ASP Microsoft .net

Etape par étape

2ème partie : Création d'un formulaire simple

(Révision : 2 du 05/11/2004 – 31 pages)

Avertissement :

Ce document peut comporter des erreurs. Cependant, tout a été mis en œuvre afin de ne pas en inclure dans ce texte. Tout code qui trouve sa place ici a été testé au préalable.

Tables des matières :

Bibliographie

Gérard Leblanc, c# et .NET, ed. Eyrolles

MSDN sur le site

Au travers de cette deuxième partie, nous allons construire un formulaire simple.

Celui-ci contiendra la plupart des composants asp.NET dits « simples ». Ces composants possèdent certaines propriétés identiques qui seront vues en premier lieu. Ensuite, nous verrons, composant par composant la majorité de leurs propriétés particulières importantes. Nous terminerons, dans la dernière partie, par voir les propriétés qui se trouvent dans la plupart des composants mais qui sont moins utilisés.

Nous utiliserons quelques commandes en c# (de nouveau, on aurait pu adopter le VB.NET comme langage de référence) afin de vérifier que les données encodées correspondent à la syntaxe attendue.

Utilisation des composants

Ce tableau permet, de manière simple et rapide, de voir quel composant utiliser suivant l'utilité envisagée.

Label	Composant très simple permettant d'afficher du texte. Il est possible de modifier ce texte de manière aisée suivant les situations rencontrées. Le texte peut inclure des balises html.
Button	Exécute des fonctions. A utiliser pour utiliser du code quand on le souhaite.
TextBox	Il s'agit d'une « case » que l'on peut compléter. Cela permet de récupérer des chaînes de caractères encodées l'utilisateur.
CheckBox	A n'utiliser que lors d'un choix qui se limite à deux valeurs (Oui/Non, Homme/Femme…)
RadioButtonList	Permet de sélectionner un choix parmi une liste de choix possible. Dans le cas d'un nombre trop élevé de choix, ce composant n'est pas adéquat car il prend de la place (plus exactement chacun des éléments de la liste de choix prend de la place).
ListBox	La place utilisée pour une telle liste est fixe. On peut voir plusieurs éléments de celle ci.
DropDownList	Il s'agit du composant le plus utilisé en terme de listes. Concrètement, il s'agit de celui qui prend le moins de place. Il s'agit d'une liste déroulante qui se met au premier plan, par dessus les autres composants lorsqu'elle est déroulée. Aucune place n'est ainsi perdue.
Image	Comme son nom l'indique, il s'agit d'une image.
ImageButton	Il s'agit d'un bouton qui a la forme d'une image.
Hyperlink	Très utilisés pour passer d'une page à l'autre.
LinkButton	Il s'agit d'un bouton qui a la forme d'un hyperlien.
Panel	Ce composant est intéressant pour afficher ou non un nombre important de composants d'un coup.
Calendar	Il s'agit d'un calendrier qui a la forme qui est la plus utilisée. Il s'agit d'un composant très simple pour choisir une date.

Propriétés communes importantes

Ici, nous allons voir la liste des propriétés qui se retrouvent au travers des différents composants. Ces propriétés sont, pour la plupart, la pièce d'œuvre de l'utilisation de ces composants. Les connaître permet un développement facile et aisé par la suite. C'est la raison pour laquelle ces propriétés sont vues avant même de voir le premier composant.

Les voici donc :

Id : Tout composant peut porter un nom, ce nom est l'Id. C'est à partir de ce nom que nous pourrons modifier ses attributs dans du code c#. On ne peut donc avoir deux fois le même Id dans une même page asp.NET. Il n'est donc pas nécessaire de lui donner de nom si celui-ci n'est pas manipulé dans du code.

OnClick : est l'événement lorsque l'on clique sur un composant, par exemple un bouton

Runat : Cet attribut permet de spécifier où le traitement du composant doit être fait.

Text : est la valeur affichée par le composant. Cette valeur peut inclure des balises html qui seront considérées en tant que telles.

OnTextChanged : est l'événement lorsque le texte change. Comme pour le ***OnClick***, il faut donc spécifier une méthode de traitement et lui affecter son nom.

AutoPostBack : Si sa valeur est à ***true***, la page est automatiquement renvoyée au serveur pour qu'il puisse faire son traitement. Dans le cas contraire, la page ne sera envoyée que lors d'un clic sur un bouton, …

Il en existe d'autres, moins importantes, qui sont détaillées à la fin de ce tome. Ces dernières propriétés sont utilisées pour la mise en forme des composants et également pour l'utilisation dynamique de ceux-ci.

