L'essentiel des composants visuels et de leurs membres

Le formulaire : `Form`

Propriétés

`Icon`

Cette propriété associe un fichier icône au formulaire. L'icône s'affiche dans la barre de titre à son extrémité gauche. Quand le formulaire est celui de démarrage, son icône est aussi celle de l'application. Visual Studio donne la possibilité de créer une icône par le menu *Fichier/Nouveau /Fichier …/Fichier d'icône* et de la modifier.

`Cursor`

Cette propriété qui peut aussi être réglée au niveau des contrôles, définit le style du curseur par défaut sur le formulaire. Les différentes valeurs sont disponibles dans la classe **`System.Windows.Forms.Cursors`**.

`BackgroundImage`

Définit l'éventuelle image de fond du formulaire. Si l'image est plus petite que le formulaire, elle est répétée en mosaïque, sinon elle est tronquée.

`WindowState`

Définit ou restitue l'état d'affichage du formulaire.

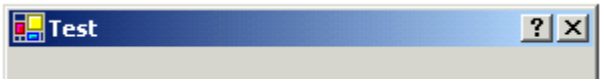
`FormWindowState.Maximized`	Plein écran. Equivalent au clic du bouton *Agrandir*
`FormWindowState.Normal`	Taille normale. Equivalent au clic du bouton *Restaurer*
`FormWindowState.Minimized`	Réduit dans la barre des tâches. Equivalent au clic du bouton *Réduire*

`ControlBox`

Indique si le bloc de boutons de contrôles situé dans le coin supérieur droit du formulaire est présent ou pas. D'autres propriétés sont dépendantes du **`ControlBox`** : **`MaximizeBox`** , **`MinimizeBox`** et **`HelpButton.`**

`MaximizeBox`	Détermine si le bouton *Agrandir* du bloc peut être utilisé. Si cette propriété a comme valeur **`False`**, ce bouton est inactif.
`MinimizeBox`	Détermine si le bouton *Réduire* du bloc peut être utilisé. Si cette propriété a comme valeur **`False`**, ce bouton est inactif.
`HelpButton`	Affiche un bouton d'aide dans le **`ControlBox`** seulement si les boutons **`MaximizeBox`** et **`MinimizeBox`** sont tous les deux inactifs, auquel cas ils ne sont d'ailleurs plus affichés.

Pour répondre au clic sur le **`HelpButton`**, il faut programmer la réponse à l'événement **`HelpRequested`** du formulaire. Il est possible que le clic de ce bouton ne produise pas cet événement sur tous les systèmes, mais avec ou sans le **`HelpButton`**, l'appui de <F1> génère l'événement.




```
Private Sub FBase_HelpRequested(ByVal sender As Object, ByVal hlpevent As
                    System.Windows.Forms.HelpEventArgs) Handles MyBase.HelpRequested
    ' Traitement pour fournir l'aide
End Sub
```

`AcceptButton`

Permet la désignation d'un bouton pour lequel est généré un événement **`Click`** lorsque le formulaire a le focus et que la touche <Enter> est pressée. C'est généralement un bouton *OK* qui est désigné comme **`AcceptButton`**.

CancelButton

Permet la désignation d'un bouton pour lequel est généré un événement **Click** lorsque le formulaire a le focus et que la touche <Escape> est pressée. C'est généralement un bouton *Annule* qui est désigné comme **CancelButton**.

AutoScale

La propriété **AutoScale** autorise ou pas le redimensionnement automatique des contrôles et du formulaire lors d'un changement de la police d'écran.

AutoScroll

La propriété **AutoScroll** autorise ou pas le placement automatiquement des barres de défilement lorsque la taille du formulaire ne permet pas l'affichage de tous les contrôles qu'il contient.

FormBorderStyle

Cette propriété détermine le style du formulaire.

None	Fenêtre sans bord ni barre de titre, non dimensionnable et non déplaçable
FixedSingle	Fenêtre ordinaire non dimensionnable mais déplaçable
Fixed3d	Mêmes propriétés que **FixedSingle**, mais avec un aspect 3D
FixedDialog	Fenêtre non dimensionnable mais déplaçable
Sizable	Fenêtre ordinaire dimensionnable et déplaçable
FixedToolWindow	Fenêtre non dimensionnable mais déplaçable, sans icône ni **ControlBox**
SizableToolWindow	Fenêtre dimensionnable et déplaçable, sans icône ni **ControlBox**

IsMDIContainer

Détermine si le formulaire est un conteneur MDI (Multiple Documents Interface), c'est-à-dire s'il est capable de contenir d'autres fenêtres. Quand une fenêtre MDI est créée, c'est par le code qu'il faut y placer les feuilles filles.

```
Dim F As Form = New NomFeuilleFille  ' NomFeuilleFille est le nom du Form à charger
F.MDIParent = Me
F.Show()
```

StartPosition

Permet de choisir la position de la fenêtre lors de son ouverture.

