Placement des composants avec .NET 2.0

par Lainé Vincent Olivier Delmotte

Date de publication : 14/11/2005

Dernière mise à jour : 14/11/2005

Danc cet article nous allons aborder le placement des composants avec .NET 2.0. Nous verrons comment faire des interfaces graphiques qui ne craignent pas le redimesionnement de la fenêtre.

I - Introduction

Quand vous développez une application graphique vous faites en sorte que l'interface soit la plus agréable possible à regarder et à manipuler. Mais bien souvent les développeurs ne pensent pas que les utilisateurs peuvent avoir une résolution d'écran différente de la leur et dès lors, la belle interface du développeur se transforme en monstruosité pour l'utilisateur. Il existe pourtant un moyen simple et puissant de rendre une interface graphique insensible au changement de résolution : Le docking et les composants de placement.

Il va sans dire que les composants de placement ne peuvent pas être utilisés sans la notion de "docking".

Nous allons donc voir au travers d'une application de démonstration comment utiliser ces deux techniques afin de rendre votre interface insensible au changement de résolution et de taille de fenêtre.

Cet article se base sur le framework 2.0 pour les composants de placement (FlowLayoutPanel et TableLayoutPanel) mais reste valable en 1.1 pour la partie "docking"

II - Le "Docking" des composants

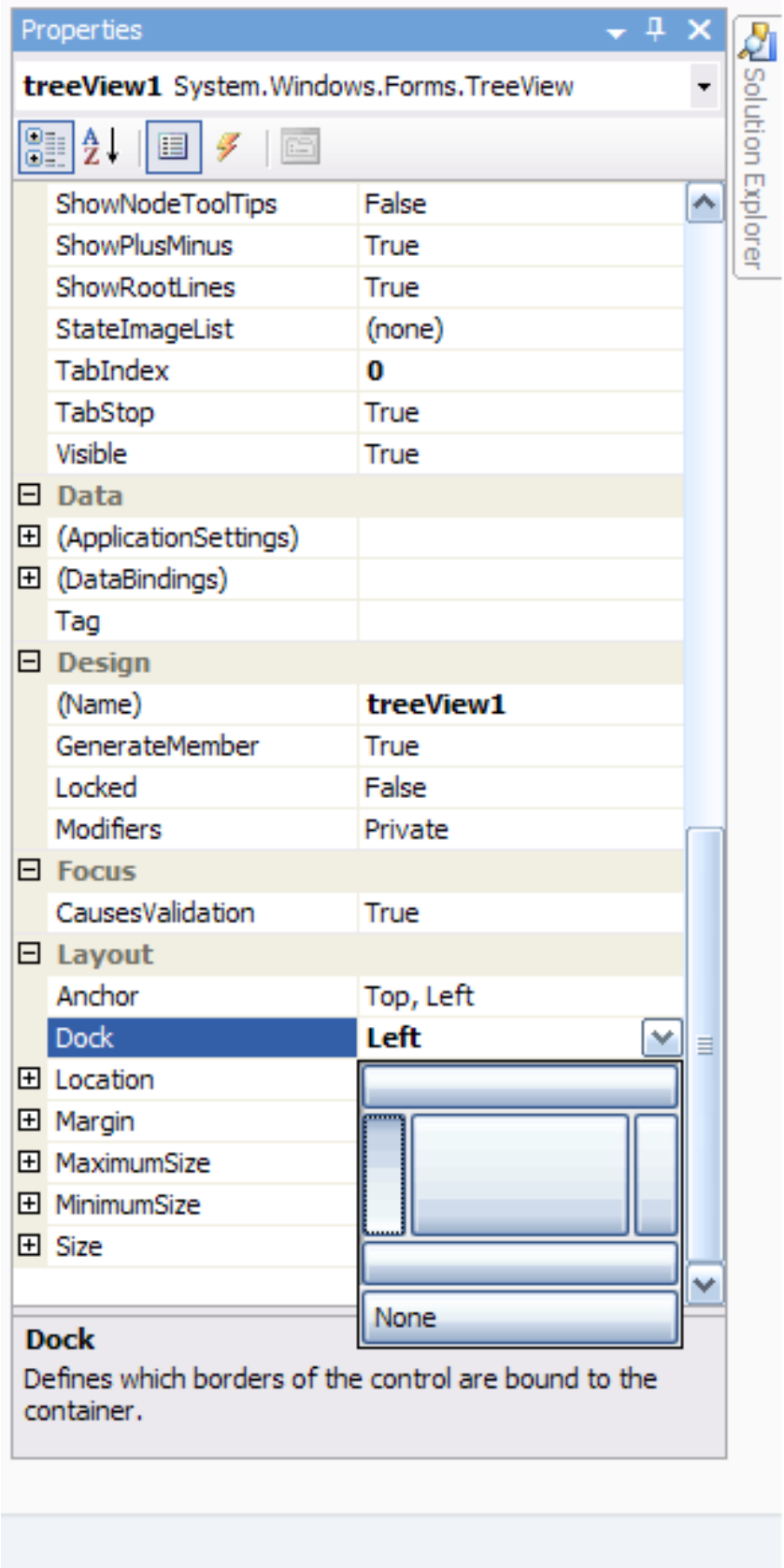
II-A - Principe de fonctionnement

Le "docking" est la technique permettant de signaler à un composant qu'il doit épouser une forme particulière de son conteneur. Si vous dockez à gauche de la fenêtre un composant vous remarquerez lors du redimensionnement de celle-ci que votre composant s'agrandit avec. Cela vous permet de définir facilement des zones possédant elles aussi leurs composants et des containeurs. Mais attention les composants dockés suivent des règles très strictes et parfois pas évidentes à comprendre. En effet vous pouvez grâce aux notions de premier plan et arrière plan définir quel composant doit être devant ou non.

II-B - Utilisation dans l'application de démonstration

Voyons maintenant comment utiliser ces propriétés de dock.

Commençons d'abord par localiser la propriété dans l'éditeur :

