PyUNO: Explota todo el potencial de OpenOffice

PYTHON NO HAY MÁS QUE UNO

¿Has visto alguna vez a los brokers de bolsa? ¿Recuerdas sus sofisticados y caros programas para ver las cotizaciones de las empresas en bolsa en tiempo real? Nosotros haremos lo mismo con 70 lineas de código Python, OpenOffice y la tecnología UNO de OpenOffice. **POR JOSÉ MARÍA RUIZ**

No es ni será la última vez que desde esta sección recordemos que la idea original de Stallman era la de que cada programa libre estuviese construido sobre librerías de funciones, de manera que su código fuese reutilizable por cualquier otro programa.

Quizás en un programa pequeño no sea muy útil este tipo de diseño, pero ¿qué pasa con esos monstruos consumidores de memoria que rondan por nuestros discos duros? Nos referimos a programas o entornos del calibre de Gnome, KDE, Mozilla u OpenOffice. Todo el mundo se queja de su tamaño excesivo, su alto consumo de recursos y su inexplicable complejidad.

Quizás con este artículo desmintamos este mito y hagamos que el lector mire con nuevos ojos a estos maravillosos programas.

Grandes sistemas de componentes

El diseño de un gran programa puede llevar años y cientos o miles de programadores. Organizar tal cantidad de personas supone ya una locura sólo por el hecho de asegurarse que todos cobren. Pero vayamos a nuestro mundillo ¿cómo podemos organizarles para que el desarrollo no acabe en un fiasco?

Esta es la gran cuestión no resuelta de la informática pero, aunque no hayamos encontrado una solución fiable, sí se disponen de técnicas que aumentan la probabilidad de que, al menos, se cree algún software útil.

Una de estas técnicas consiste en emplear un sistema de componentes como base para el desarrollo. Un componente es una cantidad de software que ofrece un servicio bien definido y que es reutilizable. Además debe ser posible reutilizarlo «de verdad»: desde cualquier lenguaje y cualquier sitio.

Cualquiera que tenga conocimiento sobre cómo funcionan los lenguajes de programación a bajo nivel sabrá que esto es muy muy complicado. Por ello se han desarrollado infraestructuras que nos permiten interactuar con los componentes de manera indirecta. A este software se le suele llamar "middleware" (algo así como "software de en medio").

Ejemplos famosos de Middleware son J2EE, que muchos conocerán, y CORBA, que a muchos les gustaría no conocer. Ambos son sistemas enormes y costosos que relegan al programador a mera herramienta en manos de ingenieros denominados arquitectos que conocen su compleja infraestructura.

Pero los sistemas de componentes también se emplean en software libre y han dado buenos resultados. Quizás el más desconocido es UNO, de Universal Network Objects, el sistema que emplea OpenOffice, ver Listado [1], y que SUN desarrolló para su precursor: StarOffice.

PyUNO

Un sistema de componentes con el que sólo se pueda programar en un lenguaje no tiene mucha utilidad. Por eso en OpenOffice se han asegurado de fomentar la creación de interfaces a distintos lenguajes de programación. Podemos acceder a UNO usando Javascript, Java, Ruby, Perl o Python (ver Recurso [2]).

PyUNO es el nombre de la interfaz y podremos encontrarlo sin problemas en nuestra distribución de Linux. Evidentemente, necesitamos también tener instalado OpenOffice. En este artículo hemos realizado los programas usando OpenOffice 2.0, que cambió la interfaz respecto a la versión 1.0, y la versión de PyUNO 0.7.0.

Un ejemplo rápido

Vamos a crear el famoso «Hola mundo» con PyUNO. Para ello primero debemos arrancar OpenOffice con el siguiente comando desde el directorio donde esté instalado:

```
$> ./soffice➥
"-accept=socket,➥
host=localhost,➥
port=2002;urp;"
```

Listado 1: programa «Hola Mundo»

```
01 import uno
02
03 localContext =
   uno.getComponentContext()
04
05 resolver =
   localContext.ServiceManager.cr
   eateInstanceWithContext("com.s
   un.star.bridge.UnoUrlResolver"
   ,
06
   localContext )
07 ctx = resolver.resolve(
   "uno:socket,host=localhost,por
   t=2002;urp;StarOffice.Componen
   tContext" )
08
09 desktop =
   ctx.ServiceManager.createInsta
   nceWithContext(
   "com.sun.star.frame.Desktop",c
   tx)
10
11 doc =
   desktop.loadComponentFromURL("
   private:factory/swriter","_bla
   nk",0,())
12
13 cursor =
   doc.Text.createTextCursor()
14
15 doc.Text.insertString( cursor,
   "Hola Mundo", 0 )
16
17 ctx.ServiceManager
```

Al arrancar OpenOffice se arranca su sistema de componentes. Podemos pensar en este proceso como en el arranque de un servidor, sólo cuando esté funcionando podrán los clientes trabajar con él.

