Python 界面程序开发应用技术

曾 浩 尚维来
(南京化工职业技术学院信息系 江苏·南京 210048)

中图分类号:TP393 文献标识码:A 文章编号:1672-7894(2010)30-087-03

摘 要 本文介绍了 Python 下开发界面程序的基本技术情况,简要分析了各种技术的优劣,并着重探讨了其中 WxPython 界面库和 Tkinter 界面库的使用方法。

关键词 Python 界面开发 wxPython Tkinter

1 python 界面开发综述

Python 界面除了标准库以外,还有许多其他高质量的库,如 wxPython、Twisted 和 Python 图形库等等,数不胜数。

Python 界面包括图形用户界面、文件处理、多媒体、正则表达式、文档生成、单元测试、线程、数据库、网络通讯、网页浏览器、CGI、FTP、电子邮件、XML、HTML、WAV 文件、密码系统。

Python 的年岁比 Java 老,到现在,这种语言已经发展出了规模,虽然不大但仍然在成长中。Python 整合了各种模块、异常、动态输入、高级动态数据类型和类等概念。它同多种系统函数和系统库乃至各类 Windows 系统建立了接口。

如果说 Python 并不一定能渗透到任何计算领域的话,至少在企业开发这一行它的处境就是这样。在企业开发领域,Python 还被当成一种可以替换 Perl 的脚本语言。本文就对 Python 界面构造企业级应用这一问题做简短阐述。

Python 语言的某些特点使其成为企业应用的合理选择:

◆免费可用 (像 Perl 一样,Python 也是开放源代码软件)。

◆稳定(Python 目前已发布到 2.2 版,我刚才已经提到,它的年头比 Java 还老)。

◆良好地支持对象、模块和其他可重用机制。

◆采用 C 和 Java 语言的集成方便性和扩展性。

以上这些特性非常有利于 Python 在企业中的应用,不过,Python 还具有一些适合企业应用但却不太令人注意的其他合理因素。IT 部分通常要完成的任务相当繁重但支撑这些工作的资源却很少,这已经成为公开的秘密。任何承诺提高编码效率、降低软件总成本的 IT 解决方案都应该进行周密的考虑。

Python 所具有的一个显著优势就是可以在企业的软件创建和维护阶段节约大量资金,而这两个阶段的软件成本占到了软件整个生命周期中总成本的 50%到 95%。Python 清晰可读的语法使得软件代码具有异乎寻常的易读性,甚至对那些不是最初接触和开发原始项目的程序员都能具有这样的强烈感觉。

虽然某些程序员反对在 Python 代码中大量使用空格,不过,几乎人人都承认 Python 代码的可读性远胜于 C 或者 Java,后两者都采用了专门的字符标记代码块结构、循环、函数以及其他编程结构的开始和结束。

提倡 Python 的人还宣称,采用这些字符可能会产生显著的编程风格差异,使得那些负责维护代码的人遭遇代码可读性方面的困难。Python 明晰的语法使其成为一种学习曲线平和的编程语言。有经验的程序员甚至可能在一天之内就掌握 Python 的基础知识。至多不过一周作用就可以上手,而编程语言的专家则肯定会比他掌握 C、C++、Java 甚至 Perl 要快很多。

Python 界面因其语法的明晰而获得的最大好处或许该算是开发时间的大大降低了。

业界开发经验告诉我们:开发时间因为 Python 的采用而大大节约,同时却并没有给软件性能带来任何负面影响。就软件开发业务而言,快速地开发出新产品比开发出性能惊人的软件产品要重要得多,开发时间相比软件的优化要贵得多(以每小时计算)。

对开发速度采用何种评价标准一直是个很难确定的问题,不过通常情况下,普遍认为开发 Python 应用程序的速度大约是用 Java 开发同类应用程序速度的 10 倍。要是相比 C/C++ 节约的时间就更多了。

Python 最大的特点就在于它的快速开发功能。作为一种胶水型语言,Python 几乎可以渗透在我们编程过程中的各个领域。这里简单介绍一下用 Python 进行 gui 开发的一些选择。

(1) Tkinter

Tkinter 似乎是与 tcl 语言同时发展起来的一种界面库。Tkinter 是 Python 配备的标准 gui 库,也是 opensource 的产物。Tkinter 可用于 windows/linux/unix/macintosh 操作系统,而且显示风格是本地化的。Tkinter 用起来非常简单,Python 自带的 IDLE 就是采用它写的。除此外,Tkinter 的扩展集 Pmw 和 Tix 功能上都要相对它强大,但 Tkinter 却是最基本的。我认为,在用 Python 做 gui 开发,Tkinter 是最基本的知识,所以这个环节是必须要学习的。你或许在以后的开发中并不

