WPF: TabControl series - Part 4: Closeable TabItems

By Olaf Rabbachin 15. February 2010 19:10

Introduction

In the last part of the series, I'd like to present one last extension to the TabControl's TabItems - a close-button.

Overview

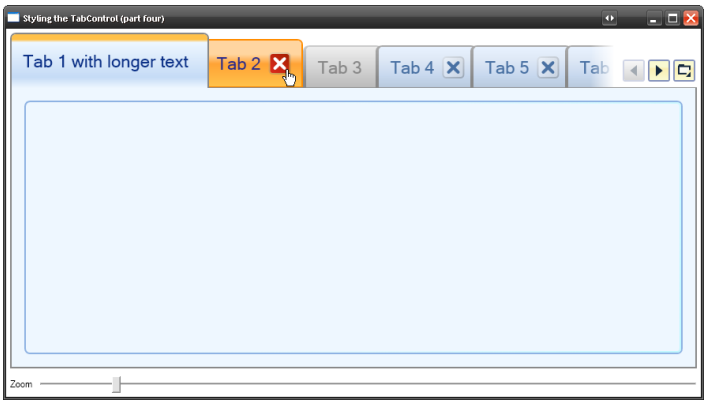
This is the last article in the multi-part series about the WPF TabControl.

Here's the four parts of the series:

- WPF: TabControl Series - Part 1: Colors and Sizes
- WPF: TabControl Series - Part 2: Animating the TabItems
- WPF: TabControl Series - Part 3: Non-wrapping TabItemPanel
- WPF: TabControl Series - Part 4: Closeable TabItems (this article)

Outcome: the result of what's covered in this article

Here's what we'll be left with at the end of this article (click to enlarge):



If you downloaded the sample solution (see the bottom for the link), click the "2. TabItem Close Button" button to show the above window.

Status quo (after *Part Three*)

As noted before, this article is based upon the stuff I introduced in the other parts, hence I'll simply assume that you read and understood what has been discussed there. Please see the other parts in case you find that I am assuming something you don't see discussed here.

Here's where we'll start in this part, that is, what the TabControl and its "sub-controls" looked like at the end of *Part Three*:

If you downloaded the sample solution (see the bottom for the link), click the "1. Base-style (ScrollableTabPanel)" button to show the above window.

Why closing TabItems

Actually, I saw the need for an easy way to close (read "*remove*") tab last year when I was working on a project in which we decided to have a "sort of" MDI application. That is, windows aren't windows but rather UserControls which are then loaded as a TabPage into a TabControl, much like what you see in recent versions of browsers like Firefox and IE (funny enough - in German, pronouncing "IE" could be translated as "yuck"; SCNR).

In the end, what I came up with is what I think is a very versatile approach as it is open to both a code-behind approach or MVVM (which is what was used in the aforementioned project), also allowing for a TabItem-related determination about whether it should even be allowed to close a TabItem or not. More on that later, let's first focus on how we deal with ...

