Les contrôles avancés en Windows Form

Sommaire

1 Introduction 3
2 Différents contrôles, différentes possibilités : les contrôles List-Based 3
2.1 Le contrôle *ListBox*, un objet nécessaire pour une liste 3
2.2 Le contrôle *CheckedListBox* 5
2.3 Le contrôle *ComboBox* 6
2.4 Le contrôle *ListView* 8
2.5 Le contrôle *TreeView* 10
3 Un contrôle à valeur numérique et un contrôle de chaîne de caractères : le *NumericUpDown* et le *DomainUpDown* 13
3.1 Le *NumericUpdown* 13
3.2 Le *DomainUpDown* 15
4 Les contrôles *Value-Setting* 16
4.1 Qu'est-ce que *Value-Setting* ? 16
4.2 La *CheckBox*, un choix multiple 17
4.3 La *RadioButton*, un choix unique 18
5 Gestion des Éléments dans un contrôle *List-Based* 19
5.1 Ajouter ou supprimer des éléments 19
5.2 Savoir gérer les éléments dans un contrôle *List-Based* 22
5.3 Le tri alphabétique 24
5.4 Mettre en place un format pour les éléments d'un contrôle *List-Based* 24
5.5 Déterminer des éléments sélectionnés grâce à deux méthodes 28
6 Le TackBar, une barre réglable utile 28
6.1 Présentation du *TrackBar* 28
6.2 Les propriétés du *TrackBar* 29
7 Les contrôles maîtres du temps 30
7.1 Le *MonthCalendar*, un calendrier facile à manipuler 30
7.2 Le *DateTimePicker*, un calendrier réduit 33
8 Utilisez la PictureBox ou l'ImageList 35

8.1 La *PictureBox* 35
8.2 L'*ImageList* 36
9 Le *WebBrowser*, le *NotifyIcon* et les raccourcis 37
9.1 Un navigateur, le *WebBrowser* 37
9.2 Le contrôle *NotifyIcon* 40
9.3 Définir des raccourcis 41
10 Conclusion 41

1 Introduction

Dans ce troisième chapitre, vous apprendrez à manipuler les différents contrôles qui permettent d'obtenir une liste d'affichage (*ListBox*, *CheckedListBox*,…). Ainsi vous pourrez permettre à l'utilisateur de sélectionner un ou plusieurs choix que vous avez attribué à votre programme. En utilisant ces contrôles vous donnerez en conséquence à l'utilisateur plus de possibilités et de choix de manipulation. Vous verrez aussi comment utiliser les dates et images sur votre *Formulaire*.

Bon cours .Net

L'équipe *Windows Form*

2 Différents contrôles, différentes possibilités : les contrôles List-Based

2.1 Le contrôle *ListBox*, un objet nécessaire pour une liste

Le contrôle ListBox permet d'afficher une liste d'éléments. Ils peuvent être récupérés à partir d'une base de données (une page XML par exemple) ou ils peuvent être ajoutés par le concepteur.

Il peut aussi permettre à un utilisateur de naviguer facilement et intuitivement dans l'application.

Il vous suffit simplement de connaître les différentes propriétés de la *ListBox* afin de mieux la manipuler. Voici les plus importantes d'entre elles :

Propriétés	Description
DataSource	Définit la source de données dans ce contrôle.
DisplayMember	Représente la source membre affichée dans ce contrôle.
FormatString	Spécifie le format d'affichage des valeurs (numérique scientifique,…).
FormattingEnabled	Si cette propriété est définie sur *True* alors la valeur attribuée à *FormatString* convertit la valeur de *DisplayMember*.
Items	Contient la collection d'éléments de la *ListBox*.
MultiColumn	Indique si les valeurs des éléments de la *ListBox* doivent être affichées dans plusieurs colonnes ou en un seul point.

SelectedIndex	Définit l'index (de base 0) de l'élément sélectionné dans la *ListBox*. Cependant si *SelectionMode* est défini sur *MultiSimple* ou *MultiExtended* alors n'importe quel élément sélectionné est retourné.
SelectedIndices	Retourne une collection des index sélectionnés dans le contrôle.
SelectedItem	Renvoie l'élément sélectionné. Cependant si *SelectionMode* est défini sur *MultiSimple* ou *MultiExtended* alors n'importe quel élément sélectionné est retourné.
SelectedItems	Retourne une collection des objets sélectionnés dans le contrôle.
SelectedValue	Si la valeur dans la propriété ValueMember n'est pas spécifiée, alors SelectedValue retourne les résultats selon la méthode ToString de l'objet.
SelectionMode	Détermine le nombre d'éléments qui peuvent être sélectionné dans la liste. L'option *MultiSimple* permet de sélectionner plusieurs éléments et *MultiExtended* permet d'utiliser en plus les touches Maj et Ctrl pour les sélectionner.
ValueMember	Indique la propriété à utiliser comme valeur réelle pour les éléments du contrôle.

Remarque : Les propriétés *SelectedItem*, *SelectedIndex* et *SelectedValue* se situent dans l'option *(DataBindings)* que vous devez développer pour les afficher.

2.2 Le contrôle *CheckedListBox*

Le contrôle *CheckedListBox* vous permet d'avoir une liste de différents éléments afin que l'utilisateur puisse sélectionner les objets selon son choix grâce au *CheckBox*. Ainsi l'utilisateur peut cocher autant qu'il souhaite, mais une seul sélection peut-être effectuée à la fois.

Types de Propriétés	Description
CheckedIndices	Renvoie une collection qui représente tous les indices « checkée ».
CheckedItems	Renvoie une collection qui expose tous les éléments « checké » dans ce contrôle.
FormatString	Détermine une chaîne de mise en forme qui va formater toutes les entrées si l'option *FormattingEnabled* est définie sur *True.*
FormattingEnabled	Indique si les entrées sont formatées en utilisant l'option *FormatString*.
Items	Renvoie à la collection d'éléments contenus dans le contrôle.
MultiColumn	Indique si ce contrôle montre plusieurs colonnes d'éléments ou un seul point d'éléments.
SelectedIndex	Représente l'index de la sélection des éléments, et si cette propriété est définie sur *MultiSimple* ou *MultiSelected*, alors elle retourne n'importe quel index de la sélection.
SelectedItem	Renvoie à la sélection des éléments. Si SelectionMode est défini sur MultiSimple ou MultiExtended alors n'importe quel élément sélectionné est retourné.

Vous pouvez avoir différents exemples au niveau du code, ci-dessous vous trouverez la méthode *SetItemCheckState* qui vous permet de définir les points « *checkés* » (donc ici cela sera le deuxième élément qui sera coché):

```
//C#

private void checkedListBoxTest_SelectedIndexChanged(object sender,
EventArgs e)

{
    checkedListBoxTest.SetItemCheckState(1, CheckState.Checked);
}
```

```
'VB

Private Sub CheckedListBoxTest_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
CheckedListBoxTest.SelectedIndexChanged

        CheckedListBoxTest.SetItemCheckState(1, CheckState.Checked)

End Sub
```

2.3 Le contrôle *ComboBox*

Le contrôle *ComboBox* est de même nature que la *ListBox* cependant on peut rajouter dans ses caractéristiques le fait qu'il prévoit une entrée pour que l'utilisateur puisse taper afin de sélectionner des éléments de son choix. Ainsi la *ComboBox* peut être configuré pour afficher une liste d'options ou encore procurer une liste déroulante de ces options.

Nous allons ainsi vous présenter les principales propriétés de ce contrôle :

Types de Propriétés	Description
DataSource	Définit la source de données obligatoire du contrôle.
DisplayMember	Représente la source membre affichée dans ce contrôle.
DropDownHeight	Fixe la hauteur maximale de la liste.
DropDownStyle	Détermine le style du contrôle.
DropDownWidth	Fixe la largeur de la liste.
FormatString	Spécifie le format d'affichage des valeurs (numérique scientifique,…).
FormattingEnabled	Si cette propriété est définit sur *True* alors la valeur attribué a *FormatString* convertit la valeur de *DisplayMember*.
Items	Définit la collection d'éléments du contrôle.
SelectedIndex	Représente l'index de la sélection des éléments et si cette propriété est définie sur *MultiSimple* ou *MultiSelected*, alors elle retourne n'importe quel index de la sélection.
SelectedItem	Renvoie à la sélection des éléments. Si *SelectionMode* est défini sur *MultiSimple* ou *MultiExtended* alors n'importe quel élément sélectionné est retourné.
SelectedValue	Si la valeur dans la propriété ValueMember n'est pas spécifiée, alors SelectedValue retourne les résultats selon la méthode ToString de l'objet.
ValueMember	Indique la propriété à utiliser comme valeur réelle pour les éléments du contrôle.

