Mixing Python and Java

How Python and Java can communicate and work together

EuroPython 2009 (June 30th 2009, Birmingham)
Andreas Schreiber <Andreas.Schreiber@dlr.de>
German Aerospace Center (DLR), Cologne, Germany
http://www.dlr.de/sc
Follow this presentation...

twitter.com/python_demo
Final Version of this Slides is available on SlideShare

http://tr.im/ep09java
DLR
German Aerospace Center

- Research Institution
- Space Agency
- Project Management Agency
Locations and employees

6000 employees across 29 research institutes and facilities at

- 13 sites.

Research Areas

- Aeronautics
- Space
- Transport
- Energy
- Space Agency
- Project Management Agency
Mixing Python and Java?

Is it all about

Jython

?
Mixing Python and Java?

Outline

Accessing Python from Java
- Jython
- JEPP

Accessing Java from Python
- JPype
- JCC

Inter-process communication
- CORBA
- SOAP
- Other remote object libraries
Java
Why People are Using Java?

- Widely used in many different application domains
  - industry/business, research, academia
- Available on many different platforms
- Good performance
  - good Garbage Collector (and no GIL)
- Many, many libraries
  - E.g., for data base access, XML processing, PDF generation, security, or user interfaces
- Availability of good documentation for all aspects of Java programming
- Very good development tools
  - Eclipse, NetBeans, IntelliJ IDEA, …
Python
Why People are Using Python?

Python in Research and Industry
➤ Scientists and engineers don’t want to write software but just solve their problems
➤ If they have to write code, it must be as easy as possible

Reasons for Python?
➤ Very easy to learn and easy to use
  (= steep learning curve)
➤ Allows rapid development
  (= short development time)
➤ Inherent great maintainability

“I want to design planes, not software!”
Python has the cleanest, most-scientist- or engineer friendly syntax and semantics.

-Paul F. Dubois

Why Mixing Java and Python?
Embedded scripting and more…

- Many mature Java applications and frameworks exist
  - Lots of commercial and Open Source software systems and libraries
  - For example, The Eclipse Universe, Apache-Software, Portal-Frameworks, Workflows systems, …

- Common use cases
  - Add embedded (Python) scripting to Java applications
  - Use Java libraries from Python code
Why Mixing Java and Python?
Integration of Python code into Java applications…

- Existing code or libraries exist either for Java or Python only
  - Effort to re-implement the functionalities could be very high
  - If the library is very well tested, it could be an enormous effort to reach a comparable level of quality

- Common use cases
  - Just use an existing library from the “other” language
  - Especially, use Python code (or C/Fortran/WHATEVER code with Python wrappers) from Java
  - Its harder to wrap C codes in Java than in Python (see CTypes)
Example
QF-Test – Automated GUI Testing

![QF-Test GUI screenshot]

Client
SUT

SUT Skript

Skriptsprache
Jython

GUI Engine

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>log name</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td></td>
</tr>
</tbody>
</table>

Verzögerung vorher (ms) | Verzögerung nachher (ms)

Bemerkung

```plaintext
1 # Textfeld Komponente holen
2 field = rc.getComponent("tfName")
3 # Wert auslesen und loggen
4 rc.logMessage("Name: " + field.getText())
```
Example
TENT – Software Integration and Workflow Management
Use Cases for Python Scripting (1)

Steering and controlling the program by user defined scripts
- Applications are complex parameter variations or steering multidisciplinary coupled simulations

Automation of repeating tasks
- In most cases, this is called “macro” recording and replaying

Extending user interfaces
- For example, with additional customized dialogs and other extensions
- In Java, this requires scripts which use the Java GUI libraries (AWT, Swing, or SWT)

Integration of additional legacy tools
- Important and widely used for integration and workflow systems
- End users can integrate external codes without changing the Java program itself
Use Cases for Python Scripting (2)

Interactive experimentation and debugging of the Java program

- Can be easily done with an embedded interactive interpreter
- Allows end users and software developers to debug and analyze the Java program during runtime

Creating automated tests for quality assurance

- Recording user actions during runtime of the Java program as a Python script ("Journaling")
- Editing and generalizing the recorded script
- Replaying the script manually or automatically
Example: Codes with Python Interfaces

Numerical Simulation Software in C++ or Fortran

Examples for high-definition CFD-Solver with Python interfaces

- DLR TAU-Code (http://www.dlr.de/as)
- ONERA elsA-Code (http://elsa.onera.fr)

Integration into Java workflow systems with the following techniques…
Tools for Python-Java-Integration
Accessing Python from Java

Use cases

- The application should have embedded scripting functionality.
- The application should use an external code written in Python.
- The application should use an external code written in Python or other languages such as C, C++, Fortran.