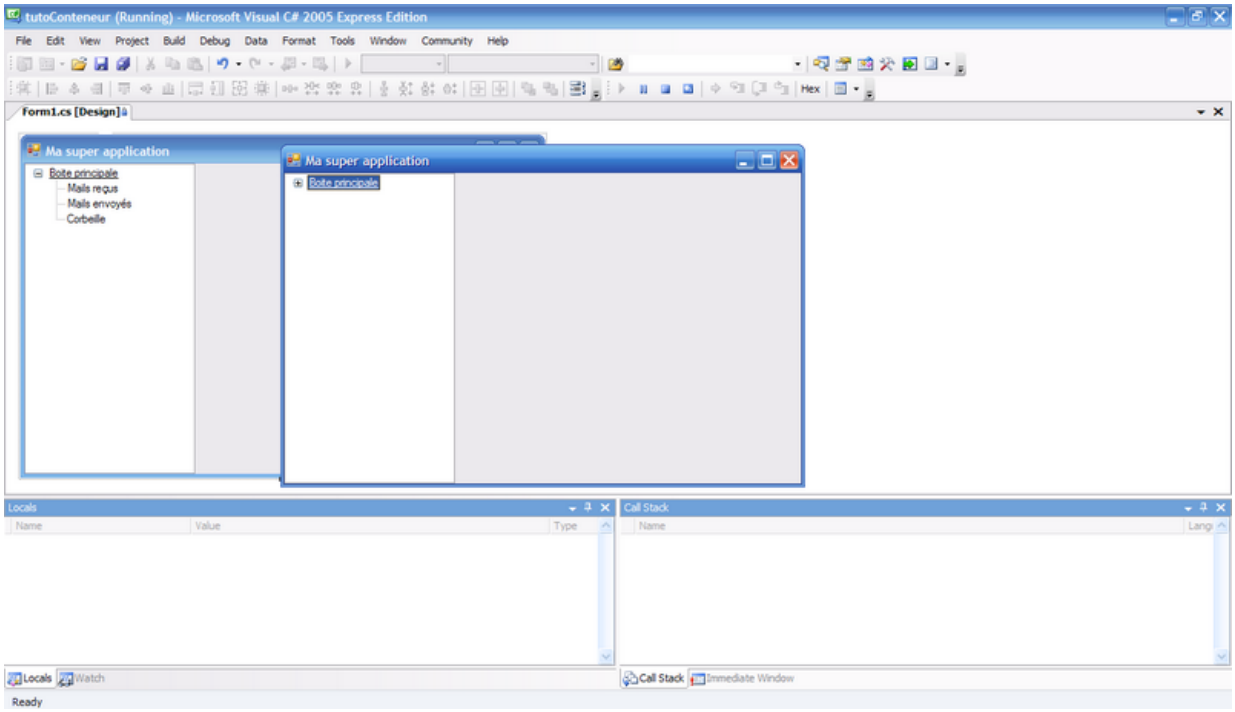


La propriété Dock dans l'explorateur de propriétés

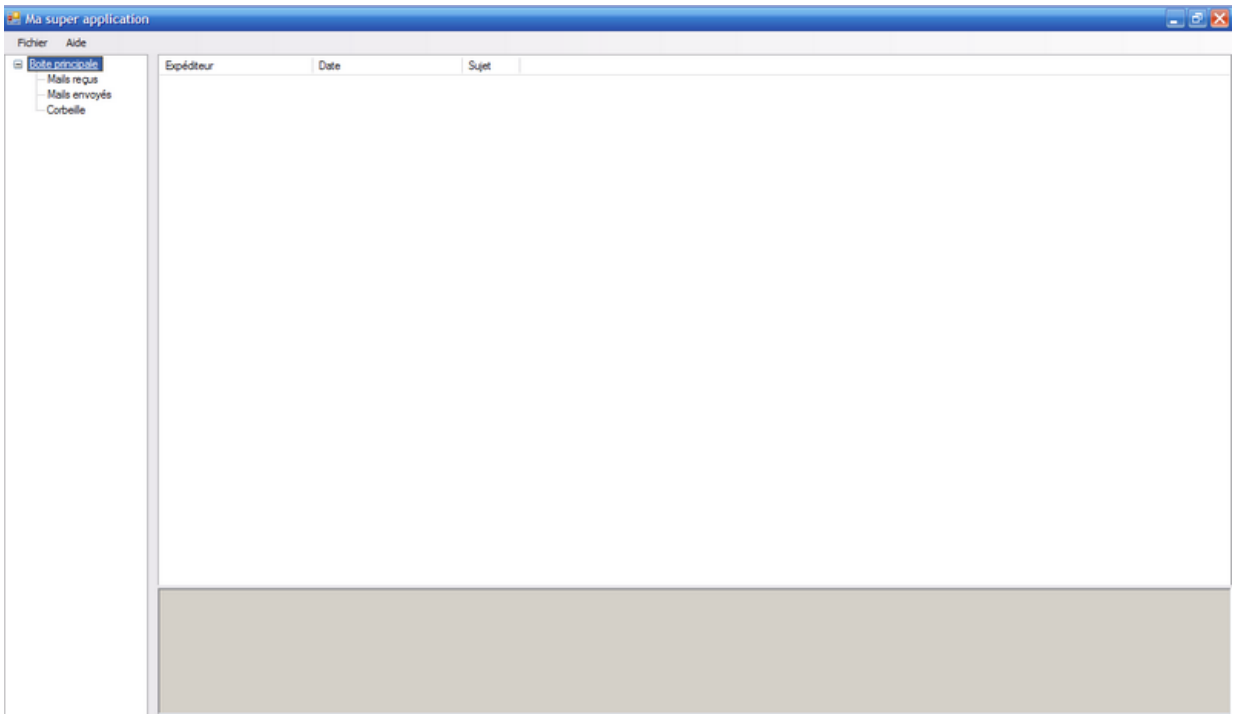
La majorité des composants possèdent cette propriété. Elle vous permet de définir où le composant doit être docké dans son containeur.

La place des boutons définit la zone de dock.

Voyons maintenant les effets du docking sur l'application une fois lancée.



L'application en taille de développement



L'application maximisée

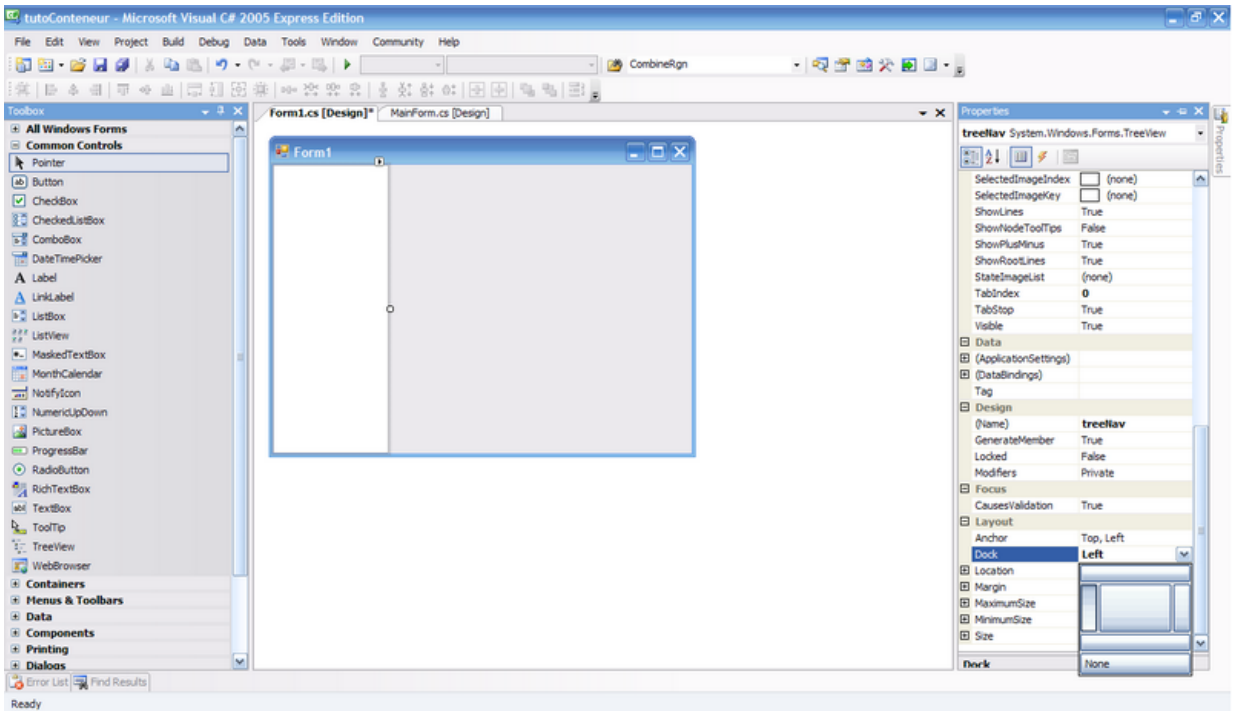
Comme vous le voyez l'application réagit très bien à la maximisation.

Vous remarquerez que le composant a été agrandit comme la fenêtre : il est "collé" au bord gauche de celle-ci. En revanche sa largeur n'a pas été modifiée car rien ne dit qu'elle doit l'être.

