# Breve tutorial de introducción a la programación con python+glade.

Este texto se preparó para los cursos 2005 del hacklab Metabolik Bio Hacklab de Bilbao por Jon Latorre Martinez. En el se explica de manera breve y directa como empezar a crear aplicaciones gráficas con el lenguaje de programación python y el diseñador de interfaces GTK glade.

Cualquier comentario es bienvenido. Puedes encontrarme en la dirección de correo electrónico moebius@etxea.net. También suelo estar en el IRC, servidor irc.freenode.org canal #metabolik, nick moebius o moe_evil.

**Revisiones**:

- 2005/03/20: v0.1 Primera versión online.

# ¿ Porque python+gtk+glade ?

## *Python*

Python es el lenguaje de script de moda. Como dicen en su web es un lenguaje interpretado (no se compila, se interpreta al ejecutarse, por lo tanto trabajamos con ficheros de texto), interactivo (tenemos disponible una consola donde podemos programar y ejecutar en tiempo real y ver como responde) y orientado a objetos (de una manera sencilla que hace muy natural el utilizarlo). Una de sus mayores ventajas es que obliga a programar de manera “limpia” (la indentación es significativa). Esta claramente de moda y cada mes aparecen nuevos módulos, bindings, APIs, etc.

Su sintaxis clara y el que sea bastante potente permite desarrollar aplicaciones en tiempos muy cortos. Como desventaja está en consumo de recursos que implica. Es muy portable, ya que funciona en distintas plataformas: UNIX (GNU/Linux, *BSD,etc), en Windows, OS/2, Mac, Amiga...

## *GTK+*

GTK viene de The GIMP toolkit y como su nombre indica se creo para que lo usase GIMP. Un toolkit gráfico se encargase de “pintar” botones y otros controles necesarios para tener una interfaz gráfica de usuario.

Internamente tiene distintos componentes (Glib para tipos de datos, pango para mostrar letras en cualquier alfabeto, ATK para accesibilidad,..) y aunque en principio fue creado para usarlo desde C/C++ ahora mismo hay soporte (bindings) GTK para multitud de lenguajes (como perl, python, ruby e incluso php :)

Es multiplataforma (funciona en las distintas arquitecturas que soporta GNU/Linux y otros Unices libres BSD etc, MS Windows, MacOSX, etc) y es software libre (aunque licenciado bajo la LGPL)

## *Glade*

Glade es el diseñador de interfaces gráficas para GTK. La idea es poder definir de manera gráfica y rápida una interfaz gráfica. Luego veremos que además de controles (widgets) GTK podremos usar controles GNOME (aunque con esto perderemos portabilidad).

## *Otras opciones*

**Otros toolkits graficos:**

Tenemos disponibles desde python unos cuantos toolkits mas:

- QT
- xWindows
- TK

**Otros lenguajes**

- ruby
- perl
- php

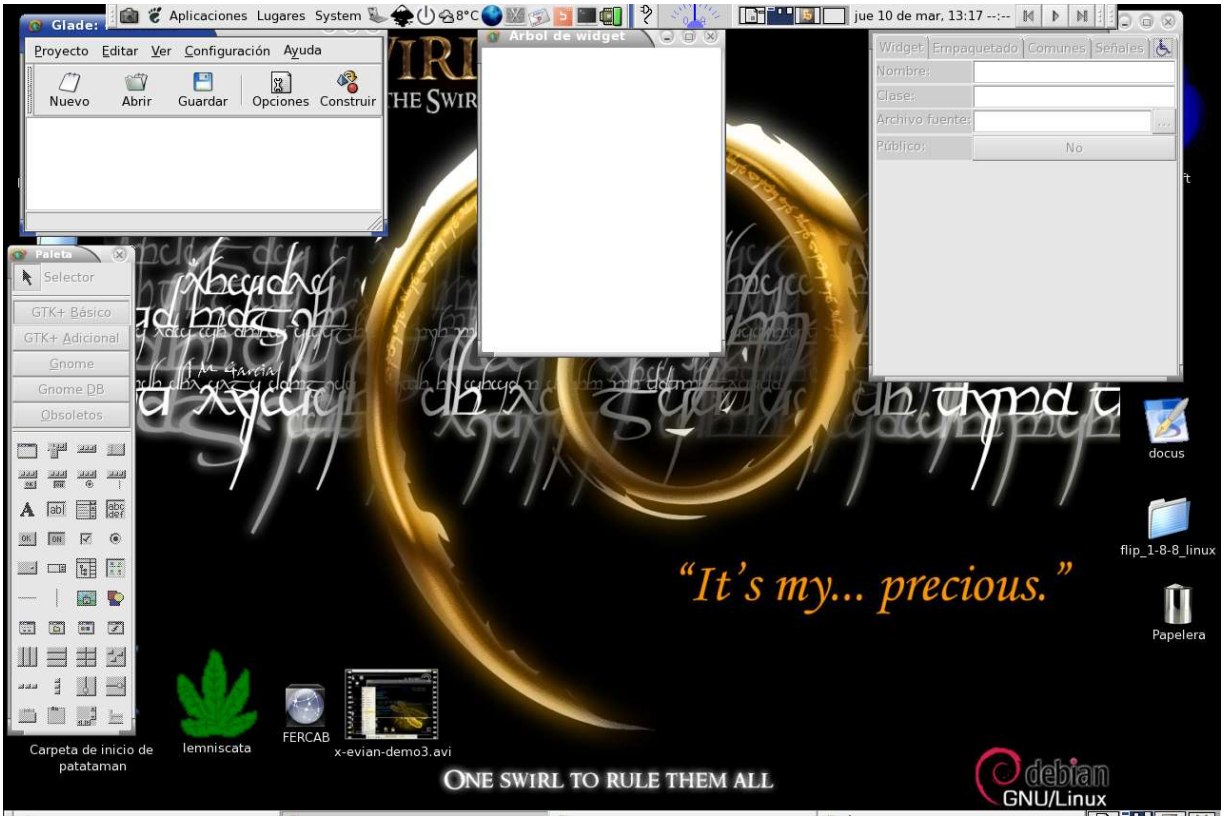
# Crear una aplicación paso a paso

Vamos a ver como crear una aplicación en 3 sencillos pasos. Crearemos una aplicación muy sencilla y por ello podemos seguir estos pasos en orden. Cuando creemos una aplicación mas compleja iremos hacia adelante y hacia atrás por estos pasos un montón de veces.

## *Paso 1: Crear la interfaz con glade*

### *Diseño de la GUI*

Lo primero será arrancar glade. Nos encontraremos que glade tiene varias ventanas:



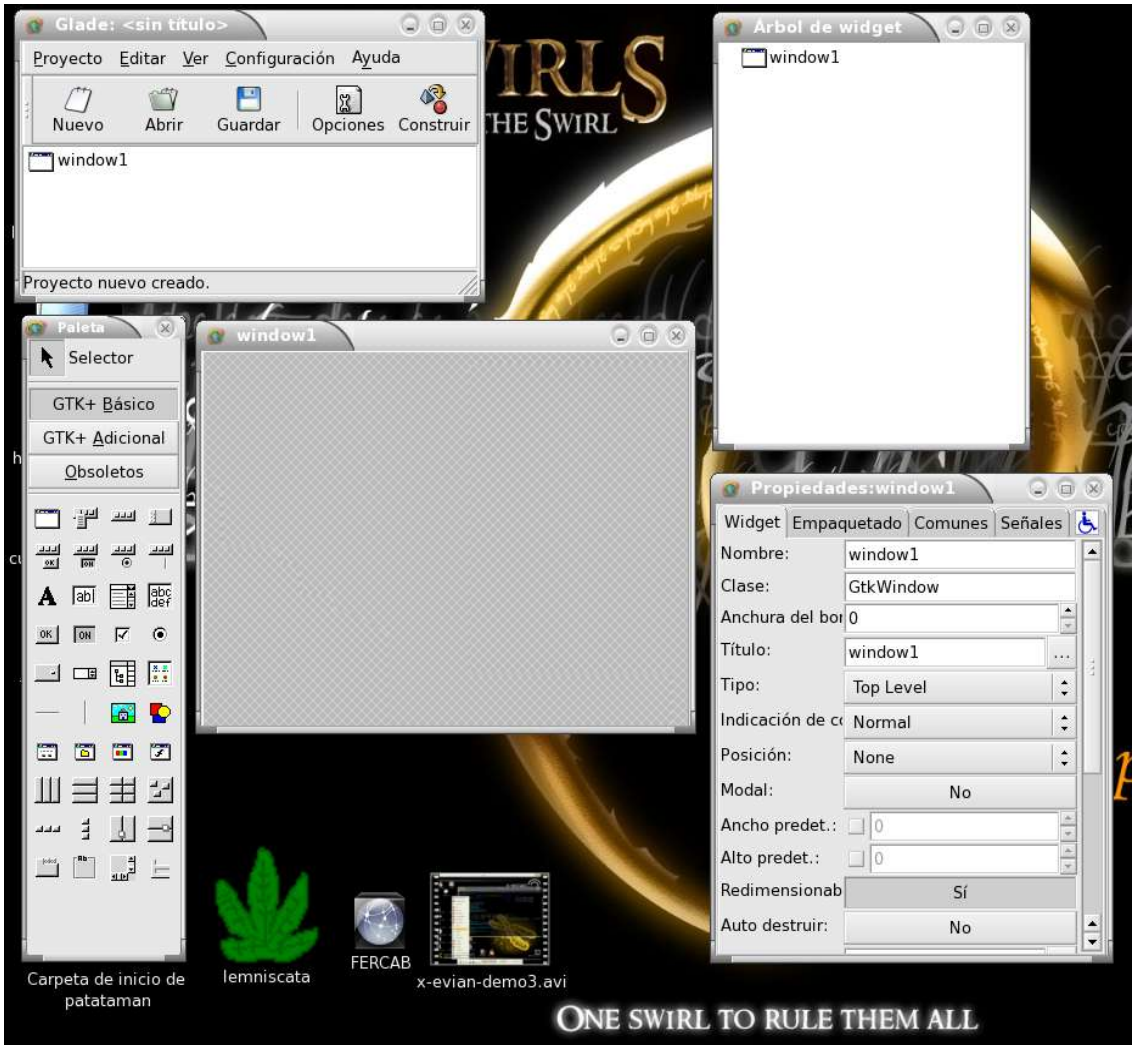

Si nos fijamos en la paleta todos los controles aparecen deshabilitados. Esto es por que no tenemos un proyecto abierto. Vamos a crear uno haciendo click en nuevo en la ventana principal de glade (arriba a la izquierda en la captura).

Nos preguntara el tipo de proyecto que deseamos. Si queremos que nuestra aplicación sea lo más portable posible tendremos que elegir GTK+ y renunciar a usar los controles específicos de GNOME.


Ahora la paleta nos muestra los controles que tenemos disponibles en 3 categorías. Como podemos ver las categorías son GTK+ Básico donde se encuentran la mayoría de los controles que usaremos. GTK+ Adicional donde hay algunos controles adicionales que no usaremos muy a menudo. Y por ultimo Obsoletos, que son controles GTK que se mantienen por compatibilidad pero que es posible que desaparezcan.

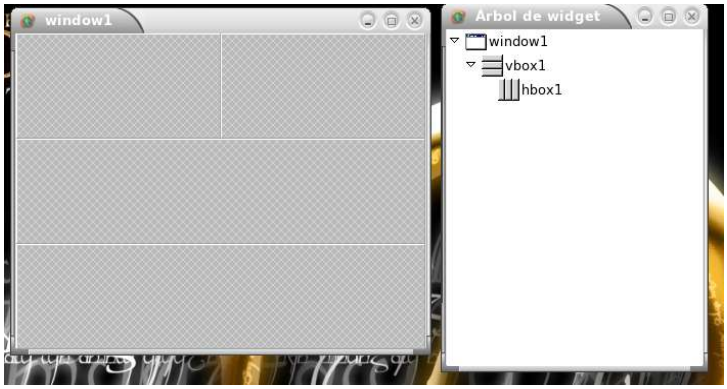
Si hubiésemos elegido un proyecto de tipo GNOME tendríamos una categoría adicional con los controles de GNOME. Entonces haciendo uso de los controles BONOBO podríamos cargar cualquier componente de GNOME en nuestra aplicación.

Una vez que hemos creado un nuevo proyecto de tipo GTK el primer paso será crear una ventana de aplicación. Para ello hacemos click en el primero icono (el superior izquierdo) de la paleta de glade (que se llama Ventana). Esto nos creara una ventana que será la ventana donde coloquemos el resto de controles.



