Jython: Mezclando Java y Python

Autor:	Milton Labanda
Bio:	Ingeniero en Informática (Universidad Técnica Particular de Loja), Master de Software Libre (Universitat Oberta de Catalunya). Actualmente Técnico y Docente de la Carrera de ingeniería en sistemas en la Universidad Nacional de Loja. Coordinador de la Comunidad de Software Libre ESOL-UNL.
Web:	http://1000tonlab.wordpress.com
Email:	1000ton.lab@gmail.com
Twitter:	@miltonlab
Identi.ca:	miltonlab
Facebook:	Milton Lab

Algunos desarrolladores, por no decir muchos, han coincidido en que “programar en Java no es lo mismo que programar en Python”, o viceversa, “programar en Python no es lo mismo que programar en Java”. Hoy en día gracias a proyectos como **Jython** o **JPype** se puede usar lo mejor de cada lenguaje en una misma aplicación.

Introducción

Los avances dentro del desarrollo de herramientas relacionadas con los diversos lenguajes de programación libres y/o de código abierto en estos tiempos, a mi criterio, están dando lugar a la programación colaborativa entre mas de un lenguaje a través de las diferentes liberías, implementaciones, APIs, bindings o como se las quiera llamar.

¿Que es Jython?

Uno los casos comentados en la introducción es Jython, una implementación de Java sobre Python. Empezamos entonces dando una definición formal, clara y concisa de lo que es y lo que permite Jython, de acuerdo al libro La Guia Definitiva de Jython de la Editorial Apress:

` “Jython es una implementación del lenguaje Python para la plataforma Java ... Jython trae el poder del lenguaje Python hacia la Maquina Virtual de Java. Provee a los desarrolladores Java la habilidad de escribir de escribir código productivo y dinámico usando una sintaxis elegante. Asi mismo permite a los desarrolladores Python ganar ventaja de la utilidad de las librerias y APIs que la JVM (Maquina Virtual de Java) tiene para ofrecer”. `

El Caso de Uso: Usando Swing desde Python

Uno de los campos donde python ofrece diversidad de posibilidades y alternativas es la creación de GUIs para aplicaciones de escritorio, en donde a través de los “bindigns” respectivos podemos usar desde **Tk**(Tkinter) pasando por **wxWindows** y llegando hasta **Gtk** o **Qt** u otras alternativas. Ahora bien si nos trasladamos al mundo Java nos encontramos en cambio con únicamente dos posibilidades **AWT** o **SWING**, aunque existe **SWT** parte del proyecto eclipse y no viene incluido en el API estandar de la plataforma. Entonces a más de uno pudo haberle surgido la interrogante de: Si se puede usar Swing en o desde python?. La respuesta es si, es por ello que en este articulo nos ocuparemos de dar una visión considerable de la aplicabilidad que tienen el poder usar Jython para permitir la creación de interfaces gráficas que incluyan widgets o componentes de las librerías Swing perteneciente a la plataforma Java con la sintaxis de Python.

