Introduction de Windows Form

Sommaire

1 Introduction

Dans ce premier chapitre vous vous initierez à la manipulation des *Formulaires* ainsi que les propriétés qui les composent. Vous découvrirez comment créer un Formulaire non-rectangulaire ou encore personnaliser votre *Formulaire* (police, bordure, boite de contrôle…) de façon intuitive ou en manipulant le code de votre projet (en Visual Basic ou en C#). Ainsi vous passerez d'une fenêtre banale à une fenêtre moderne et qui vous ressemble.

Bon cours .Net

L'équipe *Windows Form*

2 Gérer une Windows Form

2.1 Créer un nouveau projet ou insérer un nouveau Formulaire dans votre projet

Lors de sa création, votre projet contient, par défaut, une *Windows Form* (une fenêtre vide) appelée *Form1.cs*. Bien entendu, un projet peut contenir une multitude de *Windows Form*, ce qui est même en général le cas.

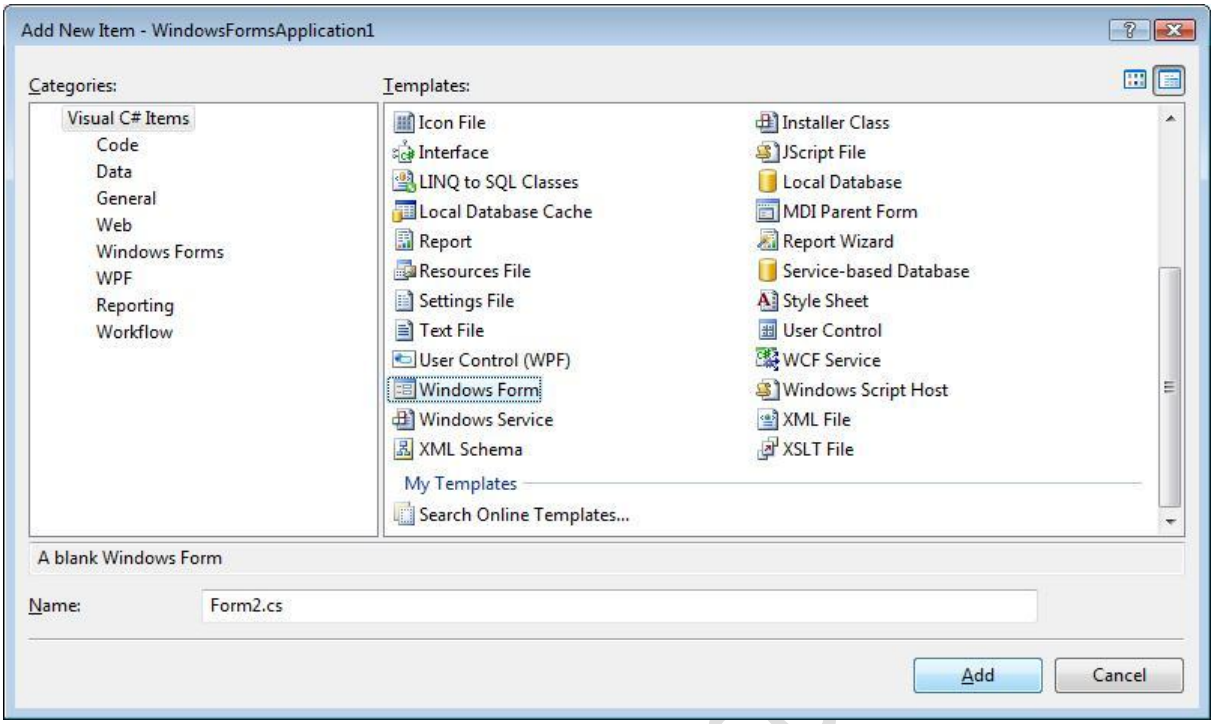
- Créer votre projet

Après avoir ouvert Visual Studio, allez dans *Fichier*, *Nouveau* et cliquez sur *Projet* (il est également possible d'utiliser le raccourci Ctrl+Shift+N). Une nouvelle fenêtre s'ouvre, sélectionnez *Application Windows Form* et nommez votre projet. Votre projet sera enregistré dans C:\Users*UserName*\Documents\Visual Studio 2008\Projects (par défaut).

- Ajouter un nouveau *Formulaire* à votre projet

Au départ, votre projet est constitué d'une *Windows Form*, la *Form1.cs* citée précédemment. En faisant un clique-droit sur *Form1* puis en sélectionnant *Voir Code*, vous passerez de la partie *Design* à la partie *Code* de votre fenêtre. De plus si vous double cliquez sur votre *Formulaire* vous accèderez à l'évènement de celle-ci.

Vous pouvez créer d'autres *Formulaires* sur votre projet. Pour cela, faites clique droit sur votre *Projet* (dans Solution Explorer), sélectionnez *Ajouter*, *Nouvel élément* puis sélectionnez *Windows Form* comme sur l'image suivante. Vous pouvez aussi personnaliser le nom de votre *Formulaire*.



2.2 Découvrir les propriétés des Windows Forms

Les *Windows Form* contiennent de nombreuses propriétés permettant de modifier de nombreux éléments tels que les tailles, les couleurs, etc.…

Pour afficher ces propriétés, faites clique droit sur votre *Formulaire* puis *Properties*. Elles s'affichent sur un onglet en bas à droite de l'écran.

Voici les principales propriétés que vous pouvez retrouver :

Types de Propriétés	Description
(Name)	Cette propriété définit le nom de la classe que vous utilisez pour créer votre *Form*.
Backcolor	Elle indique la couleur de fond du *Formulaire*.
BackgroundImage	Elle permet d'insérer une image en fond.
BackgroundImageLayout	Permet de placer l'image si celle-ci est sélectionnée grâce à la propriété précédente *BackgroundImage.*
ControlBox	Détermine si le *Formulaire* contient un système de contrôle de la fenêtre

	(agrandir, minimiser, réduire et fermer).
Cursor	Indique comment le curseur apparaît lorsqu'il se déplace sur le *Formulaire*.
Enabled	Détermine si le *Formulaire* est capable de recevoir les données saisies par l'utilisateur. Si cette propriété est déterminée sur *False* l'utilisateur ne pourra saisir aucunes données dans le *Formulaire* et l'utilisateur ne pourra pas interagir avec ce *Formulaire*.
Font	Définit la police du *Formulaire*.
ForeColor	Indique la couleur de texte utilisée dans le *Formulaire*.
FormBorderStyle	Indique le type de bordure du *Formulaire*.
HelpButton	Indique si le *Formulaire* possède un bouton Aide.
Icon	Définit l'icône qui représente le *Formulaire*.
Location	Lorsque la propriété *StartPosition* est définie sur *Manuel*, *Location* permet de déterminer la position du *Formulaire* lorsque celui-ci sera exécuté.
MaximizeBox	Permet de définir si l'utilisateur peut agrandir le *Formulaire* ou non.
MaximumSize	Définit la taille maximale du *Formulaire* lors de son agrandissement.
MinimizeBox	Permet de définir si l'utilisateur peut minimiser le *Formulaire* ou non.
MinimumSize	Définit la taille minimale du *Formulaire* lors de sa réduction.
Opacity	Représente l'opacité du *Formulaire*. Elle est déterminée selon un pourcentage. 0% correspond à la transparence complète du *Formulaire* et 100% à une forme opaque complète.
Size	Définit la taille initiale du *Formulaire*.

