Developpez.com
Club des développeurs

Le système de layout WPF

par Matthieu Dordolo (Accueil)

Date de publication : 20 October 2008

Dernière mise à jour :

Cette article parlera du système de layout WPF et des changements et des nouveautés qu'il nous sont proposées.

I - Généralités....3
II - Les différents panels....3
II-A - Le StackPanel....3
II-B - Le WrapPanel....5
II-C - Le DockPanel....8
II-D - Grid....9
II-E - GridSplitter....13
II-F - Canvas....14
III - Conclusion....16
IV - Remerciements....17

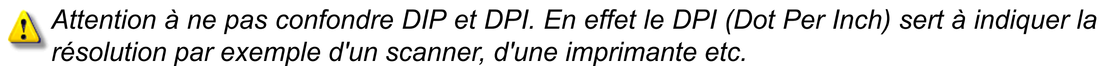
Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

I - Généralités

L'arrivée du Framework .NET 3.0 fin 2006 a apporté son lot de nouveautés comme WF, WCS, WCF, ou encore WPF qui sera le sujet que nous aborderons tout au long de cet article.

WPF est la couche graphique du Framework .NET, elle offre beaucoup plus de possibilités que les WinForms et est entièrement vectoriel ce qui permet d'adapter une fenêtre en fonction des résolutions et des redimensionnements sans pixellisation.

Cela est possible grâce au nouveau système de coordonnées : le DIP (Device Independent Pixel).



Attention à ne pas confondre DIP et DPI. En effet le DPI (Dot Per Inch) sert à indiquer la résolution par exemple d'un scanner, d'une imprimante etc.

Le DIP est en fait un pixel logique, c'est-à-dire qu'il est indépendant de la résolution de l'utilisateur.

1" = 96 Dip soit 2,54cm

On peut faire une fonction de conversion toute simple de cette façon :

```
public static double cmTodip(double cmNumber)
{
 return (cmNumber * 96.0 / 2.54);
}
public static double dipTocm(double dipNumber)
{
 return (dipNumber * 2.54 / 96.0);
}
```

Mais pourquoi un système de layout ?

Un système de layout c'est un système dit de disposition des contrôles dans notre fenêtre. C'est une partie non négligeable du développement de votre application client dans le sens ou votre IHM doit être agréable à regarder, mais surtout à manipuler. On pourra ici aborder cette idée de layout en découvrant les différents composants de placement qui nous sont proposés par WPF.

II - Les différents panels

II-A - Le StackPanel

Le panel StackPanel permet de disposer des éléments horizontalement ou verticalement (paramètre par défaut), c'est-à-dire sur une ligne ou sur une colonne. L'orientation des éléments du StackPanel est gérée par la propriété *Orientation*. Ici on ne modifiera pas cette propriété et sera donc par défaut définit comme *Vertical*.

Pour donner un exemple concret de l'utilisation d'un StackPanel on peut par exemple décider de faire un menu.

Pour une version de ce code en XAML ce sera très facile. En effet il nous suffira de déclarer les éléments dans l'ordre d'affichage dans le StackPanel défini par la balise <StackPanel>.

StackPanel XAML

```
<Window x:Class="WpfApplication.Window1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 Title="Use a StackPanel" Height="290" Width="140">
```


Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

StackPanel XAML

```
  <StackPanel Margin="10">
   <Label>UserName :</Label>
   <TextBox></TextBox>
   <Button>Sign In</Button>

   <Label>Menu n°1</Label>
   <Label>Menu n°2</Label>
   <Label>Menu n°3</Label>
  </StackPanel>
</Window>
```

Comme vous pouvez le voir la déclaration et l'utilisation d'un StackPanel en XAML est plus que facile. En revanche, une des principales utilités d'utiliser un panel comme celui-ci est de pouvoir générer des contrôles dynamiquement sans avoir à se préoccuper du placement dans notre application.

C'est pourquoi nous allons voir comment déclarer le même StackPanel via un code C#.

StackPanel C#

```
using System;
using System.Windows;
using System.Windows.Controls;

namespace WpfApplication
{
 public partial class Window1 : Window
 {
  public Window1()
  {
   Title = "Use a StackPanel";
   Height = 290;
   Width = 140;

   Label UserNameLabel = new Label();
   UserNameLabel.Content = "UserName :";

   TextBox UserNameTxtBox = new TextBox();

   Button SignInButton = new Button();
   SignInButton.Content = "Sign In";

   Label Menu1 = new Label();
   Menu1.Content = "Menu n°1";

   Label Menu2 = new Label();
   Menu2.Content = "Menu n°2";

   Label Menu3 = new Label();
   Menu3.Content = "Menu n°3";

   StackPanel MyStackPanel = new StackPanel();
   MyStackPanel.Margin = new Thickness(10);
   MyStackPanel.Children.Add(UserNameLabel);
   MyStackPanel.Children.Add(UserNameTxtBox);
   MyStackPanel.Children.Add(SignInButton);
   MyStackPanel.Children.Add(Menu1);
   MyStackPanel.Children.Add(Menu2);
   MyStackPanel.Children.Add(Menu3);

   Content = MyStackPanel;
  }
 }
}
```

