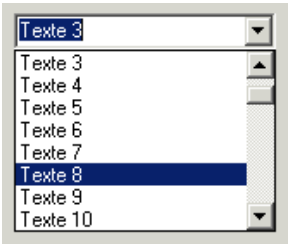
La liste de cases à cocher : `CheckedListBox`

Ce contrôle est une association des contrôles **`CheckBox`** et **`ListBox`**. Il hérite des membres de ses deux parents. Il n'accepte toutefois pas les sélections multiples.

La liste déroulante : `ComboBox`

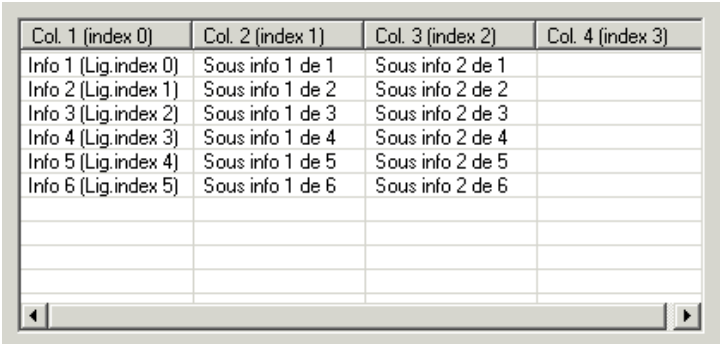


Le contrôle **`ComboBox`**, ou boîtes combinées, est l'association d'une **`ListBox`** et d'une **`TextBox`**. Il permet à l'utilisateur de sélectionner une valeur dans une liste ou de saisir une nouvelle valeur. Cependant, ce contrôle n'accepte pas les sélections multiples et ne déroule sa liste de données qu'à la demande.

Sa propriété **`DropDownStyle`** est à épingler. Ses valeurs possibles sont :

`DropDown`	Zone modifiable et déroulante (c'est la valeur par défaut)
`DropDownList`	Zone déroulante, mais non modifiable
`Simple`	Zone modifiable, mais non déroulante (balayage des valeurs par les flèches du clavier)

La liste de visualisation : `ListView`



Le contrôle **`ListView`** permet l'affichage d'une liste d'éléments avec, pour chacun, son texte et, éventuellement, une icône identifiant son type ou des informations filles le concernant et présentées en colonnes.
La classe **`ListViewItem`** représente un élément (avec ses informations filles éventuelles) du contrôle **`ListView`**. Les éléments de la liste peuvent être affichés de quatre manières différentes. Ils peuvent apparaître sous forme de grandes icônes (**`LargeImageList`**), de petites icônes (**`SmallImageList`**) ou de petites icônes dans une liste verticale (**`StateImageList`**). Les éléments peuvent aussi contenir des sous éléments comportant des informations relatives à l'élément parent. Le quatrième mode d'affichage, la vue **`Details`**, permet d'afficher l'élément et ses sous-éléments dans une grille munie d'en-têtes de colonne qui identifient les données affichées.
L'utilisateur peut réaliser des sélections par des clics sur la colonne d'indice **`0`** ou n'importe où sur la ligne selon la valeur affectée à la propriété **`FullRowSelect`**. Quelque soit le mode de sélection, il est possible d'obtenir toutes les informations de la ligne, mais à l'instar des volets de l'explorateur de Windows, seule celle de la colonne d'indice **`0`** est directement accessible.
Dans l'illustration proposée, les éléments sont contenus dans la colonne d'indice **`0`** et intitulée **`Col .1`**. Les autres colonnes contiennent des informations liées à l'élément de la même ligne.

Propriétés

`FullRowSelect`

Quand cette propriété à la valeur **`True`**, un clic sur n'importe quelle cellule de la ligne sélectionne toute la ligne. Si la valeur est **`False`**, une sélection ne peut s'opérer qu'à partir de la colonne d'indice **`0`** (comme dans l'explorateur).

`FocusedItem`

Cette propriété contient des informations de la ligne sélectionnée.

`FocusedItem.Index`	Indice de la ligne sélectionnée
`FocusedItem.Text`	Contenu de la ligne sélectionnée en colonne d'indice **`0`**
`FocusedItem.SubItems`	Collection des informations de la ligne
`FocusedItem.SubItems.Count`	Nombre de colonnes de la ligne
`FocusedItem.SubItems(n).Text`	Contenu de la ligne sélectionnée en colonne d'indice **`n`**

CheckBoxes

Quand cette propriété a la valeur **True**, les informations de la colonne d'indice **0** sont assorties de cases à cocher.

