

Membuat Teks Editor dengan wxPython

wxPython telah diakui sebagai *toolkit* GUI standar untuk pemrograman Python. Di samping itu, Python sendiri merupakan bahasa yang sangat *powerful*. Dengan menggabungkan keduanya, tentu saja Anda akan lebih produktif.

Membuat sebuah aplikasi jadi, sebenarnya bukanlah merupakan sesuatu yang sulit. Mungkin selama ini, Anda memiliki segudang ide untuk membuat sebuah aplikasi, akan tetapi kesulitan dalam menuangkannya ke dalam bentuk kode program. Penulis akan membahas pembuatan aplikasi jadi dengan wxPython.

Artikel ini sebenarnya merupakan lanjutan dari artikel sebelumnya tentang wxPython. Mungkin sebagian besar dari Anda yang telah membaca artikel sebelumnya bertanya-tanya apa *sih* yang bisa dilakukan dengan kontrol-kontrol tersebut. Jawabnya banyak sekali. Penulis menyadari hal ini, oleh karena itu penulis akan merancang suatu aplikasi berupa sebuah teks editor dengan menggunakan kontrol-kontrol dasar yang telah Anda pelajari pada artikel sebelumnya.

Penulis memilih teks editor karena teks editor merupakan aplikasi yang paling sering digunakan baik untuk menyunting naskah, membaca dokumen, membuat laporan, dan sebagainya. Di lingkungan Linux, aplikasi teks editor sangatlah banyak dan bervariasi dari yang paling sederhana sampai yang kompleks seperti Joe, KWrite, GEdit, VIM, XEmacs, KWord dan masih banyak lagi.

Dalam hal ini, penulis bermaksud memberikan suatu contoh aplikasi jadi yang konkrit (nyata) yang dapat dibuat dengan wxPython. Tentu saja teks editor yang akan dibahas tidak dibuat untuk menggantikan teks editor bawaan Linux. Tapi jika Anda memang berminat, Anda dapat mengembangkannya lebih lanjut.

Untuk membuat teks editor ini, diperlukan pengetahuan tentang Python dan wxPython (baca artikel sebelumnya: Python dan wxPython), tentu saja ditambah dengan

logika yang baik. Dengan kontrol-kontrol dasar yang telah dipelajari, Anda telah dapat membuat sebuah teks editor sederhana. Ada satu kontrol tambahan yang dipakai dalam membuat aplikasi teks editor ini, yakni:

wxFileDialog()

Kontrol ini berupa sebuah dialog box yang dilengkapi file browser.

Format sintaks kontrol ini adalah:

```
wxFileDialog(parent,message,defaultDir,
defaultFile,wildcard,style,pos)
```

Parent	Frame induk
message	Pesan yang akan ditampilkan berupa judul pada dialog box.
defaultdir	Direktori default, atau string kosong.
defaultfile	File default, atau string kosong.
wildcard	Seperti BMP Files (*.bmp) *.bmp All Files (*.*) *.*
Style	Style dialog box.

Jenis style	Keterangan
wxOPEN	berupa dialog box Open File.
wxSAVE	berupa dialog box Save File.
wxMULTIPLE	berupa dialog box Open File di mana Anda dapat memilih lebih dari satu file.
pos	posisi dialog box.

Contoh:

```
...
dlg = wxFileDialog(self,"Choose a file to
open","", "", "All files (*.*)|*.*",wxOPEN)
dlg.ShowModal()
self.filename = dlg.GetFilename()
self.dirname = dlg.GetDirectory()
...
```



Kode di atas akan menampilkan sebuah dialog box untuk membuka file. Kita juga dapat memperoleh data nama file dan nama direktori melalui metode `GetFilename()` dan `GetDirectory()`.

