

Ironpython

# DE SERPIENTES Y PRIMATES

.NET está avanzando, y Python no se ha quedado atrás. En lugar de combatirla, ha entrado en simbiosis con ella. Con Ironpython podremos hacer uso de toda la potencia de .NET desde nuestro lenguaje favorito.

**POR JOSÉ MARÍA RUIZ.**

**E**n el año 2003, Jim Hugunin leyó que la plataforma de Microsoft .NET no era adecuada para la creación de lenguajes dinámicos. Un lenguaje dinámico es aquel al que no debes decirle qué tipo de variable vas a usar. Por ejemplo, si voy a almacenar un número en la variable *num*, en un lenguaje estático debería declarar que *num* será de tipo *entero* (números positivos y negativos sin coma decimal), mientras que en un lenguaje dinámico sólo tendría que almacenar en *num* el número y el lenguaje se encargará de averiguar el tipo.

## Python sobre .NET

Hugunin se extrañó bastante del comentario porque ya había creado un intérprete de Python, Jython, para la máquina virtual de Java. Este intérprete consiguió cierto reconocimiento, y prueba de ello es el artículo sobre Jython que aparece en el número 7 de esta misma revista.

Así que ni corto ni perezoso Jim se tomó el comentario como algo personal y se puso manos a la obra. Como

resultado de su esfuerzo, el 5 de Septiembre de 2006 apareció la versión 1.0 de IronPython (ver Recurso [1]), nombre que dio a su intérprete (ver Recurso [2]).

Antes de la aparición de la versión 1.0, IronPython ya había llamado la atención de gran cantidad de desarrolladores. La posibilidad de crear aplicaciones gráficas o web multiplataforma aprovechando los recursos de .NET era demasiado interesante para dejarla escapar.

Incluso Miguel de Icaza, de fama mundial gracias a Gnome y Mono, reconoció la importancia de IronPython para Mono en el artículo que aparece referenciado en el Recurso [3]: «Prácticamente cada nueva versión de IronPython ha expuesto las limitaciones de nuestro runtime (Mono), nuestras librerías de clases o nuestros compiladores. IronPython realmente ha contribuido para que Mono se convierta en un mejor runtime.»

Vamos a echar un buen vistazo a las posibilidades de IronPython y veremos cómo nos permite explotar el poder de las librerías .NET, pero siempre desde nuestro lenguaje favorito.

## Preparativos

Para poder hacer uso de IronPython necesitamos instalarlo. IronPython es una implementación de Python sobre .NET, por lo que necesitamos una implementación de .NET para Linux.

Mono es la implementación libre de .NET más famosa. Como vimos antes, Miguel de Icaza está usando IronPython como banco de pruebas para Mono. Esto nos asegura que IronPython funcionará bastante bien sobre Mono. Tenemos que instalar IronPython y Mono.

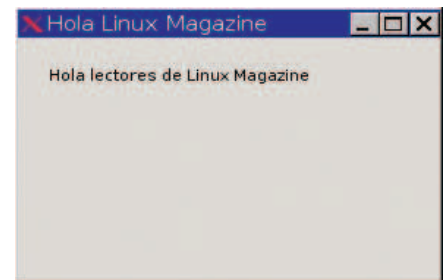


Figura 1: Hola Mundo con WinForms.

Falta un componente en la ecuación: *libgdiplus* (ver Recurso [4]). En este artículo haremos uso del interfaz gráfico de usuario que .NET pone a disposición de sus desarrolladores. Este sistema, llamado WinForms, ha sido reimplementado en forma de software libre en la librería *libgdiplus*. Sin ella no podremos hacer uso de WinForms, y por ello necesitamos instalarla.