Tools

- Jython
- JEPP
Jython

- Complete re-implementation of the Python interpreter in Java
- Python code runs in the Java VM
- Website: http://www.jython.org
- Latest version: Jython 2.5
- For details & questions, catch
  - Tobias Ivarsson
  - Frank Wierzbicki
  - and others here at EuroPython 😊
Jython
Code Example 1: Java code

> Execute Python code

```java
import org.python.util.PythonInterpreter ;
import org.python.core.*;

class TestClass {
    public static void main(String[] args) {
        try {
            org.python.util.PythonInterpreter python =
            new org.python.util.PythonInterpreter ();
            python.execfile("python_script.py");
        } catch (Exception e) {
            System.out.println("Some error!");
        }
    }
}
```
Jython

Code Example 2: Python code

Use Swing

```python
from javax.swing import *

frame = JFrame("Hello Jython")
label = JLabel("Hello Jython!", JLabel.CENTER)
frame.add(label)
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
frame.setSize(200, 100)
frame.show()
```

![Image of Hello Jython window]
JEPP
Java Embedded Python

- Embeds CPython interpreter via Java Native Interface (JNI) in Java
- Python code runs in CPython
- Website: http://jepp.sourceforge.net/
JEPP
Code Example

- Execute ("evaluate") Python statements

```java
Jep jep = new Jep(false, SCRIPT_PATH, cl);
jep.eval("print 'hello'");
jep.close();
```

- Execute a Python script file

```java
Jep jep = new Jep(false, SCRIPT_PATH, cl);
jep.runScript(SCRIPT_PATH + file);
jep.close();
```
Accessing Java from Python

Use cases

- The application should use an external Java application.
- The application should use a Java library.

Tools

- JType
- JCC
JPyte
Java to Python Integration

- Interface on native level of both (Java & Python) Virtual Machines/Interpreters
- Starts a Java Virtual Machine

- Website:  
JPype
Code Example (1)

Hello World

```python
from jpype import *
# Start JVM
startJVM (path to jvm.dll, "-ea")
# Print "Hello World"
java.lang.System.out.println("Hello World")
# Shutdown JVM
shutdownJVM()
```
JPype
Code Example (2)

Call Java methods from Python

```python
import jpype
# Start JVM
jpype.startJVM(path to jvm.dll, "-ea")
# Create reference to java package
javaPackage = jpype.JPackage("JavaPackageName")
# Create reference to java class
javaClass = javaPackage.JavaClassName
# Create instance of java class
javaObject = javaClass()
# Call java methods
javaObject.JavaMethodName()
# Shutdown JVM
jpype.shutdownJVM()
```
JCC
PyLucene's Code Generator

- C++ code generator for calling Java from C++/Python
- C++ object interface for wrapping a Java library via JNI
  - Generates complete CPython extension
- Supported on Mac OS X, Linux, Solaris and Windows, requires C++ compiler

- Website: http://lucene.apache.org/pylucene/jcc
  - Part of PyLucene

Python Interpreter

Python Code

Generated C++ Wrapper Code

JNI

Java Application Code

pythonExtension()

Java Virtual Machine
JCC

Code Example

- Requirement: VM initialization

```python
import jcc
# Start JVM
jcc.initVM(maxheap='2000m', ...)```
JCC Code Example
Using PyLucene: Search Lucene Index

Search for "Query" in directory "index"

