Apresentaçao da linguagem Groovy

Eduardo Motta Ramos
Caio Saracuza Luz
Beatriz

23/03/2018

Universidade Federal Fluminense

Linguagem escolhida

História

Início do desenvolvimento em 2003, mas a versão 1.0 foi lançada oficialmente em 2007.
Similar a Java, porém mais compacta por não precisar de todos os elementos que Java precisa.

"Minha idéia inicial é fazer uma linguagem Dinâmica, que seja compilada diretamente em classes Java e que tenha toda a produtividade e elegância encontrada em Ruby e Python"

Linguagem escolhida

Instalação:

- Versão atual: 3.0 (2.4)
- http://groovy-lang.org/download.html
- Para uma instalação rápida, os usuários de Mac OSX e Linux podem usar a ferramenta SDKMAN!
- $ curl -s get.sdkman.io — bash
- $ source "$HOME/.sdkman/bin/sdkman-init.sh"
- $ sdk install groovy

Linguagem escolhida

Groovy:

- Paradigmas: Orientada a objeto, imperativa, funcional, script
- Interpretada em tempo de execução
- Também é possivel compilar o código para conseguir melhorar ainda mais a performance. (comparável com Java)
- Possui inferência de tipos (def)
- Sintaxe nativa para listas, vetores e expressões regulares.
- Sobrecarga de operadores

Linguagem escolhida

Groovy:

- Safe Navigation Operator
- Possui alta legibilidade, redigibilidade e confiabilidade
- Groovy traz suporte nativo para várias linguagens de marcação, como XML e HTML.
- Compilação de Groovy em algum ponto é representada em uma AST(Abstract Syntax Tree), com propósito de deixar os desenvolvedores modificarem a AST antes de se transformar no bytecode que a JVM utilizará

Exemplos de uso da linguagem

Groovy traz uma forma simples e consistente de lidar com Listas e HashMap.

```
// Looks like an array, but is a list
def movieList = ['Dersu Uzala', 'Ran', 'Seven Samurai']
assert movieList[2] == 'Seven Samurai'
movieList[3] = 'Casablanca' // Adds an element to the list
assert movieList.size() == 4

// Declares a map
def monthMap = [ 'January' : 31, 'February' : 28, 'March' : 31 ]
assert monthMap['March'] == 31 // Accesses an entry
monthMap['April'] = 30 // Adds an entry to the map
assert monthMap.size() == 4
```

Exemplos de uso da linguagem

Definindo Classes:

```
class AGroovyBean {
        String color
}

def myGroovyBean = new AGroovyBean()

myGroovyBean.setColor('baby blue')
assert myGroovyBean.getColor() == 'baby blue'

myGroovyBean.color = 'pewter'
assert myGroovyBean.color == 'pewter'
```

Exemplos de uso da linguagem

Usando GStrings:

```
BigDecimal account = 10.0
def text = "The account shows currently a balance of $account"
assert text == "The account shows currently a balance of 10.0"

BigDecimal minus = 4.0
text = "The account shows currently a balance of ${account - minus}"
assert text == "The account shows currently a balance of 6.0"
```

Closures

Closures são pedaços de código que podem conter declarações. Semelhantes a ponteiros de funções em C. Podendo ser passados como argumento em outras funções.

```
// This block of code contains expressions without reference to an implementation
def operations = {
        declare 5
        sum 4
        divide 3
        print
}
```

Suporte à construção de compiladores

```
def closure = { ' HELLO WORLD! '}
assert closure() == 'HELLO WORLD!'

def sum = { a, b -> a + b }
assert sum(2,3) == 5
```

Recursive

```
def factorial ={ n ->
        if (n == 0) 1
        else n * factorial(n - 1)
}
```

Recursive

factorial(6)
6 * factorial(5)
6 * (5 * factorial(4))
6 * (5 * (4 * factorial(3)))
6 * (5 * (4 * (3 * factorial)))
6 * (5 * (4 * (3 * (2 * factorial(1)))))
6 * (5 * (4 * (3 * (2 * 1))))
6 * (5 * (4 * (3 * 2)))
6 * (5 * (4 * 6))
6 * (5 * 24)
6 * 120
720

Curry

```
def percentage = { percentage, x ->
        x/100 * percentage
}

def tenPercent = percentage.curry(10)
assert percentage(10,100) == 10
assert tenPercent(100) == 10
```

Domain-Specific Languages

```
// equivalent to: turn(left).then(right)
turn left then right

// equivalent to: take(2.pills).of(chloroquinine).after(6.hours)
take 2.pills of chloroquinine after 6.hours

// equivalent to: paint(wall).with(red, green).and(yellow)
paint wall with red, green and yellow
```

