Jython

Python on the JVM

Robert Bachmann

JSUG Meeting #49

Outline

- Use Cases
- Limitations
- Usage & `javax.script` / JSR223
- A (short) case study

Python?

- Scripting language
- Dynamically typed
- Standard interpreter: (C)Python

A broader view…

- JRuby (Ruby on the JVM)
- Groovy (native JVM language)
- CLR: IronPython & IronRuby

Use cases

- Re-use of Java libraries
- Re-use of Java infrastructure
- Adding scripting abilities to a Java software
- Prototyping
- Test scripting
- Performance improvements w.r.t (C)Python

Limitations

- Not a Java replacement (cf. Scala)
- No current compiler
- Can not use Python modules with C code
- Jython lags behind (C)Python & IronPython
  - Jython: 2.5.3 stable / 2.7a2 alpha
  - (C)Python: 2.7 stable
  - IronPython: 2.7 stable
- Performance worse than Java

Usage

- “Hello World” with Jython
- Using Java from Jython
- Using Jython with `javax.script`
- Using Jython from Java
- Deployment options

Hello World

```
# program.py
print ”Hello␣World”
```

```
$ jython program.py
Hello World
```

Hello World with classes and modules

```
### demo.py
class Hello:
      def greet(self, name):
          print "Hello " + name

### program.py
from demo import Hello
h = Hello()
h.greet("JSUG")

# or:
import demo
h = demo.Hello()
h.greet("JSUG")
```

Using Java from Jython – Example

```
# Hello World with Swing

from javax.swing import JOptionPane

JOptionPane.showMessageDialog(None, ”Hello!”)
```

Using Java from Jython – Notes

- Jython classes can
  - implement Java interfaces
  - extend Java classes
- Classpath
  - `import` uses classpath via `sys.path`
  - Add JARs via `sys.path.append()`

javax.script

- JSR223: Scripting for the JavaTM Platform
- API for using scripting languages with Java
- Central class: ScriptEngine

javax.script

```
ScriptEngineManager factory =
 new ScriptEngineManager();
ScriptEngine engine =
 factory.getEngineByName(”python”);

engine.eval(”print␣’Hello,␣World’”);
engine.put(”x”, 10)
engine.eval(”y␣=␣x␣*␣2”);
Object y = engine.get(”y”)
System.out.println(y)
```

Using Jython classes from Java

- Steps:
  - Derive from Java class / interface
  - Use `JythonInterperter` to create an instance
  - Call the instance's `__tojava__` method
- Complete soltion:
  `http://www.jython.org/jythonbook/en/1.0/JythonAndJavaIntegration.html`

Deployment options

- Servlet Contaier
  - `org.python.util.PyServlet`
  - WSGI via modjy
- Standalone .jar

Case study

- Scenario: A C library with Java and .Net wrappers
- Challenge: Automated testing of all libraries
- Solution: Single-source test automation with Jython/IronPython

Questions?

Thanks

Twitter @robertbachmann

Email rb@ — .at