	Id	OnClick	Runat	Text	OnTextChanged	AutoPostBack
Label	√		√	√		
Button	√	√	√	√		
TextBox	√		√	√	√	√
CheckBox	√		√		√	√
RadioButtonList	√		√		√	√
ListBox	√		√			√
DropDownList	√		√			√
Image	√		√			
ImageButton	√	√	√			
Hyperlink	√		√	√		
LinkButton	√	√	√	√		
Panel	√		√			
Calendar	√		√			

Passons maintenant aux composants proprement dits et ce, de manière progressive, c'est à dire en fonction du niveau de difficulté de ceux-ci.

Label (zone d'affichage)

Le premier composant asp.NET que nous allons inclure dans notre page web est une zone d'affichage. Comme il a été expliqué dans la première partie, le composant sera compilé par le serveur et traduit en html avant d'être envoyé au client.

La syntaxe la plus simple pour ce composant est la suivante :

```
<asp:Label id="zaValeur" text="Notre premier composant asp" runat="server"/>
```

Ce qui donne dans notre code :

```
<head>
</head>
<body>
    <form runat="server">
        <asp:Label id="zaValeur" text="Notre premier composant asp" runat="server"/>
    </form>
</body>
```

Label.aspx

Ce qui donne sur le navigateur du client :

Comme il a déjà été dit, il est possible de passer des balises html dans la valeur du texte. Ainsi

```
<asp:Label id="zaValeur" text="<b>Notre premier composant asp</b>" runat="server"/>
```

affiche le même texte si ce n'est que celui-ci est affiché en gras. Ceci est très utile dans certains cas que nous verrons par la suite.

Button (bouton)

Un bouton est un composant permettant lors d'un clic de souris sur celui-ci d'exécuter la méthode de l'événement **OnClick**.

Voici un exemple de syntaxe d'une balise Button :

```
<asp:Button id="bEnvoi" text="Envoi !" OnClick="Envoi()" runat="server"/>
```

Nous remarquons le « b » comme premier caractère du nom du bouton. Nous emploierons cette norme de nommer un bouton.

A notre page, nous allons ajouter ce bouton, ainsi que le code de la méthode de traitement correspondante. Nous allons nous limiter ici à changer le texte affiché dans la zone d'affichage définie précédemment.

Nous allons d'abord montrer le code et ensuite nous l'analyserons.

```
<head>
	<script language="c#" runat="server">
		void Envoi(Object Sender, EventArgs E)
		{
			zaValeur.Text="Notre deuxième composant asp fonctionne :)";
		}
	</script>
</head>
<body>
	<form runat="server">
		<asp:Label id="zaValeur" text="Notre premier composant asp" runat="server"/>
		<br clear="all">
		<asp:Button id="bEnvoi" text="Envoi !" OnClick="Envoi" runat="server"/>
	</form>
</body>
```

Button.aspx

On voit, comme on s'y attendait, que nous avons mis la balise asp.NET du bouton entre les balises ***<form>*** et ***</form>***. Nous devions également ajouter le code du traitement du bouton. C'est ce que l'on trouve dans la partie du script en c# (entre ***<script… >*** et ***</script>***).

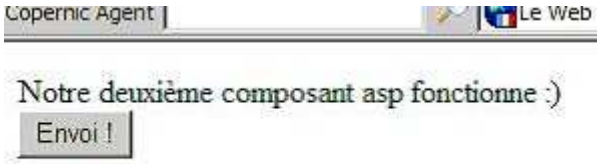
La fonction Envoi() ne renvoie rien, cependant, elle change la propriété ***Text*** de la zone d'affichage zaValeur. On remarquera également la présence des paramètres ***Object Sender*** et ***EventArgs E*** comme dans les Forms Windows. Si cela ne vous paraît pas clair, peut être qu'un petit tour sur Le langage c# , les premiers pas pourrait vous aider.

On se rend vite compte que le principe est semblable à la programmation Windows. Dans le cadre de ce tutorial, les méthodes et fonctions se trouvent dans la page dont elles dépendent. Cependant il est possible d'utiliser la méthode dite du « Code_behind » qui consiste à séparer le code de l'interface.

En plus des propriétés habituelles, un bouton a également les propriétés suivantes :

- ***CommandName*** est la commande associée au bouton. Il s'agit de la même fonction de traitement quand dans la propriété ***OnClick***. Le seul avantage étant si cette même méthode est utilisée par plusieurs boutons.
- ***CommandArgument*** est l'argument éventuellement associée à la commande.

Après avoir cliqué sur le bouton, on obtient :



TextBox (zone d'édition)

Une zone d'édition est une zone de texte qui peut être modifiée par l'utilisateur. Nous allons principalement nous en servir pour récupérer des données qu'il devra encoder.

