Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

Ajax Control Toolkit

Version 1.0

GERAUD Cédric
BEDE Nicolas
BASCANS Jérémy
HEROGUEL Quentin

James RAVAILLE
http://blogs.dotnet-france.com/jamesr

Sommaire

1 Introduction

1.1 Présentation d'AJAX

AJAX signifie **A**synchronous **Ja**vaScript and **X**ML. Sur un plan macroscopique, AJAX est une technologie étendant la technologie ASP .NET, dont Microsoft n'est pas propriétaire. Comme beaucoup d'éditeurs de logiciels, Microsoft en propose une implémentation, connue sous le nom de Microsoft ASP .NET Ajax, pour développer des applications Web :

- Plus riches sur le plan graphique.
- Plus interactives avec les utilisateurs.

L'implémentation proposée par Microsoft, se décompose en deux parties distinctes :

- Microsoft Ajax Library : constitué d'un Framework Ajax côté serveur, ainsi que d'un Framework Ajax côté client (ensemble de fichiers JavaScript). Cette partie est incluse dans le Framework .NET 3.5.
- Microsoft Ajax Toolkit : ensemble de contrôles permettant d'améliorer l'ergonomie et les performances des applications Web.

AJAX n'est pas un langage de programmation. Son principe de base est de permettre d'exécuter, depuis du code JavaScript s'exécutant dans un navigateur Web, des requêtes HTTP de manière asynchrone. Alors que dans une application ASP .NET traditionnelle, les échanges entre les clients et le serveur sont effectués de manière synchrone.

Microsoft Ajax Library permet d'exécuter des actions côté serveur, sans avoir à, recharger intégralement la page, grâce à l'interaction entre JavaScript côté client et le serveur au travers de Handlers HTTP.

Pour de plus amples informations sur les bases fondamentales de Microsoft, nous vous invitons à lire le cours suivant : http://www.dotnet-france.com/Documents/Web/ASP_NET35/Les%20bases%20fondamentales%20de%20Microsoft%20ASP.NET%20Ajax.pdf

1.2 Présentation d'AJAX Control Toolkit (ACT)

AJAX Control Toolkit (nommé ci-dessous ACT) est une extension de la plateforme ASP .NET, permettant d'apporter à cette dernière, de nouvelles fonctionnalités AJAX, au travers de contrôles ASP .NET. Il a été conçu de manière à faire partie intégrante à ASP .NET, et par conséquent de s'intégrer intégralement dans les applications ASP .NET 3.5 existantes. C'est aussi un projet Open Source, qui a été conçu et développé par des experts .NET Ajax indépendants, des personnes travaillant chez Microsoft, …

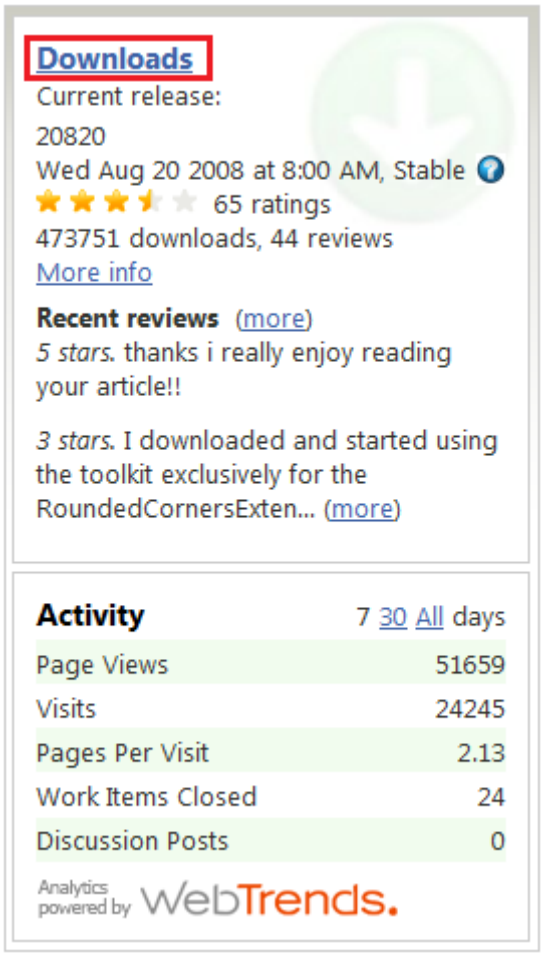
Pour pouvoir utiliser les contrôles de l'AJAX Control Toolkit dans une page ASP .NET, il est nécessaire de posséder lors de l'exécution de la page du contrôle ScriptManager. En effet c'est un contrôle essentiel qui assure le bon fonctionnement des fonctionnalités AJAX dans votre application ASP .NET.

Les contrôles ACT se distinguent en deux catégories :

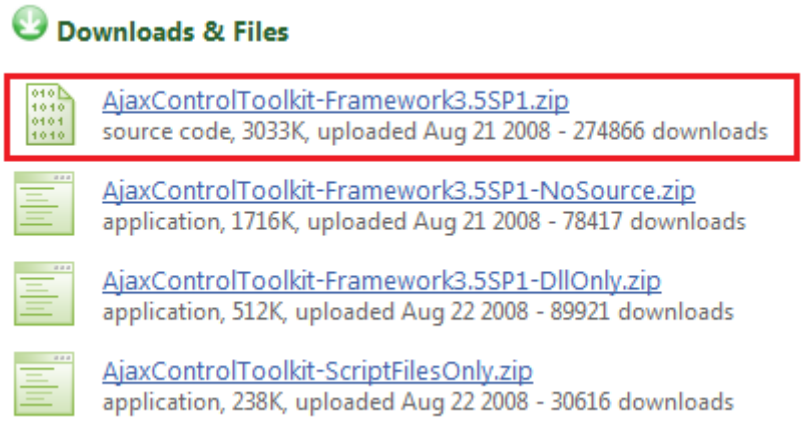
- Les extenders : contrôles permettant d'étendre les fonctionnalités d'autres contrôles ASP .NET présents dans la page ASP .NET.
- Les contrôles ASP .NET Ajax « à part entière » : contrôles autonomes (ex : le tableau d'onglets) permettant de proposer des fonctionnalités Ajax, sans étendre les contrôles existants dans la page ASP .NET.

1.3 Installation des ACT

Cette partie va vous permettre de pouvoir implémenter puis utiliser les contrôles Ajax Toolkit dans votre projet ASP.NET. Il va tout d'abord falloir les télécharger; pour cela vous devrez vous rendre sur le site www.codeplex.com. Ensuite vous devriez voir dans les plus téléchargés une partie « AJAX Control Toolkit », sinon dans la barre de recherche tapez « Ajax toolkit ». Vous aurez alors la partie appropriée en haut de la liste de recherche. Cliquez dessus. Vous arrivez donc sur la partie Ajax Control Toolkit où vous pouvez apercevoir à droite une partie downloads sur laquelle vous cliquerez :

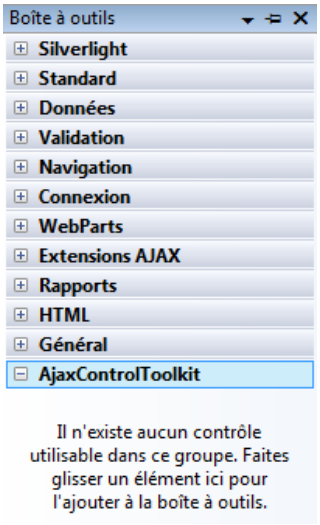


Une fois la page chargée, récupérez le fichier « AjaxControlToolkit-Framework3.5SP1.zip » :

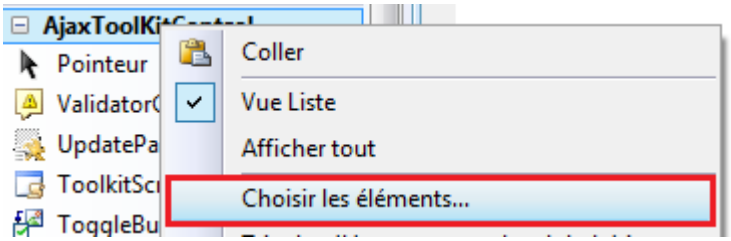


Ensuite, il suffit d'en extraire le contenu.

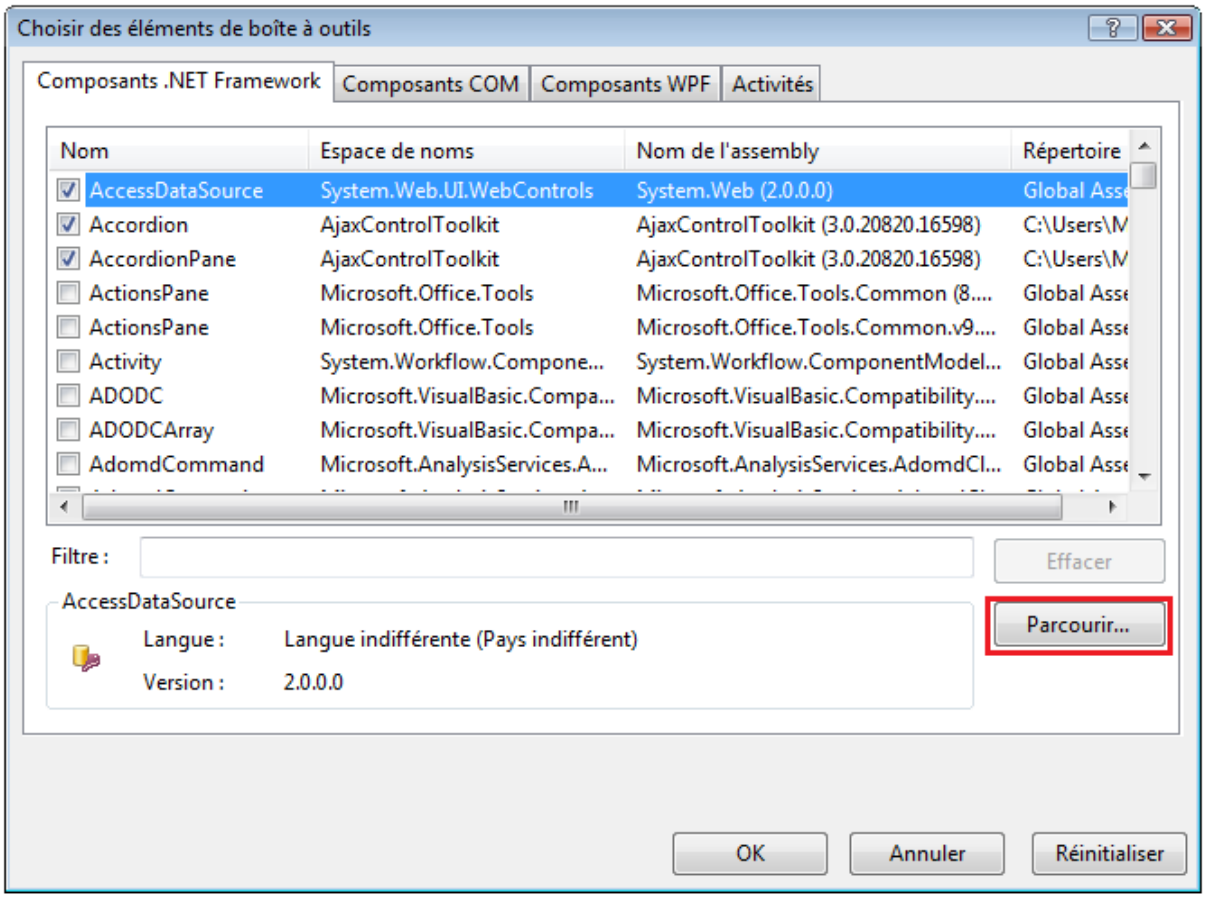
Une fois tout cela fait, vous pouvez ouvrir Visual Studio et créer un projet ASP.NET. Ouvrez ensuite la boite à outils, faites click droit dedans et ajouter un onglet que vous pouvez appeler par exemple « AjaxControlToolkit ». Vous obtiendrez donc ceci :



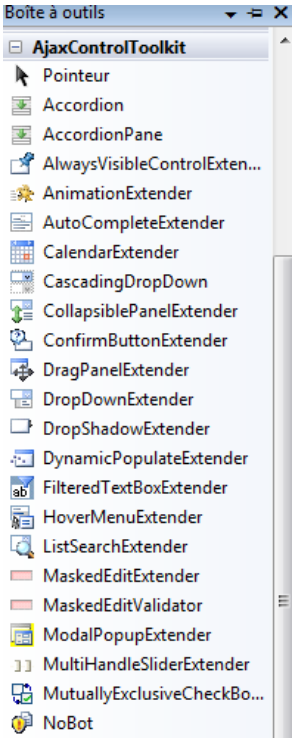
Faites ensuite click droit sur votre onglet et « Choisir les éléments » :



Une fenêtre va s'ouvrir. Vous pourrez à partir de la récupérer une dll qui se trouve dans le dossier appelé « SampleWebSite », dans lequel se trouve un répertoire appelé « Bin » qui contient une dll appelée « AjaxControlToolkit.dll ». C'est cela qui va permettre d'ajouter les contrôles à votre projet :



Une fois la dll ajoutée, tous les contrôles seront automatiquement ajoutés dans votre boîte à outils :



Les contrôles AJAX Toolkit sont désormais disponibles et utilisable dans votre projet ASP.NET.

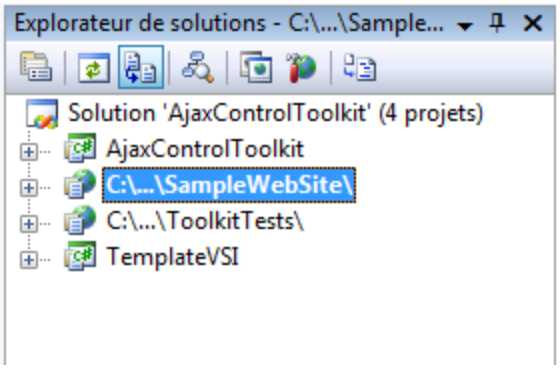
Remarque : Vous pouvez aussi télécharger le fichier « AjaxControlToolkit-Framework3.5SP1-DllOnly.zip ». Ensuite dé zipper le, dedans se trouve un répertoire appelé « Bin » qui contient une dll appelée « AjaxControlToolkit.dll ». Il suffit de prendre cette dll et de la glisser en drag and drop dans votre onglet « AjaxControlToolkit ». Les contrôles vont s'ajouter automatiquement.

2 Découvrez les contrôles ACT

Dans un premier temps vous devez télécharger le **pack SP1** de L'ACT (Ajax Control Toolkit) via le lien suivant AjaxControlToolkit-Framework3.5SP1.zip

Une fois le fichier télécharger, décompressez-le et ouvrez-le. Double-cliquez sur la solution *AjaxControlToolkit.sln*. Un avertissement de sécurité peut s'afficher au cours de l'ouverture de la solution, vous devez donc sélectionnez l'option appropriée (qui décrit le fait que vous faites confiance au projet que vous allez ouvrir).

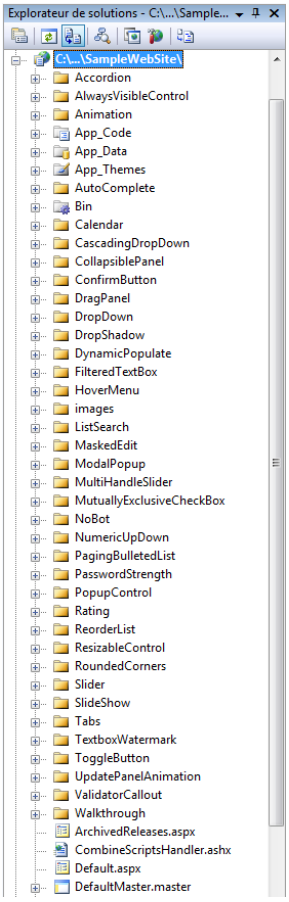
Vous obtiendrez ainsi sur une page vierge dans Visual Studio 2008 avec l'explorateur de solution ouvert contenant les 4 différents projets possible : AjaxControlToolkit, SampleWebSite, ToolkitTests et TemplateVSI.



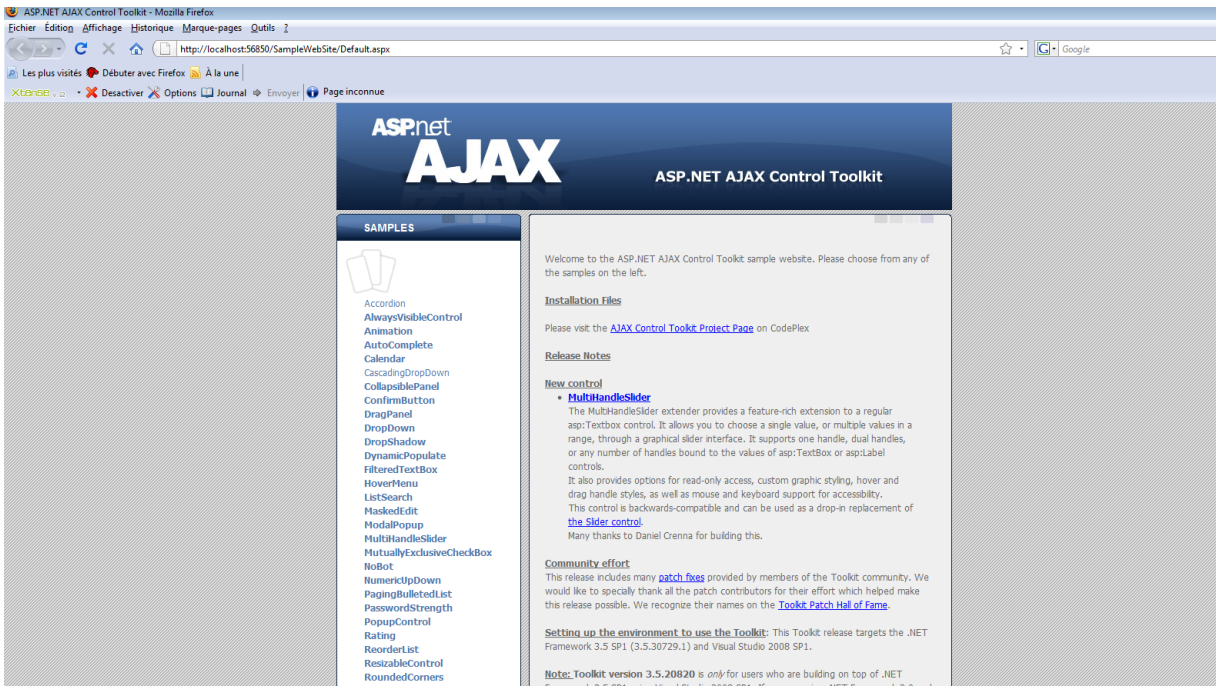
- **AjaxControlToolkit** est un projet qui contient tout simplement le code-source des contrôles Ajax. Ainsi lorsque vous compilerez vous obtiendrez la dll contant tous les contrôles disponibles.
- **SampleWebSite** est un "projet-site" vous permettant d'avoir une démonstration des ACTs disponibles (ce que nous allons décrire ci-dessous dans le but de vous appropriez chaque contrôle que vous manipulerez préalablement).
- **ToolkitTests** est un projet permettant de tester les contrôles ACT dans le cas où l'on modifierait leur code-source.
- **TemplateVSI** est une extension de Visual Studio permettant dans un nouveau projet de créer vos propres extenders.

Remarque : Les extenders sont des contrôles qui étendent d'autres fonctionnalités de contrôles.