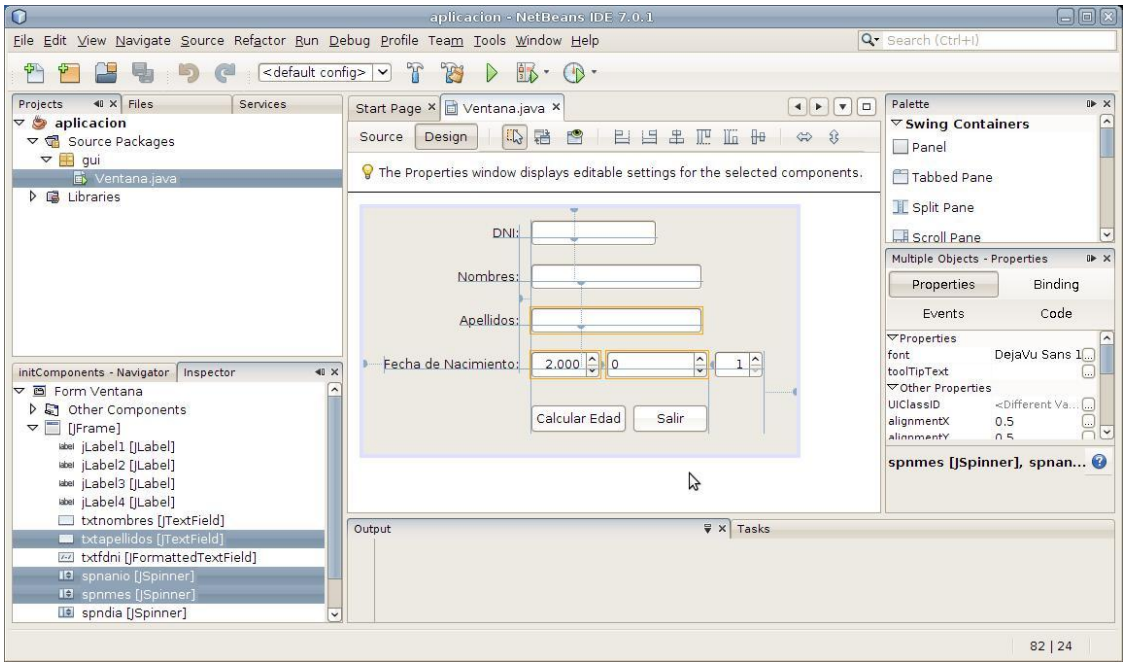
A pesar de que podemos construir íntegramente toda una aplicación usando código Jython, existe la interesante posibilidad y ventaja de crear la GUI totalmente en Java aprovechando de esta manera las herramientas e IDEs que existen para generación automática de Interfaces gráficas de usuario, tal es el caso de NetBeans o Eclipse con el plugin Visual Editor por ejemplo.

Construyendo la Vista del Ejemplo

Para ejemplificar el uso de Swing en una aplicación Jython hemos escogido algo muy simple como el cálculo de edad de una persona dada su fecha de nacimiento a más de calcular el tiempo exacto que falta para su próximo cumpleaños.

Aunque la temática de la interfaz gráfica al igual que la aplicación en general es de muchísima sencillez, hemos decidido usar el IDE Netbeans con el fin de ejemplificar las ventajas de poder generar con la ayuda de asistentes la GUI en un lenguaje y el resto de componentes de la aplicación en otro lenguaje de programación.

Veamos ahora creación de la Interfaz Gráfica de Usuario con el asistente de **Netbeans**:



Vale la pena señalar que aprovechando las novísimas características de la versión 7 de Java hemos fijado a**Nimbus** como tema de presentación de Swing (`javax.swing.plaf.nimbus.NimbusLookAndFeel`) incluido en esta ultima versión, con el siguiente código dentro de la clase creada con el designer:

```
    private void fijarLookNimbusJava7(){
            /* Set the Nimbus look and feel */
            //<editor-fold defaultstate="collapsed" desc="
    Look and feel setting code (optional) ">
            /* If Nimbus (introduced in Java SE 6) is not
    available, stay with the default look and feel.
             * For details see
    http://download.oracle.com/javase/tutorial/uiswing/loo
    kandfeel/plaf.html
             */
            try {
                for (javax.swing.UIManager.LookAndFeelInfo
    info :
    javax.swing.UIManager.getInstalledLookAndFeels()) {
                    if ("Nimbus".equals(info.getName())) {

    javax.swing.UIManager.setLookAndFeel(info.getClassName
    ());
                        break;
                    }
                }
            } catch (ClassNotFoundException ex) {

    java.util.logging.Logger.getLogger(Ventana.class.getNa
    me()).log(java.util.logging.Level.SEVERE, null, ex);
            } catch (InstantiationException ex) {
```