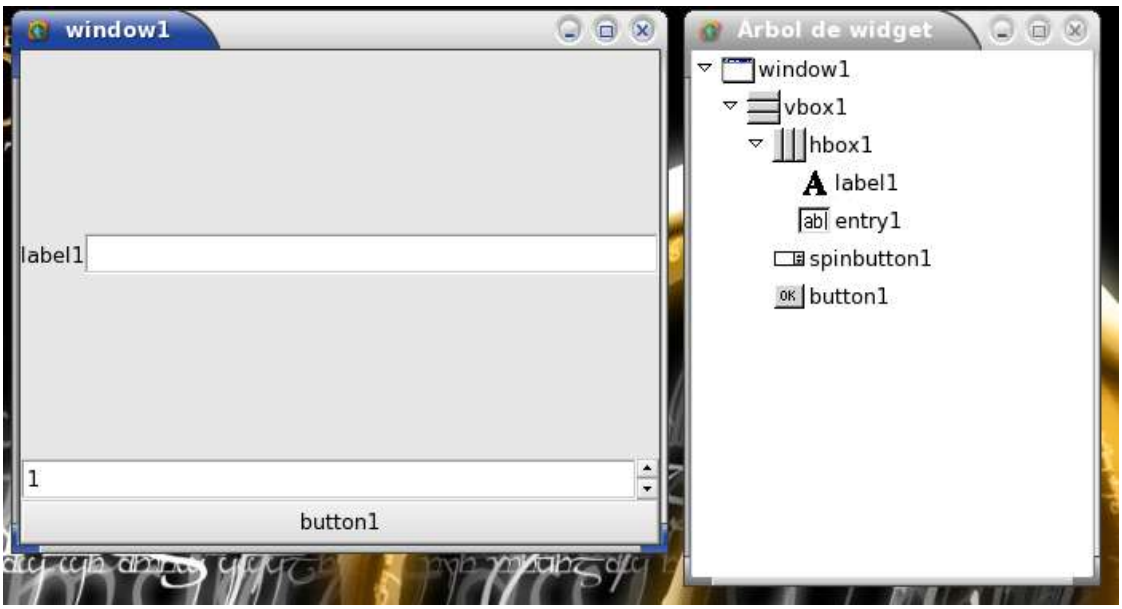

 Metabolik Bio Hacklab

Una vez tengamos la ventana principal lo que tenemos que hacer es pensar como vamos a repartir los controles y en función de ello y usando las cajas (tanto horizontales como verticales o usando tablas). También podríamos hacer uso del contenedor que permite colocar estáticamente los controles, pero de esta manera al cambiar el usuario el tamaño de la ventana no aprovecharíamos todo el tamaño de ella. Bueno, vamos a empezar por crear una caja vertical con 3 cajones.


Y dentro del primer cajón crearemos una caja horizontal de 2 cajones. Aquí podéis ver el resultado. En el árbol de controles (widgets) podemos ver como una caja esta anidada dentro de otra.

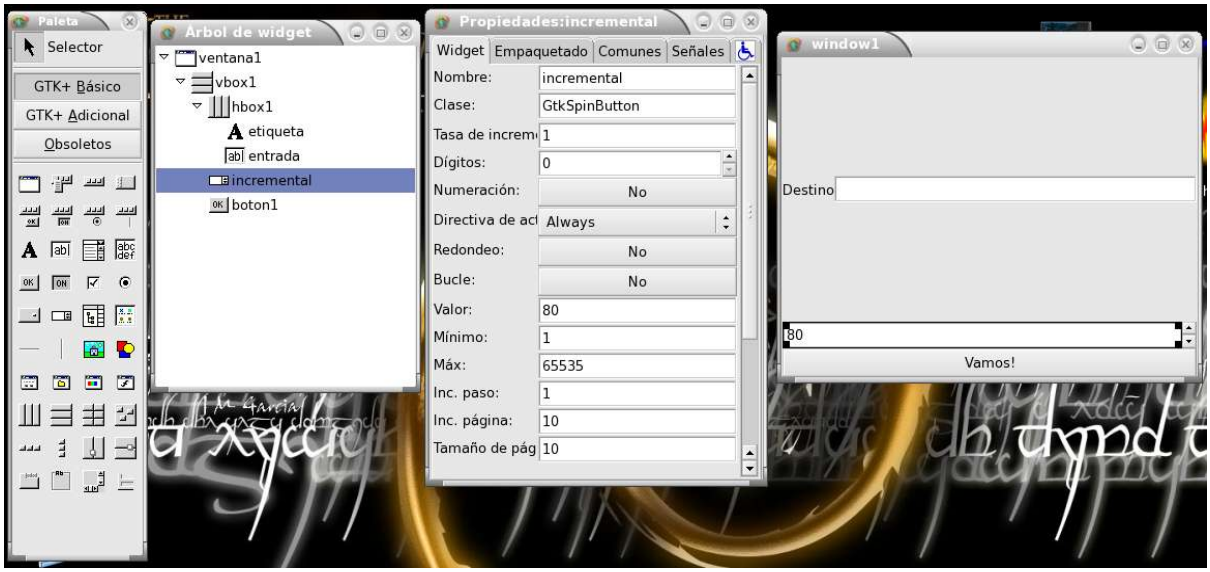
Es hora de colocar los controles de nuestra aplicación. Vamos a colocar una etiqueta en la primera caja a la izquierda, una entrada de texto en la segunda (arriba a la derecha), un botón de incremento en la caja central y para terminar un botón normal en el inferior. El resultado debería ser algo parecido a esto:




 Metabolik Bio Hacklab

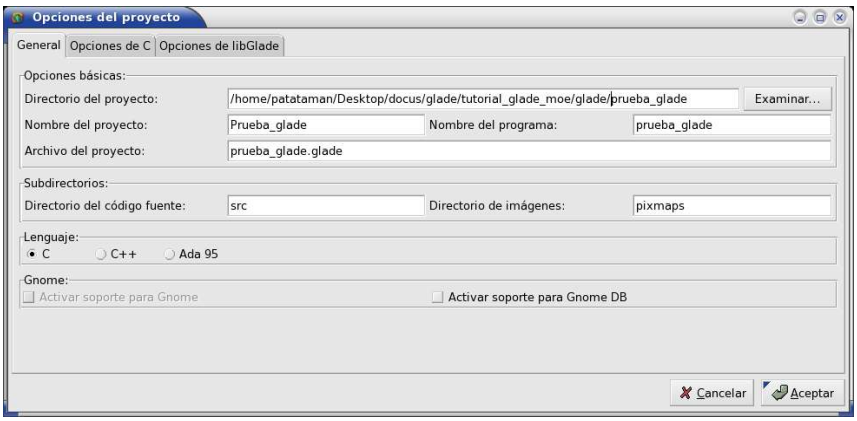
Antes de aprender como funciona el empaquetado de los controles (tamaño, que se expandan etc) vamos a cambiar los nombres de los controles a algo que nos sea más significativo (en este caso solo vamos a traducir sus nombres). Para ello elegimos el control que queremos modificar (bien en la ventana donde esta o en el árbol) y la ventana de propiedades , pestaña widget, vamos cambiado los nombres. También cambiaremos el texto que muestra la etiqueta y el texto del botón.


Una vez hecho esto la ventana debería tener un aspecto similar al la captura de debajo de este párrafo



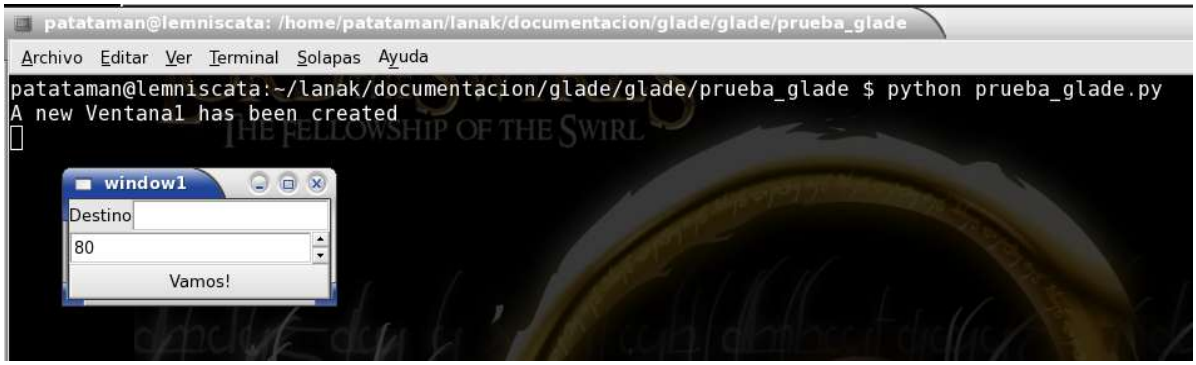
Antes de seguir con las señales vamos a guardar nuestro proyecto glade. De la ventana de opciones de guardar el proyecto lo que nos interesa es el nombre del archivo glade, ya que es lo que luego necesitaremos.




Vamos a hacer un poco de trampa y dar un salto ene l manual :) Una vez guardado nuestro glade vamos a generar una prueba de lo que luego será la aplicación python.

```
~/prueba_glade $ simple-glade-codegen.py prueba_glade.glade
prueba_glade.py
~/prueba_glade $ls prueba_glade.* -l
-rw-r--r--  1 patataman patataman 4462 2005-03-10 13:49 prueba_glade.glade
-rw-r--r--  1 patataman patataman  285 2005-03-10 13:49 prueba_glade.gladep
-rwxr-xr-x  1 patataman patataman 1017 2005-03-10 18:48 prueba_glade.py
-rwxr-xr-x  1 patataman patataman 1017 2005-03-10 18:48 prueba_glade.py.orig
```

Vemos, además de otros ficheros que luego explicaremos , el fichero ***prueba_glade.py*** . Si lo lanzamos con por ejemplo **"`python prueba_glade.py`"** veremos que se lanza nuestra ventana. Podemos escribir o cambiar el número del incremental, pero enseguida vemos que no hace nada nuestra aplicación. De hecho si intentamos cerrarla vemos que en la terminal sigue el proceso activo. Para lograr que nuestra ventana responda a nuestras acciones tenemos que usar las señales de los controles GTK+.