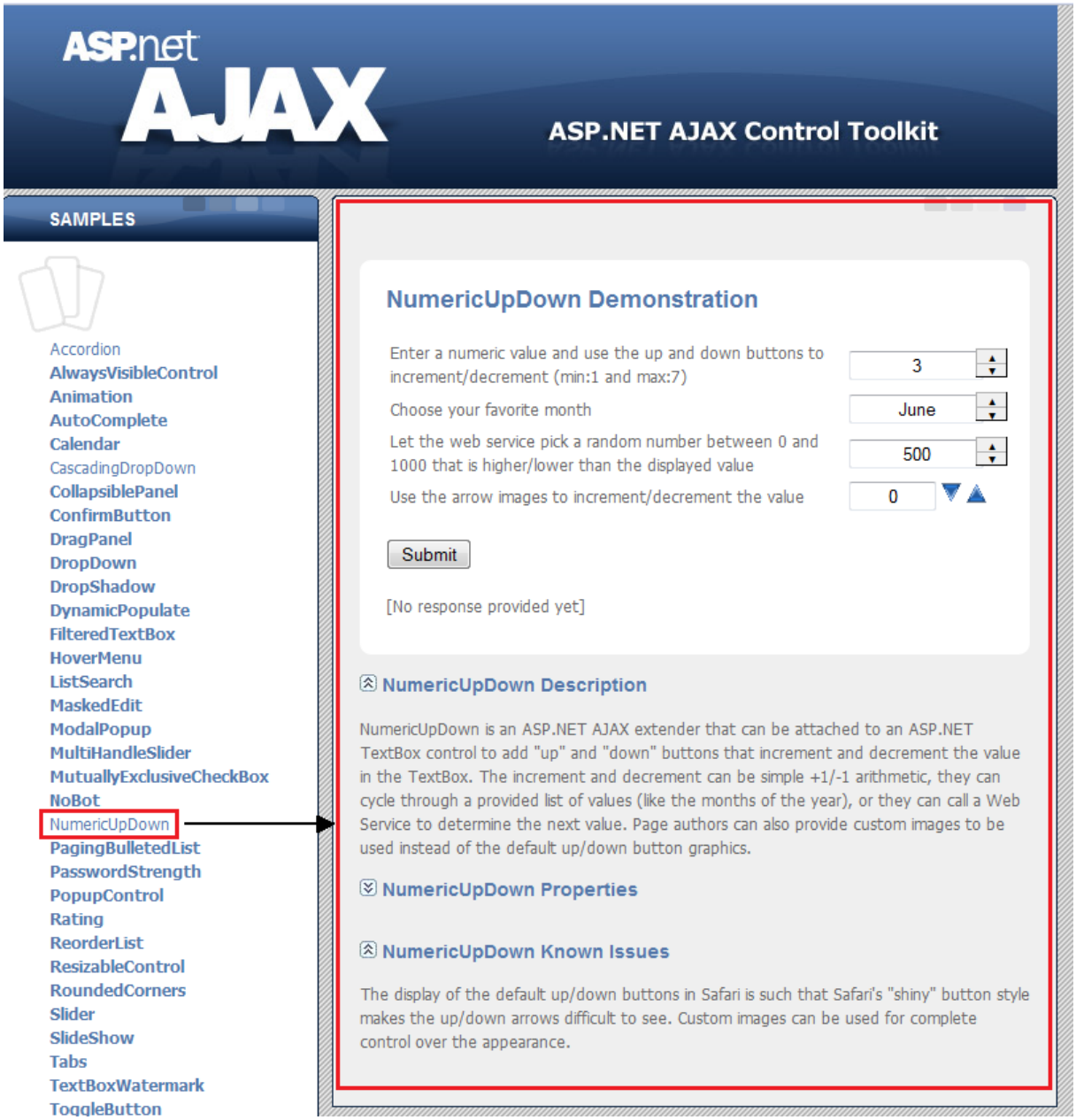
Nous allons donc nous intéresser au projet *SampleWebSite*, qui vous permettra d'accéder à une démonstration de chaque contrôle ACT. Pour accéder au site vous devrez développer le projet *SampleWebSite* comme dans le screen-shot suivant :



Ensuite, sélectionnez le ficher *Default.aspx* à l'aide du clique-droit et choisissez l'option "Afficher dans le navigateur". Ainsi votre navigateur web s'ouvre et vous accéder ainsi à la page suivante :



Dans la partie gauche de votre page, nommée “Sample”, vous obtenez tous les contrôles de l’AjaxControlToolkit. Il vous suffit donc de sélectionner un contrôle et vous accéderez à la démonstration de celui-ci comme dans l’exemple ci-dessous (NumericUpDown)



3 Extenders

Un extender est un contrôle qui va étendre les fonctionnalités d'un autre contrôle. Par exemple il existe un contrôle, qui permet d'ajouter la possibilité d'afficher un calendrier grâce à un bouton accolé à un contrôle TextBox. Un autre permet de créer une zone de saisie, avec des boutons permettant d'augmenter ou diminuer la valeur numérique qu'elle contient (un nombre contenu dans la TextBox). Par conséquent il n'est pas possible de les utiliser seul dans notre page : ils ont besoin d'être lié à un autre contrôle (qu'ils étendent). Dans cette partie du cours, nous allons les présenter et les détailler. Vous pouvez aussi tout tester par vous-même via le site ASP.NET AJAX donné dans la partie précédente.

3.1 AlwaysVisibleControl

Définition :

Ce contrôle permet à un de vos contrôles de rester par-dessus le reste de la page (à un endroit que l'on aura défini) et de suivre le défilement de celle-ci.

Propriétés :

Nom	Description	Obligatoire
TargetControlId	Prend comme valeur l'ID du contrôle ciblé.	OUI
HorizontalOffset	Permet de définir une distance qui servira pour positionner le contrôle ciblé par rapport au bord haut de la page du navigateur (en px).	NON
HorizontalSide	Défini le positionnement horizontal du contrôle (Left, Top, Middle).	NON
VerticalOffset	Permet de définir une distance qui servira pour positionner le contrôle ciblé par rapport au bord gauche de la page du navigateur (en px).	NON
VerticalSide	Défini le positionnement vertical du contrôle (Left, Top, Middle).	NON

Si vous ne remplissez pas les propriétés de placement du contrôle, le contrôle ciblé sera placé en haut à gauche de la page.

Exemple d'utilisation :

Le contrôle AlwaysVisibleControlExtender peut s'appliquer à n'importe quel contrôle de type bloc du moment qu'il possède la propriété *runat="server"*.

ASP.NET

```
<body style="background-color: Silver;">
    <form id="form1" runat="server">
    <div>

        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>

        <p runat="server" id="essai" style="background-color: Yellow; Width: 200px;
           height: 65px; text-align: center;">
            test
        </p>


        <cc1:AlwaysVisibleControlExtender ID="AlwaysVisibleControlExtender1"
           runat="server" TargetControlID="essai" HorizontalSide="Center"
           VerticalSide="Top" />
        <%--On associe ce contrôle avec la balise p et on défini la position en haut
           au milieu--%>


        <p style="height: 1500px;">
            Scrollez la page.
            Le bloc de texte reste en haut au milieu.
        </p>
```

3.2 Animation

Définition :

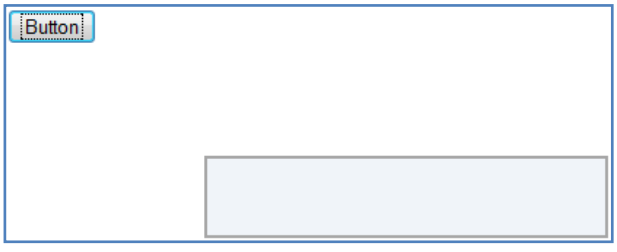
Il s'agit un contrôle qui permet la création d'animation sur votre page Web avec facilité. Ce contrôle possède un Template assez conséquent, par conséquence nous ne décrirons pas entièrement ici par soucis de clarté et de lisibilité. L'exemple vous donne cependant un aperçu de ce qu'il est possible de faire avec ce contrôle.

Propriétés du contrôle :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du contrôle ciblé (sur lequel s'appliqueront les effets du contrôle AJAX).	OUI

Exemple d'utilisation :

Voici une animation qui va déplacer le bloc tout en le redimensionnant.



Page .aspx

```
<asp:Button ID="StartAnimation" runat="server" Text="Button" />
<div ID="AnimateDiv" runat="server"
        style="border: solid black 2px; background-color: #d3dfee;width:
200px;height: 65px;position: absolute;" />
```

```
<cc1:AnimationExtender ID="AnimationExtender1" runat="server"
TargetControlID="StartAnimation">
    <Animations>
        <%--Quand la souris survole l'élément. Il y a aussi OnMouseOut,
OnMouseOver, OnLoad, OnHoverOut et OnHoverOver--%>
        <OnClick>
            <%--On créer une séquence d'animation pour le div AnimateDiv--%>
            <Sequence AnimationTarget="AnimateDiv">
                <%--Ce qui suit s'éxecutera en même temps--%>
                <Parallel Duration="2" Fps="30">
                    <%--Déplacer le bloc suivant les valeurs données--%>
                    <Move Horizontal="500" Vertical="300" />
                    <%--Redimensionner le bloc--%>
                    <Resize Width="400" Height="20" />
                </Parallel>
            </Sequence>
        </OnClick>
    </Animations>
</cc1:AnimationExtender>
```

3.3 AutoComplete

Définition :

AutoComplete est un extender de TextBox qui fournira une auto-complétion de ce que qui est écrit dans la TextBox. Il fonctionne grâce à un service web qui doit avoir pour signature :

```
C#

public string[] RecupererListe(string Text, int count) { ... }
```

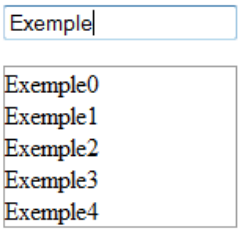
Remarquez que l'on ne peut modifier que le nom de la méthode (RecupererListe) et son contenu.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID de la TextBox cible.	OUI
ServiceMethod	Correspond au nom de la méthode du WebService à appeler.	OUI
ServicePath	Correspond au chemin d'accès vers le fichier contenant le web service. S'il n'est pas rempli, le web service doit correspondre à une méthode de la page.	NON
MinimumPrefixLength	Défini le nombre minimum de caractères qui doivent être entré dans la TextBox pour que le contrôle s'active et appel le service web.	NON
CompletionInterval	Défini le temps en millisecondes après la fin de l'écriture pour que le contrôle fasse sa requête au Web service.	NON
EnableCaching	Active ou non le cache.	NON
FirstRowSelected	Indique si la première ligne de propositions retournées par le service web doit être sélectionnée.	NON

Exemple d'utilisation :

Voici un petit exemple qui, tel quel, ne vous sera pas d'une grande utilité puisqu'il ne renvoi que le texte suivit d'un chiffre. En revanche si vous changez le code du web service pour qu'il recherche dans une base de données, un fichier XML ou autres, cet exemple prend tout son sens. C'est par soucis de compréhension que nous avons simplifié le web service et il vous sera plus facile de tester l'exemple sans base de données à gérer derrière.



Cet exemple est composé d'un web service (un fichier avec l'extension asmx dont nous vous fournissons le code behind) et du code de la page en .aspx. Le délai entre la fin de l'écriture dans la TextBox et le début de la recherche est d'une seconde.

C# du web service

```
[System.Web.Script.Services.ScriptService]
public class WebService1 : System.Web.Services.WebService
{
   public WebService1()
    {
    }

    [WebMethod]
    public string[] RecupererListe(string prefixText, int count)
    {
        string[] tableau ={"", "", "", "", "", ""};


        for(int i = 0; i < 5; i++)
        {
            tableau[i] = prefixText + i;
        }

        return tableau;
    }
}
```

VB.NET du web service

```
<System.Web.Script.Services.ScriptService()> _
Public Class WebService1
    Inherits System.Web.Services.WebService

    Public Sub New()

    End Sub


    <WebMethod()> _
    Public Function RecupererListe(ByVal prefixText As String, ByVal count
As Integer) As String()
        Dim tableau As String() = {"", "", "", "", "", ""}
        Dim i As Integer = 0

        While i < 5
            tableau(i) = prefixText + i
            System.Math.Max(System.Threading.Interlocked.Increment(i), i -
1)
        End While

        Return tableau
```

```
    End Function

End Class
```

ASP.NET

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<cc1:AutoCompleteExtender ID="AutoCompleteExtender1" runat="server"
        TargetControlID="TextBox1"
        ServiceMethod="RecupererListe"
        ServicePath="WebService1.asmx"
        CompletionInterval="1000"
        MinimumPrefixLength="2" />
```

3.4 Calendar

Définition :

Le contrôle Calendar n'est pas qu'un simple calendrier. En effet il s'agit d'un extender qui s'applique communément à une TextBox. Ce contrôle va afficher un calendrier à partir duquel on pourra récupérer le jour choisit par l'utilisateur.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID de la TextBox cible.	OUI
Format	Défini le format de la date (exemple : MMMM d, yyyy).	NON
PopupButtonID	Prend comme valeur l'ID du bouton qui servira à afficher le calendrier.	NON
PopupPosition	Détermine la position du popup par rapport à la TextBox.	NON
SelectedDate	Détermine la date qui sera sélectionné par défaut (à l'ouverture de la popup contenant le calendrier).	NON
CssClass	Défini une classe css qui va permettre d'appliquer un thème au calendrier.	NON

Exemple d'utilisation :

```
ASP.NET

<div>
    <asp:TextBox ID="calendarTextbox" runat="server" />
    <asp:Button ID="calendarButton" runat="server" Text="Calendri
      CausesValidation="false" />
</div>

<cc1:CalendarExtender
        runat="server"
        TargetControlID="calendarTextBox"
        Format="d MMMM yyyy"
        PopupButtonID="calendarButton"
        PopupPosition="BottomLeft" />
```

Vous pouvez voir sur cette image ce qui se passe lorsque l'on a appuyé sur le bouton. Remarquez que comme on a spécifié le PopupButtonID, il faut se servir du bouton pour ouvrir le calendrier. La TextBox ne servant plus qu'à afficher et recevoir la date choisit. Le format de la date dans la TextBox correspond bien à ce que l'on a demandé dans la propriété Format, c'est-à-dire « jour mois année ».

3.5 CascadingDropDown

Définition :

Il permet de remplir (binder) un DropDownList suivant le choix de l'utilisateur sur une autre DropDownList. Par exemple vous avez deux DropDownList : la première correspond à une fourchette de prix et le second à un produit correspondant a cette fourchette. Quand l'utilisateur change de fourchette, les produits disponibles sont mis à jour. Il fonctionne grâce à un service web qui doit avoir pour signature :

```
C#

public CascadingDropDownNameValue[] RecupererContenu(
        string CategoryValues, string category) { ... }
```

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID de la DropDownList à remplir.	OUI
Category	Permet de définir le nom de la catégorie utilisée (correspond, sur le XML, à un nom de tag).	OUI
PromptText	Message qui sera affiché tant que l'utilisateur n'aura pas fait un choix dans la DropDownList parente.	NON
PromptValue	Valeur allant avec PromptText.	NON
EmptyText	Texte à afficher quand la DropDownList n'a aucun contenu à afficher.	OUI
EmptyValue	Valeur correspondant à l'affichage d'EmptyText.	NON

LoadingText	Message qui sera affiché pendant le chargement de la DropDownList.	OUI
ServiceMethod	Correspond au nom du WebService à appeler.	OUI
ServicePath	Correspond au chemin d'accès vers le fichier contenant le web service. S'il n'est pas rempli, le web service doit correspondre à une méthode de la page.	NON
ParentControlID	Prend comme valeur l'ID de la DropDownList parente.	OUI
SelectedValue	Défini la valeur sélectionné par défaut.	NON

Exemple d'utilisation :

Dans cet exemple nous donnerons la possibilité de choisir instrument de musique dans plusieurs DropDownList en fonction de son type, de son modèle et de sa couleur.

Voici le fichier XML qui nous permettra de récupérer toutes les sélections possibles dans nos DropDownList :

XML
```xml
<?xml version="1.0" encoding="utf-8" ?>
<InstrumentService>
  <type name="Cuivre">
    <model name="Flute Traversière">
      <color name="Bleu"/>
      <color name="Blanc"/>
      <color name="Noir"/>
    </model>
    <model name="Trompette">
      <color name="Doré"/>
      <color name="Argenté"/>
      <color name="Cuivre"/>
    </model>
    <model name="Saxophone">
      <color name="Doré"/>
      <color name="Argenté"/>
      <color name="Cuivre"/>
    </model>
  </type>
  <type name="Corde">
    <model name="Guitare">
      <color name="Bois"/>
      <color name="Noir"/>
      <color name="Fantaisie"/>
    </model>
    <model name="Piano">
      <color name="Blanc"/>
      <color name="Noir"/>
      <color name="Bleu"/>
    </model>
    <model name="Violon">
      <color name="Bois"/>
      <color name="Noir"/>
      <color name="Fantaisie"/>
    </model>
  </type>
</InstrumentService>
```

Voici le code source du WebService avec lequel interagit le contrôle AjaxToolkit :

C# du webservice

```
[System.Web.Script.Services.ScriptService]
public class InstrumentService : System.Web.Services.WebService
{
    private static XmlDocument _document;
    private static object _lock = new object();

    public static XmlDocument Document
    {
        get
        {
            lock (_lock)
            // Permet d'éviter des accès simultanés
            {
                if (_document == null)
                {
                    // Récupère le fichier XML
                    _document = new XmlDocument();

                    _document.Load(HttpContext.Current.Server.MapPath("~/InstrumentS
                    ervice.xml"));
                }
            }
            return _document;
        }
    }

    public static string[] Hierarchy
    {
        get
        {
            // Permet de définir les catégory utilisé pour le traitement
            return new string[] { "type", "model" };
        }
    }

    public InstrumentService()
    // Contructeur par défaut
    {
    }

    [WebMethod]
    public AjaxControlToolkit.CascadingDropDownNameValue[]
           GetDropDownContents(string knownCategoryValues, string category)
    {
        // On crée un dictionnaire de valeur qui récupère aussi bien le nom des
           catégories que les valeurs associées
        StringDictionary knownCategoryValuesDictionary =
         AjaxControlToolkit.CascadingDropDown.ParseKnownCategoryValuesString(knownC
         ategoryValues);

        // Requète à effectuer pour retourner les valeurs attendus par le controle
           Ajaxtoolkit.
        return
         AjaxControlToolkit.CascadingDropDown.QuerySimpleCascadingDropDownDocument(
         Document, Hierarchy, knownCategoryValuesDictionary, category);
    }
}
```

VB.NET du webservice

```
Public Class InstrumentService
    Inherits System.Web.Services.WebService

    Shared _Document As XmlDocument
    Shared _lock As New Object

    Public ReadOnly Property Document() As XmlDocument
        Get
            If (_Document Is Nothing) Then
                SyncLock _lock
                ' Permet d'éviter des accès simultanés
                    _Document = New XmlDocument
                    'Récupère le fichier XML
                    _Document.Load(HttpContext.Current.Server.MapPath("~/InstrumentS
                     ervice.xml"))
                End SyncLock

            End If
            Document = _Document
            Exit Property
        End Get
    End Property

    Public ReadOnly Property Hierarchy() As String()
        Get
            ' Permet de définir les catégory utilisé pour le traitement
            Dim _Heirarchy As String() = {"type", "model"}
            Return _Heirarchy
        End Get
    End Property

    <WebMethod()> _
    Public Function GetDropDownContents(ByVal knownCategoryValues As String, ByVal
         category As String) As AjaxControlToolkit.CascadingDropDownNameValue()
        ' On crée un dictionnaire de valeur qui récupère aussi bien le nom des
         catégories que les valeurs associées
        Dim knownCategoryValuesDictionary As New StringDictionary
        knownCategoryValuesDictionary =
            AjaxControlToolkit.CascadingDropDown.ParseKnownCategoryValuesString(know
            nCategoryValues)

