

Drag and Drop, Timers et Globalisation

Sommaire

1	Introduction.....	3
2	Le Drag and Drop.....	3
2.1	Qu'est ce que le Drag and Drop ?	3
2.1.1	Notions générales du Drag and Drop	3
2.1.2	Le Drag and Drop en Windows Forms	3
2.2	Les évènements du Drag and Drop	4
2.2.1	Les évènements liées au Drag	5
2.2.2	Les évènements liées au Drop	6
2.3	Les « objets » de données.....	7
2.3.1	Introduction aux « objets » de données.....	7
2.3.2	Les méthodes des « objets » de données.....	7
2.3.3	Les formats des « objets » de données	7
2.4	Les éléments permettant le Drag and Drop.....	8
2.4.1	Le <i>DoDragDrop</i>	8
2.4.2	Le <i>DragDropEffects</i>	9
2.5	Exemple descriptif d'un Drag and Drop	10
2.6	Cas spécial d'utilisation du Drag and Drop : au sein d'un <i>ListView</i> et <i>TreeView</i>	12
3	L'internationalisation	12
3.1	Introduction	12
3.2	Définition de la <i>Globalization</i> et de la <i>Localization</i>	12
3.2.1	La <i>Globalization</i>	12
3.2.2	La <i>Localization</i>	12
3.3	La culture et la culture UI.....	13
3.3.1	La culture	13
3.3.2	Modifier la culture actuelle	13
3.3.3	La culture UI.....	16
3.3.4	Modifier la culture UI actuelle.....	17
3.4	Mise en page par rapport à la culture	17
3.4.1	Écriture et Lecture de droite à gauche	17



3.4.2	L'effet miroir	19
4	Les Formulaires MDI	19
4.1	Qu'est ce qu'une MDI ?	19
4.2	Utiliser le processus MDI	20
4.2.1	Créer et lier un formulaire parent et enfant	20
4.2.2	Gérer les formulaires enfants	22
4.3	Utiliser le presse-papier avec une application MDI	23
4.4	Créer un menu dans une application MDI	23
5	Conclusion	24

1 Introduction

Dans ce chapitre Drag-and-Drop & Timer, vous apprendrez à manipuler les Drag-and-Drop, l'internationalisation permettant d'avoir une application universelle et compréhensible par tous, enfin vous verrez les applications MDI avec les formulaires parents et les formulaires enfants.

2 Le Drag and Drop

2.1 Qu'est ce que le Drag and Drop ?

2.1.1 Notions générales du Drag and Drop

Le Drag and Drop porte un nom très explicite : « glisser et déposer ». Cette fonctionnalité permet donc de pouvoir saisir toutes sortes de données (images, chaîne de caractères, ...) sur votre application puis avec la souris de la déplacer ou copier vers une autre application. La méthode est assez intuitive, il suffit de faire un click gauche sur votre donnée puis tout en restant appuyé sur la touche, le déplacer puis relâcher la touche à un endroit pouvant contenir ce type de données. C'est là qu'il faudra bien faire attention au format de vos données, par exemple vous ne pourrez pas déplacer ou copier des données de types images sur une application ne pouvant contenir que des données de types chaîne de caractères.

2.1.2 Le Drag and Drop en Windows Forms

Pour bien comprendre comment fonctionne le Drag and Drop et surtout pour bien savoir l'utiliser en Windows Forms, il faut bien avoir compris la notion de contrôle ainsi que les formats qu'ils peuvent contenir, par exemple une *TextBox* contient des chaînes de caractères à la différence d'une *ImageBox* qui contient des images. Pour plus de détails sur les contrôles, voir les chapitres précédents. Ensuite, le Drag and Drop est tout simplement en Windows Forms, le déplacement ou la copie de données d'un contrôle à un autre. Il existe un grand nombre d'événements disponibles autour d'un Drag and Drop permettant par conséquent beaucoup de scénarios possibles lors d'un Drag and Drop. Ces événements sont divisés en deux styles (décrit dans la prochaine partie) : les événements liés au Drag et ceux liés au Drop.

Attention, lors d'un Drag and Drop, avec pour paramètre « Déplacer », dans le même système de stockage, les données seront déplacées (une sorte de « Couper/Coller ») ; en revanche lors de cette même opération d'un UNC (Universal Naming Convention) vers un autre les données seront copiées (une sorte de « Copier/Coller ») : (exemple : « C:\MonFichier » vers « D:\MonDossier\ »).

De plus, le Drag and Drop prend en charge le presse-papier qui permet les opérations les plus courantes tel qu'un « Copier/Coller ». Enfin, le Drag and Drop fonctionne parfaitement entre vos applications Windows Form et une toute autre application : par exemple lors d'un Drag and Drop entre un *Label* de votre application et un Bloc Note.

2.2 Les évènements du Drag and Drop

Comme dit dans la partie 1.1.2, il existe beaucoup d'évènement. Les évènements liés au Drag sont les évènements qui se déclenchent à partir du contrôle source alors que les évènements liés au Drop sont déclenchés à partir du contrôle ciblé.

2.2.1 Les événements liés au Drag

Voici les événements se déclenchant au niveau du contrôle source :

Évènement	Description
<code>GiveFeedBack</code>	Cet événement permet de modifier l'apparence du curseur de la souris lorsqu'une opération de Drag and Drop est lancée. Il s'agit d'un événement « Bubbling »* (dit bouillonnant).
<code>MouseDown</code>	Cet événement se déclenche lorsque le bouton de la souris est appuyé avec le curseur sur le contrôle source.
<code>PreviewGiveFeedBack</code>	Il s'agit de la version tunnel de l'évènement <code>GiveFeedBack</code> .
<code>PreviewQueryContinueDrag</code>	Il s'agit de la version tunnel de l'évènement <code>QueryContinueDrag</code> .
<code>QueryContinueDrag</code>	Cet événement permet d'annuler une opération Drag and Drop lorsqu'il y a modification du comportement de la souris ou du clavier durant celle-ci. Il s'agit d'un événement « Bubbling »*.

* Évènement Bubbling : C'est lorsque l'on cherche à programmer de façon événementielle dans une hiérarchie d'objet, c'est un événement qui touche un fils qui peut se mettre à remonter chez tous ses parents.

2.2.2 Les événements liés au Drop

Voici les événements se déclenchant au niveau du contrôle ciblé :

Évènement	Description
DragDrop	Cet événement se produit lorsque le bouton de la souris est relâché avec le curseur de celle-ci sur le contrôle <u>ciblé</u> . Il s'agit d'un événement « Bubbling »*.
DragEnter	Cet événement se produit lorsqu'un objet est déplacé sur un contrôle <u>lié</u> . Il s'agit d'un événement « Bubbling »*.
DragLeave	Cet événement se produit lorsqu'un objet est déplacé hors d'un contrôle <u>lié</u> . Il s'agit d'un événement « Bubbling »*.
DragOver	Cet événement se produit lorsqu'un objet est déplacé sur un contrôle <u>ciblé</u> . Il s'agit d'un événement « Bubbling »*.
PreviewDragEnter	Il s'agit de la version tunnel de l'évènement <i>DragEnter</i> .
PreviewDragLeave	Il s'agit de la version tunnel de l'évènement <i>DragLeave</i> .
PreviewDragOver	Il s'agit de la version tunnel de l'évènement <i>DragOver</i> .
PreviewDragDrop	Il s'agit de la version tunnel de l'évènement <i>DragDrop</i> .

