Groovy

Trabalho 01 - Estruturas de Linguagem 2016.2

Igor Lessa Morse Alves

Origem

“I'd rather a dynamic language that builds right on top of all the groovy Java code out there & the JVM.“

“We're starting simple with the nice tuples, sequences, maps from python & closures from ruby and being concise & dynamically typed with a java-look-and-feel though where we end up is anyone's guess right now.”

James Strachan, 2003

Influências

Várias linguagens influenciaram o Groovy, dentre estas, **Java** , **Python** e **Ruby** estão entre as principais.

Outras linguagens que influenciaram o Groovy:

- Perl
- Smalltalk
- Objective-C
- PHP

Classificação

- Tipagem Dinâmica e Estática.
- Funcional.
- Imperativa.
- Orientada a Objetos.
- Linguagem de Script.

```java
public class AnimalSounds {

    public static void main(String[] args){

        HashMap<String,String> animals = new HashMap<>();

        animals.put("Cachorro","AuAu")
        animals.put("Gato","Miau")

        animals.each{ animal, animalSound->

            System.out.println("Animal: " + animal + "\tSom: " + animalSound)

        }
    }
}
```

Groovy web console

```groovy

// Tipagem dinâmica
i = 10001
i = false
i = "Hello World"
// Todas as atribuições são válidas.


// Tipagem Estática
int j = 10

j = 300

// Atribuição inválida, j é do tipo int e não pode ser Atribuido uma String.
j = "oi"
```

Actions ➤ | Execute script | New script | Publish script | View recent scripts

Result | Output | Stacktrace

```
org.codehaus.groovy.runtime.typehandling.GroovyCastException: Cannot cast object 'oi' with class 'java.lang.String' to class 'int'
        at Script1.run(Script1.groovy:14)
```

Expressividade em Relação ao Java

- A maioria dos comandos em Java podem ser utilizados no Groovy.
- O Groovy possui Closures, tipagem dinâmica e estática, interpolação de strings, verificações em tempo de execução, sobrecarga de métodos verificada em tempo de execução, operadores de métodos com ponteiros e Metaprogramação (MOP) em tempo de execução e isto **não é possível em Java.**

```
/*

Outro uso da Metaprogramação, adicionando um Método na Biblioteca existente de String e um Atributo.

Saída:

igor estuda na uerj e tem 23

*/

// Inserção de um Método na Classe String Padrão.
String.metaClass.faculdade = { -> "$delegate estuda na uerj" }

// Inserção de um Atributo na Classe String Padrão.
String.metaClass.idade = 23

println ('igor'.faculdade() + ' e tem ' + 'igor'.idade)
```

```
String.metaClass.modify = { delegate = it }

// Saída: Mas vai sair outra
println 'Tenho uma String'.modify('Mas vai sair outra')
```

Podemos com certeza afirmar que o **Groovy** é **mais expressivo** que o **Java**!

Exemplos Comparativos Java e Groovy

```java
import java.util.*;
import java.lang.*;
import java.io.*;

/*

Calcula 10 termos da série de Fibonacci.

*/
class Ideone {

        public static void main (String[] args) throws java.lang.Exception {

                for (int i = 0 ; i <= 10 ; i++)
            System.out.println(String.valueOf(itFibN(i)));
        }

        public static long itFibN(int n){

                 if (n < 2)
                         return n;
                 long ans = 0;
                 long n1 = 0;
                 long n2 = 1;
                 for(n--; n > 0; n--){
                  ans = n1 + n2;
                  n1 = n2;
                  n2 = ans;
                 }

                 return ans;
        }
}
```

```groovy
def iFib = {
    it == 0   ? println(0)
    : it == 1 ? println(1)
    : it > 1  ? println((2..it).inject([0,1]){i, j -> [i[1], i[0]+i[1]]}[1])
    : println("Negativo nao pode"); return
}

0.upto(10,iFib)
```

Observando os exemplos acima, temos que, sem dúvidas o Groovy possui maior **facilidade para Escrita**. Quanto a legibilidade, ambos são **igualmente legíveis.**

Exemplos Comparativos Java e Groovy

```java
import java.util.regex.Matcher;
import java.util.regex.Pattern;
/**

	Classe que filtra Lista Wifi por uma Regra de Formação Específica.

*/
class regEx {

	public static void main(String[] args){

		String[] wifiList = { "Subway", "FulaninhoTal", "WifiEstranho", "UERJ-6andar", "UERJ-7andar" };

		Pattern pattern = Pattern.compile("(UERJ-)[1-9](andar)");

		for (String wifi : wifiList){

			Matcher matcher = pattern.matcher(wifi);

			if (matcher.matches())
				System.out.println(wifi);

		}

	}
}
```

```groovy
wifiList = [ 'Subway', 'FulaninhoTal', 'WifiEstranho', 'UERJ-6andar', 'UERJ-7andar' ]

wifiList.each { wifi->

    if ( wifi ==~ /(UERJ-)[1-9](andar)/ )
            println(wifi)
}
```

\+ Writability

Conclusão

Devido sua facilidade de integração, o Groovy é utilizado em vários sistemas no mercado atualmente, principalmente em conjunto com sistemas em Java. A Netflix e o Linkedin utiliza o Groovy para realizar pequenas tarefas de Script em seus sistemas.

Sua facilidade de uso e seus diversos recursos para facilitar a vida do programador deixam tentador o seu uso, porém o Groovy perde um pouco no quesito performance e este pode ser considerado seu ponto negativo.

Fim !