Les Contrôles Web personnalisés

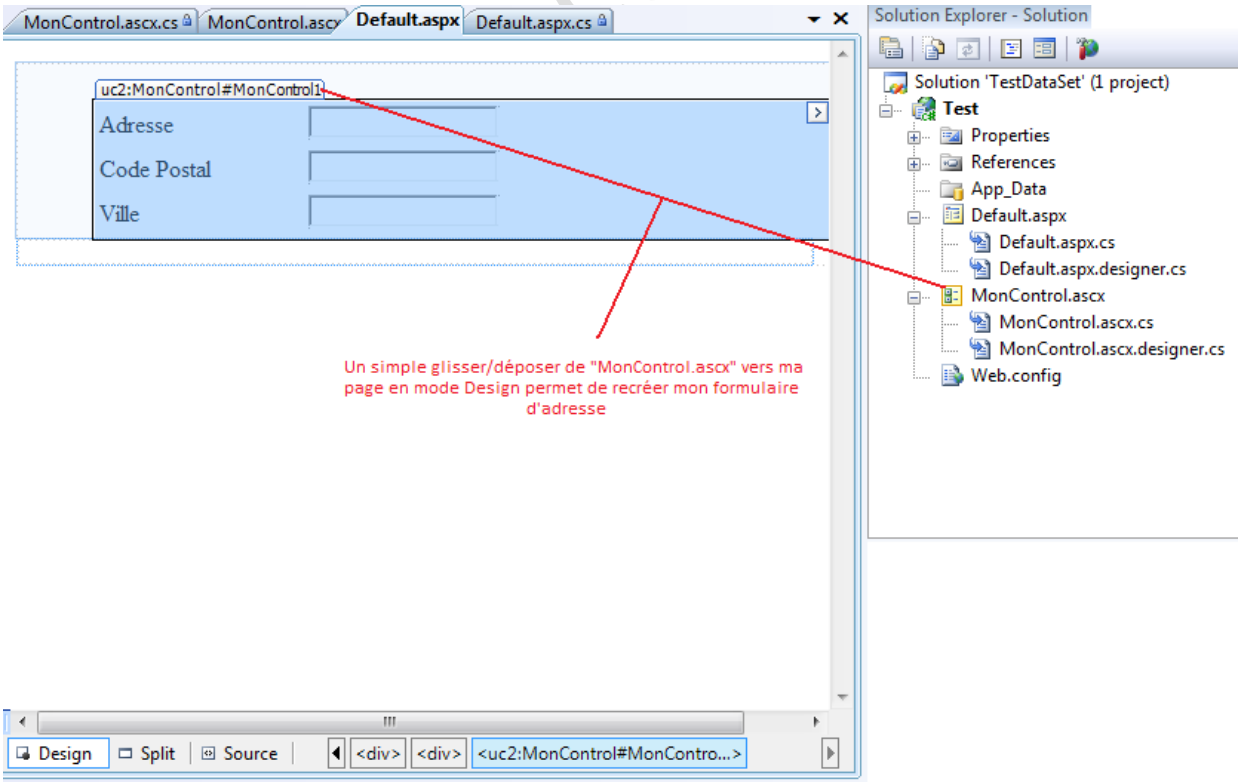
Sommaire

1 Présentation

Lors du développement d'une solution, Visual Studio met à notre disposition une multitude d'outils pour la création de page Web. Par exemple les *Label* ou encore les *TextBox* que l'on peut incorporer dans notre solution par un simple Drag&Drop (glisser/déposer).

Ceci étant, on s'est aperçu que l'on perdait beaucoup de temps à coder les même choses mais pour des pages web différentes. Par exemple, on utilise très souvent des formulaires pour entrer des adresses, contenant un *Label* pour chaque instruction (adresse, code postal, ville etc…) et une *TextBox* pour chaque entrée de l'utilisateur :

L'ASP.NET nous permet de créer nos propres contrôles que l'on va ajouter à notre solution. Ainsi notre formulaire d'adresse deviendra accessible tout comme une *TextBox* mais restera un élément de notre solution.



Nous aborderons ce chapitre en trois parties :

- Les User Controls (Contrôle Utilisateur) ;

- Les Web Server Controls (Contrôle Web Serveur) ;
- Les Composite Control (Contrôle Composé).

2 Les User Controls

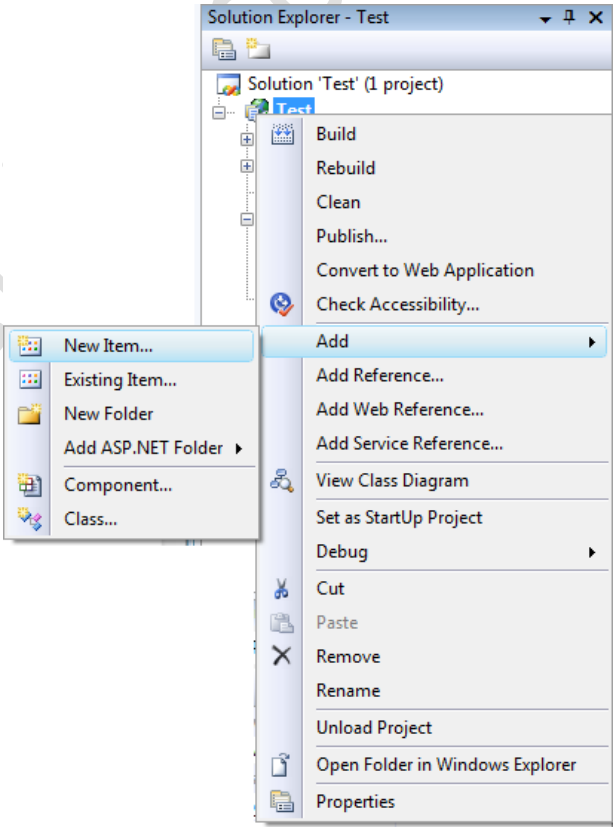
Les User Controls sont des templates (modèles/schémas) personnalisables que l'on va pouvoir ajouter à sa solution. Les User Controls pourront être constitués d'un ou plusieurs Controls proposés dans la boite à outils (*TextBox, DropDownList,…*). Ils pourront être utilisés autant de fois qu'on le souhaite pour un même projet.

En revanche, ils ne peuvent être utilisés que pour un seul projet. Si on veut réutiliser ce même contrôle dans d'autres projets, il faudra l'inclure dans chacun au préalable.

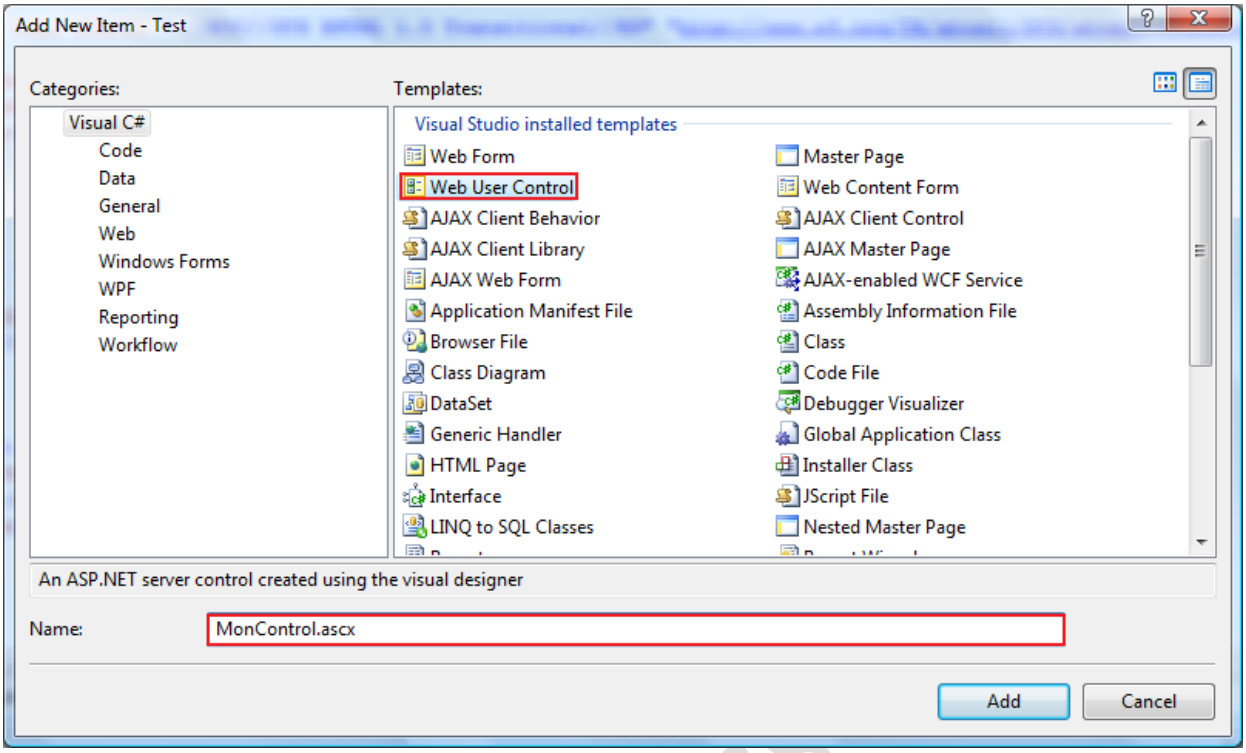
2.1 Créer et ajouter un User Control dans une page Web

Commençons par ajouter un nouveau Control à notre projet.

