MahApps

MahApps est un projet OpenSource qui permet de faire de **belles interfaces** au style « Metro » avec **Wpf**

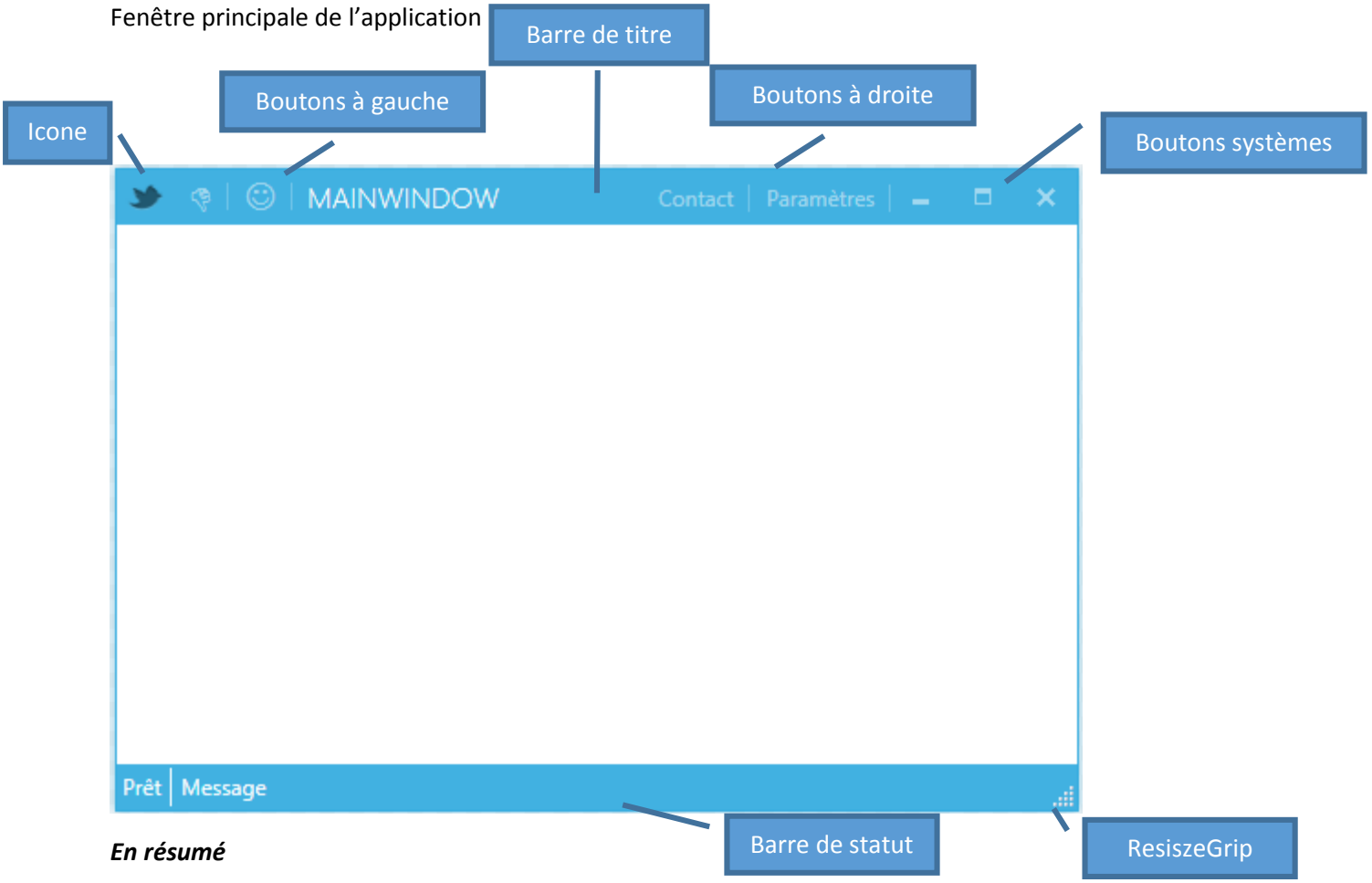
Site, Quick-start, Sources sur Github, exemples d'applications

1. Installation

```
PM> install-package MahApps.Metro
```

2. MainWindow

Fenêtre principale de l'application



En résumé

Code pour une fenêtre de base : ResizeGrip, Couleur de bordure (GlowBrush ou BorderBrush) et icône.

```
<controls:MetroWindow …
        xmlns:controls="http://metro.mahapps.com/winfx/xaml/controls"
        NonActiveGlowBrush="Red"
        ResizeMode="CanResizeWithGrip"
        GlowBrush="{DynamicResource AccentColorBrush}"
        WindowStartupLocation="CenterScreen"
        Icon="/Images/Twitter.png">
    <Grid>

    </Grid>
</controls:MetroWindow>
```

Changer la classe de base de la fenêtre

```
public partial class MainWindow : MetroWindow
{
```

Dans le **détail …**

a. Changer l'apparence de la fenêtre

1. « MainWindow » : Ajouter le namespace et changer la balise de la fenêtre

```
<controls:MetroWindow x:Class="MahAppsDemo.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:controls="http://metro.mahapps.com/winfx/xaml/controls"
        Title="MainWindow" Height="350" Width="525">
    <Grid>

    </Grid>
</controls:MetroWindow>
```