Un detalle también algo molesto, ¿cómo se sale del intérprete? La verdad es que es algo rebuscado. No funcionan ni *exit* ni *quit*. Cuando ejecutemos cual-

## Listado 1: Mostrar el contenido de un fichero de texto

```
01 import clr
02
03 from System.IO import *
04
05 fichero =
    File.OpenText("mifichero.txt")
06
07 linea = fichero.ReadLine()
08
09 while s:
10     print linea
```



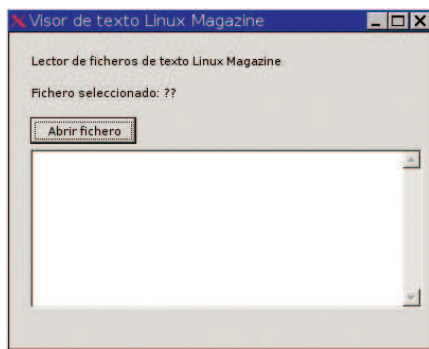


Figura 2: Visor de texto.

quiera de los dos seremos informados de que debemos pulsar *Control z* para salir del intérprete. Esto puede resultar curioso para todo aquel que haya trabajado mucho en la consola de Linux, porque *Control z* sirve para mandar a un programa a background. La secuencia sería:

```
01 IronPython 1.0 (1.0) on
.NET 2.0.50727.42
02 Copyright (c) Microsoft
Corporation. All rights
reserved.
```

```
03 >>> quit
04 'Use Ctrl-Z plus Return to
exit'
05 >>> exit
06 'Use Ctrl-Z plus Return to
exit'
07 >>>
08 Suspended
09 >
10 >
```

Jim tuvo que crear este sistema para que funcionara tanto en Windows como en Linux. Si ya tienes todos estos paquetes instalados en tu distribución podemos pasar al siguiente paso... el infame «Hola Mundo».

### Hola mundo consola

La primera sorpresa de IronPython es que realmente es casi igual a Python... pero más lento. Para ejecutar un programa .NET en Linux debemos emplear el comando *mono* que lo ejecuta dentro de la máquina virtual de .NET.

```
> mono miprograma.exe
```

Probablemente, el comando *ironpython* de vuestra distribución sea en realidad un script shell que ejecuta el intérprete (llamado *ipy.exe*) empleando *mono*. El resultado será:

```
IronPython 1.0 (1.0) on
.NET 2.0.50727.42
Copyright (c) Microsoft
Corporation. All rights
reserved.
>>>
```

¿Microsoft Corporation? ¡Tranquilos, tranquilos! IronPython posee una licencia casi libre, parte del Microsoft's Shared Source. Aunque su licencia no está aprobada por OSI, Jim dice que la licencia que lo cubre sigue todas las normas de OSI (ver Recurso [5]).

Vayamos al grano, nuestro «hola mundo» es reconocible:

```
>>> print "Hola Mundo"
Hola Mundo
>>>
```

Como el lector podrá apreciar, no ha cambiado nada. IronPython mantiene una gran compatibilidad con Python. Pero no todo el monte es orégano. Debido a que se ejecuta sobre .NET, IronPython no puede acceder a librerías de Python escritas en C. Esto es un problema porque, por ejemplo, la librería *sys* está escrita en C.

Otro detalle bastante molesto es que el intérprete de IronPython no tiene historia, ni autocompletación, ni ... nada de nada. Esto puede poner a más de uno de los nervios, yo al menos me puse, por lo que es recomendable trabajar en un fichero y ejecutarlo con IronPython.

### Librerías .NET

Hasta el momento no hemos hecho nada especial, es hora de comenzar con lo interesante. IronPython está escrito en C# sobre .NET. Una de las virtudes de .NET es que una vez que está compilado a su lenguaje intermedio puedes hacer uso de cualquier librería escrita en cualquier lenguaje para .NET.

IronPython tiene a su disposición todas las librerías del proyecto Mono. Esto incluye la librería para programación de aplicaciones gráficas WinForms, las librerías para programación web ASP.NET, estructuras de datos, librerías