StartPosition	Indique la position du *Formulaire* lors de son affichage.
Text	Détermine le texte de la légende du Formulaire.
TopMost	Indique si la forme apparaît au-dessus des autres *Formulaires* qui n'ont pas cette propriété. Pour cela il faut la définir à *True*.
Visible	Détermine si le *Formulaire* est visible lors de l'exécution.
WindowState	Détermine si le *Formulaire* est maximisée, minimisée ou si la taille est définie lors de l'exécution.

2.3 Déterminer le statut de votre *Formulaire* lors du premier démarrage

La propriété *WindowState* permet de déterminer le statut du *Formulaire* lors du premier démarrage de celle-ci. On peut lui attribuer 3 valeurs : *Normal*, *Minimized* and *Maximized*. Par défaut la propriété est *Normal*. *Minimized* permettra de réduire la *Form* qui se trouvera par conséquent dans la barre des tâches, *Maximized* agrandira au maximum le *Formulaire*.

Même si cette option est réglable au niveau du code, cela sera sans effet sur le *Formulaire*. Il est donc plus judicieux de définir cette propriété au niveau des propriétés lors de sa conception.

2.4 Situer la position de votre *Formulaire* au démarrage

L'emplacement au démarrage du *Formulaire* est déterminé par une propriété.

Cette propriété est la *StartPosition* qui définit l'emplacement du *Formulaire* au démarrage.

Cette propriété peut-être définie par n'importe quelles valeurs contenues dans l'énumération *FormStartPosition* :

Valeurs	Description
Manual	La position de démarrage du *Formulaire* est définie selon la propriété *Location*.
CenterScreen	Le *Formulaire* commence au centre de l'écran.
WindowsDefaultLocation	Le Formulaire est placé selon la position de Windows par défaut et démarre à la taille déterminée par la propriété *Size*.
WindowsDefaultBounds	Le Formulaire est placé selon la position de Windows par défaut et démarre à la taille déterminée par la taille Windows par défaut.
CenterParent	La position du *Formulaire* commence au centre du *Formulaire* parent.

Exemple :

```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        Me.StartPosition = FormStartPosition.WindowsDefaultLocation

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)
{
       this.StartPosition = FormStartPosition.WindowsDefaultLocation;
}
```

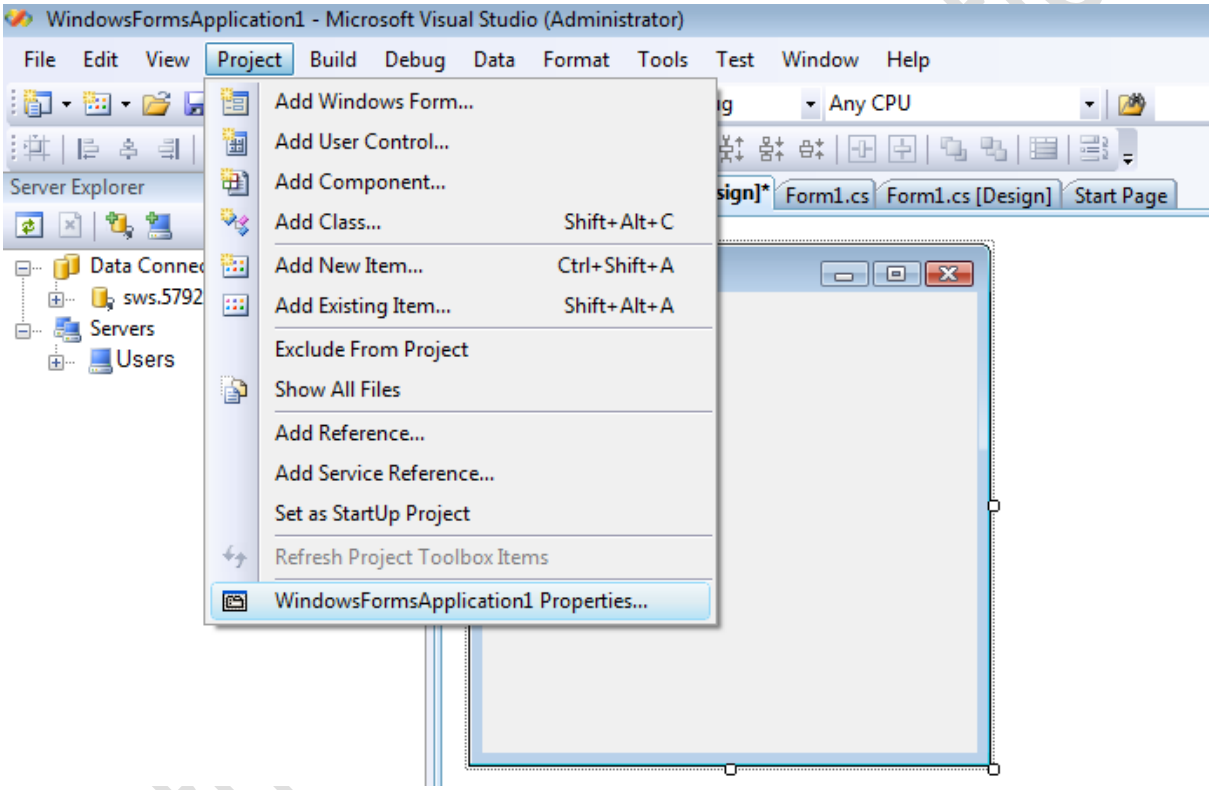
2.5 Définir votre Formulaire de démarrage

Lorsqu'une application *Windows Form* contient une multitude de Formulaire, une seule doit être désignée pour le démarrage.

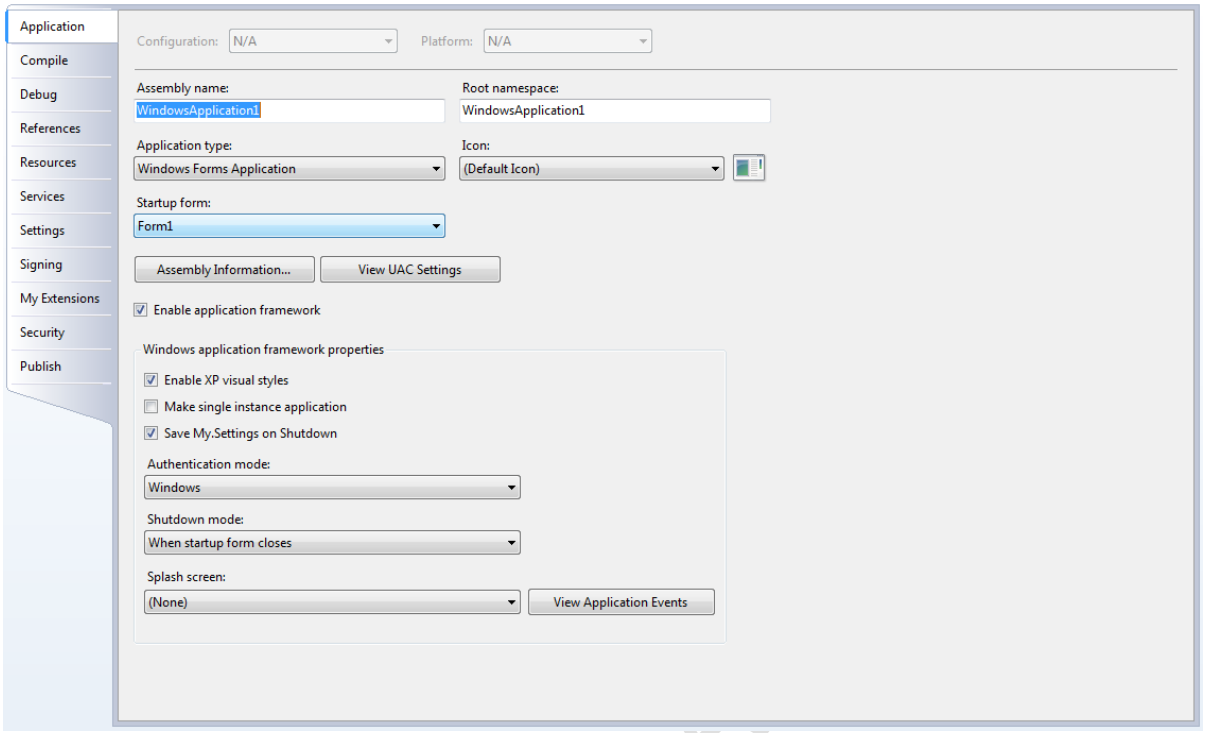
La méthode qui permet de désigner ce Formulaire est différente suivant le langage utilisé :

- Avec **Visual Basic**, vous pouvez designer le Formulaire au démarrage grâce aux propriétés de votre projet :

Pour cela aller dans la barre de menu, sélectionner *Project* puis glisser sur *(nom de votre projet) properties*.