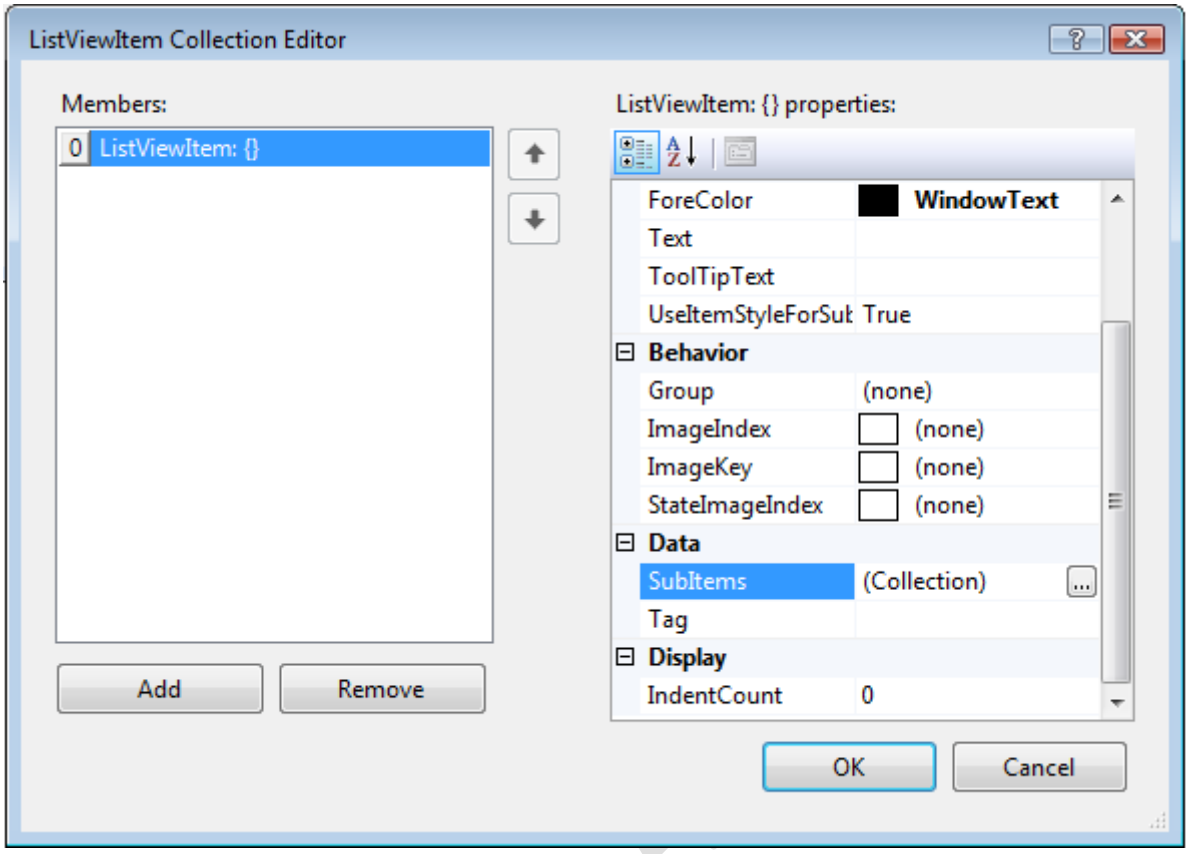
2.4 Le contrôle *ListView*

Le contrôle *ListView* permet d'avoir une liste d'élément avec des icônes de la même façon que Windows Explorer.

Voici les plus quelques propriétés de la *ListView* :

Propriétés	Description
Columns	Permet d'obtenir une collection d'en-tête de colonnes lorsque la propriété *View* est définit sur *Details*.
Groups	Contient une collection de groupes qui permet de classer les collections d'éléments qui seront présents dans la *ListView*.
Items	Charge une collection d'éléments qui seront affichés dans la *ListView*.
LargeImageList	Définit comment les images du contrôle *ImageList* seront affichés. Ici elles seront sous forme de grands icones.
ShowGroups	Détermine si le groupe contenu dans la collection *Groups* est visible.
SmallImageList	Définit comment les images du contrôle *ImageList* seront affichés. Ici elles seront sous forme de petits icones.
View	Indique de quelle façon la *ListView* sera affichée.

La propriété la plus importante de ce contrôle est la propriété *Items*. En effet, vous accédez à la collection d'éléments du contrôle puis grâce à cette propriété vous pouvez éditer ceux-ci. Par ailleurs lorsque vous avez sélectionné la propriété *Items* vous ouvrez une fenêtre appelée *ListViewItems Collection Editor* et vous personnalisez vos éléments en changeant les propriétés de cet *Editor* (*ListViewItems properties*).



Aperçu de l'éditeur des éléments du contrôle *ListView*.

Dans le tableau ci-joint vous trouverez les propriétés de l'éditeur les plus utilisées :

Propriétés	Description
Group	Détermine à quel groupe l'élément est assigné.
ImageIndex	Définit l'index de l'image attribué à l'élément. La valeur par défaut est -1 et cette propriété dépend de la propriété *ImageList*.
ImageKey	Définit la clé de l'image attribuée à l'élément. Cette propriété retourne une chaine vide (" ") si elle n'est pas définie ou si *ImageIndex* est définie.
SubItems	Détermine un sous-élément lorsque la propriété *View* est définie sur *Details*.
Text	Le texte qui apparaîtra dans la propriété *ListView*.

2.5 Le contrôle *TreeView*

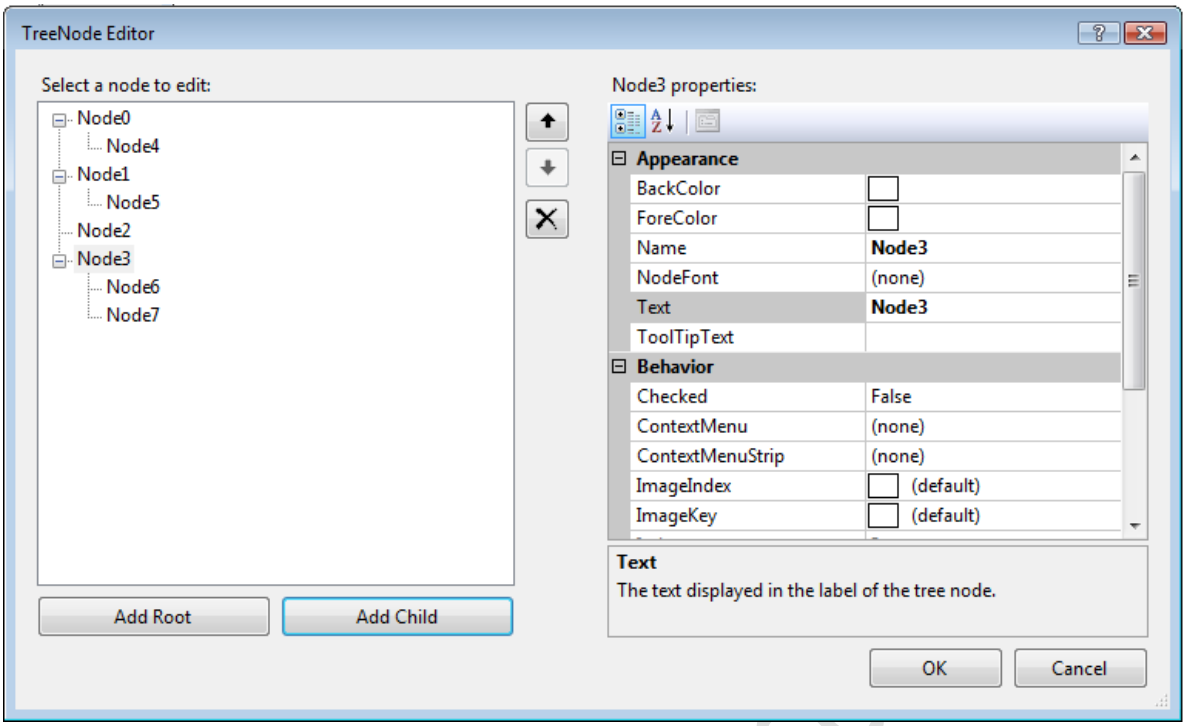
Ce contrôle vous permet d'afficher des objets ou des informations de façon hiérarchique en utilisant un système de nœuds (appelé *TreeNode*).

La principale propriété du contrôle *TreeView* est la propriété *Node*. Cette propriété contient une collection de *TreeNodes* qui constituent la racine du contrôle. Chaque objet *TreeNodes* contient des « nœuds enfants ». Voici les propriétés les plus importantes de la classe *TreeNode* :

Propriétés	Description
FirstNode	Renvoie au premier nœud issu du groupe des « nœuds enfants ».
LastNode	Renvoie au dernier nœud issu du groupe des « nœuds enfants ».
NextNode	Renvoie au nœud suivant dans l'arbre des noeuds.
NextVisibleNode	Renvoie au nœud visible suivant.
Nodes	Renvoie à la collection de nœuds enfants du nœud sélectionné.
Parent	Renvoie le nœud parent du nœud courant. Si le nœud courant est le nœud racine du *TreeView*, la propriété retournera à *NullReferenceException*.
PrevNode	Renvoie au nœud précédent.
PrevVisibleNode	Renvoie au nœud visible précédent.
TreeView	Renvoie une référence du contrôle TreeView auquel le TreeNode est assigné.

