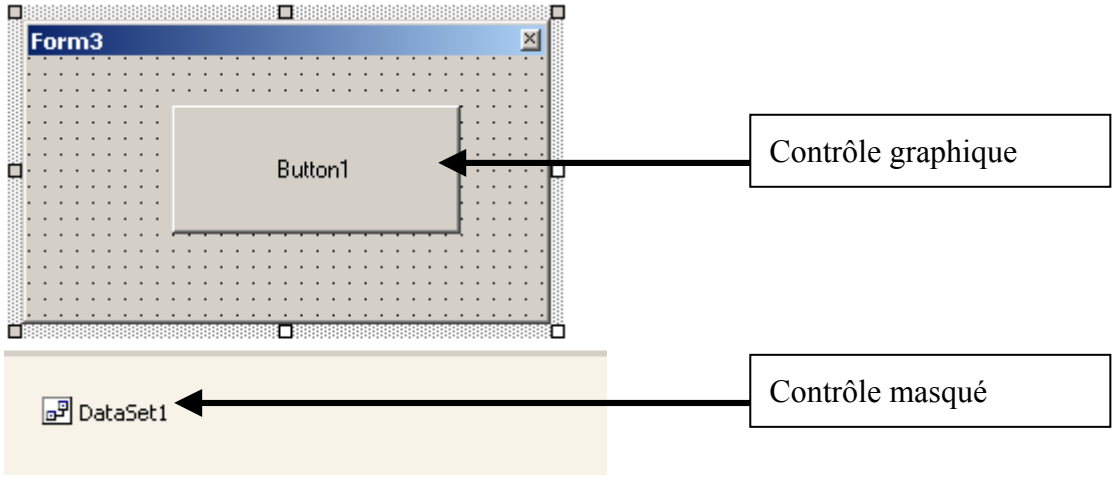
Les contrôles en VB.NET

Les contrôles permettent de créer l'interface entre l'utilisateur et notre application. C'est via les contrôles que l'utilisateur pourra saisir des données, effectuer des sélections et déclencher des actions par l'intermédiaire des événements.

De manière générale, les contrôles sont des objets graphiques, c'est à dire qu'il seront placés et visibles sur le formulaire. Cependant, certains contrôles offrant des fonctionnalités de programmation n'apparaîtront pas sur le formulaire mais dans une zone située en bas et uniquement en mode « Design ».



2.1 Membres communs

Les contrôles Visual Basic .Net sont des classes issues de la classe de base « control ». Cette dernière assure les fonctions élémentaires comme le placement sur une feuille, leur position … A cette classe est ajoutée une classe dérivée permettant la personnalisation des différents contrôles.

2.1.1 propriétés

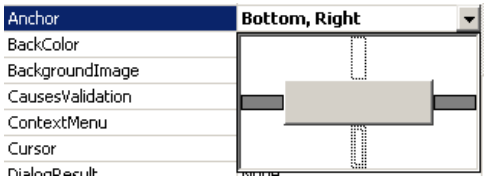
- Name

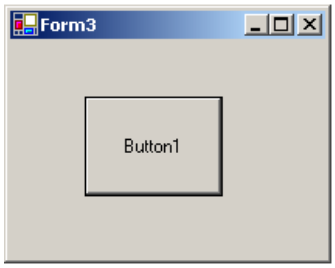
Nom du contrôle. Ce nom ne comporter que des lettres et le caractère underscore « _ ».

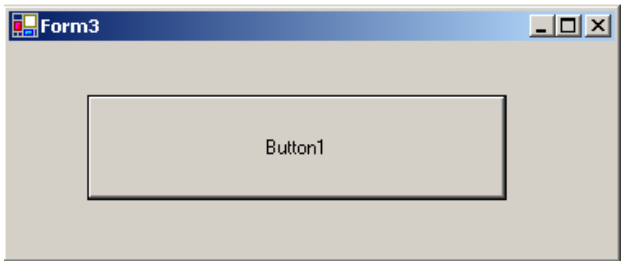
- Anchor

Les ancres permettent de modifier automatiquement la taille d'un contrôle lors du redimensionnement d'un formulaire. Chaque contrôle possède sa propre ancre.

Lors du paramétrage, vous devez définir sur quels bords du conteneur est ancré le contrôle. Dans l'exemple suivant, nous créons un contrôle ancré à gauche et à droite :







- CanFocus

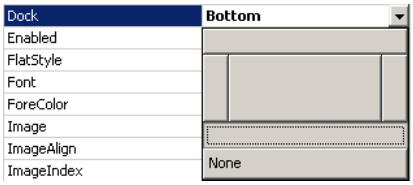
Booléen spécifiant si le contrôle peut recevoir le focus.

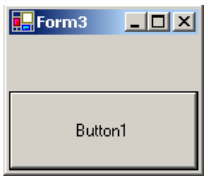
- CanSelect

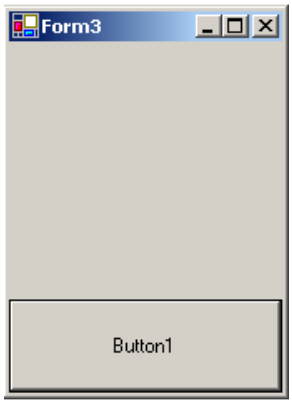
Booléen spécifiant si le contrôle peut être sélectionné.