Remarque : Le deuxième paramètre demandé, pour la méthode des événements *DragDrop*, *DragEnter* et *DragOver*, est appelé gestionnaire d'états d'évènements (contenant les informations relative à un événement levé qui doit être spécifique à celui-ci); ici il faut changer *EventArgs* en *DragEventArgs*.

2.3 Les « objets » de données

2.3.1 Introduction aux « objets » de données

Comme dit précédemment, le Drag and Drop permet de copier ou déplacer des données. Pour cela, ces données sont stockées dans des « objets » de données. Ces objets sont constitués de deux choses : les données qu'ils contiennent et leur format.

Remarque : un objet peut contenir plusieurs formats, c'est-à-dire qu'il peut fournir des données sous différents formats.

2.3.2 Les méthodes des « objets » de données

Voici les méthodes permettant de gérer un objet de données :

Méthode	Description
GetData	Elle permet de récupérer un objet de données dans un format donné.
GetDataPresent	Elle permet de vérifier si des données sont disponibles ou si elles peuvent être converties en un format donné.
GetFormats	Elle permet de retourner la liste de formats de données stockées dans un objet de données.
SetData	Elle permet de stocker dans un objet de données des données spécifiées.

Remarque : Windows Form propose une classe *DataObject*, se trouvant dans l'espace de nom *System.Windows.Forms*, qui permet de faire appel à ces méthodes. Dans d'autres styles d'applications, vous devrez implémenter l'interface *IDataObject* afin de pouvoir vous en servir.

2.3.3 Les formats des « objets » de données

Les objets de données possèdent un dispositif qui leur permet automatiquement de convertir des données d'un format à un autre. Le processus est simple, les données sont

stockées dans cet objet de données dans un certain format, puis l'objet de données change leur format lors de l'extraction des données. Vous pouvez aller sur ce lien pour avoir une liste complète des formats existant dans la classe *DataFormats* :

[http://msdn.microsoft.com/fr-fr/library/system.windows.dataformats\(VS.85\).aspx](http://msdn.microsoft.com/fr-fr/library/system.windows.dataformats(VS.85).aspx)

Remarque : Vous pouvez empêcher l'auto-conversion des données en paramétrant les méthodes *GetDataPresent* ou *GetFormats* en mettant *False* sur le paramètre *Auto-convert*.

2.4 Les éléments permettant le Drag and Drop

2.4.1 Le *DoDragDrop*

Le *DoDragDrop* est une méthode de l'espace de noms *System.Windows.Forms* qui permet de démarrer une opération de Drag and Drop. Il retourne pour valeur l'énumération *DragDropeffects* (il sera décrit dans la partie suivante) qui sera le résultat final de l'opération.

2.4.2 Le *DragDropEffects*

L'énumération *DragDropEffects* donne la valeur finale d'une opération de Drag and Drop. Il peut retourner six valeurs différentes :

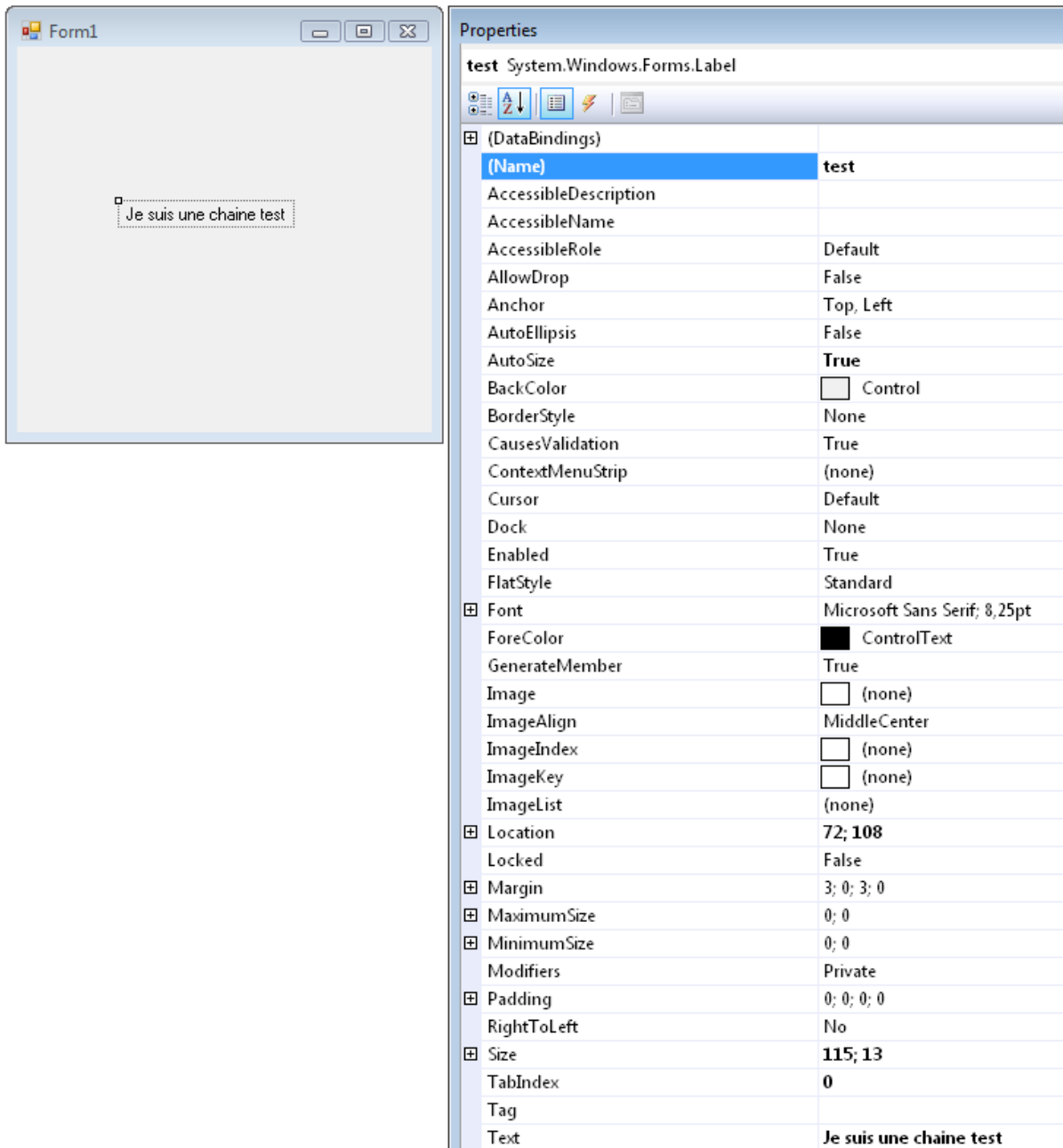
Valeur	Description
All	Cette valeur signifie que les données ont bien été transférées : supprimées du contrôle source et bien ajoutées au contrôle cible.
Copy	Cette valeur signifie que les données ont bien été copiées dans le contrôle cible.
Link	Cette valeur signifie que les données du contrôle source sont liées aux données du contrôle cible.
Move	Cette valeur signifie que les données ont bien été déplacées vers le contrôle cible.
None	Cette valeur signifie que le contrôle cible n'accepte pas les données.
Scroll	Cette valeur signifie qu'il est possible de faire défiler une cible pendant le Drag. Par exemple, vous souhaitez transférer des données de type chaîne de caractères vers un fichier BlocNote, dans ce cas vous pourrez défiler dans le BlocNote pour bien cibler l'endroit où le transfert doit avoir lieu.