Las opciones que pasamos son para que se cree un socket y se escuche en localhost en el puerto 2002. Por defecto OpenOffice no abre el socket, de manera que no podrán controlar nuestro OpenOffice sin nuestro consentimiento.

OpenOffice incorpora de serie varios intérpretes de lenguajes, entre ellos uno de Python que ya viene preconfigurado para poder hacer uso de la librería UNO. Está junto al resto de ejecutables de OpenOffice, así que lo ejecutaremos desde allí. El programa que usaremos se encuentra en el Listado [2].

El proceso es el siguiente:

- Obtenemos un contexto local (un sitio donde guardar los datos de la conexión)
- Arrancamos el componente *UnoUrlResolver* que nos sirve para acceder a otro OpenOffice en otro equipo (en nuestro caso accederemos a nuestro propio equipo)
- Emplearemos el objeto *resolver* para acceder al OpenOffice remoto
- Arrancamos un «Desktop» (escritorio) de OpenOffice (esto es una instancia de OpenOffice vacía)
- Arrancamos un *SWriter* (es decir, el procesador de textos) en el escritorio
- Obtenemos un cursor, con el que podremos posicionarnos dentro del texto
- e insertamos texto en el cursor

El resultado, no muy espectacular, podemos verlo en la Figura [1]. Ya tenemos nuestro «hola mundo» insertado en SWriter.

¿Demasiado código? Piensa por un momento lo que estamos haciendo. Hemos levantado dos componentes y hecho acceso remoto a otro OpenOffice. Este segundo OpenOffice puede estar en una máquina al otro lado del mundo. Es algo bastante impresionante, pero por el momento poco útil. Veamos un poco más sobre UNO antes de realizar un programa más útil.

Arquitectura de UNO

OpenOffice está implementado en C++. UNO se usa internamente para realizar cualquier cosa. Básicamente OpenOffice no es más que una gran cantidad de componentes que interactúan entre sí. Todo dentro de OpenOffice es un componente, así que podemos acceder a cualquier parte de la aplicación, ¡incluso reconstruir OpenOffice en Python!

Los sistemas de componentes usan un registro de componentes al que se le puede pedir que arranque componentes. El registro localiza el componente en disco y lo carga en memoria, de manera que puede ser usado. Las llamadas a las funciones no se realizan directamente, sino que se suele emplear algún sistema no dependiente de lenguaje o plataforma, como puede ser XML o un formato ASCII.

El registro también debe ser capaz de gestionar los recursos que consume el componente, descargándolo de memoria cuando ya no sea necesario.

Los componentes pueden ser programados en cualquier lenguaje con el que se tenga interfaz. Un componente es un conjunto de ficheros que proporcionan un servicio. Se acompañan de un fichero XML que describe su funcionalidad. Lo mejor es que podemos vincular ese servicio a algún componente gráfico, como por ejemplo un botón o menú.

Comenzaremos por realizar un programa que funcionará de manera externa a OpenOffice y después crearemos un componente con él y lo integraremos en OpenOffice.

Nuestro programa de Stocks

Comencemos con la parte útil, ver Listado [2]. Vamos a crear un sistema que nos permita controlar las acciones de una serie de empresas que están en bolsa dentro de un índice tecnológico, el Nasdaq (para algo estamos en una revista de informática), usando la hoja de cálculo SCalc y un programa Python. Nuestro programa accederá usando Internet a un sitio web donde podrá recoger los datos en CSV (Valores Separados por Comas), los procesará y los introducirá en SCalc.