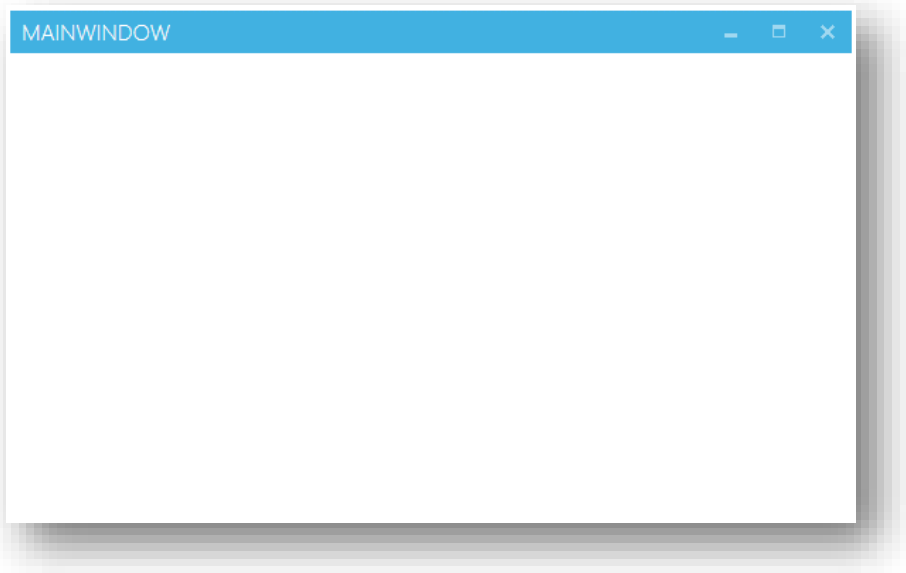
2. Code-behind : Changer la classe base de la fenêtre

```
public partial class MainWindow : MetroWindow
{
```

3. « App » : Ajouter les ressources

```
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Colors.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Blue.xaml" />
                <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseLight.xaml
" />
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
```

On obtient la fenêtre de base



a. Cacher la barre de titre

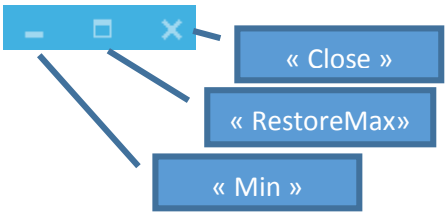
```
<controls:MetroWindow …
        ShowTitleBar="False">

</controls:MetroWindow>
```

b. Désactiver la mise en majuscules du titre de la fenêtre

Ajouter

```
TitleCaps="False"
```

c. Cacher les boutons system




```
<controls:MetroWindow …
        ShowMinButton="False"
        ShowMaxRestoreButton="False"
        ShowCloseButton="False">
</controls:MetroWindow>
```

d. ResizeMode with « Grip »

```
ResizeMode="CanResizeWithGrip"
```

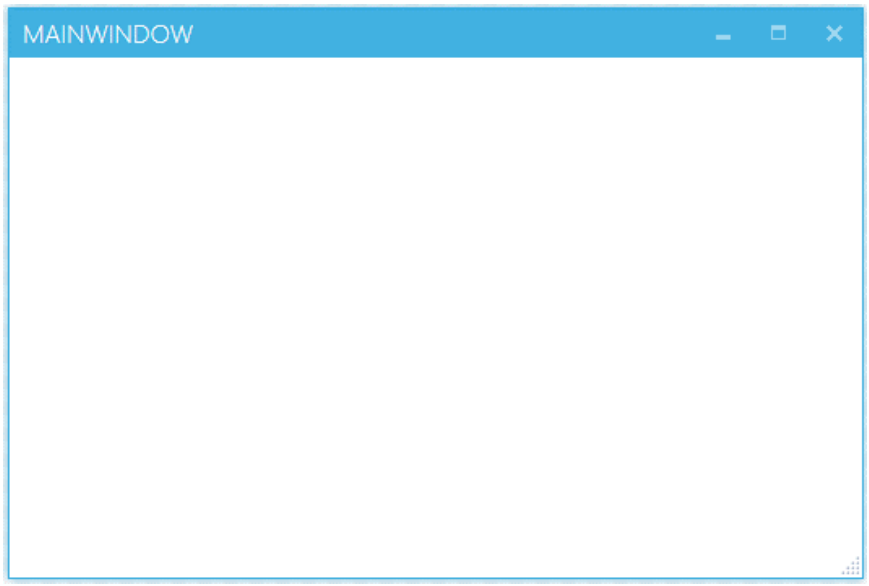
Une « poignée » est ajoutée en bas à droite permettant de redimensionner la fenêtre

e. Changer la couleur de bordure de la


```
BorderBrush="{DynamicResource AccentColorBrush}"
```

Ou si on veut un effet de « Glow »

```
GlowBrush="{DynamicResource AccentColorBrush}"
```



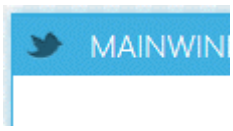
On peut de plus régler la largeur de la bordure avec « BorderThickness »

```
BorderThickness="5"
```

Changer la couleur des bordures et barre de titre à l'état « **non actif** » (perte de focus)

```
NonActiveBorderBrush="Red"
NonActiveGlowBrush="Red"
NonActiveWindowTitleBrush="Red"
```

f. Changer l'icône de la fenêtre



(Taille de l'image ici 48x48) (*.ico, *.png, etc.)

```
Icon="/Images/Twitter.png"
```

Avoir une icône (dans la barre de tache) et une autre pour la barre de titre de la fenêtre

```
<controls:MetroWindow
        Icon="/Images/Twitter.ico">
    <controls:MetroWindow.IconTemplate>
        <DataTemplate>
            <Image Source="/Images/Twitter.png" />
        </DataTemplate>
    </controls:MetroWindow.IconTemplate>
</controls:MetroWindow>
```

Affichée dans la barre de tâches

Template pour l'image affichée dans la barre de titre de la fenêtre

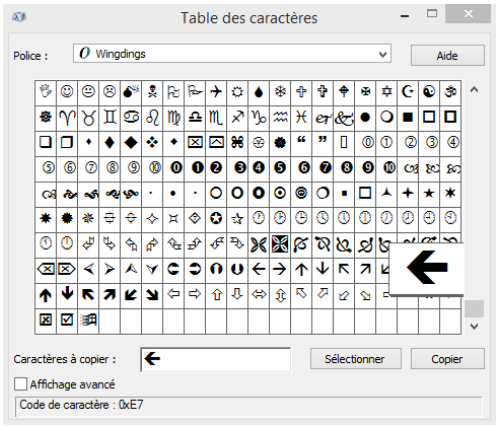
g. Ajout de boutons à la barre de titre

```
<controls:MetroWindow …>

    <controls:MetroWindow.LeftWindowCommands>
        <controls:WindowCommands>
            <Button FontSize="20" FontFamily="Wingdings">D</Button>
            <Button FontSize="20" FontFamily="Wingdings">J</Button>
        </controls:WindowCommands>
    </controls:MetroWindow.LeftWindowCommands>

    <controls:MetroWindow.RightWindowCommands>
        <controls:WindowCommands>
            <Button Content="Contact"></Button>
            <Button Content="Paramètres"></Button>
        </controls:WindowCommands>
    </controls:MetroWindow.RightWindowCommands>

</controls:MetroWindow>
```