Extending the TabItem-style

As for the style of the Button control that is to be rendered in TabItems, there isn't really anything special. Here's the style/template of the button:

```
<Style x:Key="TabItemCloseButtonStyle" TargetType="{x:Type Button}">
   <Setter Property="SnapsToDevicePixels" Value="false"/>
   <Setter Property="Height" Value="{StaticResource CloseButtonWidthAndHeight}"/>
   <Setter Property="Width" Value="{StaticResource CloseButtonWidthAndHeight}"/>
   <Setter Property="Cursor" Value="Hand"/>
   <Setter Property="Focusable" Value="False"/>
   <Setter Property="OverridesDefaultStyle" Value="true"/>
   <Setter Property="Template">
      <Setter.Value>
         <ControlTemplate TargetType="{x:Type Button}">
            <Border x:Name="ButtonBorder"
                    CornerRadius="2"
                    BorderThickness="1"
                    Background="{StaticResource TabItemCloseButtonNormalBackgroundBrush}"
                    BorderBrush="{StaticResource TabItemCloseButtonNormalBorderBrush}">
               <Grid>
                  <!-- The Path below will render the button's X. -->
                  <Path x:Name="ButtonPath"
                        Margin="2"
                        Data="{StaticResource X_CloseButton}"
                        Stroke="{StaticResource TabItemCloseButtonNormalForegroundBrush}"
                        StrokeThickness="2"
                        StrokeStartLineCap="Round"
                        StrokeEndLineCap="Round"
                        Stretch="Uniform"
                        VerticalAlignment="Center"
                        HorizontalAlignment="Center"/>
                  <!-- We don't really need the ContentPresenter, but what the heck ... -->
                  <ContentPresenter HorizontalAlignment="Center"
                                    VerticalAlignment="Center"/>
               </Grid>
            </Border>
            <ControlTemplate.Triggers>
               <Trigger Property="IsMouseOver" Value="True">
                  <Setter TargetName="ButtonBorder"
                          Property="Background"
                          Value="{StaticResource
                             TabItemCloseButtonHoverBackgroundBrush}" />
                  <Setter TargetName="ButtonPath"
                          Property="Stroke"
                          Value="{StaticResource
                             TabItemCloseButtonHoverForegroundBrush}"/>
               </Trigger>
               <Trigger Property="IsEnabled" Value="false">
                  <Setter Property="Visibility" Value="Collapsed"/>
               </Trigger>
               <Trigger Property="IsPressed" Value="true">
                  <Setter TargetName="ButtonBorder"
                              Property="Background"
                              Value="{StaticResource
                                 TabItemCloseButtonPressedBackgroundBrush}" />
                  <Setter TargetName="ButtonBorder"
                              Property="BorderBrush"
                              Value="{StaticResource
                                 TabItemCloseButtonPressedBorderBrush}" />
                  <Setter TargetName="ButtonPath" Property="Stroke"
                              Value="{StaticResource
                                 TabItemCloseButtonPressedForegroundBrush}"/>
                  <Setter TargetName="ButtonPath"
                          Property="Margin" Value="2.5,2.5,1.5,1.5" />
               </Trigger>
            </ControlTemplate.Triggers>
         </ControlTemplate>
```

- The style includes a Trigger that is applied when the button is down/pressed; here, a Margin shifts the button's content down and to the right. To only have a ***slight*** change (a shift by 1px would be to drastic), the Margin is incremented by 0.5 for top/left and decremented by 0.5 for bottom/right. To make this work, *SnapToDevicePixels* is explicitly set to *False* in the style. While this setter isn't required (*SnapToDevicePixels* is *False* by default), I prefer to explicitly point this out.
- The button's "image" is, again (see the previous parts), a Path. In this case, it's really only two lines plus round start-/end-caps (the latter being defined in the Path, of course):

```
<Geometry x:Key="X_CloseButton">M0,0 L10,10 M0,10 L10,0</Geometry>
```

To actually integrate this button into the TabItem's ControlTemplate only takes a few minor changes. Here's the part of the template that contains the amendments:

```
<Border Name="Border"
        Background="{StaticResource TabItem_BackgroundBrush_Unselected}"
        BorderBrush="{StaticResource TabItem_BorderBrush_Selected}"
        Margin="{StaticResource TabItemMargin_Base}"
        BorderThickness="2,1,1,0"
        CornerRadius="3,3,0,0">
   <Grid>
      <Grid.ColumnDefinitions>
         <!-- Text / TabItem's Caption -->
         <ColumnDefinition/>
         <!-- Close button -->
         <ColumnDefinition/>
      </Grid.ColumnDefinitions>
      <!-- This is where the Content of the TabItem will be rendered. -->
      <ContentPresenter x:Name="ContentSite"
                        VerticalAlignment="Center"
                        HorizontalAlignment="Center"
                        ContentSource="Header"
                        Margin="7,2,12,2"
                        RecognizesAccessKey="True"/>
      <Button x:Name="cmdTabItemCloseButton"
              Style="{StaticResource TabItemCloseButtonStyle}"
              Command="{Binding Path=Content.DataContext.CloseCommand}"
              CommandParameter="{Binding
              RelativeSource={RelativeSource FindAncestor,
              AncestorType={x:Type TabItem}}}"
              Grid.Column="1"
              Margin="-7,5,7,5"/>
   </Grid>
</Border>
```