La balise minimale pour une zone d'édition est la suivante :

```
<asp :TextBox id="zeSaisie" runat="server"/>
```

Une zone d'édition peut également avoir les propriétés suivantes :

- ***Columns*** : il s'agit de la largeur du contrôle exprimée en nombre de caractères. Si le nombre est inférieur à ***MaxLength***, il y aura défilement horizontal dans la boîte d'édition
- ***MaxLength*** : est le nombre maximal de caractères qui seront admis. Pour avoir un effet, il faut que ***TextMode*** vaut ***SingleLine*** ou ***Password***.
- ***Rows*** : est le nombre de lignes visibles. Il faut dans ce cas que ***TextMode*** soit ***MultiLine***. On peut saisir plus de lignes mais dans ce cas, il y aura défilement vertical
- ***TextMode*** : Les trois modes qui sont possibles sont ***Single***, ***MultiLine*** ou ***Password***. Chacune des significations a été expliquée dans les propriétés précédentes
- ***Wrap*** : permet de stipuler si oui ou non il y passage automatique à la ligne dans une zone d'édition de plusieurs lignes

Dans notre page web, nous allons faire que la valeur affichée dans la zone d'affichage soit la valeur saisie dans la zone d'édition.

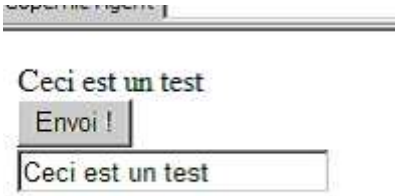
On obtient ainsi :

```
<head>
	<script language="c#" runat="server">
		void Envoi(Object Sender, EventArgs E)
		{
			zaValeur.Text=zeSaisie.Text;
		}
	</script>
</head>
<body>
	<form runat="server">
		<asp:Label id="zaValeur" text="Notre premier composant asp" runat="server"/>
		<br clear="all">
		<asp:Button id="bEnvoi" text="Envoi !" OnClick="Envoi" runat="server"/><br clear="all">
		<asp:TextBox id="zeSaisie" runat="server"/>
	</form>
</body>
```

TextBox.aspx

Le fonctionnement du code suivant est simple. Lors du clic sur le bouton, la méthode de traitement de celui-ci est appelée. Dans notre exemple, cette méthode effectue l'opération suivante : il lit la valeur saisie dans la boite, en ressort et le texte et l'affecte à la propriété Text de la zone d'affichage.

Le résultat est le suivant :



Premières modifications et améliorations

Nous allons modifier légèrement la page afin d'arriver à ce que l'on attend. Ainsi, avec les éléments déjà présent, nous pouvons déjà inclure à la page les éléments suivants :

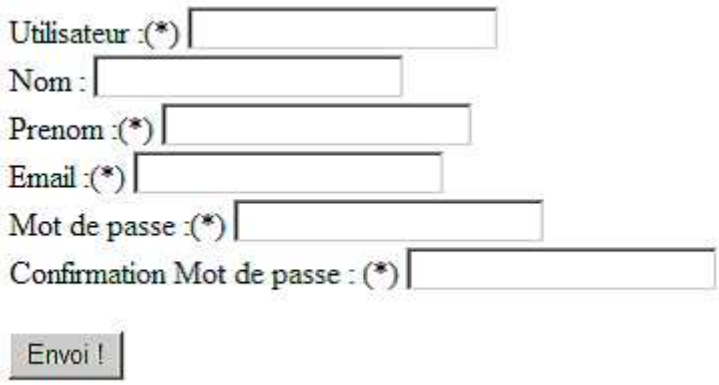
- Un nom d'utilisateur d'une longueur minimale de 5 caractères et de maximum 10
- Un nom de famille facultatif dont la longueur doit être inférieure à 25 caractères. La zone fera à l'écran 10 caractères de large
- Un prénom non facultatif. La longueur est identique au nom
- Un email
- Un mot de passe
- Une confirmation de mot de passe

On se rend vite compte qu'il sera déjà nécessaire de faire des contrôles sur les diverses valeurs qui seront saisies.

