第 8 章

容器控件

本章摘要

8-1 框架 Frame

8-2 标签框架 LabelFrame

8-3 顶层窗口 Toplevel

Frame 可翻译为框架，它在 Widget 中的类别名称就是 Frame。LabelFrame 可翻译为标签框架，它在 Widget 中的类别名称就是 LabelFrame。这两个控件主要是当作容器使用，设计时 LabelFrame 可以在外观看到标签名称。本章要介绍的另一个 Widget 是 Toplevel，它与 Frame 类似，但是将产生一个分离的窗口容器。

8-1 框架 Frame

8-1-1 框架的基本概念

这是一个容器控件，当我们设计的 GUI 程序很复杂时，此时可以考虑将一系列相关的 Widget 组织在一个框架内，这样可以方便管理。它的构造方法语法如下。

```
Frame( 父对象 ,options, … )     # 父对象可以省略，可参考 ch8_1_1.py
```

Frame( ) 方法的第一个参数是父对象，表示这个框架将建立在哪一个父对象内。下列是 Frame( ) 方法内其他常用的 options 参数。

(1)bg 或 background：背景色彩。

(2)borderwidth 或 bd：标签边界宽度，默认是 2。

(3)cursor：当鼠标光标在框架上时的光标形状。

(4)height：框架的高度，单位是像素。

(5)highlightbackground：当框架没有取得焦点时的颜色。

(6)highlightcolor：当框架取得焦点时的颜色。

(7)highlightthickness：当框架取得焦点时的厚度。

(8)relief：默认是 relief=FLAT，可由此控制框架外框，可参考 ch8_4.py。

(9)width：框架的宽度，单位是像素，省略时会自行调整为实际宽度。

程序实例 ch8_1.py：建立三个不同底色的框架。

```
1  # ch8_1.py
2  from tkinter import *
3
4  root = Tk()
5  root.title("ch8_1")
6
7  for fm in ["red","green","blue"]:    # 建立三个不同底色的框架
8      Frame(root,bg=fm,height=50,width=250).pack()
9
10 root.mainloop()
```

执行结果

从上述实例应该了解，框架也是一个 Widget 控件，所以最后也需要使用控件配置管理员包装与定位，此例中是使用 pack( )。

程序实例 ch8_1_1.py：在调用 Frame 构造方法时，省略父对象。

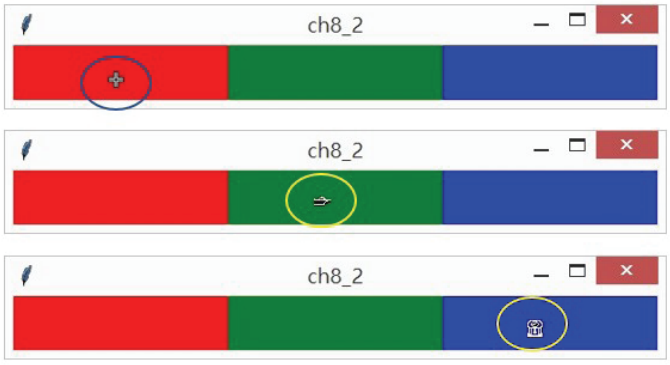
```
8       Frame(bg=fm,height=50,width=250).pack()
```

执行结果 与 ch8_1.py 相同。

程序实例 ch8_2.py：使用横向配置方式 (side=LEFT) 重新设计 ch8_1.py，同时让鼠标光标在不同的框架上有不同的形状。

```
1   # ch8_2.py
2   from tkinter import *
3
4   root = Tk()
5   root.title("ch8_2")
6
7   # 用字典存储框架颜色与光标形状
8   fms = {'red':'cross','green':'boat','blue':'clock'}
9   for fmColor in fms:          # 建立三个不同底色的框架与光标形状
10      Frame(root,bg=fmColor,cursor=fms[fmColor],
11            height=50,width=200).pack(side=LEFT)
12
13  root.mainloop()
```

执行结果



8-1-2 在框架内创建 Widget 控件

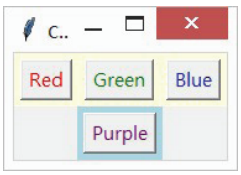
创建框架时会传回框架对象，假设此对象是 A，以后在此框架内建立 Widget 控件时，此对象 A 就是框架内 Widget 控件的父容器。下面是在框架内创建功能按钮对象的讲解。

```
A = Frame(root, … )          # 传回框架对象 A
btn = Button(A, … )          # 框架对象 A 是 btn 功能按钮的父容器
```