        ' Requète à effectuer pour retourner les valeurs attendus par le controle
         Ajaxtoolkit.
        Return
            AjaxControlToolkit.CascadingDropDown.QuerySimpleCascadingDropDownDocumen
            t(Document, Hierarchy, knownCategoryValuesDictionary, category)

    End Function
End Class
```

Voici le code source ASP.NET de la page par défaut :

```
ASP.NET
<%-- Ne pas oublier dans la directive Page de mettre la propriété
     EnableEventValidation à false--%>
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<table>
    <tr>
        <td>
            Type de l'instrument
        </td>
        <td>
            <asp:DropDownList ID="DropDownList1"
                runat="server" Width="170" />
        </td>
    </tr>
    <tr>
        <td>
            Modèle de l'instrument
        </td>
        <td>
            <asp:DropDownList ID="DropDownList2" runat="server"
                Width="170" />
        </td>
    </tr>
    <tr>
        <td>
            Couleur
        </td>
        <td>
            <asp:DropDownList ID="DropDownList3" runat="server"
                Width="170"
                AutoPostBack="true"
                OnSelectedIndexChanged="DropDownList3_SelectedIndexChanged" />
        </td>
    </tr>
</table>
<br />
<ajaxtoolkit:CascadingDropDown ID="CascadingDropDown1" runat="server"
    TargetControlID="DropDownList1"
    Category="type"
    PromptText="Choisissez un type"
    LoadingText="[Récupération des types...]"
    ServicePath="InstrumentService.asmx"
    ServiceMethod="GetDropDownContents" />

<ajaxtoolkit:CascadingDropDown ID="CascadingDropDown2" runat="server"
    TargetControlID="DropDownList2"
    Category="model"
    PromptText="Choisissez un modèle"
    LoadingText="[Récupération des modèles...]"
    ServicePath="InstrumentService.asmx"
    ServiceMethod="GetDropDownContents"
    ParentControlID="DropDownList1" />

<ajaxtoolkit:CascadingDropDown ID="CascadingDropDown3" runat="server"
    TargetControlID="DropDownList3"
    Category="color"
    PromptText="Choisissez une couleur"
    LoadingText="[Récupération des couleurs...]"
    ServicePath="InstrumentService.asmx"
    ServiceMethod="GetDropDownContents"
    ParentControlID="DropDownList2" />

<asp:UpdatePanel ID="UpdatePanel1" runat="server"
    UpdateMode="Conditional"
    RenderMode="inline">
    <ContentTemplate>
        <asp:Label ID="Label1" runat="server"
            Text="[Aucune réponse donnée ...]" />
```

```
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="DropDownList3"
            EventName="SelectedIndexChanged" />
    </Triggers>
</asp:UpdatePanel>
</form>
```

Voici le code behind de cette page par défaut :

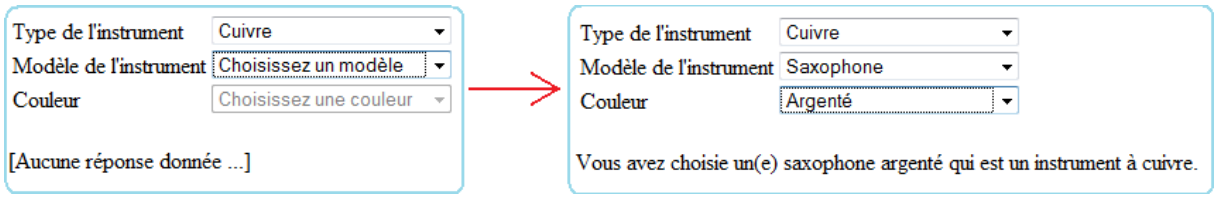
C# de default.aspx.cs

```
protected void DropDownList3_SelectedIndexChanged(object sender, EventArgs
e)
{
    Label1.Text = "Vous avez choisie un(e) "
                + DropDownList2.Text.ToLower()
                + " " + DropDownList3.Text.ToLower()
                + " qui est un instrument à "
                + DropDownList1.Text.ToLower()
                + ".";
}
```

VB.NET de default.aspx.cs

```
Protected Sub DropDownList3_SelectedIndexChanged(ByVal sender As Object, ByVal e As
EventArgs)

    Label1.Text = "Vous avez choisie un(e) " + DropDownList2.Text.ToLower() + " " +
       DropDownList3.Text.ToLower() + " qui est un instrument à " +
       DropDownList1.Text.ToLower() + "."

End Sub
```

Voici un aperçu de cet exemple :



3.6 CollapsiblePanel

Définition :

C'est un extender pour les Panel qui permet d'afficher/cacher son contenu. Il n'y a pas grand-chose à dire dessus, les propriétés et l'exemple vous feront mieux comprendre ce contrôle.

Propriétés du contrôle :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du Panel.	OUI
CollapsedSize	Taille du Panel quand il est en mode « fermé ».	NON
ExpandedSize	Taille du Panel quand il est en mode « étendu ».	NON
Collapsed	Elle permet de définir si lors de l'affichage de la page, le	NON

	contrôle doit être fermé ou étendu.	
AutoCollapse	Si elle est mise à True, le Panel passera en mode fermé quand la souris n'est plus sur le Panel.	NON
AutoExpand	Si elle est mise à True, le Panel passera en mode étendu quand la souris se trouvera sur le Panel.	NON
ScrollContents	Permet d'ajouter une Scrollbar si le contenu dépasse du Panel (si à True).	NON
CollapseControlID	Indique quel contrôle va déclencher le passage en mode collé.	OUI
ExpandControlID	Indique quel contrôle va déclencher le passage en mode fermé.	OUI
TextLabelID	Prend comme valeur l'ID du Label qui va servir pour afficher un texte personnalisé suivant que le Panel soit fermé ou ouvert (respectivement Collapsed et Expanded).	NON
CollapsedText	Texte qui sera affiché dans le Label quand le panel est fermé.	NON
ExpandedText	Texte qui sera affiché dans le Label quand le Panel est ouvert.	NON
ImageControlID	Prend l'ID du contrôle Image qui va servir à afficher une image personnalisé suivant que le Panel soit fermé ou ouvert.	NON
CollapsedImage	Image affiché quand le Panel est fermé.	NON
ExpandedImage	Image affiché quand le Panel est ouvert.	NON
ExpandDirection	Indique le sens d'ouverture du Panel (Verticale ou Horizontale).	NON

Exemple d'utilisation :

ASP.NET

```
<p id="Head" style="border: outset 2px black">
Cliquez ici pour étendre ou fermer <asp:Label ID="ShowText" runat="server" Text=""
/>
</p>
<div>
    <asp:Panel ID="TargetPanel" runat="server">
        L'AJAX Control Toolkit est une extension de la plateforme ASP .NET qui
        permet d'apporter à cette dernière les fonctionnalités AJAX aux applications
        ASP .NET. AJAX Control Toolkit a été conçu de manière à faire partie
        intégrante de l'ASP .NET et par conséquent de s'intégrer correctement avec
        les plateformes et modèle d'application existant.
    </asp:Panel>
</div>

<cc1:CollapsiblePanelExtender
        runat="server"
        CollapseControlID="Head"
        TargetControlID="TargetPanel"
        ExpandControlID="Head"
        TextLabelID="ShowText"
        CollapsedText="(Ouvrir ...)"
        ExpandedText="(Fermer ...)" />
```

Cet exemple est assez simple puisqu'il n'utilise pas de css pour son style et qu'il ne montre pas toutes les propriétés du contrôle.

Cliquez ici pour étendre ou fermer (Fermer ...)

L'AJAX Control Toolkit est une extension de la plateforme ASP .NET qui permet d'apporter à cette dernière les fonctionnalités AJAX aux applications ASP .NET. AJAX Control Toolkit a été conçu de manière à faire partie intégrante de l'ASP .NET et par conséquent de s'intégrer correctement avec les plateformes et modèle d'application existant.

3.7 ConfirmButton

Définition :

Il s'agit d'un extender de bouton qui permet d'ouvrir une fenêtre de confirmation lors d'un clic sur le bouton en question (Button ou LinkButton). Si le bouton « ok » est choisit, le bouton agit normalement. Si l'utilisateur annule, le bouton n'exécute pas son action (il est possible d'exécuter un script lors de l'annulation).

Propriétés du contrôle :

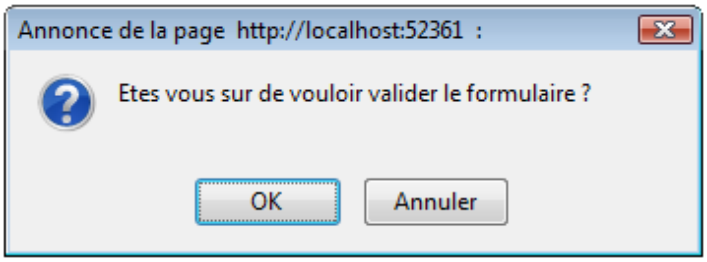
Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du contrôle à étendre.	OUI
ConfirmText	Texte affiché lors de la demande de confirmation (il n'est pas nécessaire pour le fonctionnement du contrôle mais sans lui il n'y aura aucun texte affiché).	OUI
OnClientCancel	Permet d'exécuter un script client (javascript) lors de l'annulation de la confirmation.	NON
ConfirmOnFormSubmit	Spécifie que la demande de confirmation doit attendre que le formulaire soit valide (contrôle de validation ASP.NET ou autre script).	NON

Exemple d'utilisation :

ASP.NET

```
<div>
    <asp:Label runat="server" Text="Label" />
    <asp:TextBox runat="server" />
    <asp:Button ID="ConfirmButton" runat="server" Text="Button" />
</div>

<cc1:ConfirmButtonExtender
        runat="server"
        TargetControlID="ConfirmButton"
        ConfirmText="Etes vous sur de vouloir valider le formulaire ?" />
```

On a donc bien lié notre ConfirmButton avec le bouton. Voici le résultat lorsque l'on clic sur le bouton :



3.8 DragPanel

Définition :

Le DragPanel est un extender de Panel qui permet d'ajouter très facilement la possibilité de déplacer le Panel ciblé sur la page.

Propriétés du contrôle :

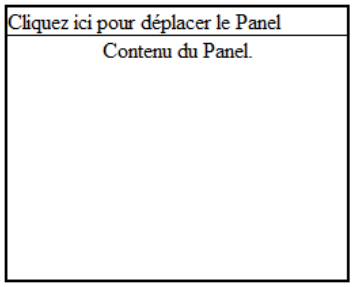
Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du Panel à rendre déplaçable.	OUI
DragHandleID	Prend comme valeur l'ID de l'élément dans le Panel qui va permettre son déplacement.	OUI

Exemple d'utilisation :

ASP.NET

```
<asp:Panel ID="DragPanel" runat="server" style="position: absolute;border: solid 2px
     black;width: 250px;height : 200px">
    <div id="DragElement" style="display: block;border-bottom: solid 1px black">
         Cliquez ici pour déplacer le Panel</div>
    <div style="text-align: center">Contenu du Panel.</div>
</asp:Panel>


<cc1:DragPanelExtender
        runat="server"
        TargetControlID="DragPanel"
        DragHandleID="DragElement" />
```

Voici ce que l'on obtient avec ce code :



3.9 DropDown

Définition :

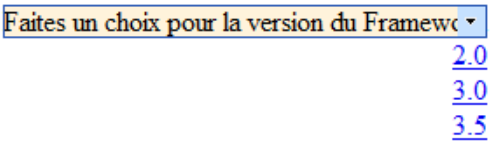
Il permet de transformer un Label en DropDownList à partir d'un Panel. Très simple à utiliser, avec seulement deux propriétés vous pouvez ajouter à un Label la possibilité d'afficher le contenu d'un Panel.

Propriétés du contrôle :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du Label ciblé.	OUI
DropDownControlID	Prend comme valeur l'ID du Panel qui sera utilisé en tant que DropDownPanel.	OUI

Il y a d'autres méthodes qui peuvent vous être utile comme celle permettant de changer l'image de la flèche mais nous n'en parlons pas ici par soucis de lisibilité.

Exemple d'utilisation :



Dans l'exemple que l'on va faire, on va permettre le choix d'une version du Framework. Quand l'utilisateur clic sur un des LinkButton du Panel on va effectuer une action. Dans notre cas on va juste changer le texte du Label pour afficher quel version a été sélectionné. On aurait aussi pu afficher les caractéristiques, le Changelog de la version choisit ou tout autres choses à partir du choix qui a été effectué.

ASP.NET

```
<asp:Label ID="Selection" runat="server" Text="Faites un choix pour la version du
Framework" />

<asp:Panel ID="PanelChoix" runat="server">
    <asp:LinkButton ID="Possibilite1" runat="server"
OnClick="ChangerLabel">2.0</asp:LinkButton><br />
    <asp:LinkButton ID="Possibilite2" runat="server"
OnClick="ChangerLabel">3.0</asp:LinkButton><br />
    <asp:LinkButton ID="Possibilite3" runat="server"
OnClick="ChangerLabel">3.5</asp:LinkButton>
</asp:Panel>

<cc1:DropDownExtender runat="server"
        TargetControlID="Selection"
        DropDownControlID="PanelChoix" />
```

3.10 DropShadow

Définition :

Ce contrôle est un extender qui permet d'ajouter une ombre et/ou des bouts arrondis à un Panel.

Propriétés du contrôle :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du Panel qu'il cible.	OUI
Width	Définit la taille de la DropShadow (par défaut elle est définie à 5).	NON
Opacity	Définit l'opacité de la DropShadow. Elle prend des valeurs entre 0 (complètement transparent) et 1 (opaque). La valeur par défaut est 0.5.	NON
TrackPosition	Définit si le contrôle DropShadow.	NON
Rounded	Définit si le contrôle ciblé aura des bouts arrondis ou non.	NON
Radius	Définit la taille de l'arrondi.	NON

Exemple d'utilisation :

ASP.NET

```
<asp:Panel ID="Panel1" runat="server" style="background: #A8C1DF; width:
        250px;height:175px;" />
<cc1:DropShadowExtender ID="DropShadowExtender1" runat="server"
        TargetControlID="Panel1" Rounded="true" Radius="10" />
```

Comme vous pouvez le constater sur l'image ci-contre, le Panel affiché est vide mais quand même visible puisque l'on a défini une hauteur et une largeur. Il y a bien l'ombre sur le Panel et les bouts du Panel sont arrondis.

3.11 DynamicPopulate

Définition :

DynamicPopulate est un extender qui permet de remplacer le contenu d'un contrôle par le résultat d'un web service ou d'une méthode de la page. Le résultat sera inséré dans le contrôle cible en tant qu'enfant. Il est donc possible de renvoyer directement du HTML et de l'insérer dans un contrôle.

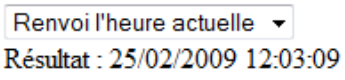
Signature du web service

```
string DynamicPopulateService(string contextKey) { ... }
```

Propriétés du contrôle :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du contrôle à remplir dynamiquement (un Panel).	OUI
ClearContentsDuringUpdate	Permet de supprimer le contenu HTML de la cible durant la mise à jour de celui-ci.	NON
ServiceMethod	Correspond au nom de la méthode à appeler (qui se trouve sur le WebService spécifié avec ServicePath s'il n'est pas dans la page ASPX).	OUI
ServicePath	Correspond au chemin d'accès vers le fichier contenant le web service. S'il n'est pas rempli, le web service doit correspondre à une méthode de la page.	NON
PopulateTriggerControlID	Correspond à l'ID du contrôle qui, lorsqu'il est cliqué servira à mettre à jour le contrôle ciblé.	NON
CacheDynamicResult	Permet de mettre en cache les résultats.	NON

Exemple d'utilisation :



Pour cet exemple on a un contrôle qui va permettre deux choix : afficher le texte « test » ou afficher l'heure. Sur l'évènement Onclick de ces items, on va exécuter une fonction JavaScript qui va récupérer le contrôle DynamicPopulate par son ID et ensuite appeler la méthode populate en lui passant une valeur. Le WebService va être appelé et retourner ce qu'il faut suivant l'option choisi.

Le résultat est envoyé au navigateur qui l'affiche dans le Panel. Remarquez que seul le Panel est rechargé et non toute la page.

Code de la page ASP.NET

```
<script type="text/javascript">
    function dynamicPopulateJS(val)
    {
        var ControlDynamicPopulate = $find("DynPop");
        if(ControlDynamicPopulate)
            ControlDynamicPopulate.populate(val);
    }
</script>

<select>
    <option onclick="dynamicPopulateJS(this.value);" value="0">Renvoi le string
"test"</option>
    <option onclick="dynamicPopulateJS(this.value);" value="1">Renvoi l'heure
actuelle</option>
</select>

<asp:Panel ID="Panel1" runat="server" />

<cc1:DynamicPopulateExtender ID="DynPop"
        runat="server"
        TargetControlID="Panel1"
        ServiceMethod="DynamicPopulateService"
        ServicePath="MonService.asmx" />
```

Code C# du WebService

```
[System.Web.Script.Services.ScriptService]
public class MonService : System.Web.Services.WebService
{
    public MonService()
    {
    }

    [WebMethod]
    public string DynamicPopulateService(string contextKey)
    {
        string valeur = "";

        //On regarde quel valeur a été choisit pour renvoyer le bon texte
        if (contextKey == "0")
            valeur = "test";
        else if (contextKey == "1")
            valeur = DateTime.Now.ToString();

        //On concatène le texte qui doit être renvoyé avec du HTML
        return String.Format("<div>Résultat : {0}</div>", valeur);
    }
}
```

Code VB.NET du WebService

```
<System.Web.Script.Services.ScriptService()> _
Public Class MonService
    Inherits System.Web.Services.WebService

    Public Sub New()
    End Sub


    <WebMethod()> _
    Public Function DynamicPopulateService(ByVal contextKey As String) As String

        Dim valeur As String = ""

        'On regarde quel valeur a été choisit pour renvoyer le bon texte
        If contextKey = "0" Then
            valeur = "test"

        ElseIf contextKey = "1" Then
            valeur = DateTime.Now.ToString()
        End If


        'On concatène le texte qui doit être renvoyé avec du HTML
        Return [String].Format("<div>Résultat : {0}</div>", valeur)

    End Function
End Class
```

3.12 FilteredTextBox

Définition :

Ce contrôle permet de filtrer les caractères que l'utilisateur entre dans une zone de texte. Par exemple on pourra empêcher qu'il entre des nombres : s'il essaye, aucun nombre ne sera écrit dans la zone de texte.

Attention : Comme toutes les autres vérifications et filtres qui agissent sur le navigateur du client, ce système peut être contourné (en désactivant le JavaScript par exemple). Il faut donc toujours vérifier les données envoyé par l'utilisateur sur le serveur pour s'assurer qu'elles correspondent à ce que nous attendons.

Propriétés :

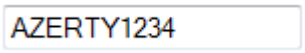
Nom	Description	Obligatoire
TargetControlID	Prend l'ID du contrôle TextBox cible.	OUI
FilterType	Correspond aux types que l'on acceptera dans la TextBox. On peut en mettre plusieurs séparé par une virgule (peut prendre comme valeur : Number, LowercaseLetters, UppercaseLetters et Custom). Par défaut il est sur Custom.	OUI
FilterMode	Spécifie le mode de filtrage si FilterType est sur Custom. Il prend comme valeur ValidChars et InvalidChars.	NON
ValidChars	Spécifie les caractères valides qui peuvent donc être mis dans la TextBox. Ne fonctionne que si FilterType est mis à Custom.	NON
InvalidChars	Spécifie les caractères valides qui ne peuvent pas être mis dans la TextBox. Ne fonctionne que si FilterType est mis à Custom.	NON
FilterInterval	Spécifie un intervalle de temps (en milliseconde) qui correspond à l'attente qu'il y a avant de filtrer la TextBox.	NON

Exemple d'utilisation :

ASP.NET