- Ajouter et supprimer des nœuds dans le contrôle *TreeView* :

Dans un premier temps vous pouvez ajouter ou supprimer des nœuds depuis l'interface graphique en sélectionnant la propriété *Node* du contrôle. Vous accédez ainsi à l'éditeur *TreeNode* et gérez votre arborescence.

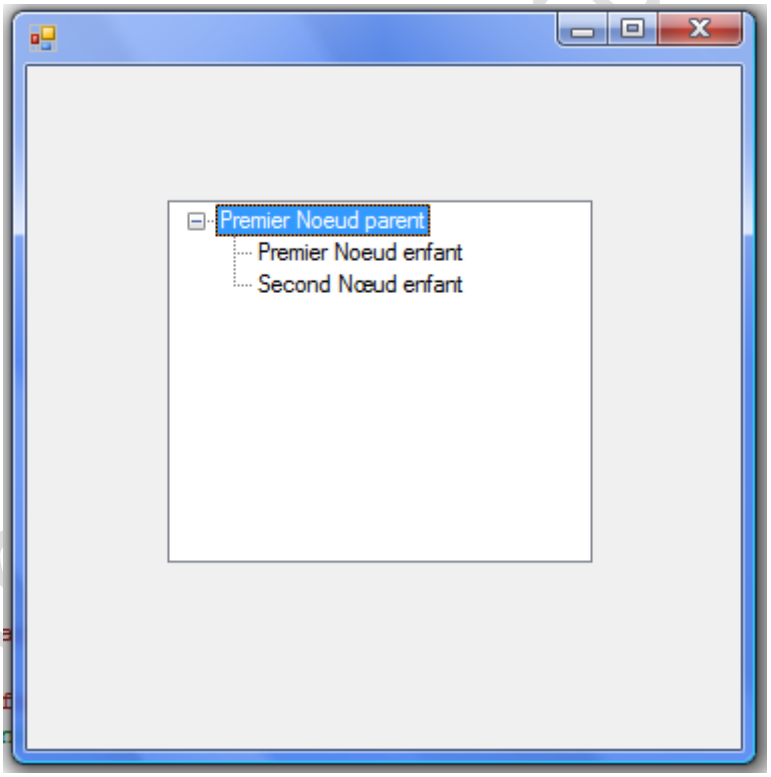


Dans un second temps, vous avez la possibilité de taper votre code afin d'ajouter ou supprimer vos nœuds avec :

- La méthode *Nodes.Add* (ajouter) :

```
'VB

'Création d'un noeud parent

Dim unNoeud As New TreeNode("Nouveau Noeud parent")

'Ajout d'un noeud enfant au noeud parent

unNoeud.Nodes.Add(New TreeNode("Nouveau Noeud enfant")

'Ajout d'un noeud et son enfant dans l'aborescence du controle
treeView nommé treeView1 ici

TreeView1.Nodes.Add(unNoeud)

'Ajout d'un second noeud enfant au premier noeud dans le
treeView1

TreeView1.Nodes(0).Nodes.Add(New TreeNode("Second Noeud enfant")
```

```
//C#

//Création d'un noeud parent

TreeNode unNoeud = new TreeNode("Premier Noeud parent");

//Ajout d'un noeud enfant au noeud parent

unNoeud.Nodes.Add(new TreeNode("Premier Noeud enfant"));

//Ajout d'un noeud et son enfant dans l'aborescence du controle
treeView nommé treeView1 ici

treeViewTest.Nodes.Add(unNoeud);

//Ajout d'un second noeud enfant au premier noeud dans le
treeView1

treeViewTest.Nodes[0].Nodes.Add(new TreeNode("Second Nœud
enfant")) ;
```



Voici le résultat du débbuging du code précédent

- La méthode Nodes.Remove (supprimer) :

```
'VB
'Supprime le noeud nommé unNoeud de la collection
TreeViewTest.Nodes.Remove(unNoeud)
'Supprime le noeud d'index 1 de la collection
TreeViewTest.Nodes.RemoveAt(1)
```

```
//C#
//Supprime le noeud nommé unNoeud de la collection
treeViewTest.Nodes.Remove(unNoeud);
//Supprime le noeud d'index 1 de la collection
treeViewTest.Nodes.RemoveAt(1);
```

3 Un contrôle à valeur numérique et un contrôle de chaîne de caractères : le *NumericUpDown* et le *DomainUpDown*

3.1 Le *NumericUpdown*

Le contrôle *NumericUpDown* est une boite contenant une valeur numérique. L'utilisateur choisit ainsi le nombre qu'il souhaite en cliquant sur les flèches.

Dans le tableau suivant vous retrouverez les propriétés les plus importantes du contrôle *NumericUpDown* :

Propriétés	Description
Hexadécimal	Définit la valeur indiquant si la zone de sélection numérique doit afficher cette valeur hexadécimale.
Increment	Définit la valeur pour incrémenter ou décrémenter la zone de sélection numérique lors d'un clic.
Maximum	Indique la valeur maximale du contrôle.
Minimum	Indique la valeur minimale du contrôle.
ThousandsSeparator	Détermine si un séparateur des milliers sera présent sur les valeurs supérieures à 1,000.
Value	Fixe la valeur actuelle du contrôle.

Voici un exemple de manipulation de ces propriétés :

```
'VB
Public Class Form1

  Public Sub New()

    InitializeComponent()

'On initialise la valeur qui sera affichée lors du débuggage

    NumericUpDownTest.Value = 5

  End Sub

Private Sub NumericUpDownTest_ValueChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
NumericUpDown1.ValueChanged

'On définit une valeur maximum pour le contrôle

        NumericUpDownTest.Maximum = 20

'On définit une valeur minimum pour le contrôle

        NumericUpDownTest.Minimum = -20

    End Sub

End Class
```

```
//C#

public Form1()

        {
            InitializeComponent();

            //On initialise la valeur qui sera affichée lors du
débuggage
            numericUpDownTest.Value = 10;
        }

        private void numericUpDownTest_ValueChanged(object
sender, EventArgs e)

        {
            //On définit une valeur maximum pour le contrôle

            numericUpDownTest.Maximum = 20;

            //On définit une valeur minimum pour le contrôle

            numericUpDownTest.Minimum = -20;
        }
```

3.2 Le *DomainUpDown*

Ce contrôle possède la même définition que le contrôle *NumericUpDown*. Cependant ce contrôle affiche des chaînes de caractères au lieu de valeurs numériques.

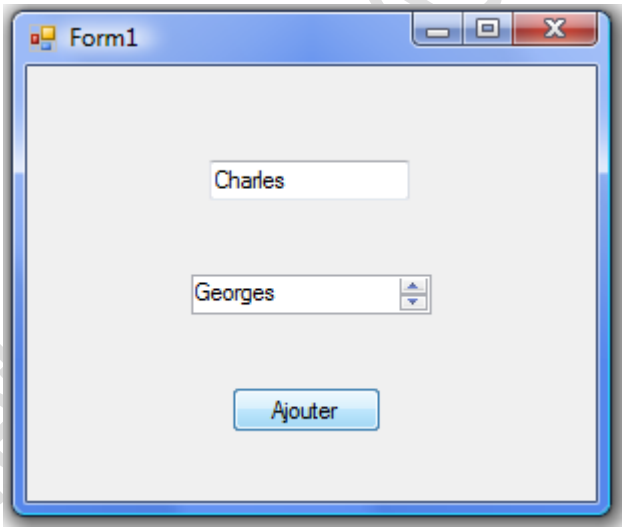
Voici quelques propriétés les plus utilisées et les plus importantes :

Propriétés	Description
Items	Contient les collections de chaîne qui s'affichera dans ce contrôle.
ReadOnly	Détermine si le texte du contrôle pourra être modifié par l'utilisateur.
Text	Obtient ou définit le texte du contrôle *DomainUpDown*.

Par exemple vous pouvez ajouter grâce à un bouton le texte issu d'une *TextBox* dans votre contrôle *DomainUpDown*, comme dans le code ci-dessous (nécessite d'insérer dans votre *Formulaire* un bouton, une *TextBox* et un *DomainUpDown*) :

```
'VB

Private Sub BtAjouter_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BtAjouter.Click

        DomainUpDownTest.Items.Add(TextBoxTest.Text)

End Sub
```

```
//C#

private void btAjouter_Click(object sender, EventArgs e)

        {
           domainUpDownTest.Items.Add(textBoxTest.Text);
        }
```



Les chaines de caractères issues de la TextBox sont ajoutées dans le contrôle DomainUpDown

4 Les contrôles *Value-Setting*

4.1 Qu'est-ce que *Value-Setting* ?

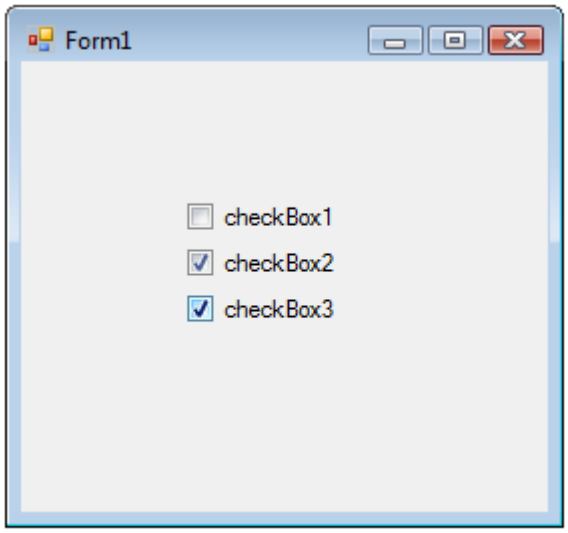
Le *Value-Setting* permet à un utilisateur d'inscrire des valeurs ou d'en choisir dans une liste déjà préconçue. Par exemple, la *CheckBox* permet à un utilisateur de sélectionner et désélectionner une ou plusieurs valeurs grâce à des cases.

