Ecole Nationale des Sciences Appliquées de Safi

A.U:2019-2020

Développement en c#.NET

Ing. Meryem OUARRACHI

Plan du module

Langage C#

- L'environnement .Net
- Initiation à la programmation C#
- Programmation Orienté Objet C#

Programmation avancée en .Net ,C#

- Programmation distribuée
- Gestion de base de donnée
- **Application WPF**

Chapitre 7:

WPF

Plan du chapitre

1-Les contrôles WPF

2-Les triggers en wpf

3-Les animations en wpf

4-Le binding en WPF

5-Les validateurs en wpf

6-Les bibliothèques en wpf

Remarque

-En WPF il y’a deux niveaux de travail:

- **Niveau1:** on se base sur la technique drag/drop de windowsForms et tout ce qui est vu dans cette partie est valable ici
- **Niveau2(but de ce cours):** se base sur la richesse de wpf ,là on utilise les techniques de expression blend,les bibliothèques de wpf(telerik, syncfusion...);le principe de notification…

Intoduction

- **Les inconvénients de windowsForms:**

-Interface graphique non riche.(Windows xp)

-Pas de possibilité de Séparation code / design.

```
Button b=new Button();
b.Text="valider";
b.Width=100;
```

Les avantages WPF

-WPF est une solution Microsoft dont le but est de permettre le développement d'applications riches, proposant une expérience utilisateur innovante et interactive, au moyen d’animations, de transformations, de l'utilisation des médias (audio/vidéo), etc…

-WPF propose un ensemble de nouvelles APIs constituant la base du système d’affichage des nouvelles versions de Windows.

Productivité Développeur-Designer

Designer

Conçoivent les UI

XAML

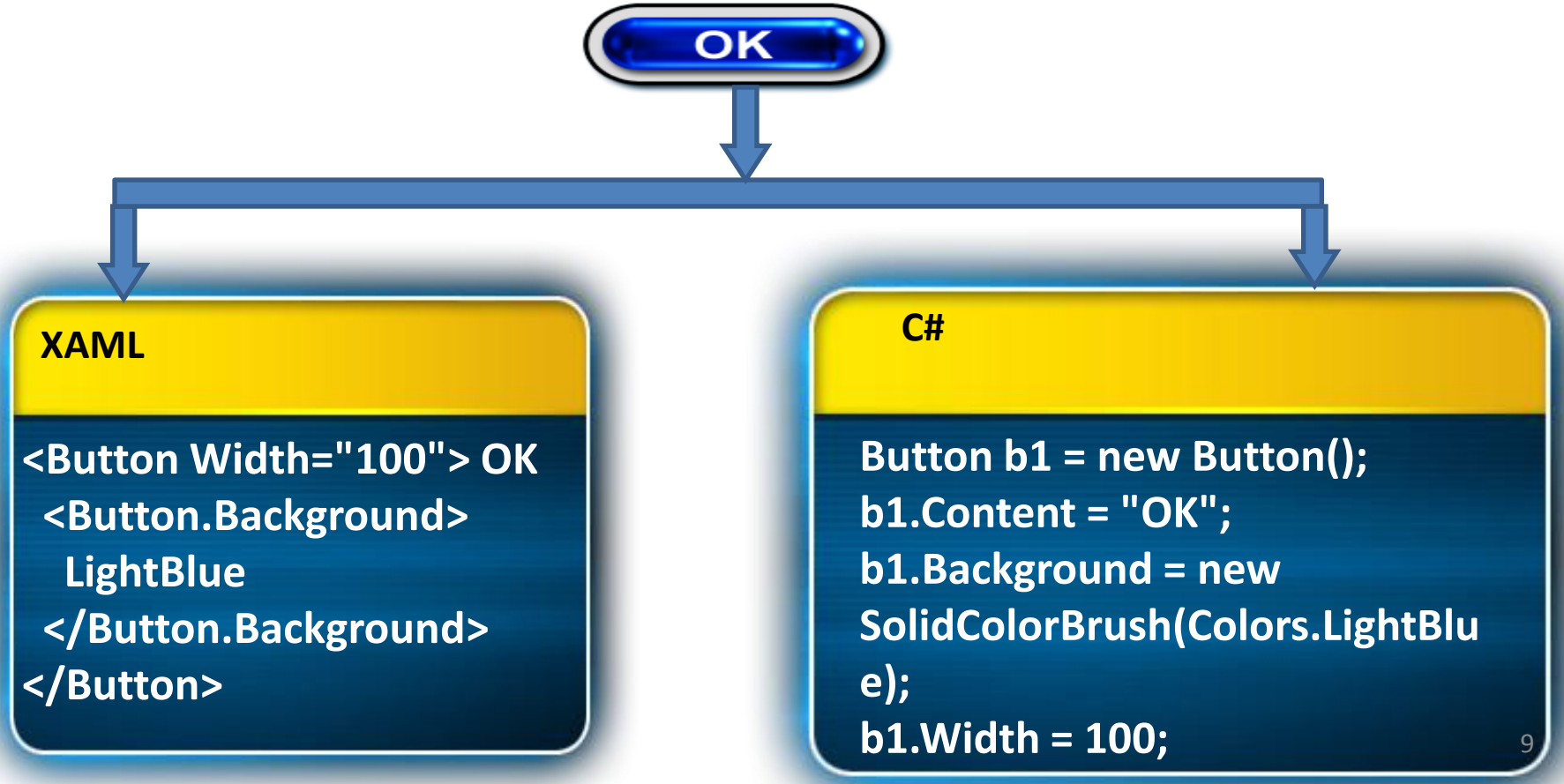
Avec XAML, les développeurs et les designers peuvent affiner leur collaboration

Développeur

Ajoute la logique métier

XAML - eXtensible Application Markup Language-

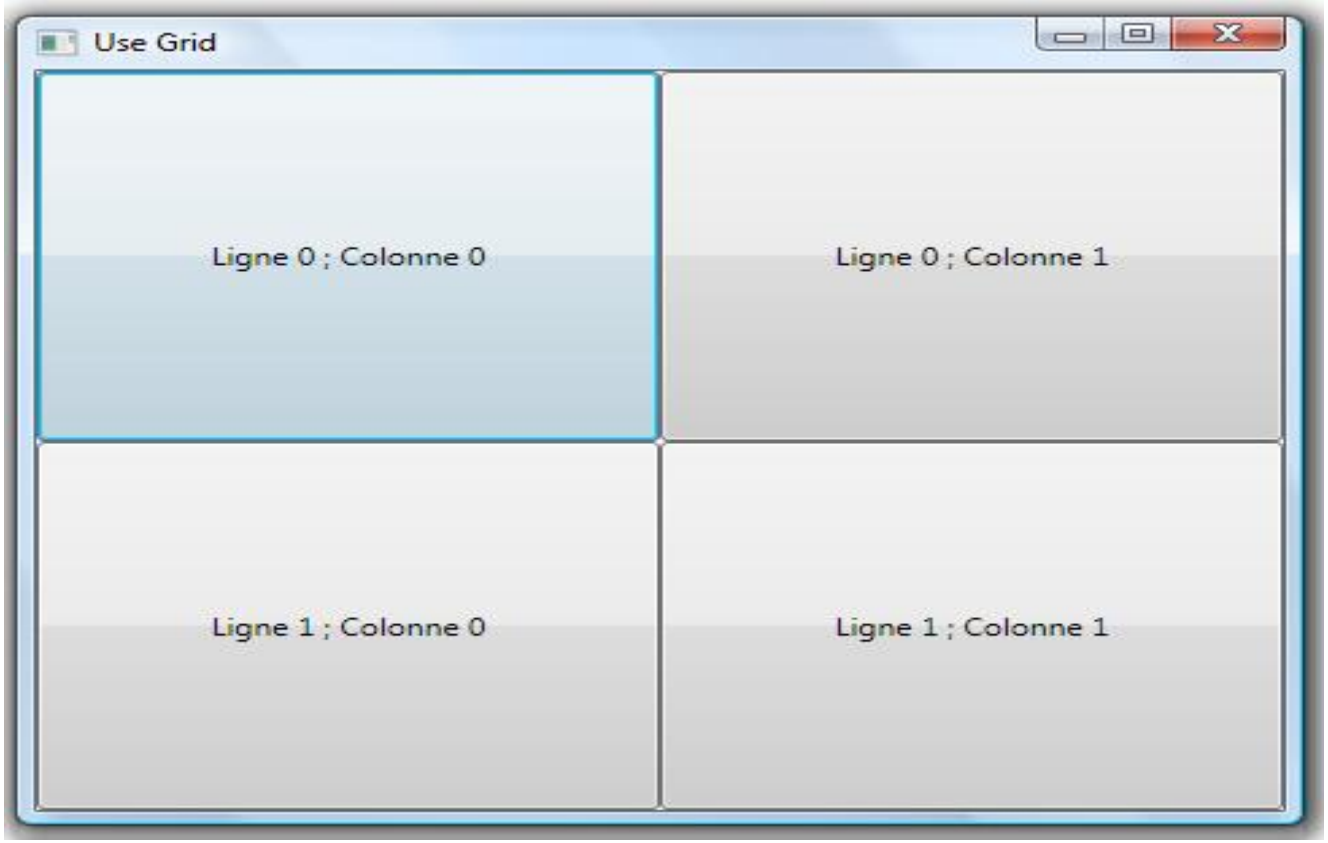
- Facilement utilisable, basé sur le XML
- Peut-être affiché dans un navigateur ou une application



