

# Les Contrôles de bases en Windows Form

---

## Sommaire

1	Introduction.....	2
2	Gérer les contrôles dans une Windows Form .....	2
2.1	Contrôles de bases.....	2
2.2	Gestion de la taille et de l'emplacement à la conception.....	4
2.3	La <i>ToolBar</i> .....	4
2.4	Le <i>Snaplines</i> .....	5
2.5	Modifier les propriétés de vos contrôles à leur conception.....	5
2.6	Le <i>Smart Tags</i> .....	7
2.7	Le <i>Document Outline</i> .....	8
3	Manipuler les affichages de textes.....	9
3.1	Le contrôle Bouton.....	9
3.2	La propriété <i>FlatAppearance</i> et <i>FlatStyle</i> .....	11
3.3	Le contrôle <i>Label</i> .....	11
3.4	Le contrôle <i>LinkLabel</i> .....	13
4	Utiliser le contrôle Text Edit.....	14
4.1	La <i>TextBox</i> .....	14
4.2	La <i>MaskedTextBox</i> .....	16
4.3	La propriété <i>Mask</i> .....	17
5	Conclusion .....	18

## 1 Introduction

Dans ce second chapitre, vous approfondirez la façon de créer une interface utilisateur et de configurer des contrôles. Vous découvrirez comment gérer les contrôles de votre application, les propriétés spécifiques de celui-ci et le contrôle d’affichage de texte. A nouveau vous pourrez manipuler ces propriétés de façon intuitive grâce à Visual Studio ou en manipulant les code de votre projet.

Bon cours .Net

L’équipe *Windows Form*

## 2 Gérer les contrôles dans une Windows Form

### 2.1 Contrôles de bases

Les contrôles permettent de combiner une interface graphique avec des fonctionnalités propres. Ces contrôles peuvent être utilisés plusieurs fois et chaque contrôle a été créé pour exécuter une ou plusieurs tâches particulières.

De plus ces contrôles contiennent des propriétés, des méthodes et des événements permettant de les utiliser pour une tâche précise. Par exemple avec un contrôle permettant de modifier la taille d’une fenêtre ou encore son opacité, plusieurs configurations sont possibles.

Voici les contrôles de bases :

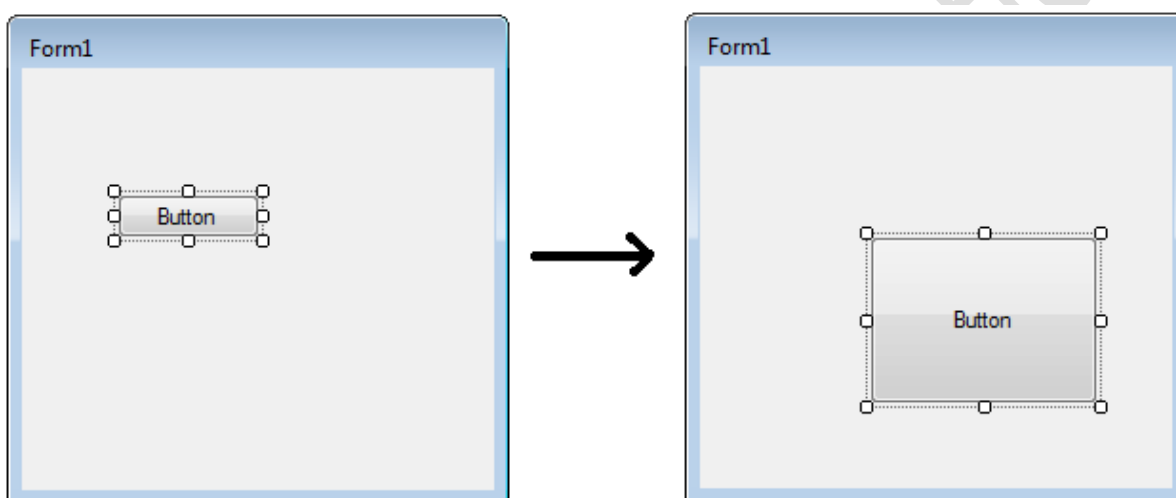
Types de Propriétés	Description
Anchor	Détermine comment le contrôle est rattaché à la <i>Form</i> mère ou dans <i>Container Control</i> .
BackColor	Définit le contrôle de <i>BackColor</i> .
BackgroundImage	Représente l’image affichée comme arrière plan du contrôle.
CausesValidation	Permet de vérifier que les données saisies par l’utilisateur correspondent à la valeur et la mise en forme demandée.
ContainsFocus	Indique si le contrôle est sélectionné.
Controls	Représente la collection de contrôles contenus dans ce contrôle, mais ne peut être utilisé que dans un <i>Container</i> .