4.2 La *CheckBox*, un choix multiple

La *ChekBox* permet aux utilisateurs de marquer une case à côté d'un nom pour indiquer qu'ils le sélectionnent ou pas. On peut avoir plusieurs *CheckBox* dans une même *Form*, elles peuvent être enregistrées en même temps ou petit à petit. Voici les propriétés principales d'une *CheckBox* :

Types de Propriétés	Description
AutoCheck	Permet de déterminer si la case est automatiquement cochée lorsque que l'utilisateur clic sur le texte.
Checked	Définit si la *CheckBox* est cochée.
CheckAlign	Définit la position de la case à cocher du contrôle.
CkeckState	Permet de définir si la case sera au départ cochée, vide ou indéterminée.
Text	Permet d'écrire le texte à côté de la case.

En général, on utilise la *CheckBox* pour demander des choix multiples à un utilisateur et des *RadioButton* pour lui demander qu'un seul choix.



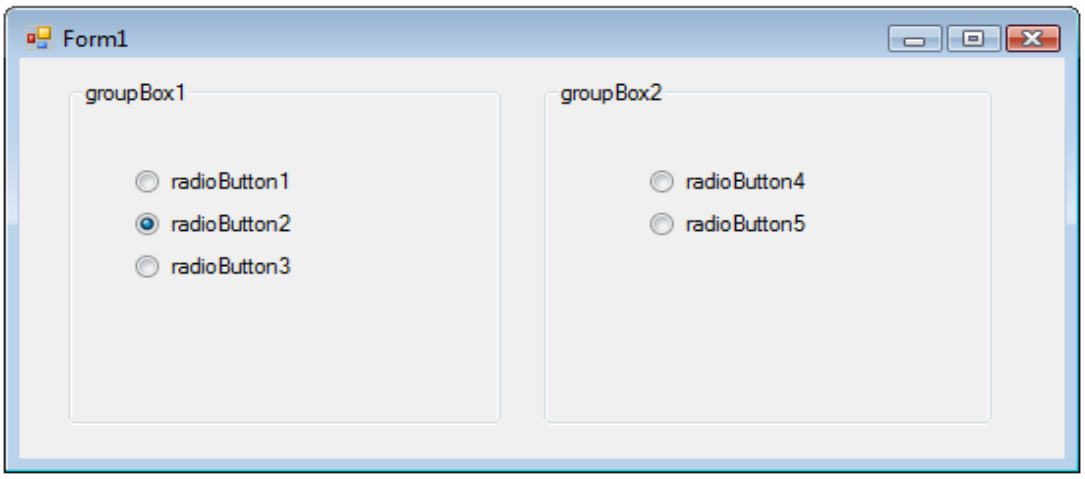
Remarque : Pour laisser à l'utilisateur un maximum de choix concernant vos *TextBox*, sélectionnez la valeur *True* à *ThreeState*. Cela lui permettra de pouvoir cocher, décocher ou mettre une valeur indéterminée.

4.3 La *RadioButton*, un choix unique

A la différence avec une *CheckBox*, le *RadioButton* ne permet à l'utilisateur de ne pouvoir cocher qu'un unique choix. D'où le sens pratique dans un projet d'allier les deux contrôles. Le contrôle *RadioButton* possède pratiquement les mêmes propriétés que son homologue, la *CheckBox*.

Types de Propriétés	Description
AutoChecked	Permet de déterminer si la case est automatiquement cochée lorsque que l'utilisateur clic sur le texte.
Checked	Cette option vous permet de définir si un *RadioButton* est sélectionné à la base ou pas.
CheckAlign	Définit la position de la case à cocher du contrôle.
Text	Permet d'écrire le texte à côté de votre *RadioButton*.

Attention, si vous voulez utiliser plusieurs groupes de *RadioButton* dans une même *Form*, le moyen le plus simple pour y parvenir est de créer plusieurs *GroupBox* dans lesquels vous insérerez vos *RadioButton*.

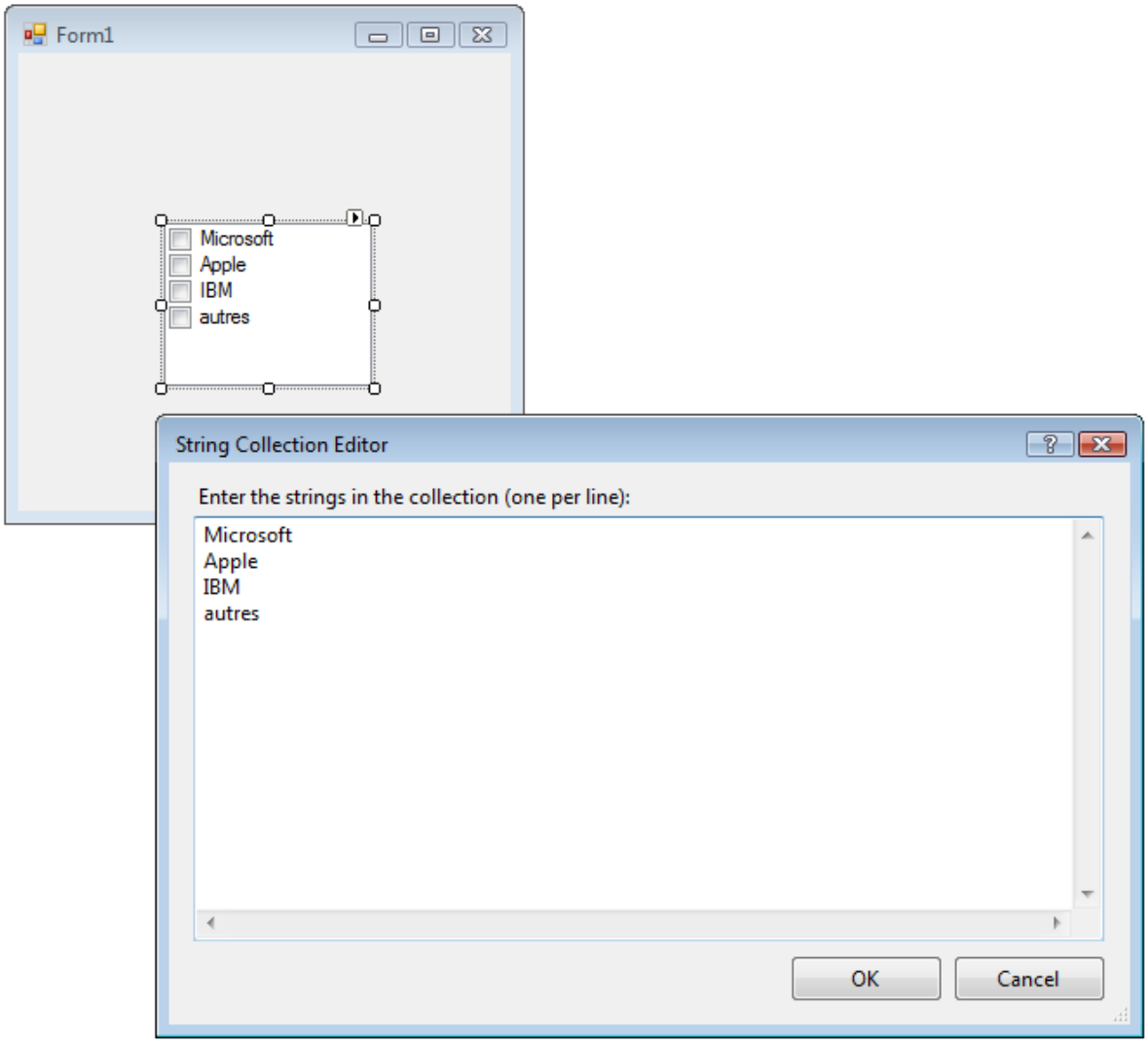


5 Gestion des Éléments dans un contrôle *List-Based*

5.1 Ajouter ou supprimer des éléments

Vous avez deux manières de manipuler les éléments (ou *Items*) d'un *List-Based Control* (*ListBox*, *CheckBox*…) : dans le code ou au moment de la conception.

- Pour ajouter ou supprimer des éléments dans une *List-Based Control* lors de la conception, vous devez tout d'abord sélectionner votre contrôle puis vous allez dans ses propriétés et accédez à la collection de la propriété *Items*. L'éditeur de la collection s'ouvre, ainsi vous pouvez rentrer ou effacer les éléments de votre choix.