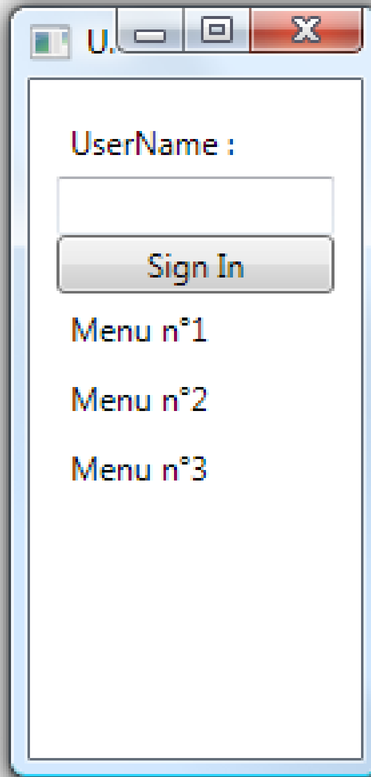
Comme vous pouvez l'observer ici vous déclarez vos contrôles de façon normal rien ne change, vous définissez vos propriétés puis tout se passe après la déclaration de votre StackPanel.

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

En effet c'est la collection Children qui va vous permettre d'ajouter vos contrôles au StackPanel.

Ensuite vous n'avez plus qu'à ajouter votre Panel à la fenêtre courante.

Que vous ayez utilisé XAML ou C# pour la mise en place de votre StackPanel le rendu restera le même :



Un gros avantage que nous n'avions pas avec les WinForms c'est l'utilisation de la collection Children qui nous permet de par la méthode Insert d'insérer entre deux contrôles un autre contrôle sans nous soucier encore une fois de sa position. En effet toutes les positions des autres contrôles vont être recalculées automatiquement.

II-B - Le WrapPanel

Le panel WrapPanel est très proche du StackPanel. Il propose en revanche la possibilité de changer automatiquement de ligne ou de colonne en fonction de l'orientation du panel si celle-ci est pleine.

Cette option peut en effet être très pratique pour une liste d'éléments comme pour lister les fichiers d'un dossier ou lister des produits ? Lors du redimensionnement futur de la fenêtre ces derniers changeront de ligne/colonne sans aucun souci.

Pour illustrer cet exemple nous allons composer le code suivant :

WrapPanel C#

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.IO;
```


Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

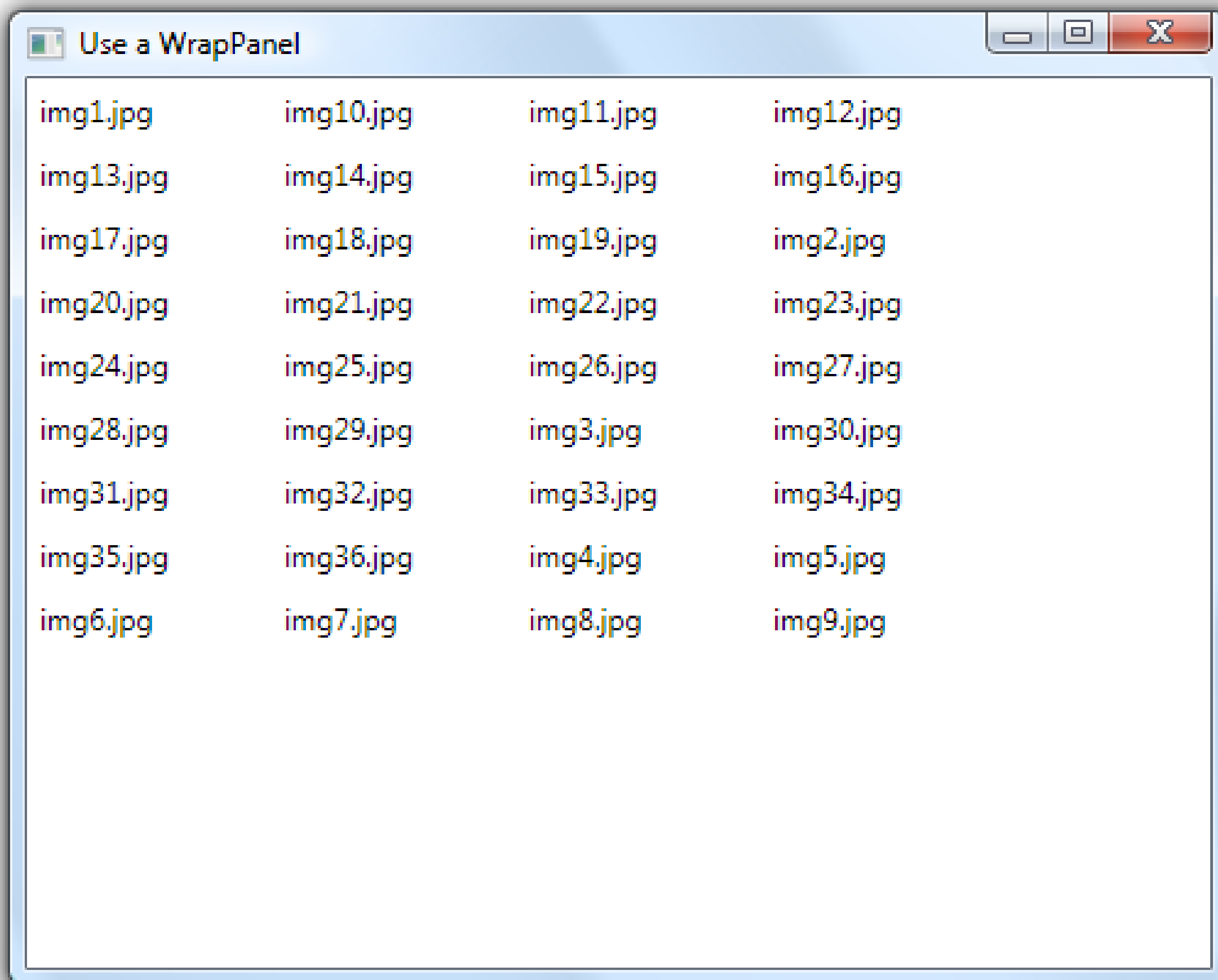
WrapPanel C#

