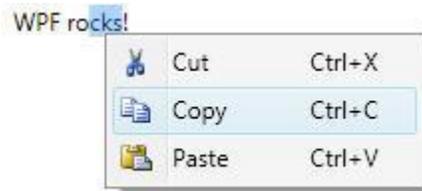


Context Menus in WPF

Context Menus can be defined on any WPF controls by setting the *ContextMenu* property to an instance of a *ContextMenu*. The items of a context menu are normal *MenuItem*s.



```
<RichTextBox>
  <RichTextBox.ContextMenu>
    <ContextMenu>
      <MenuItem Command="Cut">
        <MenuItem.Icon>
          <Image Source="Images/cut.png" />
        </MenuItem.Icon>
      </MenuItem>
      <MenuItem Command="Copy">
        <MenuItem.Icon>
          <Image Source="Images/copy.png" />
        </MenuItem.Icon>
      </MenuItem>
      <MenuItem Command="Paste">
        <MenuItem.Icon>
          <Image Source="Images/paste.png" />
        </MenuItem.Icon>
      </MenuItem>
    </ContextMenu>
  </RichTextBox.ContextMenu>
</RichTextBox>
```

Show ContextMenus on a disabled controls

If you rightclick on a disabled control, no context menu is shown by default. To enable the context menu for disabled controls you can set the *ShowOnDisabled* attached property of the *ContextMenuService* to *True*.

```
<RichTextBox IsEnabled="False" ContextMenuService.ShowOnDisabled="True">
  <RichTextBox.ContextMenu>
    <ContextMenu>
      ...
```

```

        </ContextMenu>
    </RichTextBox.ContextMenu>
</RichTextBox>

```

Merge ContextMenus

If you want to fill a menu with items coming from multiple sources, you can use the *CompositeCollection* to merge multiple collection into one.

```

<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib">
    <Grid Background="Transparent">
        <Grid.Resources>
            <x:Array Type="{x:Type sys:Object}" x:Key="extensions">
                <Separator />
                <MenuItem Header="Extension MenuItem 1" />
                <MenuItem Header="Extension MenuItem 2" />
                <MenuItem Header="Extension MenuItem 3" />
            </x:Array>
        </Grid.Resources>
        <Grid.ContextMenu>
            <ContextMenu>
                <ContextMenu.ItemsSource>
                    <CompositeCollection>
                        <MenuItem Header="Standard MenuItem 1" />
                        <MenuItem Header="Standard MenuItem 2" />
                        <MenuItem Header="Standard MenuItem 3" />
                        <CollectionContainer Collection="{StaticResource
extensions}" />
                    </CompositeCollection>
                </ContextMenu.ItemsSource>
            </ContextMenu>
        </Grid.ContextMenu>
    </Grid>
</Window>

```

How to bind a Command on a ContextMenu within a DataTemplate using MVVM

Since the *Popuup* control has it's separate visual tree, you cannot use *find ancestor* to find the *Grid*. The trick here is to use the *PlacementTarget* property, that contains the element, the *ContextMenu* is aligned to, what is the *Grid* in our case.

But this is only half of the solution. Because of the data template, the *DataContext* is set to a dataitem, and not the view model. So you need another relative source lookup, to find the view model. Trick Nr. 2 is to use the *Tag* property to bind the view model from outside to the grid, which is the *PlacementTarget* used above. And there we are.

```
<DataTemplate>
  <Grid Tag="{Binding DataContext, RelativeSource={RelativeSource
AncestorType={x:Type ListBox}}}">
    <Grid.ContextMenu>
      <ContextMenu DataContext="{Binding Path=PlacementTarget.Tag,
RelativeSource={RelativeSource Self}}">
        <MenuItem Content="Cut" Command="{Binding CutCommand}" />
        <MenuItem Content="Copy" Command="{Binding CopyCommand}" />
        <MenuItem Content="Paste" Command="{Binding PasteCommand}" />
      </ContextMenu>
    </Grid.ContextMenu>
  </Grid>
</DataTemplate>
```

How to open a context menu from code

The following sample shows you how to open a context menu of a control programmatically:

```
private void OpenContextMenu(FrameworkElement element)
{
    if( element.ContextMenu != null )
    {
        element.ContextMenu.PlacementTarget = element;
        element.ContextMenu.IsOpen = true;
    }
}
```