Exemple d'éléments entrés grâce à l'éditeur de la collection d'*Items*.

- Vous pouvez aussi grâce au code de programmation gérer vos éléments du contrôle (ajouter et supprimer des chaînes de caractères depuis une *TextBox* par exemple) :

```
'VB

'Ajout d'éléments issus de la TextBox

Private Sub ButtonTest_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        ListBoxTest.Items.Add(TextBox1.Text)

    End Sub
```

```
//C#

//Ajout d'éléments issus de la TextBox

private void buttonTest_Click(object sender, EventArgs e)

        {
            listBoxTest.Items.Add(textBox1.Text);
        }
```

```
'VB

'Suppression d'éléments issus de la TextBox

Private Sub ButtonTest_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        ListBoxTest.Items.Remove(TextBox1.Text)

    End Sub
```

```
//C#

//Suppression d'éléments issus de la TextBox

private void buttonTest_Click(object sender, EventArgs e)

        {
            listBoxTest.Items.Remove(textBox1.Text);
        }
```

- Et si vous avez plusieurs éléments à ajouter vous utilisez alors la méthode *AddRange* à la place de la méthode *Add* :

```
'VB

Public Sub New()

InitializeComponent()

ListBoxTest.Items.AddRange(New String() {"Élément1", "
Élément2","Élément3"})

End Sub
```

```
//C#

public Form1()

{

InitializeComponent();

listBoxTest.Items.AddRange(new string[] {"Element1","Element2","Element3"
});

}
```

- Vous avez aussi la possibilité d'ajouter un élément possédant un index spécifique dans le contrôle *List-Based*. Cependant l'index d'éléments est basé sur zéro c'est-à-dire que le premier élément correspond à la valeur 0. Par exemple dans le code ci-dessous vous pouvez apercevoir que l'élément ajouté à une liste est placé en deuxième position :

```
'VB

ListBoxTest.Items.Insert(1, "cet élément est ajouter en deuxième
position")
```

```
//C#

listBoxTest.Items.Insert(1, "cet élément est ajouter en deuxième
position");
```

- Si vous voulez supprimer un élément présent dans une liste ayant une place particulière dans celle-ci sans pour autant détruire toute la liste, vous devez pour cela utiliser la méthode *RemoveAt* comme dans l'exemple suivant :

```
'VB

'Le deuxième élément de la liste est supprimé
ListBoxTest.Items.RemoveAt(1)
```

```
//C#

//Le deuxième élément de la liste est supprimé
listBoxTest.Items.RemoveAt(1) ;
```

- Ou bien si vous voulez supprimer tous les éléments d'un contrôle *List* vous taperez la méthode *Items.Clear*, comme ci-dessous :

```
'VB

ListBoxTest.Items.Clear()
```

```
//C#

listBoxTest.Items.Clear() ;
```

5.2 Savoir gérer les éléments dans un contrôle *List-Based*

- Grâce à la méthode *Items.IndexOf*, vous pourrez déterminer où apparaîtra un élément dans une *List*. Cette méthode permet de prendre un élément de votre List et de retourner une valeur qui représente l'indice de cet élément. Si cet élément n'est pas trouvé dans votre liste d'élément, il renverra la valeur -1. Voici la structure en code :

```
'VB

Dim anIndex As Integer

anIndex = ListBoxTest.Items.IndexOf("un élément")
```

```
//C#
int anIndex;
anIndex = listBoxTest.Items.IndexOf("un élément");
```

- Vous pouvez également, en utilisant la propriété *SelectedIndex*, définir l'index des éléments sélectionnés par l'utilisateur. Cette propriété renvoie l'élément que l'utilisateur à sélectionné au moment de l'exécution.

```
'VB
Dim anIndex As Integer
anIndex = ListBoxTest.SelectedIndex
```

```
//C#
int anIndex;
anIndex = listBoxTest.SelectedIndex;
```

- Lorsque la propriété *SelectionMode* des contrôles est définie sur *MultiSimple* ou *MultiExtended* vous pouvez renvoyer les index choisis dans votre contrôle grâce à la propriété *SelectedIndices* :

```
'VB
For Each l As Integer In ListBoxTest.SelectedIndices
    Console.WriteLine(ListBoxTest.Items(i).ToString)
Next
```

```
//C#
foreach (int l in listBoxTest.SelectedIndices)
     {
       Console.WriteLine(listBoxTest.Items[i].ToString());
     }
```

5.3 Le tri alphabétique

Vous pouvez classer les objets présents dans les contrôles List-*Based* en sélectionnant l'option *True* de la propriété *Sorted*, comme indiqué ci-dessous. Ceci a pour effet de trier les éléments de la liste selon l'ordre alphabétique.

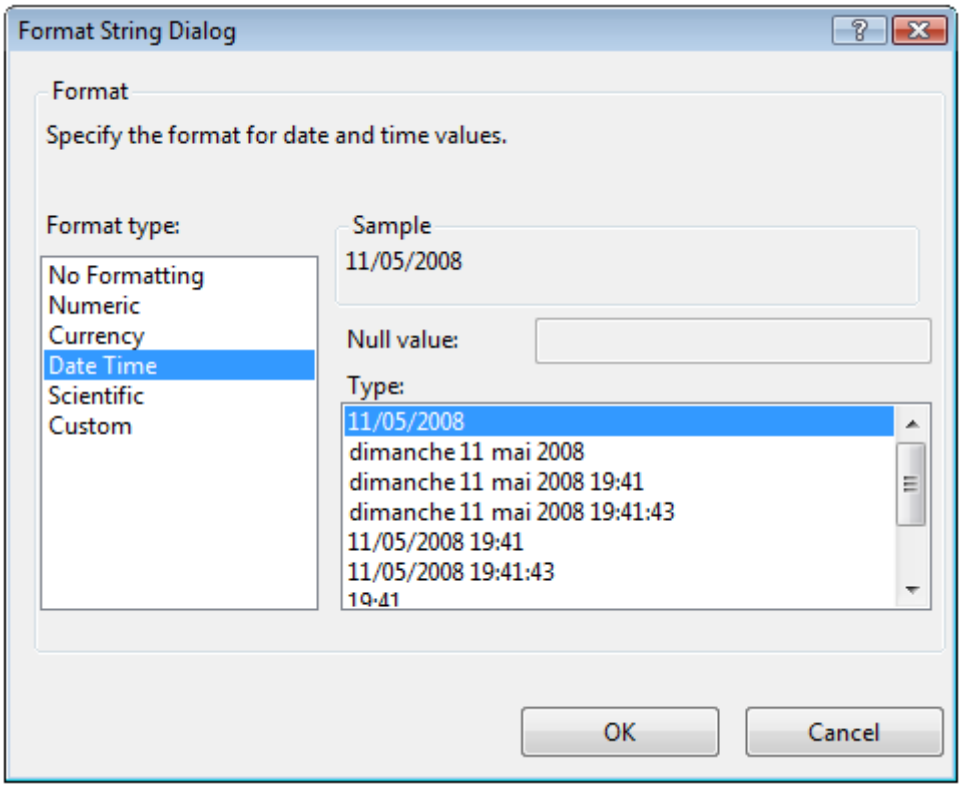
```
'VB

ListBoxTest.Sorted = True
```

```
//C#

listBoxTest.Sorted = true;
```

5.4 Mettre en place un format pour les éléments d'un contrôle *List-Based*

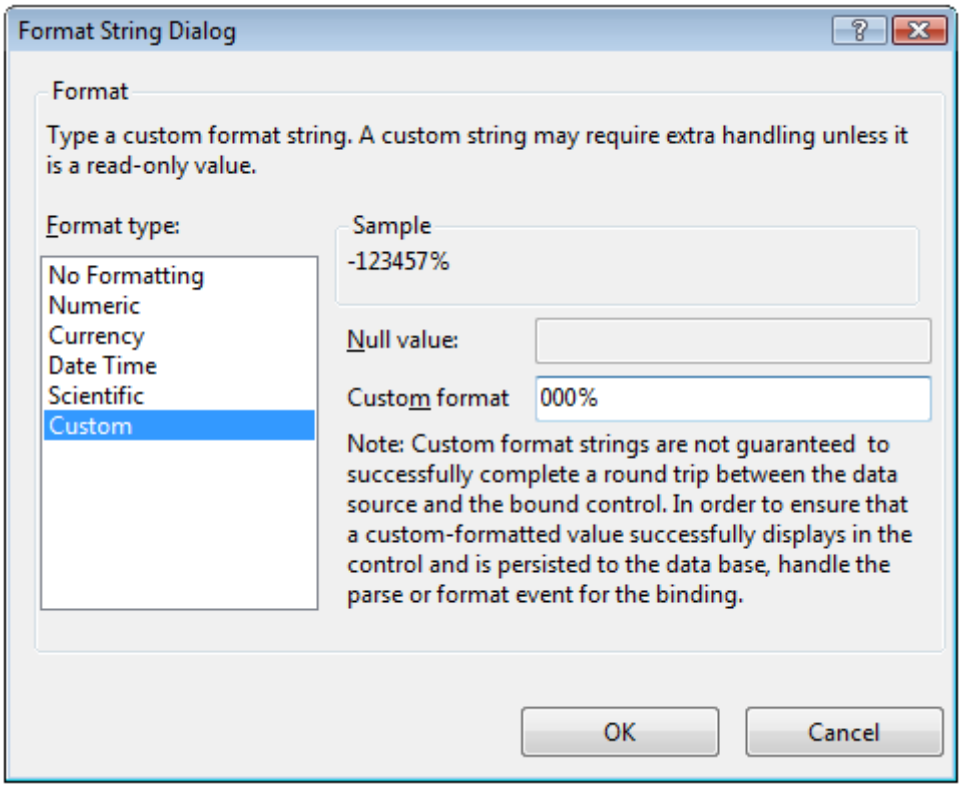
Vous avez la possibilité par l'intermédiaire de la propriété *FormatString* de choisir le format des éléments affichés dans votre contrôle. Ainsi vous pouvez choisir des valeurs monétaires ou encore une écriture scientifique.