```
namespace WpfApplication
{
 public partial class Window1 : Window
 {
  public Window1()
  {
   Title = "Use a WrapPanel";
   Height = 400;
   Width = 500;

   WrapPanel MyWrapPanel = new WrapPanel();

   string[] dirs = Directory.GetFiles(@"C:\Windows\Web\Wallpaper");
   foreach (string dir in dirs)
   {
    FileInfo file = new FileInfo(dir);

    Label lbl = new Label();
    lbl.Width = 100;
    lbl.Content = file.Name;

    MyWrapPanel.Children.Add(lbl);
   }
   Content = MyWrapPanel;
  }
 }
}
```

Rien de très spécial comme on peut l'observer, on créer un nouveau WrapPanel dans lequel on affiche la liste des fichiers que contient le répertoire C:\Windows\Web\Wallpaper sous forme de labels.

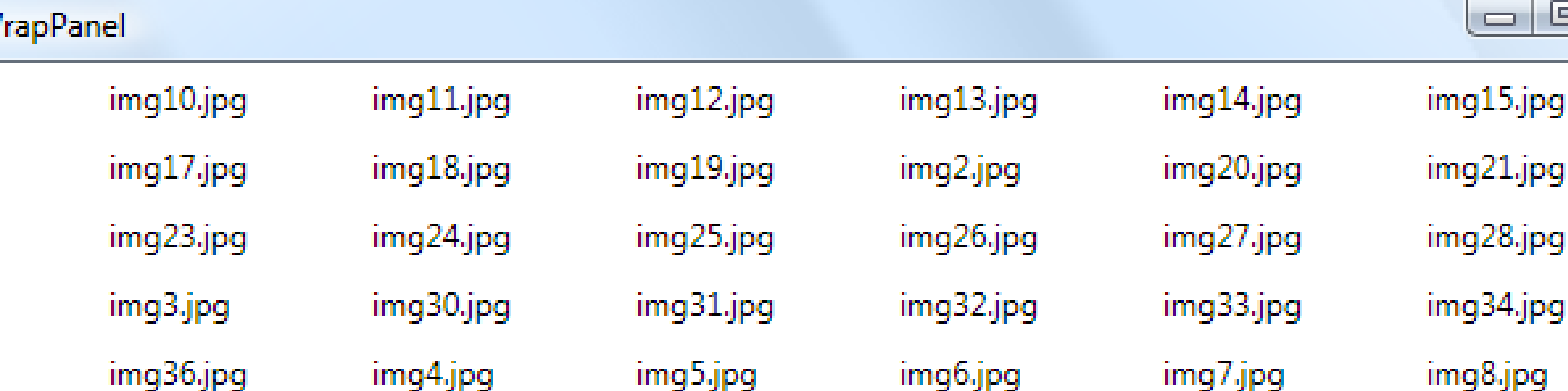
Voici le résultat lorsque l'on redimensionne la fenêtre :

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.



En redimensionnant :

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.



Vous l'aurez donc compris le WrapPanel est très similaire au StackPanel c'est pourquoi je ne détaillerai pas ici le code XAML pour utiliser ce panel.