Nous allons maintenant enrichir notre application avec tout ce qu'il faut pour en faire un petit client mail.

Contruisons pas à pas cette première fenêtre ensemble :

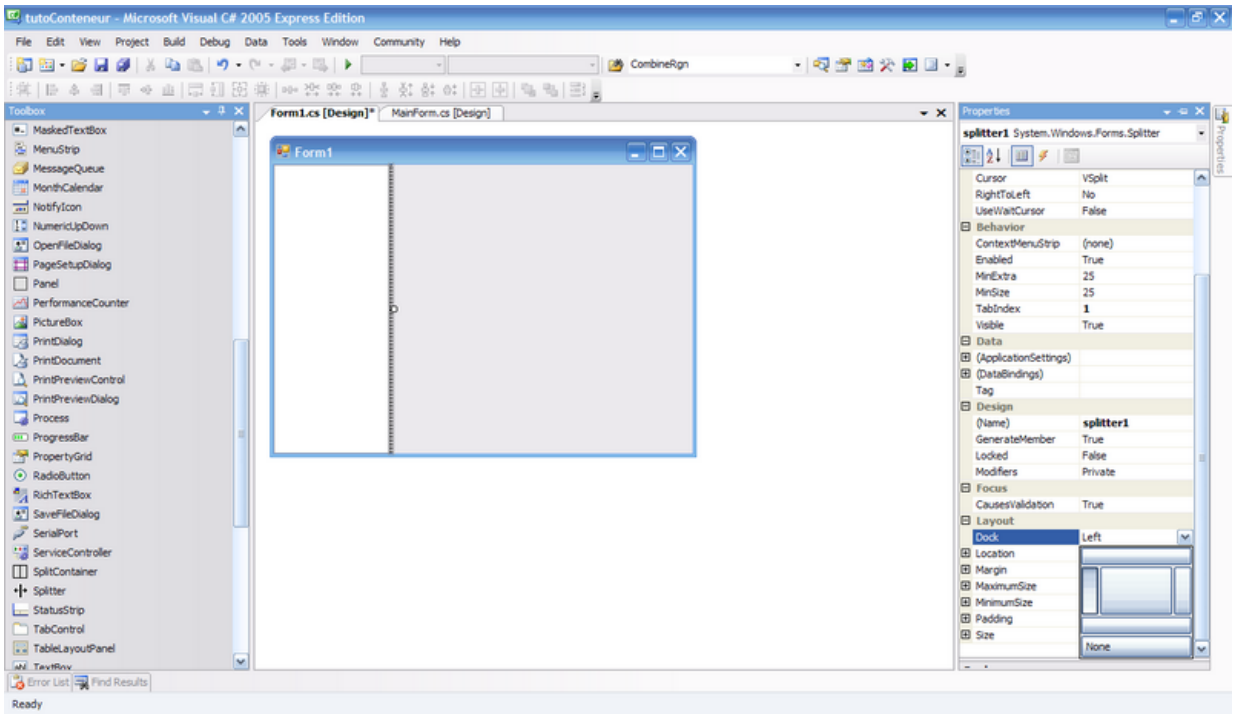
Prenons une form vide et ajoutons-y un TreeView. Ce TreeView est docké à gauche.



Placement du treeView

Je vous laisse le soin de nommer à votre guise les composants que vous ajoutez à votre formulaire. Le nommage n'est là qu'à titre d'exemple.

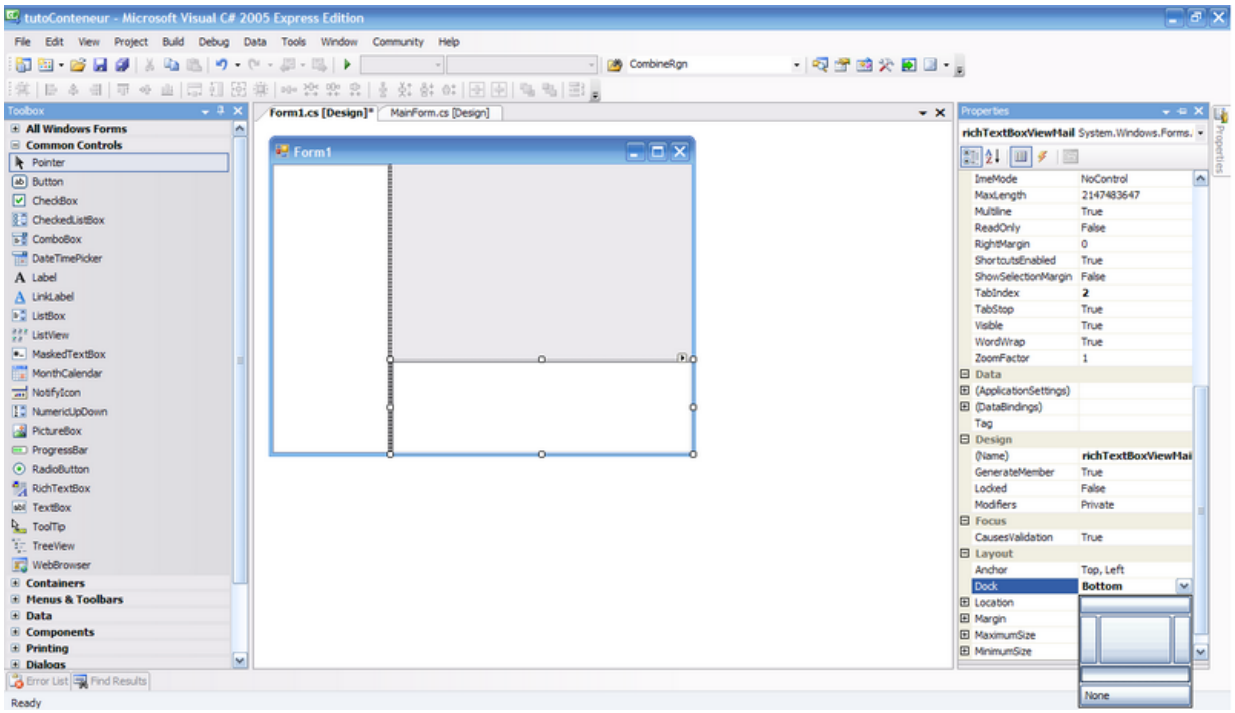
Ajoutons ensuite un Splitter vertical afin de pouvoir redimensionner nos composants facilement pendant l'éxécution du programme. Ce Splitter est docké par défaut à gauche. Nous ne modifions pas cette propriété vu qu'elle nous convient