- Dock

Dans le même esprit, la propriété « Dock » permet d'ancrer un contrôle aux à un bord du conteneur. Dans l'exemple suivant, le bouton est ancré en bas :







- Enabled

Cette propriété est une valeur booléenne spécifiant si le contrôle est accessible ou non. Dans le second cas, le contrôle apparaîtra grisé.

- Location

La propriété Location est un objet permettant de définir l'emplacement du contrôle par rapport à son conteneur. Il est composé de deux propriétés (X et Y) qui définissent ses coordonnées par rapport au coin supérieur gauche du conteneur.

- Locked

Contrairement à la version précédente, cette propriété ne bloque pas le contrôle lors de l'exécution mais lors de la conception. Il permet d'éviter de modifier les propriétés d'un contrôle.

- Modifiers

Cette propriété paramètre la visibilité au niveau programmation de l'objet. Elle peut prendre les valeurs suivantes :

Valeur	Description
Public	Accessible à partir de tous les éléments de la solution
Protected	Accessible à partir des membres de la classe et des sous classes
Protected Friend	Correspond à l'union des visibilités Friend et Protected
Friend	Accessible à partir du programme et des assemblages liés
Private	Accessible à partir des membres de la classe

Par défaut, la visibilité est friend.

- Size

Cet objet permet de définir la taille du contrôle. Il est composé de deux propriétés, width (largeur) et height (hauteur).

- TabIndex

Indice définissant l'ordre de tabulation du contrôle par rapport à son conteneur.

- Text

Cet propriété référence le texte contenu ou affiché dans un contrôle (Par exemple, le texte affiché sur un bouton).

- Visible

Cet propriété détermine si le contrôle est visible lors de l'exécution. Attention, aucun changement n'est visible lors de la conception.

2.1.2 Méthodes

Méthode	Description
Focus	Donne le focus au contrôle

2.1.3 Evénements

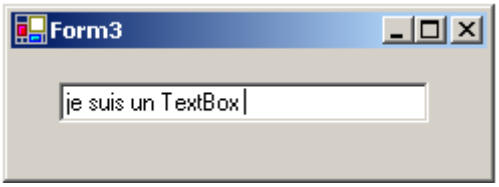
Evénements	Description
Click	Activé lors du clic sur le contrôle
DoubleClick	Activé lors du double clic sur le contrôle
Enter	Activé lorsque l'utilisateur entre sur le contrôle
GotFocus	Activé lorsque le contrôle reçoit le focus
KeyDown	Touche enfoncée
KeyPress	Touche enfoncée et relachée
KeyUp	Touche relachée
LostFocus	Activé lorsque le contrôle perd le focus
MouseDown	Bouton souris enfoncé
MouseUp	Bouton souris relaché
MouseMove	Souris déplacée sur le contrôle
MouseWheel	Déplacement de la roulette
Resize	Déclenché lorsque le contrôle est redimensionné

2.2 Principaux Contrôles

Nous ne listerons dans cette partie que les principaux contrôles.

2.2.1 TextBox

Le contrôle TextBox est certainement le contrôle le plus utilisé : il permet de saisir des chaînes de caractère de 2000 à 32 000 caractères en fonction de la configuration.



Propriété	Description
CanFocus	Détermine si le contrôle peut recevoir le focus
CharacterCasing	Détermine la casse du texte : majuscules (upper) ou minuscules (lower)
Focused	Indique si le contrôle détient le focus
ForeColor	Couleur du texte
HideSelection	Définit si le contrôle masque la sélection lorsqu'il perd le focus
Lines	Tableau correspondant aux lignes du contrôle
MaxLength	Nombre de caractères maximum du contrôle
Modified	Spécifie si le contenu du champs a été modifié depuis sa création
MultiLine	Définit si le contrôle est multi lignes
PasswordChar	Définit le caractère servant à masquer un mot de passe

Readonly	Contenu du champs en lecture seule
Scrollbars	Affiche ou masque les barres de défilement
Selectionlength	Longueur de la sélection
SelectionStart	Indice de début de la sélection dans le champs
Text	Contenu du champs
TextLength	Longueur du texte dans le contrôle

Méthode	Description
Clear	Efface le contenu du champs texte
Copy / Cut	Copie / coupe la sélection dans le presse papier
Focus	Donne le focus au contrôle
ResetText	Rétabli la valeur initiale du champs

Evénement	Description
TextChanged	Déclenché lorsque le texte change

L'exemple suivant permet de copier dans le presse papier tout le texte contenu dans le champs « textbox1 » et de vider ce dernier lorsqu'il reçoit le focus.

```
Private Sub TextBox1_GotFocus(ByVal sender As Object, ByVal e As System.EventArgs)
Handles TextBox1.GotFocus
    With Me.TextBox1
        .SelectionStart = 0
        .SelectionLength = .TextLength
        .Copy()
        .Text = ""
    End With
End Sub
```

2.2.2 Label

Le contrôle label est utilisé pour afficher du texte qui ne sera pas éditable par l'utilisateur. Il est généralement utilisé pour afficher le rôle des différents contrôles.