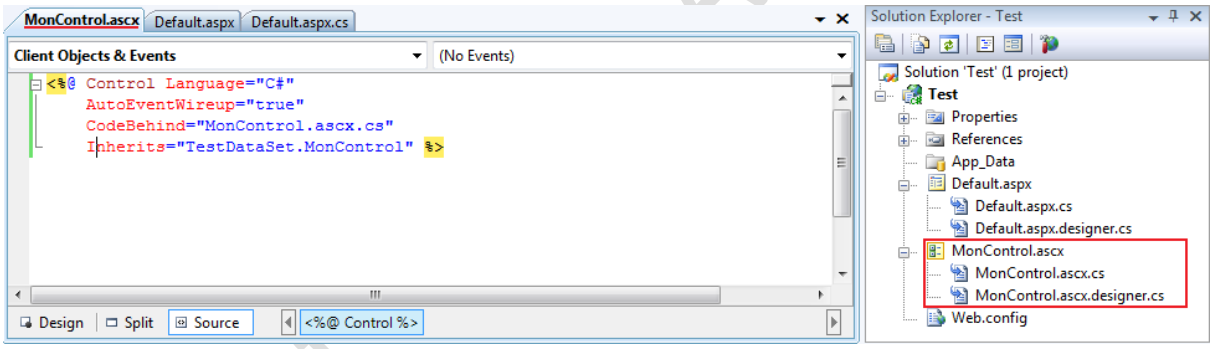
Tout d'abord faites un clic droit sur votre projet dans l'explorateur de Solution. Puis "Ajouter un nouvel élément".



Ensuite Choisissez un Web User Control. Puis renommez-le (« *MonControl.ascx* » par exemple).



Votre premier contrôle est créé, on peut maintenant le consulter dans la solution.



Du *CodeBehind* est automatiquement généré afin de pouvoir effectuer une gestion des évènements. Nous y reviendrons plus tard.

Commençons par ajouter des éléments dans notre User Control. Pour cela passez en mode Design. Dans cet exemple nous effectuerons un formulaire d'adresse pour récupérer les informations suivantes : NOM, Prénom, Adresse 1, Adresse 2, Code Postal, VILLE. Dans un souci d'esthétique nous mettrons tous nos éléments dans un tableau HTML pour que tous nos champs soient alignés. Rappelons que le raccourci pour utiliser la boite à outil est Ctrl+W puis la touche X. Ce cours n'ayant pas pour but de vous familiariser avec Visual Studio nous estimerons que la mise en place des éléments tant en mode Design qu'en mode Source est acquise.

Voici le Code Source de MonControl.ascx.

Page ASCX – C#

```
<%@ Control Language="C#"
    AutoEventWireup="true"
    CodeBehind="MonControl.ascx.cs"
    Inherits="Test.MonControl" %>
<style type="text/css">
    .style1{width: 162px;}
    .style2{width: 82px;}
    .style3{width: 102px;}
    .style4{width: 126px;}
</style>
<table style="width:100%;">
    <tr>
        <td class="style3">
            <asp:Label ID="Label1" runat="server" Text="NOM"></asp:Label>
        </td>
        <td class="style4">
            <asp:TextBox ID="lastname" runat="server" Width="190px"></asp:TextBox>
        </td>
        <td class="style2">
            <asp:Label ID="Label2" runat="server" Text="Prénom"></asp:Label>
        </td>
        <td class="style1">
            <asp:TextBox ID="firstname" runat="server" Width="190px"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td class="style3">
            <asp:Label ID="Label3" runat="server" Text="Adresse 1"></asp:Label>
        </td>
        <td colspan="3">
            <asp:TextBox ID="firstAddress" runat="server"
            Width="300px"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td class="style3">
            <asp:Label ID="Label4" runat="server" Text="Adresse 2"></asp:Label>
        </td>
        <td colspan="3">
            <asp:TextBox ID="secondAddress" runat="server"
            Width="300px"></asp:TextBox>
        </td>
    </tr>
    <tr>
        <td class="style3">
            <asp:Label ID="Label5" runat="server" Text="Code Postal"></asp:Label>
        </td>
        <td class="style4">
            <asp:TextBox ID="zipCode" runat="server" Width="190px"></asp:TextBox>
        </td>
        <td class="style2">
            <asp:Label ID="Label6" runat="server" Text="Ville"></asp:Label>
        </td>
        <td class="style1">
            <asp:TextBox ID="town" runat="server" Width="190px"></asp:TextBox>
        </td>
    </tr>
</table>
```

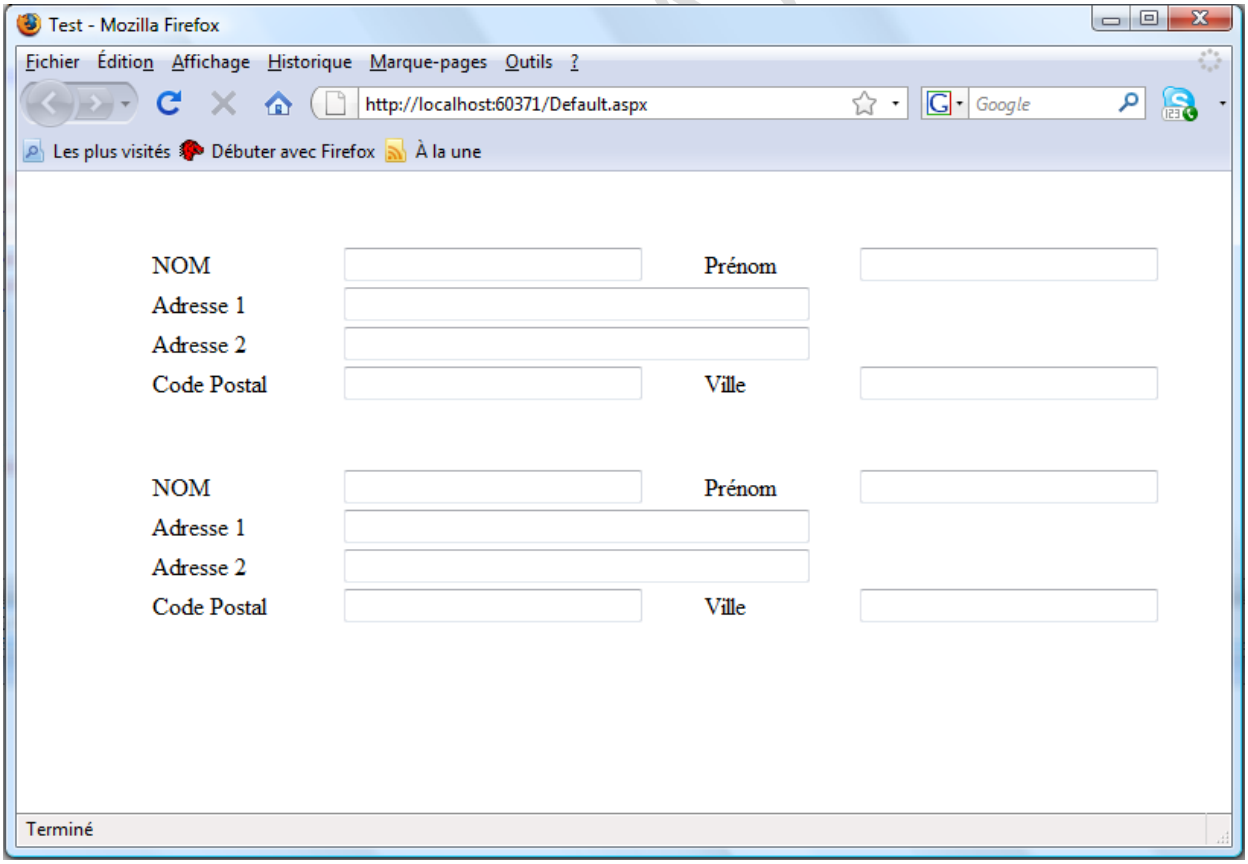
Page ASCX – VB.NET

```
<%--Changer la directive Control de l'exemple C# par : --%>
<%@ Control Language="vb"
    AutoEventWireup="false"
    CodeBehind="WebUserControl1.ascx.vb"
   Inherits="WebVBNET.WebUserControl1" %>
```

Voici ce que vous devriez obtenir après avoir renommé vos *Label*, vos *TextBox* et ajusté votre tableau en mode Design.