常用 Tkinter，但是一些小型的应用上面，它还是很有用的，而且开发速度也很快。

⑵ WxPython

WxWidgets 应该算是近几年比较流行的 GUI 跨平台开发技术了。WxWidgets 有不同的版本应用，有 c++ 的，也有 basic 的，现在在 Python 上面也有较好的移植。WxPython 的功能上面要强于 Tkinter，它提供了超过 200 个类，面向对象的编程风格，设计的框架类似于 MFC。对于大型 GUI 应用上面，WxPython 还是具有很强的优势的。boa constructor 可以帮助我们快速可视地构建 WxWidgets 界面。

⑶ PyQT

Qt 同样是一种开源的 GUI 库，Qt 的类库有 300 多个，函数有 5700 多个。Qt 同样适合于大型应用，由它自带的 qt designer 可以让我们轻松来构建界面元素。

⑷ pyGtk

Gtk 是 linux 下 Gnome 的核心开发库，功能上面非常齐全。值得说明的是，在 Windows 平台下 gtk 的显示风格并不是特别本地化。不过它自带的 glade 界面设计器还是可以帮你省不少事的。

⑸ Jython

尝试过用 Python 访问 Java 类库吗，那么就用 Jython 吧。Jython 其实可以认为是另外一个 Python 开发环境，基于 Java 的，但是大多数的 CPython 调用 jython 下还是可以的。你可以在 Jython 环境下像使用 Java 一样来通过 Python 的语法来调用 Java 语言，真的很酷。

⑹ MFC

Windows Pywin32 允许你像 VC 一样的形式来使用 PYTHON 开发 win32 应用。代码风格可以类似 win32 sdk，也可以类似 MFC，由你选择。

⑺ PythonCard

PythonCard 其实是对 WxPython 的再封装，不过封装得更加简单，使用起来觉得比 WxPython 更直观，也更简单化了。

⑻ Dabo

仍是一个基于 WxPython 的再封装库，没用过，不太了解。它提供数据库访问，商业逻辑以及用户界面。

⑼ AnyGui

通过底层的 api 来访问其他工具集，像 Tkinter，WxPython 和 qt 具体也没怎么用过。

⑽ WPY

MFC 风格的 Gui 开发库，代码风格也类似于 MFC，尽管如此，你依旧可以使用这个库来开发 GUI 应用，而不用担心平台移植的问题。它同样是一个跨平台的库。

⑾ IronPython

如果你要想开发.net 下面的应用的话，那么 IronPython 就是你的选择，与 jython 有点类似，它同样支持标准的 Python 模块，但同样增加了对.net 库的支持。你也可以理解为它是另一个 Python 开发环境。你可以非常方便地使用 Python 语法进行.net 应用的开发，这一点听起来真的挺有意思。

总之，介绍了这么多，我个人意见是，如果你是 java 用户，那么你就用 jython 吧，除了可以享受 Python 的模块功能及语法外，你可以找到许多 java 的影子 如果你是.net 用户，那么就用 iron Python 吧 如果你对 Visual C++ 很熟悉，那么你可以使用 MFC，WPY 或是 WxPython，当然我更建议使用 WxPython 了。当然，我认为对于 Tkinter 是每一个原来搞 C 的人都应该了解和学习的 GUI 库，因为它很轻便，小型应用就可以使用它来搞定，而对于较大型应用可以采用 PyGtk，PyQt，WxPython 或 PythonCard 来搞定，这样的话，既可以注重知识的衔接性，也可以快速进行软体的开发了。

下面具体介绍一下两个具体的界面库：

2 WxPython

WxPython 是 Python 编程语言的一个 GUI 工具箱。它使得 Python 程序员能够轻松地创建功能强大的图形用户界面程序。它是 Python 语言对流行的 WxWidgets 跨平台 GUI 工具库的绑定。而 wxWidgets 是用 C++ 语言写成的。和 Python 语言与 WxWidgets GUI 工具库一样，WxPython 是开源软件。这意味着任何人都可以免费地使用它并且可以查看和修改它的源代码，或者贡献补丁、增加功能。WxPython 是跨平台的。这意味着同一个程序可以不经修改地在多种平台上运行。现今支持的平台有 32 位微软 Windows 操作系统、大多数 Unix 或类 Unix 系统、苹果 Mac OS X。由于使用 Python 作为编程语言，WxPython 编写简单、易于理解。

基本的 WxPython 程序说明了开发任一 WxPython 程序所必需的五个基本步骤：

⑴ 导入必需的 WxPython 包

⑵ 子类化 WxPython 应用程序类

⑶ 定义一个应用程序的初始化方法

⑷ 创建一个应用程序类的实例

⑸ 进入这个应用程序的主事件循环

对于初学者或者对程序结构要求不多的用户来说，使用 Boa Constructor 这样的基于 WxPython 的开发平台，可以轻而易举地开发出优秀的 WxPython 程序。