### Listado 2: «Hola Mundo» con WinForms

```
01 import clr                                16
02                                           17 self.panel1 = Panel()
    clr.AddReference('System.Drawing')      18 self.panel1.Location = Point
03                                           19 self.panel1.Width =
    clr.AddReference('System.Windows        self.Width
    s.Forms')                                 20 self.panel1.Height =
04                                           self.Height
05 from System.Drawing import                21
    Color, Point                               22 self.generaSaludo()
06 from System.Windows.Forms                23
    import (Application,                       24
    BorderStyle, Button, Form,
    FormBorderStyle, Label, Panel,
    Screen)                                    25
07                                           self.Controls.Add(self.panel1)
08 class HolaMundo(Form):                    26
09 def __init__(self):                        27 def generaSaludo(self):
10 self.Text = "Hola Linux                    28 self.label1 = Label()
    Magazine"                                   29 self.label1.Text = "Hola
11 self.FormBorderStyle =                    lectores de Linux Magazine"
    FormBorderStyle.FixedDialog                30 self.label1.Location =
12                                           Point(20,20)
13 pantalla =                                31 self.label1.Height = 25
    Screen.GetWorkingArea(self)                32 self.label1.Width =
14 self.Height = pantalla.Height              self.Width
    / 5                                           33
15 self.Width = pantalla.Width /              34 form = HolaMundo()
    5                                           35 Application.Run(form)
```

de cifrado..., a las que hay que sumar aquéllas que ha incorporado el proyecto Mono, como por ejemplo las que nos permiten empotrar el motor Gecko (de Firefox) en una aplicación gráfica, de forma que podemos crear un navegador web con muy poco código.

Veamos un ejemplo de uso, vamos a abrir un fichero y a mostrarlo e impri-

mirlo en el terminal, ver Listado [1]. No es muy complicado ¿verdad? Lo más importante de este ejemplo es la sentencia:

```
import clr
```

que nos permite hacer uso de todas las librerías de .NET.

## Hola mundo Winforms

Hasta ahora todo lo que hemos hecho era fácilmente realizable con Python, a partir de este momento observaremos cambios. Vamos a realizar el mismo «Hola Mundo» pero empleando WinForms. Microsoft ha simplificado bastante el desarrollo de aplicaciones gráficas con esta librería. IronPython lo ha simplificado aún más.

### Listado 3: Visor de texto simple

```

01 import clr                                29
02                                           self.panel1.Controls.Add(self.
   clr.AddReference('System.Drawi         label1)
   ng')                                    30
03                                           self.panel1.Controls.Add(self.
   clr.AddReference('System.Windo         label2)
   ws.Forms')                              31
04                                           self.panel1.Controls.Add(self.
05 from System.IO import *                 boton1)
06 from System.Drawing import             32
   Color, Point                             self.panel1.Controls.Add(self.
07 from System.Windows.Forms             areaTexto)
   import (Application,                    33
   BorderStyle, Button, Form,             34
   FormBorderStyle, Label, Panel,         self.Controls.Add(self.panel1)
   Screen, OpenFileDialog,               35
   DialogResult, TextBox,                36 def generaAreaTexto(self):
   ScrollBars)                             37     texto = TextBox()
08                                           38     texto.Height = self.Height /
09 class LectorTXT(Form):                 2
10 def __init__(self):                     39     texto.Width = self.Width -
11     self.Text = "Visor de texto         30 # para que no se salga
   Linux Magazine"                         40     texto.Location =
12     self.FormBorderStyle =             Point(20,110)
   FormBorderStyle.FixedDialog            41     texto.Multiline = True
13                                           42     texto.ScrollBars =
14     pantalla =                          ScrollBars.Vertical
   Screen.GetWorkingArea(self)            43     self.areaTexto = texto
15     self.Height = 300                   44
16     self.Width = 400                     45 def generaLabel1(self):
17                                           46     self.label1 = Label()
18     self.panel1 = Panel()               47     self.label1.Text = "Lector
19     self.panel1.Location = Point         de ficheros de texto Linux
   (0,0)                                    Magazine"
20     self.panel1.Width =                 48     self.label1.Location =
   self.Width                               Point(20,20)
21     self.panel1.Height =                 49     self.label1.Height = 25
   self.Height                             50     self.label1.Width =
22     self.panel1.BorderStyle =           self.Width
   BorderStyle.FixedSingle                51
23                                           52 def generaLabel2(self):
24     self.generaLabel1()                 53     self.label2 = Label()
25     self.generaLabel2()                 54     self.label2.Text = "Fichero
26     self.generaBoton1()                 seleccionado: ???"
27     self.generaAreaTexto()             55     self.label2.Location =
28                                           Point(20,50)
                                           56     self.label2.Height = 25
                                           57     self.label2.Width =
                                           self.Width
                                           58
                                           59 def generaBoton1(self):
                                           60     self.boton1 = Button ()
                                           61     self.boton1.Name= 'Botón 1'
                                           62     self.boton1.Text = 'Abrir
   fichero'
                                           63     self.boton1.Location =
                                           Point(20,80)
                                           64     self.boton1.Height = 25
                                           65     self.boton1.Width = 100
                                           66     self.boton1.Click +=
                                           self.abreFichero
                                           67
                                           68 def abreFichero(self, sender,
   event):
                                           69     color = OpenFileDialog()
                                           70     color.Filter = "Ficheros txt
   (*.txt)|*.txt"
                                           71     color.Title = "Selecciona un
   fichero de texto"
                                           72
                                           73     nombre = ""
                                           74
                                           75     if (color.ShowDialog() ==
   DialogResult.OK ):
                                           76     nombre = color.FileName
                                           77     self.label2.Text = "Fichero
   seleccionado: " + nombre
                                           78 # cargamos el texto
                                           79     fichero =
   File.OpenText(nombre)
                                           80     texto = ""
                                           81
                                           82     s = fichero.ReadLine()
                                           83     while s :
                                           84     texto += s
                                           85     s = fichero.ReadLine()
                                           86
                                           87     self.areaTexto.Text = texto
                                           88
                                           89 form = LectorTXT()
                                           90
                                           91 Application.Run(form)