NOM		Prénom	
Adresse 1			
Adresse 2			
Code Postal		Ville	

Revenez maintenant sur votre page Default.aspx en mode Design et faite un Drag&Drop de votre Contrôle sur votre page. Compilez (F5). Votre Control apparait sur votre page Web comme n'importe quel autre formulaire. Il est bien sur possible d'afficher plusieurs fois dans la même page ce petit formulaire en répétant l'action.



Nous savons maintenant comment créer un User Control.

Du code a été généré pour nous. Voici le Code Source de la page *Default.aspx.*

ASP.NET

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="Test._Default" %>

<%@ Register src="MonControl.ascx" tagname="MonControl" tagprefix="uc1" %>

<!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 runat="server">
    <title>Test</title>
</head>
<body>
    <form id="form1" runat="server">

        <%-- On fait une petite marge en haut et à gauche--%>
        <div style="padding-top:5%;padding-left:10%">
            <uc1:MonControl ID="MonControl1" runat="server" />
            <br />
            <br />
        </div>

    <asp:Panel ID="Panel1" runat="server">
    </asp:Panel>
    </form>
</body>
</html>
```

Les parties du Code soulignées sont celles qui nous interessent.

```
<%@ Register src="MonControl.ascx" tagname="MonControl" tagprefix="uc1" %>
```

Cette portion de code se place avant le Doctype de la page et après la directive Page.

@ Register	Cette balise est obligatoire pour placer le Control sur la Page.
src="MonControl.ascx"	Permet de déterminer la source où se trouve notre Control.
tagname="MonControl"	Permet de donner un nom à notre Control qui autorisera son accès pour l'utiliser.
tagprefix="uc1"	Permet de créer un espace de nomage qui aura pour but d'identifier nos Controls.

```
<uc1:MonControl ID="MonControl1" runat="server" />
```

Ce code se place dans notre page à l'endroit où vous désirez afficher votre Control par exemple dans une *div* ayant des propriété d'affichage particulière.

uc1	Cette balise est obligatoire et fait référence au *tagprefix*, afin de pouvoir récupérer et utiliser notre Control.
MonControl	Cette balise permet de récupérer notre Control afin de l'instancier.
ID="MonControl1"	Cette propiété permet de donner un identifiant à l'instance de notre Control afin de pouvoir y accéder.

Remarque :

Il est possible de créer un User Control à partir d'une page Web.

Dans ce cas, il ne faut pas oublier qu'un Control n'utlilise pas de balise *html et body*. Il faut donc les retirer de notre page Web. Par ailleurs la directive Page doit devenir une directive Control.

Directive Page :	Directive Control :
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="MaPage.aspx.cs" Inherits="Test.MaPage" %>	<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="MaPage.ascx.cs" Inherits="Test.MaPage" %>

On peut remarquer que la Directive `@ Page` devient une Directive `@ Control` et que l'extension de MaPage en CodeBehind doit aussi être modifiée. En effet *MaPage.aspx.cs* devient *MaPage.ascx.cs* : *aspx* étant l'extension des pages Web en ASP.NET et *ascx* étant celle des User Controls.

Enfin, il est nécessaire de renommer *MaPage.aspx* en *MaPage.ascx* pour terminer la procédure et que notre page Web devienne définitivement un User Control utilisable.

2.2 Accéder aux données des User Controls

Il existe deux façons d'accéder aux données des User Controls. Bien entendu ces deux méthodes ne sont pas au choix, chacune présente un intérêt que l'on va déterminer au cours de cette partie.

2.2.1 Accéder aux données dans les User Controls

De façon très intuitive, il parait évident que l'on peut accéder aux données à l'intérieur de notre User Control via le CodeBehind dans le fichier *MonControl.ascx.cs*.

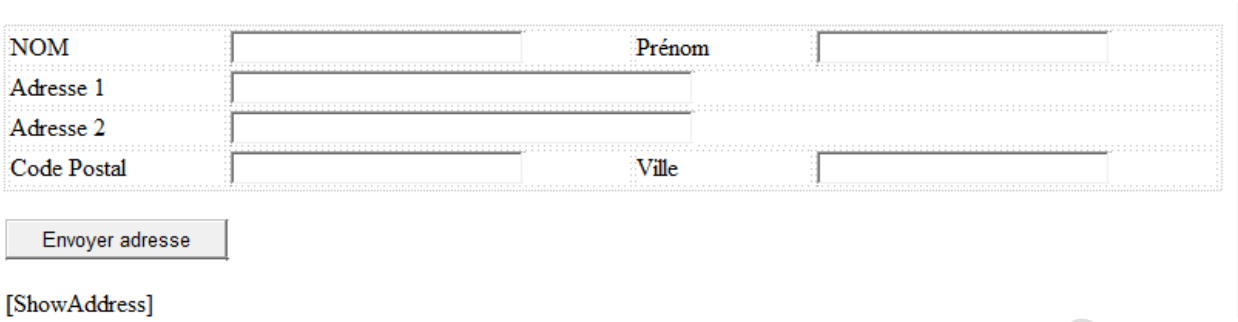
Comme nous sommes à l'intérieur de notre Control, on peut facilement accéder à tous ses attributs ainsi qu'à toutes ses méthodes privées. Il est, cela dit, fortement recommandé de créer des accesseurs (*get/set*) et de les utiliser même au sein de notre Control.

Pour les besoins de l'exemple suivant, nous allons réutiliser notre Control créé dans la partie précédente et y ajouter un *Button* puis un *Label* dans le but d'afficher dans ce *Label* nos entrées utilisateur du formulaire d'adresse.

On ajoutera aussi un évènement *onclick* sur le *Button.*

Voir ci-dessous le Design et le Code Source :

Design :



Code Source Supplémentaire dans *MonControl.ascx*:

```
ASP.NET

<%-- A rajouter à la fin de code --%>
<p>
    <asp:Button ID="Button1" runat="server" Text="Envoyer adresse"
        onclick="Button1_Click" />
</p>
<p>
    <asp:Label ID="ShowAddress" runat="server"></asp:Label>
</p>
```

Passons maintenant au CodeBehind dans le fichier *MonControl.ascx.cs*. Dans un souci de sécurité nous créerons tous les accesseurs adéquat.

Nous allons aussi gérer l'évènement du *Button1* afin que, lorsqu'on cliquera sur ledit bouton, toutes nos informations s'affichent dans notre *Label : ShowAddress*.

Code Source de *MonControl.ascx.cs* :

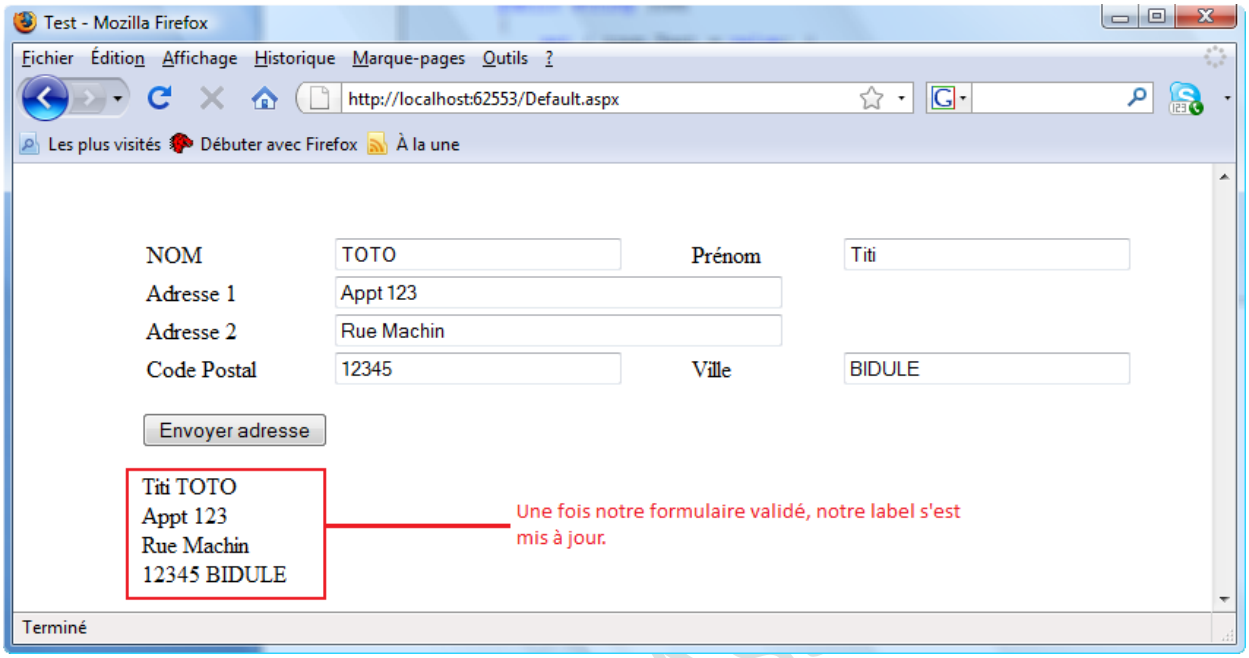
C#