Contrôles WPF

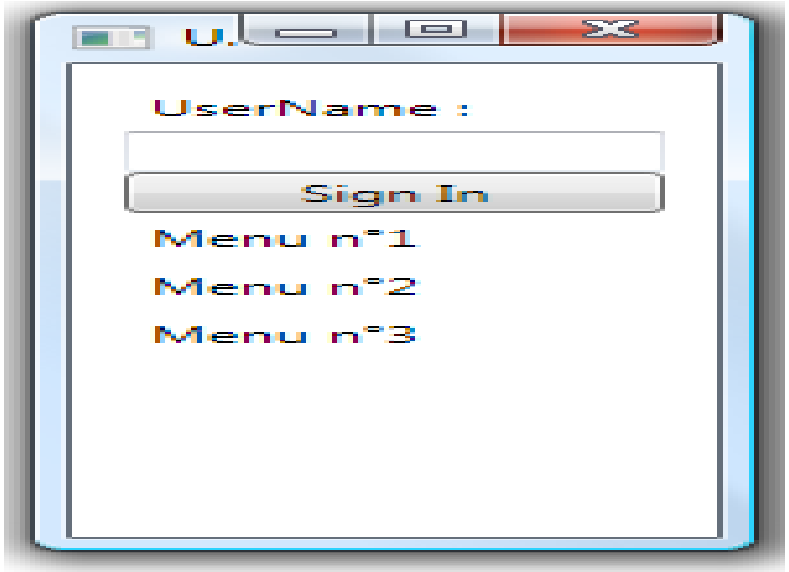
Les contrôles de WPF

-Grid: est le conteneur par défaut où nous pouvons organiser les éléments sous forme de tableau.



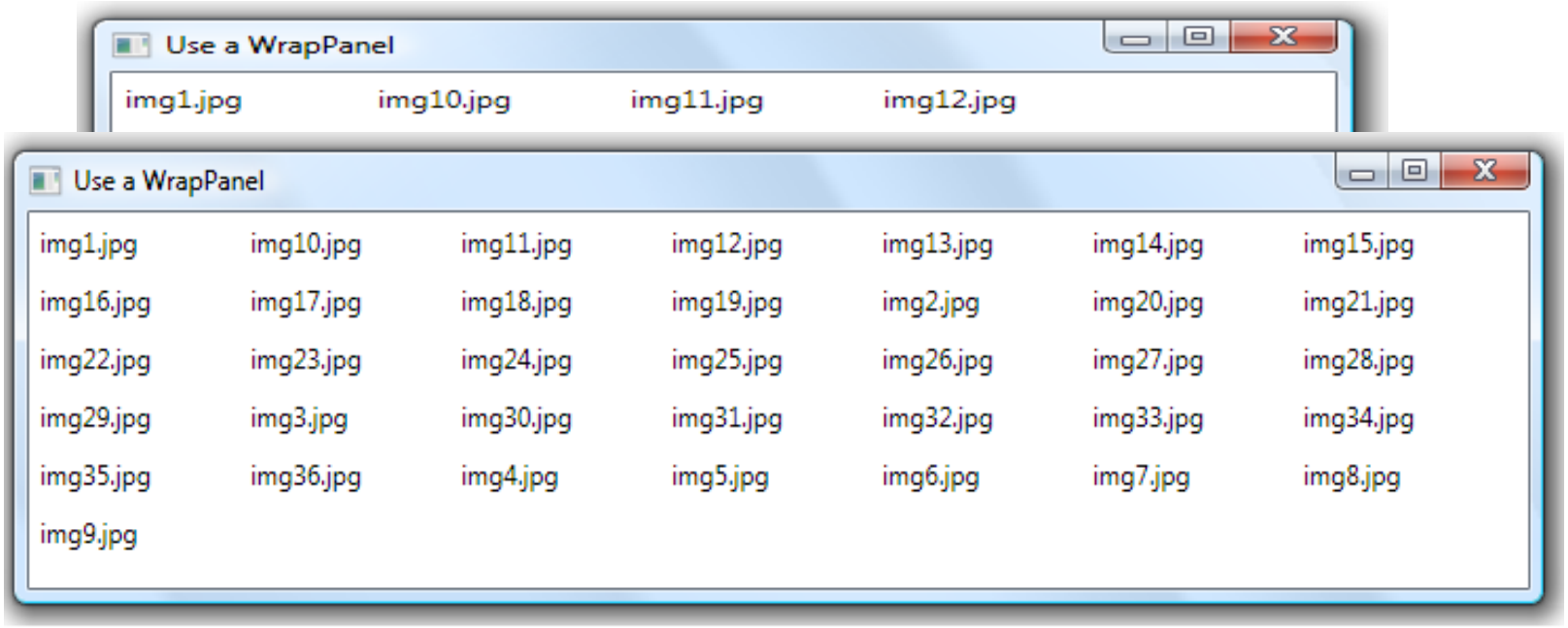
Les contrôles de WPF

-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*



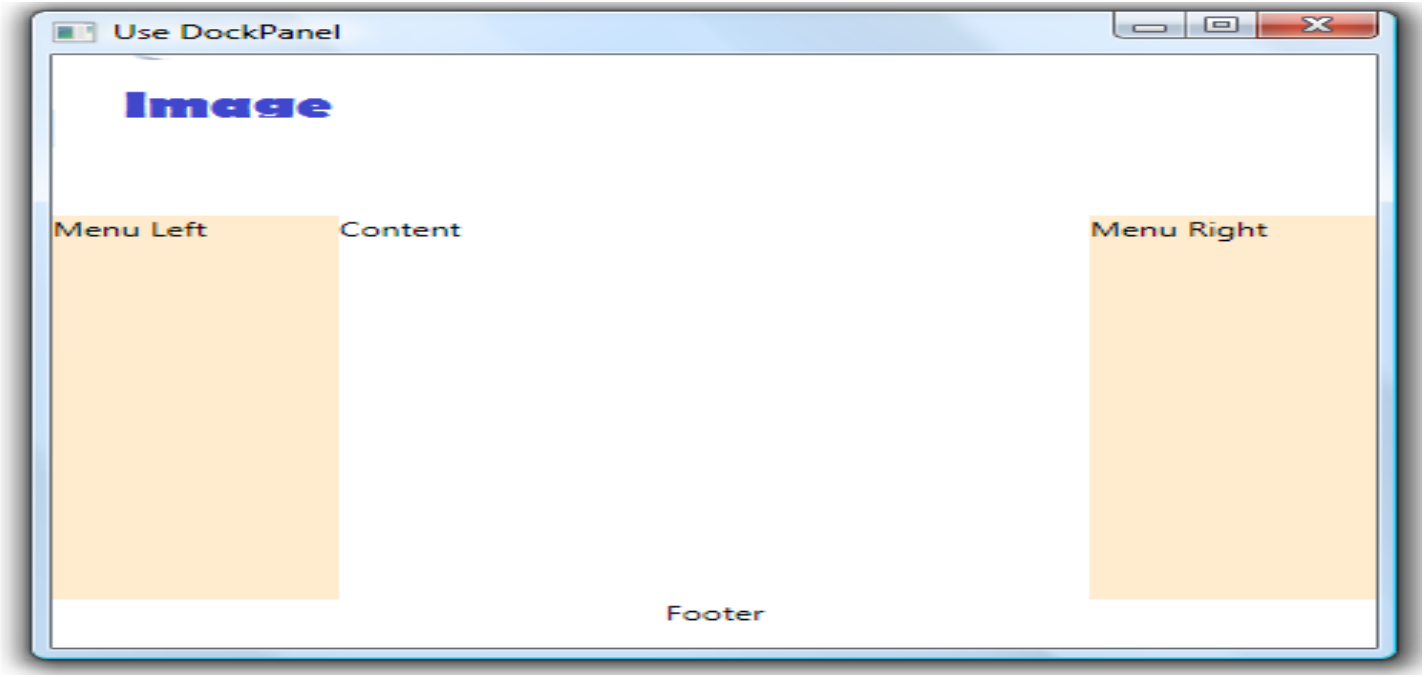
Les contrôles de WPF

-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.



