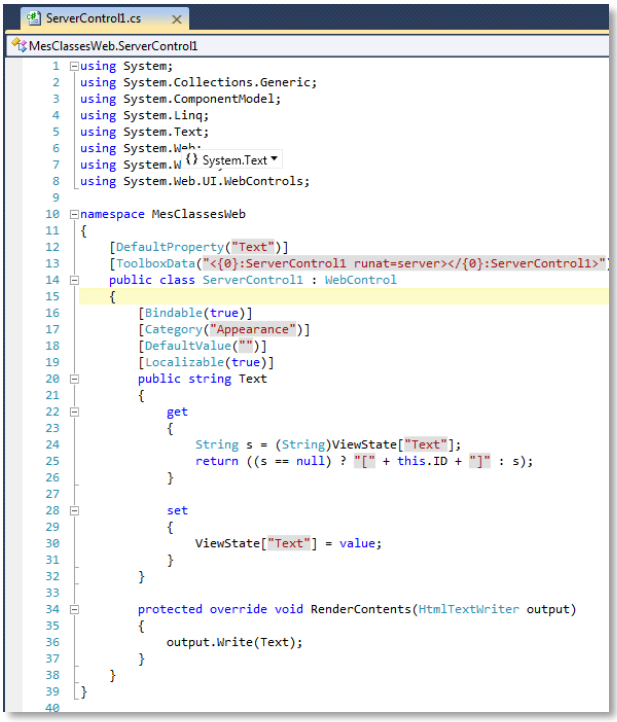


C#

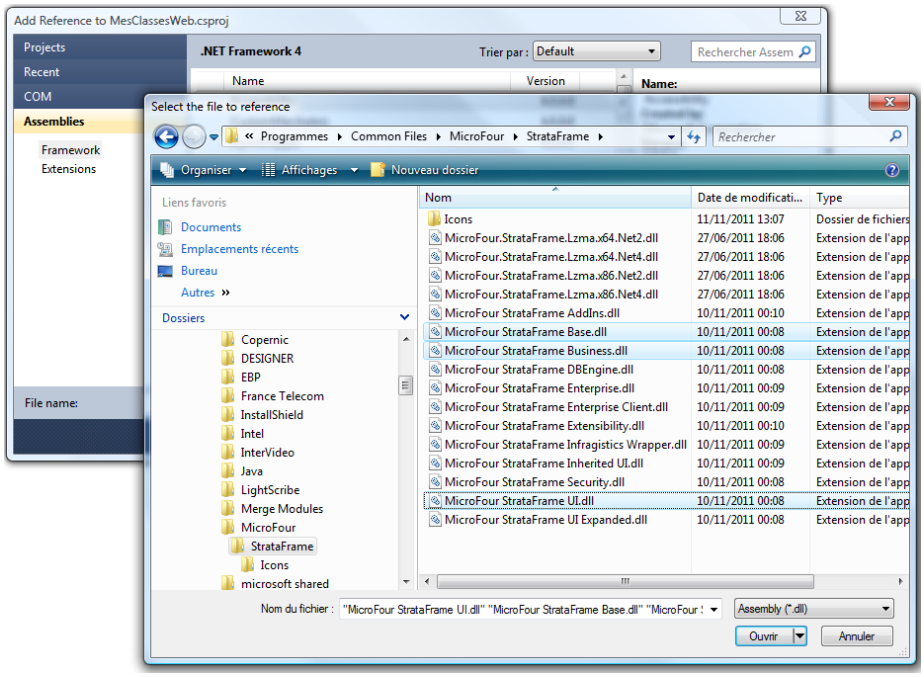


b. Votre projet est créé avec une première définition de classe que vous pouvez supprimer immédiatement