So, all we really do in the above is to add another ColumnDefinition to the (already existing) Grid control, placing the button into the second column. This doesn't influence i.e. the TabItemMenu, where the textual content is shown, as that is refering to the ContentPresenter's content. The only thing noteable here is the definition of the button's margin, which sort of "replaces" the right margin of the TabItem by applying a negative left margin - this helps to consider the fact that the button may not be visible at all times, in which case the TabItem's Margin should remain as it was before we added the button.

Now, showing a button wasn't much work, but we're talking about a ControlTemplate for the TabControl, so how can we react to a button-click ..?

Enter ICommand

The close-button itself is not worth much if there isn't an easy, versatile and independent way to react to clicks. The "WPF-way" of dealing this is, of course, to use the ICommand interface. For the sake of simplicity and since there's already plenty of tutorials on the web, I won't dig into the specifics of ICommand here. Instead, I've taken over the RelayCommand class, an approach that Josh Smith's introduced in his article on MVVM, published in the MSDN magazine and dropped it into the code behind of the TabControl window. The fact that there's now code-behind in the window is actually neglectable - it could as well be part of a ViewModel instead. This is because, if you look at the binding in the XAML above, the Command associated with the Button control is targetting the DataContext rather than any code-behind, and it also passes a reference to the parent TabItem to the Command - this is all we need in order to utilize the command. Here's the complete code-behind of the window (for the VB-version, please refer to the sample solution - see the bottom of this article for the download link):

```
using System;
using System.Windows;
using System.Windows.Input;
using System.Diagnostics;
using System.Windows.Controls;

namespace TabControlStyle
{
   public partial class TabControl_2_CloseButton : Window
   {
      ///
      /// C'tor
      ///
      public TabControl_2_CloseButton()
      {
         InitializeComponent();
```

```
        private Utils.RelayCommand _cmdCloseCommand;
        ///
        /// Returns a command that closes a TabItem.
        ///
        public ICommand CloseCommand
        {
           get
           {
              if (_cmdCloseCommand == null)
              {
                 _cmdCloseCommand = new Utils.RelayCommand(
                     param => this.CloseTab_Execute(param),
                     param => this.CloseTab_CanExecute(param)
                     );
              }
              return _cmdCloseCommand;
           }
        }

        ///
        /// Called when the command is to be executed.
        ///
        /// <param name="parm">
        /// The TabItem in which the Close-button was clicked.
        ///
        private void CloseTab_Execute(object parm)
        {
           TabItem ti = parm as TabItem;
           if (ti != null)
              tc.Items.Remove(parm);
        }

        ///
        /// Called when the availability of the Close command needs to be determined.
        ///
        /// <param name="parm">
        /// The TabItem for which to determine the availability of the Close-command.
        ///
        private bool CloseTab_CanExecute(object parm)
        {
           //For the sample, the closing of TabItems will only be
           //unavailable for disabled TabItems and the very first TabItem.
           TabItem ti = parm as TabItem;
           if (ti != null && ti != tc.Items[0])
              //We have a valid reference to a TabItem, so return
              //true if the TabItem is enabled.
              return ti.IsEnabled;

           //If no reference to a TabItem could be obtained, the command
           //cannot be executed
           return false;
        }

        #endregion

    }
 }
```

For the sample, the only two conditions that would prevent the Command from being executed (resulting in a disabled button) refer to disabled TabItems and the very first one (this way there'll always be at least one enabled item).

Note that, if you were using the MVVM pattern, the CloseCommand-region would be part of the ViewModel and the window's DataContext (see the constructor) would rather point to that ViewModel.

The last word

This concludes the last part of the TabControl series. This series has been way longer than I originally thought, but I had enough fun with it to play around with a couple of things that weren't part of the original plan ... 🙂

As always, comments are appreciated. Let me know if you have any questions or suggestions for improving the control.