Propriété	Description
BorderStyle	Style de bordure
AutoSize	Le contrôle s'adapte à la taille du texte
Text	Contenu du label

L'exemple suivant affiche successivement « Bonjour » en gras et « Au revoir » en rouge lorsque l'utilisateur double clic sur le contrôle label1 :

```
Private Sub Label1_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Label1.DoubleClick
```

```
        With Me.Label1
            If .Text = "Bonjour" Then
                .Text = "Au revoir"
                .Font = New Font(.Font, FontStyle.Regular)
                .ForeColor = System.Drawing.Color.Red
            Else
                .Text = "Bonjour"
                .Font = New Font(.Font, FontStyle.Bold)
                .ForeColor = System.Drawing.Color.Black
            End If
        End With
End Sub
```

2.2.3 CheckBox

Le contrôle Checkbox (Case à cocher) est utilisé pour proposer plusieurs options à l'utilisateur parmi lesquelles il pourra effectuer plusieurs choix.

Propriété	Description
Checked	Valeur booléenne indiquant si la case est cochée ou non
CheckState	Retourne ou modifie la valeur de la case à cocher en gérant le 3ème mode (grisé).
ThreeState	En standard, une case à cochée peut être cochée ou non. Il existe cependant un 3ème état « Indéterminé » permettant de grisé la case. Cette propriété permet d'activer ce 3ème état.
CheckAlign	Alignement de la case à cocher par rapport au contrôle
Text	Texte associé au contrôle

Evénement	Description
CheckedChanged	Se produit lorsque la propriété « Checked » change
CheckStateChanged	Se produit lorsque la propriété « CheckState » change

L'exemple suivant comporte 3 cases à cocher (pour une sélection d'options voiture) :

- ch1 libellée « Décapotable »
- ch2 libellée « Toit ouvrant »
- ch3 libellée « Ailerons »

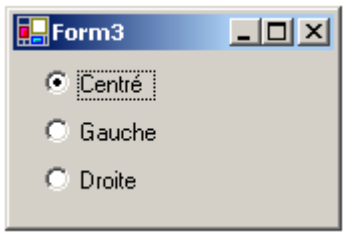
Pour des raisons logiques, il n'est pas possible de choisir ch2 et ch3 si ch1 est sélectionné. Le code suivant permet de décocher et griser les cases.

```
    Private Sub ch1_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ch1.CheckedChanged
```

```
        If Me.ch1.CheckState = CheckState.Checked Then
            Me.ch2.Checked = False
            Me.ch3.Checked = False
            Me.ch2.Enabled = False
            Me.ch3.Enabled = False
        Else
            Me.ch2.Enabled = True
            Me.ch3.Enabled = True
        End If
    End Sub

    Private Sub ch2_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ch2.CheckedChanged
        If Not Me.ch2.Checked = True Then Me.ch1.CheckState = CheckState.Unchecked
        If Not Me.ch2.Enabled Then Me.ch2.Enabled = True
        If Not Me.ch3.Enabled Then Me.ch3.Enabled = True
    End Sub

    Private Sub ch3_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ch3.CheckedChanged
        If Not Me.ch3.Checked = True Then Me.ch1.CheckState = CheckState.Unchecked
        If Not Me.ch2.Enabled Then Me.ch2.Enabled = True
        If Not Me.ch3.Enabled Then Me.ch3.Enabled = True
    End Sub
```

2.2.4 RadioButton

Contrairement aux cases à cocher, les boutons radio permettent à l'utilisateur d'effectuer un seul choix parmi plusieurs options. Cette dernière contrainte impose donc qu'il n'y ait jamais deux boutons cochés en même temps : Visual basic s'occupe de faire basculer l'état des boutons pour les boutons présents dans le même conteneur. Dans l'exemple suivant, c'est le formulaire qui est conteneur. Nous verrons plus loin les conteneurs « GroupBox » et « Panel ».



Les boutons radios possèdent les même propriétés et événements que les cases à cocher.

L'exemple suivant travaille avec 3 boutons radio (rad_blue, rad_red, rad_black) qui modifie la couleur de fond du formulaire en fonction de celui sélectionné. Notez qu'une seule procédure est utilisée pour implémenter les l'événement « CheckedChanged » de chaque bouton radio.

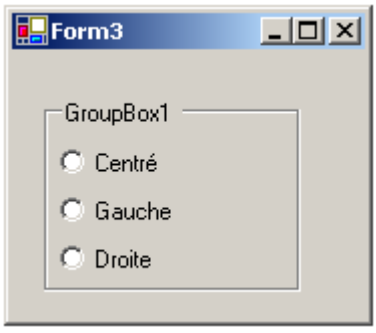
```
Private Sub rad_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles rad_blue.CheckedChanged, rad_black.CheckedChanged,
rad_red.CheckedChanged
    If Me.rad_black.Checked Then
        Me.BackColor = System.Drawing.Color.Black
    ElseIf Me.rad_blue.Checked Then
        Me.BackColor = System.Drawing.Color.Blue
    Else
        Me.BackColor = System.Drawing.Color.Red
    End If
End Sub
```

2.2.5 GroupBox et Panel

Au même titre qu'un formulaire, les contrôles GroupBox et Panel sont des conteneurs, c'est à dire qu'il contiennent eux même d'autres contrôles. Ces contrôles présentent deux intérêts majeurs :

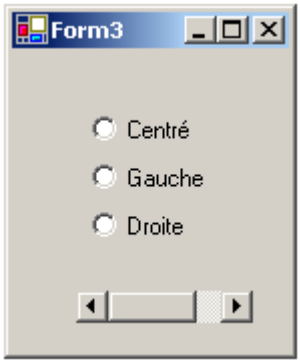
- Regrouper de manière logique des contrôles afin de les isoler (pour les boutons radio par exemple)
- Faciliter le placement de plusieurs contrôles car en modifiant la position du conteneur, vous modifiez la position de tous les contrôles contenus

- Le GroupBox



Ce contrôle possède une seule propriété particulière« text » qui correspond au texte affiché.