Manual	Position définie par la propriété **Location**
CenterScreen	Centré par rapport à l'écran
WindowsDefaultlocation	Situé à l'emplacement par défaut de Windows avec la taille définie dans **Size**
WindowsDefaultBounds	Situé à l'emplacement par défaut de Windows avec la taille par défaut de Windows
CenterParent	Centré par rapport à la fenêtre ayant déclenché l'ouverture.

TopMost

Si cette option est activée (**True**) le formulaire est toujours à l'avant plan, même quand il n'est pas actif. Cette option est utile pour les fenêtres dont les contenus doivent toujours être visibles.

Locked

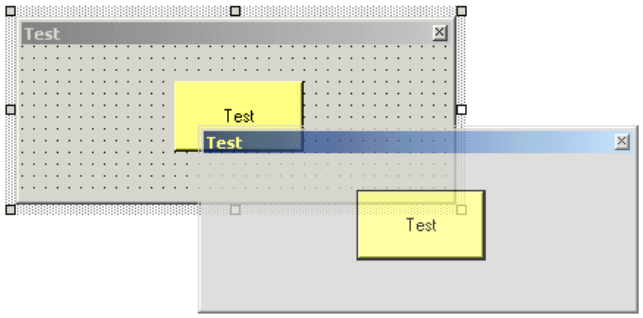
C'est une propriété au service du programmeur. En effet, quand elle a la valeur **True** elle empêche toute modification des propriétés visuelles du composant pendant le développement. Elle n'a aucune incidence sur l'application en cours d'exécution.

`LayoutMode`, `SnapLines` et `SnapToGrid`

C'est par le menu *Outils/Options/Concepteur Windows Forms/Général* qu'il convient de régler la propriété **`LayoutMode`** à **`SnapLines`** ou **`SnapToGrid`** (l'effet du réglage s'observe après le redémarrage de l'environnement).
La propriété **`LayoutMode`** aide le programmeur à dimensionner et à positionner les composants visuels sur un formulaire. La valeur **`SnapLines`** provoque l'apparition de lignes de repérage par rapport aux composants déjà placés. La valeur **`SnapToGrid`** provoque l'apparition d'une grille de points auxquels s'accrochent les composants.

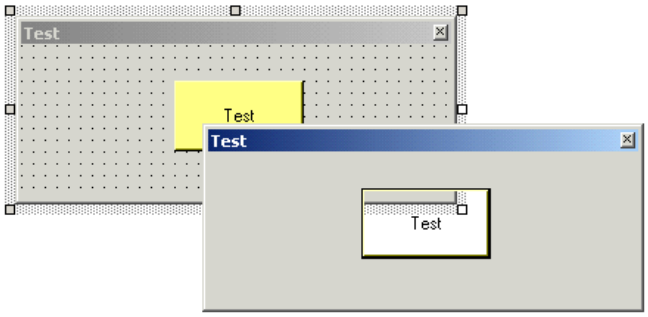
`Opacity`

Définit l'opacité générale d'un formulaire par une valeur allant de 0% à 100%.
Pour obtenir l'illustration ci contre, l'opacité a été réglée à 75%.



`TransparencyKey`

Permet la transparence totale de tout objet du formulaire qui a la couleur désignée.
Pour obtenir l'illustration ci contre, la couleur choisie a été le jaune, le bouton *Test* ayant cette couleur.



`ShowInTaskBar`

Cette propriété, qui a la valeur **`True`** par défaut, empêche la visibilité de l'application dans la barre des tâches lorsque sa valeur est **`False`**.

Méthodes

`Activate`

La méthode **`Activate`** permet de remettre un formulaire au premier plan et de lui donner le focus.

`Close`

Cette méthode ferme le formulaire et libère la mémoire des ressources utilisées. Dans le premier programme de cette partie du cours, l'usage de l'instruction **`End`** ou de la méthode **`Me.Close()`**produit le même résultat.

`ShowDialog`

Affiche le formulaire en tant que feuille modale, c'est à dire que la fenêtre reste au premier plan tant qu'elle n'est pas fermée et empêche l'accès aux autres feuilles de l'application. L'objet **`MessageBox`** est toujours une fenêtre modale.

`BringToFront`

Cette méthode *pousse* le formulaire à l'avant plan. Contrairement à l'effet produit par la propriété **`TopMost`**, qui place définitivement le formulaire en avant, celui de **`BringToFront`** est temporaire.