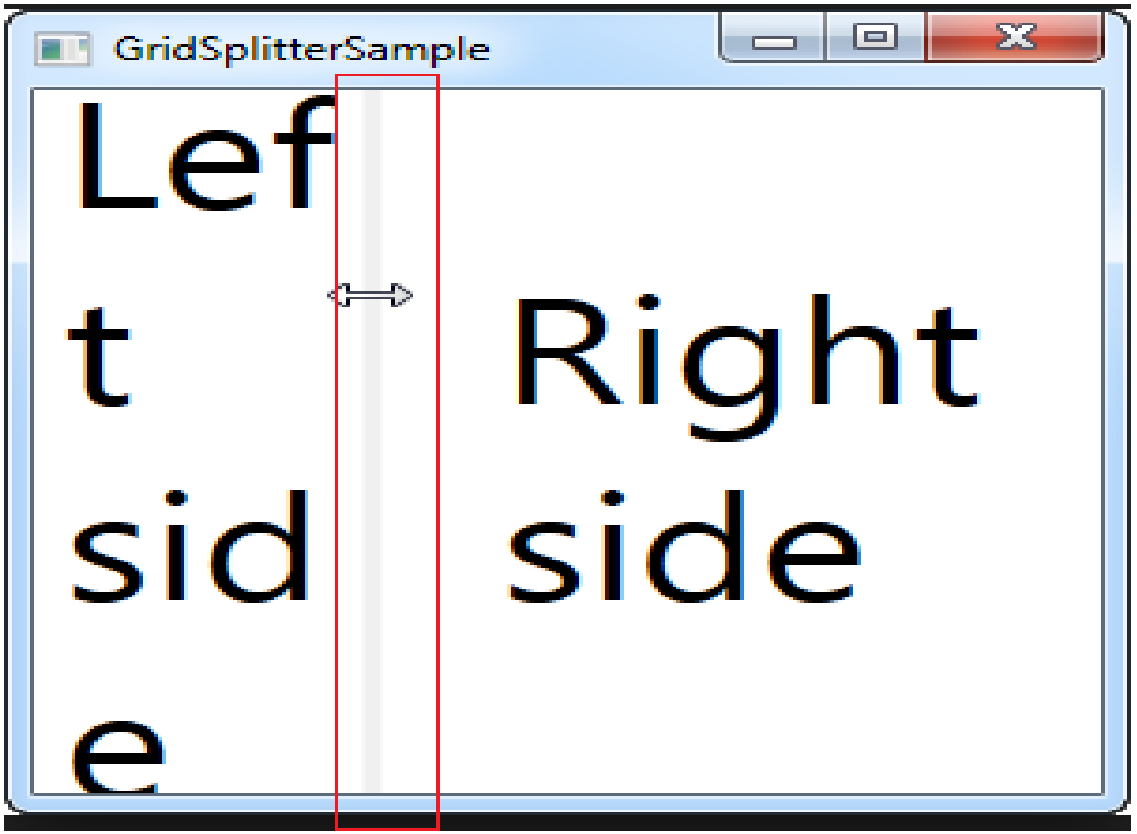
Les contrôles de WPF

-DockPanel: permet de placer le contrôle dans un endroit précis comme en Gauche, Droite, Haut et Bas.Pour spécifier l'endroit on utilise la propriété DockPanel.Dock (Top,Left,Right,Bottom)



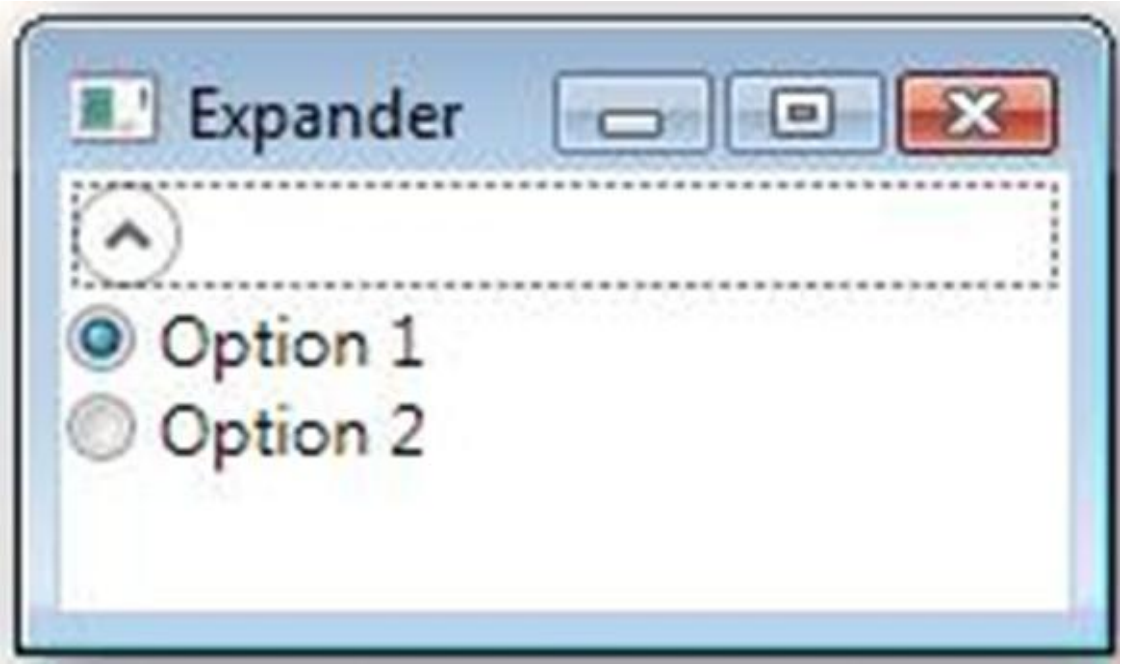
Les contrôles de WPF

-GridSplitter: Représente le contrôle qui redistribue l'espace entre les colonnes ou les lignes d'une grille de contrôle



Les contrôles de WPF

-Explander: est comme un GroupBox mais avec la caractéristique supplémentaire pour réduire et développer son contenu.



Les contrôles de WPF

Exemple:

-Créer un projet WPF

-Tester ces composants et ajouter des effets sur ce projet en utilisant Expression Blend

Remarque: Si on a une erreur pour ouvrir projet visual studio en blend: Ouvrir Cprojet file-->chercher la ligne <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />

Remplacer par :<Import Project="$(MSBuildBinPath)\Microsoft.CSharp.targets" />

Les contrôles utilisateurs

UserControl est tout simplement un Control contenant d'autres contrôles

-Pour le créer : Add →User control

-Pour l'utiliser: ajouter cette ligne dans la balise de windows: **xmlns:lc="clr-namespace:NomProjetWPF"**

-Pour l'appeler: <lc:Usercontrol1></lc:Usercontrol1>

Les styles

-Définissent l'apparence des éléments au sein de l'application(comme les CSS en html). quelques propriété de la classe **Style** :

-Setters : Contient une collection de Setter ou d'EventSetter.

-TargetType : Indique pour quel type d'élément le style sera utilisé.

-BasedOn : Permet de créer le style sur base d'un autre style. Cela permet l'héritage de style.

