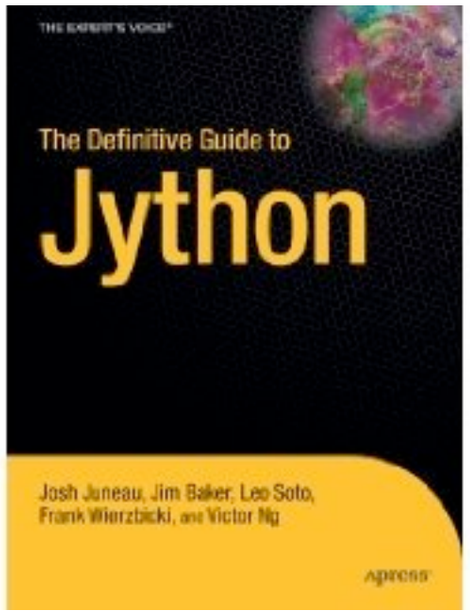
Extending Java Applications with

Frank Wierzbicki
frank@saucelabs.com
http://fwierzbicki.blogspot.com
twitter: fwierzbicki

Definitive Guide to Jython



Josh Juneau, Jim Baker, Leo Soto,
Frank Wierzbicki, and Victor Ng
http://jythonbook.com

Jython calls into Java very nicely

```java
import javax.swing.JFrame;

public class Hello {

    public static void main(String[] args) {
        JFrame f = new JFrame("Hello from Java");
        f.setSize(300, 300);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setVisible(true);
    }

}
```

```
from javax.swing import JFrame

f = JFrame("Hello from Jython",
           size = (300,300),
           defaultCloseOperation = JFrame.EXIT_ON_CLOSE)
f.visible = True
```

Jython classes inherit from Java easily

```
from javax.swing import JFrame

class MyFrame(JFrame):
    def __init__(self, *args, **kw):
        JFrame.__init__(self, *args, **kw)
        print "I was inited"

f = MyFrame("Hello from Jython",
        size = (300,300),
        defaultCloseOperation = JFrame.EXIT_ON_CLOSE)
f.visible = True
```

Java calling into Jython is a bit more difficult

- Currently there is no one standard way to do Java to Jython calling
- most common: org.python.util.PythonInterpreter + inherit from a Java interface or class.

A Java interface

```
public interface Calculator {
    int add(int a, int b);

    int subtract(int a, int b);
}
```

Extend With Jython

```
import Calculator

class JyCalculator(Calculator):
    def add(self, a, b):
        return a + b

    def subtract(self, a, b):
        return a - b

if __name__ == '__main__':
    c = JyCalculator()
    print c.add(2,2)
    print c.subtract(3,1)
```

Call With PythonInterpreter

```
import org.python.core.PyObject;
import org.python.util.PythonInterpreter;

public class CalcDemo {
    public static void main(String[] args) {
        PythonInterpreter interpreter = new PythonInterpreter();
        interpreter.exec("from calc import JyCalculator");
        PyObject klazz = interpreter.get("JyCalculator");
        PyObject o = klazz.__call__();
        Calculator c = (Calculator)o.__tojava__(Calculator.class);
        System.out.println(c.add(2, 2));
    }
}
```

PlyJy

- http://kenai.com/projects/plyjy
- Wraps up PythonInterpreter and other mess into a nice object factory

```java
import org.plyjy.factory.JythonObjectFactory;

public class PlyDemo {

    public static void main(String[] args) {
        JythonObjectFactory factory =
            JythonObjectFactory.getInstance();
        Calculator c = (Calculator)factory.createObject(
                Calculator.class, "JyCalculator");
        System.out.println(c.add(2,2));
    }
}
```

Making an Executable Jar

- Make a copy of jython.jar (e.g. myapp.jar)
- zip Jython Lib into myapp.jar
- put your .py files into a Lib dir
- Make a __run__.py that starts your app
- zip your Lib/*.py files into myapp.jar
- create a manifest file and zip into myapp.jar

Make a copy of jython.jar

```
$ cd $JYTHON_HOME
$ cp jython.jar myapp.jar
```

Zip in the Jython Lib

```
$ cd $JYTHON_HOME
$ zip -r myapp.jar Lib
```

Create manifest.txt

```
Main-Class: org.python.util.JarRunner
```

Put manifest into jar

```
$ jar ufm myapp.jar manifest.txt
```

Zip in the Jython Lib

```
$ cd $YOUR_APP_DIR
$ zip -r myapp.jar Lib
```

Alternatives

- Clamp http://github.com/fwierzbicki/clamp
- Jynx http://code.google.com/p/jynx

Where to find out more

http://www.jython.org
http://wiki.python.org/jython
http://fwierzbicki.blogspot.com
twitter: fwierzbicki
frank@saucelabs.com