Cursor	Représente le curseur lorsqu'il lorsque le pointeur de la souris est sur le contrôle.
Dock	Détermine comment le contrôle est amarré dans la <i>Form</i> mère ou le <i>Container Control</i> .
Enabled	Permet de contrôler si le contrôle est activé. S'il ne l'est pas il apparaîtra en gris et il sera impossible de le sélectionner.
Font	Définit la police utilisée pour afficher le texte soumis à ce contrôle.
ForeColor	Représente la couleur affichée en avant-plan, on l'utilise en général pour la couleur d'un texte.
HasChildren	Permet d'obtenir une valeur qui montre si ce contrôle a des contrôles enfants.
Height	Représente la hauteur du contrôle en pixel.
Location	Indique l'emplacement du coin supérieur gauche du contrôle par rapport au coin supérieur gauche de la <i>Form</i> mère ou du <i>Container Control</i> .
MaximumSize	Définit la taille maximale du contrôle.
MinimumSize	Définit la taille minimale du contrôle.
Name	Représente le nom utilisé pour référer le contrôle dans le code.
Parent	Définit la <i>Form</i> mère ou le <i>Container Control</i> pour ce contrôle.
Region	Définit l'emplacement de la fenêtre par rapport au contrôle.
Size	Représente la taille du contrôle.
TabOrder	Définit dans quel ordre les contrôles vont être sélectionné quand la <i>TabKey</i> sera utilisée pour naviguer de contrôle en contrôle.
Tag	Permet au programmeur de stocker une valeur ou un objet associé à un contrôle.
Text	Définit le texte associé à un contrôle.

Visible	Définit si le contrôle est visible ou pas.
Width	Définit la largeur du control en pixel.

## 2.2 Gestion de la taille et de l'emplacement à la conception

Vous pouvez modifier un contrôle de façon intuitive à la souris. Par exemple, vous pouvez définir la taille d'un bouton ou encore son emplacement. Il suffit juste de le sélectionner avant (votre contrôle est sélectionné lorsque qu'il est entouré de pointillés).

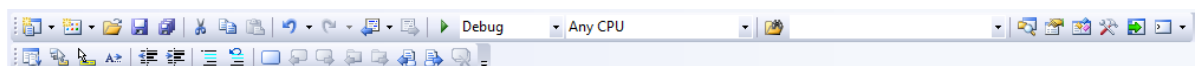


Vous pouvez modifier la taille de votre *Form* en tirant directement sur l'un des coins (dans la partie Designer). Vous modifiez un contrôle de la même manière, vous le déplacez en cliquant dessus, en gardant le clic de la souris enfoncé puis en déplaçant la souris.

Pour déplacer un groupe de contrôle, il suffit de les sélectionner un par un en gardant la touche *Ctrl* enfoncée.

## 2.3 La ToolBar

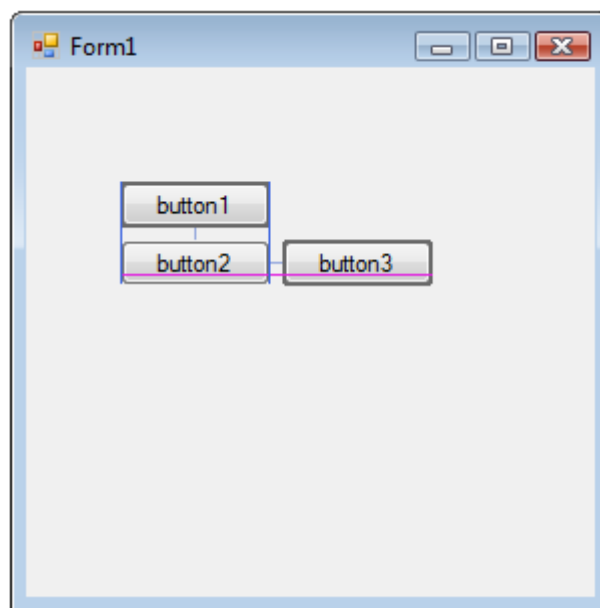
La *ToolBar* permet d'exécuter un nombre d'actions importantes et répétitives de manière rapide et simple. Vous pouvez modifier cette *ToolBar* en allant sur *View* puis *ToolBar*.



## 2.4 Le Snaplines

Le Snapline est une aide visuelle lors de la conception d'une *Form*, c'est-à-dire qu'il permet de mieux placer nos contrôles entre eux et par rapport à la *Form* elle-même. Entre autre le Snapline apparaît lorsque :

- vous rapprochez un contrôle de la bordure de votre *Form* ou d'un autre contrôle, c'est une marge,
- vous êtes aligné avec un autre contrôle verticalement ou horizontalement,
- vous êtes aligné avec le texte affiché dans certains cas, comme par exemple une *TextBox* ou un *Button*.