- Le Panel



Le contrôle panel reprend les fonctionnalités du contrôle GroupBox avec en plus la possibilité de gérer les barres de défilement (propriété AutoScroll).

2.2.6 Button

Le contrôle « button » est principalement utilisé pour déclencher une action lors du clic.

2.2.7 ListBox

Le contrôle ListBox permet l'affichage d'une liste de choix, généralement des chaînes de caractères, dans laquelle l'utilisateur pour effectuer un ou plusieurs choix.

Propriété	Description
MultiColumn	Permet un défilement horizontal de la liste
Integralheight	Evite l'affichage d'une partie d'un élément de la liste
Items	Collection représentant les éléments contenus dans la liste
Sorted	Eléments classés par dans l'ordre
Itemheight	Hauteur d'un élément de la liste
SelectedIndex	Indice de l'élément sélectionné
SelectedIndices	Indices des éléments sélectionnés
SelectionMode	Mode de sélection des éléments (« MultiExtended » permet une sélection multiple, « One » permet une seule sélection et « None » aucune.

Méthode	Description
FindString	Retourne l'indice de l'élément commençant par le texte recherché
SetSelected	Définit un élément en tant que sélectionné ou non
GetSelected	Retourne un booléen permettant de savoir si un élément est sélectionné ou non

Evénement	Description
SelectedIndexChange	Déclenché lorsque la propriété « SelectedIndex » change

L'exemple suivant remplit un ListBox à l'aide d'une boucle.

```
Dim i As Int16
Me.ListBox1.Items.Clear()
For i = 1 To 50
   Me.ListBox1.Items.Add("Element no " & i)
Next
```

Celui ci affiche l'élément sélectionné lors d'un double clic sur le ListBox.

```
Private Sub ListBox1_DoubleClick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ListBox1.DoubleClick
    Dim elt As String
    Dim indice As Int16
    indice = Me.ListBox1.SelectedIndex
    elt = Me.ListBox1.Items(indice)
```

```
    MsgBox("Elément sélectionné: " & elt)
End Sub
```

Enfin, le code permettant d'afficher la liste des éléments sélectionnés :

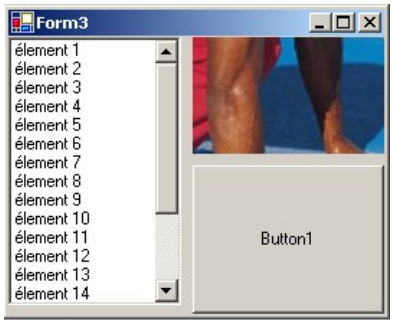
```
Dim elt As String
For Each elt In Me.ListBox1.SelectedItems
    MsgBox("Libellé: " & elt)
Next
```

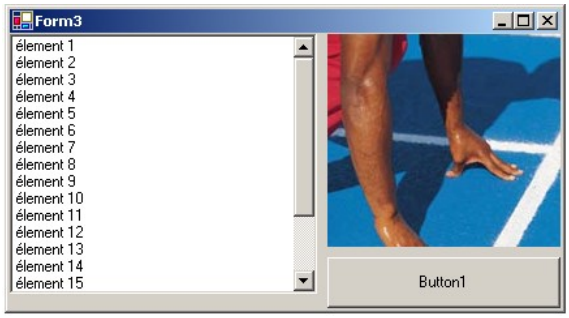
2.2.8 ComboBox

Le contrôle ComboBox est l'association du contrôle listBox et 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.

2.2.9 Splitter

Le contrôle « Splitter » est une nouveauté de la version .Net. Il permet de créer des barres de séparation redimensionnables pour distribuer l'espace du formulaire entre les différents contrôles. Splitter est particulièrement utilisé dans les interfaces de type « explorateur ».





Plutôt qu'un long discours, ci-dessous figure le mode opératoire afin de réaliser l'interface montrée en exemple.

- Placer le contrôle « liste » et paramétrer la propriété « Dock » à « left »
- Placer le contrôle « splitter » à droite de la liste et paramétrer sa propriété « dock » à left
- Placer le contrôle « image » à droite et paramétrer la propriété « Dock » à « top »
- Placer le second splitter en dessous de l'image et paramétrer la propriété « Dock » à « top »
- Enfin, placer le contrôle « bouton » en dessous et paramétrer la propriété « Dock » à « fill »

