Linux Day 2007

by

PyGtk

come creare applicazioni

grafiche

in modo semplice e veloce.

perche' PyGtk

perche' e' un Free Software (LGPL license)

libero di essere:

usato

modificato

distribuito

studiato

by Orcim

un accenno alle LGPL

La licenza LGPL stabilisce il copyleft sul singolo file di codice sorgente non sull'intero software.

Questo ci permette di includere liberamente parti di applicazioni commerciali (closed-source) in un software libero.

In altre parole ci permette di rendere compatibili programmi liberi e non.

Applicazione con pyGtk

una applicazione che usa una interfaccia grafica
comunemente chiamata
GUI (Graphics User Interface)
prevede in pyGtk normalmente la seguente
Struttura:

- l'importazione delle librerie necessarie
- la creazione della finestra principale
- la creazione di un packing dove inserire i vari widgets
- la connessione dei widgets ai relativi signal handlers
- l'inserimento del packing nella finestra principale
- l'abilitazione della visualizzazione dei widgets
- l'avvio della procedura principale.

by Orcim

vediamo alcune **Terminologie**

che useremo nel corso della presentazione

- Widgets (oggetti)
- Packing (impacchettamento)
- Signals (segnali)
- Handlers (gestione segnali)
- Callbacks (procedure)
- Events (eventi)

by Orcim

Widget (oggetto)

in generale si intende un qualsiasi oggetto incluso nelle finestre:

es:

Menus, Toolbar, Text view, Status bar, etc.

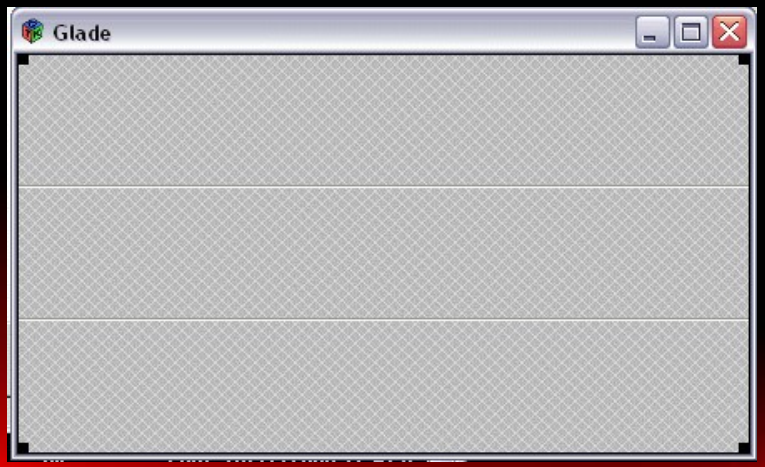
Packing (impacchettamento)

e' un contenitore che gestisce l' ordinamento degli oggetti al suo interno.

un packing puo' contenere altri packing per gestire aree suddivise.

esempi di packing:

horizontal Box, vertical Box, Tables, etc.



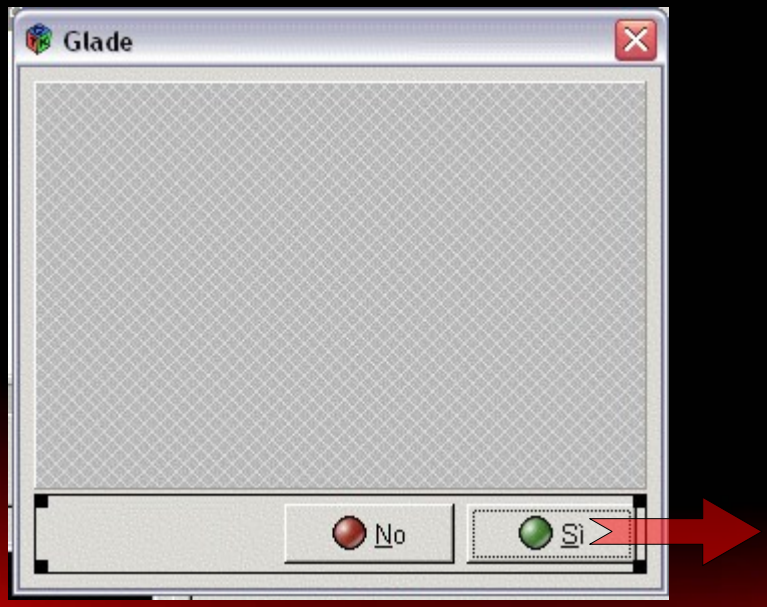
by Orcim

Signal (segnale)