Donc pour cela vous sélectionnez la propriété *FormatString* de votre contrôle (depuis l'interface utilisateur), puis vous choisissez votre format. Par défaut cette propriété est définie sur *No Formatting*.



Remarque : Par ailleurs, il faut déterminer la propriété *FormattingEnable* sur *True* afin que la propriété *FormatString* puisse convertir.

De plus l'option *Custom* vous permet de créer votre propre format dans le cas où les autres ne vous conviennent pas.



Le tableau suivant décrit les différents caractères qui peuvent être utilisés pour créer une chaîne de format personnalisé grâce à l'option *Custom*.

Types de caractères	Description
0	Lorsque vous attribuez une valeur, celle-ci prend l'emplacement du caractère « 0 ». Si aucune valeur n'est attribuée alors un « 0 » apparaît dans la chaîne. De plus le « 00 » impose un arrondissement du chiffre le plus proche de la virgule.
#	Ce caractère possède les mêmes caractéristiques que le caractère « 0 ». Cependant lorsqu'aucune valeur n'est attribuée il n'y a pas de valeur affichée. Le « ## » provoque aussi un arrondissement.
.	Le '.' permet de créer des nombres décimaux.
,	Ce caractère permet de séparer les nombres en milliers. Ce qui ajoute une autre vision du format des nombres.

	Par exemple 1000000 pourrait s'écrire grâce à la chaîne « #,# » 1, 000,000.
%	Le « % » multiplie la valeur par 100, on obtient ainsi une mise en forme du nombre en pourcentage.
E0, E+0, E-0, e0, e+0, e-0	Ces caractères mettent en place une notation scientifique. Ils sont suivit obligatoirement du caractère « 0 » et indiquent le nombre minimal d'exposants affichés. Les caractères « E+ » et « e+ » montrent qu'un signe (+ ou -) doit précéder l'exposant. Les caractères « E », « E-», « e » et « e- » montrent qu'un signe doit être affiché seulement pour les nombres négatifs.
\	C'est le caractère d'échappement qui entraîne une interprétation du caractère qui suit l'antislash « \ ». Si vous voulez introduire un antislash vous devrez taper « \\ » pour l'insérer.
''ABC'', 'ABC'	Vous pouvez ancrer des valeurs littérales entre « » ou ''.
;	Le « ; » permet de séparer en sections les nombres positifs, nuls et négatifs. La partie gauche détermine les nombres positifs ou nul et la partie droite détermine les nombres négatifs. Si votre chaine comprend 3 sections, la partie la plus gauche correspond aux positifs, celle du milieu aux nuls et la plus à droite aux négatifs.
Autres caractères	Les autres caractères dans les chaines sont copiés dans le résultat et n'affectent pas la mise en forme.

5.5 Déterminer des éléments sélectionnés grâce à deux méthodes

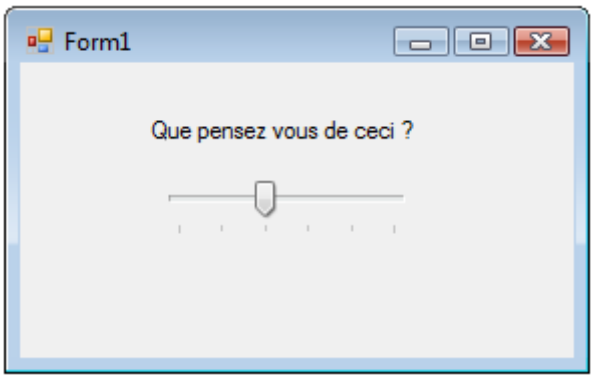
Pour déterminer des éléments sélectionnés ou des index depuis votre code vous utiliserez les méthodes respectives *SelectedItem* ou *SelectedIndex*.

Remarque : Si vous voulez avoir une collection d'éléments ou d'index il faut que (dans votre contrôle List) la propriété *SelectedMode* soit définie sur *MultiSimple* ou *MultiExtended*.

6 Le TackBar, une barre réglable utile

6.1 Présentation du *TrackBar*

Le *TrackBar* est une barre permettant de définir graphiquement une valeur. L'échantillon de valeurs est défini par le concepteur. Cette option permet par exemple de pouvoir donner une note à un service.

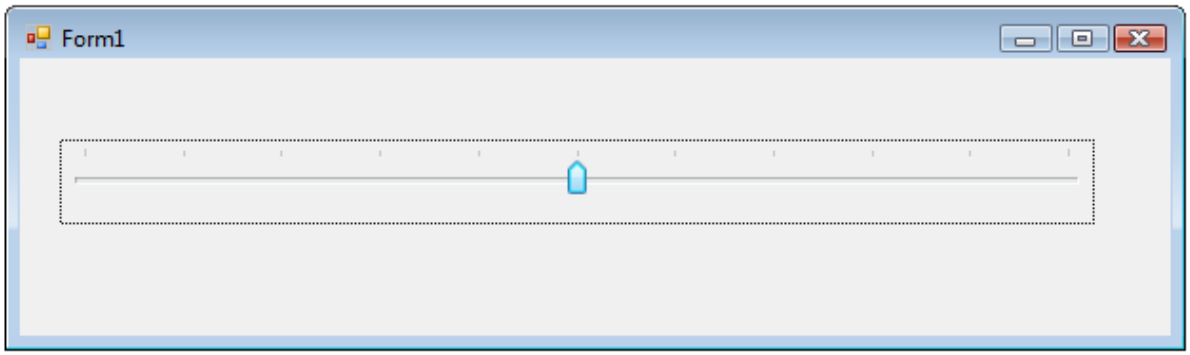


6.2 Les propriétés du *TrackBar*

Voici les principales propriétés de ce contrôle :

Types de Propriétés	Description
LargeChange	Il permet de définir de combien de rang augmente votre flèche en cliquant avec la souris. Par exemple si la valeur est 3, en cliquant avec votre souris, le curseur augmentera de 3 barres.
Maximum	Définit le nombre maximum de valeur sur votre barre.
Minimum	Définit le nombre minimum de valeur sur votre barre.
SmallChange	Il permet de définir de combien de rang augmente votre flèche en cliquant avec les touches de votre clavier. Par exemple si la valeur est 3, en tapant avec votre clavier, le curseur augmentera de 3 barres.
TickFrequency	Permet d'indiquer le nombre de position que le curseur peut prendre entre chaque rangée.
TickStyle	Permet d'indiquer comment s'affiche les *Ticks* (pointeurs) sur la *TrackBar*.
Value	Définit la valeur retournée par la *TrackBar*.

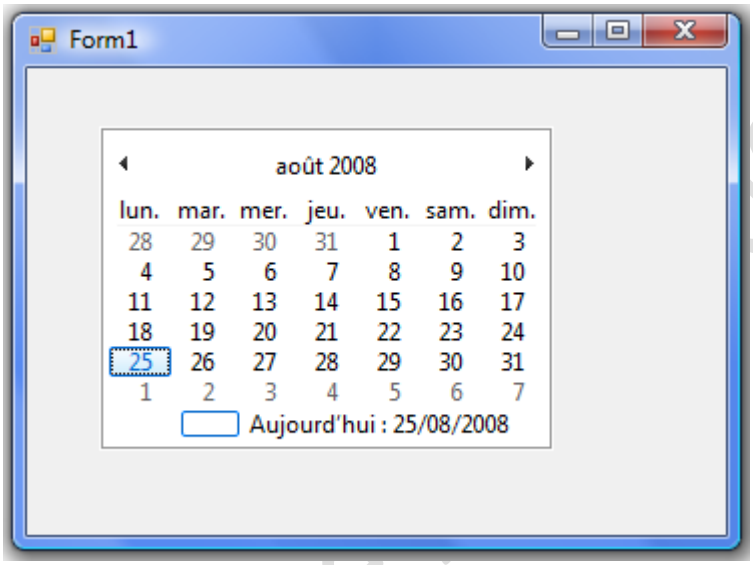
Vous pouvez donc modifier jusqu'au design de votre *TrackBar*. Par exemple avec le *TickStyle*, vous pouvez mettre les *Tick* au-dessus de la barre :



7 Les contrôles maîtres du temps

7.1 Le *MonthCalendar*, un calendrier facile à manipuler

Le *MonthCalendar* est un calendrier avec interface graphique très intuitif. De plus, il dispose d'un gros éventail de configuration possible.



