

25.3. `tkinter.tix` — Extension widgets for Tk

Source code: [Lib/tkinter/tix.py](#)

Deprecated since version 3.6: This Tk extension is unmaintained and should not be used in new code. Use `tkinter.ttk` instead.

The `tkinter.tix` (Tk Interface Extension) module provides an additional rich set of widgets. Although the standard Tk library has many useful widgets, they are far from complete. The `tkinter.tix` library provides most of the commonly needed widgets that are missing from standard Tk: `HList`, `ComboBox`, `Control` (a.k.a. `SpinBox`) and an assortment of scrollable widgets. `tkinter.tix` also includes many more widgets that are generally useful in a wide range of applications: `NoteBook`, `FileEntry`, `PanedWindow`, etc; there are more than 40 of them.

With all these new widgets, you can introduce new interaction techniques into applications, creating more useful and more intuitive user interfaces. You can design your application by choosing the most appropriate widgets to match the special needs of your application and users.

See also:

[Tix Homepage](#)

The home page for Tix. This includes links to additional documentation and downloads.

[Tix Man Pages](#)

On-line version of the man pages and reference material.

[Tix Programming Guide](#)

On-line version of the programmer's reference material.

[Tix Development Applications](#)

Tix applications for development of Tix and Tkinter programs. Tide applications work under Tk or Tkinter, and include **TixInspect**, an inspector to remotely modify and debug Tix/Tk/Tkinter applications.

25.3.1. Using Tix

```
class tkinter.tix.Tk(screenName=None, baseName=None, className='Tix')
```

Toplevel widget of Tix which represents mostly the main window of an application. It has an associated Tcl interpreter.

Classes in the `tkinter.tix` module subclasses the classes in the `tkinter`. The former imports the latter, so to use `tkinter.tix` with Tkinter, all you need to do is to import one module. In general, you can just import `tkinter.tix`, and replace the toplevel call to `tkinter.Tk` with `tix.Tk`:

```
from tkinter import tix
from tkinter.constants import *
root = tix.Tk()
```

To use `tkinter.tix`, you must have the Tix widgets installed, usually alongside your installation of the Tk widgets. To test your installation, try the following:

```
from tkinter import tix
root = tix.Tk()
root.tk.eval('package require Tix')
```

If this fails, you have a Tk installation problem which must be resolved before proceeding. Use the environment variable `TIX_LIBRARY` to point to the installed Tix library directory, and make sure you have the dynamic object library (`tix8183.dll` or `libtix8183.so`) in the same directory that contains your Tk dynamic object library (`tk8183.dll` or `libtk8183.so`). The directory with the dynamic object library should also have a file called `pkgIndex.tcl` (case sensitive), which contains the line:

```
package ifneeded Tix 8.1 [list load "[file join $dir tix8183.dll]" Tix
```

25.3.2. Tix Widgets

Tix introduces over 40 widget classes to the `tkinter` repertoire.

25.3.2.1. Basic Widgets

`class tkinter.tix.Balloon`

A `Balloon` that pops up over a widget to provide help. When the user moves the cursor inside a widget to which a `Balloon` widget has been bound, a small pop-up window with a descriptive message will be shown on the screen.

`class tkinter.tix.ButtonBox`

The `ButtonBox` widget creates a box of buttons, such as is commonly used for Ok Cancel.

`class tkinter.tix.`**ComboBox**

The **ComboBox** widget is similar to the combo box control in MS Windows. The user can select a choice by either typing in the entry subwidget or selecting from the listbox subwidget.

`class tkinter.tix.`**Control**

The **Control** widget is also known as the **SpinBox** widget. The user can adjust the value by pressing the two arrow buttons or by entering the value directly into the entry. The new value will be checked against the user-defined upper and lower limits.

`class tkinter.tix.`**LabelEntry**

The **LabelEntry** widget packages an entry widget and a label into one mega widget. It can be used to simplify the creation of “entry-form” type of interface.

`class tkinter.tix.`**LabelFrame**

The **LabelFrame** widget packages a frame widget and a label into one mega widget. To create widgets inside a **LabelFrame** widget, one creates the new widgets relative to the frame subwidget and manage them inside the frame subwidget.

