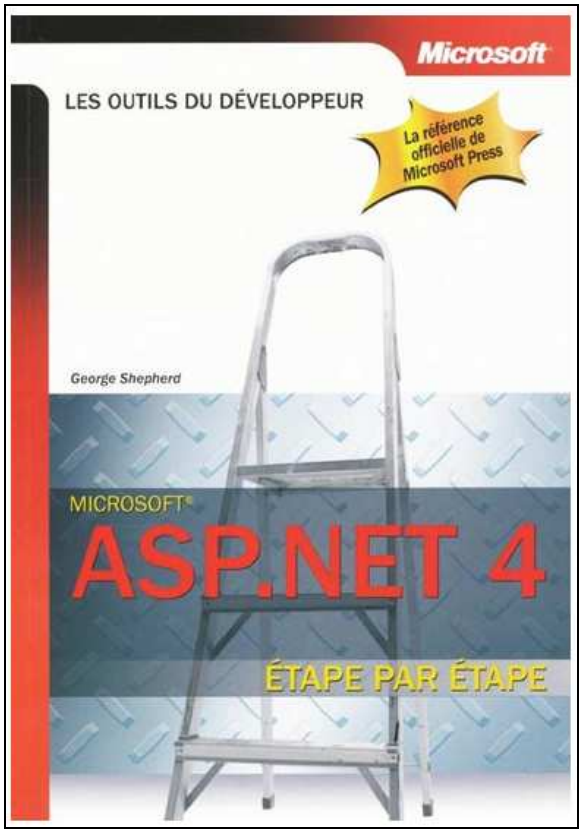
Introduction à ASP.NET
Partie 1 (concepts de base)

Auteurs : Ph. Lacomme (placomme@isima.fr) et R. Phan (phan@isima.fr)
Date : juin 2011
Avertissement : Les exemples proposés dans ce tutorial viennent en partie du livre de George Shepherd. Nous avons refait certains exemples du livre, nous les avons complétées et nous espérons avoir amélioré les explications (parfois très rapides) qui sont données dans le livre.

Ceci n'est pas un cours.

Nous vous conseillons le livre suivant :



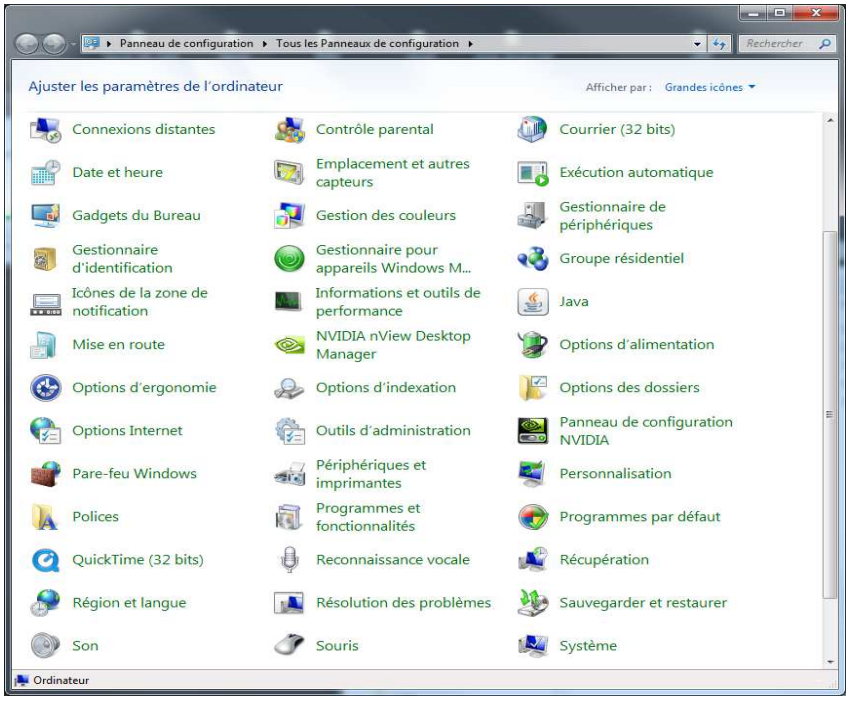
George Shepherd
Editeur : Microsoft Press, 2010 - 606 pages
ISBN : 2100547429

Préliminaires

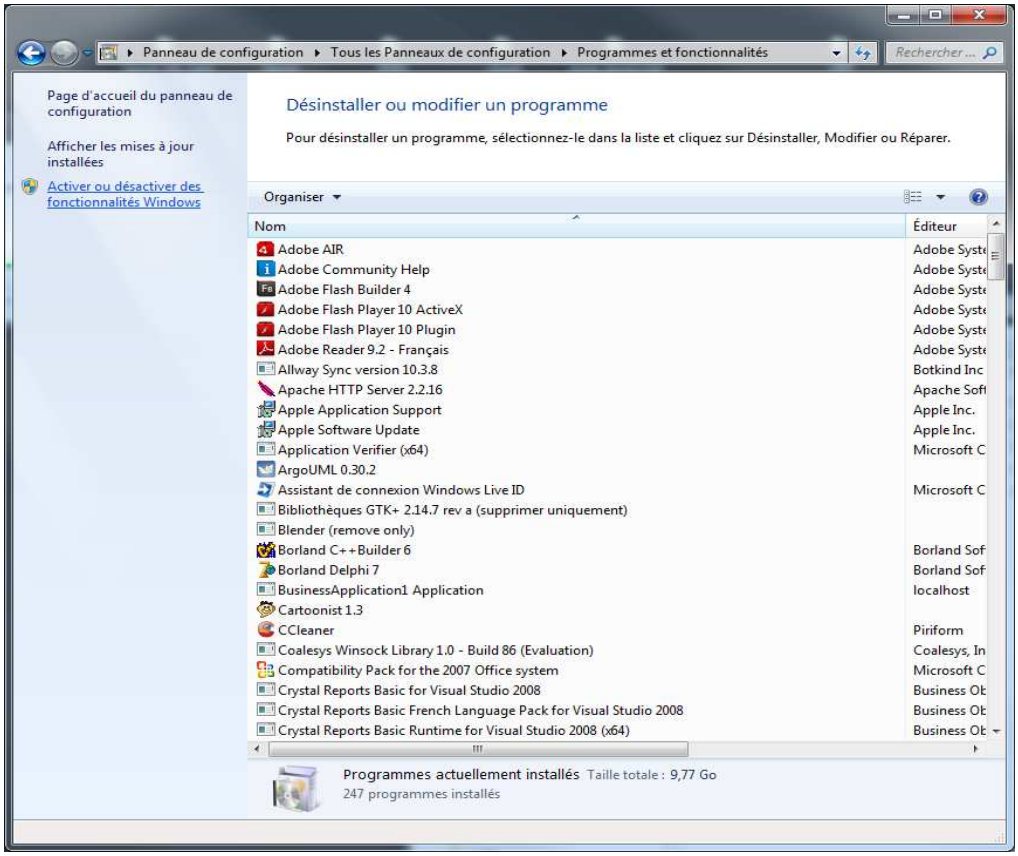
Activer IIS

Aller dans le menu **Demarrer** et choisir **Panneau de Configuration.**

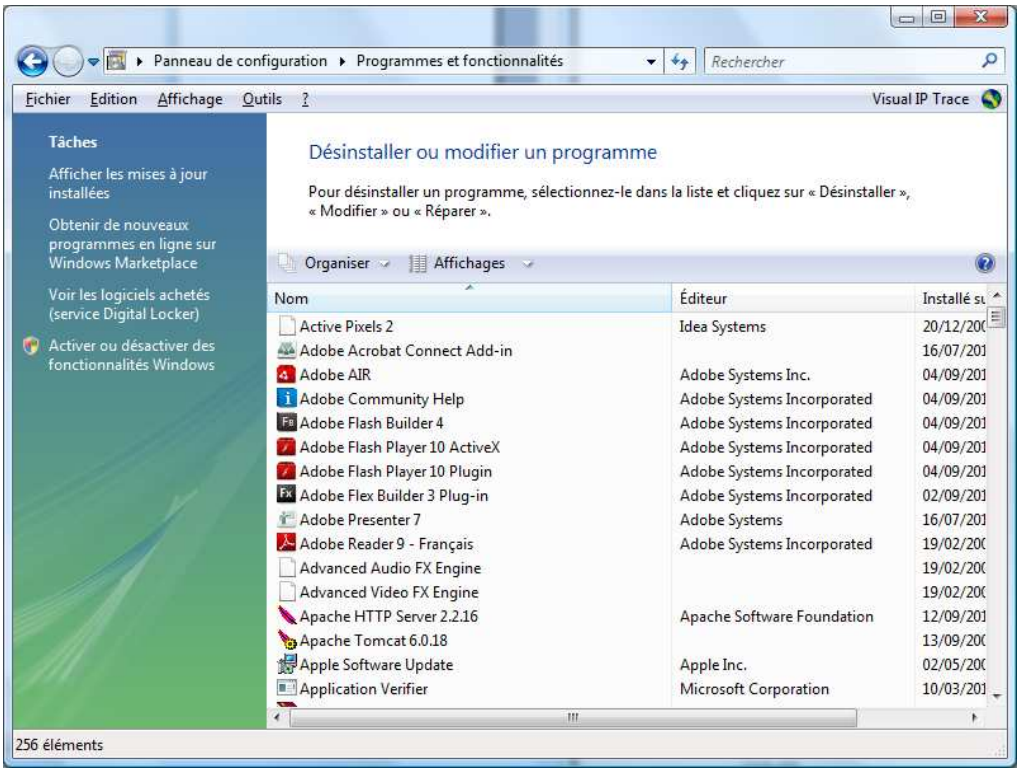
Sous Windows 7 cela donne un menu comme celui-ci :



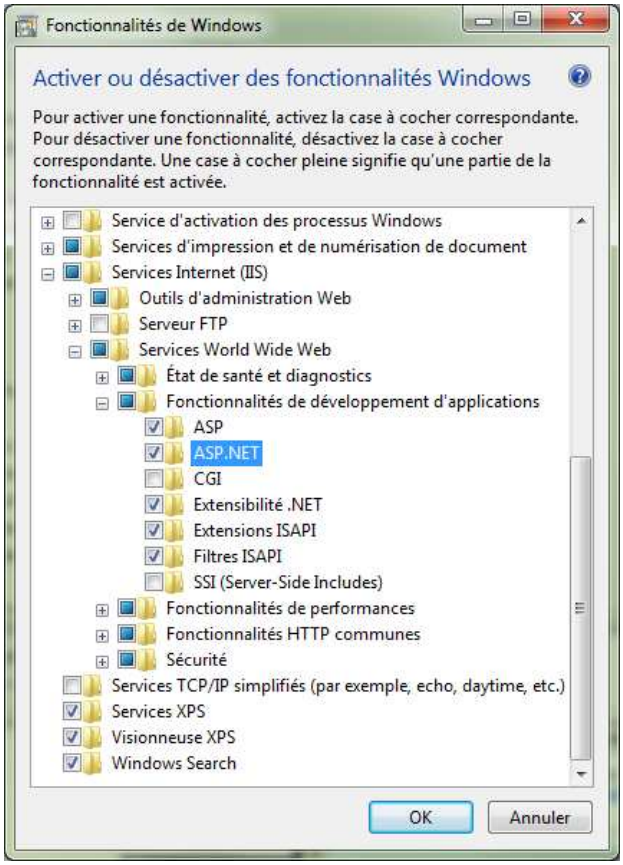
Sous Windows 7, choisir **Activer des fonctionnalités windows**.



Sous Windows Vista, choisir **Programme et Fonctionnalités**.



Choisir le menu **Activer et désactiver des fonctionnalités de Windows.**
Cocher service IIS et penser à activer la fonctionnalité ASP et ASP.NET

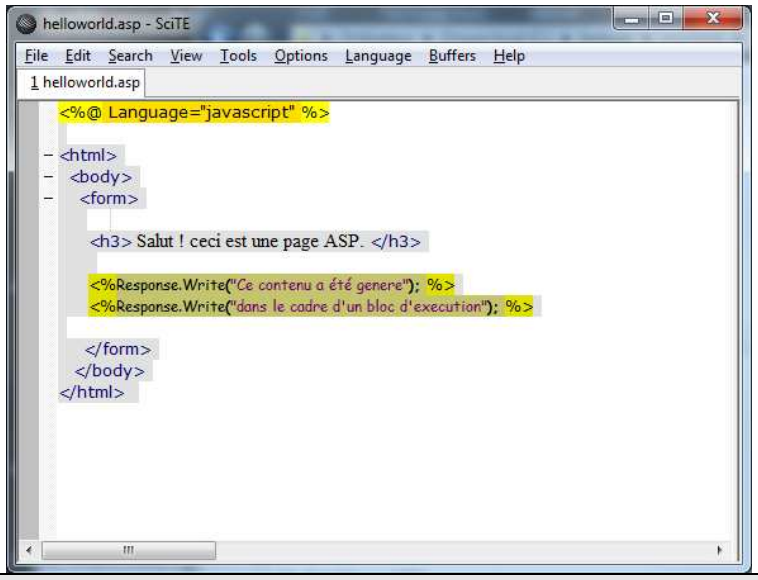


Cela créera un répertoire **c:\Inetpub\wwwroot\ dans lequel vous aller mettre vos page .asp. Cela permet de simuler un serveur acceptant les fichier ASP.**

1. Mes premières pages ASP simples

1.1. Exemple

Ouvrir un éditeur de texte quelconque et taper le texte suivant !



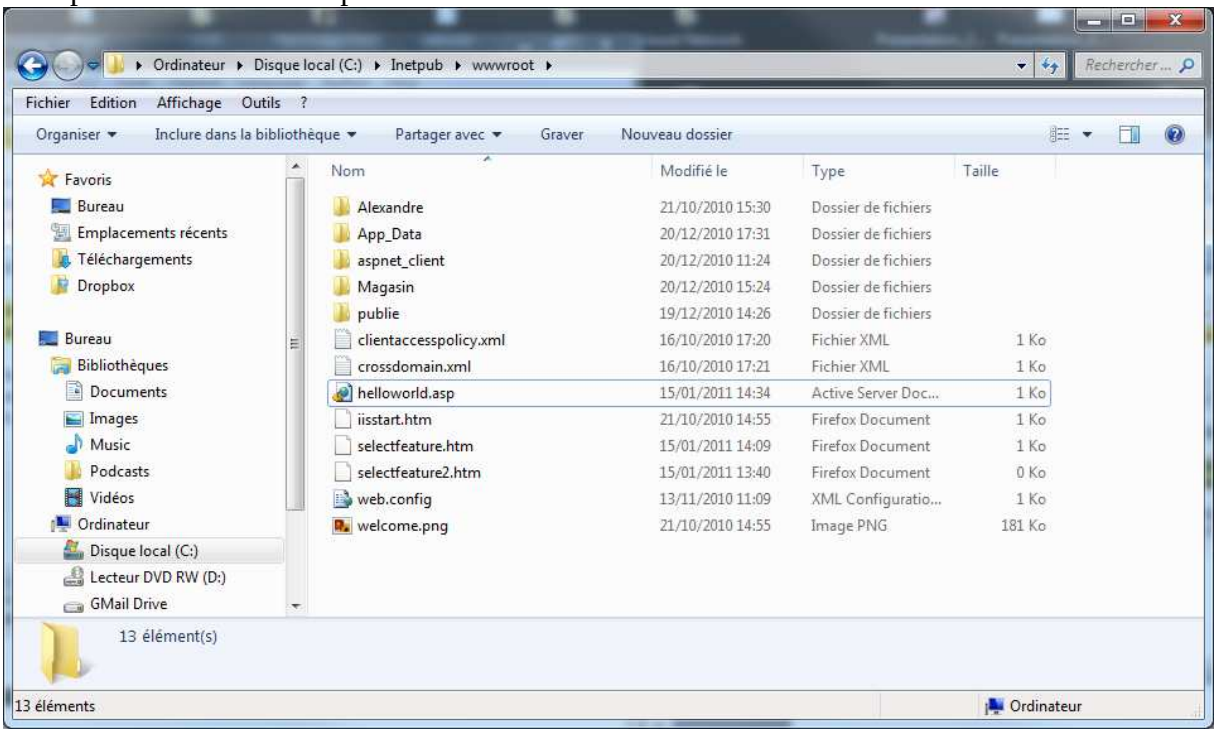

```
<%@ Language="javascript" %>
```

```
<html>
  <body>
    <form>

      <h3> Salut ! ceci est une page ASP. </h3>

      <%Response.Write("Ce contenu a été genere"); %>
      <%Response.Write("dans le cadre d'un bloc d'execution"); %>

     </form>
   </body>
</html>
```

Nommez cette page : **helloworld.asp**

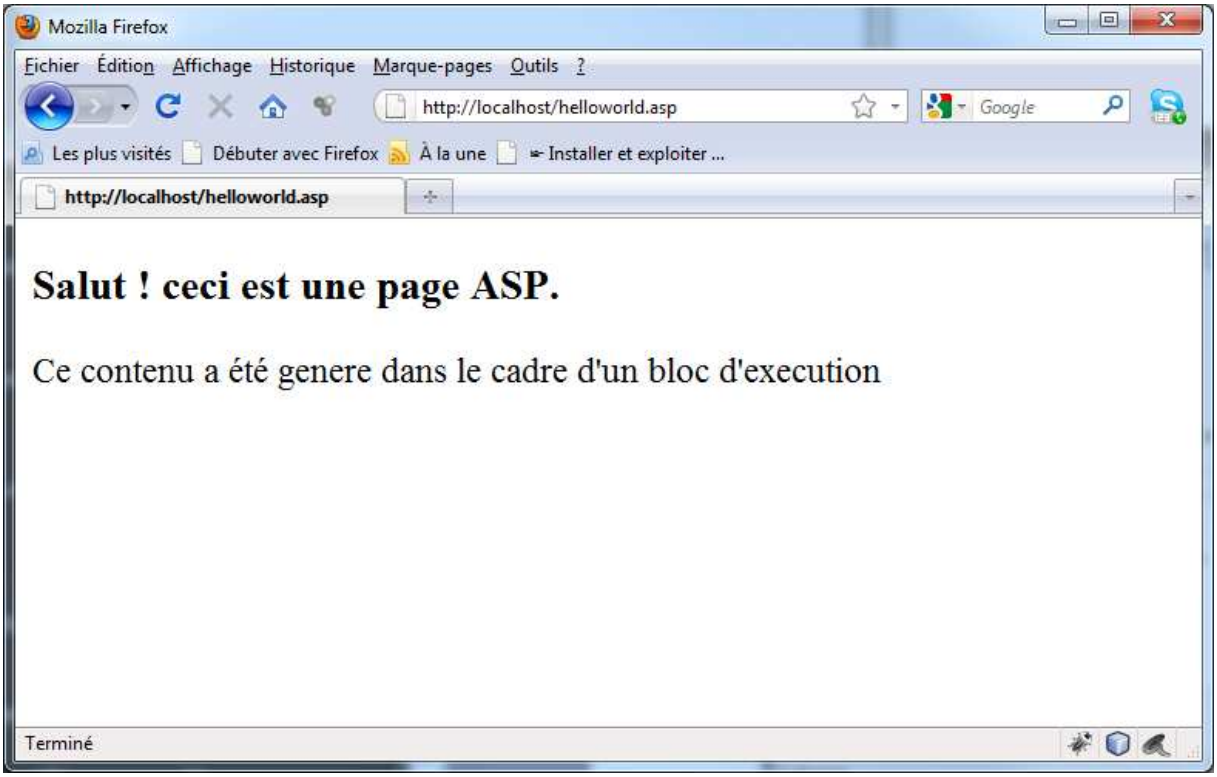
Copiez cette page dans le répertoire : **c:\Inetpub\wwwroot**

Le répertoire doit alors se présenter comme suit :



Dans un navigateur tapez l'adresse : **http://localhost/helloworld.asp**

Vous devez obtenir :

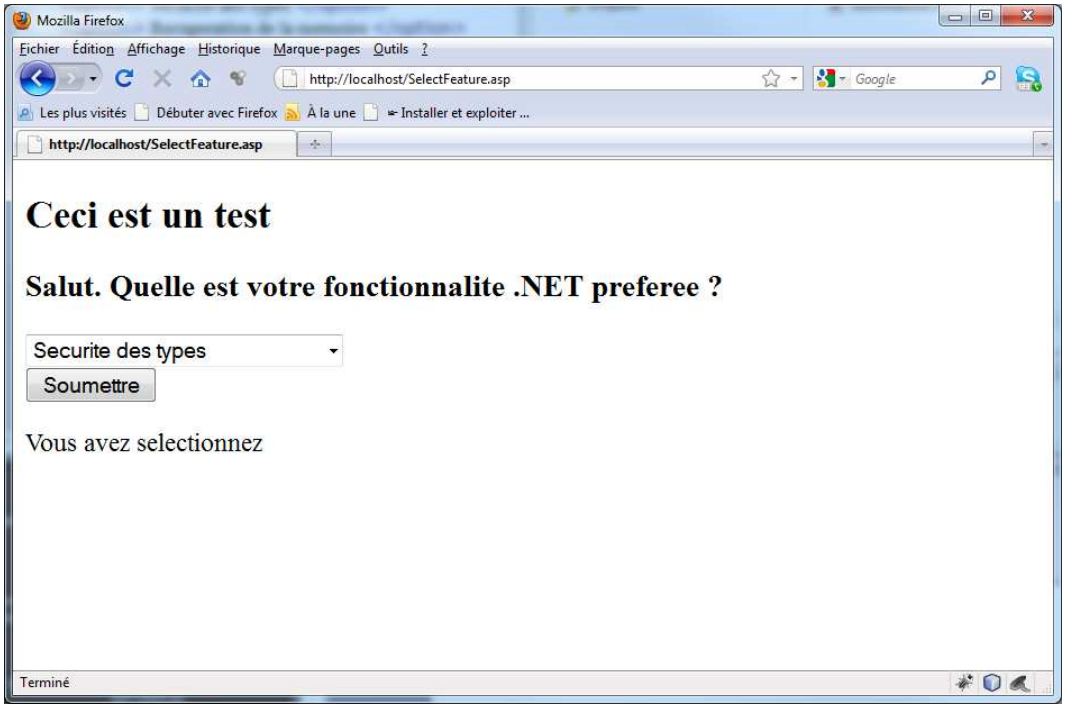


1.2. Exemple

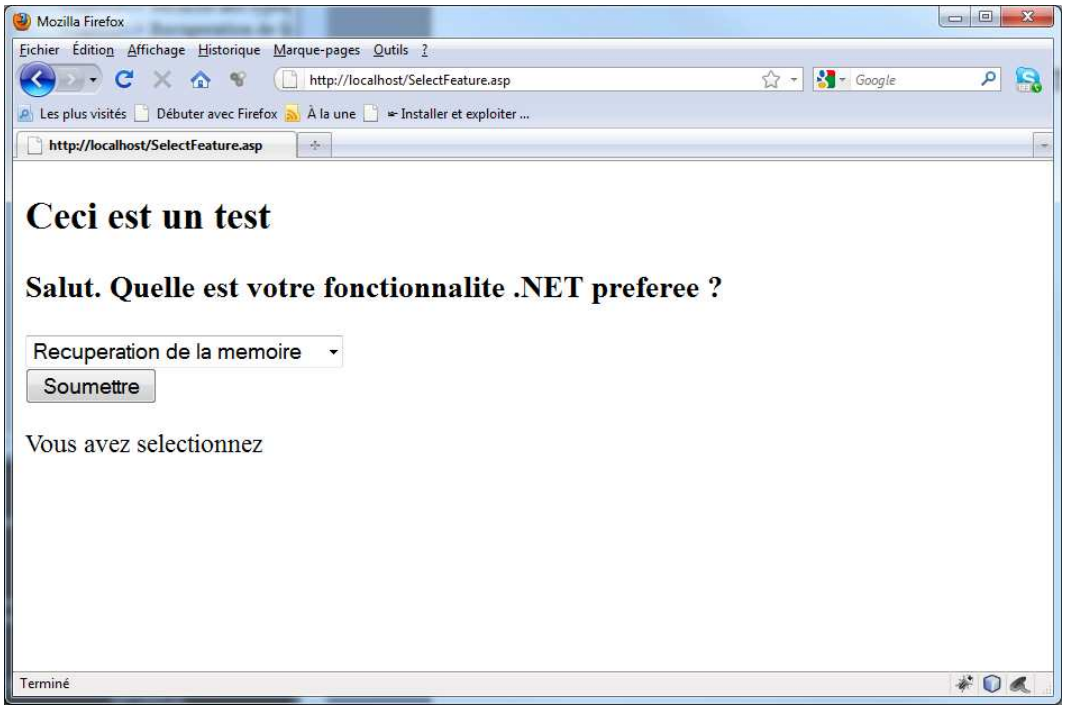
Créez la page suivante et sauvegardez la sous le nom de « SelectFeature.asp » dans le répertoire **c:\Inetpub\wwwroot** :

```
<%@ Language="javascript" %>
<html>
	<body>
		<form>

		<h2> Ceci est un test </h2>
		<h3> Salut. Quelle est votre fonctionnalite .NET preferee ? </h3>

		<select name='fonctionnalite'>
			<option> Securite des types </option>
			<option> Recuperation de la memoire </option>
			<option> Syntaxes multiples </option>
			<option> Securite des acces au code </option>
			<option> Modele de thead plus simples </option>
			<option> Purgatoire des versions </option>
		</select>

		<br>
		<input type=submit name='Soumettre' value='Soumettre'> </input>

		<p>
			Vous avez selectionnez <%=Request("fonctionnalite") %>
		</p>

		</form>
	</body>
</html>
```

Dans un navigateur tapez l'adresse : **http://localhost/SelectFeature.asp**

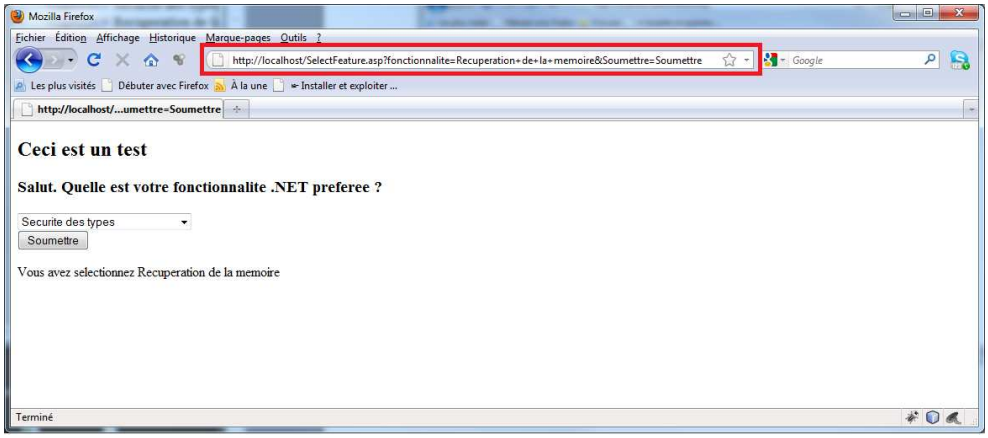


Faites un choix dans la liste déroulante :



Puis un clic sur le bouton **Soumettre**.

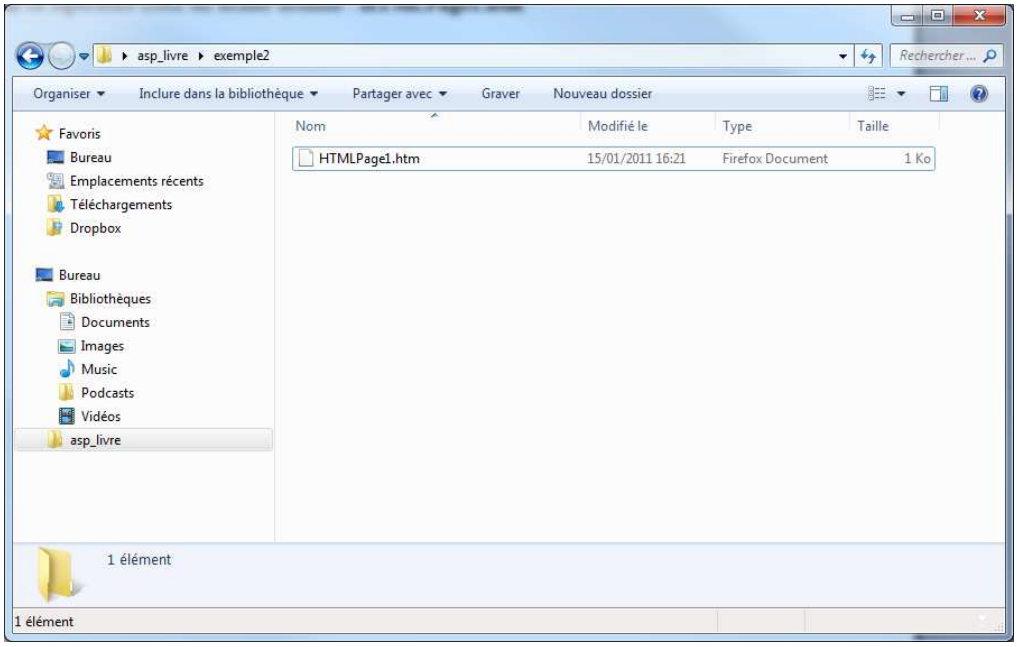
Examiner alors l'url en haut de la page :



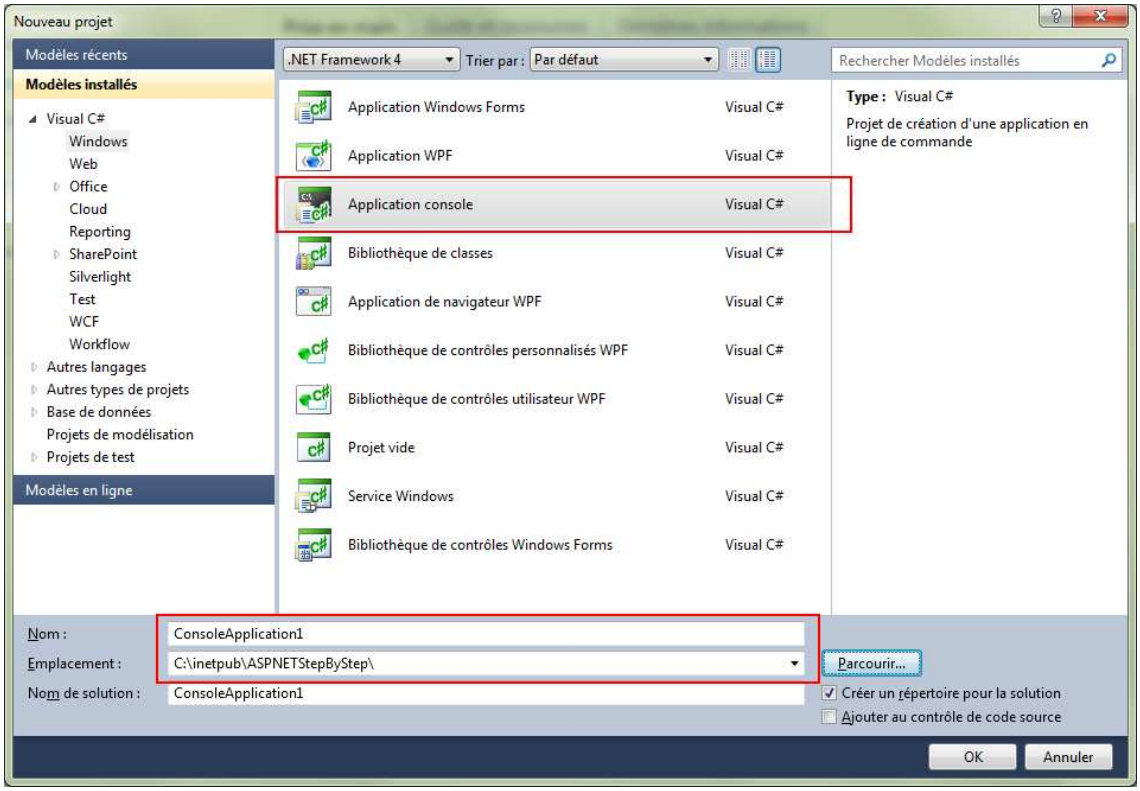
2. Ma première page ASP.NET : programme 1

Créer un répertoire **c:\Inetpub\wwwroot\ASPNETStepByStep**

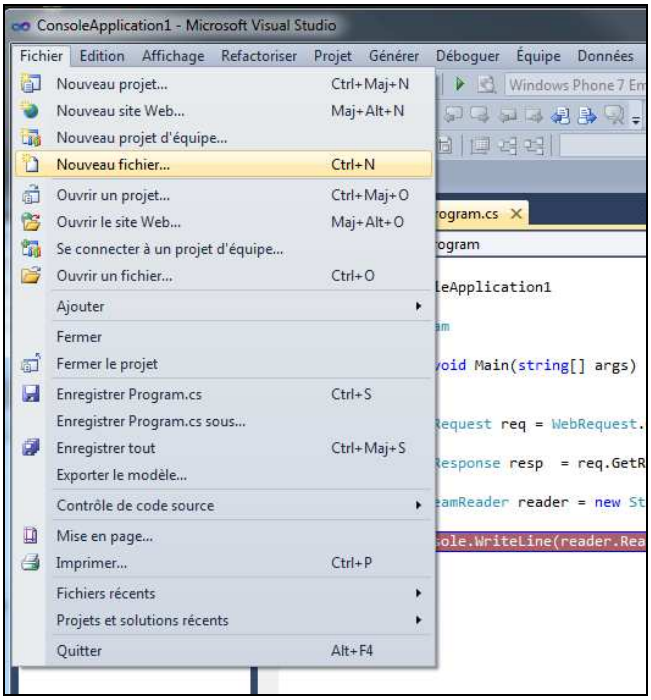
Dans ce répertoire créez un fichier nommé : **HTMLPage1.htm**



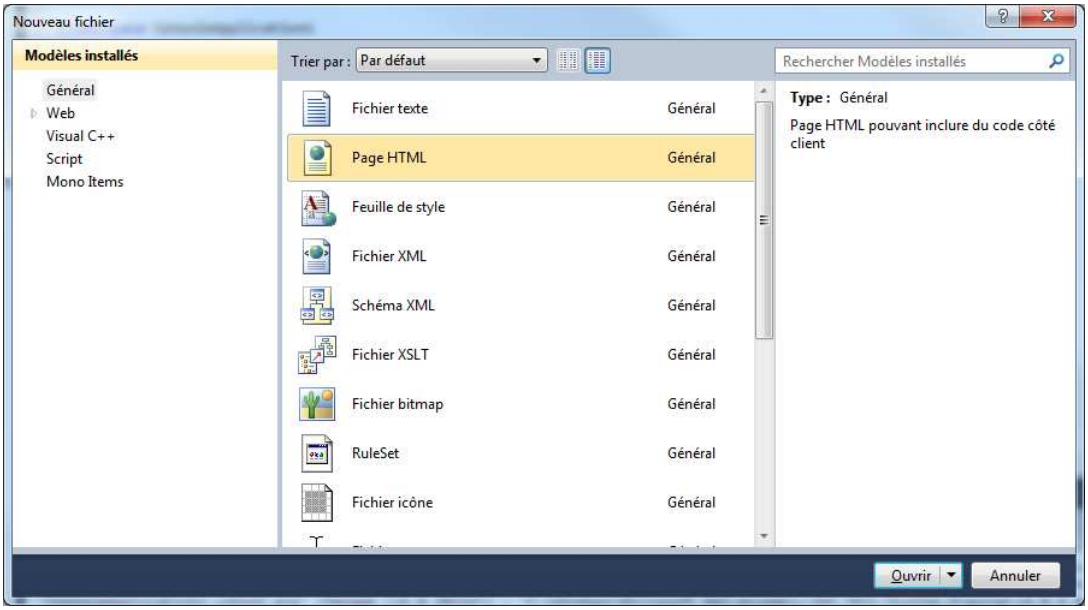
Démarrez **Visual Studio.** Créer un nouveau projet de type *Application console*. Mettre comme dossier du projet celui qu'on vient de créer : **c:\Inetpub\wwwroot\ASPNETStepByStep\.**



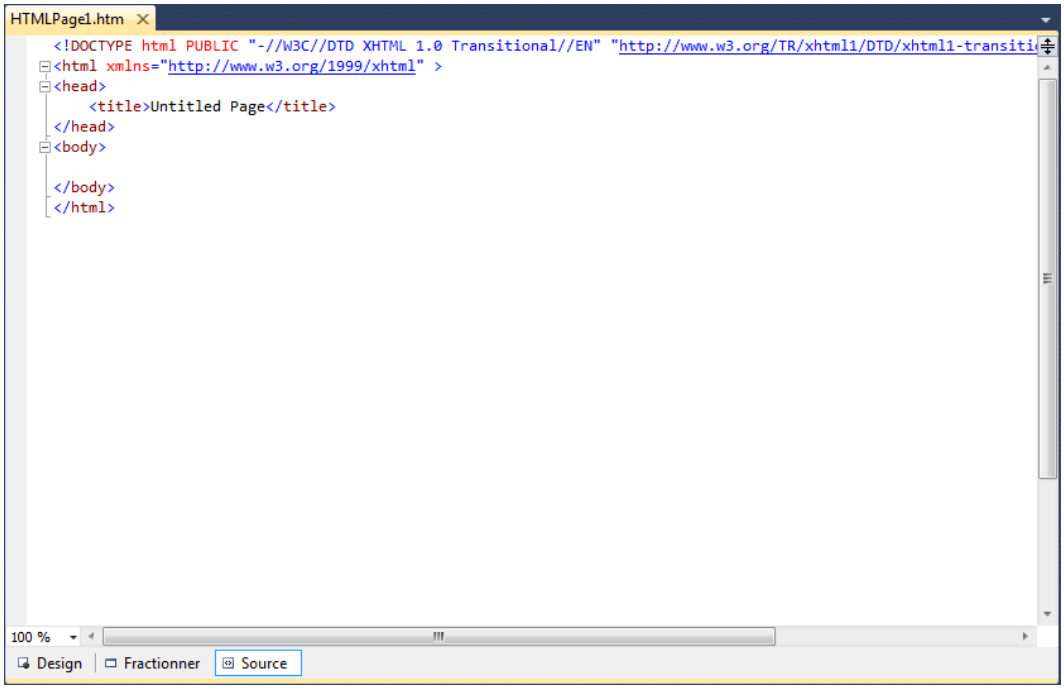
Créer un **Nouveau Fichier**.



Choisir ensuite **Page HTML**.



Ce qui devrait donner :

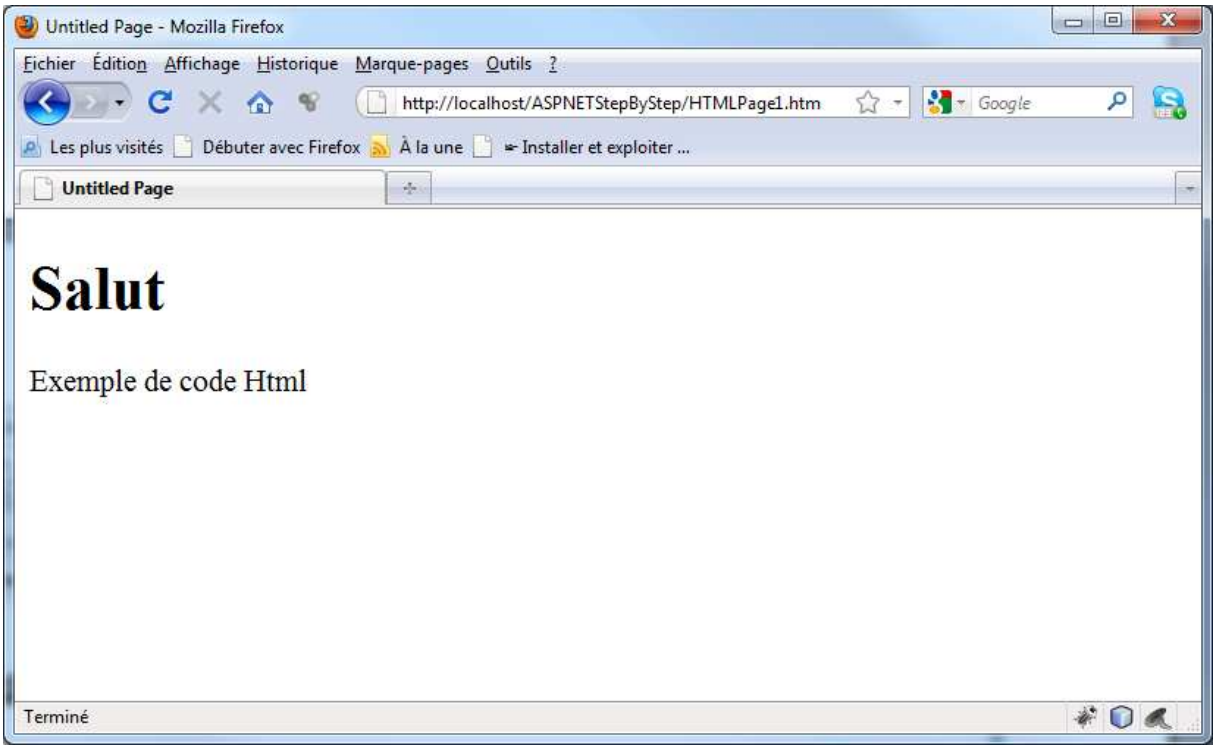


Modifier le contenu de la page comme ci-dessous :

```
<!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>Untitled Page</title>
</head>
<body>

  <h1>Salut</h1>

  Exemple de code Html

</body>
</html>
```

Sauvegarder dans le répertoire « **c:\Inetpub\wwwroot\ASPNETStepByStep** » et accéder à la page http://localhost/ASPNETStepByStep/HTMLPage1.htm

Le résultat devrait être comme suit :

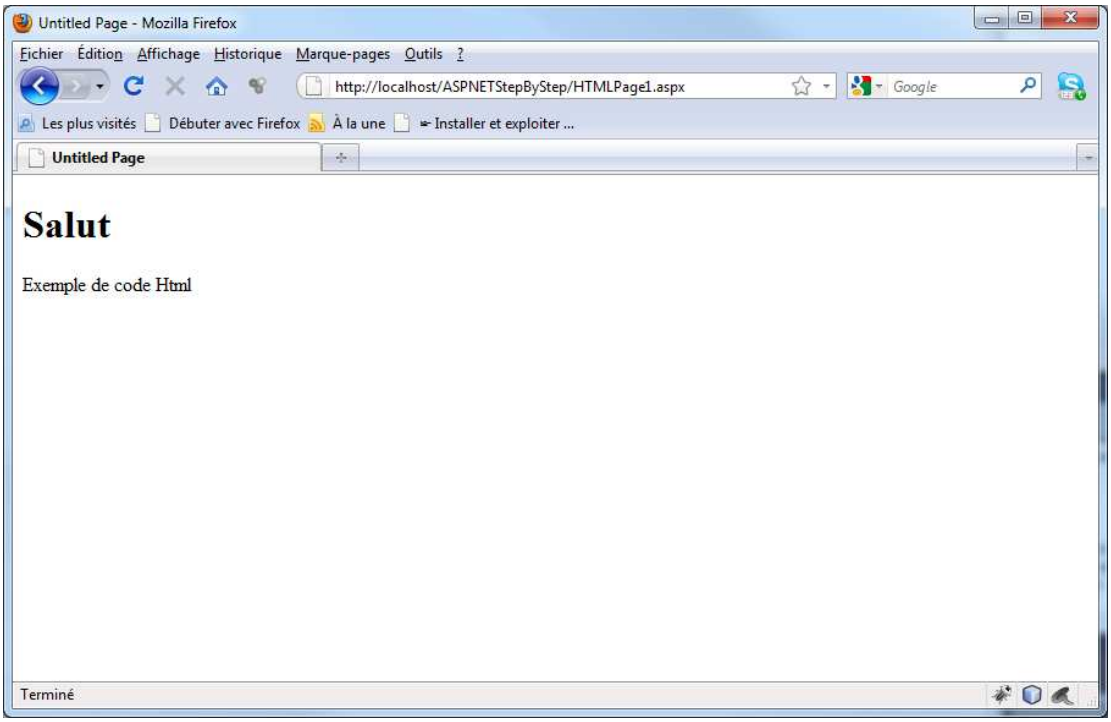


Modifier le fichier :

- son nom devient : **HTMLPage1.aspx**
- son contenu est modifié comme suit :

```
<%@ Page Language="C#" %>

<!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>Untitled Page</title>
</head>
<body>

  <h1>Salut</h1>

  Exemple de code Html

</body>
</html>
```

Visualiser ensuite la page : http://localhost/ASPNETStepByStep/HTMLPage1.aspx



3. Mélanger du HTML et du code exécutable ASP.NET : programme 2

Créer un fichier nommé : **HTMLPage2.aspx** et contenant le code suivant :

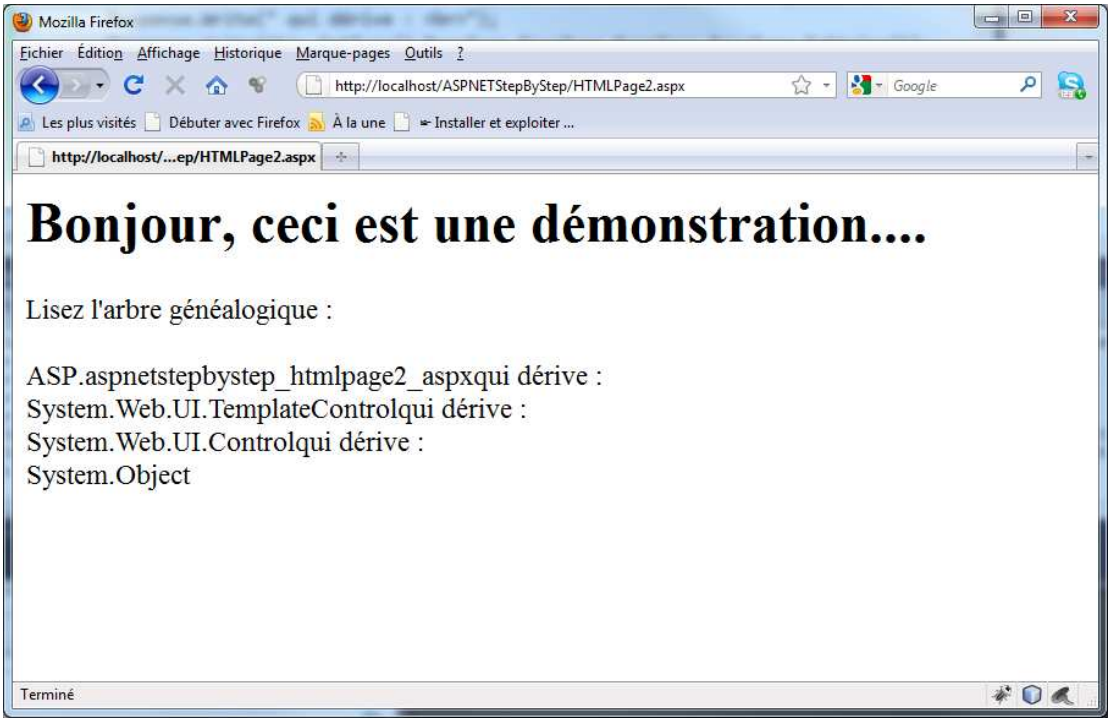
```
<%@ Page Language="C#" %>
<html>
  <body>
     <h1> Bonjour, ceci est une démonstration....</h1>
     <%
       Response.Write("Lisez l'arbre généalogique : <br> <br>");
       Response.Write(this.GetType().ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.BaseType.ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.BaseType.BaseType.ToString());
     %>
  </body>
</html>
```

Le code qui se trouve entre les deux balises <% et %> est du code exécuté du coté serveur.

Enregistrez le fichier dans C:\Inetpub\wwwroot\ASPNETStepByStep\

Accédez ensuite à l'adresse : http://localhost/ASPNETStepByStep/HTMLPage2.aspx

Le résultat devrait être :

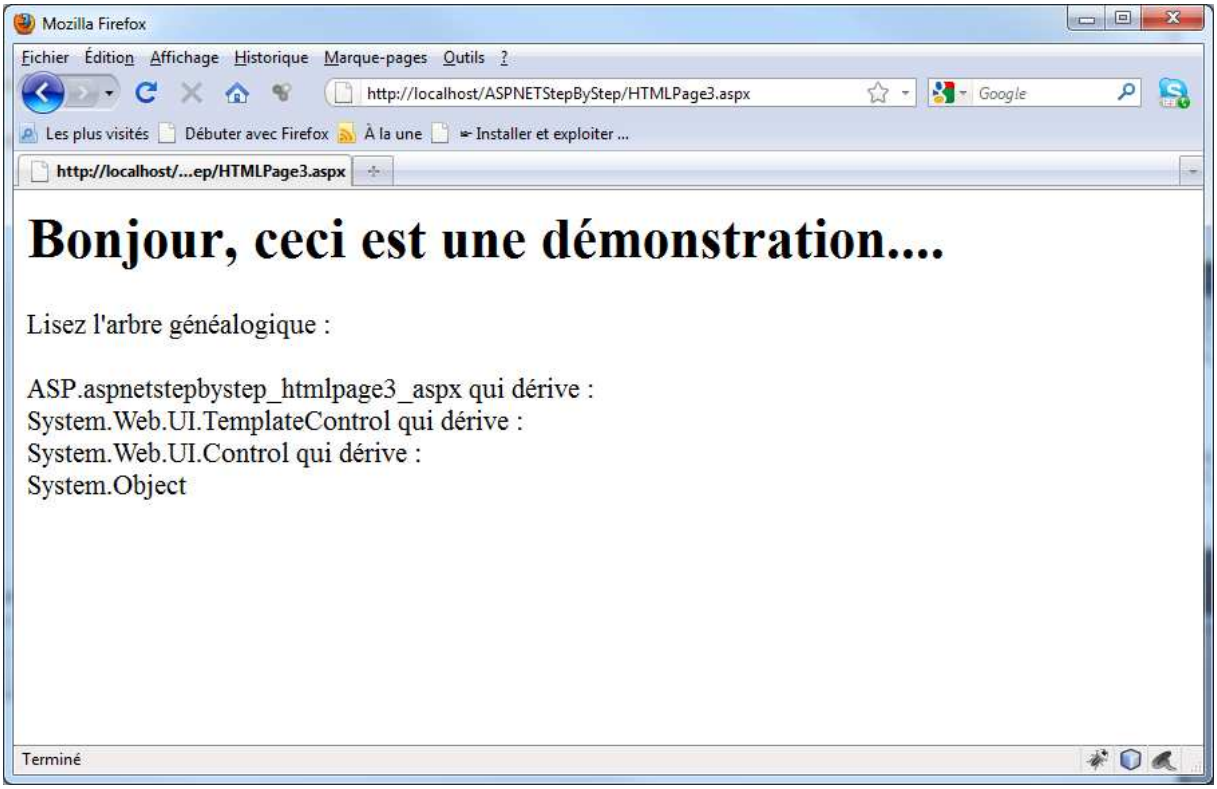


Il est possible de faire la même chose en utilisant un bloc de script. Modifiez le nom du fichier en HTMLPage3.aspx déclarez une procésure nommée ShowLineage comme indiquée ci-dessous :

```
<%@ Page Language="C#" %>
<script runat="server">

  void ShowLineage()
  {
       Response.Write("Lisez l'arbre généalogique : <br> <br>");
       Response.Write(this.GetType().ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.BaseType.ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.BaseType.BaseType.ToString());
  }

</script>
 <html>
  <body>
     <h1> Bonjour, ceci est une démonstration....</h1>
     <%
        ShowLineage();
     %>
  </body>

</html>
```

Accédédez ensuite à la page HTMLPage3.aspx :
http://localhost/ASPNETStepByStep/HTMLPage3.aspx

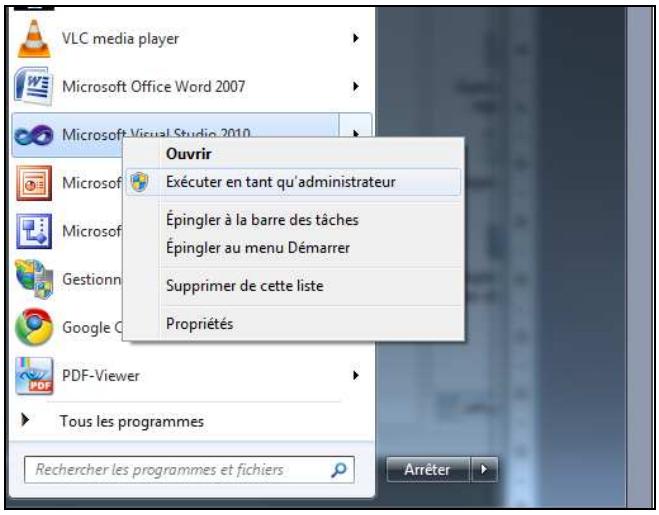
Le résultat devrait être rigoureusement identique au précédent.



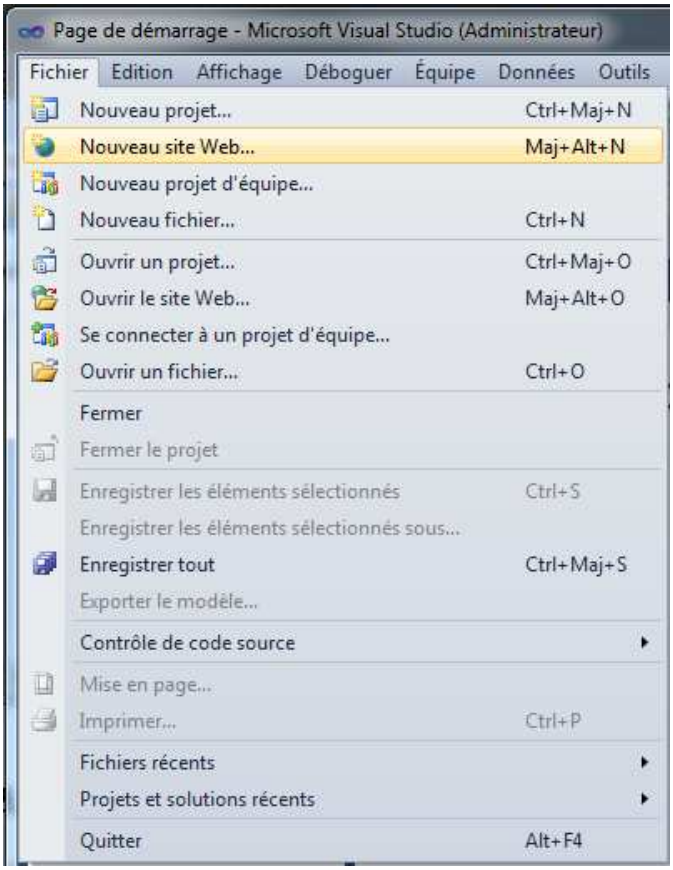
3. Premier « vrai » programme ASP.NET

3.1. Créer une application ASP.NET.

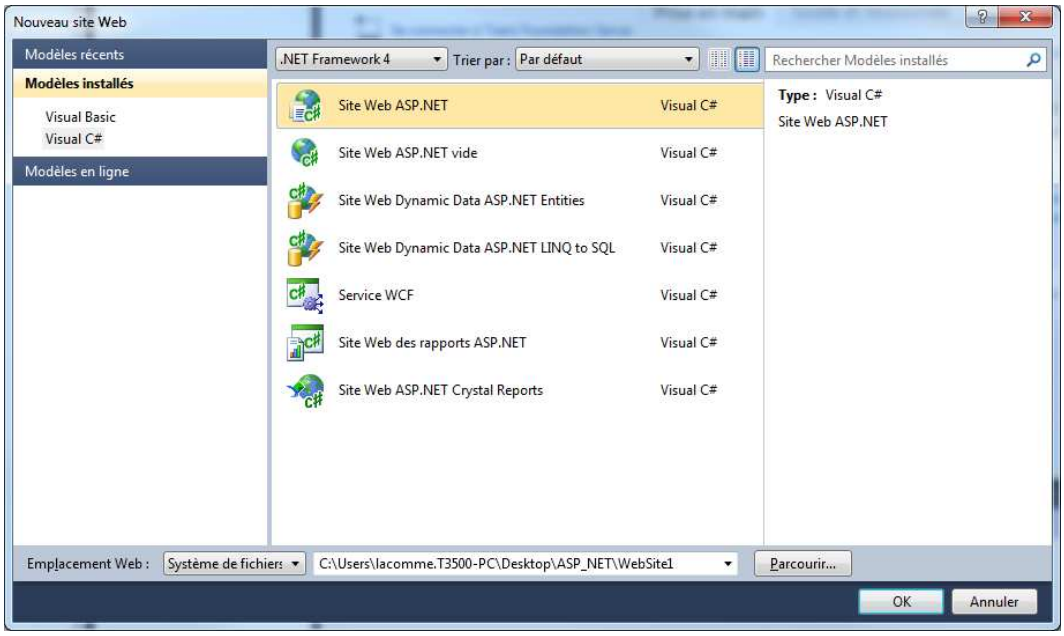
Démarrer Visual Studio en mode administrateur. Pour cela faites un click droit sur l'exécutable et choisir **Exécuter en tant qu'administrateur**.



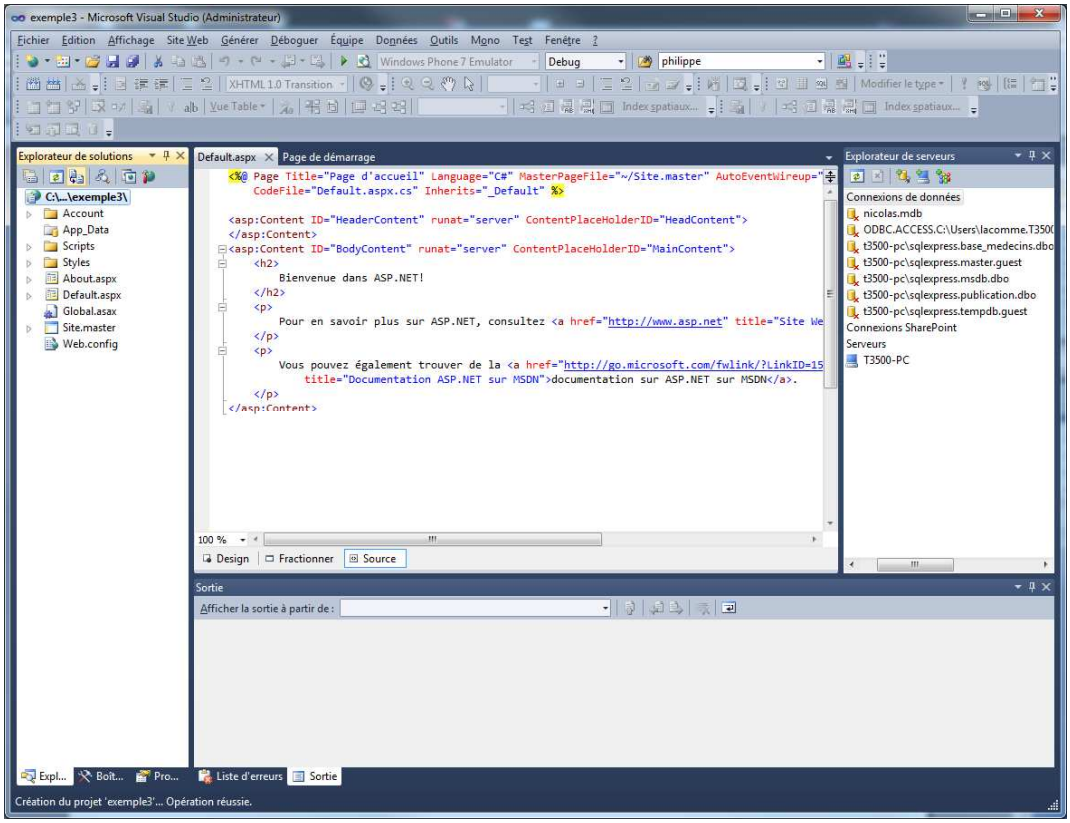
Faire ensuite **Fichier / Nouveau Site Web**



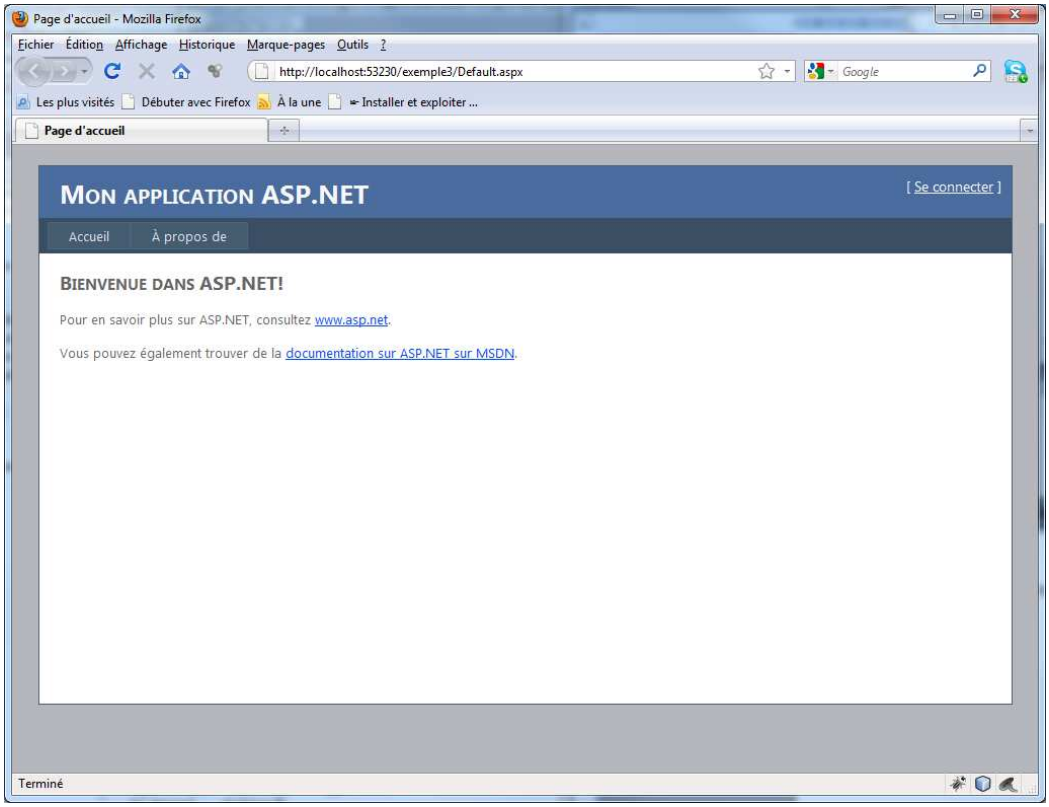
Choisir ensuite **Site Web ASP.NET**



La création du projet donne une configuration similaire à celle-ci :



Vérifier que le projet compiler et s'exécute. Cela devrait donner :



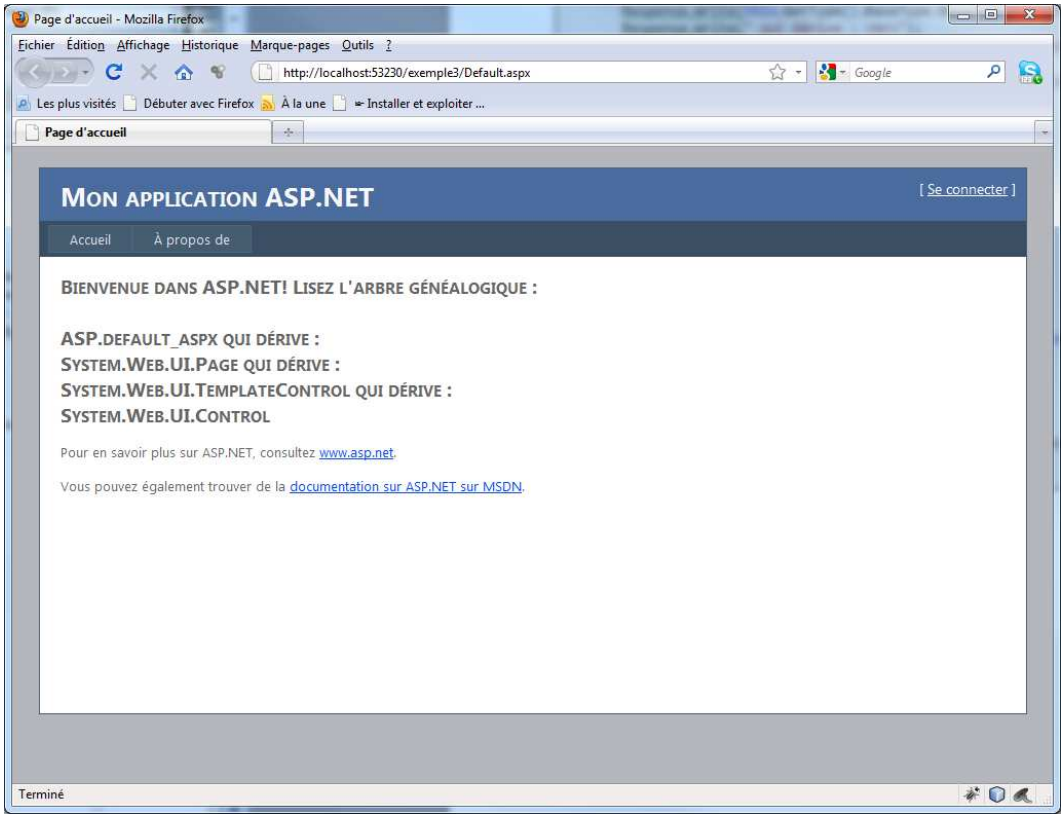
Notons le port 53230 qui correspond au serveur de développement :
http://localhost:53230/exemple3/Default.aspx

Reprenez la fonction **ShowLineage()** vue précédemment et modifiez le code comme suit :

```
<%@ Page Title="Page d'accueil" Language="C#" MasterPageFile="~/Site.master"
AutoEventWireup="true"
    CodeFile="Default.aspx.cs" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">

<script runat="server">
  void ShowLineage()
  {
       Response.Write("Lisez l'arbre généalogique : <br> <br>");
       Response.Write(this.GetType().ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.BaseType.ToString());
       Response.Write(" qui dérive : <br>");
       Response.Write(this.GetType().BaseType.BaseType.BaseType.BaseType.ToString());
  }

</script>

    <h2>
        Bienvenue dans ASP.NET!

       <%
```

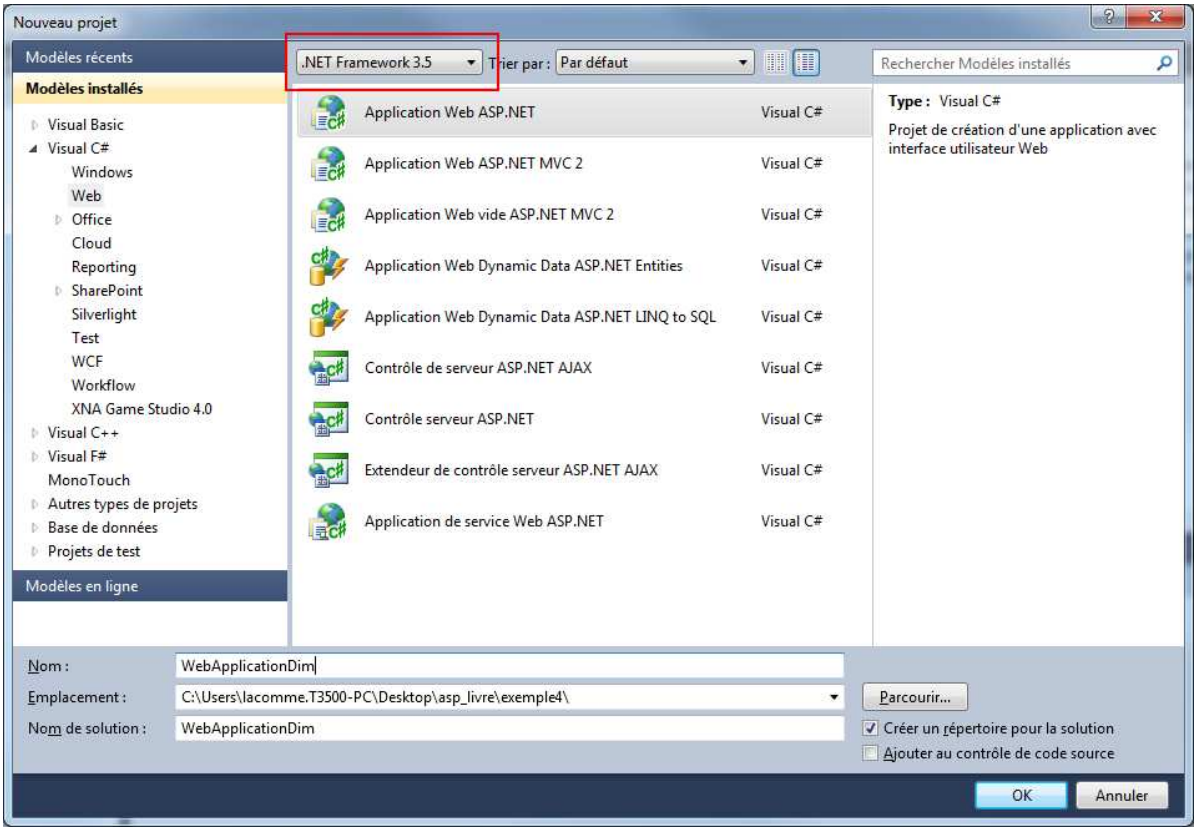
```
        ShowLineage();
       %>


    </h2>
    <p>
        Pour en savoir plus sur ASP.NET, consultez <a href="http://www.asp.net" title="Site
Web ASP.NET">www.asp.net</a>.
    </p>
    <p>
        Vous pouvez également trouver de la <a
href="http://go.microsoft.com/fwlink/?LinkID=152368"
           title="Documentation ASP.NET sur MSDN">documentation sur ASP.NET sur MSDN</a>.
    </p>
</asp:Content>
```

Ce qui donne :



3.2. Publier une application ASP.NET.

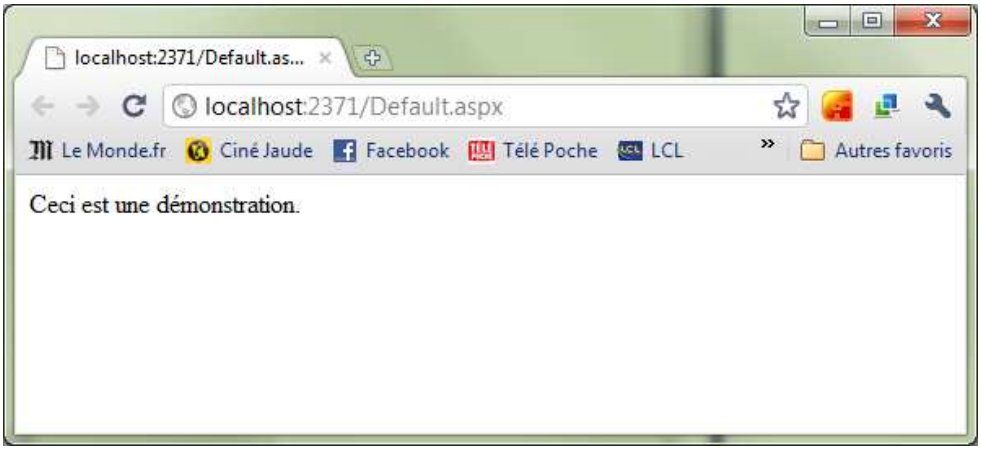
Créer un nouveau projet application Web ASP.NET et l'appeler *WebApplicationDim*.
Bien choisir la .NET Framework 3.5.