Voici un aperçu du contrôle en question

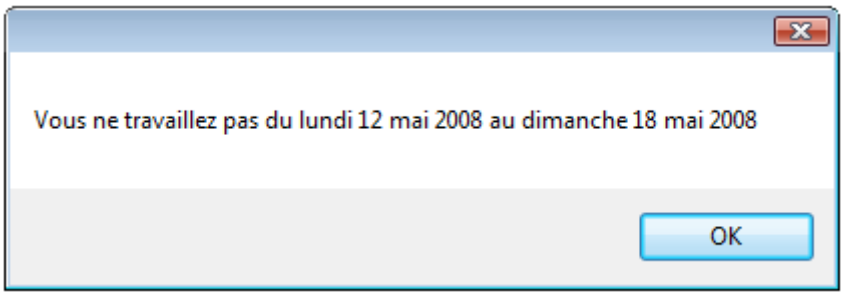
Voici les principales propriétés d'un *MonthCalendar* :

Types de Propriétés	Description
AnnuallyBoldedDates	Permet de définir une ou plusieurs dates qui apparaîtront en gras sur le calendrier chaque année.
BoldedDates	Permet de définir une ou plusieurs dates qui apparaîtront en gras sur le calendrier.
FirstDayOfWeek	Permet de définir le premier jour de la semaine affiché sur le calendrier, par défaut c'est le lundi.
MaxDate	Permet de définir la date maximum jusqu'à laquelle le calendrier peut aller.
MinDate	Permet de définir la date minimum jusqu'à laquelle le calendrier peut aller.
MaxSelectionCount	Permet de définir le nombre de jour maximum pouvant être sélectionnés dans le *MonthCalendar*.
MonthlyBoldedDates	Permet de définir une ou plusieurs dates qui apparaîtront en gras sur le calendrier chaque mois.
SelectionEnd	Définit le dernier jour de la sélection de votre *SelectionRange*.
SelectionRange	Par défaut votre *SelectionRange* sélectionnera le jour actuel mais vous pouvez définir une rangée de jour qui sera sélectionnée.
SelectionStart	Définit le premier jour de la sélection de votre *SelectionRange*.

L'utilisateur peut sélectionner une date en cliquant dessus ou en sélectionner plusieurs en maintenant la touche « Maj » enfoncée. En revanche, il ne pourra pas sélectionner plus de jours que le nombre autorisé par le *MaxSelectionCount*.

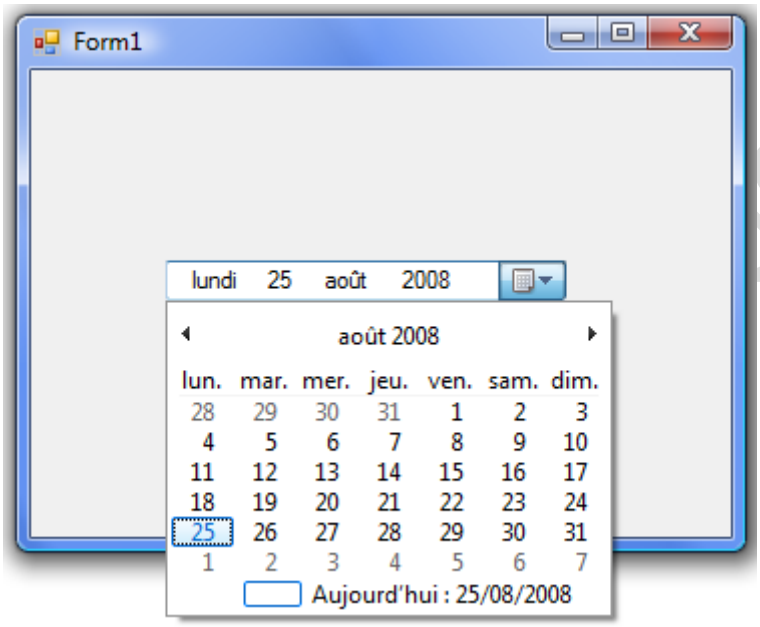
Vous pouvez aussi rappeler toute sorte de chose à l'utilisateur lors du lancement de l'application. Par exemple lui rappeler trois jours au cours desquels il ne travaille pas.

```
'VB

Message.Show("Vous ne travaillez pas du " &
monthCalendarTest.SelectionStart.ToLongDateString & " au " &
monthCalendarTest.SelectionEnd.ToLongDateString)

//C#

MessageBox.Show("Vous ne travaillez pas du " +
monthCalendarTest.SelectionStart.ToLongDateString() + " au " +
monthCalendarTest.SelectionEnd.ToLongDateString());
```



7.2 Le *DateTimePicker*, un calendrier réduit

Le *DateTimePicker* est un contrôle permettant d'afficher l'heure, le jour, le mois et l'année, c'est un calendrier possédant un menu déroulant affichant un calendrier général. Ceci permet à l'utilisateur d'avoir bien sûr le choix sur la date qu'il souhaite afficher mais surtout d'avoir accès simplement à un calendrier.



Voici en image le contrôle DateTimePicker sous forme développée

Remarque : Par défaut la valeur de la propriété *Value* est la date et heure de votre ordinateur.

Voici les principales propriétés de ce contrôle :

Types de Propriétés	Description
CustomFormat	Lorsque la valeur de la propriété Format est sur Custom, le texte affiché ne sera plus la date ou l'heure mais la valeur inscrite dans cette propriété.
Format	Définit le format de votre contrôle, c'est-à-dire de montrer la date ou l'heure par exemple.
MaxDate	Permet de définir la date maximum jusqu'à laquelle le calendrier peut aller.
MinDate	Permet de définir la date minimum jusqu'à laquelle le calendrier peut aller.
Value	Il s'agit de la valeur à laquelle votre *DateTimePicker* est situé au départ.

8 Utilisez la PictureBox ou l'ImageList

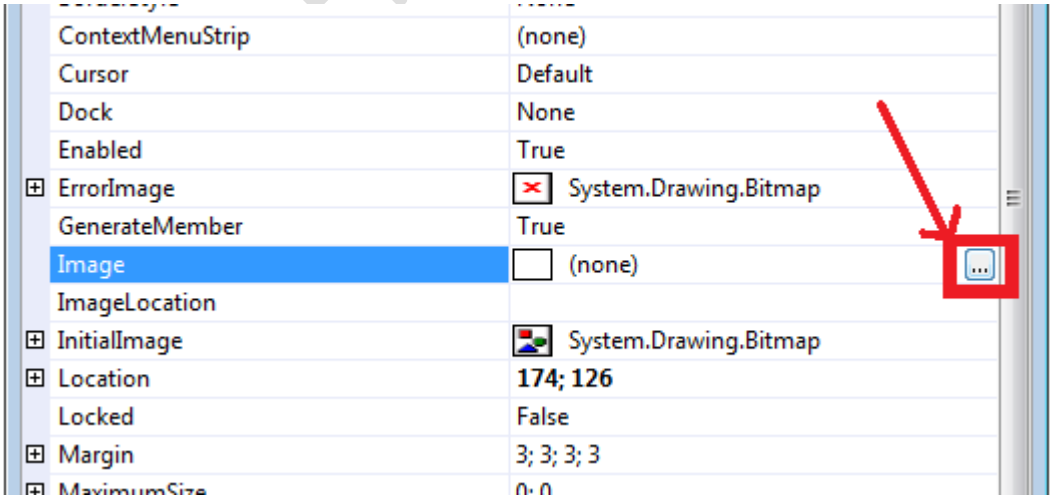
8.1 La *PictureBox*

La *PictureBox* est le contrôle le plus utilisé pour afficher des images. En effet, il permet d'utiliser un grand nombre de formats d'images (GIF, JPG, …). De plus, vous pouvez aussi charger des images d'un disque ou encore d'une page web.

Voici les principales propriétés de la *PictureBox* :

Types de Propriétés	Description
ErrorImage	Permet de choisir une image qui sera affichée si l'image demandée n'arrive pas à se charger.
Image	Permet de choisir l'image qui sera affichée dans la *PictureBox*.
ImageLocation	Permet de définir l'adresse sur le disque ou d'une page web afin d'y récupérer l'image.
InitialImage	Permet de définir une image qui sera affichée durant le chargement d'une autre image.
SizeMode	Permet de définir comment l'image sera affichée (redimensionné, centré, …)

Vous avez accès aux propriétés *Image* en cliquant sur l'icône ci-dessous :



C'est à partir de ce menu que vous pourrez récupérer et afficher vos images.