3 Tkinter

Tkinter 是一个 Python 模块，是一个调用 Tcl/Tk 的接口，它是一个跨平台的脚本图形界面接口。Tkinter 不是唯一的 Python 图形编程接口，却是其中比较流行的一个。最大的特点是跨平台，缺点是性能不太好，执行速度慢。

一般使用 Tkinter 的方法是：

From Tkinter import *

或者：import Tkinter

两者的区别我们前面讲模块的时候已经说过了。

先看一下 GUI 程序的开发，熟悉 MFC 的朋友应该不会陌生。在 GUI 程序中，我们会有一个顶层窗口，在这个顶层窗口上可以包括所有的小窗口对象，像标签、按钮、列表框等等，也就是说顶层窗口是我们放置其他窗口或者控件的地方。我们用下面的语句可以创建一个顶层窗口，或者叫根窗口：

```
Import Tkinter
top = Tkinter.Tk()
```

（如果前面是用的 from Tkinter import * ，那么 Tk()就够了）

然后我们就可以在这个根窗口上设置“组件”了。通常这些组件会有一些相应的行为，比如鼠标点击、按下等等，这些称为事件，而程序会根据这些时间采取相应的反应，称为回调。这个过程称为事件驱动。

所有的创建和放置完毕后，就立刻进入主循环，代码如下：

```
Tkinter.mainloop( )
```

Tk 的组件有很多，不可能一一介绍，现在通过一个小例子看看其中一个标签的使用。

```
>>> import Tkinter
>>> top = Tkinter.Tk()
>>> label = Tkinter.Label(top,text='Hello World')
>>> label.pack()
>>> Tkinter.mainloop()
```

下面解释一下：

第一行，是导入模块。

第二行，创建主窗口。

第三行，创建 Label 标签，它是用 Tkinter 的一个方法Label 来实现的，关于 Label 的帮助可以 help 一下。

第四行，pack（）是用来管理和显示组件的，它的参数我们以后再说。

第五行，mainloop（）进入主循环。剩下的事就是系统的了。

下面看看组件的配置。Tk 中的每一个组件都有很多 option，通过改变这些 option 可以改变组件的外观，比如显示的内容、颜色、大小、位置、事件处理函数等。

比如：w=label(root,text='hello',fg='red')

创建一个 w，第一个参数是它的 master widget，是 root，所有参数都是默认的。我们可以用默认的来创建，w.cget(option)得到一个 option 的值。同样可以用 w.config(option='')来设置某个参数的值。

Tkinter 的几何管理器：

熟悉 GUI 编程的人知道，放好每个组件是很繁琐的，不仅要调整自身大小，还要整合其他组件的相对位置。Tk 提供了三个管理器来帮助我们：Pack、Grid、Place。

（1）pack

Pack 使用很简单，就是 w.pack (option)。常用的 option 有：

Side 表示把组件放到哪一边，TOP（上），BOTTOM（下），LEFT，RIGHT。

Padx 和 pady 表示 parcel 的每一个边和组件的预留空间。

Ipadx 和 ipady 表示组件的每一个边和它包含的内容之间的预留空间。

Anchor 表示在 parcel 放置组件的方式，缺省时 CENTER。

（2）grid

使用方法和 pack 类似。

（3）place

精确地摆放一个组件的位置，一般不太用。

看一下代码：

```
from Tkinter import *  # 引入模块
#resize 函数是用来改变文字大小的，当进度条改变时调用。
def resize(ev=None):
label.config(font='Helvetica - %d bold' % scale.get())
#config 函数就是通过设置组件的参数来改变组件的，这里改变的是 font 字体大小
top=Tk()  # 主窗口
top.geometry ('600x400') # 设置了主窗口的初始大小 600x400
label=Label (top,text='Hello world!',font='Helvetica - 12 bold') # 设置标签字体的初始大小。
label.pack(fill=Y,expand=1)
#scale 创建进度条，设置
scale=Scale (top,from_=10,to=40,orient=HORIZONTAL, command=resize)
scale.set(12) # 设置起始位置
scale.pack(fill=X,expand=1)
quit = Button(top,text='QUIT',command=top.quit,activeforeground='white',
activebackground='red')
quit.pack()
mainloop()
```


参考文献

[1] Magnus Lie Hetland.Python 基础教程[M].司维,译.北京:人民邮电出版社,2010.7.

[2] Mark Lutz.Python 学习手册[M].南京:机械工业出版社,2006.

[3] (美)马特利.Python Cookbook（第 2 版）中文版[M].高铁军,译.北京:人民邮电出版社,2010(5).

编辑 李叶亚