TButtonBar component

Overview

TButtonBar is a single standalone Delphi component which displays and manages a panel of buttons. TButtonBar supports "smart" incremental scrolling in the direction of alignment -- either horizontal or vertical -- when necessary, much like a TScrollBar component. Additionally, each button behaves much like a TSpeedButton, in that it can display a caption and optional glyph. TSpeedBar does not use TSpeedButtons or TScrollBars, though. It is a single TCustomPanel which manages its own drawing and for this reason is very resource-friendly.

<u>Properties</u>

BorderOn	when True, a border is drawn around the entire component area and empty space is shaded.
ButtonCount	Number of buttons within the component.
ButtonProps	Structure for accessing attributes of specific buttons within the component. ButtonProps is a group of attributes which are accessible both at runtime and design-time (at design-time, double-click ButtonProps in the Object Inspector to view the attribute list). Each attribute is described below:

ButtonIndex	Value from 1..ButtonCount which determines the particular button the rest of the properties affect.
Caption	Text displayed at the bottom of the button face.
Enabled	When True, the button can receive focus and clicks. When False, the Caption is grayed-out and an invalid Glyph is displayed (if available).
Glyph	Bitmap displayed on the button face.
Hint	Standard hint property. Currently not supported.
Visible	If False, the button is not displayed. Careful with this!

ButtonHeight	Sets the button height for the component. Note that if the TButtonBar is aligned horizontally, this can also be accomplished by dragging to resize the component's height.
ButtonWidth	Sets the button width for the component. Note that if the TButtonBar is aligned vertically, this can also be accomplished by dragging to resize the component's width.
InitialFocus	If TabStop is True, this is the default button which receives focus initially.
Orientation	The component can be aligned either horizontally or vertically. Further, it can optionally be constrained to an edge of its parent control.

ParentName This property is streamed; however, it provides no functionality with a single TButtonBar.

Sticky If False, buttons pop back up immediately after they are pressed. If True, the TButtonBar acts as a radio control, where a single button remains pressed until another one is selected.

Runtime-Only Properties

ButtonProps : **StickyOverride** This boolean property allows an application to programmatically override the default Sticky behavior of the component for a single button. Valid values are:

`stDefault`	(use component Sticky property)
`stNever`	never sticky
`stAlways`	always sticky

ButtonProps : **Pressed** This boolean property allows buttons to be pressed programmatically. It is accessed the same way as all the other ButtonProps.

Additionally, TButtonBar supports additional standard component properties.

Events

OnClick If assigned, OnClick is called each time a button is activated, or pressed. In addition to a sender object, the OnClick notify event receives the Name (Caption property) and index (ButtonIndex property) of the activated button.

Step-by-Step Tutorial

In this section, I'll walk you throu the process of creating a working application with a TButtonBar speed menu. Before you start, make sure the TButtonBar is installed (see README.WRI for instructions). When you're ready, let's go.....

1. Open a new project (that's New Project, under the File menu). If the Form wizard pops up, tell him you just want a blank form.

create a TButtonBar and some buttons...

2. Click on the Samples tab in your palette. Select the ButtonBar.
3. Click on your form to drop a TButtonBar. It looks like a gray rectangle.
4. Head over to the Object Inspector. Set the value of the ButtonCount property to some number. Use 5. NIow look at your ButtonBar -- 5 buttons!

add a caption and a bitmap to one of the buttons...

5. Double-Click the ButtonProps property. The list of properties drops down.
6. We're going to assign a caption to the middle button, so set ButtonIndex to 2.
7. Set the Caption property to 'Clock'.
8. Double-click on the Glyph property, and choose the ALARM.BMP bitmap from your DELPHI\IMAGES\BUTTONS directory. Say 'Ok.'

align him vertically, for grins...

9. Set the Orientation property to sbVRight, for fun. Look what happened to the TButtonBar -- it rotated and stuck to the right side of the form.

add an event handler...

10. Click on the Events tab in the Object Inspector.
11. Double-click the OnClick event. Delphi creates a method in your form source for handling click events from your TButtonBar.
12. Add this code to the new event:

```
If ButtonIdx = 2 then ShowMessage('Clicked!');
```

12.5 Since we just used the ShowMessage procedure, we need to add Dialogs to the form's Uses clause at the top.

and test it out!

13. Hit F9 to compile and run. Click some of the buttons. Resize the form to get some scroll arrows.

Contacting the Author

TButtonBar was written by Sam Johnson, the original BlueGuy. Sam appreciates all comments and criticisms (well, almost all) and can be reached via email at sam@moai.com.

Cheers!

Moai Technologies is a software group focused on delivering business automation solution for the Internet and the World Wide Web. To find out more about who they are and what they're doing, visit the Moai home page at http://www.moai.com/ or drop a note to feedback@moai.com.