The sample solution

I've created a sample solution that contains everything discussed here, containing one project for each the C# and the VB versions.

Download: TabControlStyle - Part Four.zip (76.83 kb)

Location: SinglePost

Currently rated 4.8 by 16 people

Related posts

WPF: TabControl series - Part 3: Non-wrapping scrollable TabPanel; TabItem DropDown-Menu
How to style the WPF TabControl and the TabItems resp. the TabPanel (part three - non-wrapping scro...

WPF: TabControl Series - Part 1: Colors and Sizes
How to style the WPF TabControl and the TabItems resp. the TabPanel (part one - colors and sizes)

WPF: TabControl series - Part 2: Animating TabItems
How to style the WPF TabControl and the TabItems resp. TabItemPanel (part two - animating TabItems)

Comments

France Benoît H. June 21. 2010 10:57

Hi Olaf,

Great job with this and thanks for sharing.

I have a problem when I try to apply this to my application.
It's working correctly with the TabItems already added to the TabControl in the xaml, but not with the TabItems I'm dynamically adding to it (with a Hyperlink click system).

Here's how I proceed:

TabItem tabtest = new TabItem();
tabControl1.Items.Add(tabtest);

I'm pretty sure I'm missing something but I'm really a beginner so sorry if that's stupid.

Thanks
Benoît

Olaf Rabbachin (website link) June 21. 2010 11:19

Salut Benoît,

actually what you have should be working "as is", the only side effect being that you don't see any header - simply because you don't set it. Try ...

```
TabItem tabtest = new TabItem() { Header = "Test" };
tc.Items.Add(tabtest);
```

... to set it. If you i.e. add a button control to the TC of the article's window and paste the above into its click-event, you will see the dynamically added TabItem appear both in the TabPanel as well as the menu (last item).

À bientôt et bon courage,
Olaf

France Benoît H. June 21. 2010 11:26

Thanks for replying so fast Olaf

In fact I didn't explain my problem very clearly.
I indeed set the header of my generated TabItem as you suggest, here's the whole thing:

```
TabItem tabtest = new TabItem();
Hyperlink h = (Hyperlink)sender;
CasUtilisation c = (CasUtilisation)h.DataContext;
tabtest.DataContext = c;
tabtest.Header = c.Libelle;
```

What I meant by "not working" was about the closeable behavior. The TabItems already added in xaml are closing fine, but not the dynamically added ones. (I imagine I have to attach them the command part but I don't figure out how)

Thanks
Ben

Olaf Rabbachin (website link) June 21. 2010 11:39

Hi Ben,

Cheers,
Olaf

France Benoît H. June 21. 2010 14:01

Hi again,

I had a look but I can't find what's wrong for now.

The general structure of the app is an outlook style TabControl menu on the left (like this www.codeproject.com/KB/WPF/XAML_OutlookBar.aspx) containing hyperlinks, which create TabItems in a second (but basic) TabControl on the main part of the app.
As I'm very new in WPF, it's pretty hard for me to debug, but I'll keep searching.

Thanks again for your help and availability Olaf

Ben

Olaf Rabbachin (website link) June 21. 2010 14:50

Hi Ben,

if you're really only after the closeable tabs, my tutorial - with all the fuzz - might be the wrong choice. I don't have any link at hand that would give you an easier intro to closeable tabs, but I'm sure you'll find bunches of them on the web, meaning intros that only cover the how-to for this specific task.
I know there is closeable tabs in Josh Smith's MVVM-intro, but, again, it's only a secondary area of interest there. In case you still want to peek at the article he wrote for the MSDN mag, here's the link: msdn.microsoft.com/en-us/magazine/dd419663.aspx

Alternatively, if you upload what you have unto this point and post the link here (or e-mail me), I'll take a look, but that might take until later this or even next week (go through the contact-page for my e-mail address), that is, until I find the time (way too limited at present).

Cheers,
Olaf

France Benoît H. June 21. 2010 17:02

Hi Olaf,

I think your tutorial is very convenient with my needs, but it's just that I must be having some details that are messing up the closing function but I'll try to figure that out.