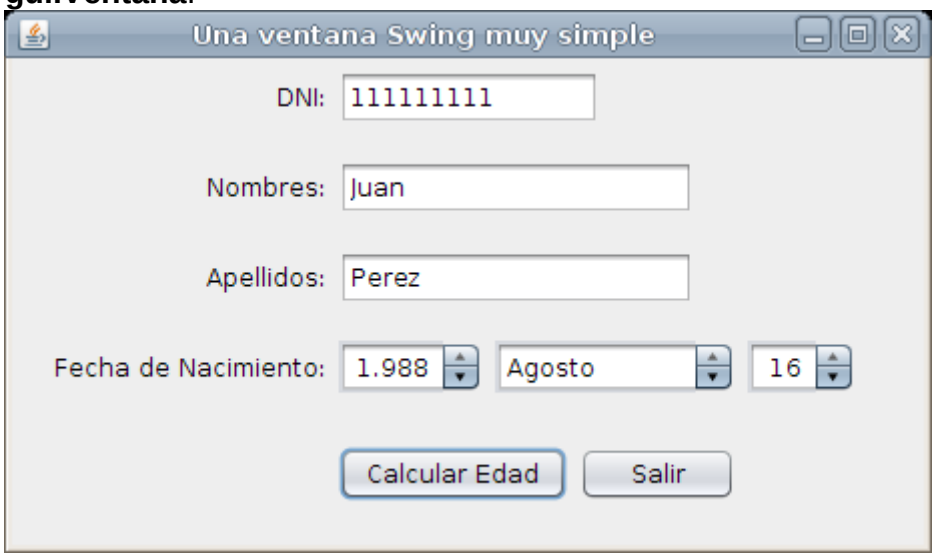
```
java.util.logging.Logger.getLogger(Ventana.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Ventana.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch
(javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Ventana.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        }
}
```

Por otro lado es necesario definir como propiedades (con los métodos set y get) a los componentes de la GUI (mostramos algunos, pero el ide ayuda a crearlos a todos de una manera rapida y sencilla):

```
/**
 * @return the btncalcular
 */
public javax.swing.JButton getBtncalcular() {
    return btncalcular;
}

/**
 * @param btncalcular the btncalcular to set
 */
public void setBtncalcular(javax.swing.JButton
btncalcular) {
    this.btncalcular = btncalcular;
}

/**
 * @return the btnsalir
 */
public javax.swing.JButton getBtnsalir() {
    return btnsalir;
}

/**
 * @param btnsalir the btnsalir to set
 */
public void setBtnsalir(javax.swing.JButton btnsalir)
{
    this.btnsalir = btnsalir;
}

/**
 * @return the spnanio
 */
public javax.swing.JSpinner getSpnanio() {
    return spnanio;
}

/**
```

```
     * @param spnanio the spnanio to set
     */
    public void setSpnanio(javax.swing.JSpinner spnanio) {
        this.spnanio = spnanio;
    }
```

Una vista previa de la vista de la aplicación generada en la clase **java gui.Ventana**:



Programando la lógica de la aplicación en Python

Para generalizar la solución del problema relacionado con el cálculo de la edad, hemos reducido la lógica central a una resta de fechas de acuerdo a nuestro calendario. Antes de mostrar el código en Python mostraremo la lógica del algoritmo utilizado.

Aunque pueden haber muchas otras soluciones usando por ejemplo la clase `datetime.timedelta` hemos escogido usar un algoritmo que sea lo más generalizado para este tipo de situaciones. Así, si queremos saber el tiempo que hay entre las fechas *2011-08-12* y *1975-09-13*, el procedimiento que aplicamos es:

```
  2011 - 08 - 12      31 - 1 = 30 (days)    Respuesta : 35 years, 2 months 30 days
  1975 - 09 - 13      12 - 2 = 2 (months)
  ---------------
    35 -(-2)-(-1)
```

Cuando el día es negativo aumentamos uno al mes de la fecha menor y lo restamos del numero total de días del mes anterior de la fecha mayor, en este caso 31 (por el mes 07 es decir Julio). Asi mismo si el mes resulta negativo le sumamos uno al año de la fecha mayor y lo restamos de 12 para obtener el mes real resultante.