```
public partial class MonControl : System.Web.UI.UserControl
{

    public string Lastname
    {
        set { lastname.Text = value; }
        get { return lastname.Text; }
    }

    public string Firstname
    {
        set { firstname.Text = value; }
        get { return firstname.Text; }
    }

    public string FirstAddress
    {
        set { firstAddress.Text = value; }
        get { return firstAddress.Text; }
    }
    public string SecondAddress
    {
        set { secondAddress.Text = value; }
        get { return secondAddress.Text; }
    }

    public string ZipCode
    {
        set { zipCode.Text = value; }
        get { return zipCode.Text; }
    }

    public string Town
    {
        set { town.Text = value; }
        get { return town.Text; }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        ShowAddress.Text = Firstname + " " + Lastname + "<br />"
                         + FirstAddress + "<br />"
                         + SecondAddress + "<br />"
                         + ZipCode + " " + Town;
    }
}
```

VB.NET

```
Public Partial Class MonControl
    Inherits System.Web.UI.UserControl


    Public Property GetSetLastName()
        Get
            Return lastname.Text
        End Get
        Set(ByVal value)
            lastname.Text = value
        End Set
    End Property

    Public Property GetSetFirstName()
        Get
            Return firstname.Text
        End Get
        Set(ByVal value)
            firstname.Text = value
        End Set
    End Property

    Public Property GetSetFirstAddress()
        Get
            Return firstAddress.Text
        End Get
        Set(ByVal value)
            firstAddress.Text = value
        End Set
    End Property

    Public Property GetSetSecondAddress()
        Get
            Return secondAddress.Text
        End Get
        Set(ByVal value)
            secondAddress.Text = value
        End Set
    End Property

    Public Property GetSetZipCode()
        Get
            Return zipCode.Text
        End Get
        Set(ByVal value)
            zipCode.Text = value
        End Set
    End Property


    Public Property GetSetTown()
        Get
            Return town.Text
        End Get
        Set(ByVal value)
            town.Text = value
        End Set
    End Property

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs)
        ShowAddress.Text = GetSetFirstName & " " & GetSetLastName & "<br />" &
GetSetFirstAddress & "<br />" & GetSetSecondAddress & "<br />" & GetSetZipCode & " "
& GetSetTown
    End Sub
End Class
```

Voici à quoi doit ressembler notre projet une fois compilé :



L'avantage de cette procédure est que l'on peut créer un Control ayant une multitude de fonctionnalités déjà toutes prêtes. Ceci étant il faut garder en tête que lorsqu'on crée un Control, il a pour but d'être réutilisé. Un Control avec des fonctionnalités trop ciblés perdra de son utilité et on préfèrera ajouter ces fonctionnalités en dehors de ce control.

Ce qui nous amène à notre seconde sous-partie : comment récupérer des données en dehors de notre Control.

2.2.2 Accéder aux données à l'extérieur des User Controls

Tout comme pour notre Control, ajoutons un *Button* et un *Label* à la page *Default.aspx* juste après notre control.

ASP.NET

```
    <form id="form1" runat="server">
    <div style="padding-top:5%;padding-left:10%">

        <uc1:MonControl ID="MonControl1" runat="server" />

        <br />
        Evenement géré à l&#39;extérieur de MonControl<br />
        <br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
            Text="Envoyer adresse" />
        <br />
        <br />
        <asp:Label ID="ShowAddress" runat="server"></asp:Label>
        <br />
        <br />

    </div>
    </form>
```

Pour pouvoir récupérer les données de votre Control, créer des accesseurs est maintenant, non plus fortement conseillé, mais **indispensable**. Pour accéder aux données de votre Control dans le CodeBehind il faudra commencer par donner l'ID de votre Control suivi de la méthode à utiliser : *MonControl1.maMethode()*.

Remarque : Les accesseurs ne prennent pas d'argument : *MonControl1.monAccesseur*.

Si vous utilisez plusieurs fois le même contrôle dans la même page, l'*ID* permettra de différencier chaque instance de votre Control et sera le seul moyen d'y accéder, n'hésitez pas à les renommer pour plus de clarté dans votre code.

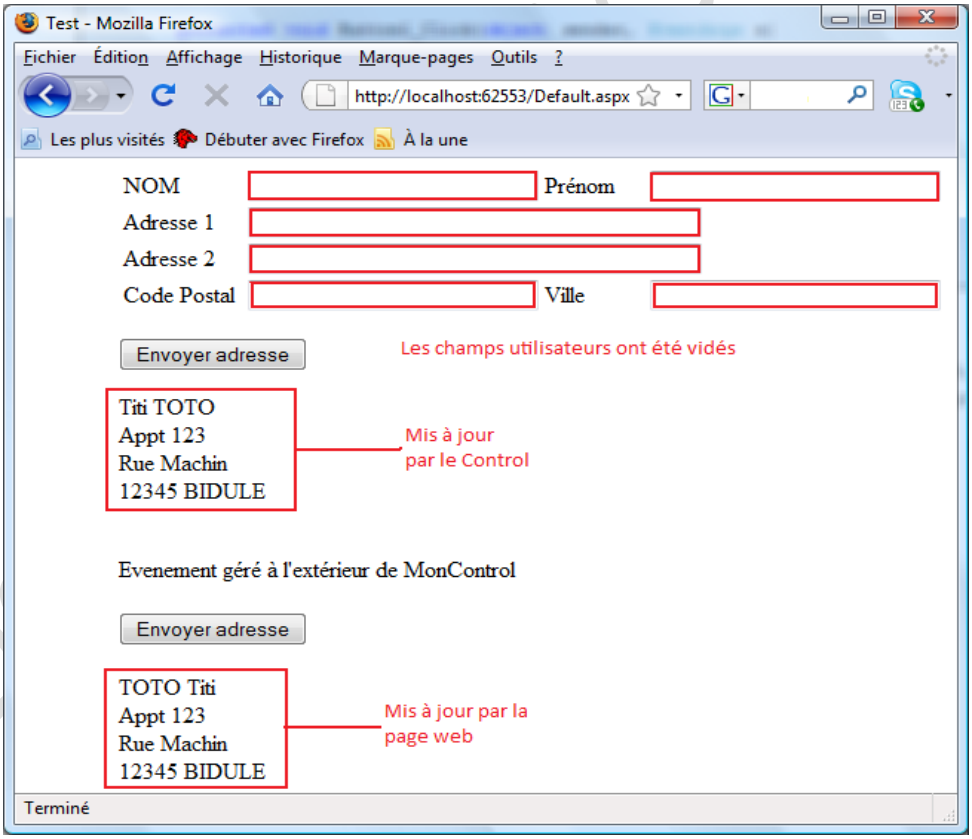
Afin de bien démontrer que toute les données sont accéssibles et utilisables nous apporterons une légère modification : les champs utilisateur du Control *MonControl1* seront vidés après avoir cliqué sur le bouton *Button1*.

C#

```
public partial class _Default : System.Web.UI.Page
    {
        protected void Button1_Click(object sender, EventArgs e)
        {
            ShowAddress.Text = MonControl1.Lastname + " " + MonControl1.Firstname
                             + "<br />"
                             + MonControl1.FirstAddress + "<br />"
                             + MonControl1.SecondAddress + "<br />"
                             + MonControl1.ZipCode + " " + MonControl1.Town;

            MonControl1.Firstname = "";
            MonControl1.Lastname = "";
            MonControl1.FirstAddress = "";
            MonControl1.SecondAddress = "";
            MonControl1.ZipCode = "";
            MonControl1.Town = "";
        }
    }
```

```
VB.NET

Partial Public Class _Default
    Inherits System.Web.UI.Page

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As EventArgs)
    ShowAddress.Text = MonControl1.GetSetLastName + " " _
         + MonControl1.GetSetFirstName + "<br />" _
         + MonControl1.GetSetFirstAddress + "<br />" _
         + MonControl1.GetSetSecondAddress + "<br />" _
         + MonControl1.GetSetZipCode + " " + MonControl1.GetSetTown