`class tkinter.tix.`**Meter**

The **Meter** widget can be used to show the progress of a background job which may take a long time to execute.

`class tkinter.tix.`**OptionMenu**

The **OptionMenu** creates a menu button of options.

`class tkinter.tix.`**PopupMenu**

The **PopupMenu** widget can be used as a replacement of the `tk_popup` command. The advantage of the Tix **PopupMenu** widget is it requires less application code to manipulate.

`class tkinter.tix.`**Select**

The **Select** widget is a container of button subwidgets. It can be used to provide radio-box or check-box style of selection options for the user.

`class tkinter.tix.`**StdButtonBox**

The **StdButtonBox** widget is a group of standard buttons for Motif-like dialog boxes.

25.3.2.2. File Selectors

`class tkinter.tix.`**DirList**

The [DirList](#) widget displays a list view of a directory, its previous directories and its sub-directories. The user can choose one of the directories displayed in the list or change to another directory.

class tkinter.tix.**DirTree**

The [DirTree](#) widget displays a tree view of a directory, its previous directories and its sub-directories. The user can choose one of the directories displayed in the list or change to another directory.

class tkinter.tix.**DirSelectDialog**

The [DirSelectDialog](#) widget presents the directories in the file system in a dialog window. The user can use this dialog window to navigate through the file system to select the desired directory.

class tkinter.tix.**DirSelectBox**

The [DirSelectBox](#) is similar to the standard Motif(TM) directory-selection box. It is generally used for the user to choose a directory. DirSelectBox stores the directories mostly recently selected into a ComboBox widget so that they can be quickly selected again.

class tkinter.tix.**ExFileSelectBox**

The [ExFileSelectBox](#) widget is usually embedded in a tixExFileSelectDialog widget. It provides a convenient method for the user to select files. The style of the [ExFileSelectBox](#) widget is very similar to the standard file dialog on MS Windows 3.1.

class tkinter.tix.**FileSelectBox**

The [FileSelectBox](#) is similar to the standard Motif(TM) file-selection box. It is generally used for the user to choose a file. FileSelectBox stores the files mostly recently selected into a [ComboBox](#) widget so that they can be quickly selected again.

class tkinter.tix.**FileEntry**

The [FileEntry](#) widget can be used to input a filename. The user can type in the filename manually. Alternatively, the user can press the button widget that sits next to the entry, which will bring up a file selection dialog.

25.3.2.3. Hierarchical ListBox

class tkinter.tix.**HList**

The [HList](#) widget can be used to display any data that have a hierarchical structure, for example, file system directory trees. The list entries are indented and connected by branch lines according to their places in the hierarchy.

`class tkinter.tix.CheckList`

The `CheckList` widget displays a list of items to be selected by the user. CheckList acts similarly to the Tk `checkboxbutton` or `radiobutton` widgets, except it is capable of handling many more items than checkboxes or radiobuttons.

`class tkinter.tix.Tree`

The `Tree` widget can be used to display hierarchical data in a tree form. The user can adjust the view of the tree by opening or closing parts of the tree.

25.3.2.4. Tabular ListBox

`class tkinter.tix.TList`

The `TList` widget can be used to display data in a tabular format. The list entries of a `TList` widget are similar to the entries in the Tk `listbox` widget. The main differences are (1) the `TList` widget can display the list entries in a two dimensional format and (2) you can use graphical images as well as multiple colors and fonts for the list entries.

25.3.2.5. Manager Widgets

`class tkinter.tix.PanedWindow`

The `PanedWindow` widget allows the user to interactively manipulate the sizes of several panes. The panes can be arranged either vertically or horizontally. The user changes the sizes of the panes by dragging the resize handle between two panes.

`class tkinter.tix.ListNoteBook`

The `ListNoteBook` widget is very similar to the Tk `NoteBook` widget: it can be used to display many windows in a limited space using a notebook metaphor. The notebook is divided into a stack of pages (windows). At one time only one of these pages can be shown. The user can navigate through these pages by choosing the name of the desired page in the `hlist` subwidget.

`class tkinter.tix.NoteBook`

The `NoteBook` widget can be used to display many windows in a limited space using a notebook metaphor. The notebook is divided into a stack of pages. At one time only one of these pages can be shown. The user can navigate through these pages by choosing the visual “tabs” at the top of the `NoteBook` widget.