Nous allons suivre le formalisme suivant pour une ligne :

- Une zone d'affichage comprenant l'intitulé
- Une zone d'affichage signifiant si l'élément est facultatif ou non
- Une zone d'édition dans laquelle les valeurs seront saisies
- Une zone d'affichage qui permettra de signaler une erreur d'encodage

Ce qui donne l'affichage suivant :



Voici le code de la page correspondante côté serveur:

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

   void Envoi(Object Sender, EventArgs E)
   {
      // A AJOUTER: 'strlen' 5 a 10 caracteres
      if (zeUtilisateur.Text=="") zaErrUtilisateur.Text="Vous devez entrer un nom d'utilisateur dont la longueur varie entre 5 et 10 caractres";
      else zaErrUtilisateur.Text="";
```

```
        if (zePrenom.Text=="") zaErrPrenom.Text="Vous devez entrer votre prnom dont la longueur varie entre 5 et 10 caractres";
        else zaErrPrenom.Text="";

        if (zeMdp.Text=="") zaErrMdp.Text="Vous devez entrer un mot de passe";
        else zaErrMdp.Text="";

        if (zeConfMdp.Text=="") zaErrConfMdp.Text="Vous devez entrer une deuxime fois votre mot de passe";
        else { if (zeConfMdp.Text!=zeMdp.Text) zaErrConfMdp.Text="Vos mots de passe ne concident pas";
        else zaErrConfMdp.Text="";}
    }

</script>
<html>
<head>
</head>
<body>
    <form runat="server">
        <!-- Utilisateur --><asp:Label id="zaUtilisateur" runat="server" text="Utilisateur :"></asp:Label><asp:Label id="zaFacUtilisateur" runat="server" text="(*)"></asp:Label>
        <asp:TextBox id="zeUtilisateur" runat="server"></asp:TextBox>
        <asp:Label id="zaErrUtilisateur" runat="server" text=""></asp:Label>
        <br clear="all" />
        <!-- Nom --><asp:Label id="zaNom" runat="server" text="Nom :"></asp:Label><asp:Label id="zaFacNom" runat="server" text=""></asp:Label>
        <asp:TextBox id="zeNom" runat="server"></asp:TextBox>
        <asp:Label id="zaErrNom" runat="server" text=""></asp:Label>
        <br clear="all" />
        <!-- Prenom --><asp:Label id="zaPrenom" runat="server" text="Prenom :"></asp:Label><asp:Label id="zaFacPrenom" runat="server" text="(*)"></asp:Label>
        <asp:TextBox id="zePrenom" runat="server"></asp:TextBox>
        <asp:Label id="zaErrPrenom" runat="server" text=""></asp:Label>
        <br clear="all" />
        <!-- Email --><asp:Label id="zaEmail" runat="server" text="Email :"></asp:Label><asp:Label id="zaFacEmail" runat="server" text="(*)"></asp:Label>
        <asp:TextBox id="zeEmail" runat="server"></asp:TextBox>
        <asp:Label id="zaErrEmail" runat="server" text=""></asp:Label>
        <br clear="all" />
        <!-- Mot de passe --><asp:Label id="zaMdp" runat="server" text="Mot de passe :"></asp:Label><asp:Label id="zaFacMdp" runat="server" text="(*)"></asp:Label>
        <asp:TextBox id="zeMdp" runat="server"></asp:TextBox>
        <asp:Label id="zaErrMdp" runat="server" text=""></asp:Label>
        <br clear="all" />
        <!-- Confirmation mot de passe --><asp:Label id="zaConfMdp" runat="server" text="Confirmation Mot de passe :"></asp:Label><asp:Label id="zaFacConfMdp" runat="server" text="(*)"></asp:Label>
        <asp:TextBox id="zeConfMdp" runat="server"></asp:TextBox>
        <asp:Label id="zaErrConfMdp" runat="server" text=""></asp:Label>
        <br clear="all" />
        <br clear="all" />
        <!-- Bouton d'envoi -->
        <asp:Button id="bEnvoi" onclick="Envoi" runat="server" text="Envoi !"></asp:Button>
    </form>
</body>
</html>
```

1stLogin2.aspx

Et voici maintenant le code de la page html envoyé au client.

```
<html>
<head>
</head>
<body>
   <form name="_ctl0" method="post" action="1stLogin2.aspx" id="_ctl0">
<input type="hidden" name="__VIEWSTATE"
value="dDwxOTc3ODg3MTQ1OztsPGNjTWFpbGluZ0xpc3Q7Pj73NPpQr/3jNgkKYWjOd9CoS/gS
Lw==" />

