

WPF et XAML

Présentation	2
Introduction à la mise en forme WPF	3
Pourquoi les composants layout sont si importants.....	3
Bonne pratiques.....	3
Alignement Vertical et Horizontal	3
Margin et Padding	4
Height et Width.....	4
Scrolling.....	5
Grid Panel	5
Définir les lignes et les colonnes (Rows Columns)	6
Comment ajouter des contrôles dans un tableau	6
StackPanel	7
Introduction.....	7
Empiler des Items horizontalement	8
Dock Panel	9
Introduction.....	9
Plusieurs éléments du même coté.....	9
Ajouter une fenêtre dans un projet WPF.....	12
Ressources	13
Définition d'une ressource.....	13
Utilisation d'une ressource	13
Styles	14
Déclaration d'un style	14
Utilisation du style.....	14
Style utilisant un TargetType	14

Présentation

Introduction à la mise en forme WPF

Pourquoi les composants layout sont si importants

La mise en forme de contrôles est une phase importante dans la création d'une application et dans sa future utilisation. Positionner les contrôles en utilisant les coordonnées absolues en pixel peut fonctionner dans un environnement statique, mais si vous voulez que votre application s'adapte à différentes résolutions d'écrans, vous devez utiliser un système de positionnement différent. WPF fournit un large éventail de contrôle parent permettant le positionnement d'autres contrôles enfants pour construire des interfaces riches.

Il y a cinq composants (layout panel) utilisés couramment

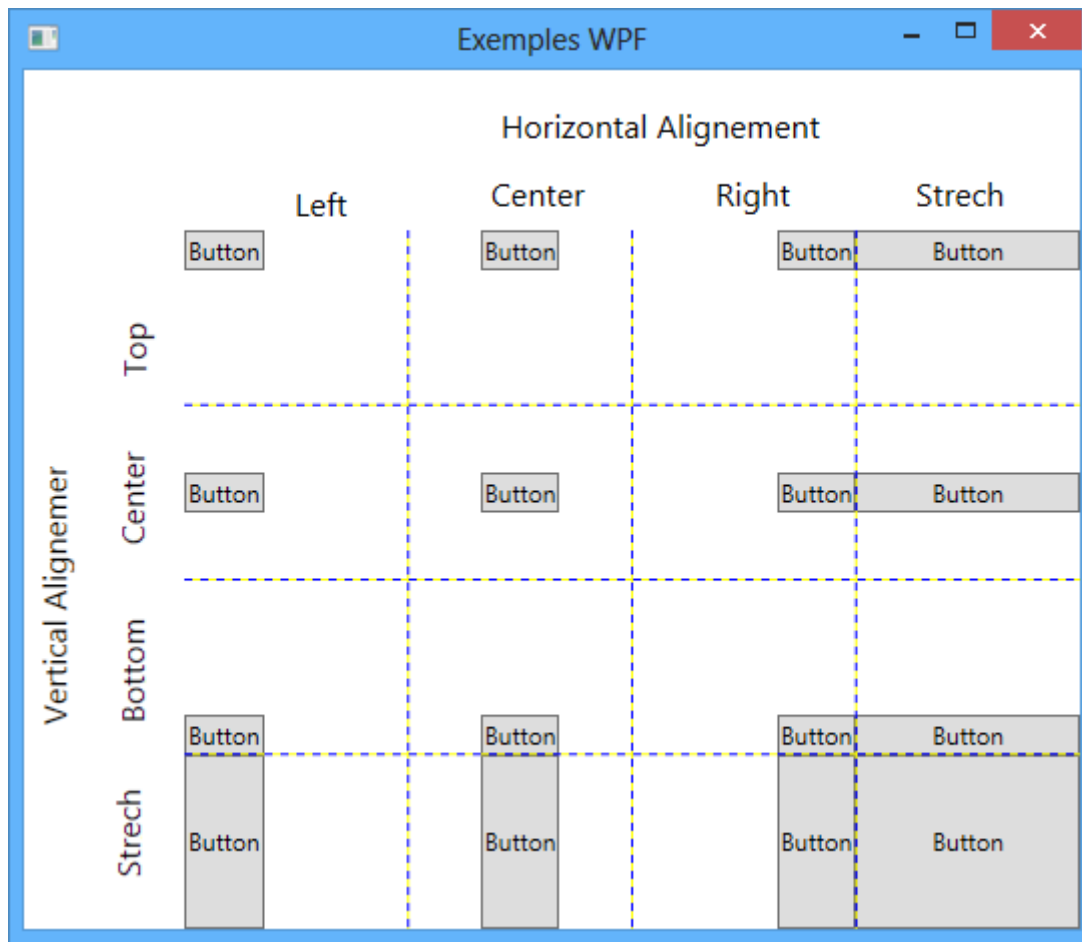
- Grid Panel (positionnement en grille)
- Stack Panel (positionnement par empilage vertical ou horizontal)
- Dock Panel (positionnement par accrochage Haut, Bas, Droite, Gauche, Milieu)
- Wrap Panel (positionnement horizontal avec retour à la ligne)
- Canvas Panel (positionnement absolu (x,y) des contrôles)