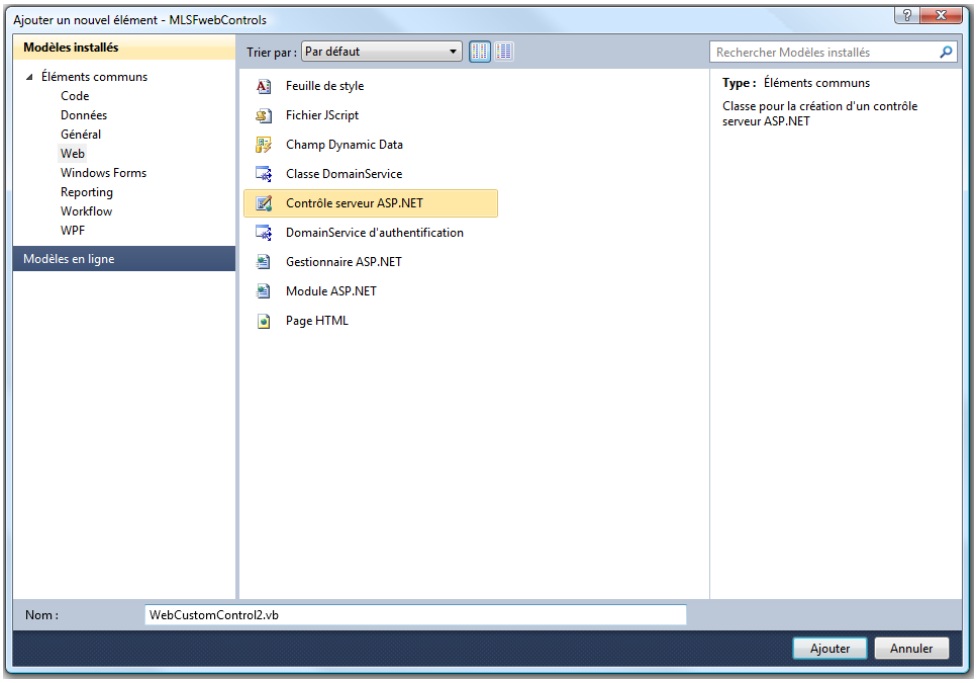


c. Si nécessaire ajoutez les références aux assembly de StrataFrame, que vous trouverez dans C:\Program Files\Common Files\MicroFour\StrataFrame\
(sélectionnez au moins Base, Business, et UI)

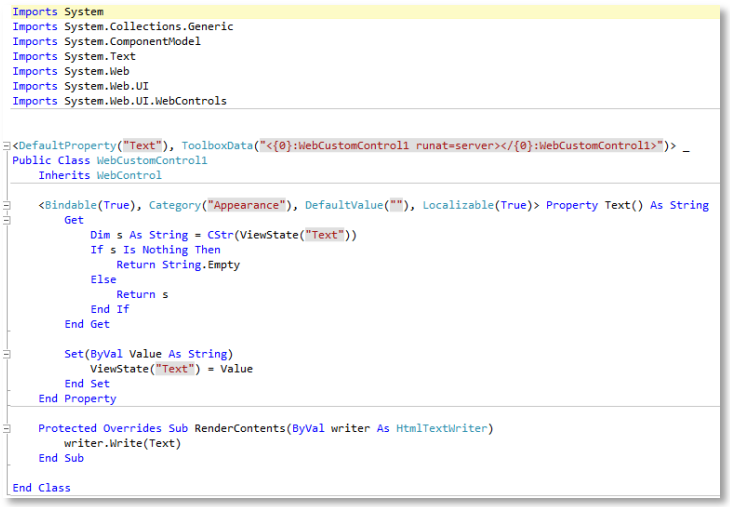


2. Création des classes

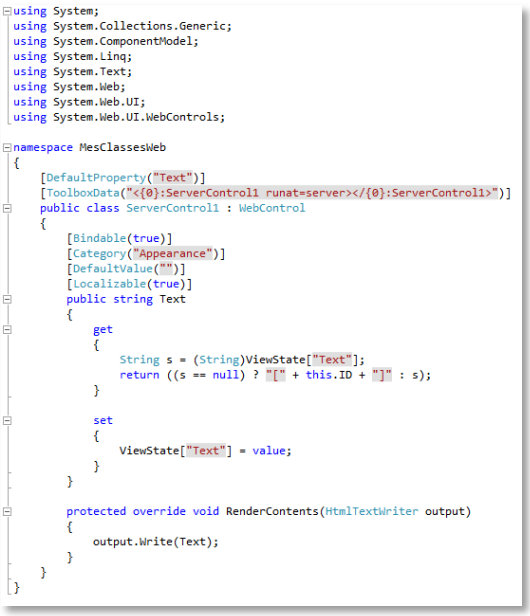
a. Dans le menu « Projet », choisissons « Ajouter un nouvel élément », et dans la boite de dialogue, pointons sur la catégorie Web et sélectionnons un élément de type « Contrôle Serveur ASP.NET »