Boutons à gauche

Boutons à doite

Utiliser la table de caractères avec la « font » Wingdings



h. Status bar

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="25"/>
    </Grid.RowDefinitions>

    <StatusBar Grid.Row="1">
        <StatusBarItem>Prêt</StatusBarItem>
        <Separator Style="{StaticResource MetroStatusBarSeparator}"></Separator>
        <StatusBarItem>Message</StatusBarItem>
    </StatusBar>
</Grid>
```

3.Contrôles

Des démos sont disponibles avec les [sources sur Github]

a. Boutons

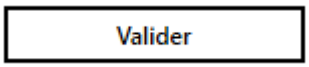
« **Default button** »

Un style est appliqué par défaut

Désactiver la mise en majuscules

```
<Button Content="Valider" controls:ButtonHelper.PreserveTextCase="True"></Button>
```

« Square button »




```
Style="{DynamicResource SquareButtonStyle}"
```


```
Style="{DynamicResource AccentedSquareButtonStyle}"
```

« Circle button »


```
<Button Content="Envoyer" Height="60" Width="60"
        Style="{DynamicResource MetroCircleButtonStyle}"></Button>
```

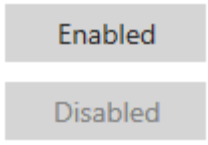
Avec image

```
<Button  Height="60" Width="60" Style="{DynamicResource MetroCircleButtonStyle}">
                <Button.Content>
                    <Image Source="/images/Twitter.png" />
                </Button.Content>
</Button>
```

Ou tout autre contenu (rectangle, …)

```
<Button Width="50" Height="50" Style="{DynamicResource MetroCircleButtonStyle}">
                <Button.Resources>
                    <Canvas x:Key="appbar_city" Width="76" Height="76" Clip="F1 M
0,0L 76,0L 76,76L 0,76L 0,0">
                        <Path Width="38" Height="38.7916" Canvas.Left="19"
Canvas.Top="18.2084" Stretch="Fill" Fill="{DynamicResource BlackBrush}" Data="F1 M
44.3333,30.0833L 57,30.0833L 57,57L 44.3333,57L 44.3333,30.0833 Z M 46.3125,35.2292L
46.3125,38L 49.0833,38L 49.0833,35.2292L 46.3125,35.2292 Z M 52.25,35.2292L
52.25,38L 55.0208,38L 55.0208,35.2292L 52.25,35.2292 Z M 46.3125,39.9792L
46.3125,42.75L 49.0833,42.75L 49.0833,39.9792L 46.3125,39.9792 Z M 52.25,39.9792L
52.25,42.75L 55.0208,42.75L 55.0208,39.9792L 52.25,39.9792 Z M 46.3125,44.7292L
46.3125,47.5L 49.0833,47.5L 49.0833,44.7292L 46.3125,44.7292 Z M 52.25,44.7292L
52.25,47.5L 55.0208,47.5L 55.0208,44.7292L 52.25,44.7292 Z M 46.3125,49.4792L
46.3125,52.25L 49.0833,52.25L 49.0833,49.4792L 46.3125,49.4792 Z M 52.25,49.4792L
52.25,52.25L 55.0208,52.25L 55.0208,49.4792L 52.25,49.4792 Z M 23.75,25.3333L
25.3333,22.1667L 26.9167,22.1667L 26.9167,18.2084L 28.5,18.2084L 28.5,22.1667L
31.6667,22.1667L 31.6667,18.2084L 33.25,18.2084L 33.25,22.1667L 34.8333,22.1667L
36.4167,25.3333L 36.4167,34.8334L 38.7917,34.8333L 41.1667,37.2083L 41.1667,57L
19,57L 19,37.2083L 21.375,34.8333L 23.75,34.8334L 23.75,25.3333 Z M 25.7291,27.3125L
25.7291,30.0834L 28.1041,30.0834L 28.1041,27.3125L 25.7291,27.3125 Z M
32.0625,27.3125L 32.0625,30.0834L 34.4375,30.0834L 34.4375,27.3125L 32.0625,27.3125
Z M 25.7291,32.0625L 25.7291,34.8334L 28.1041,34.8334L 28.1041,32.0625L
25.7291,32.0625 Z M 32.0625,32.0625L 32.0625,34.8334L 34.4375,34.8334L
34.4375,32.0625L 32.0625,32.0625 Z M 30.875,39.9792L 28.8958,39.9792L 28.8958,42.75L
30.875,42.75L 30.875,39.9792 Z M 24.5416,39.9792L 24.5416,42.75L 26.9166,42.75L
26.9166,39.9792L 24.5416,39.9792 Z M 36.0208,39.9792L 33.25,39.9792L 33.25,42.75L
36.0208,42.75L 36.0208,39.9792 Z M 30.875,44.7292L 28.8958,44.7292L 28.8958,47.5L
30.875,47.5L 30.875,44.7292 Z M 26.9166,44.7292L 24.5416,44.7292L 24.5416,47.5L
26.9166,47.5L 26.9166,44.7292 Z M 36.0208,44.7292L 33.25,44.7292L 33.25,47.5L
36.0208,47.5L 36.0208,44.7292 Z M 30.875,49.4792L 28.8958,49.4792L 28.8958,52.25L
30.875,52.25L 30.875,49.4792 Z M 26.9166,49.4792L 24.5416,49.4792L 24.5417,52.25L
26.9167,52.25L 26.9166,49.4792 Z M 36.0208,49.4792L 33.25,49.4792L 33.25,52.25L
36.0208,52.25L 36.0208,49.4792 Z "/>
                    </Canvas>
                </Button.Resources>
                <Rectangle Width="20" Height="20"
                           Fill="{Binding Path=Foreground,
RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type Button}}}">
                    <Rectangle.OpacityMask>
                        <VisualBrush Stretch="Fill"
                                     Visual="{DynamicResource appbar_city}" />
                    </Rectangle.OpacityMask>
                </Rectangle>
</Button>
```

« Flat button »

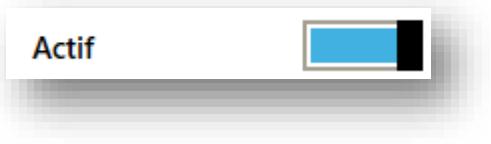