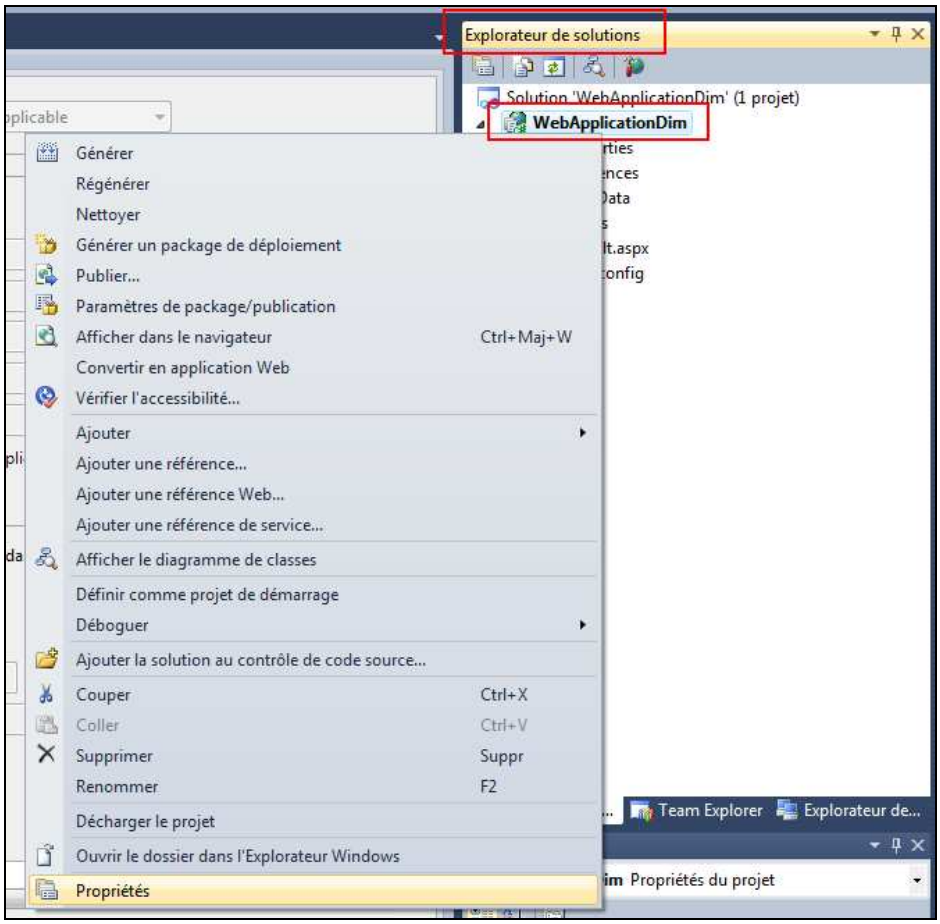
Modifiez le code en ajoutant un affichage dans la partie body comme indiqué ci-dessous :

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

<!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></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    </div>
    </form>
    Ceci est une démonstration.
</body>
</html>
```

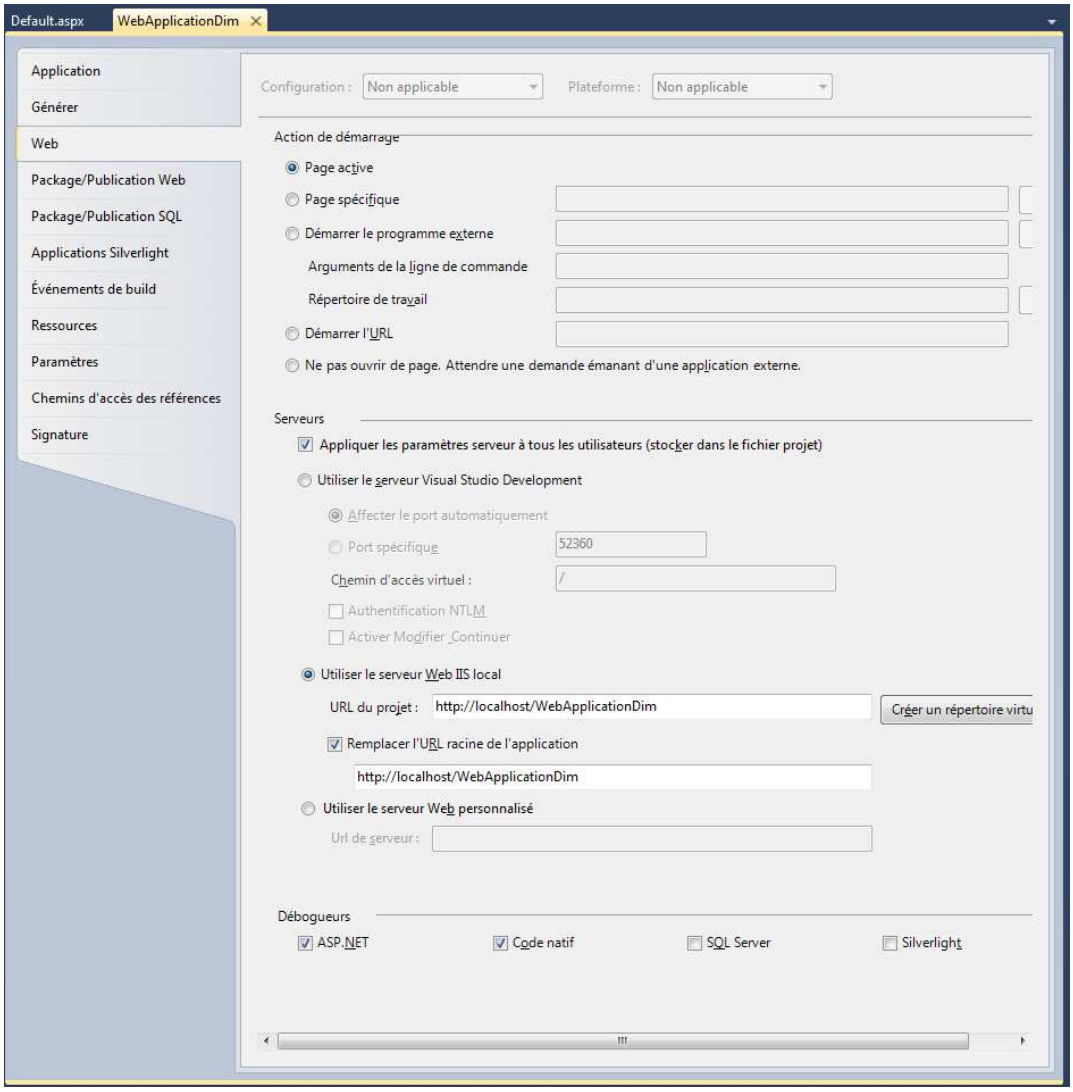
Vérifiez que l'application démarre sur le serveur ASP.NET de développement :



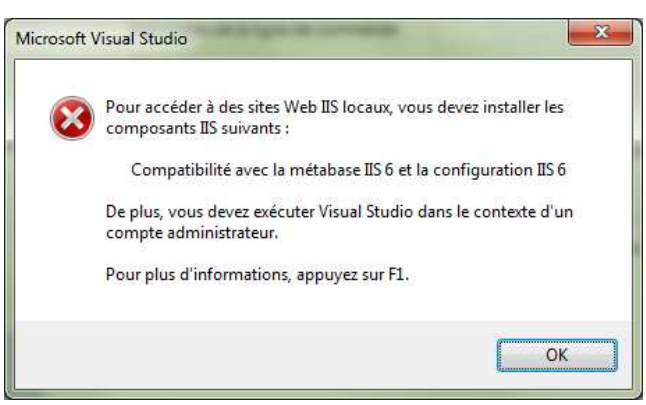
Faire un click droit sur le projet et choisir **Propriétés**. (Aller dans l'affichage pour afficher *l'Explorateur de solution* si besoins)

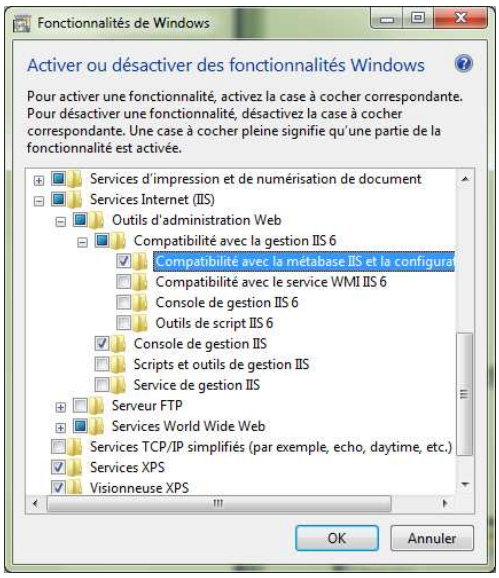


Modifiez les options comme suit :

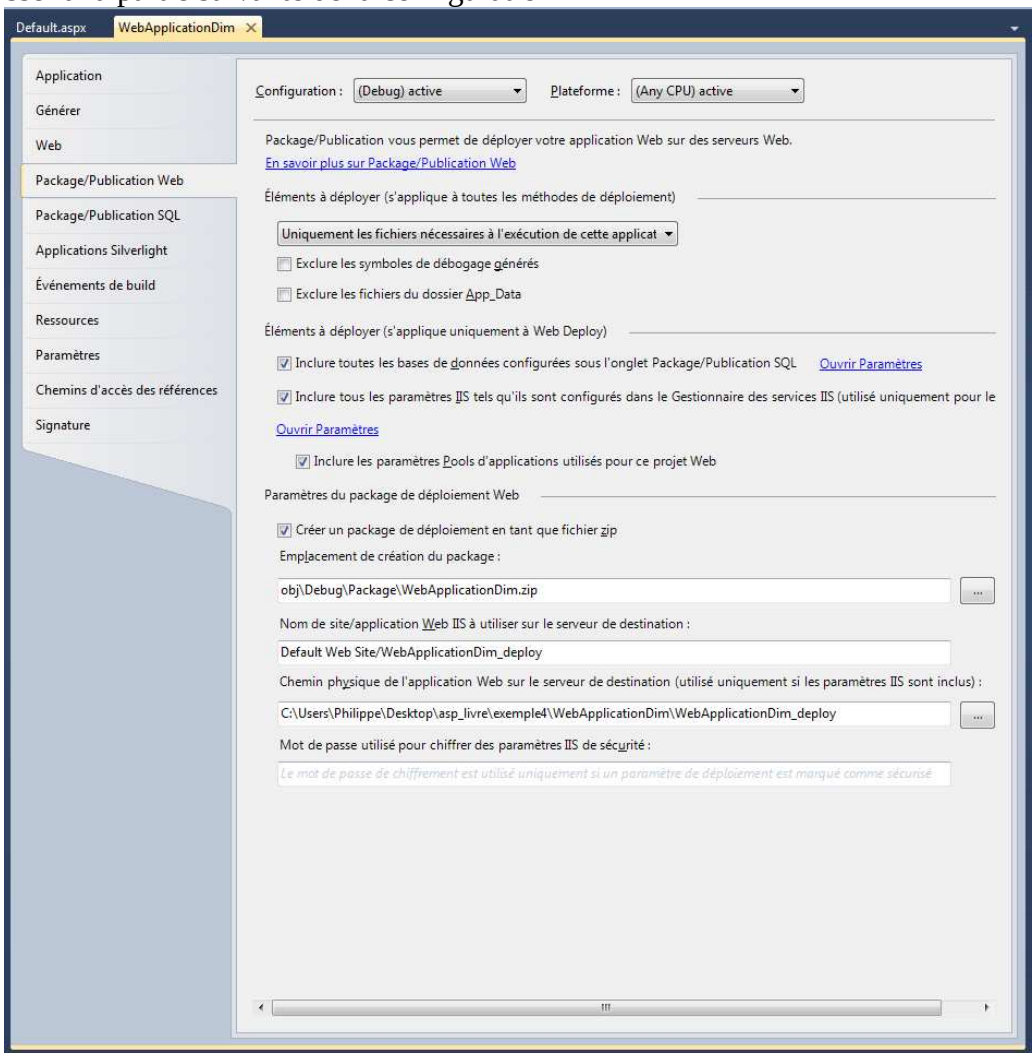


Si un message d'erreur suivant apparait faite la modification comme le montre l'image qui suit (vous pouvez vous aider de la partie sur la configuration du service IIS du début du tuto) :

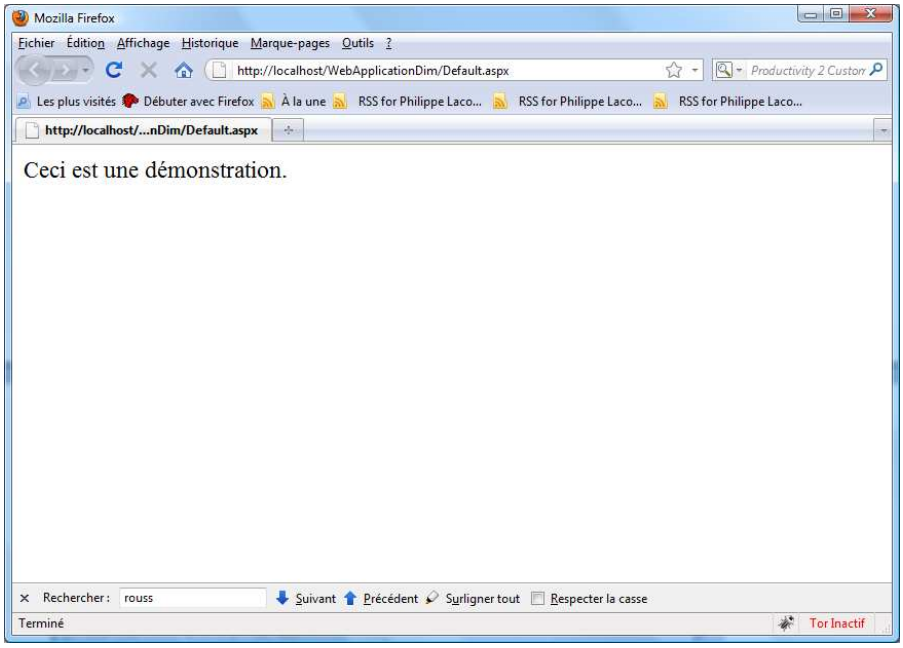




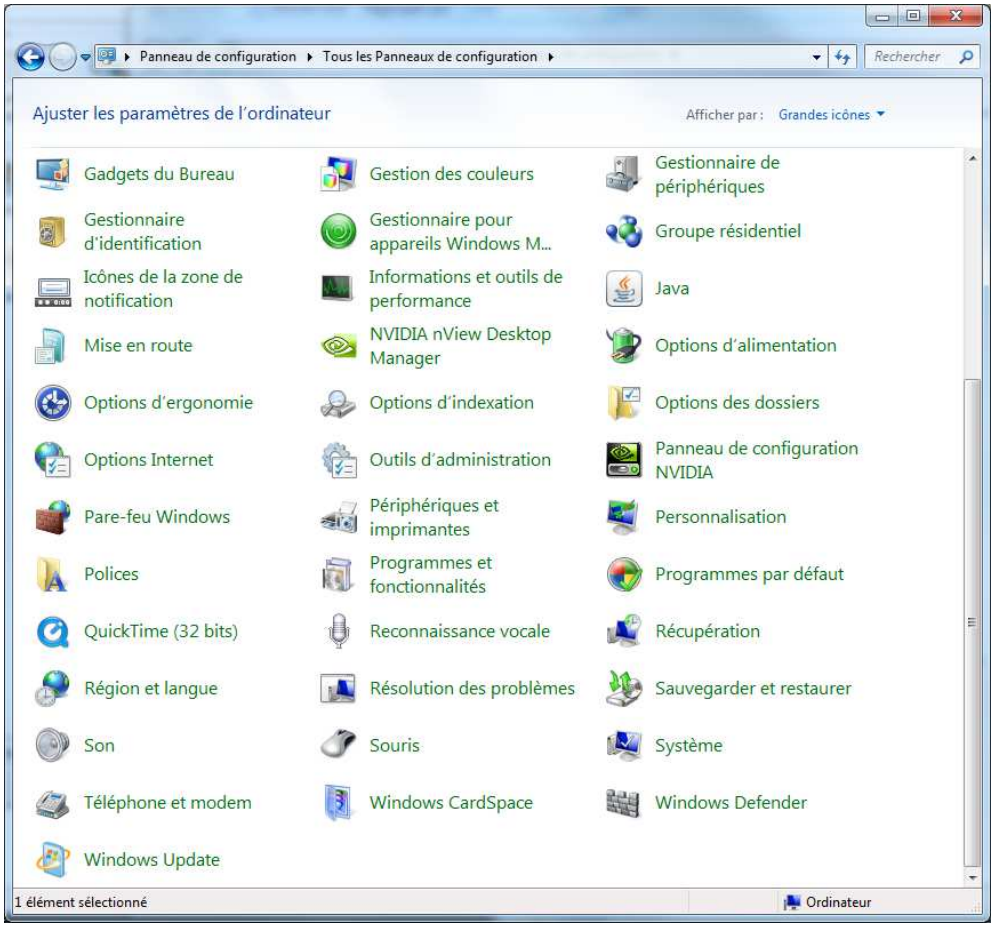
Puis passer à la partie suivante de la configuration



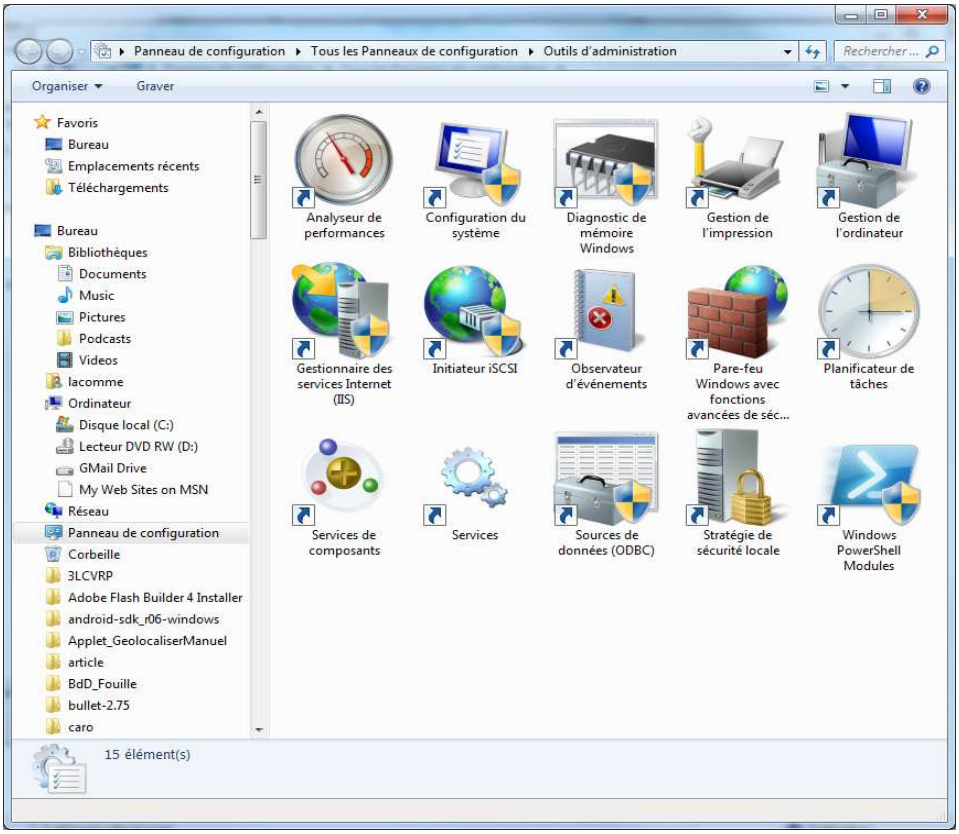
Notons que l'exécution se fait alors directement sur le serveur de développement IIS avec l'adresse : http://localhost/WebApplicationDim/Default.aspx



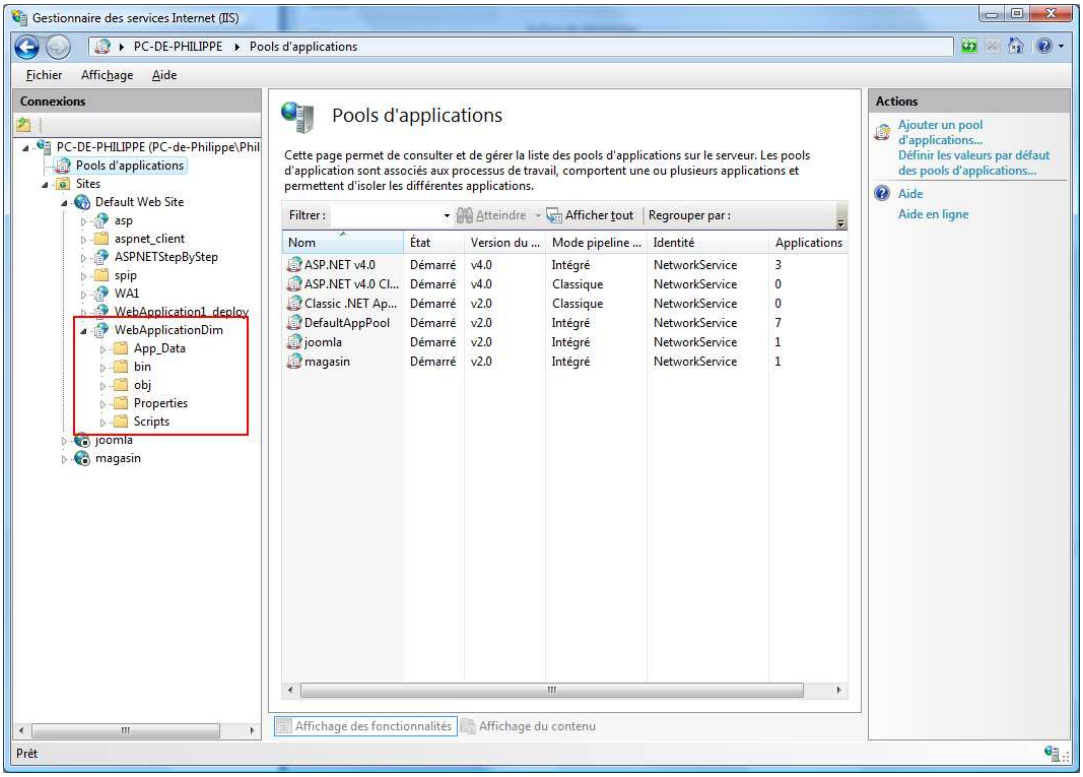
Lancer IIS.
Aller dans le menu **Demarrer** et choisir **Panneau de Configuration.**



Choisir **Outils d'administration** puis **Gestionnaire des services Internet (IIS)**.



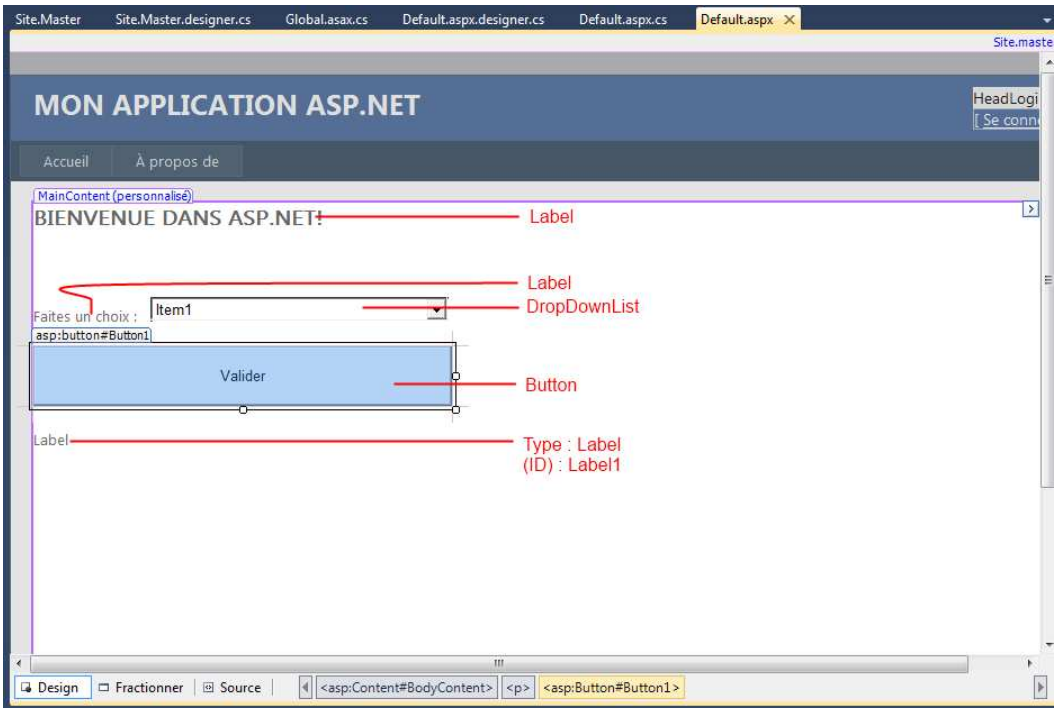
On retrouve alors l'application *WebApplicationDim* sur le **Default Web Site** :



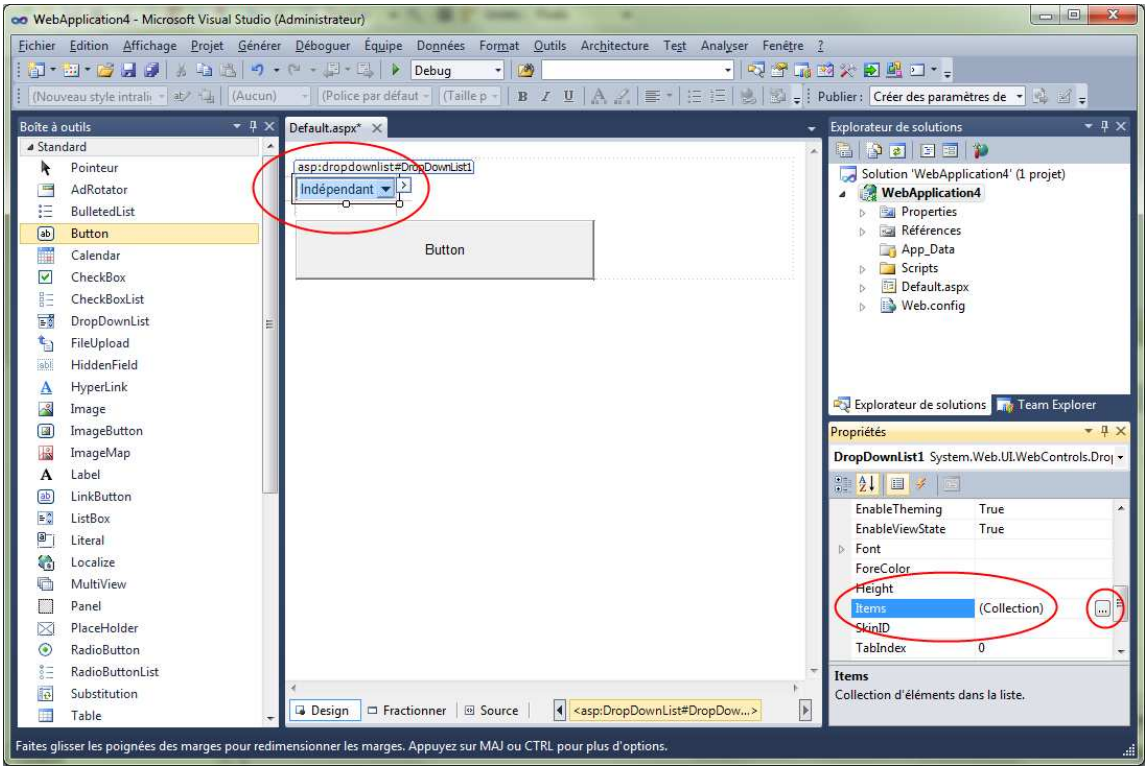
3.3. Utiliser les composants

Avec Visual Studio, créer une application ASP.NET nommé *Mon Application ASP*.

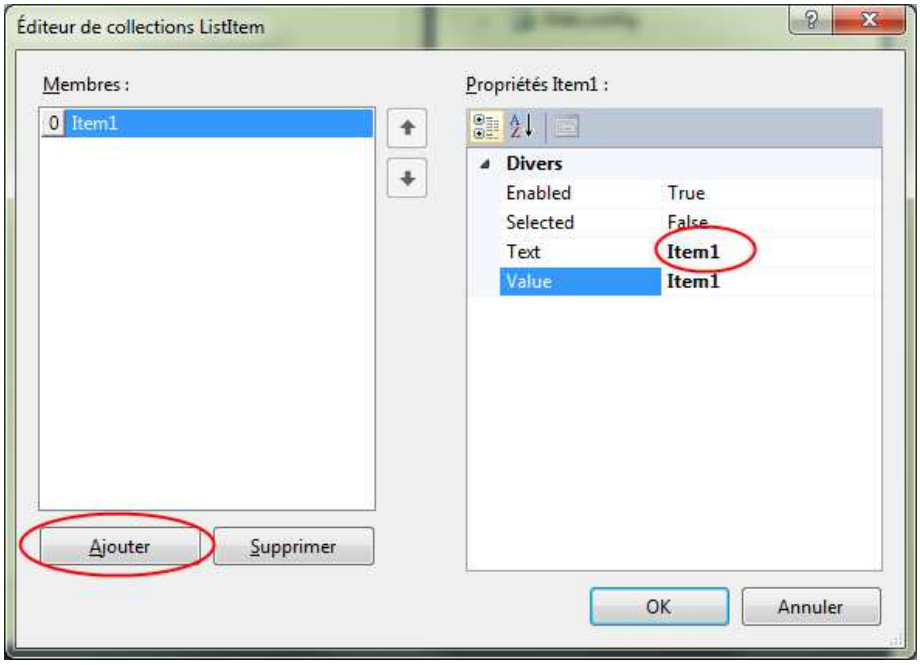
Modifier le fichier Default.aspx comme suit (pour configurer le « DropDownList » voir juste après) :



Pour configurer le « DropDownList », il faut aller modifier la propriété de l'objet.



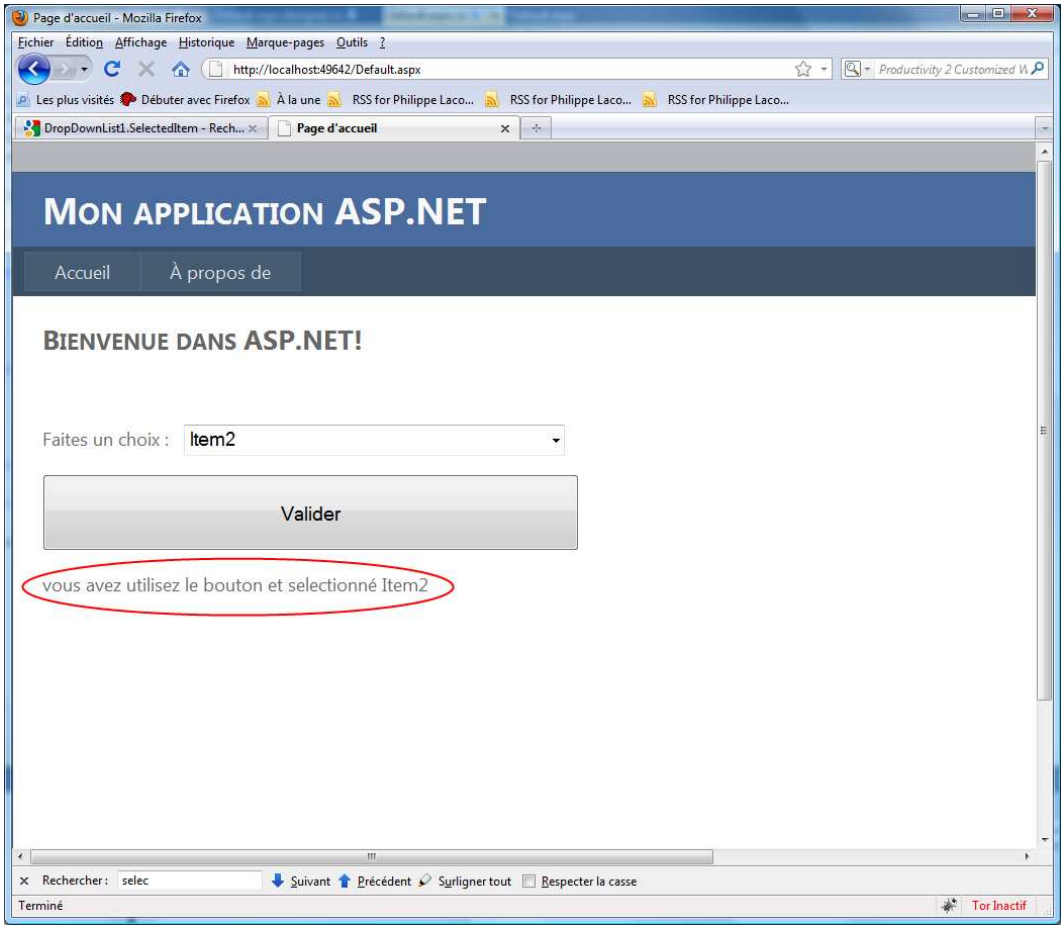
Puis ajouter les Items.



Faire un double click sur le bouton afin d'obtenir le code C# (vous vous retrouvez alors dans le fichier Default.aspx.cs qui contient le code source de votre programme) :

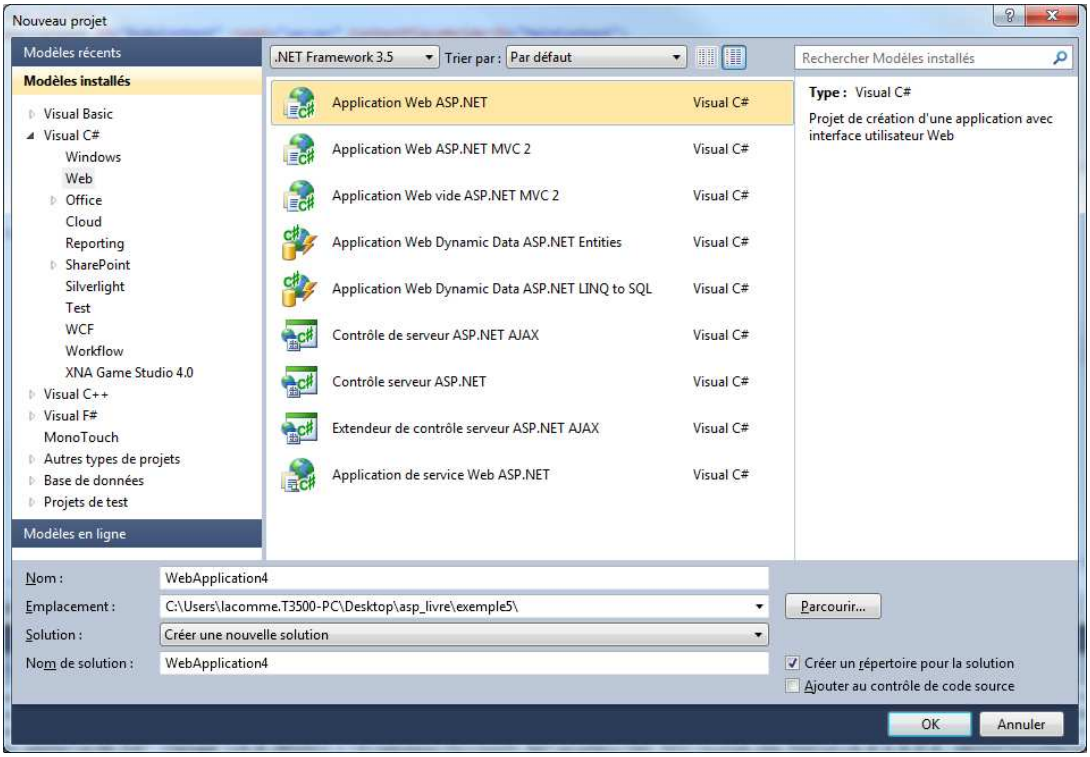
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Mon_Application_ASP
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs
e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            String i = DropDownList1.SelectedItem.Value;

            Label1.Text = "vous avez utilisez le bouton et selectionné "+i;
        }
    }
}
```

On obtient alors le code suivant à l'exécution :

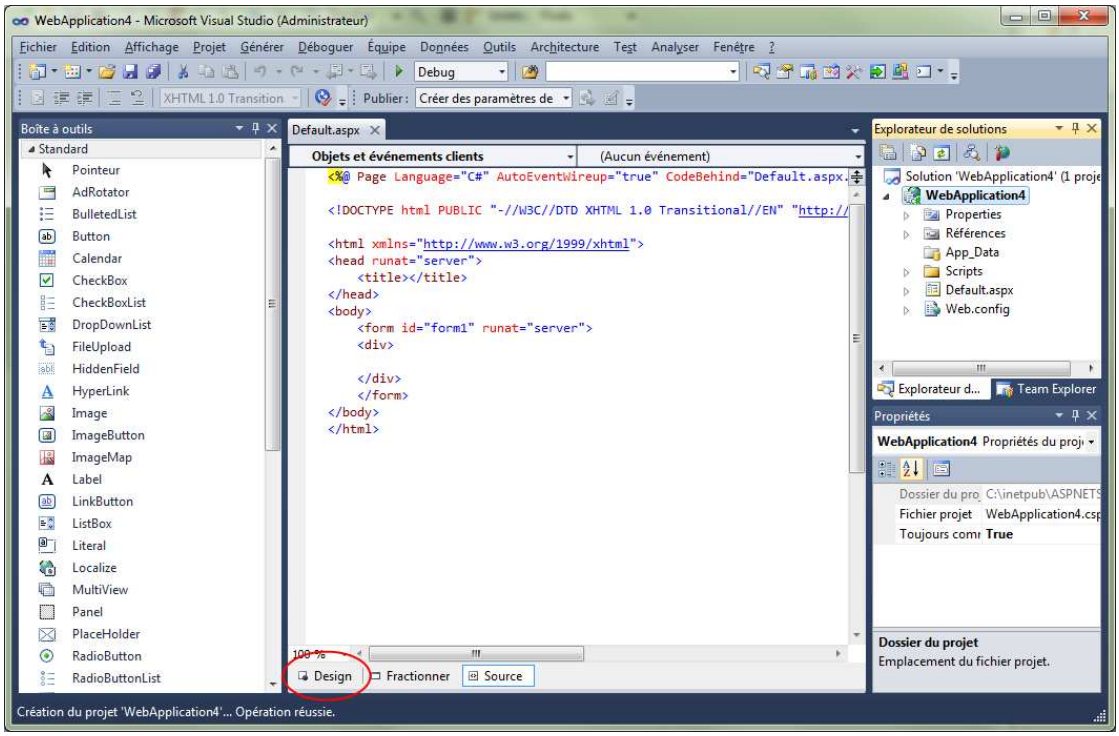


4. Utilisation des contrôles personnalisés

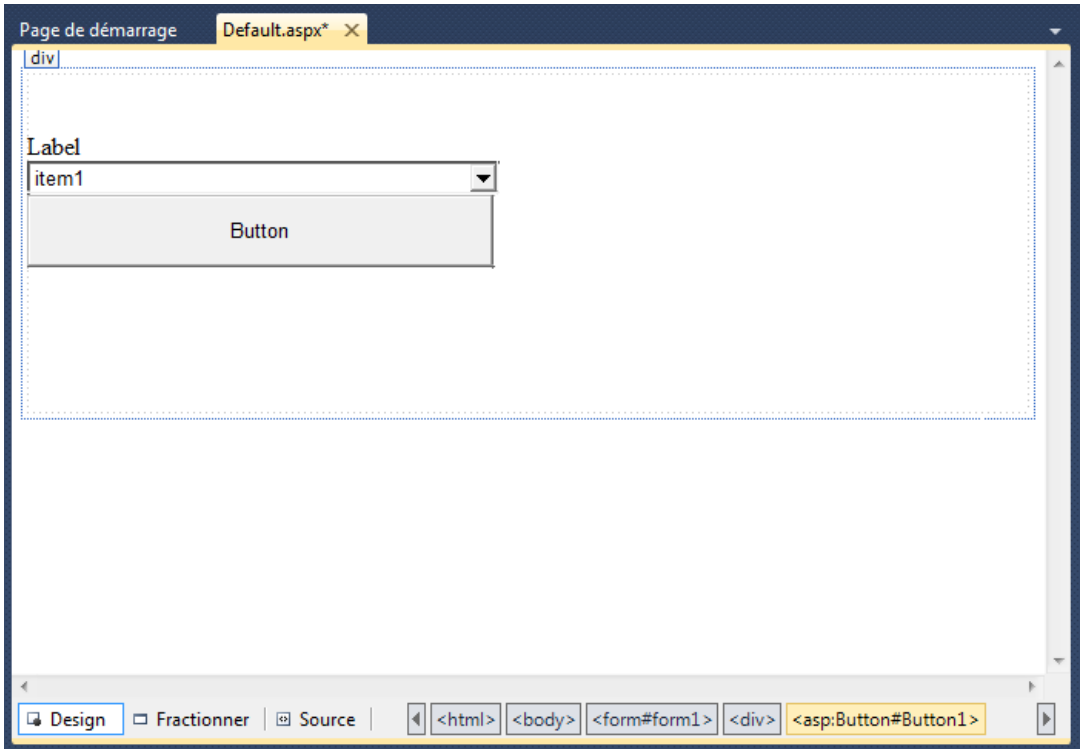
Créer un projet de type Application Web ASP.Net Framework 3.5 et appelez-le WebApplication4.



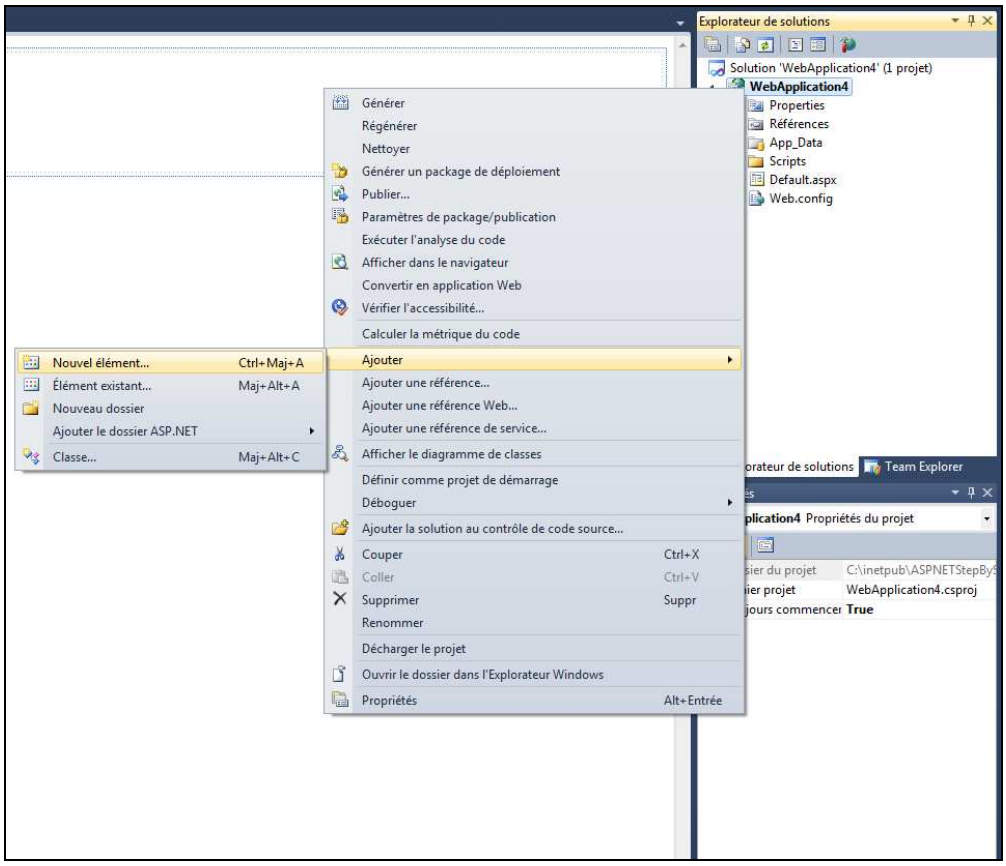
Si besoin, cliquer sur le bouton l'onglet « Design » en bas à gauche pour avoir la vue « Design » de votre projet.



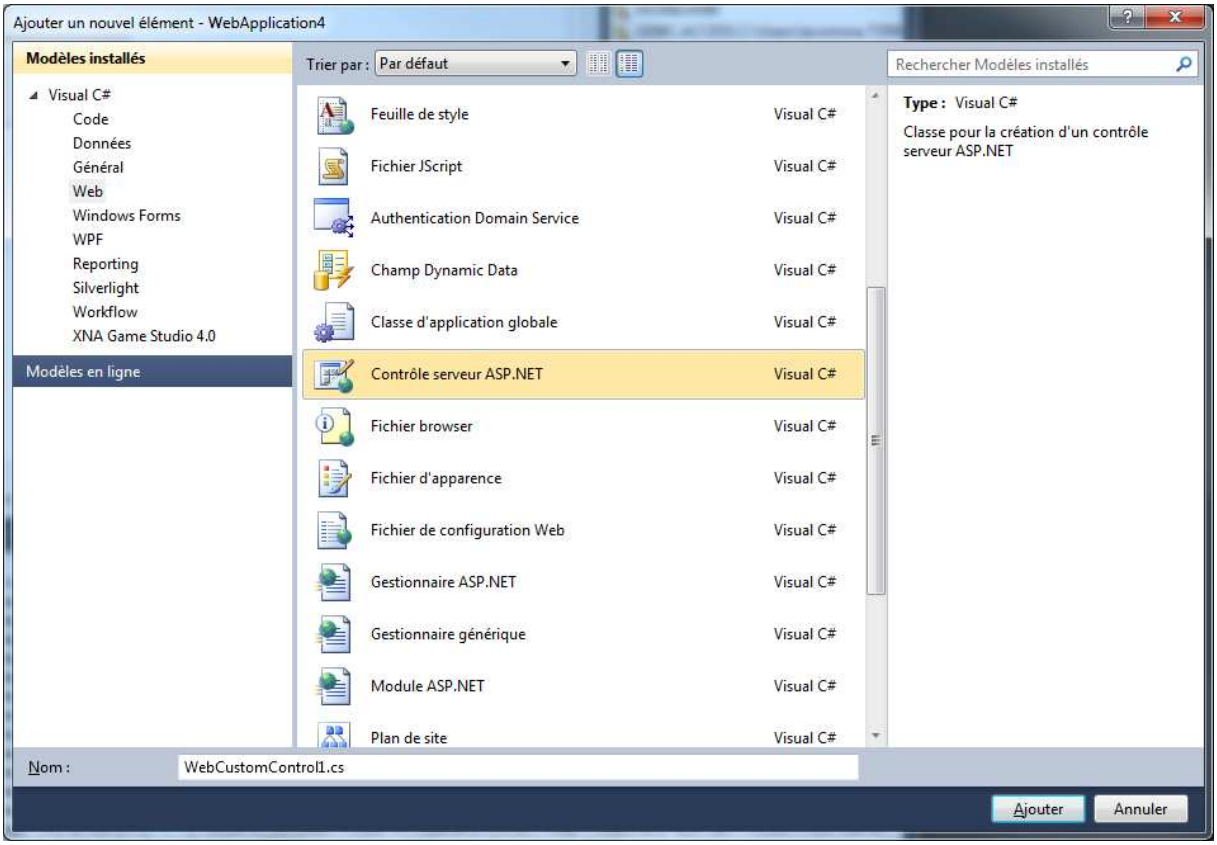
Ouvrir le fichier **Default.aspx** et créer une interface de la forme suivante :



Faire un click sur le projet et choisir **Ajouter nouvel élément**.



Choisir ensuite **Contrôle Server ASP.NET**



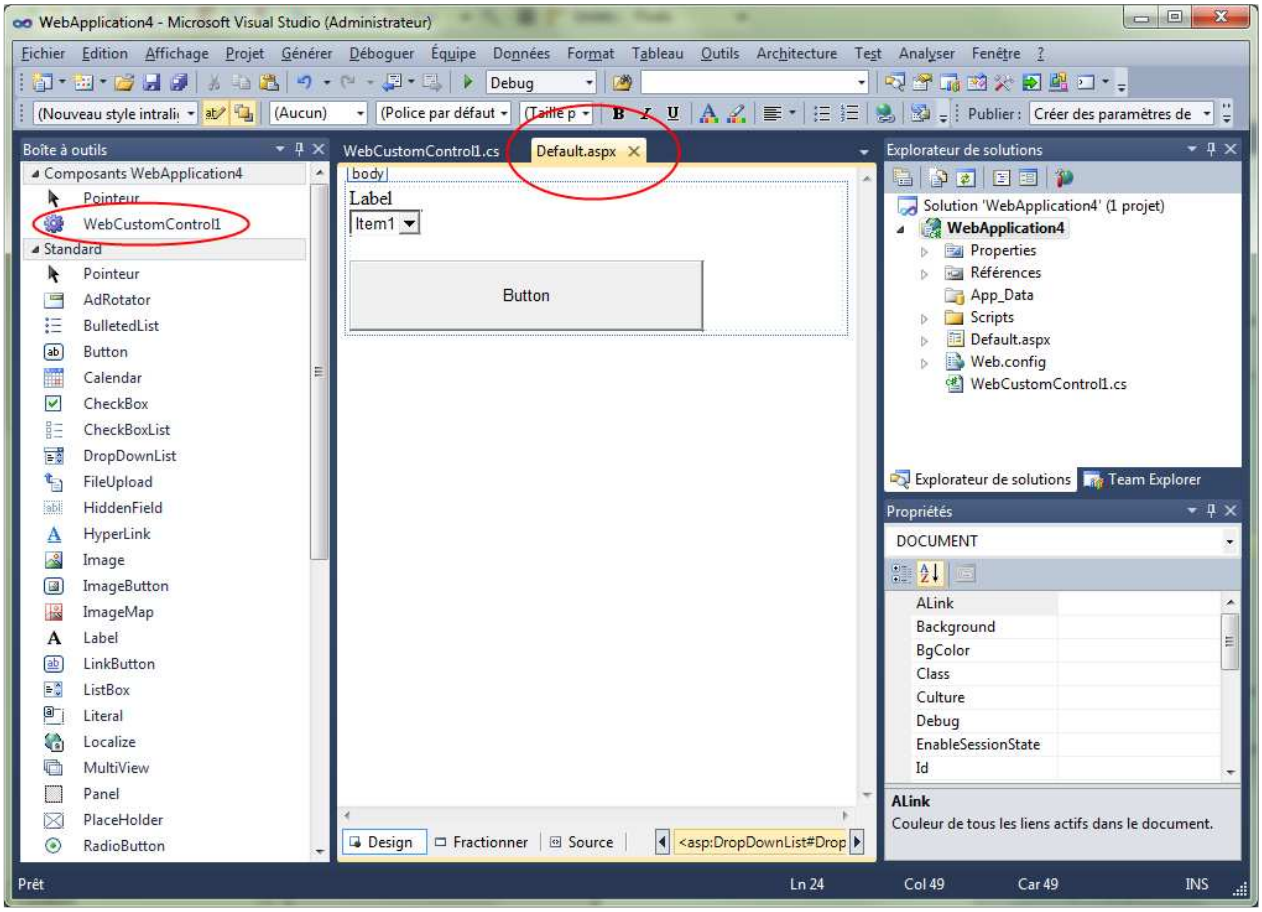
Visual Studio génère alors le code suivant :

```
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 WebApplication4
{
    [DefaultProperty("Text")]
    [ToolboxData("<{0}:WebCustomControl1 runat=server></{0}:WebCustomControl1>")]
    public class WebCustomControl1 : WebControl
    {
        [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;
```

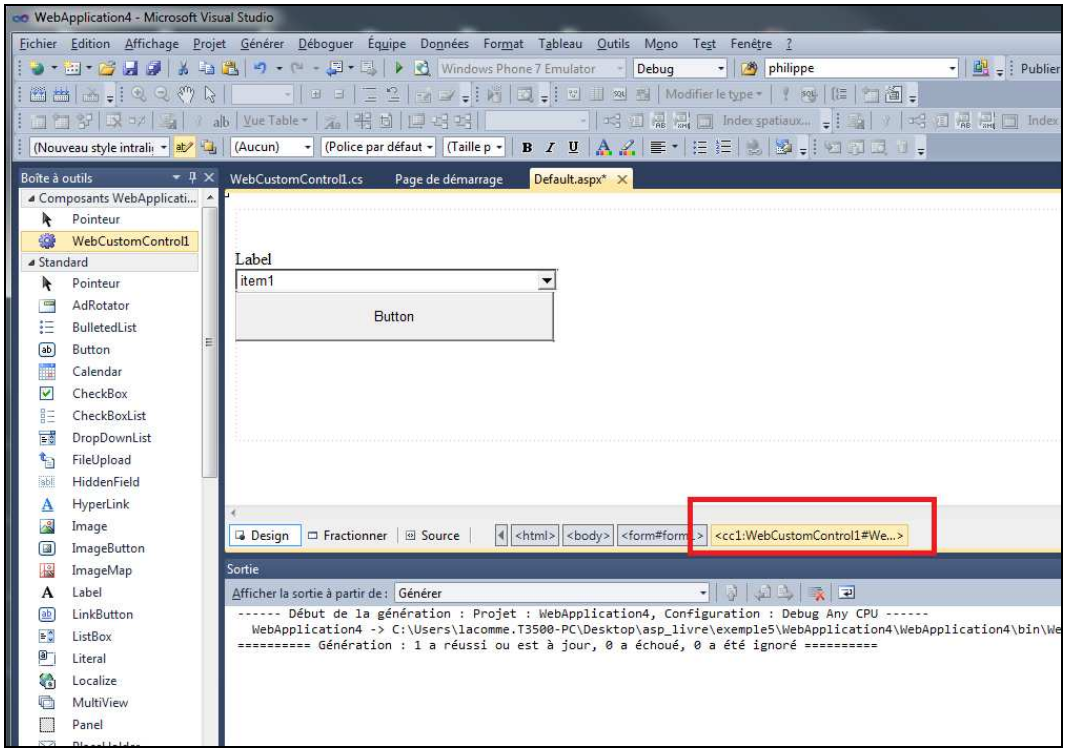
```
            }
        }
        protected override void RenderContents(HtmlTextWriter output)
        {
            output.Write(Text);
        }
    }
}
```

Générer le projet.

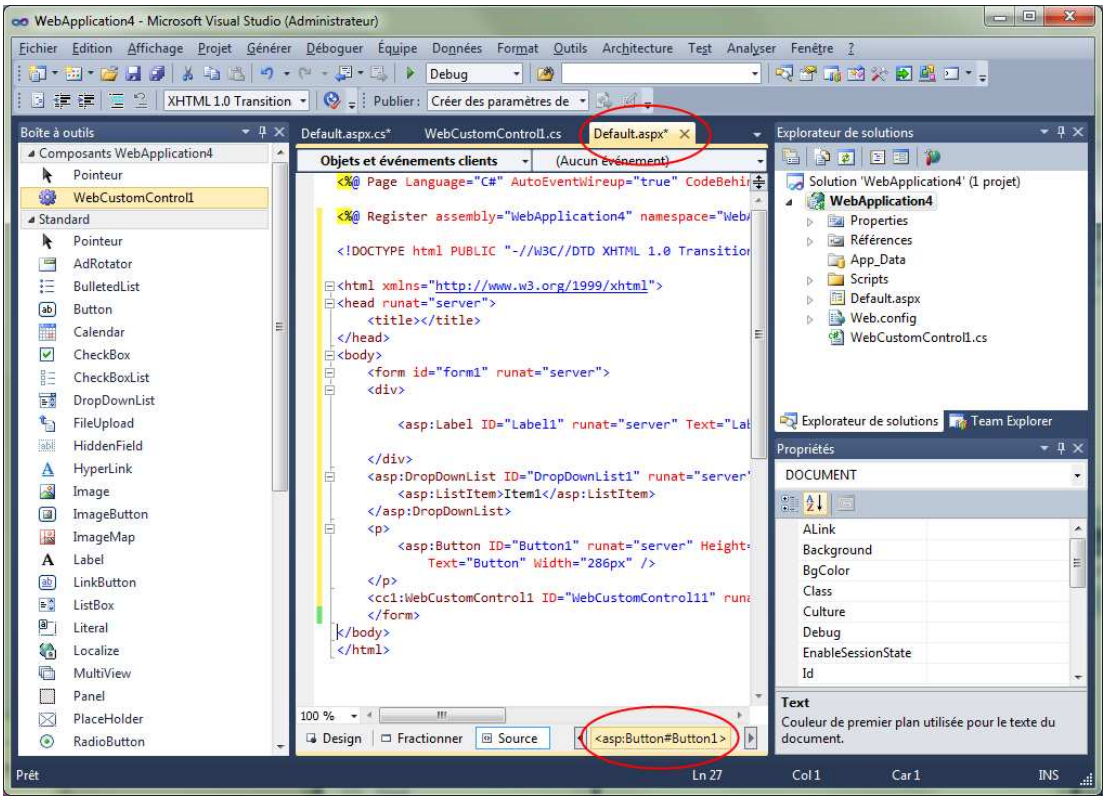
Consulter ensuite à la boite à outils.
Vous devriez retrouver votre **WebCustomControl1**.



Sélectionnez WebCustomControl1 et à l'aide de la souris, posez-le sur la page.

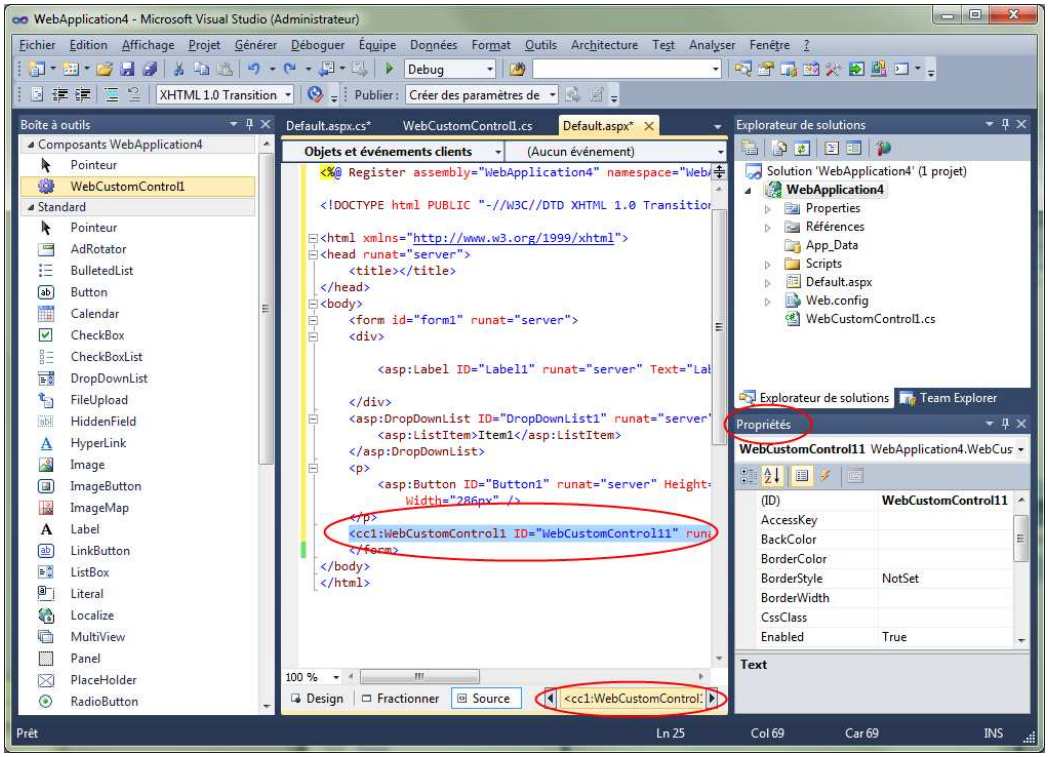


Visualisez le code aspx. Le contrôle est présent à la fin du fichier.

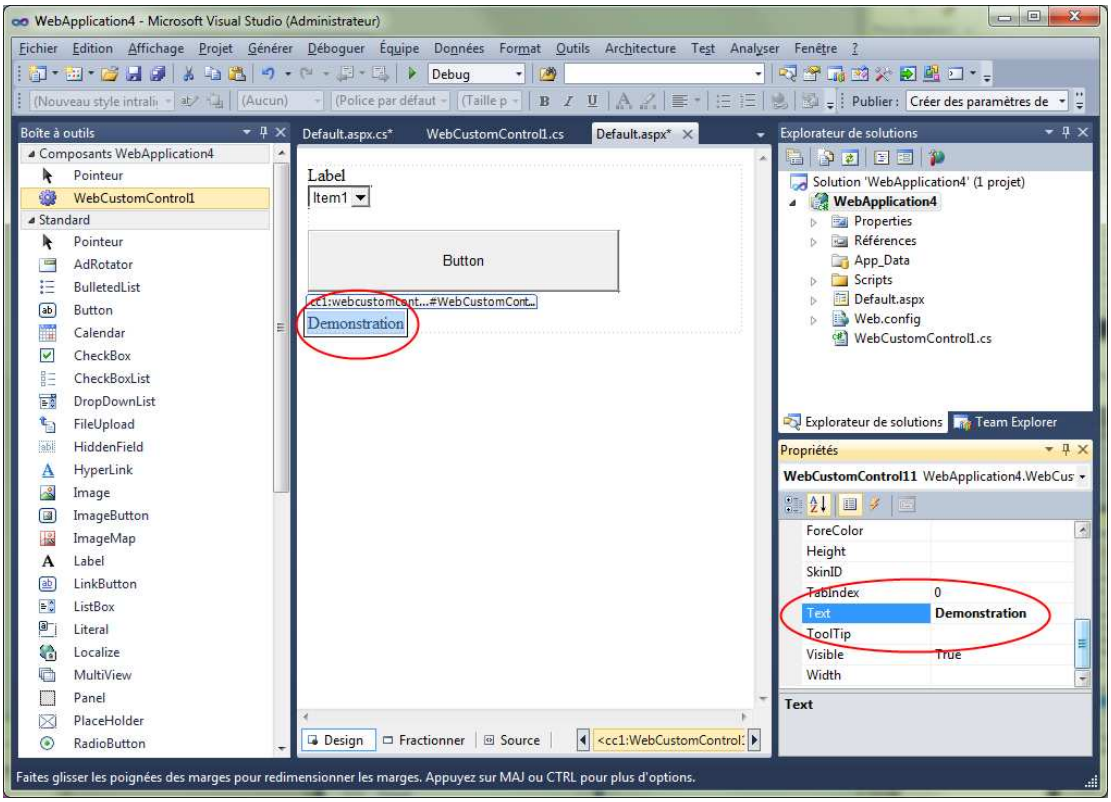


```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="WebApplication4._Default" %>

<%@ Register assembly="WebApplication4" namespace="WebApplication4" tagprefix="cc1" %>

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

        <br />
        <br />
        <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        <br />
        <asp:DropDownList ID="DropDownList1" runat="server" Height="16px"
Width="306px">
            <asp:ListItem>item1</asp:ListItem>
            <asp:ListItem>item2</asp:ListItem>
            <asp:ListItem Value="item3">item3</asp:ListItem>
            <asp:ListItem>item4</asp:ListItem>
        </asp:DropDownList>
        <br />
        <asp:Button ID="Button1" runat="server" Height="47px" Text="Button"
            Width="303px" />
        <br />
        <br />
        <br />
        <br />
        <br />

    </div>
    <cc1:WebCustomControl1 ID="WebCustomControl11" runat="server" />
    </form>
</body>
</html>
```

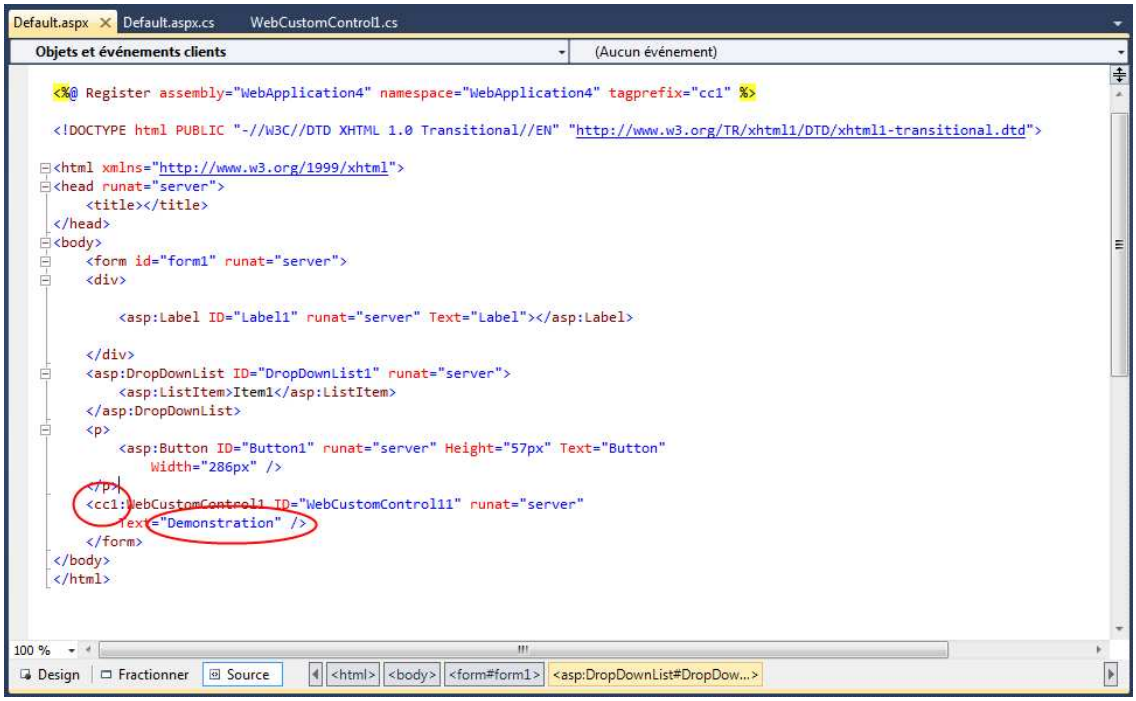
Sélectionnez le **WebCustomControl1** et consultez le panneau **Propriétés**.



Modifiez le champ **text** et constatez ce qui se passe sur le concepteur d'interface.



Cela modifie le code (mode « Source ») comme suit :



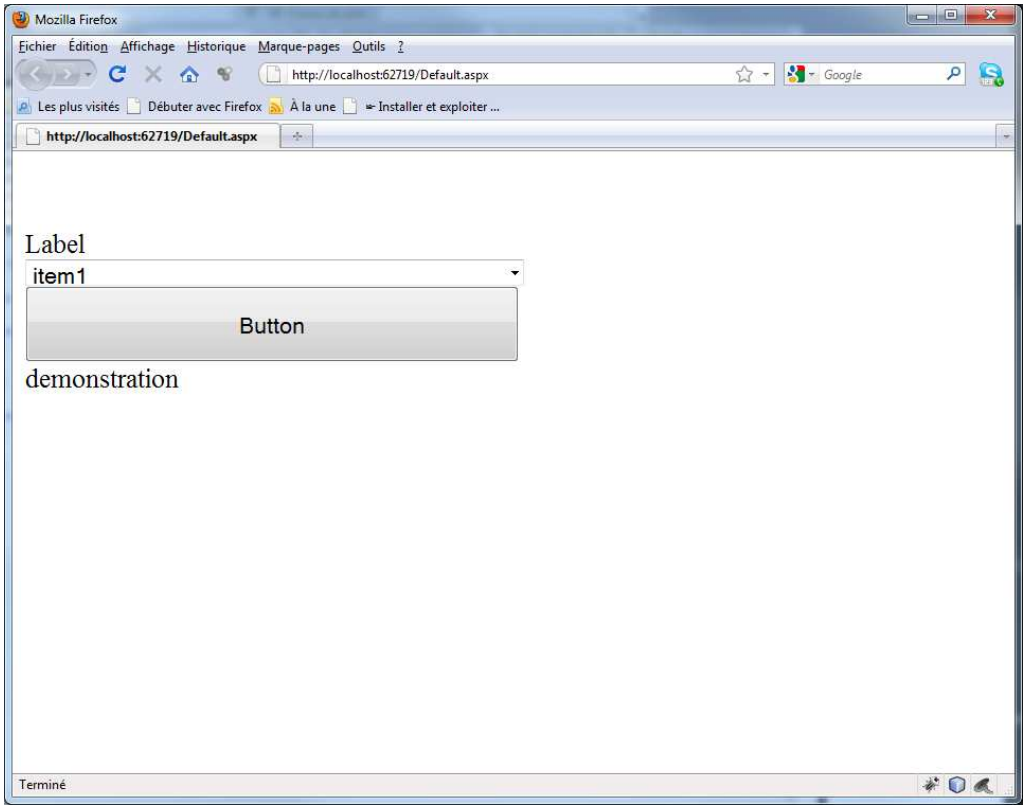
Notez l'utilisation du préfixe **cc1** au lieu de **asp**.

