Jython. Меньше кода, больше возможностей

creative commons

Copyright © 2008 Андрей Власовских
Лицензировано по Creative Commons Attribution-Noncommercial-Share Alike 3.0 License

Обо мне

- Андрей Власовских vlasovskikh@aivt.ftk.spbstu.ru
- ТКЦ ФТК, ведущий программист
- NC, SE, PL
- Мой блог: http://vlasovskikh.wordpress.com/

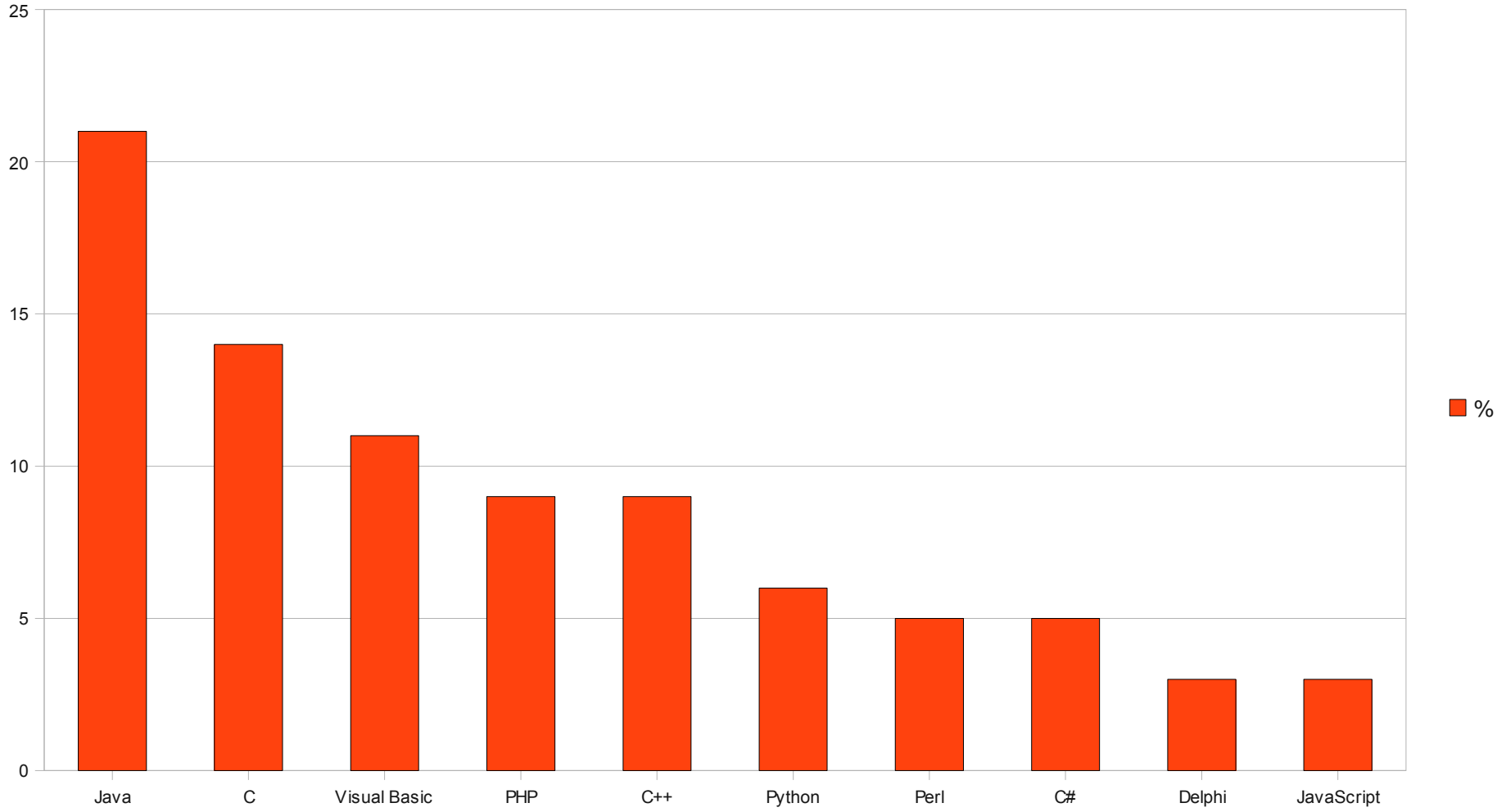
Структура

- Java + Python? (15 мин)
- Язык Python (15 мин)
- Как Jython меняет Java (15 мин)

Вначале вопросы —
к вам!