   <!-- Utilisateur -->
      <span id="zaUtilisateur">Utilisateur :</span><span id="zaFacUtilisateur">(*)</span>
      <input name="zeUtilisateur" type="text" id="zeUtilisateur" />
      <span id="zaErrUtilisateur"></span>
      <br clear="all" />
   <!-- Nom -->
                  <span id="zaNom">Nom :</span><span id="zaFacNom"></span>
      <input name="zeNom" type="text" id="zeNom" />
      <span id="zaErrNom"></span>
      <br clear="all" />
   <!-- Prenom -->
            <span id="zaPrenom">Prenom :</span><span id="zaFacPrenom">(*)</span>
      <input name="zePrenom" type="text" id="zePrenom" />
      <span id="zaErrPrenom"></span>
      <br clear="all" />
   <!-- Email -->
               <span id="zaEmail">Email :</span><span id="zaFacEmail">(*)</span>
      <input name="zeEmail" type="text" id="zeEmail" />
      <span id="zaErrEmail"></span>
      <br clear="all" />
   <!-- Mot de passe -->
            <span id="zaMdp">Mot de passe :</span><span id="zaFacMdp">(*)</span>
      <input name="zeMdp" type="text" id="zeMdp" />
      <span id="zaErrMdp"></span>
      <br clear="all" />
   <!-- Confirmation mot de passe -->
                  <span id="zaConfMdp">Confirmation Mot de passe :</span>
      <span id="zaFacConfMdp">(*)</span>
      <input name="zeConfMdp" type="text" id="zeConfMdp" /><span
id="zaErrConfMdp"></span>
      <br clear="all" />
   <!-- Bouton d'envoi -->
      <input type="submit" name="bEnvoi" value="Envoi !" id="bEnvoi" />
   </form>
</body>
</html>
```

On remarque qu'il s'agit bien d'HTML pur. Plus aucune trace d'asp.NET.

Lors d'un clic sur le bouton « Envoi ! », sans avoir entré de valeurs, nous obtenons toutes une série de messages tels que ceux-ci :

Utilisateur :(*) Vous devez entrer un nom d'utilisateur dont la longueur varie entre 5 et 10 caractères
Nom :
Prenom :(*) Vous devez entrer votre prénom dont la longueur varie entre 5 et 10 caractères
Email :(*)
Mot de passe :(*) Vous devez entrer un mot de passe
Confirmation Mot de passe : (*) Vous devez entrer une deuxième fois votre mot de passe

Envoi !

Effectivement ! La méthode Envoi() qui a été appelée par le clic sur le bouton a effectuer quelques contrôles rudimentaires sur la valeur des zones de texte. Il y a également un contrôle sur l'égalité des deux zones de mots de passe.

Ici, tous ces contrôles se font du côté du serveur. Cependant, cela pourrait se faire côté client. Il existe par ailleurs des composants permettant de faire cela. Il s'agit des Validator (contrôles de validation). Ceux-ci sont détaillés dans l'article suivant :

Nous allons maintenant continuer notre petit tour des composants ASP .NET.

En regardant attentivement, on se rend compte que la page envoyée ne contient que des balises html. Cependant, on remarque un champ caché appelé ***__VIEWSTATE***. Ce champ permet, lors du renvoi de la page au serveur, de ne pas perdre les données des différents composants. Ainsi, la page ne doit être initialisée qu'une seule fois.

Cependant, il faut avouer que ce n'est pas très convivial tel quel. Quelques modifications en html donnent ceci :

Utilisateur :(*)
Nom :
Prenom :(*)
Email :(*)
Mot de passe :(*)
Confirmation Mot de passe :(*)

Envoi !

Cette page est déjà beaucoup plus agréable. En utilisant les feuilles de styles et autres, le design de la page ne peut que s'améliorer, cependant il ne sera pas vu dans ce cadre-ci.

CheckBox (case à cocher)

Une case à cocher permet de faire un choix de type oui/non ou vrai/faux.

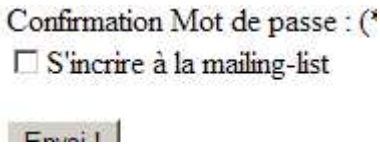
Comme toujours, ce contrôle possède au minimum un id. Il a également comme propriétés :

- ***Checked*** : est à ***true*** si la case est cochée et à ***false*** dans le cas contraire
- ***TextAlign*** : est la position du libellé par rapport à la case. Les valeurs possibles sont ***Right*** ou ***Left***

Nous obtenons donc, dans sa forme la plus simple un contrôle tel que celui-ci :

```
<asp:CheckBox id="ccMailingList" runat=server" text="S'incrire à la mailing-list"/>
```

Nous allons le rajouter aux composants déjà existants dans notre page d'inscription (1stLogin2.aspx), ce qui donne comme résultat :



RadioButtonList (cases à cocher)

Parmi plusieurs cases à cocher d'un même ensemble, on ne peut en choisir qu'une seule, ce qui implique que lorsque l'on cocher une autre case, celle qui était au préalable cochée est décochée. On les utilise donc lorsque l'on a plusieurs choix mais qu'une seule valeur ne peut être choisie.

Pour les propriétés, elles sont identiques à un CheckBox si ce n'est que ces cases à cocher doivent être dans un même groupe. On lui donne donc un nom (propriété GroupName).

Pour notre page d'inscription, on pourrait demander le titre d'une personne (Mme, Mr, Mlle). C'est ce que nous allons faire.

Pour ce composant il existe deux syntaxes :