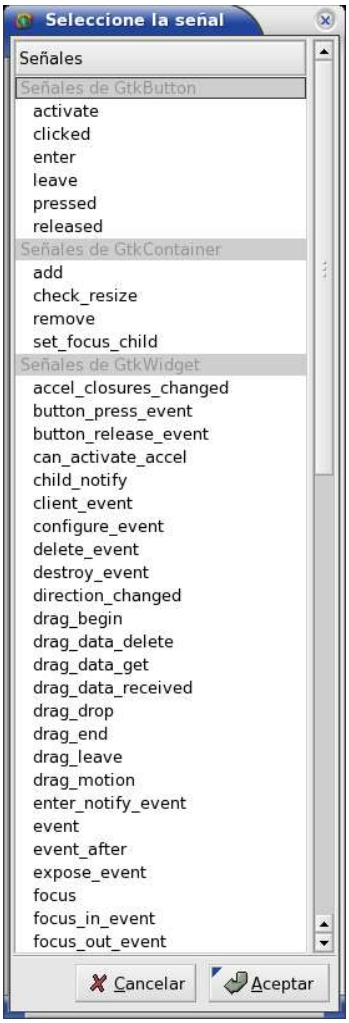

 

## *Señales*

Si queremos que cuando ocurra un evento en la gui (se pulse un botón, se edite un texto, etc) nuestro código python haga algo necesitamos usar señales. Lo primero sera en glade adjuntar las señales deseadas al control adecuado. Por ejemplo, elegimos el control botón y en la ventana de propiedades del control (widget) vamos a la pestaña señales.



Una vez en ella hacemos click en los puntos suspensivos (...) a la derecha del campo señal para elegir el tipo de señal. Nos aparecerá entonces una larga lista de señales soportadas por ese control. Primero tendremos las señales especificas de ese control (GtkButton en este caso), luego las señales del contenedor del control y por último las genéricas a todos los controles GTK+.



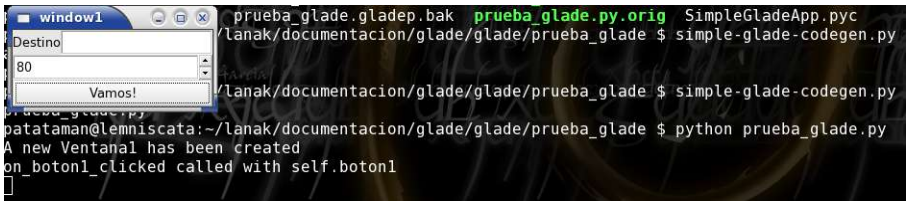

En este caso elegimos la señal clicked, que nos mandara una señal cada vez que el botón sea pulsado. Le damos a aceptar y si no queremos cambiar el nombre de la señal (on_boton1_clicked) le damos a añadir. Tendremos entonces la primera señal de nuestro programa.

Guardamos nuestro proyecto glade y vamos una vez mas a dar un salto en el manual. Volvemos a abrir una consola en la carpeta donde tengamos el fichero glade y volvemos a lanzar el simple-glade-codegen que nos regenerara el script python:

```
$ simple-glade-codegen.py prueba_glade.glade
prueba_glade.py
```

Si lanzamos la aplicación (`python prueba_glade.py`) y hacemos click en el botón veremos que en la terminal nos aparece un mensaje.




Estamos listos ya para ver con mas detalle como se genera el esqueleto en python de nuestra aplicación.