II-C - Le DockPanel

Le DockPanel n'est pas une véritable évolution offert par WPF. Il permet en effet de docker les éléments sur les bords du panel de façon similaire à la propriété *Dock* des WinForms.

Le DockPanel, contrairement aux autres panels le positionnement des éléments ne va pas dépendre essentiellement de leur ordre de déclaration, mais aussi de ce qu'on appelle les *attached property*.

Pour en savoir plus sur les *attached properties*, je vous invite à lire la doc officiel qui résume très bien leur utilité et de quelles façons elles sont utilisés : **http://msdn.microsoft.com/fr-fr/library/ms749011.aspx**

Dans l'exemple suivant vous allez voir qu'on peut facilement faire devenir notre programme comme une interface de page web avec un *Header* un *Menu* du *Contenu* et un *Footer* pour finir.

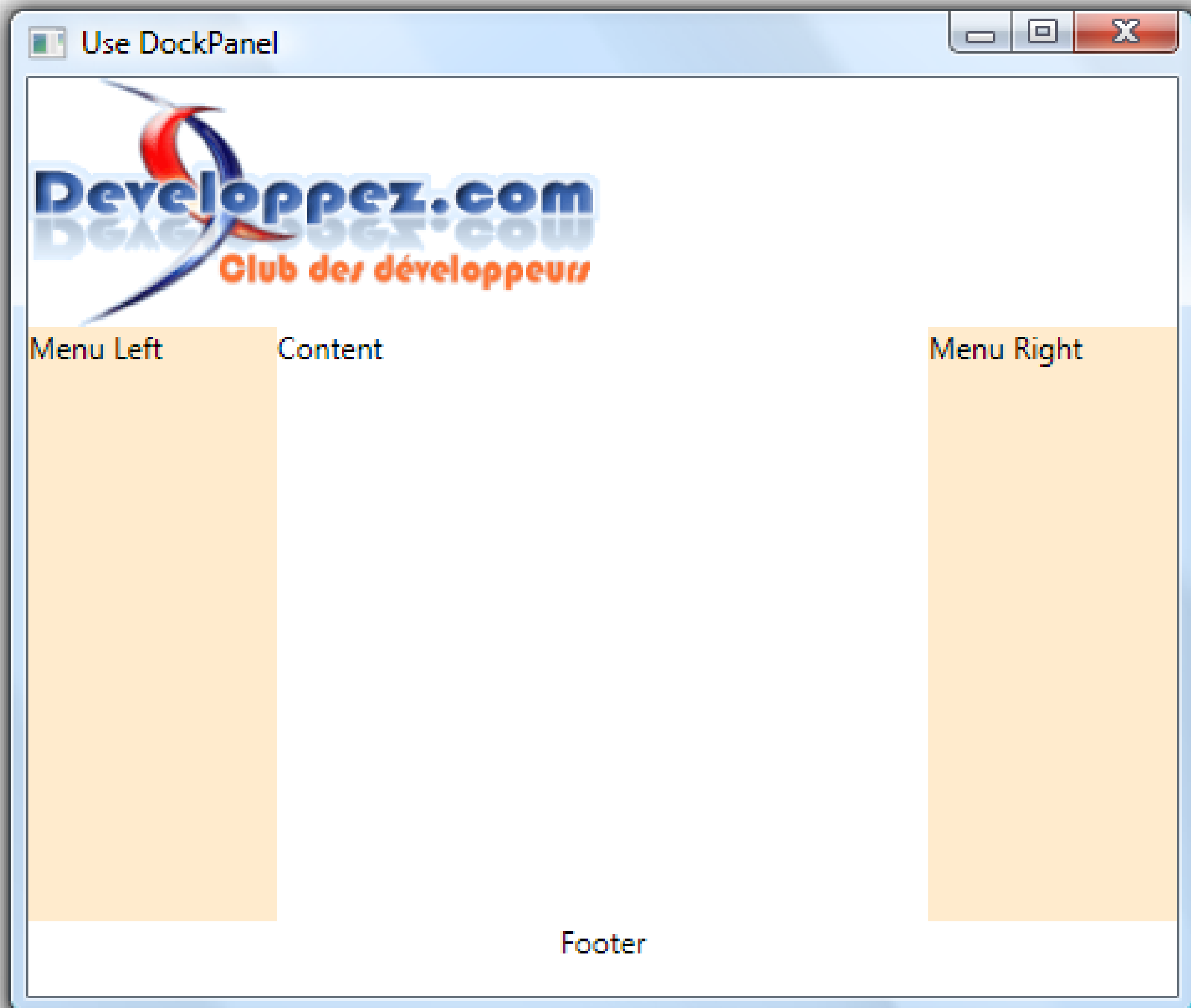
DockPanel XAML

```
<Window x:Class="WpfApplication.Window1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 Title="Use DockPanel" Height="405" Width="478">