    MonControl1.GetSetFirstName = ""
    MonControl1.GetSetLastName = ""
    MonControl1.GetSetFirstAddress = ""
    MonControl1.GetSetSecondAddress = ""
    MonControl1.GetSetZipCode = ""
    MonControl1.GetSetTown = ""
End Sub
End Class
```

Après compilation, on peut toujours utiliser le bouton créé dans le Control même, les fonctionnalités ajoutées dans la page web ne remplacent pas celles déjà existante, même si leur but est identique.



Remarque:

Il bien entendu possible d'ajouter un Control à notre page de façon dynamique en passant pas le CodeBehind.

Ceci peut-être utile lorsqu'on veut, par exemple, initialiser des champs utilisateurs au chargement de la page.

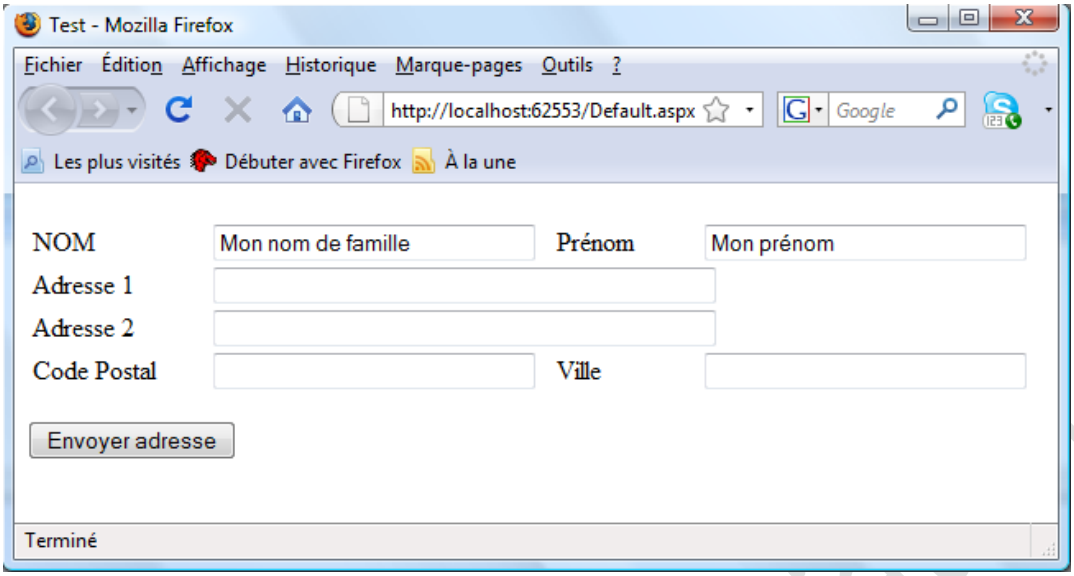
C#

```
public partial class _Default : System.Web.UI.Page
    {
         protected void Page_Load(object sender, EventArgs e)
        {
            // Lorsqu'on veut appeler un Control dynamiquement il faut
            // aussi le charger via le nom de sa page ou il est créé
            MonControl c1 = (MonControl)LoadControl("MonControl.ascx");
            // On crée un nouveau Control de type MonControl de nom c1
            // Il faut charger MonControl à cette adresse : "MonControl.ascx"
            c1.Lastname = "Mon nom de famille";
            c1.Firstname = "Mon prénom";
            // On initialise des attributs
            //Puis on l'ajoute à notre page Web
            form1.Controls.Add(c1);

        }
    }
```

VB.NET

```
Partial Public Class _Default
    Inherits System.Web.UI.Page


    Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)

        ' Lorsqu'on veut appeler un Control dynamiquement il faut
        ' aussi le charger via le nom de sa page ou il est créé
        Dim c1 As MonControl = CType(LoadControl("MonControl.ascx"), MonControl)
        ' On crée un nouveau Control de type MonControl de nom c1
        ' Il faut charger MonControl à cette adresse : "MonControl.ascx"
        c1.GetLastname = "Mon nom de famille"
        c1.GetFirstname = "Mon prénom"
        ' On initialise des attributs
        ' Puis on l'ajoute à notre page Web
        form1.Controls.Add(c1)
    End Sub

End Class
```

Après compilation :



2.2.3 Lever un évènement vers la page Web

Lors de la création d'un contrôle, on peut facilement créer un *Button*, cela dit il n'est pas évident de prévoir comment celui-ci sera implanté. Afin de pallier à ce problème, il est possible de lever un évènement directement sur la page Web. Le code suivant va nous montrer comment gérer ce type de problème en passant par un *delegate*.

N.B. : Afin de bien comprendre cette partie il est nécessaire de connaitre et de comprendre l'utilité des *delegates* et des *events.*

Code du Control *Test* :

ASPX de Test.ascx

```
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="Test.ascx.cs"
Inherits="Test.Test" %>
<p>
    Entrer Nom :
    <asp:TextBox ID="Nom" runat="server"></asp:TextBox>
</p>
<p>
    <asp:Button ID="SendNom" runat="server" Text="Button" onclick="SendNom_Click" />
</p>
```

C# de Test.ascx.cs

```
//On commence par déclarer notre type de délégué
public delegate void SendNom(string message);

namespace Test
{
    public partial class Test : System.Web.UI.UserControl
    {
        //On crée un nouvel event appelé SendMessage
        // du type de notre delegate : SendNom
        public event SendNom SendMessage;

        //On gère notre évenement SendNom_Click
        protected void SendNom_Click(object sender, EventArgs e)
        {
            //Si notre délégué n'est pas vide on prend en
            // paramêtre notre TextBox Nom.Text
            if (SendMessage != null) SendMessage(Nom.Text);
            //On peut remarquer que bien que notre delegate ait été instancié
            // il ne contient actuellement aucune méthode(SendMessage == null),
            // elles seront rajoutées par la suite directement sur la page qui
            // contiendra notre contrôle
        }
    }
}
```

VB.NET

```
Partial Public Class Test
    Inherits System.Web.UI.UserControl

    ' On commence par déclarer notre type de délégué
    Public Delegate Sub delSendNom(ByVal message As String)

    ' On crée un nouvel event appelé SendMessage
    ' du type de notre delegate : SendNom
    Public Event SendMessage As delSendNom

    ' On gère notre évenement SendNom_Click
    Protected Sub SendNom_Click(ByVal sender As Object, ByVal e As EventArgs)
        ' On utilise l'évènement que l'on a créé pour envoyer le texte de la TextBox
        RaiseEvent SendMessage(Nom.Text)
        'On peut remarquer que bien que notre delegate ait été instancié
        ' il ne contient actuellement aucune méthode(SendMessage == null),
        ' elles seront rajoutées par la suite directement sur la page qui
        ' contiendra notre contrôle
    End Sub

End Class
```

Code de notre Page Web *MaPage* :

ASPX de MaPage.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="MaPage.aspx.cs"
Inherits="Test.MaPage" %>

<%@ Register Src="Test.ascx" TagName="Test" TagPrefix="uc1" %>
<!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 runat="server">
    <title>Ma Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <uc1:Test ID="Test1" runat="server" />
        <asp:Label ID="Label1" runat="server"></asp:Label>
        <br />
    </div>
    </form>
</body>
</html>
```

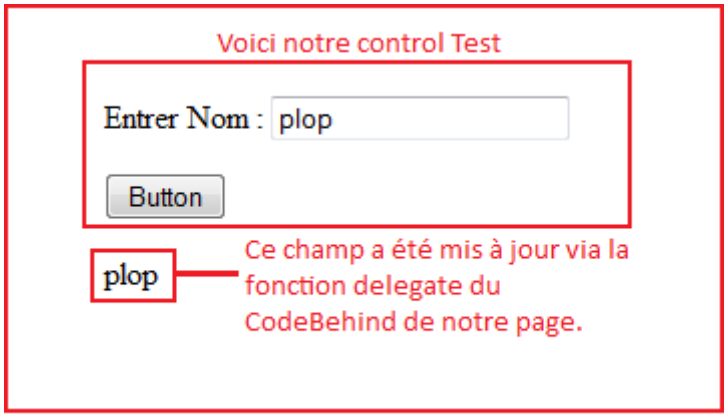
C# de MaPage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
   {
       //Nous ajoutons une méthode anonyme à notre delegate
       // notre SendMessage sera alors != null
       Test1.SendMessage += delegate(string message) { Label1.Text = message ; } ;
       //Il ne faut pas oublier que notre delegate anonyme prendra comme paramètre
       // le Text de la TextBox Nom de notre control
   }
```

VB.NET de MaPage.aspx.vb

```
Partial Public Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        Handles Me.Load
        AddHandler Test1.SendMessage, AddressOf MessageReceived
            ' Le AddHandler va récupérer le SendMessage de notre contrôle Test1
            ' Dès qu'il aura reçu, il va exécuter la méthode qui se trouve dans
              AdressOf
    End Sub