Comenzaremos por crear una función que recoja el fichero CSV y lo procese. Lo que hacemos es conectarnos con la página web *finance.yahoo.com*. Yahoo tiene un sitio web bastante avanzado sobre cotizaciones de bolsa, y uno de sus servicios nos permite recoger los datos de cotización de una empresa en tiempo real en formato CSV. Sin embargo, Yahoo no nos permitirá acceder a los datos demasiado a menudo, ya que dispone de un servicio de pago para ello, así que puede cortarnos el «grifo» en cualquier momento si hacemos demasiadas consultas por minuto. Por eso recogeremos los datos cada 10 segundos. La función *getSimbolo()* se encargará de ello.

Ahora ya tenemos los datos, tenemos que mandarlos a SCalc. Hemos creado un objeto llamado *Calc* para gestionar el acceso. Hemos metido en el método constructor (*__init__*) el código que conecta con el OpenOffice, puesto que sólo lo haremos una vez. Una hoja de cálculo posee varias «hojas», así que tendremos que solici-

Listado 2: «OfficeBroker»

```
01 import uno
02 import random
03 import time
04 import httplib
05 import csv
06
07 class Calc:
08     def __init__(self):
09         self.conecta()
10
11     def conecta (self):
12         self.local =
   uno.getComponentContext()
13         self.resolver =
   self.local.ServiceManager.crea
   teInstanceWithContext("com.sun
   .star.bridge.UnoUrlResolver",
   self.local)
14
15         self.context =
   self.resolver.resolve("uno:soc
   ket,host=localhost,port=2002;u
   rp;StarOffice.ComponentContext
   ")
16
17         self.desktop =
   self.context.ServiceManager.cr
   eateInstanceWithContext("com.s
   un.star.frame.Desktop",
18
   self.context)
19
20         #self.doc =
   self.desktop.getCurrentCompone
   nt()
21         self.doc =
   self.desktop.loadComponentFrom
   URL("private:factory/scalc","_
   blank",0,())
22
23         self.hojas =
   self.doc.getSheets()
24         self.s1 =
   self.hojas.getByIndex(0)
25
26     def actualiza(self,
   cotizacion, fila):
27
28         i = 0
29         for entrada in
   cotizacion:
30             if (i == 0) or (i
   == 2) or (i ==3):
31
   self.s1.getCellByPosition(i,fi
   la).setString(entrada)
32             else:
33
   self.s1.getCellByPosition(i,fi
   la).setValue(float(entrada))
34
35             i = i + 1
36
37 def getSimbolo(simbolo):
38     c =
   httplib.HTTPConnection("financ
   e.yahoo.com")
39
   c.request("GET","/d/quotes.csv
   ?s="+simbolo+"&f=sl1d1t1c1ohgv
   &e=.csv")
40     r = c.getresponse()
41     cad = r.read()
42     reader = csv.reader([cad])
43     resultado = []
44     for row in reader:
45         resultado = row
46     return resultado
47
48 if __name__ == '__main__':
49
50     simbolos =
   ["GOOG","MSFT","RHAT"]
51
52     c = Calc()
53
54     while(1):
55         i = 0;
56         for s in simbolos:
57
   c.actualiza(getSimbolo(s),i)
58             i = i + 1
59
60             time.sleep(10)
```

tar una usando el método *getSheets()*, que nos devuelve una lista con las distintas hojas. Dentro de esta lista usaremos *getByIndex()* para seleccionar la primera hoja, que es la que se ve cuando arrancamos SCalc.

El método *actualiza()* admite una lista con los datos de cotización y número que representa la fila donde aparecerá en SCalc. Una hoja de cálculo se compone de celdas y éstas tienen un tipo. La función *getCellByPosition()* nos permite acceder a una celda pasándole la columna y la fila (al revés de lo normal, así que cuidado).