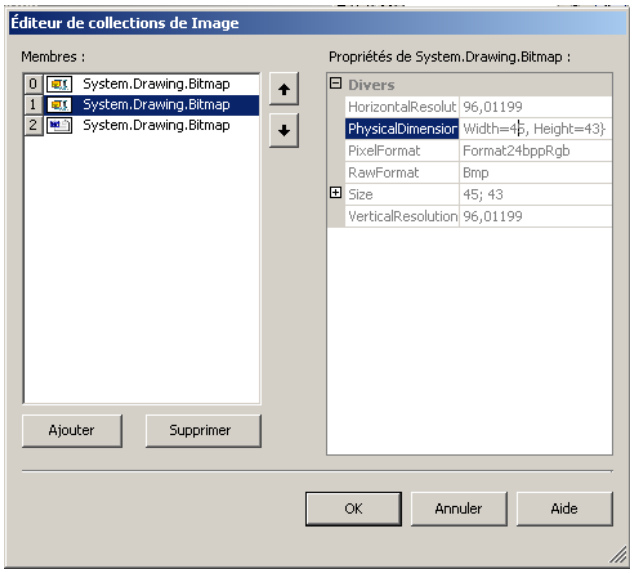
Propriété	Description
MinSize	
MinExtra	

2.2.10 ImageList

Le contrôle « ImageList » est un conteneur d'images destinées à être utilisée dans l'application ou alors par d'autre contrôles (Listview, TreeView …). Ce contrôle n'est pas visible sur le formulaire et peut contenir tous types d'images (Gif, Jper, Bmp …).

Propriété	Description
ColorDepth	Nombre de couleurs à utiliser pour les images
ImageSize	Taille en pixels des images
Transparent	Définit la couleur de transparence
Images	Collection contenant les images

Chacune des images possède un Index qui sera ensuite utilisé pour les lier aux autres contrôles. La gestion des images se fait à l'aide d'un assistant :

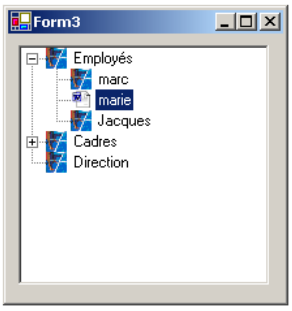


Le code suivant permet d'ajouter une image dans un Imagelist à partir d'un fichier physique et la supprimer du contrôle :

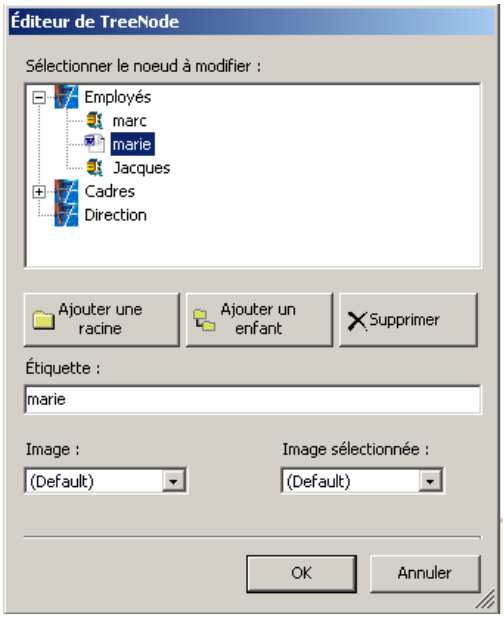
```
Dim chemin as string
Chemin = ''C:\mes documents\toto.gif''
Me.Imagelist1.images.add(Image.fromfile(chemin))
Me.Imagelist1.images.RemoveAt(0)
```

2.2.11 Treeview

Le contrôle TreeView permet un affichage hiérarchique des données à la façon de l'explorateur Windows. Chaque élément du treeview est un nœud pouvant à son tour contenir d'autres nœuds.



Pour remplir le Treeview, vous pouvez utiliser les méthodes liées au contrôle ou utiliser l'assistant fourni par le framework. Pour l'ouvrir, utiliser le bouton situé à droite de la propriété « Nodes » :



- Ajouter une racine : Ajoute un élément à la racine (Employés par exemple)
- Ajouter un enfant : Ajoute un nœud enfant au nœud sélectionné
- Etiquette : Texte affiché au niveau de l'élément sélectionné
- Image : Image du contrôle imagelist lié.
- Image sélectionnée : Image affichée lorsque l'élément est sélectionné

Propriété	Description
CheckBoxes	Afficher les cases à cocher au niveau des éléments
FullrowSelect	La surbrillance s'étend sur toute la largeur du contrôle
ImageIndex	Indice de l'image par défaut du contrôle ImageList
ImageList	Contrôle ImageList contenant les images utilisées par le treeview
Indent	Valeur en pixel de l'indentation
LabelEdit	Permet à l'utilisateur de modifier l'étiquette
Nodes	Collection de nœuds
SelectedImageIndex	Indice de l'image par défaut pour les éléments sélectionnés
ShowPlusMinus	Affiche les signes + et – devant les nœuds parents
Sorted	Indique si les nœuds sont triés

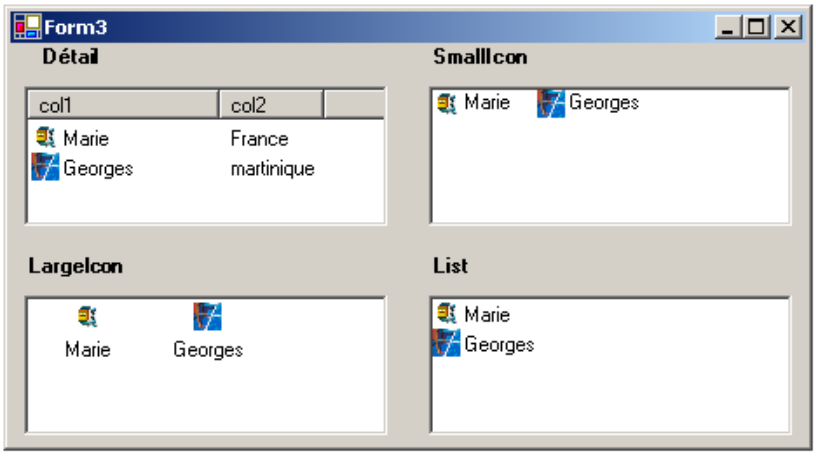
Méthode	Description
Nodes.add	Ajoute un élément