Bonne pratiques

- Éviter les positionnements fixes - utilisez les propriétés `Alignment` en combinaison avec `Margin` pour positionner un élément dans un panel
- Éviter les tailles fixes - positionnez `Width` et `Height` sur `Auto` autant que possible.
- N'abusez pas des canvas panel
- Utilisez des `StackPanel` pour positionner des Bouttons dans une fenêtre.
- Utilisez un `GridPanel` pour positionner un formulaire. Créez une colonne à taille `Auto` pour les labels et une autre à taille `*` pour les `TextBox`.

Alignement Vertical et Horizontal

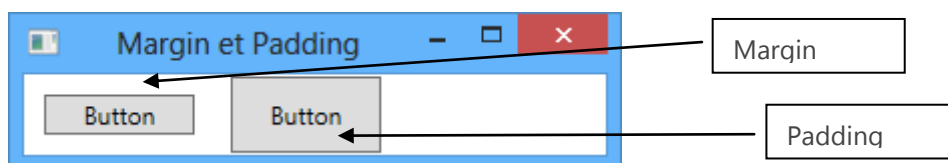
Utilisez les propriétés `VerticalAlignment` et `HorizontalAlignment` pour accrocher un contrôle aux bords du panel. L'illustration suivante montre les différentes combinaisons.



Margin et Padding

Les propriétés `Margin` and `Padding` sont utilisées pour réserver de l'espace dans et autour du contrôle.

- `Margin` spécifie de l'espace autour du contrôle.
- `Padding` spécifie de l'espace dans le contrôle.



Height et Width

Bien que ce ne soit pas recommandé, les contrôles fournissent des propriétés `Height` et `Width` pour fixer la taille des éléments. Utilisez plutôt `MinHeight`, `MaxHeight`, `MinWidth` et `MaxWidth` propriétés pour spécifier la largeur et la hauteur des contrôles.

Si vous positionnez la largeur ou la hauteur sur `Auto`, le contrôle prendra la taille du panel.

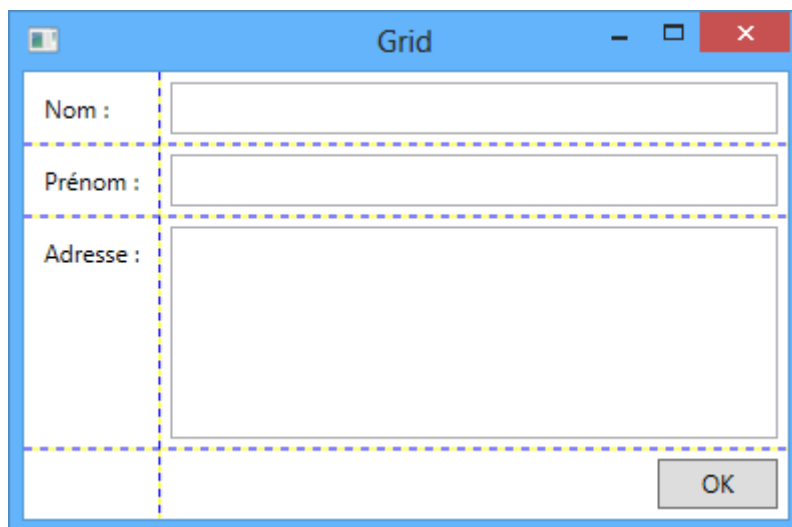
Scrolling

Quand un contenu est trop grand pour être affiché, vous pouvez utiliser un `ScrollView`. Le `ScrollView` utilise deux scrollbars pour choisir la portion visible.