Revenez en mode « Design » et faites un double Click sur le bouton. Visual Studio crée alors automatiquement le gestionnaire d'événements lié au bouton. Ajouter alors la ligne suivante :

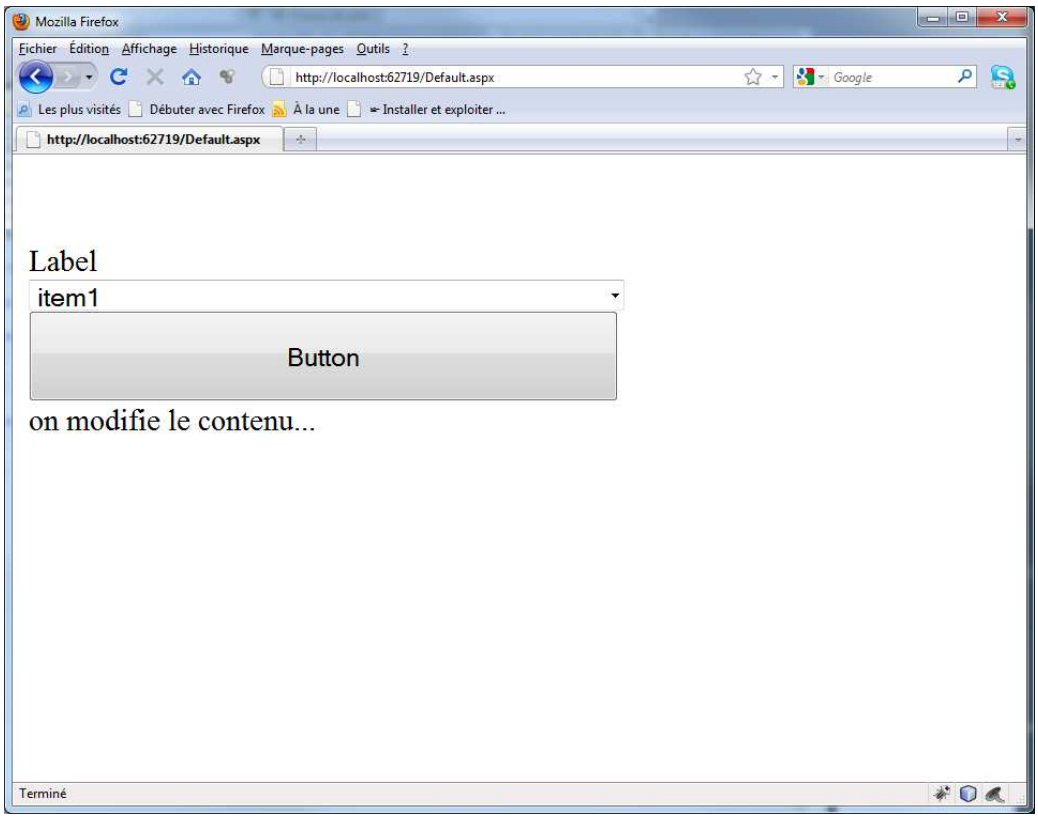
```
this.WebCustomControl12.Text = "on modifie le contenu...";
```

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

namespace WebApplication4
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            this.WebCustomControl12.Text = "on modifie le contenu...";
        }
    }
}
```

Ce qui donne à l'exécution :



Puis après le click :



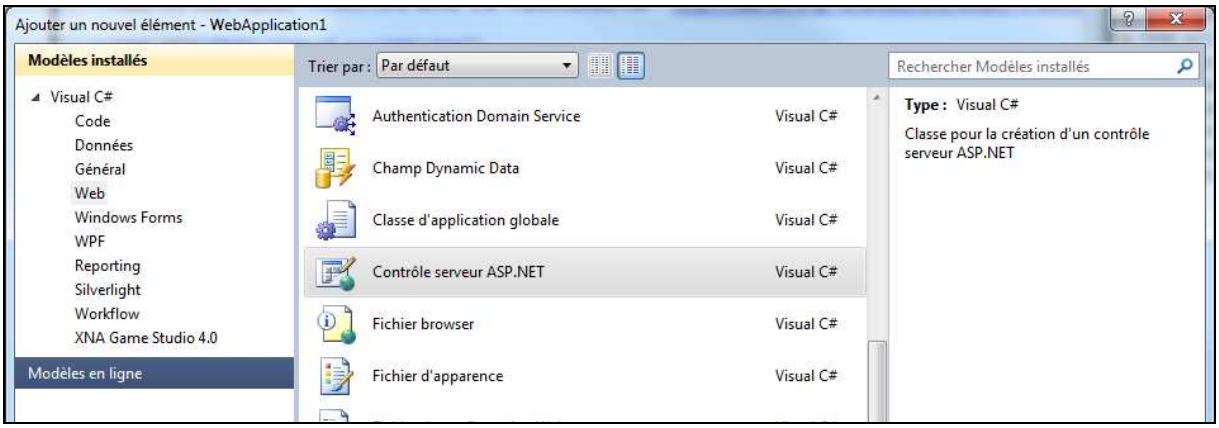
5. Utilisation d'un contrôle personnalisé : traitement des palindromes

5.1. Gestion des contrôles

Nous allons montrer comment un controleur peut faire une restitution différente en fonction de la valeur de sa propriété **text**.

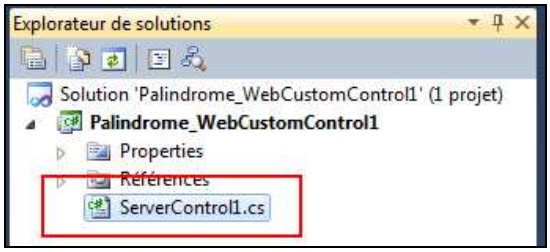
Créer un projet de type .Net Framework 3.5

Ajouter un **Contrôle serveur ASP.NET**



Choisir comme nom : **Palindrome_WebCustomControl1**

Ouvrir le fichier **ServerControl1.cs**



Ajouter ces deux procédures dans le « public class ServerControl1 : WebControl » du « namespace Palindrome_WebCustomControl1 » :

- traitement_chaine qui traite la chaine de caractères (suppression des caractères etc…);
- tester_palindrome qui renvoie True ou False suivant les cas.

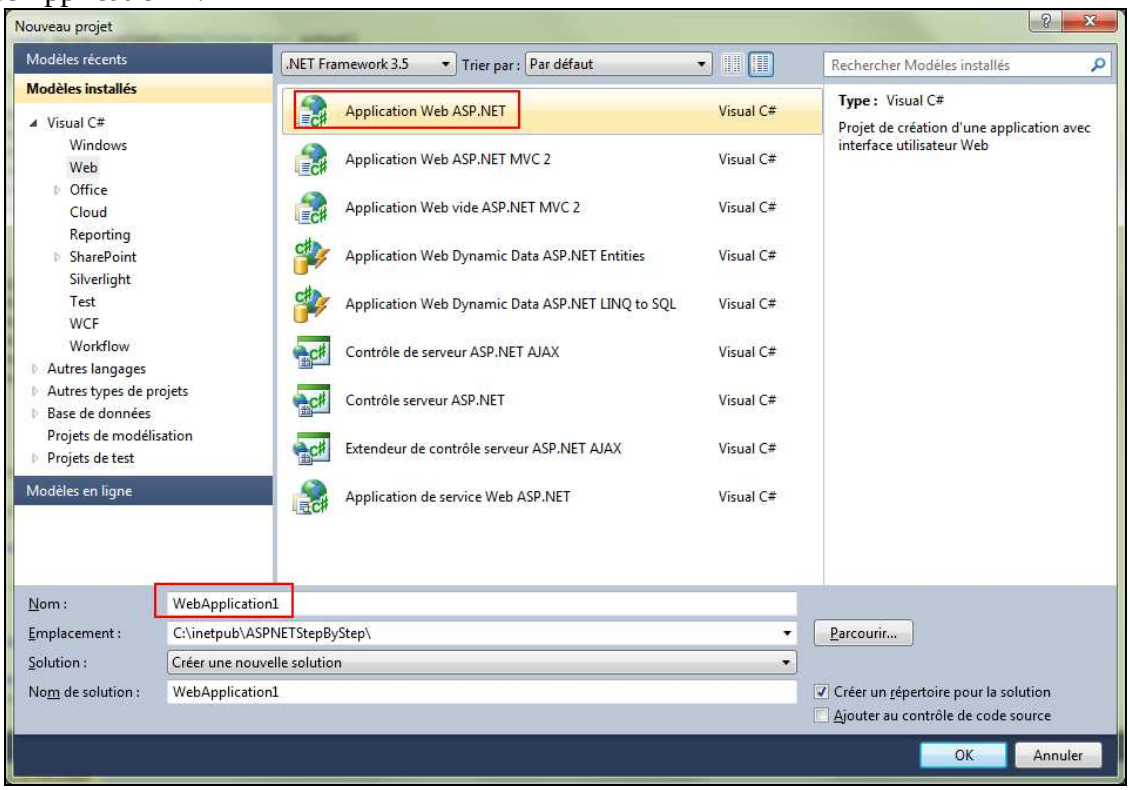
```
//------------------------------------------------------------------------------
        protected string traitement_chaine(string chaine)
        {
          string chaine_clone = (String) chaine.Clone();

          if (chaine != null)
          {
              char[] rgc = chaine_clone.ToCharArray();

              int i = 0;
              foreach (char c in rgc)
```

```
            {
                if (char.IsLetterOrDigit(c))
                    i++;
                else
                    chaine_clone = chaine_clone.Remove(i, 1);
            }
        }

        return chaine_clone;
    }

    protected bool Tester_Palindrome()
    {
        if (this.Text != null)
        {
            String StrConrolText = this.Text;
            String StrMajuscule = null;
            StrMajuscule = Text.ToUpper();

            StrConrolText = this.traitement_chaine(StrMajuscule);

            char[] chaine_inverse = StrConrolText.ToCharArray();
            Array.Reverse(chaine_inverse);

            String Chaine_inverse_finale = new string(chaine_inverse);

            if (StrConrolText == Chaine_inverse_finale)
                return true;
            else
                return false;

        }
        else
            return false;

    }
```

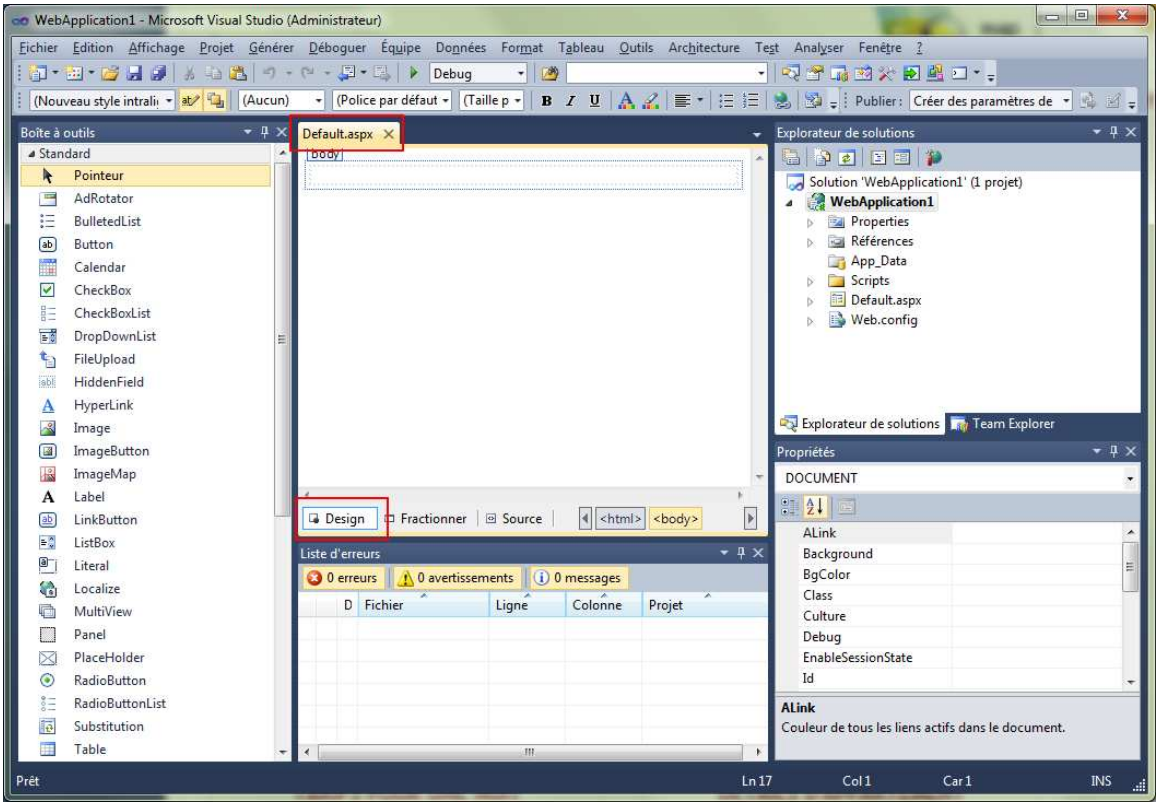
Modifier ensuite la procedure **RenderContents** comme suit :

```
    protected override void RenderContents(HtmlTextWriter output)
    {
        if (this.Tester_Palindrome() == true)
        {
            output.Write("c'est un palindrome: <br>");
            output.Write("<FONT size=5 color=blue>");
            output.Write("<B>");
            output.Write(this.Text);
            output.Write("</B>");
            output.Write("</FONT>");
        }
        else
        {
            output.Write("ce n'est pas un palindrome: <br>");
            output.Write("<FONT size=5 color=red>");
            output.Write("<B>");
            output.Write(this.Text);
            output.Write("</B>");
            output.Write("</FONT>");
        }

    }
```

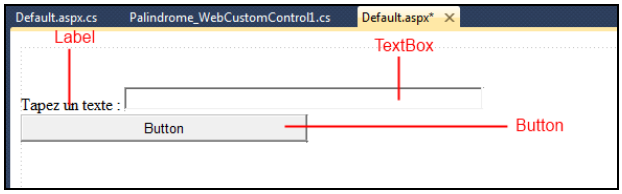
Créer une « Application Web ASP.NET » comme dans la partie 4 et nommé la WebApplication1 :



Ouvrir le fichier « Default.aspx » et se mettre en mode « Design » pour pouvoir créer l'interface graphique.

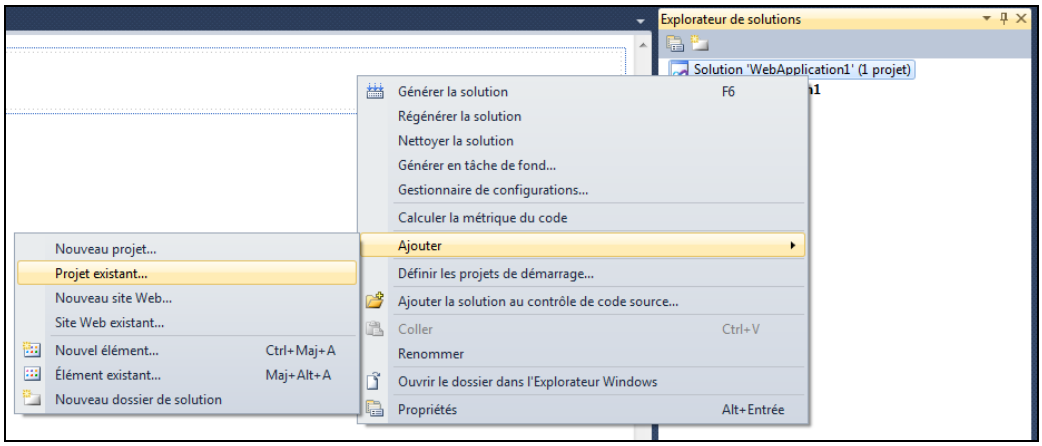


Créez la page web suivante :

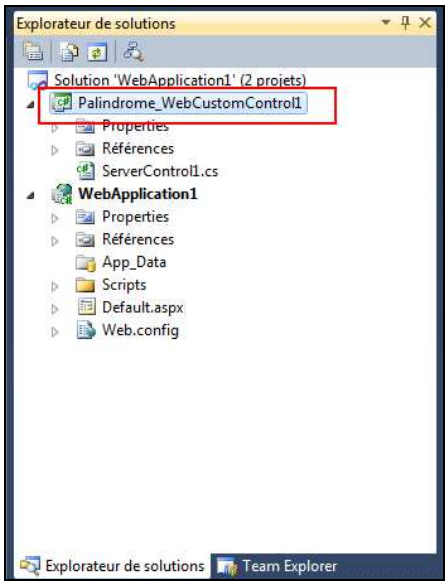


NB :Le « TextBox » s'appelle doit « TexBox1 ».

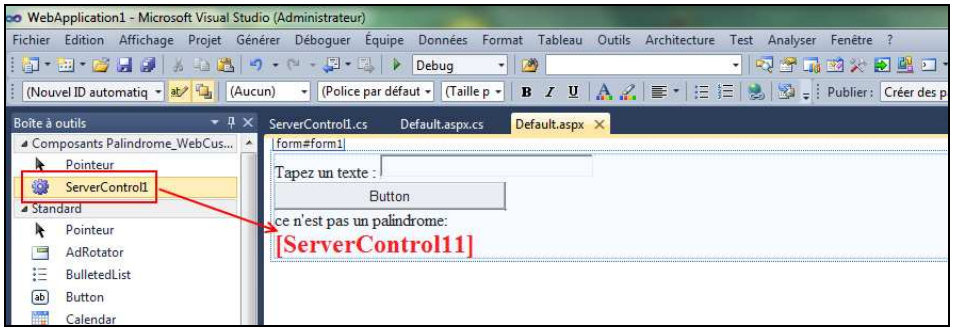
Puis ajouter le projet « **Palindrome_WebCustomControl1** » dans le projet courant :



Générer le projet « Palindrom_WebCustomControl1 » si ce n'est déjà fait.



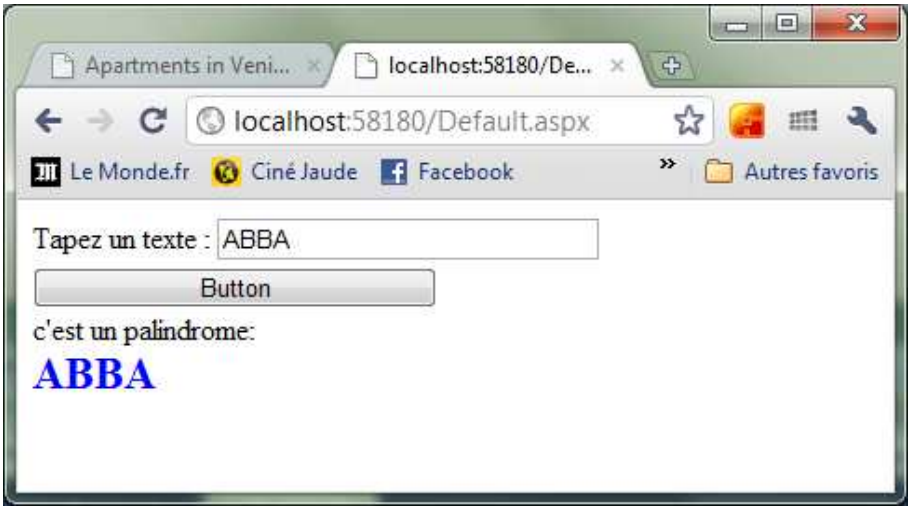
Le ServerControl1 devrait alors apparaitre à gauche. Faire un glisser-déposser pour le mettre sur la page web.



Faire un double click sur le bouton et modifier le code comme suit :

```
protected void Button1_Click(object sender, EventArgs e)
{
    this.ServerControl11.Text = this.TextBox1.Text;
}
```

L'exécution du code devra donner :

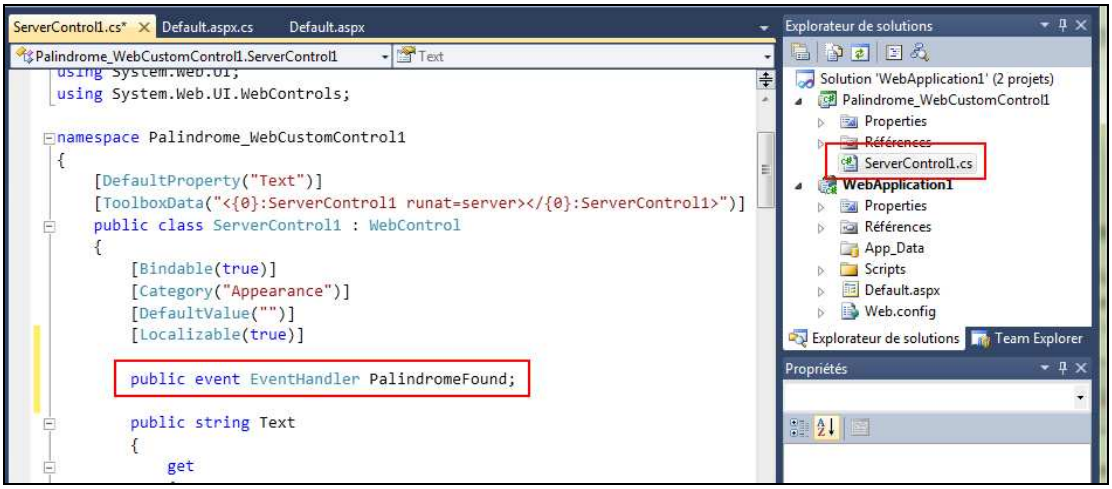


5.2. Gestion des contrôles

Nous allons ajouter un événements à la classe « Palindrome_WebCustomControl1 ».

Toujours dans le projet « WebApplication1 » dans lequel nous avons ajouté le projet « Palindrome_WebCustomControl1 », ajouter la ligne suivante dans le « public class ServerControl1 : WebControl » du fichier « ServerControl1.cs » :

```
public event EventHandler PalindromeFound;
```



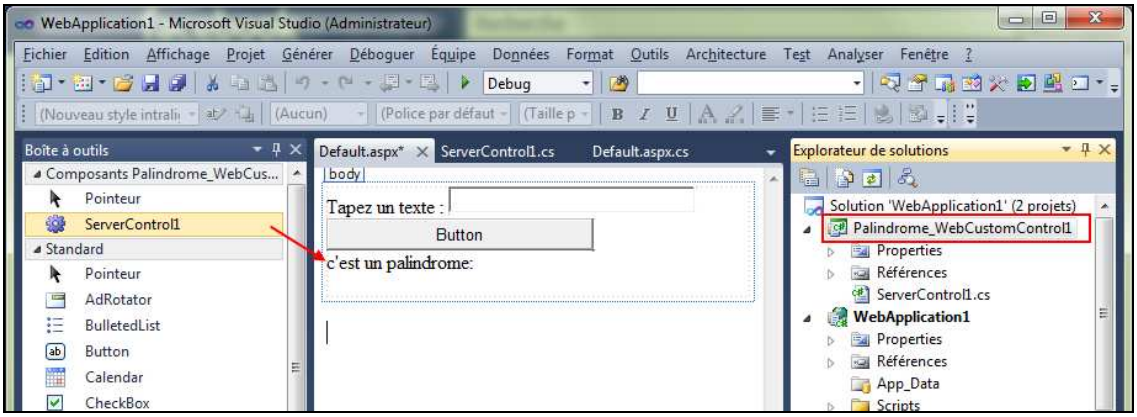
Nous allons générer un événement chaque fois que le texte est modifié. Pour cela il faut modifier « Text » dans le fichier « ServerControl1.cs » comme suit :

```
        public event EventHandler PalindromeFound;

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

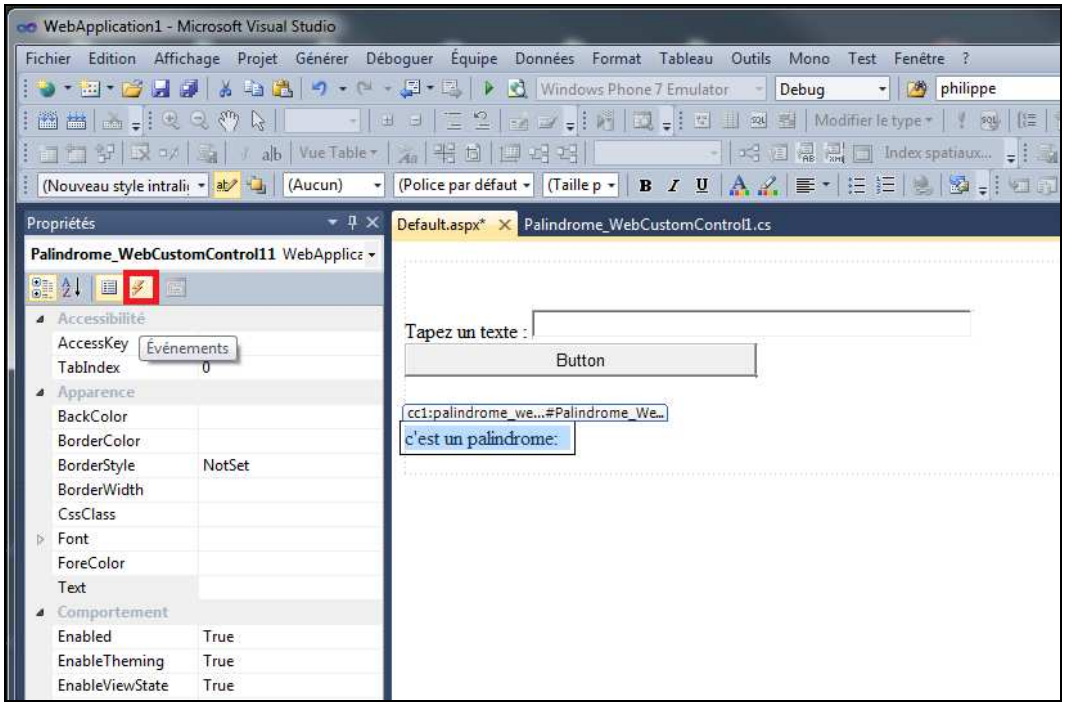
```
        set
        {
            ViewState["Text"] = value;
            if (this.Tester_Palindrome() == true)
            {
                if (PalindromeFound != null)
                    PalindromeFound(this, EventArgs.Empty);

            }
        }
    }
```

Editez le fichier **Default.aspx**.
Passez en mode **Design**.

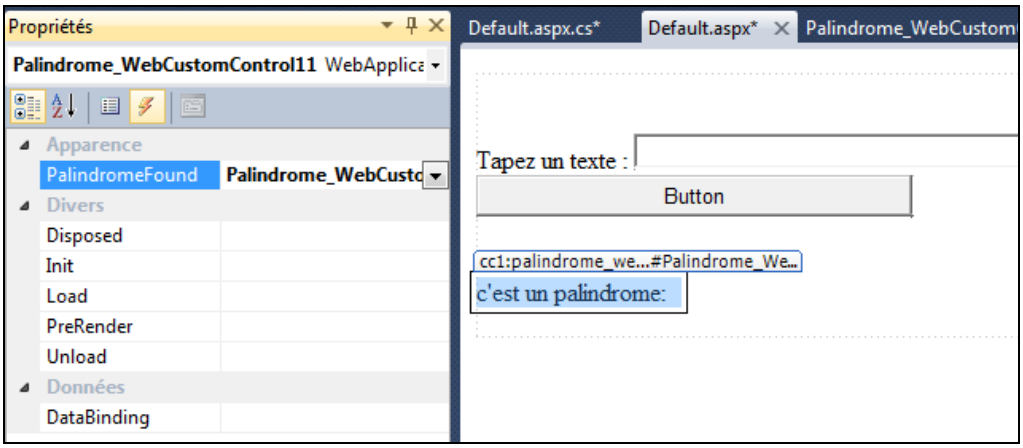
Supprimer le contrôle (« ce n'est pas un palindrome :… ») déjà présent. Régénérer le projet Palindrome_WebCustomControl1 et recréez le contrôle.



Sélectionnez le contrôle puis dans la partie **Propriétés** choisissez **Evenements** (bouton avec un éclair).



Allez à droit de PalindromeFound et faire un double click.

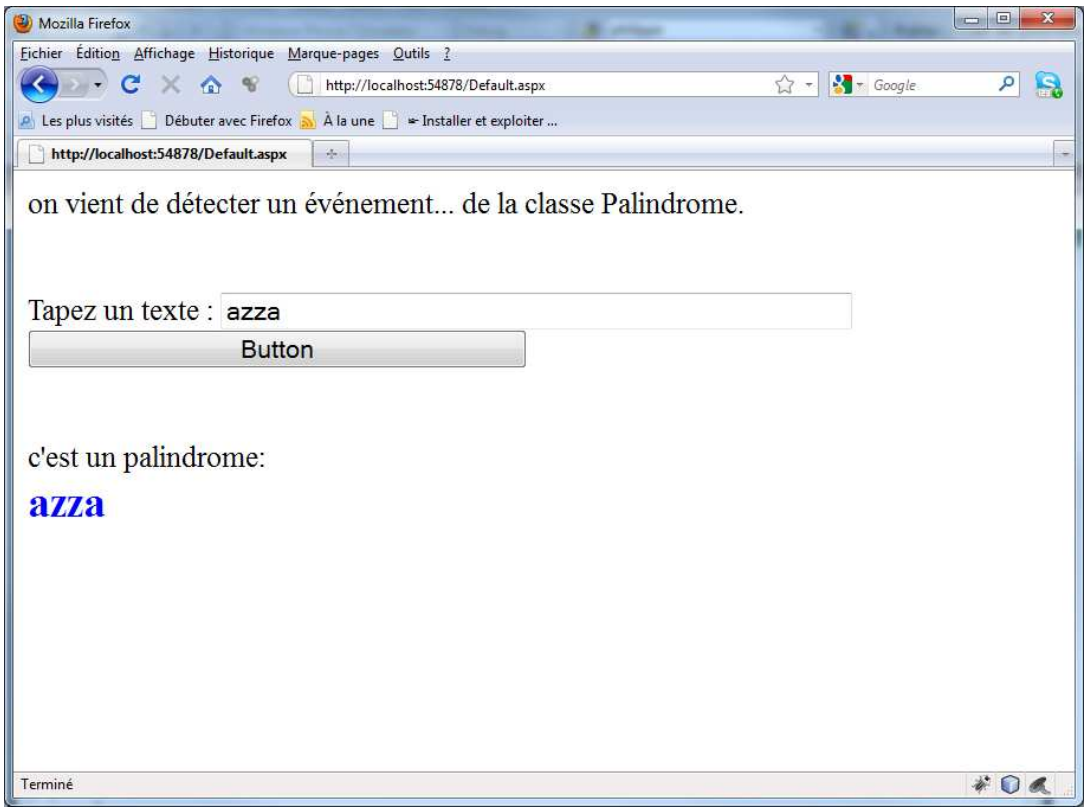


Visual Studio génère automatiquement le gestionnaire d'événements et on peut ajouter facilement un affichage dans la procédure de gestion de l'événement.

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

namespace WebApplication1
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
```

```
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            this.Palindrome_WebCustomControl11.Text = this.TextBox1.Text;
        }

        protected void Palindrome_WebCustomControl11_PalindromeFound(object sender,
EventArgs e)
        {
            Response.Write("on vient de détecter un événement... de la classe
Palindrome.");
        }
    }
}
```

Ceci doit donner à l'exécution :



Nous allons stocker les différents palindromes. Cela se fait en ajoutant un attribut dans la classe « ServerControl1 » du namespace « Palindrome_WebCustomControl1 » du fichier « ServerControl1.cs ».

```
    public class Palindrome_WebCustomControl1 : WebControl
    {
        [Bindable(true)]
        [Category("Appearance")]
        [DefaultValue("")]
        [Localizable(true)]

        ArrayList SavePalindrome = new ArrayList();
```

NB : ajouter si besoin le package System.Collections :

```
using System.Collections;
```

Ensuite, il faut modifier le setter de la classe comme suit :

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

            set
            {
                ViewState["Text"] = value;
                string texte = value;

                this.SavePalindrome = (ArrayList) this.ViewState["palindromes"];

                if (this.SavePalindrome == null)
                {
                  this.SavePalindrome = new ArrayList();
                }


                if (this.Tester_Palindrome() == true)
                {
                    if (PalindromeFound != null)
                        PalindromeFound(this, EventArgs.Empty);
                    SavePalindrome.Add(texte);
                }
                ViewState.Add("palindromes", SavePalindrome);
            }
        }
```

Ajouter la méthode suivante dans « public class ServerControl1 : WebControl » :

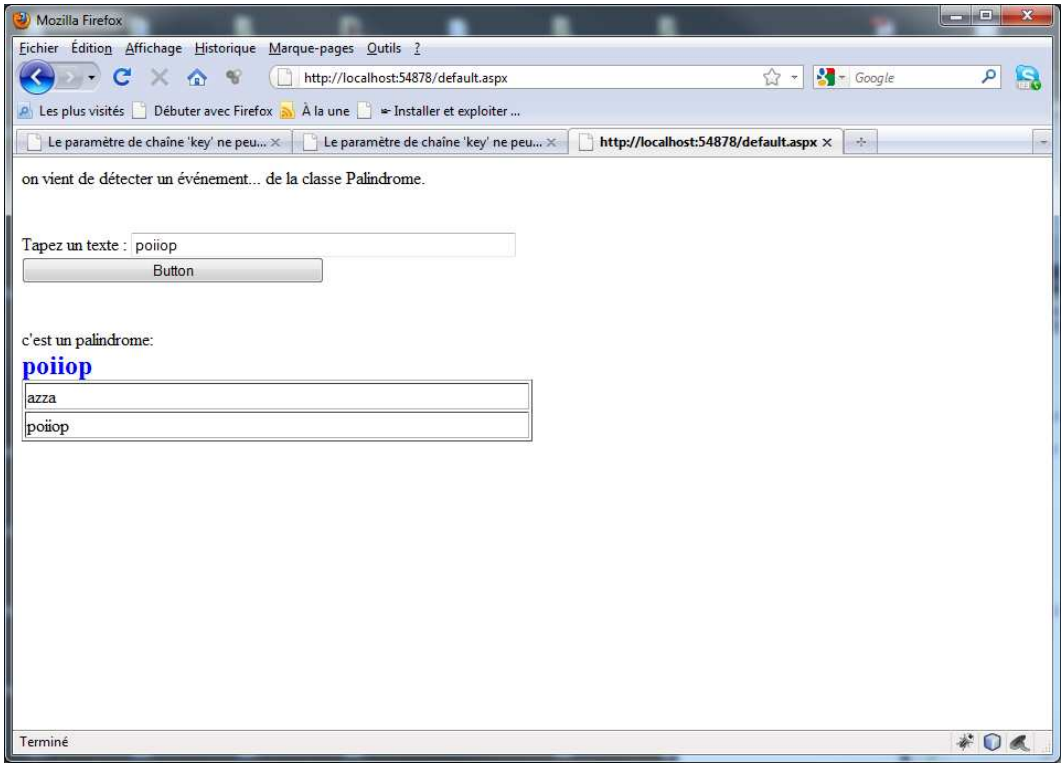
```
       protected void RenderPalindromInTable (HtmlTextWriter output)
       {
            output.AddAttribute(HtmlTextWriterAttribute.Width, "50%");
            output.AddAttribute(HtmlTextWriterAttribute.Border, "1");
            output.RenderBeginTag(HtmlTextWriterTag.Table);

            foreach (string s in this.SavePalindrome)
            {
               output.RenderBeginTag(HtmlTextWriterTag.Tr);
               output.AddAttribute(HtmlTextWriterAttribute.Align, "left");
               output.RenderBeginTag(HtmlTextWriterTag.Td);
```

```
            output.Write(s);

            output.RenderEndTag();  // fin </td
            output.RenderEndTag();  // fin </tr

        }
        output.RenderEndTag();  // fin </tableau
    }
```

Modifier la méhode RenderContents en ajoutant à la fin un appel à `RenderPalindromInTable`

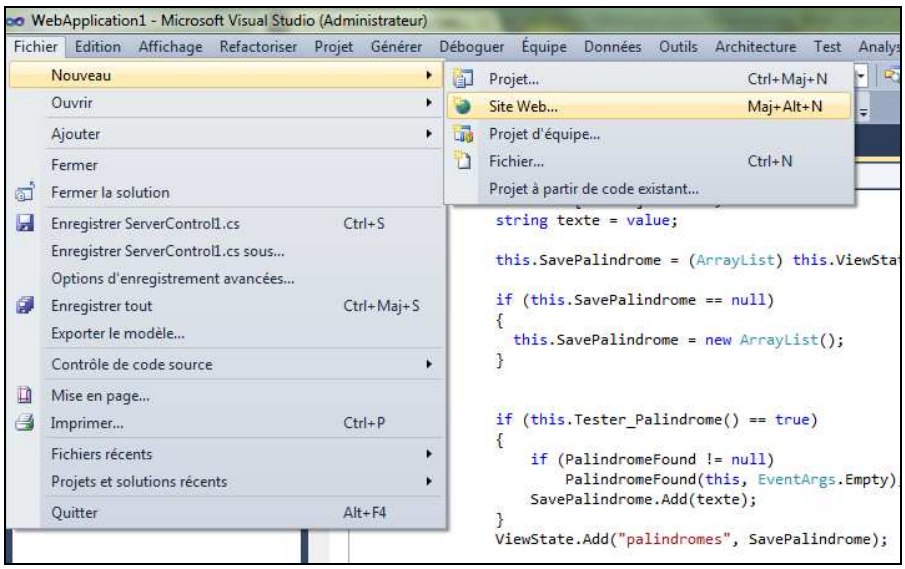
```
    protected override void RenderContents(HtmlTextWriter output)
    {
        if (this.Tester_Palindrome() == true)
        {
            output.Write("c'est un palindrome: <br>");
            output.Write("<FONT size=5 color=blue>");
            output.Write("<B>");
            output.Write(this.Text);
            output.Write("</B>");
            output.Write("</FONT>");
        }
        else
        {
            output.Write("ce n'est pas un palindrome: <br>");
            output.Write("<FONT size=5 color=red>");
            output.Write("<B>");
            output.Write(this.Text);
            output.Write("</B>");
            output.Write("</FONT>");
        }

        output.Write("<br>");
        RenderPalindromInTable(output);

    }
```

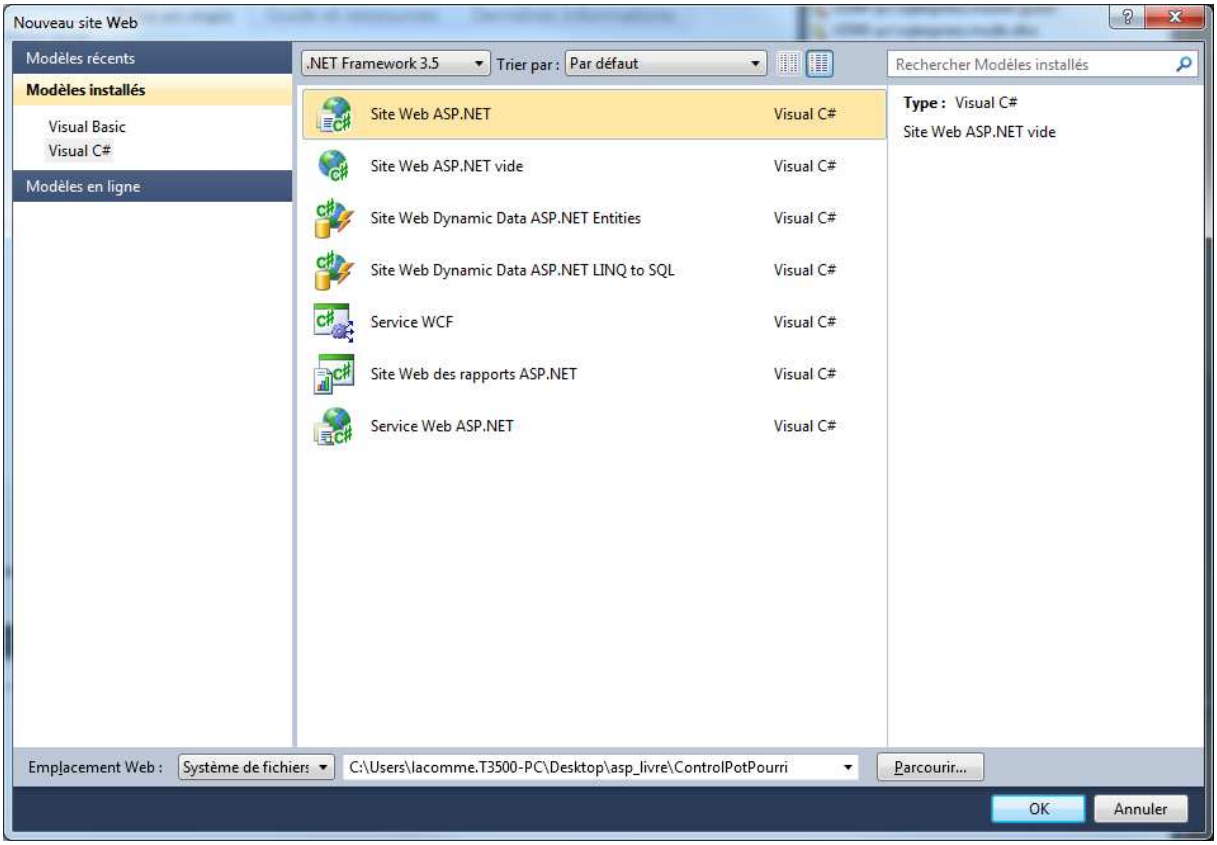
Ceci donne à l'exécution :



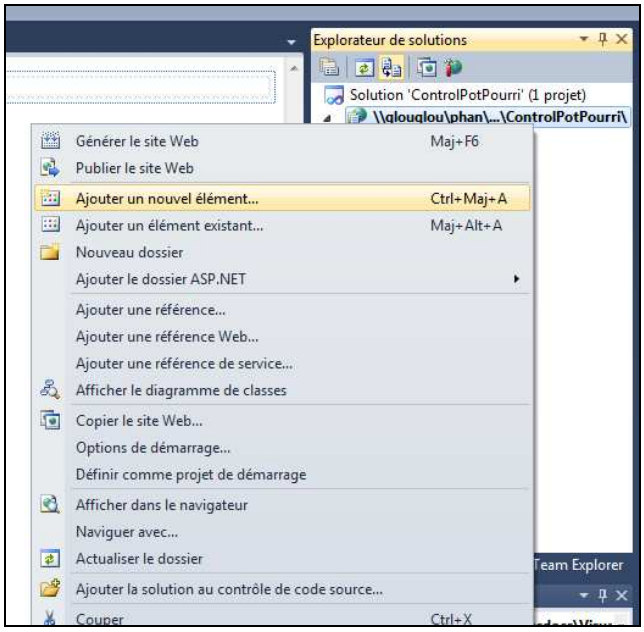
6. Les controles de validation

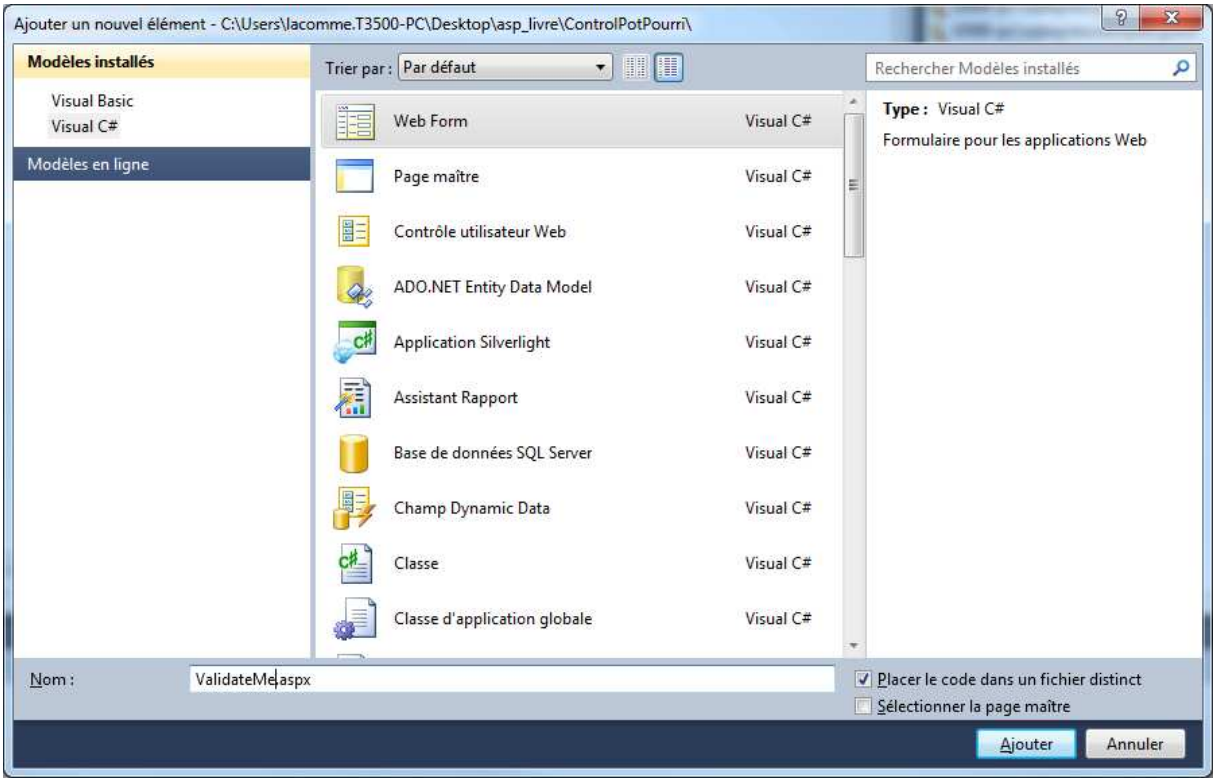
6.1) Créer un projet « site web » nommé **ControlPotPourri**



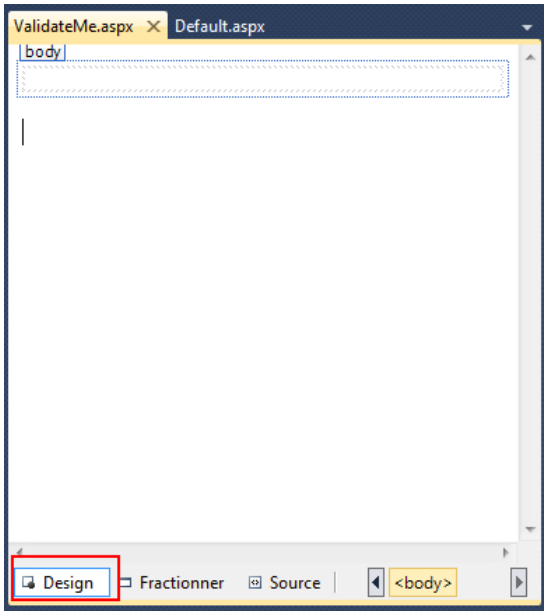


6.2) Ajouter un formulaire nommé **ValidateMe.aspx**





Passez en mode **Design**

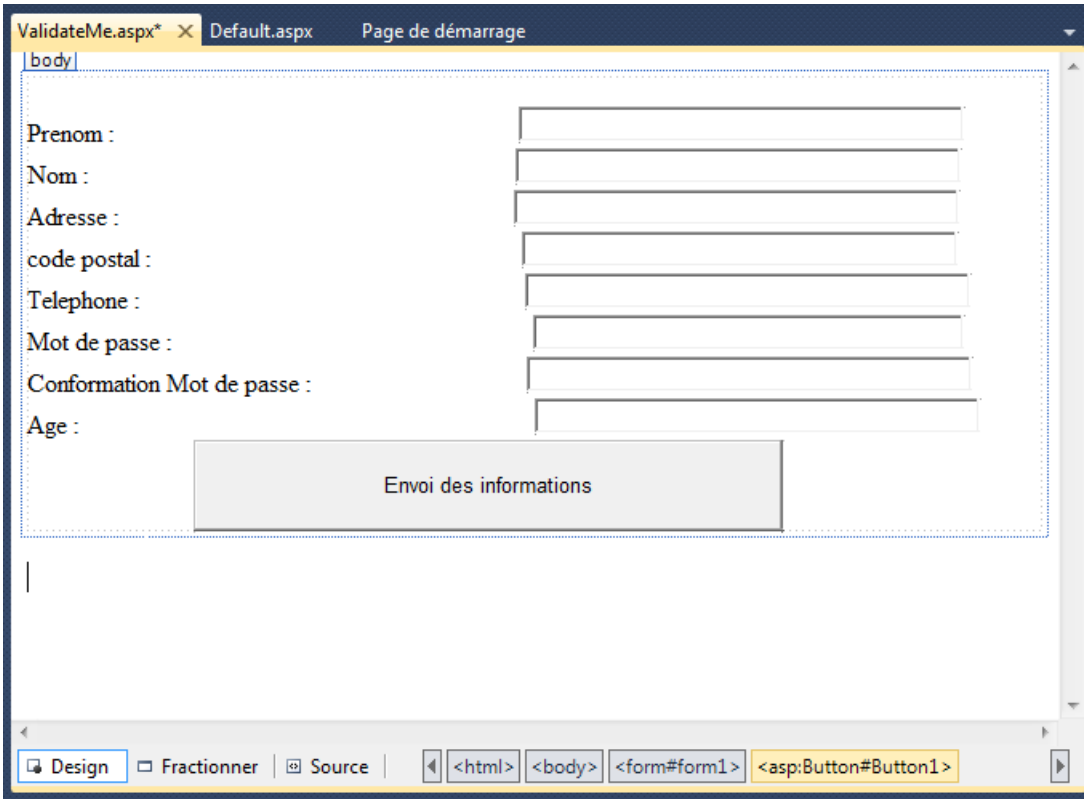


Dans un premier temps copier le code entre les balises <body> </body> dans votre code.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ValidateMe.aspx.cs"
Inherits="ValidateMe" %>

<!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></title>
```

```
</head>
<body>
    <form id="form1" runat="server">


    <br />
    <asp:Label ID="Label1" runat="server" Text=" Prenom :"></asp:Label>
    <asp:TextBox ID="TextBox1" runat="server" Width="278px"
        style="margin-left: 253px"></asp:TextBox>

    <br />
    <asp:Label ID="Label2" runat="server" Text=" Nom :"></asp:Label>
    <asp:TextBox ID="TextBox2" runat="server" style="margin-left: 267px"
        Width="278px"></asp:TextBox>

    <br />
   <asp:Label ID="Label3" runat="server" Text=" Adresse :"></asp:Label>
    <asp:TextBox ID="TextBox3" runat="server" Width="278px"
        style="margin-left: 247px"></asp:TextBox>

    <br />
    <asp:Label ID="Label4" runat="server" Text=" code postal :"></asp:Label>
    <asp:TextBox ID="TextBox4" runat="server" style="margin-left: 232px"
        Width="272px"></asp:TextBox>

    <br />
    <asp:Label ID="Label5" runat="server" Text=" Telephone : "></asp:Label>
    <asp:TextBox ID="TextBox5" runat="server" Width="278px"
        style="margin-left: 237px"></asp:TextBox>

    <br />
    <asp:Label ID="Label6" runat="server" Text=" Mot de passe : "></asp:Label>
    <asp:TextBox ID="TextBox6" runat="server" Width="269px"
        style="margin-left: 222px"></asp:TextBox>

    <br />
    <asp:Label ID="Label7" runat="server" Text=" Conformation Mot de passe :"></asp:Label>
    <asp:TextBox ID="TextBox7" runat="server" Width="278px"
        style="margin-left: 135px"></asp:TextBox>

    <br />
    <asp:Label ID="Label8" runat="server" Text=" Age :"></asp:Label>
    <asp:TextBox ID="TextBox8" runat="server" style="margin-left: 285px"
        Width="278px"></asp:TextBox>
    <br />
    <asp:Button ID="Button1" runat="server" Height="57px"
        style="margin-left: 104px" Text="Envoi des informations" Width="368px" />
    </form>
</body>
</html>
```

Cela vous donnera le formulaire suivant :



Choisissez ensuite les ID et évitez les noms par défaut …

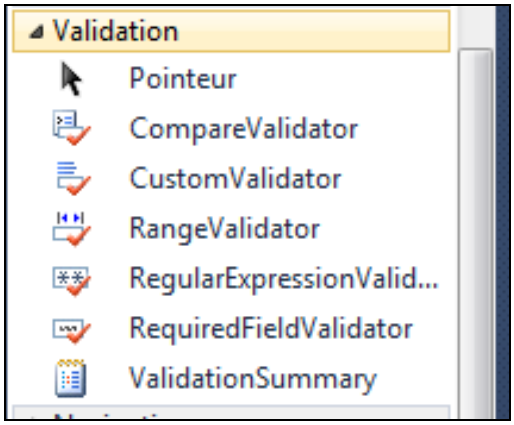
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ValidateMe.aspx.cs"
Inherits="ValidateMe" %>

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


    <br />
    <asp:Label ID="Label1" runat="server" Text=" Prenom :"></asp:Label>
    <asp:TextBox ID="TextBoxFirstNBame" runat="server" Width="278px"
        style="margin-left: 253px"></asp:TextBox>

    <br />
    <asp:Label ID="Label2" runat="server" Text=" Nom :"></asp:Label>
    <asp:TextBox ID="TextBoxLastName" runat="server" style="margin-left: 267px"
        Width="278px"></asp:TextBox>

    <br />
   <asp:Label ID="Label3" runat="server" Text=" Adresse :"></asp:Label>
    <asp:TextBox ID="TextBoxAdress" runat="server" Width="278px"
        style="margin-left: 247px"></asp:TextBox>

    <br />
    <asp:Label ID="Label4" runat="server" Text=" code postal :"></asp:Label>
```

```
    <asp:TextBox ID="TextBoxPostalCode" runat="server" style="margin-left: 232px"
        Width="272px"></asp:TextBox>

    <br />
    <asp:Label ID="Label5" runat="server" Text=" Telephone : "></asp:Label>
    <asp:TextBox ID="TextBoxPhone" runat="server" Width="278px"
        style="margin-left: 237px"></asp:TextBox>

    <br />
    <asp:Label ID="Label6" runat="server" Text=" Mot de passe : "></asp:Label>
    <asp:TextBox ID="TextBoxPassword" runat="server" Width="269px"
        style="margin-left: 222px"></asp:TextBox>

    <br />
    <asp:Label ID="Label7" runat="server" Text=" Conformation Mot de passe
:"></asp:Label>
    <asp:TextBox ID="TextBoxPasswordAgain" runat="server" Width="278px"
        style="margin-left: 135px"></asp:TextBox>

    <br />
    <asp:Label ID="Label8" runat="server" Text=" Age :"></asp:Label>
    <asp:TextBox ID="TextBoxAge" runat="server" style="margin-left: 285px"
        Width="278px"></asp:TextBox>
    <br />
    <asp:Button ID="Button1" runat="server" Height="57px"
        style="margin-left: 104px" Text="Envoi des informations" Width="368px" />
    </form>
</body>
</html>
```

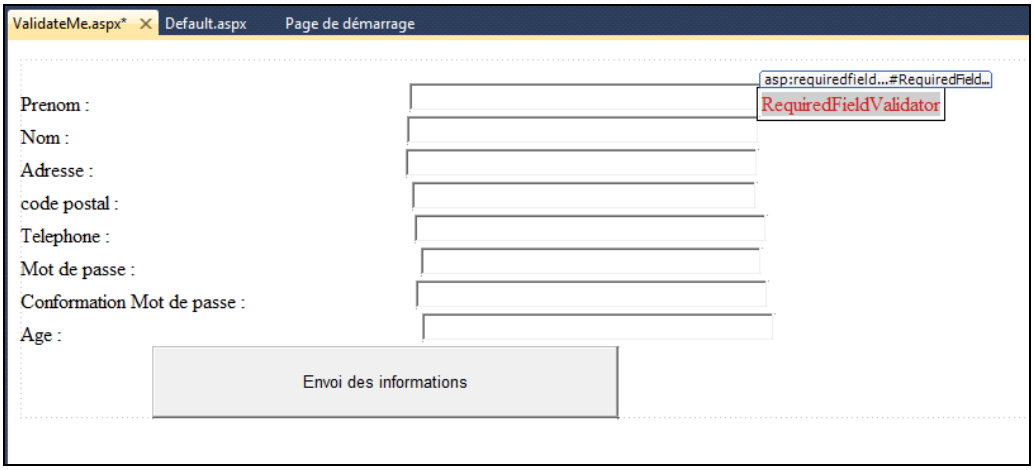
6.3) Ajout des validateurs

Ajouter un contrôle **RequiredFieldValidator** pour le prénom.

Dans la boite à outils, choisir la section **Validation**.

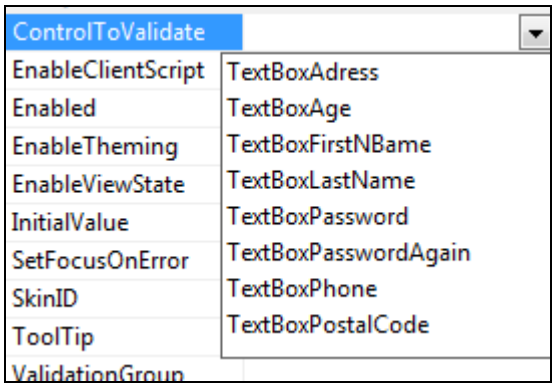


Poser le contrôle RequiredFiledValidator à droite du `TextBoxFirstNBame;`

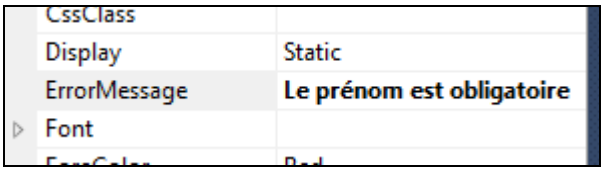


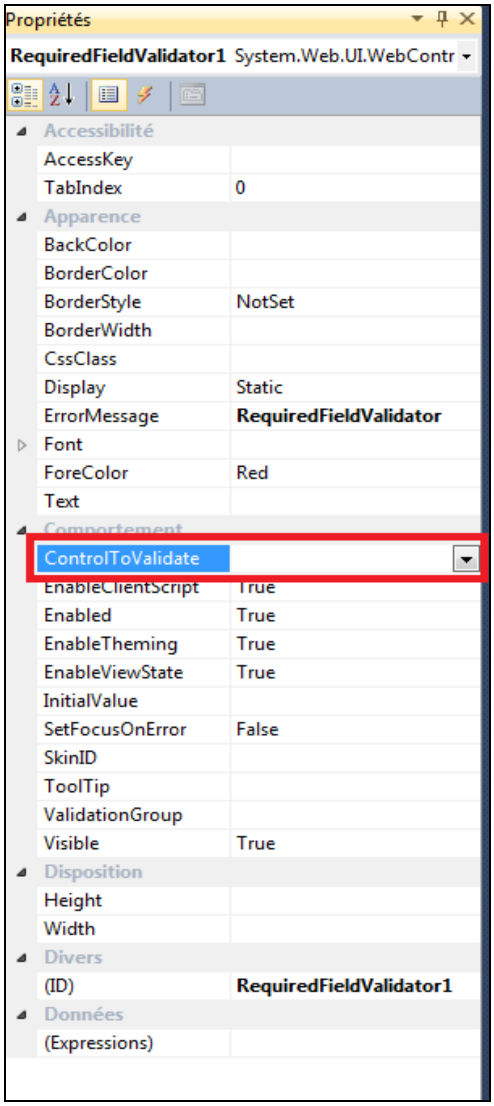
Dans les propriétés, choisir **ControlToValidate.**

Dans la liste déroulante, choisir **TextBoxFirstName.**

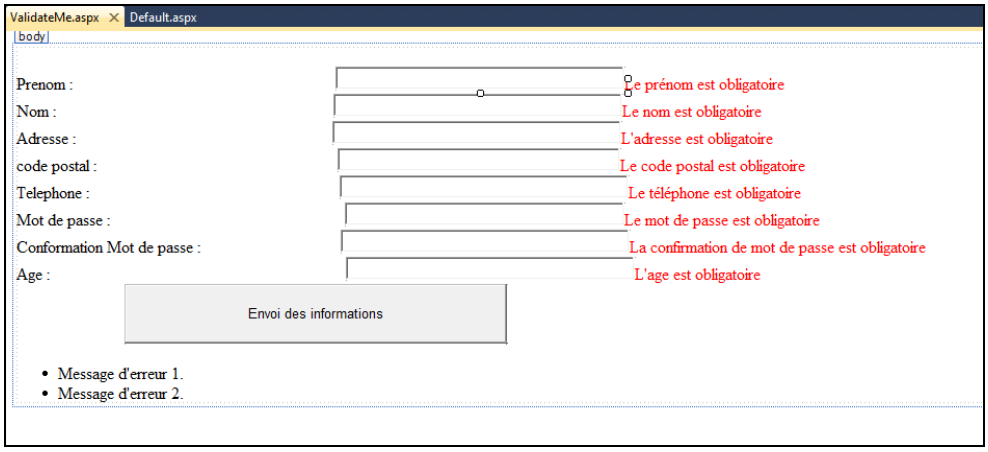


Modifier la propriété **ErrorMessage** en lui donnant un message clair.

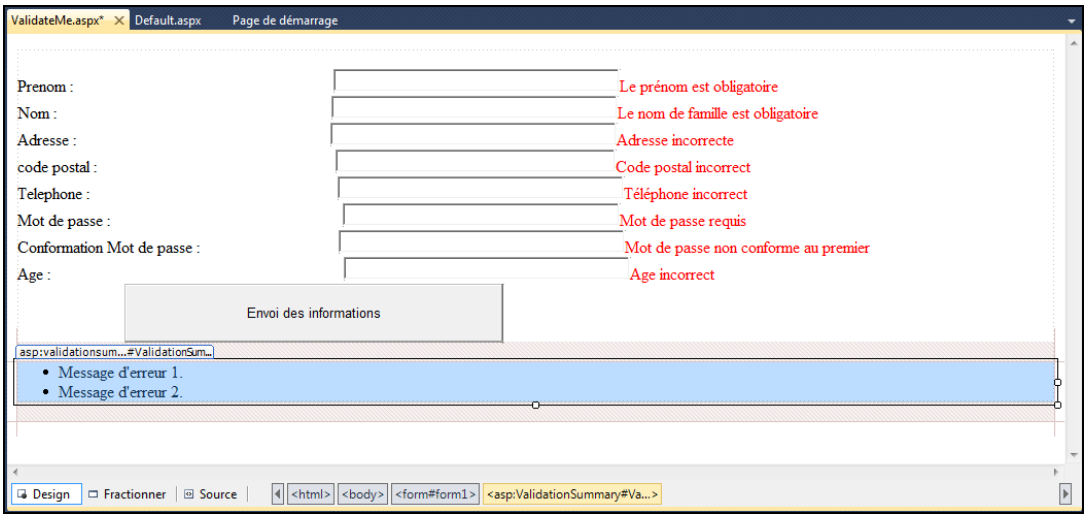




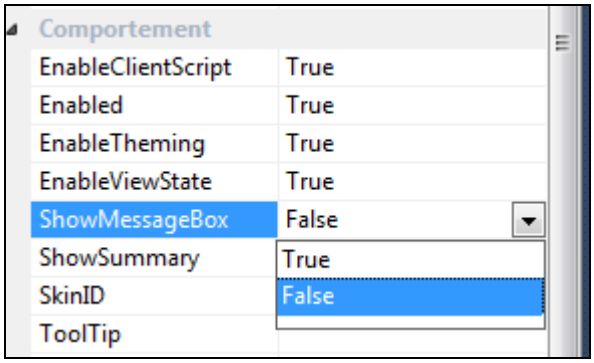
Réitérer l'opération pour le Nom, le code postal, le téléphone, le mot de passe, le second mot de passe, et l'age



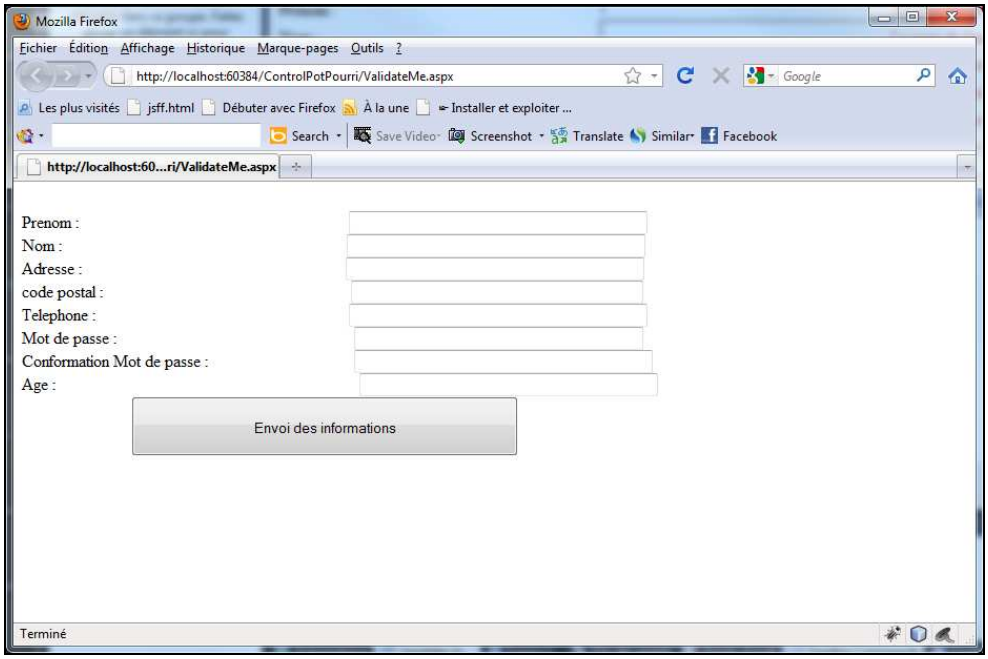
Ajouter un contrôle **ValidationSummary**.



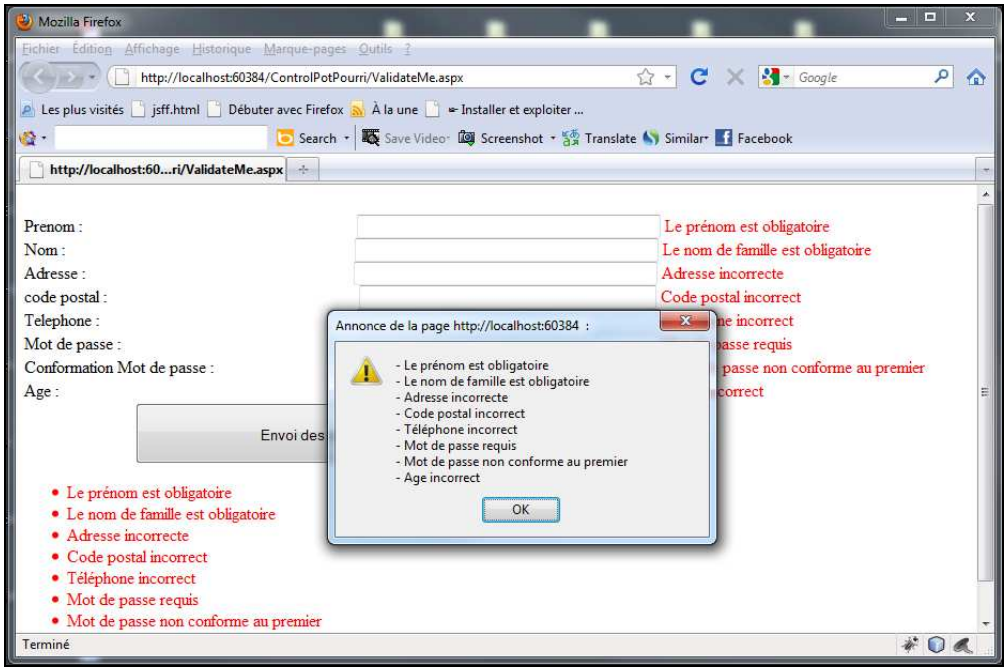
Mettre la propriété ShowMessageBow à true.



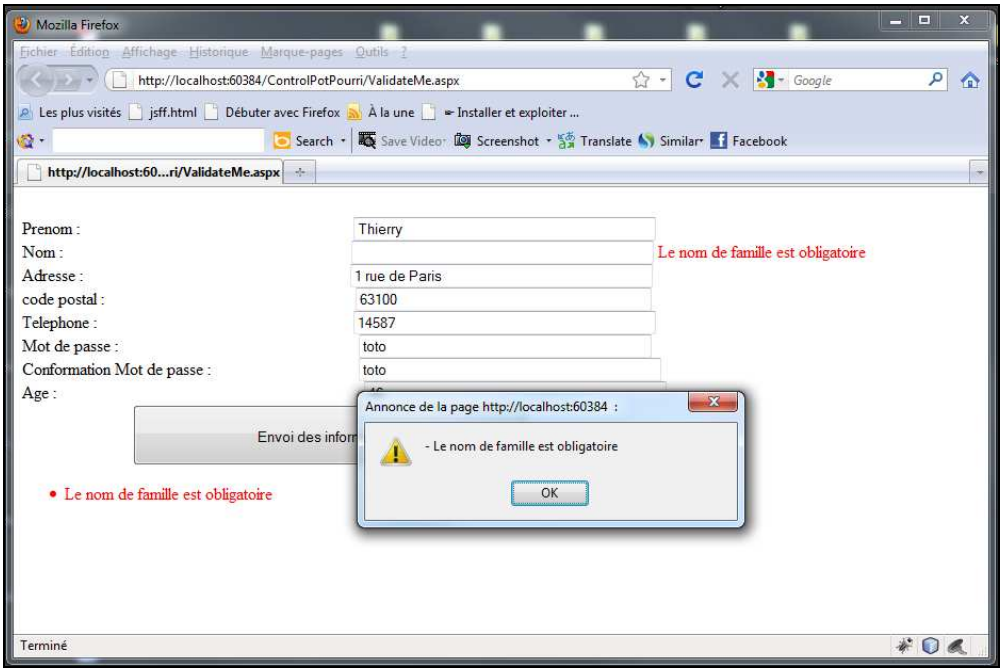
A l'exécution on obtient ceci :



Et les erreurs sont traitées comme ceci :



Ou encore ceci si une seule erreur apparait :



Remarquons que la validation au niveau de la saisie se fait du coté client et donc ne sollicite pas le réseau ni le serveur.

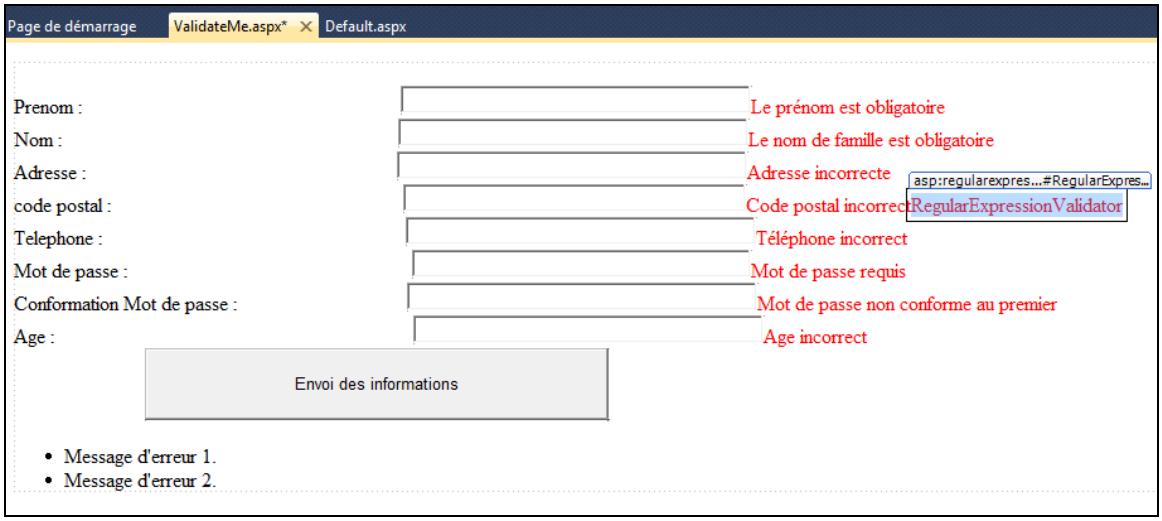
7) Validation des pages

7.1) Ajout d'une validation plus granulaire

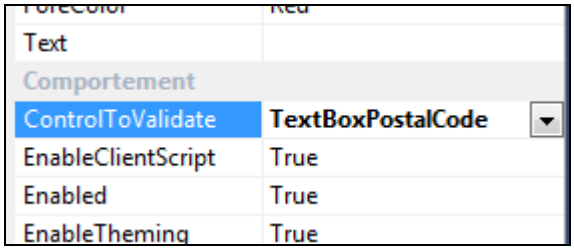
Nous allons revoir la validation au niveau du code postal en nous assurant que l'utilisateur à bien saisi un chiffre.

Pour cela nous allons utiliser un contrôle **RegularExpressionValidator**.

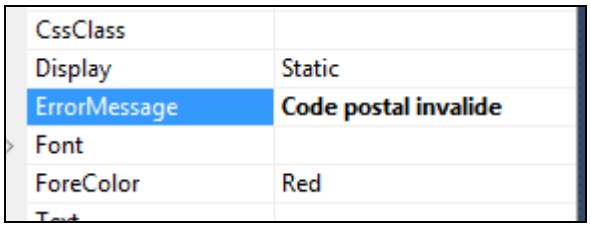
Poser ce contrôle à coté du champ Code Postal comme ceci :



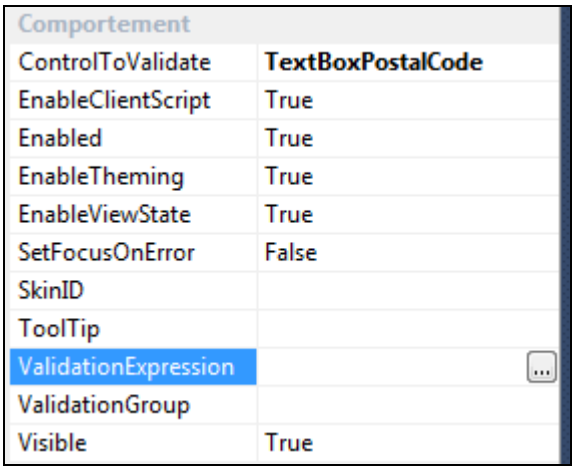
Dans les propriétés, modifier le ControlToValidate pour qu'il pointe sur le code postal.