Nommez votre nouveau contrôle, vous obtenez le nouvel élément avec du code par défaut .



```
Imports System
Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.Text
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls


<DefaultProperty("Text"), ToolboxData("<{0}:WebCustomControl1 runat=server></{0}:WebCustomControl1>")> _
Public Class WebCustomControl1
    Inherits WebControl

    <Bindable(True), Category("Appearance"), DefaultValue(""), Localizable(True)> Property Text() As String
        Get
            Dim s As String = CStr(ViewState("Text"))
            If s Is Nothing Then
                Return String.Empty
            Else
                Return s
            End If
        End Get

        Set(ByVal Value As String)
            ViewState("Text") = Value
        End Set
    End Property

    Protected Overrides Sub RenderContents(ByVal writer As HtmlTextWriter)
        writer.Write(Text)
    End Sub

End Class
```



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MesClassesWeb
{
    [DefaultProperty("Text")]
    [ToolboxData("<{0}:ServerControl1 runat=server></{0}:ServerControl1>")]
    public class ServerControl1 : WebControl
    {
        [Bindable(true)]
        [Category("Appearance")]
        [DefaultValue("")]
        [Localizable(true)]
        public string Text
        {
            get
            {
                String s = (String)ViewState["Text"];
                return ((s == null) ? "[" + this.ID + "]" : s);
            }

            set
            {
                ViewState["Text"] = value;
            }
        }

        protected override void RenderContents(HtmlTextWriter output)
        {
            output.Write(Text);
        }
    }
}
```

b. Remplacez ce code par le code de définition de la classe que vous voulez.

Dans un premier exemple, nous allons créer une classe de Web textbox dérivée du Web textbox de StrataFrame.

Nous lui ajouterons 2 propriétés supplémentaires (une de type String, une de type Booléen), et une méthode Overrides qui affichera le premier plan en noir, quand on demandera un affichage en bleu.

i. VB

```
Imports System
Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.Text
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls


<DefaultProperty("Text"), ToolboxData("<{0}:MyTextBox runat=server></{0}:MyTextBox>")> _
Public Class MyTextBox
    Inherits MicroFour.StrataFrame.UI.Web.TextBox

#Region "Propriétés Privées"
    Private _MaProp As String = ""
    Private _AutreProperty As Boolean = False
#End Region

#Region "Propriétés Publiques"

    <Bindable(True),
    Category("Mes Propriétés"),
    DefaultValue(""),
    Description("Ma propriété de type String nouvelle dans ma classe à moi"),
    Localizable(True)>
    Property MaProperty() As String
        Get
            Return _MaProp
        End Get
```

```
        Set(ByVal Value As String)
            _MaProp = Value
        End Set
    End Property

    <Category("Mes Propriétés"),
    DefaultValue(False),
    Description("Ma propriété de type booléen")>
    Property AutreProperty() As Boolean
        Get
            Return _AutreProperty
        End Get
        Set(value As Boolean)
            _AutreProperty = value
        End Set
    End Property


    ''' <summary>
    ''' si la couleur de premier plan est bleu, on affiche noir
    ''' </summary>
    ''' <value></value>
    ''' <returns></returns>
    ''' <remarks></remarks>
    Public Overrides Property ForeColor As System.Drawing.Color
        Get
            Return MyBase.ForeColor
        End Get
        Set(value As System.Drawing.Color)
            If value = Drawing.Color.Blue Then
                MyBase.ForeColor = Drawing.Color.Black
            Else
                MyBase.ForeColor = value
            End If
        End Set
    End Property
#End Region


End Class
```

Il n'y a absolument rien d'extraordinaire dans ce code, qui est très ressemblant à celui que nous aurions écrit pour une bibliothèque de classe pour WinForms. La seule particularité est dans la balise ToolBoxData.
La valeur du ToolBoxData définit le code de la balise qui sera généré sur notre page aspX, quand on déplacera le contrôle depuis la boite à outil vers la surface de design.

ii. C#

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.ComponentModel;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
[DefaultProperty("Text"),
ToolboxData("<{0}:MonTextBoxCsharp runat=server></{0}:MonTextBoxCsharp>")]
public class MonTextBoxCsharp : MicroFour.StrataFrame.UI.Web.TextBox
{

    #region "Propriétés Privées"
        private string _MaProp = "";
        private bool _AutreProperty = false;
    #endregion

    #region "Propriétés Publiques"

    [Bindable(true),
    Category("Mes Propriétés"),
    DefaultValue(""),
    Description("Ma propriété de type String nouvelle dans ma classe à moi"),
    Localizable(true)]
    public string MaProperty {
        get { return _MaProp; }

        set { _MaProp = value; }
    }

    [Category("Mes Propriétés"),
    DefaultValue(false),
    Description("Ma propriété de type booléen")]
    public bool AutreProperty {
        get { return _AutreProperty; }
        set { _AutreProperty = value; }
    }


    /// <summary>
    /// si la couleur de premier plan est bleu, on affiche noir
    /// </summary>
    /// <value></value>
    /// <returns></returns>
    /// <remarks></remarks>
    public override System.Drawing.Color ForeColor {
        get { return base.ForeColor; }
        set {
            if (value == System.Drawing.Color.Blue) {
                base.ForeColor = System.Drawing.Color.Black;
            } else {
                base.ForeColor = value;
            }
        }
    }
    #endregion


}
```