## 2.5 Modifier les propriétés de vos contrôles à leur conception

Il est possible de toucher les propriétés d'une *Form*, d'un composant ou d'un contrôle lors de leur conception grâce à une interface pour les propriétés. Elle est accessible en faisant clic droit sur votre *Form* (ou composant ou contrôle) puis *Properties*. Une fois la fenêtre de propriété ouverte, vous pouvez directement modifier les valeurs, soit en tapant une nouvelle valeur, soit avec une liste prédéfinie.

Pour certaines méthodes, comme le Dock ou l'Anchor, la fenêtre des propriétés fournit en plus des interfaces graphiques qui permettent de mieux ajuster vos valeurs.



Properties

**button1** System.Windows.Forms.Button

DialogResult: None

Dock: None

Enabled: True

FlatAppearance

FlatStyle: Standard

Font: Microsoft Sans Serif; 8,;

ForeColor: ControlText

GenerateMember: True

Image: (none)

ImageAlign: MiddleCenter

ImageIndex: (none)

ImageKey: (none)

ImageList: (none)

Location: **47; 57**

Locked: False

Margin: 3; 3; 3; 3

MaximumSize: 0; 0

MinimumSize: 0; 0

Modifiers: Private

Padding: 0; 0; 0; 0

RightToLeft: No

Size: **75; 23**

TabIndex: **0**

TabStop: True

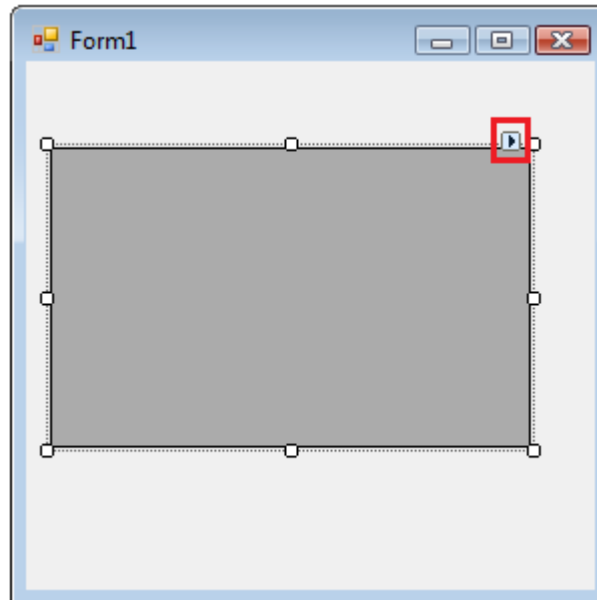
Tag:

**Text** **button1**

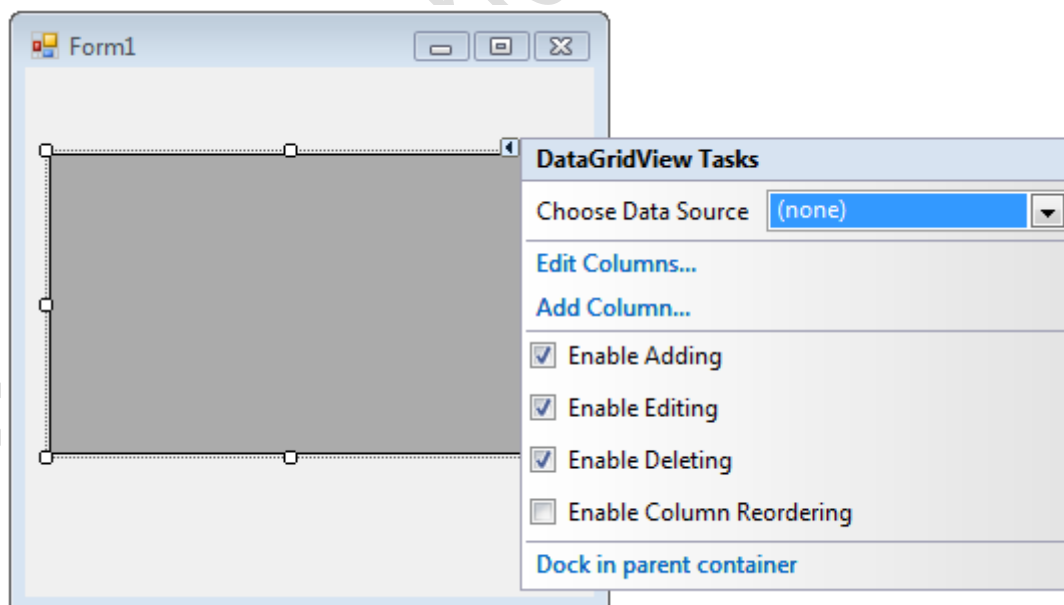
**Text**  
The text associated with the control.

## 2.6 Le Smart Tags

Certains contrôles possèdent des étiquettes intelligentes qui permettent d'avoir accès à leurs tâches les plus courantes.