```

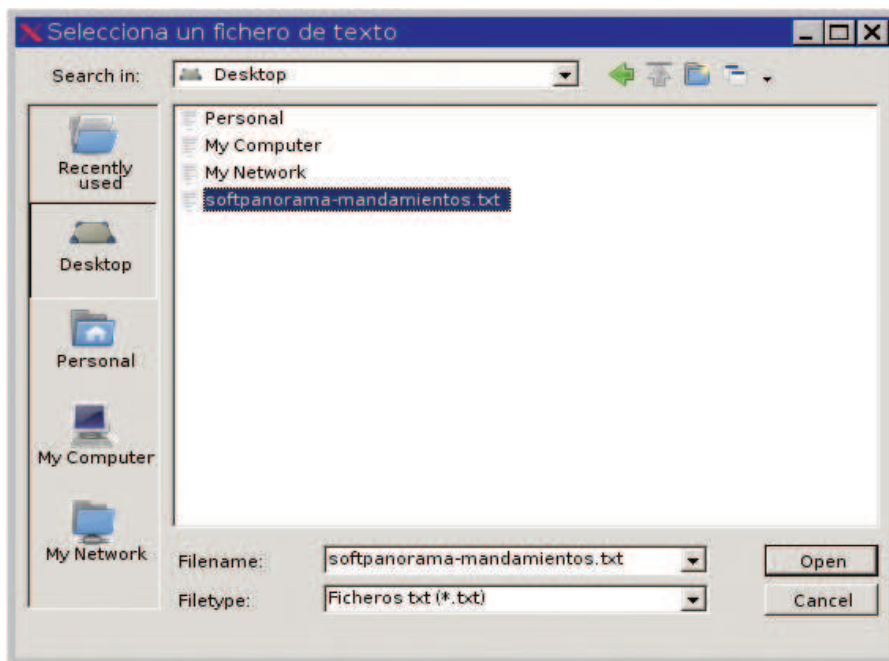


Figura 3: Diálogo de elección de fichero.

WinForms, ver Recurso [6], funciona de forma similar a como lo hacen otras librerías gráficas. Creas una ventana, dentro de la cual creas un panel, dentro del cual se pueden disponer widgets. Es muy parecido a las muñecas rusas Matroskas. Veamos el código en el Listado [2] y el resultado en la Figura [1].

Comencemos por el principio. Una vez importado *clr* hemos de hacer algo que no es normal en Python: debemos añadir referencias a las librerías de .NET que vamos a utilizar. Para ello utilizamos el método *clr.AddReference()* con el nombre de la librería que vamos a usar.