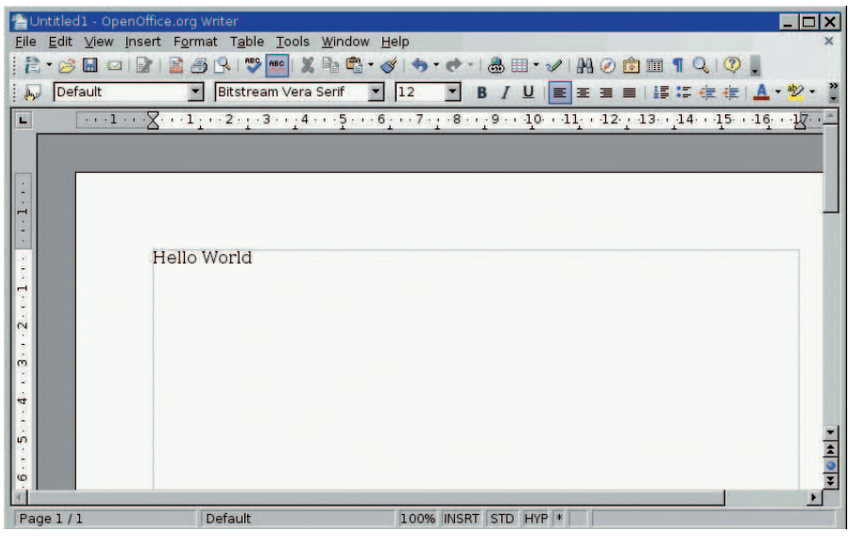


Figura 1: Un documento de Write de OpenOffice con el ineludible "Hello World" generado a partir de PyUNO.

Una vez localizada la celda tenemos varias funciones para poder asignar un valor:

- *setString()*: para poner una cadena
- *setValue()*: para poner un número
- *setFormula()*: para poner una fórmula

El dato *cotización* es la lista de parámetro de cotización, pero vienen dados como cadenas de caracteres. Las posiciones 0, 2 y 3 son realmente cadenas, pero el resto son números. Por eso tenemos que convertir ciertos valores al tipo *float()* mediante la función *float()*.

El resultado se puede ver en la Figura [2], veremos cómo se abre una ventana

Listado 3: Addons.xcu

```
01 <?xml version="1.0"
   encoding="UTF-8"?>
02 <oor:node
   xmlns:oor="http://openoffice.o
   rg/2001/registry"
03
   xmlns:xs="http://www.w3.org/20
   01/XMLSchema"
04           oor:name="Addons"
   oor:package="org.openoffice.Of
   fice">
05     <node oor:name="AddonUI">
06
07         <node
   oor:name="AddonMenu">
08             <node
   oor:name="org.openoffice.comp.
   pyuno.linuxmagazine.Stock"
   oor:op="replace">
09                 <prop
   oor:name="URL"
   oor:type="xs:string">
10
   <value>service:org.openoffice.
   comp.pyuno.linuxmagazine.Stock
   ?insert</value>
11                 </prop>
12                 <prop
   oor:name="Title"
   oor:type="xs:string">
13                     <value/>
14                     <value
   xml:lang="en-US">Stock
   Market</value>
15                     <value
   xml:lang="es">Cotización en
   Bolsa</value>
16                 </prop>
17                 <prop
   oor:name="Target"
   oor:type="xs:string">
18
   <value>_self</value>
19                 </prop>
20                 <prop
   oor:name="ImageIdentifier"
   oor:type="xs:string">
21
   <value>private:image/3216</val
   ue>
22                 </prop>
23
24             </node>
25         </node>
26     </node>
27 </oor:node>
```

de SCalc y se rellena con los valores de las contizaciones, además de cómo se actualizan cada 10 segundos. Si creamos un gráfico que use esos valores se actualizará con ellos.

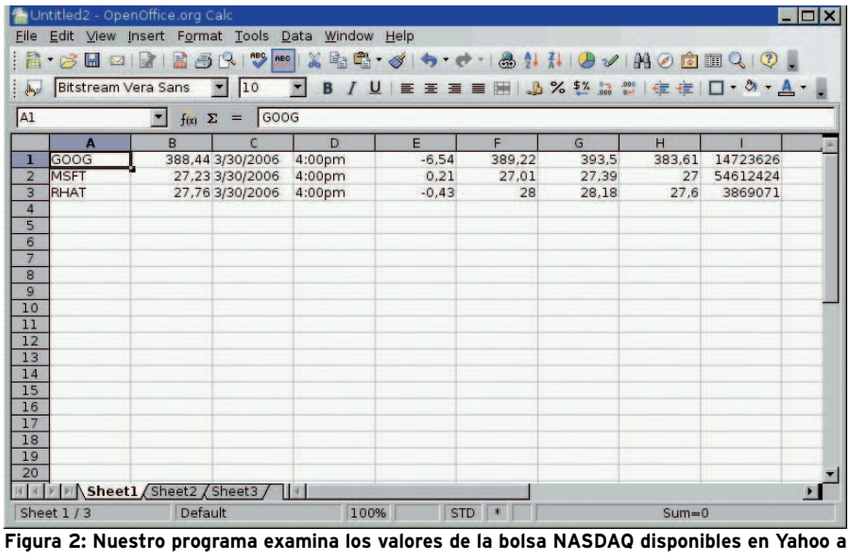
Pero este es un programa externo... estaría bien que pudiésemos hacer eso pulsando un botón