Les styles

-Utilisation des Setter pour paramétrer certaines valeurs de notre style.On a 2 types de setters:

-Les setters de propriétés

```
<Setter Property="Button.Background" Value="Green" />
```

-Les setter d'évènement

```
<EventSetter Event="Button.Click" Handler="bt1_Click"
/>
```

Les styles

Exemple: On met ce code entre <Window.Resources>

</Window.Resources>

```
<!--XAML-->
<Style x:Key="MonStyleBleu">
    <Setter Property="Button.Background" Value="Aqua" />
    <Setter Property="Control.Margin" Value="10" />
    <Setter Property="TextBox.Height" Value="50" />
</Style>
```

- Style qui hérite les propriétés d'un autre style

```
<!--XAML-->
<Style x:Key="MonStyleVert" BasedOn="{StaticResource MonStyleBleu}">
    <Setter Property="Button.Background" Value="Green" />
</Stvle>
```

Les styles

Exemple:

- Utiliser un style

```
<!--XAML-->
<StackPanel>
    <Button Style="{StaticResource MonStyleVert}" />
    <TextBox Style="{StaticResource MonStyleBleu}" ></TextBox>
</StackPanel>
```

Les styles

Remarque: pour appliquer un style dans l'ensemble des fenêtres: il faut le définir dans la racine « Application »

```
<Application x:Class="WpfApplicationtest.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="Window4.xaml">
    <Application.Resources>
        <Style x:Key="MonStyleBleu" TargetType="Button">
            <Setter Property="Button.Background" Value="Aqua" />
            <Setter Property="Control.Margin" Value="10" />
            <Setter Property="TextBox.Height" Value="50" />
            <EventSetter Event="Click" Handler="bt1_Click" />
        </Style>
    </Application.Resources>
</Application>
```

ResourceDictionary

Fournit une implémentation de table de hachage/dictionnaire qui contient des ressources WPF utilisées par les composants et d'autres éléments d'une application de WPF.

ResourceDictionary

-ResourceDictionary1.xaml

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <!-- Les entrées du dictionnaire de ressources sont définies ici. -->

  <Style TargetType="TextBlock">
            <Setter Property="HorizontalAlignment" Value="Center" />
            <Setter Property="FontFamily" Value="Comic Sans MS"/>
            <Setter Property="FontSize" Value="13"/>
        </Style>
    <Style x:Key="s3">
        <Setter Property="Button.Background" Value="Green" />
    </Style>
</ResourceDictionary>
```

ResourceDictionary

-La collection *MergedDictionaries* pour ajouter les *ResourceDictionary* auxquels on souhaite avoir accès dans nos projets:

```
<Application.Resources>
    <ResourceDictionary>
        <ResourceDictionary.MergedDictionaries>
            <ResourceDictionary Source="ResourceDictionary1.xaml"/>
        </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
</Application.Resources>
```

-Pour pouvoir appliquer le style , il suffit ensuite de l'appeler:

```
<Button Content="Button" HorizontalAlignment="Left" Style="{StaticResource s3}" />
```

Triggers / Actions

Les Triggers

Un trigger est un déclencheur qui peut assigner des propriétés ou déclencher des actions quand une propriété prend une valeur quelconque.

- Ses Types:

-Property-triggers

-Multi-trigger

-Data-triggers

-Event-triggers

-Timer-trigger

Property-triggers

Les property-triggers sont des triggers qui ne vont monitorer qu'une propriété et changer la valeur de la propriété cible du trigger lorsque la valeur de la propriété monitorée va se changer.

-Les éléments de Tigger

- Property: Il faut lui affecter la propriété à surveiller (Ex. la propriété IsMouseOver du bouton).
- Value: la valeur que la propriété surveillée doit prendre pour déclencher le trigger.
- Setters : Contient une collection de setter à exécuter.

Property-triggers

-Exemple : Nous avons un bouton et nous voudrons que lorsque la souris passe au-dessus de celui-ci, la couleur de fond du bouton devient rouge

```
<Window.Resources>
    <Style x:Key="forExemple">
        <Style.Triggers>
            <Trigger Property="Button.IsMouseOver" Value="True">
                <Setter Property="Button.Background" Value="Red"/>
            </Trigger>
        </Style.Triggers>
    </Style>
</Window.Resources>
```

Property-triggers

- Pour l'appeler

```
<Button Content="Button" HorizontalAlignment="Left"  Width="135"
        Style="{StaticResource forExemple}"/>
```

Property-triggers

-Remarque:

Si on veut appliquer le trigger sur tous les objets d'un type quelconque, on utilise la propriété TargetType sans spécifier la clé.

```
<Style TargetType="{x:Type Button}">
    <Style.Triggers>
        <Trigger Property="Button.IsMouseOver" Value="True">
            <Setter Property="Button.Background" Value="Red"/>
        </Trigger>
    </Style.Triggers>
</Style>
```

Multi-Trigger

-Les multi-triggers sont comparables aux property-triggers à l'exception qu'ils monitorent plusieurs propriétés en même temps.

-Le trigger déclenchera le(s) setter(s) lorsque toutes les propriétés surveillées auront la valeur attendue

MultiTrigger

Exemple: Changer la couleur de Foreground si le curseur de la souris est dessus et qu'en plus il a le focus (il est sélectionné).

```
<Window.Resources>
<Style x:Key="forExemple">
    <Style.Triggers>
        <MultiTrigger>
            <MultiTrigger.Conditions>
                    <Condition Property="Button.IsMouseOver" Value="True"/>
                    <Condition Property="Button.IsFocused" Value="True"/>
            </MultiTrigger.Conditions>
            <MultiTrigger.Setters>
                   <Setter Property="Button.Foreground" Value="Red"/>
           </MultiTrigger.Setters>
        </MultiTrigger>
     </Style.Triggers>
    </Style>
</Window.Resources>
```

DataTrigger

-Les Data-trigger sont un type de trigger qui va surveiller la valeur d'une propriété liée .

-Exemple: On va faire en sorte que lorsque l'on tape trigger dans le textbox, la couleur d'arrière-plan du label se change en rouge

DataTrigger

-Exemple:

```
<Window.Resources>
    <Style x:Key="forExemple">
        <Style.Triggers>
            <DataTrigger Binding="{Binding ElementName=X , Path=Text}" Value="trigger">

                <Setter Property="Label.Background" Value="Red"/>

            </DataTrigger>
        </Style.Triggers>
    </Style>
</Window.Resources>
```

```
<TextBox HorizontalAlignment="Left" Name="X" Height="23" Margin="26,37,0,0"
         TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top" Width="120"/>
```

MultiDataTrigger

Ce type permet d'utiliser, comme pour le multi-triggers, plusieurs Multi-triggers. Ils s'emploient exactement de la même manière que les multi-triggers.

Exemple:

Changer la couleur de contenu de textBox ,si on coche les deux CheckBox

MultiDataTrigger

-Exemple:

```
        <Style.Triggers>
            <MultiDataTrigger>
                <MultiDataTrigger.Conditions>
                    <Condition Binding="{Binding ElementName=a, Path=IsChecked}" Value="True" />
                    <Condition Binding="{Binding ElementName=b, Path=IsChecked}" Value="True" />
                </MultiDataTrigger.Conditions>
                <MultiDataTrigger.Setters>
                <Setter Property="TextBox.Text" Value="Verified" />
                <Setter Property="TextBox.Foreground" Value="Green" />
                </MultiDataTrigger.Setters>
            </MultiDataTrigger>
        </Style.Triggers>
    </Style>
```