Cliquez sur l'onglet *Application* puis dans l'option *Startupform* choisissez votre Formulaire dans le menu déroulant.



- En **C#,** vous devez définir le démarrage du Formulaire à partir du code de votre programme depuis la méthode *Main* :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }

}
```

Donc pour définir le Formulaire vous avez juste à remplacer *Form1* par le nom du Formulaire que vous voulez utiliser au démarrage.

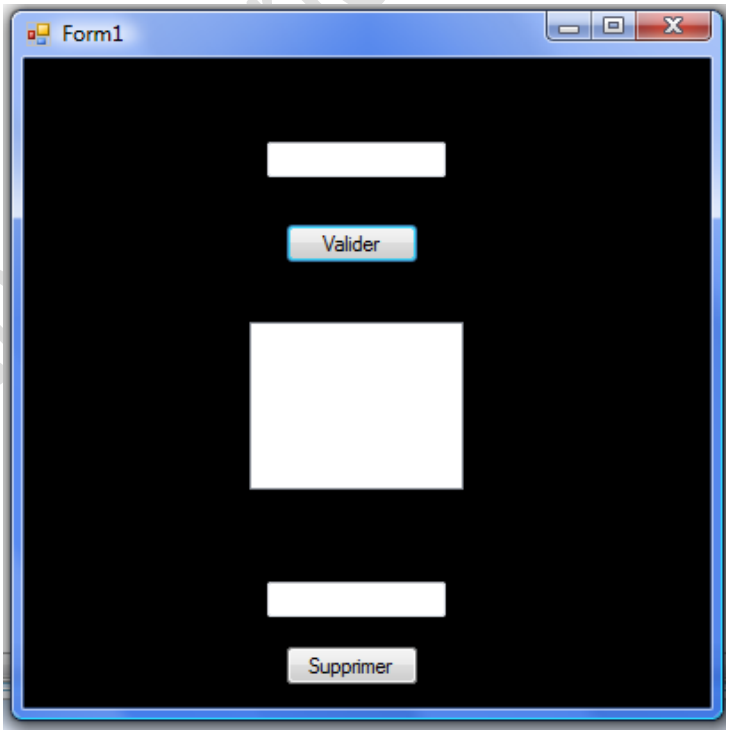
2.6 L'apparence du Formulaire

Vous pouvez modifier votre Formulaire en visuel à l'aide de la partie *Design* de celle-ci. Il suffit de modifier manuellement les propriétés.

Cependant, il est possible de modifier les apparences directement dans le code, par exemple avec une propriété *BackColor*.

```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

Me.BackColor = Color.Black;

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)
{
    this.BackColor = Color.Black;

}
```



Par contre, d'autres propriétés sont plus complexes d'utilisation. La valeur d'une propriété est représentée par l'*instance* d'une *classe* ou d'une *structure*. On peut définir cette propriété à une

instance existante de la *classe* ou bien créer une nouvelle *instance* qui spécifie toutes les valeurs de la propriété.

2.7 Configurer le style de bordure du Formulaire

Le style de la bordure de votre Formulaire désigne comment apparaît son contour et comment elle se comporte durant l'exécution, c'est à dire par exemple si l'utilisateur peut ou non modifier sa taille.

La propriété *FormBorderStyle* possède sept options :

Valeurs	Description
None	Le Formulaire n'a aucune bordure ni d'aide, ni de boite de contrôle.
FixedSingle	Le Formulaire possède un seul type de bordure (une seule ligne) et la taille ne peut être modifiée par l'utilisateur.
Fixed3D	La bordure du Formulaire est en 3D (tridimensionnelle fixe) mais ne peut être modifiée au niveau de sa taille par l'utilisateur.
FixedDialog	Le Formulaire possède une bordure épaisse, sa taille ne peut être modifiée par l'utilisateur. Elle possède l'apparence d'une boîte de dialogue.
Sizable	Paramètre par défaut dans votre Formulaire. La taille peut être modifiée par l'utilisateur.
FixedToolWindow	Le *Formulaire* contient une simple bordure et sa taille ne peut être modifiée par l'utilisateur. Seul le bouton fermer est présent dans le Formulaire.
SizableToolWindow	Le Formulaire contient une simple bordure qui est redimensionnable. Seul le bouton fermer est présent dans le Formulaire.

Ces propriétés peuvent être modifiées, soit à la conception en modifiant directement les propriétés du Formulaire, soit au moment de l'exécution, en tapant un code comme celui-ci :

```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    Me.FormBorderStyle = Windows.Forms.FormBorderStyle.FixedSingle

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)
{
    this.FormBorderStyle = FormBorderStyle.FixedSingle;
}
```

2.8 Redimensionner votre *Form*

Lorsque la valeur de la propriété *WindowState* est attribuée à *Normal*, la *Form* aura la taille définie par la propriété *Size*.

Le redimensionnement peut se faire au niveau de la conception avec la souris en tirant sur les coins. Les paramétrages se feront alors automatiquement. On peut aussi le faire en paramétrant directement les valeurs des propriétés *Height* et *Width* dans le code, comme l'exemple suivant :

```
'VB

Form1.Width = 300
Form1.Height = 200
Form1.Size = new Size(300, 200)
```

```
//C#

Form1.Width = 300;
Form1.Height = 200;
Form1.Size = new Size(300, 200);
```


Attention, si la propriété *StartPosition* du Formulaire est définie sur *WindowsDefaultBounds*, la taille du Formulaire sera la taille de la fenêtre par défaut plutôt que la taille définie par la propriété *Size*.

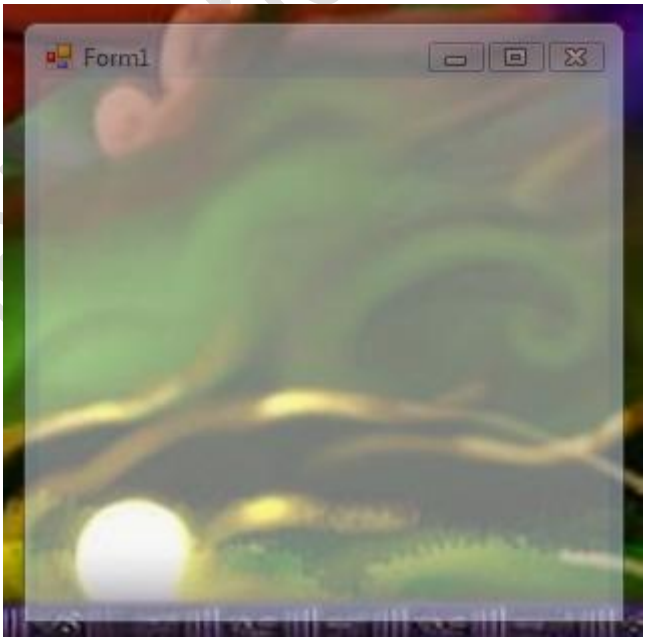
2.9 Transparence et opacité de le Formulaire

La propriété *Opacity* permet de régler l'opacité et la transparence de votre Formulaire, c'est-à-dire avec une opacité à 100%, le Formulaire sera visible et solide en apparence, alors qu'avec une opacité à 0% le Formulaire sera totalement transparent.

On peut déterminer cette propriété par les propriétés dans la conception ou bien directement dans le code comme par exemple :