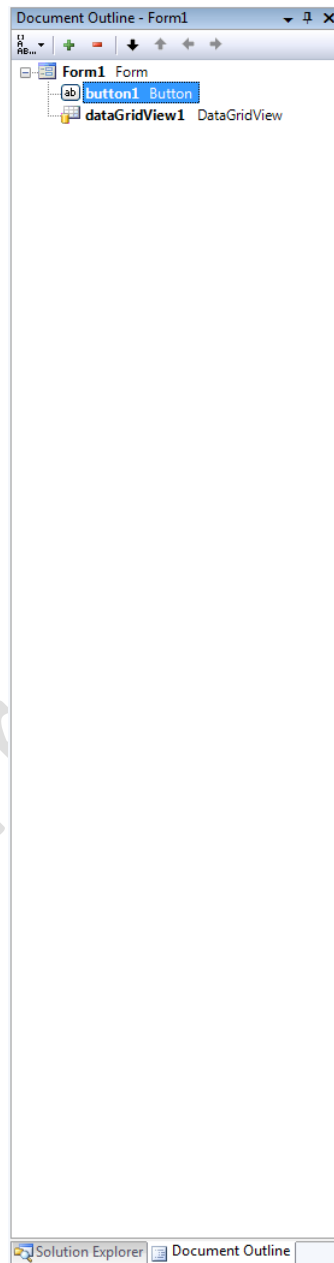


Lorsque vous cliquez dessus, une Box s'ouvrira juste en dessous, en vous présentant les tâches les plus communes.



## 2.7 Le *Document Outline*

Vos *Form* peuvent contenir plusieurs contrôles et conteneur, le *Document Outline* vous permet de mieux les gérer. Il affiche graphiquement dans une autre fenêtre (à droite sur visual studio) l'ensemble des conteneurs et contrôle existant dans votre *Form*. Vous pouvez, par exemple, déplacer vos contrôles d'un conteneur à un autre, les supprimer ou les ajouter en faisant copier/coller de la *ToolBox*.







## 3 Manipuler les affichages de textes

### 3.1 Le contrôle Bouton

Le contrôle le plus utilisé en .Net est le Bouton. En effet celui-ci permet d'interagir entre l'utilisateur et l'interface de l'utilisateur. On peut lui attribuer beaucoup de fonctions (comme fermer l'application en cours) et modifier son apparence grâce à ses propriétés et son évènement.

Voici les principales propriétés du contrôle Bouton :

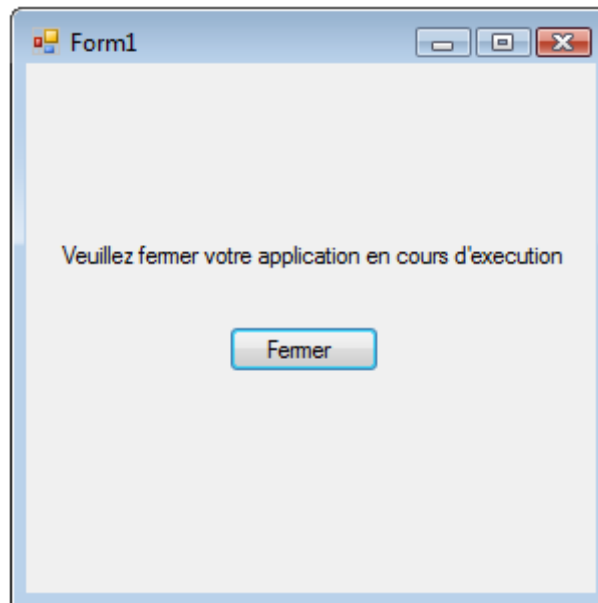
Types de Propriétés	Types de Propriétés
AutoEllipse	Permet au texte qui s'affiche sur le bouton de s'étendre au-delà de sa largeur.
DialogResult	Fixe une valeur au bouton telle que <i>Yes</i> ou encore <i>OK</i> .
FlatAppearance	Définit le design du bouton par rapport au <i>FlatStyle</i> définit (couleur...)
FlatStyle	Détermine le style de bouton, on obtient donc des effets visuels.
Text	Définit le texte qui s'affiche dans le bouton.
TextAlign	Indique comment le texte sera affiché sur le bouton ( <i>TopLeft...</i> )

Par ailleurs vous pouvez attribuer au bouton (au niveau du code) la valeur que vous voulez. Dans l'exemple ci-dessous nous montrons une application où le bouton, lorsqu'on l'exécute, ferme cette application :


```
'VB
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
Application.Exit()
End Sub

//On peut aussi écrire Me.Close() ; qui ne ferme que la forme concernée
// sauf si celle-ci est la forme principale ou parent.
```

```
//c#  
  
private void button1_Click(object sender, EventArgs e)  
{  
    Application.Exit();  
    //On peut aussi écrire this.Close() ; qui ne ferme que la forme concernée  
    //sauf si celle-ci est la forme principale ou parent.  
}
```



Exemple d'attribution de différents clicks au bouton :

➔ Dans la partie *Design*, ajouter votre bouton puis dans ses propriétés cliquez sur la partie events (bouton éclair) 

➔ Ensuite vous allez sur la propriété *MouseDown* puis vous double-cliquez dessus, vous accédez ainsi à l'événement *MouseDown* du bouton.