程序实例 ch8_3.py：建立两个框架，同时在上层框架 frameUpper 内建三个功能按钮，下层框架是 frameLower，同时在此建立一个功能按钮。

```
1  # ch8_3.py
2  from tkinter import *
3
4  root = Tk()
5  root.title("ch8_3")
6
7  frameUpper = Frame(root,bg="lightyellow")    # 建立上层框架
8  frameUpper.pack()
9  btnRed = Button(frameUpper,text="Red",fg="red")
10 btnRed.pack(side=LEFT,padx=5,pady=5)
11 btnGreen = Button(frameUpper,text="Green",fg="green")
12 btnGreen.pack(side=LEFT,padx=5,pady=5)
13 btnBlue = Button(frameUpper,text="Blue",fg="blue")
14 btnBlue.pack(side=LEFT,padx=5,pady=5)
15
16 frameLower = Frame(root,bg="lightblue")      # 建立下层框架
17 frameLower.pack()
18 btnPurple = Button(frameLower,text="Purple",fg="purple")
19 btnPurple.pack(side=LEFT,padx=5,pady=5)
20
21 root.mainloop()
```

执行结果



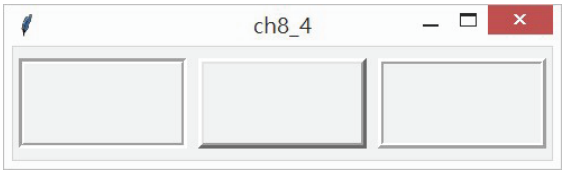
8-1-3 活用 relief 属性

可以利用 relief 属性的特性，将 Widget 控件建立在框架内。

程序实例 ch8_4.py：建立三个框架，分别使用不同的 relief 属性。

```
# ch8_4.py
from tkinter import *

root = Tk()
root.title("ch8_4")

fm1 = Frame(width=150,height=80,relief=GROOVE, borderwidth=5)
fm1.pack(side=LEFT,padx=5,pady=10)

fm2 = Frame(width=150,height=80,relief=RAISED, borderwidth=5)
fm2.pack(side=LEFT,padx=5,pady=10)

fm3 = Frame(width=150,height=80,relief=RIDGE, borderwidth=5)
fm3.pack(side=LEFT,padx=5,pady=10)

root.mainloop()
```

执行结果



8-1-4　在含 raised 属性的框架内创建复选框

程序实例 ch8_5.py：创建一个含 raised 属性的框架，同时在此框架内创建标签和复选框。

```
# ch8_5.py
from tkinter import *

root = Tk()
root.title("ch8_5")

fm = Frame(width=150,height=80,relief=RAISED,borderwidth=5) # 创建框架
lab = Label(fm,text="请复选常用的程序语言")          # 创建标签
lab.pack()
python = Checkbutton(fm,text="Python")              # 创建Phthon复选框
python.pack(anchor=W)
java = Checkbutton(fm,text="Java")                  # 创建Java复选框
java.pack(anchor=W)
ruby = Checkbutton(fm,text="Ruby")                  # 创建Ruby复选框
ruby.pack(anchor=W)
fm.pack(padx=10,pady=10)                            # 包装框架

root.mainloop()
```

执行结果

8-1-5 额外对 relief 属性的支持

在标准的 Frame 框架中，对于 relief 属性并没有完全支持，例如，solid 和 sunken 属性，此时可以使用 tkinter.ttk 的 Frame 和 Style 模块。下面将直接以实例讲解。

程序实例 ch8_6.py：建立 6 个框架，每个框架有不同的 relief。

```
# ch8_6.py
from tkinter import Tk
from tkinter.ttk import Frame, Style

root = Tk()
root.title("ch8_6")
style = Style()                    # 改用Style
style.theme_use("alt")             # 改用alt支持Style

fm1 = Frame(root,width=150,height=80,relief="flat")
fm1.grid(row=0,column=0,padx=5,pady=5)

fm2 = Frame(root,width=150,height=80,relief="groove")
fm2.grid(row=0,column=1,padx=5,pady=5)

fm3 = Frame(root,width=150,height=80,relief="raised")
fm3.grid(row=0,column=2,padx=5,pady=5)

fm4 = Frame(root,width=150,height=80,relief="ridge")
fm4.grid(row=1,column=0,padx=5,pady=5)

fm5 = Frame(root,width=150,height=80,relief="solid")
fm5.grid(row=1,column=1,padx=5,pady=5)

fm6 = Frame(root,width=150,height=80,relief="sunken")
fm6.grid(row=1,column=2,padx=5,pady=5)

root.mainloop()
```

执行结果

上述程序中需使用 tkinter.ttk 模块内的 Frame 作为支持才可正常显示 relief 外框，同时留意第 8 行中的 alt 参数主要是此机制内对于 relief 支持的参数。

8-2 标签框架 LabelFrame

8-2-1 标签框架的基本概念

这也是一个容器控件，主要是将一系列相关的 Widget 组织在一个标签框架内，然后给它一个名称。它的构造方法语法如下。

```
LabelFrame( 父对象 ,options, … )
```