```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        Me.Opacity = 0.7

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)
        {
            this.Opacity = 0.7;
        }
```



Exemple d'un Formulaire avec une opacité réduite.

2.10 Créer un Formulaire visible ou invisible

Certaines applications peuvent avoir un long processus de démarrage et il n'est pas forcément recommandé d'afficher le Formulaire de démarrage durant le déroulement du processus.

La propriété *Visible* permet de cacher le Formulaire pendant le déroulement du processus. Cette propriété peut être modifiée directement dans les propriétés lors de la conception ou bien dans le code. Dans ce dernier cas la valeur *False* permettra de cacher le *Formulaire* et la valeur *True* l'affichera, par exemple :

```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        Me.Visible = False

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)
        {
            this.Visible = false;
        }
```

Ensuite vous pouvez aussi utiliser les méthodes *Show ()* et *Hide ()*.

```
'VB

'Votre forme est visible

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

Me.Show()

End Sub

'Ou bien

'Votre forme est invisible

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load


Me.Hide ()

End Sub
```

```
//C#

//Votre forme est visible
private void Form1_Load(object sender, EventArgs e)
{
this.Show () ;

}

//Ou bien

//Votre forme est invisible
private void Form1_Load(object sender, EventArgs e)
{
this.Hide () ;

}
```

Remarque: Les méthodes *Show* et *Hide* peuvent être définie aussi sur des contrôles tels que des boutons, des Panels …

2.11 Créer un Formulaire non-rectangulaire

Les Formulaires ne sont pas forcément toujours rectangulaires. On peut créer des Formulaires ovales, ronds, etc.…

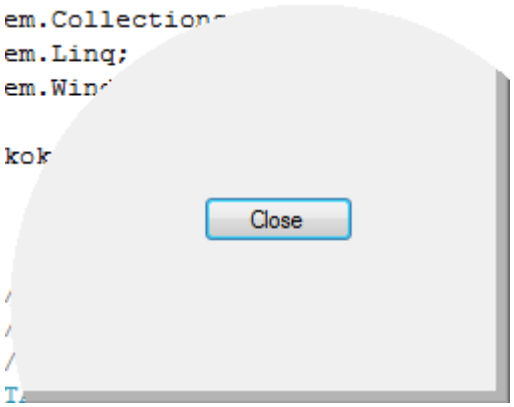
La propriété *Region* dans *Form_Load* permet de définir des Formulaires non rectangulaires. En effet le changement de forme se fait au moment de l'exécution et non au cours de sa conception, c'est-à-dire que vous ne verrez le résultat qu'une fois l'application lancée.

La propriété *Region* est une instance de *System.Drawing.Region*, elle représente une surface de l'écran qui se trouve à l'intérieur d'une forme graphique, c'est-à-dire que votre Formulaire sera découpé pour entrer dans cette forme graphique.

Pour créer facilement un Formulaire non rectangulaire, il suffit de créer une nouvelle instance de la classe *GraphicsPath* puis de créer une nouvelle *Region* dedans. A cause de ce « découpage » n'oubliez de créer un bouton Fermer pour sortir. Par exemple :

```vb
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

   Dim myPath As New System.Drawing.Drawing2D.GraphicsPath
   myPath.AddEllipse(50, 100, Me.Width, Me.Height)
   Dim myRegion As New Region (myPath)
   Me.Region = myRegion

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
           Me.Close()

End Sub
```

```csharp
//C#

private void Form1_Load(object sender, EventArgs e)
        {
            System.Drawing.Drawing2D.GraphicsPath myPath = new
System.Drawing.Drawing2D.GraphicsPath();
            myPath.AddEllipse(50, 100, this.Width, this.Height);
            Region myRegion = new Region(myPath);
            this.Region = myRegion;
        }

private void button1_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
```



3 La mise en pages avec les Contrôles Containers

3.1 Ajouter un nouveau contrôle

Il existe plusieurs façons d'ajouter un contrôle, vous pouvez ajouter votre contrôle intuitivement sur votre onglet *Form[Designer]* grâce à la *Toolbox* :

- Glissez votre contrôle depuis la Toolbox sur votre Formulaire.
- Sélectionnez votre contrôle depuis votre *Toolbox* puis dessinez votre bouton sur votre Formulaire.
- Sélectionnez votre contrôle depuis votre *Toolbox*, ensuite double-cliquez sur votre Formulaire.
- Double-cliquez sur votre contrôle depuis votre *Toolbox*.

Ensuite vous avez la possibilité de créer des contrôles depuis votre code. Dans l'exemple suivant on ajoute à un panneau simplement un bouton :

```
'VB

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'On crée un nouveau bouton
Dim aButton As New Button()
'On définit la position du bouton (largeur, Hauteur)
aButton.Location = New Point(100, 150)
aButton.Text = "OK"
'on ajoute un bouton au panneau nommé par défaut panel1
panel1.Controls.Add(aButton)
'On ajoute le bouton à la forme nommée Form1  par defaut

Me.Controls.Add(aButton)

End Sub
```

```
//C#

private void Form1_Load(object sender, EventArgs e)

{

//On crée un nouveau bouton
Button aButton = new Button();
//On définit la position du bouton (largeur, Hauteur)
aButton.Location = new Point(100, 150);
aButton.Text = "OK";
//on ajoute un bouton au panneau nommé par défaut panel1
panel1.Controls.Add(aButton);
//On ajoute le bouton à la forme nommée Form1  par defaut

this.Controls.Add(aButton);