Ваш основной язык?

TPCI 2008-01


25
20
15
10
5
0
Java
C
Visual Basic
PHP
C++
Python
Perl
C#
Delphi
JavaScript
%

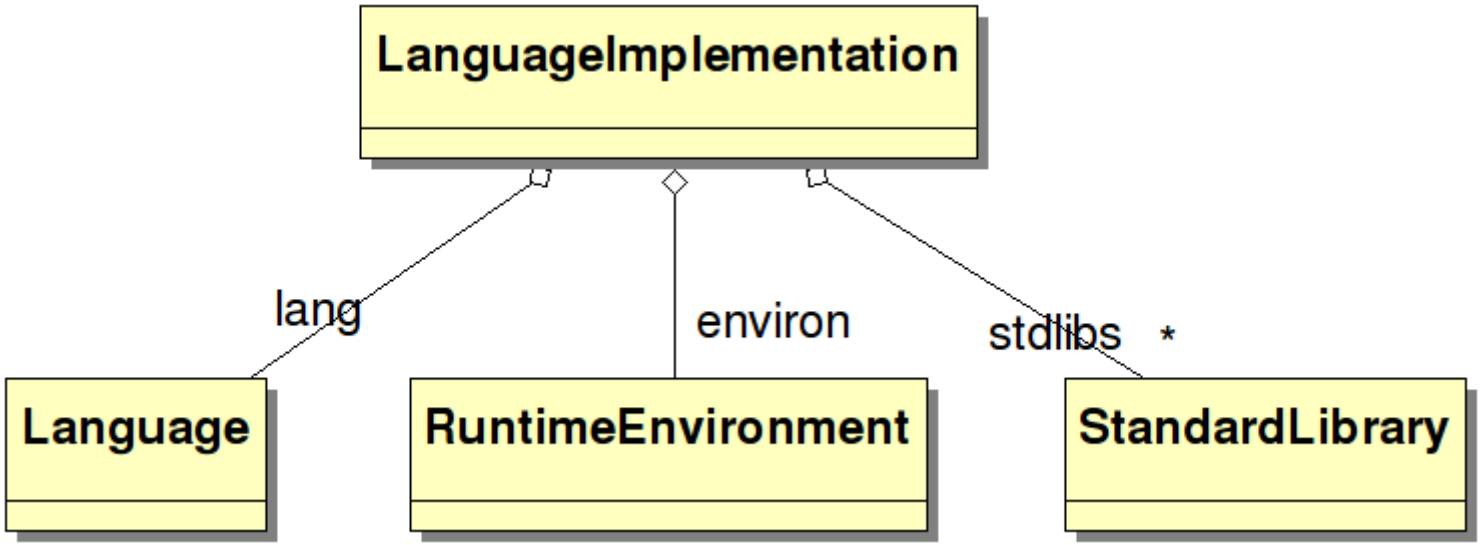
Насколько вы знаете Java?

Насколько вы знаете Python?