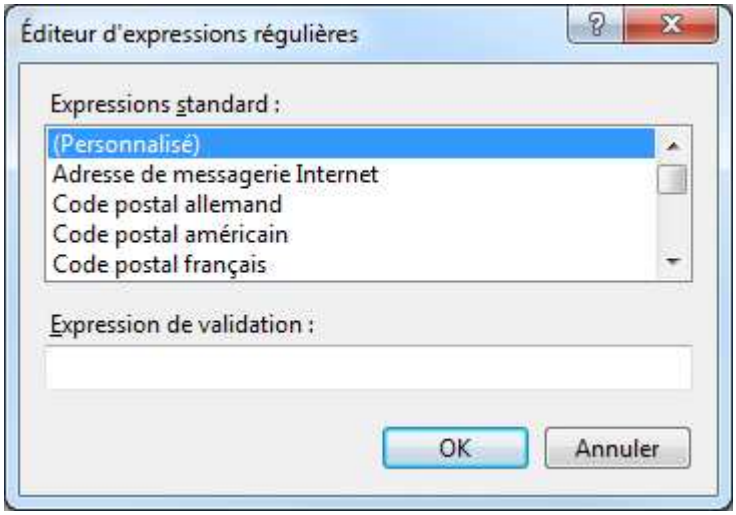
Modifier le champ ErrorMessage avec un message du type **"Code postal non valide…"**



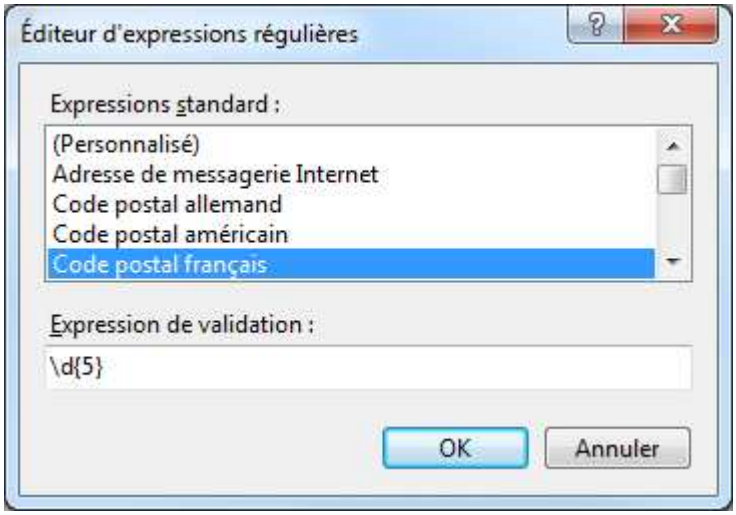
Choisissez ensuite **ValidationExpression**.



Ceci devrait faire apparaitre une nouvelle fenêtre…

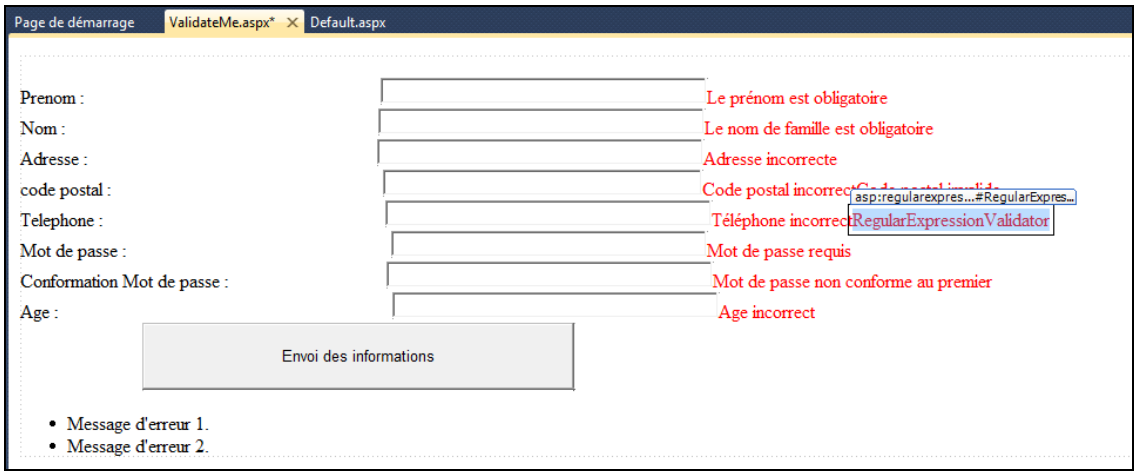


Choisissez code postal Français.

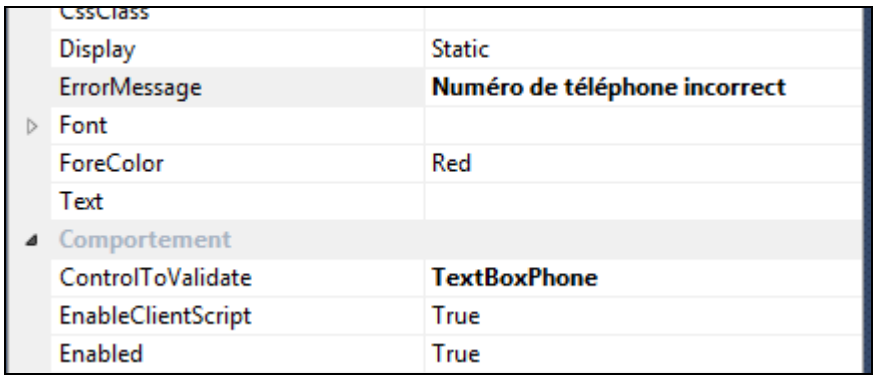


Nous allons ajouter un validateur d'expressions régulières sur le contrôle TextBoxPhone.

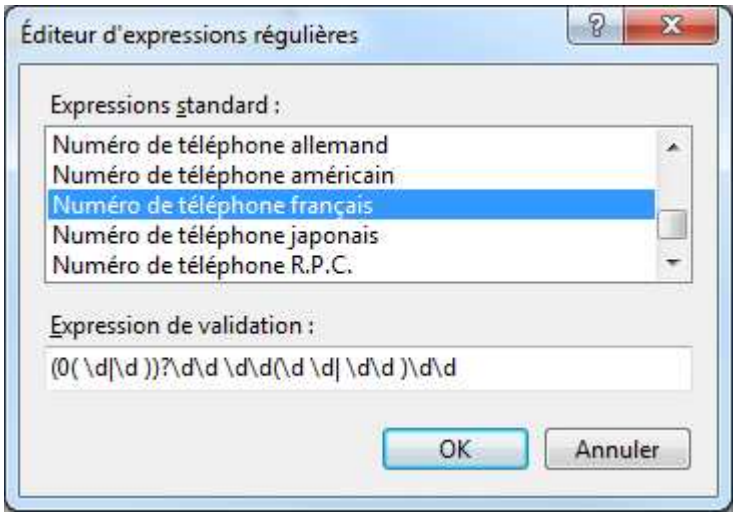
Pour cela nous allons utiliser un contrôle **RegularExpressionValidator** que nous posons à coté du champ de saisie du téléphone.



Comme précédemment, modifiez le champ **ControlToValidate** et le champ **ErrorMessage**.

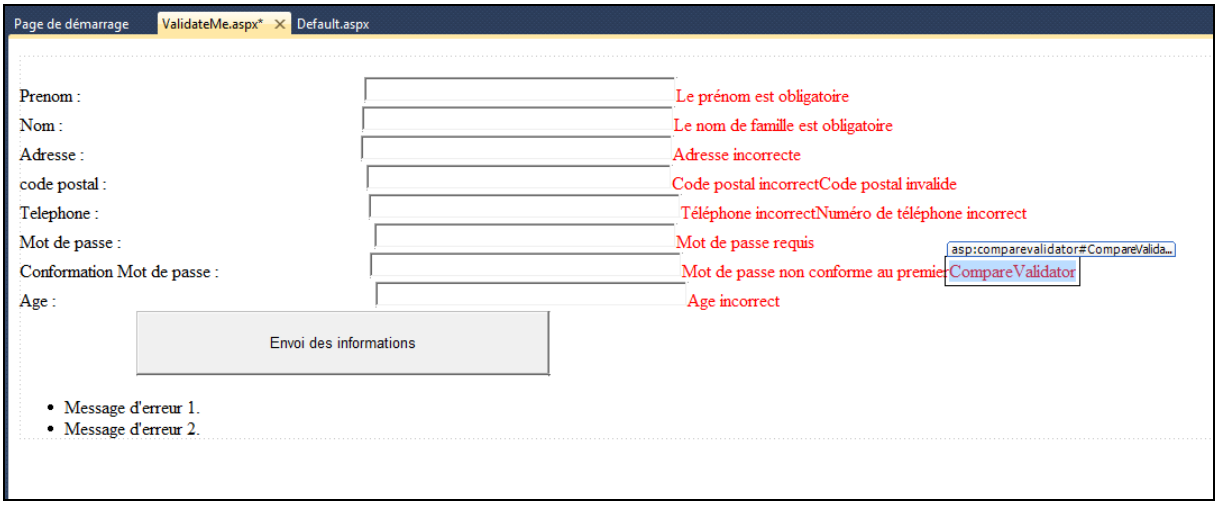


Choisisse esuite le champ ValidationExpression et choisissez numéro de téléphone Français.

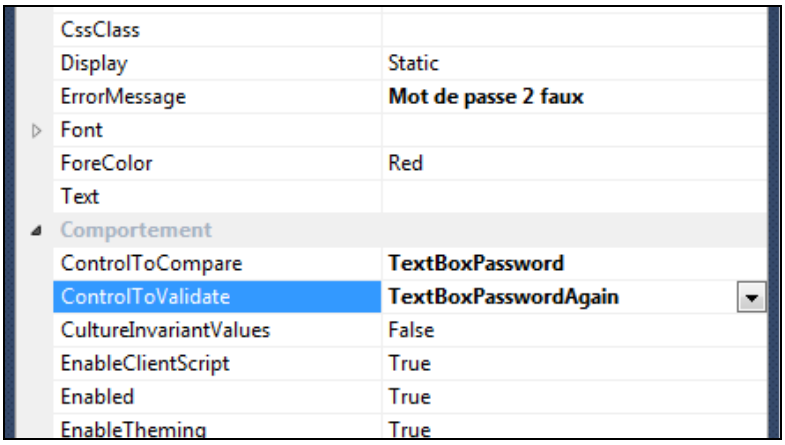


Nous allons ajouter un validateur d'expressions régulières sur le contrôle TextBoxPasswordAgain.

Pour cela nous allons utiliser un contrôle **CompareValidator** que nous posons à coté du champ de saisie du password.

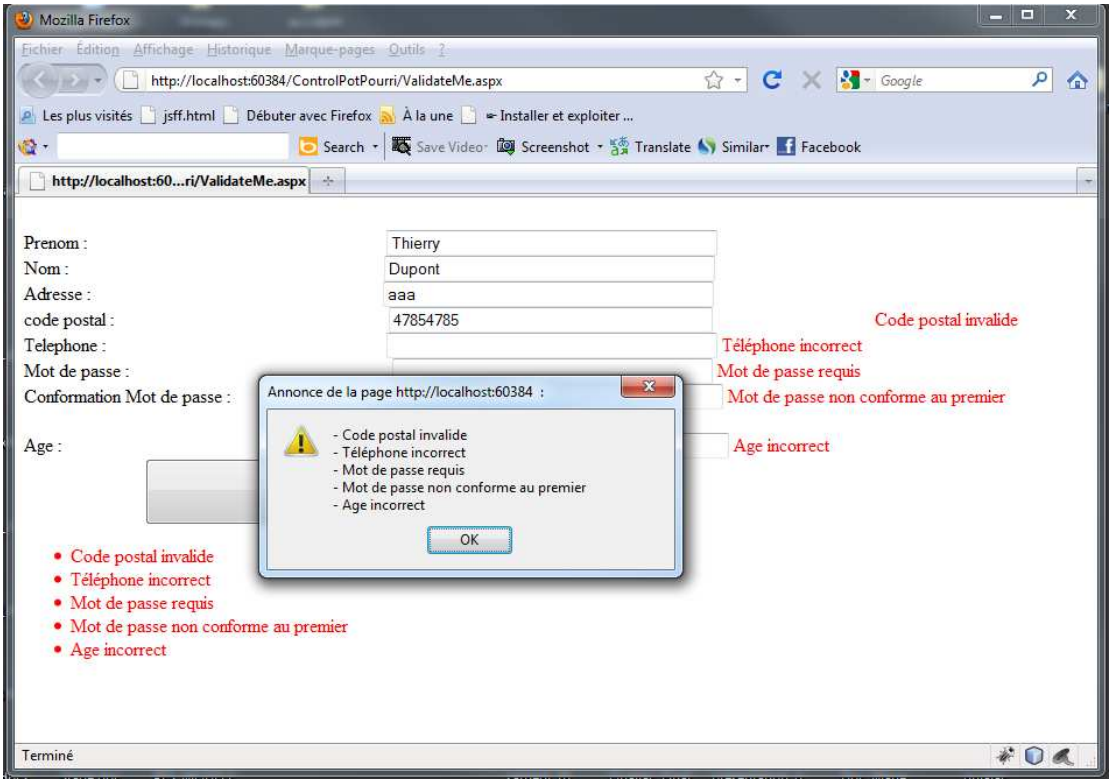


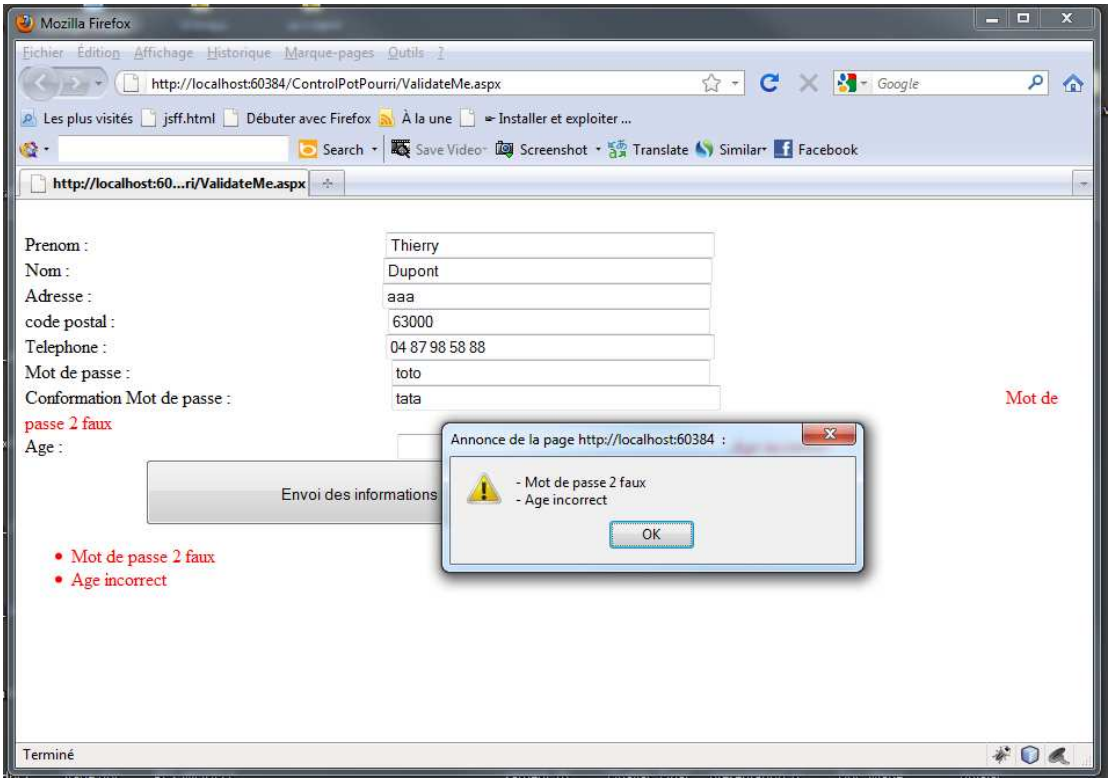
Modifiez le champ **ControlToCompare** et le champ **ErrorMessage**.



Le champ **ControlToCompare** donne la zone avec laquelle le contenu doit être comparé.

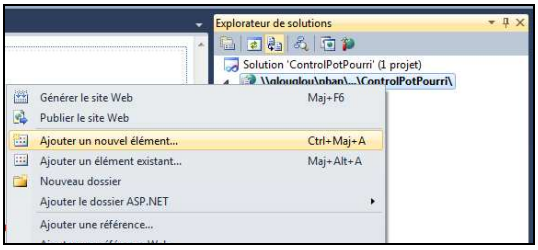
Ceci donne :

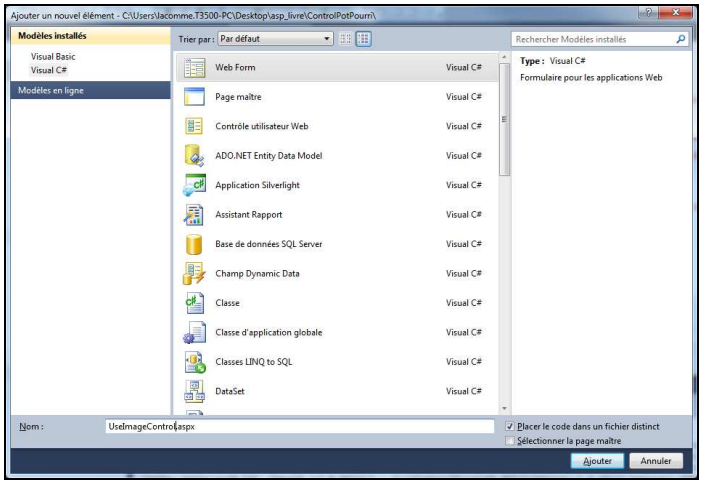




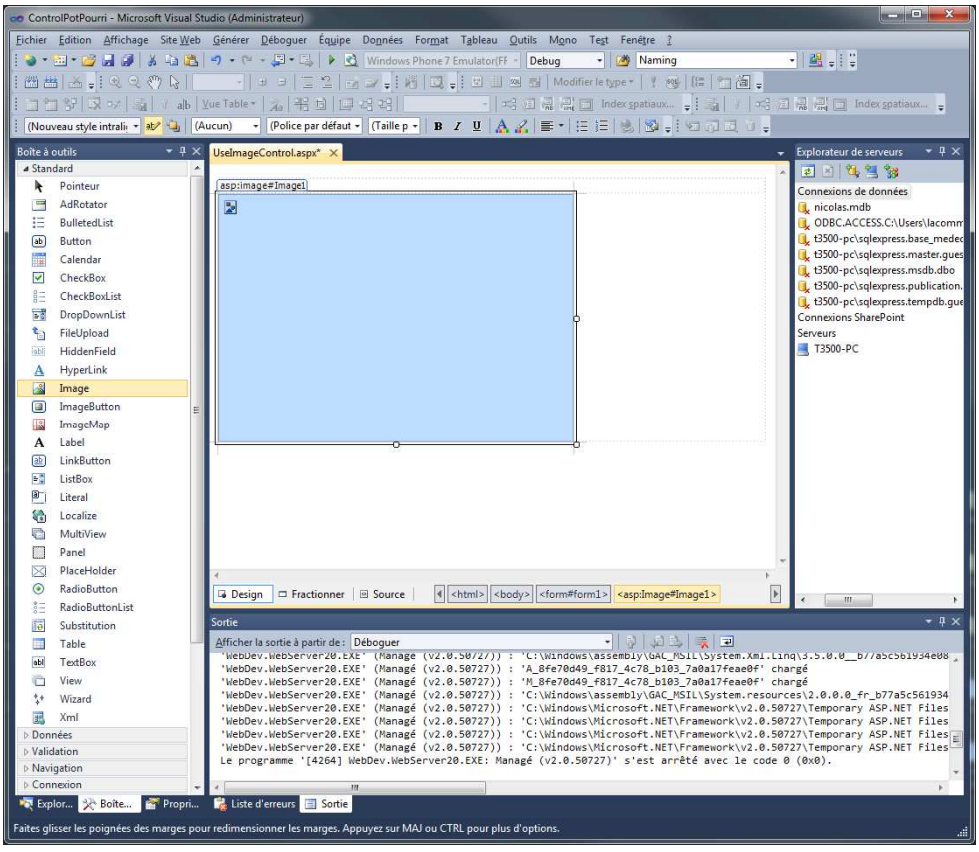
7.2) Contrôles basés sur des images

Ajoutez au projet un nouveau formulaire nommé : **UseImageControl.aspx**.

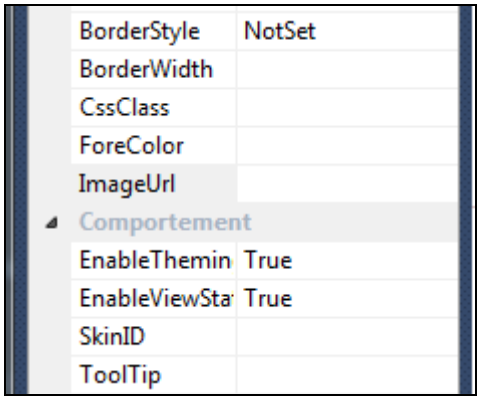




Posez sur la page un control **Image**.

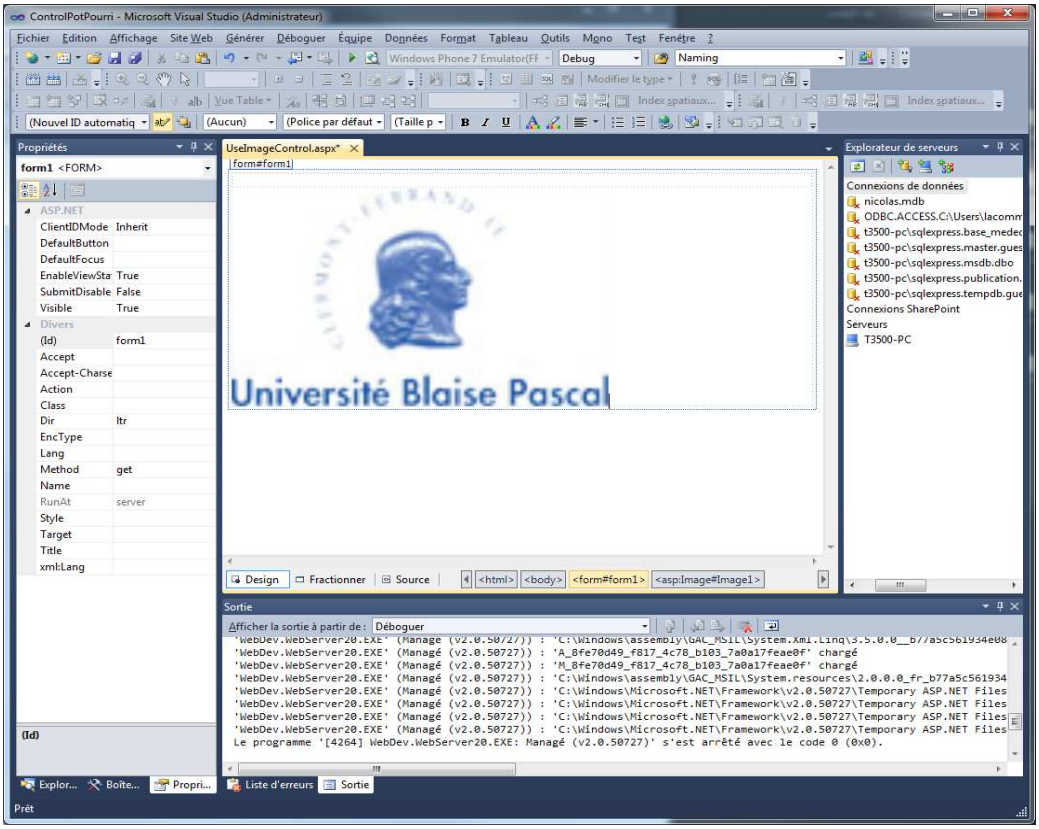


On s'intéresse au champ **ImageUrl**.

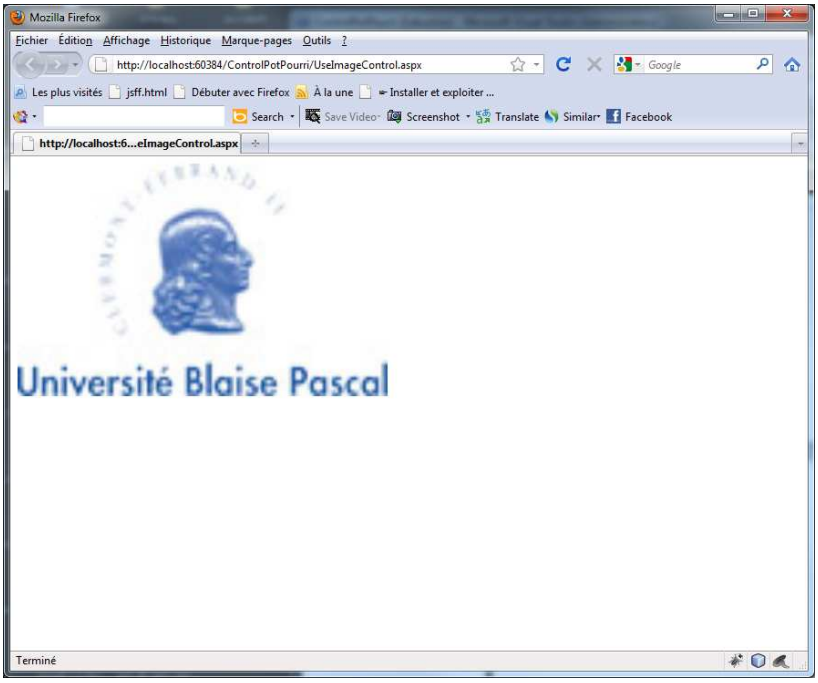


Choisissons par exemple : http://www.isima.fr/~lacomme/img/logo_UBP.jpg

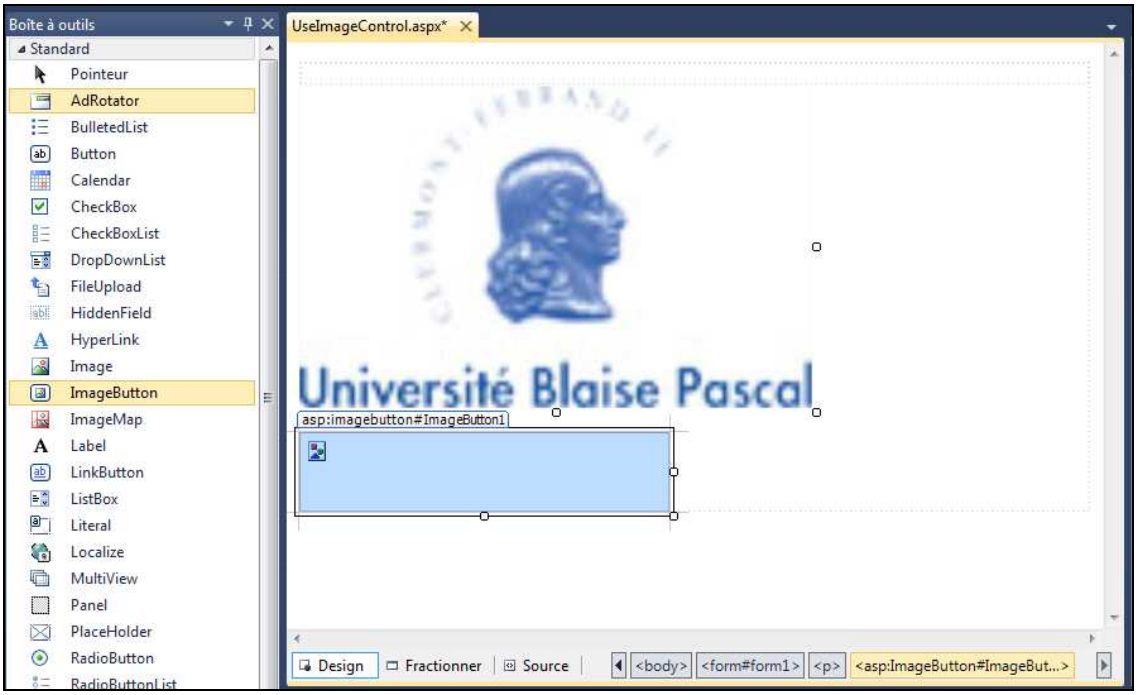
Vous devriez obtenir :



Ceci donne à l'exécution :

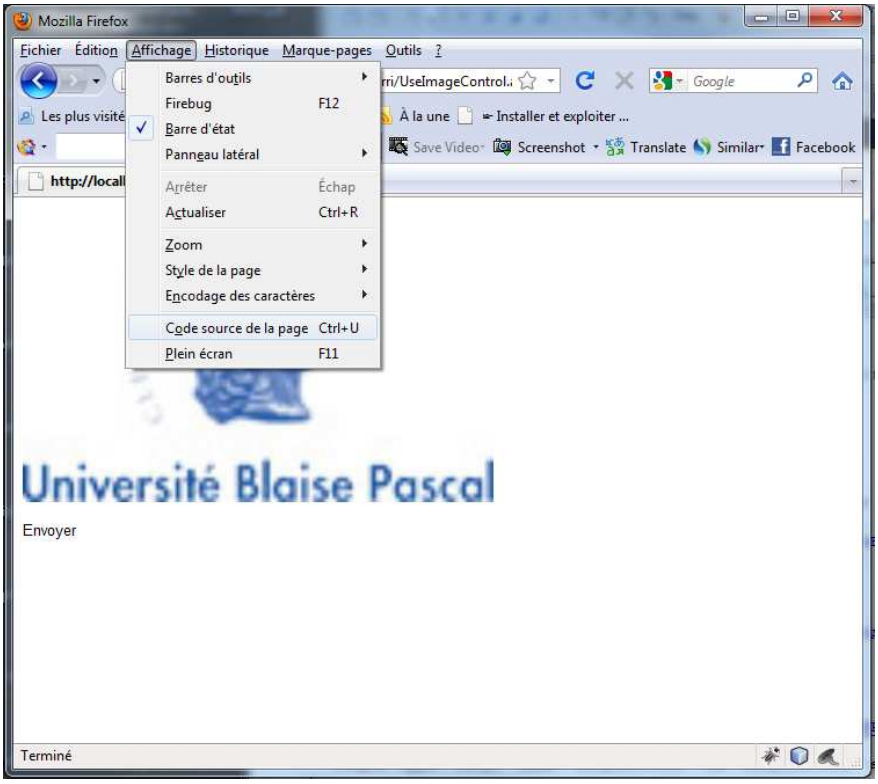


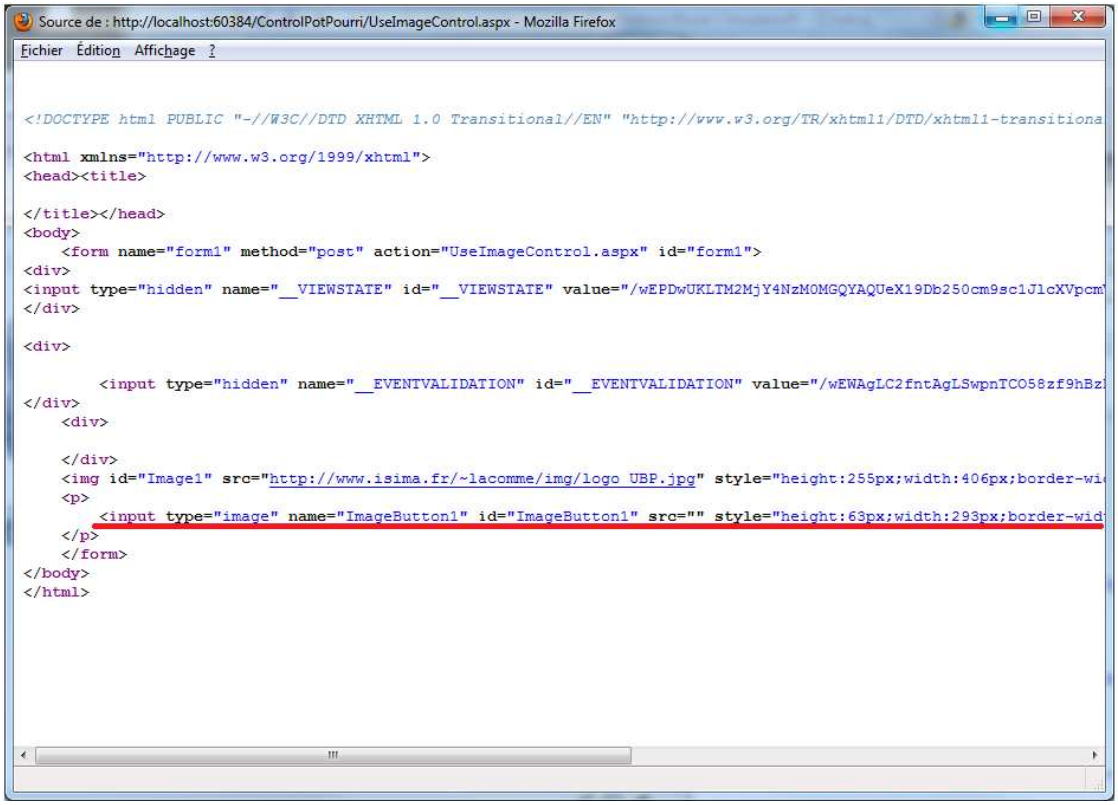
Ajoutons un **ImageButton** à la page.



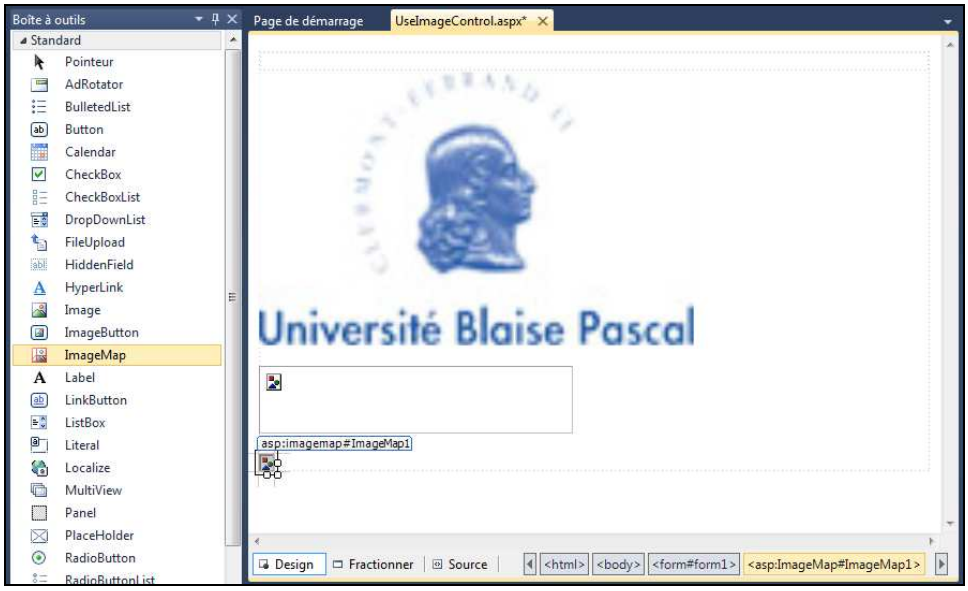
NB : Redémarrer le projet si besoin pour que le bouton apparait.

Exécuter et examiner le code HTML.



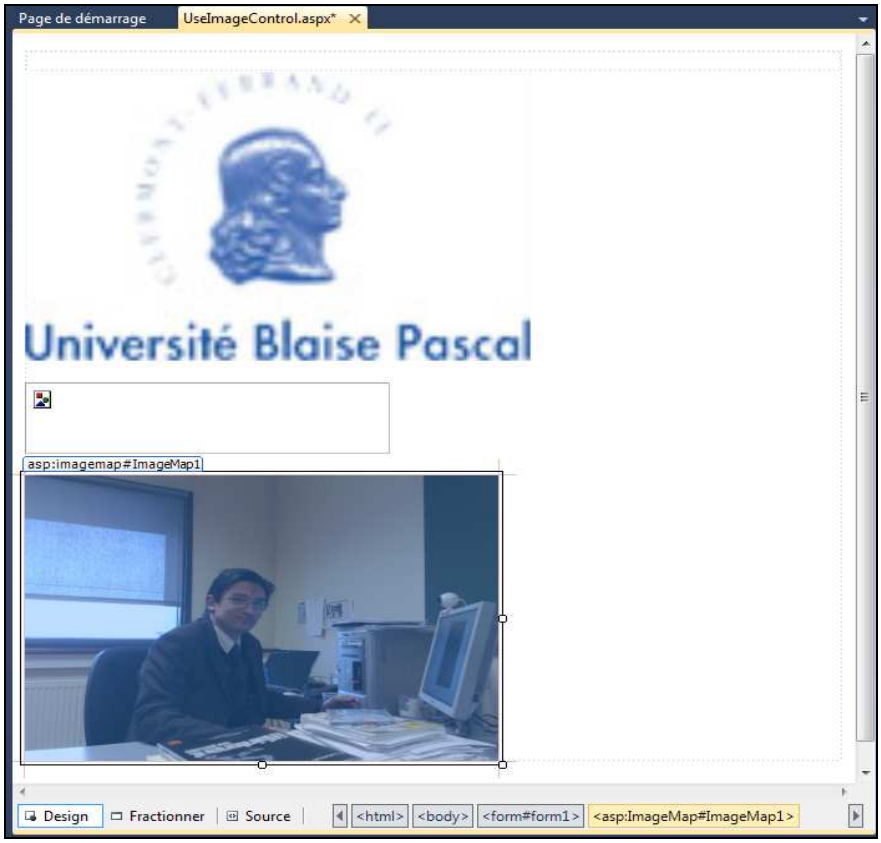


Ajoutons un **ImageMap** à la page.



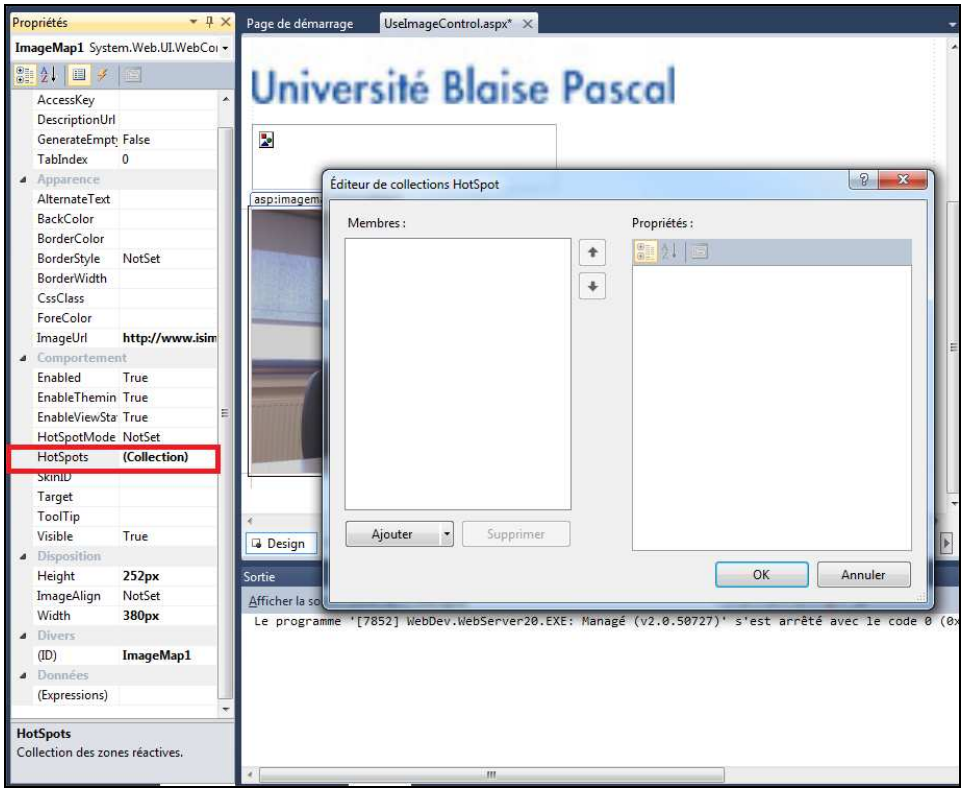
Il va nous permettre de définir des parties cliquables dans un bitmap.

Modifier la partie ImageUrl en prenant une image sur internet.
Par exemple : http://www.isima.fr/~lacomme/img/Bureau.JPG
Agrandissez un peu l'image pour obtenir une présentation comme celle-ci :

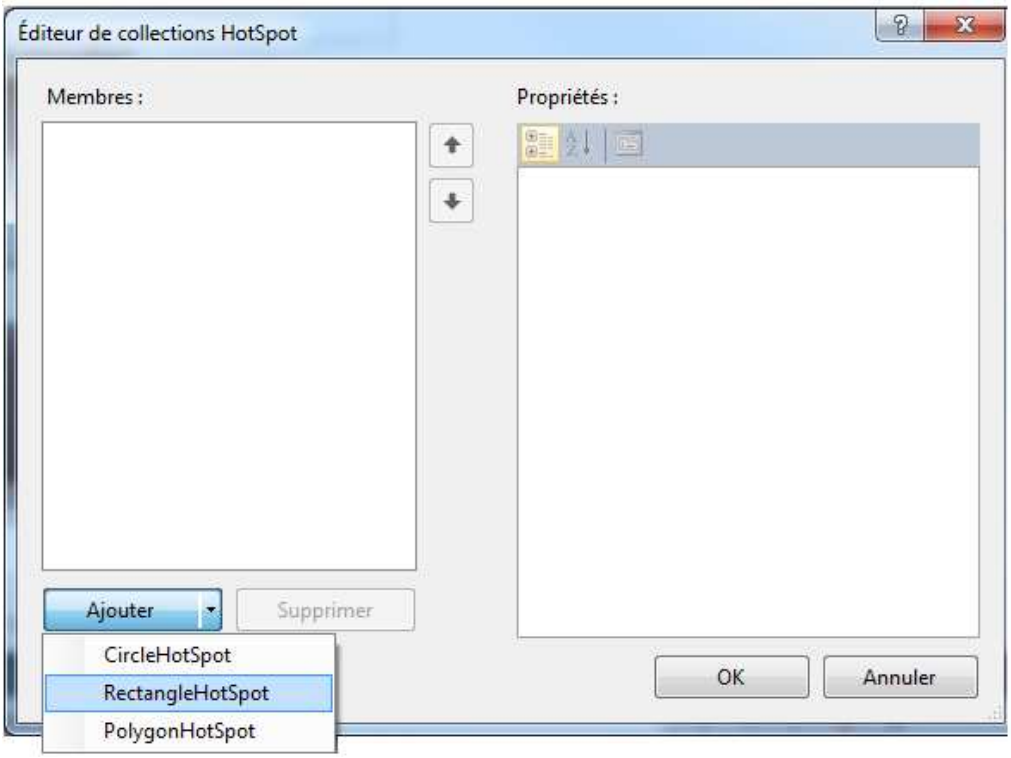


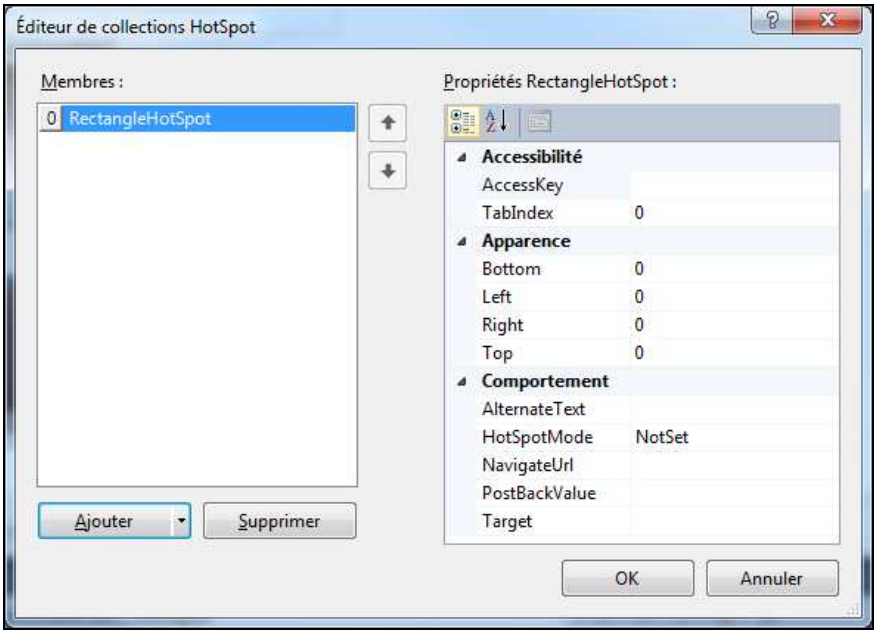
NB : Redémarrer le projet si besoin pour que le bouton apparait.

Cliquez sur **HotSpot**

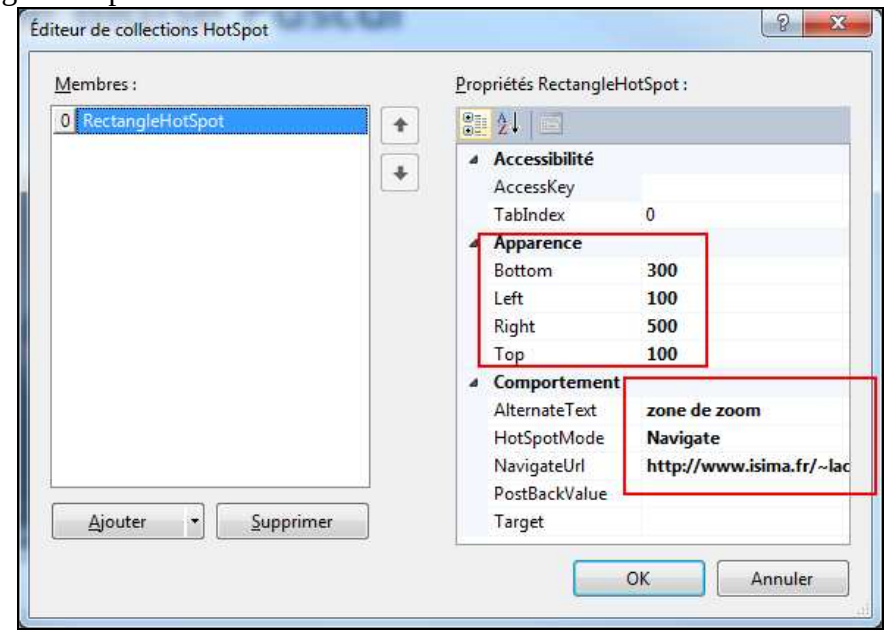


Choisir le bouton **Ajouter** puis choisir la création d'une zone rectangulaire.

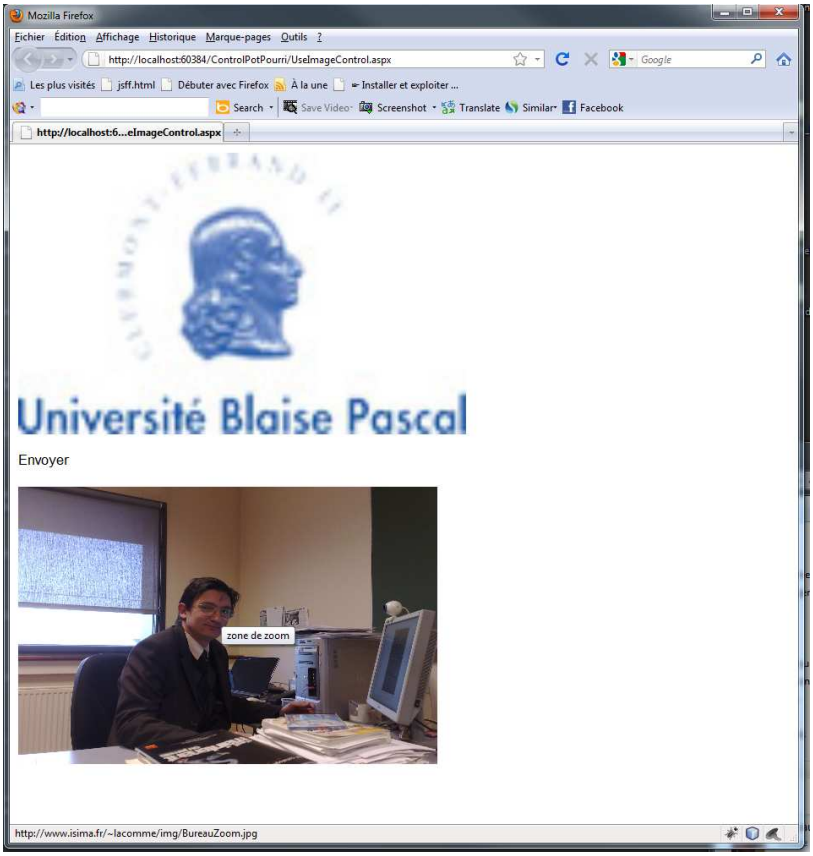




Faire les modifications comme suite et dans « NavigateUrl » mettre :
http://www.isima.fr/~lacomme/img/BureauZoom.jpg
Tester aussi différentes paramètre de « Apparence » afin que la cadre cliquable soit sur la tête du personnage de la photo.

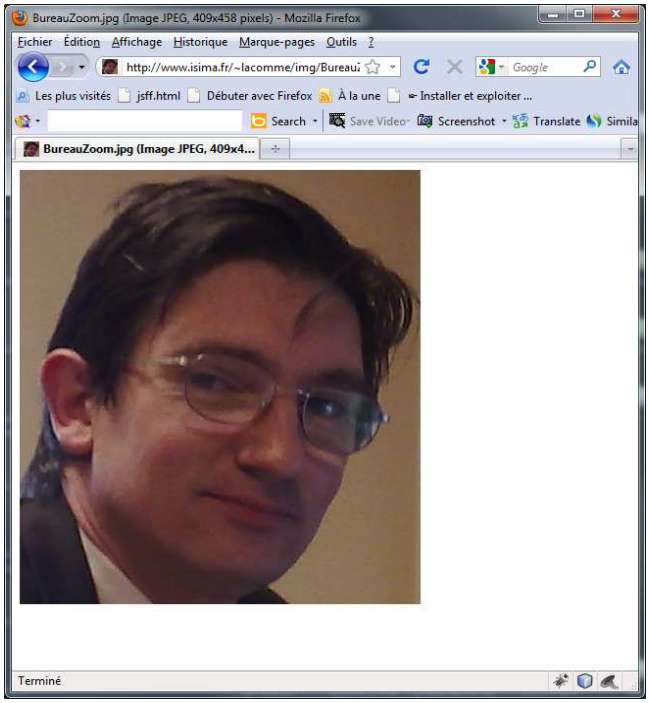


Ceci donne à l'exécution :



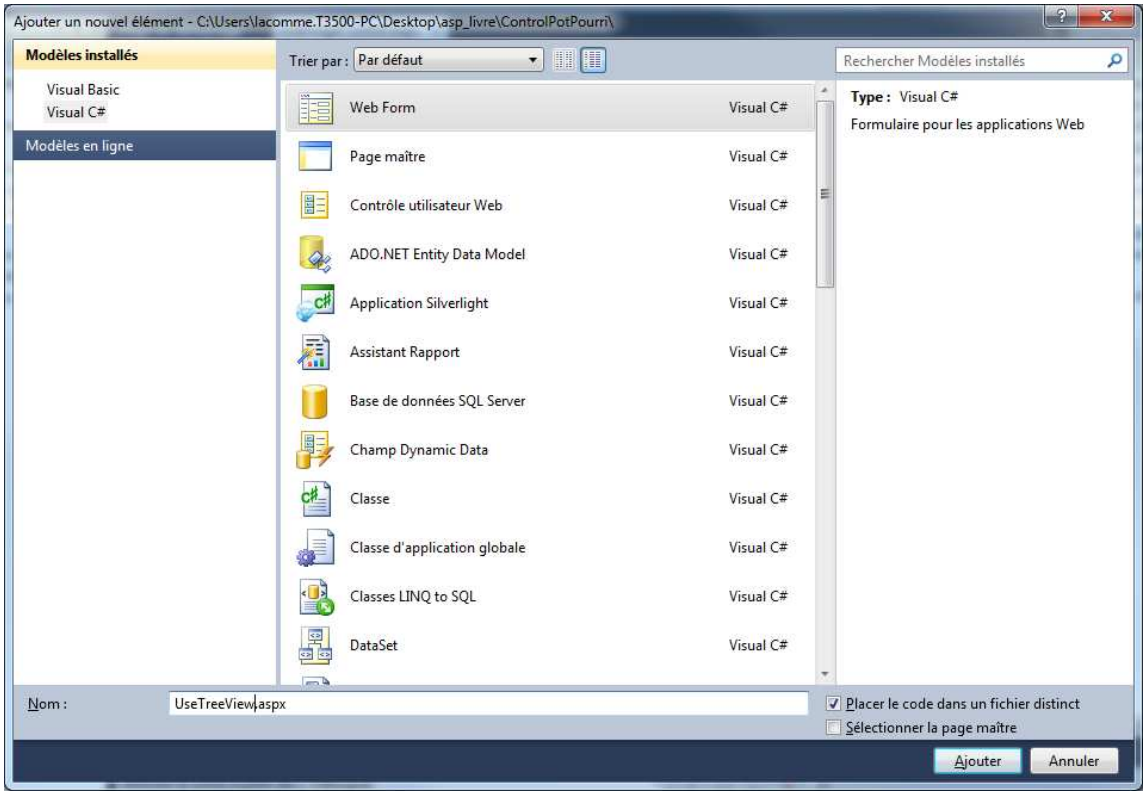
NB : Penser à redémarrer le projet si les modifications ne semblent pas être prises en compte.

Et après le click :

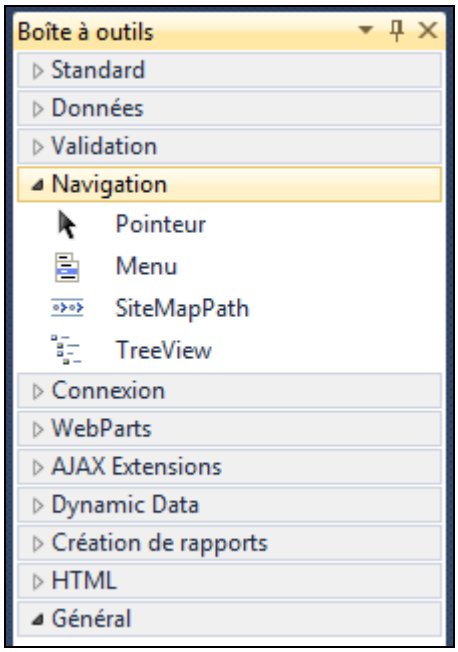


7.3) Contrôles basés sur des **TreeView**

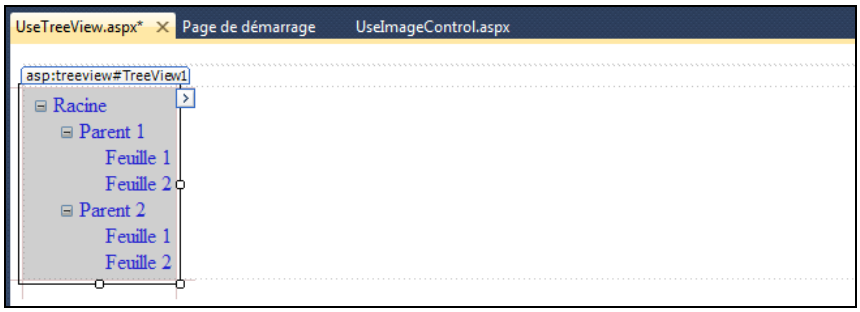
Ajoutons un formulaire nommé **UseTreeView**.



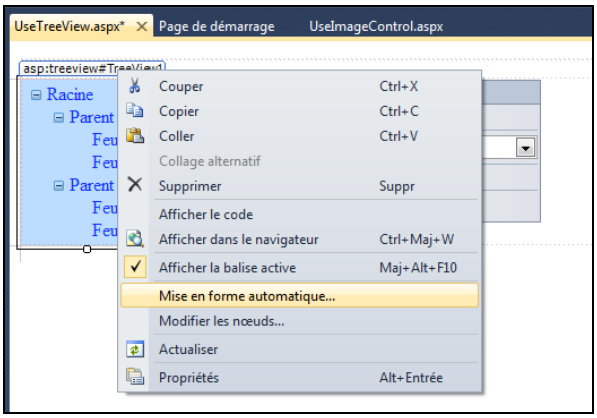
Choisissez un **TreeView** dans la section **Navigation**.



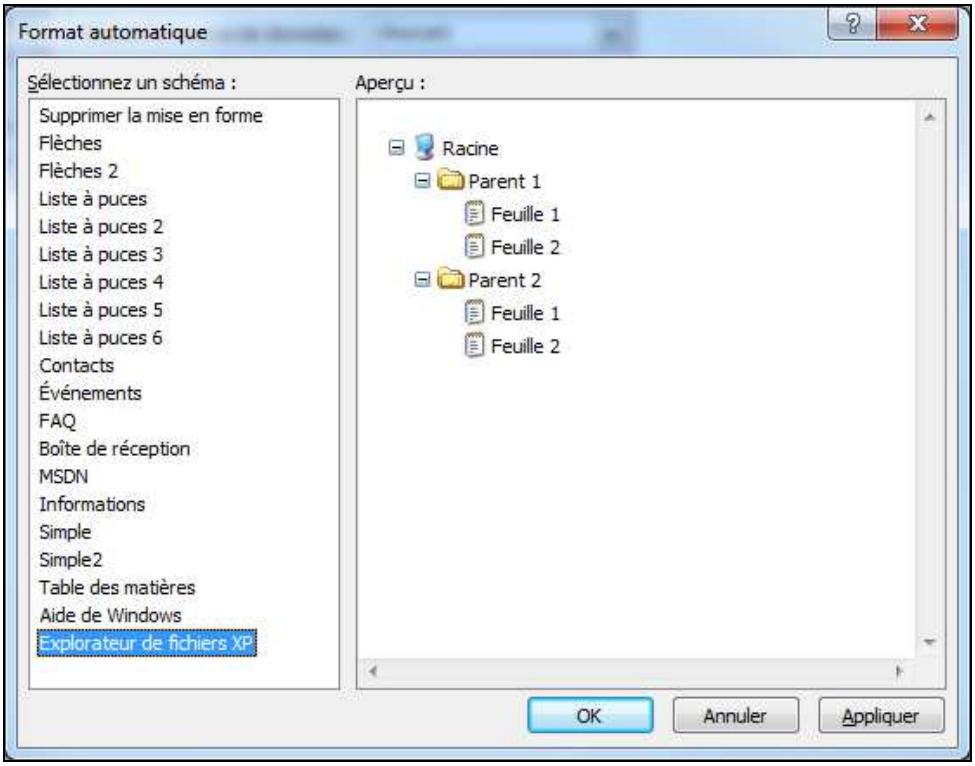
Ceci devrait donner :



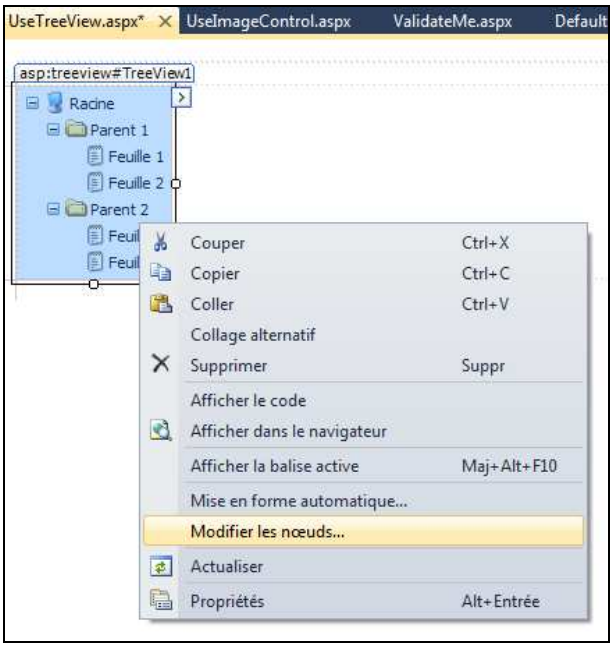
Faire un click droit et choisir **Mise en forme automatique**



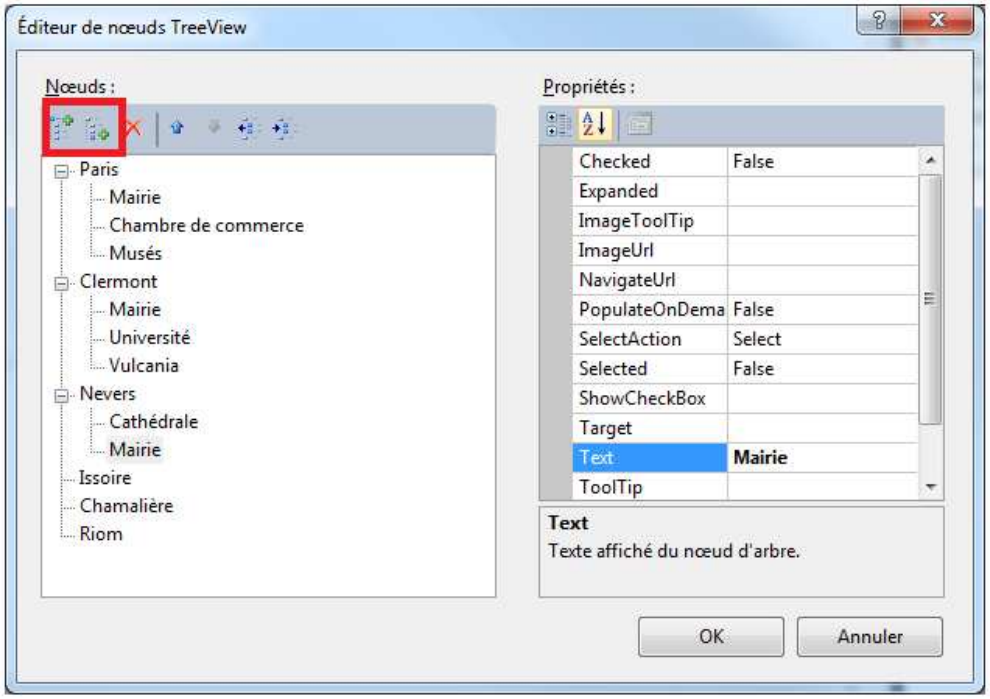
Choisir par exemple le stype **Explorateur de fichiers XP**.



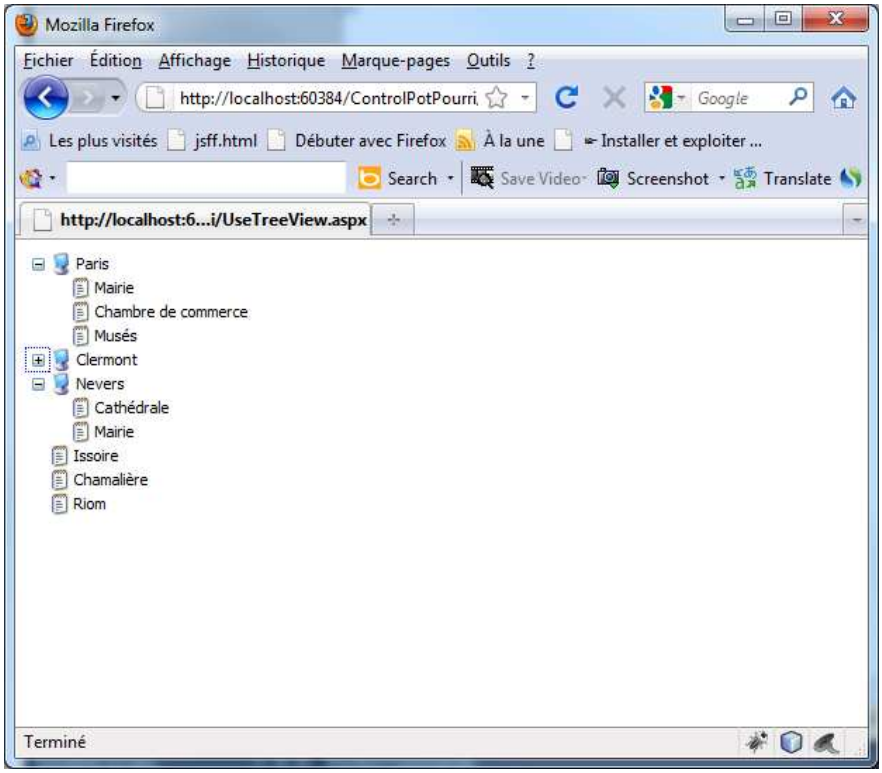
Modifier les nœuds de l'arbre :



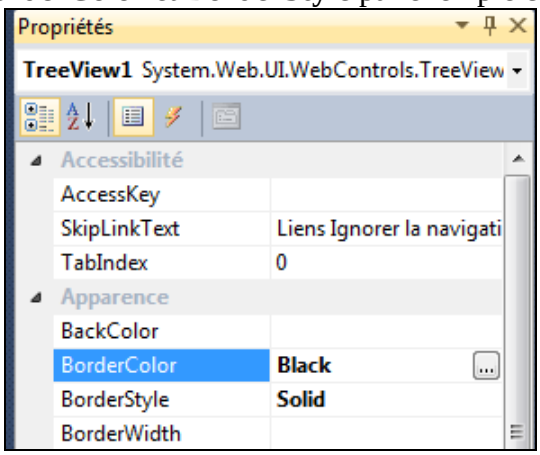
En utilisant les deux icones sous le mot Nœuds, ajouter les éléments de la liste.



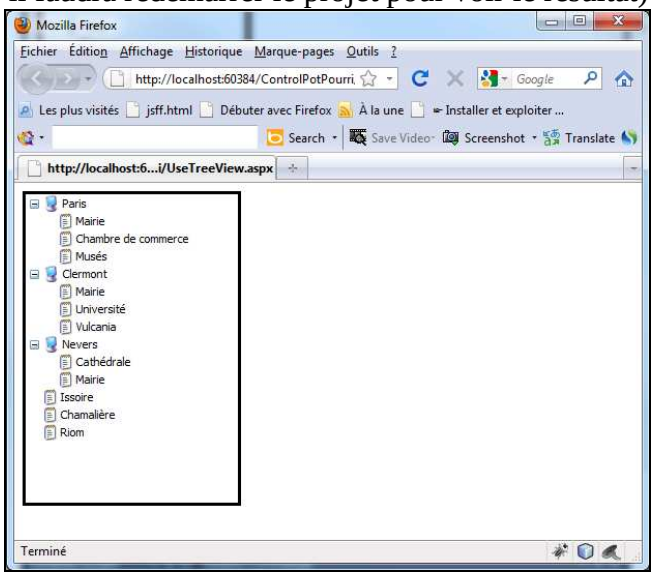
Le résultat à l'exécution est assez spectaculaire :



Modifiez les propriétés **BorderColor** et **BorderStyle** par exemple comme ci-dessous :



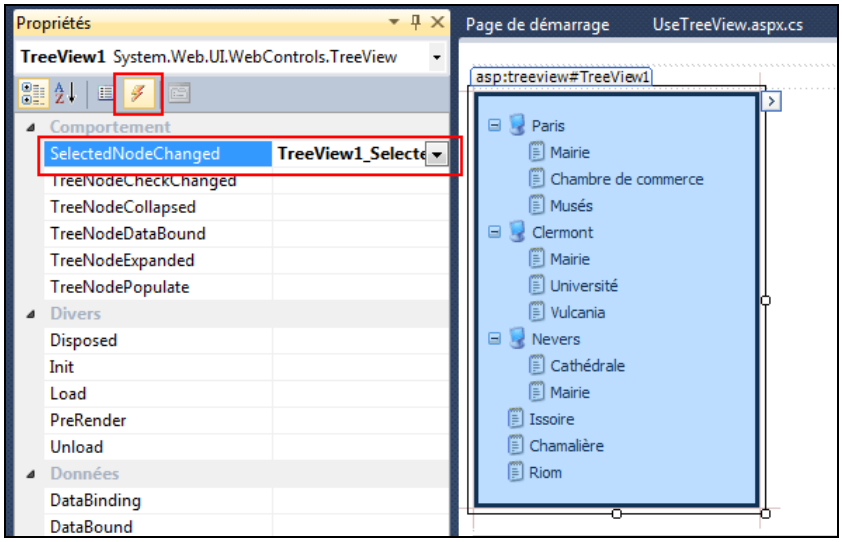
Ce qui donne (parfois il faudra redémarrer le projet pour voir le résultat) :



En éditant le code aspx vous constaterez qu'il n'y a aucune étiquette.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="UseTreeView.aspx.cs"
Inherits="UseTreeView" %>

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

    </div>
    <asp:TreeView ID="TreeView1" runat="server" BorderColor="Black"
        BorderStyle="Solid" Height="298px" ImageSet="XPFileExplorer" NodeIndent="15"
        onselectednodechanged="TreeView1_SelectedNodeChanged" Width="205px">
        <HoverNodeStyle Font-Underline="True" ForeColor="#6666AA" />
        <Nodes>
            <asp:TreeNode Text="Paris" Value="Paris">
                <asp:TreeNode Text="Mairie" Value="Mairie"></asp:TreeNode>
                <asp:TreeNode Text="Chambre de commerce" Value="Chambre de commerce">
                </asp:TreeNode>
                <asp:TreeNode Text="Musés" Value="Musés"></asp:TreeNode>
            </asp:TreeNode>
            <asp:TreeNode Text="Clermont" Value="Clermont">
                <asp:TreeNode Text="Mairie" Value="Mairie"></asp:TreeNode>
                <asp:TreeNode Text="Université" Value="Université"></asp:TreeNode>
                <asp:TreeNode Text="Vulcania" Value="Vulcania"></asp:TreeNode>
            </asp:TreeNode>
            <asp:TreeNode Text="Nevers" Value="Nevers">
                <asp:TreeNode Text="Cathédrale" Value="Cathédrale"></asp:TreeNode>
                <asp:TreeNode Text="Mairie" Value="Mairie"></asp:TreeNode>
            </asp:TreeNode>
            <asp:TreeNode Text="Issoire" Value="Issoire"></asp:TreeNode>
            <asp:TreeNode Text="Chamalière" Value="Chamalière"></asp:TreeNode>
            <asp:TreeNode Text="Riom" Value="Riom"></asp:TreeNode>
        </Nodes>
        <NodeStyle Font-Names="Tahoma" Font-Size="8pt" ForeColor="Black"
            HorizontalPadding="2px" NodeSpacing="0px" VerticalPadding="2px" />
        <ParentNodeStyle Font-Bold="False" />
        <SelectedNodeStyle BackColor="#B5B5B5" Font-Underline="False"
            HorizontalPadding="0px" VerticalPadding="0px" />
    </asp:TreeView>
    </form>
</body>
</html>
```

Ajouter un **TextBox**. Et ajouter un gestionnaire sur le **SelectedNodeChanged** en double cliquant juste à droite de « SelectedNodeChanged ».



Ce qui nous amène au code suivant :

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

public partial class UseTreeView : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
    {

    }
}
```

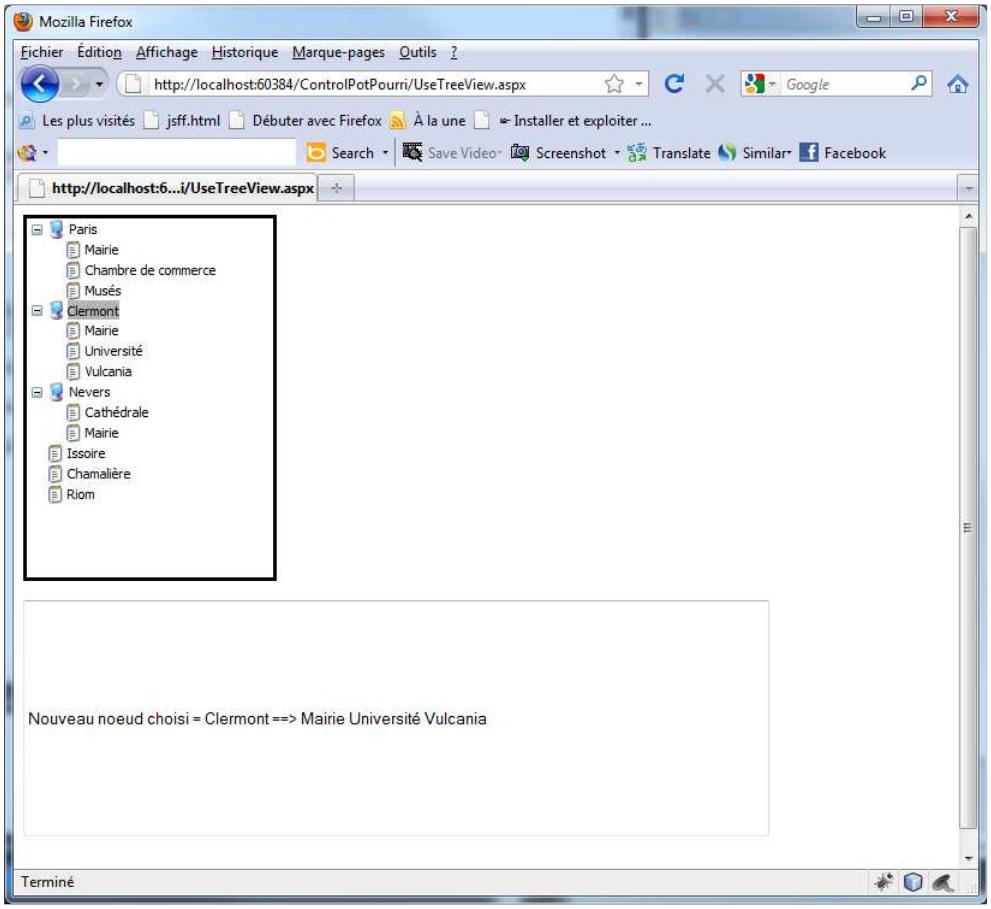
Modifiez la procédure comme suit :

```
    protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
    {

        String S = this.TreeView1.SelectedNode.Text;
        String chaine = "Nouveau noeud choisi = " + S + " ==> ";


        TreeNodeCollection Childnodes = this.TreeView1.SelectedNode.ChildNodes;

        if (Childnodes != null)
        {
            this.TextBox1.Text = String.Empty;
            StringBuilder sb = new StringBuilder();
```

```
            sb.Append(chaine);
            foreach (TreeNode chilnode in Childnodes)
            {
              sb.AppendFormat("{0}\n", chilnode.Value);
            }
            this.TextBox1.Text = sb.ToString();

        }
    }
```

NB : Ajouter la package « System.Text » si besoin.