```
<StackPanel>
    <StackPanel.Resources>
       <ResourceDictionary>
           <ResourceDictionary.MergedDictionaries>
               <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/FlatButton.xaml" />
           </ResourceDictionary.MergedDictionaries>
       </ResourceDictionary>
    </StackPanel.Resources>
    <Button Width="100"
                   Margin="0, 10, 0, 0"
                   Content="Enabled" />
    <Button Width="100"
                   Margin="0, 10, 0, 0"
                   Content="Disabled"
                   IsEnabled="False" />
</StackPanel>
```

Requiert

b. Toggle switch

Peut remplacer par exemple une case à cocher (checkbox)



2 labels (un pour chaque état du bouton)

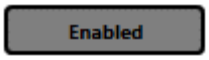
```
<controls:ToggleSwitch Width="200"
                       IsChecked="True"
                       OnLabel="Actif"
                       OffLabel="Inactif" />
```

Pour ne pas avoir de label laisser la chaine « vide »

Customiser le style

```
<Style TargetType="{x:Type controls:ToggleSwitch}" x:Key="CustomMetroToggleSwitch"
                       BasedOn="{StaticResource {x:Type controls:ToggleSwitch}}">
       <Setter Property="SwitchForeground" Value="Red" />
       <Setter Property="OnLabel" Value="True" />
       <Setter Property="OffLabel" Value="False" />
</Style>
```

« Toggle button »



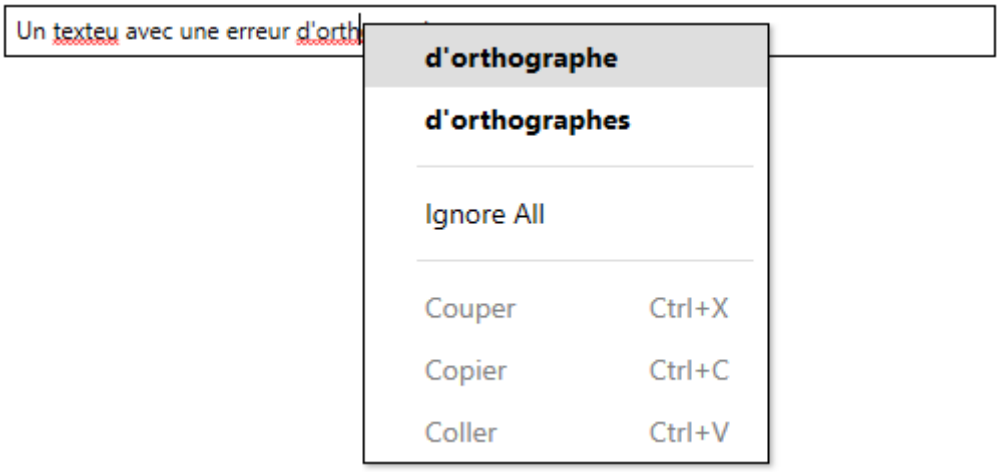

```
<ToggleButton Width="100" Content="Enabled" />
```

c. TextBox

Correction orthographique

```
<TextBox Text="Un texteu avec une erreur d'orthographeu"
         controls:TextBoxHelper.IsSpellCheckContextMenuEnabled="True" />
```

(Idem pour RichTextBox)



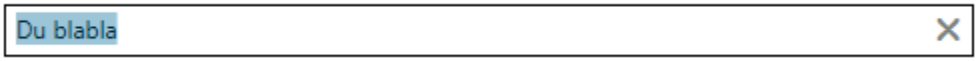
Placeholder

```
<TextBox controls:TextBoxHelper.Watermark="Entrez votre nom" />
```

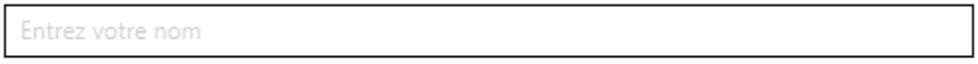
Autre exemple

```
<TextBox controls:TextBoxHelper.Watermark="Entrez votre nom"
         controls:TextBoxHelper.UseFloatingWatermark="True"
         controls:TextBoxHelper.IsWaitingForData="True" />
```

« Clear TextBox »



…




```
<TextBox controls:TextBoxHelper.ButtonCommand="{Binding TextBoxButtonCmd,
Mode=OneWay}" Text="Du blabla">
                <TextBox.Style>
```

```
                    <Style TargetType="TextBox" BasedOn="{StaticResource
MetroTextBox}">
                        <Setter Property="controls:TextBoxHelper.ClearTextButton"
Value="True"></Setter>
                        <Style.Triggers>
                            <Trigger Property="controls:TextBoxHelper.HasText"
Value="False">
                                <Setter
Property="controls:TextBoxHelper.ClearTextButton" Value="False" />
                                <Setter Property="controls:TextBoxHelper.Watermark"
Value="Entrez votre nom" />
                            </Trigger>
                            <Trigger Property="controls:TextBoxHelper.HasText"
Value="True">
                                <Setter
Property="controls:TextBoxHelper.ClearTextButton" Value="True" />
                            </Trigger>
                        </Style.Triggers>
                    </Style>
                </TextBox.Style>
</TextBox>
```

« Search box »


```
<TextBox controls:TextBoxHelper.Watermark="Rechercher"
         controls:TextBoxHelper.ButtonCommand="{Binding
TextBoxButtonCmdWithParameter, Mode=OneWay}"
         controls:TextBoxHelper.ButtonCommandParameter="{Binding ElementName=test2,
Path=Text}"
         Style="{DynamicResource SearchMetroTextBox}" />
```

Custom icon