That's funny, at the exact same time you linked Josh Smith's MVVM tutorial I was reading it to know more on MVVMs and indeed I noticed the closeable tabs on his demo application. In fact my app has the exact same system of links opening closeable tabs so I was pretty angry because I could have been looking how he had done to help myself for the last weeks o_o

Anyway I'll dig this and carry on.
I'm not gonna go and loose your time in my stuff, I'll try to make it myself, but I appreciate a lot your help and advices ;)

Ben

Jerome July 16. 2010 18:49

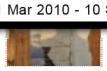
Hi Olaf,

Thanks a lot for this tutorial. It is the best digging-into the TabControl out there, so far as I am able to determine.

A question I have for you is: what could I do to reduce the TabItem's height? I found where I can edit the TabItemPanel's height, but that just cuts off the top of the TabItems.

Thank you,
Jerome

PS> Your commenting system doesn't seem to work in Chrome



eh?

Thanks again,
-Jerome

Olaf Rabbachin (website link) July 16. 2010 19:00

Hi Jerome,

glad you got it yourself - was just about to dig into things here.

As for the commenting system, I know there's a problem which comes from the Captcha plug-in I'm using. Haven't had the time to fix it yet (its code is from somebody else). However, the Captcha is (besides the Aksiment comment checker) the only means to at least dumb *most* of those dumb spammers' comments ... :-(

FWIW - it works if you click the captcha (to create a new code/image) first, enter the code and then enter the comment (not that I'd think this would help much for those that'd like to post a comment ...).

Cheers,
Olaf

Australia Mark F July 18. 2010 13:23

Heya Olaf,

thanks so much for writing this series, you saved me hours of work and created a control that did exactly what I was after. It also answered quite a few WPF questions I've had along the way, notably how you do command notification in templated controls (I'm very new to this stuff).

The only small problem I did have was when I added it to a custom control, for some reason the drop-down menu had a 1-pixel black border around it (only that button, not the scroll left/right buttons). Your xaml is very easy to understand though so it only took me a minute fix it by adding BorderBrush="Transparent" to the <Menu> tag.

Thanks again for all your hard work and for making this code public.

South Africa Dean July 18. 2010 19:25

Hi Olaf
This is great thanks
One small thing, when nesting tab controls the navigation icons dont display in the Parent tab control and all of the tabs in the Child tab have "Close" crosses.
Great work none the less

Tx,
Dean

France Benoît H. July 21. 2010 14:31

Hi again Olaf,

Still using your nice piece of work for my project, and I managed to get out of the little problems I had (the ones I posted about earlier).

I'm encountering the same problem Dean gets, in a UserControl nesting 2 of your tabcontrols, and looking for a solution too.

Thanks again
Ben

Switzerland Antoine November 12. 2010 09:23

Hello Olaf!
Your tutorials are just great!

Have you ever thought giving a class?
I have not found any other tutorial on Customizing Tabcontrol that comes even close to what you did.

Thank you for sharing
Do you work with WPF or Silverlight on SAP?

Salut Antoine,

> Your tutorials are just great!

Thanks!

> Have you ever thought giving a class?

Actually I do give classes from time to time, just not on a regular basis and, up to today, I've never given one for WPF. Given the fact that I've only worked with WPF for less than 1.5 years I guess that makes sense, too.

> Do you work with WPF or Silverlight on SAP?

Well, not sure if I understand you correctly. There are plans to build a stand-alone application that serves as an offline client to SAP's cProjects. This project was started with WinForms, but the prototype we made had been put into the drawer due to the economic situation back in 2008. All our potential clients are coming from the automotive branch and were cutting down their budgets to zero, back then. However, we're convinced that it's just about to come back to life; if that's the case, we'll most probably shift to WPF. However, I've never done anything to integrate WPF into a SAP-/ABAP-app and I doubt it'd be possible.

Cheers,
Olaf

Switzerland Antoine November 12. 2010 10:01

Hi again,
Thank you for your answers.
I don't know SAP well, just heard of it. But I was indeed thinking as client application

Have you ever worked with Blend for your styles? I'm learning it for a month now, it's great!