# Paso 2: Crear el esqueleto de la aplicación python *(simple-glade-codegen.py)*

Podríamos directamente usando pygtk cargar el fichero glade e importar los widgets que queremos usar. Afortunadamente **Sandino "tigrux" Flores** como se sentía vago (una de las cualidades de los hackers, que odian repetir la misma tarea) creó un guión en python que nos creará un esqueleto de aplicación python que no solo nos cargará el fichero glade, sino que además creará los esqueletos de las funciones asociadas a las señales. Nos lo podemos bajar desde aquí: http://etxea.net/docu/simple-glade-codegen.py .De esta manera al ejecutarla tendremos algo parecido a esto:

```
#!/usr/bin/env python
# -*- coding: UTF8 -*-

# Python module prueba_glade.py
# Autogenerated from prueba_glade.glade
# Generated on Fri Mar 11 20:58:46 2005

# Warning: Do not delete or modify comments related to context
# They are required to keep user's code

import os, gtk
from SimpleGladeApp import SimpleGladeApp

glade_dir = ""

# Put your modules and data here

# From here through main() codegen inserts/updates a class for
# every top-level widget in the .glade file.

class Ventana1(SimpleGladeApp):
        def __init__(self, glade_path="prueba_glade.glade", root="ventana1", domain=None):
                glade_path = os.path.join(glade_dir, glade_path)
                SimpleGladeApp.__init__(self, glade_path, root, domain)

        def new(self):
                #context Ventana1.new {
                print "A new Ventana1 has been created"
                #context Ventana1.new }

        #context Ventana1 custom methods {
        #--- Write your own methods here ---#
```

```
        #context Ventana1 custom methods }

        def on_boton1_clicked(self, widget, *args):
                #context Ventana1.on_boton1_clicked {
                print "on_boton1_clicked called with self.%s" % widget.get_name()
                #context Ventana1.on_boton1_clicked }

def main():
        ventana1 = Ventana1()

        ventana1.run()

if __name__ == "__main__":
        main()
```

Vemos que es muy sencillo. Importa distintos módulos, uno de los cuales SimpleGladeApp.py nos creará el mismo en el directorio donde estemos. Luego simplemente define una clase con el nombre de nuestra ventana y que es heredera de SimpleGladeApp y le añade las funciones asociadas a las señales que hayamos definido (con un útil print dentro de ellas para que sin editar el código podamos comprobar que la señal actúa como queremos).

Además de un archivo con el mismo nombre que nuestro .glade pero con extensión .py también vemos otros ficheros. Ya hemos hablado de SimpleGladeApp.py, que es donde se carga el glade y se asocian las funciones y las señales. Vemos también un fichero . orig. Es importante mantenerlo, ya que este fichero lo utilizará para que cada vez que lancemos simple-glade-codegen.py (por ejemplo porque hemos cambiado el fichero glade) mantener el código que hayamos introducido a mano.

# *Paso 3: Programar los algoritmos de la aplicación en python*

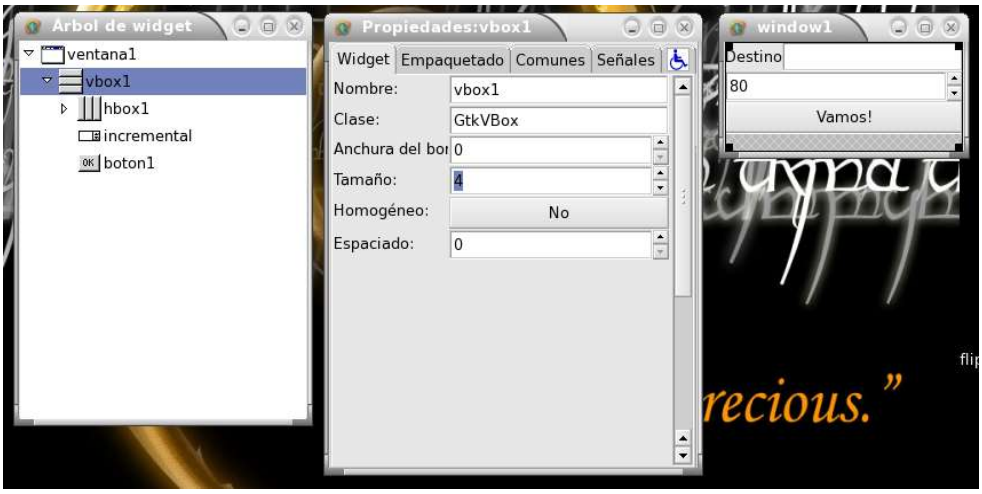
Bien, ya tenemos la interfaz y tenemos también las señales que harán que nuestros algoritmos actúen. Es hora de programar un poco de python. Vamos a hacer en 8 lineas (python mola :) un pequeño programa que se conecte a un servidor web y nos vuelque el resultado por consola:

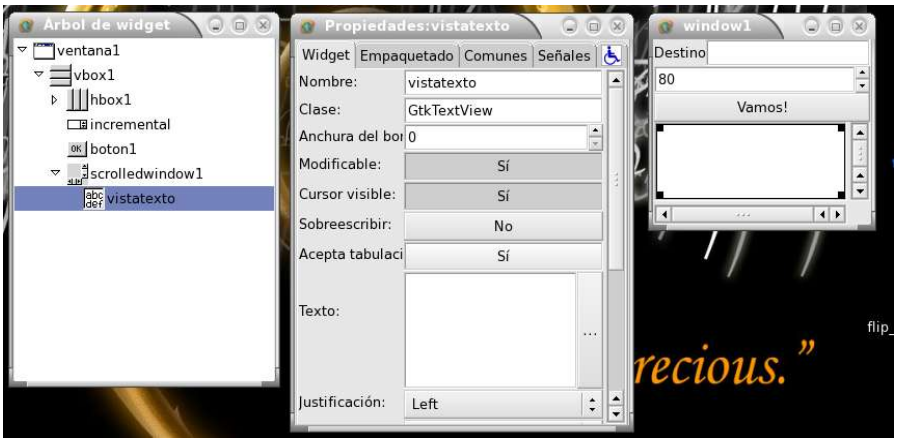
```
servidor=self.wTree.get_widget("entry1").get_text()
puerto=int(self.wTree.get_widget("spinbutton1").get_value())
if host=="":
  return
import urllib
page=urllib.urlopen("http://"+servidor+":"+str(puerto)+"/")
data=page.read()
print data
```

# *Paso 4: Queremos ampliar el programa.*

Como estamos haciendo una aplicación gráfica no tiene mucho sentido que la salida la obtengamos a través de una terminal. Por ello vamos a añadir un control nuevo que nos muestre la salida de la petición HTML.

Lo primero será hacerle un hueco en la cajonera vertical. Para ello elegimos el control en el árbol de widgets y en la ventana de propiedades cambiamos su tamaño de 3 a 4. Con esto lograremos un nuevo cajón vacío.




En este cajón vamos a añadir un control de “Vista del Texto”. De paso le cambiamos el nombre a vistatexto por ejemplo.

Lo siguiente es guardar el proyecto glade y volver a lanzar a nuestro buen amigo simple-glade-codegen.py (aunque en este caso no es necesario). Una vez actualizado el glade vamos a cambiar la manera en la que representamos los datos en nuestro guión python. Si recordamos teniamos un simple print data, que vamos a cambiar por lo siguiente:

```
mi_buffer=gtk.TextBuffer()
mi_buffer.insert_at_cursor(data)
self.vistatexto.set_buffer(mi_buffer)
```

Hemos tenido que hacer uso de un buffer GTK de tipo texto para poder pasarle a través de el el texto al control vistatexto. Con todo esto la función on_boton1_clicked será tal que así:

```
print "on_boton1_clicked called with self.%s" % widget.get_name()
host=self.entrada.get_text()
print "En etiqueta tenemo: %s" % host
port=int(self.incremental.get_value())
if host=="" :
            print "no hostname!"
            return
import urllib
page=urllib.urlopen("http://"+host+":"+str(port)+"/")
data=page.read()
print data
mi_buffer=gtk.TextBuffer()
mi_buffer.insert_at_cursor(data)
self.vistatexto.set_buffer(mi_buffer)
#context Ventana1.on_boton1_clicked }
```

El resultado debería ser algo tal que así:


La utilidad de esta aplicación es mas bien poca :) pero nos ha servido para ver lo rápido y sencillo que es desarrollar sobre python+glade. Suerte y que la fuente este con vosotros :).

# Bibliografía

## *Tutorial esencial. Es del creador de simple-glade-codegen.py:*

- http://primates.ximian.com/~sandino/python-glade/

## *Documento en el cual se basa el ejemplo de este tutorial:*

- http://www.linuxjournal.com/article/6586

## *Que es python?*

- http://users.servicios.retecal.es/tjavier/python/Que_es_Python.html

## *Web de pygtk, con la referencia a todos los widgets*

- http://pygtk.org/
- http://www.pygtk.org/pygtk2reference/index.html

# Licencia

**Reconocimiento-NoComercial-CompartirIgual 2.1 España**

**Usted es libre de:**

- copiar, distribuir y comunicar públicamente la obra
- hacer obras derivadas

**Bajo las condiciones siguientes:**

**Reconocimiento**. Debe reconocer y citar al autor original.

**No comercial**. No puede utilizar esta obra para fines comerciales.

**Compartir bajo la misma licencia**. Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

- Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.
- Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor

**Los derechos derivados de usos legítimos u otras limitaciones no se ven afectados por lo anterior.**

Esto es un resumen legible por humanos del texto legal (la licencia completa) disponible en los idiomas siguientes:
Catalán Castellano

 Metabolik Bio Hacklab