Vous pouvez utiliser le *DragDropEffects* pour modifier le curseur de votre souris, c'est-à-dire qu'il prendra des formes différentes pour chaque action : défiler, déplacer, copier, ...

Remarque : Tous les événements décrits précédemment dans les parties 1.2.1 et 1.2.2 font parti des éléments incontournables pour le bon déroulement d'une opération de Drag and Drop.

2.5 Exemple descriptif d'un Drag and Drop

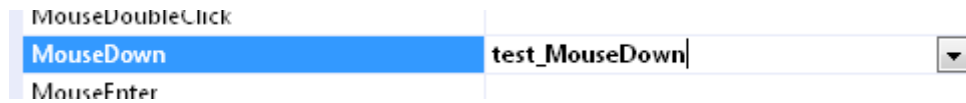
Dans cette partie, nous allons monter pas à pas comment créer et exécuter un Drag and Drop en copiant une chaîne de caractère d'un Label vers un fichier texte. Tout d'abord, il faut créer notre Label puis le renommer, ici nous le nommerons "test" ; ensuite nous écrirons comme chaîne de caractères « Je suis une chaîne test » :



The screenshot shows a Windows Forms application window titled 'Form1'. Inside the window, there is a single label with the text 'Je suis une chaîne test'. To the right of the window is the 'Properties' window, which displays the properties of the selected label. The label's name is 'test' and its text is 'Je suis une chaîne test'.

Properties	
test System.Windows.Forms.Label	
<div> <div>(DataBindings)</div> <div> <div>(Name)</div> <div>test</div> </div> </div>	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	True
BackColor	<input type="color"/> Control
BorderStyle	None
CausesValidation	True
ContextMenuStrip	(none)
Cursor	Default
Dock	None
Enabled	True
FlatStyle	Standard
Font	Microsoft Sans Serif; 8,25pt
ForeColor	<input type="color"/> ControlText
GenerateMember	True
Image	<input type="image"/> (none)
ImageAlign	MiddleCenter
ImageIndex	<input type="image"/> (none)
ImageKey	<input type="image"/> (none)
ImageList	(none)
Location	72; 108
Locked	False
Margin	3; 0; 3; 0
MaximumSize	0; 0
MinimumSize	0; 0
Modifiers	Private
Padding	0; 0; 0; 0
RightToLeft	No
Size	115; 13
TabIndex	0
Tag	
Text	Je suis une chaîne test

Ensuite toujours dans les propriétés de votre Label "test" nous allons créer l'événement « *MouseDown* » qui lui sera associé :



Ensuite il suffira d'écrire ce code :

```
'VB
Private Sub test_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles test.MouseDown
    test.DoDragDrop(test.Text, DragDropEffects.Copy)
End Sub
```

```
//C#
private void test_MouseDown(object sender, MouseEventArgs e)
{
    test.DoDragDrop(test.Text, DragDropEffects.Copy);
}
```

Vous n'avez plus qu'à tester votre Drag and Drop comme expliqué au début de ce chapitre :

Je suis une chaine test



Vous pourrez constater le changement de forme de votre curseur avec un « + » lorsque vous êtes face à une cible pouvant contenir une chaîne de caractère et un panneau interdit dans le cas contraire.

2.6 Cas spécial d'utilisation du Drag and Drop : au sein d'un *ListView* et *TreeView*

Le Drag and Drop dans un *TreeView* ou *ListView* fonctionne quasiment comme un Drag and Drop classique. La différence réside dans la partie Drag de l'opération, lorsqu'on Drag un élément du *TreeView* ou *ListView*, l'évènement *ItemDrag* se déclenche. C'est cet évènement qui va permettre d'effectuer les tâches nécessaires pour le déplacement d'éléments. Ceci va instancier le gestionnaire *ItemDragEventArgs* qui contient une référence au *TreeNode* (qui permet la gestion des nœuds dans un *ListView* ou *TreeView*) qui va permettre de copier les données dans un objet de données. Le reste des opérations nécessaires au Drag and Drop se poursuit de façon classique.

3 L'internationalisation

3.1 Introduction

Une application adaptée à toutes les cultures géographiques est mondialisée et localisée. Ce qui permet d'avoir une application internationale et compréhensible par tous grâce à Visual Studio.

3.2 Définition de la *Globalization* et de la *Localization*

3.2.1 La *Globalization*

La *Globalization* (ou encore Globalisation/Mondialisation) est donc le processus permettant un design et une programmation d'une application pour plusieurs cultures ou avec plusieurs paramètres régionaux. Donc ce processus permet d'adapter la mise en forme des données en fonction de la culture appropriée. Par exemple pour les caractères, la monnaie ou encore le format d'heure des zones géographiques spécifiques.

3.2.2 La *Localization*

La *Localization* (ou encore Localisation) consiste à adapter une application globalisée/mondialisée à une culture ou à des paramètres régionaux caractéristiques. Par exemple ce processus adapte l'interface de l'utilisateur et les illustrations en fonction des paramètres culturels ou régionaux.

3.3 La culture et la culture UI

3.3.1 La culture

La culture contient un ensemble d'information sur la langue concernée (caractères, format d'heure,...). Dans le Framework .NET chaque culture est représentée par un code appelé *culture code*. Un *culture code* spécifiant que la langue est une culture alors qu'un *culture code* spécifiant la langue et la région en paramètre est une culture spécifique. De plus dans un culture code vous pouvez préciser les jeux de caractères utilisés (Latin, Cyrillique,...). Tous ces *cultures codes* se trouvent dans leur classe respective *CultureInfo*.

Vous apercevrez ci-dessous différents exemples de *cultures codes* :

Cultures Codes	Signification
fr	Français
fr-BE	Français Belge
az-Latn-AZ	Azéris en alphabet Latin Azerbaïdjan
nn-NO	Norvégien Nynorsk Norvège

Pour obtenir toute la liste des *cultures codes* et de plus amples informations sur la classe *CultureInfo*, vous devez aller sur le site [MSDN](http://msdn.microsoft.com/fr-fr/library/system.globalization.cultureinfo.aspx).