Placement du Splitter

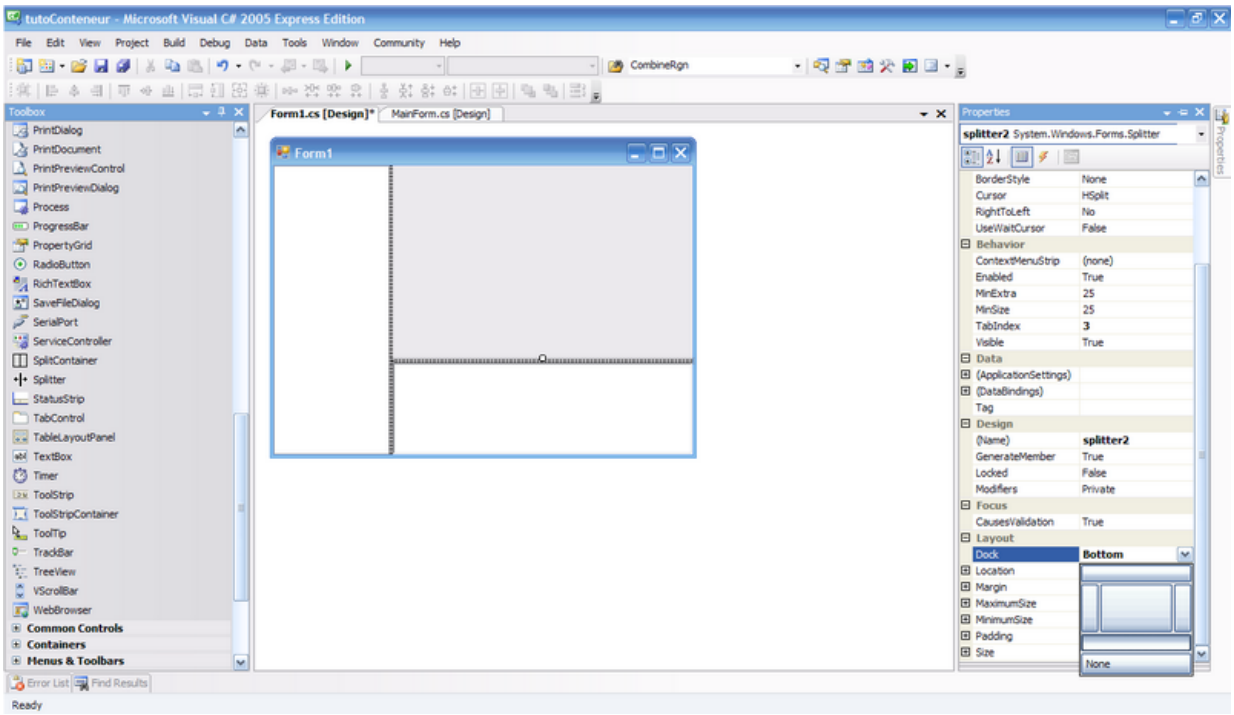
Ajoutons maintenant de quoi voir les mails de notre application.

Pour ce faire nous ajoutons un RichTextBox et nous positionnons sa propriété Dock à bottom.



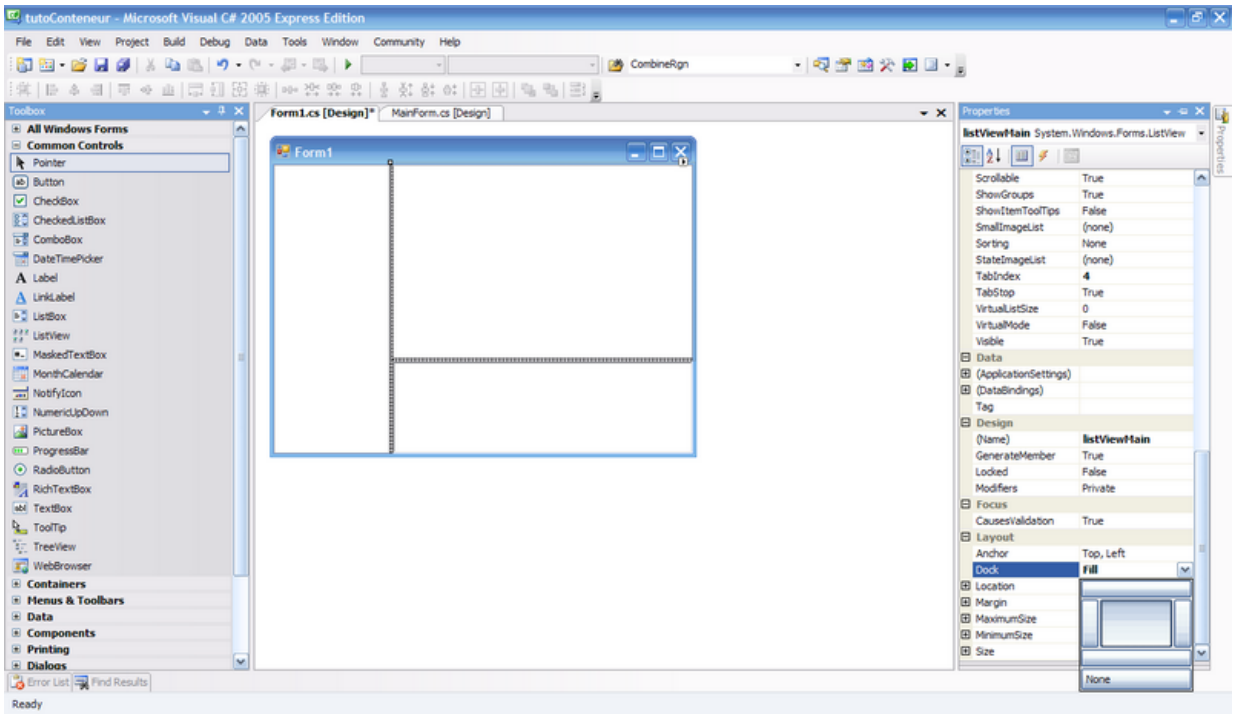
Placement du RichTextBox

Afin de pouvoir redimensionner le RichTextBox nous allons ajouter un deuxième splitter entre ce RichTextBox et le futur ListView. Nous mettrons sa propriété Dock à bottom afin qu'il agisse sur les composants situé au-dessus et en dessous de lui.



Placement du second Splitter

Et pour finir nous allons ajouter un ListView qui va prendre toute la place restante dans la fenêtre. Pour ce faire nous allons définir sa propriété Dock à Fill.



Placement du ListView

Les propriétés de docking ne sont pas bien compliquées à comprendre mais leur ordre de priorité si. Le mieux pour les comprendre reste encore de faire les tests soit-même.

Voici quelques informations qui vous aideront à mieux comprendre cela. Sur une fenêtre :

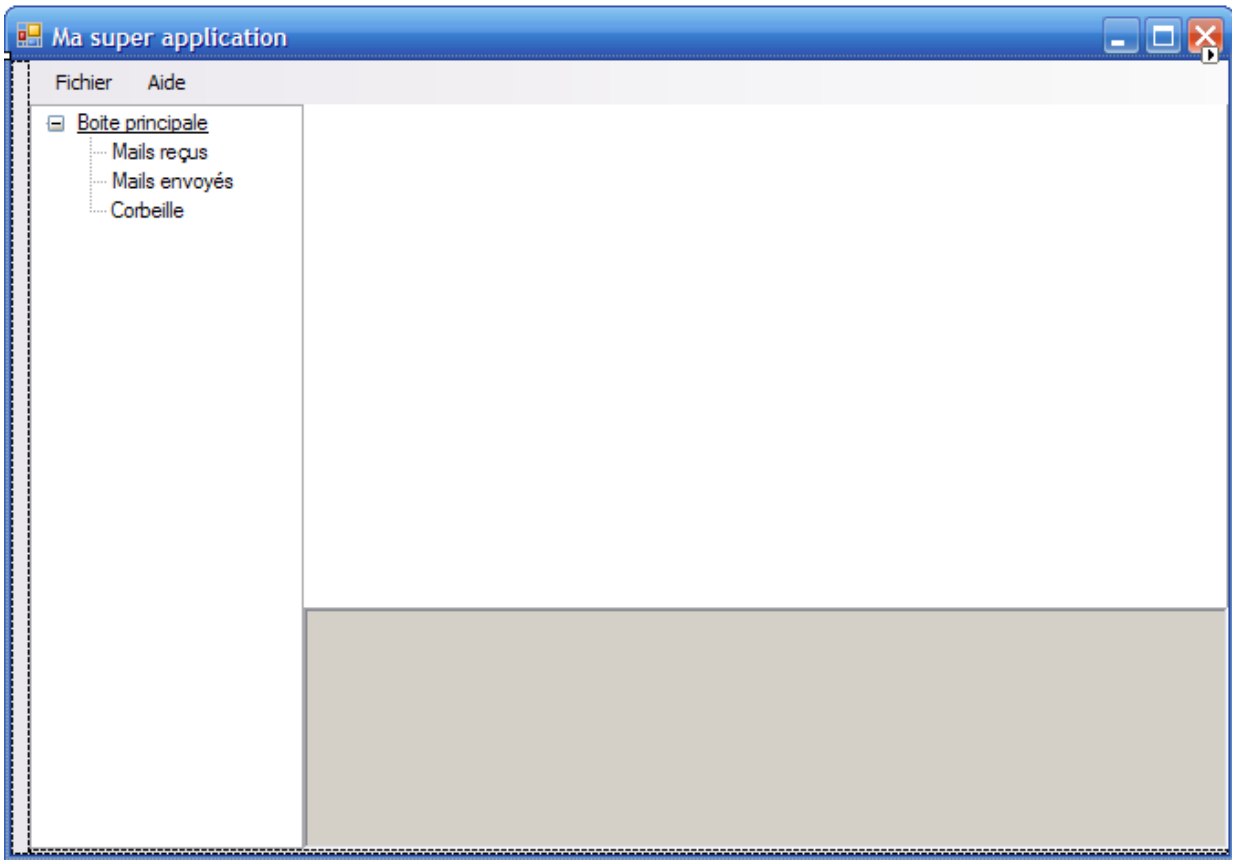
- L'arrière plan vertical est à gauche la fenêtre.