LabelFrame( ) 方法的第一个参数是父对象，表示这个标签框架将建立在哪一个父对象内。下列是 LabelFrame( ) 方法内其他常用的 options 参数。

(1)bg 或 background：背景色彩。

(2)borderwidth 或 bd：标签边界宽度，默认是 2。

(3)cursor：当鼠标光标在框架上时的光标形状。

(4)font：标签框架中文字的字形。

(5)height：框架的高度，单位是像素。

(6)highlightbackground：当框架没有取得焦点时的颜色。

(7)highlightcolor：当框架取得焦点时的颜色。

(8)highlightthickness：当框架取得焦点时的厚度。

(9)labelAnchor：设置放置标签的位置。

(10)relief：默认是 relief=FLAT，可由此控制框架的外框。

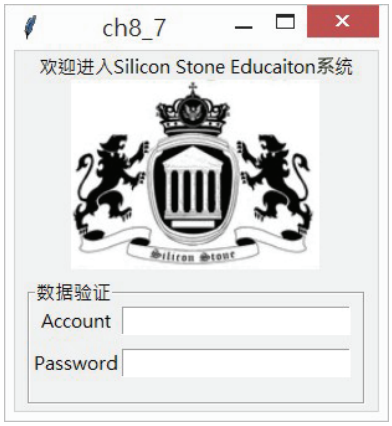
(11)text：标签内容。

(12)width：框架的宽度，单位是像素，省略时会自行调整为实际宽度。

程序实例 ch8_7.py：重新设计 ch5_3.py，将账号和密码字段使用标签框架框起来，此框架标签的文字是“数据验证”。

```
# ch8_7.py
from tkinter import *

root = Tk()
root.title("ch8_7")                              # 窗口标题

msg = "欢迎进入Silicon Stone Educaiton系统"
sseGif = PhotoImage(file="sse.gif")              # Logo图像文件
logo = Label(root,image=sseGif,text=msg,compound=BOTTOM)
logo.pack()

# 以下是LabelFrame标签框架
labFrame = LabelFrame(root,text="数据验证")  # 创建框架标签
accountL = Label(labFrame,text="Account")    # account标签
accountL.grid(row=0,column=0)
pwdL = Label(labFrame,text="Password")       # pwd标签
pwdL.grid(row=1,column=0)

accountE = Entry(labFrame)                       # account文本框
accountE.grid(row=0,column=1)                    # 定位account文本框
pwdE = Entry(labFrame,show="*")                  # pwd文本框
pwdE.grid(row=1,column=1,pady=10)                # 定位pwd文本框
labFrame.pack(padx=10,pady=5,ipadx=5,ipady=5)    # 包装与定位标签框架

root.mainloop()
```

执行结果



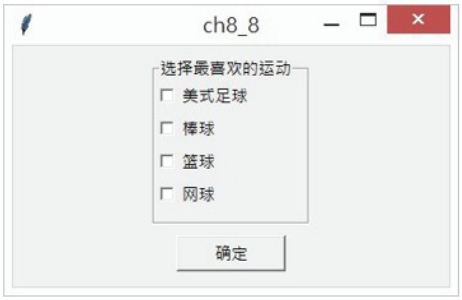
8-2-2　将标签框架应用于复选框

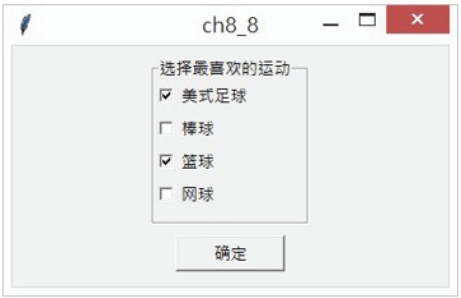
标签框架的应用范围很广泛，也常应用于将选项按钮或是复选框组织起来。下面将直接以实例讲解。

程序实例 ch8_8.py：重新设计 ch7_8.py，将复选框用标签框架框起来，同时设置了 root 窗口的宽度和高度。

```
# ch8_8.py
from tkinter import *

def printInfo():
    selection = ''
    for i in checkboxes:                        # 检查此字典
        if checkboxes[i].get() == True:         # 被选取则执行
            selection = selection + sports[i] + "\t"
    print(selection)

root = Tk()
root.title("ch8_8")                             # 窗口标题
root.geometry("400x220")
#以下建立标签框架与字典
labFrame = LabelFrame(root,text="选择最喜欢的运动")
sports = {0:"美式足球",1:"棒球",2:"篮球",3:"网球"}     # 运动字典
checkboxes = {}                                 # 字典存放被选取项目
for i in range(len(sports)):                    # 将运动字典转成复选框
    checkboxes[i] = BooleanVar()                # 布尔变量对象
    Checkbutton(labFrame,text=sports[i],
                variable=checkboxes[i]).grid(row=i+1,sticky=W)
labFrame.pack(ipadx=5,ipady=5,pady=10)          # 包装定位标签框架

btn = Button(root,text="确定",width=10,command=printInfo)
btn.pack()

root.mainloop()
```