Berikut ini adalah uraian kode program teks editor yang akan kita buat. Kita akan membahas setiap bagian yang penting saja.

```
"""Text Editor v1.0
Author : Eric
"""
from wxPython.wx import *
import os

ID_NEW = 101
ID_OPEN = 102
ID_SAVE = 103
ID_SAVE_AS = 104
ID_EXIT = 105
ID_ABOUT = 106
ID_WORDCOUNT = 107

class main_window(wxFrame):
    def __init__(self,parent,id,title):
        wxFrame.__init__(self,parent,-1,title)

        self.filename = ""
        self.dirname = ""
        self.documentChanged = 0

        self.textbox = wxTextCtrl(self,5,"",
style=wxTE_MULTILINE)
        EVT_TEXT(self,5,self.OnChange)

        self.CreateStatusBar()

        menufile = wxMenu()
        menufile.Append(ID_NEW,"&New\
tCtrl+N","Create new document")
        menufile.Append(ID_OPEN,"&Open\
tCtrl+O","Open a file")
        menufile.Append(ID_SAVE,"&Save\
```

```
tCtrl+S","Save the current document")
menufile.Append(ID_SAVE_AS,"Save
&As...", "Save the current document
with a different name")
menufile.AppendSeparator()
menufile.Append(ID_WORDCOUNT,
"&Word Count", "Count the words")
menufile.AppendSeparator()
menufile.Append(ID_EXIT,
"E&xit\tAlt+X", "Exit the program")
menuhelp = wxMenu()
menuhelp.Append(ID_ABOUT, "&About")

menubar = wxMenuBar()
menubar.Append(menufile, "&File")
menubar.Append(menuhelp, "&Help")

self.SetMenuBar(menubar)

EVT_MENU(self, ID_NEW, self.
OnFileNew)
EVT_MENU(self, ID_OPEN, self.
OnFileOpen)
EVT_MENU(self, ID_SAVE, self.
OnFileSave)
EVT_MENU(self, ID_SAVE_AS, self.
OnFileSaveAs)
EVT_MENU(self, ID_WORDCOUNT, self.
OnFileWordCount)
EVT_MENU(self, ID_EXIT, self.
OnFileExit)
EVT_MENU(self, ID_ABOUT, self.
OnHelpAbout)

EVT_CLOSE(self, self.OnClose)
```

Pertama-tama kita membuat sebuah kelas `main_window` yang diturunkan dari kelas `wxFrame`. Kemudian meng-customize atribut-atribut dari kelas `main_window` yang baru kita buat dalam konstruktor `__init__`. Di dalam konstruktor `__init__`, kita mendeklarasikan atribut-atribut yang diperlukan dan membuat kontrol-kontrol yang hendak ditempatkan ke dalamnya serta memproses event-event dari kontrol-kontrol tersebut. Atribut `filename` dan `dirname` kita isi dengan *string* kosong karena pada awalnya program dijalankan belum ada file yang terbentuk. Sementara atribut `documentChanged` merupakan indikator apakah isi dokumen berubah atau tidak, pada awal program belum ada dokumen yang berubah

sehingga kita isi dengan nilai `False (0)`. Selanjutnya kita membuat berbagai kontrol seperti menu, text control, dan status bar. Untuk shortcut pada menu, Anda tinggal menuliskan kombinasi tombol apa yang diinginkan. Dan terakhir kita membuat event handling untuk kontrol-kontrol tersebut.

```
def OnChange(self, evt):
    self.documentChanged = 1
    if self.filename == "":
        filename = "Untitled"
    else:
        filename = self.filename
    self.SetTitle('*' + filename + '*' + '-Text
Editor')
```

Jika terjadi perubahan teks pada isi textbox, maka akan dipanggil fungsi `OnChange()` di mana kita membuat suatu indikator bahwa isi dokumen telah berubah dengan mengubah isi atribut `documentChanged` menjadi bernilai `True (1)` dan mengubah judul window (frame) untuk menandakan bahwa isi dokumen telah berubah.

```
def saveChanges(self):
    if self.filename == "":
        filename = "Untitled"
    else:
        filename = self.filename
    msg = "The text in the %s file has
changed.\n\nDo you want to save the
changes?" % filename
    dlg = wxMessageDialog(self, msg, 'Text
Editor', wxICON_EXCLAMATION |
wxYES_NO | wxCANCEL)
    ret = dlg.ShowModal()
    dlg.Destroy()
    return ret
```

Fungsi `saveChanges()` di atas berfungsi untuk menampilkan kotak pesan untuk mengkonfirmasi bahwa isi dokumen telah berubah dan apakah Anda ingin menyimpan perubahan tersebut. Fungsi ini mengembalikan nilai tombol yang kita pilih, yakni apakah tombol *Yes*, *No*, atau *Cancel*.

```
def newFile(self):
    self.textbox.SetValue("")
    self.documentChanged = 0
    self.filename = ""
    self.dirname = ""
    self.SetTitle("Untitled-Text Editor")
```