- L'arrière plan horizontal est en bas de la fenêtre.

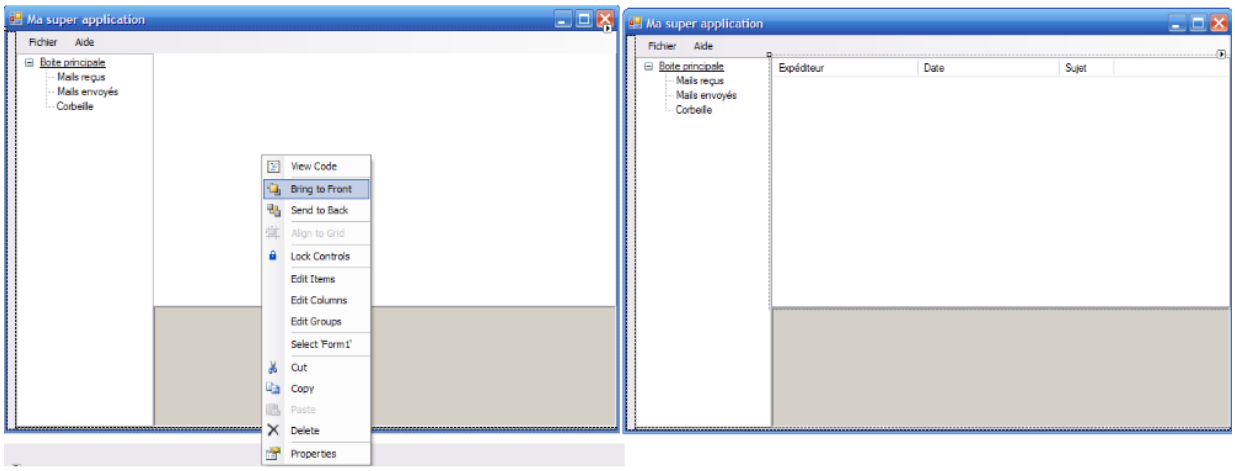
- L'ordre d'insertion des composants définit leur position initiale dans la hiérarchie de docking.

- Une mise à l'avant plan ou à l'arrière plan ne repositionne un composant que par rapport à ses deux composants voisins.

Afin d'illustrer le principe voyons cela sur des captures d'écran. Les actions effectuées sont détaillées en dessous :



Les composants de l'application sont tous mélangés



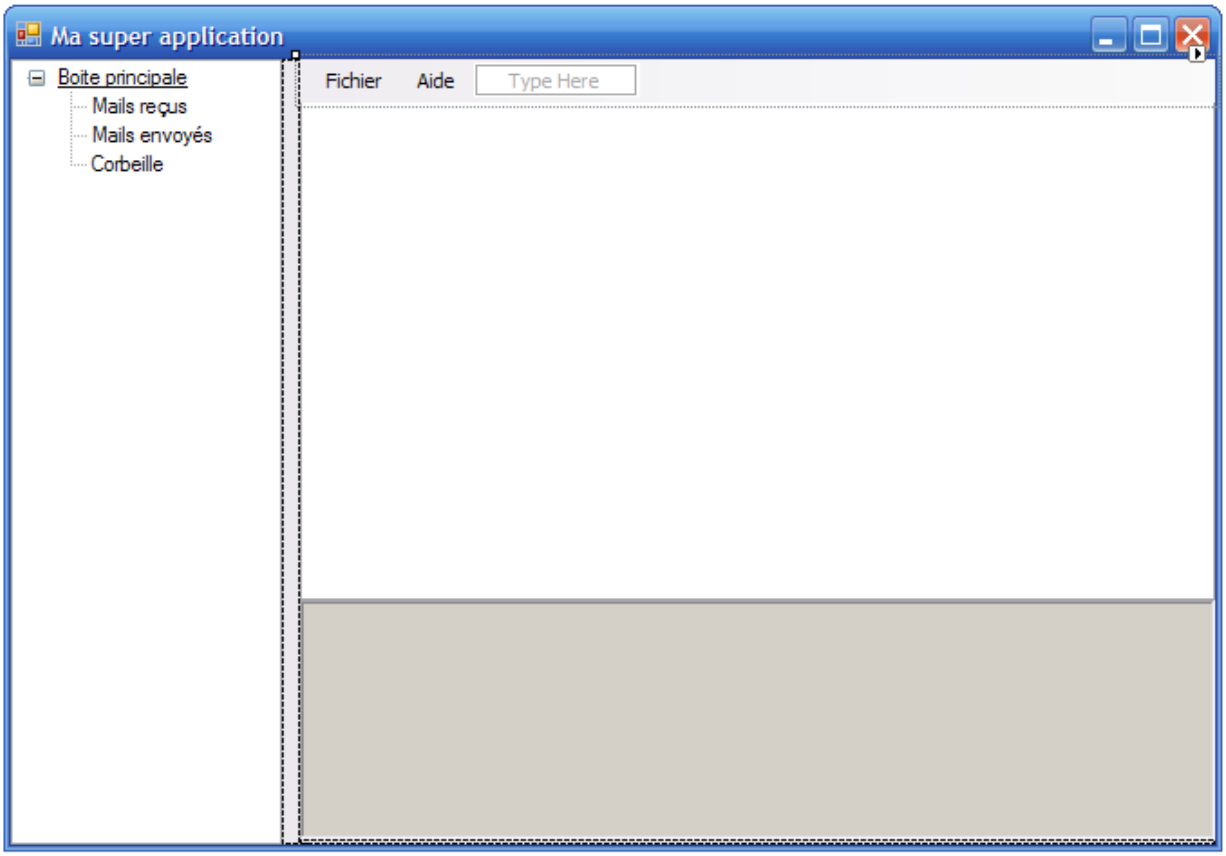
On ramène la ListView au premier plan

A cette étape nous n'avons que ramené la ListView au premier plan. En mettant la ListView au premier plan nous demandons à ce qu'elle soit affichée entièrement, or à cause des propriétés de docking, celle-ci doit être redimensionnée afin que tout soit affiché.

La ListView est donc maintenant à sa place .

Globalement il ne reste pas grand chose, l'application apparait comme il faut si vous regardez bien, vous

remarquerez que les Splitter ne sont pas bien positionnés. Nous allons remédier à cela.