Le code suivant remplit un ImageList, insère à l'intérieur du TreeView 2 catégories principales (Renault & Peugeot) et place ensuite à l'intérieur les différents modèle en leur affectant des images :

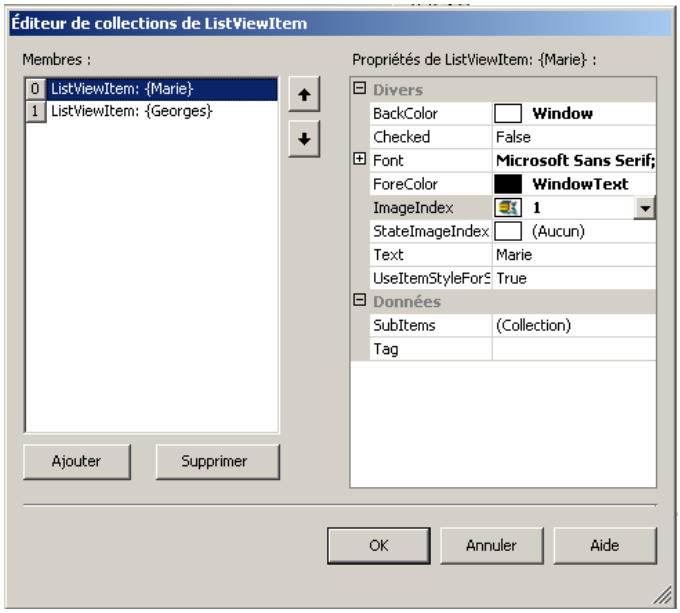
```
Me.ImageList1.Images.Add(Image.FromFile("C:\peugeot.jpg"))
Me.ImageList1.Images.Add(Image.FromFile("C:\renaud.bmp"))

Me.TreeView1.ImageList = Me.ImageList1
Me.TreeView1.ImageIndex = 0

Dim noeud1, noeud2 As TreeNode
noeud1 = New TreeNode
noeud2 = New TreeNode

With noeud1
   .Text = "Peugeot"
   .ImageIndex = 0
   .Nodes.Add("307")
   .Nodes.Add("806")
   .Nodes.Add("309")
End With
With noeud2
   .Text = "Renault"
   .ImageIndex = 1
   .Nodes.Add("Mégane")
   .Nodes.Add("4L")
   .Nodes.Add("Laguna")
End With

With Me.TreeView1
   .Nodes.Add(noeud1)
   .Nodes.Add(noeud2)
End With
```

2.2.12 ListView

Le contrôle listview permet l'affichage d'une liste plate selon les 4 modes de présentation de l'explorateur Windows :



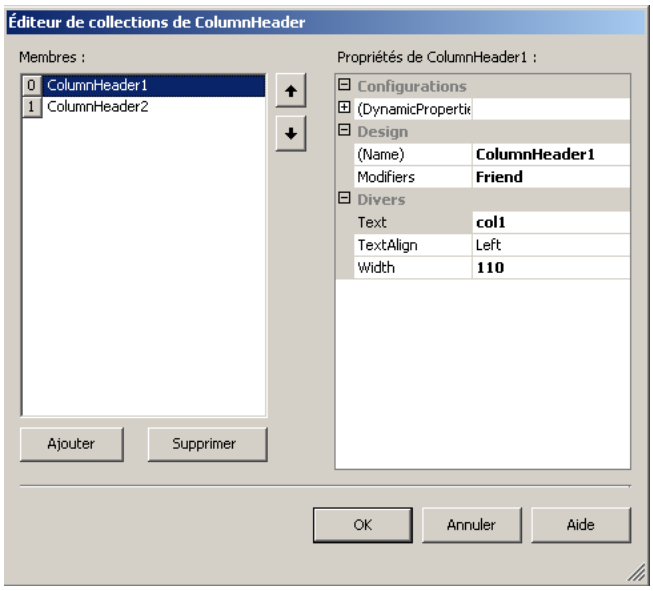
Au même titre que le TreeView, le ListView posséde des assistants afin de définir leur contenu. Pour ouvrir l'assistant permettant de gérer la liste, cliquer sur le bouton à droite de la propriété Items :



Pour chaque élément, il est possible de paramétrer :

- Checked : définit si l'élément apparaît coché par défaut
- Font : police de l'élément
- ForeColor : couleur d'affichage du texte
- ImageIndex : indice de l'image liée à l'élément
- Text : libellé de l'élément
- UseItemStyleForSubitems : répercute les propriété de l'élément sur les sous éléments
- SubItem : dans un affichage par détail, correspond aux colonnes à partir de la seconde. Lors du clic sur le bouton correspondant à cette propriété, un nouvel assistant est lancé pour définir les autres colonnes.