Ici aussi, on retrouve la balise ToolBoxData.

Dans un deuxième exemple, nous allons créer une classe de Web Button dérivée du Web Button natif de .NET.

Les seules surcharges dans cette classe seront la couleur par défaut, et le texte affiché par défaut.

i. VB

```
Imports System
Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.Text
Imports System.Web
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Drawing


<DefaultProperty("Text"),
ToolboxData("<{0}:MyButton runat=server></{0}:MyButton>"),
ToolboxBitmap(GetType(Button))> _
Public Class MyButton
    Inherits Button

#Region "Propriétés Privées"

#End Region

#Region "Propriétés Publiques"

#End Region


    Public Sub New()
        Me.BackColor = Drawing.Color.Aquamarine
        Me.Text = "Coucou!"
    End Sub
End Class
```

ii. C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing;

namespace MesClassesWeb
{
    [DefaultProperty("Text")]
    [ToolboxData("<{0}:MonBoutonCsharp runat=server></{0}:MonBoutonCsharp>")]
    [ToolboxBitmap(typeof(Button))]
    public class MonBoutonCsharp : Button
    {
        [Bindable(true)]
        [Category("Appearance")]
        [DefaultValue("")]
        [Localizable(true)]
```

```
        public string Text
        {
            get
            {
                String s = (String)ViewState["Text"];
                return ((s == null) ? String.Empty : s);
            }

            set
            {
                ViewState["Text"] = value;
            }
        }

