Contents

Overview

With Winword 6.0, Microsoft has introduced a new standard for the Windows user interface, the so called **tabbed dialogs**. This technique allows to present complicated dialogs to the user without placing a great number of controls into one window or splitting the dialog into many sub windows. With the EZ-Tab custom control, the Visual Basic developer is able to implement tabbed dialogs without any additional coding or sub controls.

Related Topics:

EZ-Tab Implementation

EZ-Tab Implementation

EZ-Tab was designed with the objective, that the developer does not have to learn any new technique to set-up a tabbed dialog. Here are some of the EZ-Tab highlights:

- The EZ-Tab control is used like the standard Visual Basic Frame control, i.e. each pane of the dialog is in a separate control.
- All sibling EZ-Tab controls have the same position and size automatically.
- The 'tab-area' is also active in **design mode**, i.e. the developer can switch between the different panes by **double clicking with the right** mouse button.
- The first EZ-Tab instance is initially set-up with the BackColor and TabColor set to the BackColor property of it's parent control. The second and further instances automatically take the color and font properties from their ancestors. Also the 'tab-area' is automatically moved to a reasonable position.
- EZ-Tab has an **Auto 3-D** facility, i.e. depending on the BackColor property, the control is painted plain or with a 3-D appearance. You can use the **style property** to bypass auto 3-D detection.
- EZ-Tab has built-in **focus tracking**, i.e. when a container is shown for the first time, focus is moved to the first control on that container. Any further activation of this pane restores the focus to the control that had the focus the last time.
- EZ-Tab supports the standard Caption and Enabled properties. This includes the ability to use **mnemonic** access keys.
- Multiline Tabs as in the Winword 6.0 Options dialog can be realized without any coding, but simply by setting the **TabGroup** property
- The position of the 'tab-area' can be on top or on the bottom of the control. This is controlled by the **TabPosition** property.
- The **TabStyle** property is used to choose between **chamferred** (like in WinWord's option dialog) and **slanted** (like an Excel spread sheet) tabs.
- The 'tab-area' can be colored different from the tab's background. Setting **UsePalette** to True permits the use of up to 256 solid colors for the **TabColor** property, if this is supported by the video driver.

Components and Installation

The following sections describe the different components of the EZ-Tab package and how to install it on your computer.

Related Topics:

Contents of the distribution disk
The EZ-Tab demo version
Installation

Contents of the distribution disk

The EZ-Tab distribution disk contains the following files (demo versions may contain fewer or additional components, look for a READ.ME file with explanations). For electronically distributed versions, the EZ-Tab components may be compressed into a single file:

Filename	Description
EZTAB.VBX	The custom control file.
EZTAB.LIC	This file, which should reside in the Windows System directory, contains license information. It is needed to compile your EZ-Tab application to an EXE file. If the license file cannot be found or is corrupted, the control is loaded in demonstration mode. The information in the license file can be displayed by selecting the **(About)** property.
EZTAB.HLP	On-line help for EZ-Tab.
PROJECT1.MAK	Sample project file.
FORM*.FRM	Sample form files.
LICENSE.WRI	License agreement in Windows Write format.

You are not allowed to distribute the license information file EZTAB.LIC. Distribution of the control file is free of charge.

The EZ-Tab demo version

If you load EZ-Tab into the Visual Basic environment without a valid license file in your Windows System Directory, the control switches into **demonstration mode**. EZ-Tab is fully functional in the VB development environment, and any program you develop in demonstration mode is compatible with the registered version, but you should be aware of the following:

If you compile a project without a valid license file, the resulting EXE file will fail to load.

Installation

If the installation is not accomplished by an installation program, simply copy the files from the distribution disk to a separate directory on your hard disk. The custom control file EZTAB.VBX should also be copied to your Windows system directory to allow access for compiled programs.

How to use EZ-Tab

EZ-Tab is used like any other Visual Basic custom control. For a quick start open a new project and follow these instructions:

- Use the Add File Command from the File menu to include the control library EZTAB.VBX into your project.
- You can now draw the first EZ-Tab container on a form. You will notice that the control initially has the BackColor property of the form. Depending on the control's background color, the EZ-Tab border is displayed plain or in 3-D. So if you want a 3-D appearance of the tabbed dialog, set the BackColor property of the form and of the EZ-Tab control e.g. to &H8000000F (light gray).
- The easiest technique to add other EZ-Tab instances to the form is to simply double click on the toolbox icon. All subsequent containers take the size and position of the existing controls. Resizing one control automatically adjusts it's siblings.
- Now you can draw the child controls of the different containers in the same way as you do for a Frame control. A right mouse button double click can be used to switch between the panes in design mode.