Il existe également un assistant pour les colonnes : pour l'ouvrir, utiliser le bouton à droite de la propriété « Columns ». Celles ci n'apparaîtront que lors d'un affichage au détail.



Propriété	Description
AllowColumnReorder	Autorise ou non la modification de l'ordre des colonnes
AutoArrange	Organise automatiquement la présentation des éléments
Columns	Collection de colonnes
FullrowSelect	La surbrillance s'étend sur toute la largeur du contrôle
HeaderStyle	Style des entête de colonne
Items	Collection d'éléments
LabelEdit	Permet à l'utilisateur de modifier l'étiquette des éléments
LargeImageList	ImageList utilisé pour la présentation « LargeIcon »
MultiSelect	Permet la sélection multiple
SmallImageList	ImageList utilisé pour toutes les présentation (sauf LargeIcon)
Sorting	Mode de tri
View	Mode de représentation de la liste

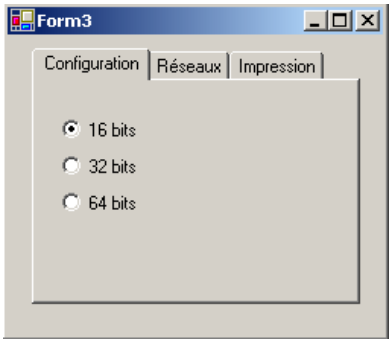
Le code suivant affiche pour chaque caractère son code ascii, le caractère en minuscule et le caractère en majuscule. Il crée également les colonnes :

```
Dim elt As ListViewItem
Dim i As Byte

With Me.ListView1
   .View = View.Details
   .Columns.Add("Ascii", 50, HorizontalAlignment.Center)
   .Columns.Add("Min", 50, HorizontalAlignment.Center)
   .Columns.Add("Maj", 50, HorizontalAlignment.Center)
   For i = 0 To 255
      elt = New ListViewItem
      elt.Text = CType(i, String)
      elt.SubItems.Add(LCase(Chr(i)))
      elt.SubItems.Add(UCase(Chr(i)))
      Me.ListView1.Items.Add(elt)
   Next
```

```
End With
```

2.2.13 TabControl

Le contrôle « TabControl » permet l'affichage d'onglet contenant chacun plusieurs contrôles. Ce dernier est généralement utilisé pour regrouper logiquement des contrôle ou pour placer beaucoup de contrôles dans la même formulaire.



TabControl possède également un assistant permettant de le configurer. Pour ouvrir l'assistant, utiliser le bouton à droite de la propriété « TabPages » :

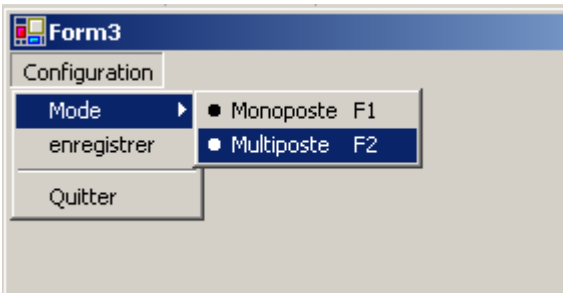
Pour chaque « page », il est possible de configurer les options d'apparence qui sont analogues à celles d'un formulaire.

Propriété	Description
Alignment	Définit la position des onglets par rapport aux pages
HotTrack	Modifie l'apparence des onglets lorsque la souris passe dessus
Imagelist	ImageList lié pour les icônes d'onglets
Multiline	Permet l'affichage des onglets sur plusieurs lignes
TabPages	Collection de pages.

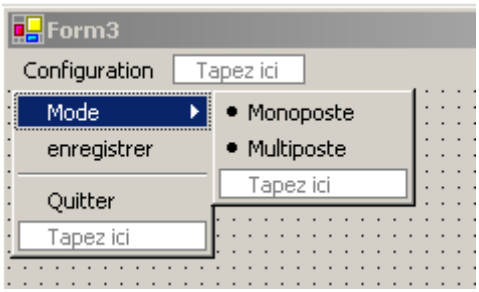
2.2.14 Menus

les menus permettent d'offrir à l'utilisateur un ensemble de fonctionnalités sans pour autant surcharger la présentation du formulaire. Il existe 2 types de menu :

- Menu d'application situé en haut du formulaire
- Menu contextuel activé généralement lors d'un clic droit



Pour créer un ou plusieurs menus, vous devez ajouter à votre formulaire le contrôle « MainMenu ». A ce moment, le contrôle apparaît en dessous du formulaire et un menu est ajouté. Il n'y a pas d'assistants particulier : pour ajouter un élément, cliquer sur les zones « Tapez ici » :



Pour définir une barre de séparation, créer un élément avec « - » (tiret) en libellé.

Propriété	Description
Checked	Indique si l'élément est coché
DefaultItem	Définit l'élément en tant qu'élément par défaut
MdiList	Affiche la liste des fenêtres enfants dans le cas d'une fenêtre MDI
RadioCheck	Indique si l'élément est activé
ShortCut	Permet de définir un raccourci pour le menu
ShowShorcut	Affiche le raccourci
Text	Libellé du menu

2.2.15 DateTimePicker

Le contrôle « DateTimePicker » associe une zone de texte et un calendrier permettant la sélection d'une date.