3.3.2 Modifier la culture actuelle

Pour configurer la culture actuelle, vous devez tout simplement définir la propriété *CurrentThread.CurrentCulture* de la classe *Thread* comme étant une nouvelle instance de la classe *CultureInfo* avec en paramètre le *culture code*. Vous devez aussi insérer les deux espaces de noms *System.Globalization* et *System.Threading*. Dans l'exemple ci-dessous vous trouverez une configuration de la culture espagnole-mexicaine.

```
'VB
Imports System.Threading
Imports System.Globalization

Thread.CurrentThread.CurrentCulture = New CultureInfo("es-MX", False)
```

```
//c#
using System.Threading;
using System.Globalization;

Thread.CurrentThread.CurrentCulture = new CultureInfo("es-MX", false);
```

Ou encore vous pouvez voir ci-dessous du code permettant d'afficher des dates selon différents formats et la culture choisie, c'est de la *Globalization* :

```
'VB
Imports System.Globalization

Public Class Form1

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

'On crée une nouvelle instance de la classe CultureInfo en récupérant les
dates de la classe DateTimeFormatInfo
Dim mesDates As DateTimeFormatInfo = New CultureInfo("en-US",
False).DateTimeFormat

'On défini une date à notre objet maDate de la classe DateTime
Dim maDate As DateTime = New DateTime(2008, 8, 14)

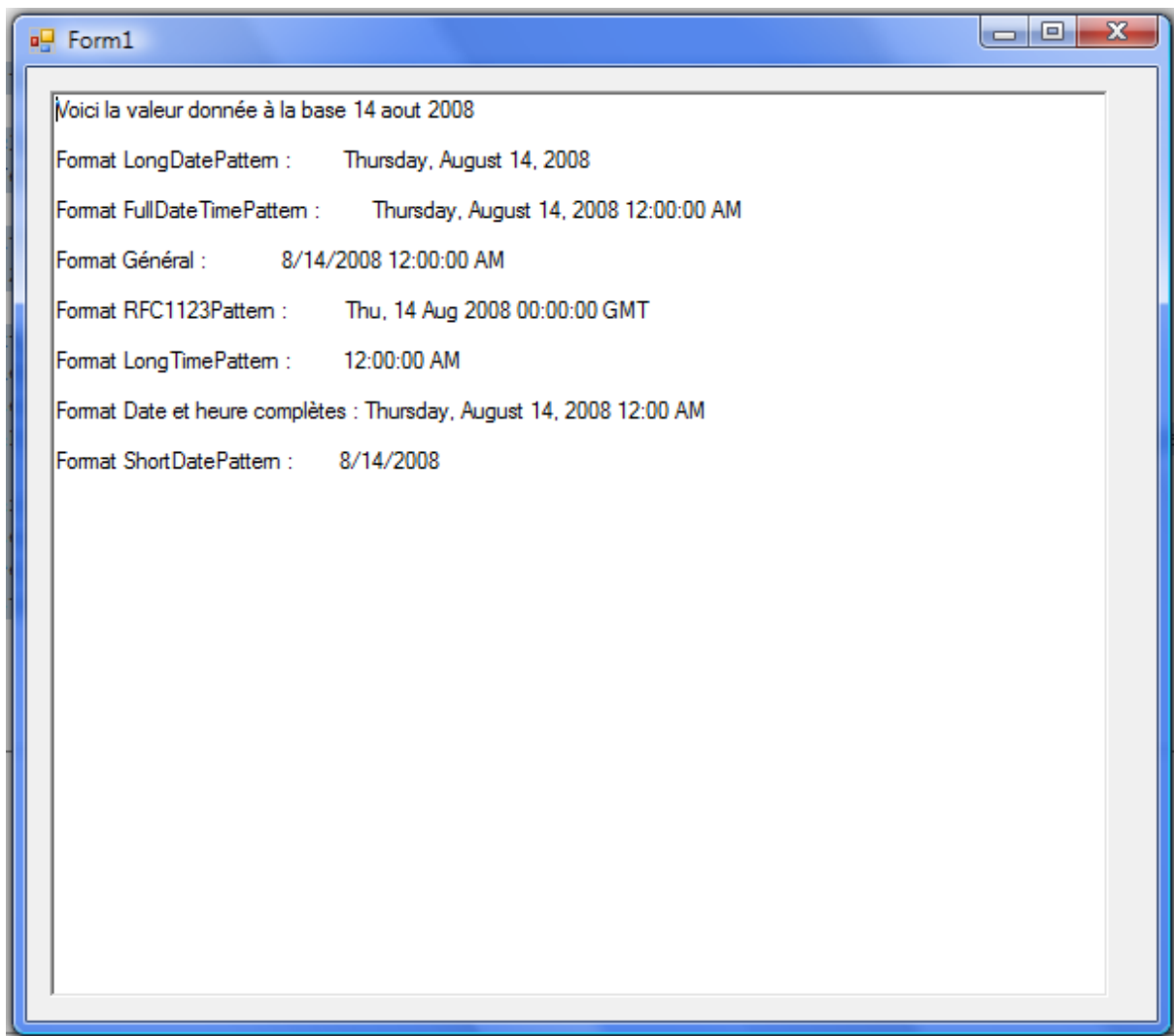
'Enfin on ajoute la date à une RichTextBox selon différents formats
maTextBox.Text = "Voici la valeur donnée à la base 14 aout 2008 " +
vbNewLine + vbNewLine
maTextBox.Text += "Format LongDatePattern : " +
maDate.ToString("D", mesDates) + vbNewLine + vbNewLine
maTextBox.Text += "Format FullDateTimePattern : " +
maDate.ToString("F", mesDates) + vbNewLine + vbNewLine
maTextBox.Text += "Format Général : " + maDate.ToString("G",
mesDates) + vbNewLine + vbNewLine
maTextBox.Text += "Format RFC1123Pattern : " +
maDate.ToString("R", mesDates) + vbNewLine + vbNewLine
maTextBox.Text += "Format LongTimePattern : " +
maDate.ToString("T", mesDates) + vbNewLine + vbNewLine
maTextBox.Text += "Format Date et heure complètes : " +
maDate.ToString("f", mesDates) + vbNewLine + vbNewLine
maTextBox.Text += "Format ShortDatePattern : " +
maDate.ToString("d", mesDates) + vbNewLine + vbNewLine

End Sub

End Class
```

```
//c#  
  
using System.Globalization  
  
private void Form1_Load(object sender, EventArgs e)  
{  
    //On crée une nouvelle instance de la classe CultureInfo en récupérant  
    //les dates de la classe DateTimeFormatInfo  
    DateTimeFormatInfo mesDates = new CultureInfo("en-US",  
false).DateTimeFormat;  
  
    //On définit une date à notre objet maDate de la classe DateTime  
    DateTime maDate = new DateTime(2008, 8, 14);  
  
    //Enfin on ajoute la date à une RichTextBox selon différents formats  
    maTextBox.Text = "Voici la valeur donnée à la base 14 aout 2008 \n\n";  
    maTextBox.Text += "Format LongDatePattern : " +  
maDate.ToString("D", mesDates) + "\n\n";  
    maTextBox.Text += "Format FullDateTimePattern : " +  
maDate.ToString("F", mesDates) + "\n\n";  
    maTextBox.Text += "Format Général : " +  
maDate.ToString("G", mesDates) + "\n\n";  
    maTextBox.Text += "Format RFC1123Pattern : " +  
maDate.ToString("R", mesDates) + "\n\n";  
    maTextBox.Text += "Format LongTimePattern : " +  
maDate.ToString("T", mesDates) + "\n\n";  
    maTextBox.Text += "Format Date et heure complètes : " +  
maDate.ToString("f", mesDates) + "\n\n";  
    maTextBox.Text += "Format ShortDatePattern : " +  
maDate.ToString("d", mesDates) + "\n\n";  
}
```

On a donc pour résultat lors du débogage :



Remarque : Vous avez pu constater que nous avons utilisé la classe `DateTime` dans le but d'utiliser des formats de date en fonction de la culture (d'où l'utilisation de la classe `CultureInfo`). De plus il existe un répertoire de tous les formats de la classe `DateTime`.