```
<CheckBox Content="CheckBox1" Name="a" HorizontalAlignment="Left" Margin="26,198,0,0" VerticalAlignment="Top"/>
<CheckBox Content="CheckBox2" Name="b" HorizontalAlignment="Left" Margin="126,198,0,0" VerticalAlignment="Top"/>
```

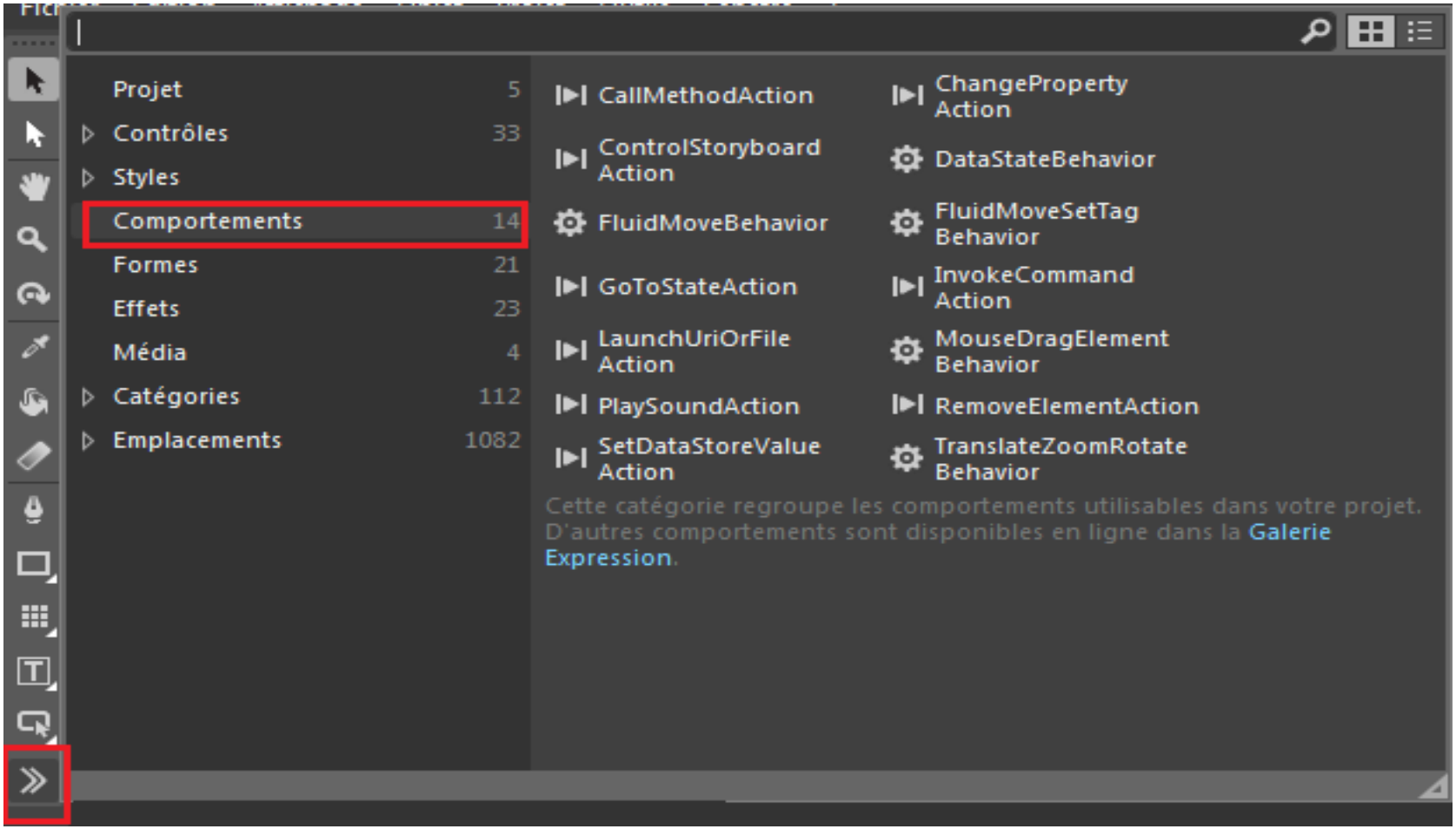
Les actions

-Action est une tache quelconque(fermer le programme;modifier un contenu,jouer un son…) qui ne s'exécute que à travers un trigger

-Remarque: Le DataTrigger et les triggers que nous allons voir par la suite permettent l'exécution des actions

Les actions

- En Blend: Onglet « Composants »→ Liste des Comportements



Les actions

-Exemple: Changer le contenu d'un TextBox si un checkBox est coché

```
<CheckBox x:Name="checkBox" Content="CheckBox" HorizontalAlignment="Left" Margin="22.785,23.285,0,0"
VerticalAlignment="Top"/>
```

```
<TextBox HorizontalAlignment="Left" Name="t1" Height="23" Margin="254.214,23.285,0,0"
TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top"       Width="120">
    <i:Interaction.Triggers>
      <ei:DataTrigger Binding="{Binding IsChecked, ElementName=checkBox}" Value="True">
            <ei:ChangePropertyAction PropertyName="Text" >
              <ei:ChangePropertyAction.Value>verifie</ei:ChangePropertyAction.Value>
            </ei:ChangePropertyAction>
      </ei:DataTrigger>
    </i:Interaction.Triggers>
</TextBox>
```

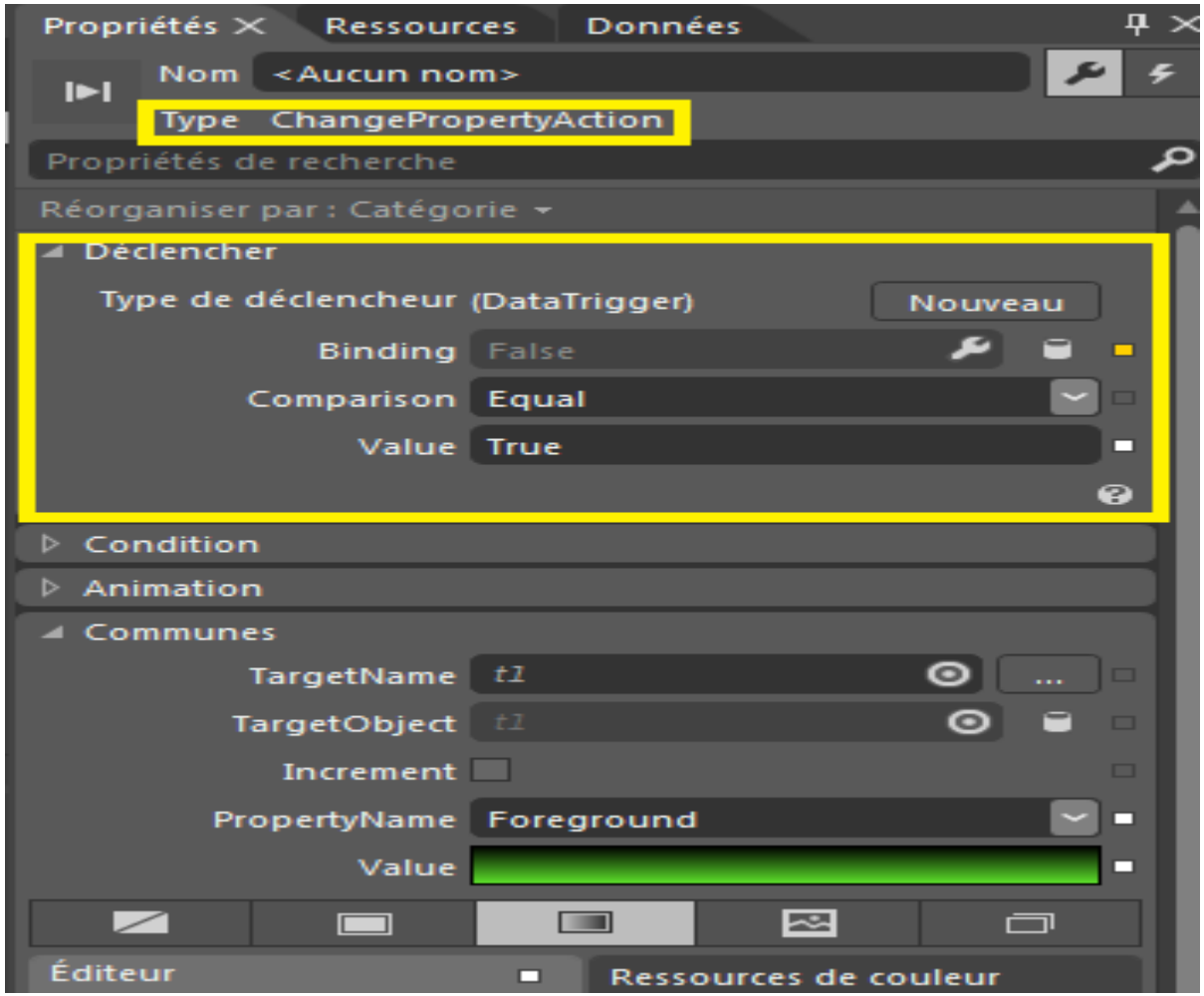
Les actions

-Exemple: Changer le contenu d'un TextBox si un checkBox est coché