Winforms hace uso de «System.Drawing» y de «System.Windows.Forms». Estas librerías contienen todos los widgets necesarios para crear aplicaciones gráficas. Una vez que añadamos las referencias a estas librerías podemos usarlas como cualquier librería de Python.

Importamos todos los widgets necesarios: *Application*, *BorderStyle*, *Button*... Para hacer referencias a posiciones en la pantalla emplearemos la clase *Point* de *System.Drawing*, expresando la posición en pixels.

Con todas las librerías cargadas, podemos comenzar inicializando nuestra clase *HolaMundo*, que representa la ventana de la aplicación. Comenzamos dándole título, con *self.Text*, a la ventana. Definimos el tipo de ventana que utilizaremos con *FormBorderStyle* indi-

cando que será fijo, nuestra ventana no se podrá redimensionar.

Calculamos el tamaño de la ventana en base al de la pantalla. Conseguimos los datos de la pantalla mediante el método *Screen.GetWorkingArea()*, y hacemos que nuestra ventana tenga un quinto de la altura (*Height*) y ancho (*Width*) de la pantalla. Podríamos haber indicado el tamaño mediante un número, digamos 100 pixels.

Creamos un panel que pasará a contener todos los widgets que utilizemos. De nuevo ajustamos su altura y anchura, así como su posición dentro de la ventana. Como queremos que ocupe toda la superficie de la ventana lo posicionamos en (0,0), y le damos el mismo ancho y la misma altura que la ventana. He añadido un método, para simplificar el código, que genera una etiqueta donde realizamos el saludo. De nuevo el proceso es repetitivo: texto de etiqueta, posición, altura y anchura.

Por último, añadimos la etiqueta al panel y el panel a la ventana (esto último mediante *self.Controls.Add()*). Con estas últimas sentencias terminamos de definir nuestra clase.

Para poder hacer uso de ella creamos una instancia de *HolaMundo* y se la pasamos a *Application.Run()*, que es un bucle sin fin que se dedicará a gestionar los eventos sobre la ventana.

La explicación es bastante más larga que el texto, pero el lector se habrá dado

cuenta de lo simple que es realmente el proceso. Incluso llega a ser aburrido por repetitivo.

Pero hemos logrado nuestro objetivo, realizar una aplicación gráfica con un mínimo de líneas de código. Vayamos a algo más interesante.

## Conclusión

Poder acceder a la enorme librería de .NET con IronPython nos permite crear aplicaciones gráficas multiplataforma. Una posibilidad realmente esperanzadora para todos aquellos que quieran llevar sus desarrollos en Python de un equipo a otro.

El lector puede profundizar en el desarrollo de aplicaciones que empleen WinForms desde IronPython en el Recurso [7]. Es un tutorial, en inglés, donde se da un repaso a los conocimientos básicos de desarrollo de aplicaciones gráficas mediante WinForms.

Python está consiguiendo con IronPython atraer a gran número de desarrolladores de otros sistemas operativos, de forma que se están comenzando a crear aplicaciones de las que nosotros podremos disfrutar en Linux gracias a Mono.

.NET se va estableciendo poco a poco en el mundo empresarial como un estándar a tener en cuenta. Pero esto no debe atemorizar a los que usen Python. Tanto si triunfa .NET o Java, IronPython y Jython están ahí para que Python siga vigente y demostrando al mundo que la programación no tiene por qué ser complicada. ■

## RECURSOS

- [1] <http://www.codeplex.com/IronPython>
- [2] <http://blogs.msdn.com/huginin/archive/2006/09/05/741605.aspx>
- [3] <http://tirania.org/blog/archive/2007/Jan-11-1.html>
- [4] <http://www.mono-project.com/Libgdiplus>
- [5] <http://www.codeplex.com/Project/License.aspx?ProjectName=IronPython>
- [6] <http://www.mono-project.com/WinForms>
- [7] <http://www.voidspace.org.uk/ironpython/winforms/index.shtml>
- [8] Los listados de este artículo: <http://www.linux-magazine.es/Magazine/Downloads/25/Python>