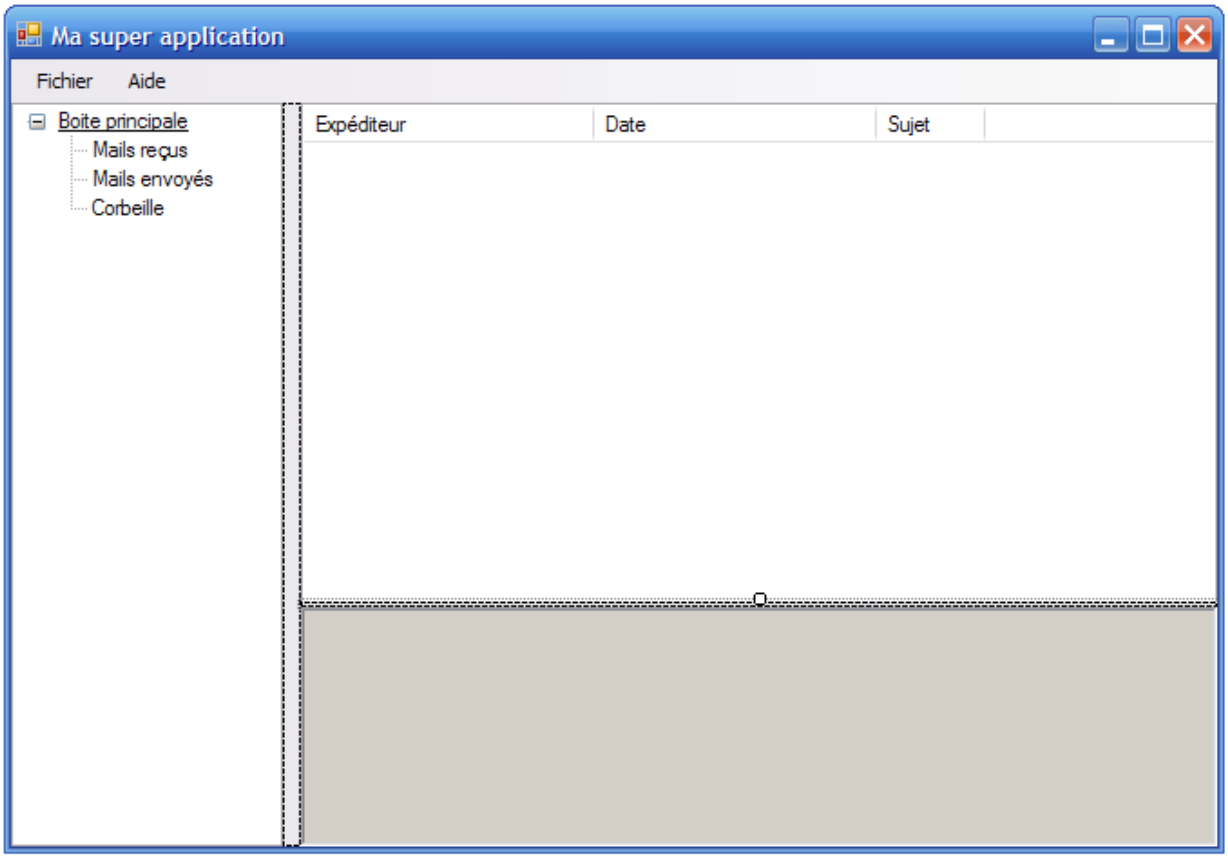


Le splitter vertical est à sa place

Nous avons maintenant remis le Splitter vertical à sa place en le mettant en arrière plan.

Mais vous remarquerez qu'un autre problème, en plus du deuxième Splitter, est apparu : Le menu n'est plus correct ! Il n'occupe que la place au dessus de la ListView.

Résolvons ces deux petits problèmes :



Tous les composants sont bien placés

Le menu a été mis en arrière plan et le splitter en avant plan.

Le placement des composants avec l'avant plan et l'arrière plan n'est pas une science exacte. Il arrive parfois des choses un peu étranges comme par exemple le fait de mettre le menu en arrière plan pour qu'il occupe tout le haut de la fenêtre.

Si vous refaites l'application de A à Z, je ne peux pas garantir que vous aurez les même résultats avec le jeu des avant/arrière plans.

Voila notre fenêtre principale est finie et supporte sans problème le redimensionnement.

Dans la suite nous allons voir comment combiner le docking aux composants de placement afin d'avoir une interface propre.

III - Placement dynamique des composants

III-A - Présentation

Le but est de réaliser un menu basé sur un fichier XML. Cette approche nous permettra de voir les différentes façons de procéder.

Le menu que nous allons créer est tout simple (dans la théorie). Nous allons ajouter un contrôle de type Button et le "coller" en haut d'un contrôle de type Panel (propriété Dock à Top). A ce bouton nous associerons un TabPage, plusieurs boutons pouvant être associés au même TabPage.

Vous verez que si l'idée est simple, la réalisation l'est moins :(

Le code de base de l'application

```
Imports System.Xml

Public Class Form1

    Private Sub ToolStripButton1_Click(ByVal sender As System.Object,
            ByVal e As System.EventArgs) Handles ToolStripButton1.Click


            Dim xmlfile As XmlDocument = New XmlDocument()
        xmlfile.Load("ListeMenu.xml")
        Dim elems As XmlNodeList = xmlfile.SelectNodes("/menu").Item(0).ChildNodes
        Dim mess As String = ""
        For i As Integer = 0 To elems.Count - 1
            AddElemMenu(elems(i).InnerText, elems(i).Attributes("tabpage").Value)
        Next

    End Sub

    Private Sub BtnMenu_click(ByVal sender As Object, ByVal e As EventArgs)
        Dim tbpname As String = CType(sender, Control).Name.Replace("btn", "")
        Me.TabControl1.SelectTab(tbpname)
    End Sub

    Private Sub AddElemMenu(ByVal itemText As String, ByVal associatedTP As String)

            ' C'est ici que nous nous occuperons d'ajouter les différents éléments du menu.

    End Sub

End Class
```

III-B - Sans la gestion de placement

En utilisant AddElemMenu dans la version suivante, vous allez avoir des surprises.

Ajout des éléments au menu sans réfléchir

```
Private Sub AddElemMenu(ByVal itemText As String, ByVal associatedTP As String)
    ' Création du bouton pour le menu
    Dim b As Button = New Button()
    b.Size = New Size(100, 40)
    b.Location = New Point(10, 10)
    b.Dock = DockStyle.Top
    b.Text = itemText
    b.FlatStyle = FlatStyle.Popup
    b.Name = "btn" & associatedTP
```

Ajout des éléments au menu sans réfléchir

```
        AddHandler b.Click, AddressOf BtnMenu_click
        ' Si le tabpage associé n'éxiste pas, on le crée
        If Me.TabControl1.Controls(associatedTP) Is Nothing Then
            Dim tbp As TabPage = New TabPage
            tbp.Name = associatedTP
            tbp.Text = associatedTP
            Me.TabControl1.TabPages.Add(tbp)
        End If
        ' On ajoute le bouton au menu
        Me.Panel1.Controls.Add(b)
    End Sub
```

Tous les éléments du menu apparaîssent inversés. Pourquoi ? Jettons un oeil au code généré par Visual Studio pour l'interface de base de ce programme, principalement la partie concernant Form1.