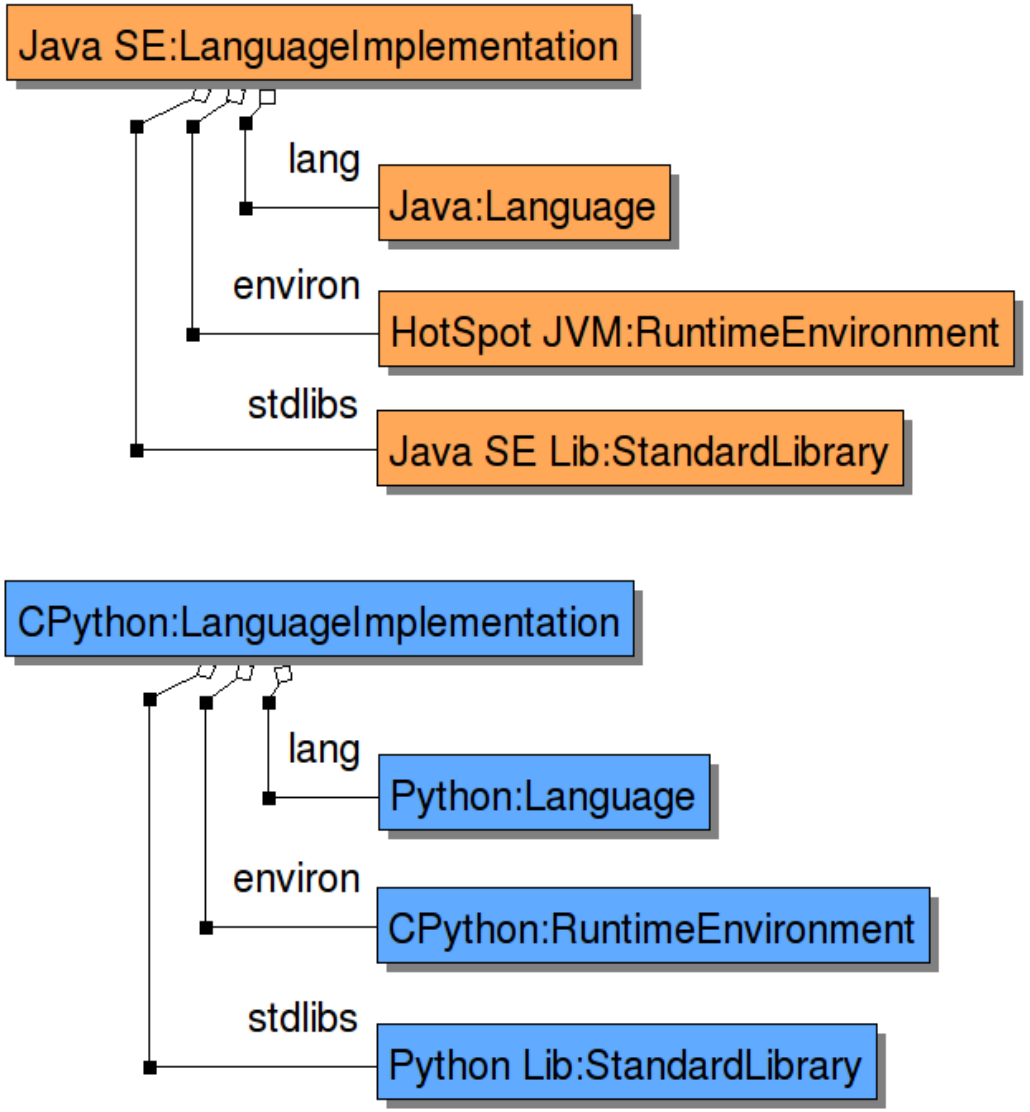
Часть 1. Java + Python?