I'll take your work as an example and try to implement my TabControl Style in Blend.

Greetings from Switzerland,
Herzliche Grüsse

Antoine

Olaf Rabbachin (website link) November 12. 2010 10:08

Hi Antoine,

> Have you ever worked with Blend for your styles?

not all too much - I'm a code-guy and prefer to do things in XAML. So, even though I have it installed, I haven't made all too much use of it. But maybe I should. ;)

Gruß,
Olaf

Belgium Ludwig Stuyck November 13. 2010 17:18

Hi, thanks for this great tutorial! It works great, I integrated it into a Prism application with its own region adapter. There is still 1 issue that has already been reported previously, concerning nested tab controls, where the navigation arrows dissapear. Has anyone solved this yet?
Thanks!

Derrick November 16. 2010 00:26

Also have to add that I have worked through this and it's helped me quite a bit.

But I too am having a problem with the arrows disappearing with nested tabs. If I nest 3 deep and put the 3rd level on a tab that isn't shown right away, the upper tier will show it's arrows till i pick the tab with the lowest tier, then only the lowest tier has arrows.

Not sure yet what's going on, trying to fix it though.

some duplication of things, to be able to set up each button, but it allows for the path to be defined in the style and applied that way. I don't know if that's the correct or best way to do it, but it's what I did.

I tried to use some style inheritance, but it seems that they styles will not inherit triggers, so i just had to define them across all the styles.

Not pretty, dunno how else to do it but my level of experience with this type of programming is very limited.

Olaf Rabbachin (website link) November 18. 2010 09:06

Hi Derrick,

thanks for the comments. I will eventually see into this but, knowing that this will most probably quite a hassle and thus time-consuming, I don't know when that'll be. Also, despite the fact that I spent quite a substantial amount of time with the TabControl and its controls, it might very well be possible that I don't find an adequate and/or straight forward way to deal with this either.

So, it's good to know that you found a work-around - thanks for sharing!

Cheers,
Olaf

Muhammad K. Shehzad February 1. 2011 10:57

Hi Olaf,
Thanks for such an explanatory and helpful article. I used this tab control to extend my custom tab control with very minor changes but all the functionality same.

For the very first tab item, close button do not display, can you help me please...

Once again thanks for your helpful article.

Olaf Rabbachin (website link) February 1. 2011 11:27

Hi Muhammad,

> For the very first tab item, close button do not display, [...]

the close-button is, by convention, neither rendered for the first tab nor for disabled tabs. See the CloseTab_CanExecute method.

Cheers,
Olaf

Muhammad K. Shehzad February 1. 2011 12:20

Thanks a lot man!

Actually I was dived deep just in xaml code which is obviously marvolous so that's why I could not check code behind so keenly... now I am also using code behind to customize it as per my requirements...

Once again thanks!

Muhammad Kanwal Shehzad March 6. 2011 16:47

Hi Olaf,

I was thinking about to extend this control and binding it with commands. I mean can we have a button to close all of the opened TabItems with single click or with some key combinations.

Or at least help me how add custom commands in custom controls.

Regards,

Olaf Rabbachin (website link) March 7. 2011 17:27

Hi Muhammad,

Cheers,
Olaf

Muhammad K. Shehzad March 7. 2011 17:51

Hi Olaf,

Thanks for your response. Yes the exactly the same thing I was looking for. But my main concern would be to add some custom commands in user controls like in this case. I have used custom commands with controls but not with user controls. So please help me to sort this issue out.

Regards,
Muhammad

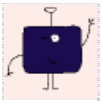
Olaf Rabbachin (website link) March 7. 2011 18:29

Hi again,

I suppose I don't quite understand - you could simply follow the ***exact*** same approach for this as shown in the article ..? I mean, you'd of course have to add another Button to the StackPanel that hosts the line-buttons and the TabItem-menu (see "*Hosting the ScrollButtons*" in part 3 @www.blogs.intuidev.com/.../...yling_PartThree.aspx), but once that has been done, it should basically be the same thing ...