La visibilité des scrollbars peut être contrôlé par la propriété `ScrollbarVisibility`.

```
<ScrollView>
  <StackPanel>
    <Button Content="Clique 1" />
    <Button Content=" Clique 2" />
    <Button Content=" Clique 3" />
  </StackPanel>
</ScrollView>
```

Grid Panel



Nom :	<input type="text"/>
Prénom :	<input type="text"/>
Adresse :	<input type="text"/>
	<input type="button" value="OK"/>

Le composant `Grid` permet d'arranger ses composants enfants dans un tableau structuré en lignes et colonnes. Les fonctionnalités sont comparables aux tableau HTML.

La taille des enfants est définie par les propriétés `HorizontalAlignment` et `VerticalAlignment` qui définissent des ancrs.

La distance entre l'ancre et l bord du tableau est spécifié par la propriété `margin` du contrôle enfant.

Définir les lignes et les colonnes (Rows Columns)

Un composant grid possède une ligne et une colonne par défaut.. Pour créer des lignes et de colonnes additionnelles, vous devez ajouter des éléments `RowDefinition` à la propriété `RowDefinitions` du Grid et des éléments `ColumnDefinition` à la propriété `ColumnDefinitions` du Grid. L'exemple suivant permet de construire un tableau de trois ligne et deux colonnes.

La taille peut spécifier en indiquant le valeur en nombre de pixels, comme un pourcentage ou automatiquement.

Auto Prend la taille désirée par les contrôles enfants

* Prend tout l'espace disponible (après avoir dimensionnée les éléments fixes et Auto)..

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
    <RowDefinition Height="28" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="200" />
  </Grid.ColumnDefinitions>
</Grid>
```

Comment ajouter des contrôles dans un tableau

Pour ajouter des éléments dans les cases du tableau, vous devez spécifier pour chaque éléments, le numéro de la ligne et de la colonne avec les propriétés `Grid.Column` et `Grid.Row`.

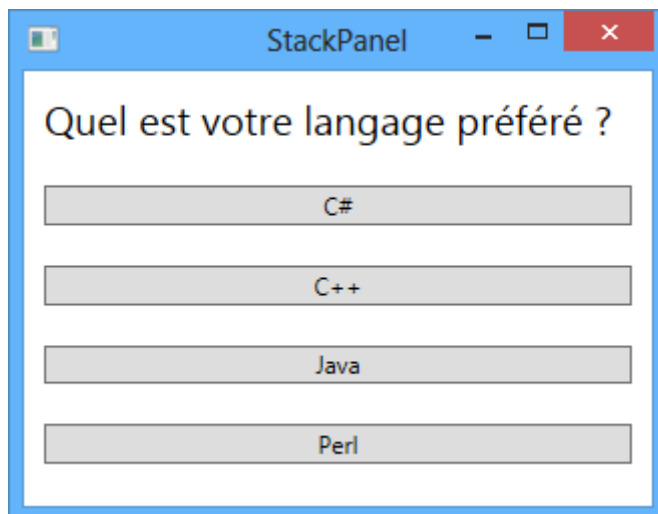
```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
    <RowDefinition Height="28" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
```

```

    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="200" />
</Grid.ColumnDefinitions>
<Label Grid.Row="0" Grid.Column="0" Content="Name:"/>
<Label Grid.Row="1" Grid.Column="0" Content="E-Mail:"/>
<Label Grid.Row="2" Grid.Column="0" Content="Comment:"/>
<TextBox Grid.Column="1" Grid.Row="0" Margin="3" />
<TextBox Grid.Column="1" Grid.Row="1" Margin="3" />
<TextBox Grid.Column="1" Grid.Row="2" Margin="3" />
<Button Grid.Column="1" Grid.Row="3" HorizontalAlignment="Right"
        MinWidth="80" Margin="3" Content="Send" />
</Grid>

```

StackPanel



Introduction

Le composant `StackPanel` est très utilisé, il permet d'empiler facilement des contrôles horizontalement ou verticalement en fonction de l'orientation. C'est un composant très pratique pour créer des listes, les contrôles WPF `ItemsControls` comme les `ComboBox`, `ListBox` ou `Menu` utilisent le `StackPanel` pour leur agencement interne.