```
<div>
    <asp:Label runat="server" Text="Label" />
    <asp:TextBox id="TextBoxCible" runat="server" />
</div>


<cc1:FilteredTextBoxExtender
        runat="server"
        TargetControlID="TextBoxCible"
        FilterType="Numbers,UppercaseLetters"
        ValidChars="&éà" />
```

Vous pouvez voir sur l'image ci-contre le type de caractères accepté par la TextBox. Remarquez que comme on n'a pas mis FilterType sur Custom, les caractères du ValidChars ne sont pas pris en compte.



Pour faire fonctionner InvalidChars il aurait fallu mettre FilterType sur Custom et FilterMode sur InvalidChars. Ainsi les caractères de la propriété InvalidChars sont pris en compte.

3.13 ListSearchExtender

Définition :

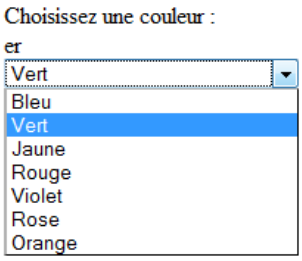
Ce contrôle permet d'étendre les fonctionnalités des ListBox et des DropDownList en rajoutant la possibilité d'effectuer une recherche dans ceux-ci en tapant au clavier ce que nous cherchons. C'est un contrôle très utile quand il y a beaucoup de données.

Propriétés du contrôle :

Nom	Description	Obligatoire
PromptText	Message à afficher quand le contrôle a le focus. Le message sera ensuite remplacé par la recherche de l'utilisateur (par défaut le message est "Type to search").	NON
TargetControlID	Prend comme valeur l'ID du contrôle à étendre.	OUI
PromptCssClass	Définit le nom de la classe CSS qui sera appliqué au message du prompt.	NON
PromptPosition	Définit la position du prompt par rapport eu contrôle. Il peut se trouver au dessus, en dessous, à gauche ou à droite. Par défaut il se trouve au dessus.	NON
QueryPattern	Indique la façon dont la recherche doit être effectuée. Par défaut il recherche ce que l'on a entré au début des mots (StartsWith). On peut aussi lui dire de recherche les entrées qui contiennent la recherche (Contains).	NON
QueryTimeout	Défini à partir de combien de temps (en millisecondes) le navigateur va attendre avant de redémarrer une recherche (reset) si aucune correspondance n'est trouvé. Par défaut à zéro, il ne reset pas la recherche.	NON

Exemple d'utilisation :

Dans cette exemple on propose de choisir une couleur parmis celles contenues dans la DropDownList. Pour l'utilisation de ce contrôle, il faut faire attention à ce qui se trouve au dessus de la DropDownList. Ici nous avons mis deux *br* pour que le prompt ne soit pas sur le texte qui se trouve avant.



ASP.NET

```
Choisissez une couleur :
<br /><br />
<asp:DropDownList ID="DropDownList1" runat="server" style="width: 200px">
    <asp:ListItem>Bleu</asp:ListItem>
    <asp:ListItem>Vert</asp:ListItem>
    <asp:ListItem>Jaune</asp:ListItem>
    <asp:ListItem>Rouge</asp:ListItem>
    <asp:ListItem>Violet</asp:ListItem>
    <asp:ListItem>Rose</asp:ListItem>
    <asp:ListItem>Orange</asp:ListItem>
</asp:DropDownList>

<cc1:ListSearchExtender
        runat="server"
        TargetControlID="DropDownList1"
        PromptText="Tappez votre recherche :"
        QueryPattern="Contains"
        QueryTimeout="3000" />
```

3.14 MaskedEdit

Définition :

Ce contrôle est un extender de TextBox qui permet d'empêcher la saisie de caractère non voulu. Il vérifie que ce que rentre l'utilisateur correspond bien à ce que nous avons spécifié dans le code (une lettre suivit de deux chiffres par exemple). On peut aussi effectuer un formatage de la TextBox. On aura ainsi le format de ce que l'on attend de l'utilisateur déjà présent. Par exemple on peut demander une date à l'utilisateur. Dans ce cas on peut afficher - - / - - / - - dans la TextBox au moment de la saisie de l'utilisateur. En outre il n'a pas besoin des contrôles de validation puisqu'il vérifie déjà lui-même que les caractères entré sont correct et qu'il possède déjà un contrôle de validation spécifique : MaskedEditValidator.

Propriétés du contrôle MaskedEditExtender :

Nom	Description	Obligatoire
MaskType	Définit le type de validation qui sera effectuée. None : pas de validation Number : Validation d'un nombre Date : Validation d'une date Time : Validation d'un horaire DateTime : Validation d'une date et d'un horaire	NON
AcceptNegative	Permet d'accepter les nombres négatifs : Left : Afficher le signe – à gauche de la TextBox Right : Afficher le signe - à droite de la TextBox	NON
ClearMaskOnLostFocus	Définit si on vide la TextBox quand cette dernière perd le focus.	NON
ClearTextOnInvalid	Définit si on vide la TextBox quand ce qui a été saisie n'est pas valide.	NON
ClipboardEnable	Permet d'accepter ou de refuser la saisie de données depuis le presse papier.	NON
DisplayMoney	Spécifie la manière dont le symbole monétaire est affiché. S'utilise comme AcceptNegative.	NON
ErrorTooltipEnabled	Permet d'afficher un message lors du survol de la TextBox si elle est invalide.	NON
Filtered	Permet de spécifier les caractères autorisés dans la TextBox avec un masque de type C (voir Mask).	NON
InputDirection	Définit la direction dans laquelle sera ajouté ce que l'on écrit dans la TextBox. De gauche à droite (LeftToRight) ou de droite à gauche (RightToLeft).	NON
TargetControlID	Prend comme valeur l'ID de la TextBox à étendre.	OUI
Mask	Prend comme valeur une chaîne de caractère qui sera le masque à appliquer à cette TextBox. On créait un Template qui défini ce que l'on est autorisé à entrer dans le zone de texte. Voici la correspondance des caractères que l'on peut mettre dans cette propriété et leurs actions dans le masque : ? -> Tous les caractères L -> Une lettre $ -> Une lettre ou un espace 9 -> Un chiffre	OUI

	C -> Seuls les caractères spécifiés dans la propriété Filtered sont acceptés A -> Une lettre ou les caractères personnalisés (C) N -> Un chiffre ou les caractères personnalisés (C) Il existe aussi des séparateurs pour les masques comme l'heure et la date : / -> Séparateur pour la date : -> Séparateur pour l'heure . -> Séparateur de décimale Le caractère \ est le caractère d'échappement pour les caractères spéciaux que l'on vient de voir.	

Propriétés du contrôle MaskedEditValidator :

Nom	Description	Obligatoire
ControlToValidate	Prend comme valeur l'ID de la TextBox à valider.	OUI
ControlExtender	Prend comme valeur l'ID du MaskedEditExtender.	OUI
InitialValue	Définit la valeur initiale de la TextBox.	NON
IsValidEmpty	Définit si la Textbox est valide lorsqu'elle est vide.	NON
MaximumValue	Définit le nombre maximum que peut prendre la TextBox.	NON
MinimumValue	Définit le nombre minimum que peut prendre la TextBox.	NON
ValidationExpression	Définit une expression régulière qui sera utilisé pour la validation.	OUI

Exemple d'utilisation :

On va créer une TextBox dans lequel on ne rentrera que des nombres entre 1000 et 9000 et qui, lorsqu'il sera sélectionné, affichera des underscores « _ » aux emplacements où il manque un nombre.

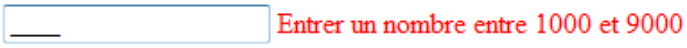
Page ASPX

```
<asp:TextBox ID="MaskedEditTextBox" runat="server"></asp:TextBox>

<cc1:MaskedEditExtender ID="Extender" runat="server"
    TargetControlID="MaskedEditTextBox"
    Mask="9999"
    MaskType="Number"
    AutoComplete="true"
    ClearMaskOnLostFocus="true"
    MessageValidatorTip="true"
    InputDirection="LeftToRight" />

<cc1:MaskedEditValidator ID="Validator" runat="server"
    ControlExtender="Extender"
    ControlToValidate="MaskedEditTextBox"
    EmptyValueMessage="Le nombre est vide"
    IsValidEmpty="false"
    TooltipMessage="Entrer un nombre entre 1000 et 9000"
    MinimumValue="1000"
```

```
    MaximumValue="9000"
    MinimumValueMessage="Le nombre est plus petit que 1000"
    MaximumValueMessage="Le nombre est plus grand que 9000" />
```

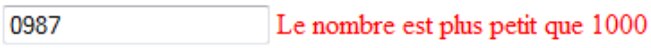
Voici une liste de ce qui va se passer dans cet exemple suivant ce que vous faites :

Si vous sélectionnez la TextBox :



Si vous le laissez vide et le désélectionnez :

Si vous mettez un nombre plus petit que 1000 :



Si vous mettez un nombre plus grand que 9000 :

Et si vous rentrez un nombre valide vous aurez ceci :

3.15 ModalPopup

Définition :

La ModalPopup permet d'afficher un message dans un bloc qui sera affiché au dessus du reste du site. Ce contrôle est très pratique pour afficher un complément d'informations lors du clic sur l'élément que l'on a défini, mais il permet aussi d'exécuter du code Javascript sur le client lorsque celui-ci appuie sur le bouton de validation ou d'annulation. Ces deux boutons correspondent à des contrôles que nous avons placés nous même dans la ModalPopup et définis dans les propriétés du contrôles.

Il est à prendre en considération que lorsque la ModalPopup est affichée, il est impossible d'interagir avec le reste de la page.

Propriétés :

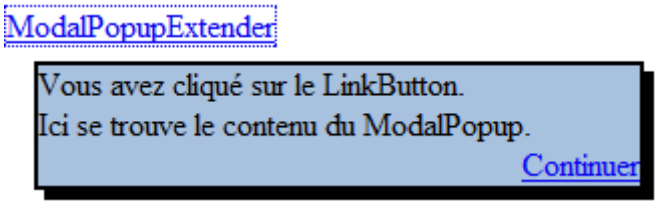
Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du contrôle à étendre pour activer la ModalPopup.	OUI
PopupControlID	Prend comme valeur l'ID de l'élément à afficher comme une ModalPopup.	OUI
BackgroundCssClass	Définit le nom de la classe CSS à appliquer au fond de la ModalPopup.	NON
DropShadow	Définit si la ModalPopup aura une ombre (à true) ou non.	NON
OkControlID	Prend comme valeur l'ID du contrôle qui correspond au bouton Ok.	OUI

OnOkScript	Script à exécuter sur le client (javascript) lorsque celui-ci appuie sur le contrôle de validation.	NON
CancelControlID	Prend comme valeur l'ID du contrôle qui correspond au bouton cancel.	NON
OnCancelScript	Script à exécuter sur le client (javascript) lorsque celui-ci appuie sur le contrôle d'annulation.	NON
PopupDragHandleControlID	Prend comme valeur l'ID de l'élément qui sera utilisé pour déplacer la ModalPopup.	NON
X	Indique la coordonné X pour positionner la ModalPopup (par rapport au côté gauche de la page du navigateur). Elle sera centrée par défaut.	NON
Y	Indique la coordonné Y pour positionner la ModalPopup (par rapport au haut de la page du navigateur). Elle sera centrée par défaut.	NON
RepositionMode	Permet de définir si la ModalPopup doit être redimensionnée dans le cas où la fenêtre sera redimensionnée ou "scrollée".	NON

Exemple d'utilisation :

ASP.NET

```
<asp:LinkButton ID="OuvrirModalPopup"
runat="server">ModalPopupExtender</asp:LinkButton>

<asp:Panel ID="ModalPopup" runat="server" style="border: solid 2px black;background-
color: #A8C1DF;width: 300px">
    Vous avez cliqué sur le LinkButton.<br />
    Ici se trouve le contenu du ModalPopup.<br />
    <div style="text-align: right">
        <asp:LinkButton ID="ValidButton" runat="server">Continuer</asp:LinkButton>
    </div>
</asp:Panel>

<cc1:ModalPopupExtender runat="server"
        TargetControlID="OuvrirModalPopup"
        PopupControlID="ModalPopup"
        OkControlID="ValidButton"
        DropShadow="True" />
```



3.16 MultiHandleSlider

Définition :

Ce contrôle est un extender de TextBox qui permet d'avoir plusieurs Handle (ou le pointeur). Il permet de choisir plusieurs valeurs sur le Slider (la barre sur laquelle (lesquelles) se trouve(nt) le(s) curseur(s)). Le reste ressemble à l'extender Slider qui lui ne possède qu'un Handle.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Prend comme valeur l'ID du TextBox qui sera remplacé par un Slider.	OUI
Minimum	Valeur minimum du Slider.	NON
Maximum	Valeur maximum du Slider.	NON
Length	Taille du Slider en pixel (correspond au Width).	NON
Decimals	Nombre de décimale pour la valeur du Slider.	NON
Steps	Nombre de valeurs possibles dans le Slider (exemple : a 5 il y aura 5 valeurs possibles dans le Slider : Le minimum, a 1 quart, le milieu (2 quart), a 3 quart et le maximum).	NON
Orientation	Définit le sens du Slider : Horizontal ou Vertical.	NON
EnableHandleAnimation		NON
EnableRailClick	Au clic sur le rail, déplace le Handle le plus proche jusqu'à cette valeur.	NON
EnableInnerRangeDrag	Permet de déplacer deux Handles simultanément lorsque l'on clic sur le rail et qu'on déplace la souris.	NON
EnableKeyboard	Permet le changement des valeurs à partir du clavier.	NON
EnableMouseWheel	Permet le changement des valeurs avec la roulette de la souris.	NON
ReadOnly	Définit si le Slider est en ReadOnly ou non. Si oui les valeurs ne pourront pas être modifiées.	NON
Increment	Détermine de combien la valeur sera modifiée si on utilise le clavier ou la roulette de la souris.	NON
HandleAnimationDuration	Définit en secondes le temps de l'animation.	NON
RaiseChangeOnlyOnMouseUp	Définit un événement du contrôle doit être levé lorsque le Handle est modifié.	NON
TooltipText	Texte à afficher lorsque le Handle est survolé : l'argument {0} sera remplacé par la valeur courante du Handle en question.	NON
MultiHandleSliderTargets	Définit une description pour chaque Handle sur le Slider.	NON

Exemple d'utilisation :

Dans cet exemple vous allez voir comment utiliser ce contrôle avec deux Handler. Nous y avons activé l'utilisation de la molette et le clic sur le rail pour déplacer le HandleSlider le plus proche.

Page ASPX

```
<asp:TextBox ID="Handle1" runat="server" /><br />
<asp:TextBox ID="Handle2" runat="server" />
<asp:TextBox ID="Slider" runat="server" />
```

```
<cc1:MultiHandleSliderExtender runat="server" TargetControlID="Slider"
    Minimum="-200" Maximum="200" Steps="5" Length="100"
    EnableMouseWheel="true" EnableRailClick="true">
    <%--C'est ici que ce trouveront vos HandleSlider. Leur ControlID pointe
    vers une TextBox qui contiendra dynamiquement la valeur du HandleSlider--%>
    <MultiHandleSliderTargets>
        <cc1:MultiHandleSliderTarget ControlID="Handle1" />
        <cc1:MultiHandleSliderTarget ControlID="Handle2" />
    </MultiHandleSliderTargets>
</cc1:MultiHandleSliderExtender>
```

3.17 MutuallyExclusiveCheckBox

Définition :

Le MutuallyExclusiveCheckBox permet de transformer vos CheckBox en RadioButton, tout en conservant l'aspect graphique d'une CheckBox. Ce contrôle permet de conserver une homogénéité de votre site, par exemple lorsque vous proposez un formulaire à choix multiple et que vous désirez empêcher des sélections incohérentes.

Propriétés du contrôle :

Nom	Description	Obligatoire
TargetControlID	Permet de lier la MutuallyExclusiveCheckBox à une CheckBox qui bénéficiera des fonctionnalités de ce contrôle.	OUI
Key	En éditant cette propriété, vous créez un identifiant unique qui permettra aux autres MutuallyExclusiveCheckBox de communiquer entre eux.	OUI

Exemple d'utilisation :

Dans cet exemple, nous allons créer plusieurs CheckBox, donnant la possibilité à l'utilisateur de choisir son sexe. Pour cela nous mettrons nos CheckBox dans un tableau HTML. L'utilisateur aura la possibilité de choisir entre **homme**, **femme**, **anonyme**.

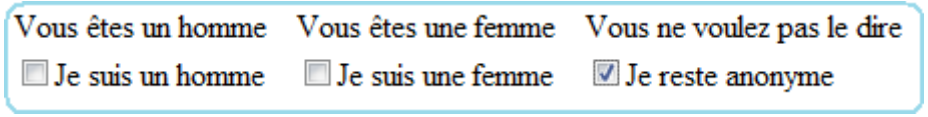
Voici le code correspondant :

ASP.NET

```
<table style="width: 40%;">
    <tr>
        <td>
            <span lang="fr">Vous êtes un homme</span>
        </td>
        <td>
            <span lang="fr">Vous êtes une femme</span>
        </td>
        <td>
            <span lang="fr">Vous ne voulez pas le dire</span>
        </td>
    </tr>
    <tr>
        <td>
            <asp:CheckBox ID="Male" runat="server" Text="Je suis un homme" />
            <ajaxToolkit:MutuallyExclusiveCheckBoxExtender ID="MaleEx"
                TargetControlID="Male"
                Key="Sexe" runat="server">
            </ajaxToolkit:MutuallyExclusiveCheckBoxExtender>
        </td>
        <td>
            <asp:CheckBox ID="Female" runat="server" Text="Je suis une femme" />
            <ajaxToolkit:MutuallyExclusiveCheckBoxExtender ID="FemaleEx"
                TargetControlID="Female"
                Key="Sexe" runat="server">
            </ajaxToolkit:MutuallyExclusiveCheckBoxExtender>
        </td>
        <td>
            <asp:CheckBox ID="Anonyme" runat="server" Text="Je reste anonyme" />
            <ajaxToolkit:MutuallyExclusiveCheckBoxExtender ID="AnonymeEx"
                TargetControlID="Anonyme"
                Key="Sexe" runat="server">
            </ajaxToolkit:MutuallyExclusiveCheckBoxExtender>
        </td>
    </tr>
</table>
```

Comme vous pouvez le constater, chaque MutuallyExclusiveCheckBox est lié à la CheckBox du dessus grâce à la propriété `TargetControlID` qui prend en paramètre l'ID d'une CheckBox. Vous remarquerez que chaque propriété `Key` possède la même valeur.

Voici un aperçu de cet exemple :



Nous pouvons souligner que vous ne pouvez pas sélectionner plusieurs CheckBox en même temps.

3.18 NumericUpDown

Définition :

Ce contrôle est une alternative à la DropDownList, à la différence près que ce contrôle ne donne pas la possibilité choisir parmi une liste de valeur mais plutôt d'incrémenter ou décrémenter la valeur visualisée en fonction des paramètres que vous avez définis.

Propriété :

Nom	Description	Obligatoire
TargetControlID	Permet de lier la NumericUpDown à une TextBox qui bénéficiera des fonctionnalités de ce contrôle.	OUI
Width	Combine la taille de la TextBox associée avec la taille des boutons que ce contrôle rajoute.	OUI
RefValues	Permet de définir une liste exhaustive dans laquelle les valeurs vont osciller.	NON
Step	Permet de définir le pas d'incrémentation (1 par défaut).	NON
Minimum **Maximum**	Permet de définir une valeur maximum/minimum que va pouvoir prendre notre contrôle.	NON
TargetButtonDownID **TargetButtonUpID**	Permet de définir une image pour un affichage personnalisé du bouton d'incrémentation ou de décrémentation.	NON
ServiceDownPath **ServiceUpPath**	Permet de définir l'emplacement d'un Web Service qui comportera des méthodes destinées à un traitement personnalisé de votre incrémentation/décrémentation	NON
ServiceDownMethod **ServiceUpMethod**	Permet de spécifier la méthode à utiliser contenue dans votre Web Service pour traiter l'incrémentation/décrémentation.	NON

Exemple d'utilisation :

Pour cet exemple nous allons créer deux contrôles NumericUpDown. Le premier nous permettra d'afficher des nombres allant de 1 à 31, le second contiendra les mois de l'année.

Voici le code correspondant :

ASP.NET