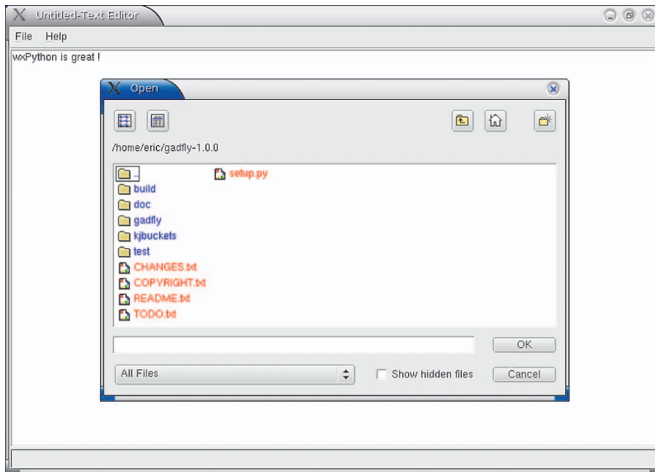
```
def OnFileNew(self, evt):
    if self.documentChanged:
        save = self.saveChanges()
        if save == wxID_CANCEL:
            pass
        elif save == wxID_NO:
            self.newFile()
        else:
            if self.filename == "" and self.
dirname == "":
                if self.OnFileSaveAs(None):
                    self.newFile()
            else:
                self.OnFileSave(None)
                self.newFile()
        else:
            self.newFile()
```

Kedua fungsi di atas merupakan fungsi untuk membuat file baru. Ketika user mengklik menu *File|New*, maka akan dijalankan fungsi `OnFileNew()`.

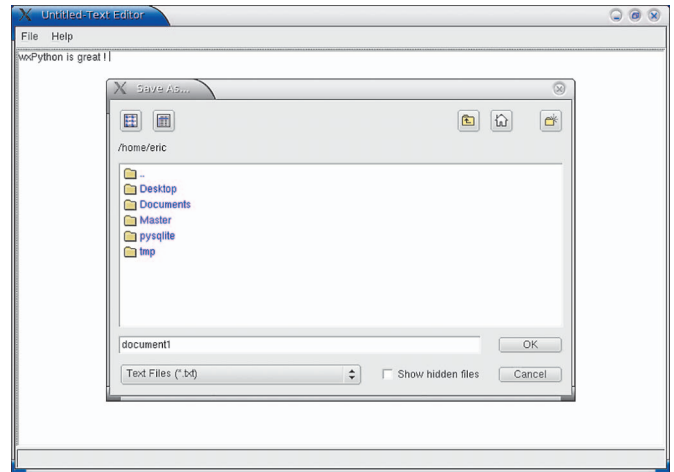
Dalam fungsi `OnFileNew()`, kita mengecek apakah isi dokumen telah berubah atau tidak. Jika isi dokumen telah berubah, maka kita memanggil fungsi `saveChanges()` di atas untuk mengkonfirmasi apakah dokumen yang telah berubah tersebut mau disimpan atau tidak. Seandainya jika kita tidak mau menyimpan perubahan tersebut, maka akan langsung dibuat file baru dengan memanggil fungsi `newFile()`. Apabila pada awalnya isi dokumen tidak berubah, maka akan dipanggil langsung fungsi `newFile()` tanpa konfirmasi lagi.

Fungsi `newFile()` sendiri berfungsi mengosongkan isi textbox, mengatur variabel `filename` dan `dirname` menjadi `None` serta mengubah judul window (frame).

```
def openFile(self):
    dlg = wxFileDialog(self, "Open", self.
dirname, self.filename, "TextFiles(*.
txt)|*.txt|All Files|*.*", wxOPEN)
    if (dlg.ShowModal() == wxID_OK):
        self.filename = dlg.GetFilename()
        self.dirname = dlg.GetDirectory()
    try:
        f = file(os.path.join(self.dirname, self.
filename), 'r')
        self.textbox.SetValue(f.read())
        self.documentChanged = 0
        self.SetTitle(self.filename + '-Text
Editor')
```



▲ Penggunaan Kotak dialog Open.



▲ Penggunaan Kotak dialog Save As.

```
self.PushStatusText(self.filename + '
opened.')
f.close()
except:
    pass
dlg.Destroy()
```

Fungsi `openFile()` berperan dalam operasi pembukaan file. Pertama akan muncul file dialog box yang meminta input file mana yang akan dibuka. Jika user mengklik tombol OK, maka akan dilakukan operasi pembacaan file melalui metode `read()` kemudian hasilnya dicetak di textbox. Berikutnya kita mengubah judul window(frame) menjadi nama file aktif. Sesudah operasi pembacaan file, lazimnya file ditutup dengan metode `close()`.

```
def OnFileOpen(self, evt):
    if self.documentChanged:
        save = self.saveChanges()
        if save == wxID_CANCEL:
            return
        elif save == wxID_NO:
            self.openFile()
        else:
            if self.filename == "" and self.
                dirname == "":
                if self.OnFileSaveAs(None):
                    self.openFile()
            else:
                return
        else:
            self.saveFile(self.dirname, self.
                filename)
            self.openFile()
```

```
else:
    self.openFile()
```