Vous pouvez aussi afficher une image directement en le tapant dans votre code, comme par exemple :

```
'VB

Dim yourImage As New System.Drawing.Bitmap("C:/Votre
adresse/yourImage.bmp")
PictureBoxTest.Image = yourImage

//C#

System.Drawing.Bitmap yourImage = new
System.Drawing.Bitmap(@"C:/VotreAdresse/yourImage.bmp");
pictureBoxTest.Image = yourImage;
```

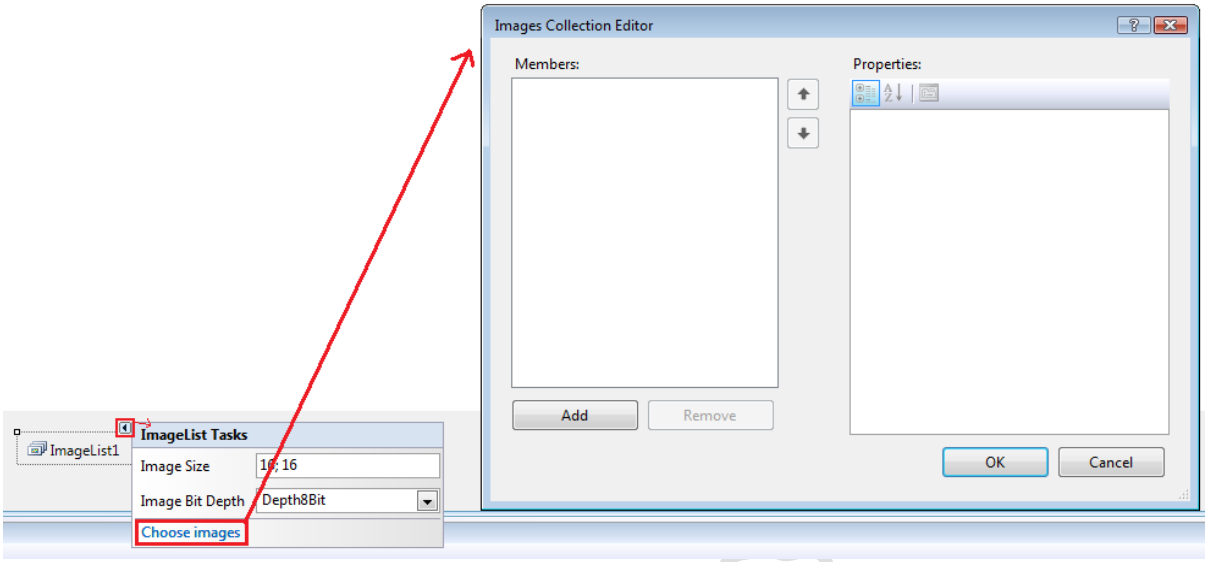
8.2 L'*ImageList*

L'*ImageList* n'est pas un contrôle comme les autres. En effet, c'est plus un gestionnaire d'Image. Ainsi l'*ImageList* vous permettra de fournir des images à d'autres contrôles. Les images contenues dedans pourront être gérées tout comme dans toute autre collection.

Voici les principales propriétés d'une *ImageList* :

Types de Propriétés	Description
ColorDepth	Permet de choisir la qualité des images, contenues dans votre *ImageList*, que vous souhaitez afficher.
Images	Il s'agit de la collection d'images gérées par l'*ImageList*.
ImageSize	Permet de définir la taille des images, contenues dans votre *ImageList*, que vous souhaitez afficher.

Pour ajouter des images à une *ImageList* (elle se situe en bas de votre page Form [Design]), faites comme si dessous :



Vous pouvez utiliser les images de votre *ImageList* directement dans votre code. Comme par exemple en utilisant une de ces images dans une *PictureBox* :

```
'VB

PictureBoxTest.Image = ImageListTest.Images(2) 'le 2 est le numéro associé
à l'image, ces numéros sont accessibles par le "Images Collection Editor"

//C#

pictureBoxTest.Image = imageListTest.Images(2) ; //le 2 est le numéro
associé à l'image, ces numéros sont accessibles par le "Images Collection
Editor"
```

9 Le *WebBrowser*, le *NotifyIcon* et les raccourcis

9.1 Un navigateur, le *WebBrowser*

Le contrôle *WebBrowser* permet d'accueillir toutes les fonctionnalités des pages Web (HTML) et d'autres types de fichiers et de documents. Grâce à ce contrôle vous pourrez afficher une aide en ligne pour votre programme ou télécharger des fichiers ou encore imprimer des documents en ligne ou bien même afficher les fichiers sous plusieurs formats.

Le *WebBrowser* possède beaucoup de propriétés, voici les plus importantes :

Propriétés	Description
AllowWebBrowserDrop	Détermine si les documents chargés dans le contrôle sont ouvert automatiquement.
CanGoBack	Indique si la page précédente est accessible ou non.
CanGoForward	Indique si la page suivante est accessible ou non.
Document	Obtient un document HTML qui s'affiche dans le *WebBrowser*.
DocumentStream	Renvoie les flux associés au document affiché dans le *WebBrowser*.
DocumentText	Retourne une chaîne représentant le document qui s'affiche dans le contrôle.
DocumentTitle	Définit le titre du document qui s'affiche dans le contrôle.
DocumentType	Définit le type de document qui s'affiche dans le contrôle.
IsOffline	Détermine si le contrôle travaille hors connexion.
IsWebBrowserContextMenuEnabled	Indique si le menu contextuel est activé pour le contrôle.
ScriptErrorsSupressed	Indique si les boîtes de dialogues d'erreur pour les erreurs de scripts sont affichées.
ScrollBarsEnabled	Détermine si la barre de défilement est activée ou non.
URL	Obtient ou définit l'URL (Uniform Resource Locator) pour le document qui s'affiche dans le contrôle
WebBrowserShortcutsEnabled	Détermine si les raccourcis claviers sont activés dans le contrôle.

De plus le contrôle *WebBrowser* contient une variété de méthodes :

Méthodes	Description
GoBack	Navigue à la page précédente dans l'historique de navigation s'il en existe une.
GoForward	Navigue à la page suivante dans l'historique de navigation s'il en existe une.
GoHome	Navigue à l'home page du navigateur.
GoSearch	Navigue à la page de recherche du navigateur.
Naviguate	Navigue à la page URL spécifiée.
Print	Imprime le document affiché dans le contrôle.
ShowPageSetupDialog	Affiche une boite de dialogue de mise en page d'Internet Explorer.
ShowPrintDialog	Affiche une boîte de dialogue d'impression d'Internet Explorer.
ShowPrintPreviewDialog	Affiche une boîte de dialogue d'aperçu avant impression d'Internet Explorer.
ShowPropertiesDialog	Affiche une boîte de dialogue dans Internet Explorer des propriétés du document affiché.
ShowSaveAsDialog	Affiche une boîte de dialogue dans Internet Explorer "Enregistrer la page web" ou "Enregistrer" si le document n'est pas une page HTML.
Stop	Stop toutes les pages dynamiques (animations, fond sonore…) ainsi que la navigation en cours.

Exemple de structure des méthodes du contrôle *WebBrowser* :

```
'VB

'renvoie à la page d'acceuil
WebBrowserTest.GoHome()
```

```
//C#

//renvoie à la page d'acceuil
webBrowserTest.GoHome();
```

Vous pouvez aussi charger des documents personnels :

```
'VB

WebBrowserTest.Navigate("C:\Documents\MonDocument.doc")


//C#

webBrowserTest.Navigate ("@C:\Documents\MonDocument.doc")
```

9.2 Le contrôle *NotifyIcon*

Ce contrôle permet de spécifier un élément qui crée une icône dans une zone de notification. Ce sont en quelque sorte des images en raccourcis permettant d'accéder aux processus qui s'exécutent en arrière plan comme un antivirus ou encore une mise à jour.

Ci-dessous vous trouverez quelques propriétés :

Propriétés	Description
BalloonTipIcon	Définit l'icône à associée à l'info-bulle de la NotifyIcon.
BalloonTipText	Définit le texte à associé à l'info-bulle de la NotifyIcon.
BalloonTipTitle	Définit le titre à associé à l'info-bulle de la NotifyIcon.
ContextMenuStrip	Obtient ou définit le menu contextuel associé à la NotifyIcon.
Icon	Indique l'icône affiché dans la barre d'état du système
Text	Définit le texte affiché lorsque le curseur de l'utilisateur reste sur l'icône.
Visible	Détermine si l'icône est visible dans la barre d'état du système.

9.3 Définir des raccourcis

Vous pouvez créer des raccourcis clavier pour permettre à l'utilisateur d'avoir une meilleure accessibilité aux différents contrôles de votre programme. Il lui suffira de presser la touche *Alt* et la clé que vous définissez :

➔ Allez dans la propriété *Text* de votre contrôle.

➔ Ensuite pour définir la clé vous entrez l'esperluette (&) avant la lettre qui servira de clé.

➔ Dans la fenêtre des propriétés, si elle est affichée la propriété *UseMnemonic* doit être définie sur *True* et la lettre précédant l'esperluette sera soulignée (qui sera la clé du raccourci).

Remarque : Vous pouvez aussi vous reportez au Chapitre 2 de *Windows Form* dans la partie explicative du *Label*.

10 Conclusion

Vous avez terminé ce 3ème chapitre de *Windows Form*. Pour obtenir plus de pratique nous vous conseillons de vous aider du site MSDN.

L'équipe *Windows Form.*