```
xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
xmlns:ei="http://schemas.microsoft.com/expression/2010/interactions"
```

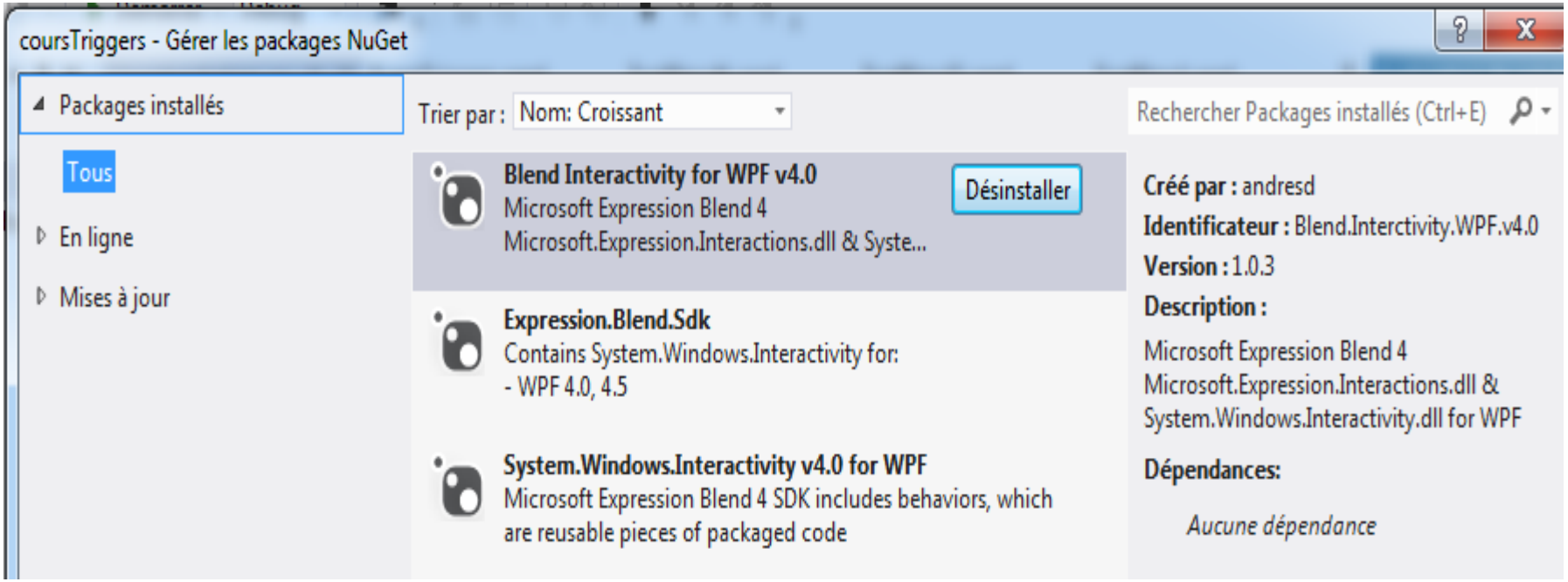
Les actions

-Si on travaille en Blend:



Les actions

-Si on travaille en Visual studio:



Event-triggers

-Là où les autres types de triggers monitorent la valeur d'une propriété, l'event-trigger lui va surveiller le déclenchement d'un évènement (event).

-De plus ce type ne contient pas une collection de setter. Il a, à la place, une collection d'Actions.

-Exemple: Jouer un son lorsqu'un bouton est cliqué :

Event-triggers

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"
    xmlns:ei="http://schemas.microsoft.com/expression/2010/interactions"
    x:Class="BlendCours.EventTrigger"
    x:Name="Window"
    Title="EventTrigger"
    Width="640" Height="480">

    <Grid x:Name="LayoutRoot">
        <Button Content="Button" HorizontalAlignment="Left" Margin="97.997,54.699,0,0"
            VerticalAlignment="Top"   Width="75">
            <i:Interaction.Triggers>
                <i:EventTrigger EventName="Click">
                    <ei:PlaySoundAction Source="cheval.wav"/>
                </i:EventTrigger>
            </i:Interaction.Triggers>
        </Button>
    </Grid>
</Window>
```

Timer-triggers

-Déclenche une action qui est basé sur une minuterie

- Ses paramétres:

-*MillisecondsPerTick :* nombre de millisecondes séparant chaque déclenchement (ex : 2000 pour 2 secondes)

- *TotalTicks :* par défaut « infinite », indiquer le nombre de fois à exécuter (exemple : 1 pour une seule fois)

Timer-triggers

-Exemple: Lancer un audio après 3 seconde de démarrage de l'application

```
<Grid x:Name="LayoutRoot">

    <i:Interaction.Triggers>
      <ei:TimerTrigger MillisecondsPerTick="3000">
          <ei:PlaySoundAction Source="cheval.wav"/>
      </ei:TimerTrigger>
  </i:Interaction.Triggers>


</Grid>
```

Key-triggers

-Déclenché lorsque la touche (ou combinaison de touches) est pressée

- *FiredOn :* à indiquer uniquement si « KeyUp »
- *Key :* la touche du clavier
- *Modifiers :* à indiquer si on fait une combinaison de touche (ex « Alt » + « 1 »)

Key-triggers

Exemple: jouer un son lorsque la touche A+ctrl est pressée

```
<Grid x:Name="LayoutRoot">

    <i:Interaction.Triggers>

      <ei:KeyTrigger Key="A" Modifiers="Ctrl">
          <ei:PlaySoundAction Source="chat.wav" Volume="1"/>
      </ei:KeyTrigger>
  </i:Interaction.Triggers>

</Grid>
```

Action personnalisée

- En Blend: Fichier→Nouvel élément→Action→ok

```
public class Action1 : TriggerAction<DependencyObject>
{
    public Action1()
    {   // Insérez le code requis sur la création de l'objet à
        //partir de ce point.
    }

    protected override void Invoke(object o)
    {   // Insérez le code qui définit ce que fera l'Action une fois
        //déclenchée/invoquée.
    }
}
```

Action personnalisée

-Pour appeler une nouvelle action définie

1. Ajouter l'emplacement de la classe dans les références

```
xmlns:local="clr-namespace:coursTriggers"
```

2. Ajouter le nom de l'action

```
<i:Interaction.Triggers>
    <ei:KeyTrigger Key="B">
        <local:Action1/>
    </ei:KeyTrigger>
</i:Interaction.Triggers>
```

Animation

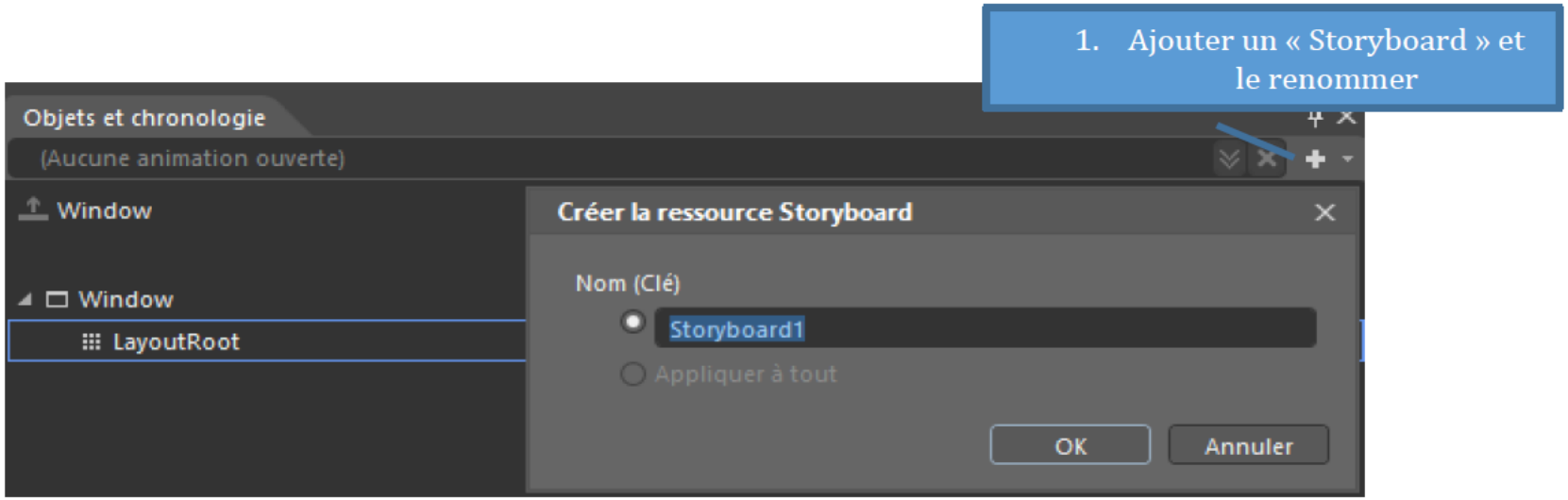
Animation

- Pour effectuer une animation,on utilise l'objet storyboard.
- Storyboard est tout simplement un objet englobant des animations diverses en suivant une timeline .
- Peuvent être utilisées sur toutes les Dependency Properties
- **Type d'animation :**
  - Translation
  - Rotation
  - Agrandissement/Réduction
  - Couleur