Dalam operasi pembukaan file, kita akan berhubungan dengan fungsi `openFile()` dan `OnFileOpen()`. Pada prinsipnya, saat kita mengklik menu `File|Open` maka akan dipanggil fungsi `OnFileOpen()` yang akan mengecek apakah isi dokumen telah berubah atau tidak. Jika isi dokumen mengalami perubahan, maka akan dipanggil fungsi `saveChanges()` yang mengonfirmasi melalui sebuah kotak pesan apakah ingin menyimpan perubahan tersebut atau tidak. Jika Anda tidak ingin menyimpan perubahan tersebut, selanjutnya akan dipanggil fungsi `openFile()` di atas. Tampilannya akan tampak seperti gambar Kotak dialog Open.

```
def saveFile(self, dirname, filename):
    try:
        path = os.path.join(dirname, filename)
        f = open(path, 'w')
        f.write(self.textbox.GetValue())
        f.close()
        self.documentChanged = 0
        self.SetTitle(filename + "-Text Editor")
        self.PushStatusText(filename + "
saved. ")
    except:
        pass
```

Fungsi di atas digunakan untuk operasi penyimpanan file. Yang dilakukan fungsi tersebut adalah menggabungkan `dirname`

dan `filename` menjadi path yang utuh, kemudian menciptakan sebuah objek file baru dengan tipe `writable`. Selanjutnya, kita membaca isi textbox melalui fungsi `GetValue()`, kemudian kita menuliskannya ke dalam file dengan metode `write()`. Dan atribut `documentChanged` kita ubah menjadi `False (0)` serta judul window(frame) kita ganti dengan nama file yang kita simpan tadi.

```
def OnFileSave(self, evt):
    if self.filename == "" and self.
        dirname == "":
        self.OnFileSaveAs(None)
    else:
        self.saveFile(self.dirname, self.
            filename)
```

Baris kode di atas akan diproses ketika user mengklik menu `File|Save`. Fungsi di atas akan memeriksa apakah file pernah disimpan sebelumnya dengan `filename` tertentu atau belum. Jika atribut `filename` dan `dirname` masih kosong berarti file belum pernah disimpan sebelumnya, maka akan dijalankan fungsi `OnFileSaveAs()`. Jika file telah pernah disimpan sebelumnya, maka akan langsung disimpan dengan menjalankan fungsi `saveFile()`.

```
def OnFileSaveAs(self, evt):
    dlg = wxFileDialog(self, "Save As...",
        self.dirname, self.filename, "Text Files
        (*.txt) | *.txt | All Files | *.*", wxSAVE)
    if (dlg.ShowModal() == wxID_OK):
        self.filename = dlg.GetFilename()
        self.dirname = dlg.GetDirectory()
        self.saveFile(self.dirname, self.
```

```

filename)
self.SetTitle(self.filename + '-Text
Editor')
dlg.Destroy()
return 1
else:
dlg.Destroy()
return 0
    
```

Fungsi OnFileSaveAs() akan dijalankan apabila user mengklik menu File|Save As. Fungsi ini akan menampilkan sebuah file dialogbox untuk menyimpan dokumen Anda dengan nama file yang baru. Jika Anda mengklik tombol OK, maka file akan disimpan dengan memanggil fungsi saveFile(), sesudahnya kita mengubah judul window (frame) menjadi nama file tersebut dan mengembalikan nilai True (1). Sebaliknya jika Anda mengklik Cancel maka kita akan kembali ke teks editor dan mengembalikan nilai False (0). Tampilannya dapat Anda lihat pada Kotak dialog Save As.

```

def OnFileWordCount(self,evt):
    text = self.textbox.GetValue()
    chars = len(text)
    words = len(text.split())
    lines = len(text.split('\n'))
    info = """Number of characters : %s
Number of words : %s
Number of lines : %s""" %(chars,words,
lines)
    dlg = wxMessageDialog(self,info,"Word
Count",wxOK|wxICON_INFORMATION)
    dlg.ShowModal()
    dlg.Destroy()
    
```

Fungsi di atas berguna untuk menghitung jumlah karakter, kata dan baris pada dokumen yang sedang aktif. Pertama kita mengambil isi textbox dengan fungsi

GetValue(), dengan demikian kita dapat mengetahui banyaknya karakter dengan menggunakan fungsi len().