执行结果





8-3 顶层窗口 Toplevel

8-3-1 Toplevel 窗口的基本概念

这个控件的功能类似于 Frame，但是这个控件所产生的容器是一个独立的窗口，有自己的标题栏和边框。它的构造方法语法如下。

```
Toplevel(options, … )
```

下列是 LabelFrame( ) 方法内其他常用的 options 参数。

(1)bg 或 background：背景色彩。

(2)borderwidth 或 bd：标签边界宽度，默认是 2。

(3)cursor：当鼠标光标在 Toplevel 窗口上时的光标形状。

(4)fg：文字前景颜色。

(5)font：字形。

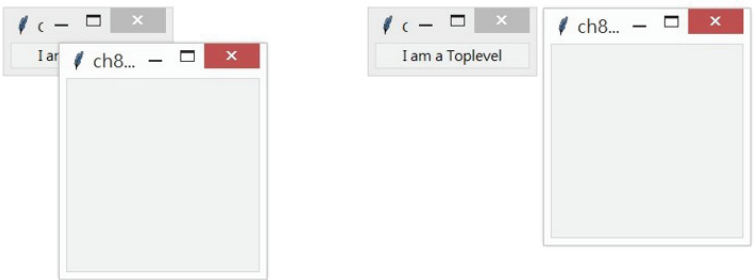
(6)height：窗口高度。

(7)width：窗口宽度。

程序实例 ch8_9.py：建立一个 Toplevel 窗口，为了区分在 Toplevel 窗口中增加字符串“I am a toplevel.”。

```
1  # ch8_9.py
2  from tkinter import *
3
4  root = Tk()
5  root.title("ch8_9")
6
7  tl = Toplevel()
8  Label(tl,text = 'I am a Toplevel').pack()
9
10 root.mainloop()
```

执行结果 下方左图是执行结果画面，右图是适度移动主窗口后的结果。

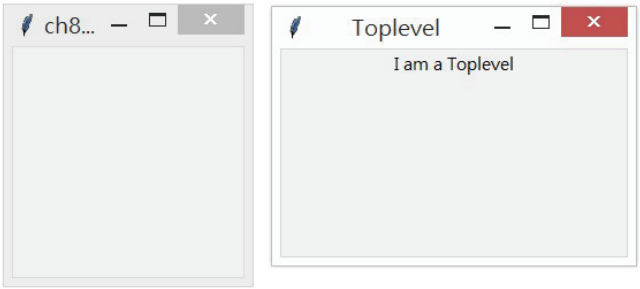


Toplevel 窗口建立完成后，如果关闭 Toplevel 窗口，原主窗口仍可以继续使用，但是如果关闭了主窗口，Toplevel 窗口将自动关闭。在第 1 章介绍建立主窗口时有介绍过窗口属性设置的方法，这些方法中有些可以供 Toplevel 窗口使用。

程序实例 ch8_10.py：设置 Toplevel 窗口的标题和大小。

```
1  # ch8_10.py
2  from tkinter import *
3
4  root = Tk()
5  root.title("ch8_10")
6
7  tl = Toplevel()
8  tl.title("Toplevel")
9  tl.geometry("300x180")
10 Label(tl,text = 'I am a Toplevel').pack()
11
12 root.mainloop()
```

执行结果



8-3-2　使用 Toplevel 窗口仿真对话框

程序实例 ch8_11.py：这个程序执行时会有一个 Click Me 按钮，当单击此按钮时会由一个随机数产生 Yes、No、Exit 字符串，这些字符串会出现在 Toplevel 窗口内。

```
1  # ch8_11.py
2  from tkinter import *
3  import random
4
5  root = Tk()
6  root.title("ch8_11")
7
8  msgYes, msgNo, msgExit = 1,2,3
9  def MessageBox():                    # 创建对话框
10     msgType = random.randint(1,3)    # 随机数产生对话框方式
11     if msgType == msgYes:            # 产生Yes字符串
```

```
        labTxt = 'Yes'
    elif msgType == msgNo:          # 产生No字符串
        labTxt = 'No'
    elif msgType == msgExit:        # 产生Exit字符串
        labTxt = 'Exit'
    tl = Toplevel()                 # 建立Toplevel窗口
    tl.geometry("300x180")          # 设置对话框大小
    tl.title("Message Box")
    Label(tl,text=labTxt).pack(fill=BOTH,expand=True)

btn = Button(root,text='Click Me',command = MessageBox)
btn.pack()

root.mainloop()
```

执行结果