➔ Enfin vous pouvez taper le code qui permettra d'attribuer différents clique de souris au bouton.

```
'VB  
  
Private Sub Button1_MouseDown(ByVal sender As System.Object, ByVal e As  
System.Windows.Forms.MouseEventArgs) Handles Button1.MouseDown  
    Select Case e.Button  
        Case Windows.Forms.MouseButtons.Left  
            MsgBox("Vous avez utilisé le clique gauche")  
        Case Windows.Forms.MouseButtons.Right  
            MsgBox("Vous avez utilisé le clique droit")  
        Case Else  
            MsgBox("Vous avez utilisé un autre type de clique")  
    End Select  
End Sub
```

```
//c#

private void button1_MouseDown(object sender, MouseEventArgs e)
{
    switch (e.Button)
    {
        case MouseButtons.Left:
            MessageBox.Show("Vous avez utilisé le clique gauche");
            break;
        case MouseButtons.Right :
            MessageBox.Show("Vous avez utilisé le clique droit");
            break;
        default :
            MessageBox.Show("Vous avez utilisé un autre type de
clique");
            break;
    }
}
```

### 3.2 La propriété *FlatAppearance* et *FlatStyle*

La propriété *FlatStyle* permet de choisir différents styles de boutons (*Flat*, *Popup*, *Standard* et *System*). Les trois dernières options sont prédéfinies et donc ne peuvent être modifiées grâce à la propriété *FlatAppearance*. Donc si vous définissez la valeur *Flat* dans la propriété *FlatStyle* vous pourrez modifier l'apparence de votre bouton (couleur du bouton lorsque la souris passe sur celui-ci, épaisseur de la bordure...) :

Types de Propriétés	Description
<code>BorderColor</code>	Définit la couleur de la bordure du bouton.
<code>BorderSize</code>	Détermine la taille de la bordure du bouton.
<code>MouseDownBackColor</code>	Définit la couleur lorsque l'utilisateur clique sur le bouton.
<code>MouseOverBackColor</code>	Définit la couleur du bouton lorsque le pointeur passe sur le bouton.

Vous pouvez ainsi donner à l'utilisateur des procédures à suivre grâce à *FlatAppearance*.

### 3.3 Le contrôle *Label*

Le contrôle *Label* est utilisé essentiellement pour afficher des informations textuelles à l'utilisateur. Par exemple pour décrire à quoi fait référence un autre contrôle comme la *TextBox* (Login, password...).

Vous pouvez aussi définir des raccourcis à vos labels pour accéder aux contrôles qui leur correspondent. Pour cela il suffit de presser la touche Alt + « La Clé » c'est-à-dire le chiffre ou la lettre que vous avez défini pour accéder au contrôle :

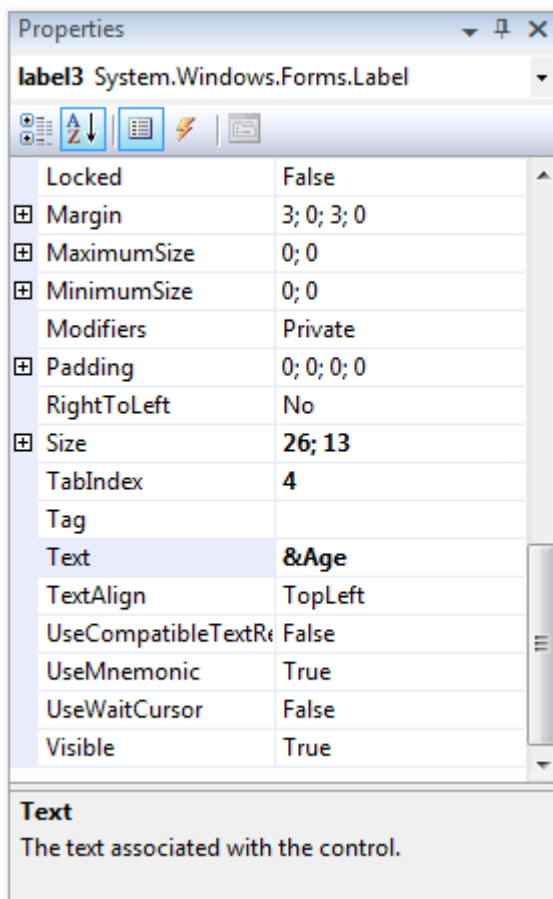
-Dans un premier temps vous glissez sur votre *Form* un label puis un contrôle qui sera lié avec ce label.

