

Bevezetés a Tkinter használatába (3. rész)

Sorozatunk e részében a Tkinterhez létező nagyszerű kiegészítő modullal foglalkozunk, amelynek segítségével az alkalmazásaink írásához szükséges időt még jobban lerövidíthetjük.

Ezzel a modullal még egyszerűbben hozhatunk létre párbeszédablakokat, menüket, gördítősávval ellátott ablakokat, és még egy sor egyéb elemet, mint eddig. A kiegészítés neve PMW, azaz Python Megawidgets. A PMW-t teljes egészében Pythonban írták, és a Tkinterhez hasonlóan teljesen felületfüggetlen. Az elkövetkezőkben néhány gyakran használt PMW-s elemmel fogunk megismerkedni. Ha egy programban ki szeretnénk használni a PMW képességeit, első lépésben a PMW-modult importálnunk kell a következő utasítással:

```
import Pmw
```

```

1. lista A balloon.py
1 from Tkinter import *
2 import Pmw
3
4 root = Tk()
5
6 status = Pmw.MessageBar(root,
7     ↪entry_relief=GROOVE, labelpos=W,
8     ↪label_text='Sæg :')
9 status.pack(fill=X, expand=YES,
10    ↪side=BOTTOM)
11
12 balloon = Pmw.Balloon(root)
13 balloon.configure
14    ↪(statuscommand=status.helpmessage)
15
16 start = Button(root, text='Start')
17 start.pack()
18
19 balloon.bind(start,
20    ↪"Program ind tÆsa",
21    ↪"A program elind tÆsa egy æj ablakban")
22
23 root.mainloop()

```

Ezt követően a PMW elemekre a `Pmw` elemneve objektummal hivatkozhatunk.

A PMW-t a *Kapcsolódó címek* részben megadott címről tölthetjük le, illetve Debian esetén az `apt-get install python-pmw` parancs kiadásával.

Helyzetérzékeny sűgőrendszer – Balloon

Már a legegyszerűbb alkalmazások készítése során is felmerül a kérdés, hogy vajon mindenki ugyanolyan könnyedséggel használja-e majd a programot, mint mi magunk. Ha a leírás

```

2. lista A combobox.py
1 from Tkinter import *
2 import Pmw
3
4 def changeChoice(item):
5     choice.configure(text=item)
6
7 root = Tk()
8
9 menuitems = ('első elem', 'második elem',
10    ↪'harmadik elem')
11
12 choice = Label(root, text='VÆlassz!')
13 choice.pack(side=TOP, expand=YES,
14    ↪fill=BOTH)
15
16 items = Pmw.ComboBox(root, label_text=
17    ↪'Elemek:', labelpos=W, dropdown=1,
18    ↪selectioncommand=changeChoice,
19    ↪scrolledlist_items=menuitems)
20 items.pack(expand=YES, fill=X)
21
22 root.mainloop()

```

készítésén már túl vagyunk, okos ötlet lehet helyzetérzékeny sűgőt készíteni az alkalmazás felületén található ikonokhoz, gombokhoz, legördülő listákhoz és egyéb elemekhez. Ez annyit tesz, hogy amikor a felhasználó az egérrel megáll valamelyik gomb felett, akkor a gomb mellett egy kis mezőben, úgynevezett „lufiban” (buborékban) megjelenteszük a gombhoz tartozó tudnivalókat. Ilyen módon programunk kinézete átláthatóbbá tehető, mivel nem kell az ablakát magyarázó szövegekkel telezsűfolni – ezzel ahelyett, hogy megkönnyítenénk a használatát, inkább túlterheljük és a felhasználót is megijesztjük. Az *1. listában* látható, hogyan valósítható meg mindez a gyakorlatban.

A listában hivatkozom egy `Pmw.MessageBar` osztályra, amellyel a állapotjelzősor készíjtük el, erről az osztályról a későbbiekben még szót ejtünk.

A lufik elkészítése a `Pmw.Balloon` osztályával történik. Ez az osztály felelős a állapotsor működéséért is, melynek `bind()` eljárása rendeli a szövegeket az egyes elemekhez. A `bind()` első értéke annak az objektumnak a változója, amelyhez a szöveget rendeljük, második értéke a lufiban megjelenő szöveg, mely a gombon lévő szöveghez hasonlóan még elég tömör, illetve a harmadik értékben a állapotjelzősorban megjelenő szövegen módosíthatunk, ez ugyanis jóval hosszabb is lehet.