```
<asp:TextBox runat="server" ID="TxtDay" Text="1"></asp:TextBox>
<ajaxToolkit:NumericUpDownExtender runat="server" ID="NUDDay"
TargetControlID="TxtDay"
    Width="100" Minimum="1" Maximum="31">
</ajaxToolkit:NumericUpDownExtender>
<asp:TextBox runat="server" ID="TxtMonth" Text="Janvier"></asp:TextBox>
<ajaxToolkit:NumericUpDownExtender runat="server" ID="NUDMonth"
    TargetControlID="TxtMonth"
    Width="100"
    RefValues="Janvier;Février;Mars;Avril;Mai;Juin;Juillet;Aout;Septembre;Octobre;No
    vembre;Decembre">
</ajaxToolkit:NumericUpDownExtender>
```

Comme vous pouvez le constater nous avons lié nos deux contrôle Ajax à nos contrôles standards. Pour la propriété `RefValues`, notez que nous avons ajouté les mois de l'année par ordre croissant, l'incrémentation par défaut respectera cet ordre.

Voici un aperçu de cet exemple :

Bug répertorié :

- La propriété Minimum ne peut pas prendre une valeur supérieure à 100, au-delà de cette valeur, le contrôle ne répond plus.

3.19 PagingBulletedList

Définition :

Le PagingBulletedList n'est ni plus ni moins qu'une extension du BulletedList. En effet ce contrôle Ajax permet de paginer les éléments contenus dans votre BulletedList. Il y a deux manière de paginer : par index et par éléments maximum par page.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le PagingBulletedList à une BulletedList qui bénéficiera des fonctionnalités de ce contrôle.	OUI
ClientSort	Prend un booléen en paramètre et permet de trier les éléments de notre liste par ordre alphanumérique (à *false* par défaut).	NON
IndexSize	Permet de déterminer la taille de l'index (à 1 par défaut).	NON
MaxItemPerPage	Détermine le nombre d'éléments maximum à afficher sur la page (désactivé par défaut). Cette propriété désactive automatiquement l'**IndexSize.**	NON
Separator	Définis le caractère de séparation entre les différents index (« - » par défaut).	NON
SelectIndexCssClass **UnselectIndexCssClass**	Permet de choisir un style dans votre CSS pour les index sélectionnés ou non.	NON

Exemple d'utilisation :

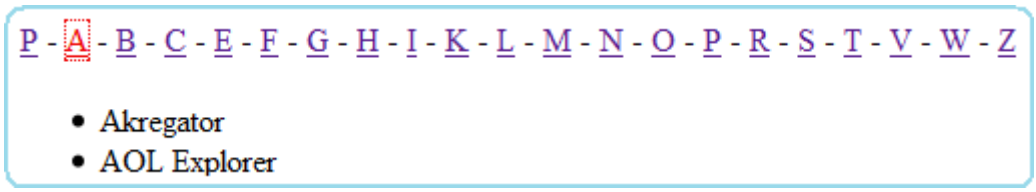
Nous allons créer une liste d'éléments dans une BulletedList puis nous lierons notre contrôle standard à notre contrôle Ajax.

ASP.NET

```
<asp:BulletedList ID="BulletedList1" runat="server">
    <asp:ListItem>Pistach </asp:ListItem>
    <asp:ListItem>Akregator </asp:ListItem>
    <asp:ListItem>AOL Explorer </asp:ListItem>
    <asp:ListItem>Blam! </asp:ListItem>
    <asp:ListItem>Camino </asp:ListItem>
    <asp:ListItem>Epiphany </asp:ListItem>
    <asp:ListItem>FeedBeast </asp:ListItem>
    <asp:ListItem>Gnus </asp:ListItem>
    <asp:ListItem>Hubdog </asp:ListItem>
    <asp:ListItem>IBM Lotus Notes </asp:ListItem>
    <asp:ListItem>iCab </asp:ListItem>
    <asp:ListItem>K-Meleon </asp:ListItem>
    <asp:ListItem>Liferea </asp:ListItem>
    <asp:ListItem>Mail </asp:ListItem>
    <asp:ListItem>NetNewsWire </asp:ListItem>
    <asp:ListItem>Omea </asp:ListItem>
    <asp:ListItem>Pegasus Mail </asp:ListItem>
    <asp:ListItem>RSS Bandit </asp:ListItem>
    <asp:ListItem>Safari </asp:ListItem>
    <asp:ListItem>Tencent Traveler </asp:ListItem>
    <asp:ListItem>Vienna </asp:ListItem>
    <asp:ListItem>Windows Live Mail </asp:ListItem>
    <asp:ListItem>Zimbra </asp:ListItem>
</asp:BulletedList>
<ajaxToolkit:PagingBulletedListExtender ID="PagingBulletedListExtender1"
    runat="server"
    TargetControlID="BulletedList1">
</ajaxToolkit:PagingBulletedListExtender>
```

Comme vous pouvez le constater nous avons géré le minimum de propriété possible.

Voici un aperçu de cet exemple :



Nous n'avons pas mis *True* à la propriété **ClientSort** ce qui explique que notre index n'est pas trié.

3.20 PasswordStrength

Définition :

Le PasswordStrength est une extension aux contrôles standards de type TextBox. Il permet de signifier à l'utilisateur si son mot de passe est assez sécurisé ou non, selon votre politique de sécurité : un mot de passe de minimum 10 caractère comprenant 3 symboles et 2 chiffres peut représenter un mot de passe sécurisé selon votre politique.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le PasswordStrength à une TextBox qui bénéficiera des fonctionnalités de ce contrôle.	OUI

DisplayPosition	Permet de déterminer la position de l'indicateur de sécurité du mot de passe par rapport à la TextBox auquel le PasswordStrength est lié.	NON
StrengthIndicatorType	Détermine le type d'indicateur de sécurité (BarIndicator ou Text).	NON
PreferredPasswordLength	Permet de définir la taille de votre mot de passe que vous considérez comme sécurisé.	NON
PrefixText	Permet de définir un texte qui apparaitra juste avant l'indicateur de sécurité (uniquement pour le type Text).	NON
TextCssClass	Permet d'appliquer un style sur les indicateurs de type Text.	NON
MinimumNumericCharacters **MinimumSymbolCharacters**	Permet de définir un nombre minimum de symboles/nombres à entrer pour avoir un mot de passe sécurisé.	NON
RequiresUpperAndLowerCaseCharacters	Permet de définir si on prend en compte la gestion des lettres majuscules ou minuscules pour avoir un mot de passe sécurisé (a *false* par défaut).	NON
MinimumLowerCaseCharacters **MinimumUpperCaseCharacters**	Permet de définir un nombre minimum de caractère majuscule/minuscule pour avoir un mot de passe sécurisé.	NON
TextStrengthDescriptions	Permet de personnaliser le texte affiché en fonction de l'entrée utilisateur pour un indicateur de type Text. Peut prendre 2 à 10 indications différentes par ordre de sécurité croissante. Chacune d'elle doit être séparée par un point virgule.	NON
CalculationWeightings	Permet de définir nous même le niveau d'importance pour un type de vérification. Par défaut ce sont les valeurs suivantes qui sont prises : 50 ;15 ;15 ;20. La somme de ces valeurs doit impérativement être égal à 100 et chacune à une signification : - La première valeur est l'indice d'importance pour la taille du mot de passe ; - La seconde est l'indice d'importance pour le nombre de caractère numérique ; - La troisième est l'indice d'importance pour la casse (majuscule et minuscule) ; - La dernière valeur est l'indice d'importance pour les symboles.	NON
BarBorderCssClass	Permet d'appliquer un style à la bordure de l'indicateur de type BarIndicator.	NON
BarIndicatorCssClass	Permet d'appliquer un style à l'indicateur de type BarIndicator.	NON
StrengthStyles	Permet d'appliquer plusieurs styles différents en fonction du niveau de sécurité du mot de passe. Il faut séparer chacun des styles par un point virgule.	NON
HelpStatusLabelID	Permet de lier notre à un Label d'aide d'où il pourra indiquer les caractères manquant pour avoir un mot de passe le plus sécurisé.	NON
HelpHandleCssClass	Permet d'appliquer un style à l'élément **HelpHandlePosition**. Obligatoire si vous utilisez	NON

	HelpHandlePosition.	
HelpHandlePosition	Permet d'afficher et de choisir la position d'un mini bouton qui lors d'un clic affichera une boite de dialogue informant des caractères manquant pour avoir une sécurité maximum pour le mot de passe. A lier impérativement avec **HelpHandleCssClass.**	NON

Exemple d'utilisation :

Nous allons créer une TextBox de type Password qui sera liée à notre contrôle Ajax.

ASP.NET

```
<asp:TextBox ID="TxtPassword" runat="server" TextMode="Password"></asp:TextBox>
<ajaxToolkit:PasswordStrength ID="PasswordStrength1" runat="server"
TargetControlID="TxtPassword">
</ajaxToolkit:PasswordStrength>
```

Une fois encore nous avons choisi d'éffectuer un exemple minimaliste.

Voici un aperçu de cet exemple :

Vous remarquerez qu'en fonction de notre entré utilisateur, le texte sur la droite se modifie au fur et à mesure.

3.21 PopupControl

Définition :

Le PopupControl est un contrôle Ajax qui permet à la sélection d'une TextBox d'afficher un contrôle permettant de remplir automatiquement la TextBox. Tous les contrôles disposant d'une méthode `OnSelectedIndexChanged` peuvent être utilisés pour cette sélection (Ex : RadioButtonList, Calendar). La gestion de ce contrôle s'effectue en partie en code behind. Les contrôles utilisés pour la sélection doivent être imbriqués dans un Panel puis dans un UpdatePanel.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le PopupControl à une TextBox qui bénéficiera des fonctionnalités de ce contrôle.	OUI
PopupControlID	Permet de lier un Panel avec le contrôle Ajax, qui bénéficiera des fonctionnalités du contrôle Ajax.	OUI
Position	Permet de déterminer le lieu d'affichage de notre Panel.	NON
CommitProperty	Permet de définir la variable qui sera prise en compte pour afficher votre sélection.	OUI

CommitScript	Permet de modifier l'affichage après la sélection (ajouter du texte avant ou après l'élément sélectionné dans la TextBox).	NON
OffsetX/OffsetY	Permet d'ajouter un décalage en pixel du Panel par rapport à la TextBox liée (à *0* par défaut pour les deux propriétés).	NON

Exemple d'utilisation :

Dans cet exemple nous donner la possibilité de choisir différents OS à la sélection de la TextBox.

ASP.NET

```
Sous quelle OS tournez vous ?
<asp:TextBox ID="TxtBxOS" runat="server" Width="350px"></asp:TextBox>
<br />
<asp:Panel ID="PnlOS" runat="server" CssClass="popupControl">
    <asp:UpdatePanel ID="UpdtPnlOS" runat="server">
        <ContentTemplate>
            <asp:RadioButtonList ID="RdBtnLstOS" runat="server"
                AutoPostBack="true"
                OnSelectedIndexChanged="RdBtnLstOS_SelectedIndexChanged"
                Width="150px">
                <asp:ListItem Text="XP Home"></asp:ListItem>
                <asp:ListItem Text="XP Pro"></asp:ListItem>
                <asp:ListItem Text="Vista Home"></asp:ListItem>
                <asp:ListItem Text="Vista Business"></asp:ListItem>
                <asp:ListItem Text="Vista Ultimate"></asp:ListItem>
            </asp:RadioButtonList>
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Panel>
<br />
<ajaxtoolkit:PopupControlExtender ID="PCExOS" runat="server"
    TargetControlID="TxtBxOS"
    PopupControlID="PnlOS"
    CommitProperty="value"
    CommitScript="e.value += ' est votre système d\'exploitation !';"
    Position="Bottom" OffsetX="10" OffsetY="10">
</ajaxtoolkit:PopupControlExtender>
```

Code Behind :

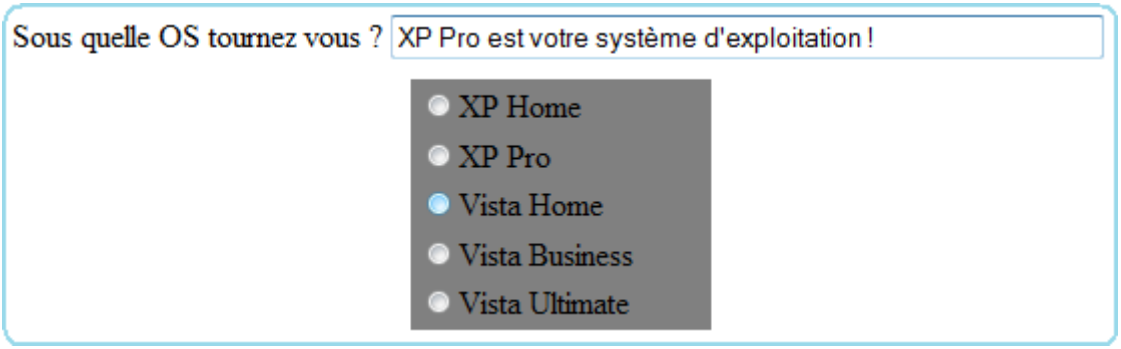
Vous devez impérativement ajouter l'espace de nomage `AjaxControlToolkit`.

C#

```
protected void RdBtnLstOS_SelectedIndexChanged(object sender, EventArgs e)
{
    if ((String.IsNullOrEmpty(RdBtnLstOS.SelectedValue)) == false)
    {
        PopupControlExtender.GetProxyForCurrentPopup(this.Page).Commit(RdBtnLstOS.
        SelectedValue);
    }
    RdBtnLstOS.ClearSelection();
}
```

```
VB.NET

Protected Sub RdBtnLstOS _SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles RdBtnLstOS.SelectedIndexChanged

    If Not (String.IsNullOrEmpty(RdBtnLstOS.SelectedValue)) Then
         PopupControlExtender.GetProxyForCurrentPopup(Me.Page).Commit(RdBtnLstOS.Se
         lectedValue)
    End If
    RdBtnLstOS.ClearSelection()

End Sub
```

Voici un aperçu de cet exemple :



Vous remarquerez qu'une fois votre OS sélectionné, le Panel disparait.

3.22 ResizableControl

Définition :

Le ResizableControlExtender va nous permettre de redimensionner n'importe quel contrôle contenue dans un Panel.

Propriétés :

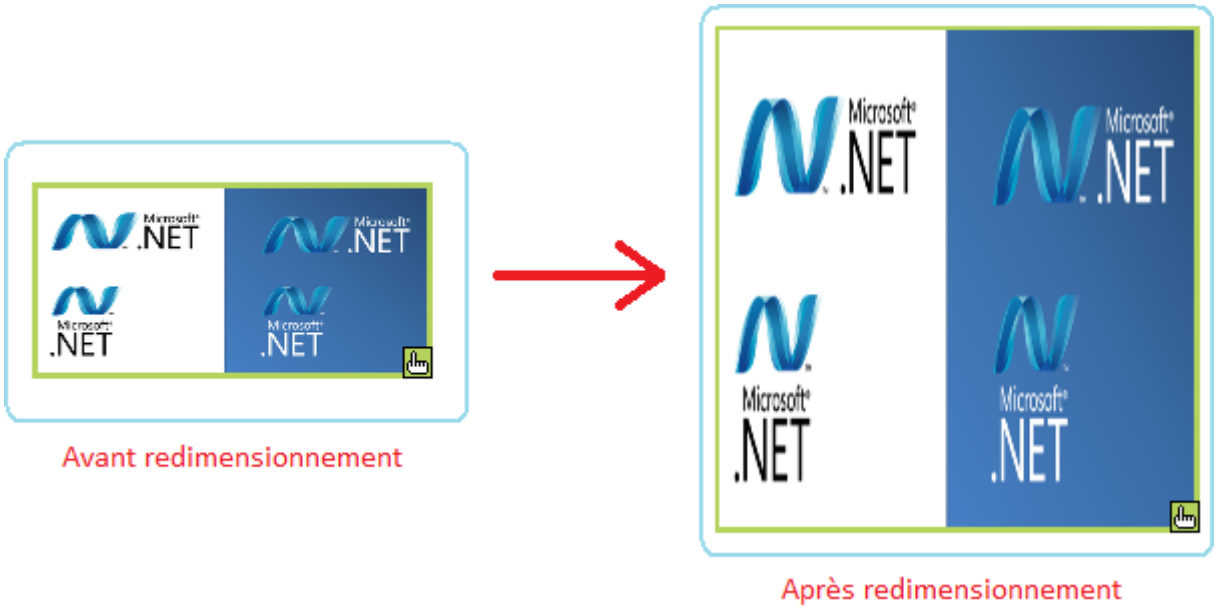
Nom	Description	Obligatoire
TargetControlID	Permet de lier le ResizableControlExtender à un Panel qui bénéficiera des fonctionnalités de ce contrôle.	OUI
MinimumHeight **MinimumWidth**	Ces propriétés permettent de définir la valeur minimum de la hauteur et de la largeur.	OUI
MaximumHeight **MaximumWidth**	Ces propriétés permettent de définir la valeur maximum de la hauteur et de la largeur.	OUI
ResizableCssClass	Permet de lier le panel à une classe CSS, utilisée lorsqu'on est sur le bouton pour redimensionner le contrôle.	NON
HandleCssClass	Permet de lier le panel à une classe CSS qui sera appliquée sur le bouton de redimensionnement.	NON
HandleOffsetX **HandleOffsetY**	Permet de définir le décalage du bouton de redimensionnement par rapport au coin inférieur droit du panel.	NON

Exemple d'utilisation :

Dans cet exemple nous afficherons une image qui pourra être redimensionnée.