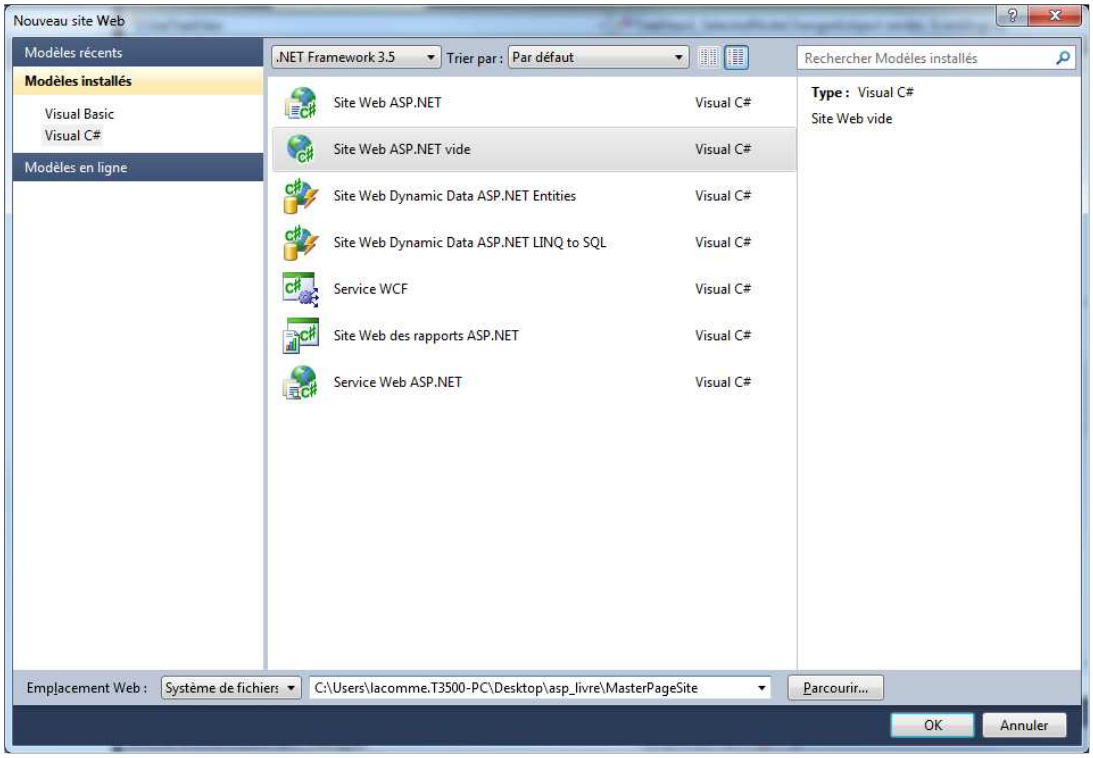
Ceci donne à l'exécution :



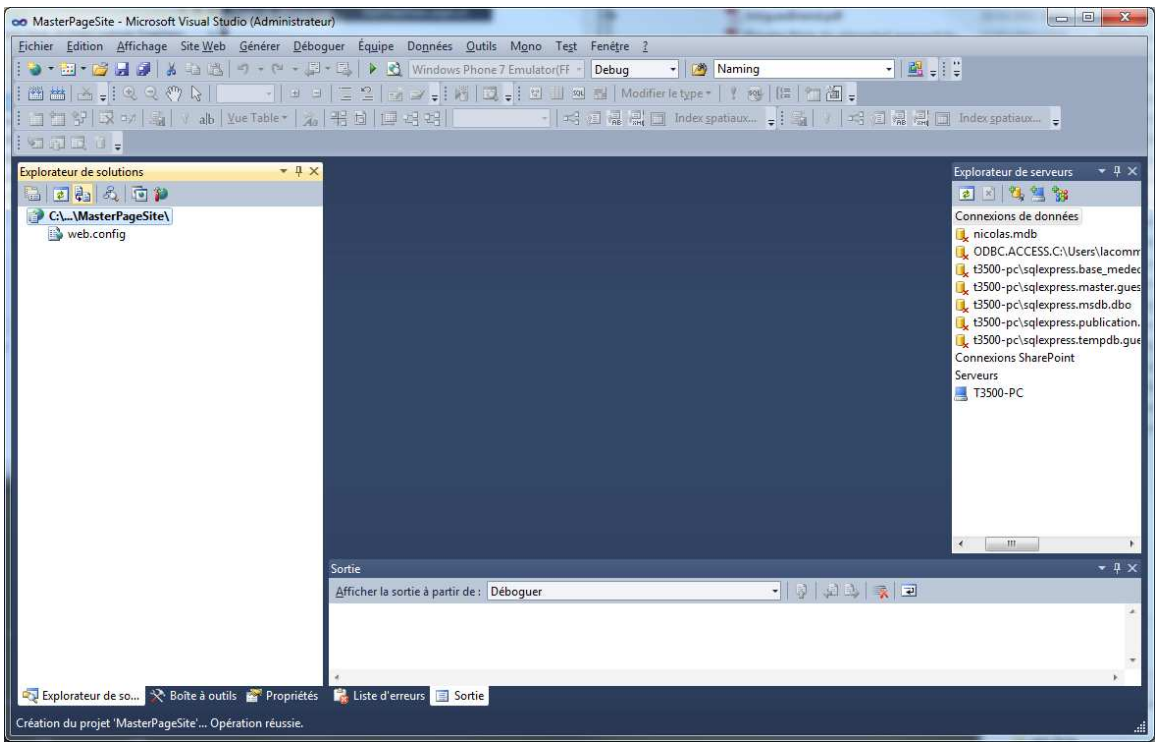
8) Cohérence de l'interface utilisateur

8.1) Utilisation d'une page Maître

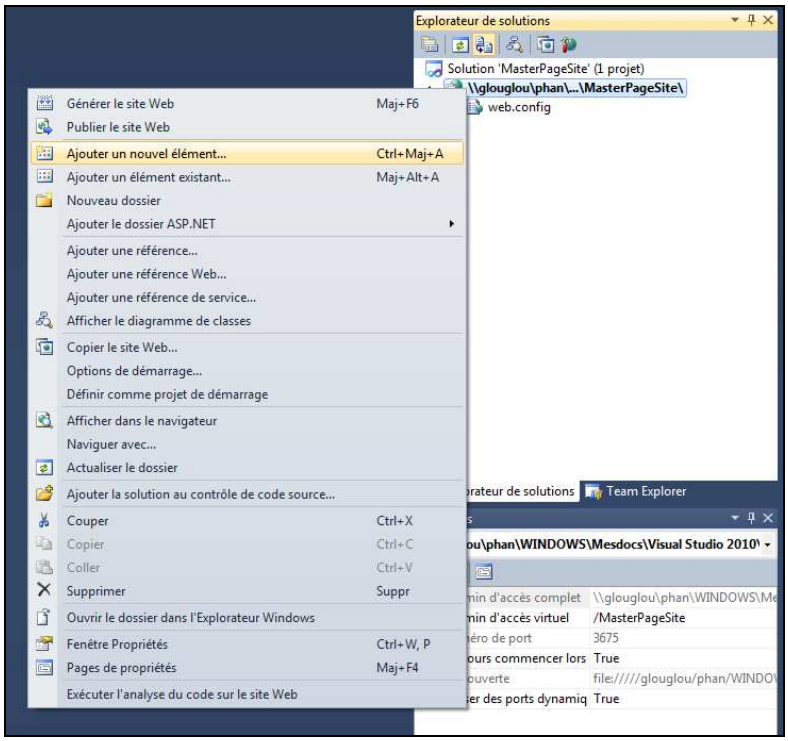
Créer un nouveau site **vide** nommé : **MasterPageSite**.

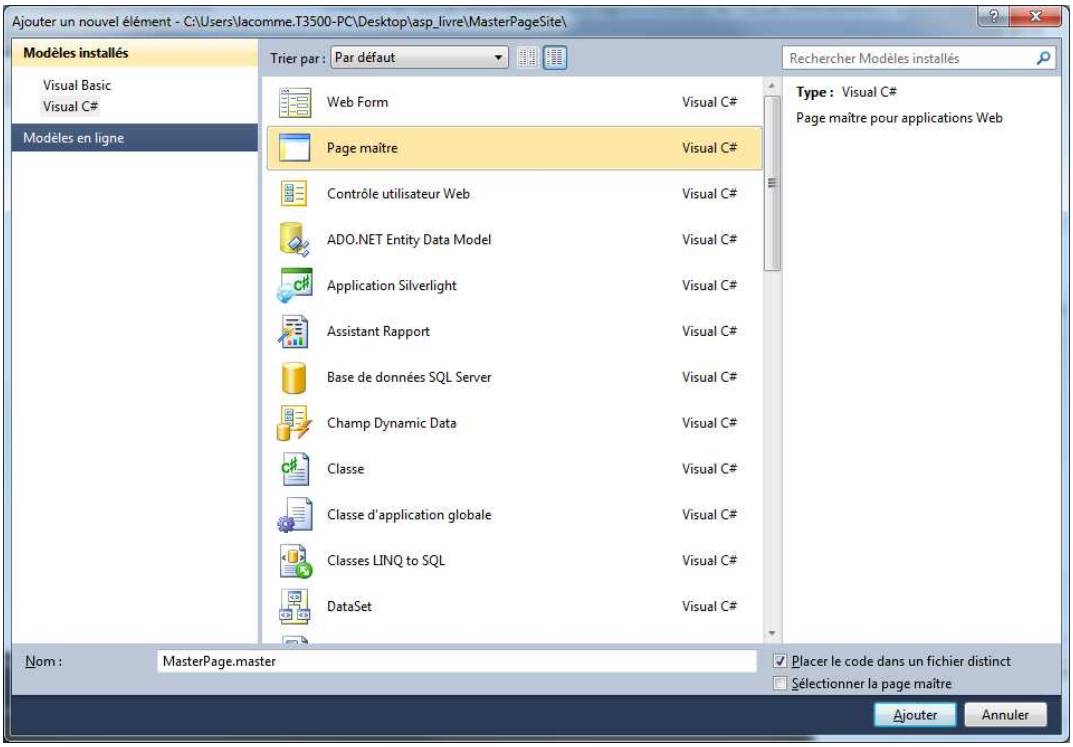


Ce qui donne un site vide.

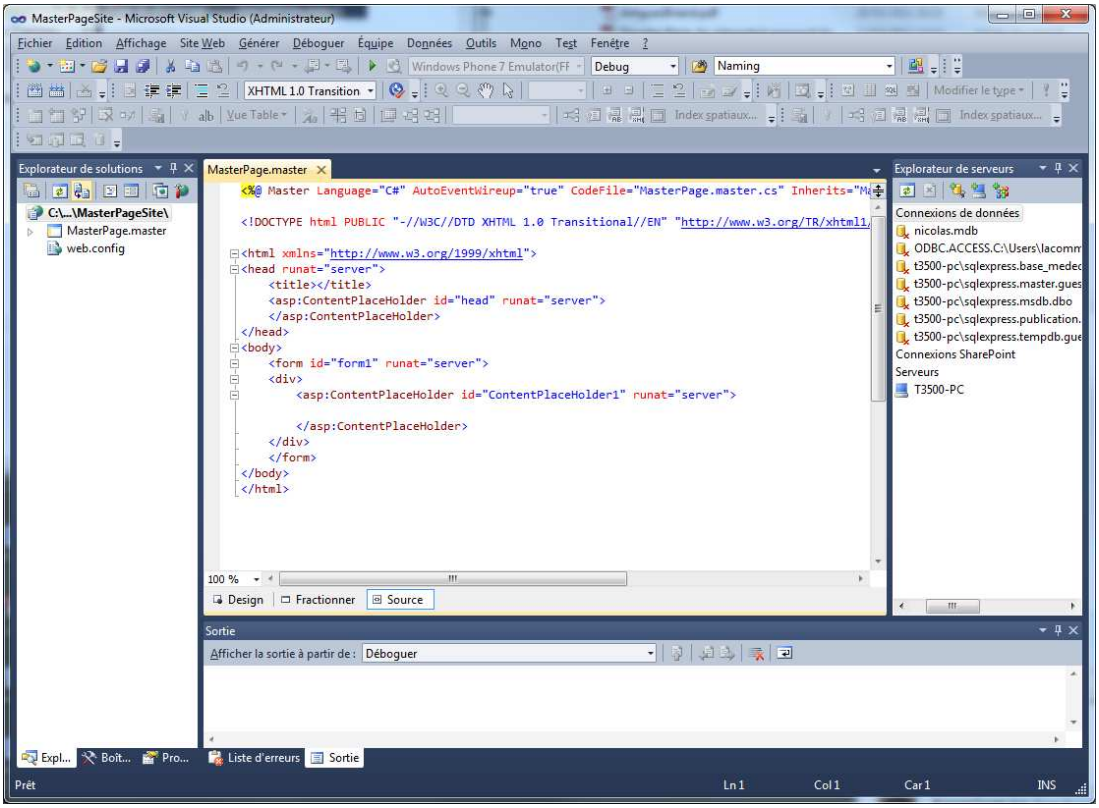


Ajoutez un nouvel élément à la page : sélectionnez **page maître**.





Ce qui donne :



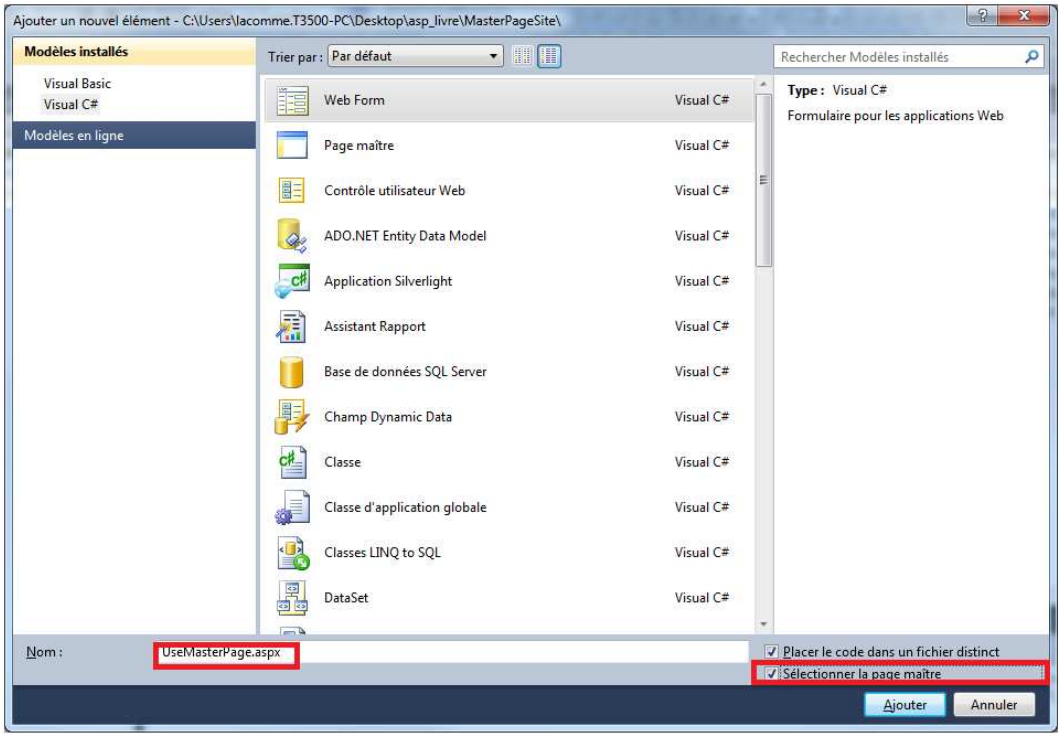
Le code généré est le suivant :

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="MasterPage" %>

<!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></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

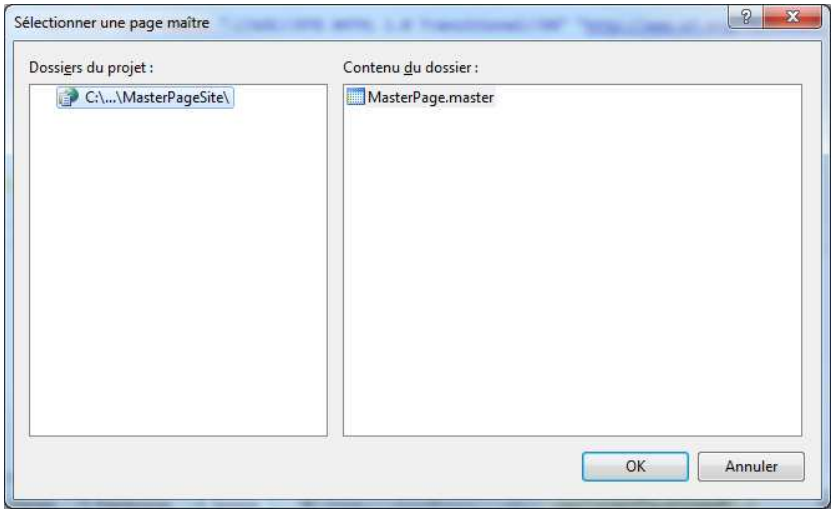
On peut facilement modifier la couleur d'arrière-plan comme ceci :

```
<body style="background-color: #bbbbbb;">
    <form id="form1" runat="server">
    <div>
        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
```

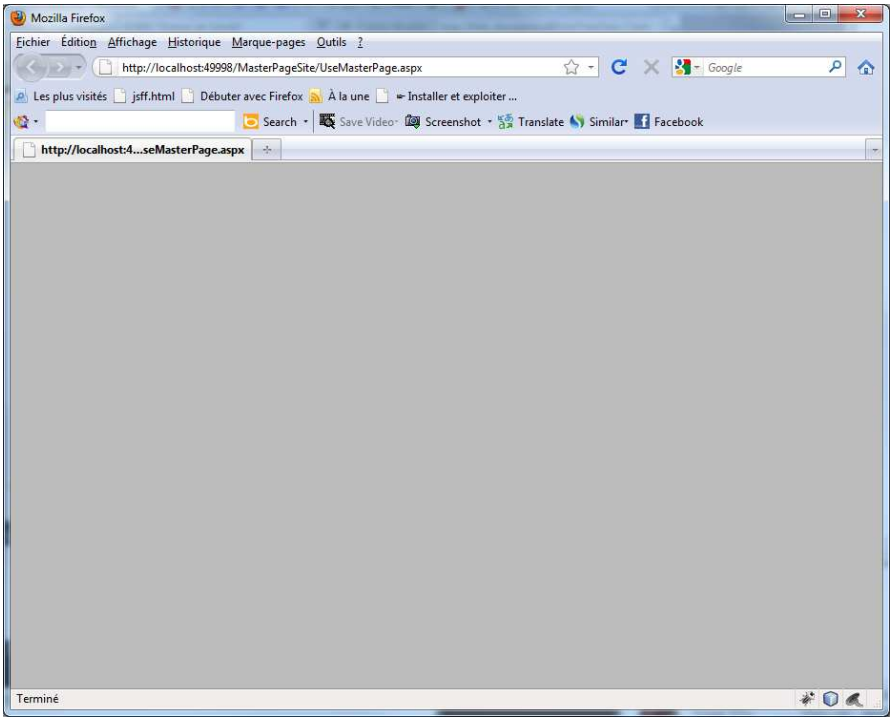
Ajoutez un nouvel élément à la page de type **Web Form** avec page maître et l'appeler **UseMasterPage**.



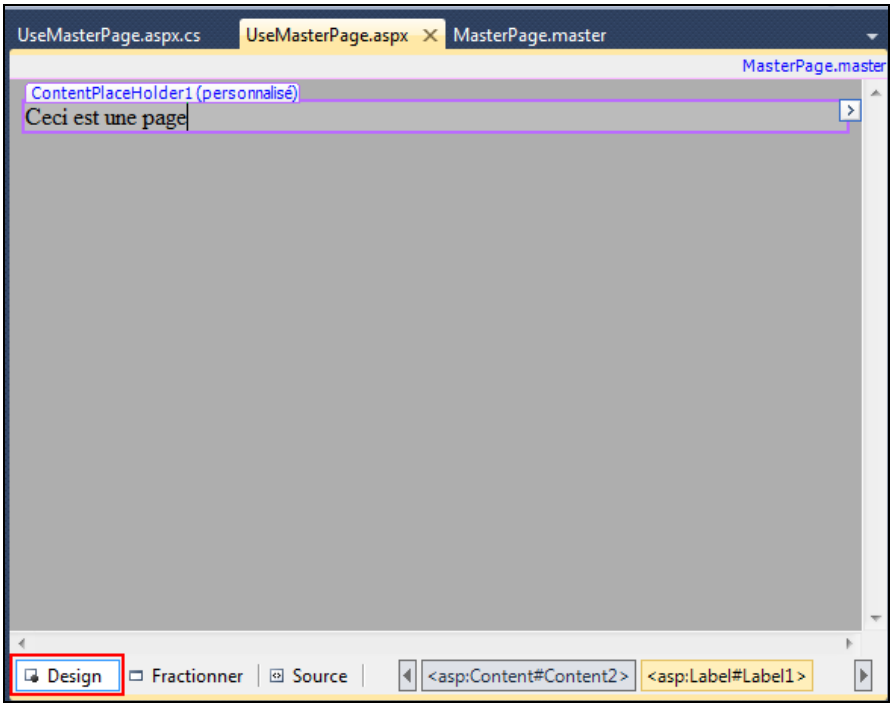
Sélectionner ensuite la page maître précédemment créée :



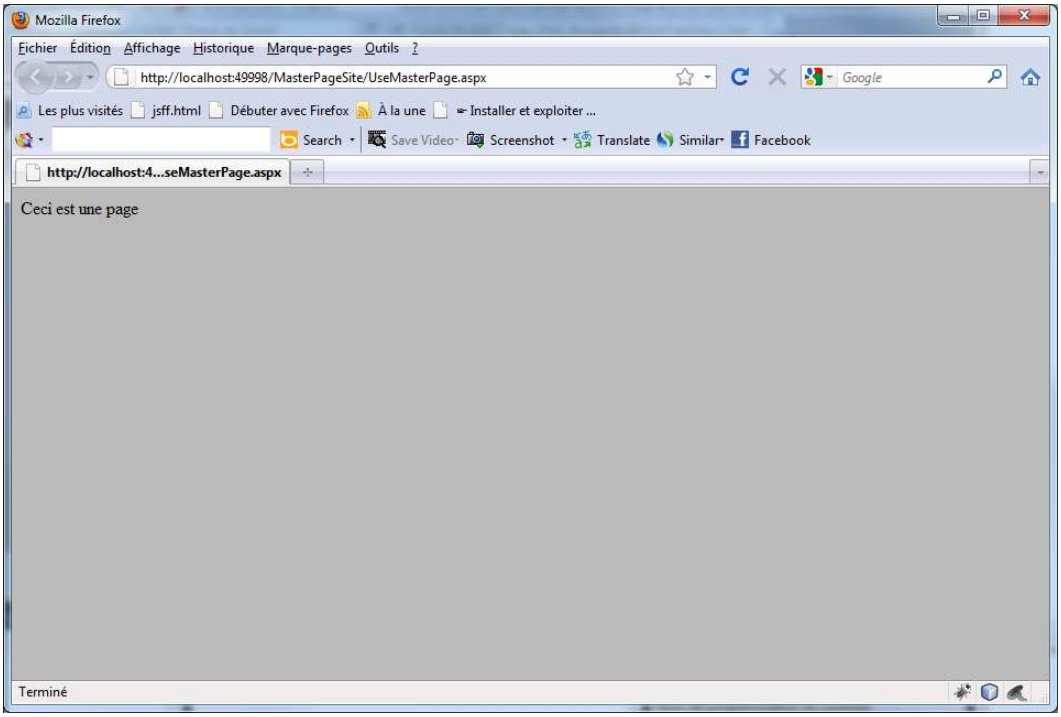
L'exécution de « UseMasterPage.aspx » donne une page grise ce qui montre que la page utilise la page maître. Essayer de modifier la couleur dans « MasterPage.master » pour voir le changement.



On peut manipuler cette page de la manière usuelle… par exemple en ajoutant un texte simple comme suit :

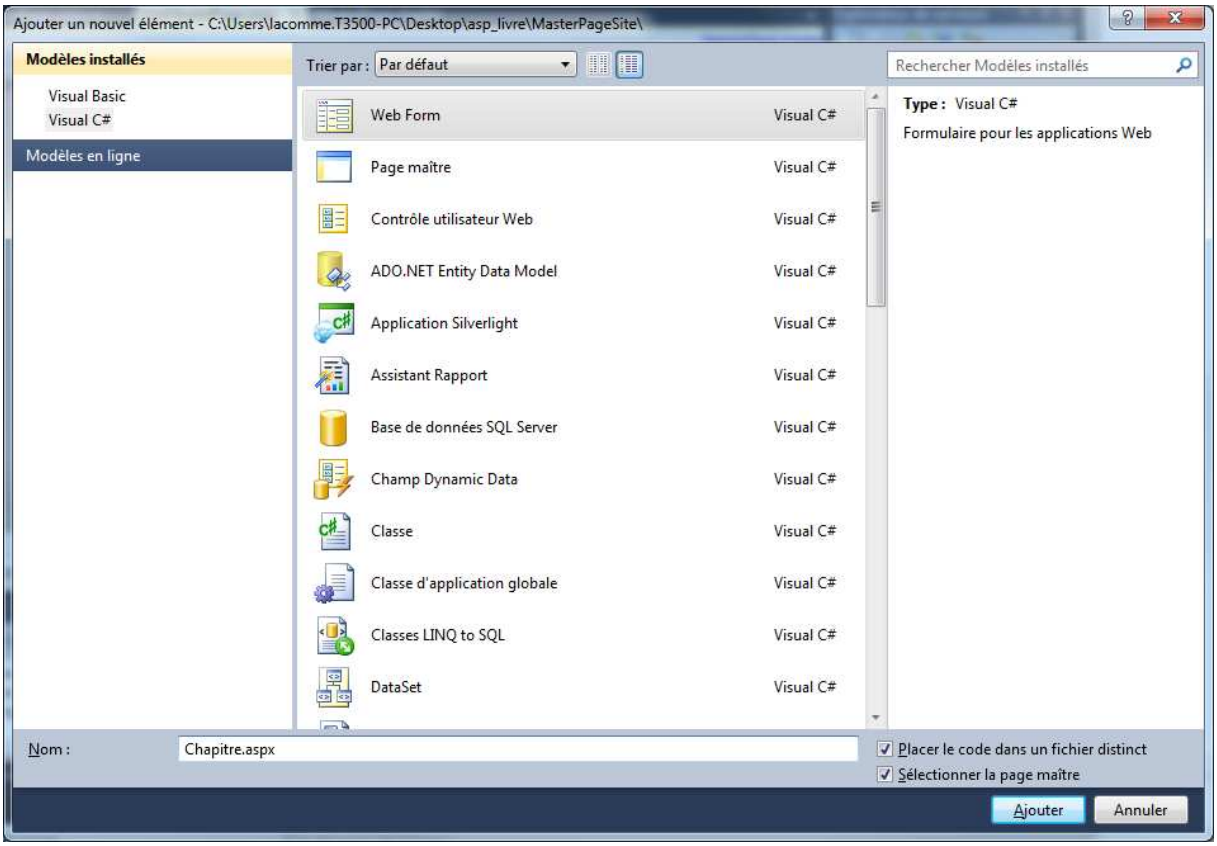


Ce qui donnera :

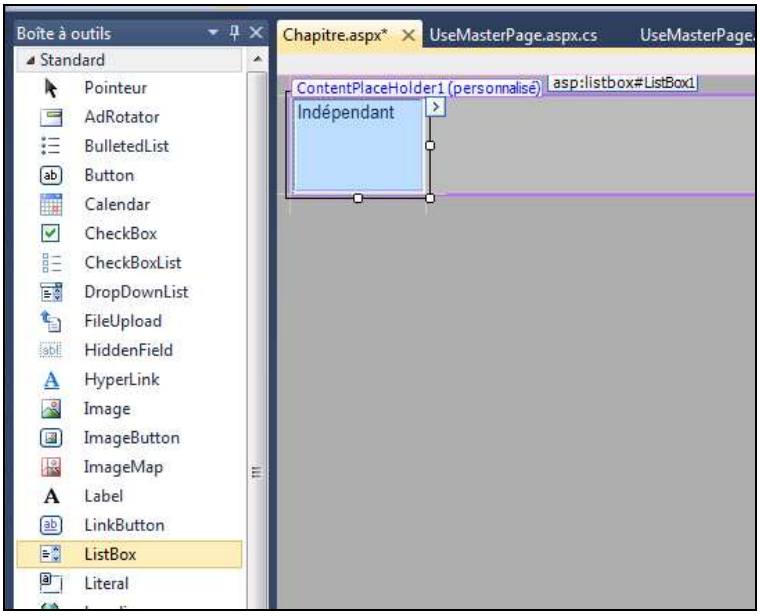


Ajoutez un nouvel élément à la page de type **Web Form** avec page maître.

Nommez cette page : **Chapitre.aspx**

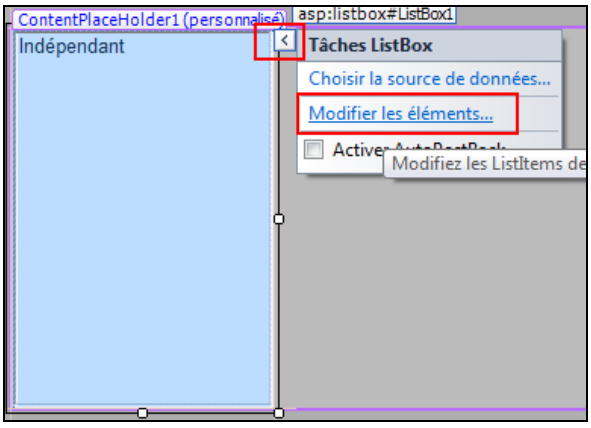


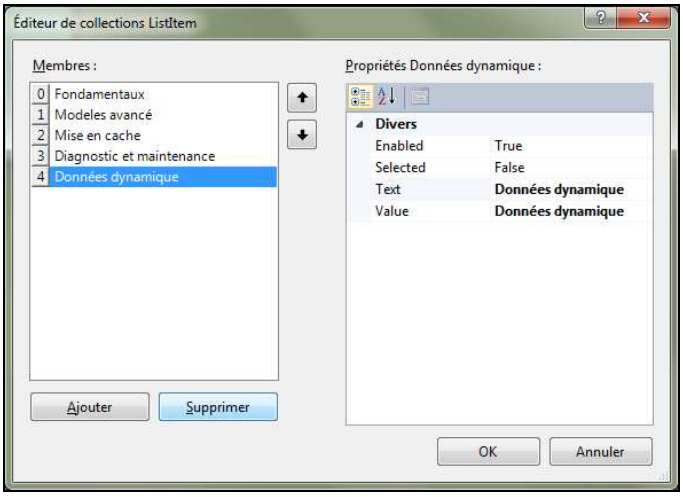
Poser une **ListBox** sur la page.

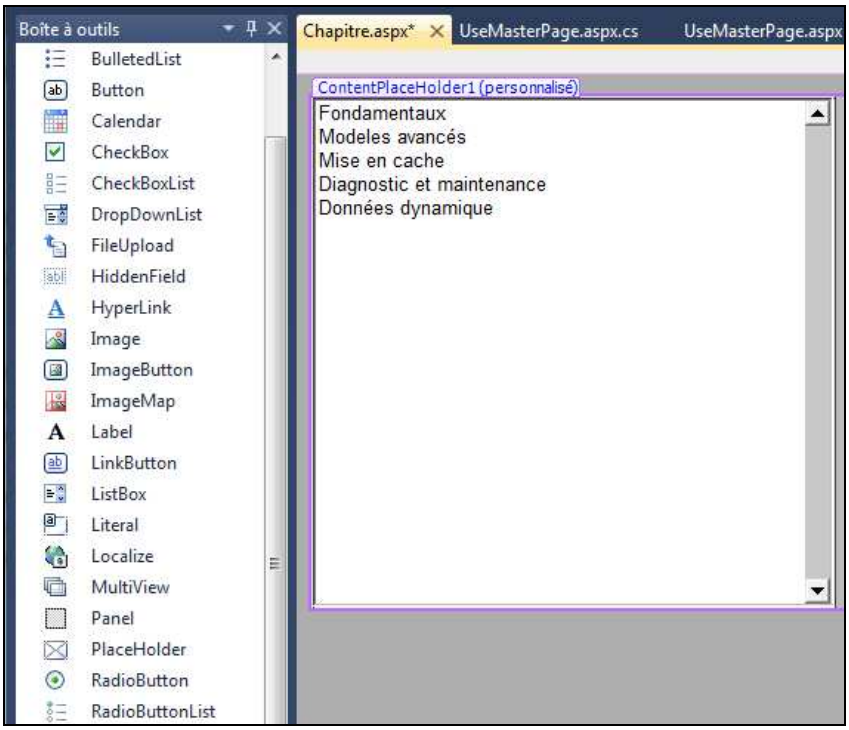


Ajouter quelques éléments dans la listbox :

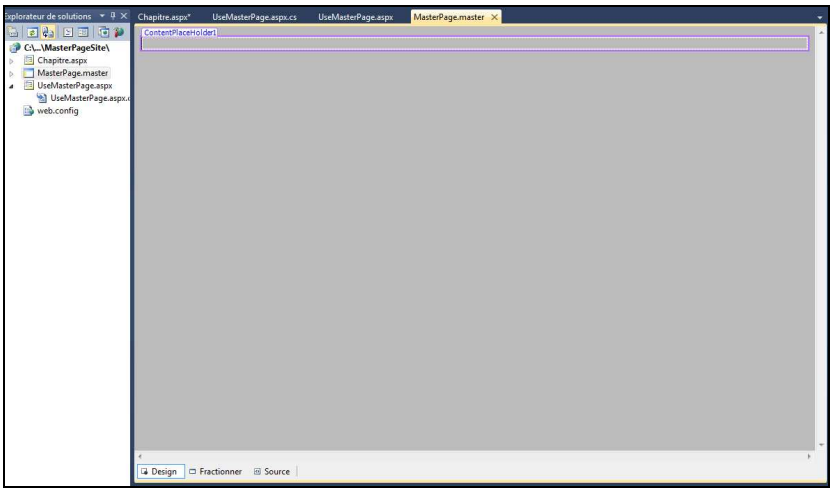
```
Fondamentaux
Modeles avancé
Mise en cache
Diagnostic et maintenance
Données dynamique
```



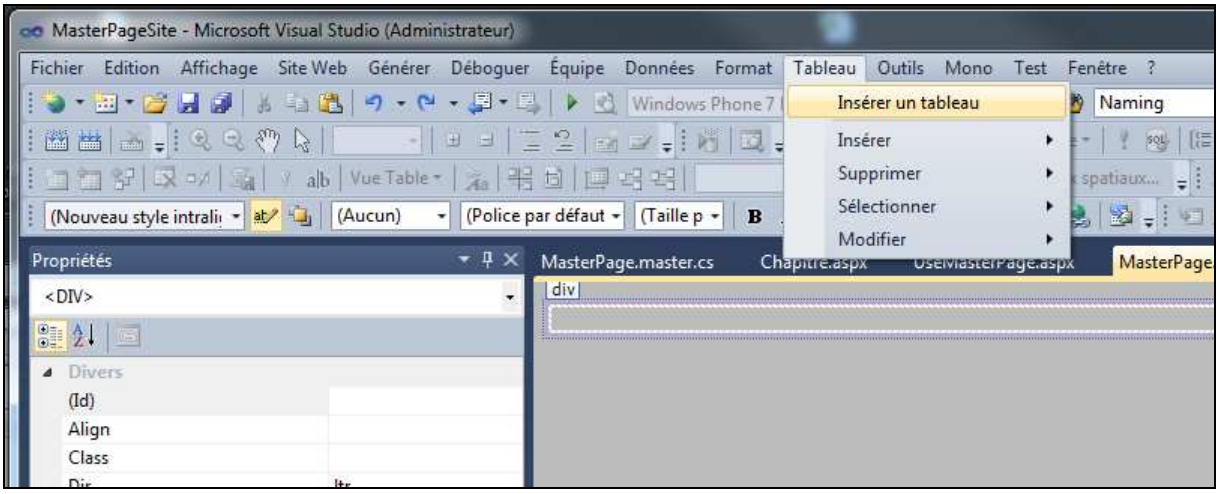


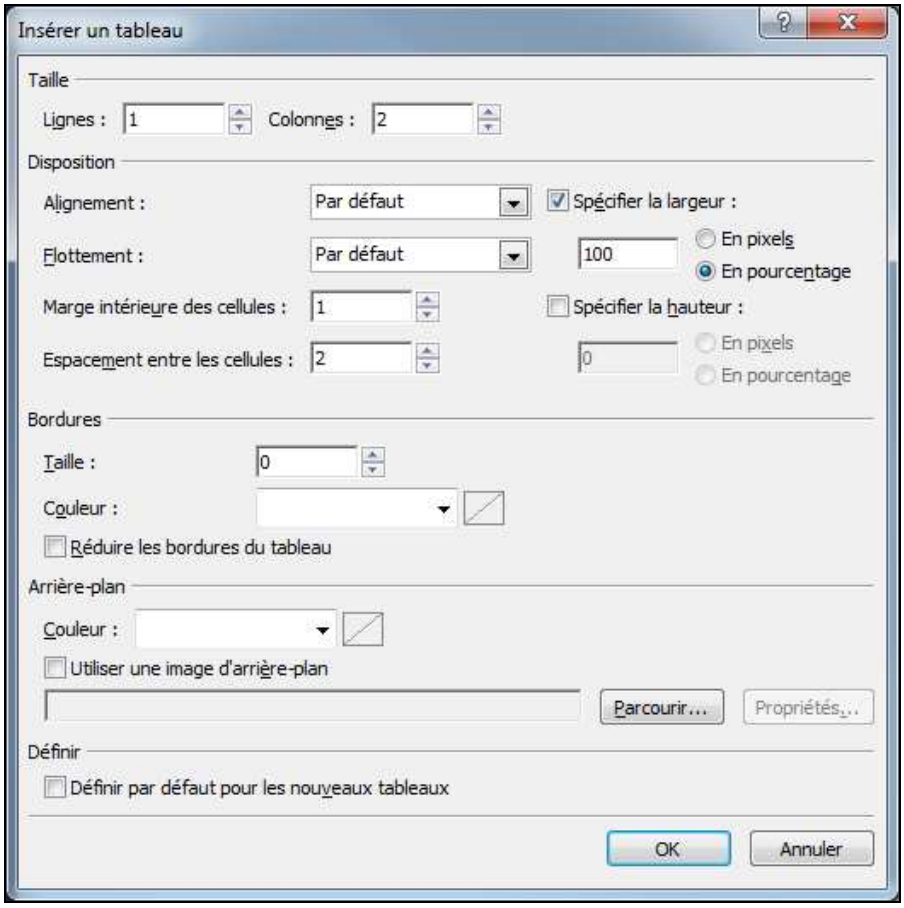


Revenez au fichier maitre.

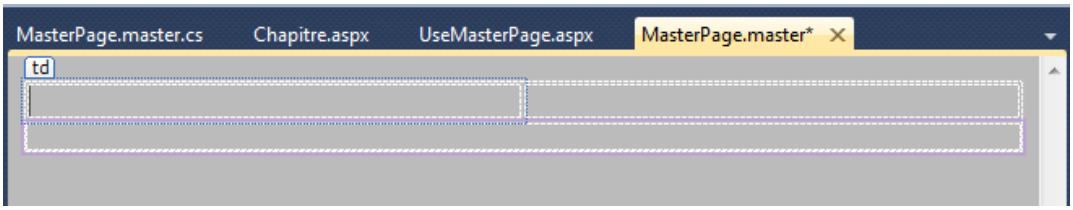


Insérez un tableau juste au-dessus du volet représentant le contenu de la page.



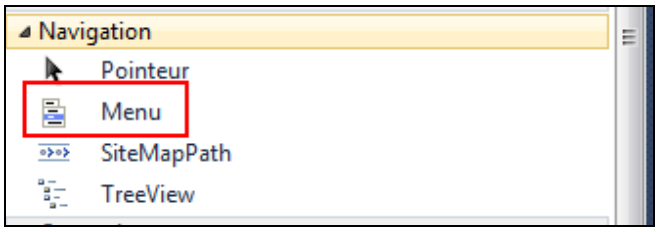


Ce qui donne :

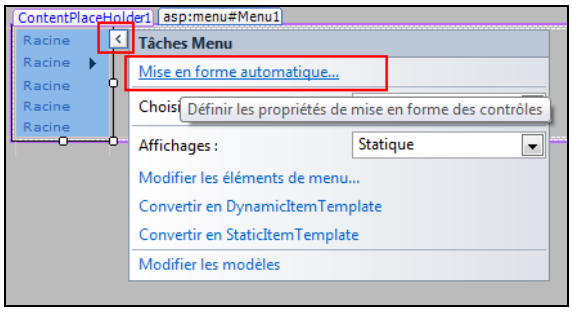


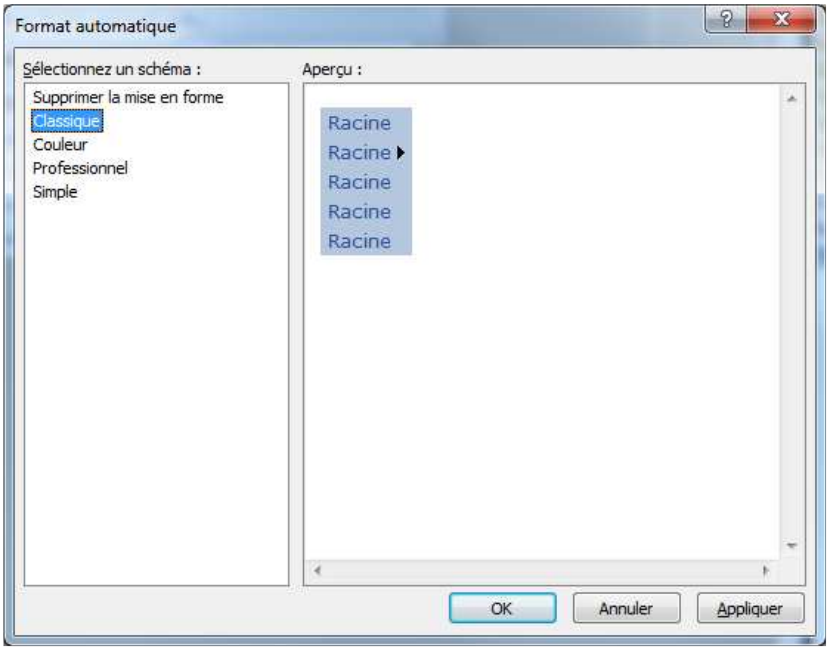
NB : Remarquer bien que le tableau a été ajouté au-dessus de la zone violette. Elle représente en fait la zone de la page « Chapitre ».

Dans la cellule de gauche on va ajouter un menu qui permettre de naviguer entre les pages du site. Allez dans l'onglet **Navigation** et choisir **Menu.** Déplacer le menu sur la colonne de gauche de la page web.

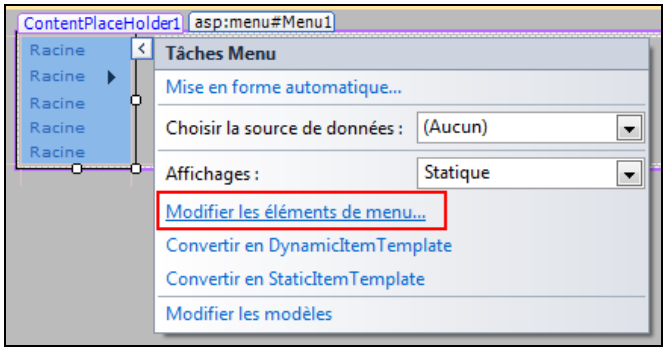


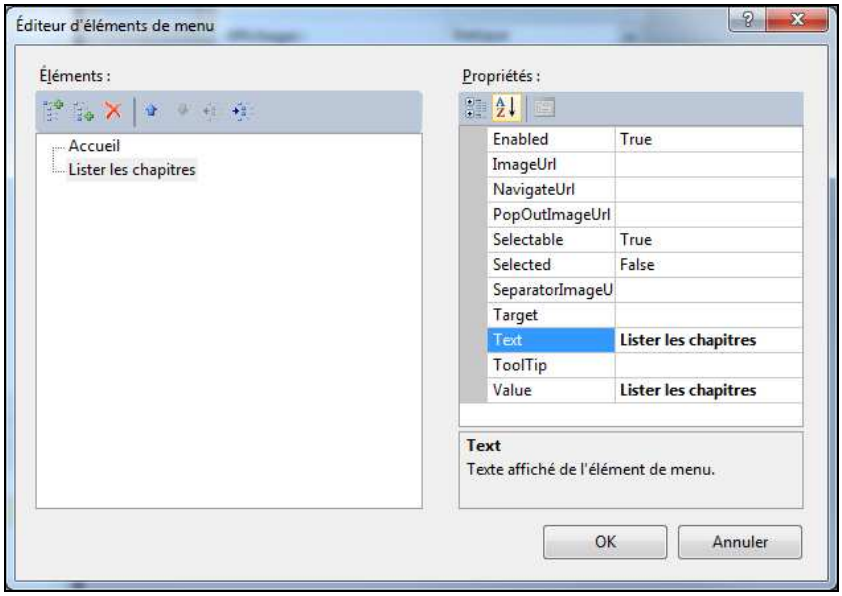
Choisir une mise en forme "classique" :



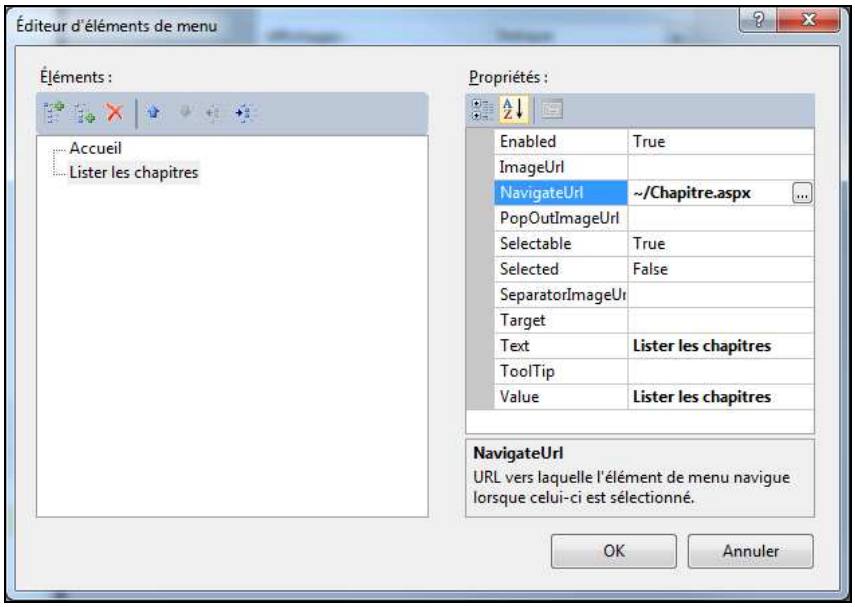


Donner un contenu à votre menu :

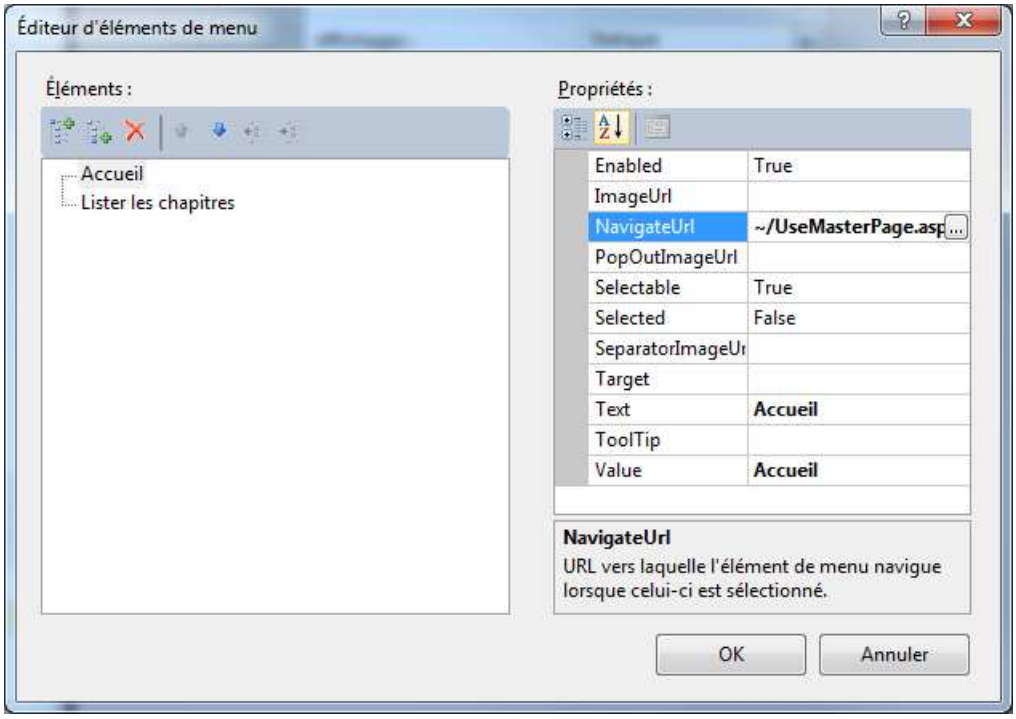




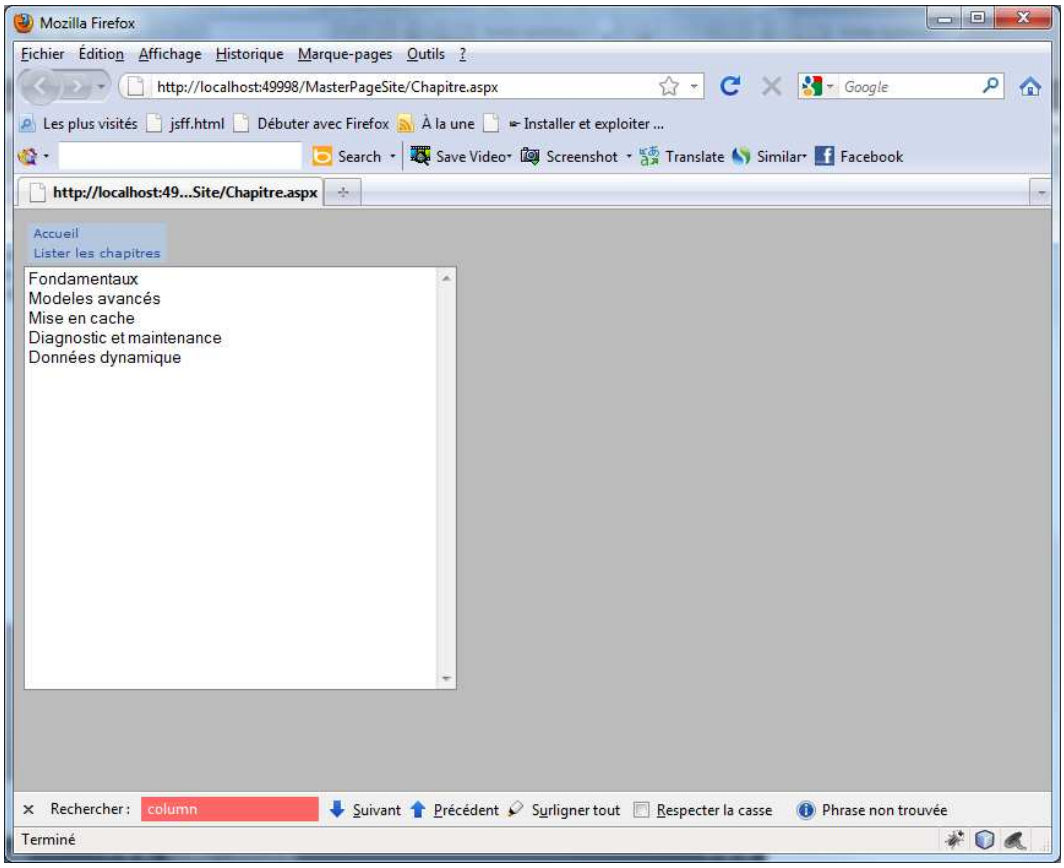
Connecter l'élément **Lister les Chapitres** avec la page **Chapitre.aspx**



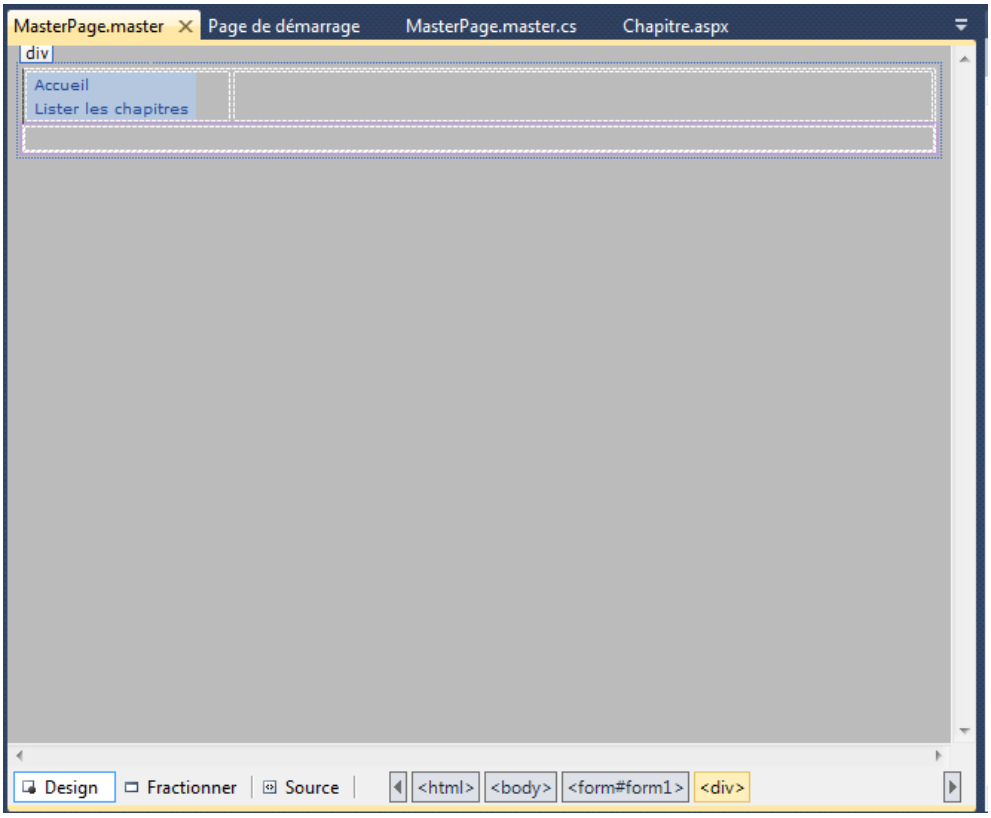
Connecter l'élément **Accueil** avec la page **UseMasterpage.aspx**



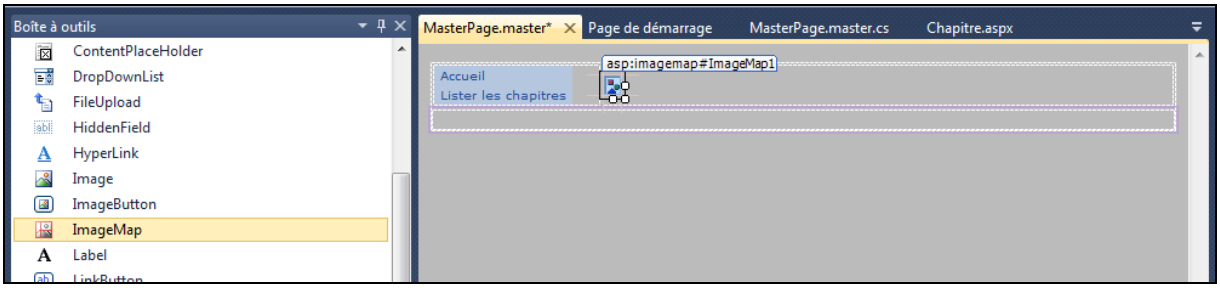
Si on exécute le fichier Chapitre.aspx on obtient :



Revenez à la page maître.

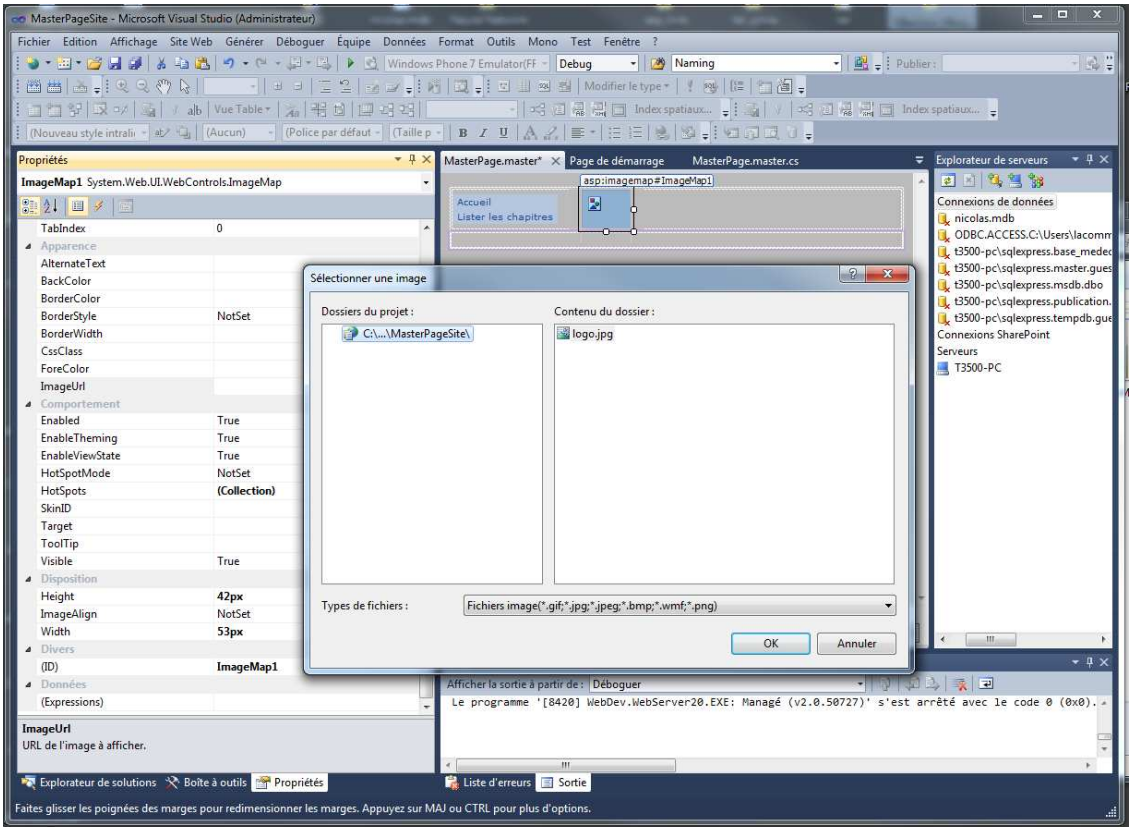


Dans la case droite du tableau nous allons ajouter une image.
Poser dans cette case une **ImageMap**.

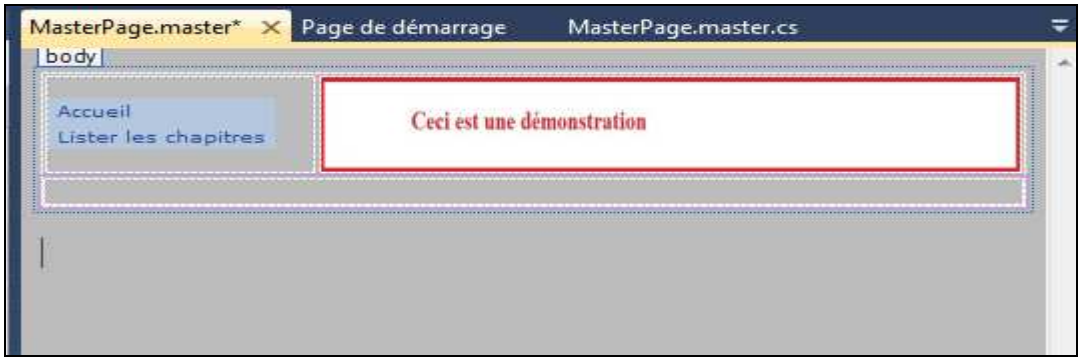


Télécharger l'image à cette adresse :
http://fc.isima.fr/~phan/tuto/ASP/logo.jpg

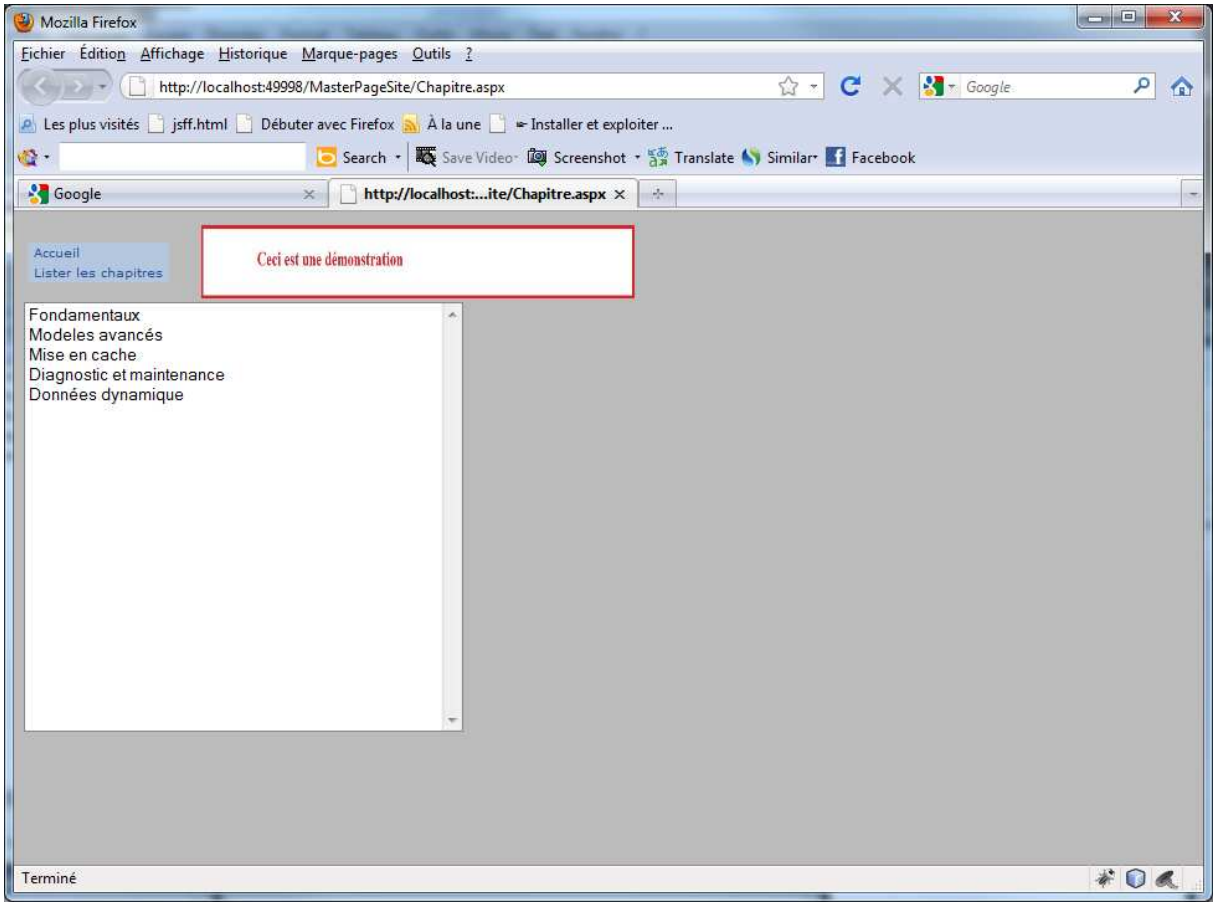
Revenir à la zone image et modifier le champ **ImageUrl** pour mettre l'image que vous avez téléchargé. Penser à bien mettre votre image dans le bon dossier… Et redémarrer (encore) le projet au cas où vous ne verrais pas votre image.



Le résultat devrait ressembler à ceci :

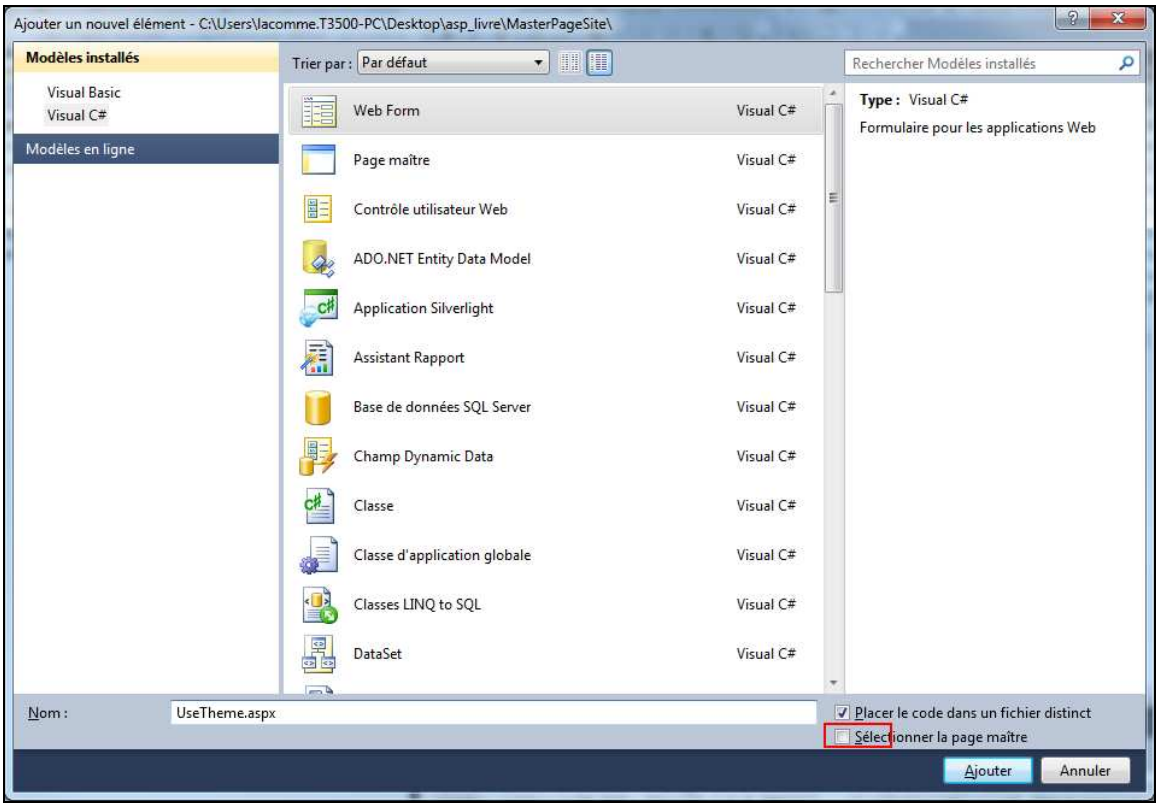


Si on exécute la page Chapitre.aspx on obtient :

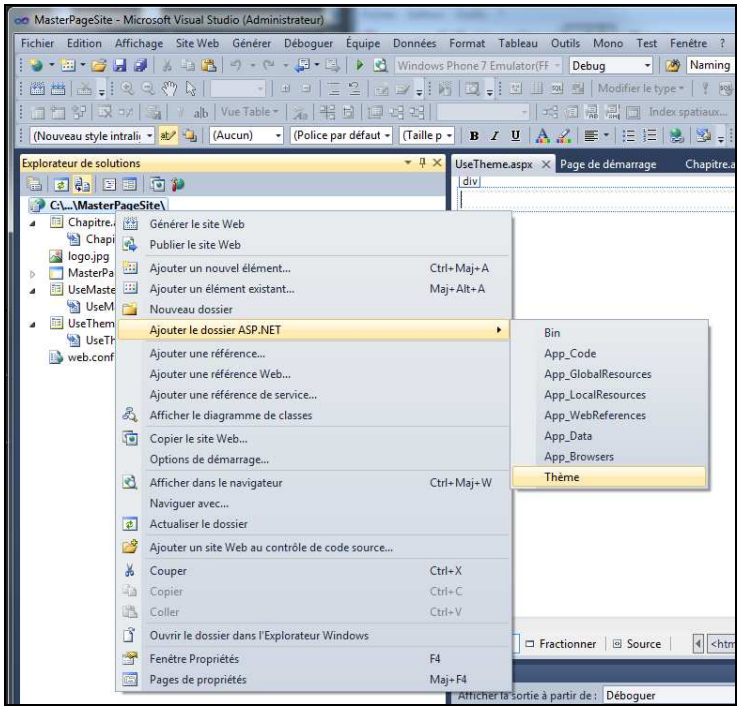

Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils ?
http://localhost:49998/MasterPageSite/Chapitre.aspx
Google
Les plus visités jsff.html Débuter avec Firefox À la une Installer et exploiter ...
Search Save Video Screenshot Translate Similar Facebook
Google
http://localhost:...ite/Chapitre.aspx
Accueil
Lister les chapitres
Ceci est une démonstration
Fondamentaux
Modeles avancés
Mise en cache
Diagnostic et maintenance
Données dynamique
Terminé

8.2) Utilisation d'un **thème**

Ajouter un formulaire nommé **UseTheme.aspx** au projet sans que celui-ci dépende de la page maître.