3.3.3 La culture UI

Comme la culture, la culture interface utilisateur (UI) est une instance de la classe `CultureInfo` mais la propriété `CurrentUICulture` est différente de `CurrentCulture`. En effet cette propriété recherche les ressources spécifiques à la culture au moment de l'exécution de l'application. Si ces ressources spécifiques sont indisponibles alors la propriété retourne la culture par défaut.

3.3.4 Modifier la culture UI actuelle

La configuration de la culture de l'interface utilisateur s'opère de la même manière que la culture.

```
'VB
Imports System.Threading
Imports System.Globalization

Thread.CurrentThread.CurrentUICulture = New CultureInfo("es-MX", False)
```

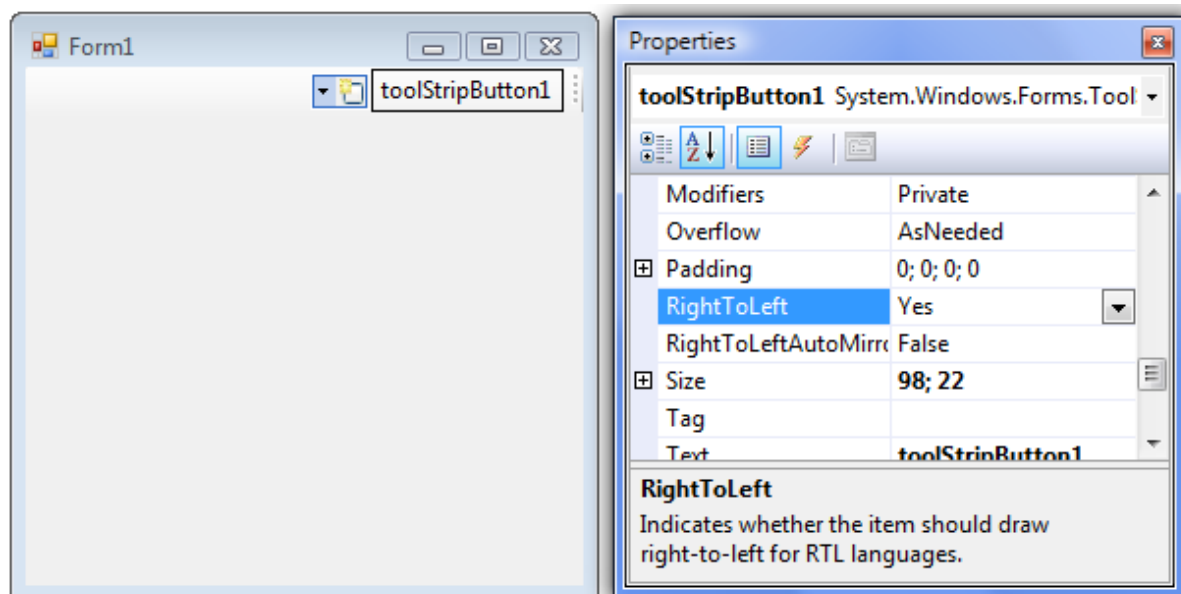
```
//C#
using System.Threading;
using System.Globalization;

Thread.CurrentThread.CurrentUICulture = new CultureInfo("es-MX", false);
```

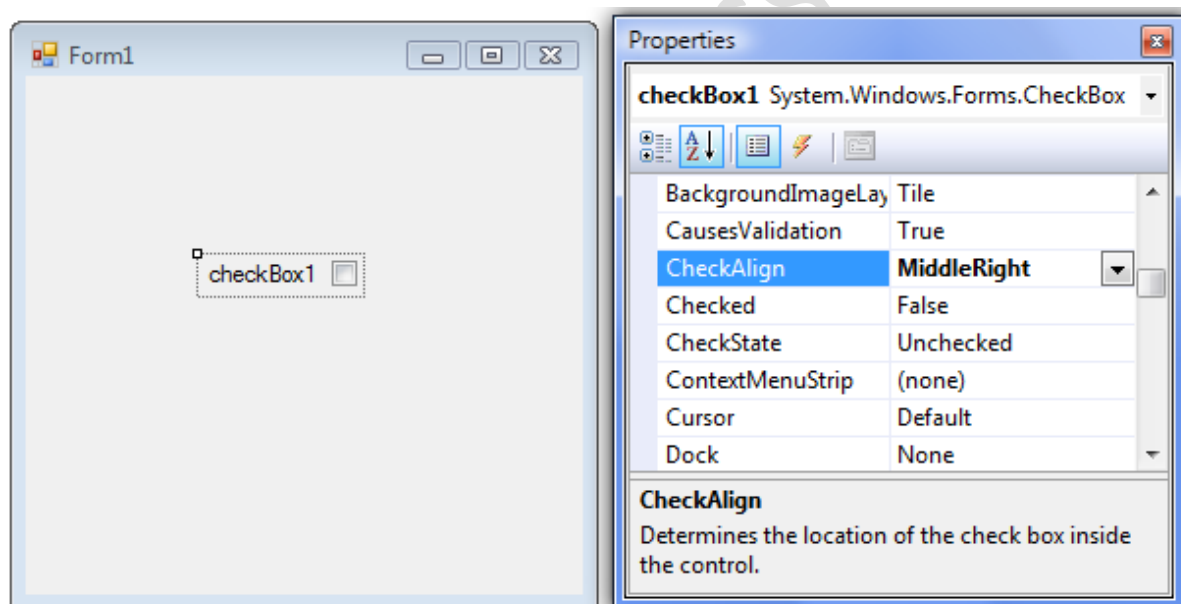
3.4 Mise en page par rapport à la culture

3.4.1 Écriture et Lecture de droite à gauche

Par défaut votre écriture et votre lecture se font de gauche à droite. Mais dans certaines cultures l'écriture et la lecture se font de droite à gauche. Ainsi vous pouvez définir pour vos contrôles la propriété `RightToLeft` sur `Yes`, afin d'obtenir une écriture de droite à gauche pour la culture correspondante. Elle possède trois valeurs : `Yes`, `No` et `Inherit` (valeur par défaut). Cette dernière valeur détermine le sens de la lecture et d'écriture hérité du contrôle parent. Ainsi lorsque la culture est arabe ou hébraïque l'écriture et la lecture se font de droite à gauche, la barre de défilement est à gauche, ...