```
<TextBox controls:TextBoxHelper.Watermark="Custom..."
         controls:TextBoxHelper.ButtonContent="s"
         controls:TextBoxHelper.ButtonCommand="{Binding TextBoxButtonCmd,
Mode=OneWay}"
          Style="{DynamicResource MetroButtonTextBox}" />
```

Autre exemple

```
<TextBox controls:TextBoxHelper.Watermark="Rechercher..."
         controls:TextBoxHelper.UseFloatingWatermark="True"
         controls:TextBoxHelper.ClearTextButton="True"
         controls:TextBoxHelper.ButtonCommand="{Binding TextBoxButtonCmd,
Mode=OneWay}">
                <TextBox.Style>
                    <Style TargetType="{x:Type TextBox}"
                               BasedOn="{StaticResource SearchMetroTextBox}">
                        <Style.Triggers>
                            <Trigger Property="controls:TextBoxHelper.HasText"
                                           Value="True">
                                <Setter
Property="controls:TextBoxHelper.ButtonTemplate"
                                                Value="{DynamicResource
ChromelessButtonTemplate}" />
                            </Trigger>
                        </Style.Triggers>
                    </Style>
                </TextBox.Style>
</TextBox>
```

« Numeric up down »


```
<controls:NumericUpDown Value="5" TextAlignment="Left" Minimum="0" Maximum="10"
IsReadOnly="True"/>
```

d. Progress ring et progress bar

```
<controls:ProgressRing IsActive="True" Width="40" Height="40" />
```



```
<controls:MetroProgressBar IsIndeterminate="True"
                           Value="{Binding ElementName=horizSlider, Path=Value}"
                           Minimum="0"
                           Maximum="100"
                           Width="200"
                           Foreground="{DynamicResource AccentColorBrush}" />
```

e. Slider

```
<Slider VerticalAlignment="top"
                    HorizontalAlignment="Left"
                    Width="132"
                    TickPlacement="BottomRight"
                    TickFrequency="25"
                    Value="50" />
```

f. ComboBox et ListBox

Auto-complétion


```
<ComboBox Width="200"
                          Margin="0, 10, 0, 0"
                          Style="{DynamicResource VirtualisedMetroComboBox}"
                          controls:TextBoxHelper.Watermark="Entrez un nom"
                          DisplayMemberPath="Name"
                          IsEditable="True"
                          ItemsSource="{Binding People}"
                          MaxDropDownHeight="125"
                          Text="{Binding Name}" />
```

Bindé sur une liste de personnes, on affiche la propriété « Name » de la classe « Person »

Listbox

Style appliqué par défaut

```
<ListBox ItemsSource="{Binding People}"
                         DisplayMemberPath="Name"
                         Margin="1"
                         IsEnabled="False"
                         SelectedIndex="0" />
```

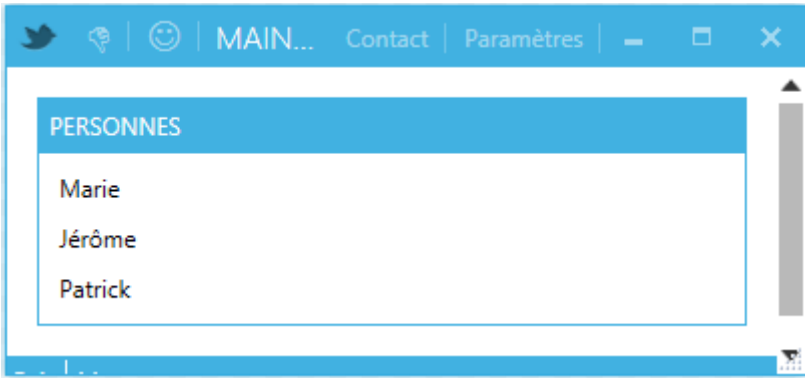
g. GroupBox, Expander


```
<GroupBox Header="Personnes" >
                <ListBox ItemsSource="{Binding People}" />
</GroupBox>
```

Expander

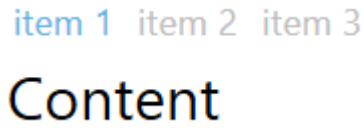

```
<Expander Header="Personnes" >
                <ListBox ItemsSource="{Binding People}" />
</Expander>
```

h. ScrollViewer




```
<ScrollViewer>
 <Grid>
   <!-- etc. -->
 </Grid>
 </ScrollViewer>
```

i. TabControl




```
<TabControl>
     <TabItem Header="item 1" controls:ControlsHelper.HeaderFontSize="18">
                    <TextBlock FontSize="30" Text="Content" />
      </TabItem>
      <TabItem Header="item 2" controls:ControlsHelper.HeaderFontSize="18">
                    <TextBlock FontSize="30" Text="More content" />
       </TabItem>
       <TabItem Header="item 3" controls:ControlsHelper.HeaderFontSize="18">
                    <TextBlock FontSize="30" Text="More more content" />
       </TabItem>
</TabControl>
```

Animé

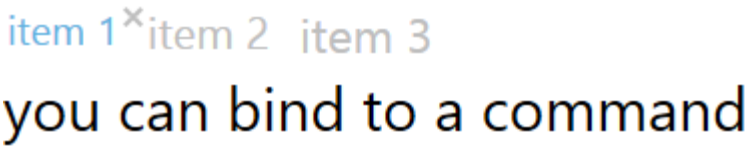
1. Ajouté le dictionnaire en ressources de l'application (ou dans conteneur pour l'exemple)
2. Utiliser MetroTabItem

```
<Grid>
    <Grid.Resources>
         <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.AnimatedTabCo
ntrol.xaml" />
     </Grid.Resources>

       <TabControl>
          <controls:MetroTabItem Header="item 1"
controls:ControlsHelper.HeaderFontSize="18">
                    <TextBlock FontSize="30" Text="Content" />
            </controls:MetroTabItem>
            <controls:MetroTabItem Header="item 2"
controls:ControlsHelper.HeaderFontSize="18">
                    <TextBlock FontSize="30" Text="More content" />
              </controls:MetroTabItem>
              <controls:MetroTabItem Header="item 3"
controls:ControlsHelper.HeaderFontSize="18">
                    <TextBlock FontSize="30" Text="More more content" />
               </controls:MetroTabItem>
        </TabControl>