}
```

3.2 Le contrôle de mise en page avec le Panel

Le contrôle *Panel* a pour fonction de créer une sous-partie du Formulaire afin d'y introduire d'autres contrôles. Il possède beaucoup de ressemblances avec le *GroupBox*. Cependant la particularité principale du *Panel* réside dans le fait qu'il peut posséder une barre de défilement grâce à l'option *AutoScroll.*

Nous allons vous présenter les principales propriétés du contrôle *Panel* :

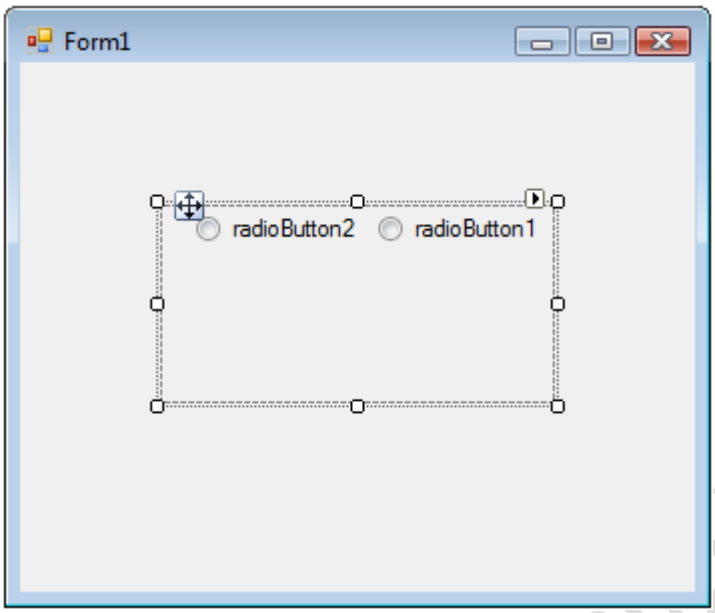
Propriétés	Description
AutoScroll	Détermine si le Panel possède une barre de défilement lorsque les contrôles dépassent la limite visible du Panel.
BorderStyle	Détermine l'apparence visuelle de la bordure de votre *Panel*. Elle peut être définie sur None pour aucun style de bordure, FixedSingle pour une simple bordure et Fixed3D pour avoir une bordure avec une apparence en 3 dimensions.
Cursor	Indique l'apparence du curseur lorsque celui-ci survole le contrôle.
Visible	Détermine si le contrôle est visible ou caché.

➢ *FlowLayoutPanel*

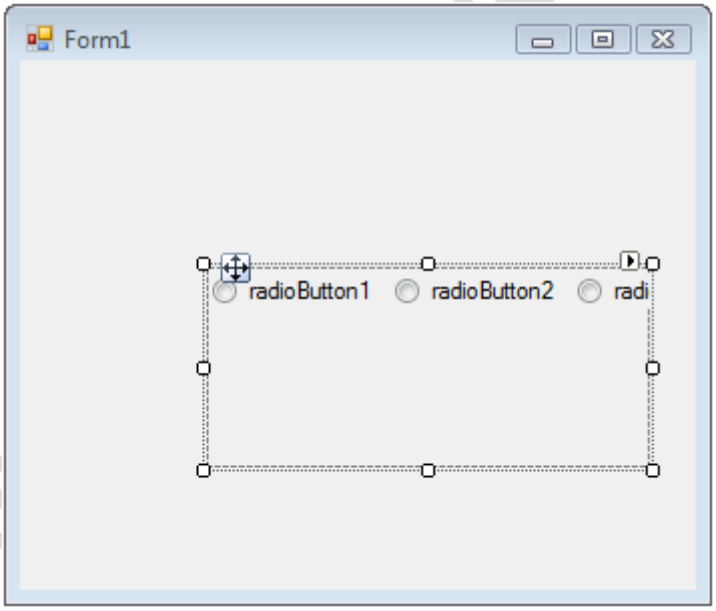
Le contrôle *FlowLayoutPanel* est une sous-classe du contrôle *Panel*. En effet celui-ci permet, contrairement au *Panel*, de réajuster dynamiquement les contrôles (horizontalement ou verticalement) qu'il contient lorsque l'utilisateur fait un redimensionnement de la forme au moment de la conception ou de l'exécution.

Dans le tableau ci-dessous vous trouverez quelques propriétés communes au Panel mais aussi des spécifiques au contrôle ainsi que deux méthodes :

Propriétés	**Description**
AutoScroll	Détermine si le Panel possède une barre de défilement lorsque les contrôles dépassent la limite visible du Panel.
FlowDirection	Détermine la direction d'écoulement des contrôles du *FlowLayoutPanel*. Elle peut prendre comme valeur : *LeftToRight*, *TopDown*, *RightToLeft* et *BottomUp*.
WrapContent	Indique si le contenu du *FlowLayoutPanel* est encapsulé ou attaché à la limite du contrôle.
BorderStyle	Indique le style de bordure du contrôle.
Cursor	Indique l'apparence du curseur lorsque celui-ci survole le contrôle.
Visible	Détermine si le contrôle est visible ou caché.
VerticalScroll	Définit les caractéristiques de la barre de défilement lorsque le contrôle a un affichage vertical.
Méthodes	**Description**
GetFlowBreak	Retourne une valeur qui représente le paramètre d'interruption du flux du contrôle *FlowLayoutPanel*.
SetFlowBreak	Définit la valeur qui représente le paramètre d'interruption du flux du contrôle *FlowLayoutPanel*.



Exemple de FlowLayoutPanel contenant 2 RadioButton où la propriété FlowDirection est définit sur RightToLeft.



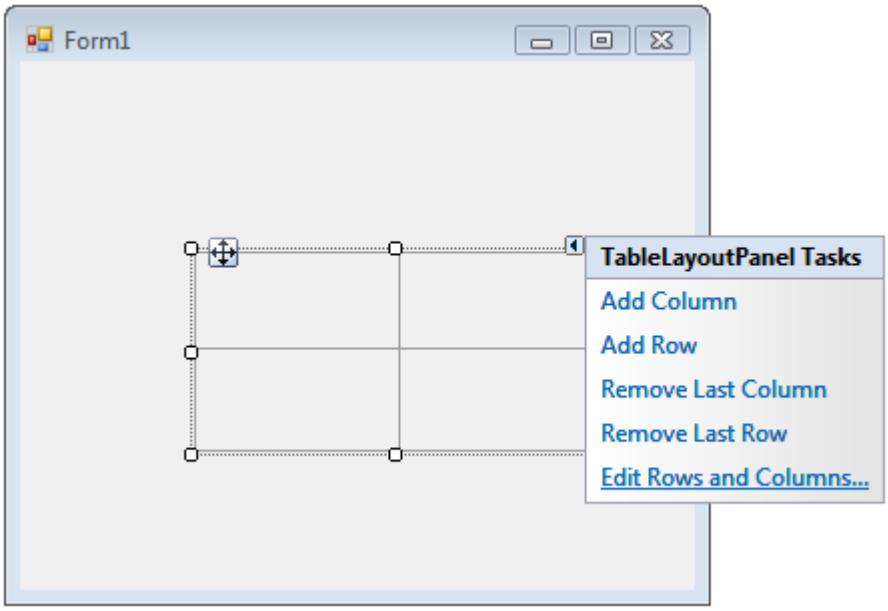
Exemple de l'utilisation de la propriété WrapContent qui est définie ici sur la valeur False.

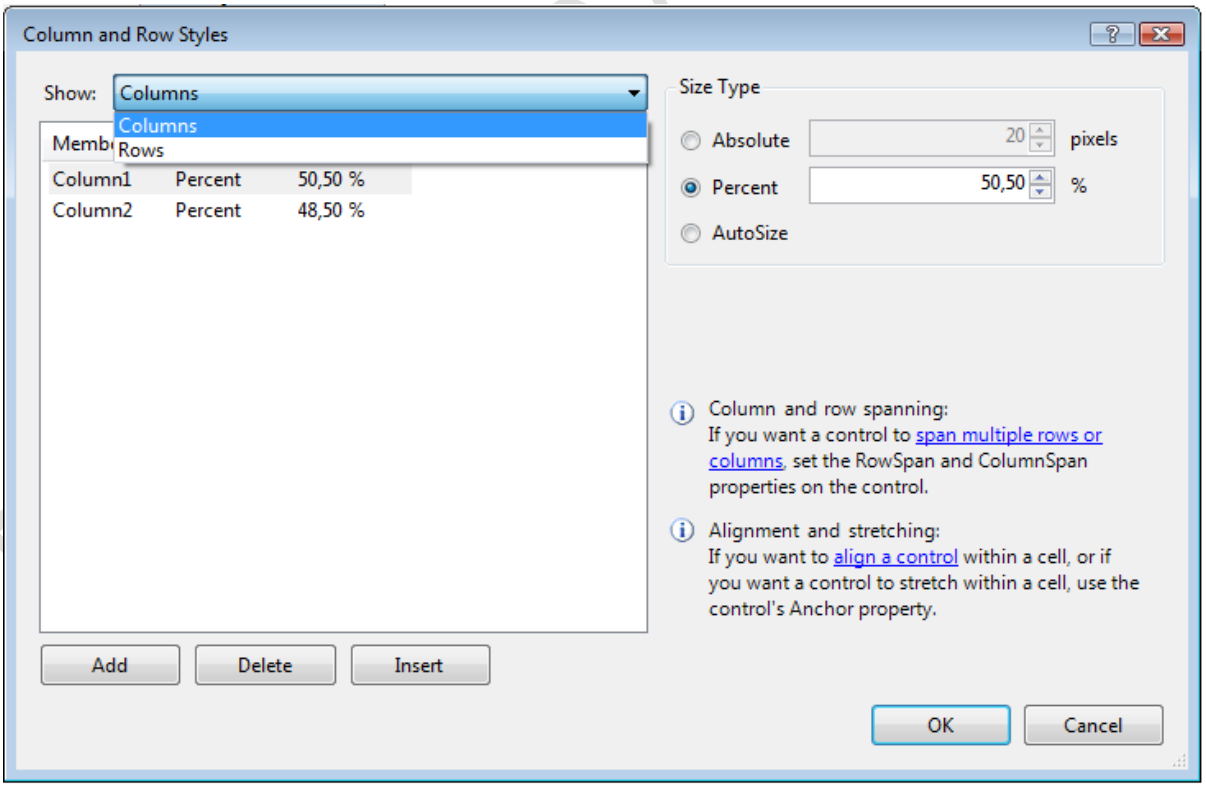
- *TableLayoutPanel*

Comme le *FlowLayoutPanel,* le *TableLayoutPanel* est une sous-classe du contrôle *Panel* qui a sa propre spécialité. En effet celui-ci contient plusieurs cellules afin de répartir individuellement les différents contrôles que vous voudriez insérer. C'est en quelque sorte un tableau contenant des contrôles dans ses lignes et colonnes (d'ou son nom).

Vous pouvez manipuler vos cellules de deux façons différentes, soit lors de la conception en sélectionnant la propriété *Edit Rows and Columns…*, soit directement dans le code en double-cliquant sur le contrôle :

De façon intuitive :





Au niveau du code :