Creamos un componente UNO

Los componentes UNO no son más que código debidamente empaquetado. Los paquetes que OpenOffice admite tienen una estructura fija. Son ficheros ZIP que contienen los ficheros con el código fuente, recursos (como imágenes) y un fichero de configuración XML.

Los ficheros deben tener nombres especiales. El fichero de configuración debe llamarse *Addons.xcu* y permite asignar el código fuente del paquete con el widget que deseemos, un botón, una entrada de un menú... Ver Listado [3].

La sintaxis del fichero parece bastante complicada, cuando en realidad no es muy difícil de entender. Básicamente decimos que queremos que nuestro componente se asocie con una entrada en el menú «Addons» que está en *Tools* o *Herramientas* en castellano. Nuestro componente tiene una ruta que especificaremos después y que es:



	A	B	C	D	E	F	G	H	I
1	GOOG	388,44	3/30/2006	4:00pm	-6,54	389,22	393,5	383,61	14723626
2	MSFT	27,23	3/30/2006	4:00pm	0,21	27,01	27,39	27	54612424
3	RHAT	27,76	3/30/2006	4:00pm	-0,43	28	28,18	27,6	3869071

Figura 2: Nuestro programa examina los valores de la bolsa NASDAQ disponibles en Yahoo a intervalos regulares y los inserta en una hoja de cálculo de OpenOffice.

```
org.openoffice.comp.pyuno.➥
linuxmagazine.Stock
```

Esta ruta la hemos creado nosotros y tenemos que tener cuidado de que sea única, por eso hemos incorporado *linuxmagazine* en ella ;). Definimos un título, que puede estar en varios idiomas, y una imagen, que hemos escogido de entre las que proporciona OpenOffice.

El fichero con el código fuente Python en sí se puede ver en el Listado 4. Tenemos un objeto llamado *StockJob* que será el que se invocará en caso de pulsar la entrada en el menú. Ese objeto se vincula a la ruta que vimos antes. Cada vez que se pulse sobre la entrada del menú se ejecutará el método *trigger*, que descargará de Internet las cotizaciones y las mostrará en la hoja de cálculo. Es posible hacer que sólo se muestre el menú cuando arrancamos la hoja de cálculo SCalc, pero por motivos de espacio no hemos puesto la restricción. Aún así si no estamos en una hoja de cálculo no sucederá nada, simplemente no funcionará.

Manejo de paquetes en OpenOffice

Ahora tenemos que generar nuestro paquete UNO. Para ello necesitaremos el programa ZIP, gzip no nos vale, y crear un fichero:

```
$> zip stock.zip➥
stock_comp.py Addons.xcu
updating: ...../Addons.xcu➥
```