That's all to implement tabbed dialogs! If you look at the demo application, you will see that the only code there is to demonstrate some of the EZ-Tab capabilities and not to support the dialog behavior. You might want to include in the Form_Load event a call to the ZOrder method of the EZ-Tab control that you want to be on top initially.

*You should not draw a new EZ-Tab instance **within** an existing one as long as you do not want to display sort of a 'hierarchical tabbed dialog'.*

For more information on how to use the EZ-Tab control see Tips and Techniques.

EZ-Tab Properties

EZ-Tab supports the same properties as the Visual Basic Frame control. These can be used as documented in the VB Programmer's Reference. The only specialty is that all siblings controls automatically have the same size and position. There are only few additional properties.

Starting with version 1.25 the Visible property is supported. See Tips and Techniques for an example.

Related Topics:

EZ-Tab non-standard properties

EZ-Tab non-standard properties

Here is a list of EZ-Tab's non standard properties. See Tips and Techniques for more information on how to use these properties.

Related Topics:

(About)
AlignTabs
OnTop
Style
TabLeft, TabWidth
TabPosition
TabStyle
TabColor
TabGroup
UsePalette

(About)

Displays a dialog box with copyright and version information.

AlignTabs

Displays a dialog box to adjust the size and position of the 'tab-area'. The tabs are arranged starting from left to right. You can decide to

- let the TabWidth property unchanged.
- set the TabWidth of all controls to the current maximum value.
- let the Tabs fill the complete border area.

It is recommended that you experiment a little with this property to get a feeling on how it can be used.

OnTop

This property, which is available only at runtime, determines if the corresponding container is currently active (True/False).

Style

Use this property to bypass the automatic 3-D detection. The value of this property is automatically copied to all sibling controls.

- Style 0: automatic 3-D detection (default).
- Style 1: the control is always painted flat.
- Style 2: the control is always painted 3-D.

This property cannot be changed at runtime.

TabLeft, TabWidth

These properties determine the position and width of the 'tab-area'. The height is automatically adjusted to the font size.

TabPosition

Determines the placement of the tab area.

- 0 - Top (default)
- 1 - Bottom

TabStyle

Determines the style of the tab area.

- 0 - Chamferred (default)
- 1 - Slanted

TabColor

This property allows to give the tab area a background color different from the control's pane.

TabGroup

This property allows to group different containers to set-up a Multiline tabbed dialog.

UsePalette

This property allows to use up to 256 solid colors for the TabColor property, if this is supported by the video driver.

EZ-Tab Events and Methods

EZ-Tab supports the same events as the Visual Basic Frame control. You should notice that the Click and DblClick events are only fired, if the mouse button is pressed within the 'control-area'. This is the combination of the control's 'tab-area' and 'pane'. Though EZ-Tab controls appear to be non-rectangular, the 'out-of-bounds' area is part of the control's window. Therefore the other mouse events are also fired, if the mouse is over this region. You can even draw a child window in the out-of-bounds area (but you certainly won't like to do so).

You will also notice that the Click event is only fired, if the control is currently on top, and not when the control is activated. This is intentional and allows to distinguish these situations in the code (The Activate event described below is fired in the latter situation).

Related Topics:

EZ-Tab non-standard events
EZ-Tab methods

EZ-Tab non-standard events

There are only few non-standard events for EZ-Tab. Here is a list of these events.

Related Topics:

Activate, Deactivate
QueryDeactivate

Activate, Deactivate

```
Sub EZTab_Activate ()
Sub EZTab_Deactivate ()
```

These events are fired, when a control is brought to the top or put into the background respectively.

QueryDeactivate

```
Sub EZTab_QueryDeactivate (Cancel As Integer)
```

This event is called before a control is put into the background (either through user action or programatically with the ZOrder method. Setting the Cancel parameter to True aborts that operation and leaves the corresponding container on top.

EZ-Tab methods

There are no non-standard methods for the EZ-Tab control.

The ZOrder method is the preferred way to bring a specific container to the front.

Tips and Techniques

The following sections contain some commonly used techniques used with the EZ-Tab control.

Related Topics:

Bringing a specific control to the front

Switching controls is normally user controlled either by clicking in the tab area or by using shortcuts. If you want to change the active control programatically, e.g. to start with a specific part of the dialog, you can simply use the VB standard **ZOrder** method:

```
EZTab(Index).ZOrder
```

This brings the corresponding control to the front.