25.3.2.6. Image Types

The `tkinter.tix` module adds:

- `pixmap` capabilities to all `tkinter.tix` and `tkinter` widgets to create color images from XPM files.
- `Compound` image types can be used to create images that consists of multiple horizontal lines; each line is composed of a series of items (texts, bitmaps, images or spaces) arranged from left to right. For example, a compound image can be used to display a bitmap and a text string simultaneously in a Tk Button widget.

25.3.2.7. Miscellaneous Widgets

`class tkinter.tix.InputOnly`

The `InputOnly` widgets are to accept inputs from the user, which can be done with the `bind` command (Unix only).

25.3.2.8. Form Geometry Manager

In addition, `tkinter.tix` augments `tkinter` by providing:

`class tkinter.tix.Form`

The `Form` geometry manager based on attachment rules for all Tk widgets.

25.3.3. Tix Commands

`class tkinter.tix.tixCommand`

The `tix commands` provide access to miscellaneous elements of Tix's internal state and the Tix application context. Most of the information manipulated by these methods pertains to the application as a whole, or to a screen or display, rather than to a particular window.

To view the current settings, the common usage is:

```
from tkinter import tix
root = tix.Tk()
print(root.tix_configure())
```

`tixCommand.tix_configure(cnf=None, **kw)`

Query or modify the configuration options of the Tix application context. If no option is specified, returns a dictionary all of the available options. If option is specified with no value, then the method 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 method modifies the given option(s) to have the given value(s); in this case

the method returns an empty string. Option may be any of the configuration options.

`tixCommand.tix_cget(option)`

Returns the current value of the configuration option given by *option*. Option may be any of the configuration options.

`tixCommand.tix_getbitmap(name)`

Locates a bitmap file of the name *name*.xpm or *name* in one of the bitmap directories (see the `tix_addbitmapdir()` method). By using `tix_getbitmap()`, you can avoid hard coding the pathnames of the bitmap files in your application. When successful, it returns the complete pathname of the bitmap file, prefixed with the character @. The returned value can be used to configure the bitmap option of the Tk and Tix widgets.

`tixCommand.tix_addbitmapdir(directory)`

Tix maintains a list of directories under which the `tix_getimage()` and `tix_getbitmap()` methods will search for image files. The standard bitmap directory is \$TIX_LIBRARY/bitmaps. The `tix_addbitmapdir()` method adds *directory* into this list. By using this method, the image files of an applications can also be located using the `tix_getimage()` or `tix_getbitmap()` method.

`tixCommand.tix_filedialog([dlgclass])`

Returns the file selection dialog that may be shared among different calls from this application. This method will create a file selection dialog widget when it is called the first time. This dialog will be returned by all subsequent calls to `tix_filedialog()`. An optional *dlgclass* parameter can be passed as a string to specified what type of file selection dialog widget is desired. Possible options are `tix`, `FileSelectDialog` or `tixExFileSelectDialog`.

`tixCommand.tix_getimage(self, name)`

Locates an image file of the name *name*.xpm, *name*.xbm or *name*.ppm in one of the bitmap directories (see the `tix_addbitmapdir()` method above). If more than one file with the same name (but different extensions) exist, then the image type is chosen according to the depth of the X display: xbm images are chosen on monochrome displays and color images are chosen on color displays. By using `tix_getimage()`, you can avoid hard coding the pathnames of the image files in your application. When successful, this method returns the name of the newly created image, which can be used to configure the `image` option of the Tk and Tix widgets.

`tixCommand.tix_option_get(name)`

Gets the options maintained by the Tix scheme mechanism.

`tixCommand.tix_resetoptions(newScheme, newFontSet[, newScmPrio])`

Resets the scheme and fontset of the Tix application to *newScheme* and *newFontSet*, respectively. This affects only those widgets created after this call. Therefore, it is best to call the `resetoptions` method before the creation of any widgets in a Tix application.

The optional parameter *newScmPrio* can be given to reset the priority level of the Tk options set by the Tix schemes.

Because of the way Tk handles the X option database, after Tix has been imported and inited, it is not possible to reset the color schemes and font sets using the `tix_config()` method. Instead, the `tix_resetoptions()` method must be used.