Реализация языка

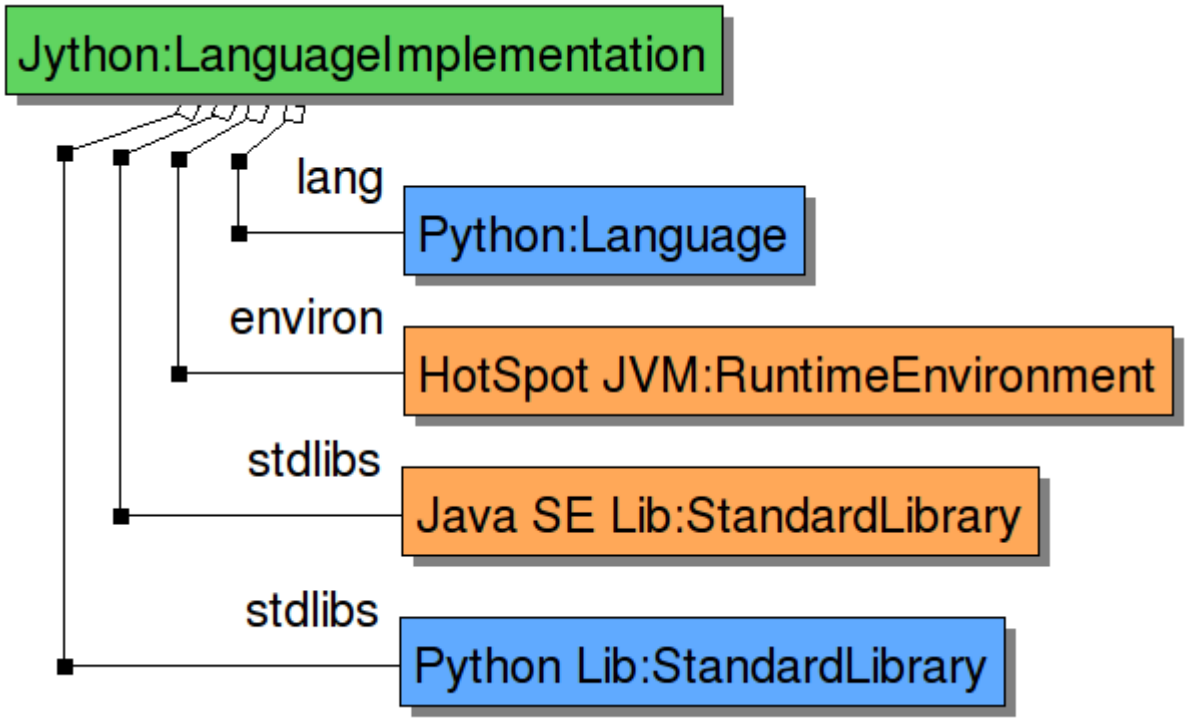
Реализация языка



Java SE и CPython



Реализация Jython



Jython — программа на Java

```
$ java -cp "/path/to/jython.jar:$CLASSPATH"
  -Dpython.home="/jython/home"
  org.python.util.jython
  /path/to/program.py args
```

...или проще:

```
$ jython /path/to/program.py args
```

...или ещё проще:

```
$ ./program.py
```

В начале файла: `#!/bin/env jython`

Интерактивный Jython

```
$ jython
>>> s = "Hello World"
>>> s
'Hello World'
```

Jython — класс на Java

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

public class TestInterperter {
    public static void main(String[] args)
    throws PyException {
        PythonInterpreter p =
            new PythonInterpreter();
        p.set("a", new Integer(2));
        p.exec("s = a + a");
        System.out.println(p.get("s"));
    }
}
```

Jython поддерживает JSR 223

Scripting for the Java Language

Почему с Java/Python на Jython?

Java → Jython

Нужно быстрое прототипирование

Нужно интерактивное изучение API

Нужна динамическая расширяемость

Нужен удобный язык

Python → Jython

Сложности ‖ на SMP

Нужны библиотеки на Java

?

Часть 2. Язык Python

Парадигмы: OOP, SP, элементы FP, AOP

Всё — это объект

Экземпляры, классы, методы, функции, модули,
а также
классы классов, классы классов классов...

Объект: идентичность, тип, значение

`id(o)`
`type(o)`, `isinstance(o, t)`
`o`, `dir(o)`

Типы объектов

- None
- Числа: int, long, bool, float, complex, ...
- Контейнеры
  - Последовательности: str, unicode, tuple, list, ...
  - Отображения: dict, ...
- Вызываемые
  - Функции и методы (isinstance(f, FunctionType))
  - Классы (isinstance(c, type))
  - Экземпляры классов (isinstance(o, c))
- Модули (isinstance(m, ModuleType))

Базовые контейнеры: [], {}, ()

`list`, `dict`, `tuple`

`[1, 2, 3]`, `{1: 2, 2: 4}`, `(1, 2, 3)`

Demo

Интерактивное изучение API

Типизация

Строгая динамическая типизация («утиная»)

Python:
строгая динамическая

```
bird = make_some_bird()
if hasattr(bird, "quack"):
    bird.quack(2)
    # Run-time error
    bird.quack(2 + "3")
```

Java:
строгая статическая

```
// Compile-time error
Duck duck = BirdGenerator.makeSomeBird();
```

...поэтому:

```
// Possible run-time error
Duck duck = (Duck) BirdGenerator.makeSomeBird();
duck.quack(5);
```

C:
слабая статическая

```
int size = sizeof(duck_t) * 4;
duck_t *ducks = malloc(size);
duck_t *duck = ducks + size;
duck_quack(duck, 1);
```

Здесь баг?

Связывание и разрешение имён

Динамическое разрешение имён,
позднее связывание атрибутов объектов,
статические области видимости

Семантика интерпретации

```
try:
    from cStringIO import StringIO
except ImportError:
    from StringIO import StringIO
>>> StringIO
<java function StringIO 1>
```

Функции — полноправные объекты

Замыкания, вложенные функции,
функции высших порядков

Замыкания в Python