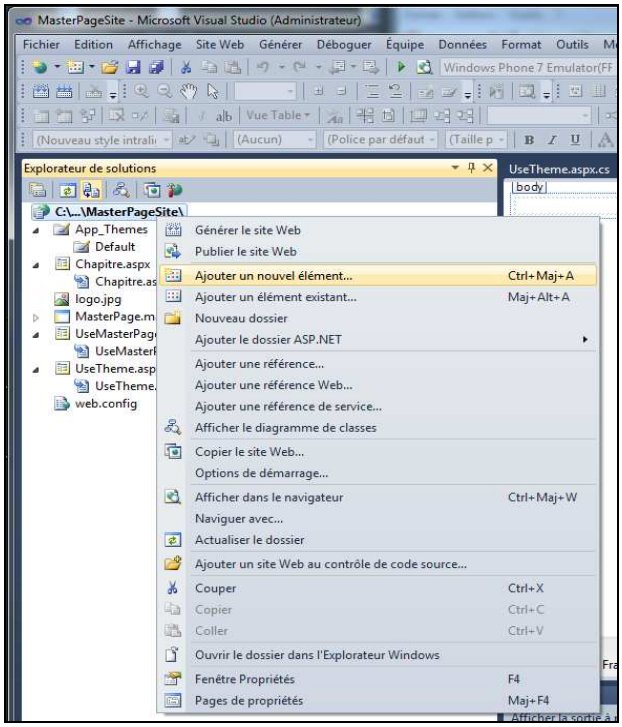


Ajoutez un dossier de thèmes au projet : click droit sur le projet / Ajouter le dossier ASP.NET / Thème.

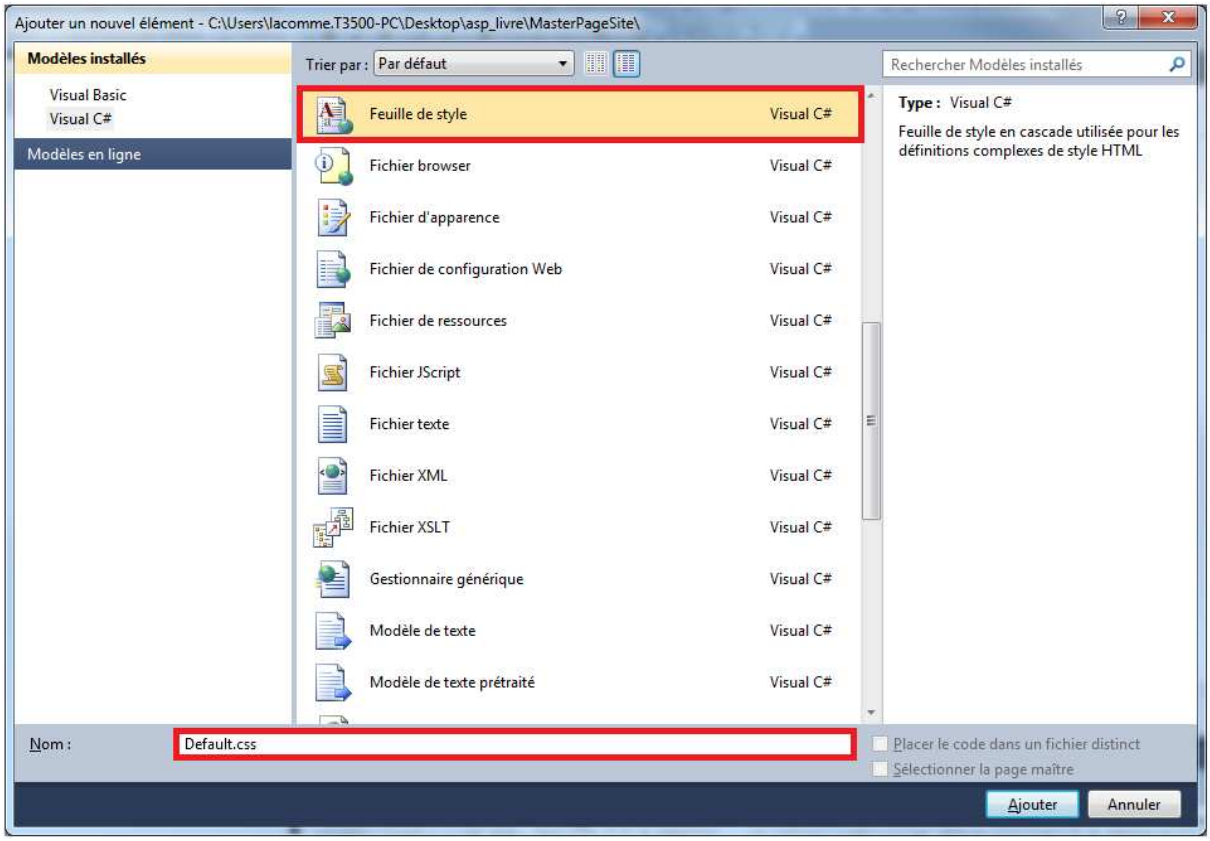


Renommez le dossier **Thème1** en **Default**.

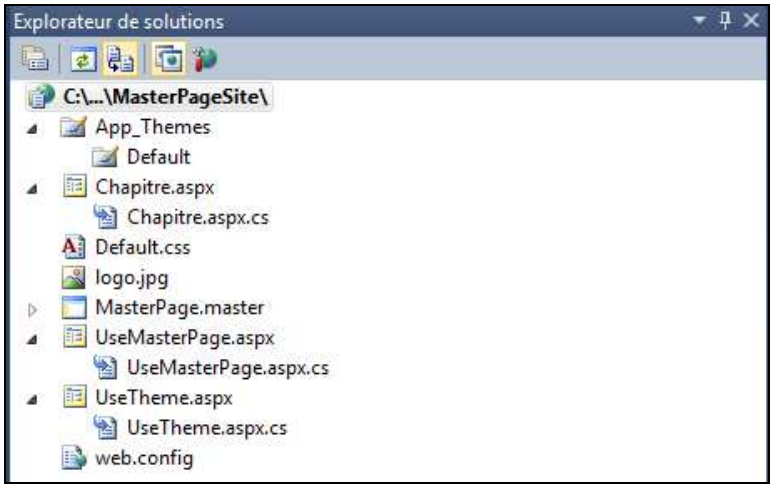
Faire un clic droit sur le projet puis Ajouter un nouvel élément.



Choisir **Feuille de style** et **Default.css** comme nom.

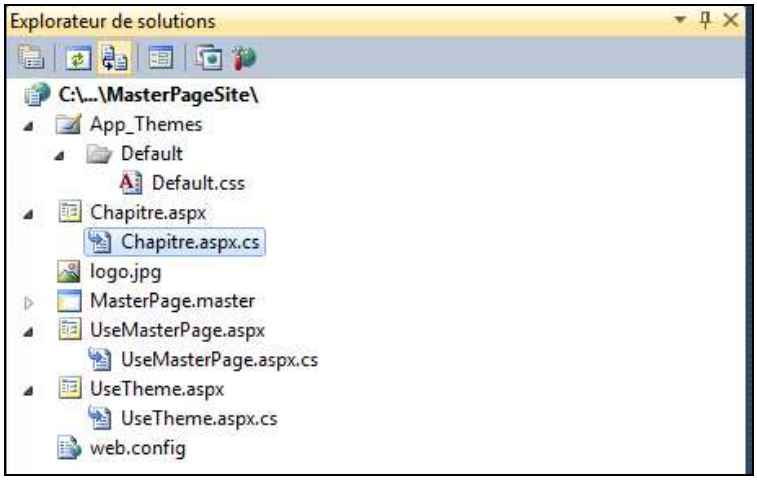


Ceci devrait donner un projet ressemblant à ceci :

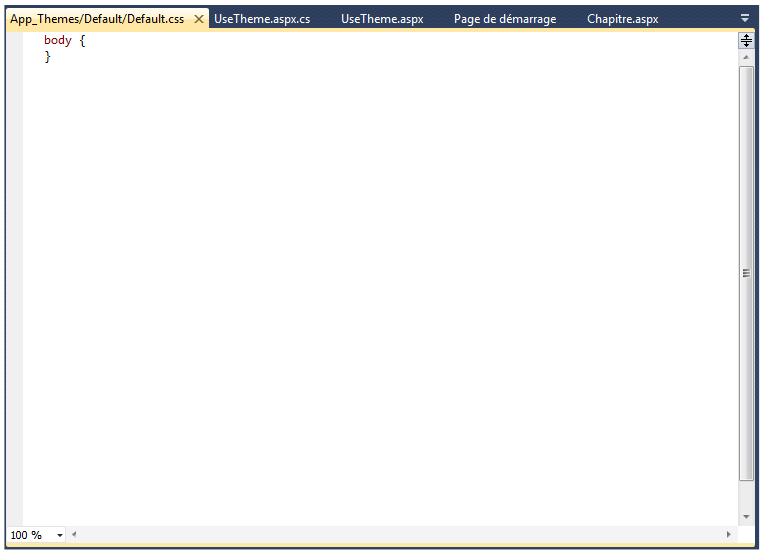


Déplacer le fichier **Default.css** dans le répertoire nommé **Default**.

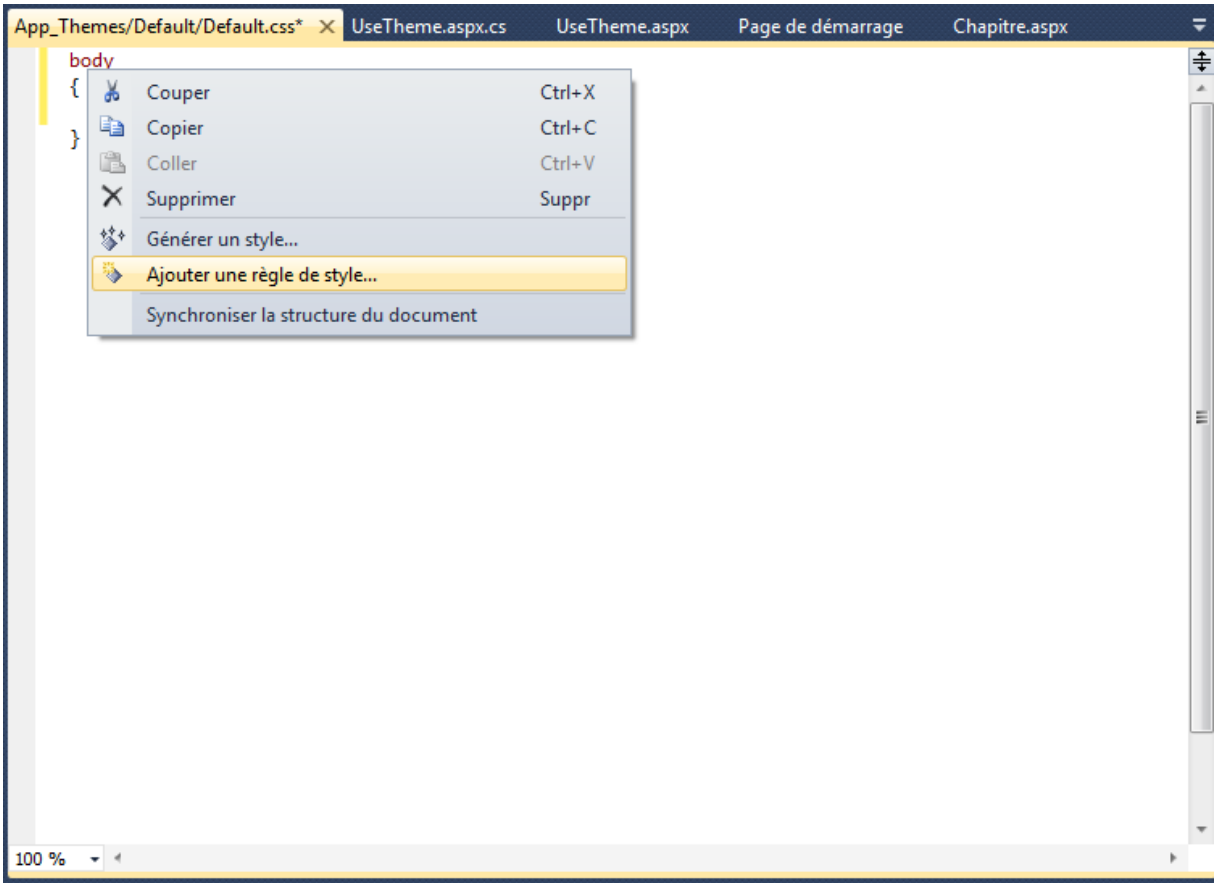
Cela devrait donner ceci :

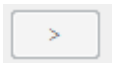


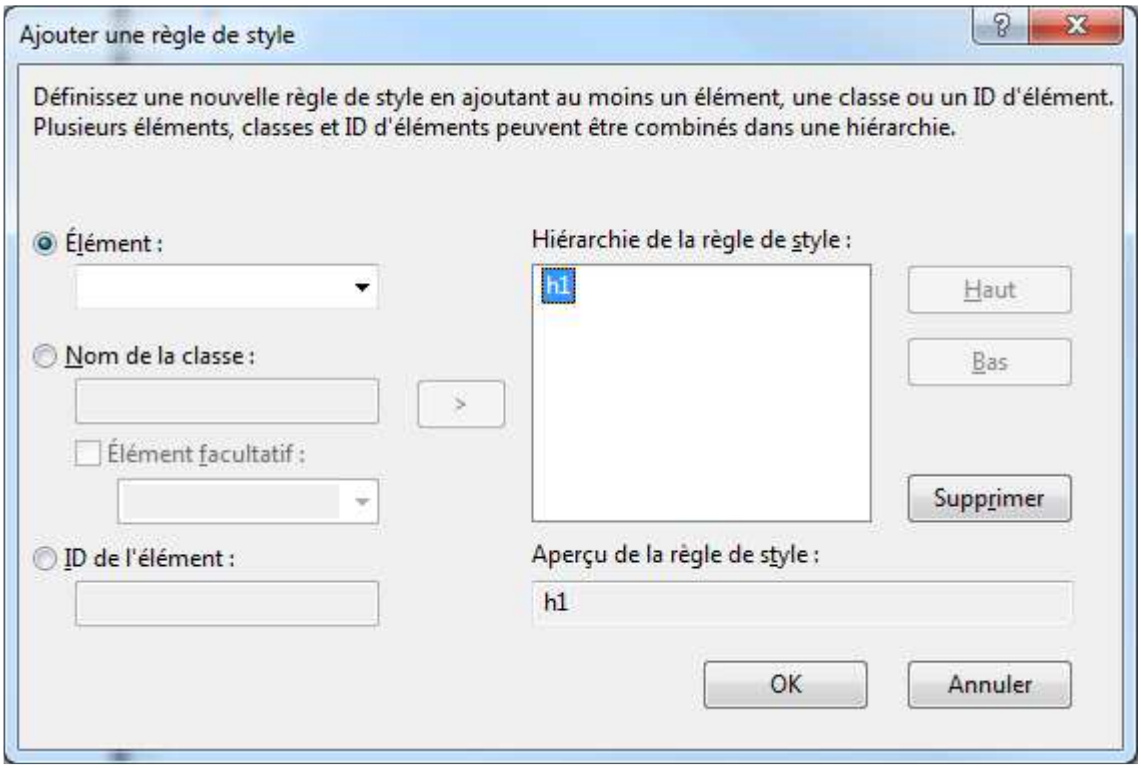
Ouvrir le fichier **Default.css**.



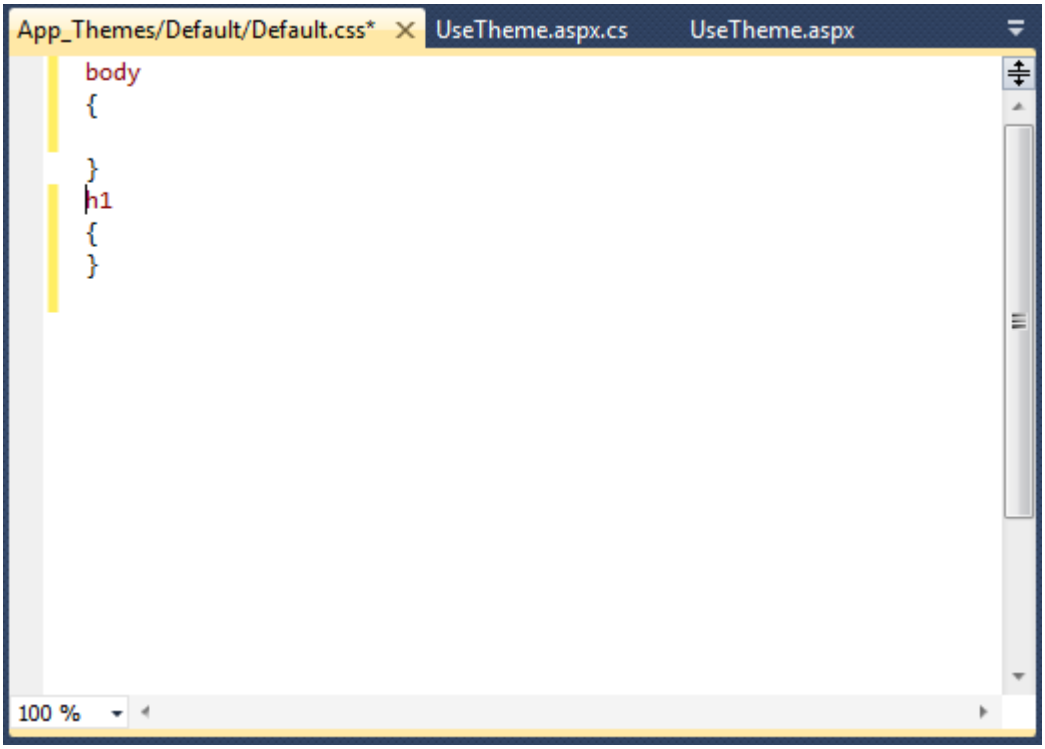
Faire un clic droit sur **body**.



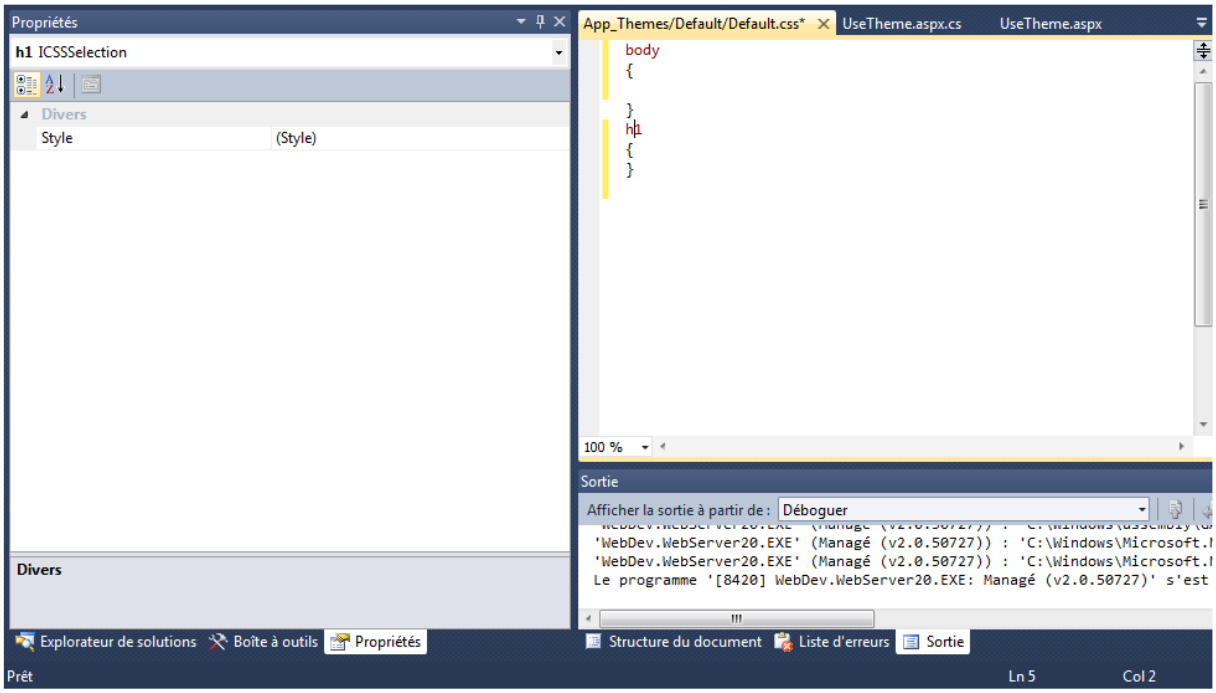
Dans élément, choisir h1 et cliquez sur le bouton  :



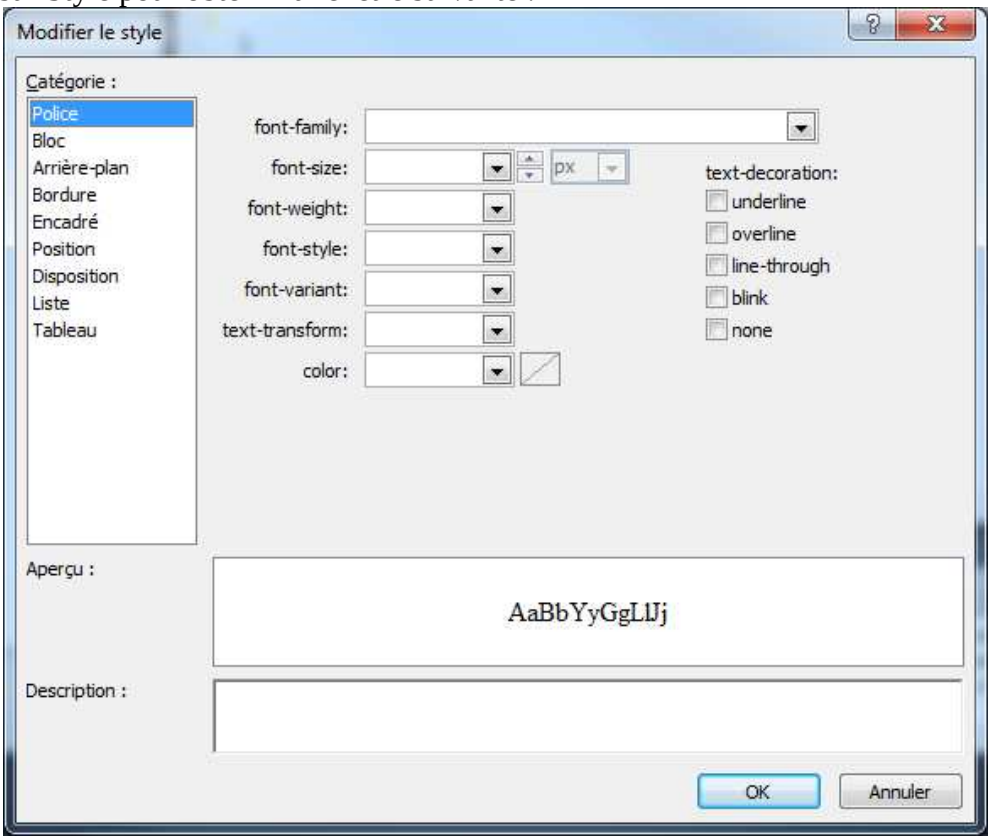
Et faire OK. Ceci devrait donner :



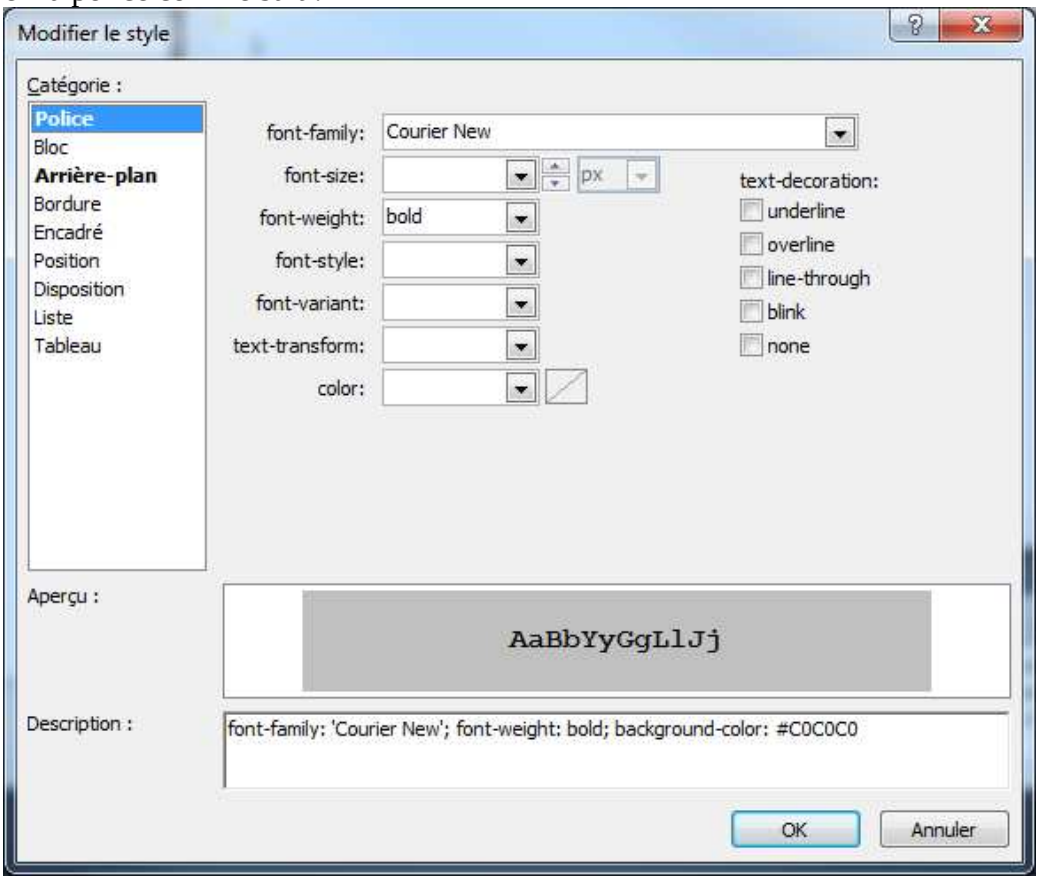
Sélectionnez **h1** et consultez le volet **Propriétés**.



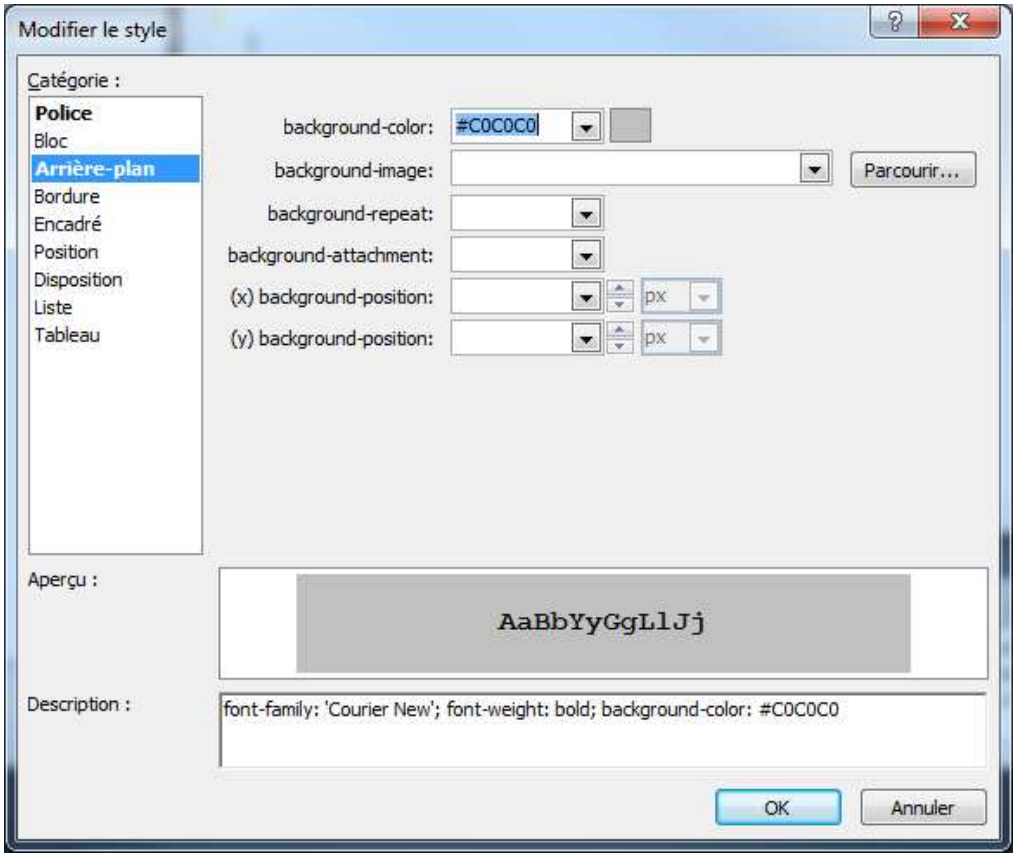
Cliquez sur Style pour obtenir la fenêtre suivante :



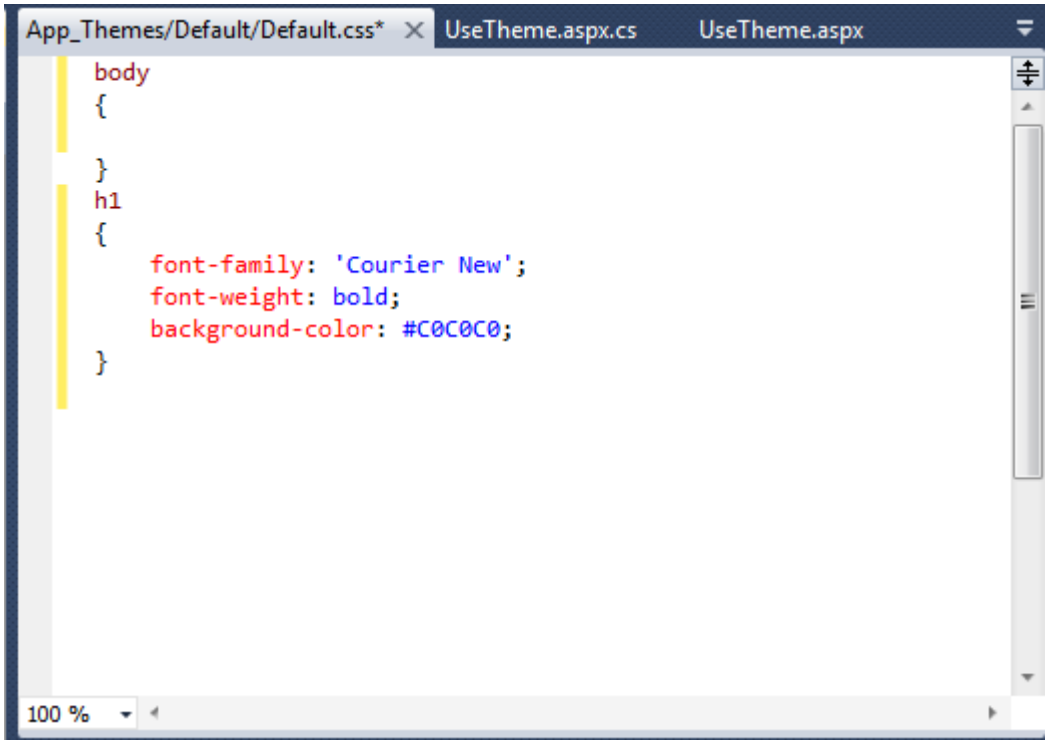
Modifiez la police comme suit :



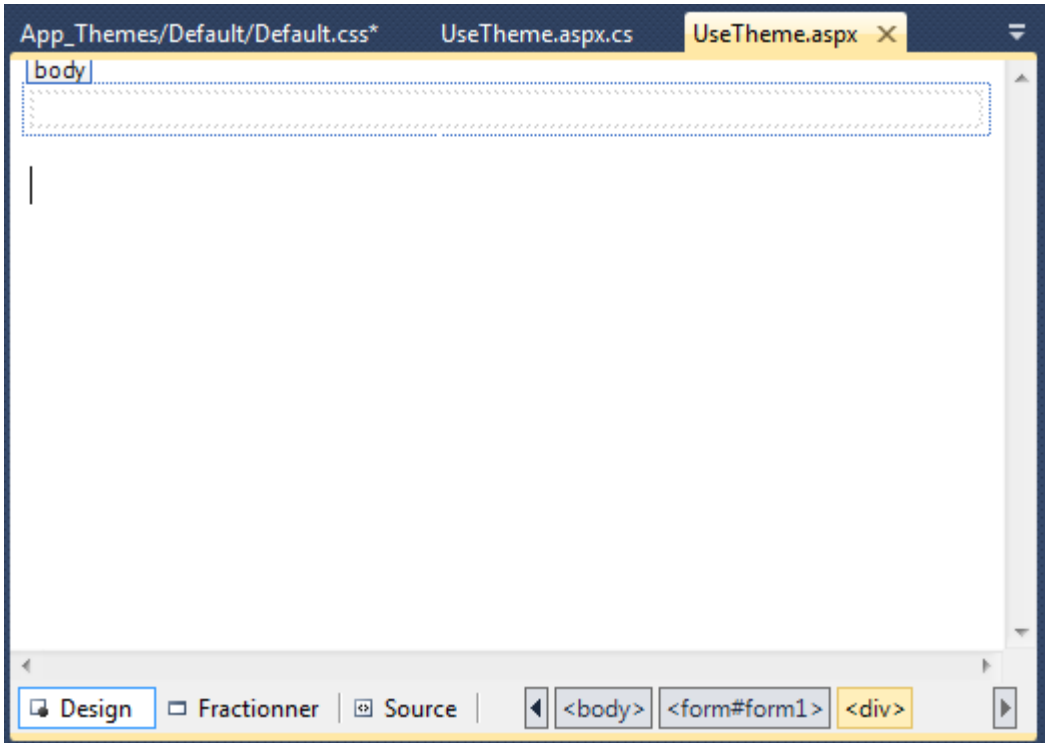
Modifiez l'arrière-plan.



Faire OK. Le fichier css doit ressembler à ce qui suit :



Revenez à la page **Usetheme.aspx**.



Le code source doit ressembler à ce qui suit :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="UseTheme.aspx.cs"
Inherits="UseTheme" %>

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

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

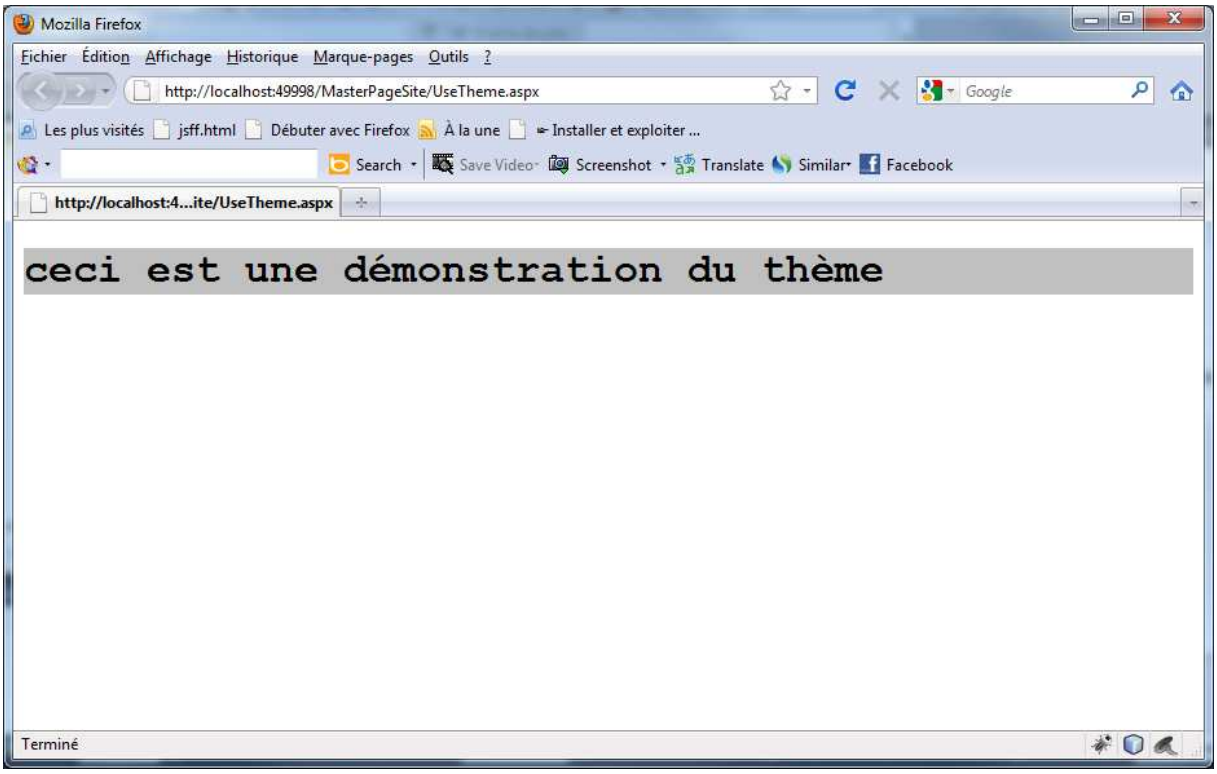
Modifiez la section **div** comme suit :

```
    <div>

    <h1> ceci est une démonstration du thème </h1>

    </div>
```

Et la première ligne du fichier comme suit :

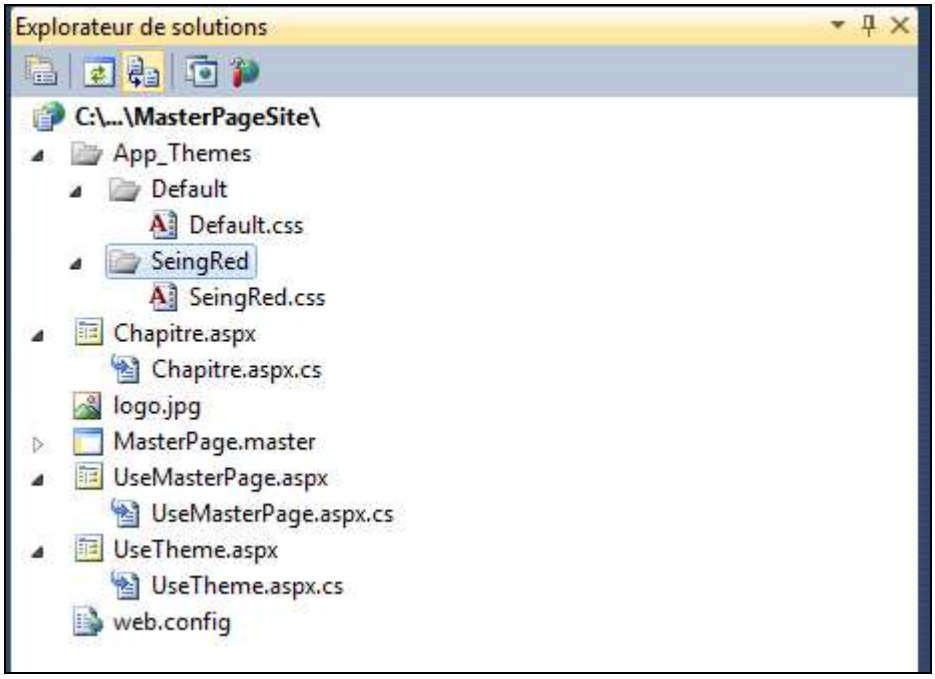
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="UseTheme.aspx.cs"
Inherits="UseTheme" Theme="Default" %>
```

Ce qui donnera à l'exécution :

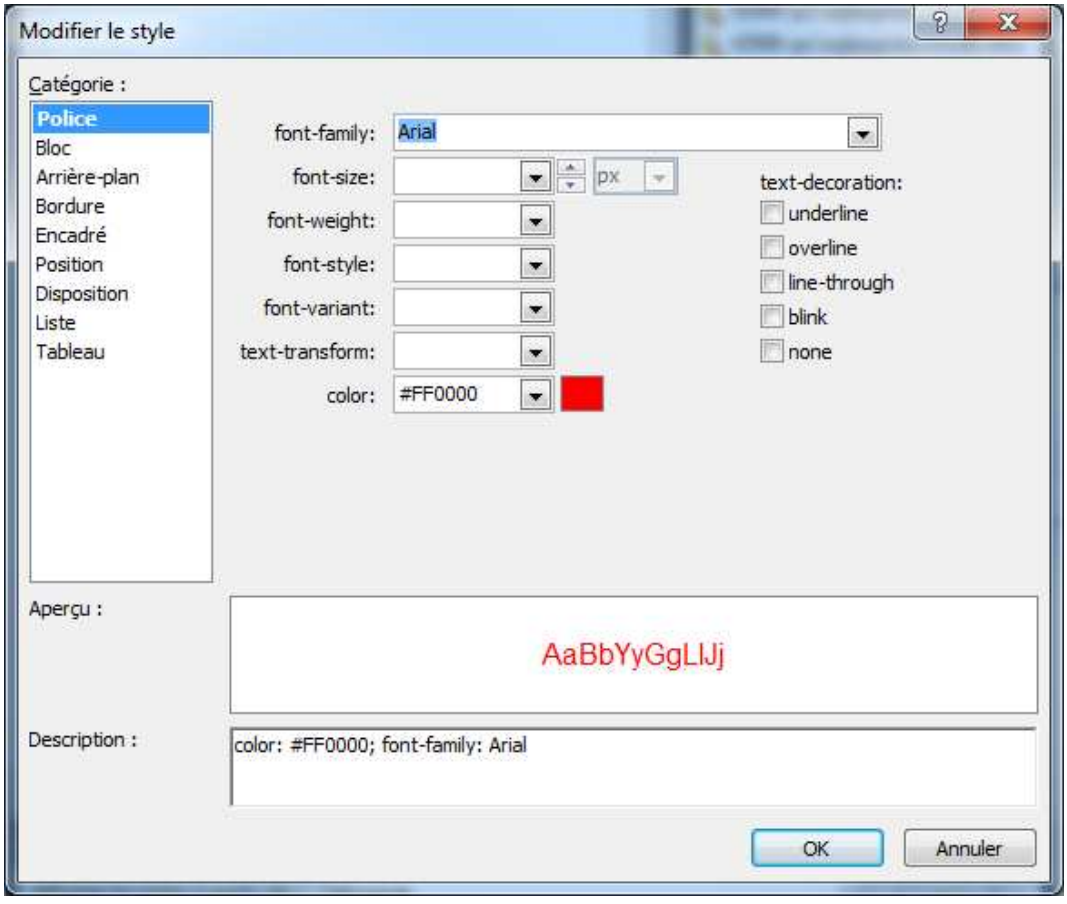


On peut recommencer l'opération pour créer un thème nommé SeingRed par exemple.

Le projet devrait ressembler à ceci :



Modifiez ensuite comme précédemment le fichier **SeingRed.css**.

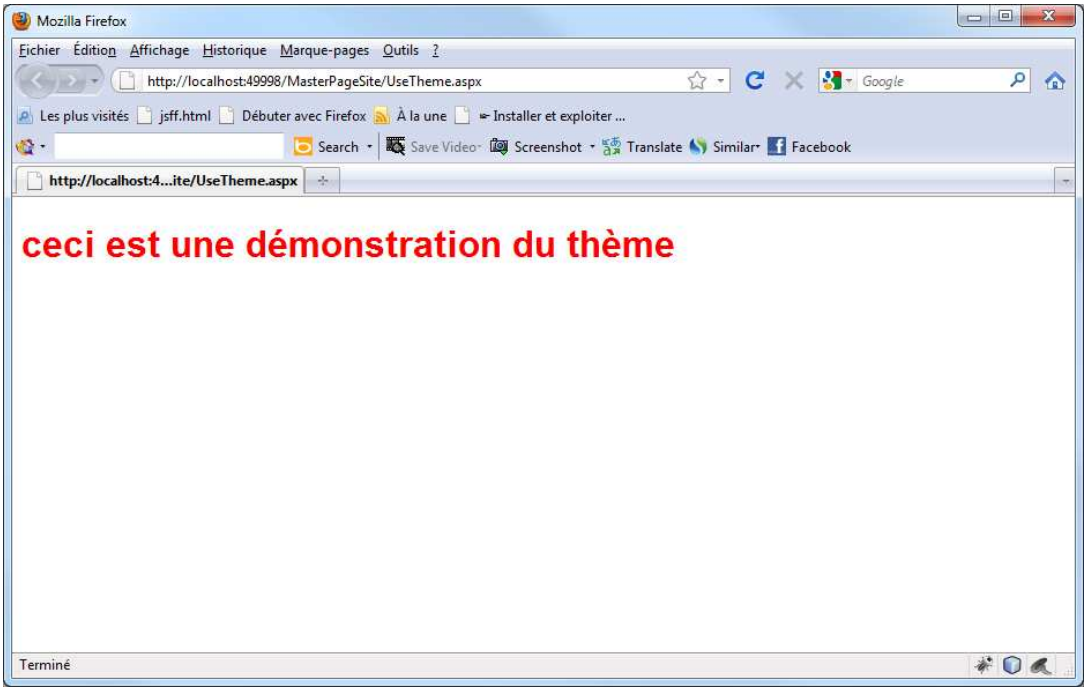


Le nouveau fichier doit ressembler à ceci (par exemple).

```
body {
}
h1
{
    color: #FF0000;
    font-family: Arial;
}
```

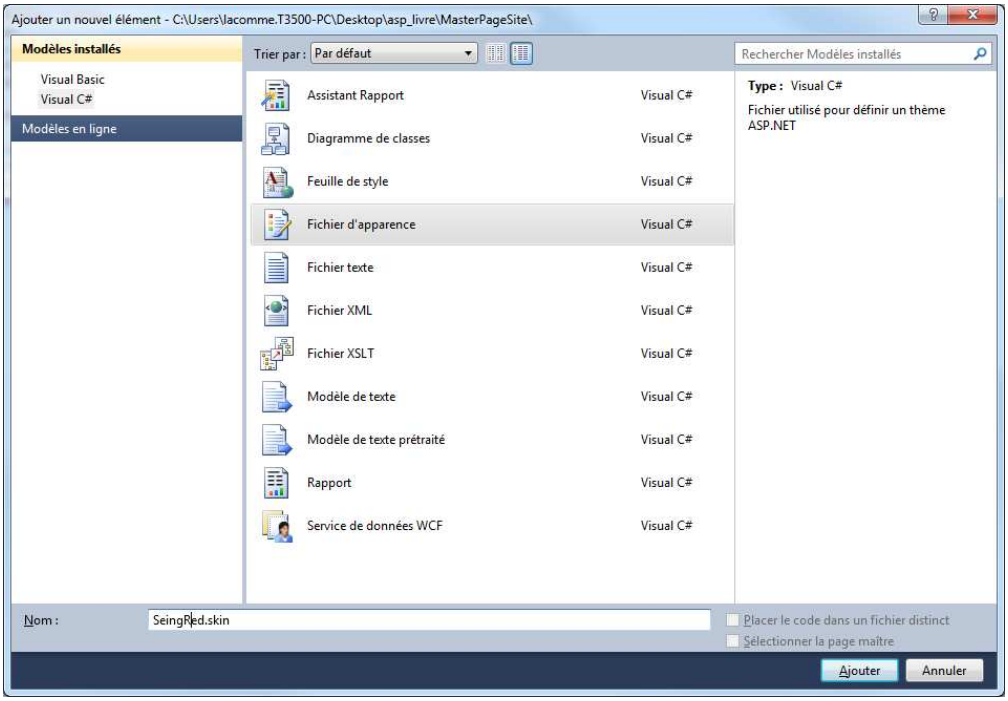
On peut modifier le fichier **UseTheme.aspx** en faisant référence au thème SeingRed.

La page UseTheme.aspx se visualisera alors comme ceci :



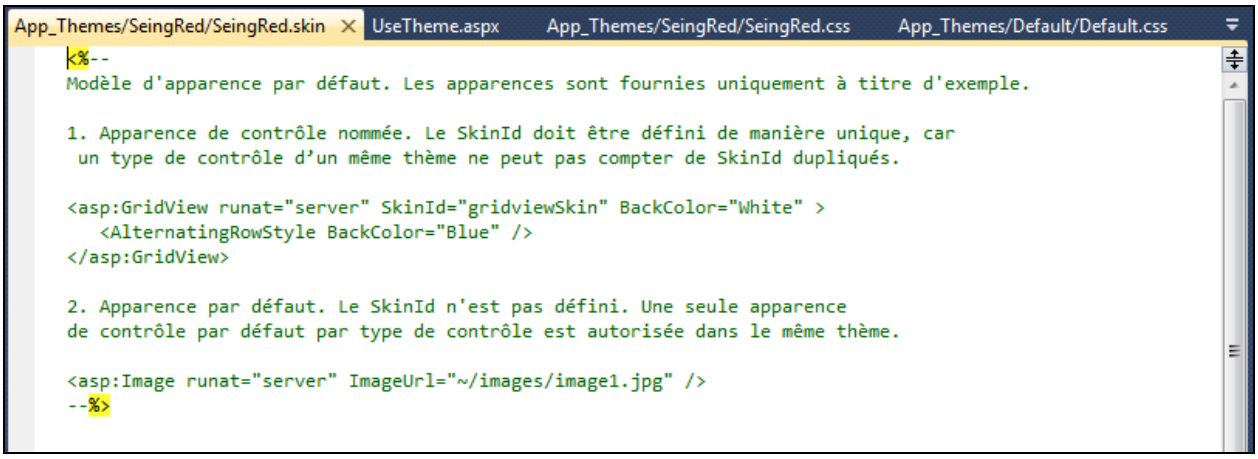
8.3) Utilisation d'un **Skin**

Dans le même projet, créer un fichier **Skin** (apparence) nommé **SeingRed.skin**



Répondre oui si on vous pose la question :

Cela devrait donner ceci :



Ajoutez dans ce fichier, les contrôles pour lesquels on souhaite définir des valeurs de propriétés par défaut.

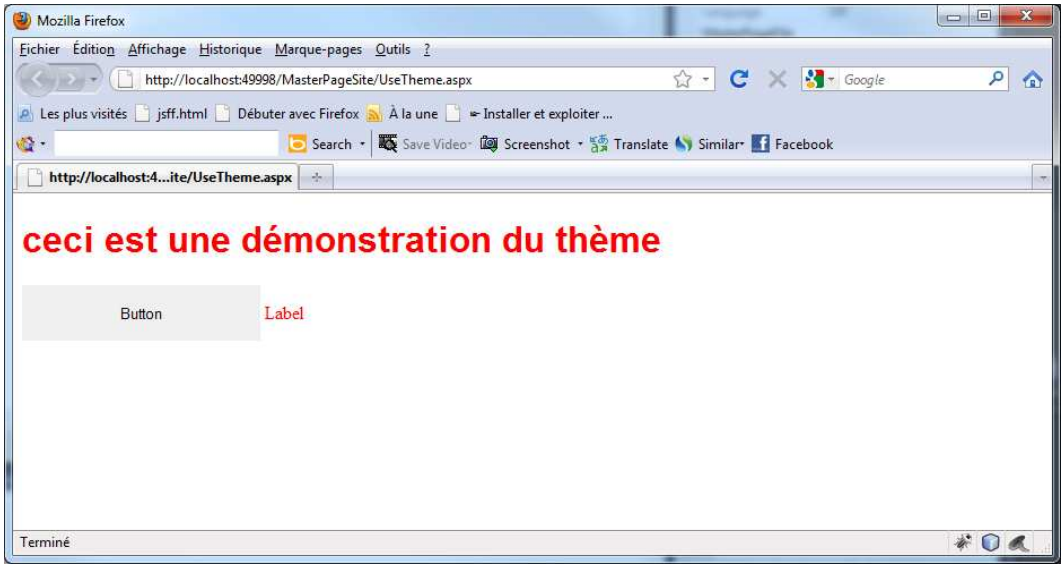
On peut par exemple, inclure le code ci-dessous :

```
<asp:Label runat="server" ForeColor="red" FontSize="14pts" FontName="vernada" />

<asp:Button runat="server" BorderStyle="Solid" Borderwidth="2px"/>

<asp:RadioButtonlist runat="server" ForeColor="#ff9999" />
```

Revenir à la page **UseTheme.aspx**.

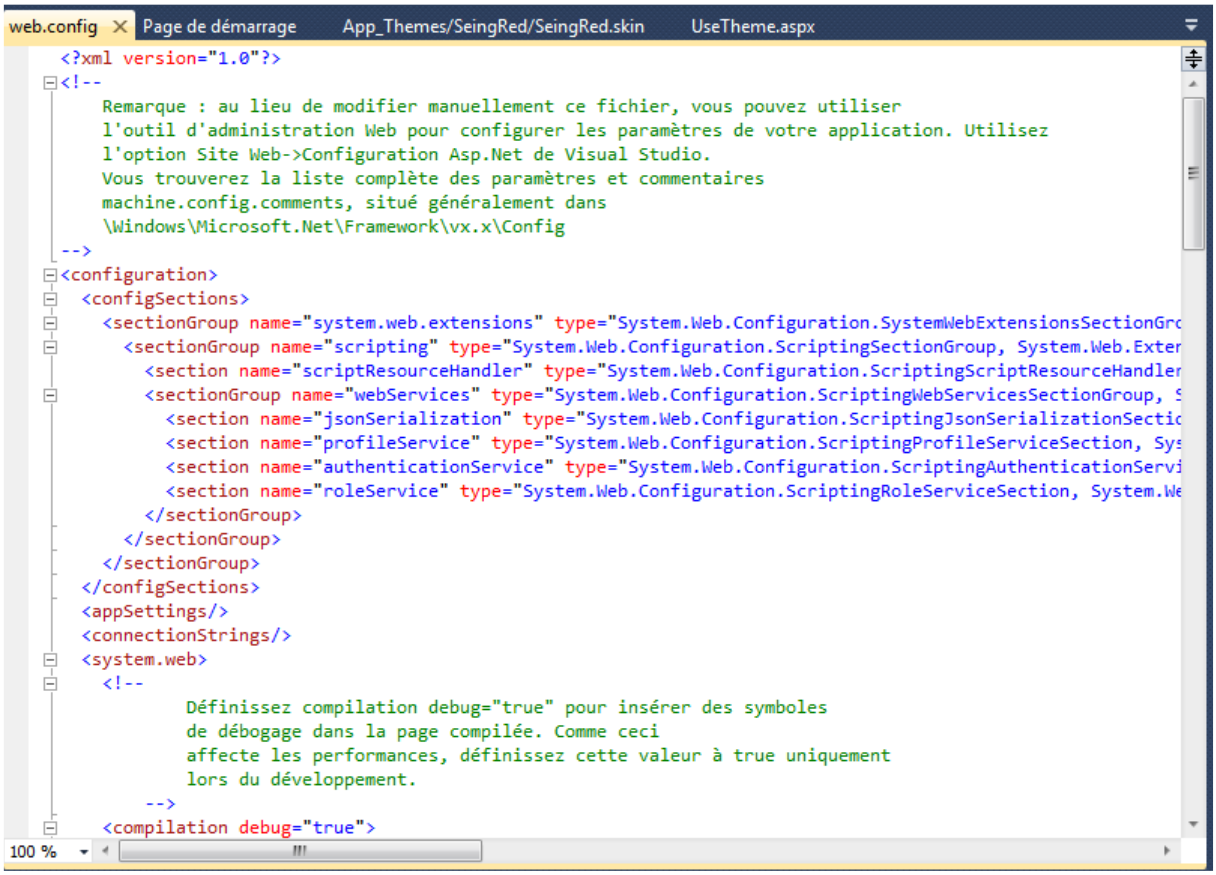
Posez sur la page quelques éléments concernés par le skin : un bouton, un label etc…

A l'exécution vous devriez obtenir ceci:



9) Connexion des utilisateurs

Ouvrir le fichier **web.config**.



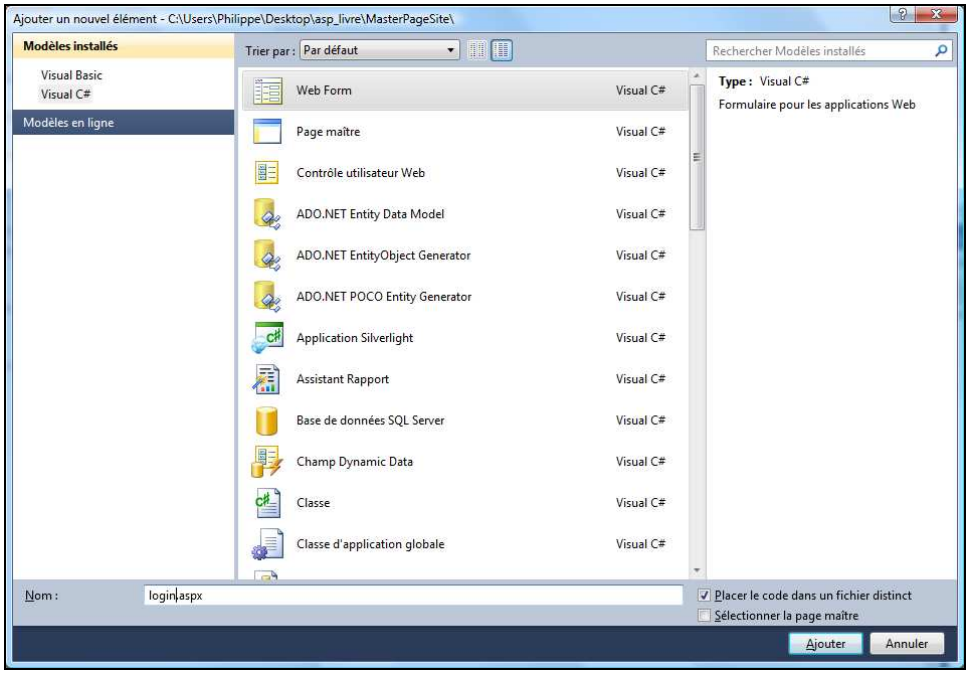
Rechercher dans le fichier la ligne :

<authentication mode="Windows"/>

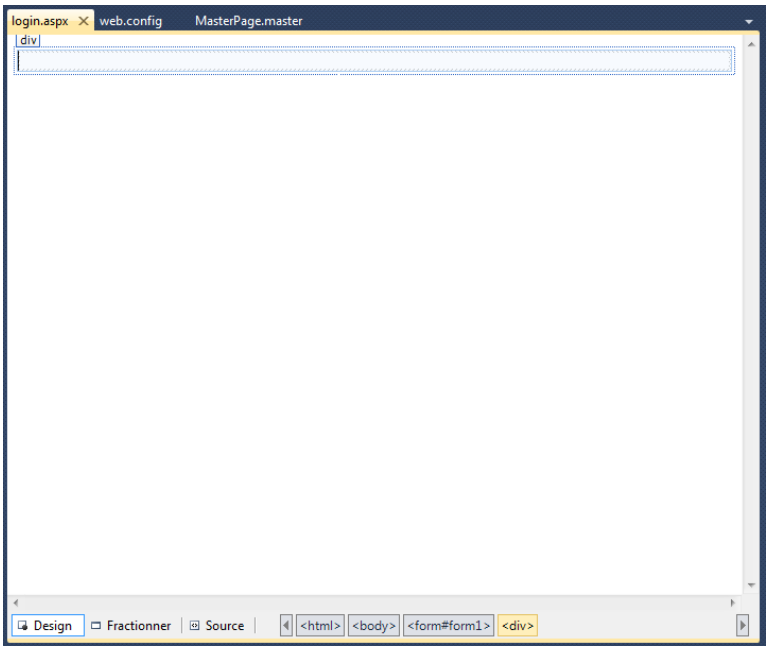
Modifiez le fichier en remplaçcant la ligne par ce qui suit. Cela permet de lancer la page « login.aspx » (que nous allons créer juste après) afin de demander l'authentification à l'utilisateur avant de l'autoriser à voir le contenue du site web.