CheckedIndices et SelectedIndices

Ces propriétés contiennent respectivement la collection des indices des lignes cochées (si **CheckBoxes** vaut **True**) et celle des indices des lignes sélectionnées. Comme toutes les collections, elles disposent des *sous* propriétés **Count** et **Item**. Une ligne cochée n'est pas une ligne sélectionnée et une ligne sélectionnée n'est pas une ligne cochée.

CheckedItems et SelectedItems

Ces propriétés contiennent respectivement la collection des lignes cochées (si **CheckBoxes** vaut **True**) et celle des lignes sélectionnées. Comme toutes les collections, elles disposent des *sous* propriétés **Count** et **Item**.

View

C'est par l'affectation de cette propriété que se définit le mode d'affichage souhaité pour la **ListView** : **Details**, **LargeImageList**, **SmallImageList** et **StateImageList**.

Columns

En mode **Details**, cette propriété désigne la collection des colonnes de la liste. La largeur de chaque colonne se définit en paramètre de la méthode **Add** qui la crée. Il est toutefois possible de modifier la largeur d'une colonne par sa propriété **Width**. Il est ainsi possible d'affecter la largeur **0** à la colonne d'indice **0**. Cette faculté peut être utile par exemple, pour cacher l'identifiant de l'information présentée.

HeaderStyle

La détermination du type d'entête de colonne s'effectue par l'affectation de cette propriété par une des valeurs suivantes :

ColumnHeaderStyle.Clickable	Le clic de l'entête de colonne provoque l'événement **ColumnClick**
ColumnHeaderStyle.Nonclickable	Le clic de l'entête de colonne ne provoque pas d'événement
ColumnHeaderStyle.None	Pas d'entête de colonne

AllowColumnReorder

Quand cette propriété a la valeur **True**, l'utilisateur peut déplacer les colonnes de sorte à les placer dans un ordre différent. Cette manœuvre n'affecte que l'apparence de la **ListView**. Les indices de colonnes ne sont pas modifiés.

LabelEdit

Selon sa valeur **True** ou **False**, cette propriété autorise ou interdit la modification d'un élément par l'utilisateur. Seul le contenu de la colonne d'indice **0** peut être modifié.

SubItems

La collection des informations d'une ligne n'est pas une propriété de **ListView**, mais bien celle d'un objet de type **ListViewItem**, comme **FocusedItem** par exemple. C'est par l'ajout d'éléments à cette propriété que sont créés les champs d'informations liés au parent, c'est-à-dire l'élément de la colonne d'indice **0**.

Sorting

La valeur de cette propriété détermine la façon dont les informations peuvent être triés sur la colonne d'indice **0**.

SortOrder.Ascending	Tri dans l'ordre croissant
SortOrder.Descending	Tri dans l'ordre décroissant
SortOrder.None	Pas de tri

Méthodes

Il n'y a guère de méthodes spécifiques à la **ListView** qui n'aient déjà été vues. Par contre, quelques méthodes accessibles par des membres sont indispensables.

Columns.Add

L'ajout de colonnes est réalisé par cette méthode. Sa syntaxe est la suivante :

```
MaListView.Columns.Add("Libellé", Largeur, Alignement)
```

Le paramètre **Alignement** peut prendre une des valeurs **HorizontalAlignment.Center**, **HorizontalAlignment.Left** et **HorizontalAlignment.Right.**

Items.Add

Cette méthode réalise l'ajout d'éléments, c'est-à-dire d'information en colonne d'indice **0**. Elle crée les lignes. Sa syntaxe est :

```
MaListView.Items.Add("Information")
```

Outre de créer la ligne et d'y placer l'information donnée, la méthode retourne un objet de type **ListViewItem**. C'est par la récupération de cet objet et de sa collection **SubItems** qu'il est ensuite possible d'ajouter les informations filles, en principe autant qu'il y a de colonnes dans la **ListView**.

```
Dim UneLigne As ListViewItem              ' Déclaration d'une variable de type ListViewItem
UneLigne = MaListView.Items.Add("Info 1") ' Création de la ligne avec la donnée de colonne 0
UneLigne.SubItems.Add("Sous info 1 de 1") ' Crée un sous élément avec la donnée de colonne 1
UneLigne.SubItems.Add("Sous info 2 de 1") ' Crée un sous élément avec la donnée de colonne 2
```

Evénements

ColumnClick

Cet événement est déclenché par le clic de l'entête d'une colonne.