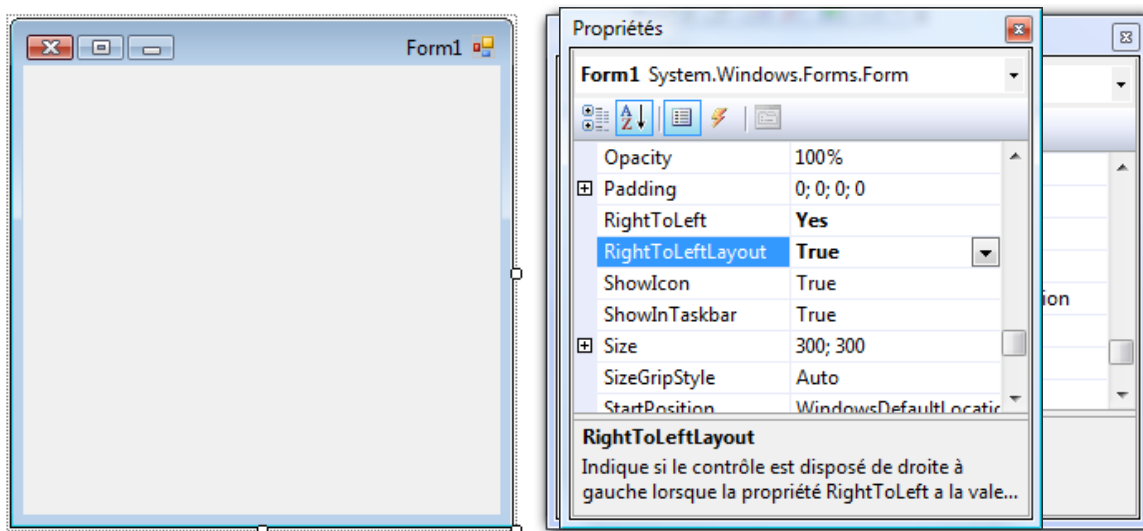


Pour les contrôles CheckBox et RadioButton, c'est la propriété CheckAlign qu'il vous faut définir.



3.4.2 L'effet miroir

Lorsque vous avez défini `RightToLeft` sur `Yes`, vous avez la possibilité d'effectuer le même processus pour l'aspect visuel de la Form ou encore des images en assignant la valeur `True` à la propriété `RightToLeftLayout`.



4 Les Formulaires MDI

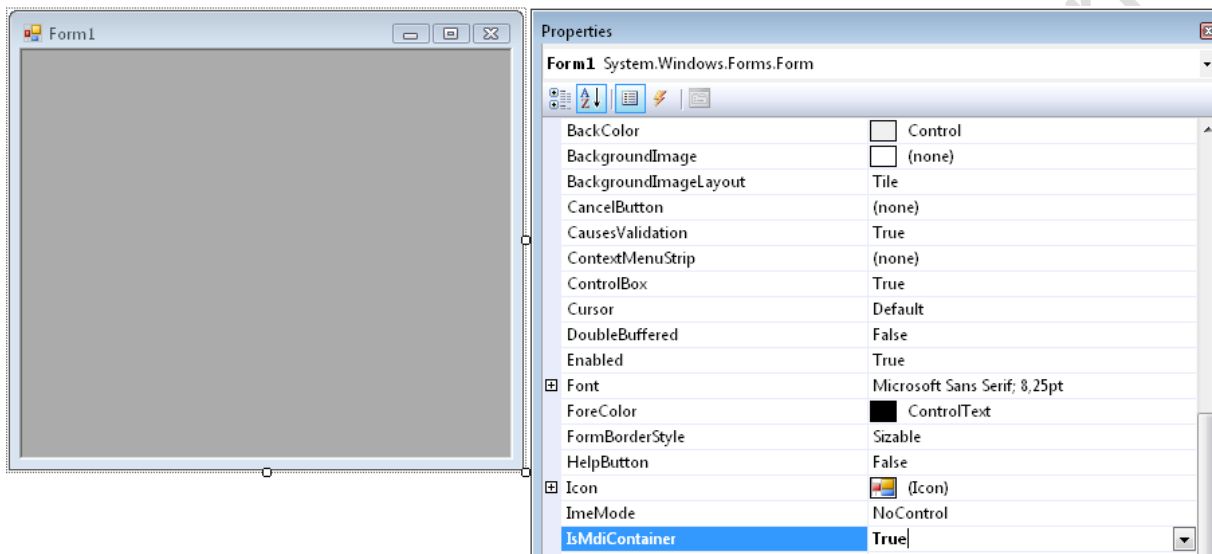
4.1 Qu'est ce qu'une MDI ?

Une application MDI est un modèle de formulaires parent/enfant. Cela permet de pouvoir travailler sur plusieurs formulaires tout en restant dans un seul, dit "parent". Par exemple, sur votre système d'exploitation vous avez plusieurs fenêtres ouvertes en parallèle et par conséquent vous pouvez travailler sur plusieurs choses à la fois. Le principe est le même, vous avez un formulaire parent (une sorte de conteneur) qui va contenir et mettre en relation tous vos formulaires enfants. De plus, une application MDI va plus loin avec des options de presse papier ou le moyen de créer une liste de tous vos formulaires enfants.