```
    <authentication mode="Forms">
      <forms loginUrl="login.aspx" />
    </authentication>
    <authorization>
      <deny users="?"/>
    </authorization>
```

Créer un fichier nommé **login.aspx**.

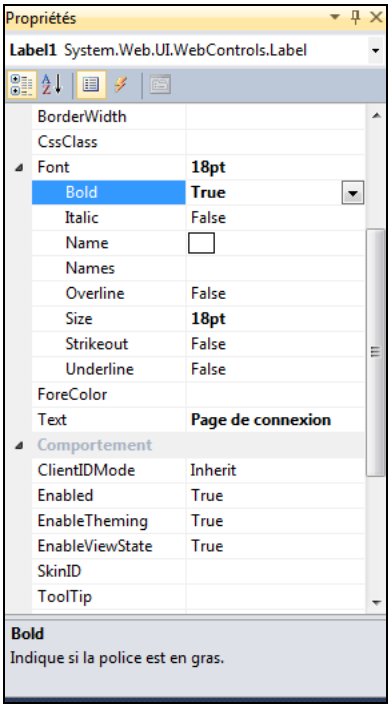


Basculer l'affichage en mode **Design**.

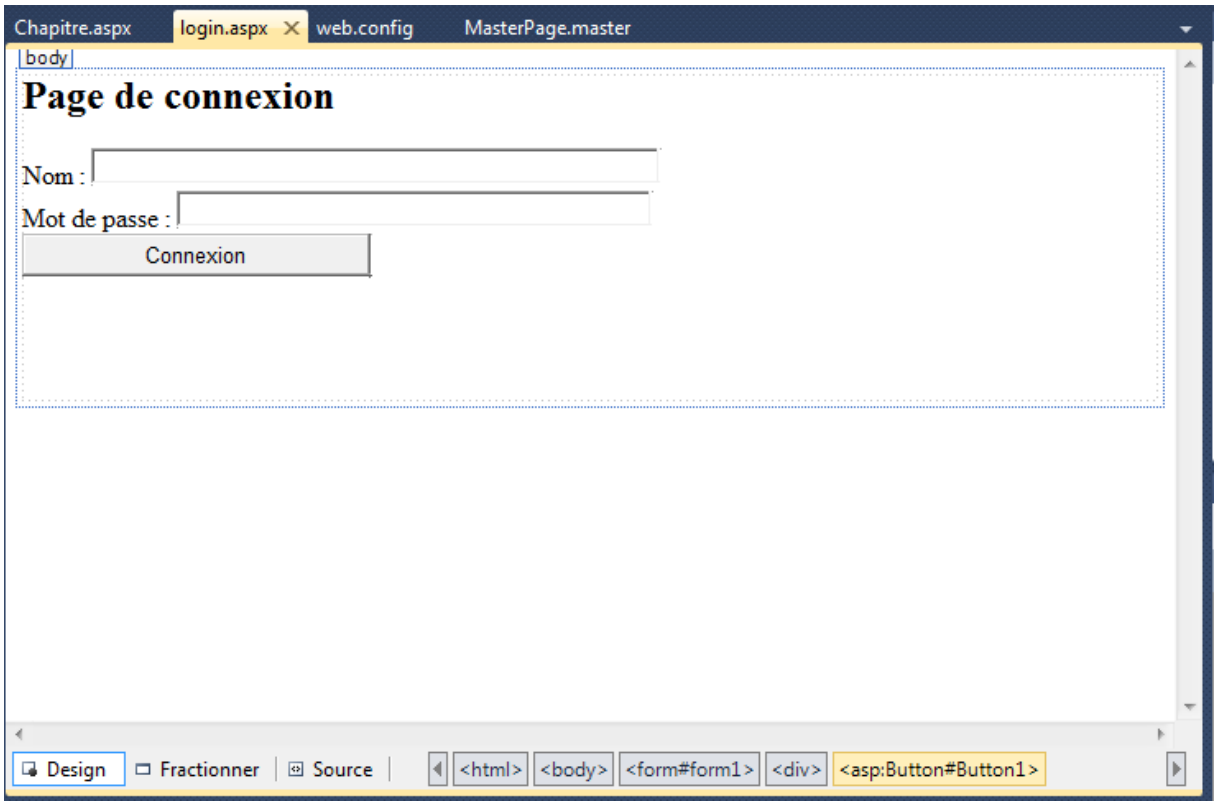


Donner un titre à la page en utilisant un label :

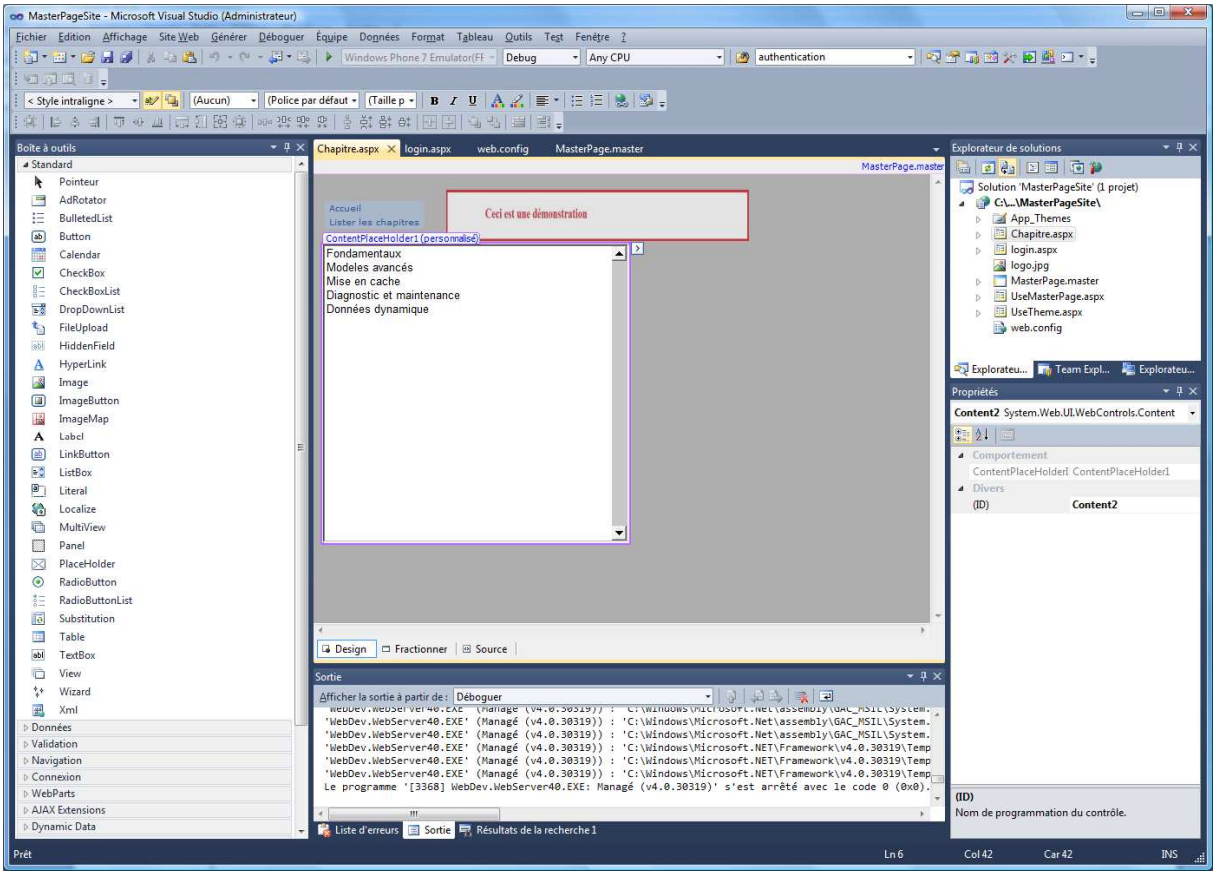
En utilisant les attributs passer la police en taille 18 et en **bold**.



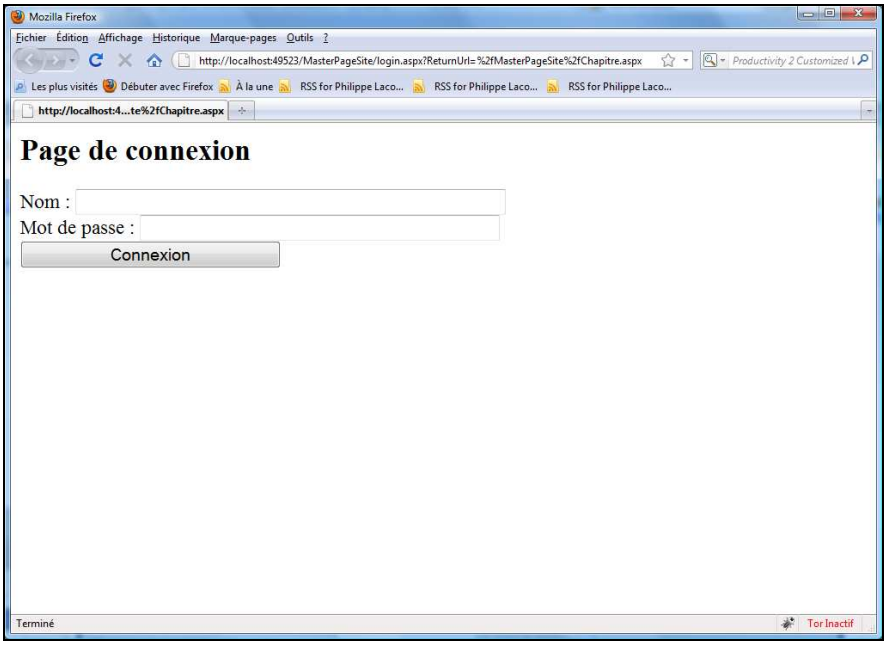
Complétez ensute la page en incluant des champs de saisies et un bouton de connexion comme indiqué ci-dessous :



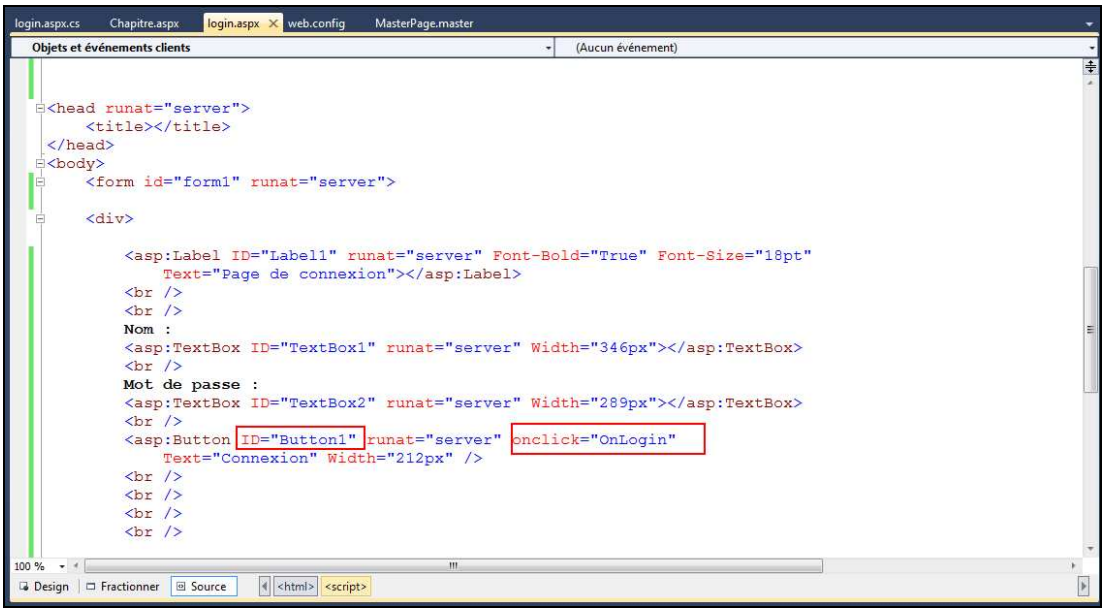
Ouvrez la page que nous avons précédemment crée nommée **Chapitre.aspx**



A l'exécution vous allez obtenir la page **login.aspx** de manière automatique.



Editez le code source de la page **login.aspx**. Nous allons inclure dans cette page des scripts qui seront exécutés coté serveur. Vérifier bien la cohérence de vos éléments graphique car sinon, le code suivant ne marchera pas. Notamment, ajouter un appel de fonction à votre bouton…



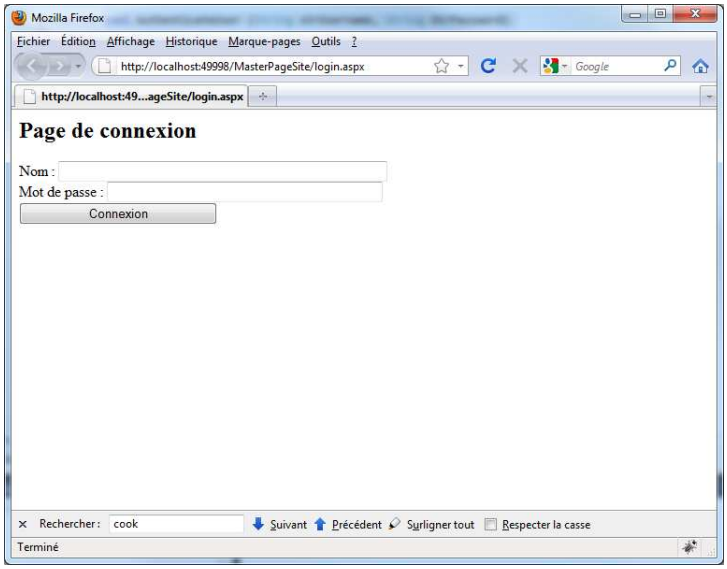
Avant la section head et après le tag html il faut créer deux procédures nommées :

- **AuthenticateUser**
- **OnLogin**

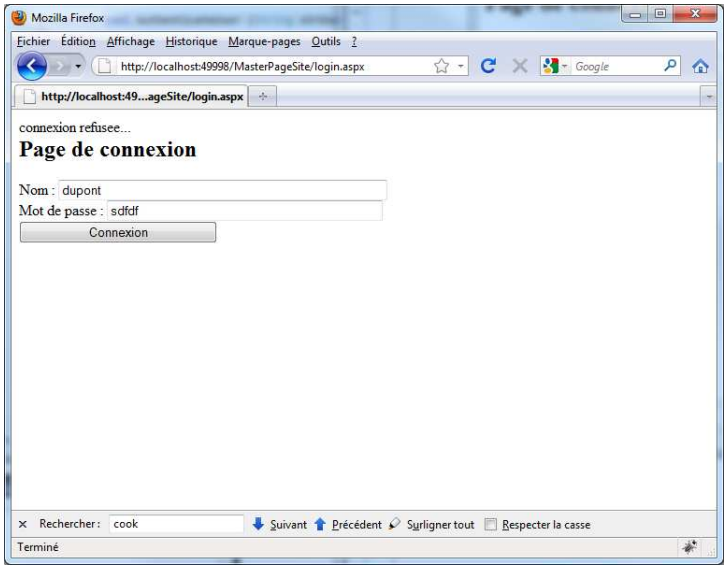
```
<html xmlns="http://www.w3.org/1999/xhtml">

    <script runat=”server” >

    protected bool AuthenticateUser (String strUsername, String StrPassword)
    {

      if (strUsername=="toto")
      {
        if (StrPassword=="aaa")
          return true;
      }

      return false;
    }


    public void OnLogin (Object src, EventArgs e)
    {
        if (AuthenticateUser(TextBox1.Text, TextBox2.Text) == true)
     {
         Response.Write("connexion OK...");
       //  FormsAuthentication.RedirectFromLoginPage("Chapitre.aspx", true);

         // on ecrit un cookie
         Response.Cookies["UserSettings"]["Font"] = "Arial";
         Response.Cookies["UserSettings"]["Color"] = "Blue";
         Response.Cookies["UserSettings"].Expires = DateTime.Now.AddDays(1d);
```

```
            Response.Redirect("Chapitre.aspx");
        }
        else
          Response.Write("connexion refusee...");