```

<StackPanel>
    <TextBlock Margin="10" FontSize="20">Quel est votre langage préféré ?</TextBlock>

```

```

<Button Margin="10">C#</Button>
<Button Margin="10">C++</Button>
<Button Margin="10">Java</Button>
<Button Margin="10">Perl</Button>
</StackPanel>

```

Empiler des Items horizontalement

Un bon exemple d'empilage horizontal de contrôles est les boutons OK et Annuler de boîtes de dialogue. La taille du texte peut changer si la police est modifiée ou lors de la traduction du logiciel un texte plus ou moins long peut apparaître. Le StackPanel aligne les éléments en utilisant la taille demandée.

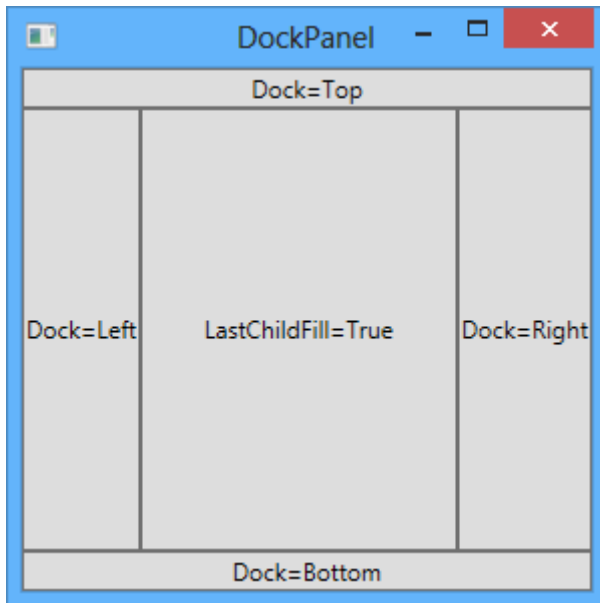


```

<StackPanel Margin="8" Orientation="Horizontal">
  <Button MinWidth="93">OK</Button>
  <Button MinWidth="93" Margin="10,0,0,0">Cancel</Button>
</StackPanel>

```


Dock Panel



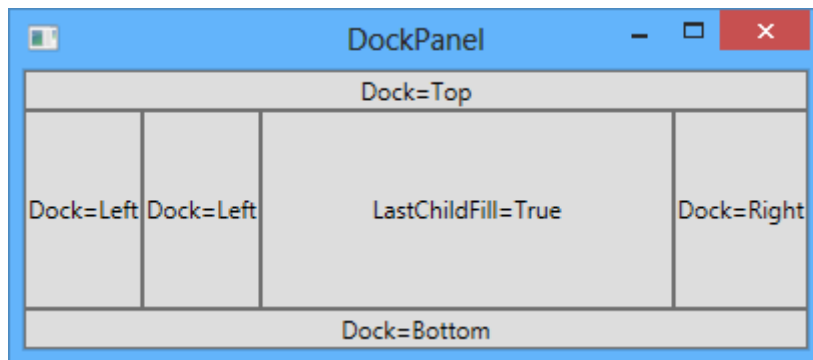
Introduction

Le DockPanel est un composant de mise en forme qui permet facilement d'accrocher des éléments sur les bords (left, right, top, bottom, center). Le point d'accroche est défini par la propriété `DockPanel.Dock`. Pour accrocher un élément au milieu, il suffit d'ajouter la propriété `LastChildFill=True` dans le dockPanel et que l'élément soit le dernier dans la liste.