Propriété	Description
Checked	Si activé, spécifie lorsque l'utilisateur a sélectionné une date
CustomFormat	Chaine de format pour l'affichage de la date sélectionnée
Format	Mode d'affichage de la date
MaxDate	Date maximale sélectionnable
MinDate	Date minimale sélectionnable

Evénement	Description
ValueChanged	Déclenché lorsque la valeur change

Il existe un second contrôle pour la gestion des dates : MonthCalendar. Ce dernier reprend les fonctionnalités du contrôle DateTimePicker avec en plus la possibilité de définir des jours fériés ou des périodes sélectionnées.

2.2.16 Timer

Le contrôle Timer permet de déclencher un événements à intervals réguliers.

Propriété	Description
Interval	Définit l'interval en milliseconde. La valeur doit être comprise entre 1 et 65536
Enabled	Active ou désactive le timer

Evénement	Description
Tick	Déclenché à chaque interval.

2.3 Le Drag and Drop

Le drag and Drop (ou Glisser Déposer) est une des fonctionnalités en terme d'ergonomie qui fit le succès de Windows. Il permet de déplacer des informations (Fichiers, Images, Texte, Objet) au sein d'une même application ou entre plusieurs en accrochant un élément au curseur pour ensuite le déposer sur l'objet de destination (Liste, champs texte…).

2.3.1 Démarrer le drag and drop

Pour démarrer le drag and drop, vous devez détecter lorsque l'utilisateur quitte un contrôle avec un bouton enfoncé. Pour cela, on utilise l'événement « MouseMove » :

```
  Private Sub TextBox1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles TextBox1.MouseMove
      If e.Button = MouseButtons.Left Then
```

```
        …
    End If
End Sub
```

Ensuite, vous activer le drag and drop en spécifiant l'élément à déplacer ainsi que les effets. L'élément à déposer est de type Object (donc ce peut être n'importe quel élément). Pour cela, on utilise la méthode « DoDragDrop » prenant deux arguments : l'objet à déplacer et les effets voulus. Les effets sont membres de la classe « DragDropEffects ».

Effet	Description
All	Les données sont copiées, supprimées et parcourues dans la zone cible
Copy	Les données sont copiées dans la zone de déplacement
Link	Les données sont liées à la cible de déplacement
Move	Les données sont déplacées vers la cible de déplacement

2.3.3 Contrôler la réception

En premier lieu, vous devez spécifier que le contrôle cible peut recevoir des éléments : pour cela, paramétrez la propriété « AllowDrop » à « True ».

Le contrôle de la réception consiste à modifier l'apparence du curseur et du contrôle cible en vérifiant que l'élément déplacé corresponde aux besoins. Pour cela, nous disposons de plusieurs événements (ces 3 événements sont valides tant que l'utilisateur ne lâche pas le bouton de la souris).

Evénement	Description	Utilisation
DragEnter	Se produit lorsque le curseur entre dans la zone du contrôle.	Utilisé pour modifier l'apparence du contrôle cible et du curseur et vérifier la validité de l'élément déplacé
DragOver	Se produit tant que le curseur reste au dessus de la zone du contrôle	
DragLeave	Se produit lorsque le curseur quitte la zone du contrôle	Utilisé pour rétablir l'apparence du contrôle cible et du curseur

Sur chacun des événement est passé en paramètre « e » de type DragEventsArg contient toutes les informations sur le déplacement :

Propriété	Description
Data	Correspond à l'élément déplacé
Effect	Défini l'action autorisée par le contrôle de destination et permet de modifier l'apparence de la souris
KeyState	Permet de connaître l'état des touches Shift, Alt, Ctrl
X, Y	Position du curseur sur le contrôle

Pour récupérer l'élément ou l'objet déplacé, vous devez utiliser la méthode suivante :

```
e.Data.GetData(DataFormats.type_données)
```

Cette méthode est utilisée pour vérifier le type d'objet déplacé (dans la méthode DragEnter).

2.3.3 Récupérer l'élément

La récupération de l'élément se fait lorsque l'événement « DragDrop » est déclenché.

Ci dessous figure un exemple complet permettant de déplacer du texte d'une textbox à une autre :

```
   Private Sub TextBox1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles TextBox1.MouseMove
      If e.Button = MouseButtons.Left Then
         Me.TextBox1.DoDragDrop(Me.TextBox1.Text, DragDropEffects.Move)

      End If
   End Sub

   Private Sub TextBox2_DragEnter(ByVal sender As Object, ByVal e As
System.Windows.Forms.DragEventArgs) Handles TextBox2.DragEnter
      e.Effect = DragDropEffects.Move
      Me.TextBox2.BorderStyle = BorderStyle.FixedSingle
   End Sub

   Private Sub TextBox2_DragDrop(ByVal sender As Object, ByVal e As
System.Windows.Forms.DragEventArgs) Handles TextBox2.DragDrop
      Me.TextBox2.Text = e.Data.GetData(DataFormats.Text)
      Me.TextBox2.BorderStyle = BorderStyle.FixedSingle
      If CBool(e.KeyState And 32) Then
         Me.TextBox1.Clear()
      End If
   End Sub

   Private Sub TextBox2_DragLeave(ByVal sender As Object, ByVal e As
System.EventArgs) Handles TextBox2.DragLeave
      Me.TextBox2.BorderStyle = BorderStyle.Fixed3D
   End Sub
```