-Puis dans la propriété *Text* de votre label vous taper le texte approprié au contrôle en rajoutant **&** devant votre texte (par exemple si vous voulez que l'utilisateur entre son âge tapez dans la propriété *Text* de votre label **&Age**). Ainsi vous définissez la clé qui sera ici la lettre A.

-Ensuite faites en sorte que la propriété *TabIndex* du label et de votre contrôle soit différente et que la *TabIndex* soit supérieur à celle du label d'une valeur de 1.

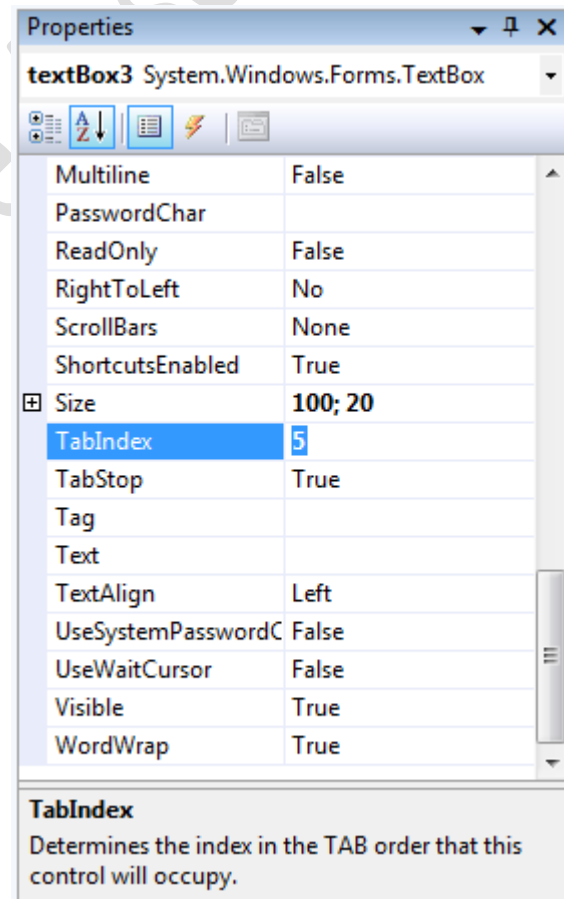
-Ainsi lorsque vous débutez votre *Form* et que vous utilisez le raccourcis **Alt+A** vous accédez au contrôle lié au label Age.

*Ci-dessous vous avez l'exemple en image des propriétés du label Age et d'un contrôle associé à ce label. Vous pouvez remarquer que le *TabIndex* du contrôle est supérieur de 1 au *TabIndex* du label.*



Properties	
label3 System.Windows.Forms.Label	
Locked	False
Margin	3; 0; 3; 0
MaximumSize	0; 0
MinimumSize	0; 0
Modifiers	Private
Padding	0; 0; 0; 0
RightToLeft	No
Size	26; 13
TabIndex	4
Tag	
Text	<b>&amp;Age</b>
TextAlign	TopLeft
UseCompatibleTextRe	False
UseMnemonic	True
UseWaitCursor	False
Visible	True

**Text**  
The text associated with the control.



Properties	
textBox3 System.Windows.Forms.TextBox	
Multiline	False
PasswordChar	
ReadOnly	False
RightToLeft	No
ScrollBars	None
ShortcutsEnabled	True
Size	100; 20
TabIndex	5
TabStop	True
Tag	
Text	
TextAlign	Left
UseSystemPasswordC	False
UseWaitCursor	False
Visible	True
WordWrap	True