</Grid>
```

Ou simplement

```
<Grid>
    <Grid.Resources>
         <ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.AnimatedTabCo
ntrol.xaml" />
     </Grid.Resources>

     <TabControl>
                <TabItem Header="item 1">
                    <TextBlock FontSize="30" Text="Content" />
                </TabItem>
                <TabItem Header="item 2">
                    <TextBlock FontSize="30" Text="More content" />
                </TabItem>
                <TabItem Header="item 3">
                    <TextBlock FontSize="30" Text="More more content" />
                </TabItem>
     </TabControl>
</Grid>
```

TabItems pouvant être fermés

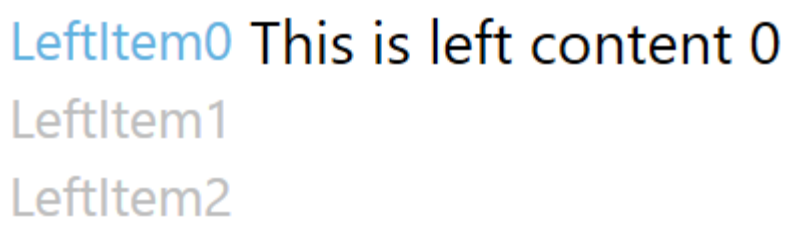


you can bind to a command

On ajoute « CloseButtonEnabled » aux tabitems pouvant être fermés

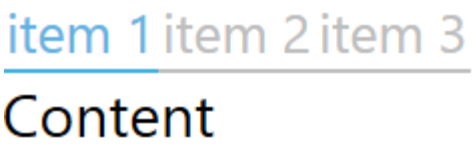
```
     <controls:MetroTabControl
TabItemClosingEvent="MetroTabControl_TabItemClosingEvent">
                <controls:MetroTabItem Header="item 1"

controls:ControlsHelper.HeaderFontSize="20"
                                       CloseButtonEnabled="True"
                                       CloseTabCommand="{Binding
SingleCloseTabCommand}"
                                       CloseTabCommandParameter="{Binding
RelativeSource={RelativeSource Self}, Path=Header}">
                    <TextBlock FontSize="30" Text="you can bind to a command" />
                </controls:MetroTabItem>
                <controls:MetroTabItem Header="item 2"

controls:ControlsHelper.HeaderFontSize="22"
                                       CloseButtonEnabled="True"
                                       CloseTabCommand="{Binding
NeverCloseTabCommand}"
                                       CloseTabCommandParameter="{Binding
RelativeSource={RelativeSource Self}, Path=Header}">
                    <TextBlock FontSize="30" Text="and ensure the command never
closes" />
                </controls:MetroTabItem>
                <controls:MetroTabItem Header="item 3"

controls:ControlsHelper.HeaderFontSize="24"
                                       CloseButtonEnabled="True">
                    <TextBlock FontSize="30" Text="More more content" />
                </controls:MetroTabItem>
            </controls:MetroTabControl>
```

Onglets à gauche (ou droite)




```
<TabControl TabStripPlacement="Left" Height="100">
      <TabItem Header="LeftItem0">
          <TextBlock FontSize="30" Text="This is left content 0" />
       </TabItem>
        <TabItem Header="LeftItem1">
            <TextBlock FontSize="30" Text="This is left content 1" />
         </TabItem>
         <TabItem Header="LeftItem2">
             <TextBlock FontSize="30" Text="This is left content 2" />
          </TabItem>
</TabControl>
```

Ligne et choix de transition



Content

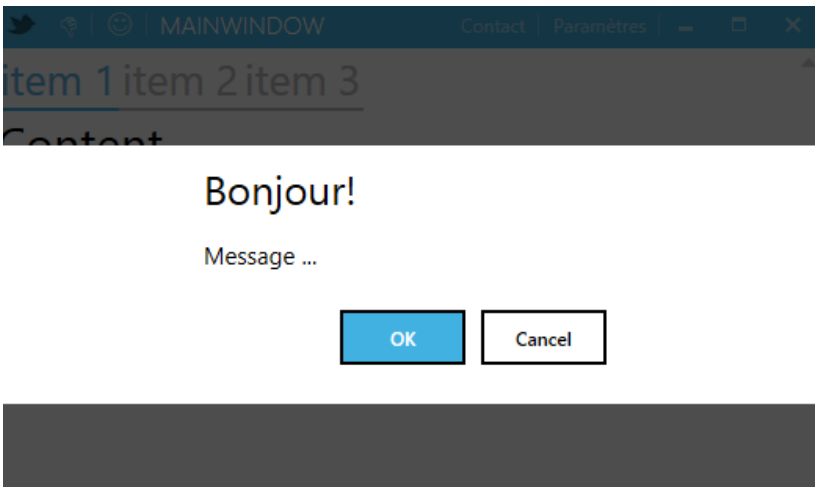
```
<controls:MetroAnimatedTabControl controls:TabControlHelper.IsUnderlined="True"

controls:TabControlHelper.Transition="Up">
                <controls:MetroTabItem Header="item 1">
                    <TextBlock FontSize="30" Text="Content" />
                </controls:MetroTabItem>
                <controls:MetroTabItem Header="item 2">
                    <TextBlock FontSize="30" Text="More content" />
                </controls:MetroTabItem>
                <controls:MetroTabItem Header="item 3">
                    <TextBlock FontSize="30" Text="More more content" />
                </controls:MetroTabItem>
</controls:MetroAnimatedTabControl>
```

Le contenu arrivera par le bas

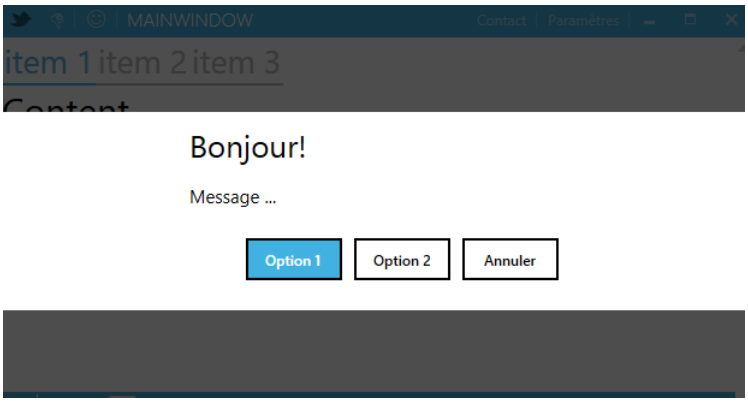
j. MessageDialog

Appelé au click sur un bouton par exemple