e' un messaggio che viene emesso da un oggetto quando esegue qualcosa

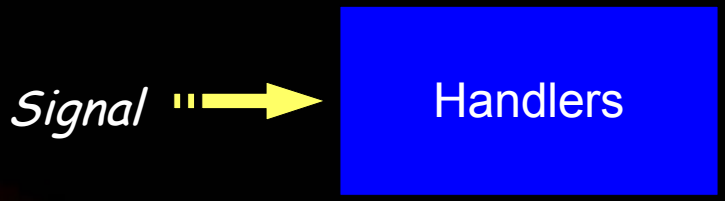
es:

clicked, destroy, toggled, etc.



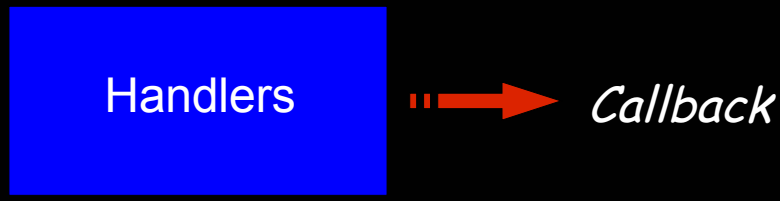
Handlers (gestore segnali)

sono quelle funzioni che attendono di identificare un segnale...



Handlers (gestore segnali)

per avviarne il metodo ad esso associato.



Callbacks (procedure)

sono le funzioni che vengono eseguite dal gestore in risposta al segnale ricevuto.

Event (evento)

e' un segnale NON generato da un Widget

es:

destroy_event, focus_in_event,

key_press_event, etc..

Analizziamo la nostra prima Applicazione:

*file: mia*Applicazione.py

```
#/usr/bin/env python
# -*- coding: utf-8 -*-
"""     multistringa di documentazione
    Class: MyWindow
    Metod:
      rev: 070928
"""
import gobject
import gtk
import pango

#----------------------------------------------------------------------
class MyWindow(object):
#----------------------------------------------------------------------
    def __init__(self):
    #
    # Attributs
        self.myFlg=True
    #
    # Window
        self.win = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.win.set_title("La mia prima applicazione")
        self.win.set_border_width(4)
        self.win.set_default_size(320,40)
```

Importiamo le Librerie:

```
#/usr/bin/env python
# -*- coding: utf-8 -*-
"""     multistringa di documentazione
    Class: MyWindow
    Metod:
      rev: 070928
"""
import gobject
import gtk
import pango

#----------------------------------------------------------------------
class MyWindow(object):
#----------------------------------------------------------------------
    def __init__(self):
    #
    # Attributs
        self.myFlg=True
    #
    # Window
        self.win = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.win.set_title("La mia prima applicazione")
        self.win.set_border_width(4)
        self.win.set_default_size(320,40)
```

Istanziamo l'oggetto principale la Window:

```
#/usr/bin/env python
# -*- coding: utf-8 -*-
"""     multistringa di documentazione
    Class: MyWindow
    Metod:
      rev: 070928
"""
import gobject
import gtk
import pango

#----------------------------------------------------------------------
class MyWindow(object):
#----------------------------------------------------------------------
    def __init__(self):
    #
    # Attributs
        self.myFlg=True
    #
    # Window
        self.win = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.win.set_title("La mia prima applicazione")
        self.win.set_border_width(4)
        self.win.set_default_size(320,40)

        # Signals
        self.win.connect("destroy", self.on_destroy)
        self.win.connect("delete_event", self.on_delete_event)
```


Associamo i Segnali alle Callbacks:

```
import gobject
import gtk
import pango
#-----------------------------------------------------------------------
class MyWindow(object):
#-----------------------------------------------------------------------
    def __init__(self):
    #
    # Attributs
        self.myFlg=True
    #
    # Window
        self.win = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.win.set_title("La mia prima applicazione")
        self.win.set_border_width(4)
        self.win.set_default_size(320,40)

        # Signals
        self.win.connect("destroy", self.on_destroy)
        self.win.connect("delete_event", self.on_delete_event)
        self.win.connect("hide", self.on_hide)

    #
    # Vbox
        # istanzio una vertical box
        #gtk.VBox(homogeneous=False, spacing=0)
        self.vbox = gtk.VBox(False, 0)
```