SelectedIndexChanged

L'événement se produit lors de chaque changement de sélection. Il est toujours suivi de l'événement **Click** de la **ListView**.

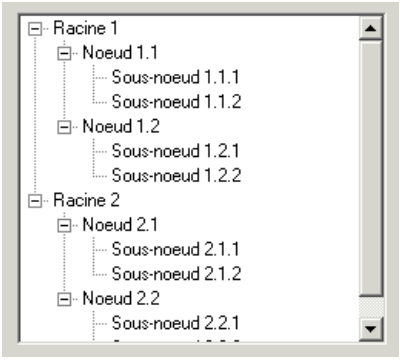
Quelques exemples

```
MaListV.HeaderStyle = ColumnHeaderStyle.Clickable     ' Type d'entête de colonne
MaListV.View = View.Details                           ' Mode d'affichage
MaListV.Clear()                                       ' Vide la liste
MaListV.Sorting = SortOrder.Ascending                 ' Mode de tri

' Création de 4 colonnes de largeur 100 avec intitulé : Col.0 … Col.3

For j As Integer = 0 To 3
  MaListV.Columns.Add("Col." & j, 100, HorizontalAlignment.Left)
Next j

' Création de 6 lignes, chacune munie de 2 sous éléments

Dim UneLigne As ListViewItem
For i As Integer = 0 To 5
  UneLigne = MaListV.Items.Add("Info " & i)
  UneLigne.SubItems.Add("Sous info 1 de " & i)
  UneLigne.SubItems.Add("Sous info 2 de " & i)
Next i
```

```
' Affectation de la largeur 0 à la colonne d'indice 0

MaListV.Columns(0).Width = 0

' Affiche l'indice de la ligne qui a le focus

MessageBox.Show(MaListV.FocusedItem.Index)

' Affiche le texte de la colonne 0 de la ligne qui a le focus

MessageBox.Show(MaListV.FocusedItem.Text)

' Affiche le nombre de cellules de la ligne qui a le focus

MessageBox.Show(MaListV.FocusedItem.SubItems.Count)

' Affiche tous les textes situés en colonne d'indice 0

Dim x As ListViewItem
For Each x In MaListV.Items
    MessageBox.Show(x.Text)
Next

' Affiche le texte de chaque cellule de la ligne qui a le focus

For i As Integer = 0 To MaListV.FocusedItem.SubItems.Count - 1
  MessageBox.Show(MaListV.FocusedItem.SubItems(i).Text)
Next i

' Affiche aussi tous les textes de la ligne qui a le focus

Dim x As ListViewItem.ListViewSubItem
For Each x In MaListV.FocusedItem.SubItems
    MessageBox.Show(x.Text)
Next

' Marque la ligne qui a le focus comme étant sélectionnée

MaListV.FocusedItem.Selected = True

' Affiche tous les indices de lignes cochées

For Each i As Integer In MaListV.CheckedIndices
    MessageBox.Show(i)
Next i

' Affiche tous les textes de la colonne d'indice 0 des lignes cochées

For i As Integer = 0 To MaListV.CheckedIndices.Count - 1
    MessageBox.Show(MaListV.CheckedItems.Item(i).Text)
Next i

' Affiche tous les indices de lignes sélectionnées

For Each i As Integer In MaListV.SelectedIndices
    MessageBox.Show(i)
Next i

' Affiche tous les textes de la colonne d'indice 0 des lignes sélectionnées

For i As Integer = 0 To MaListV.SelectedIndices.Count - 1
    MessageBox.Show(MaListV.SelectedItems.Item(i).Text)
Next i
```

```
' Affiche l'indice de la colonne cliquée (sur l'entête)

Private Sub MaListV_ColumnClick(ByVal sender As Object, ByVal e As
            System.Windows.Forms.ColumnClickEventArgs) Handles MaListV.ColumnClick
   MessageBox.Show(e.Column)
End Sub
```

Les vues en arborescence : `TreeView`



Le **`TreeView`**, qui s'apparente un peu à la **`ListView`**, présente toutefois une énorme différence. Alors que dans la **`ListView`**, chaque élément peut être assorti de plusieurs informations filles sans distinction hiérarchique entre elles, dans le **`TreeView`**, chaque élément appelé *node* peut être assorti de **`Nodes`** fils qui lui sont subordonnés et qui sont aussi des **`Nodes`** à part entière. Le premier nœud d'une arborescence est aussi appelé *racine*.

Propriétés

`Nodes`

Cette propriété contient la collection des **`Nodes`**.