Code généré par le concepteur Windows Forms

```
    '
    'Form1
    '
    Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
    Me.ClientSize = New System.Drawing.Size(703, 462)
    Me.Controls.Add(Me.TabControl1)
    Me.Controls.Add(Me.Panel1)
    Me.Controls.Add(Me.ToolStrip1)
    Me.Name = "Form1"
    Me.Text = "Exemple de placement des composants"
    Me.ToolStrip1.ResumeLayout(False)
    Me.ToolStrip1.PerformLayout()
    Me.ResumeLayout(False)
    Me.PerformLayout()
```

Pour réaliser cette interface, constituée d'une ToolStrip, d'un Panel et d'un TabControl, j'ai commencé par poser le ToolStrip, dont la propriété Dock est par défaut Top, ensuite le panel, que j'ai Docké à Left et pour finir le TabControl, Docké en Fill.

Vous constatez que le dernier contrôle que j'ai ajouté est placé en premier dans la collection de contrôles de ma Form et que le premier se situe en dernier. En partant de ce constat, on comprend que le dernier élément ajouté est "prioritaire" au niveau de la propriété Dock, et que c'est donc lui qui sera traité en premier lors de la gestion du placement.

Comment remédier à ce "problème" ?

III-C - Première méthode - En trichant sur l'ordre de chargement des éléments du menu

Pour palier à ce problème, la première solution est la ruse, mais elle n'est pas forcément toujours applicable malheureusement.

Comment faire ? Tout simplement en traitant le contenu de notre fichier XML à l'envers, en insérant le dernier élément en premier et ainsi de suite.

Modification du sens de parcours de la boucle

```
    Private Sub ToolStripButton1_Click(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles ToolStripButton1.Click

        ' ...
```

Modification du sens de parcours de la boucle

```
        For i As Integer = elems.Count - 1 To 0 Step -1
            AddElemMenu(elems(i).InnerText, elems(i).Attributes("tabpage").Value)
        Next

        ' ...

    End Sub
```

Toujours pas convainquant et surtout pas applicable dans tous les cas. Passons à la deuxième méthode.

III-D - Deuxième méthode - En réordonnant nous même les contrôles

Réagencement 'manuel' des contrôles

```
    Imports System.Xml

    Public Class Form1

    ' ...

    Private Sub AddElemMenu(ByVal itemText As String, ByVal associatedTP As String)

        ' ...

      ' On ajoute le bouton au menu avec la gestion du placement
      If Panel1.Controls.Count = 0 Then
        Me.Panel1.Controls.Add(b)
      Else
        Dim controls() As Control = New Control(Panel1.Controls.Count) {}
        Me.Panel1.Controls.CopyTo(controls, 0)
        Array.Copy(controls, 0, controls, 1, controls.Length - 1)
        controls(0) = b
        Panel1.Controls.AddRange(controls)
      End If
    End Sub

    ' ...

    End Class
```

Comme ControlCollection ne possède pas de méthode InsertAt malheureusement, nous allons la réécrire nous même. On copie son contenu dans un tableau, on décale tous ses éléments d'un cran et on insère le nôtre en position 0.

Si par exemple, vous aviez besoin de rajouter un élément en 2ème position du menu, il faudrait jouer sur les paramètres sourceIndex, destinationIndex et length de la méthode Copy de la classe Array.

Pour rajouter un élément du menu en deuxième position

```
    Array.Copy(controls, 1, controls, 2, controls.Length - 2)
    controls(1) = b
```

Voila, j'espère que je vous aurais aidé un peu à comprendre le placement des contrôles à l'exécution. Amusez-vous bien ;)

III-E - Sources

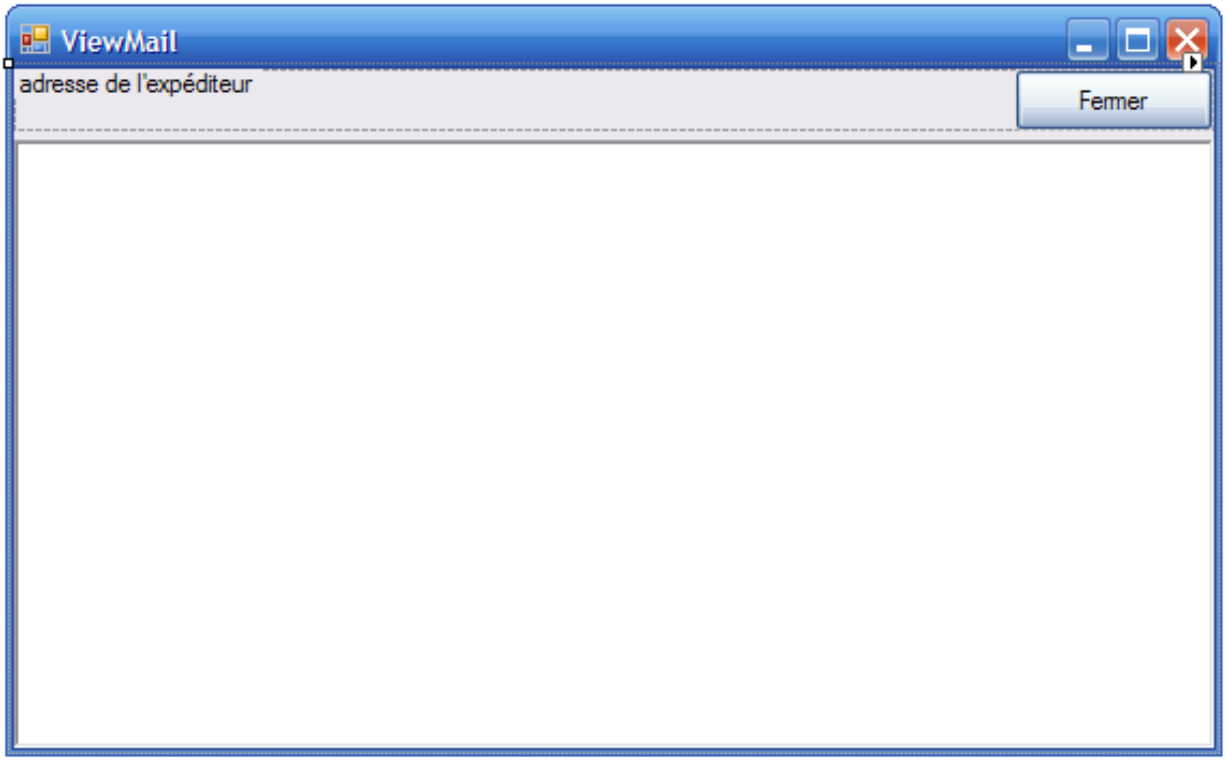
Vous pouvez télécharger les sources de ces exemples.

Le zip contient la version du projet pour la deuxième méthode.

IV - Le composant SplitContainer

Le composant SplitContainer vous permet de créer deux Panels séparés par un Splitter. La nouveauté étant que c'est le SplitContainer lui même qui gère directement les deux Panels.

Cela vous permet de définir facilement deux zones dans lesquels vous pourrez ajouter des composants. Vous pourrez également facilement définir l'orientation du Splitter (horizontal ou vertical).