Animation

-En blend→ Depuis le panneau « Objets et chronologie »

→Click sur «+ » pour ajouter un nouveau storyboard

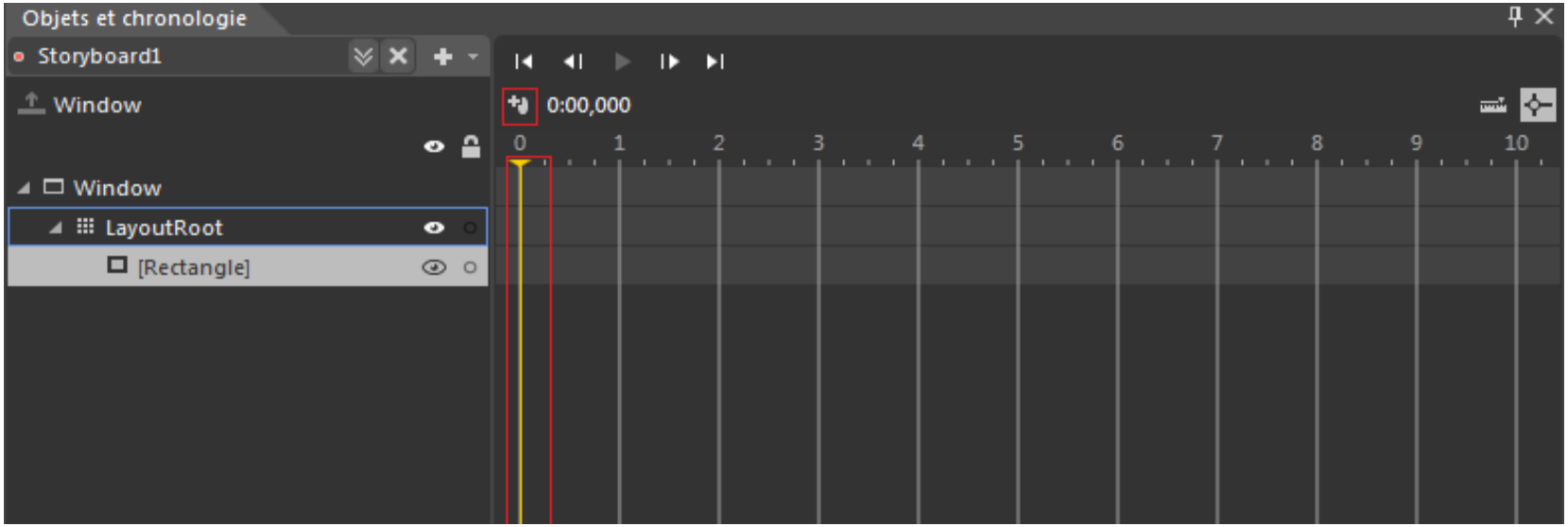


Basculer vers l'espace de travail « Animation » qui est plus indiqué pour créer les animations (Menu « Fenêtre »)

Animation

- l'enregistrement est actif

- La chronologie



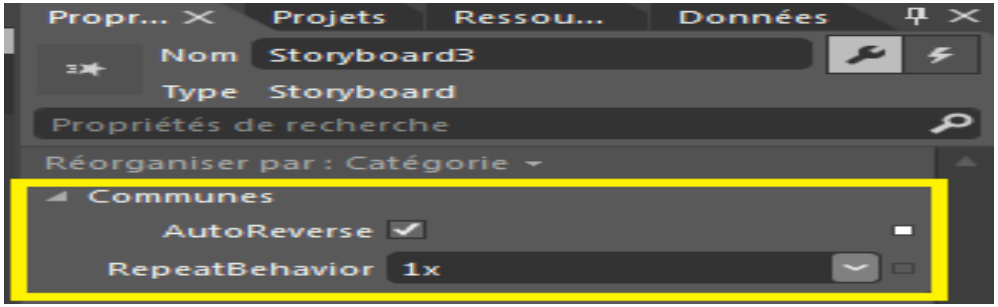
Animation

-Pour effectuer une animation:

1.Sélectionner les éléments, puis placer le curseur sur la timeline la où effectuer « un changement » (changement de propriété, ou déplacement de l'élément)

2. Effectuer l'animation souhaitée.

Remarque:on a la possibilité de faire un effet retour dans les propriétés de storyboard,cocher AutoReverse



Activation de l'Animation

-L'animation est activée par un Trigger

-Par défaut :activer lors de démarrage de l'application(Windows Load)

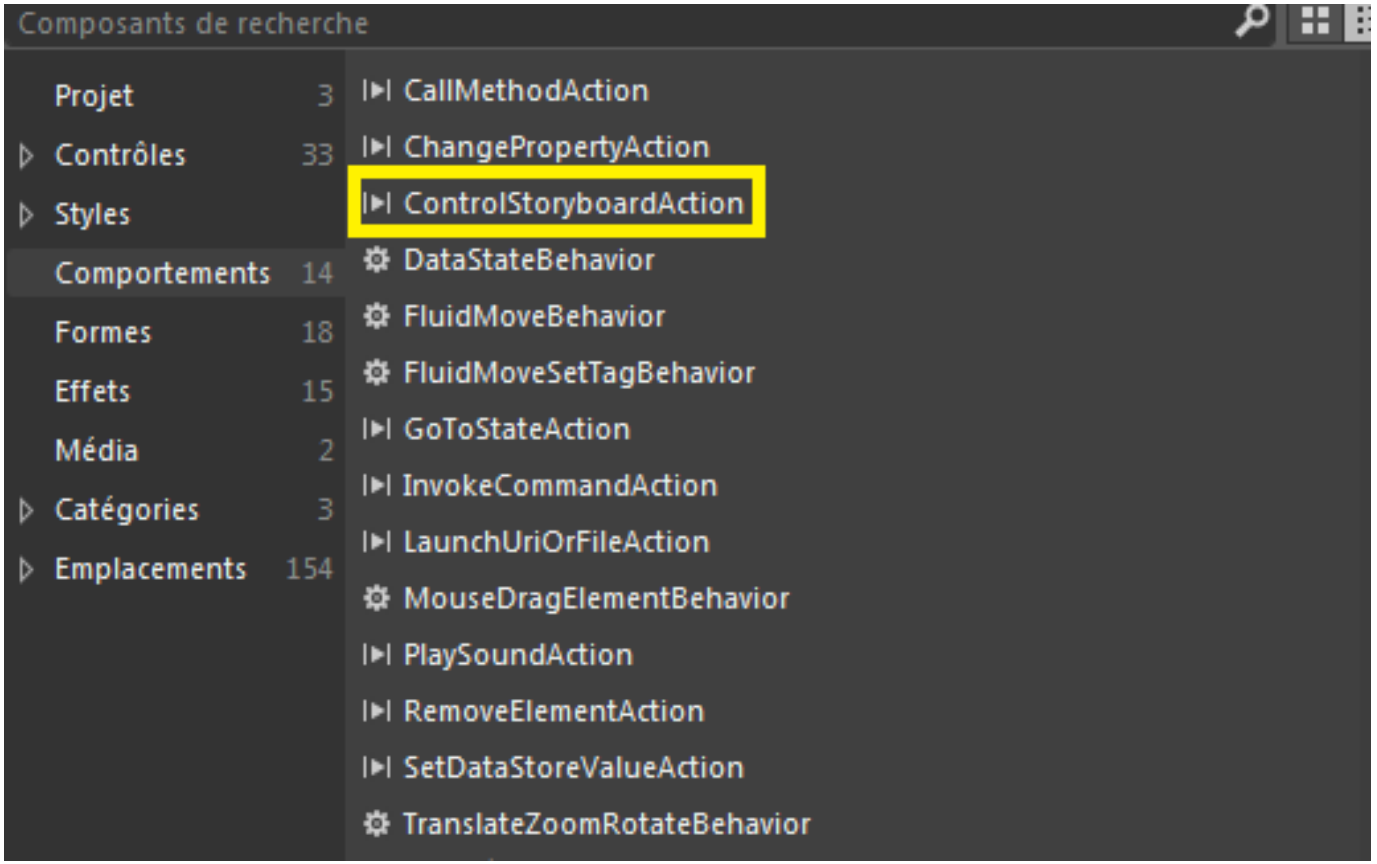
```
<Window.Triggers>
    <EventTrigger RoutedEvent="FrameworkElement.Loaded">
        <BeginStoryboard Storyboard="{StaticResource Storyboard1}"/>
    </EventTrigger>
</Window.Triggers>
```