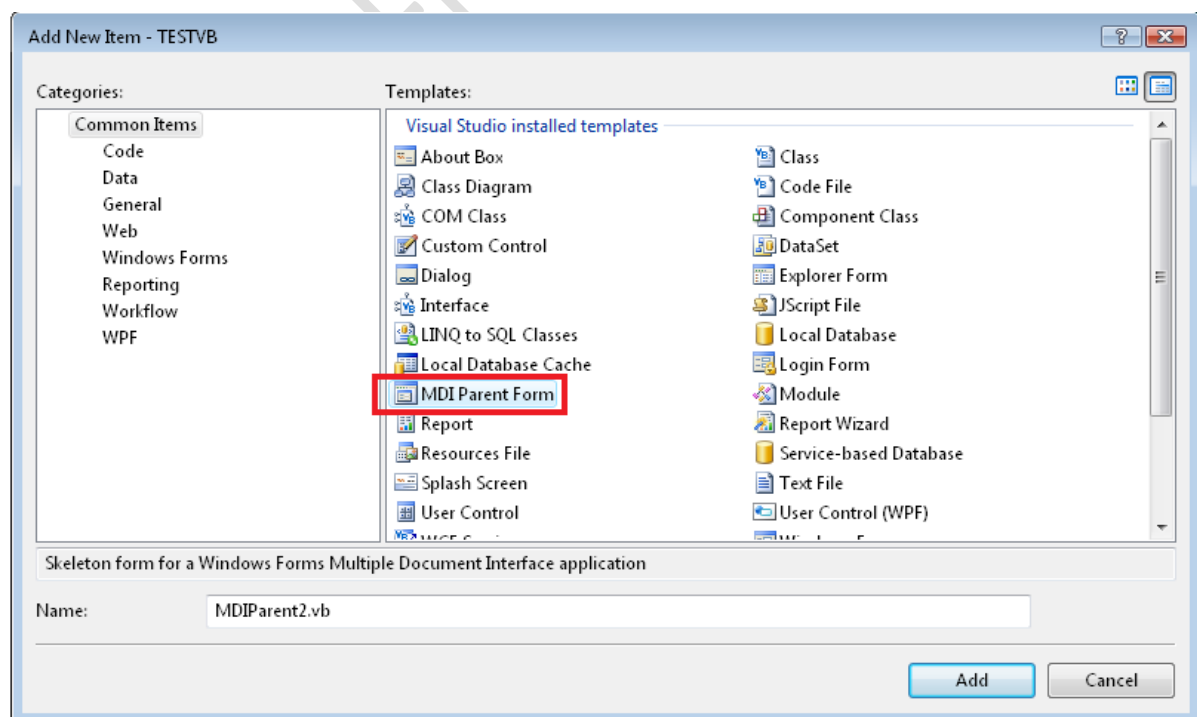
4.2 Utiliser le processus MDI

4.2.1 Créer et lier un formulaire parent et enfant

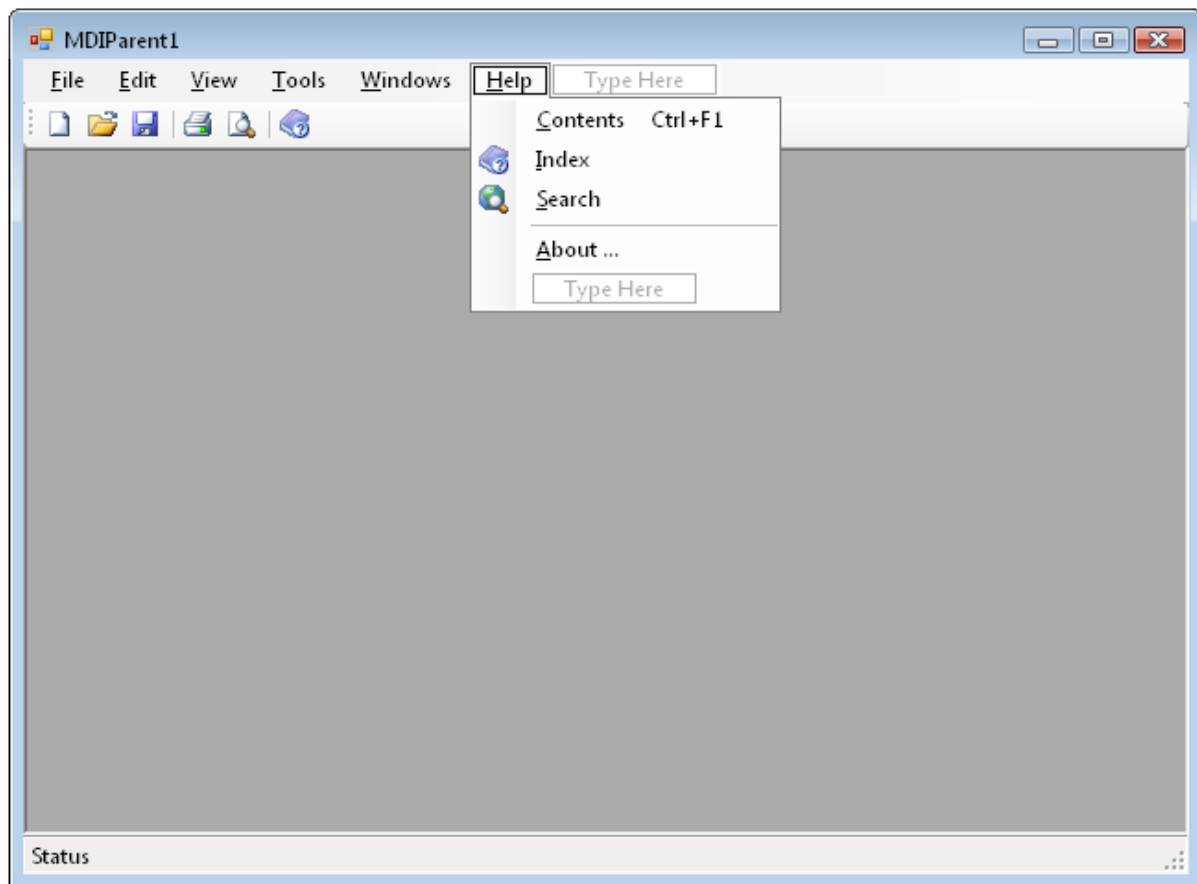
Tout d'abord, votre formulaire parent doit être le formulaire de départ de votre application, sinon votre application ne fonctionnera pas correctement voir ne fonctionnera pas tout court. Une fois votre formulaire parent désigné, allez sur la propriété *IsMdiContainer* et donnez la valeur *True*. Vous verrez que votre formulaire prendra un fond ressemblant à un conteneur :



Il existe aussi une deuxième solution afin de créer un formulaire parent, il suffit de créer un formulaire « MDI Parent Form », comme ceci :



Puis vous devriez avoir un bon départ personnalisable pour votre application MDI :