```vb
'VB


Private Sub TableLayoutPanel1_Paint(ByVal sender As System.Object,
ByVal e As System.Windows.Forms.PaintEventArgs) Handles
TableLayoutPanel1.Paint

tableLayoutPanel1.ColumnStyles(0).SizeType = SizeType.Percent
tableLayoutPanel1.ColumnStyles(0).Width = 70
tableLayoutPanel1.RowStyles(0).SizeType = SizeType.Percent
tableLayoutPanel1.RowStyles(0).Height = 60

End Sub
```

```csharp
//C#


private void tableLayoutPanel1_Paint(object sender, PaintEventArgs e)
        {
            tableLayoutPanel1.ColumnStyles[0].SizeType =
SizeType.Percent;
            tableLayoutPanel1.ColumnStyles[0].Width = 70;
            tableLayoutPanel1.RowStyles[0].SizeType = SizeType.Percent;
            tableLayoutPanel1.RowStyles[0].Height = 60;
        }
```

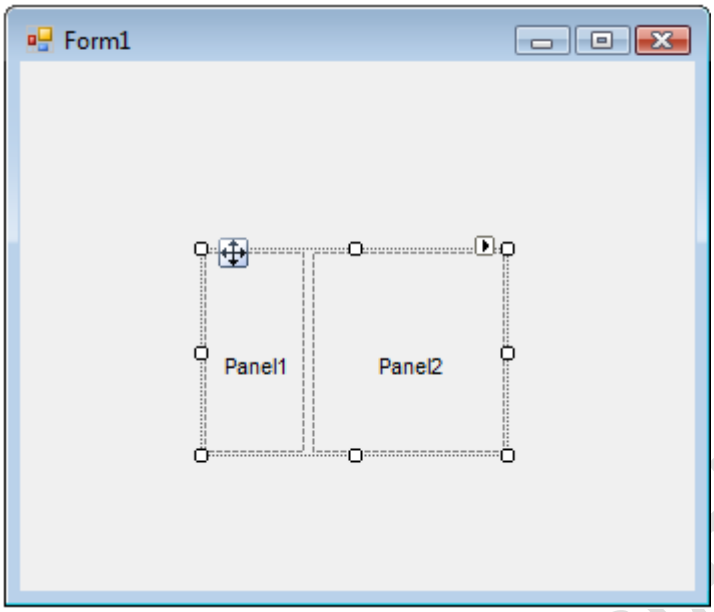
Remarque : La première colonne et la première ligne ont pour indice 0 et non 1.

Voici les différentes propriétés et méthodes du *TableLayoutPanel* :

Propriétés	Description
AutoScroll	Détermine si le contrôle possède une barre de défilement lorsque les contrôles dépassent la limite visible du *TableLayoutPanel*.
CellBorderStyle	Détermine le style de bordure des cellules. Par défaut la propriété est définie sur *None* donc il n'y pas de bordure.
ColumnCount	Indique le nombre de colonnes.
Columns	Accède aux propriétés des colonnes
ColumnStyles	Représente la liste de style de colonnes. Cette propriété n'est disponible que lors de l'exécution.
GrowStyle	Représente l'agrandissement du contrôle lorsqu'on ajoute un nouveau contrôle.
RowCount	Indique le nombre de lignes
Rows	Accède aux propriétés des lignes
RowStyles	Représente la liste de style de lignes. Cette propriété n'est disponible que lors de l'exécution.
Méthodes	**Description**
Controls.Add	Méthode permettant d'ajouter un nouveau contrôle.

3.3 Du deux en un, le SlipContainer

Cet outil vous permet de diviser votre forme ou une zone en deux Panel séparés par une barre mobile(le *Splitter*) afin d'y ajouter des contrôles. Par défaut *SplitContainer* est défini sur *Fill* afin de remplir tout votre Formulaire pour faire en sorte que celle-ci soit divisée en deux parties (cette option est modifiable dans la propriété *Dock*). Grâce à *SplitContainer* vous avez la possibilité de grouper vos autres contrôles et de les organiser selon vos choix. En effet *SplitContainer* est distinct du reste du Formulaire. Ainsi vous pourrez ajouter des contrôles en dehors des *Panels* afin qu'ils soient indépendant des contrôles se trouvant dans le *SplitContainer*.



Exemple de *SplitContainer*. Celui-ci a sa propriété *Dock* défini sur *None.*

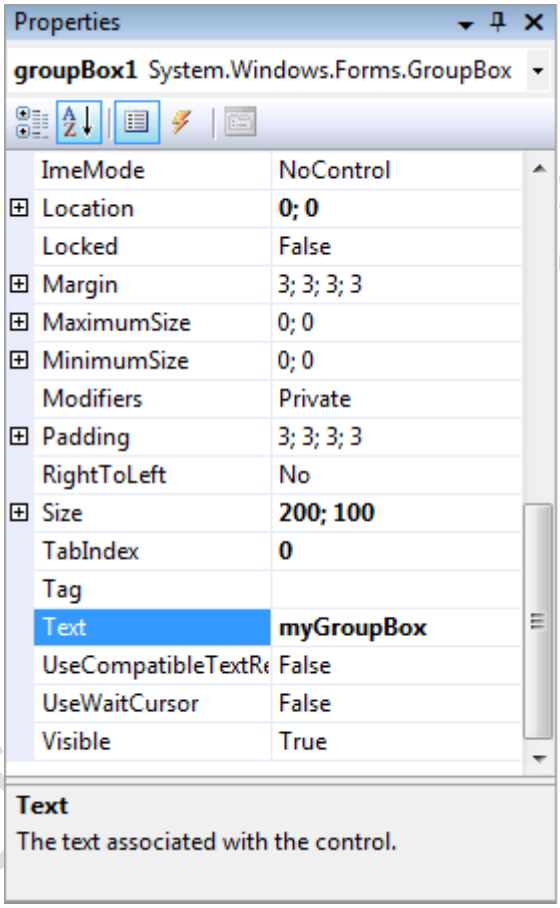
Voici les principales propriétés du contrôle SplitContainer :

Propriétés	Description
BorderStyle	Détermine l'apparence visuelle de la bordure de votre *TabControl*. Elle peut être définie sur *None* pour aucun style de bordure, *FixedSingle* pour une simple bordure et *Fixed3D* pour avoir une bordure avec une apparence en 3 dimensions.