Ahora el código en Python:

```
# Archivo: logic.py
def restarfechas(fecha1, fecha2):
    """
    Resta dos objetos datetime.date, el menor del
mayor
    Devuelve una tupla con el numero de anios, meses y
dias
    """
    f1,f2 = max(fecha1,fecha2),min(fecha1,fecha2)
    dias = f1.day - f2.day
    # se devuelve 1 a los meses en caso de ser
negativos los dias
    f2 = f2.replace(month = f2.month + 1 if dias < 0
else f2.month)
    # numero de dias que tiene el mes anterior de la
primera fecha
    if f1.month == 1:
        dias_mes_anterior =
calendar.monthrange(f1.year-1,12)[1]
    else:
        dias_mes_anterior =
calendar.monthrange(f1.year,f1.month-1)[1]
    dias = dias_mes_anterior + dias if dias < 0 else
dias
    meses = f1.month - f2.month
    # se devuelve 1 a los anios en caso de ser
negativos los meses
    f2 = f2.replace(year = f2.year + 1 if meses < 0
else f2.year)
    meses = 12 + meses if meses < 0 else meses
    anios = f1.year - f2.year
    return (anios, meses, dias)
```

De esta manera entonces reduciremos el cálculo de la edad y del tiempo del próximo cumpleaños prácticamente a una simple resta de fechas:

```
def edad(nacimiento):
    """ Devuleve la edad en una tupla de anios, meses
y dias """
    hoy = datetime.date.today()
    return restarfechas(hoy,nacimiento)

def cumpleanios(nacimiento):
    """ Devuelve el tiempo restante para el siguiente
cumpleanios """
    hoy = datetime.date.today()
    cumpleanios =
datetime.date(hoy.year,nacimiento.month,nacimiento.day
)
    if cumpleanios < hoy:
    cumpleanios = cumpleanios.replace(year = hoy.year
+ 1)
    faltan = restarfechas(cumpleanios,hoy)
    return (faltan[1],faltan[2])
```

Enlazando las partes de la aplicación con Jython

Finalmente enlazamos todo el escenario con un script en el cual básicamente creamos los manejadores Swing o AWT para que procesen los eventos y las interacciones del usuario interconectando con la lógica ya programada en Python puro.

Mostramos la clase principal por decirlo así, en donde creamos la GUI y registramos los manejadores de eventos, evidenciando además el uso de las librerías de la JVM, lo cual puede ser interpretado por Jython:

```
# Archivo: aplicacion.py
from java.awt.event import ActionListener
from javax.swing.event import ChangeListener
from javax.swing import JOptionPane
from java.lang import Integer

import datetime, calendar

import gui, logic

class Aplicacion:

    form = None

    @staticmethod
    def run():
        Aplicacion.form = gui.Ventana()
        Aplicacion.form.visible = True

Aplicacion.form.btnsalir.addActionListener(ManejadorSa
lir())

Aplicacion.form.spnmes.addChangeListener(ManejadorMes(
))

Aplicacion.form.btncalcular.addActionListener(Manejado
rCalcular())
```

Para muestra el manejador de eventos que ejecuta el calculo de la edad:

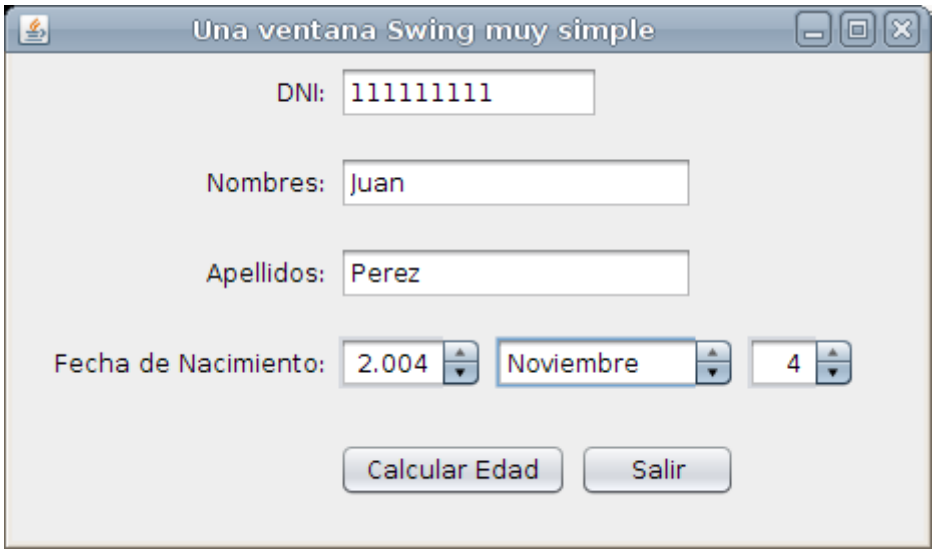
```
class ManejadorCalcular(ActionListener):
    def actionPerformed(self,ev):
        mesint =
Aplicacion.form.spnmes.model.list.indexOf(Aplicacion.f
orm.spnmes.value) + 1
        fecha_n =
datetime.date(Aplicacion.form.spnanio.value,mesint,Apl
icacion.form.spndia.value)
        edad = logic.edad(fecha_n)
```

```
        nombre = '%s %s' %
(Aplicacion.form.txtnombres.text,Aplicacion.form.txtap
ellidos.text)
        msg1 = '%s tiene %d anios %d meses %d dias' %
(nombre,edad[0], edad[1], edad[2])
        JOptionPane.showMessageDialog(None, msg1)
        cumple = logic.cumpleanios(fecha_n)
        meses,dias = cumple[0],cumple[1]
        if meses == 0 and dias == 0:
            msg2 = 'Hoy es su cumpleanios.
Felicitaciones :)'
        else:
            msg2 = 'Faltan %d meses %d dias para su
cumpleanios' % (cumple[0],cumple[1])
        JOptionPane.showMessageDialog(None, msg2)
```

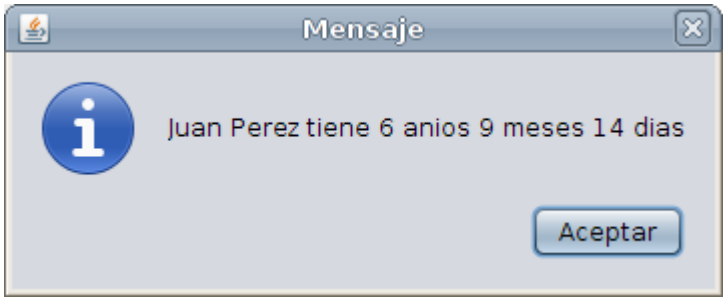
A continuación unas muestras de la ejecución de la aplicación (si la fecha actual es 18 de agosto del 2011) que se debe ejecutarse con:

```
miltonlab@debianlab:~/edadjpy$ jython aplicacion.py
```

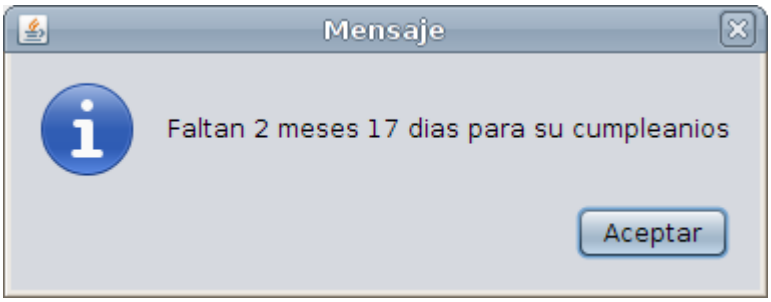
Ventana inicial



Presentación de la edad



Tiempo restante para el cumpleaños



Para aquellos que deseen revisar el código completo de la aplicación pueden encontrarlo enhttp://docencia-programacion.googlecode.com/files/edadjpy.tar.gz

Las versiones del software con el cual se ha desarrollado son: Java 6 (la versión 7 si se desea usar el Look And Feel “Nimbus'), **Python 2.6.6** y **Jython 2.5.1**. El Sistema Operativo utilizado para la ejecución fue**Debian 6.0**