 <DockPanel>
  <TextBlock DockPanel.Dock="Top" Height="100">
   <Image Source="http://www.developpez.com/template/logo.gif"/>
  </TextBlock>
  <TextBlock DockPanel.Dock="Bottom" HorizontalAlignment="Center" Height="30">
   Footer
  </TextBlock>
  <TextBlock DockPanel.Dock="Right" Background="BlanchedAlmond" Width="100">
```


Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

DockPanel XAML
```
    Menu Right
  </TextBlock>
  <TextBlock DockPanel.Dock="Left" Background="BlanchedAlmond" Width="100">
    Menu Left
  </TextBlock>
  <TextBlock >
    Content
  </TextBlock>
 </DockPanel>
</Window>
```

Comme vous pouvez le voir c'est très facile d'arriver à ce résultat :



Je vous invite à consulter un article de Microsoft qui vous fera voir les similitudes entre DockPanel et StackPanel et quand les utiliser : **http://msdn.microsoft.com/fr-fr/library/ms754213.aspx**

II-D - Grid

Le panel Grid permet de disposer des contrôles dans une grille c'est-à-dire lignes/colonnes basique.

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

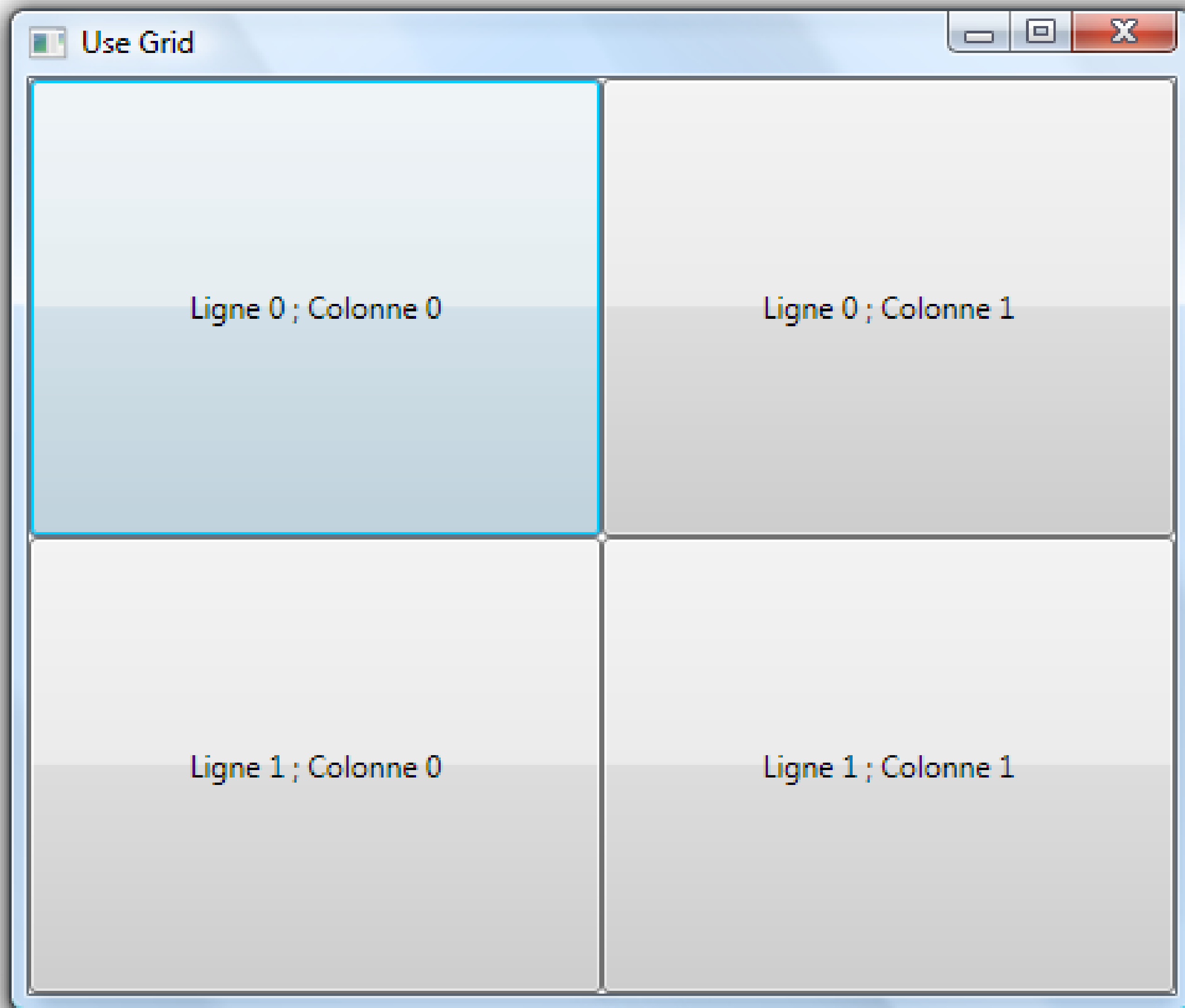
Avant de passer à l'utilisation montrant la réelle utilité du grid panel voici comment faire un gris panel de 2 colonnes et 2 lignes en XAML :

Grid XAML

```
<Window x:Class="WpfApplication.Window1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 Title="Use Grid" Height="405" Width="478">

 <Grid>
  <Grid.RowDefinitions>
   <RowDefinition />
   <RowDefinition />
  </Grid.RowDefinitions>

  <Grid.ColumnDefinitions>
   <ColumnDefinition />
   <ColumnDefinition />
  </Grid.ColumnDefinitions>

  <Button Grid.Row="0" Grid.Column="0">Ligne 0 ; Colonne 0</Button>
  <Button Grid.Row="0" Grid.Column="1">Ligne 0 ; Colonne 1</Button>