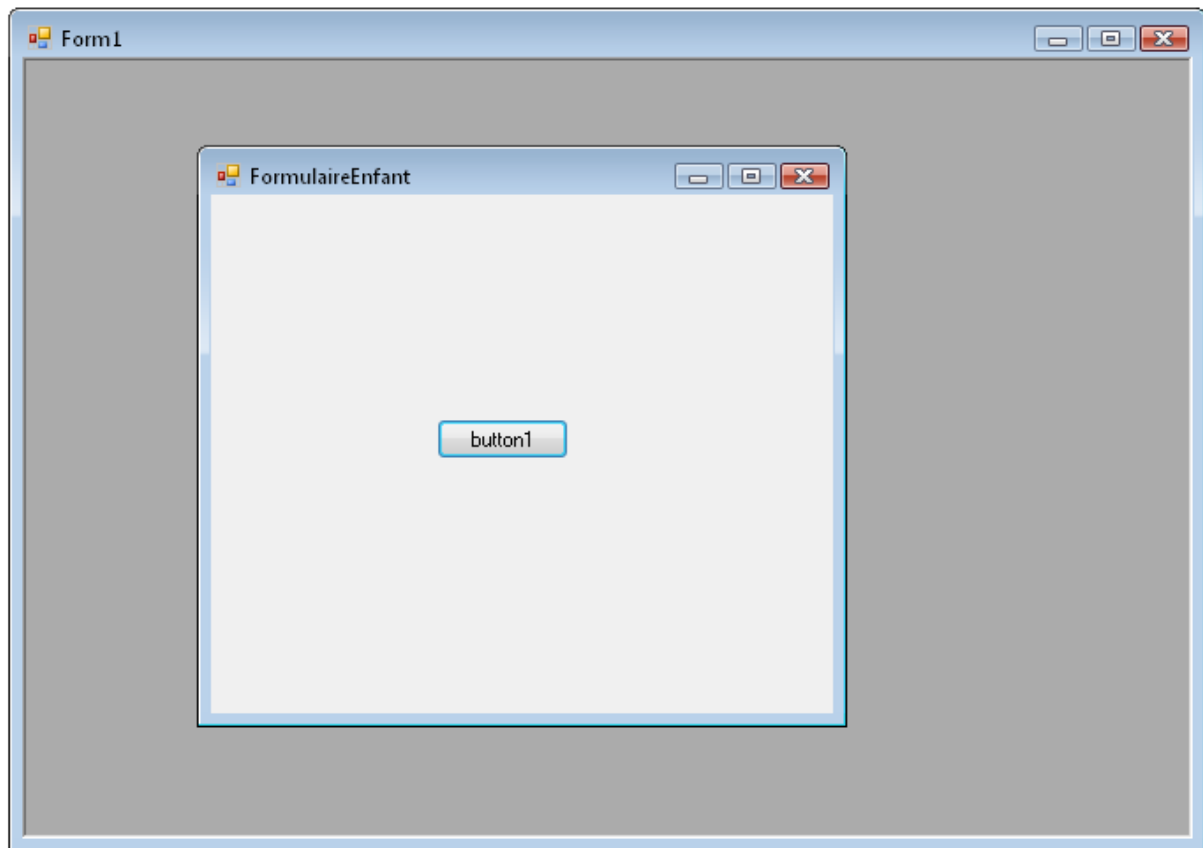


Ensuite, vous devez créer un ou plusieurs autres formulaires dit "enfants" qui seront contenue dans le parent. Dans notre cas, nous avons créé un formulaire nommé « FormulaireEnfant ». Ensuite, vous devez créer un événement *OnClick* sur votre formulaire parent afin de lui associer par la suite votre formulaire enfant. Il faut tout d'abord instancier votre classe « FormulaireEnfant » puis l'associer grâce à *MpiParent*. Voici dans notre cas :

```
'VB
Dim monFormulaireEnfant As New FormulaireEnfant
monFormulaireEnfant.MdiParent = Me
monFormulaireEnfant.Show()
```

```
//C#
FormulaireEnfant monFormulaireEnfant = new FormulaireEnfant();
monFormulaireEnfant.MdiParent = this;
monFormulaireEnfant.Show();
```

Vous aurez ce résultat (nous avons ajouté un bouton afin de montrer qu'il s'agit bien de notre formulaire enfant :



4.2.2 Gérer les formulaires enfants

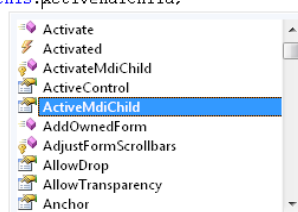
Vous pouvez avoir une multitude de formulaires enfants dans un formulaire parent. Pour connaître le formulaire qui aura le focus au démarrage de l'application, vous pouvez utiliser la propriété *ActiveMDIChild* comme ceci :

```
private void Form1_Load_1(object sender, EventArgs e)
{
    FormulaireEnfant monFormulaireEnfant = new FormulaireEnfant();
    monFormulaireEnfant.MdiParent = this;

    FormulaireEnfant2 monSecondFormulaireEnfant = new FormulaireEnfant2();
    monSecondFormulaireEnfant.MdiParent = this;

    monSecondFormulaireEnfant.Show();
    monFormulaireEnfant.Show();

    FormulaireEnfant monFormulaireDepart;
    monFormulaireDepart = (FormulaireEnfant) this.ActiveMdiChild;
}
```



Form.ActiveMdiChild
Gets the currently active multiple-document interface (MDI) child window.

De plus, lors du lancement de l'application donc du formulaire parent puis des formulaires enfants contenus dedans, il est appréciable de pouvoir classer nos formulaires enfants dedans. Ceci est possible grâce à la méthode *LayoutMdi* qui prend ses paramètres dans l'énumération *MdiLayout*.

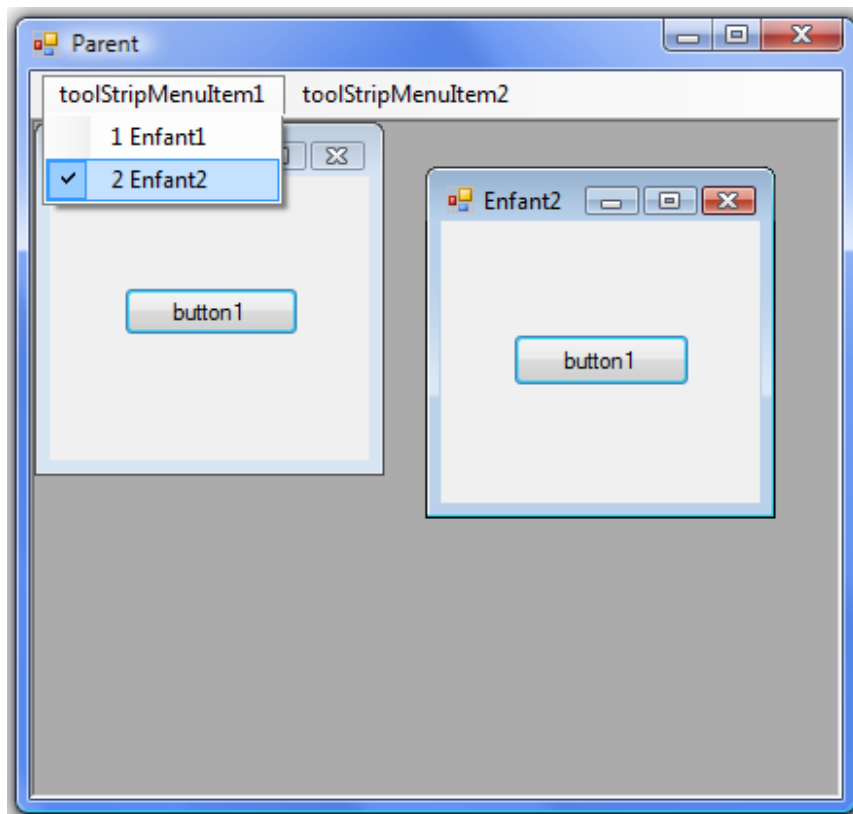
Membre	Description
ArrangeIcons	Tous les formulaires enfants seront placés dans la zone client (la zone "conteneur") du formulaire parent.
Cascade	Tous les formulaires enfants seront placés en cascade dans la zone client du formulaire parent.
TileHorizontal	Tous les formulaires enfants seront placés horizontalement en mosaïque dans la zone client du formulaire parent.
TileVertical	Tous les formulaires enfants seront placés verticalement en mosaïque dans la zone client du formulaire parent.

4.3 Utiliser le presse-papier avec une application MDI

Dès que votre application MDI est prête, vous pouvez utiliser le Presse-papier afin de coller ou récupérer des données. Pour cela vous devez appeler les méthodes *GetDataObject* (pour copier/récupérer les données) ou *SetDataObject* (pour coller/envoyer les données) de la classe *Clipboard* depuis un contrôle. Les données seront retournées dans un objet implémentant l'interface *IDataObject*. De plus si vous ne connaissez pas le format des données à récupérer vous avez la possibilité par exemple d'utiliser la méthode *GetFormats* pour obtenir une liste d'ensemble des formats de vos données. Enfin vous utilisez la méthode *GetData* afin de prendre les données et les modifier pour votre application.

4.4 Créer un menu dans une application MDI

Un menu dans une application MDI peut s'avérer très utile dans le but de gérer vos formulaires enfants. Pour cela vous devez ajouter un *MenuStrip* dans le formulaire parent, ajouter des éléments dans celui-ci puis vous définissez un élément dans la propriété *MdiWindowListItem* du *MenuStrip*. Ainsi le menu générera automatiquement des entrées pour les formulaires enfants et ils seront activés lorsqu'ils seront sélectionnés dans le menu.



5 Conclusion

En conclusion, nous avons vu dans ce chapitre différents éléments s'avérant utiles pour votre application. L'internationalisation vous permet d'avoir une application compréhensible par tous, quelque soit la culture, les applications MDI sont nécessaires si l'on veut avoir un formulaire contenant d'autres formulaires, et le Drag-and-Drop est intuitif et utile.

L'équipe WinForm