Finding the active control

If you want to know, if a specific control is currently in front of its siblings, you have to query the **OnTop** property of that control.

```
If EZTab(Index).OnTop Then
    'Action
End If
```

Inserting a new Tab before an existing one

The common technique to add an EZ-Tab instance to a form is to simply double click on the control bitmap in the Toolbar and let the control take care of its size and position. This method also puts the tab area of the new instance at the end of the control cluster. But under certain circumstances, you might want to insert a new pane with its tab area between two existing controls (or to reorder the current panes). This can be achieved by changing the **TabLeft** property after control creation to a value between that of those controls where the new instance shall be inserted. The **AlignTabs** dialog can then be used to automatically rearrange the complete dialog.

Using control arrays

It is highly recommended to put all EZ-Tab controls, that set-up one tabbed dialog, together to one control array (i.e. to use the same control name). This reduces the code in response to the various EZ-Tab events, because each pane of the dialog can be addressed by an index value.

If you want to load additional EZ-Tab instances at runtime, you have to set the position of each control after it has been loaded. Here is a short code fragment that shows, how a new instance can be set-up:

```
Sub LoadEZTab ()
    Static i As Integer

    i = i + 1 :   If i > 4 Then Exit Sub

    Load EZTab(i)

    EZTab(i).Caption = "TAB &" + Format$(i)
    EZTab(i).TabLeft = i * EZTab(0).TabWidth

    ' Don't forget this !!!
    EZTab(i).Visible = True
    EZTab(i).Enabled = True
Exit Sub
```

Multiline Tabbed Dialogs

If you have to set-up a dialog with many different panes, a single line tab area will result in very small tabs (though you could decide to use overlapping tabs). The way out of this problem is to group panes together to build a Multiline Tabbed Dialog. Here is how this can be done.

Start setting up your EZ-Tab controls as usual. Don't care about the size and position of the tabs. After you have created the instances which will constitute the first 'tab line', create the next container and set its **TabGroup** property to a value different from the default '0'. You should reset the **TabLeft** property of this control to '0' to let the new group start at the left border of the control cluster. Now create the other controls for the second line (eventually create a third or even a fourth line with a different **TabGroup** property). If you are finished use the **AlignTabs** dialog to bring the tab areas to their final position (**Fill Border Area** is the preferred alignment for multiline dialogs).

Putting EZ-Tab controls in a container

Normally, EZ-Tab controls are directly drawn on a form, but there might be situations, where you have to put a tabbed dialog into another container control (e.g. a picture box). In this case you cannot use the double click technique to add a new control instance, but you have to draw each control on the container. Remember to draw outside and not within an existing EZ-Tab control, as this would make the new instance a child of the old one.

Switching between panes in design mode

To switch between different EZ-Tab instances in design mode, you can

- double click with the right mouse button in the tab area of the control you want to make active.
- select a control in the Properties Window, activate the form and use the 'Bring to Front' or 'Send To Back' commands of the VB menu.

Permanent child controls

To reduce code size and resource consumption, you may want to appear some controls on any EZ-Tab instance in a tabbed dialog. You can use the following techniques to achieve this:

- Draw the controls (e.g. OK and Cancel buttons) on **one** EZ-Tab instance, and use the **SetParent** API to move them to the active control .
- Create the controls outside the tabbed dialog, then move them **over** the EZ-Tab controls. Use the **ZOrder** method bring these controls to the front of the dialog.

For both techniques, the **Activate** event is the place to move the child controls.

The ClipControls property

The recommended setting for the ClipControls property is 'True' for the containing form and 'False' for any EZ-Tab instance.

Unsing the Visible Property

To change the Visible property at runtime it is necessary that the container in question is on top of its sibling controls. Here are 2 procedures that can be used to show and hide EZ-Tab controls:

```
Sub HideTab (ctl As EZTab)
    ' do nothing if already hidden
    If Not ctl.Visible Then
        Exit Sub
    End If

    ' bring control to the front
    ctl.ZOrder
    ' ... hide ...
    ctl.Visible = False
    ' and force into background
    ctl.ZOrder 1
End Sub

Sub Showtab (ctl As EZTab, iBringToTop As Integer)
    ' bring control to the front if invisible or especially requested
    If iBringToTop Or ctl.Visible = False Then
        ctl.ZOrder
    End If

    '   That's all if the control is already visible
    If ctl.Visible Then
        Exit Sub
    End If

    ' show the control ...
    ctl.Visible = True

    ' ... and force into background if necessary
    If Not iBringToTop Then
        ctl.ZOrder 1
    End If
End Sub
```