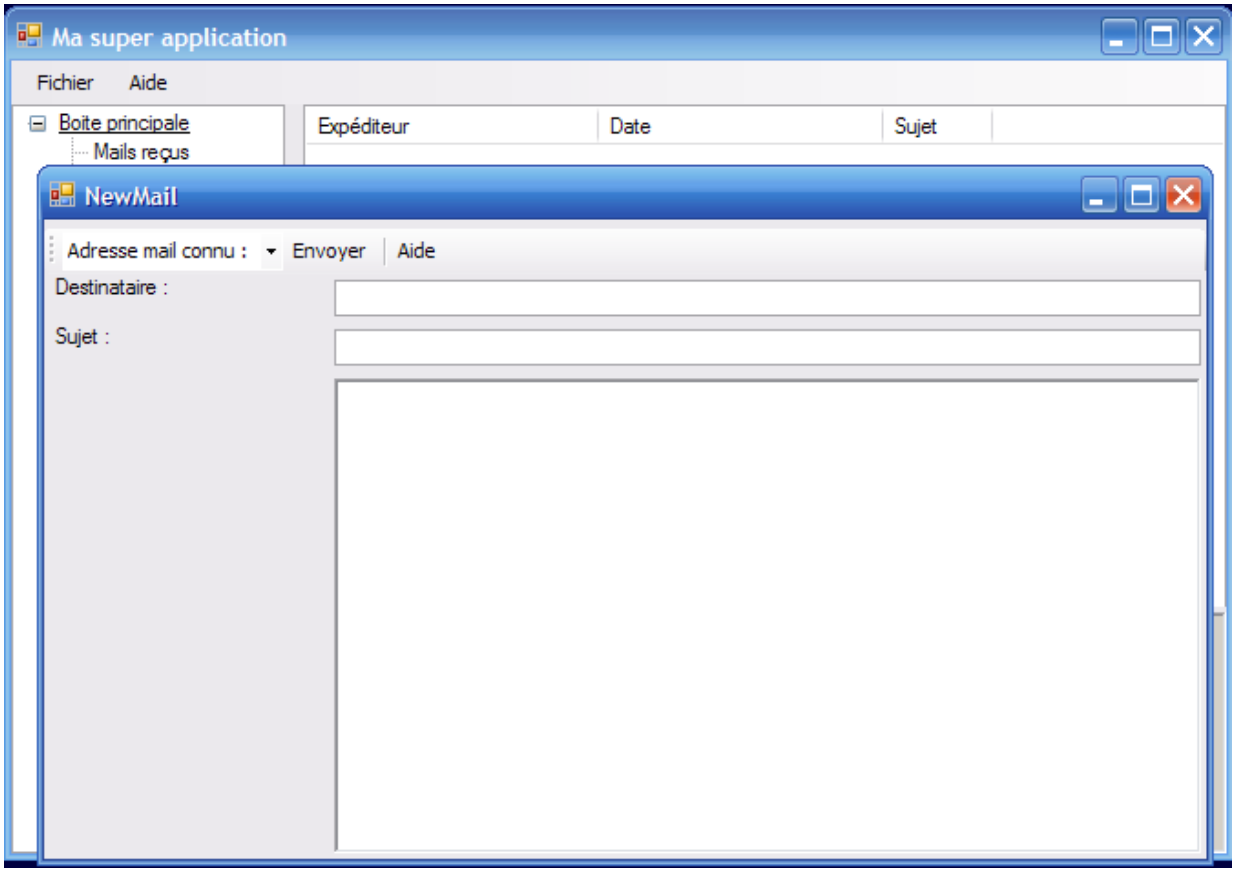
Les deux parties sont séparées par un splitter. Les Panels contiennent les composants.

V - Le composant TableLayoutPanel

Le composant TableLayoutPanel est un composant permettant de placer des composants à la manière d'un tableau. Cela se révèle pratique dans le cas où vous souhaitez aligner des composants qui n'ont pas nécessairement la même taille.

Les principales propriétés de ce composant sont évidemment le nombre de ligne et le nombre de colonne. (Rows et Columns).

Grâce à ce composant nous allons aligner les TextBox du destinataire et du sujet. De plus le RichTextBox sera également aligné sur les deux TextBox donnant un aspect plus fini à l'application.



LesTextBox sont alignées ainsi que le RichTextBox

Ce composant est très simple d'emploi. Il suffit de le créer et d'y mettre le nombre de lignes et de colonnes que l'on souhaite. Ensuite il suffit d'y faire glisser les composants et ceux-ci adoptent la bonne place.

Le gros avantage est donc d'avoir un bon rendu mais malheureusement ce composant ne supporte pas très bien le redimensionnement.

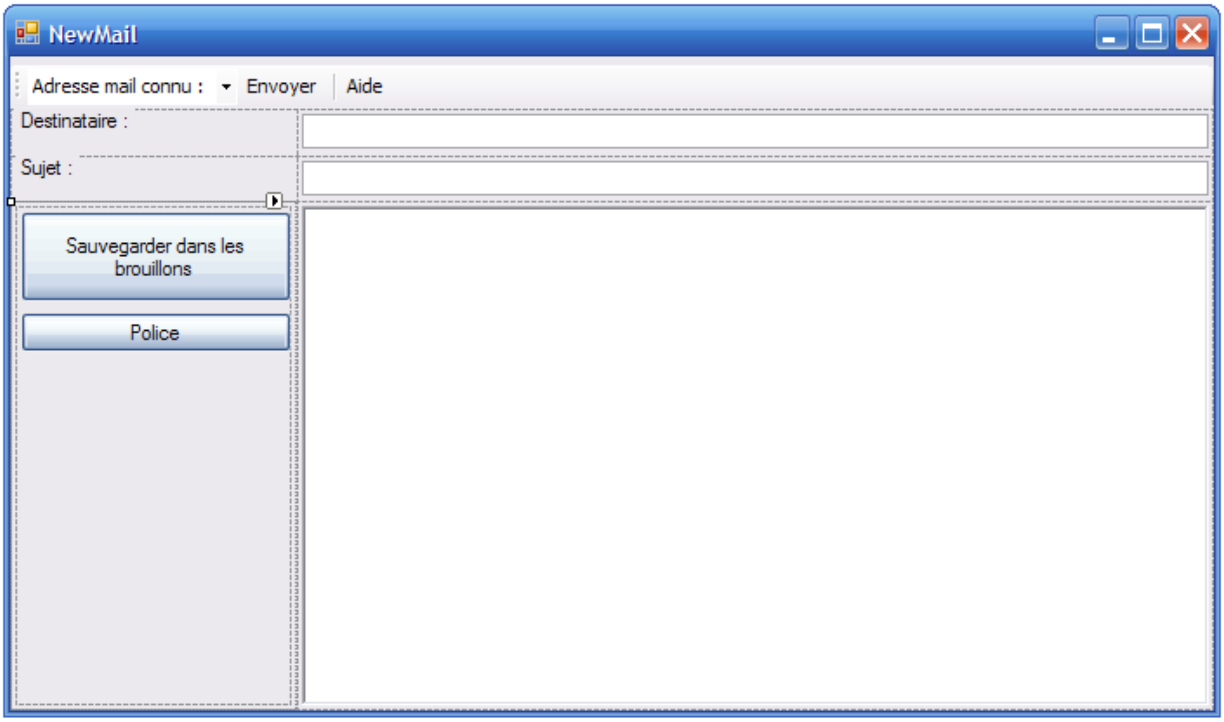
VI - Le composant FlowLayoutPanel

Le composant FlowLayoutPanel vous permet de définir l'ordre de placement des composants à l'intérieur de celui-ci.

En effet si vous ajoutez des composants vous remarquerez qu'il vont tous se coller en haut à gauche du conteneur.

Ce contrôle vous permet donc de faire par, exemple, une barre d'outils très facilement et ainsi d'arranger les divers composants afin de garder une uniformité dans l'interface graphique.

De plus vous pouvez par le code modifier facilement l'organisation du contenu en modifiant la propriété *FlowDirection*. Cette méthode vous permet de maitriser à tout moment la disposition de vos composants.



Les boutons à gauche sont contenus dans un FlowLayoutPanel avec la propriété FlowDirection à TopDown

VII - Remerciements

Je tiens à remercier titi pour la partie sur le placement dynamique des composants.

Je remercie également Khany pour la correction orthographique.

VIII - Téléchargements

Le projet Visual C# 2005

L'article au format pdf