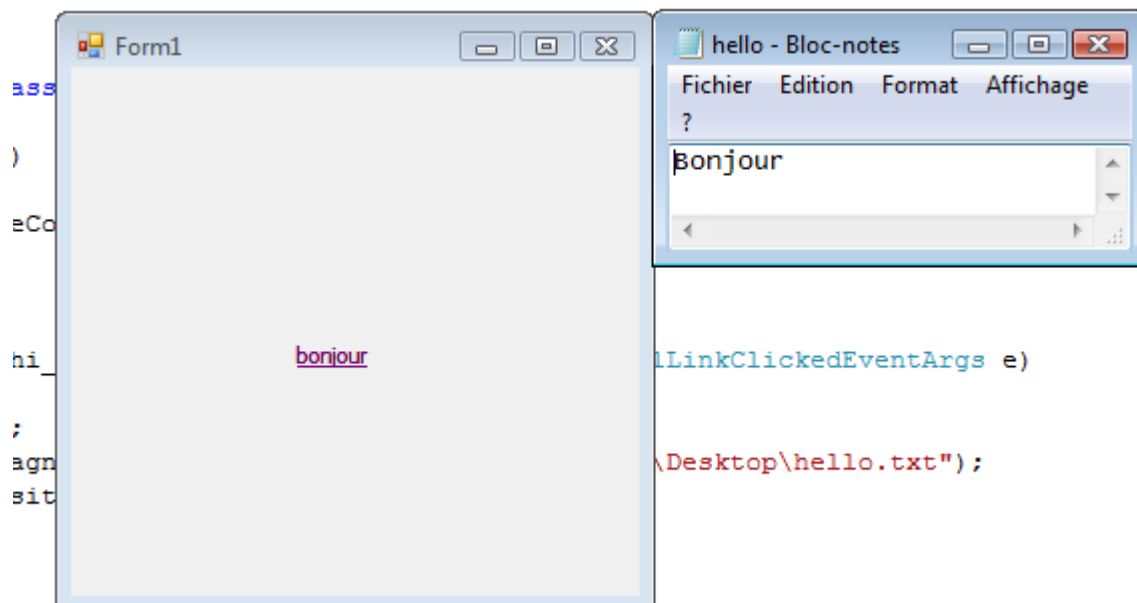
**TabIndex**  
Determines the index in the TAB order that this control will occupy.

### 3.4 Le contrôle *LinkLabel*

Le contrôle *LinkLabel* permet de créer un lien vers une page web ou effectue d'autres actions (un fichier par exemple). Ce contrôle contient plusieurs propriétés, voici les principales :

Types de Propriétés	Description
<code>ActiveLinkColor</code>	Détermine la couleur du lien lorsqu'on clique sur celui-ci
<code>LinkArea</code>	Détermine la zone de lien dans le <code>LinkLabel</code>
<code>LinkBehavior</code>	Définit le comportement du lien lorsque celui-ci est cliqué ou lorsqu'on passe le curseur sur celui-ci.
<code>LinkColor</code>	Définit la couleur du lien avant que celui-ci ne soit cliqué
<code>LinkVisited</code>	Indique si le lien a été visité
<code>VisitedLinkColor</code>	Indique la couleur du lien lorsque celui-ci a été cliqué

Remarque : Si vous voulez que votre lien affiche un dossier ou un fichier il faut entrer le chemin absolu du document ou du fichier (exemple : @"C:\Users\Jean\Desktop\cours.txt"). Le caractère @ permet d'entrer correctement les \ (antislash).



Voici la méthode pour créer un lien vers une page Internet ([www.dotnet-france.fr](http://www.dotnet-france.fr)) :

➔ Tout d'abord ajoutez un contrôle *LinkLabel* à votre *Form* dans la partie *Design* de celle-ci. Définissez le nom de référence du *LinkLabel* dans la propriété (*Name*) puis le texte dans la propriété *Text* (exemple : dotNet-france).

➔ Ensuite double-cliquez sur votre contrôle dans la partie *Design* de la *Form* pour accéder à l'événement de celui-ci.

➔ Puis tapez le code suivant :

```
'VB
Private Sub DotNet_LinkClicked(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventArgs) Handles
DotNet.LinkClicked

    ' On affiche la Form qui contient le lien
    DotNet.Show()
    ' On définit vers quel site le lien va renvoyer
    System.Diagnostics.Process.Start("www.dotnet-france.fr")
    ' On indique que le lien une fois cliqué a été visité
    DotNet.LinkVisited = True

End Sub

//C#
private void DotNet_LinkClicked_1(object sender,
LinkLabelLinkClickedEventArgs e)
{
    // on affiche la Form qui contient le lien
    DotNet.Show();
    // on définit vers quel site le lien va renvoyer
    System.Diagnostics.Process.Start(@"www.dotnet-france.fr");
    // On indique que le lien une fois cliqué a été visité
    DotNet.LinkVisited = true;
}
```

## 4 Utiliser le contrôle Text Edit

### 4.1 La TextBox

La *TextBox* est le contrôle le plus utilisé pour recevoir du texte d'un utilisateur. C'est une zone qui permet de recevoir et d'afficher le texte que l'utilisateur inscrit. Ces zones peuvent afficher un

nombre limité de caractère, afficher plusieurs lignes ou encore afficher des étoiles (ou points) à la place des caractères lors d'une saisie de mot de passe.

