NAME

button − Create and manipulate ’button’ action widgets

SYNOPSIS

button *pathName* ?*options*?

STANDARD OPTIONS

−activebackground	**−font**	**−relief**
−activeforeground	**−foreground**	**−repeatdelay**
−anchor	**−highlightbackground**	**−repeatinterval**
−background	**−highlightcolor**	**−takefocus**
−bitmap	**−highlightthickness**	**−text**
−borderwidth	**−image**	**−textvariable**
−compound	**−justify**	**−underline**
−cursor	**−padx**	**−wraplength**
−disabledforeground	**−pady**	

See the **options** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Command-Line Name: **−command**
Database Name: **command**
Database Class: **Command**

Specifies a Tcl command to associate with the button. This command is typically invoked when mouse button 1 is released over the button window.

Command-Line Name: **−default**
Database Name: **default**
Database Class: **Default**

Specifies one of three states for the default ring: **normal**, **active**, or **disabled**. In active state, the button is drawn with the platform specific appearance for a default button. In normal state, the button is drawn with the platform specific appearance for a non-default button, leaving enough space to draw the default button appearance. The normal and active states will result in buttons of the same size. In disabled state, the button is drawn with the non-default button appearance without leaving space for the default appearance. The disabled state may result in a smaller button than the active state.

Command-Line Name: **−height**
Database Name: **height**
Database Class: **Height**

Specifies a desired height for the button. If an image or bitmap is being displayed in the button then the value is in screen units (i.e. any of the forms acceptable to **Tk_GetPixels**); for text it is in lines of text. If this option is not specified, the button’s desired height is computed from the size of the image or bitmap or text being displayed in it.

Command-Line Name: **−overrelief**
Database Name: **overRelief**
Database Class: **Ov erRelief**

Specifies an alternative relief for the button, to be used when the mouse cursor is over the widget. This option can be used to make toolbar buttons, by configuring **−relief flat −overrelief raised**. If the value of this option is the empty string, then no alternative relief is used when the mouse cursor is over the button. The empty string is the default value.

Command-Line Name: **−state**
Database Name: **state**
Database Class: **State**

Specifies one of three states for the button: **normal**, **active**, or **disabled**. In normal state the button is displayed using the **−foreground** and **−background** options. The active state is typically used when the pointer is over the button. In active state the button is displayed using the **−activeforeground** and **−activebackground** options. Disabled state means that the button should be insensitive: the default bindings will refuse to activate the widget and will ignore mouse button presses. In this state the**−disabledf oreground** and **−background** options determine how the button is displayed.

Command-Line Name: **−width**
Database Name: **width**
Database Class: **W idth**

Specifies a desired width for the button. If an image or bitmap is being displayed in the button then the value is in screen units (i.e. any of the forms acceptable to **Tk_GetPixels**). For a text button (no image or with **−compound none**) then the width specifies how much space in characters to allocate for the text label. If the width is negative then this specifies a minimum width. If this option is not specified, the button's desired width is computed from the size of the image or bitmap or text being displayed in it.

DESCRIPTION

The **button** command creates a new window (given by the *pathName* argument) and makes it into a button widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the button such as its colors, font, text, and initial relief. The **button** command returns its *pathName* argument. At the time this command is invoked, there must not exist a window named *pathName*, but *pathName*'s parent must exist.

A button is a widget that displays a textual string, bitmap or image. If text is displayed, it must all be in a single font, but it can occupy multiple lines on the screen (if it contains newlines or if wrapping occurs because of the **−wraplength** option) and one of the characters may optionally be underlined using the **−underline** option. It can display itself in either of three different ways, according to the **−state** option; it can be made to appear raised, sunken, or flat; and it can be made to flash. When a user invokes the button (by pressing mouse button 1 with the cursor over the button), then the Tcl command specified in the **−command** option is invoked.

WIDGET COMMAND

The **button** command creates a new Tcl command whose name is *pathName*. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg ...*?

Option and the *arg*s determine the exact behavior of the command. The following commands are possible for button widgets:

pathName **cget** *option*

Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **button** command.

pathName **configure** ?*option*? ?*value option value ...*?

Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **button** command.

pathName **flash**
Flash the button. This is accomplished by redisplaying the button several times, alternating between the configured activebackground and background colors. At the end of the flash the button is left in the same normal/active state as when the command was invoked. This command is ignored if the button's state is **disabled**.

pathName **invoke**
Invoke the Tcl command associated with the button, if there is one. The return value is the return value from the Tcl command, or an empty string if there is no command associated with the button. This command is ignored if the button's state is **disabled**.

DEFAULT BINDINGS

Tk automatically creates class bindings for buttons that give them default behavior:

[1] A button activates whenever the mouse passes over it and deactivates whenever the mouse leaves the button. Under Windows, this binding is only active when mouse button 1 has been pressed over the button.

[2] A button's relief is changed to sunken whenever mouse button 1 is pressed over the button, and the relief is restored to its original value when button 1 is later released.

[3] If mouse button 1 is pressed over a button and later released over the button, the button is invoked. However, if the mouse is not over the button when button 1 is released, then no invocation occurs.

[4] When a button has the input focus, the space key causes the button to be invoked.

If the button's state is **disabled** then none of the above actions occur: the button is completely non-responsive.

The behavior of buttons can be changed by defining new bindings for individual widgets or by redefining the class bindings.

PLATFORM NOTES

On Aqua/macOS, some configuration options are ignored for the purpose of drawing of the widget because they would otherwise conflict with platform guidelines. The **configure** and **cget** subcommands can still manipulate the values, but do not cause any variation to the look of the widget. The options affected notably include **−background** and **−relief**.

EXAMPLES

This is the classic Tk "Hello, World!" demonstration:

```
button .b -text "Hello, World!" -command exit
pack .b
```

This example demonstrates how to handle button accelerators:

```
button .b1 -text Hello -underline 0
button .b2 -text World -underline 0
bind . <Key-h> {.b1 flash; .b1 invoke}
bind . <Key-w> {.b2 flash; .b2 invoke}
pack .b1 .b2
```

SEE ALSO

ttk::button(n)

KEYWORDS

button, widget