```python
from lucene import QueryParser, IndexSearcher, StandardAnalyzer, FSDirectory, Hit, VERSION, initVM, CLASSPATH

initVM(CLASSPATH)
directory = FSDirectory.getDirectory("index", False)
searcher = IndexSearcher(directory)
analyzer = StandardAnalyzer()
command = raw_input("Query:")
query = QueryParser("contents", analyzer).parse(command)
hits = searcher.search(query)
for hit in hits:
    doc = Hit.cast_(hit).getDocument()
    print 'path:', doc.get("path"), 'name:', doc.get("name")
searcher.close()
```
Inter-Process Communication

- Inter-process communication (IPC) is data exchange between different processes on one or more computers connected by a network.

- Typical IPC techniques are:
  - remote procedure calls (RPC)
  - message passing.

- The most common APIs are:
  - Object Request Broker (ORB) or
  - Web Services protocols based on XML.

- A drawback is the need for additional services, such as a directory service for registration and location of remote objects:
  - Naming Service for CORBA
  - Universal Description, Discovery and Integration (UDDI) for Web Services
Object Request Broker (ORBs)
CORBA
CORBA Implementations for Python

**Fnorb**
- Pure Python implementation that works with CPython and Jython.
- It is a light-weight CORBA implementation and interoperable with the standard Java ORB.
- The development of Fnorb has ended.

**omniORB**
- An ORB implementation for C++ and Python that can be used from CPython.
- The Python ORB (omniORBpy) uses the C++ implementation of omniORB.
- omniORB is actively developed with regular releases.
- Website: [http://omniorb.sourceforge.net](http://omniorb.sourceforge.net)
CORBA Example

Interface

Hello World interface in Interface Definition Language (IDL)

```idl
module HelloWorld {

    const string Message = "Hello CORBA World!";

    interface Hello {
        string hello_world();
    }
};
```
CORBA Example
Python Server (Fnorb): Implementation of Interface

```python
# Standard/built-in modules.
import sys

# Fnorb modules.
from Fnorb.orb import BOA, CORBA

# Stubs and skeletons generated by 'fnidl'.
import HelloWorld, HelloWorld_skel

class HelloWorldServer(HelloWorld_skel.Hello_skel):
    """ Implementation of the 'Hello' interface. ""

    def hello_world(self):
        print HelloWorld.Message
        return HelloWorld.Message
```
def main(argv):
    # Initialise ORB and BOA
    orb = CORBA.ORB_init(argv, CORBA.ORB_ID)
    boa = BOA.BOA_init(argv, BOA.BOA_ID)
    # Create object reference
    obj = boa.create('fred', HelloWorldServer._FNORB_ID)
    # Create an instance of the implementation class.
    impl = HelloWorldServer()
    # Activate the implementation
    boa.obj_is_ready(obj, impl)
    # Write the stringified object reference to a file
    open('Server.ref', 'w').write(orb.object_to_string(obj))
    boa._fnorb_mainloop()  # Start the event loop
    return 0
CORBA Example
Java Client

```java
public class Client {
    public static void main ( String args[] ) {
        java.util.Properties props = System.getProperties();
        try {
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,props);
            org.omg.CORBA.Object obj = null;
            java.io.BufferedReader in = new java.io.BufferedReader(
                new java.io.FileReader("Server.ref"));
            String ref = in.readLine();
            obj = orb.string_to_object(ref);
            Hello hello = HelloHelper.narrow(obj);

            hello.hello_world();
            orb.destroy();
        } catch (Exception e) { e.printStackTrace(); }
    }
}
```
Web Services: SOAP

WSDL

WSDL Compiler

Python Application Code (Client)

Generated Python Code (Stub)

Python Interpreter

Generated Java Code (Skeleton)

Java Application Code (Server)

Servlet Container

Java Virtual Machine

WSDL Compiler
Web Services for Python

Zolera SOAP Infrastructure (ZSI)

For Python the only remaining SOAP toolkit of considerable quality is the Zolera SOAP Infrastructure (ZSI)
Other Remote Object Libraries

Python Remote Objects (Pyro)
- Pyro is similar to Java's Remote Method Invocation (RMI).
- It is simple, very portable, and works with Jython.
- Brings its own Naming Service for locating remote objects.
- Pyro is actively developed with regular new releases.

Simple Python Interface to Remote Objects (SPIRO)
- SPIRO implements an ORB and is developed as a bridge between CPython and Jython.
- The development has ended.
Conclusions

I ❤️ Jython

… but there are alternatives which makes sense for certain use cases!
Questions?

Contact

http://tr.im/schreiber

twitter.com/onyame

python@dlr.de