La *TextBox* modifie son comportement par plusieurs propriétés. Voici les principales propriétés d'une *TextBox* :

Types de Propriétés	Description
<i>AutoCompleteCustomSource</i>	Il conserve une collection de chaînes de caractères qui contient les fichiers d'auto complétion lorsque que l' <i>AutoCompleteMode</i> n'est pas sur <i>None</i> et l' <i>AutoCompleteSource</i> est sur <i>Custom</i> .
<i>AutoCompleteMode</i>	Il définit le mode d'auto complétion. Il a pour valeur <i>None</i> , <i>Append</i> , <i>Suggest</i> et <i>Suggest-Append</i> .
<i>AutoCompleteSource</i>	Il définit la source des fichiers d'auto complétion. Cette source peut être parmi une source du système ou une source fournie par l' <i>AutoCompleteCustomSource</i> .
<i>CharacterCasing</i>	Il indique la forme des caractères (majuscule, minuscule). Il a pour valeur <i>Normal</i> , <i>Upper</i> (majuscule) et <i>Lower</i> (minuscule).
<i>Lines</i>	Lorsque <i>MultiLine</i> est sur <i>True</i> , <i>Lines</i> permet d'afficher des caractères sur plusieurs lignes, caractères qui seront affichés dans la <i>TextBox</i> lors du lancement de l'application.
<i>Maxlenght</i>	Il définit le nombre de caractères maximum que l'utilisateur peut insérer.
<i>Multiline</i>	Il permet à la <i>TextBox</i> d'avoir une ligne unique ou plusieurs avec les valeurs <i>False</i> et <i>True</i> .
<i>PasswordChar</i>	Il permet d'afficher le caractère que vous définissez à la place des caractères que l'utilisateur tape. Il faut que le <i>Multiline</i> soit sur <i>False</i> .
<i>ReadOnly</i>	Il définit si le texte inscrit dans la <i>TextBox</i> peut être modifié ou pas.
<i>ScrollBars</i>	Il permet, lorsqu'une <i>TextBox</i> permet le <i>MultiLine</i> , d'avoir une barre de défilement.

<b>Text</b>	Il définit ou récupère le texte inscrit dans la <i>TextBox</i> .
<b>UseSystemPasswordChar</b>	Il permet de mettre des points à la place du caractère défini dans le <i>PasswordChar</i> .
<b>WordWrap</b>	Il permet de bloquer une ligne à la taille de la <i>TextBox</i> . Il faut que l'option <i>MultiLine</i> soit sur <i>True</i> .

Toutes ces propriétés sont accessibles dès la conception de la *TextBox*.

## 4.2 La *MaskedTextBox*

La *MaskedTextBox* est une *TextBox* avec d'autres propriétés qu'une *TextBox* classique. Entre autre elle permet de définir si son contenu doit être accepté ou refusé. Par exemple, on peut lui définir de n'accepter que les chiffres.

Son comportement peut être modifié dès sa conception par plusieurs propriétés. Voici ses principales propriétés :

Types de Propriétés	Description
<b>AsciiOnly</b>	Il définit que seul les caractères ASCII peuvent être rentrés, c'est-à-dire entre A-Z et a-z.
<b>BeepOnError</b>	Il définit si la <i>MaskedTextBox</i> envoie un signal sonore chaque fois qu'une entrée est refusée.
<b>CutCopyMaskFormat</b>	Il détermine si les caractères littéraux sont inclus lorsque le texte est copié ou coupé.
<b>InsertKeyMode</b>	Il définit ou récupère le mode d'insertion de texte pour la <i>MaskedTextBox</i> .
<b>Mask</b>	Il définit le masque pour la <i>MaskedTextBox</i> .
<b>ResetOnSpace</b>	Détermine comment un espace doit être traité.



### 4.3 La propriété Mask

Cette propriété permet de définir le format de la chaîne de caractères entrée dans le *MaskedTextBox*. Voici les éléments qui permettent d'obtenir différents formats :

Élément de Mask	Description
0	Il représente un chiffre nécessairement entre 0 et 9.
9	Il représente un chiffre qui peut être entre 0 et 9.
#	Il représente un chiffre qui peut être entre 0 et 9 ou un espace.
L	Il représente une lettre nécessairement entre a-z et A-Z.
?	Il représente une lettre qui peut être entre a-z et A-Z.
ℓ	Il représente un caractère. Si <i>AsciiOnly</i> est sur <i>True</i> , cela revient à L.
C	Il représente optionnellement un caractère. Si <i>AsciiOnly</i> est sur <i>True</i> , cela revient à L?.
A, a	Il représente optionnellement un caractère alphanumérique.
.	Il représente un caractère décimal.
:	Il représente une séparation de temps.
/	Il représente une séparation de date.
\$	Il représente un symbole courant.
<	Il convertit tous les caractères en minuscules.
>	Il convertit tous les caractères en majuscules.
	Il permet d'annuler un « < » ou « > ».

## 5 Conclusion

Vous avez fini votre second chapitre en *Windows Form* ! Vous savez à présent manipuler les contrôles couramment utilisés. Le mieux reste de créer des petits projets pour tester les différentes possibilités offertes par ces contrôles. N'oubliez également pas que le site du MSDN peut vous apporter une aide complémentaire précieuse.

L'équipe *Windows Form*

Dotnet-France Association