```
<asp:RadioButton id="Madame" runat="server" value="Madame"
GroupName="rblMmeMlleMr"/>

    <asp:RadioButton id="Mademoiselle" runat="server" value="Mademoiselle"
GroupName="rblMmeMlleMr"/>

        <asp:RadioButton id="Monsieur" runat="server" value="Monsieur"
GroupName="rblMmeMlleMr"/>
```

ou encore

```
<asp:RadioButtonList id="rblMmeMlleMr" runat="server">
  <asp:ListItem value="Madame" runat="server"/>
  <asp:ListItem value="Mademoiselle" runat="server"/>
  <asp:ListItem value="Monsieur" runat="server"/>
</asp:RadioButtonList>
```

A l'affichage, les pages 1stLogin3.aspx et 1stLogin3b.aspx ne laissent transparaître aucune différence si ce n'est qu'avec la deuxième syntaxe, il y a eu retour à la ligne automatiquement. Il est donc équivalent d'utiliser une syntaxe ou l'autre.

La première syntaxe est avantageuse dans ce sens qu'elle permet de voir directement qu'il s'agit bien d'un composant RadioButton. La seconde permet d'avoir le groupe de cases à cocher de manière claire.

Le résultat obtenu avec la deuxième syntaxe :

Utilisateur : ( )

- Madame
- Mademoiselle
- Monsieur

ListBox (boîte de liste)

Une boîte de liste est un composant permettant d'afficher une liste de valeur possible.

Les caractéristiques de ce composant sont les suivantes :
- ***SelectionMode*** qui indique l'on peut sélectionner un (***Single***) ou plusieurs articles (***Multiple***) simultanément
- ***Items*** : qui est la collection des articles
- ***SelectedIndex*** : qui est le numéro de l'article sélectionné, le premier ayant comme valeur 0
- ***SelectedItem*** : est l'article sélectionné
- ***SelectedItems*** est la collection des articles sélectionés
- ***Rows*** est le nombre de lignes pour la liste visibles

Passons directement au code que nous allons insérer dans notre page web :

```
<asp:ListBox id="lbPays" runat="server" SelectionMode="Single" Rows="2">
<asp:ListItem text="Belgique" runat="server"/>
<asp:ListItem text="France" runat="server"/>
<asp:ListItem text="Suisse" runat="server"/>
</asp:ListBox>
```

On remarquera la syntaxe fort similaire au ***RadioButtonList***.

Ici, nous aurons 2 des 3 lignes qui seront affichées dans le contrôle. ***SelectionMode*** étant à ***Single***, nous ne pouvons en sélectionner qu'une seule valeur.

Il est évident que le contrôle est utilisé ici pour la sélection d'un nom de pays.

A l'affichage, nous obtenons donc ceci :

Continuons notre petit tour des composants…

DropDownList (boîte combo)

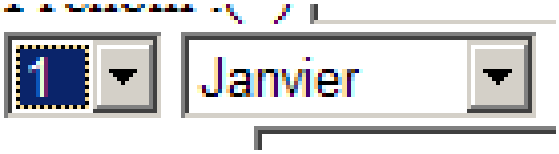
La boîte combo possède les mêmes propriétés qu'une ListBox.

Nous allons les utiliser pour par exemple une date de naissance. On pourrait utiliser trois ***DropDownList*** pour le faire. Cependant, nous ne mettrons pas l'année maintenant.

Par exemple, voici le code de la boîte combo pour le mois :

```
<asp:DropDownList id="ddlMoisNaissance" runat="server" SelectionMode="Single" Rows="10">
<asp:ListItem text="Janvier" runat="server"/>
<asp:ListItem text="Fevrier" runat="server"/>
...
<asp:ListItem text="Octobre" runat="server"/>
<asp:ListItem text="Novembre" runat="server"/>
<asp:ListItem text="Decembre" runat="server"/>
</asp:DropDownList>
```

Et voici le résultat :



Image

Pour celui qui a déjà vu les propriétés d'une image en html, les propriétés suivantes ne lui sembleront pas très compliquées. On y retrouve :

- ***ImageUrl*** qui est l'adresse de l'image (relative ou absolue)
- ***AlternateText*** est le texte à afficher si l'image ne peut l'être
- ***ImageAlign*** est l'alignement de l'image par rapport au contour de son composant. On retrouve comme valeurs acceptées **AbsBottom**, **AbsMiddle**, **BaseLine**, **Bottom**, **Left**, **Middle**, **NotSet**, **Right**, **TextTop** et **Top**.

Ce qui donne par exemple :