    Private Sub MessageReceived(ByVal message As String)
        Label1.Text = message
    End Sub
End Class
```

Aperçu :



MaPage.aspx

3 Les Customs Web Controls et Composite Control

Les Customs Web Controls (Contrôles Web Personnalisés) sont des contrôles que l'on va créer en faisant hériter une classe d'une classe déjà existante. Les Composite Control (ou contrôle composé) sont tout simplement des contrôles qui héritent de la classe *CompositeControl*. Dans l'un comme dans l'autre, on ne travaille qu'avec du code *C#/VB.NET*. Nous verrons cela plus en détails dans les parties suivantes.

3.1 Contrôle Web personnalisé

Pour modifier le rendu du contrôle dans la page, on utilise une surcharge de la méthode Render. Pour ce contrôle notre classe peut hériter d'une classe déjà existante d'un WebControl (*TextBox*, *Label* …) ou de la classe *WebControl* elle-même. Voyons cela avec un exemple qui utilisera pour héritage la classe *TextBox*.

C#

```
public class MonControl : TextBox //Notre classe MonControl hérite de la classe
Textbox
                                  //Il possèdera donc les méthodes, attributs,
évennements ... de cette classe
{
    //On créé deux attributs en private qui vont nous permettre de modifier la
largeur (width) et le texte du bouton
    private int _buttonWidth;
    private string _buttonText = "Rechercher";

    //On créer des accesseurs / mutateurs pour pouvoir changer les valeurs des
attributs
    public int ButtonWidth
    {
        get { return _buttonWidth; }
        set { _buttonWidth = value; }
    }

    public string ButtonText
    {
        get { return _buttonText; }
        set { _buttonText = value; }
    }

    //On surcharge la méthode Render pour personnalisé le contrôle
    protected override void Render(HtmlTextWriter writer)
    {
        //On utilise l'objet récupéré (l'objet HtmlTextWriter) pour créer notre
propre rendu
        writer.Write(@"<div style=""display:inline;"">"); //Cet objet possède une
méthode Write qui nous permet d'écrire dans le flux. On peut ainsi y mettre le Html
que l'on souhaite
        //Le @ devant permet de faire un guillemet avec "" mais aussi de pouvoir
ainsi le caractère \ même si ici ce n'est pas utile.
        base.BorderColor = System.Drawing.Color.Blue; // Le mot clef base permet
d'accéder aux valeurs de la classe hérité et donc de changer ce qui touche au
TextBox
        base.Text = "Tapper la recherche ici"; // On a donc changé la couleur de la
bordure mais aussi le texte du TextBox
        base.Render(writer); //Ici on applique le Render de la classe parent ... On
aura donc l'affichage d'un contrôle TextBox
        writer.Write("  ");
        writer.Write(@"<input type=""button"" Value=""{0}"" ", ButtonText); // Le
{0} correspond au premier paramètre après la fermeture du guillemet. 1 signifierai
le second etc

        if (_buttonWidth != 0) //Si on à une taille défini alors on l'applique
            writer.Write(@"style=""width:{0}px""", ButtonWidth);

        writer.Write("/>"); //sinon on ferme le bouton que l'on à écrit
        writer.Write("</div>"); //On ferme la div
    }
}
```

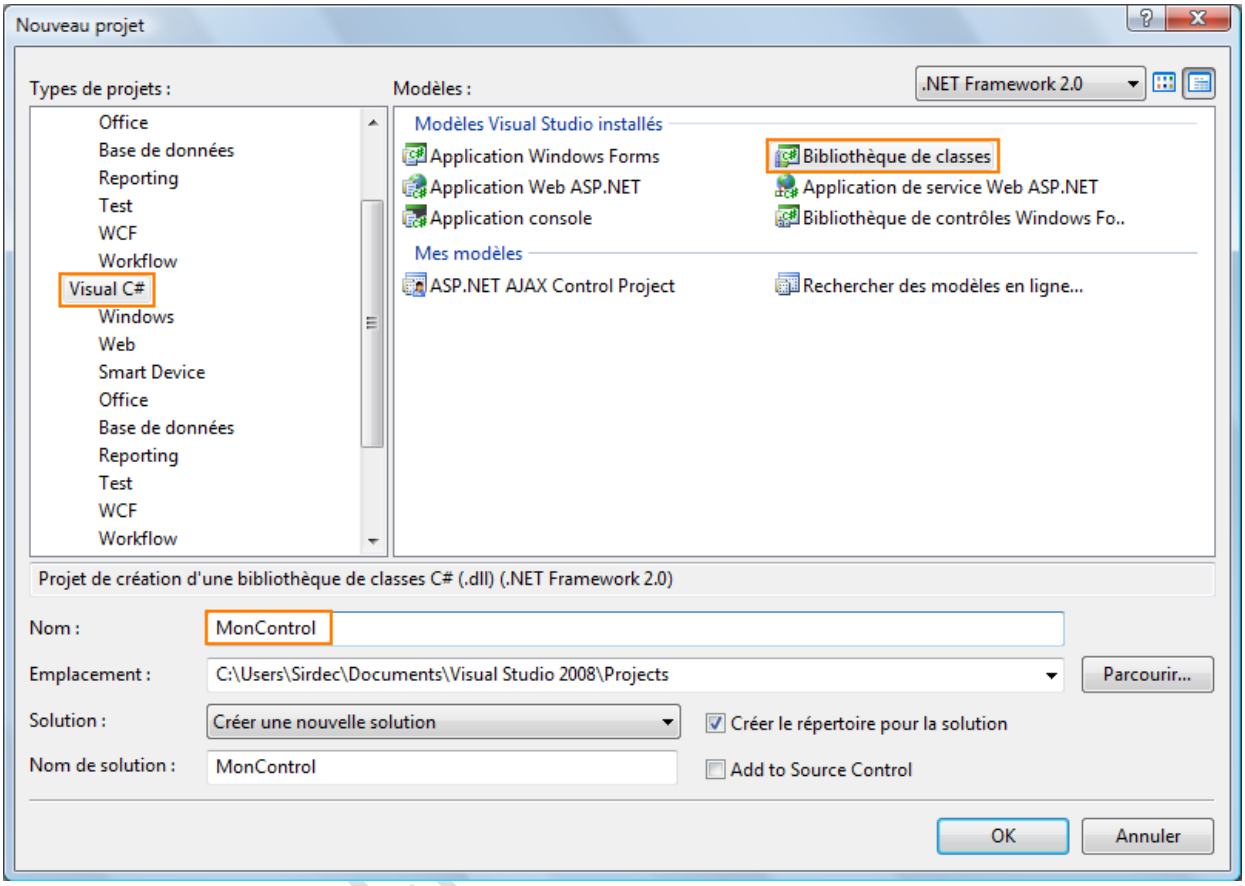
VB.NET

```
Public Class MonControl
    Inherits TextBox ' Notre classe MonControl hérite de la classe Textbox
    ' Il possèdera donc les méthodes, attributs, évennements ... de cette classe

    ' On créé deux attributs en private qui vont nous permettre de modifier la
       largeur (width) et le texte du bouton
    Private _buttonWidth As Int32
    Private _buttonText As String = "Rechercher"

    ' On créer des accesseurs / mutateurs pour pouvoir changer les valeurs des
       attributs
    Public Property ButtonWidth() As Int32
        Get
            Return _buttonWidth
        End Get
        Set(ByVal value As Int32)
            _buttonWidth = value
        End Set
    End Property

    Public Property ButtonText() As String
        Get
            Return _buttonText
        End Get
        Set(ByVal value As String)
            _buttonText = value
        End Set
    End Property

    ' On surcharge la méthode Render pour personnalisé le contrôle
    Protected Overrides Sub Render(ByVal writer As HtmlTextWriter)

        ' On utilise l'objet récupéré (l'objet HtmlTextWriter) pour créer notre
          propre rendu
        writer.Write("<div style=""display:inline;"">") ' Cet objet possède une
          méthode Write qui nous permet d'écrire dans le flux. On peut ainsi y
          mettre le Html que l'on souhaite
        ' Le @ devant permet de faire un guillemet avec "" mais aussi de pouvoir
          ainsi le caractère \ même si ici ce n'est pas utile.
        MyBase.BorderColor = System.Drawing.Color.Blue ' Le mot clef base permet
          d'accéder aux valeurs de la classe hérité et donc de changer ce qui
          touche au TextBox
        MyBase.Text = "Tapper la recherche ici" ' On a donc changé la couleur de la
          bordure mais aussi le texte du TextBox
        MyBase.Render(writer) ' Ici on applique le Render de la classe parent ... On
          aura donc l'affichage d'un contrôle TextBox
        writer.Write("  ")
        writer.Write("<input type=""button"" Value=""{0}"" ", ButtonText) ' Le {0}
          correspond au premier paramètre après la fermeture du guillemet. 1
          signifierai le second etc

