Tkinter 8.5 参考：一个 Python 的图形界面

原文作者：Joh W.Shipman
译者：William

原文发布时间：2013-06-24 12:46

摘要

本参考介绍了用 **Python** 编程语言中的 **Tkinter** 套件来创建用户图形界面。本参考涵盖了 **ttk** 主题控件。

本参考现可在线浏览[1]同时另有**PDF** 文档[2]。如有意见请发邮件到**tcc-doc@nmt.edu**。

如对翻译有任何意见请发邮件至**william0victor@gmail.com**

[1] http://www.nmt.edu/tcc/help/pubs/tkinter/
[2] http://www.nmt.edu/tcc/help/pubs/tkinter/tkinter.pdf

目录

第 1 章　Python 的一个跨平台用户图形界面

Tkinter 是 Python 的一个 GUI（graphical user interface）组件。本文档适用于运行在 Linux 操作系统的 X Window 系统中的 Python2.7 和 Tkinter8.5。你的版本或稍有差异。
相关参考：

- Fredrik Lundh, who wrote *Tkinter*, has two versions of his *An Introduction to Tkinter*: a more complete 1999 version[3] and a 2005 version[4] that presents a few newer features.
- *Python 2.7 quick reference*[5]: general information about the Python language
- For an example of a sizeable working application (around 1000 lines of code), see huey: *A color and font selection tool*[6].

我们将从 *Tkinter* 可见的部分开始：创建控件并布局在屏幕上。稍后我们将探讨如何将应用程序前端的面板关联到后端逻辑。

[3] http://www.pythonware.com/library/tkinter/introduction/
[4] http://effbot.org/tkinterbook/
[5] http://www.nmt.edu/tcc/help/pubs/python/
[6] http://www.nmt.edu/tcc/help/lang/python/examples/huey/

第 2 章　一个小程序

下面是一个只含有一个推出按钮的 *Tkinter* 小程序

```
#!/usr/bin/env python                                    1
import Tkinter as tk                                     2
class Application(tk.Frame):                             3
        def __init__(self, master=None):
                tk.Frame.__init__(self, master)          4
                self.grid()                              5
                self.createWidgets()
        def createWidgets(self):
                self.quitButton = tk.Button(self, text='Quit',
                        command=self.quit)               6
                self.quitButton.grid()                   7
app = Application()                                      8
app.master.title('Sample␣application')                   9
app.mainloop()                                           10
```

1. 假如你的系统中已正确安装 Python，该行将会使脚本自动执行。
2. 该行将 *Tkinter* 模块导入到程序的命名空间，但重命名为 tk。
3. 你的程序的类必须从 *Tkinter* 的 Frame 类继承。
4. 调用父类 Frame 的构造函数。
5. 必须让程序真正显示在屏幕上。
6. 创建一个按钮，标记为“Quit”。
7. 将按钮放入程序中
8. 通过实例化 Application 类启动主程序
9. 这个方法将窗口的 title 设为"Sample application"。
10. 启动程序的主程，等待鼠标和键盘事件。

第 3 章　解释

在我们继续下面的内容前，让我们来解释一些常用的名词。

window

在不同的语境中这个词有不同的意思，但通常它是指你电脑显示屏中的某个矩形区域。

top-level window

一个独立存在于你屏幕中的窗口，它将为你的系统桌面管理器装饰框架和控制器。你可以在桌面上四处移动它。通常，你也可以调整它的尺寸除非程序禁止了。

widget

这个词通常指组成程序用户图形界面的组件。例如：按钮、单选框、文本域、框架以及文本标签。

frame

在 *Tkinter* 中，Frame 控件是组成复杂布局的基本单元。它指一个能够包含其它控件的矩形区域。

child,parent

当任意一个控件被创建时，父级 -子级关系就已经建立。例如，当你将一个文本标签放入一个框架中，框架就是文本标签的父级。

第 4 章　布局管理

接下来我们将讲讲控件，组成程序图形界面的“积木块”。怎样布置控件到窗口中？尽管在 *Tkinter* 中有三种不同“图形管理器”，但是笔者特别愿意用.grid() 图形管理器来布局。这个管理将所有的窗口或框架当做一个表 ---有行和列的网格。