```
ASP.NET
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<div>
    <asp:Panel ID="PanelImage" runat="server" CssClass="frameImage">
        <asp:Image ID="Image1" runat="server"
            ImageUrl="~/images/Dotnet.png" AlternateText="Dotnet-France"
            Style="width: 100%; height: 100%;" />
    </asp:Panel>
    <ajaxtoolkit:ResizableControlExtender ID="ResizableControlExtender1"
        runat="server" HandleCssClass="handleImage"
        TargetControlID="PanelImage" ResizableCssClass="resizingImage"
        MinimumWidth="50" MinimumHeight="50"
        MaximumWidth="250" MaximumHeight="250"
        HandleOffsetX="3" HandleOffsetY="3"/>
</div>
</form>
```

Voici un aperçu de cet exemple :



3.23 RoundedCorners

Définition :

Ce contrôle permet d'effectuer un effet arrondi sur les coins des Panel.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le RoundedCorners à un Panel qui bénéficiera des fonctionnalités de ce contrôle.	OUI
Radius	Permet de déterminer sur combien de pixel vous effectuez un arrondissement du coin (à *5* par défaut).	NON
Corners	Permet de choisir quels coins vous voulez arrondir (tous par défaut).	NON

Exemple d'utilisation :

ASP.NET

```
<asp:Panel ID="Panel1" runat="server"
    Style="background-color: Gray;" Width="314px"
    Height="125px">
    <div style="padding-top: 10px; padding-left: 10px">
        <asp:Image ID="Image1" runat="server"
            ImageUrl="~/images/DotNet.png" />
    </div>
</asp:Panel>
<ajaxtoolkit:RoundedCornersExtender ID="RoundedCornersExtender1" runat="server"
    TargetControlID="Panel1">
</ajaxtoolkit:RoundedCornersExtender>
```

Voici un aperçu de cet exemple :

3.24 Slider

Définition :

C'est un extender pour les TextBox qui va les transformer en slider. Il possède des propriétés permettant de configurer le Slider comme les valeurs minimum et maximum, le pas, la valeur par défaut ect.

Propriétés :

Nom	Description	Obligatoire
Minimum	Valeur minimum du slider. Par défaut à 0.	NON
Maximum	Valeur maximum du slider. Par défaut à 100.	NON
Decimals	Nombre de chiffres décimaux après la virgule.	NON
Steps	Défini en combien de valeur va être coupé le Slider. Par exemple de -200 à 200 si on met 5, il y aura les valeurs : -200, -100, 0, 100 et 200, donc 5 valeurs.	NON
RailCssClass	Définit la classe CSS utilisé pour le rail du Slider.	NON
HandleImageURL	Image qui sera utilisée pour le slider.	NON
Length	Définit la taille du Slider (Width/Height).	NON
BoundControlID	Prend comme valeur l'ID du Label ou de la TextBox qui prendra automatiquement la valeur du Slider.	NON
TargetControlID	Prend comme valeur l'ID de la TextBox qui va servir de Slider.	OUI
Orientation	Définit le sens d'affichage du slider : Horizontal, Vertical. Par défaut il est Horizontal.	NON

Exemple d'utilisation :

ASP.NET

```
<asp:TextBox ID="Slider" runat="server" />

<asp:Label ID="LabelSlider" runat="server" Text="Texte non affiché" />

<cc1:SliderExtender
        runat="server"
        TargetControlID="Slider"
        BoundControlID="LabelSlider"
        Minimum="-200"
        Maximum="200"
        Steps="5"
        Decimals="2"
        Orientation="Vertical" />
```

Cet exemple vous montre simplement comment faire un Slider à partir d'une TextBox. Vous pouvez vérifier qu'il y a bien 5 pas sur le Slider.

3.25 SlideShow

Définition :

Le SliderShow va permettre de créer un diaporama d'images via un WebService. Pour mieux comprendre allez voir l'exemple.

Propriété :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le SliderShow à une Image qui bénéficiera des fonctionnalités de ce contrôle.	OUI
SlideShowServicePath	Permet de spécifier le chemin de notre WebService.	OUI
SlideShowServiceMethod	Permet de spécifier le nom de la méthode à appeler dans le WebService.	OUI
NextButtonID	Permet de spécifier l'ID du Button qui servira pour passer à l'image suivante.	OUI
PreviousButtonID	Permet de spécifier l'ID du Button qui servira pour passer à l'image précédente.	OUI
PlayButtonID	Permet de spécifier l'ID du Button qui servira à démarrer/stopper le diaporama.	OUI
PlayInterval	Permet de définir l'intervalle de temps entre chaque image.	NON
Loop	Permet de définir si le diaporama doit tourner un boucle ou non.	NON
AutoPlay	Permet de définir si le diaporama se lance au chargement de la page ou non.	NON
ImageTitleLabelID	Permet de spécifier le Label cible pour afficher le titre de l'image.	NON
ImageDescritpionLabelID	Permet de spécifier le Label cible pour afficher la description de l'image.	NON

Exemple d'utilisation :

Dans cet exemple, nous allons créer un diaporama via un WebService qui récupèrera nos images de manière prédéfinie. Il est bien entendu possible de faire cela via une base de données.

Code source de SlideShowService.asmx :

C#
```
public class SlideShowService : System.Web.Services.WebService
{
    [System.Web.Services.WebMethod]
    [System.Web.Script.Services.ScriptMethod]
    public AjaxControlToolkit.Slide[] GetSlide()
    {
        AjaxControlToolkit.Slide[] MySlide = new AjaxControlToolkit.Slide[4];
        MySlide[0] = new AjaxControlToolkit.Slide("images/Freedom.jpg", "Freedom",
                     "Let's be free !");
        MySlide[1] = new AjaxControlToolkit.Slide("images/Lemon.jpg", "Lemon",
                     "Wanna have some fruits ?");
        MySlide[2] = new AjaxControlToolkit.Slide("images/Load.jpg", "Load", "You
                     still wait a moment please");
        MySlide[3] = new AjaxControlToolkit.Slide("images/Love.jpg", "Love",
                     "You've seen my heart ?");
        return MySlide;
    }
}
```

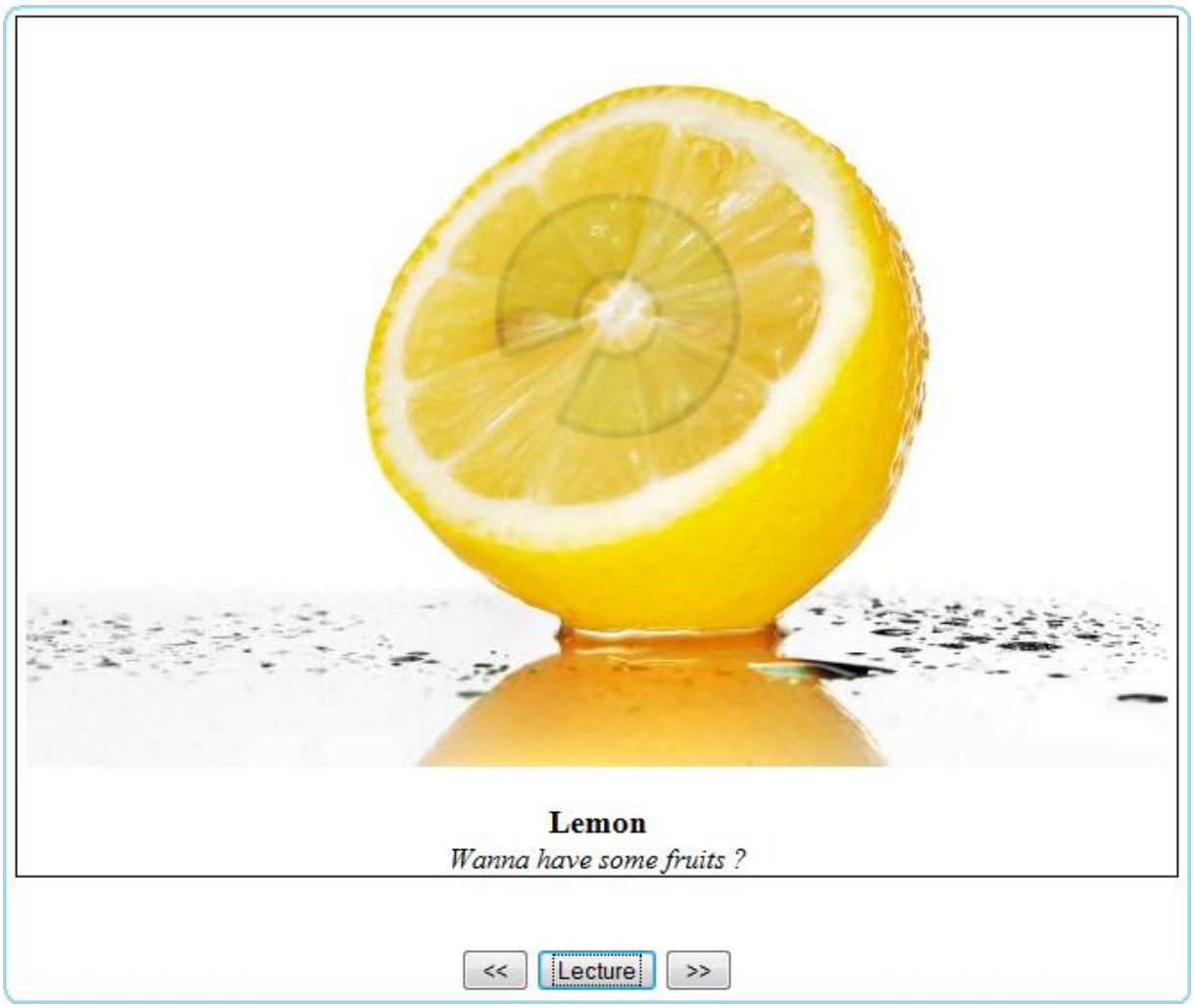
VB.NET
```
' A insérer avant la délaration de la classe
```

```
<System.Web.Script.Services.ScriptService()> _
' [...]
Public Function GetSlides() As AjaxControlToolkit.Slide()
    Dim MySlides(4) As AjaxControlToolkit.Slide
    MySlides(0) = New AjaxControlToolkit.Slide("images/Freedom.jpg", "Freedom",
                  "Let's be free !")
    MySlides(1) = New AjaxControlToolkit.Slide("images/Lemon.jpg", "Lemon", "Wanna
                  have some fruits ?")
    MySlides(2) = New AjaxControlToolkit.Slide("images/Load.jpg", "Load", "You
                  still wait a moment please")
    MySlides(3) = New AjaxControlToolkit.Slide("images/Love.jpg", "Love", "You've
                  seen my heart ?")
    Return MySlides
End Function
```

Code Source de la page par défaut :

```
ASP.NET
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<div style="border:solid 1px black;text-align:center">
    <asp:Image ID="ImageSlideShow" runat="server"
        Style="width:auto; padding-top:20px;" Height="400" />
    <br />
    <asp:Label ID="LabelTitle" runat="server" Font-Bold="true" Font-
              Size="Larger"></asp:Label>
    <br />
    <asp:Label ID="LabelDescription" runat="server" Font-Italic="true"></asp:Label>
</div>
<br /><br />
<div style="text-align:center";>
    <asp:Button ID="ButtonPrevious" runat="server" Text="<<" />
    <asp:Button ID="ButtonPlay" runat="server" Text="Lecture" />
    <asp:Button ID="ButtonNext" runat="server" Text=">>" />
    <ajaxtoolkit:SlideShowExtender ID="SlideShowExtender1" runat="server"
        TargetControlID="ImageSlideShow"
        SlideShowServicePath="SlideShowService.asmx"
        SlideShowServiceMethod="GetSlide"
        ImageTitleLabelID="LabelTitle"
        ImageDescriptionLabelID="LabelDescription"
        PreviousButtonID="ButtonPrevious"
        NextButtonID="ButtonNext"
        PlayButtonID="ButtonPlay"
        PlayButtonText="Lecture"
        StopButtonText="Pause"
        Loop="true"
        AutoPlay="true"/>
  </div>
</form>
```

Voici un aperçu de cet exemple d'utilisation :



3.26 TextBoxWatermark

Définition :

Le contrôle TextBoxWatermark va vous permettre d'afficher un texte par défaut dans une TextBox qui disparaitra à la sélection de celle-ci.

Propriété :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le TextBoxWatermark à une TextBox qui bénéficiera des fonctionnalités de ce contrôle.	OUI
WatermarkText	Permet de définir le texte affiché par défaut.	OUI
WatermarkCssClass	Permet d'appliquer un style à la TextBox quand le texte par défaut est affiché.	NON

Exemple d'utilisation :

```
ASP.NET

<table>
    <tr>
        <td>Nom : </td>
        <td>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <ajaxtoolkit:TextBoxWatermarkExtender ID="TextBoxWatermarkExtender1"
                runat="server"
                TargetControlID="TextBox1"
                WatermarkText="Entrez votre nom ici"
                WatermarkCssClass="watermarked">
            </ajaxtoolkit:TextBoxWatermarkExtender>
        </td>
    </tr>
    <tr>
        <td>Prénom : </td>
        <td>
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <ajaxtoolkit:TextBoxWatermarkExtender ID="TextBoxWatermarkExtender2"
                runat="server"
                TargetControlID="TextBox2"
                WatermarkText="Entrez votre prénom ici"
                WatermarkCssClass="watermarked">
            </ajaxtoolkit:TextBoxWatermarkExtender>
        </td>
    </tr>
</table>
```

Voici un aperçu de cet exemple :

Le texte s'efface quand vous sélectionnez une TextBox puis réapparait si vous la laissez vide.

3.27 ToggleButton

Définition :

Le contrôle ToggleButton va nous permettre d'appliquer un style pour nos CheckBox avec des images personnalisées sans perturber le fonctionnement et l'utilisation des CheckBox.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le ToggleButton à une CheckBox qui bénéficiera des fonctionnalités de ce contrôle.	OUI
ImageHeight **ImageWidth**	Permet de définir la hauteur/largeur, de nos images qui s'afficheront.	OUI
CheckedImageUrl **UncheckedImageUrl**	Permet de définir l'URL de l'image associée à la CheckBox lorsqu'elle est coché/décoché.	OUI
DisabledCheckedImageUrl **DisabledUncheckedImageUrl**	Permet de définir l'URL de l'image associée à la CheckBox lorsque celle-ci est désactivée et cochée/décochée.	NON
CheckedImageOverUrl	Permet de définir l'URL de l'image associée à	NON

UncheckedImageOverUrl	la CheckBox au survol de la souris.	
CheckedImageAlternateText **UncheckedImageAlternateText**	Permet de définir un texte alternatif à la CheckBox quant celle-ci est cochée/décochée.	NON
CheckedImageOverAlternateText **UncheckedImageOverAlternateText**	Permet de définir le texte alternatif au survol de la sourie de la CheckBox.	NON

Exemple d'utilisation :

Dans cet exemple nous allons créer deux TextBox et les lier à notre contrôle Ajax. Puis nous récupèrerons les informations entrées par l'utilisateur.

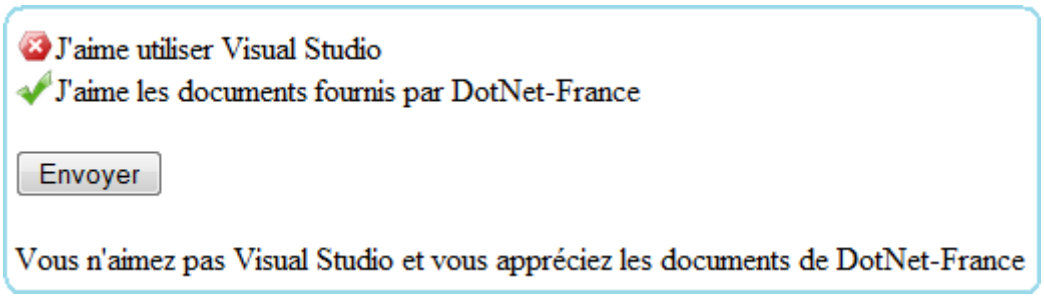
ASP.NET

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:CheckBox ID="CheckBox1" Checked="true" Text="J'aime utiliser Visual
            Studio" runat="server" />
        <br />
        <asp:CheckBox ID="CheckBox2" Checked="true" Text="J'aime les documents
            fournis par DotNet-France" runat="server" />
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" Text="Envoyer"
            OnClick="Button1_Click" />
        <br />
        <br />
        <asp:Label ID="Label1" runat="server"></asp:Label>
        <ajaxtoolkit:ToggleButtonExtender ID="ToggleButtonExtender1" runat="server"
            TargetControlID="CheckBox1"
            ImageWidth="19" ImageHeight="19"
            UncheckedImageUrl="images/Toggle_refuse.png"
            CheckedImageUrl="images/Toggle_valide.png"
            CheckedImageAlternateText="Coché"
            UncheckedImageAlternateText="Décoché">
        </ajaxtoolkit:ToggleButtonExtender>
        <ajaxtoolkit:ToggleButtonExtender ID="ToggleButtonExtender2" runat="server"
            TargetControlID="CheckBox2"
            ImageWidth="19" ImageHeight="19"
            UncheckedImageUrl="images/Toggle_refuse.png"
            CheckedImageUrl="images/Toggle_valide.png"
            CheckedImageAlternateText="Coché"
            UncheckedImageAlternateText="Décoché">
        </ajaxtoolkit:ToggleButtonExtender>
    </ContentTemplate>
</asp:UpdatePanel>
```

```
C#

protected void Button1_Click(object sender, EventArgs e)
{
    string clientSelection = "";
    if (CheckBox1.Checked)
    {
        clientSelection = "Vous aimez Visual Studio ";
    }
    else
    {
        clientSelection = "Vous n'aimez pas Visual Studio ";
    }
    if (CheckBox2.Checked)
    {
        clientSelection += "et vous appréciez les documents de DotNet-France";
    }
    else
    {
        clientSelection += "et vous n'appréciez guerre les documents de DotNet-
        France";
    }
    Label1.Text = clientSelection;
}
```

```
VB.NET

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim clientSelection As String = ""
    If (CheckBox1.Checked) Then
        clientSelection = "Vous aimez Visual Studio "
    Else
        clientSelection = "Vous n'aimez pas Visual Studio "
    End If
    If (CheckBox2.Checked) Then
        clientSelection += "et vous appréciez les documents de DotNet-France"
    Else
        clientSelection += "et vous n'appréciez guerre les documents de DotNet-
        France"
    End If
    Label1.Text = clientSelection
End Sub
```

Aperçu de cet exemple :



Comme vous pouvez le constater, la CheckBox habituelle a été remplacée par nos images.

3.28 UpdatePanelAnimation

Définition :

L'UpdatePanelAnimation est un extender de l'UpdatePanel. Ce contrôle va nous permettre de créer une animation lors d'une modification d'un UpdatePanel. Il est possible d'appliquer trois effets différents. Voici les utilisations les plus courantes :

- Réduction d'opacité : il est possible de diminuer l'opacité de l'UpdatePanel pendant le chargement et de lui faire reprendre sa valeur initiale une fois le chargement terminé.
- Animation d'affaissement : l'UpdatePanel se réduit puis retrouve sa taille initiale.
- Changement de la couleur de l'arrière plan : la couleur de l'arrière plan change de manière fluide puis retrouve sa couleur de départ.

Toutes ces animations sont cumulables et complètement personnalisables, par exemple, on peut mettre à l'arrivée une couleur différente de celle de départ pour l'arrière plan et coupler cette animation avec celle de l'affaissement.

Les propriétés sont peu nombreuses en revanche la mise en place des effets est plus longue.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Permet de lier l'AnimationUpdatePanel à un UpdatePanel qui bénéficiera des fonctionnalités de ce contrôle.	OUI
BehaviorID	Permet de créer un identifiant pour l'UpdatePanel auquel l'UpdatePanelAnimation est lié afin pouvoir récupérer ses propriétés dans le JavaScript.	NON

Balises :

Nom	Description	Obligatoire
Animation	Permet d'indiquer que l'on va créer des effets d'animation. Elle est contenue dans UpdatePanelAnimation et contient les balises *OnUpdating* et *OnUpdated*.	Recommandé
OnUpdating	C'est dans cette balise que l'on crée les effets d'animation pendant le chargement.	Recommandé
OnUpdated	C'est dans cette balise que l'on crée les effets d'animation après le chargement.	Recommandé

Exemple d'utilisation :

Dans cet exemple nous allons créer un *UpdatePanel* contenant une *Textbox* et un *Label*. Nous allons donner la possibilité à l'utilisateur de choisir l'animation qu'il veut voir.