FixedPanel	Cette propriété définit si la taille du Panel est fixe. Elle peut être réglée sur le *Panel1*, *Panel2* ou *None* pour déterminer qu'aucune taille des *Panels* ne soit fixe.
IsSplitterFixed	Détermine si le *Splitter* (la ligne qui sépare les 2 *Panels*) est fixe ou peut être modifiée par l'utilisateur.
Orientation	Attribue l'orientation des panels. Soit ils peuvent être *Vertical* (par défaut) soit ils peuvent être *Horizontal*.
Panel1	Affiche les différentes propriétés du *Panel1*.
Panel1Collapsed	Détermine si le *Panel1* est réduit ou développé.
Panel1MinSize	Attribue la taille minimum du *Panel1*.
Panel2	Affiche les différentes propriétés du *Panel2*.
Panel2Collapsed	Détermine si le *Panel2* est réduit ou développé.
Panel2MinSize	Attribue la taille minimum du *Panel2*.
SplitterDistance	Représente la distance du *Splitter* par rapport au bord du contrôle. Cette propriété dépend donc de la propriété *Orientation*.
SplitterWidth	Détermine la largeur du *Splitter*.

3.4 Le contrôle de mise en page, la GroupBox

La *GroupBox* est container qui possède une bordure et une frontière pour l'interface de l'utilisateur. Sa fonction est identique à celui du Panel. Cependant elle ne peut avoir des barres de défilement mais elle peut obtenir une légende, ce qui n'est pas le cas pour le *Panel*. Par exemple, dans les propriétés de la *GroupeBox* on peut lui définir un titre grâce à l'option *Text* (voir le screenshot ci-dessous).

En général on utilise ce contrôle afin d'y introduire des *RadioButtons*.



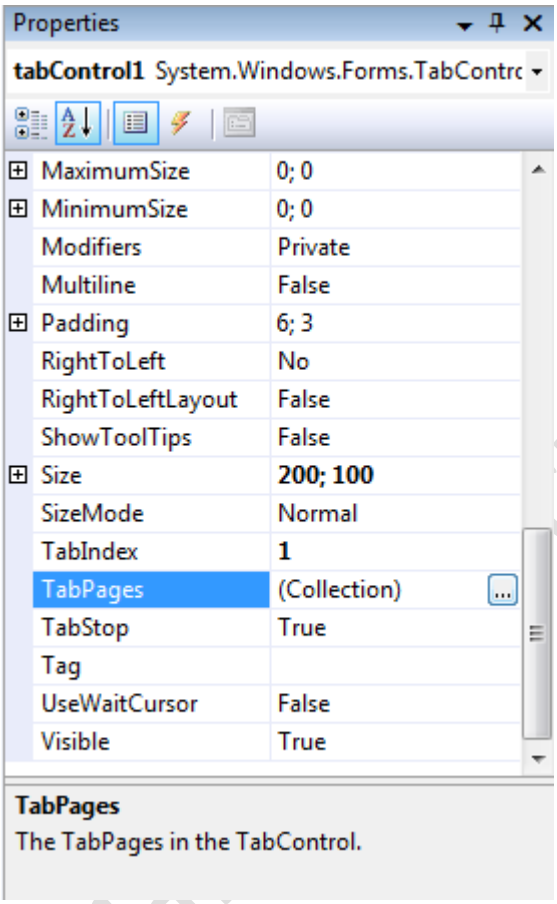
3.5 Des onglets pratique avec le TabControl

Le *TabControl* vous permet d'organiser vos contrôles ou encore des images sur différents onglets comme dans un classeur. Vous pouvez ainsi trier selon vos choix les contrôles que vous utiliserez. Ainsi l'utilisateur pourra naviguer sur les onglets du *TabControl* (donc sur les contrôles qu'il contient).

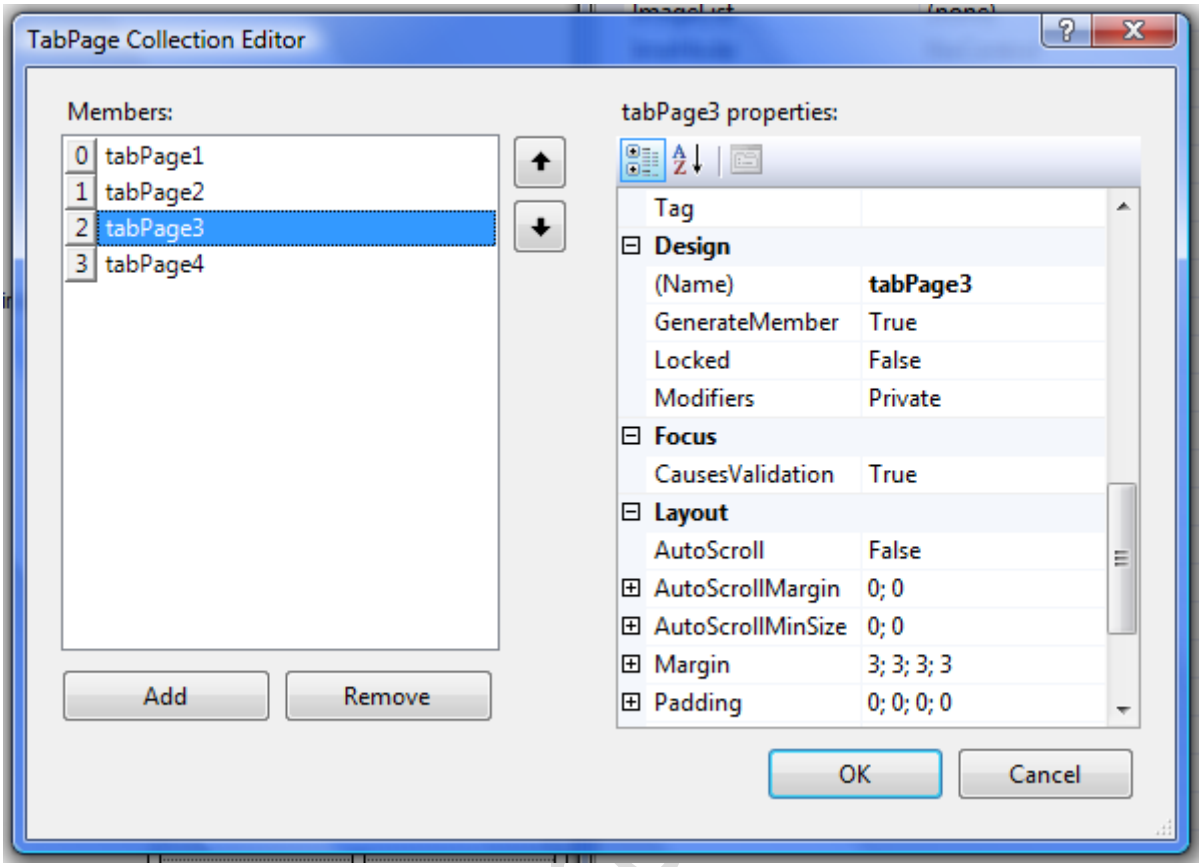
Le *TabControl* contient plusieurs propriétés utiles, nous allons donc vous présenter les plus importantes d'entre elles dans le tableau suivant :

Propriétés	Description
Appearence	Définit le style visuel de votre *TabControl*.
Alignment	Détermine si vos onglets apparaissent au *Top*, *Bottom*, *Left* or *Right* (respectivement en Haut, en Bas, à Gauche ou à Droite) de votre *TabControl*.
Multiline	Détermine si plus d'une rangée d'onglet est autorisée dans le *TabControl*.
Tabpages	Permet d'accéder aux propriétés de vos onglets.
BackgroundImage	Permet d'insérer une image dans votre contrôle.
BackgroundImageLayout	Indique la mise en forme de votre image définie dans la propriété *BackgroundImage*.
ToolTipText	Retourne un texte qui s'affiche lorsque le curseur de la souris survole l'onglet.