Mettiamo ordine nei widgets Packing:

```
#
# Window
    self.win = gtk.Window(gtk.WINDOW_TOPLEVEL)
    self.win.set_title("La mia prima applicazione")
    self.win.set_border_width(4)
    self.win.set_default_size(320,40)
    # Signals
    self.win.connect("destroy", self.on_destroy)
    self.win.connect("delete_event", self.on_delete_event)
    self.win.connect("hide", self.on_hide)
```

```
#
# Vbox
    # istanzio un vertical box
    #gtk.VBox(homogeneous=False, spacing=0)
    self.vbox = gtk.VBox(False, 0)
```

```
#
# Label
    self.lab = gtk.Label("<b><big>myFlg: %s</big></b>" %self.myFlg)
    self.lab.set_use_markup(True)
    # gli oggetti che non hanno associata una windows non possono
    #  esser modificati come background
    self.lab.modify_fg(gtk.STATE_NORMAL, gtk.gdk.color_parse("blue"))
#
# EventBox
    self.evb = gtk.EventBox()
```

by Orcim

Includiamo dei Widgets:

```
#
# Vbox
    # istanzio un vertical box
    #gtk.VBox(homogeneous=False, spacing=0)
    self.vbox = gtk.VBox(False, 0)

#
# Label
    self.lab = gtk.Label("<b><big>myFlg: %s</big></b>" %self.myFlg)
    self.lab.set_use_markup(True)
    # gli oggetti che non hanno associata una windows non possono
    #  esser modificati come background
    self.lab.modify_fg(gtk.STATE_NORMAL, gtk.gdk.color_parse("blue"))
#
# EventBox
    self.evb = gtk.EventBox()
    self.evb.add(self.lab)
    self.evb.modify_bg(gtk.STATE_NORMAL,gtk.gdk.color_parse("green"))
    # Signals
    self.evb.connect("button-release-event", self.on_clicked)
    #def pack_start(child, expand=True, fill=True, padding=0)
    self.vbox.pack_start(self.evb, False, False, 0)

#
# View
    self.win.add(self.vbox)
    self.win.show_all()
```


Rendiamo visibili gli Oggetti:

```
        #
        # EventBox
            self.evb = gtk.EventBox()
            self.evb.add(self.lab)
            self.evb.modify_bg(gtk.STATE_NORMAL,gtk.gdk.color_parse("green"))
            # Signals
            self.evb.connect("button-release-event", self.on_clicked)
            #def pack_start(child, expand=True, fill=True, padding=0)
            self.vbox.pack_start(self.evb, False, False, 0)

        #
        # View
            self.win.add(self.vbox)
            self.win.show_all()

    #--------------------------------------------------------------------
    # signal
    #--------------------------------------------------------------------
    #
    # Window
    def on_delete_event(self, widget, data=None):
        print "on_delete_event"
        return self.myFlg
    def on_hide(self, widget, data=None):
        print "on_hide"
    def on_destroy(self, widget, data=None):
        print "on_destroy"
        gtk.main_quit()
```

by Orcim

Definiamo le Callbacks: assiociate alla Window

```
        #
        # View
            self.win.add(self.vbox)
            self.win.show_all()
        #--------------------------------------------------------------
        # signal
        #--------------------------------------------------------------
        #

        # Window
        def on_delete_event(self, widget, data=None):
            print "on_delete_event"
            return self.myFlg
        def on_hide(self, widget, data=None):
            print "on_hide"
        def on_destroy(self, widget, data=None):
            print "on_destroy"
            gtk.main_quit()

        #
        # EventBox
        def on_clicked(self, widget, data=None):
            print "on_clicked"
            if self.myFlg:
                self.myFlg = False
            else:
                self.myFlg = True
```

by Orcim

Definiamo le Callbacks: assiociate all' EventBox

```
    # Window
    def on_delete_event(self, widget, data=None):
        print "on_delete_event"
        return self.myFlg
    def on_hide(self, widget, data=None):
        print "on_hide"
    def on_destroy(self, widget, data=None):
        print "on_destroy"
        gtk.main_quit()

    #
    # EventBox
    def on_clicked(self, widget, data=None):
        print "on_clicked"
        if self.myFlg:
            self.myFlg = False
        else:
            self.myFlg = True
        msg="<b><big>Flag di ritorno: %s</big></b>" %self.myFlg
        self.lab.set_text(msg)
        self.lab.set_use_markup(True)

#----------------------------------------------------------------------
if __name__ == "__main__":
    # istanzio l'oggetto
    MyWindow()
```

by Orcim

Definiamo lo Scripts:

