- det vassaste verktyget i Java-lådan

Tobias Ivarsson
tobias.ivarsson@neotechnology.com

Agenda

- Presentation av Tobias
- Introduktion till Python och Jython
- Vägen framåt för Jython
- Tobias arbete med Jython

Tobias Ivarsson

- Student vid Linköpings Universitet
- Javautvecklare på Neo Technology
- Google Summer of Code 2007 med Jython
- Committer i Jython-projektet
- Examensarbete kring Jython
  Supporting dynamic languages on the present and future implementations of the JVM

Grundläggande Python

- Ett Dynamiskt högnivåspråk
  Det som sker vid kompilering i C/Java sker i runtime i Python
  - Eval
  - Stöd för funktionell programmering
  - Runtime-modifiering av objekt och typer
  - Stora möjligheter till introspektion

Grundläggande Python - Typer

- *Dynamisk typhantering*
  - *Typ-kontroller sker sent, i runtime*
- *Duck typing*
  - Interface är enbart semantiskt
  - Har objektet en next()-metod kan vi iterera över det

Jython

- Version 2.2, kompatibel med Python 2.2
- Full möjlighet att utnyttja allt skrivet för Java
- Enkelt att bädda in för skriptning
- Kompilerar till Java Bytekod
- Aktiv utveckling (efter nästan sex år uppehåll)

Inbäddning av Jython

```
import javax.script.ScriptEngineManager;

import javax.script.ScriptEngine;

import javax.script.Invocable;

/* Provides a few shorthands, but gives less
   flexibility.
   This method is great for handing over to a
   script, and run it. If you want to integrate
   and interact, the other way will aid you
   more. */

public class EmbeddingScriptingEngine {

    public static void main(String[] args) {

        ScriptEngineManager mgr = new
          ScriptEngineManager();

        ScriptEngine py =
          mgr.getEngineByName("python");

        py.eval("from script import main");

        ((Invocable)py).invokeFunction(
          "main", (Object[])args);

    } }
```

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

import org.python.core.*;

public class EmbeddingPureJython {

    public static void main(String[] args) {

        PyString[] argv = new
          PyString[args.length];

        for(int i = 0; i < args.length; i++)
          argv[i] = new PyString(args[i]);

        PyDictionary globals = new
          PyDictionary();

        globals.__setitem__(
          new PyString("argv"),
          new PyTuple(argv));

        PythonInterpreter py = new
          PythonInterpreter(globals);

        py.exec("from script import main");

        py.exec("main(*argv)");

} }
```

Ett litet exempelskript

```
from javax import swing

from java import awt; event = awt.event

class ActionListener(event.ActionListener):
    def __init__(self, action):
        self.actionPerformed = action

def click(event):
    print "you clicked", event.source.text

click = ActionListener(click)

def makeButtons(captions):
    for text in captions:
        button = swing.JButton(text)
        button.addActionListener(click)
        yield button

def main(title="Hello JFokus",*buttons):
    frame = swing.JFrame(title)
    for button in makeButtons(buttons):
        frame.add(button)
    frame.visible = True
```

```
@ActionListener
def click(event):
    print “you clicked”, event.source.text
```

Python (2.5) vs. Jython (2.2)

- Generatorer – iterator genom återinträde
- Import-introspektion
  - .py-filer i en jar
  - import av python bytekod från .pyc-filer
- Funktionsdekoratorer
- Generatorer i kort-form “comprehension”
- Återanrop & undantagshantering i generatorer
- Kontexthantering

Jython 2.5 exempel

```
from neo4j import EmbeddedNeo,
    Transaction, neo4j, transactional

def main():

    neo = EmbeddedNeo("var/neo", True)

    with Transaction():

        root = neo.createNode()

        root["name"] = "Thomas
            Andersson"

        root["age"] = 29

        friend = neo.createNode()

        friend["name"] = "Trinity"

        root.KNOWS( friend )

        # create more data ...

    print_friends( neo, root )
```

```
@transactional

def print_friends( neo, node ):

    traverser = node.BreadthFirst[
        neo.KNOWS.Outgoing ](
        neo4j.END_OF_NETWORK,
        neo4j.ALL_BUT_START_NODE )

    for friend in traverser:

        print_person(friend, traverser)

def print_person( node, traverser ):

    print "At depth %s => %s" % (
        traverser.depth,
        person["name"] )
```

Summer of Code

- Fullt stöd för Python 2.5
- Ingen parser, istället transformation av bytekod
- Ladda Python bytekod

Jython 2.5

- Under utveckling
- Pre-release kommer på PyCon i början av mars
- Fullt stöd för alla språkets möjligheter
- Portering av C-bibliotek kommer senare
- Kompatibelt med Java 5

Efter 2.5 ...

- Bättre prestanda
  - Lägre anropsoverhead
  - Inget GIL, Javas möjligheter till parallelllitet
  - Classworking
- Ännu bättre integration
  - Generera POJO-klasser, möjliga att ladda från Java
  - Mer standardmässig undantagshantering

Jython

- Intressant idag – ännu intressantare imorgon
- Integrerar väl med Java-miljön
- Inte långsamt – kommer bli ännu bättre
- Större potential än CPython

- det vassaste verktyget i Java-lådan

SVAR!