Evénements

Activated et **Deactivate**

L'événement **Activated** survient quand un formulaire reçoit le focus, qu'il devient actif. L'événement **Deactivate** se produit évidemment dans le cas contraire. Ces événements se produisent notamment quand le focus passe de la fenêtre d'une application à celle d'une autre application, ou d'une fenêtre à l'autre dans une même application.

DoubleClick

Comme son nom l'indique …

Load

Cet événement se produit avant le premier affichage du formulaire.

Closing et **Closed**

L'événement **Closing** se produit pendant la fermeture du formulaire et **Closed** quand il est effectivement fermé.

Membres communs à la plupart des composants visibles

Propriétés

Déjà vues : **Enabled**, **Location**, **Locked**, **Text**, **Visible**

Anchor

Les ancres permettent de fixer la position d'un contrôle de sorte qu'il la conserve ensuite lors des redimensionnements du formulaire. L'effet est assuré tant en développement qu'à l'exécution. Valeur par défaut : **None**. L'affectation simultanée des quatre valeurs **Top**, **Bottom**, **Left** et **Right** offre une mise à l'échelle globale des composants. Les propriétés **Anchor** et **Dock** s'excluent mutuellement, seule la dernière affectée est valide.

Dock

Similaire à **Anchor**, sauf qu'elle définit une seule ancre, la propriété **Dock** permet d'ancrer un contrôle à un bord du conteneur. Valeur par défaut : **None**. Les propriétés **Anchor** et **Dock** s'excluent mutuellement, seule la dernière affectée est valide.

Left	L'objet occupe tout le bord gauche
Right	L'objet occupe tout le bord droit
Top	L'objet occupe tout le bord supérieur
Bottom	L'objet occupe tout le bord inférieur
Fill	L'objet occupe toute la surface du conteneur

Modifiers

Cette propriété définit la portée de l'objet au niveau de la programmation. Par défaut, sa valeur est **Friend**.

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 de tous les éléments du projet.
Private	Accessible à partir des membres de la classe

Size

Cette propriété contient la taille du contrôle. Elle est composée de deux *sous* propriétés, **Width** et **Height**.

`ClientSize`

Cette propriété contient la taille intérieure du contrôle, c'est-à-dire l'espace intérieur disponible. Elle est composée de deux *sous* propriétés, **`Width`** et **`Height`**, dont les valeurs sont toujours inférieures à celles de la propriété **`Size`** du même composant. La différence est particulièrement importante pour un **`Form`** où il faut déduire de **`Size`**, non seulement les épaisseurs des bords, mais aussi la hauteur de la barre de titre.

`TabIndex`

Ordre d'accès au contrôle par la touche <Tab>. Tous les objets posés sur un formulaire reçoivent une valeur pour cette propriété. La valeur par défaut correspond à l'ordre d'arrivée de l'objet sur la feuille : **`0`** pour le premier, **`1`** pour le deuxième, … jusqu'à **`N-1`** pour le $N^{ième}$.

`Tag`

Voici une sympathique propriété qui ne sert à rien ou, plus exactement, à n'importe quoi. C'est une variable associée (comme un signet) à l'objet et disponible pour le programmeur qui peut y stocker une valeur quelconque.

Méthode

Déjà vues : **`Focus`**, **`Select`**

Evénements

Déjà vus : **`Click`**, **`DoubleClick`**, **`GotFocus`**, **`KeyPress`**, **`LostFocus`**, **`MouseMove`**, **`MouseEnter`**, **`MouseLeave`**

`Enter`	Activé lorsque le contrôle reçoit le focus, bonne alternative à **`GotFocus`**
`Leave`	Activé lorsque le contrôle perd le focus, bonne alternative à **`LostFocus`**
`KeyDown`	Touche enfoncée
`KeyUp`	Touche relâchée
`MouseDown`	Bouton souris enfoncé
`MouseUp`	Bouton souris relâché
`MouseWheel`	Déplacement de la roulette
`Resize`	Déclenché lorsque le contrôle est redimensionné

L'étiquette : `Label`

Propriétés

`BorderStyle`

Style de bordure. Valeur par défaut : **`None`**. Autres valeurs possibles : **`FixedSingle`** et **`Fixed3d`**

`None`	Pas d'encadrement
`FixedSingle`	Encadré d'un simple trait
`Fixed3d`	Aspect 3D enfoncé

`AutoSize`

La taille de l'étiquette est adaptée à la taille du texte.

`TextAlign`

Position du texte dans l'étiquette. Valeur par défaut : **`TopLeft`**. Autres valeurs possibles : **`TopCenter`**, **`TopRight`**, **`MiddleLeft`**, **`MiddleCenter`**, **`MiddleRight`**, **`BottomLeft`**, **`BottomCenter`** et **`BottomRight`**.