  <Button Grid.Row="1" Grid.Column="0">Ligne 1 ; Colonne 0</Button>
  <Button Grid.Row="1" Grid.Column="1">Ligne 1 ; Colonne 1</Button>
 </Grid>
</Window>
```

Nous obtenons le simple résultat suivant :

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.



Comme vous pouvez le voir rien de plus simple !

Pour illustrer maintenant une utilisation dans un cas concret du grid nous allons nous lancer dans le développement de l'IHM d'un puissance 4.

Le but de ce jeu est de faire une ligne de 4 pions sur une grille comptant 6 rangées et 7 colonnes.

On part du principe que chaque case de notre grille sera un simple bouton.

Donc à partir de cette enonce vous pouvez decider de placer les 42 boutons à la main grace au RAD de visual studio; comme vous pouvez decider d'être plus raisonnable et de generer les boutons dynamiquement dans le code behind en calculant leur position sur la fenetre manuellement.

On est d'accord que tout ça n'est franchement pas pratique ?

C'est pourquoi ce cas précis va nous permettre de découvrir que le grid panel est un véritable atout dans ce genre d'application en particulier.

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

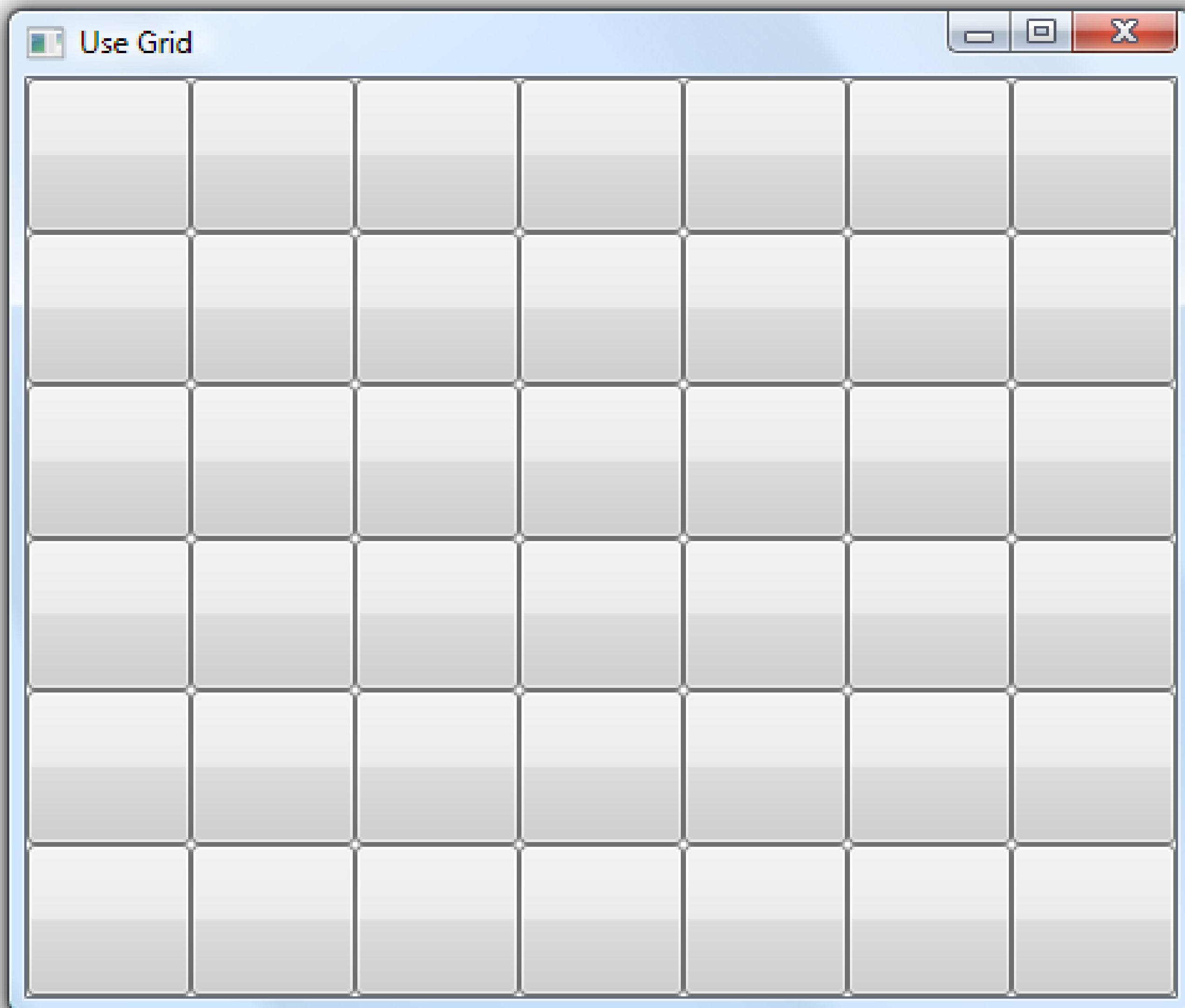
Grid C#

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace WpfApplication
{
 public partial class Window1 : Window
 {
  public Window1()
  {
   Grid MyGrid = new Grid();

   RowDefinition RowGrid1 = new RowDefinition();
   RowDefinition RowGrid2 = new RowDefinition();
   RowDefinition RowGrid3 = new RowDefinition();
   RowDefinition RowGrid4 = new RowDefinition();
   RowDefinition RowGrid5 = new RowDefinition();
   RowDefinition RowGrid6 = new RowDefinition();
   MyGrid.RowDefinitions.Add(RowGrid1);
   MyGrid.RowDefinitions.Add(RowGrid2);
   MyGrid.RowDefinitions.Add(RowGrid3);
   MyGrid.RowDefinitions.Add(RowGrid4);
   MyGrid.RowDefinitions.Add(RowGrid5);
   MyGrid.RowDefinitions.Add(RowGrid6);

   ColumnDefinition ColumnGrid1 = new ColumnDefinition();
   ColumnDefinition ColumnGrid2 = new ColumnDefinition();
   ColumnDefinition ColumnGrid3 = new ColumnDefinition();
   ColumnDefinition ColumnGrid4 = new ColumnDefinition();
   ColumnDefinition ColumnGrid5 = new ColumnDefinition();
   ColumnDefinition ColumnGrid6 = new ColumnDefinition();
   ColumnDefinition ColumnGrid7 = new ColumnDefinition();
   MyGrid.ColumnDefinitions.Add(ColumnGrid1);
   MyGrid.ColumnDefinitions.Add(ColumnGrid2);
   MyGrid.ColumnDefinitions.Add(ColumnGrid3);
   MyGrid.ColumnDefinitions.Add(ColumnGrid4);
   MyGrid.ColumnDefinitions.Add(ColumnGrid5);
   MyGrid.ColumnDefinitions.Add(ColumnGrid6);
   MyGrid.ColumnDefinitions.Add(ColumnGrid7);

   for (int i = 0; i < 6; i++ )
   {
    for (int ii = 0; ii < 7; ii++)
    {
     Button ButtonGrid = new Button();
     Grid.SetRow(ButtonGrid, i);
     Grid.SetColumn(ButtonGrid, ii);

     MyGrid.Children.Add(ButtonGrid);
    }
   }

   Content = MyGrid;
  }
 }
}
```