-Affecter après le click sur un boutton

```
<Window.Triggers>
    <EventTrigger RoutedEvent="ButtonBase.Click" SourceName="button">
                <BeginStoryboard  Storyboard="{StaticResource Storyboard3}"/>
    </EventTrigger>
</Window.Triggers>
```

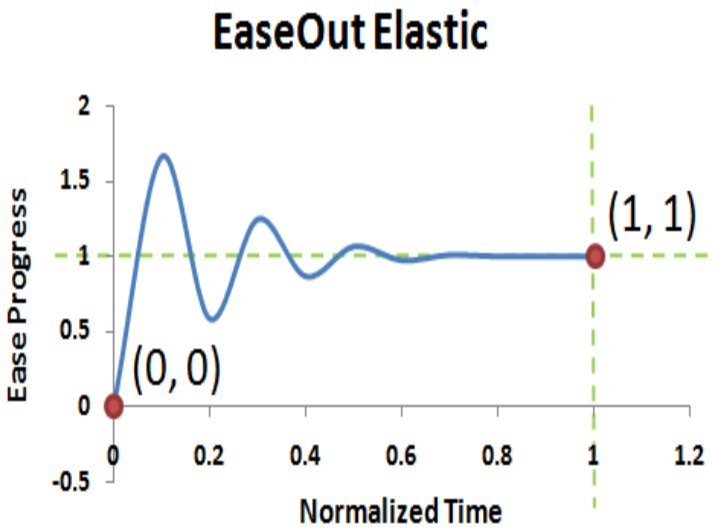
Activation de l'Animation

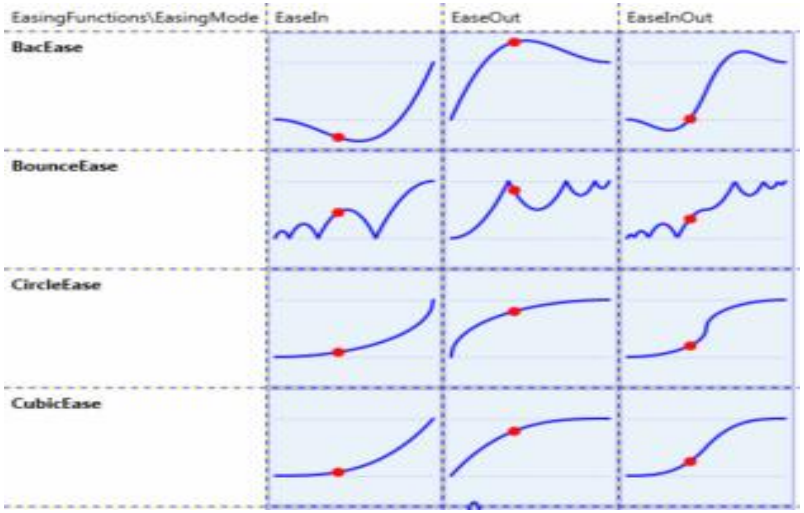
-L'animation est activée aussi par l'action ControlStoryboardAction



Fréquence de l'Animation

-Fréquence d'animation signifie la vitesse de l'animation,appelé aussi Easing



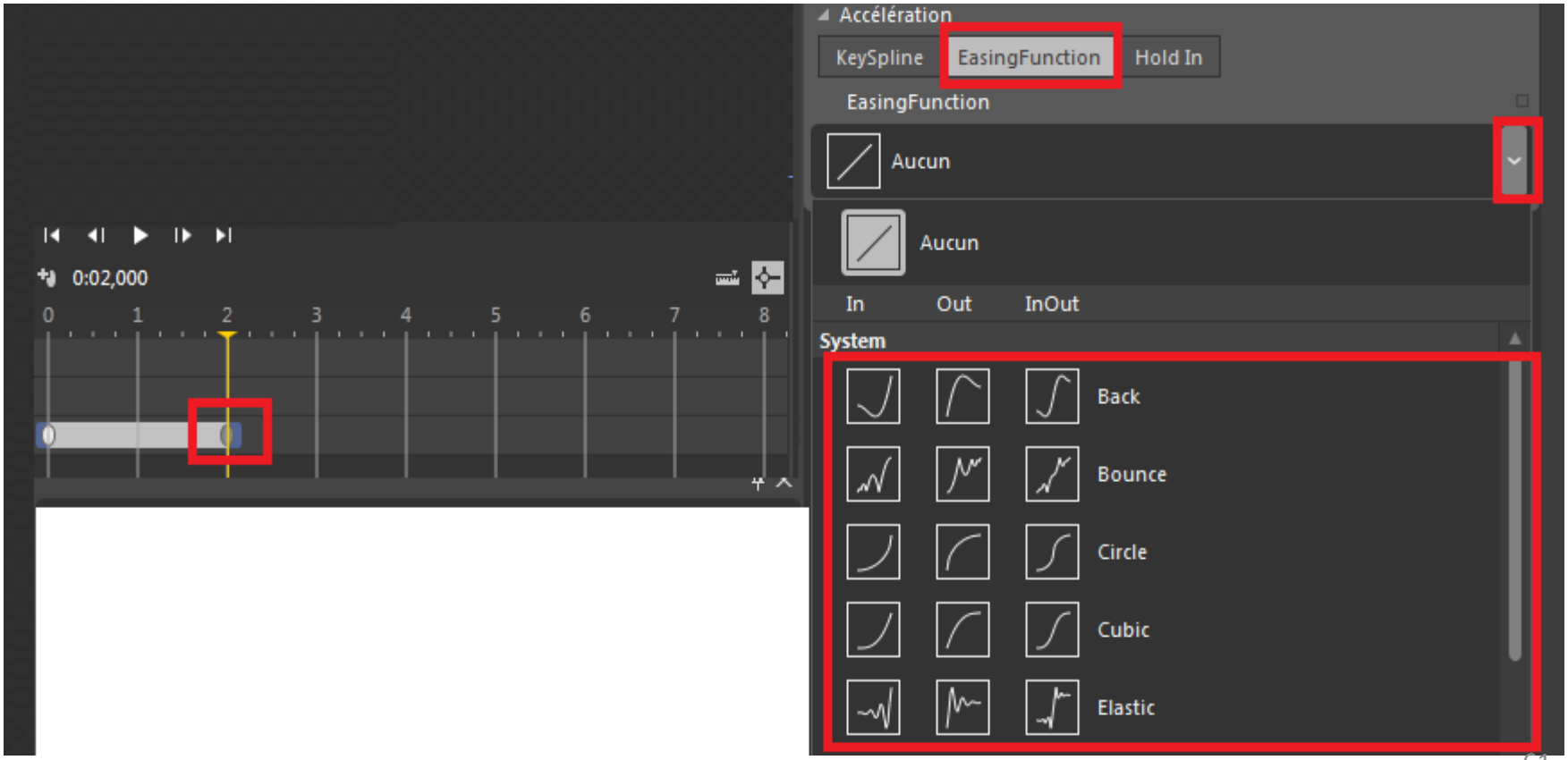


-Fonctions d'accélération qui vous permettent d'appliquer des formules mathématiques personnalisées à vos animations.

Fréquence de l'Animation

-Pour accéder aux propriétés de easing:

En blend→aller au storyboard→cliquer sur les points blancs



Fréquence de l'Animation

-Hold in : déplacer l'objet d'une position vers une autre sans montrer le chemin d'animation.

Trajectoire de l'Animation

-Définir un trajectoire de l'animation c.à.d. le chemin par lequel l'animation va passer

-créer un trajectoire(via la plume par exemple)→Click droit sur le trajectoire→Tracé→convertir en chemin →sélectionner l'objet qui va passer par le trajectoire