3. lista A dialog.py

```

1 from Tkinter import *
2 import Pmw
3
4 root = Tk()
5
6 info1 = Label(root, text=
↳ 'A dial gusablak mØg nem tØrt vissza')
7 info1.pack()
8
9 dialog = Pmw.Dialog(root,
↳ buttons=('Igen', 'Nem', 'MØgsem'),
↳ defaultbutton='Igen',
↳ title='Dial gusablak')
10 info2 = Label(dialog.interior(), text=
↳ 'Ez egy dial gusablak pØlda PMW-vel',
↳ pady=20)
11 info2.pack(fill=BOTH, expand=YES, padx=5,
↳ pady=5)
12 rv = dialog.activate()
13
14 info1.configure(text=rv)
15
16 root.mainloop()

```

4. lista Az entryfield.py

```

1 from Tkinter import *
2 import Pmw
3
4 root = Tk()
5
6 yourname = Pmw.EntryField(root,
↳ labelpos=W, label_text='Neved:')
7 yourname.pack()
8
9 email = Pmw.EntryField(root,
↳ labelpos=W, label_text='E-mail c med:')
10 email.pack()
11
12 Pmw.alignlabels((yourname, email))
13
14 root.mainloop()

```

Választás több elem közül – ComboBox

Ha több elem közül kell kiválasztani egyetlen elemet, nagyon helytakarékos és átlátható megoldást jelentenek a ComboBox-ok. A 2. listában egy egyszerű ComboBox-példa látható. A PMW-s ComboBox összetett elem, azaz sok egyéb PMW-s elemhez hasonlóan több elemből áll, jelen esetben kettőből: egy Label-ből és magából a ComboBox-ból. A lista 14. sorában látható, hogy e összevont elem belső elemeinek a tulajdonságait külön-külön is módosíthatjuk. A label_text tulajdonság megváltoztatásával például a beágyazott Label, azaz a címkeobjektum tulajdonságain változtathatunk. A labelpos értékkel határozhatjuk meg, hogy az összevont elem belső hol jelenjen meg.

5. lista A group.py

```

1 from Tkinter import *
2 import Pmw
3
4 root = Tk()
5
6 g1 = Pmw.Group(root, tag_text=
↳ 'Elso csoport')
7 g1.pack(expand=YES, fill=BOTH,
↳ padx=3, pady=3)
8 g1label = Label(g1.interior(),
↳ text='Az elso csoport tartalma')
9 g1label.pack(expand=YES, fill=BOTH,
↳ padx=5, pady=5)
10
11 g2 = Pmw.Group(root, tag_pyclass=None)
12 g2.pack(expand=YES, fill=BOTH,
↳ padx=3, pady=3)
13 g2label = Label(g2.interior(),
↳ text='A mØsodik csoport tartalma')
14 g2label.pack(expand=YES, fill=BOTH,
↳ padx=5,
↳ pady=5)
15
16 g3 = Pmw.Group(root, tag_pyclass=
↳ Checkbutton, tag_text='3. csoport')
17 g3.pack(expand=YES, fill=BOTH, padx=
↳ 3, pady=3)
18 g3label = Label(g3.interior(),
↳ text='A harmadik csoport tartalma')
19 g3label.pack(expand=YES, fill=BOTH,
↳ padx=5, pady=5)
20
21 root.mainloop()

```

Párbeszédablakok gyorsan – Dialog

Amennyiben a felhasználóval párbeszédablakon keresztül szeretnénk közölni valamit, a PMW Dialog osztályát felhasználva egyszerűen megtehetjük.

A párbeszédablak használatára látható példa a 3. listában. Ha a megjelenő párbeszédablakban a gombokon kívül egyéb elemet is el szeretnénk elhelyezni, a Dialog osztály interior() eljárásával hivatkozhatunk a párbeszédablak belsejére. Az ablak megjelenítéséért az activate() eljárás felelős, mely egyben a párbeszédablak visszatérési értéke is.

Szövegbevitel előre gyártott címkével – EntryField

Ha sokszor együttesen van szükségünk címkékre és beviteli mezőkre – például egy űrlap kitöltésénél –, jó szolgálatot tehet az EntryField osztály. Ez az osztály lényegében egy összevont Label és Entry elemet takar, kiegészítve néhány PMW-re jellemző szolgáltatással (például beviteli értékellenőrzés). Használata is éppen olyan egyszerű, mint az említett két elemé. Erre példát a 4. listában találhatunk.

Az alignlabels() eljárás segítségével az egymás alatti címkék és beviteli mezők egymás alá igazíthatók. Az eljárás értéke azoknak az elemeknek a listája, amelyeket egymáshoz kell igazítani. Az alignlabels után a kettős nyitó zárójel nem elírás, hanem azért szükséges, mert az eljárás egyetlen listaértéket vár. A feladatot így is megoldhattuk volna:

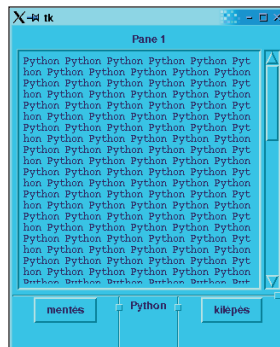
```
widgets = (yourname, email)
Pmw.alignlabels(widgets)
```

Az elemek csoportosítása – Group

Ezzel az összetevővel az ablakban található elemek csoportosítására nyílik lehetőség, így téve az ablak tartalmát még átlá-



Tagolt munkafelület a „Group”-elemmel



Egy „PanedWidget”-re épülő alkalmazás



A „NoteBook”-elem

eljárás első értéke a menüelem gazdájának nevét adja meg, a második pedig azt, hogy milyen típusú elemről van szó: elválasztóelemről vagy rendes szöveges menüelemről.