Et voilà il ne vous reste plus qu'à travailler avec votre collection habituel Children et le tour est joué, vos boutons se sont placés tout seul dans votre Window grâce a votre Grid.

Résultat :

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.



Nous n'aborderont pas ici l'UniformGrid qui reste très similaire au Grid à la seule différence que dans un UniformGrid les colonnes et les lignes ont forcément la même taille ce qui n'est pas toujours le cas dans un Grid.

II-E - GridSplitter

Je ne vais pas non plus m'attarder sur le GridSplitter tout simplement, car c'est un contrôle très similaire au SplitContainer des WinForms ? Le GridSplitter permet donc de modifier la taille des lignes/colonnes pendant l'exécution de l'application.

Il n'y a pas eu grande évolution entre les deux contrôles à part la possibilité grâce à WPF de le déclarer en XAML c'est pourquoi je vais tout de même vous montrer comment déclarer un GridSplitter en XAML.

Dans l'exemple suivant on va déclarer un Grid (voir exemple précédent) puis un GridSplitter qui permettra de modifier la taille de la ligne du Grid.

GridSplitter XAML

```
<Window x:Class="WpfApplication.Window1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```


Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

GridSplitter XAML
```
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Use Grid" Height="405" Width="478">

  <Grid>
   <Grid.RowDefinitions>
    <RowDefinition/>
    <RowDefinition/>
   </Grid.RowDefinitions>