ASP.NET
```
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<div class="firstDiv">
    <div class="secondDiv">
```

```
        <div id="up_container" class="thirdDiv">
            <%-- Panel touché par l'animation --%>
            <asp:UpdatePanel ID="update" runat="server">
                <ContentTemplate>
                    <div id="background" class="divInContentTemplate">
                        Votre nom :
                        <asp:TextBox ID="txtbxUpdate" runat="server"></asp:TextBox>
                        <br />
                        <asp:Label ID="lblUpdate" runat="server"
                        CssClass="labelInContentTemplate">
                                Nous ne connaissons pas votre nom.
                        </asp:Label>
                    </div>
                </ContentTemplate>
                <%-- Le controle AsyncPostTrigger va permettre d'effectuer une mise
                     à jour sans recharger la page entièrement --%>
                <Triggers>
                    <asp:AsyncPostBackTrigger ControlID="btnUpdate"
                    EventName="Click" />
                </Triggers>
            </asp:UpdatePanel>
        </div>
    </div>
</div>
<!-- Les controles suivants sont en HTML afin de leur permettre d'intéragir avec
     le controle AjaxToolkit -->
Choisissez les effets de l'animation et cliquez sur "Actualiser" :<br />
<input type="checkbox" id="effect_fade" checked="checked" />
<label for="effect_fade">Réduction d'opacité</label><br />
<input type="checkbox" id="effect_collapse" checked="checked" />
<label for="effect_collapse">Animation d'affaissement</label><br />
<input type="checkbox" id="effect_color" checked="checked" />
<label for="effect_color">Couleur de l'arrière plan</label><br />
<asp:Button ID="btnUpdate" runat="server" Text="Actualiser"
            OnClick="btnUpdate_Click" />
<ajaxtoolkit:UpdatePanelAnimationExtender ID="upae" BehaviorID="animation"
            runat="server" TargetControlID="update">
    <Animations>
            <OnUpdating>
                <Sequence>
                    <%-- On stocke les propriétés de l'UpdatePanel
                         que nous comptons utiliser --%>
                    <ScriptAction Script="var b = $find('animation');
                                          b._originalHeight =
                                          b._element.offsetHeight;" />

                    <%-- La balise Parallele permet de définir les actions à
                         réaliser en même temps --%>
                        <%-- La propriété duration permet de spécifier
                             en combien de temps l'action doit être réalisé --%>
                    <Parallel duration="0">
                    <%-- On désactive les controles pendant l'actualisation --%>
                        <EnableAction AnimationTarget="btnUpdate" Enabled="false"
                        />
                        <EnableAction AnimationTarget="effect_color"
                        Enabled="false" />
                        <EnableAction AnimationTarget="effect_collapse"
                        Enabled="false" />
                        <EnableAction AnimationTarget="effect_fade" Enabled="false"
                        />
                    </Parallel>
                    <StyleAction Attribute="overflow" Value="hidden" />

                    <%-- Animation à réaliser pendant le chargement en fonction des
                         effets choisis --%>
                        <%-- La propriété FPS spécifie le nombre d'image par
                             seconde à afficher --%>
                    <Parallel duration=".25" Fps="30">
                        <%-- L'animation se réalisera si la valeur
                             de retour de ConditionScript est True --%>
                        <Condition ConditionScript="$get('effect_fade').checked">
```

```
                        <FadeOut AnimationTarget="up_container"
                        minimumOpacity=".2" />
                    </Condition>
                    <Condition
                    ConditionScript="$get('effect_collapse').checked">
                        <Resize Height="0" />
                    </Condition>
                    <Condition ConditionScript="$get('effect_color').checked">
                        <Color AnimationTarget="up_container"
                         PropertyKey="backgroundColor" EndValue="#FF0000"
                         StartValue="#40669A" />
                    </Condition>
                </Parallel>
            </Sequence>
        </OnUpdating>
        <OnUpdated>
            <Sequence>
                <%-- Animation à réaliser après le chargement en fonction des
                     effets choisis --%>
                <Parallel duration=".25" Fps="30">
                    <Condition ConditionScript="$get('effect_fade').checked">
                        <FadeIn AnimationTarget="up_container"
                        minimumOpacity=".2" />
                    </Condition>
                    <Condition
                    ConditionScript="$get('effect_collapse').checked">
                        <%-- On récupère les valeurs stockées dans les
                             variables auparavant --%>
                        <Resize
                        HeightScript="$find('animation')._originalHeight" />
                    </Condition>
                    <Condition ConditionScript="$get('effect_color').checked">
                        <Color AnimationTarget="up_container"
                        PropertyKey="backgroundColor" StartValue="#FF0000"
                        EndValue="#40669A" />
                    </Condition>
                </Parallel>

                <%-- On réactive les controles après l'actualisation --%>
                <Parallel duration="0">
                    <EnableAction AnimationTarget="effect_fade" Enabled="true"
                    />
                    <EnableAction AnimationTarget="effect_collapse"
                    Enabled="true" />
                    <EnableAction AnimationTarget="effect_color" Enabled="true"
                    />
                    <EnableAction AnimationTarget="btnUpdate" Enabled="true" />
                </Parallel>
            </Sequence>
        </OnUpdated>
    </Animations>
</ajaxtoolkit:UpdatePanelAnimationExtender>
</form>
```

CSS - Feuille de style à lier

```
.firstDiv
{
      margin-bottom: 10px;
}
.secondDiv
{
      border: dashed 1px #222222;
}
.thirdDiv
{
      background-color: #40669A;
}
.divInContentTemplate
```

```
{
        text-align: center;
        vertical-align: middle;
        line-height: 44px;
    padding: 12px;
    height: 80px;
    color: #FFFFFF;
}
.labelInContentTemplate
{
        padding: 5px;
        font-size: 14px;
        font-weight: bold;
}
```

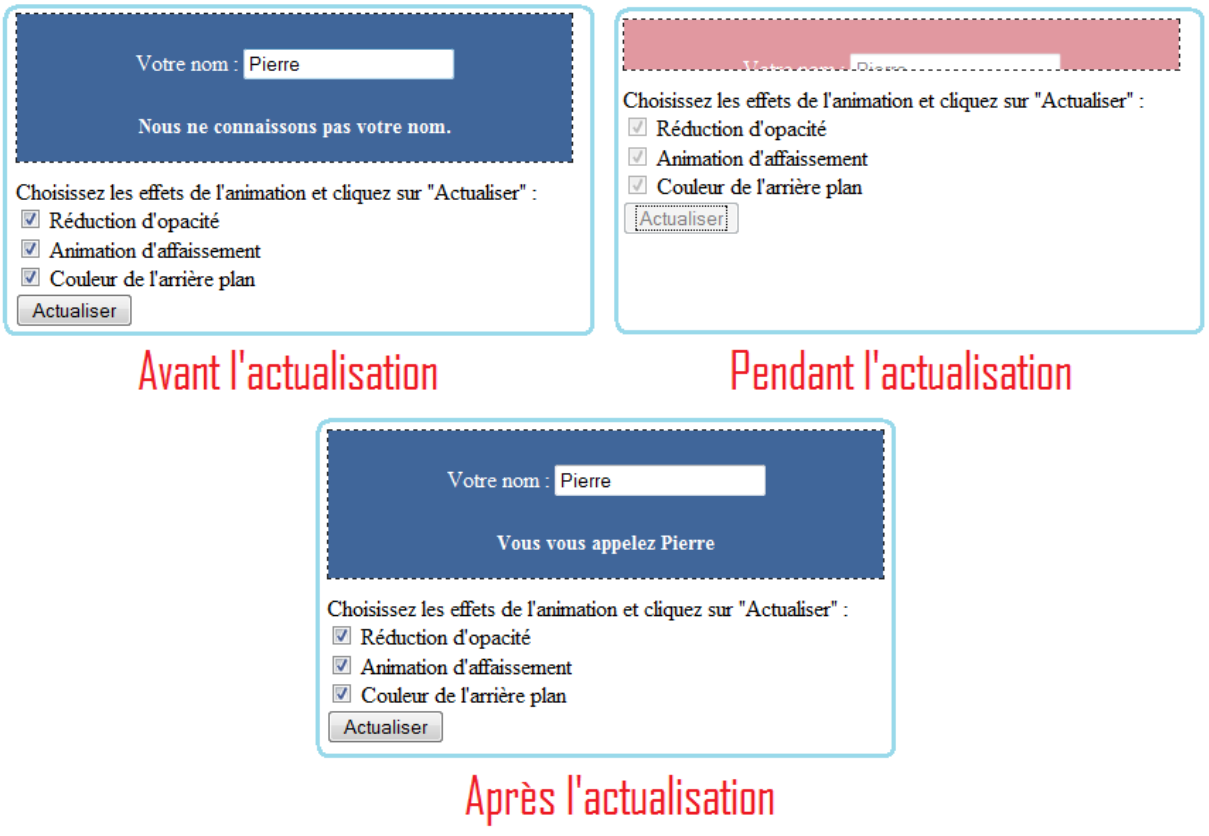
C#

```
protected void btnUpdate_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(300);
    lblUpdate.Text = "Vous vous appelez " + txtbxUpdate.Text;
}
```

VB.NET

```
Protected Sub btnUpdate_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnUpdate.Click
    System.Threading.Thread.Sleep(300)
    lblUpdate.Text = DateTime.Now.ToString()
End Sub
```

Aperçu de cet exemple :



3.29 ValidatorCallout

Définition :

Ce contrôle permet d'afficher une étiquette esthétique à la place de du texte d'erreur classique affichée par les contrôles de validation (Ex : RequireFieldValidator), ainsi que de surligner le champ ne remplissant pas les conditions de validation.

Propriétés :

Nom	Description	Obligatoire
TargetControlID	Permet de lier le ValidatorCalloutExtender à un contrôle de validation qui bénéficiera des fonctionnalités du contrôle AjaxToolkit.	OUI
HightlightCssClass	Lie votre contrôle à une classe CSS où vous pourrez configurer la couleur du surlignement.	NON

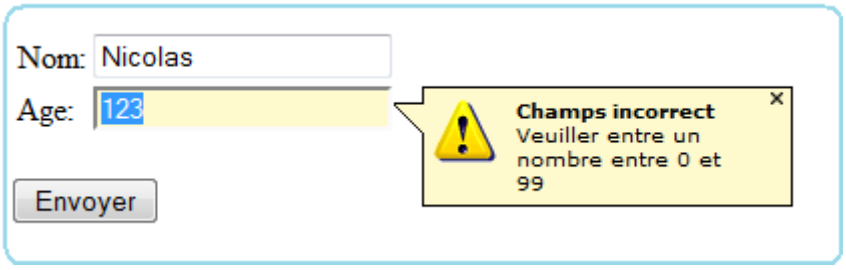
Exemple d'utilisation :

Dans cet exemple, nous afficherons deux Textbox nécessitant une validation.

```
ASP.NET
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<table>
    <tr>
        <td>Nom:</td>
        <td>
            <asp:TextBox runat="server" ID="TxtBxName" />
        </td>
    </tr>
    <tr>
        <td>Age:</td>
        <td>
            <asp:TextBox runat="server" ID="TxtBxAge" />
        </td>
    </tr>
</table>
<br />
<asp:RequiredFieldValidator runat="server"
    ID="RFVNameMissing" ControlToValidate="TxtBxName"
    Display="None"
    ErrorMessage="<b>Texte manquant</b><br />Veuillez entrer votre nom." />
<ajaxtoolkit:ValidatorCalloutExtender runat="Server"
    ID="VCENameMissing" TargetControlID="RFVNameMissing"
    HighlightCssClass="validatorCalloutHighlight" />

<asp:RequiredFieldValidator runat="server"
    ID="RFVAgeMissing" ControlToValidate="TxtBxAge"
    Display="None"
    ErrorMessage="<b>Texte manquant</b><br />Veuillez entrer votre age." />
<ajaxtoolkit:ValidatorCalloutExtender runat="Server"
    ID="VCEAgeMiss" TargetControlID="RFVAgeMissing"
    HighlightCssClass="validatorCalloutHighlight"
    Width="350px" />
<asp:RegularExpressionValidator runat="server"
    ID="REVAgeType" ControlToValidate="TxtBxAge"
    Display="None"
    ValidationExpression="^(\d{2})|(\d{1})"
    ErrorMessage="<b>Champ incorrect</b>
                <br />Veuiller entre un nombre entre 0 et 99" />
<ajaxtoolkit:ValidatorCalloutExtender runat="Server"
    ID="VCEAgeType" TargetControlID="REVAgeType"
    HighlightCssClass="validatorCalloutHighlight" />
```

```
<asp:Button ID="Button1" runat="server" Text="Envoyer" /><br />
<br />
</form>
```

Aperçu de cet exemple :



4 Les contrôles autonomes

Les contrôles autonomes sont des contrôles à part entière, qui n'ajoutent pas de fonctionnalités à d'autres contrôles, mais qui sont en eux même des contrôles « complets ».

4.1 Accordion

Définition :

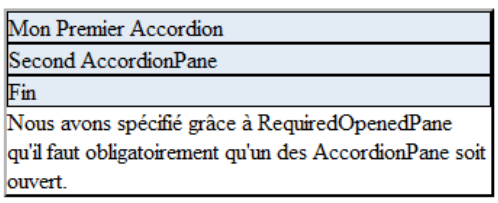
Le contrôle Accordion vous permet l'affichage de plusieurs Panel sous la forme d'un menu. Chaque Panel possède un titre. Lors d'un clic sur celui-ci, soit on déroule, soit on cache le contenu du Panel. Ce contrôle contient une liste de contrôles Panel, que l'on place dans un contrôle Panes de l'Accordion. Un Panel est implémenté par des contrôles AccordionPane contenant eux-mêmes un titre (Header) et un contenu (Content). C'est AccordionPane se trouvent dans la collection de Panels nommé *Panes.*

Pour mieux comprendre de quoi il retourne veuillez vous référer à l'exemple de ce contrôle.

Propriétés :

Nom	Description	Obligatoire
AutoSize	Défini le mode de redimensionnement. Peut prendre comme valeur : None, Limit ou encore Fill.	NON
DataSourceId	Prend en paramètre l'ID de l'objet qui représente la source de données.	NON
FadeTransitions	Active ou non les effets de transition. A False, les effets standards sont utilisés.	NON
RequireOpenedPane	Détermine si on force l'ouverture d'un seul Panel. Lors de l'ouverture d'un nouveau Panel, celui qui était déjà ouvert se ferme. Sa valeur par défaut est True.	NON
SelectedIndex	Elle permet de définir quel sera le Panel ouvert au chargement de la page. Par défaut elle possède la valeur 0 qui correspond au premier Panel.	NON
TransitionDuration	Durée de la transition en millisecondes.	NON
SuppressHeaderPostbacks	Supprime les éventuels Postback possible dans les en têtes. Pour l'accessibilité on peut mettre un Hyperlink dans l'en tête. Grâce à cette propriété on peut désactiver le lien (ce qui n'aura aucun effet sur l'accessibilité puisque le lien restera présent mais sera inactif).	NON
Panes	Contient la collection d'AccordionPane. Ce sont les panels qui seront affiché.	OUI
DataSource	Défini la source de donnée à utiliser. Attention il faudra appeler la méthode DataBind pour remplir le contrôle.	NON
DataMember	Lorsque l'on utilise DataSourceID, on a besoin de définir le nom du membre à utiliser.	NON

Exemple d'utilisation :



Dans cet exemple on va voir comment utiliser le contrôle Accordion. L'image ci-contre montre le résultat. Comme vous pouvez le constatez dans le code qui suit, le contrôle Accordion contient une collection d'AccrodionPane qui est stocké dans Panes. Chaque AccordionPane possède un

Header qui correspond à son titre ou son en-tête et un contenu (Content).

ASP.NET

```
<style type="text/css">
    .HeadClass
    {
          border: solid black 1px;
          background-color: #e3ebf4;
    }

    .ThisClass
    {
          border: outset black 2px;
    }
</style>

<div>
  <asp:ScriptManager ID="ScriptManager1" runat="server" />

  <cc1:Accordion ID="Accordion1" runat="server" SelectedIndex="0"
      FadeTransitions="true" HeaderCssClass="HeadClass"
      ContentCssClass="ContentClass" CssClass="ThisClass"
     AutoSize="None" RequiredOpenedPane="true">
       <Panes>
          <cc1:AccordionPane runat="server">
             <Header>Mon Premier Accordion</Header>
             <Content>Le contenu des AccordionPane est mis en Autosize="None".
                        Parcourez les différentes parties du contrôle Accordion
                        pour en découvrir plus.</Content>
          </cc1:AccordionPane>
          <cc1:AccordionPane runat="server">
              <Header>Second AccordionPane</Header>
              <Content>On peut mettre du code dans l'Accordion.
                      <asp:Label runat="server" Text="Ma TextBox"/><asp:TextBox
                           runat="server"/></Content>
          </cc1:AccordionPane>
          <cc1:AccordionPane runat="server">
              <Header>Fin</Header>
              <Content>Nous avons spécifié grâce à RequiredOpenedPane qu'il faut
                      obligatoirement qu'un des AccordionPane soit
                      ouvert.</Content>
          </cc1:AccordionPane>
       </Panes>
    </cc1:Accordion>
```

4.2 NoBot

Définition :

NoBot est un contrôle qui va vous permettre de savoir si la personne qui accède à votre site est un bot de la même façon qu'un Captcha pourrait le faire. Son avantage est qu'il ne demande aucune action de l'utilisateur : il est basé sur différentes hypothèses émises pour différencier un humain d'un bot que nous allons voir. L'avantage de ce système est qu'il est totalement invisible pour l'utilisateur. En revanche il n'est pas efficace à 100%, et il est préférable de ne l'utiliser que pour des sites à petit trafic comme des blog, des sites de petites entreprises … Ce genre de système est généralement utilisé pour éviter qu'un bot ne puisse s'inscrire à un site, flooder/spammer le site en postant des commentaires/articles ou autres … Donc pour empêcher un bot de nuire au site.

Pour cela, ce contrôle ce base sur plusieurs techniques dont en voila quelques-unes :

- La rapidité d'un humain à remplir un formulaire et à l'envoyer. En effet on peut configurer un temps correspondant au temps minimum que mettrait un humain à remplir le formulaire en question. Ce temps est le temps entre la requête du formulaire faite au serveur et le demande de PostBack pour envoyer les données. Un bot mettra surement moins d'une seconde pour remplir un formulaire s'il n'est pas programmé pour faire des « pauses ».

- Le nombre de fois où le formulaire est envoyé par minute. Un humain ne peut pas renvoyer le même formulaire plus d'un certain nombre de fois par minutes (par convention on prend 5).

Il permet aussi de vérifier si le code javascript est bien activé, pour éviter par exemple que l'on passe outre les vérifications faites en javascript sur le formulaire avant son envoi. Cependant nous tenons à insister sur le fait que, comme tout ce qui se trouve sur le client, ce n'est pas sur à 100%. Il faut toujours vérifier les données reçus (on peut quand même passer outre le code javascript). Quand au contrôle lui-même, il n'est pas non plus infaillible.

Propriétés du contrôle :

Nom	Description	Obligatoire
OnGenerateChallengeAndResponse	Prend en valeur le nom de la méthode permettant de vérifier que le javascript est bien activé sur le client. Le challenge correspond à quelque chose que le client va devoir calculer et à la suite de cela il retournera le résultat qui correspond à Response.	NON
ResponseMinimumDelaySeconds	Délai entre la requête pour récupérer le formulaire et le PostBack du formulaire.	NON
CutoffWindowSeconds	Défini un laps de temps durant lequel une adresse IP ne peut pas renvoyer le formulaire.	NON
CutoffMaximumInstances	Défini le nombre de fois maximum pour l'envoi du formulaire par rapport à une adresse IP.	NON

Exemple d'utilisation :

Cet exemple ne contient pas les valeurs que l'on mettrait en réalité pour un formulaire. Il est au contraire configuré pour que vous voyiez bien l'utilité des propriétés.