```
<asp:Image ImageUrl="dotnet.gif" runat="server" AlternateText="Made with ASP.NET"/>
```

ImageButton

D'apparence, ce composant est le même que le précédent, si ce n'est qu'un événement lui est associé lors du clic.

Pour ce qui est des propriétés, on ne retrouve que ***ImageUrl*** par rapport au précédent.

On obtient ainsi dans le script c# :

```
void ibClick(Object Sender, ImageClickEventArgs E)
  {
                    lImageButton.Text="Encore un contrôle qui fonctionne!";
  }
```

et dans la partie d'affichage

```
    <asp:ImageButton ImageUrl="dotnet2.gif" runat="server" AlternateText="Une info" OnClick="ibClick"/>
          <asp:Label id="lImageButton" runat="server" text=""/>
```

On remarque dans le script que l'on ne passe plus de ***EventArgs*** mais bien un ***ImageClickEventArgs***.

Un clic sur l'image de l'exemple fera afficher un texte dans une zone d'affichage.

Hyperlink (hyper-lien)

Toute personne qui souhaite faire un site web se doit de savoir ce qu'est un hyperlien. Cependant, voici quand même l'utilité : le clic sur un tel lien permet d'ouvrir une autre page web, télécharger un programme, envoyer un email ou ouvrir une adresse ftp. On peut les trouver sous deux formes : un texte ou une image.

Les caractéristiques sont simples :

- ***ImageUrl*** est l'image qui contient le lien (si l'on désire utiliser une image comme hyperlien). Si cette propriété est utilisée, il ne faut pas utiliser la propriété ***Text***.
- ***NavigateUrl*** est l'adresse destination

```
<asp:Hyperlink Text="Tutoriaux .NET" NavigateUrl="http:\\dotnet.developpez.com" runat="server"/>
```

L'exemple ci-dessus est facilement compréhensible. Un clic sur le texte « Tutoriaux .NET » permet d'ouvrir la page du site « dotnet.developpez.com ».

LinkButton (bouton avec apparence d'hyperlien)

Tout est dans le titre… Identique à un hyperlien lors d'un affichage, ce composant permet d'actionner un événement tel que l'on en a déjà vu précédemment.

Dans le script, on trouve :

```
void lbLinkButton(Object Sender, EventArgs E)
{
                    lLinkButton.Text="Bientot, nous serons des pros :)";
}
```

et dans la seconde partie :

```
<asp:LinkButton Text="LinkButton!" OnClick="lbLinkButton" runat="server"/>
<asp:Label id="lLinkButton" Text="" runat="server"/>
```

Panel (volet)

Le panneau contient d'autres composants asp.NET. On peut lui donner des effets de relief tout autour.

Ce composant est souvent utilisé pour afficher ou cacher un grand nombre de composants.

Ici, nous allons désactiver tout une série de composants à l'aide de l'ImageButton de tout à l'heure. Si ceux ci sont déjà désactivés, ils seront réactivés.

```
void ibClick(Object Sender, ImageClickEventArgs E)
  {
                                pTout.Enabled=!(pTout.Enabled);
  }
```

permet de changer l'activation ou non du panel pTout.

Les balises de début et de fin du volet sont respectivement :

```
<asp:Panel id="pTout" runat="server">
```

et

```
</asp:Panel>
```

Comme propriétés, on trouve :
- ***BackImageUrl*** qui est l'image de fond
- ***HorizontalAlign*** qui est l'alignement du contenu du volet (***Center***, ***Justify***, ***Left***, ***NotSet*** ou ***Right***)

Calendar (calendrier)

Il s'agit d'un calendrier complet prêt à l'emploi.

On trouve une liste interminable (mais logique) de propriétés pour ce composant. Si il y en a deux à retenir, il s'agit bien de celles-ci :

- ***SelectedDate*** qui donne la date sélectionnée
- ***VisibleDate*** qui est la date visible

Ceux deux éléments sont de type ***DateTime***.