Státuszsor készítése – MessageBar

Ezzel az elemmel már találkoztunk néhányszor az eddigiek során. A MessageBar elsődleges feladata, hogy programunk számára állapotsorként működjön. Az elemeknek eddig a helpmessage() nevű eljárását használtuk, mely egyenértékű a message(help, szveg) eljárással, azonban ennek a rövidítésnek köszönhetően visszahívófüggvényként egyszerűbben használhatjuk. Mint az gyanítható volt, nemcsak help-típusú üzenetek küldése lehetséges a állapotsorra, hanem léteznek még a userevent, busy, systemevent, usererror, systemerror üzenettípusok is, amelyek elsőbbségi sorrendje a felsorolással megegyező irányban növekszik. Ha nagyobb elsőbbségű üzenet érkezik és az előző üzenet megjelenítésére szánt idő még nem járt le, a magasabb

elsőbbségű üzenet akkor is megjelenik – eltüntetve az előző, kevésbé fontos üzenetet.

elsőbbeségű üzenet akkor is megjelenik – eltüntetve az előző, kevésbé fontos üzenetet.

Menük készítése – MenuBar

Ezzel az elemmel néhány egyszerű utasítás segítségével programjainkhoz bonyolult, többszintű menüszerkezetet készíthetünk. Az elemet bemutató példaprogram a 6. listában (26. CD Magazin/Tkinter könyvtárban) található. Az addmenu() eljárás első értéke a menü nevét adja meg, míg a második a állapotsor használata esetén szükséges. Az itt megadott szöveg – ha az egérrel megállunk a menüsor adott eleme felett – megjelenik a állapotsoron. Az addmenumitem()

Beviteli mezők tagolása – Notebook

A Notebook a Group elemhez hasonló szerepet tölt be, tehát segít a program űrlapjainak a ésszerű tagolásában, de a Group-pal ellentétben itt egyszerre csak az egy csoporthoz tartozó elemek látszanak. Az egyes csoportokat a fülekre kattintva érhetjük el. Ha a felhasználónak nagy mennyiségű vagy eltérő feladatú elemekbe kell adatot bevinnie, ajánlott a Notebook felhasználása, mert így elkerülhetjük, hogy a felhasználót felesleges adatokkal árásszuk el. Egy egyszerű Notebook-példa látható a 7. listában (26. CD Magazin/Tkinter könyvtár).

A Notebook-hoz a füleket az add() eljárással adhatjuk hozzá, amely az adott fül belterületére mutató változóval tér vissza.

Munkaterület felosztása – PanedWidget, ScrolledText

A PanedWidget elem segítségével munkaterületünk egyes részeit oly módon oszthatjuk fel, hogy a felosztáson a felhasználó is változtathasson. Segítségével olyan jól alakítható felületek hozhatók létre, mint amilyen például az egyes levelezőprogramokban is látható.

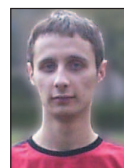
A ScrolledText egyszerű szövegbeviteli mezőt takar görgetőkkel ellátva, amelyet példánkban egy PanedWidget-elemmel vonunk össze. A példa a 8. listában (26. CD Magazin/Tkinter könyvtárban) található.

A Notebook-hoz hasonlóan új munkafelületet a PanedWidget-hez is az add() eljárással adhatunk hozzá, és a visszatérési értéke használható arra, hogy az adott munkafelületen belül újabb elemeket hozzunk létre.

Összegzés

Az eddig tanult felhasználásával most már mi magunk is készíthetünk egyszerű Tkinter- vagy PMW-alkalmazásokat. Mindemellet az egyes elemek alapos megismeréséhez olykor szükséges lehet fellapozni egy-egy elektronikus kézikönyvet. A PMW-hez és a Tkinterhez egyaránt nagyon jó, részletekbe menő leírás érhető el az Interneten, címüket a *Kapcsolódó címek*-nél találod meg.

A cikkhez kapcsolódó további kiegészítések a 26. CD Magazin/Tkinter könyvtárban találhatóak.



Gludovátz Gábor

(ggabor@sopron.hu)

1996 óta foglalkozik Linux-rendszerekkel.

Egyik kedvenc időtöltése a programozás, jelenleg éppen egy C++-ban írt KDE-s játékon dolgozik, de szívesen kódol Pythonban és

PHP-ben is. Honlapja ➔ <http://www.sopron.hu/~ggabor/>

Kapcsolódó címek

A Python honlapja ➔ <http://www.python.org/>

Tkinter-tanfolyam

➔ <http://www.pythonware.com/library/tkinter/introduction/>

A PMW honlapja, innen tölthető le a leírás és a modul is

➔ <http://www.dscpl.com.au/pmw/>

A PMW FTP-címe ➔ <ftp://ftp.dscpl.com.au/pub/pmw>