- 单元格是一行和一列的交叉区域。
- 每列最宽的单元格的宽度是该列的列宽。
- 每行最高的单元格的高度是该行的行高。
- 控件不可能完全充满单元格的所有空间，你可以对单元格进行指定。你既可以不调整控件外的多余控件，也可以伸展水平方位或垂直方位来使控件填满单元格。
- 你可以将多个单元格合并生成一个大单元格。

当你创建了一个控件，除非你在图形管理器中注册了它否则它将不显示。因此，构造和放置控件的两个步骤进行如下：

```
self.thing = tk.Constructor(parent, ...)
self.thing.grid(...)
```

Constructor 是如按钮、框架等控件的类，*Parent* 是将被构造的子类控件的父类控件。所有的控件都有.grid() 方法，你可以使用它来告诉图形管理器怎么放置控件。

第 4.1 节　.grid() 方法

将一个控件 *W* 显示到程序窗口中：

```
w.grid(option=value, ...)
```

该方法在图形管理器中注册了一个控件 *w*---如果不这么做，控件将只存在内部不会显示出来。见表一“.grid() 图形管理器参数”：

column	控件放置的列数，从 0 开始计算。默认值是 0。
columnspan	通常一个控件只占据网格中的一个单元格。然而，你可以选取行中的多个单元格，并在 columnspan 选项中设置单元格的数量来整合他们到一个大单元格。例如，*w*.grid(row=0,column=2,columnspan=3)，例中将会把 *w* 控件放置在一个横跨 0 行 2，3，4 列的一个单元格中。
in_	注册 *w* 作为某个控件如 w_2 的子类，用法 in_=w_2。当 *w* 被创建时，新的 w_2 控件必须作为 *parent* 控件的子类来使用。
ipadx	内部 x padding。这个维度是增加控件内部左边和右边。
ipady	内部 y padding。这个维度是增加控件内部顶部和底部。
padx	外部 x padding。这个维度是增加控件外部左边和右边。
pady	外部 y padding。这个维度是增加控件上部和下部。
row	你想把控件插入的行数，从 0 计数。默认是下一个更高的未被占用的行。
rowspan	通常一个控件只占据网格的一个单元格。你可以选取列内的多个单元格，然后，给选中的单元格设置 rowspan 选项。本选项和 columnspan 选项结合使用来抓取一块单元格。例子，*w*.grid(row=3,column=2,rowspan=4,columnspan=5) 例中将 *w* 微件放入一个由 3-6 列 2-6 行组成的区域中。
sticky	本项决定怎样分配单元格中的微件所占空间之外的空间。见下。

- 如果未设置 sticky 属性，默认会将微件在单元格中居中放置。
- 你可以使用 sticky=NE（右上），SE（右下），SW（左下），或者 NW（左上）来布局微件到单元格的四角。
- 你可以使用 sticky=N（中上），E（中右），S（中下），或者 W（中左）来布局微件到一边的相对中间。
- 使用 sticky=N+S 垂直扩展微件但让它水平居中。
- 使用 sticky=E+W 水平扩展微件但让它垂直居中。
- 使用 sticky=N+E+S+W 在水平和垂直方位来扩展微件填充单元格。
- 其它的组合也可使用。例如，sticky=N+S+W 将垂直扩展微件并放置微件在相对东（左）边。

第 4.2 节　其它 grid 管理方法

这些 grid 相关的方法在所有控件上都有定义：

w.grid_bbox (column=None, row=None, col2=None, row2=None)

返回一个四元组描述微件 w 中一些或所有 grid 系统的边界框。前两个数字返回左上角区域的 x 和 y 坐标，后两个数字是宽和高。

如果你传递行和列变量，返回的限定框描述所在行和列的单元格的区域。如果你也传递了 col2 和 row2 参数，返回的限定框描述包含从行 column 到 col2，列 row 到 row2 的区域。

例如，w.grid_bbox(0,0,1,1) 返回四个单元格的限定框，不是一个。

w.grid_forget()

本方法使微件 w 从屏幕上消失。它还存在，只是不可见。你可以使用.grid() 使它再次显示，但是它将不会记住它的 grid 选项。

w.grid_info()

返回一个键为 w 微件选项名字的字典，以及这些选项相应的值。

w.grid_location (x,y)