    }

    </script>

<head runat="server">
…
        <asp:Button ID="Button1" runat="server" onclick="OnLogin"
            Text="Connexion" Width="212px" />
…
```

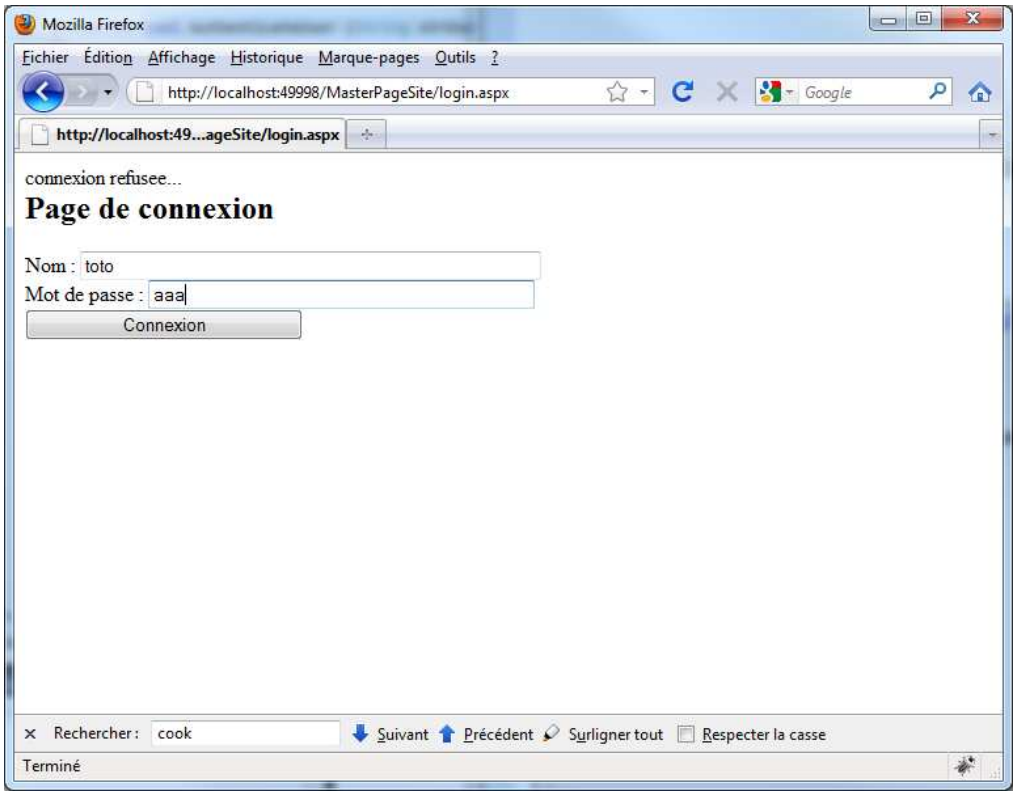
L'exécution de « Chapitre.aspx » donne :



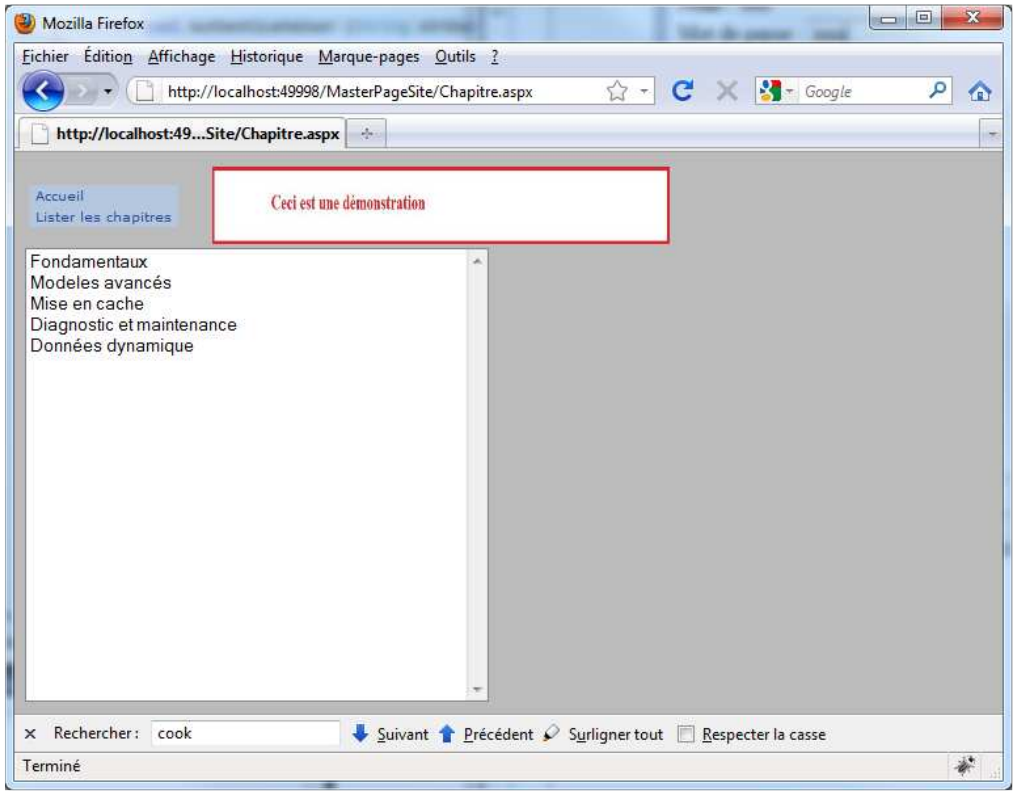
Faites ensuite une saisie erronée : **dupont** avec **sdfdf** comme mot de passe.
La page se réaffiche avec comme message en haut : **connexion refusée**



Entrez ensuite **toto** et comme mot de passe **aaa**.



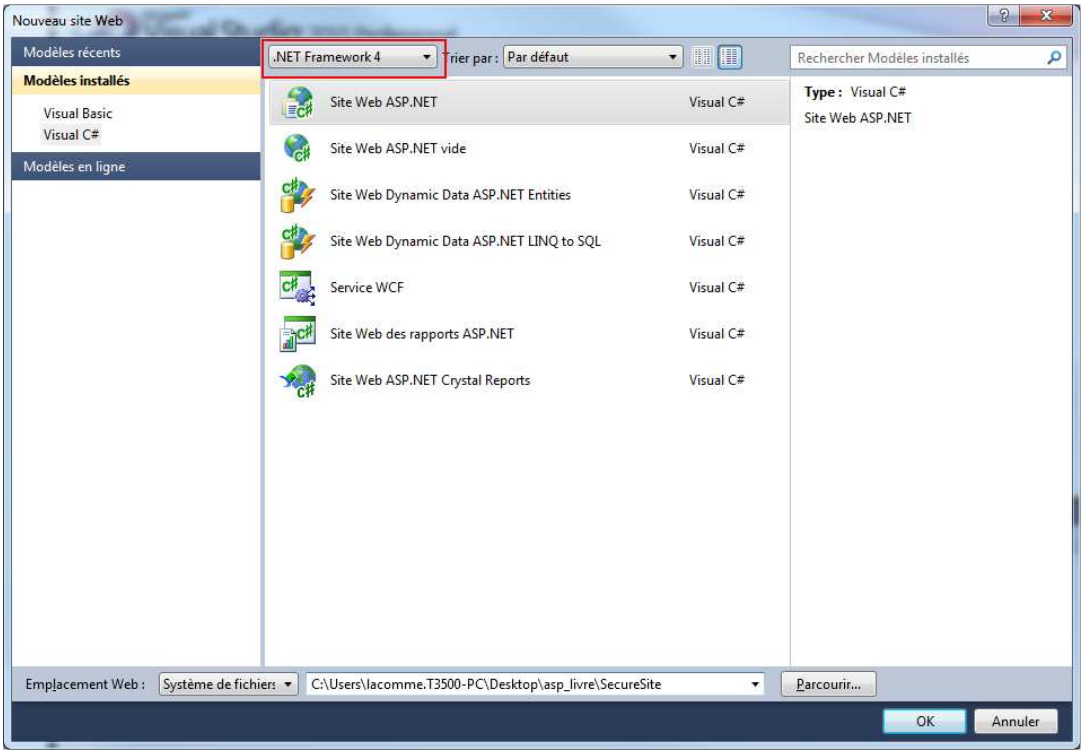
Vous allez obtenir une redirection vers la page **chapitre.aspx**.



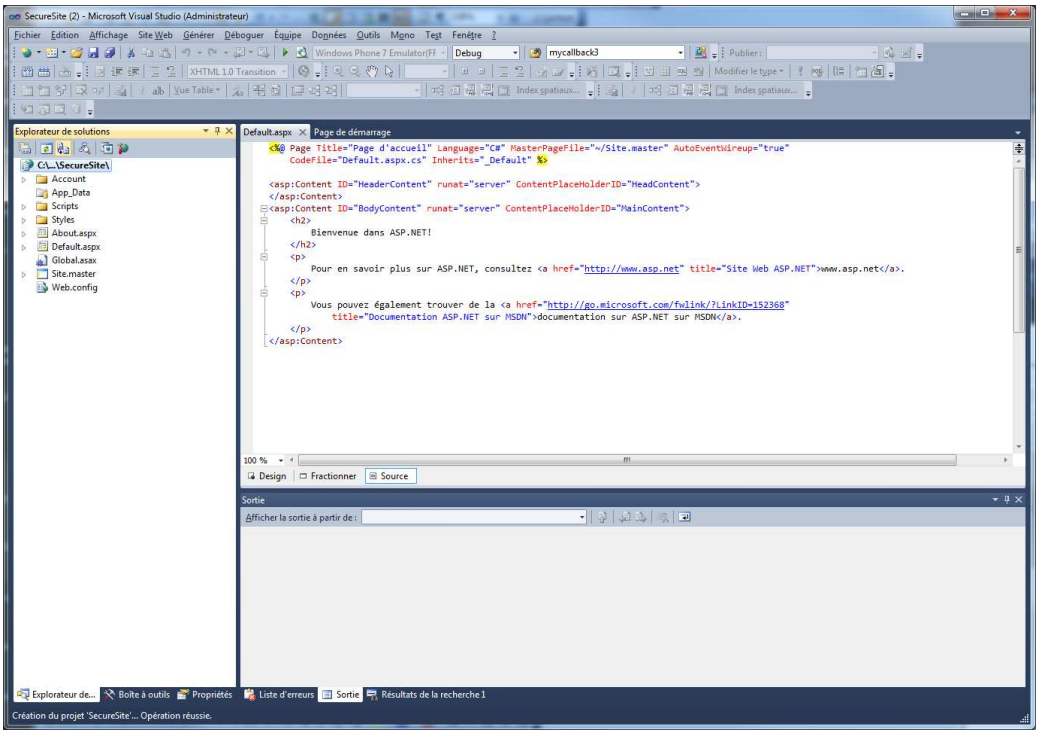
10) Gestion des utilisateurs

10.1 Création du site

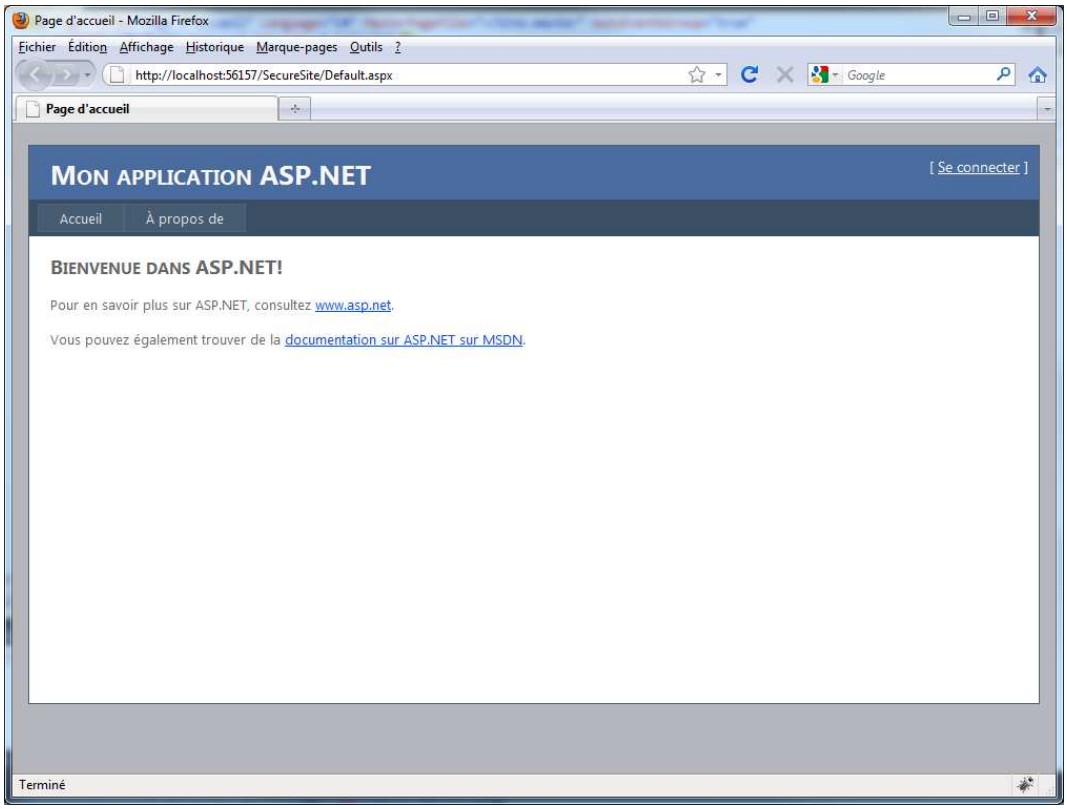
Créer un site web intitulé **SecureSite**. Utiliser bien le .NET Framework 4.



Ce qui doit donner :

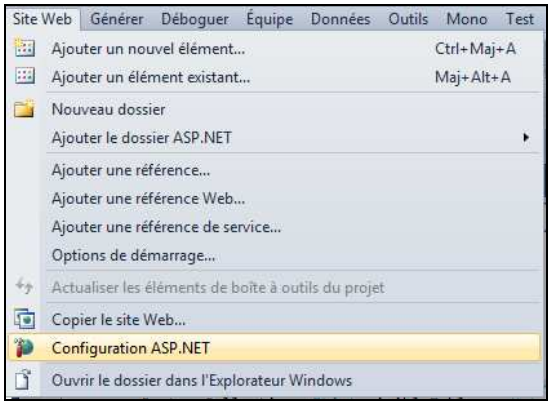


L’exécution doit donner :

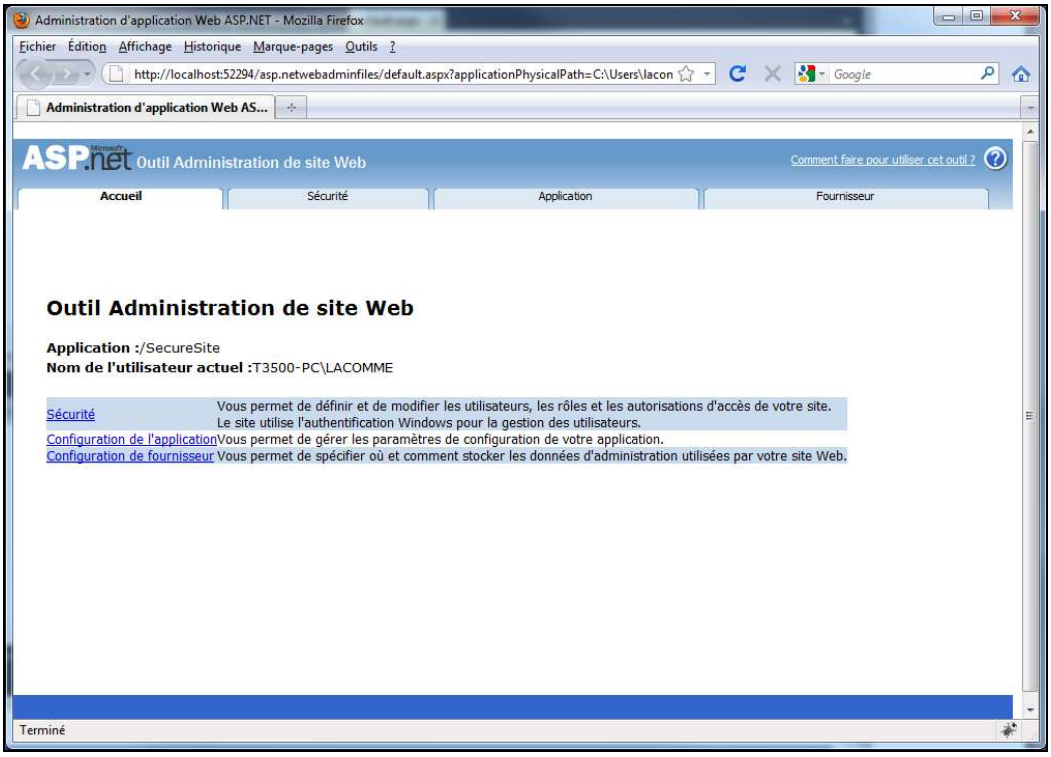


10.2 Configurer ASP.NET

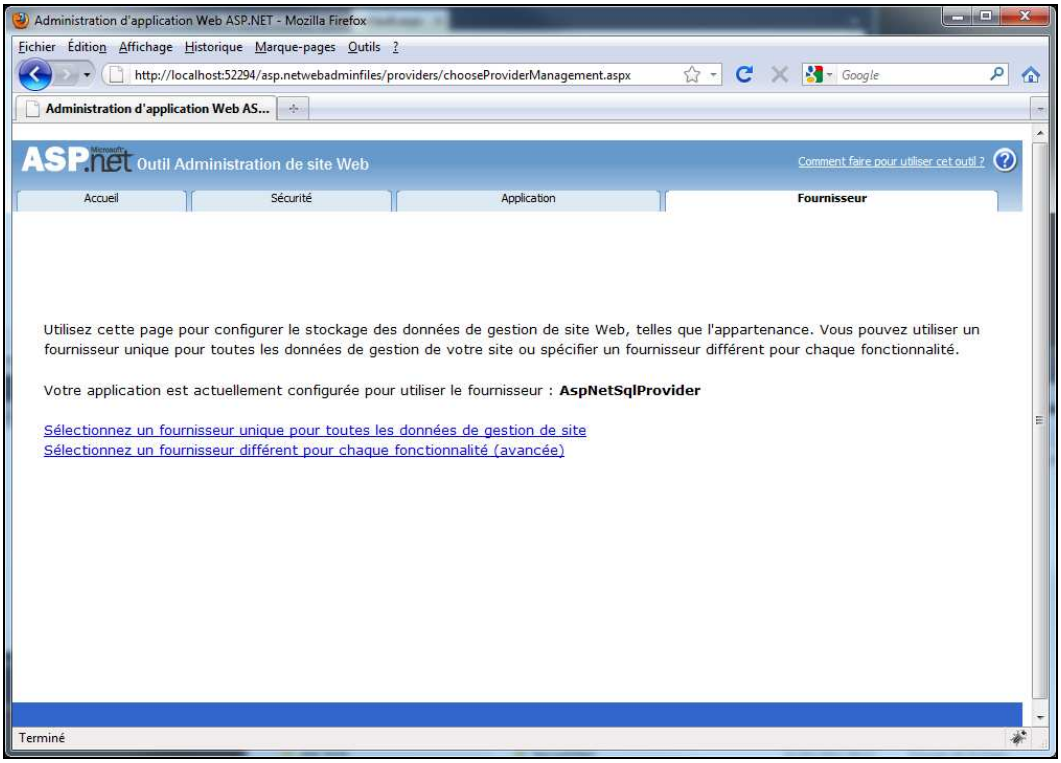
Allez dans le menu **Site Web** et choisir **Configuration ASP.NET**



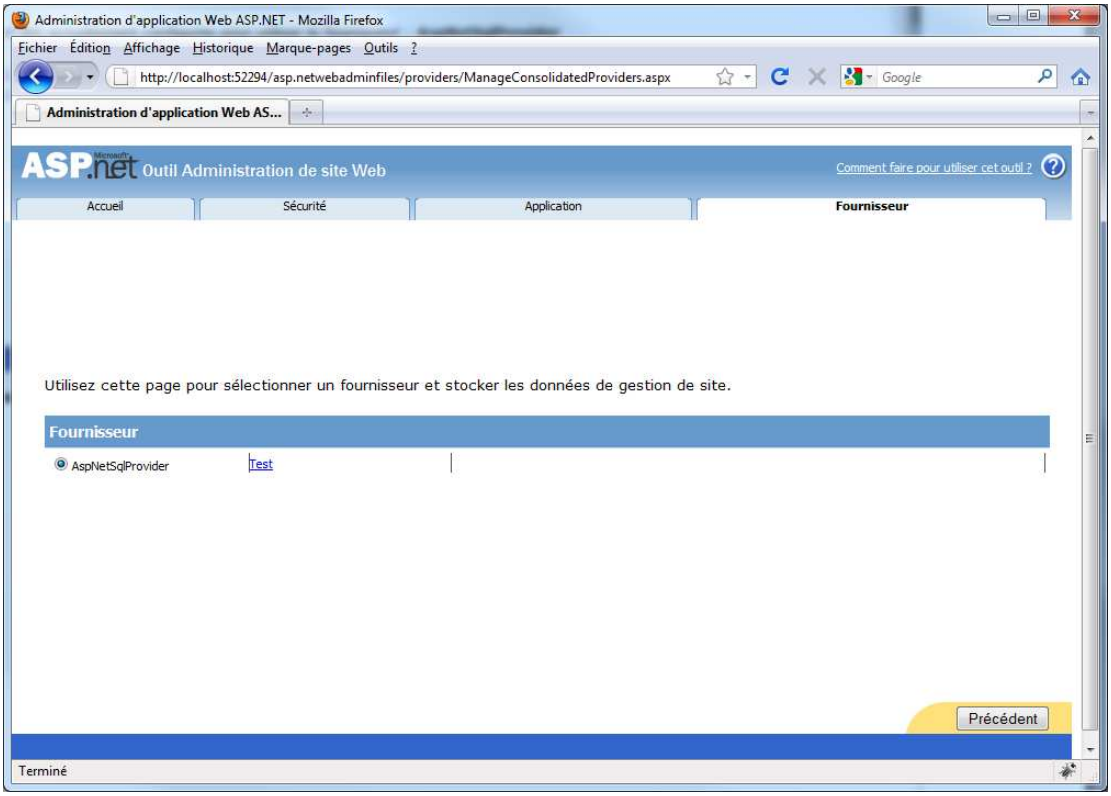
Ceci donne accès aux outils d'administration du site web.



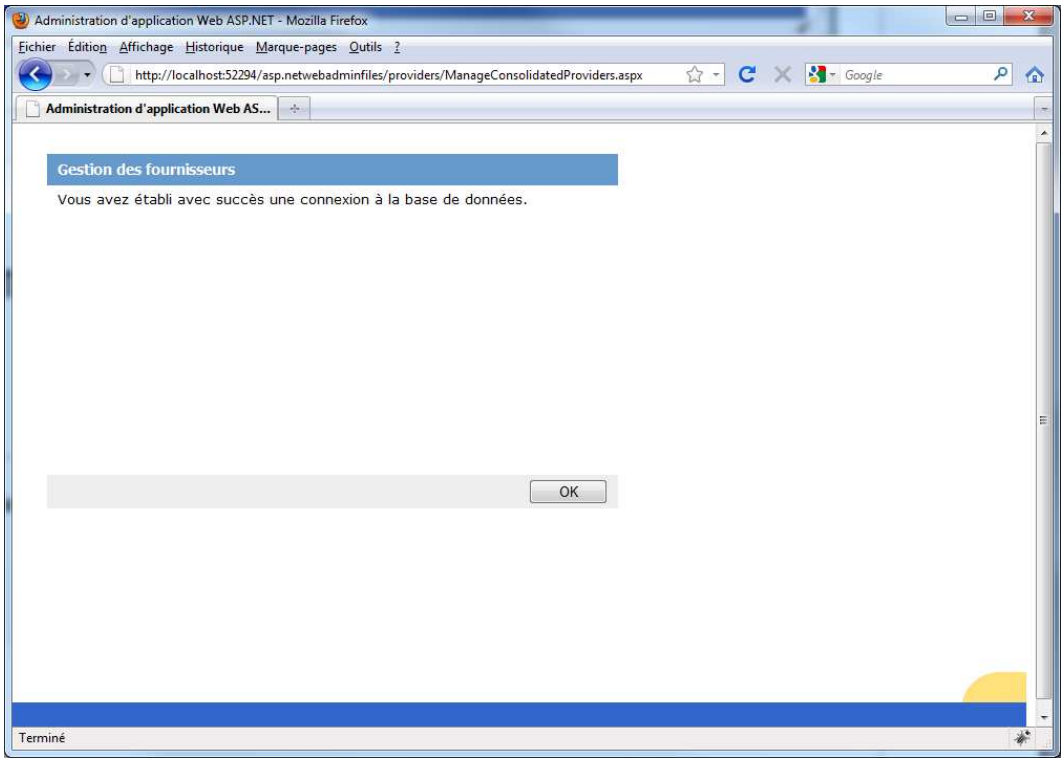
Allez dans l'onglet **Fournisseur**.



Faites le premier choix (fournisseur unique) ce qui vous amènera à la page suivante :

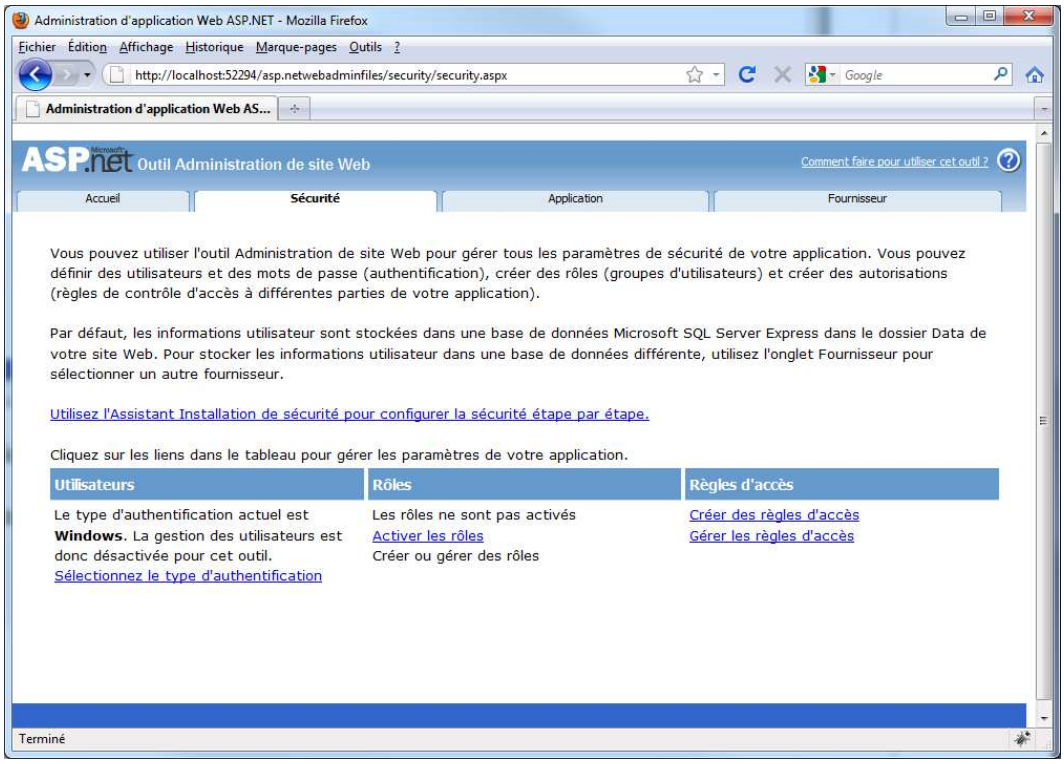


Cliquez sur **test** afin de vérifier que le fournisseur fonctionne correctement.

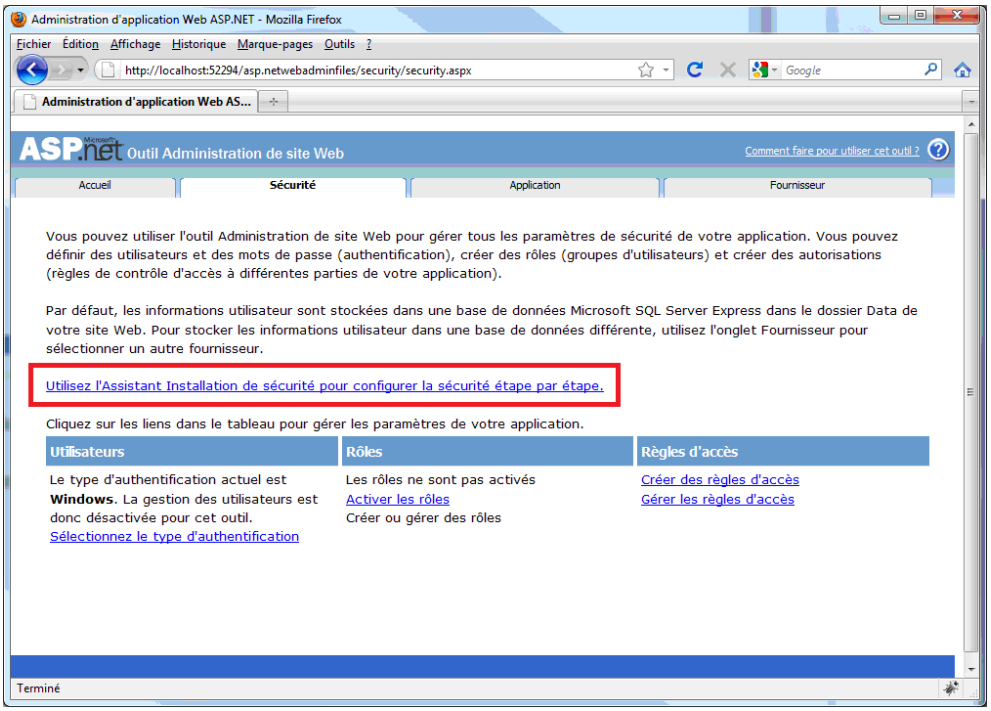


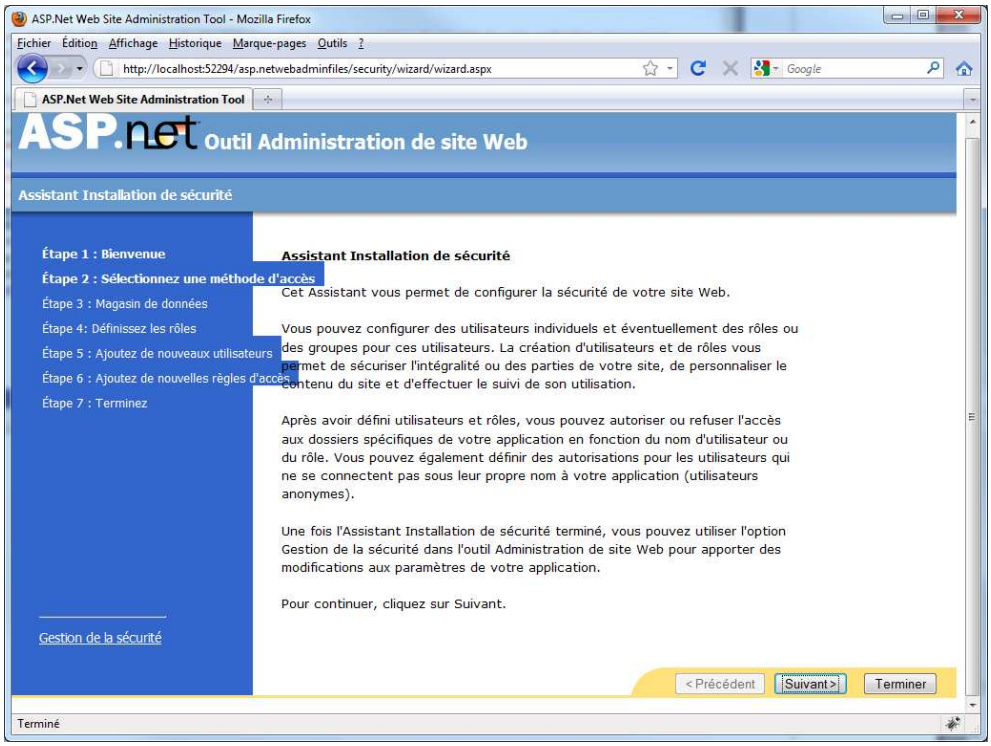
Choisissez le bouton **OK** et revenez à l'écran d'accueil. Si vous n'avez pas pu vous connecter à la base de données cela viens du fait de votre installation de « SqlServer ». Il est possible que ce dernier ne soit pas installé ou bien qu'il n'existe pas de base à laquelle se connecter.

Choisissez maintenant l'onglet **Securite**.

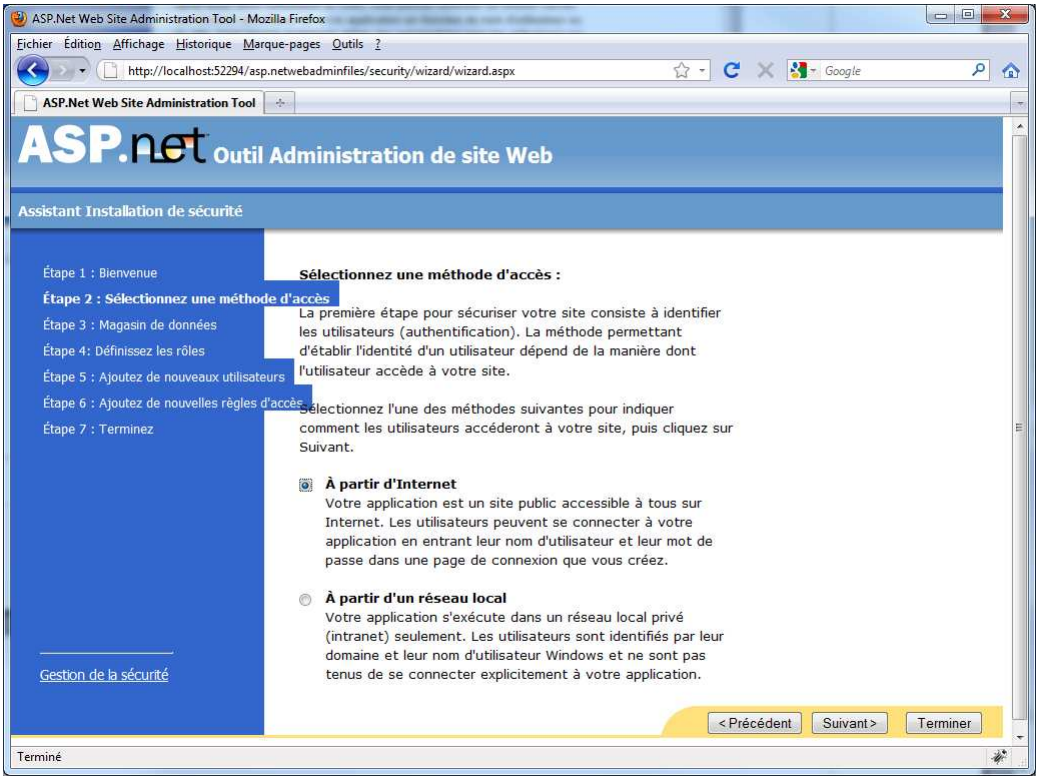


Choisissez l'assistant d'installation.

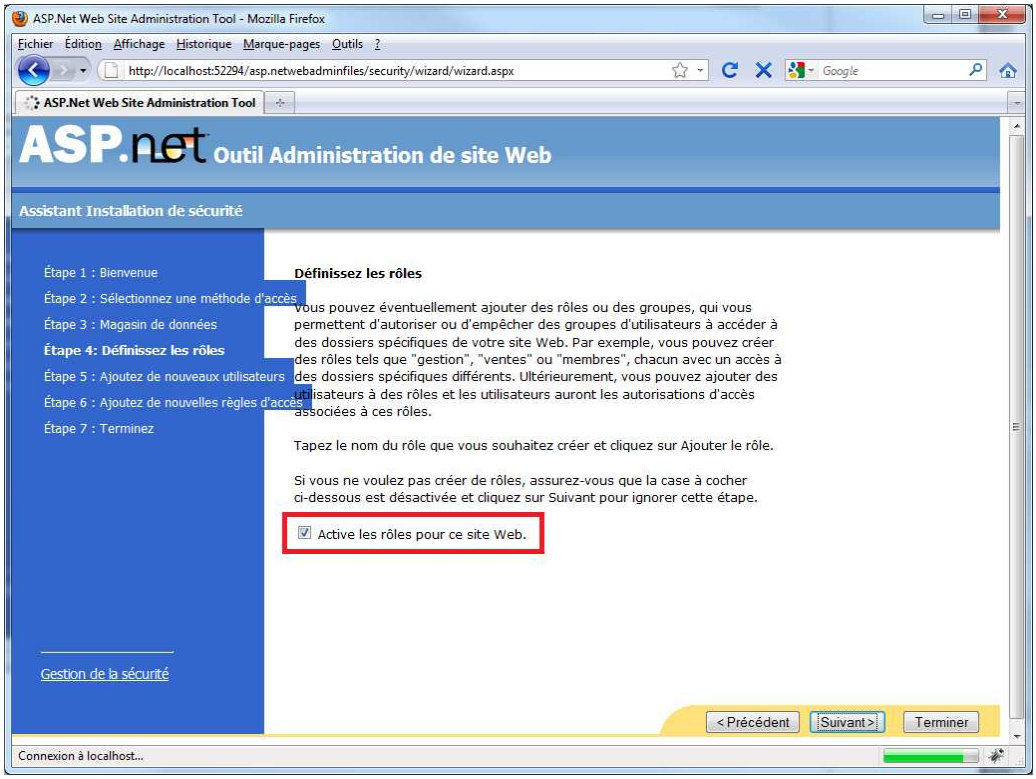


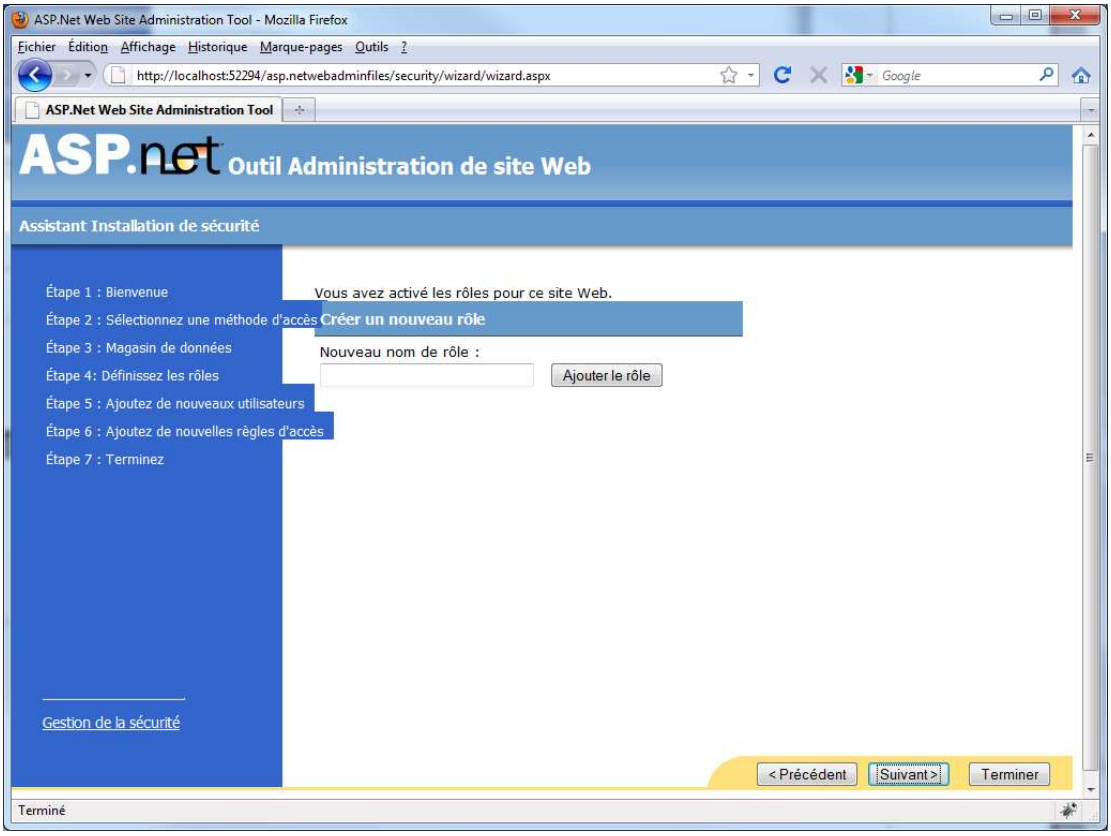


Sur l’étape 2 choisissez Internet comme méthode d’accès.

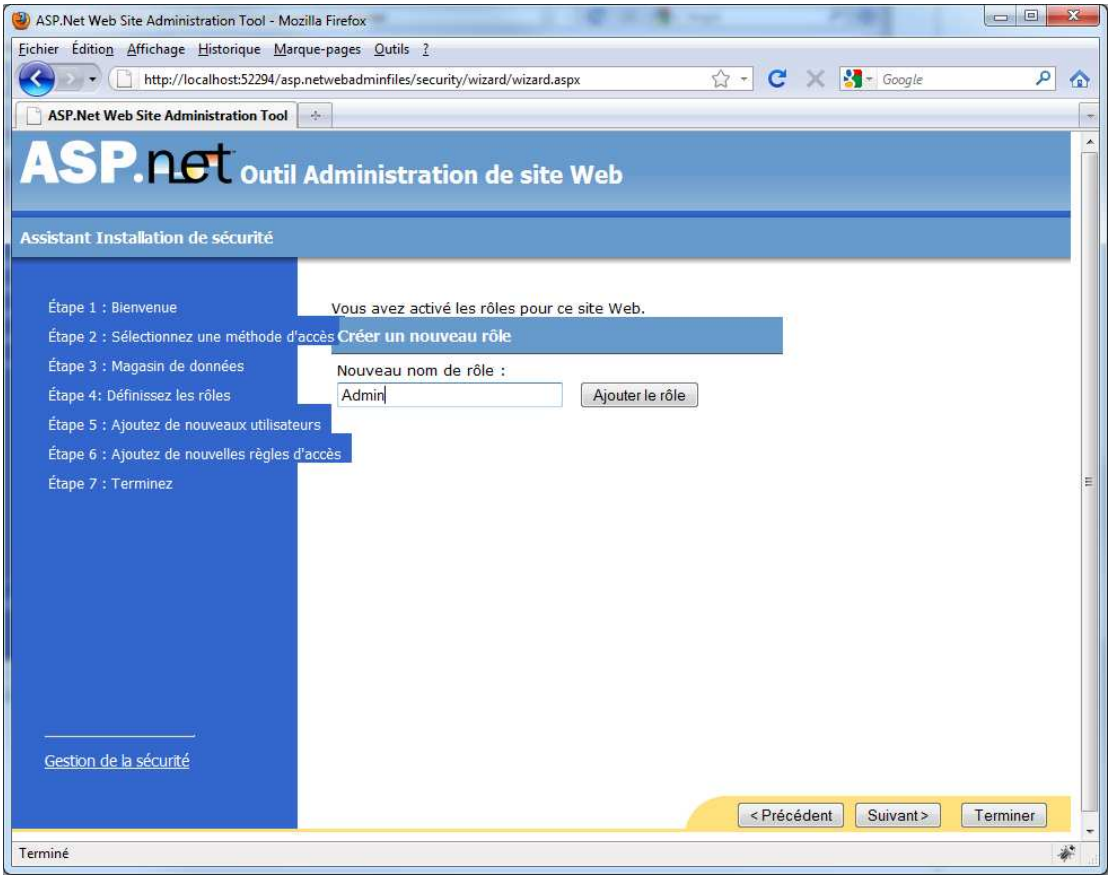


A l'étape 4 activez les rôles comme indiqué ci-dessous.

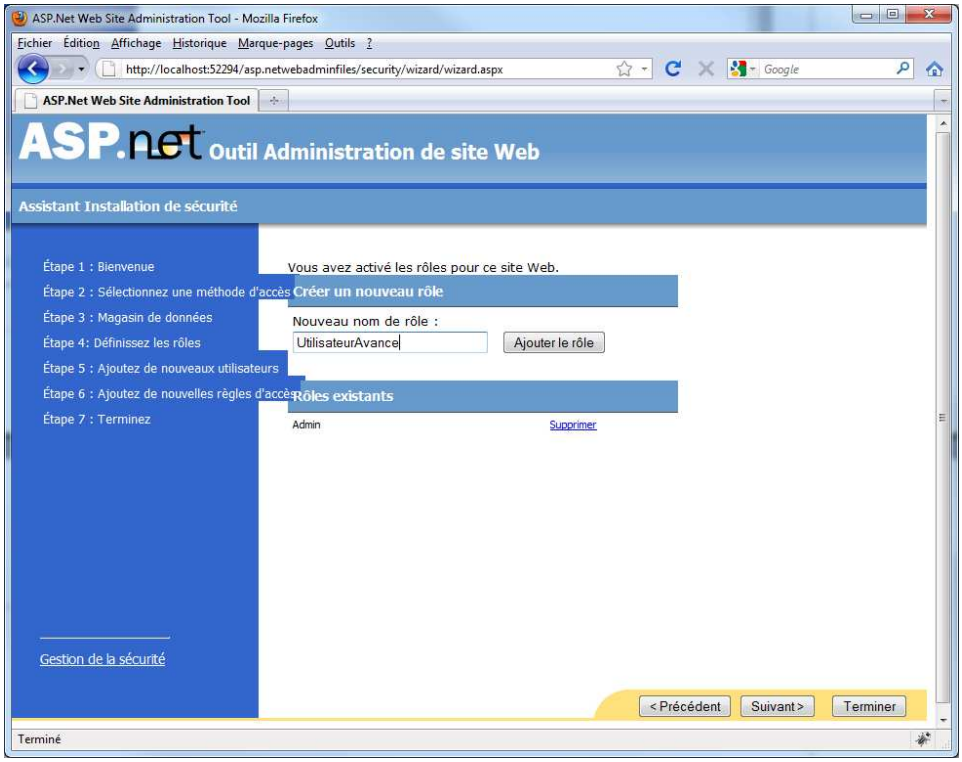




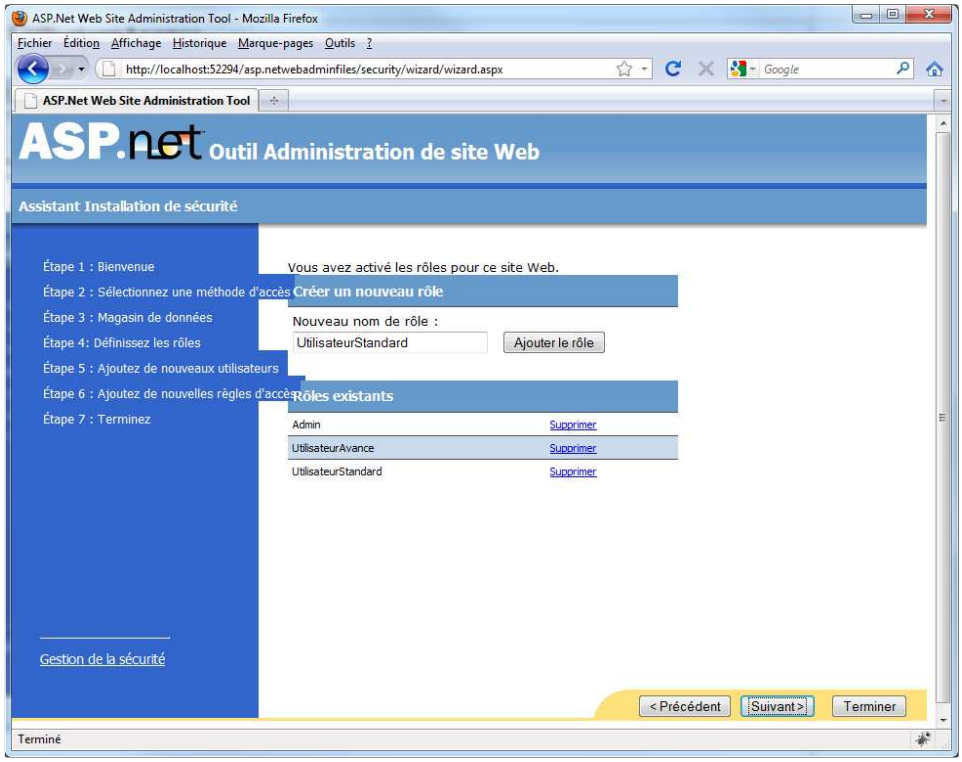
Créez un role **Admin**.



Créez un role **UtilisateurAvance**.



Créez un role **UtilisateurStandard**.

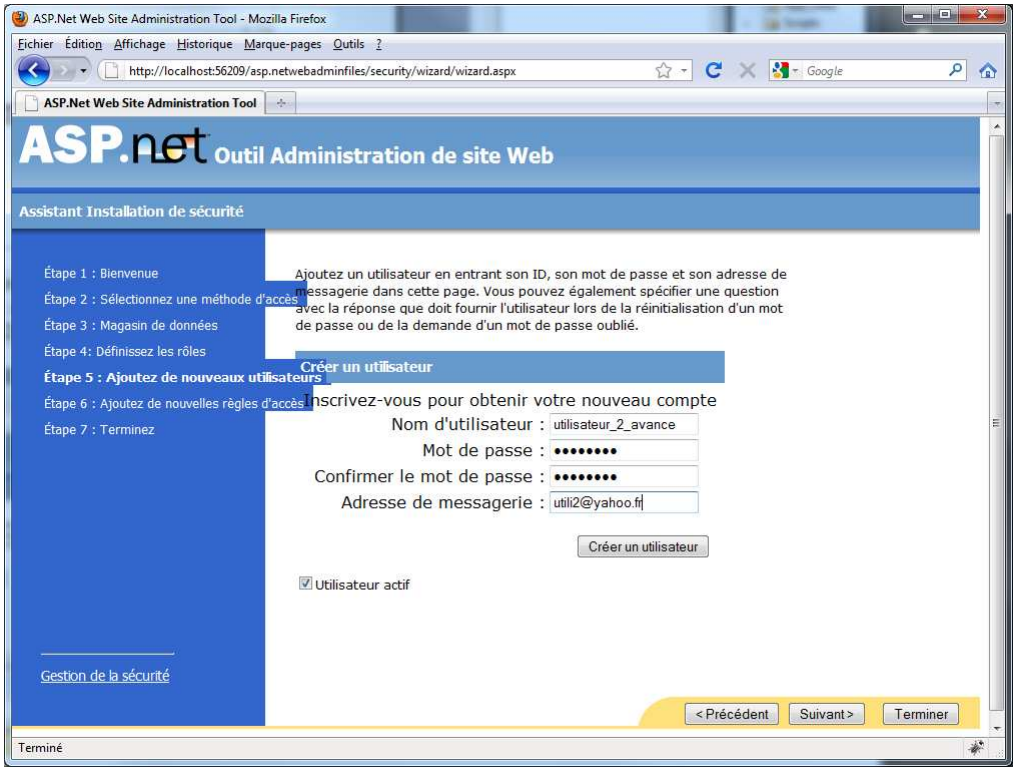


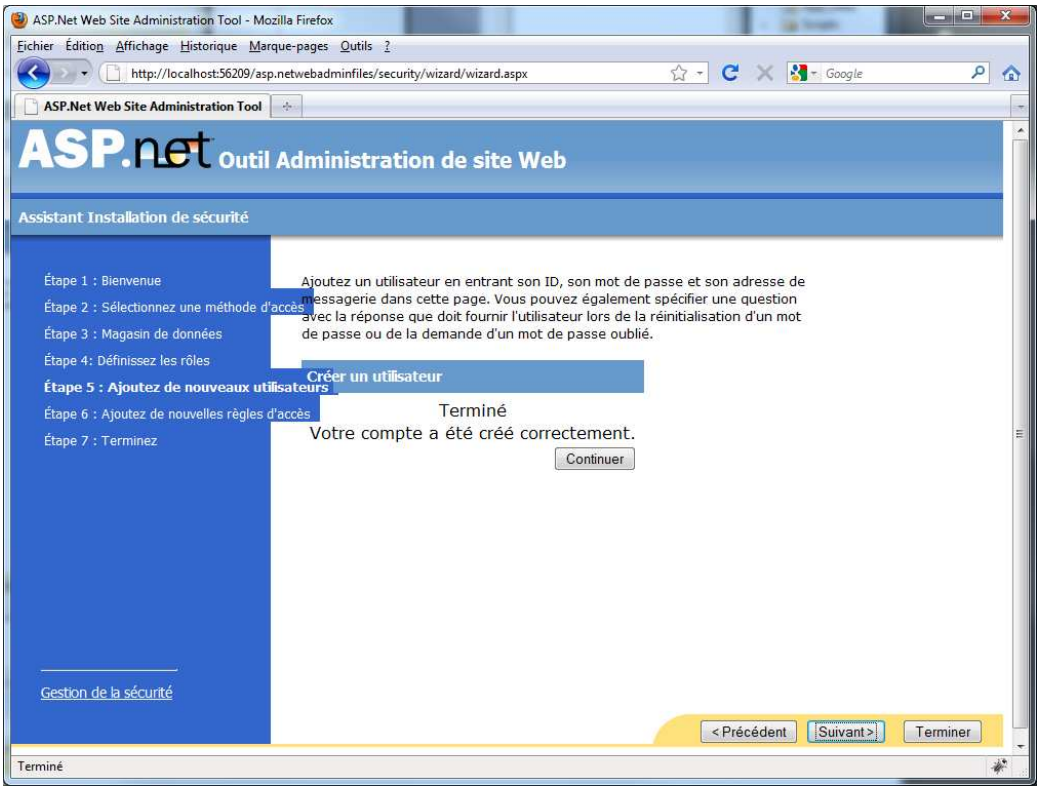
Choisissez le bouton **Suivant** afin de passer à la page permettant de définir de **nouveaux utilisateurs**.

Nous allons créer :

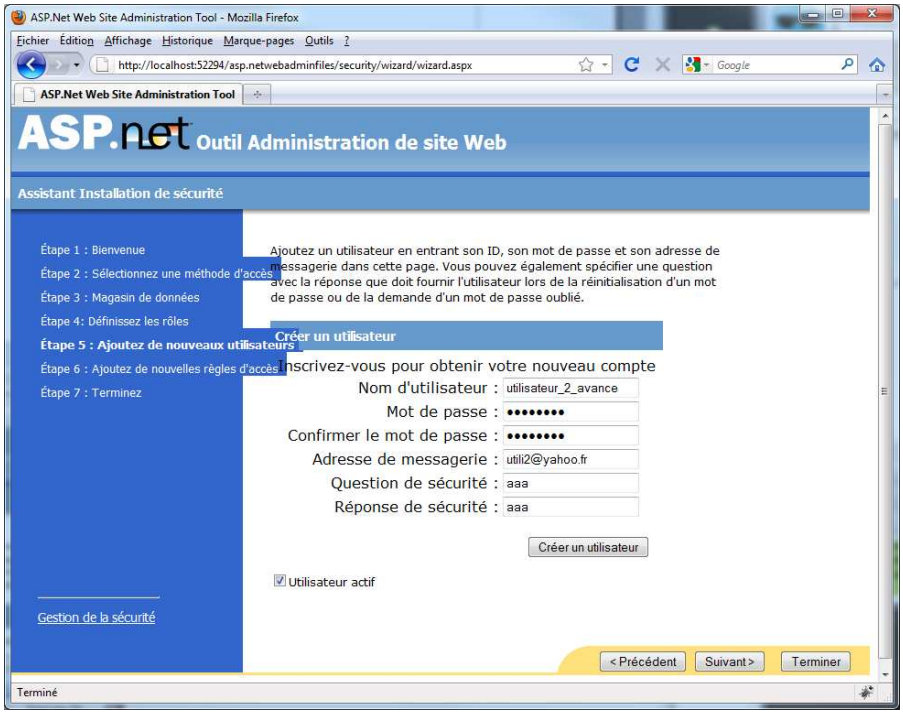
Nom : utilisateur_1_avance
Mot de passe : aaaaaaa;
Adresse : utili1@yahoo.fr

Nom : utilisateur_2_avance
Mot de passe : aaaaaaa;
Adresse : utili2@yahoo.fr



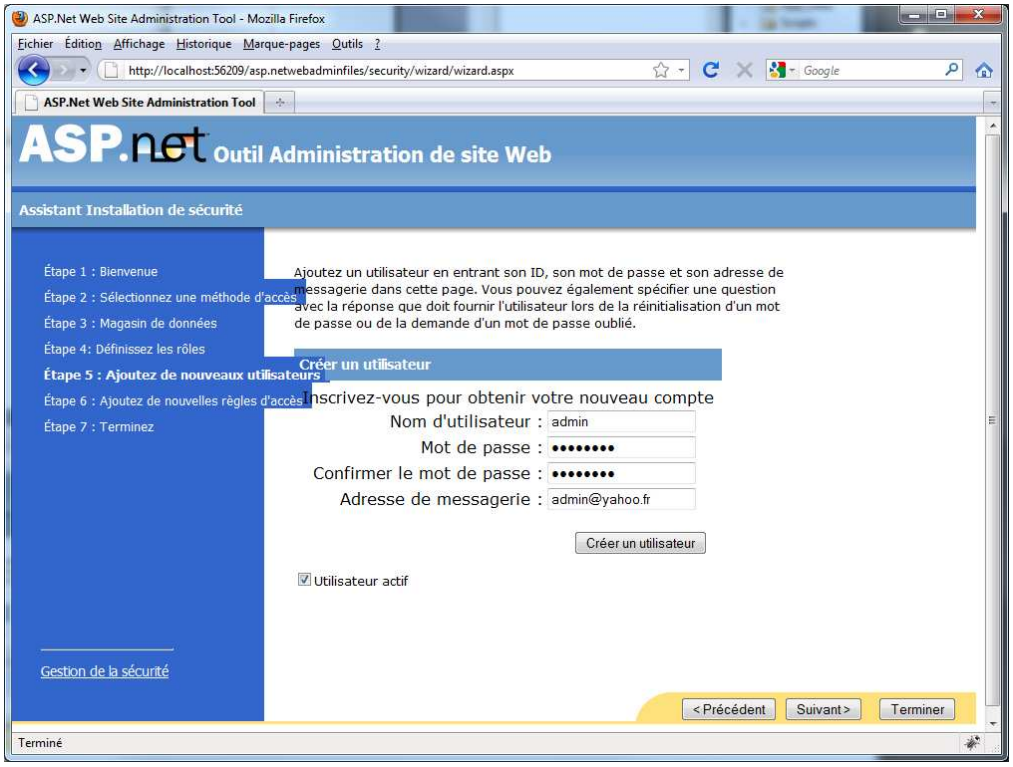


Recommencez l’opération pour l’utilisateur 2.

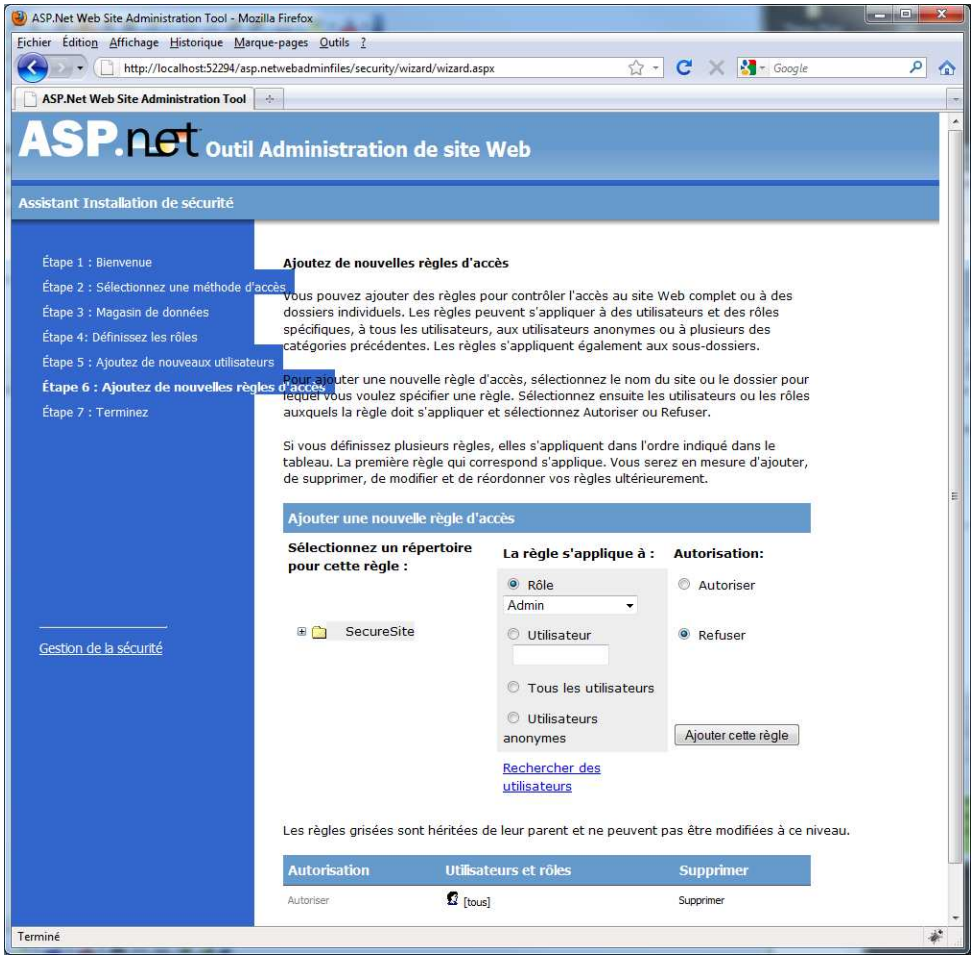


Créez ensuite un utilisateur admin

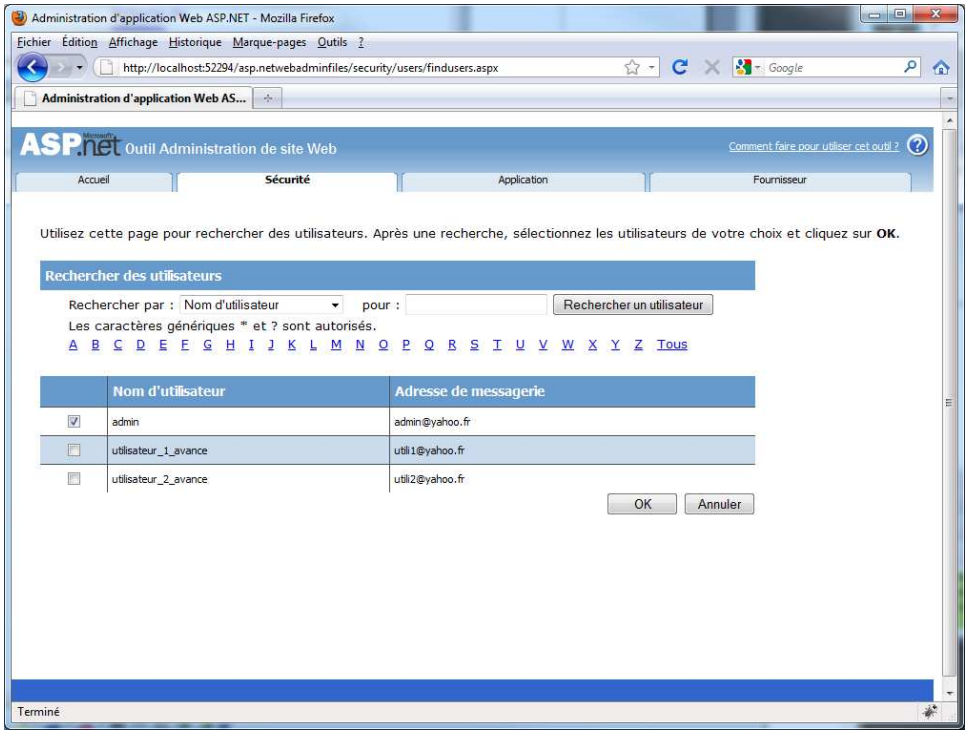
Nom : admin
Mot de passe : aaaaaaa ;
Adresse : admin@yahoo.fr



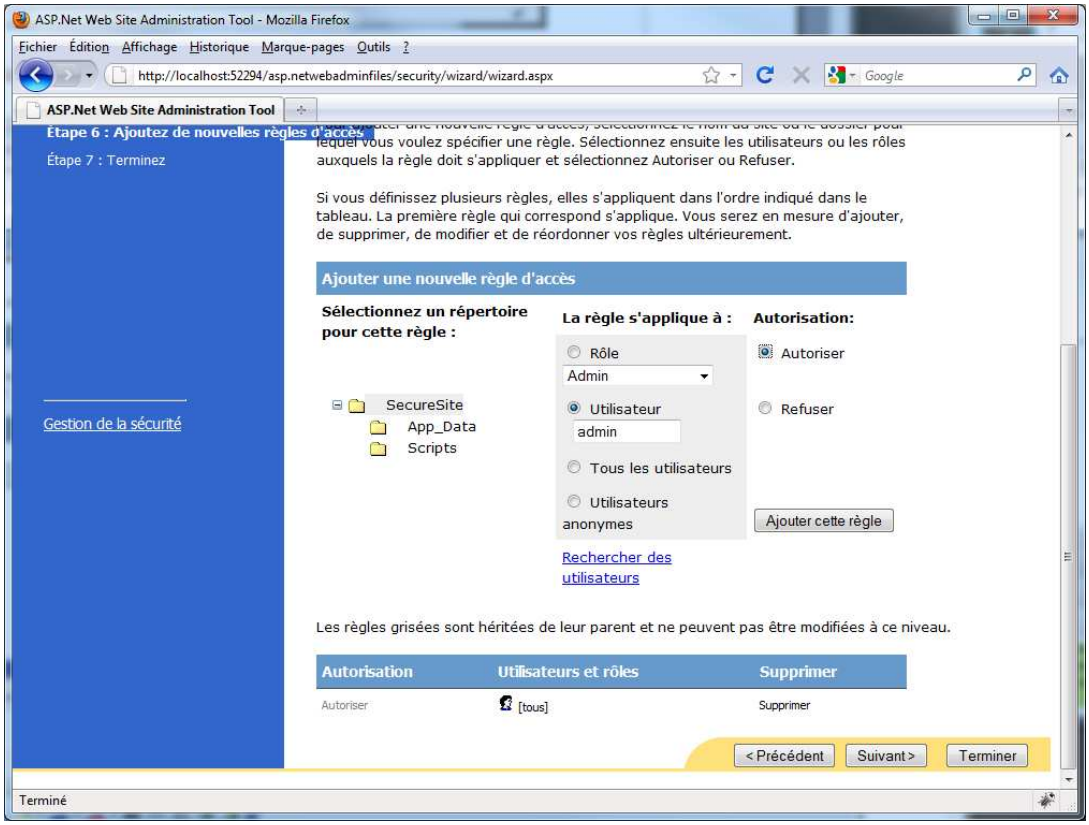
Passez à l'écran suivant qui va permettre de définir les rôles de chacun.

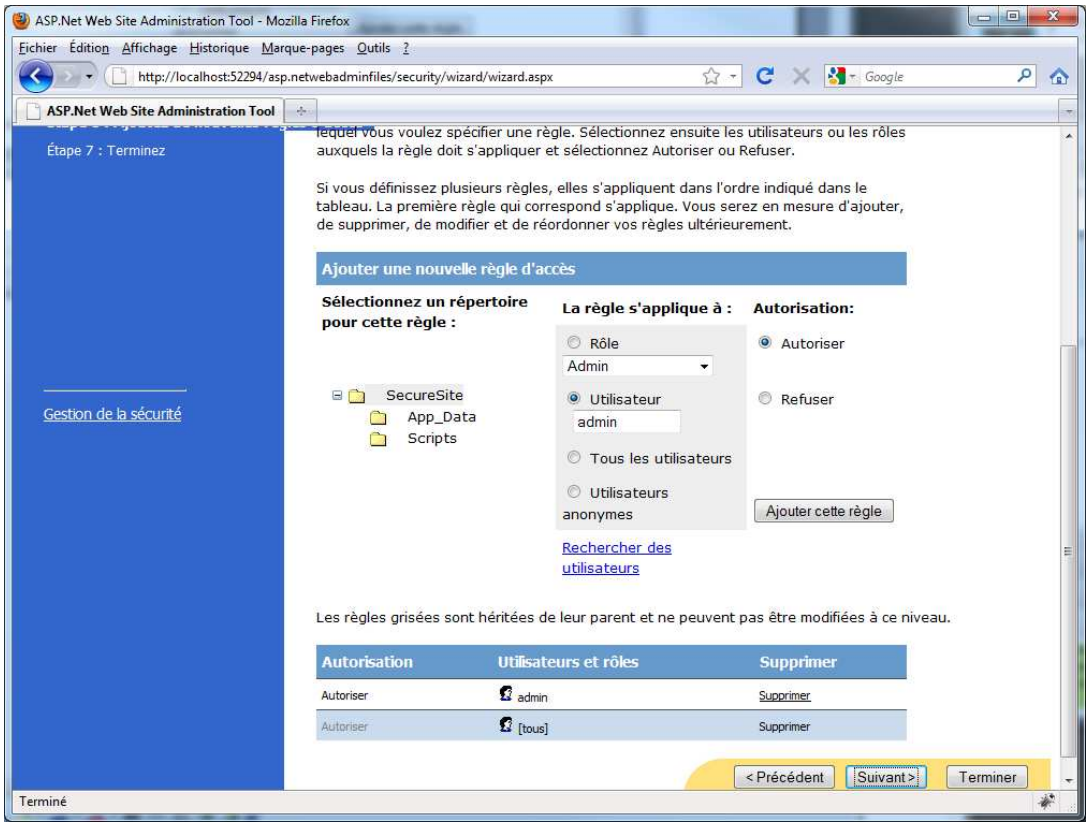


Pour le role **admin**, cliquer sur le lien **Rechercher des utilisateurs** et choisissez **admin**.

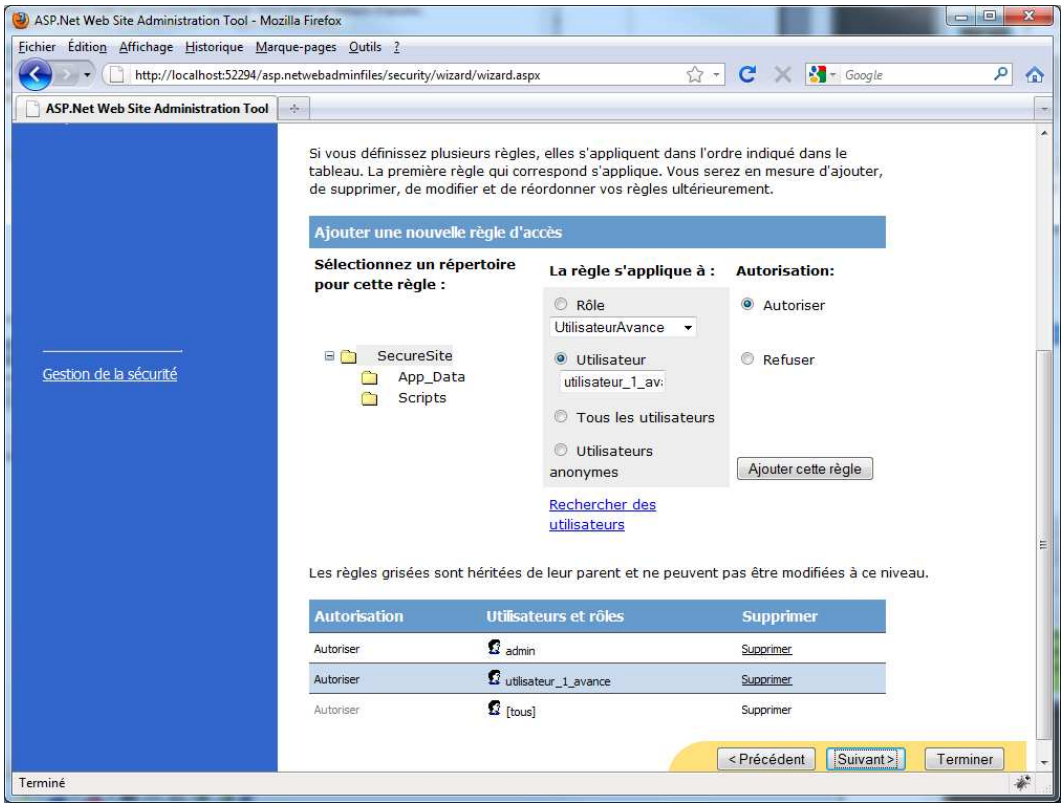


Cliquer ensuite sur « Ajouter la règle » pour créer la règle comme suit :

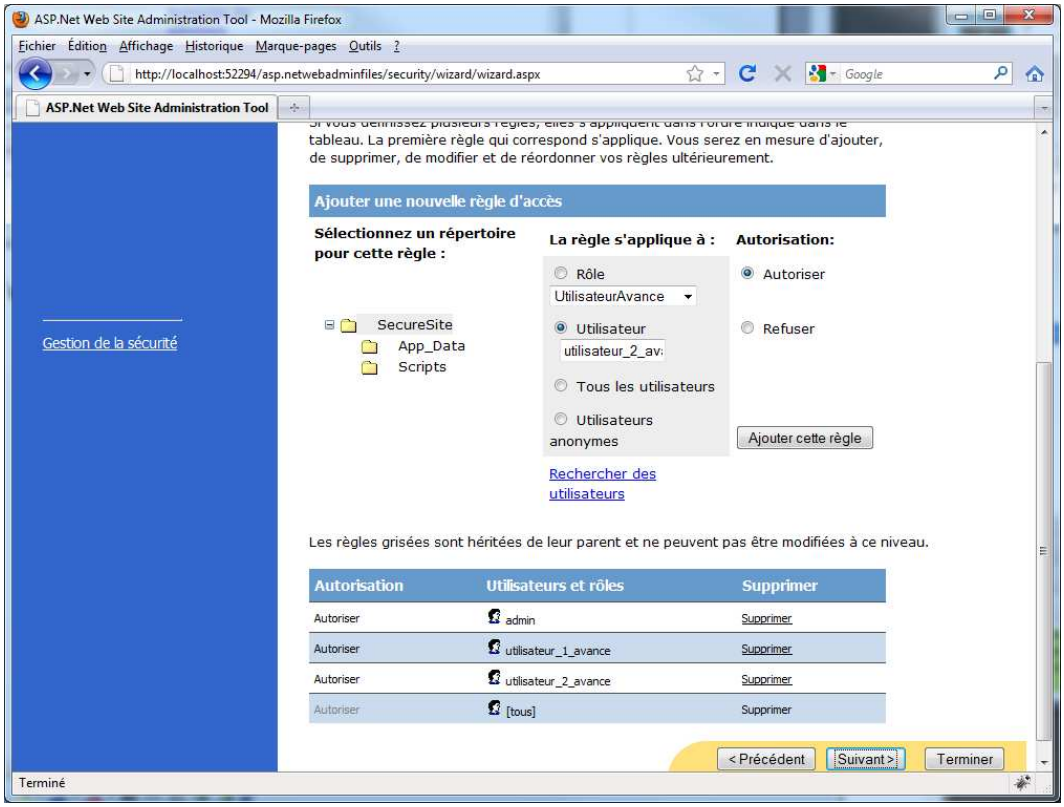




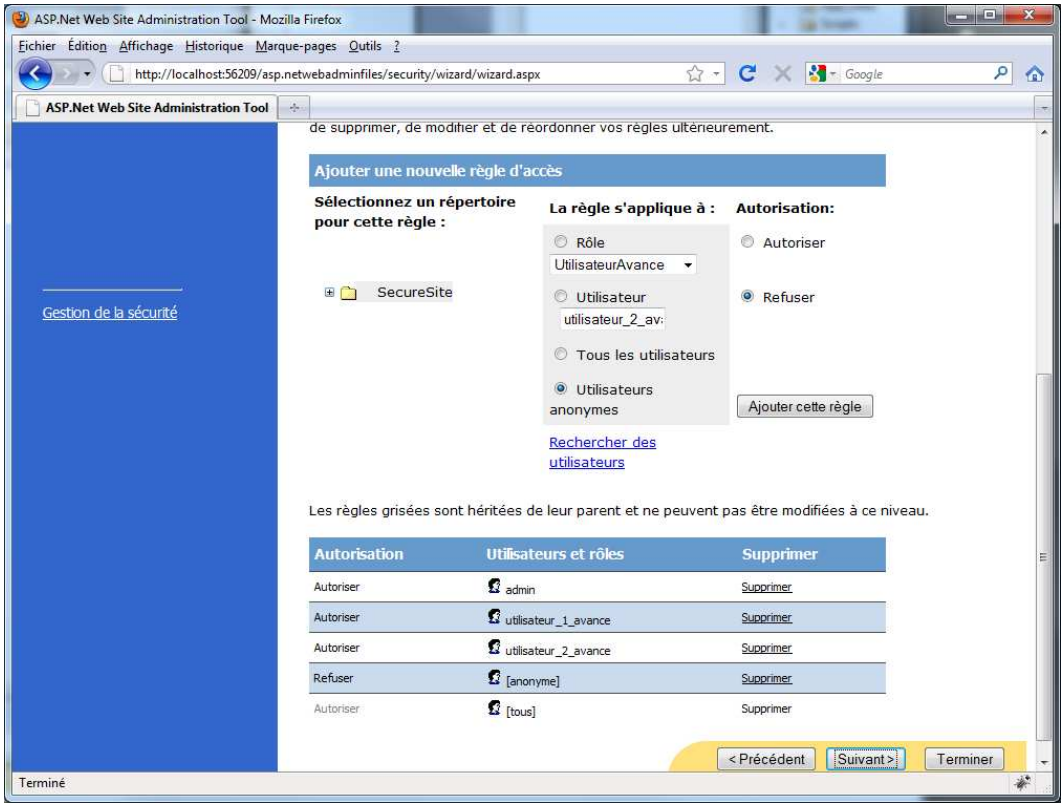
Ajouter ensuite une règle pour l'**utilisateur utilisateur_1_avance**

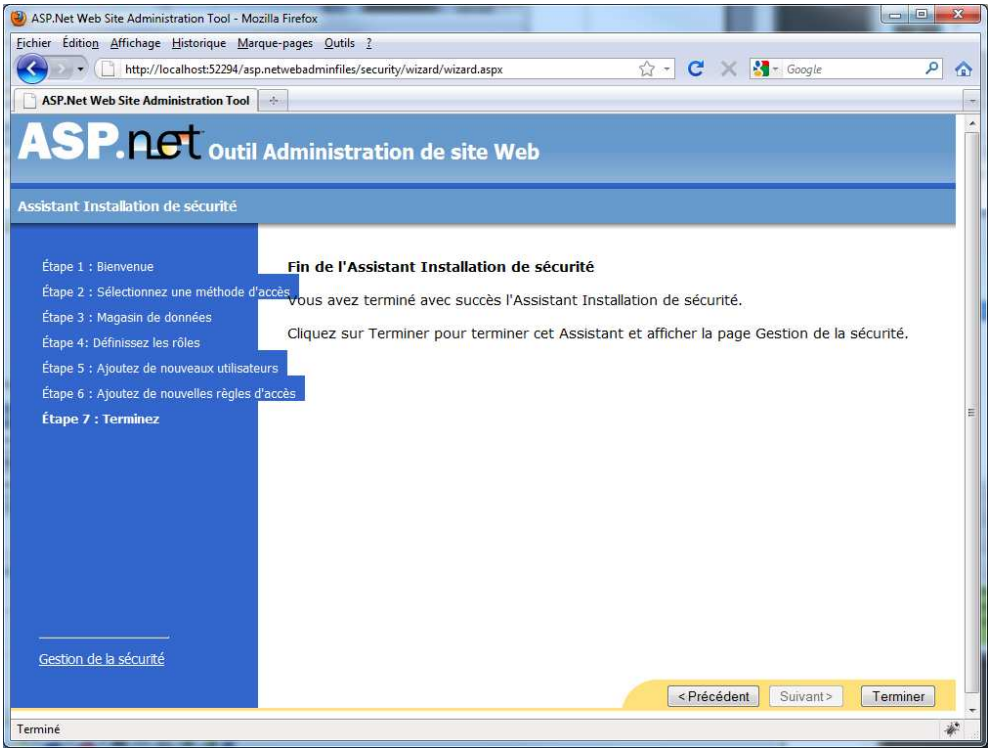


Ajouter ensuite une règle pour l'**utilisateur utilisateur_2_avance**

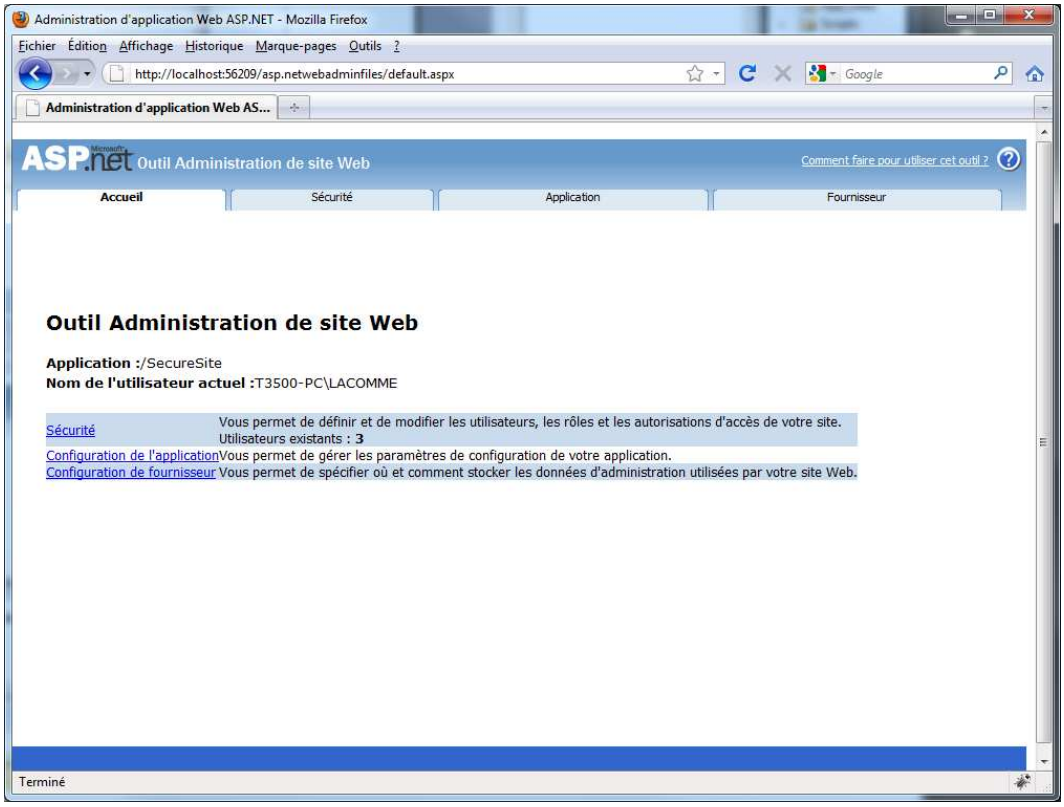


Refusez ensuite les utilisateurs anonymes.



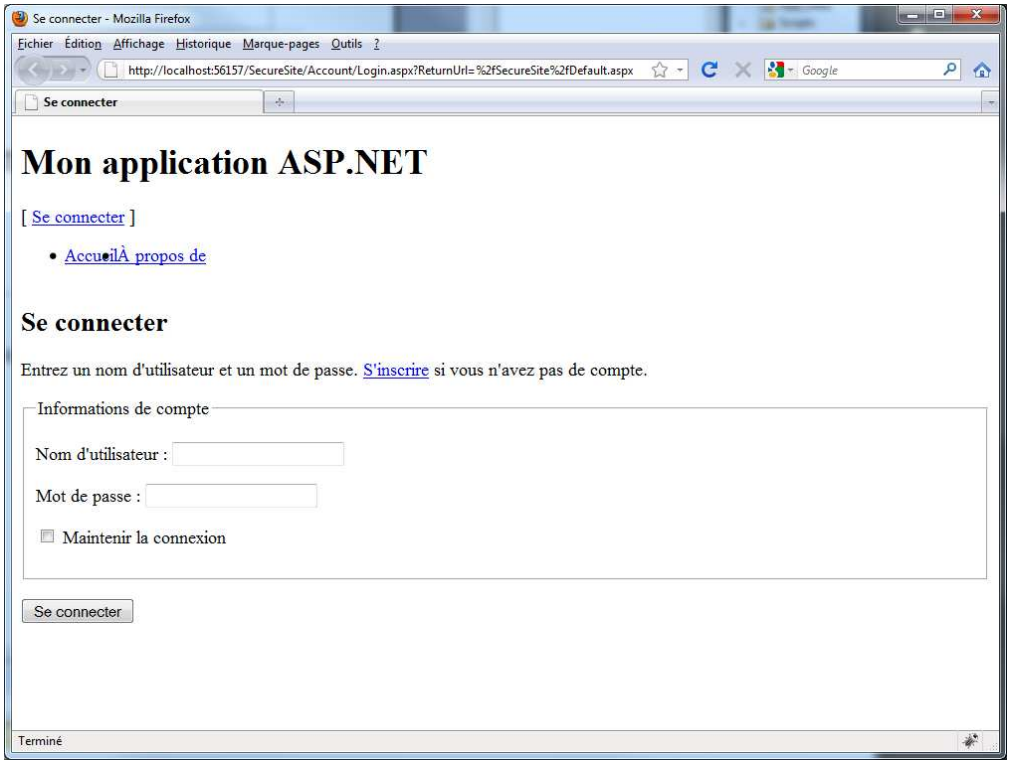


L'outil de configuration vous donne une vue d'ensemble.

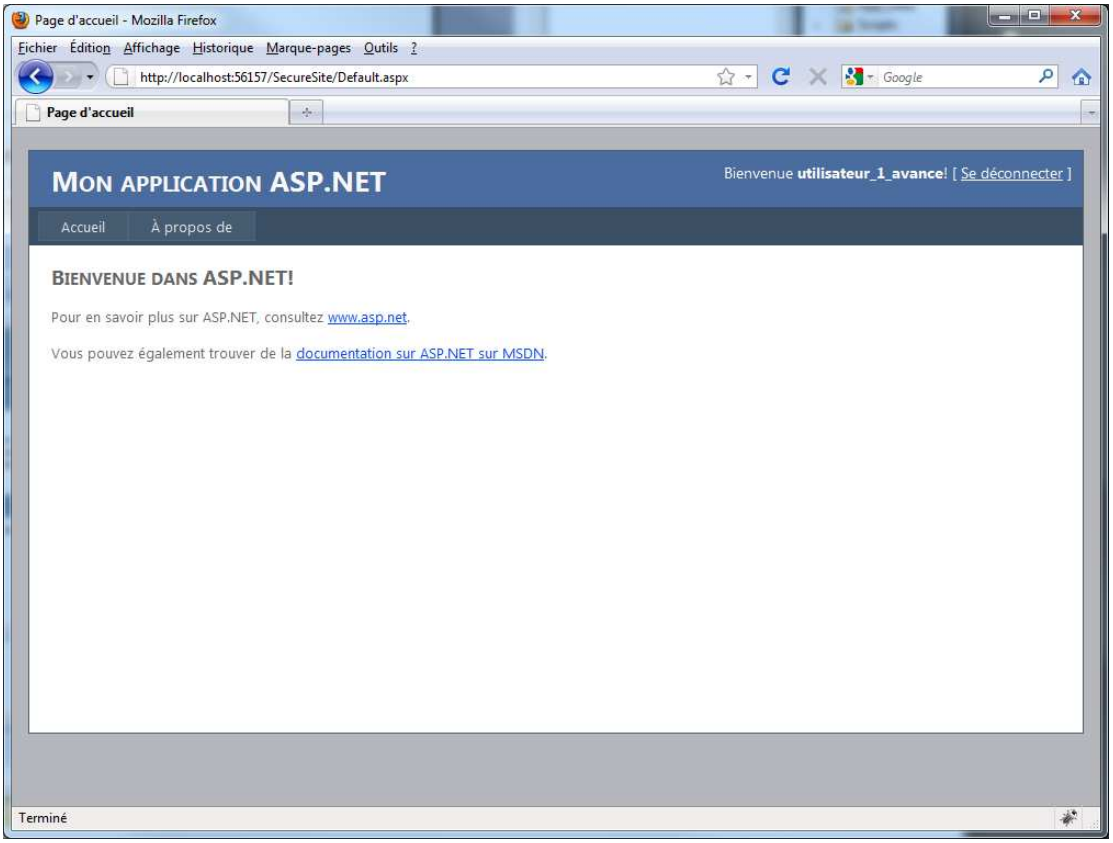


10.3. Utilisation de l'application

L'exécution de l'application entraine une redirection automatique vers la page de connexion.



Si vous entrez **utilisateur_1_avance** et le mot de passe **aaaaaaa;** vous obtiendrez ceci :



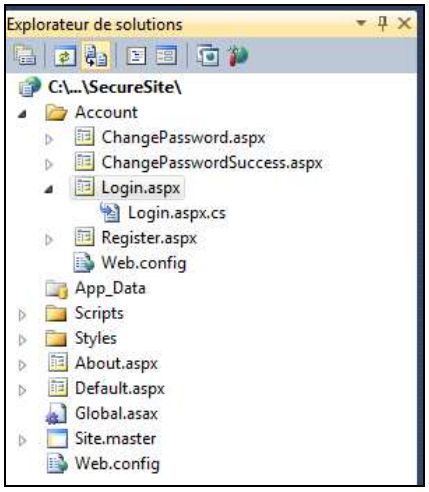
Remarquez le haut de la page :

Arrêter l'exécution du programme.

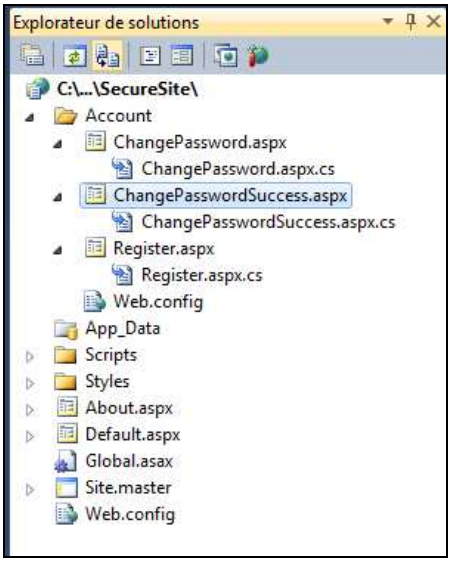
10.4. Modifier la page de connexion standard

La page de connexion par défaut à le mérite d'exister mais n'est pas très ergonomique. Nous allons la modifier.

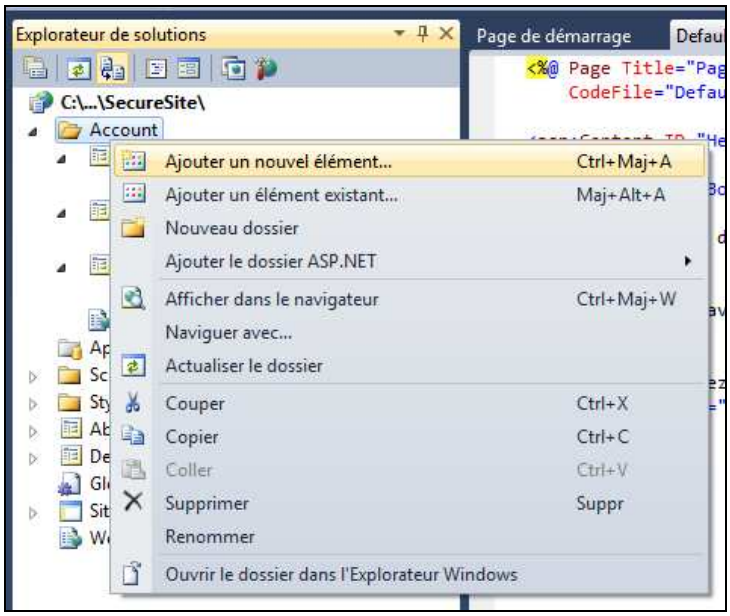
Allez dans l'**explorateur de solutions**.
Choisissez la partie **Account**.



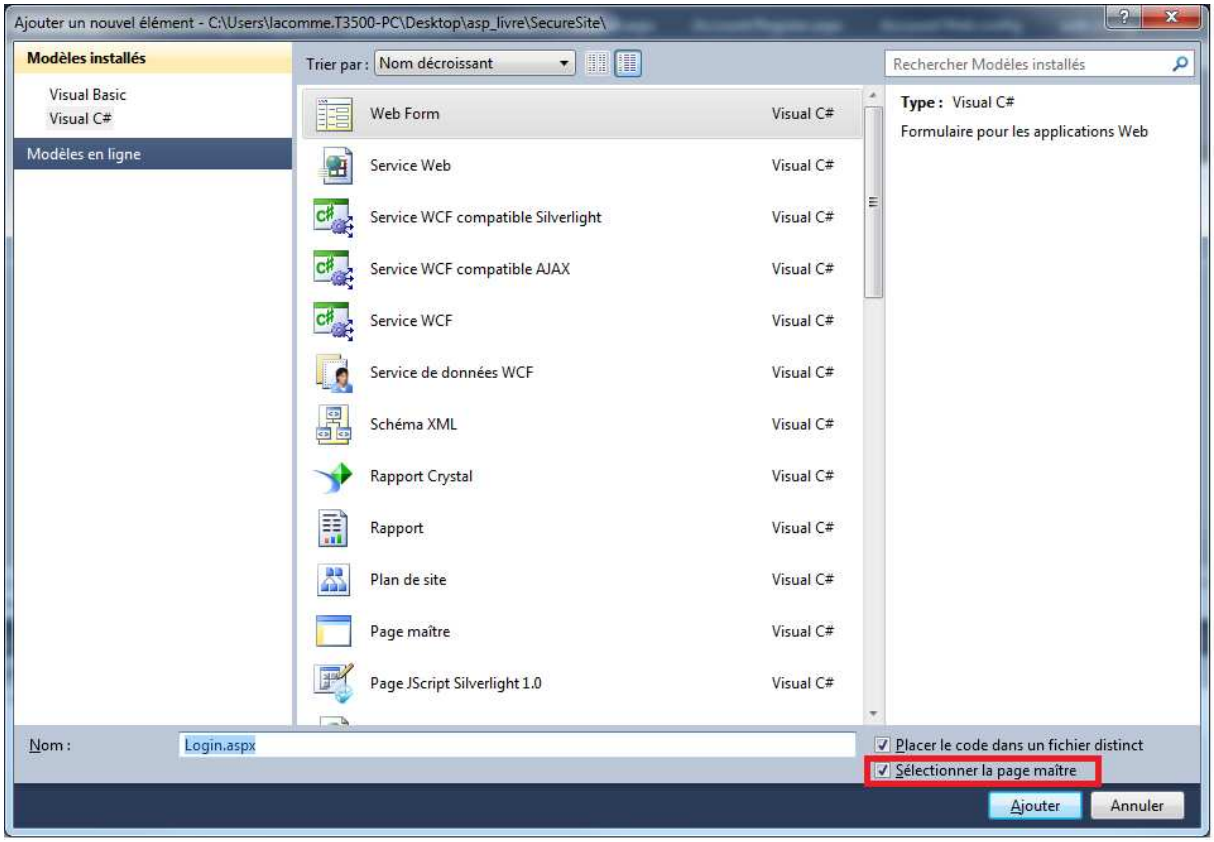
Supprimez la page **Login.aspx**.



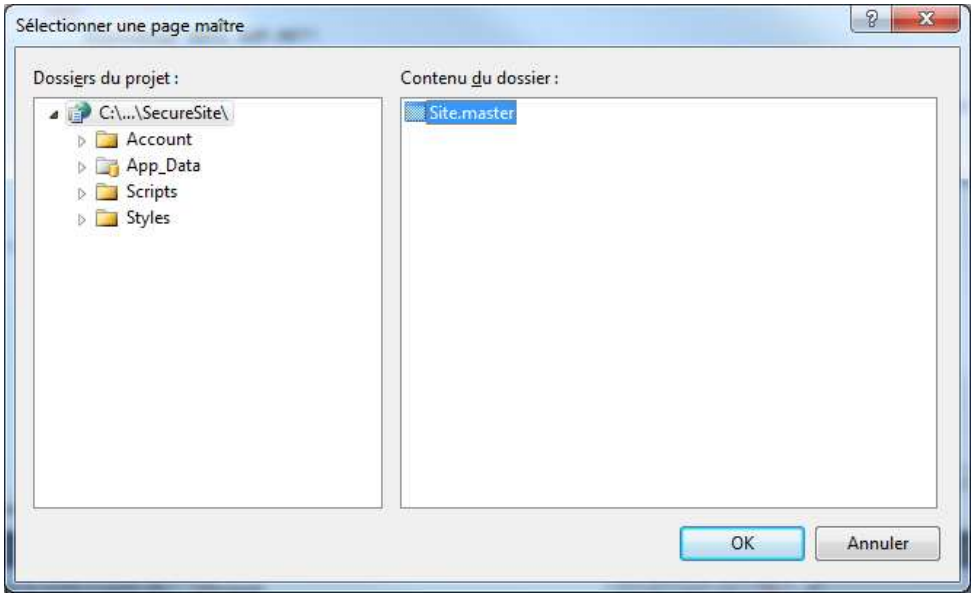
Faire un click droit sur **Account** et choisir **Ajouter un nouvel élément**.



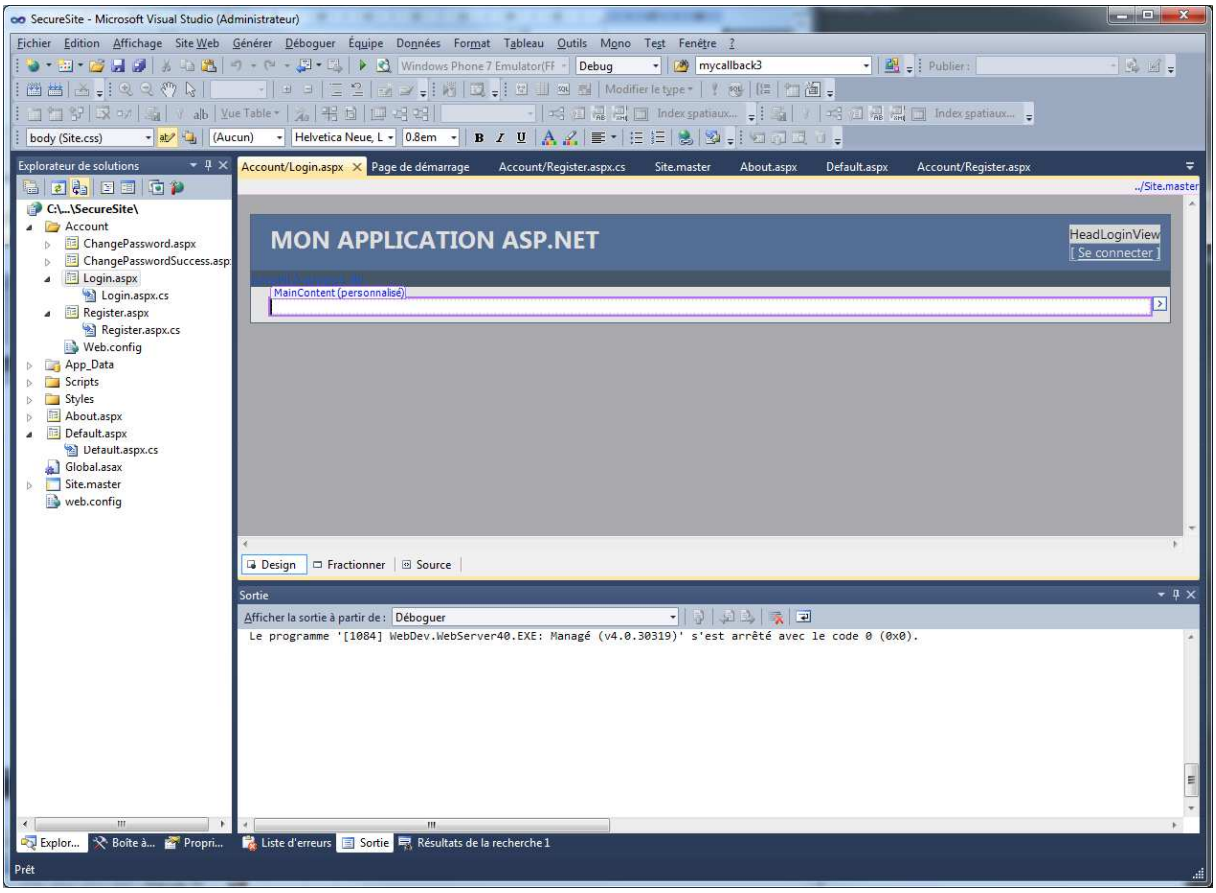
Choisir **Web Form** et comme nom **Login.aspx**.
Cochez la case **Sélectionner la page maitre**.



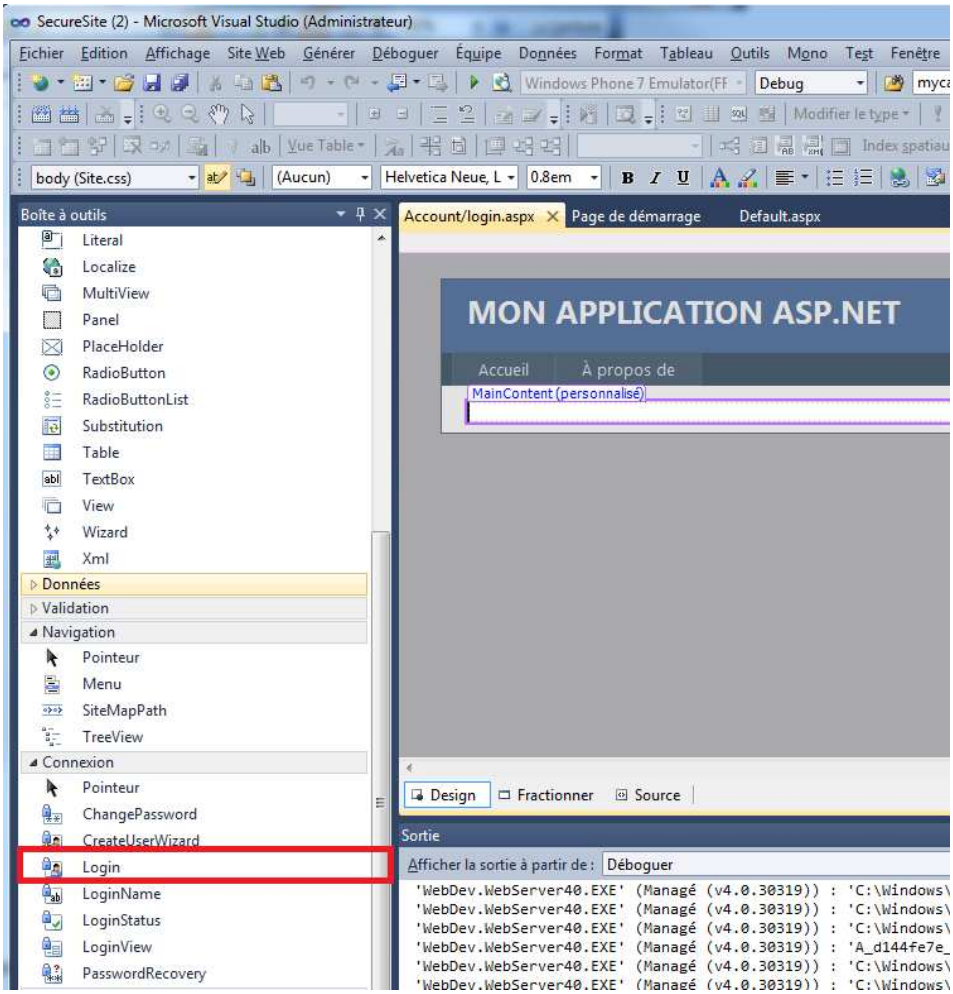
Choisissez la page maitre :



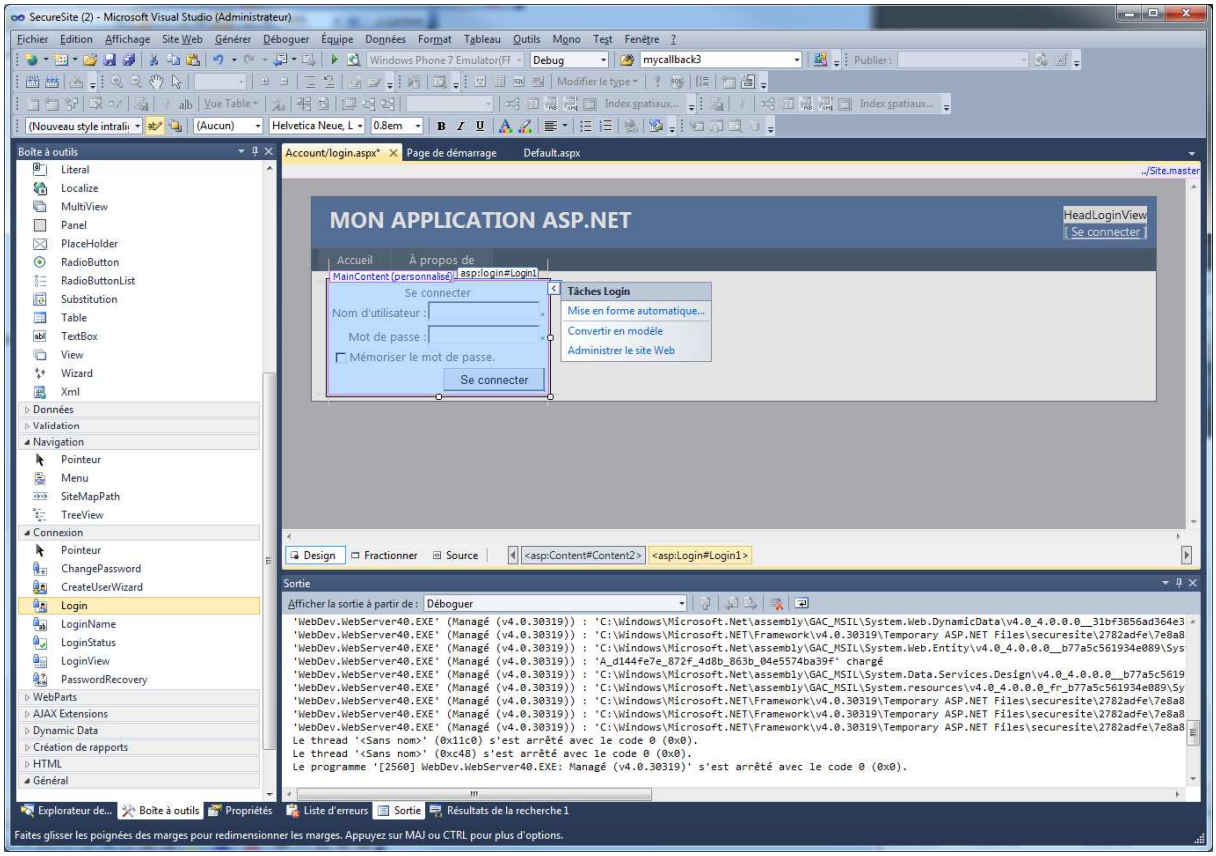
Ouvrir la page **Login.aspx**.
Passez en mode **Design**.



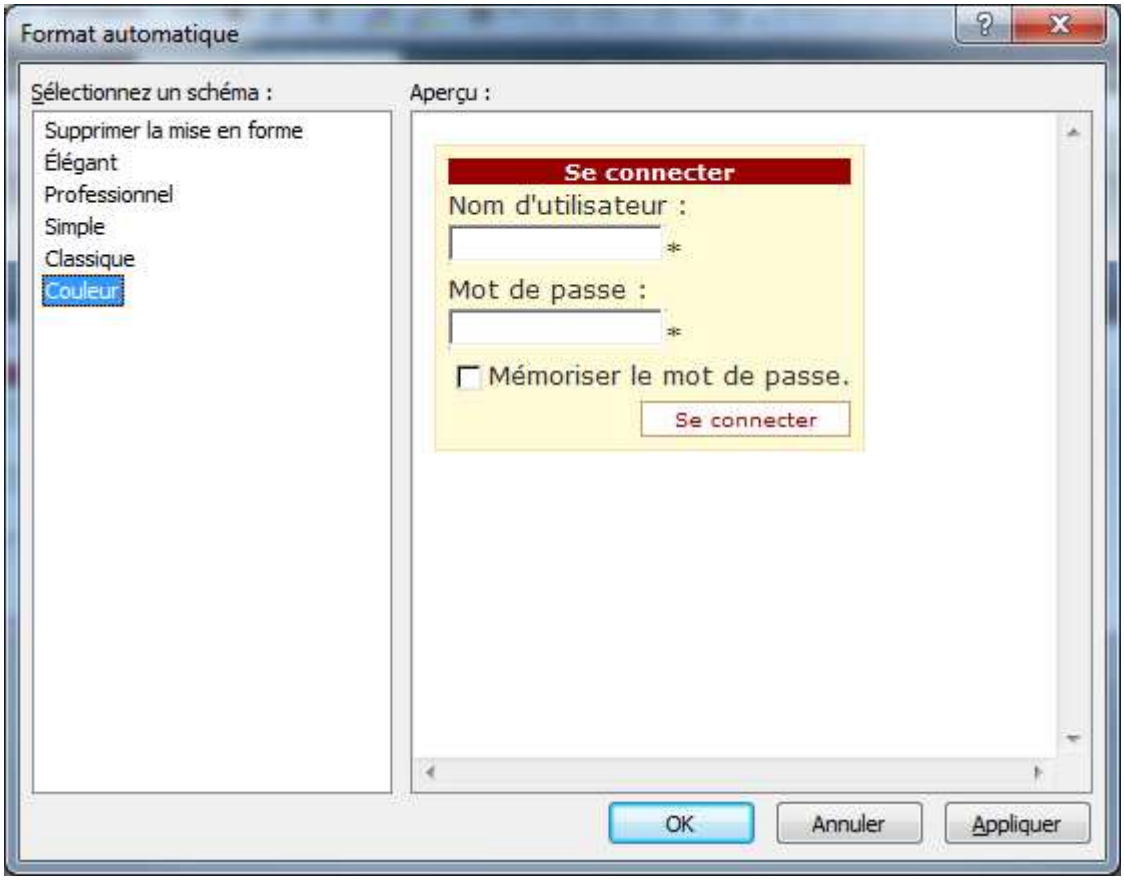
Allez dans la boite à outils et chercher la section **Connexion**.
Sélectionner le contrôle **Login**.



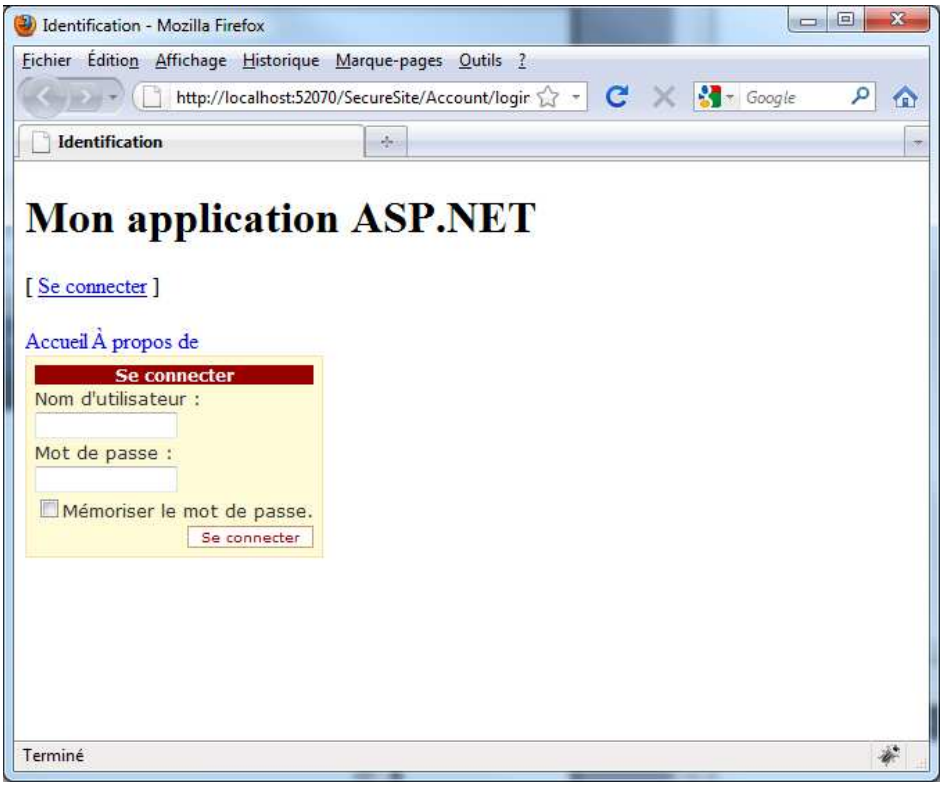
Posez le contrôle sur la page.



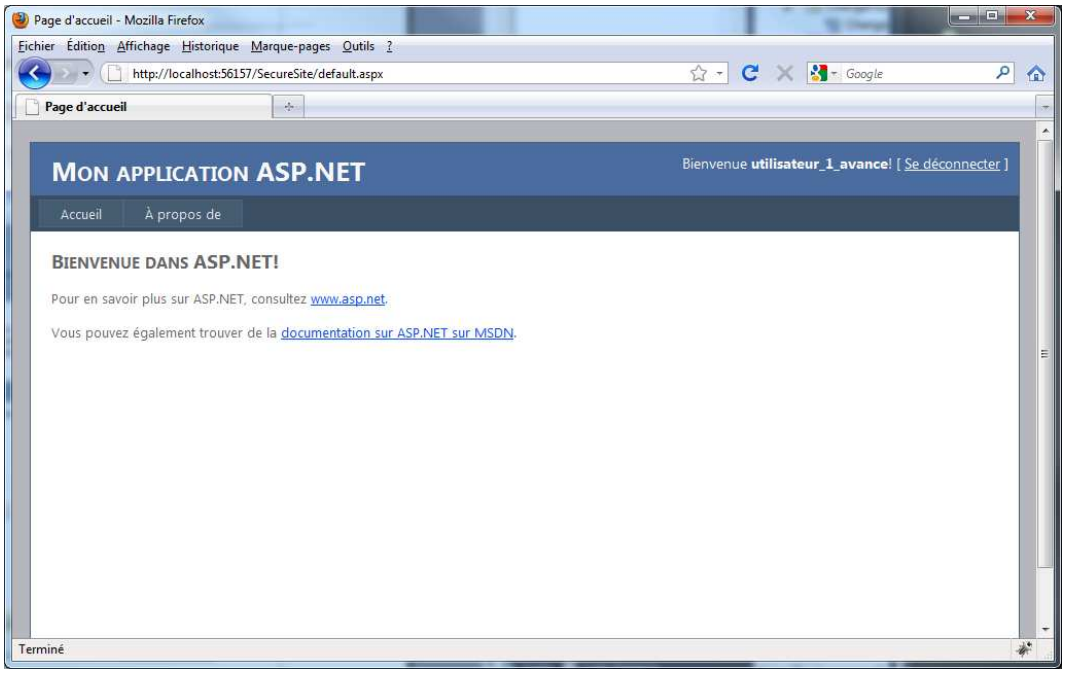
Choisissez dans Mise en forme automatique la présentation qui vous convient. Pas exemple celle-ci :



L’exécution donne maintenant :



Si vous entrez le mot de passe correct vous obtiendrez ceci :

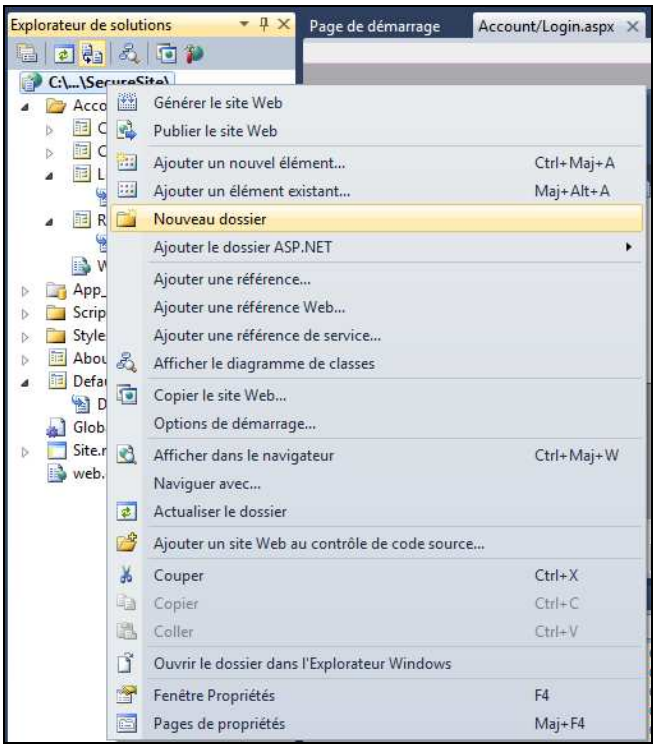


10.5. Autorisation des utilisateurs

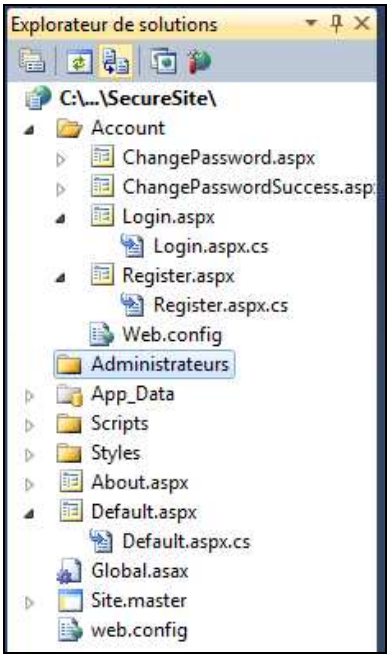
Maintenant que l'authentification est faite il faut définir les droits des utilisateurs connectés sur le site.

Etape 1. Créer un répertoire par type d'utilisateurs.

Créez un répertoire en cliquant sur le nœud du **projet / Nouveau dossier**.

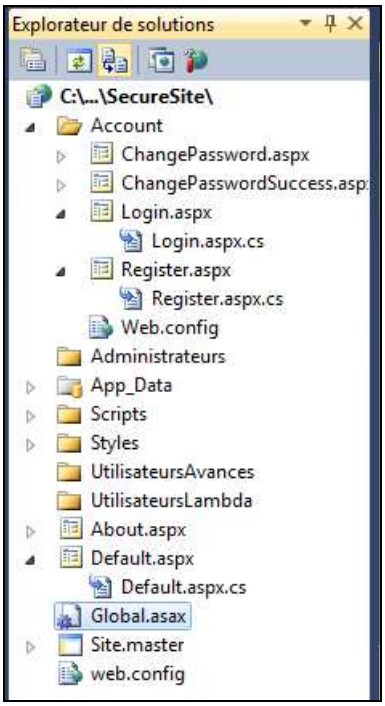


Nommez ce répertoire **Administrateurs**.



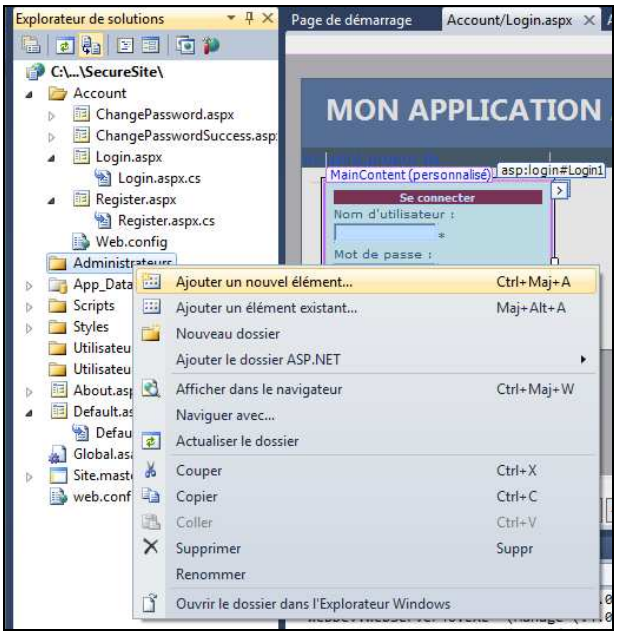
Créer ensuite deux répertoires nommés :
- **UtilisateursLambda**
- **UtilisateursAvances**

Le projet doit ressembler à ceci :

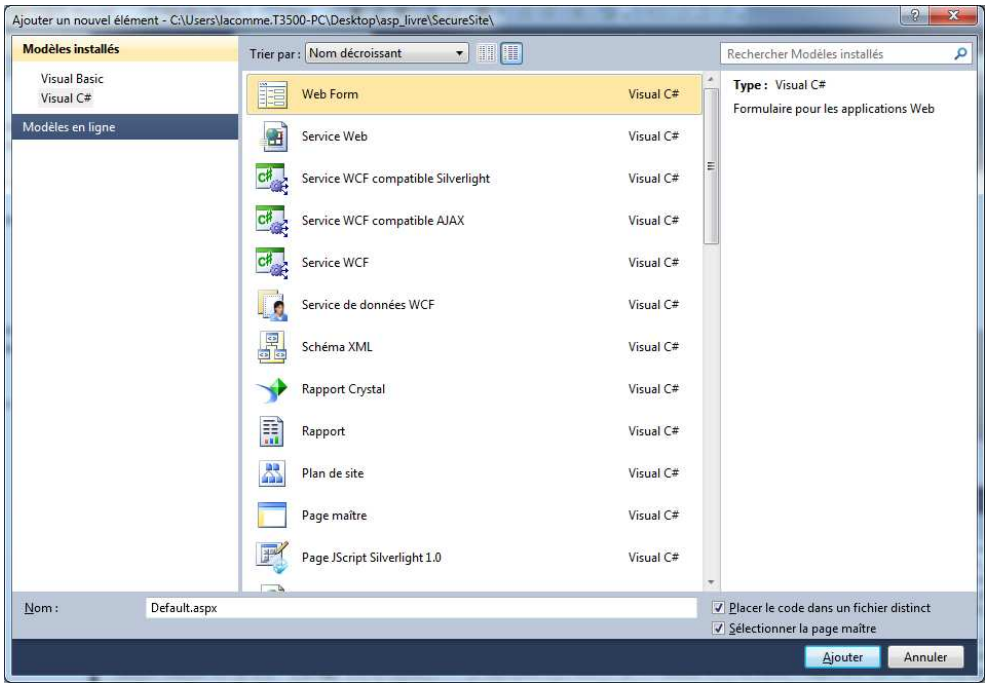


Etape 2. Définir une page pour les administrateurs et les utilisateurs

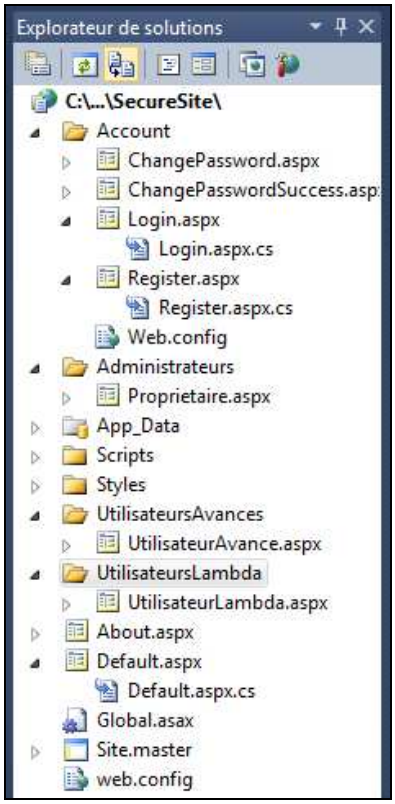
Faire un click droit sur **Administrateurs** et choisir **Ajouter un nouvel élément**.



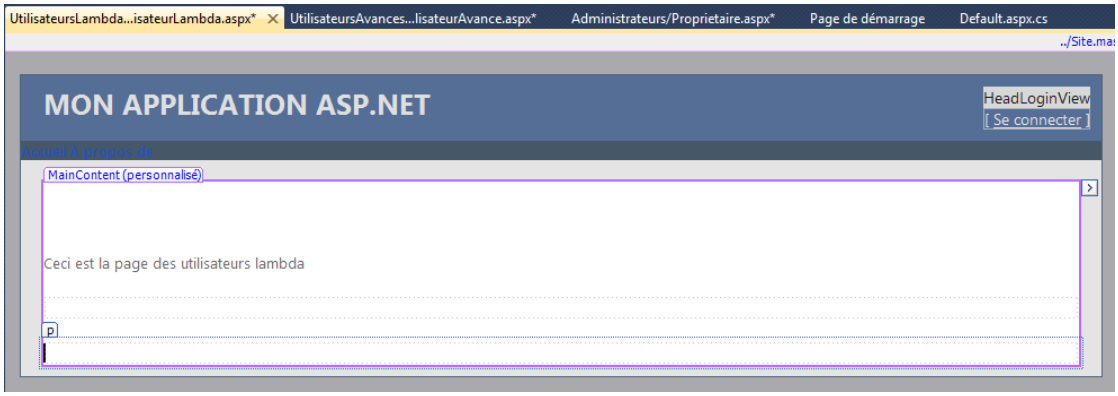
Choisir une Web Form basée sur une page maître.
Nommez cette page **Proprietaire.aspx**.



Réitérez l'opération avec les utilisateurs lambda et les utilisateurs avancés.

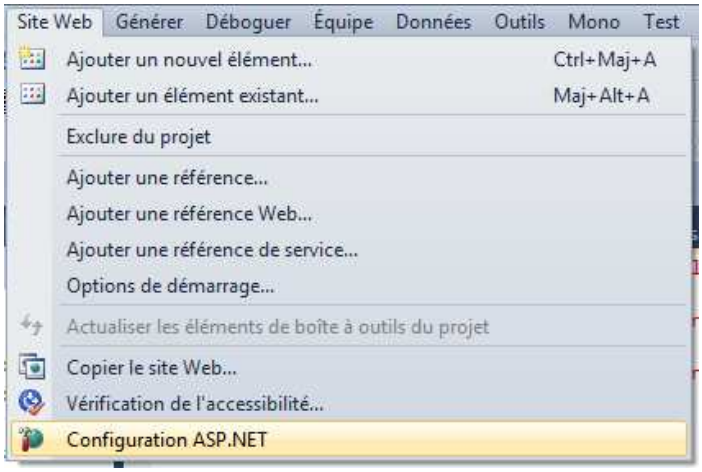


Mettre sur chacune de ces pages un mot d'accueil du type :

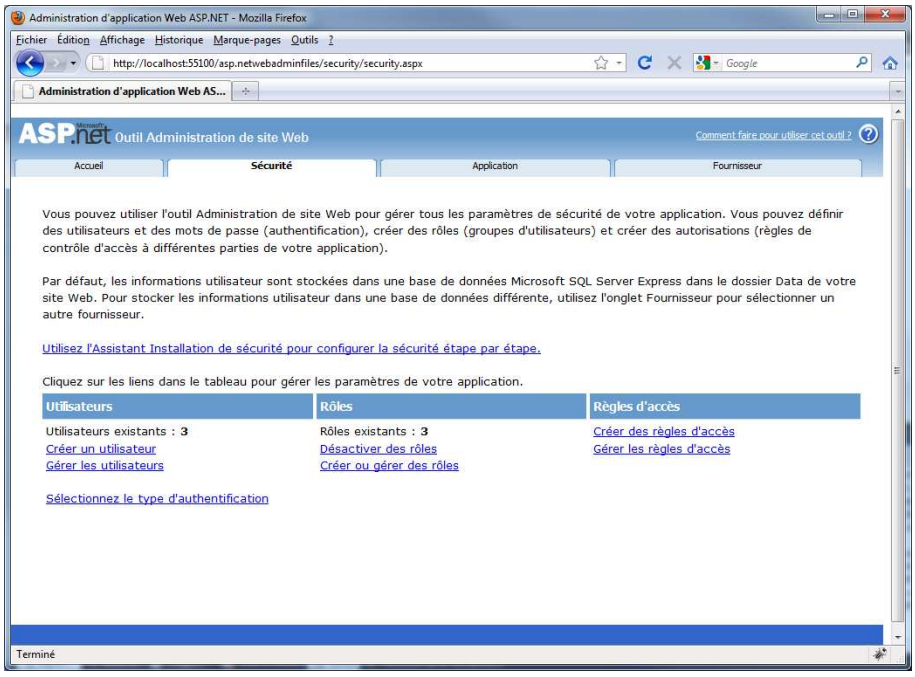


Etape 3. Définition des règles d'accès

Lancer la configuration ASP. NET.

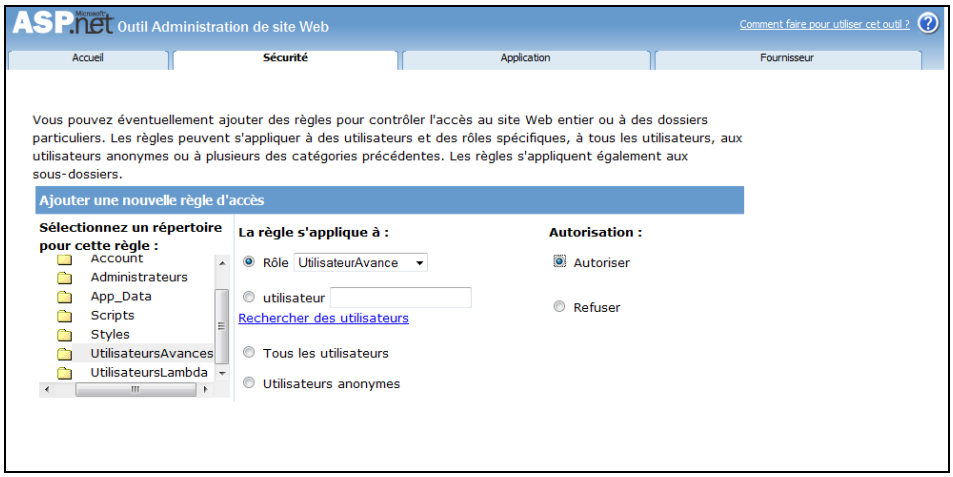


Allez dans l'onglet Sécurité.



Allez dans le lien **Gérer les règles d'accès**.

Créer une **nouvelle règle** pour le répertoire « UtilisateurAvances ».



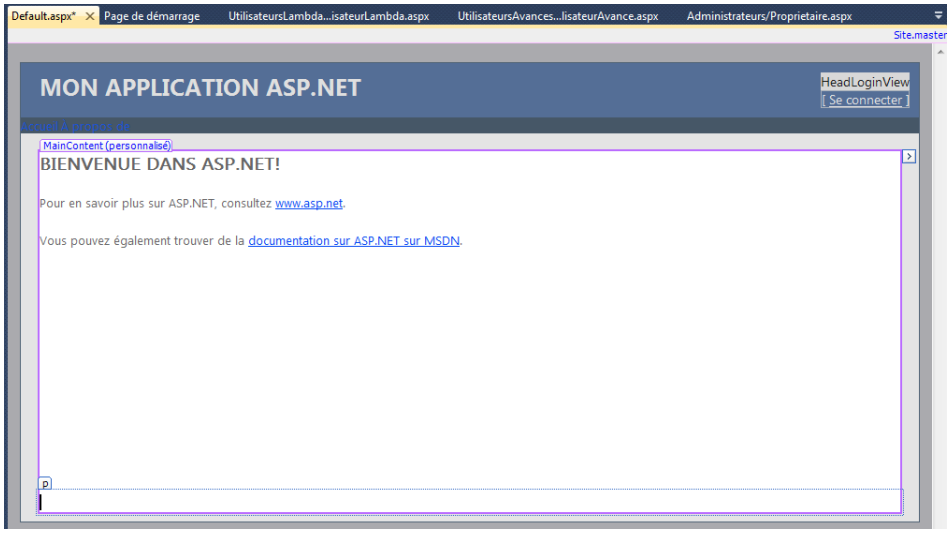
Recommencez pour l'utilisateur Lambda.

Recommencez pour l'utilisateur Administrateur

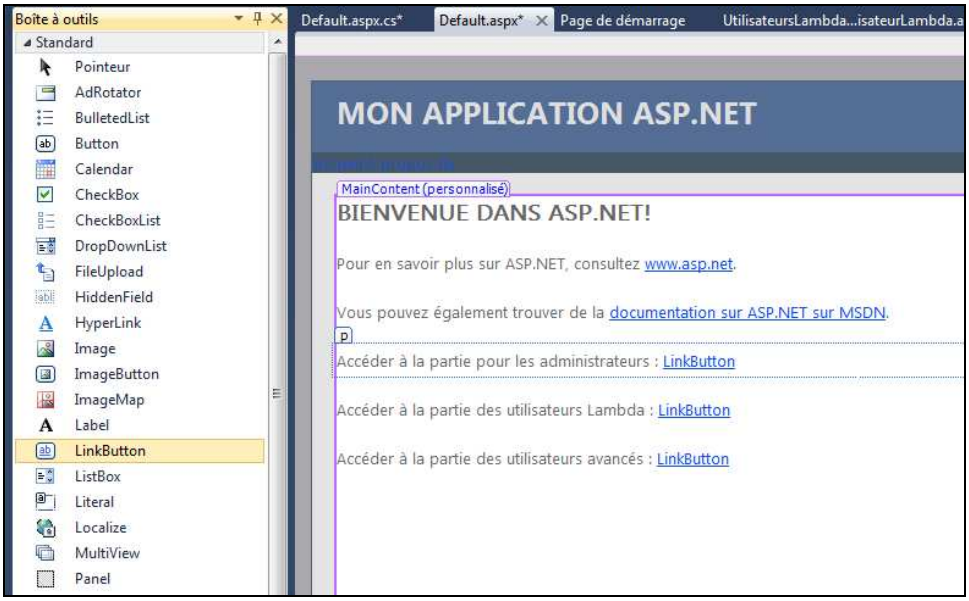
Ajouter une règle interdisant à un utilisateur non administrateur d'accéder à la partie Adminsitrateurs. Recommencez pour la partir utilisateurs avancés et utilisateurs lambda.

Etape 4. Test des règles d'accès

Ouvrir la page **Default.aspx**.



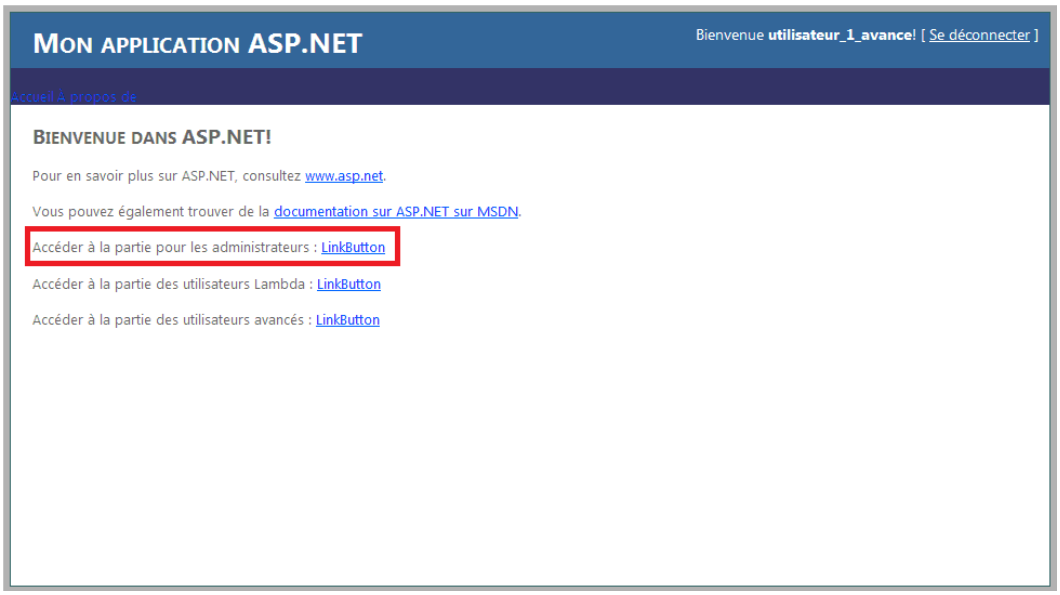
Utilisez un link button pour créer trois liens donnant accès aux pages aspx de chaque catégorie.



Lancer l'application.

Connectez-vous en tant que utilisateur avancé.

L'application doit ressembler à ce qui suit :



Si on cherche à accéder à la partie réservée aux administrateurs, on obtient alors un écran nous demandant de nous identifier à nouveau comme ceci :