   <GridSplitter Grid.Row="1" Background="Black" HorizontalAlignment="Stretch" VerticalAlignment="Top" Height="4"
  </Grid>
</Window>
```

Et le résultat de ça :

II-F - Canvas

Le Canvas vous permet de placer un contrôle sur votre Window grâce à des coordonnées X (abscisse) et Y (ordonné) en DIP bien sûr. Vous l'aurez compris cela va poser un problème au niveau du redimensionnement de la fenêtre c'est pourquoi le Canvas est peu utilisé. Préférez utiliser un panel avant tout, c'est d'ailleurs pour cela que je le cite ici en dernier dans cet article sur le Layout WPF.

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

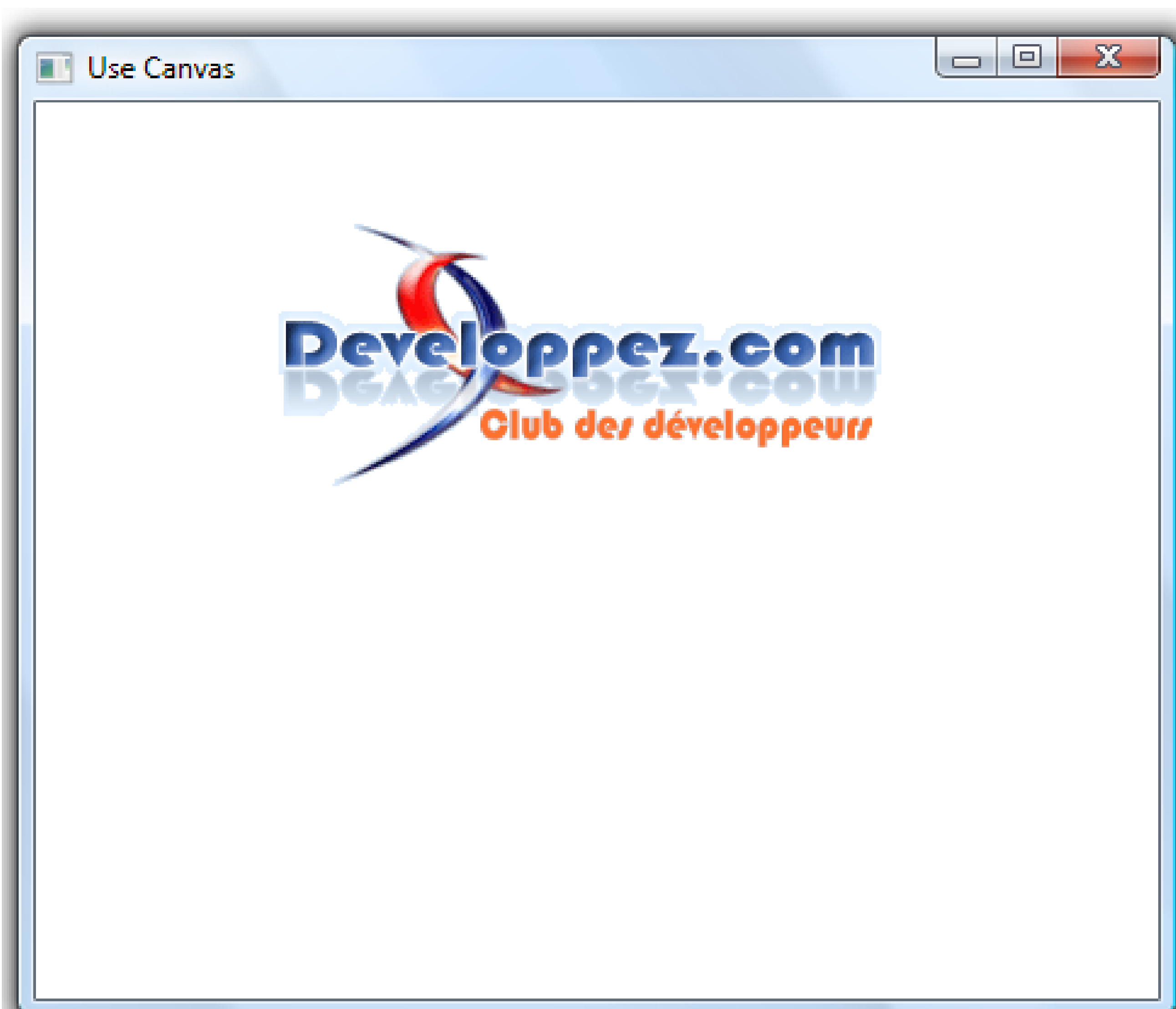
Exemple d'utilisation du Canvas en XAML :

Canvas XAML

```
<Window x:Class="WpfApplication.Window1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 Title="Use Canvas" Height="405" Width="478">

 <Canvas>
  <Image Source="http://www.developpez.com/template/logo.gif" Canvas.Left="100" Canvas.Top="50"/>
 </Canvas>
</Window>
```

On peut observer que l'utilisation du Canvas se fait via une *attached property*.

Voici le résultat :



Et maintenant le même résultat en redimensionnant notre fenêtre :

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

Use Canvas

III - Conclusion

Pour conclure, on peut voir que WPF est une avancé considérable dans le développement d'IHM en .NET.

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.

En effet si vous voulez comparer en détail ces avancés je vous conseille ce très bon article de Vincent Laine : **http://vincentlaine.developpez.com/tuto/dotnet/placementcompos/** qui traite le même sujet, mais en .NET 2.0 c'est-à-dire avant WPF.

On pourrait se demander pourquoi une telle avancée du layout entre .NET 2.0 et .NET 3.0 ? Et bien Microsoft a vu en WPF une nouvelle façon de concevoir nos applications, on aura pu voir la disparition dans WPF du DataGridView ce qui a été longtemps critiqué, mais à la base cela tiens vraiment d'une volonté de changer notre style de présentation d'où une aussi grande evolutivité du layout.

Je vous conseille de visionner également cette vidéo de Billy Hollins qui traite justement de ce sujet : **http://perseus.franklins.net/dnrtvplayer/player.aspx?ShowNum=0115**

IV - Remerciements

Je tiens à remercier toute l'équipe .NET de developpez.com pour leur relecture.

Les sources présentées sur cette pages sont libre de droits, et vous pouvez les utiliser à votre convenance. Par contre cette page de présentation de ces sources constitue une oeuvre intellectuelle protégée par les droits d'auteurs. Copyright © 2008 - Matthieu Dordolo. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à 3 ans de prison et jusqu'à 300 000 E de dommages et intérets. Droits de diffusion permanents accordés à developpez LLC.