```
<DockPanel LastChildFill="True">
    <Button Content="Dock=Top" DockPanel.Dock="Top"/>
    <Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>
    <Button Content="Dock=Left" DockPanel.Dock="Left"/>
    <Button Content="Dock=Right" DockPanel.Dock="Right"/>
    <Button Content="LastChildFill=True"/>
</DockPanel>
```

Plusieurs éléments du même côté

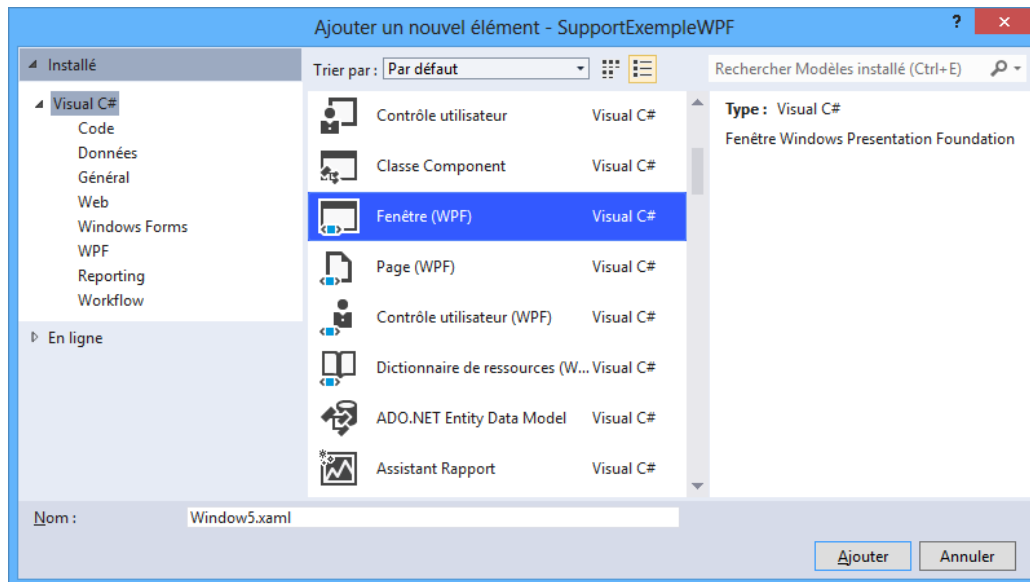
Le DockPanel autorise l'accroche de plusieurs éléments au même endroit.



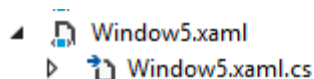
```
<DockPanel LastChildFill="True">  
    <Button Content="Dock=Top" DockPanel.Dock="Top"/>  
    <Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>  
    <Button Content="Dock=Left" DockPanel.Dock="Left"/>  
    <Button Content="Dock=Left" DockPanel.Dock="Left"/>  
    <Button Content="Dock=Right" DockPanel.Dock="Right"/>  
    <Button Content="LastChildFill=True"/>  
</DockPanel>
```


Ajouter une fenêtre dans un projet WPF

Clique droit sur le projet puis choisissez Ajouter Fenêtre, spécifier le nom de la fenêtre.



Visual Studio ajoute un fichier xaml est un fichier xaml.cs



Pour modifier la fenêtre de démarrage de votre projet, vous devez modifier le fichier App.xaml et modifier la propriété StartUri de l'objet Application

```
<Application x:Class="SupportExempleWPF.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    StartupUri="Window5.xaml" >
    <Application.Resources>

    </Application.Resources>
</Application>
```

Ressources

Une ressource WPF est un objet stocké dans un dictionnaire. Vous pouvez définir des dictionnaires au niveau de l'application, de la fenêtre ou d'un élément (grid, panel, ...).

Définition d'une ressource

Définition d'une ressource de type SolidColorBrush (pinceau) dans le dictionnaire de l'application.

```
<Application.Resources>
    <SolidColorBrush x:Key="Couleur1" Color="Aqua" />
</Application.Resources>
```

Utilisation d'une ressource

En xaml

```
<Label Background="{DynamicResource Couleur1}" />
```

En csharp

```
SolidColorBrush couleur =
(SolidColorBrush)Application.Current.FindResource("Couleur1");
```

Styles

Un style WPF permet de modifier les propriétés d'un objet, il est stocké dans un dictionnaire de ressources et il est identifié par une clé (x:Key) et par son type cible (TargetType).

Déclaration d'un style

```
<Style x:Key="MonStyleDeBouton" TargetType="{x:Type Button}" >  
    <Setter Property="Background" Value="{DynamicResource Couleur1}"></Setter>  
</Style>
```

Utilisation du style

```
<Button Style="{DynamicResource ResourceKey=MonStyleDeBouton}"/>
```

Style utilisant un TargetType

```
<Style TargetType="{x:Type Button}" >  
    <Setter Property="Background" Value="{DynamicResource Couleur1}"></Setter>  
</Style>
```

Si l'objet Style ne spécifie pas de nom (x:Key) mais seulement le TargetType, le style s'applique implicitement à tous les objets de ce type.

Les styles doivent être stockés dans un dictionnaire de ressources.