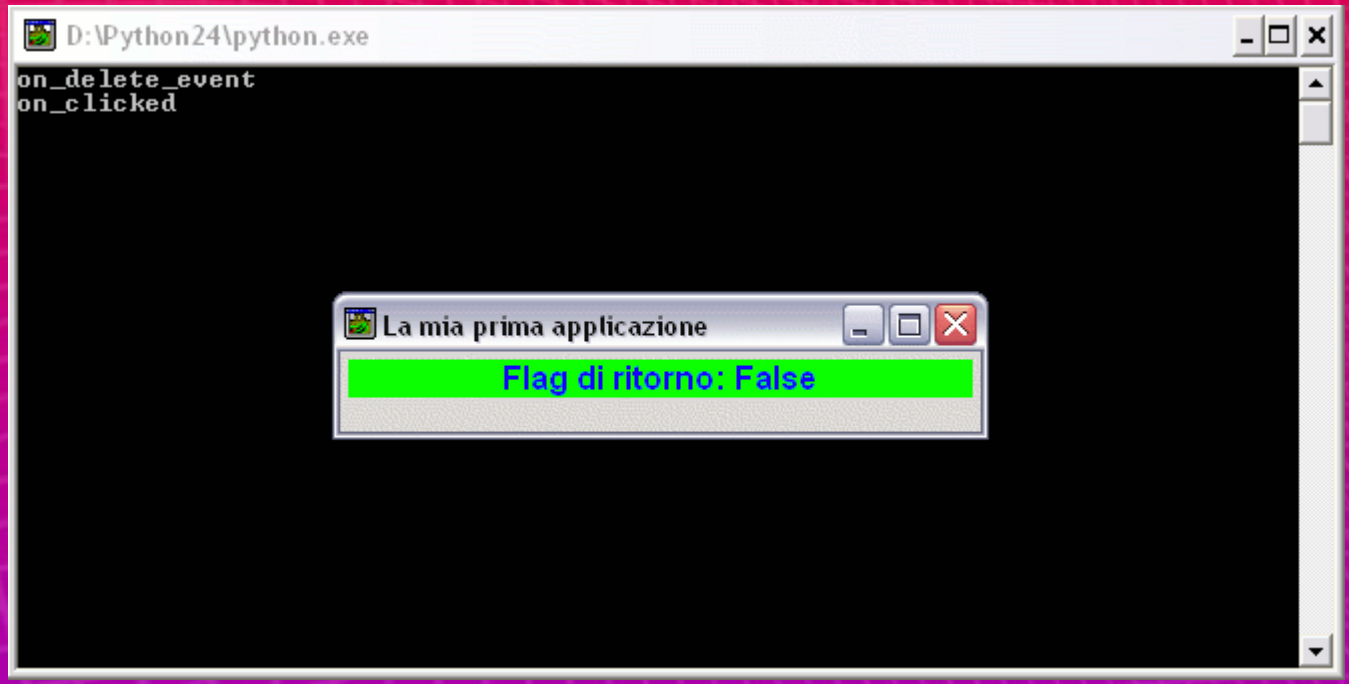
```
    #
    # EventBox
    def on_clicked(self, widget, data=None):
        print "on_clicked"
        if self.myFlg:
            self.myFlg = False
        else:
            self.myFlg = True
        msg="<b><big>Flag di ritorno: %s</big></b>" %self.myFlg
        self.lab.set_text(msg)
        self.lab.set_use_markup(True)
```

```
#----------------------------------------------------------------------
if __name__ == "__main__":
    # istanzio l'oggetto
    MyWindow()
    # avvio l'applicazione
    gtk.main()
```

Avviamo lo Scripts:

```
$ miaApplicazione.py [ENTER]
            o
$ python miaApplicazione.py [ENTER]
```

a seconda se abbiate definito o meno nel path dove si trova l'interprete.



PyGtk

Link utili:

http://www.pygtk.org/

http://live.gnome.org/PyGTK/FAQ

Domande ?