```
<asp:Calendar VisibleDate="2003/12/11" runat="server"/>
```

La date est de type anglosaxone.

Voici le code de la nouvelle page :

```
<head>
</head>
<body>
<form runat="server">
                <asp:Calendar VisibleDate="2003/12/11" runat="server"/>
</form>
</body>
```

La simplicité de celui-ci saute aux yeux. Pourtant le résultat est surprenant :

<	décembre 2003					>
lun.	mar.	mer.	jeu.	ven.	sam.	dim.
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Il possible de changer de mois sans avoir écrit une seule ligne de code ! Simplissime !

Nous allons ajouter une deuxième page à notre exemple. Cette fonction sera utilisée par l'administrateur du site. Appellons la admin.aspx.

Initialisation d'une page

Il est peu pratique de devoir encoder toutes les valeurs comme dans une DropDownList pour, par exemple, l'année de naissance. Celui qui fait un site sur l'histoire et qui remonte à plusieurs centaines d'années risque de passer plus de temps à l'encodage de ces années qu'autre chose.

Il est ainsi possible d'initialiser une page avec les valeurs correspondantes.

C'est ce que fait l'événement Page_Load. On peut ainsi y mettre du code. Ainsi on a :

```
int i;
for (i=1 ; i<2005 ; i++)
{
        DDLNaissAnnee.Items.Add=i ;
}
```

Ce type d'initialisation fera partie d'un prochain tutorial.

Propriétés communes supplémentaires

- **AccessKey** permet de spécifier une combinaison de touches de type Alt+<touche> pour servir de raccourci. La syntaxe exacte est **AccessKey= « touche »**.

- **Attributes** est la collection des attributs du composant. Ces attributs peuvent être atteints à l'aide des indexeurs (()). Ce sont eux qui gardent les diverses informations sur les composants.

- **BackColor** est, comme son nom l'indique, la couleur d'arrière plan. Les valeurs acceptés pour cette propriétés sont divers noms de couleurs (**Red**, **Blue**, **Green**, … Dans ce cas, il faut importer **System.Drawing** qui contient la définition de ces couleurs) mais aussi les valeurs exprimées en hexadécimal au format RGB (c'est à dire celle utilisée en HTML). On trouve ainsi des choses telles que : **BackColor= « #FF0012 ».**

- **BorderWidth** spécifie l'épaisseur de la bordure autour du composant. La syntaxe est donc **BorderWidth= « NbPixels »**.

- **BorderStyle** que l'on traduit par « style de bordure ». L'énumération possible est **None** (aucune), **Solid** (ligne pleine), **Double** (ligne double), **Groove** (effet de relief), **Ridge** (effet de relief également), **Inset** (image dans l'écran), **Outset** (image qui ressort fortement de l'écran).

- **CssClass** permet de spécifier le nom d'une classe de la feuille de style spécifiée au préalable. Ceci permet, dans le cas de mise en forme répétée, de simplifier l'écriture (moins de répétition au niveau de font-) mais aussi de faciliter des mises à jour de la police par exemple au travers du document.

- **Enabled** indique si il est possible d'effectuer une action sur le composant (**true** / **false**).

- **Font-Bold** indique si le texte doit être affiché en gras ou non (**true** / **false**).

- **Font-Italic** indique si le texte doit être affiché en italique ou non (**true** / **false**).

- **Font-Name** contient le nom de la police de caractères. On trouve par exemple **Font-Name= « Times New Roman »**.

- **Font-Overline** avec la valeur **true** permet de surligner le texte.

- **Font-Size** est la taille des caractères. Elle peut être exprimée de quatre manières : par défaut en pixels (**px**) ou alors en points typographiques (**pt**), en millimètres (**mm**) ou en centimètres (**cm**).

- **Font-StrikeOut** avec la valeur **true** permet de barrer le texte.

- **Font-Underline** avec la valeur **true** permet de souligner le texte.

- **ForeColor** est la couleur d'avant plan du texte.

- **Height** est la hauteur du composant exprimée par défaut en pixels (**px**) ou en pourcentage par rapport au container (**%**).

- **Style** est une chaîne de caractères comprenant plusieurs attributs de cette liste.

- **TabIndex** permet de spécifier l'ordre de passage des composants par l'utilisation des touches de tabulation.

- **ToolTip** contient la chaîne de caractères affichée dans la bulle d'aide.

- **Width** est la largeur du contrôle composant exprimée par défaut en pixels (**px**) ou en pourcentage par rapport au container (**%**).

Conclusion Tome 2

Nous avons ainsi fait un simple formulaire.

Une page peut bien entendu être entièrement dynamique par l'utilisation de tableau en mémoire, de fichiers XML ou encore d'ADO.NET (accès aux bases de données, ce qui nous permet de rendre les données persistantes), il est ainsi possible de charger des pages de manière dynamique.

Bonne continuation.