赋予关联的包含的微件一个坐标，本方法返回一个数组（col,row）描述 w 微件的网格系统的单元格包含的屏幕坐标。

w.grid_propagate()

通常，所有微件传送他们的尺寸，意味着他们调节来适应内容。然而，有时你想约束一个微件到确定的尺寸，忽略它内容的尺寸，这样做，调用 w.grid_propagate(0) 限制 w 微件的尺寸。

w.grid_remove()

本方法类似.grid_forget()，但是它的 grid 选项会记住，所以如果你再.grid() 它，它将会使用相同的 grid 配置选项。

w.grid_size()

分别在 w 微件 grid 系统中返回一个包含行数和列数的二元组。

w.grid_slaves (row=None, column=None)

返回由微件 w 管理的微件的目录。如果没有提供参数，你将会获得所有被管理的微件的目录。使用 row= 参数只选择一列微件，或者使用 column= 参数只选择一行的微件。

第 4.3 节　设置行和列的尺寸

除非你采取确切的措施，网格列内的微件宽将会等于它的最大宽度，并且网格行内的微件高将会是最高的单元格的高。微件上的 sticky 属性只控制它被放置的地方如果它没有完全填充单元格。

w.columnconfigure (N, option=value,...)

w 微件的网格层中，配置 column N 以便所给选项有所给的值。对于选项，请看下表。

w.rowconfigure (N, option=value, ⋯)

w 微件的网格层中，配置 row N 以便所给的选项有所给的值。对于选项，请看下表。

以下是用来配置 column 和 row 尺寸的选项。

minsize	以像素为最小单位，列和行的最小尺寸。
pad	若干像素将会被加入所给的列或行，及以上的列或行中最大的单元格。

weight	为了使一列或一行可拉伸，当重新分配额外的空间时，使用此选项并提供一个值，赋予该列或行相对权重。例如，如果一个微件 w 包含一个网格层，这些行将会分配 3/4 的额外空间到第一列及 1/4 到第二列：
	`w.columnconfigure(0, weight=3)`
	`w.columnconfigure(1, weight=1)`
	如果没有使用此项，列或行将不会伸展。

第 4.4 节　让根窗口可重组

如果你想让用户调整你的整个程序窗口，并分配给内部的控件吗？只需普通的几个操作就能实现。

这需要用到行和列尺寸管理的方法，在第 4.3 小节已经提到，“配置列和行大小”(p.7)，来使你的 *Application* 控件的网格可伸缩。然而，那仅仅是不够的。

思考下第二章讨论的小程序，“一个小程序”(p.2)，其只包含一个退出按钮。如果你运行此程序，并调整窗口大小，按钮会保持同样的尺寸，并居中于窗口。

以下是小程序内.__createWidgets() 方法的替代版本。在这个版本中，退出按钮一直填充可利用空间。

```
def createWidgets(self):
        top=self.winfo_toplevel()                                    1
        top.rowconfigure(0,weight=1)                                 2
        top.columnconfigure(0,weight=1)                              3
        self.rowconfigure(0,weight=1)                                4
        self.columnconfigure(0,weight=1)                             5
        self.quit=Button (self, text="Quit", command=self.quit )
        self.quit.grid( row=0, column=0, sticky=N+S+E+W)             6
```

1. 顶层窗口是屏幕上最外层的窗口。然而，这个窗口不是你的程序的窗口——它是程序实例的父类。要获取顶层窗口，在程序中的任何微件上调用.winfo_toplevel() 方法。参看 25 节，“通用微件方法”(p.91)。
2. 本行使顶层窗口的 0 行网格可伸展。
3. 本行使顶层窗口的 0 列网格可伸展。
4. 使 0 行的程序微件的网格可伸展。
5. 使 0 列的程序微件的网格可伸展
6. 参数 sticky=N+S+E+W 使按钮扩张填满它的单元格

还必须作出一个变化。在构造函数中，如下显示的改变第二行：

```
def __init__(self,master=None):
        Frame.__init__(self,master)
        self.grid(sticky=N+S+E+W)
        self.createWidgets()
```

sticky=N+S+E+W 参数对 self.grid() 是必要的，因此程序微件将会扩张填充它的顶层窗口网格的单元格。