        If _buttonWidth <> 0 Then ' Si on à une taille défini alors on l'applique
            writer.Write("style=""width:{0}px""", ButtonWidth)
        End If

        writer.Write("/>") ' sinon on ferme le bouton que l'on à écrit
        writer.Write("</div>") ' On ferme la div
    End Sub
End Class
```

3.1.1 Création d'une dll et ajout dans la ToolBox (boite à outils)

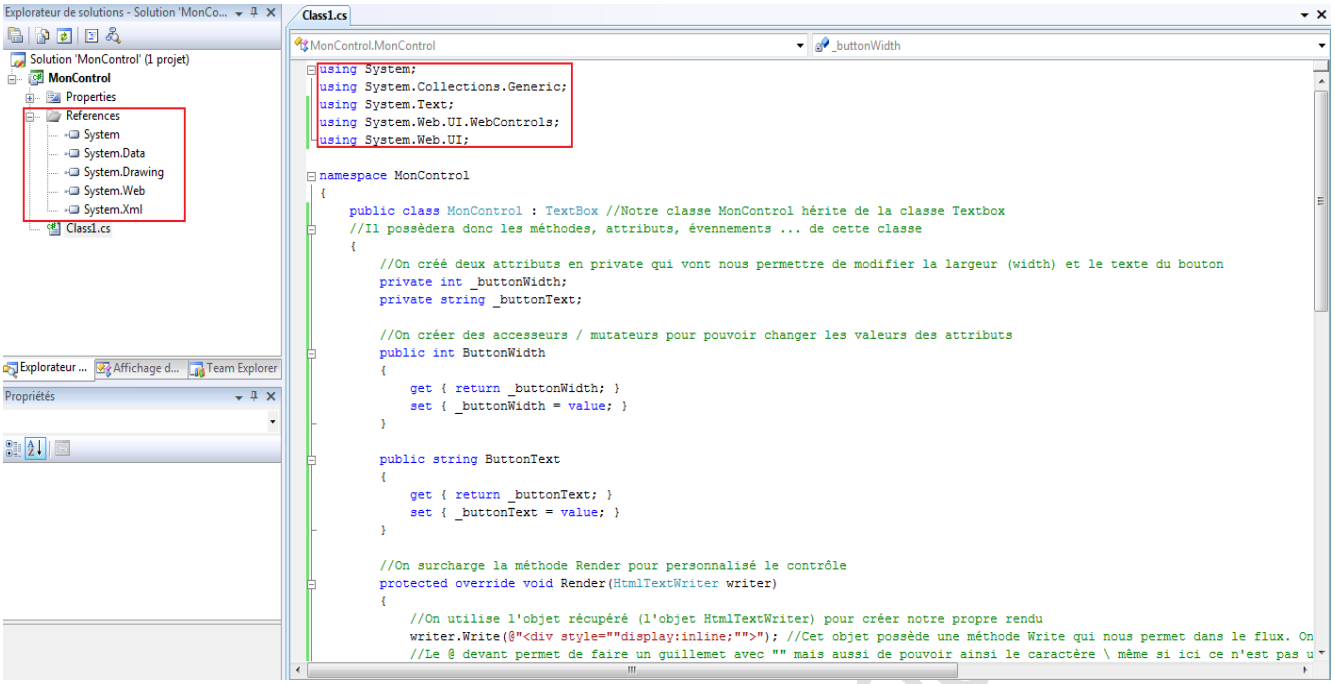
Nous allons voir comment créer une dll. Pour cela ouvrez un nouveau projet.

Sélectionnez votre langage, puis Bibliothèques de classes, le nom que vous souhaitez pour votre dll et validez.

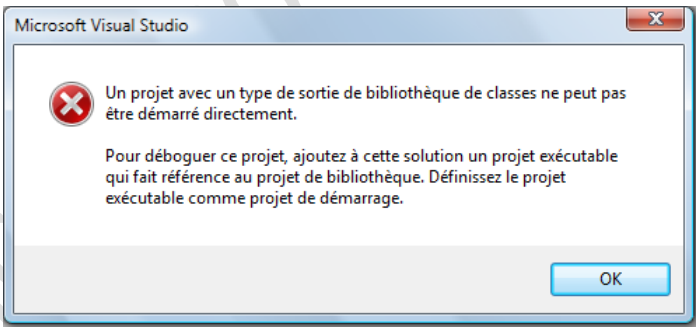


Vous arrivez dans un projet comportant une classe. Vous pouvez en créer une nouvelle ou utiliser celle-ci puisque toutes les classes seront regroupées dans une dll au final. Recopiez donc le code de la classe du contrôle que vous avez créé. Pour que votre code fonctionne il va falloir importer les bonnes assembly et rajouter des références (rappel : clic droit sur Références dans l'explorateur de solution et Ajouter une référence).

Dans le cas de la classe que nous avons créé plus haut il va falloir ajouter les références System.Web et System.Drawing. Il faudra aussi les assembly suivante : `System.Web.UI.WebControls` et `System.Web.UI`. Au final vous devriez obtenir quelque chose ressemblant à ceci (exemple sur du C#, en VB.NET ce n'est pas using mais imports, le reste du code étant celui donné plus haut) :

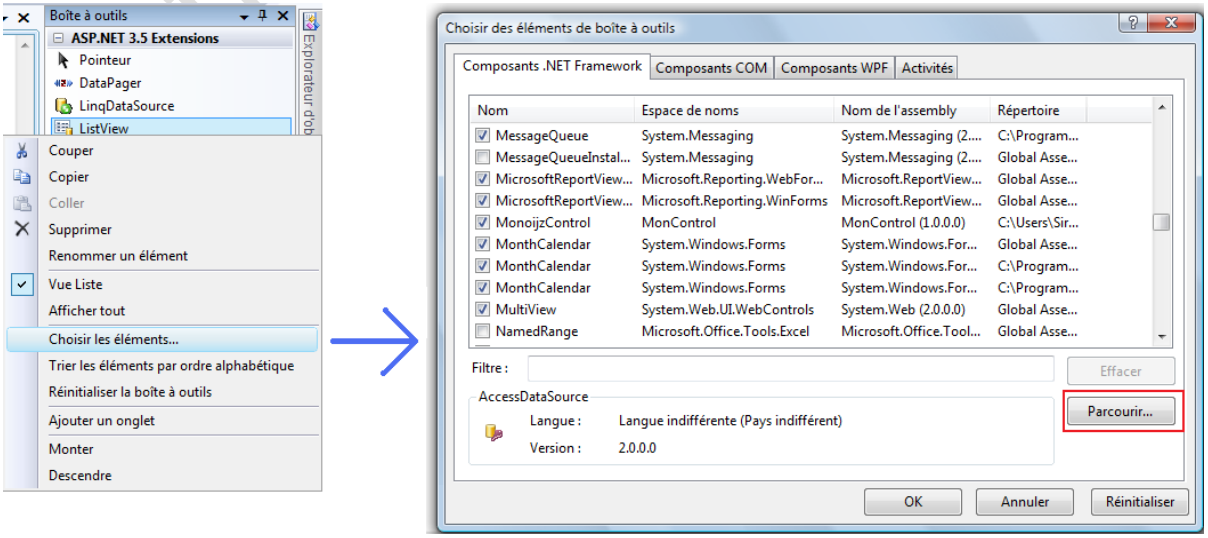


A partir de cette étape (notre contrôle est créé), il faut générer la solution (F5) pour pouvoir récupérer la dll. Lorsque l'on compile ce projet il va nous afficher un avertissement : n'en tenez pas compte, votre dll à bien été généré dans le dossier de la solution.

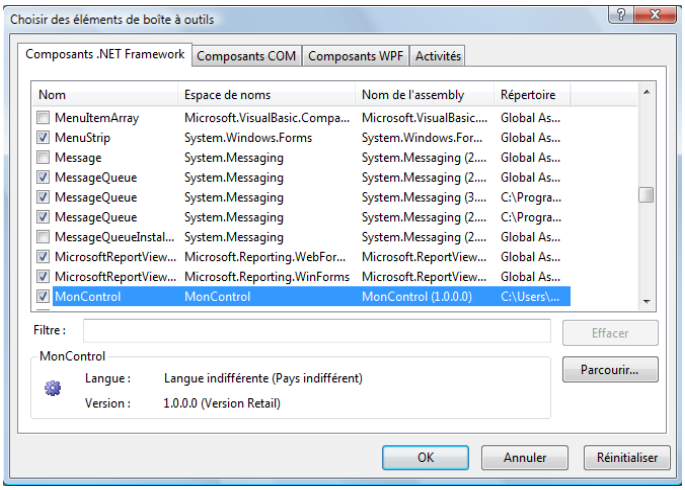


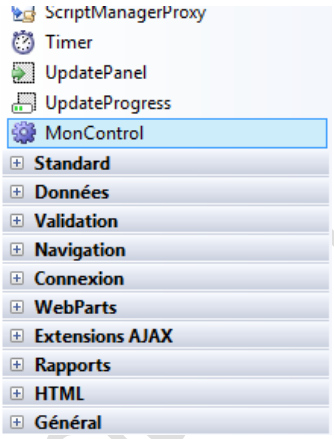
Par défaut elle se trouvera dans bin/debug (pour aller rapidement à ce dossier, faites un clic droit sur le nom du projet dans l'explorateur de solution, puis choisir "Ouvrir le dossier dans l'Explorateur Windows").

Maintenant ajoutons la à notre *ToolBox*. Pour ce faire il va falloir aller sur une page aspx (quelque soit le projet). Faire un clic droit dans la *ToolBox* et choisir "Choisir les éléments". Une fois que là fenêtre s'est ouverte choisissez Parcourir.



Ensuite vous devez aller chercher votre dll et valider. Revenez à la fenêtre du choix, vous pouvez remarquer que votre contrôle à bien été ajouté et qu'il est coché. Cliquer sur Ok : votre contrôle est maintenant ajouté à votre *ToolBox*.





On peut facilement le déplacer d'une catégorie à l'autre en le déplaçant.

Maintenant nous allons voir comment personnaliser un peu ce contrôle lorsqu'on le Drag&Drop dans une page ASPX.

3.1.2 Personnaliser l'action du contrôle lors du Drag&Drop dans la page

Cette partie va nous permettre de personnaliser ce que notre contrôle va faire lorsqu'on le rajoute dans la page. Pour l'instant si on le rajoute il possède le strict minimum. C'est-à-dire que la balise va être généré avec un *runat="server"*, un id ainsi qu'une directive *Register* en haut de la page. On pourra définir des propriétés par défaut.

Pour cela, reprenons notre projet pour créer la dll. Nous allons y ajouter une propriété `ToolboxData` qui va nous permettre de personnaliser la balise qui sera généré au *Drag&Drop*. Pour notre exemple nous allons utiliser ce code :

C#