Selanjutnya kita menggunakan metode split() untuk memisahkannya berdasarkan spasi sehingga didapatkan jumlah kata. Untuk mendapatkan jumlah baris, kita memanggil metode split() dengan pemisah berupa karakter '\n' (Enter). Setelah itu, kita membuat kotak pesan untuk menampilkan informasi tersebut. Tampilannya akan tampak seperti gambar Kotak dialog WordCount.

```

def doExit(self):
    if self.documentChanged:
        save = self.saveChanges()
        if save == wxID_CANCEL:
            return 0
        elif save == wxID_NO:
            return 1
        else:
            if self.filename == "" and self.
dirname == "":
                return self.OnFileSaveAs(None)
            else:
                self.saveFile(self.dirname,self.
filename)
            return 1
        else:
            return 1

def OnClose(self,evt):
    if self.doExit():
        evt.Skip()

def OnFileExit(self,evt):
    self.Close()
    
```

Untuk menangani proses keluar dari program, kita membuat tiga buah fungsi. Pada saat user mengklik tombol X di window program akan dipanggil fungsi OnClose() yang melakukan pengecekan dengan memanggil fungsi doExit(). Fungsi doExit() melakukan pengecekan apakah isi dokumen telah berubah atau tidak. Jika tidak, maka akan langsung keluar dari program. Dan jika isi dokumen berubah maka akan dikonfirmasi untuk menyimpan dokumen tersebut sebelum akhirnya keluar dari program. Sama halnya pula jika user mengklik menu File|Exit, maka akan dilakukan fungsi Close() yang sama

dengan fungsi OnClose(). Sebenarnya saat kita membuat EVT_CLOSE di atas, fungsi Close() telah digantikan dengan fungsi yang kita buat yakni fungsi OnClose().

```

def OnHelpAbout(self,evt):
    dlg = wxMessageDialog(self, __doc__,
About",wxOK|wxICON_INFORMATION)
    dlg.ShowModal()
    dlg.Destroy()
    
```

Fungsi ini dijalankan apabila user mengklik menu Help|About, yakni dengan menampilkan dialog box yang berisi informasi tentang program itu sendiri.

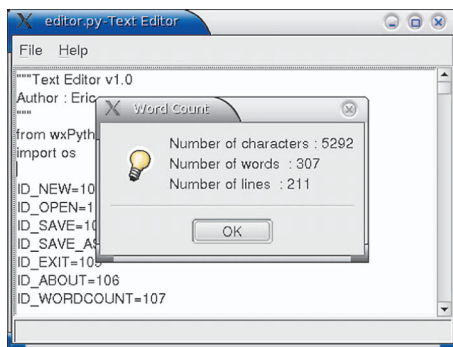
```

app = wxPySimpleApp()
frame = main_window(None,-1,"Untitled-
Text Editor")
frame.CenterOnScreen()
frame.Show(True)
app.MainLoop()
    
```

Berikutnya kita menginstansiasi objek app dengan kelas wxPySimpleApp, kemudian membuat objek frame yang diturunkan dari kelas main_window dan mengaktifkannya serta mengatur letaknya agar pas di tengah layar. Kemudian kita memanggil MainLoop untuk melakukan proses looping sampai program kita ditutup.

Program teks editor di atas masihlah sangat sederhana, mungkin Anda berniat menambahkan fitur-fitur tambahan seperti toolbar, menu Edit, Find and Replace Text, dan lain sebagainya. Semuanya dapat Anda lakukan hanya dengan menggunakan wxPython. Anda dapat mempelajari manual maupun demo yang disertakan dalam paket instalasi wxPython tentang bagaimana cara menggunakan menggunakan kontrol seperti toolbar, Find Dialog, dan masih banyak kontrol lainnya. Jika demonya belum terinstall, Anda dapat men-download-nya di www.wxpython.org.

Kini wxPython telah menjadi toolkit GUI standar untuk pemrograman Python. Berbagai aplikasi andal telah banyak yang dibuat menggunakan wxPython. Dengan contoh aplikasi di atas, kiranya cukup bagi Anda untuk menilai bagaimana kemampuan wxPython sendiri? Cukup hebat, bukan? Dengan wxPython, Anda tentunya akan lebih produktif dalam membuat aplikasi GUI di Linux. Selamat berkarya!
Eric (eric85id@plasa.com)



▲ Kotak dialog WordCount.