```
private async void ShowMessageDialog()
{
    var result = await this.ShowMessageAsync("Bonjour!", "Message ...",
MessageDialogStyle.AffirmativeAndNegative);
     if (result != MessageDialogResult.Affirmative)
     {

     }
}
```

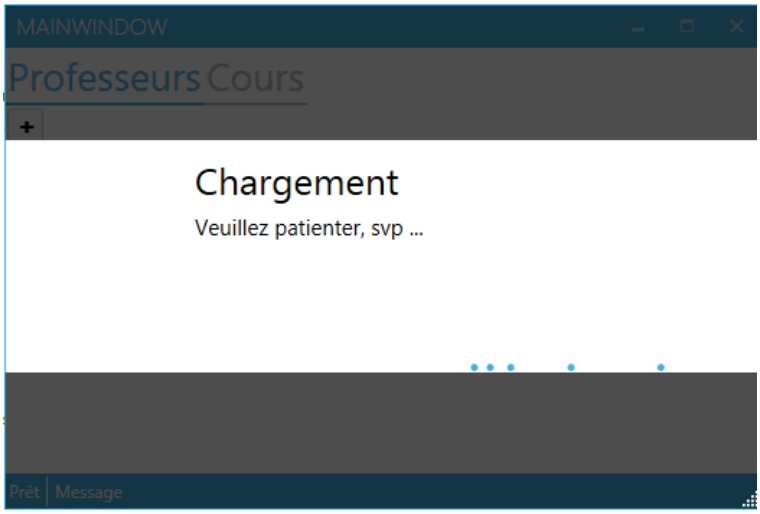


Autre exemple avec settings et boutons supplémentaires




```
private async void ShowMessageDialog()
        {
            var settings = new MetroDialogSettings()
            {
                AffirmativeButtonText = "Option 1 ",
                NegativeButtonText = "Option 2",
                FirstAuxiliaryButtonText = "Annuler",
                ColorScheme = MetroDialogColorScheme.Theme
            };
            MessageDialogResult result = await this.ShowMessageAsync("Bonjour!",
"Message ...",
MessageDialogStyle.AffirmativeAndNegativeAndSingleAuxiliary,settings);
            if (result != MessageDialogResult.FirstAuxiliary)
            {

            }
        }
```

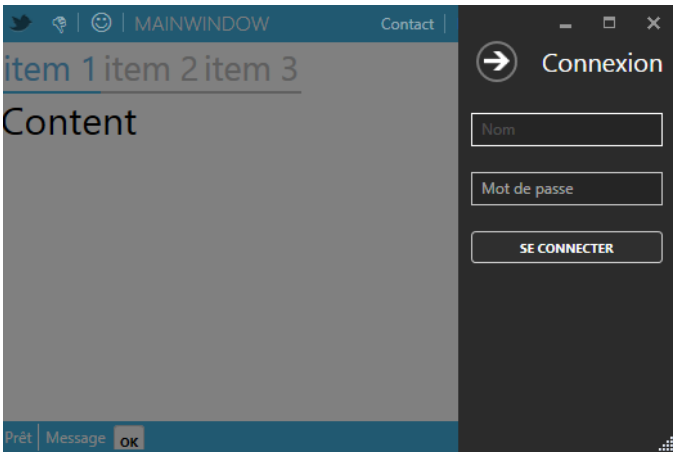
k. Progress dialog



Appelé depuis MainWindow (ou créer un service)

```
private async void ShowProgressAsync()
{
            var controller = await this.ShowProgressAsync("Chargement", "Veuillez
patienter, svp ...");
            await Task.Delay(5000);
            await controller.CloseAsync();
}
```

I. Flyout



Ajouté à MainWindow

```
<controls:MetroWindow.Flyouts>
    <controls:FlyoutsControl>
        <controls:Flyout Position="Right"
                         AreAnimationsEnabled="True"
                         Header="Connexion"
                         IsOpen="{Binding IsOpened}">
            <local:LoginFlyout />
        </controls:Flyout>
    </controls:FlyoutsControl>
</controls:MetroWindow.Flyouts>
```

Un UserControl que l'on crée, affiché quand la propriété « IsOpened » de la source de données est « true »

On ajoute également pour masquer les boutons de la fenêtre principale sous le flyout

```
<controls:MetroWindow …
        RightWindowCommandsOverlayBehavior="Never">
```

La même disponible pour les boutons à gauche de la barre de titre avec « LeftWindowCommandsOverlayBehavior »

4. Personnalisation

a. Accents, thèmes, couleurs

Accents : Red, Green, Blue, Purple, Orange, Lime, Emerald, Teal, Cyan, Cobalt, Indigo, Violet, Pink, Magenta, Crimson, Amber, Yellow, Brown, Olive, Steel, Mauve, Taupe, Sienna

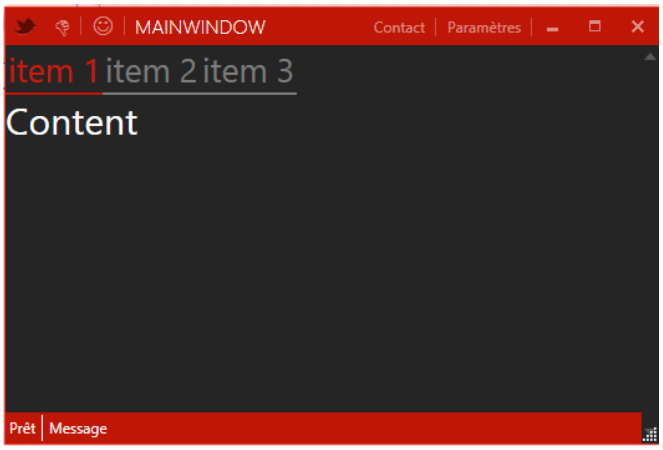
Si on change le dictionnaire par exemple …