```
[ToolboxData(@"<{0}:MonControl runat=""server"" ButtonText=""Faire une Recherche""
/>")]

public class MonControl : TextBox
```

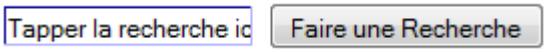
VB.NET

```
<ToolboxData("<{0}:MonControl runat=""server"" ButtonText=""Faire une Recherche""
/>")> _
Public Class LogoControl
```

Comme vous pouvez le voir la propriété doit se trouver juste après le namespace mais avant la déclaration de la classe. Ainsi lorsque nous mettons notre contrôle dans une page, nous avons une propriété en plus qui s'ajoute.

```
<cc1:MonControl ID="MonControl1" runat="server" ButtonText="Faire une
Recherche">
</cc1:MonControl>
```

A l'affichage on peut constater que le texte du bouton est "Faire une Recherche".



Il faut savoir que l'on peut aussi changer l'image bitmap (en 16x16) qui apparait dans la *toolbox* pour notre contrôle en utilisant la propriété `ToolboxBitmap` qui nécessite l'assembly *System.Drawing*.

3.2 Contrôle composé

Un contrôle composé (ou Composite Control comme nous l'avions vu), est en fait un contrôle Web personnalisé qui va hériter non pas d'un WebControl mais de la classe CompositeControl.

C#

```
public class Composite : CompositeControl // Notre contrôle hérite de la classe
CompositeControl
{
    protected override void CreateChildControls() // On surcharge cette méthode pour
ajouter nos contrôles
    {
        Panel panel = new Panel(); // On créer un Panel pour y mettre nos contrôles
        // On pourrait aussi utiliser le mot clé this pour rajouter les contrôles
dans un conteneur extérieur à la classe

        Controls.Add(panel); // On rajoute la panel dans la page

        // Création des TextBox
        TextBox txtNom = new TextBox();
        TextBox txtPrenom = new TextBox();
        TextBox txtAge = new TextBox();

        //On rempli notre Panel avec un tableau contenant un formulaire
        panel.Controls.Add(new LiteralControl(@"<table><tr><th colspan=""2"">"));
        panel.Controls.Add(new LiteralControl("Formulaire"));
        panel.Controls.Add(new LiteralControl("</th></tr><tr><td>"));
        panel.Controls.Add(new LiteralControl("Nom"));
        panel.Controls.Add(new LiteralControl("</td><td>"));
        panel.Controls.Add(txtNom);
        panel.Controls.Add(new LiteralControl("</td></tr><tr><td>"));
        panel.Controls.Add(new LiteralControl("Prenom"));
        panel.Controls.Add(new LiteralControl("</td><td>"));
        panel.Controls.Add(txtPrenom);
        panel.Controls.Add(new LiteralControl("</td></tr><tr><td>"));
        panel.Controls.Add(new LiteralControl("Age"));
        panel.Controls.Add(new LiteralControl("</td><td>"));
        panel.Controls.Add(txtAge);
        panel.Controls.Add(new LiteralControl("</td></tr></table>"));
    }
}
```

VB.NET

```
Public Class Composite
    Inherits CompositeControl ' Notre contrôle hérite de la classe CompositeControl

    Protected Overrides Sub CreateChildControls() ' On surcharge cette méthode pour
                                                    ajouter nos contrôles
        Dim panel As Panel = New Panel() ' On créer un Panel pour y mettre nos
                                           contrôles
        ' On pourrait aussi utiliser le mot clé this pour rajouter les contrôles
            dans un conteneur extérieur à la classe

        Controls.Add(panel) ' On rajoute la panel dans la page

        ' Création des TextBox
        Dim txtNom As TextBox = New TextBox()
        Dim txtPrenom As TextBox = New TextBox()
        Dim txtAge As TextBox = New TextBox()

        ' On rempli notre Panel avec un tableau contenant un formulaire
        panel.Controls.Add(New LiteralControl("<table><tr><th colspan=""2"">"))
        panel.Controls.Add(New LiteralControl("Formulaire"))
        panel.Controls.Add(New LiteralControl("</th></tr><tr><td>"))
        panel.Controls.Add(New LiteralControl("Nom"))
        panel.Controls.Add(New LiteralControl("</td><td>"))
        panel.Controls.Add(txtNom)
        panel.Controls.Add(New LiteralControl("</td></tr><tr><td>"))
        panel.Controls.Add(New LiteralControl("Prenom"))
        panel.Controls.Add(New LiteralControl("</td><td>"))
        panel.Controls.Add(txtPrenom)
        panel.Controls.Add(New LiteralControl("</td></tr><tr><td>"))
        panel.Controls.Add(New LiteralControl("Age"))
        panel.Controls.Add(New LiteralControl("</td><td>"))
        panel.Controls.Add(txtAge)
        panel.Controls.Add(New LiteralControl("</td></tr></table>"))
    End Sub
End Class
```

On l'ajoute à la page ASPX depuis le code behind avec :

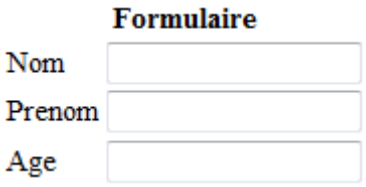
C#

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load()
    {
       Composite compo = new Composite();
       form1.Controls.Add(compo);
    }
}
```

VB.NET

```
Partial Public Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load() Handles Me.Load
        Dim compo As New Composite()
        form1.Controls.Add(compo)
    End Sub

End Class
```

Après génération du projet on obtient ceci :



4 Conclusion

Nous avons vu dans ce chapitre comment créer notre propre contrôle. Voici les points importants dont il faut se souvenir :

- Le *User Control* est un contrôle contenu dans une page *ascx,* possédant aussi du CodeBehind.
- Les *Custom Web Control* sont des contrôles contenus dans des classes (héritant d'un contrôle Web déjà existant ou de la classe WebControl) ainsi d'ailleurs que les *Composite Control* (héritant de la classe CompositeControl).
- Avec ces deux derniers on peut créer des dll et ainsi les ajouter à la *ToolBox*.
- Il faut ajouter la directive *Register* en haut de la page *ASPX* même si celle-ci est générée automatiquement, elle n'en est pas moins importante puisque sans elle le contrôle ne pourra pas s'ajouter.

L'ASP.NET nous permet de créer nos propres contrôles. C'est très utile dans le cas où nous avons besoin de placer dans nos pages plusieurs fois la même chose. On peut par exemple créer un contrôle qui permettra d'ajouter toute la partie Login (exemple : un tableau avec deux *Label*, deux *TextBox* et un *Button*). On aura un gain de temps du fait que l'on ne devra pas à chaque fois recréer la partie de Login.