```
def power(n):
    def f(x):
        return x ** n
    return f
squared = power(2)
cubed = power(3)
>>> squared(10), cubed(10)
(100, 1000)
```

λ-исчисление

1930-е, Алонсо Чёрч

FP

Нет изменяемых данных,
функции вместо состояния,
противопоставляется императивным языкам

Функциональные языки

Lisp, Scheme, ML,
Haskell,
Erlang, Python, Ruby, Scala

«Скриптовые» языки: Python, Ruby, JavaScript, ...

Динамически типизируемые,
объектно-ориентированные,
частично функциональные

λ

```
def power(n):
    def f(x):
        return x ** n
    return f

power = lambda n: lambda x: x ** n
```

?

Часть 3. Как Jython меняет Java

Все «объекты» Java становятся объектами

```
from java.lang import System
import java
>>> type(java), type(System), type(System.out)
(<type 'javapackage'>, <type 'javaclass'>,
<type 'javainstance'>)
```

Приводятся типы

Примитивные типы, строки, массивы, классы

JavaBeans на уровне языка

JavaBeans: свойства, ключевые аргументы

```
JButton b = new JButton();
b.setText("Press");
```

```
b = JButton()
b.text = "Press"
b2 = JButton(text="Press", visible=True)
```

JavaBeans: конструкторы свойств

```
JFrame f = new JFrame();
f.setSize(new Dimension(640, 480));
```

```
f = JFrame()
f.size = (640, 480)
```

JavaBeans:
события и слушатели

```
b = new JButton();
b.setText("Press");
b.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        this.text = ((JButton) e.getSource()).getText();
    }
});
```

...а на Jython:

```
def save_text(e):
    self.text = e.source.text

b = JButton(text="Press",
    actionPerformed=save_text)
```

Demo

File Tree

Каркас интерфейса на Swing

Общий вид

```
import …

def exit(e): System.exit(0)

class MainFrame(JFrame):
    def __init__(self, **kwargs): …

if __name__ == "__main__":
    frame = MainFrame(
        size=(480, 480),
        windowClosing=exit,
    )
    frame.show()
```

Java Swing: типичный конструктор формы

```
JPanel p = new JPanel();
p.setLayout(new BorderLayout());
p.setBorder(new EmptyBorder(5, 5, 5, 5));
JPanel east = new JPanel();
east.setLayout(new GridLayout(8, 1, 0, 5));
JButton b = new JButton("Beep");
b.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.out.println(((JButton) e.getSource()).getText());
    }
});
east.add(b);
east.add(new JButton("1"));
east.add(new JButton("2"));
p.add(east, BorderLayout.EAST);
JButton b2 = new JButton("Exit");
b.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }
});
p.add(b2, BorderLayout.SOUTH);
this.getContentPane().add(p);
```

GridPanel, BorderPanel

```
GridPanel(
    rows, columns, hgap=0, vgap=0,
    …,
    cells=[
        …
    ],
)

BorderPanel(
    …,
    bottom=…,
    center=…,
    left=…,
)
```

Наш MainFrame.__init__()

```
def __init__(self, **kwargs):
    JFrame.__init__(self, **kwargs)
    def print_text(e):
        print e.source.text
    self.contentPane = BorderPanel(
        border=EmptyBorder(5, 5, 5, 5),
        right=GridPanel(
            rows=8, columns=1, vgap=5,
            cells=[
                JButton(
                    text="Beep",
                    actionPerformed=print_text,
                ),
                JButton("1"),
                JButton("2"),
            ],
        ),
        bottom=JButton(
            text="Exit",
            actionPerformed=exit,
        ),
    )
```

Как будет выглядеть?

Что рассмотрели?

- Язык и реализация: Java SE, CPython, Jython
- Java/Python → Jython
- Основные свойства языка Python
  - Примитивы, средства комбинирования
  - Типизация, именование
  - Функции как полноправные объекты
- Jython меняет Java
  - Поддержка JavaBeans
  - Написание GUI на Swing

Что осталось?

- 禅 Python'a (`import this`)
- Протоколы вместо интерфейсов (file-like, iterator, ...)
- FP (чистота, выделение списков, генераторы, модели вычислений)
- Специальные методы, эмуляция типов
- Связывание атрибутов, __dict__, descriptor protocol
- Экземпляры и классы, метаклассы
- и ...

...нужна практика!

http://vlasovskikh.wordpress.com/