ASP.NET

```
<fieldset style="width: 300px">
    <legend>Remplissez le formulaire</legend>
    <br />
    Entrez votre nom : <asp:TextBox ID="Name" runat="server" />
    <br />
    <asp:Button ID="Valider" runat="server" Text="Valider" OnClick="ButtonClic" />
    <br /><br />
    <asp:Label ID="errors" runat="server" Text="No error message" />
</fieldset>

<cc1:NoBot ID="NoBot1"
        runat="server"
        ResponseMinimumDelaySeconds="10"
        CutoffMaximumInstances="4"
        CutoffWindowSeconds="5" />
```

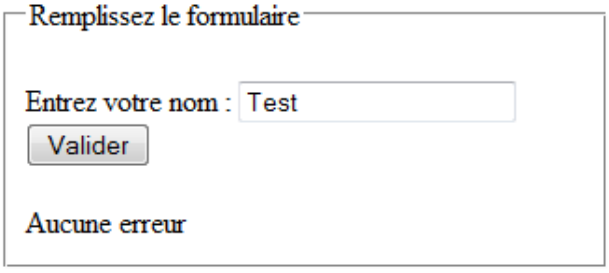
C#

```
protected void ButtonClic(object sender, EventArgs e)
{
    NoBotState etat;
    if (!NoBot1.IsValid(out etat))
    {
        errors.Text = "Message d'erreur : " + etat.ToString();
    }
    else
    {
        errors.Text = "Aucune erreur";
    }
}
```

VB.NET

```
Protected Sub ButtonClic(ByVal sender As Object, ByVal e As EventArgs)

  Dim etat As NoBotState
  If Not NoBot1.IsValid(etat) Then
    errors.Text = "Message d'erreur : " + etat.ToString()

  Else
    errors.Text = "Aucune erreur"
  End If

End Sub
```

Vous pouvez voir que suivant le temps que vous mettez à répondre au formulaire, vous pourrez être considéré comme un bot ou non. Vous ne pouvez renvoyer ce formulaire que 4 fois, au delà de quoi il vous traitera comme un bot.



Il est à prendre en considération qu'il va falloir vérifier vous-même dans le code behind, si l'utilisateur est ou non un bot.

4.3 Rating

Définition :

Le Rating est un contrôle Ajax qui va nous permettre de créer un système de notation par étoiles : 5 . La valeur de notation peut-être récupérée pour insertion dans une base de données par exemple.

Propriétés :

Nom	Description	Obligatoire
AutoPostBack	Mettre à *true* si vous voulez effectuer un PostBack à chaque clic sur le contrôle **Rating**.	NON
CurrentRating	Permet d'initialiser une note par défaut ou bien de récupérer la notation dans du code behind.	NON
MaxRating	Permet de définir le nombre d'étoiles à afficher.	NON
ReadOnly	Permet de mettre la notation en lecture seule.	NON
StarCssClass	Permet d'applique un style pour l'affichage des étoiles.	OUI
WaitingStarCssClass	Permet d'applique un style pour l'affichage des étoiles sur lesquelles on a cliqué.	OUI
FilledStarCssClass	Permet d'applique un style pour l'affichage des étoiles remplies.	OUI
EmptyStarCssClass	Permet d'applique un style pour l'affichage des étoiles vides	OUI
RatingAlign	Permet de définir comment les étoiles vont s'aligner : Vertical ou Horizontal (Horizontal par défaut).	NON
RatingDirection	Permet de définir le sens de notation : de gauche à droite (*par défaut)*, de droite à gauche, de haut en bas ou de bas en haut.	NON
OnChanged	Permet de créer un évènement géré en code behind lorsque l'on change son *rating*.	NON
Tag	Permet de définir un paramètre personnalisé utilisé lors d'un CallBack.	NON

Exemple d'utilisation :

Nous allons utiliser le contrôle **Rating** pour noter la préférence d'utilisation de Visual Studio. Puis dans du code Behind nous récupérerons la notation pour l'associer à un texte de réponse.

ASP.NET

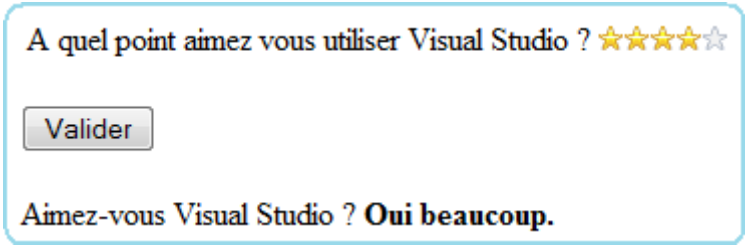
```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <table>
            <tr>
                <td>
                    A quel point aimez vous utiliser Visual Studio ?
                </td>
                <td>
                    <ajaxtoolkit:Rating ID="Rating1" runat="server"
                        StarCssClass="ratingStar"
                        FilledStarCssClass="filledRatingStar"
                        EmptyStarCssClass="emptyRatingStar"
                        WaitingStarCssClass="savedRatingStar">
                    </ajaxtoolkit:Rating>
                </td>
            </tr>
        </table>
        <br />
        <asp:Button ID="BtnSendRating" runat="server" Text="Valider"
            onclick="BtnSendRating_Click" />
        <br />
        <br />
        <asp:Label ID="LblGetRating" runat="server" Text=""></asp:Label>
    </ContentTemplate>
</asp:UpdatePanel>
```

C#

```
protected void BtnSendRating_Click(object sender, EventArgs e)
{
    string howMuch = " Je ne sais pas.";
    switch (Rating1.CurrentRating)
    {
        case 1:
            howMuch = " Pas du tout !";
            break;
        case 2:
            howMuch = " Un peu.";
            break;
        case 3:
            howMuch = " Oui.";
            break;
        case 4:
            howMuch = " Oui beaucoup.";
            break;
        case 5:
            howMuch = " Je n'utilise que ça.";
            break;
    }
    LblGetRating.Text = "Aimez-vous Visual Studio ?" + howMuch + "</b>";
}
```

```
VB.NET

Protected Sub Submit_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim howMuch As String = " Je ne sais pas."
    Select Case Rating1.CurrentRating
        Case 1
            howMuch = " Pas du tout !"
        Case 2
            howMuch = " Un peu."
        Case 3
            howMuch = " Oui."
        Case 4
            howMuch = " Oui beaucoup."
        Case 5
            howMuch = " Je n'utilise que ça."
    End Select
    LblGetRating.Text = "Aimez-vous Visual Studio ?<b>" + howMuch + "</b>"
End Sub
```

Voici un aperçu de cet exemple :



Après validation, le texte change en fonction de votre notation.

4.4 ReorderList

Définition :

Le ReorderList nous permet de créer une liste d'éléments que l'on va pouvoir organiser selon notre préférence. De plus, ce contrôle crée notre liste à partir d'éléments contenus dans une base de données.

Propriété :

Nom	Description	Obligatoire
DataSourceID	Permet de lier le ReorderList à un SqlDataSource afin de remplir notre liste.	OUI
DataKeyField	Permet d'indiquer le nom de la colonne en clé primaire	OUI
ItemInsertLocation	Permet de définir où un nouvel va s'insérer (au début ou à la fin.	NON
DragHandleAlignment	Permet de définir où se situera le bouton de déplacement d'un élément de la liste.	NON
AllowReorder	Permet d'autoriser/interdire la réorganisation de la liste (prend *true/false* en paramètre).	NON
ItemTemplate	Permet de définir le modèle d'affichage des éléments de la liste.	OUI

EditItemTemplate	Permet de définir le modèle d'affichage des éléments de la liste lors d'une édition d'un élément.	NON
ReorderTemplate	Permet de définir le modèle d'affichage d'un nouvel emplacement lors du déplacement d'un élément de la liste.	OUI
InsertItemTemplate	Permet de définir le modèle d'affichage des éléments de la liste lors de l'ajout d'un élément.	NON
DragHandleTemplate	Permet de définir le modèle d'affichage lors du déplacement d'un élément de la liste.	OUI
EmptyListTemplate	Permet de définir le modèle d'affichage pour une liste vide.	NON
PostBackOnReorder	Permet de définir si un PostBack doit être effectué après une modification de liste (prend *true/false* en paramètre)	OUI

Exemple d'utilisation :

Dans cet exemple, nous allons créer une table contenant quelques plats cuisinés, et donner la possibilité de réorganiser une liste d'éléments contenant nos plats, selon notre préférence.

Voici comment est définie notre table que nous appellerons *FavoriteFood* :

Nom de la colonne	Type de données	Null autorisé
ID	int	☐
Plat	nvarchar(50)	☐
Description	nvarchar(50)	☐
Preference	int	☐
		☐

Remarque : L'ID est en auto-incrémentation.

Voici les données contenu dans cette table :

ID	Plat	Description	Preference
1	Croque-Monsie...	Fromage et Jambon entre tranche de pain grillé	8
2	Sushi	Riz et Poisson frais	6
3	Hamburger	Steak Salade et Tranche de Pain	8
4	Salade Composé	Salade avec divers ingrédients	7

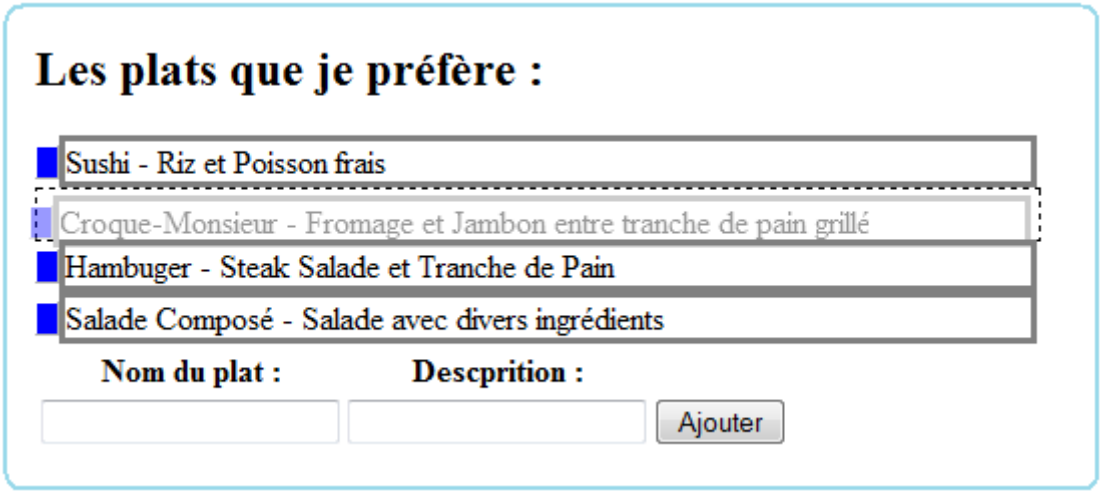
Voici le code source de cet exemple :

```
ASP.NET
<form id="form1" runat="server">
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<div>
    <h2>Les plats que je préfère :</h2>
    <br />
    <ajaxtoolkit:ReorderList ID="ReorderList1" runat="server"
        PostBackOnReorder="false"
        DragHandleAlignment="Left"
        ItemInsertLocation="End"
        DataKeyField="ID"
        SortOrderField="Preference"
        Width="500"
        DataSourceID="SqlDataSource1">
        <ItemTemplate>
            <div style="border-style:solid; border-color:Gray">
```

```
                <asp:Label ID="Label2" runat="server"
                    Text='<%#
HttpUtility.HtmlEncode(Convert.ToString(Eval("Plat"))) %>' />
                <asp:Label ID="Label1" runat="server"
                    Text='<%#
HttpUtility.HtmlEncode(Convert.ToString(Eval("Description", " - {0}"))) %>' />
            </div>
        </ItemTemplate>
        <ReorderTemplate>
            <asp:Panel runat="server" ID="Panel1" CssClass="reorderCue" />
        </ReorderTemplate>
        <DragHandleTemplate>
            <div class="dragHandle">
            </div>
        </DragHandleTemplate>
        <InsertItemTemplate>
            <table>
                <tr>
                    <th>Nom du plat :</th>
                    <th>Descprition :</th>
                    <th></th>
                </tr>
                <tr>
                    <td><asp:TextBox ID="TxtPlat" runat="server"
                        Text='<%# Bind("Plat") %>' /></td>
                    <td><asp:TextBox ID="TextDescription" runat="server"
                        Text='<%# Bind("Description") %>' /></td>
                    <td><asp:Button ID="Button1" runat="server"
                            Text="Ajouter" CommandName="Insert" /></td>
                </tr>
            </table>
        </InsertItemTemplate>
    </ajaxtoolkit:ReorderList>
</div>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConflictDetection="CompareAllValues"
    ConnectionString="<%$ ConnectionStrings:AjaxToolkitDBConnectionString %>"
    DeleteCommand="DELETE FROM [FavoriteFood] WHERE [ID] = @original_ID AND [Plat]
= @original_Plat AND [Description] = @original_Description AND [Preference] =
@original_Preference"
    InsertCommand="INSERT INTO [FavoriteFood] ([Plat], [Description], [Preference])
VALUES (@Plat, @Description, @Preference)"
    OldValuesParameterFormatString="original_{0}"
    SelectCommand="SELECT * FROM [FavoriteFood]"
    UpdateCommand="UPDATE [FavoriteFood] SET [Plat] = @Plat, [Description] =
@Description, [Preference] = @Preference WHERE [ID] = @original_ID AND [Plat] =
@original_Plat AND [Description] = @original_Description AND [Preference] =
@original_Preference">
    <DeleteParameters>
        <asp:Parameter Name="original_ID" Type="Int32" />
        <asp:Parameter Name="original_Plat" Type="String" />
        <asp:Parameter Name="original_Description" Type="String" />
        <asp:Parameter Name="original_Preference" Type="Int32" />
    </DeleteParameters>
    <UpdateParameters>
        <asp:Parameter Name="Plat" Type="String" />
        <asp:Parameter Name="Description" Type="String" />
        <asp:Parameter Name="Preference" Type="Int32" />
        <asp:Parameter Name="original_ID" Type="Int32" />
        <asp:Parameter Name="original_Plat" Type="String" />
        <asp:Parameter Name="original_Description" Type="String" />
        <asp:Parameter Name="original_Preference" Type="Int32" />
    </UpdateParameters>
    <InsertParameters>
        <asp:Parameter Name="Plat" Type="String" />
        <asp:Parameter Name="Description" Type="String" />
        <asp:Parameter Name="Preference" Type="Int32" />
    </InsertParameters>
</asp:SqlDataSource>
</form>
```

Remarque : Le contrôle SqlDataSource doit être adapté à la base de données. Il ne faut pas oublier que le contenu est généré tout seul via la configuration de notre source de données en mode Design.

Voici un aperçu de cet exemple :



4.5 Tabs

Définition :

Le contrôle Ajax Tabs, va vous permettre d'insérer des éléments de votre page dans plusieurs onglets cliquables.

Propriétés :

Pour le **TabContainer** (le conteneur d'onglet) :

Nom	Description	Obligatoire
ActiveTabChanged	Méthode appelé lors d'un changement d'onglet après un PostBack.	NON
CssClass	Permet d'appliquer un style à nos onglets.	NON
ActiveTabIndex	Permet de définir quel onglet on affiche en premier au chargement de la page.	NON
Height/Width	Permet de définir la hauteur/largeur maximum de notre conteneur par onglet.	NON
ScrollBars	Permet de définir si on affiche la barre de défilement ou non et de quelle manière.	NON
TabStripPlacement	Permet de définir ou vont s'afficher les onglets : en haut, en bas (à *TOP* par défaut).	NON

Pour le **TabPanel** (l'onglet) :

Nom	Description	Obligatoire
Enabled	Permet d'activer ou non l'affichage de l'onglet.	NON
HeaderText	Permet de définir le texte affiché en titre d'onglet.	OUI
HeaderTemplate	Permet de définir le texte affiché en titre d'onglet via ce	NON

	contrôle (Ex : <HeaderTemplate>Titre</HeaderTemplate>).	
ContentTemplate	La balise **ContentTemplate** permet d'ajouter du contenu à votre onglet.	OUI

Exemple d'utilisation :

Dans cet exemple nous allons créer un mini-formulaire basique, et récupérer les informations pour les afficher dans un contrôle Label.

ASP.NET

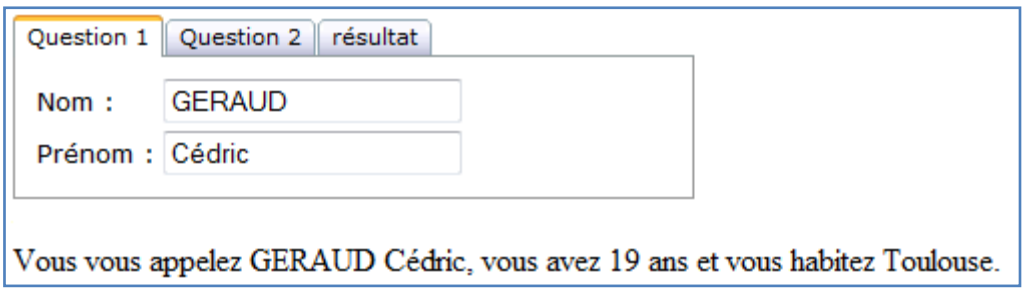
```
<ajaxtoolkit:TabContainer ID="TabContainer1" runat="server" Width="340px"
    ActiveTabIndex="2">
    <ajaxtoolkit:TabPanel runat="server" ID="Tab1" HeaderText="Question 1">
        <ContentTemplate>
            <table>
                <tr>
                    <td>
                        Nom :
                    </td>
                    <td>
                        <asp:TextBox runat="server" ID="TxtNom" />
                    </td>
                </tr>
                <tr>
                    <td>
                        Prénom :
                    </td>
                    <td>
                        <asp:TextBox runat="server" ID="TxtPrenom" />
                    </td>
                </tr>
            </table>
        </ContentTemplate>
    </ajaxtoolkit:TabPanel>
    <ajaxtoolkit:TabPanel runat="server" ID="Tab2" HeaderText="Question 2">
        <ContentTemplate>
            <table>
                <tr>
                    <td>
                        Age :
                    </td>
                    <td>
                        <asp:TextBox runat="server" ID="TxtAge" />
                    </td>
                </tr>
                <tr>
                    <td>
                        Ville :
                    </td>
                    <td>
                        <asp:TextBox runat="server" ID="TxtVille" />
                    </td>
                </tr>
            </table>
        </ContentTemplate>
    </ajaxtoolkit:TabPanel>
    <ajaxtoolkit:TabPanel runat="server" ID="Tab3" HeaderText="résultat">
        <ContentTemplate>
            Cliquez pour terminer :<br />
            <asp:Button ID="BtnSendInfo" runat="server" Text="Envoyer"
                onclick="BtnSendInfo_Click" />
        </ContentTemplate>
    </ajaxtoolkit:TabPanel>
</ajaxtoolkit:TabContainer>
<br />
<asp:Label ID="LblGetInfo" runat="server"></asp:Label>
```

C#

```
protected void BtnSendInfo_Click(object sender, EventArgs e)
{
    LblGetInfo.Text = "Vous vous appelez " + TxtNom.Text + " " + TxtPrenom.Text
                    + ", vous avez " + TxtAge.Text + " ans et vous habitez "
                    + TxtVille.Text + ".";
}
```

```
VB.NET

Protected Sub BtnSendInfo_Click(ByVal sender As Object, ByVal e As EventArgs)
Handles BtnSendInfo.Click

    LblGetInfo.Text = "Vous vous appelez " + TxtNom.Text + " " + TxtPrenom.Text + ",
                      vous avez " + TxtAge.Text + " ans et vous habitez " +
                      TxtVille.Text + "."

End Sub
```

Voici un aperçu de cet exemple après validation du formulaire :



5 Conclusion

Au travers des différents exemples de contrôles ACT présentés dans ce cours, vous avez pu observer, qu'avec un faible effort de développement, il était possible de créer interfaces graphiques conviviales dans vos applications Web.

Aussi, en tant que projet Open Source, il est tout à fait possible de récupérer leurs codes sources, pour pouvoir les étudier d'avantage. Et en approfondissant vos connaissances dans Microsoft ASP .NET Ajax, il vous sera possible de créer vous-même vos propres extendeurs en AJAX.