```
<ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Red.xaml" />
```

Thèmes : BaseLight, BaseDark

Si on change la ressource liée au thème

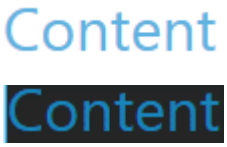
```
<ResourceDictionary
Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseDark.xaml"
/>
```



Utiliser la couleur d'accent

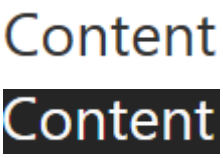
```
<TextBlock FontSize="30" Text="Content"  Foreground="{DynamicResource
AccentColorBrush}"/>
```

Ne change pas selon le thème …

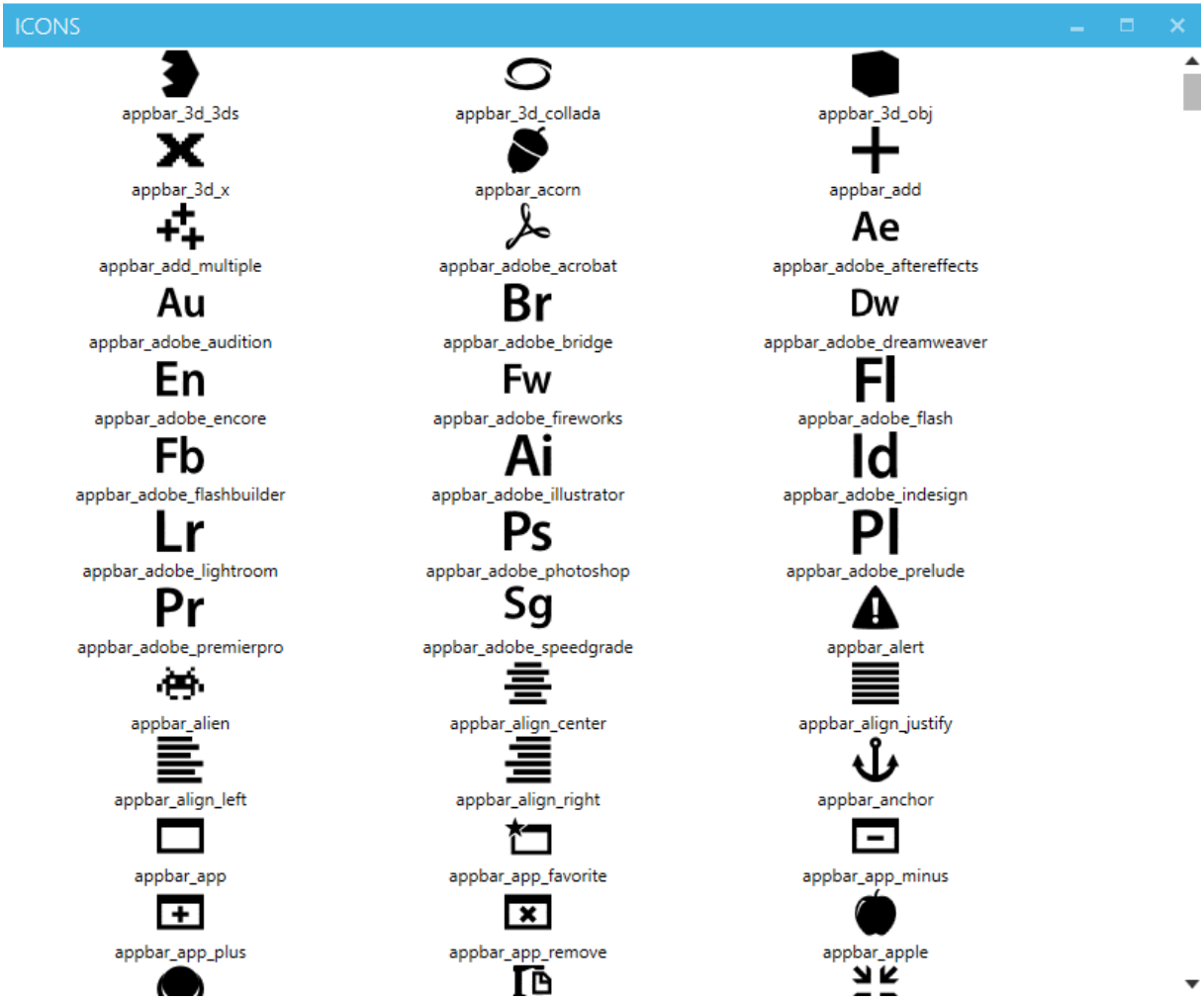


Par contre les couleurs (colors.xaml) changent selon le thème

```
<TextBlock FontSize="30" Text="Content"  Foreground="{DynamicResource GrayBrush1}"/>
```



b. Icons

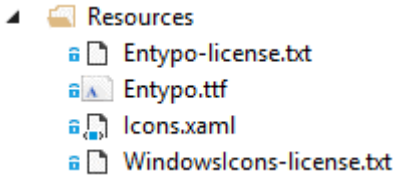


(Projet « MahApps.Metro.Ressources »)

Installation

```
PM> install-package MahApps.Metro.Resources
```

Le dossier Resources est ajouté au projet.



Référencer

```
<ResourceDictionary Source="/Resources/Icons.xaml" />
```

Utilisation

```
<Button Width="50"
                    Height="50"
                    Margin="0, 10, 0, 0"
                    Style="{DynamicResource MetroCircleButtonStyle}">
                        <Rectangle Width="20"
                           Height="20"
                           Fill="{Binding Path=Foreground,
RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type Button}}}">
                            <Rectangle.OpacityMask>
                                <VisualBrush Stretch="Fill" Visual="{DynamicResource
appbar_youtube}" />
                            </Rectangle.OpacityMask>
         </Rectangle>
</Button>
```

http://www.entypo.com/

```
 <Button Width="50" Height="50" FontSize="40" FontFamily="/Resources/#Entypo"
Style="{DynamicResource MetroCircleButtonStyle}">r</Button>
```

http://www.metrostudio.com/