TabIndex	Indique l'indice d'ordre des onglets que le contrôle occupera.
TabPages	Définit la collection des onglets.
TabStop	Détermine une valeur afin que l'utilisateur obtienne le focus du contrôle grâce à la touche TAB.

Ainsi l'une des propriétés les plus importantes de la *TabControl* est la *TabPage* qui vous permet de gérer et de modifier les onglets selon vos préférences. Vous pouvez accéder ainsi aux propriétés de chaque onglet.



Alors chaque *TabPage* (ou page d'onglet) contient son propre ensemble de propriétés.



Par exemple vous pouvez attribuer à votre premier onglet une barre de défilement (pour cela vous devez attribuer la valeur *True* dans la propriété *AutoScroll* pour activer votre barre de défilement) et ne pas la définir pour votre second onglet. Vous avez aussi la possibilité d'entrer le nom de votre onglet grâce à la propriété *Text*. Par ailleurs la propriété *TabPage* contient également une propriété *BorderStyle* qui fonctionne exactement comme la propriété du contrôle *Panel*.

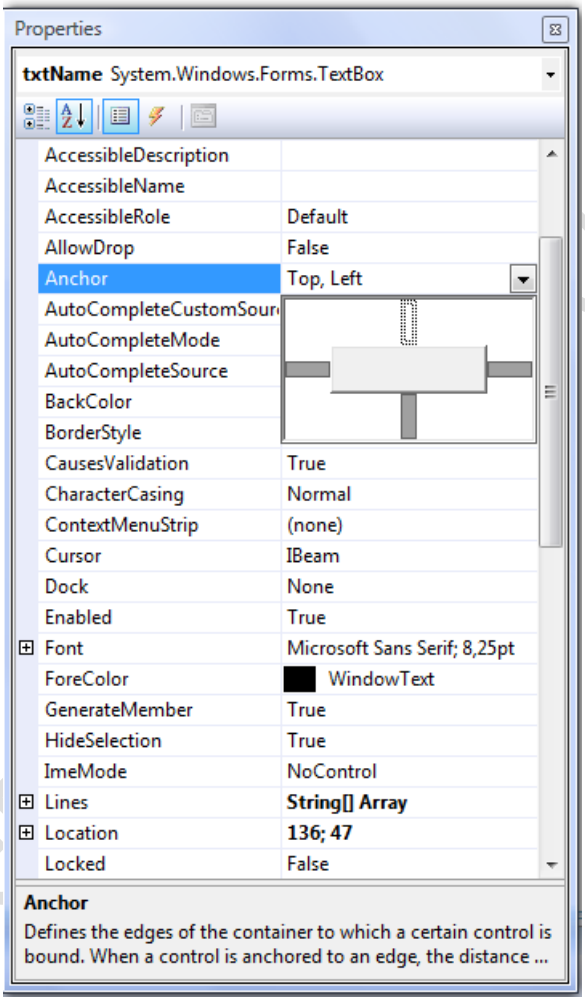
Voici donc un résumé des principales propriétés de *TabPage* dans le tableau suivant :

Propriétés	Description
AutoScroll	Définit si vous insérez une barre de défilement dans votre onglet. Sélectionner *True* pour que la barre de défilement apparaissent et *False* pour ne pas l'afficher.
BorderStyle	Détermine l'apparence visuelle de la bordure de votre *TabControl*. Elle peut être définit sur None pour aucun style de bordure, FixedSingle pour une simple bordure et Fixed3D pour avoir une bordure avec une apparence en 3 dimensions.
BackgroundImage	Permet d'insérer une image dans votre contrôle.
BackgroundImageLayout	Indique la mise en forme de votre image définie dans la propriété *BackgroundImage*.
Text	Représente le nom de votre onglet dans votre *TabControl*.

3.6 La mise en forme avec les propriétés Anchor et Dock

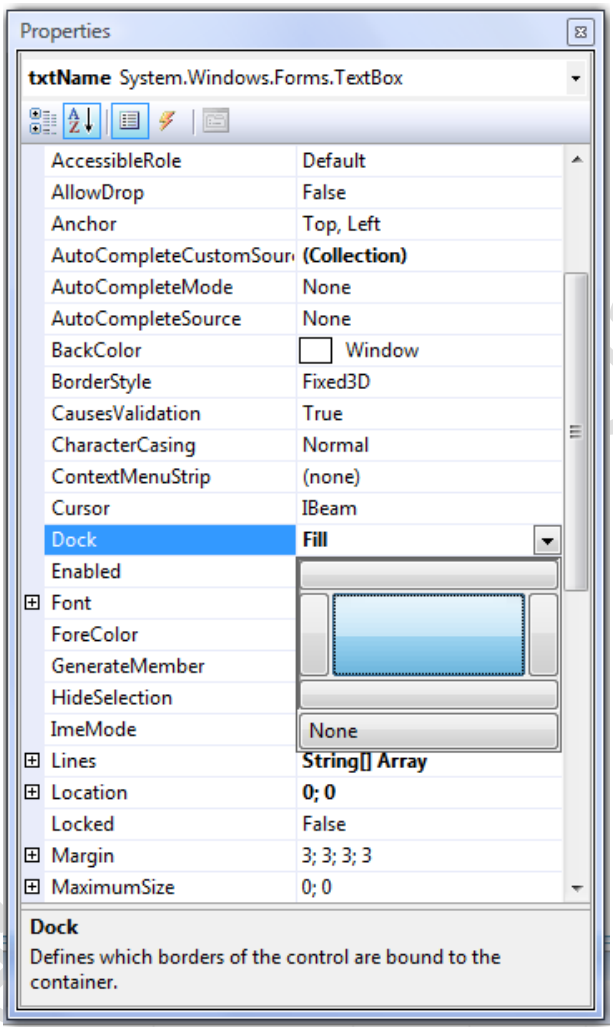
➢ Anchor

Cette propriété permet de définir comment le contrôle est redimensionné pour atteindre la même taille que son contrôle parent. Vous définissez alors avec quels bords est lié le contrôle. Donc lorsqu'il y a un redimensionnement d'un contrôle contenant un contrôle, ce dernier conservera la même position par rapport aux bords définis dans cette propriété.



- Dock

La propriété Dock permet d'avoir des effets visuels sur votre contrôle. En effet vous indiquerez dans la propriété à quelles bordures du contrôle parent le contrôle sera attaché. Ainsi le contrôle s'aligne par rapport aux bordures définies et se redimensionne automatiquement en fonction de celles-ci.



4 Conclusion

Vous avez fini votre premier chapitre en *Windows Form* ! Celui-ci vous donnera un exemple de manipulation d'un Formulaire. Il vous donnera un aperçu sur l'utilisation de Visual Studio 2008 et de ses Formulaires.

L'équipe *Windows Form*