Cheers,
Olaf

Italy Gigi Billa March 8. 2011 13:06



Very great job! I think that the last feature to make it perfect would be the possibility to switch the tabs position, in a vs-like style. Do you think it could be implemented in some way?
Thank you anyway for the great piece of code.

Spain Robert March 31. 2011 18:18

Incredible piece of work! Congratulations Olaf.

By the way, have you (or anyone following this post for that matter) implemented/tried this on Silverlight? I didn't get deep into it, but certainly does not work out of the box in SL.

I'm currently implementing an application that provides views of the same business objects in WPF and SL, and it would be great to have some lights regarding porting this into SL.

Thanks,

--
R.

Olaf Rabbachin (website link) March 31. 2011 21:00

Hi there,

@Gigi: Yes, that of course would be feasible. However, neither do I have the time to do that myself presently, nor do I personally see the need for that in the near future. You'll have to do this yourself, that is. Sorry!

@Robert: No, I haven't tried this with Silverlight. Frankly, I'm pretty much of a noob when it comes to SL, so I can't even tell as to whether there's any potential problems ahead ...

Cheers,
Olaf

Roger April 4. 2011 03:55

Spain Robert April 4. 2011 17:02

Ok Olaf no problem, if I can come up with the SL version I will let you know. Frankly, your version is superb. Thanks for sharing.

France Fabien April 19. 2011 09:33

Hi Olaf,
Thanks for your tutorial, it's great to understand how to customize a control. Not only the tabcontrol indeed.

I found a solution for the Benoît problem about closing tab problem. When Itemsource is bound the datacontext is no longer the windows but the data in the item.
My solution consists in writing command in window header like this :

```
<Window.CommandBindings>
    <CommandBinding Command="{x:Static locTongagePrincipale.CloseCommand}" Executed="CloseTab_Executed"
CanExecute="CloseTab_CanExecute" />
</Window.CommandBindings>
<!-- The button with new command binding -->
<Button x:Name="cmdTabItemCloseButton"
    Style="{StaticResource TabItemCloseButtonStyle}"
    Command="{x:Static locTongagePrincipale.CloseCommand}"
    CommandParameter="{Binding
    RelativeSource={RelativeSource FindAncestor,
    AncestorType={x:Type TabItem}}}"
    Grid.Column="1"
    Margin="-7,5,7,5"/>
```

It's a routed command.

```
      public static RoutedCommand CloseCommand = new RoutedCommand();
      ///
      /// Called when the command is to be executed.
      ///
      ///
      private void CloseTab_Executed(object sender, ExecutedRoutedEventArgs e)
      {
        TabItem ti = e.Parameter as TabItem;
        if (ti != null)
          tc.Items.Remove(ti);
      }

      ///
      /// Called when the availability of the Close command needs to be determined.
      ///
      /// The TabItem for which to determine the availability of the Close-command.
      ///
      private void CloseTab_CanExecute(object sender, CanExecuteRoutedEventArgs e)
      {
          e.canExecute =  true;
        /* I have a trouble with parameter, it's allways enabled  */
      }

      #endregion
```

I hope that's could be useful.

Regards,

Fabien

Olaf Rabbachin (website link) April 19. 2011 16:34

Salut Fabien,

Muhammad May 5. 2011 08:52

Hi Olaf,
Sorry to disturb you again. I have a problem, I need to place a tab control inside this custom TabControl. But my inner TabControl automatically inherits the style of parent TabControl. I want that inner TabControl should not inherit the style of parent TabControl.

Regards,
Muhammad

Olaf Rabbachin (website link) May 5. 2011 11:01

Hi Muhammad,

I'd suggest you simply assign a key to my TabControl's style and its sub-controls and then apply this to the outer TC while leaving your inner TC without a style-assignment.

Cheers,
Olaf

Iraq Muhammad K. Shehzad August 3. 2011 09:55

Dear Olaf, thanks for all of this article. I have used this code in my application. I got a problem, I have placed TabControl inside TabControl. On selection change of outer TabControl, selected TabItem for innner TabControl also get chnaged. Can we handle it, like we can stop an routed event to stop from routing in code behind like

e.Handled=true;

Can we do anything like that?