`SelectedNode`

La propriété **`SelectedNode`** contient les informations du nœud sélectionné.

`ShowPlusMinus`

Selon sa valeur **`True`** ou **`False`**, cette propriété détermine si les boutons **`+/-`** sont affichés ou pas. Quand ils ne sont pas affichés, les développements ou réductions d'arborescences restent possibles par double-clic.

`ShowLines`

Quand cette propriété a la valeur **`True`**, soit sa valeur par défaut, l'affichage des lignes joignant les différents nœuds est effectué.

`ShowRootLines`

Quand cette propriété a la valeur **`True`**, soit sa valeur par défaut, et que **`ShowLines`** autorise l'affichage des lignes joignant les différents nœuds, la ligne joignant les nœuds racines est également affichée. Si **`ShowLines`** a la valeur **`True`** et que **`ShowRootLines`** a la valeur **`False`**, seules les lignes joignant les nœuds secondaires sont affichées.

Méthodes

Comme pour la **`ListView`**, les méthodes les plus utiles du **`TreeView`** se trouvent au niveau de certains de ses membres. Il faut toutefois épingler **`ExpandAll`** et **`CollapseAll`** qui sont des méthodes de **`TreeView`**, et dont les *cousines* sont des méthodes de **`TreeView.Nodes`**.

`ExpandAll`

Cette méthode provoque le développement de toute l'arborescence, comme si chacun des boutons **`+`** avait été cliqué.

`CollapseAll`

Cette méthode provoque la réduction de toute l'arborescence, comme si chacun des boutons **`-`** avait été cliqué.

`SelectedNode.Expand`, `SelectedNode.ExpandAll`, `SelectedNode.Collapse` et `SelectedNode.Toggle`

Ces méthodes, qui s'appliquent à un nœud sélectionné, développent ou réduisent l'arborescence à partir de la sélection.

`SelectedNode.Expand`	Développement d'un sous niveau
`SelectedNode.ExpandAll`	Développement de tous les sous niveaux
`SelectedNode.Collapse`	Réduction de tous les sous niveaux
`SelectedNode.Toggle`	Passage à l'état suivant (développé ou réduit)

`Nodes.Add`

Cette méthode réalise l'ajout de nœuds et retourne un objet de type **`TreeNode`**. L'exécution de **`Nodes.Add`** sur l'objet retourné crée un sous niveau de l'arborescence. Il n'est pas nécessaire de récupérer l'objet retourné par la méthode lorsqu'on crée le nœud de dernier niveau. Le code suivant crée la **`TreeView`** présentée en illustration ci-dessous.

```
Dim N1, N2 As TreeNode
For i As Integer = 1 To 2                       ' Deux racines
    N1 = MaTreeView.Nodes.Add("Racine " & i)
    For j As Integer = 1 To 2                   ' Deux premiers sous niveaux
        N2 = N1.Nodes.Add("Noeud " & i & "." & j)
        For k As Integer = 1 To 3               ' Trois seconds sous niveaux
            N2.Nodes.Add("Sous-noeud " & i & "." & j & "." & k)
        Next k
    Next j
Next i
```

Evénément

Le seul événement retenu ici est **`NodeMouseClick`** : il donne accès toutes les propriétés d'un nœud cliqué.

```
Private Sub MaTreeView_NodeMouseClick(ByVal sender As Object,
                      ByVal e As System.Windows.Forms.TreeNodeMouseClickEventArgs)
                                          Handles MaTreeView.NodeMouseClick
    MessageBox.Show(e.Node.Text)                ' Donne le texte du nœud cliqué
End Sub
```

Lecture d'un TreeView

La procédure récursive suivante réalise la lecture de tous les nœuds enfants, à partir d'un nœud d'entrée donné, et copie leur donnée dans une **`ListBox`** nommée **`LBNodes`**.

```
Private Sub AfficheNodes(ByVal Entree As TreeNode)
    LBNodes.Items.Add(Entree.Text)              ' Copie des données dans une ListBox
    If Entree.GetNodeCount(True) > 0 Then
        For Each N As TreeNode In Entree.Nodes
            AfficheNodes(N)
        Next
    End If
End Sub
```

Pour faire une lecture complète du **`TreeView`**, il faut appeler la procédure pour chaque racine. Le remplissage de la **`ListBox`** illustré ci-contre, avec toutes les données du **`TreeView`**, est le résultat de l'exécution des deux lignes suivantes :

```
AfficheNodes(MaTreeView.Nodes(0))
AfficheNodes(MaTreeView.Nodes(1))
```