        public MonBoutonCsharp()
        {
            this.Text = "Coucou C#";
            this.BackColor = Color.Aqua;
        }
    }
}
```

La particularité supplémentaire ici est la balise ToolBoxBitmap, qui permet de spécifier l'icône qui figurera dans la boite à outils :
sans cette balise, votre composant apparaitra dans la barre d'outils avec une icône par défaut 
Vous pouvez spécifier un fichier image sur le disque, mais dans notre exemple, nous récupérons l'image associée par défaut à la classe parent.

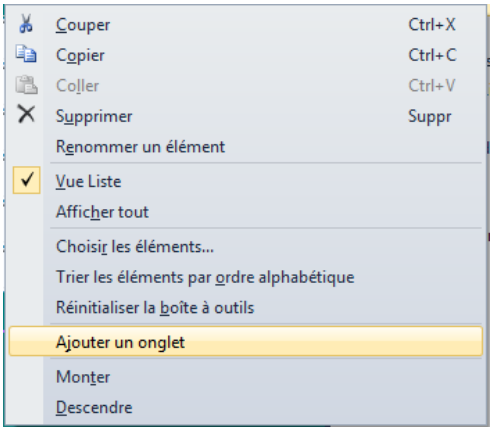
3. Compilation

Nous pouvons compiler le projet, mais faîtes attention à la version du framework dans laquelle vous allez compiler. Si vous voulez pouvoir utiliser ces contrôles dans des projets existants préalablement, alors vous devez compiler dans une version compréhensible pour ces projets.

4. Utilisation de ces classes

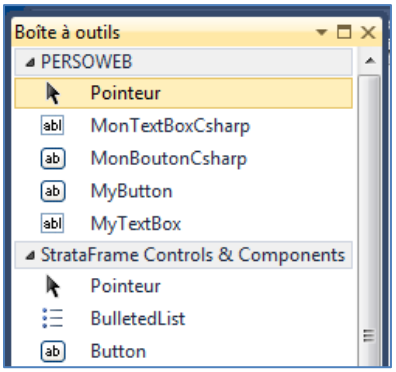
a. Configurer la Boite à Outils

Nous allons commencer par ajouter un nouvel onglet à notre boite à outils (Click droit, puis ajouter un Onglet)



Puis dans ce nouvel onglet, nous allons ajouter les éléments voulus ; même s'il est possible de « jeter » directement la dll compilée depuis l'explorateur de fichier vers ce nouvel onglet, je déconseille cette méthode qui ne permet pas de choisir les éléments à ajouter de façon individualisée. Cliquez plutôt sur « Choisir les éléments ...» dans le menu contextuel du click droit dans ce nouvel onglet vide, et allez chercher la dll compilée. Vous pouvez choisir les élémnts de façon individualisée.

Si vous ne voyez pas immédiatement votre nouvel onglet et vos contrôles ajoutés, commencez par ouvrir une page aspx. Vous devriez alors voir ceci :

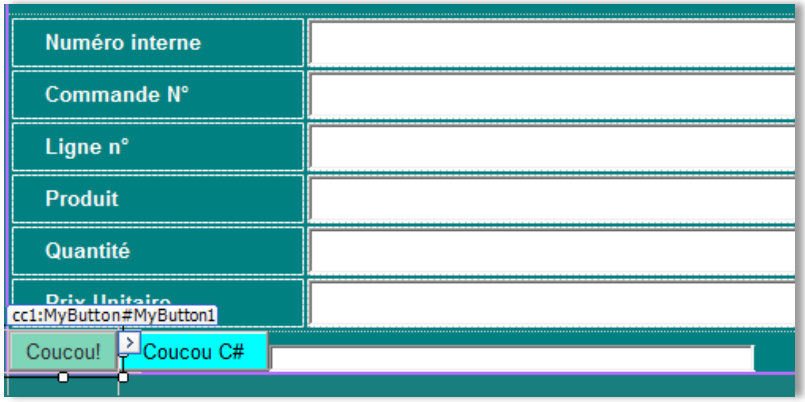


On remarque que les classes issues de « Button » sont bien identifiées par l'icône correcte.

b. Consommer dans une page ASPX

N'oubliez pas de référencer votre bibliothèque de classe dans votre projet, ensuite il suffit de procéder comme pour tout autre contrôle, en le faisant glisser depuis la boite à outil vers la surface de design ou la surface de source.

i. En mode de Design

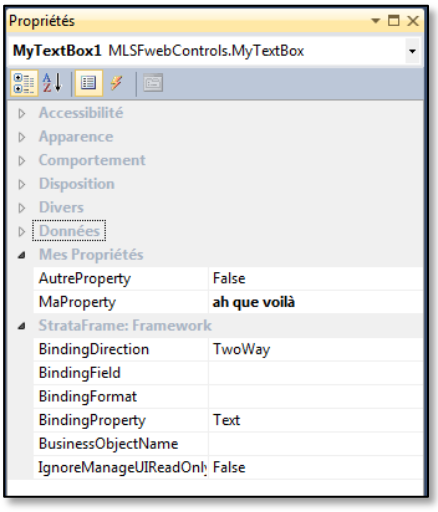


ii. En mode Source

```
<cc1:MyButton ID="MyButton1" runat="server" />
<cc2:MonBoutonCsharp ID="MonBoutonCsharp1" runat="server" />
<cc1:MyTextBox ID="MyTextBox1" runat="server" Height="17px"
    MaProperty="ah que voilà" style="margin-top: 0px" Width="299px"></cc1:MyTextBox>
```

c. Les propriétés

on retrouve bien la catégorie créée, ainsi que les 2 propriétés ajoutées. On voit que notre classe dérive bien du TextBox de StrataFrame.



d. Le code behind

```
Protected Sub MonBoutonCsharp1_Click(sender As Object, e As EventArgs) _
    Handles MonBoutonCsharp1.Click
    Me.MyTextBox1.ForeColor = Drawing.Color.Blue
End Sub

Protected Sub MyButton1_Click(sender As Object, e As EventArgs) _
    Handles MyButton1.Click
    Me.MyTextBox1.Text += Me.MyTextBox1.MaProperty.ToUpper() +
        Me.MyTextBox1.AutreProperty.ToString
    Me.MyTextBox1.ForeColor = Drawing.Color.Red
End Sub
```

Cet exemple nous permet de vérifier que nos propriétés définies dans les classes sont accessibles et utilisables, et que le code que nous avions écrit dans les classes est bien exécuté.

En conclusion…

Nous avons appris à créer une bibliothèque de contrôles web pour des pages aspx, dans le langage de notre choix. Nous savons ajouter ces contrôles à notre boite à outils, et utiliser ces contrôles comme tout autre.