La boîte de texte : `TextBox`

Propriétés

`CharacterCasing`

Détermine la casse du texte. Valeur par défaut : **`CharacterCasing.Normal`**.
Autres valeurs possibles : **`CharacterCasing.Upper`** (majuscule) et **`CharacterCasing.Lower`** (minuscule).

`Focused`

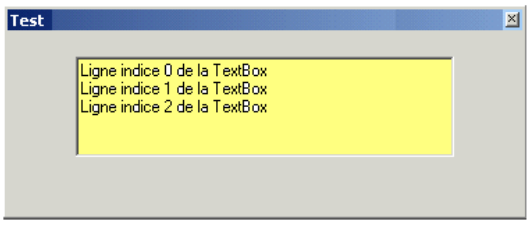
Cette propriété a la valeur **`True`** quand le contrôle détient le focus. Elle n'est accessible que par le code et en lecture seule.

`HideSelection`

Cette propriété définit si le contrôle conserve la marque d'une sélection lorsqu'il perd le focus. Sa valeur par défaut est **`True`**, ce qui signifie que le contrôle cache la marque de sélection.

`Lines` et `MultiLine`

La propriété **`Lines`** qui est un tableau de chaînes, donne accès à chacune des lignes d'une **`TextBox`**. Il est permis d'encoder plusieurs lignes de texte dans une **`TextBox`** quand sa propriété **`MultiLine`** a la valeur **`True`**.




```
Dim i As Integer
For i = 0 To MaTextBox.Lines.Length - 1
    MessageBox.Show(MaTextBox.Lines(i))
Next
```

`MaxLength`

Nombre maximum de caractères autorisés pour le texte du contrôle. Par défaut : **`32767`**.

`Modified`

Cette propriété reçoit la valeur **`True`** chaque fois que le contenu du champ est modifié. Elle est accessible par le code en lecture et écriture.

`PasswordChar`

Tout caractère encodé ou affiché est remplacé par le caractère affecté à cette propriété. Par défaut, cette propriété est vide.

`ReadOnly`

L'affectation de la valeur **`True`** à cette propriété interdit toute modification du contenu.

`SelectionStart` et `SelectionLength`

La propriété **`SelectionStart`** contient l'indice de départ d'une sélection et **`SelectionLength`** contient la longueur de sélection effectuée par un glissé du curseur sur le texte ou la longueur de la sélection à effectuer par le code.

```
MaTextBox.SelectionStart = 1        ' Commencer la sélection au deuxième caractère
MaTextBox.SelectionLength = 3       ' Sélectionner 3 caractères
MaTextBox.Select()                  ' Donne le focus et fait la sélection demandée
```

`TextLength`

Cette propriété contient le nombre de caractère contenu dans le contrôle.

Méthodes

`Clear`

Vide la **`TextBox`** de son contenu.

`Copy`, `Cut` et `Paste`

-Ce sont les méthodes *Copier*, *Couper* et *Coller* de Windows. Les méthodes **`Copy`** et **`Cut`** s'appliquent ici au texte sélectionné d'une **`TextBox`**. La méthode **`Paste`** restitue le contenu du presse-papier.

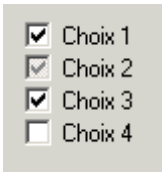
Evénement

`TextChanged`

Déclenché lorsque le texte change. L'événement se produit à chaque appui de touche.

La case à cocher : `CheckBox`

Un jeu de cases à cocher permet à l'utilisateur d'effectuer plusieurs choix parmi ceux proposés.



Propriétés

`Checked`

Cette propriété accessible en lecture et écriture a la valeur **`True`** quand la case est cochée.

`CheckState`

Cette propriété accessible en lecture et écriture peut avoir trois valeurs différentes selon que la case est cochée, cochée et grisée, et non cochée.

`CheckState.Checked`	La case est cochée (Choix 1 de l'illustration)
`CheckState.Indeterminate`	La case est cochée et grisée (Choix 2 de l'illustration)
`CheckState.Unchecked`	La case est non cochée (Choix 4 de l'illustration)

`ThreeState`

Par défaut, cette propriété a la valeur **`False`**. Il faut lui affecter la valeur **`True`** pour qu'une case à cocher puisse être cochée et grisée.

`CheckAlign`

Cette propriété permet de régler les positions respectives du texte et de la case au sein d'un contrôle **`CheckBox`**.

Evénements

`CheckedChanged`

Cet événement se produit quand la propriété **`Checked`** change.

`CheckStateChanged`

Cet événement se produit quand la propriété **`CheckState`** change.