Listado 4: stock_comp.py

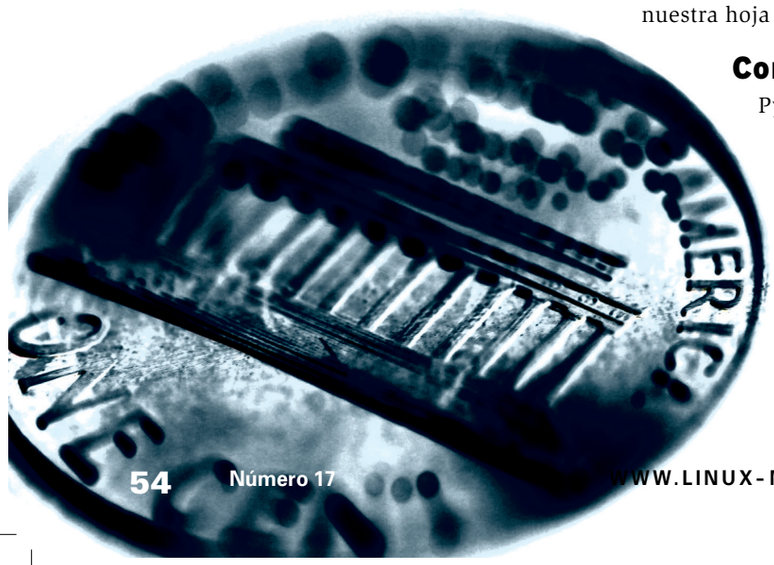
```
01 import uno
02 import unohelper
03
04 import random
05 import time
06 import httplib
07 import csv
08
09 from com.sun.star.task import
   XJobExecutor
10
11 def getSimbolo(simbolo):
12     c =
   httplib.HTTPConnection("financ
   e.yahoo.com")
13
   c.request("GET","/d/quotes.csv
   ?s="+simbolo+"&f=sl1d1t1c1ohgv
   &e=.csv")
14     r = c.getresponse()
15     cad = r.read()
16     reader = csv.reader([cad])
17     resultado = []
18     for row in reader:
19         resultado = row
20     return resultado
21
22 class StockJob(
   unohelper.Base, XJobExecutor
   ):
23     def __init__( self, ctx ):
24
25         self.ctx = ctx
26
27     def trigger( self, args ):
28         desktop =
   self.ctx.ServiceManager.create
   InstanceWithContext(
29
   "com.sun.star.frame.Desktop",
   self.ctx )
30
31         model =
   desktop.getCurrentComponent()
32
33         self.hojas =
   model.getSheets()
34
35         self.s1 =
   self.hojas.getByIndex(0)
36
37         simbolos =
   ["GOOG","MSFT","RHAT"]
38         i = 0;
39
40         for s in simbolos:
41
   self.actualiza(getSimbolo(s),i
   )
42             i = i + 1
43
44     def actualiza(self,
   cotizacion, fila):
45         i = 0
46         for entrada in
   cotizacion:
47             if (i == 0) or (i
   == 2) or (i ==3):
48
   self.s1.getCellByPosition(i,fi
   la).setString(entrada)
49             else:
50
   self.s1.getCellByPosition(i,fi
   la).setValue(float(entrada))
51
52             i = i + 1
53
54 g_ImplementationHelper =
   unohelper.ImplementationHelper
   ()
55
56
   g_ImplementationHelper.addImpl
   ementation( StockJob,
57
   "org.openoffice.comp.pyuno.lin
   uxmagazine.Stock",
58
   ("com.sun.star.task.Job",),)
```

```
(deflated 59%)
updating: ...../stock_comp.py
(deflated 57%)
>
```

Este fichero debe ser integrado en OpenOffice, iremos al directorio donde esté instalado y ejecutaremos como root:

```
$> sudo ./unopkg add
stock.zip
>
```

Con esto concluye la instalación del paquete...¡no ha sido tan difícil! Cuando arranquemos de nuevo OpenOffice podremos seleccionar la hoja de cálculo SCalc y en el menú Tools/Herramientas veremos cómo ha aparecido al final un nuevo submenú: «Complementos (add-ons)». Dentro del mismo aparecerá una nueva entrada llamada «Cotización de Bolsa». Si lo pulsamos aparecen los datos de 3 compañías (Google, Microsoft y Redhat) del Nasdaq en nuestra hoja de cálculo.



Conclusión

Python nos permite un uso nuevo de algo tan trillado como puede ser un paquete ofimático. OpenOffice entero es accesible desde Python; no es difícil imaginarse programas que podrían facilitarnos mucho la vida y no son tan difíciles de crear gracias a PyUNO. No hemos explorado la posibilidad de actuar sobre un OpenOffice remoto por falta de espacio, pero es una nueva posibilidad que abre un camino para aplicaciones muy interesantes, como puede ser la edición distribuida de documentos o un uso más creativo de la hoja de cálculo.

Todo un mundo de posibilidades se abre ante nosotros gracias a Python. ■

RECURSOS

[1] El sitio de OpenOffice: *http://www.openoffice.org*

[2] El "puente" entre Python y OpenOffice: *http://udk.openoffice.org/python/python-bridge.html*

[3] Los listados de este arículo: *http://www.linux-magazine.es/Magazine/Downloads/17*