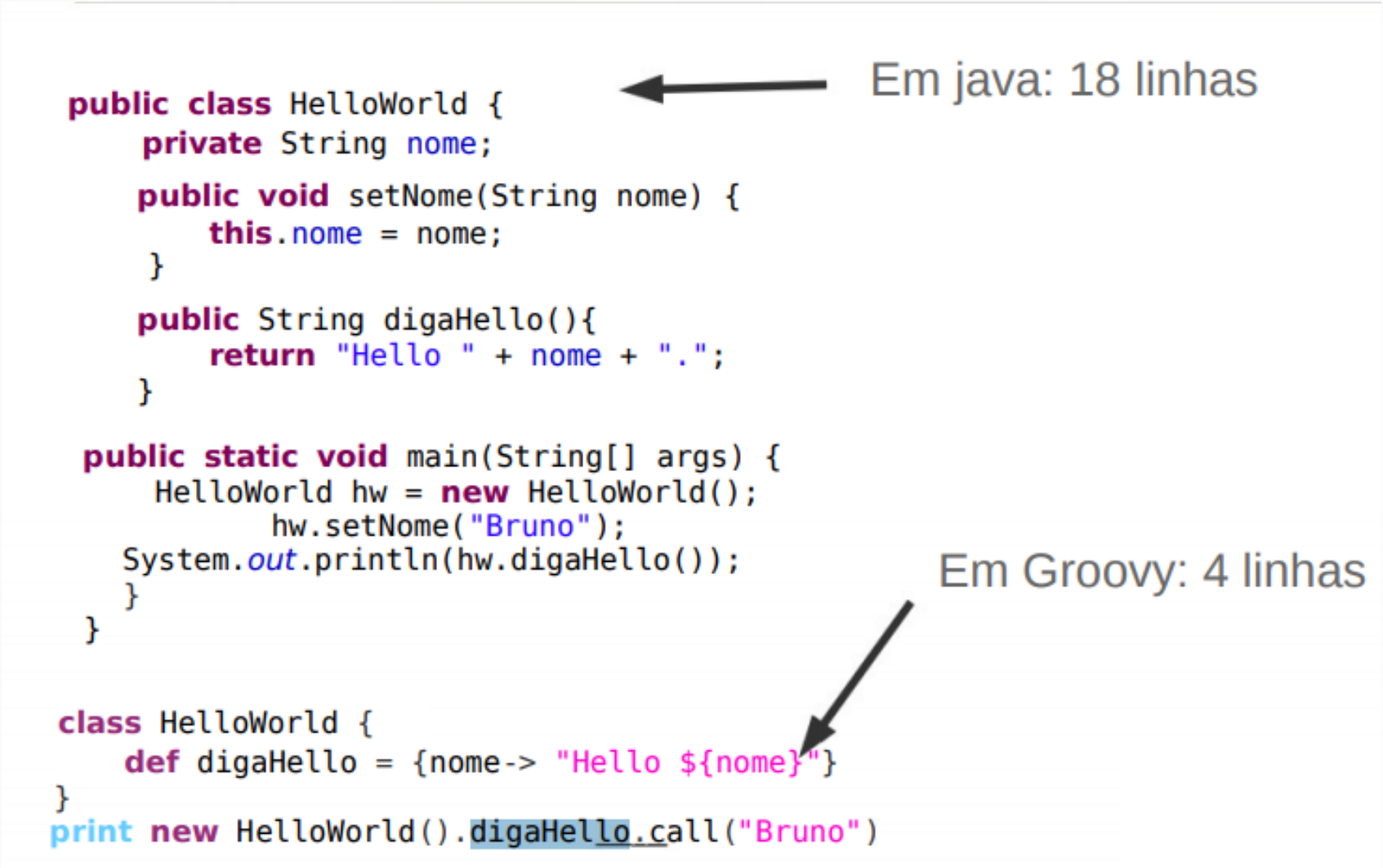
Creating a DSL

```
show = { println it }
square_root = { Math.sqrt(it) }

def please(action) {
        [the: { what ->
        [of: { n ->action(what(n)) }]
}] }

// equivalent to: please(show).the(square_root).of(100)
please show the square_root of 100
// ==>10.0
```

Domain-Specific Languages

Operador	Método
`a + b`	`a.plus(b)`
`a - b`	`a.minus(b)`
`a * b`	`a.multiply(b)`
`a ** b`	`a.power(b)`
`a / b`	`a.div(b)`
`a % b`	`a.mod(b)`
`a | b`	`a.or(b)`
`a & b`	`a.and(b)`
`a ^ b`	`a.xor(b)`
`a++ or ++a`	`a.next()`
`a-- or --a`	`a.previous()`
`a[b]`	`a.getAt(b)`
`a[b] = c`	`a.putAt(b, c)`
`a << b`	`a.leftShift(b)`

Prós e contras



Prós e contras

Vantagens

- Portátil
- Orientada a Objeto
- Similar a Java
- Código enxuto

Prós e contras

Desvantagens

- Performance não equivalente a C
- Linguagem nova
- Documentação escassa (comparado a Java)

Prós e contras

Por quê não é popular

- No início a performance era muito ruim

  Mudou a partir de Java7 com invokeDynamic
- Pessoas que vem de Java muitas vezes acham difícil de entender por ser funcional