Thanks in Advance...

Shehzad

Olaf Rabbachin (website link) August 3. 2011 10:02

Hi Muhammad,

I haven't ever used my stuff with nested TCs, so I can't tell. However, I believe Derrick has found a work-around for that - see the comments above.

Cheers,
Olaf

Netherlands Ronald Schaap (website link) September 16. 2011 13:18

To solve the TC inside TC problem it is enough to take the following steps (based on the suggestions made by Derrick):

1. Put the data of the <Geometry> tags in the Data property of the <Path> tags (to draw the arrows and the menubutton)
2. Put the complete concerned <Path> tags for the arrows in the <Content> of the concerned <RepeatButton> tags.
3. Put the complete <Path> tag into the <Content> tag of the <ContentPresenter> of the TabMenuButtonStyle.

There is no more redundancy then there was and it takes only two nimutes to solve the problem.

Olaf Rabbachin (website link) September 16. 2011 13:22

Hi Ronald,

Olaf

Niels Jensen October 8. 2011 03:30

Thank you so much to Olaf for sharing. Great work!

Also, thank you very much to Derrick and Ronald for fixing the nested tab control problem.

It took me a bit longer than two minutes, , to put in the fix suggested by Ronald, so I am listing it here for others to save time. I only moved the three path tags. I don't think it was necessary to change the static resources in the data tags.

Left button:

```
<RepeatButton.Content>
   <Path Margin="4,3"
      Data="{StaticResource ArrowLeft}"
      Stroke="{StaticResource LineButtonBrush}"
      Fill="{StaticResource LineButtonBrush}"
      Stretch="Fill"
      VerticalAlignment="Center"
      HorizontalAlignment="Center"/>
</RepeatButton.Content>
```

Right button:

```
<RepeatButton.Content>
   <Path Margin="4,3"
      Data="{StaticResource ArrowRight}"
      Stroke="{StaticResource LineButtonBrush}"
      Fill="{StaticResource LineButtonBrush}"
      Stretch="Fill"
      VerticalAlignment="Center"
      HorizontalAlignment="Center"/>
</RepeatButton.Content>
```

Tab menu button:

```
<ContentPresenter.Content>
   <Path Margin="2"
      Data="{StaticResource TabMenuButton}"
      Stroke="{StaticResource LineButtonBrush}"
      Fill="{StaticResource TabMenuButtonBrush}"
      Stretch="Fill"
      VerticalAlignment="Center"
      HorizontalAlignment="Center"/>
</ContentPresenter.Content>
```

Thanks again,

Niels

Niels Jensen October 30. 2011 02:38

I just spent a little while searching for the cause of the tab selection problem, when you have more than one tab control in your app, and I believe I found it.

Find the following border definition under the scroll viewer of the tab control style:

```
<Border BorderThickness="1"
      BorderBrush="{StaticResource TabPage_InnerBorderBrushBright}"
      CornerRadius="2"
      Margin="0"
      Padding="2,2,3,3">
   <!--
   This is where the Content of the selected TabPage
   will be rendered.
   -->
   <ContentPresenter x:Name="PART_SelectedContentHost"
                     ContentSource="SelectedContent"
                     Margin="0"/>
</Border>
```

Simply delete the bold and underlined name property above, and the tab selection in one tab control no longer

Once again thank you to Olaf for sharing,

Niels

Olaf Rabbachin (website link) November 3. 2011 17:38

Hi Niels,

I just don't find the time to dig into things here, but your comment makes a lot of sense.

I'll try to do a check & update to part III accordingly so that this potential pitfall gets cleared out. Might take a while though; 2011 seems to become the most time-consuming year in my last decade, no time for even picking my nose! ;)

Many thanks for the heads up!

Cheers,
Olaf

priyakin.org (website link) December 7. 2014 11:09

pingback Pingback from priyakin.org

TabControl with same height tab items? | Zaremski Answers

Comments are closed

Powered by: BlogEngine.NET 1.6.0.0 | Theme: stablestart | Design by: Thomas A. Bosscher