Estudo sobre a Linguagem de Programação Scala

Jaqueline Miranda Neves e Milena Rosa Mendes

Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)  
Boa Vista – RR – Brazil

jaquelinemirandan@gmail.com, milena.mends.25@gmail.com

*O trabalho aqui exposto é uma síntese da linguagem de programação Scala. Serão apresentados os dados históricos; as aplicações da linguagem; as variáveis que ela suporta, bem como seu tamanho na memória, e a sua visibilidade. A linguagem suporta o paradigma funcional e a orientação a objetos. Além disso, serão apresentados os comandos de controle da linguagem. Por fim, um exemplo prático da linguagem Scala com uso da biblioteca twitter4j.*

# 1. História do surgimento da linguagem

A linguagem Scala foi criada por Martin Odersky, um pesquisador da universidade EPFL, na Suíça. Em 1995, ele juntou-se com Philip Wadler para escrever uma linguagem de programação funcional que compila Java bytecode. Em 1999, depois de se unir a EPFL, seu objetivo era combinar programação orientada a objetos e programação funcional, porém sem as restrições que eram apresentadas no Java.

O primeiro passo era o Funnel, na qual é uma linguagem minimalista de busca baseada em redes funcionais, uma variante orientada a objetos de join calculus. Porém, em Funnel quase tudo, incluindo classes e padrões correspondentes, seria feito por bibliotecas e codificações.

O segundo passo, e atual, é o Scala, que trouxe algumas das ideias do Funnel e colocou dentro de uma linguagem mais pragmática. Ela integra os recursos de linguagens orientadas a objetos e funcional, permitindo que programadores Java e de outras linguagens sejam mais produtivos e práticos no que concerne o quesito desenvolvimento de software.O design do Scala começou em 2001 e o primeiro lançamento ao público foi em 2003.

# 2. Domínios de aplicação

HelloWorld em Scala

package main

object Main {

def main(args:Array[String]) {

print("Hello World!")

}

}

Declaração de variáveis em Scala

package main

object Main {

def main(args:Array[String]) {

var num1 = 5

var num2 = 10

var texto = "A soma é: "

println(texto + (num1 + num2))

var s = new StringBuffer

s.append("Olá!");

s.append("Vamos aprender Scala?")

println(s);

}

}

Funções Anônimas

package main

object Main2 {

def fazConta(callback: (Int, Int) => Int, x: Int, y: Int) {

println(callback(x, y))

}

def contaUmDez(f: (Int) => Unit) {

for (i <- 1 to 10) f(i)

}

def main(args: Array[String]) {

println("Realiza as operações matemáticas")

fazConta((x, y) => x + y, 5, 6)

fazConta((x, y) => x \* y, 5, 3);

fazConta((x, y) => x / y, 5, 3);

fazConta((x, y) => x - y, 5, 3);

println("Conta até 10")

contaUmDez(i => println(i))

}

}

# 3. Paradigmas suportados pela linguagem

# A linguagem Scala foi desenvolvida para expressar padrões de programação comuns de uma forma concisa, elegante e de tipagem segura. Ela suporta dois paradigmas, de linguagens orientadas a objetos e funcional.

# Orientada a Objeto

package main

object Main6{

def main(args:Array[String]) {

val num1 = 5

val num2 = 10

val texto = “A soma é: “

println(texto + (num1 + num2))

}

}

# Funcional

Scala> def criarMensagemDeErro (codigoErro:Int) : String = codigoErro match {

| case 1 => “Erro ao conectar”

| case 2 => “Erro ao abrir arquivo”

| case \_ => “Erro desconhecido”

| }

criarMensagemDeErro: (codigoErro: Int)String

scala> |

# 4. Variáveis e tipos de dados

Scala tem os mesmo tipos de dados que o Java, com a mesma precisão e tamanho de memória. Todos os tipos listados abaixo são objetos. Não existe tipo primitivo em Scala como no Java. Isso significa que em Scala pode-se chamar metodos em um int, por exemplo.

* Byte: possui 8 bits. Varia entre -128 e 127.
* Short: possui 16 bits. Varia entre -32768 e 32767.
* Int: possui 32 bits. Varia entre -2147483648 e 2147483647.
* Long: possui 64 bits. Varia entre -9223372036854775808 e 9223372036854775807.
* Float: Possui 32 bits. Varia entre ±1.18×10−38 e ±3.4×1038.
* Double: Possui 64 bits. Varia entre   
  ±2.23×10−308e ±1.80×10308.
* Char: possui 16 bits. Varia entre U+0000 e U+FFFF.
* String: sequência de chars.
* Boolean: varia entre true e false.
* Unit: corresponde a nenhum valor.
* Null: vazio.
* Nothing: subtipo de qualquer outro tipo, inclui nenhum valor.
* Any: o supertipo de qualquer outro tipo, qualquer objeto é um tipo Any.
* AnyRef: supertipo de qualquer tipo de referência.

# 5. Comandos de controle

O comando if consiste em uma expressão booleana seguida de declarações. Se a expressão for verdadeira, então o bloco de código será executado, senão o programa não executa o bloco do if. Um exemplo do comando if:

object Teste{

def main(args: Array[String]){

var x=5;

if(x<10){

println(“5 é menor que 10”);

}

}

}

O comando else é utilizado quando o comando if utilizado é falso. Por exemplo:

object Teste{

def main(args: Array[String]){

var x=5;

if(x>10){

println(“X é maior que 10”);

} else{

println(“X é menor que 10”);

}

}

}

Outro comando usando if e else é o else if, que verifica quais das condições são verdadeiras e a as executa.

object Teste{

def main(args: Array[String]){

var x=20;

if(x==5){

println(“X é 5”);

} else if (x==10){

println(“X é 10”);

} else if (x==20){

println(“X é 20”);

}

}

}

O comando while repete uma declaração enquanto uma condição é verdadeira. Ela testa a condição antes de executar o bloco código.

object Teste{

def main(args: Array[String]){

var x=1;

while(x<5){

println(“O valor de x é: ” + x);

x = x +1;

}

}

}

O comando do-while repete uma declaração enquanto uma condição é verdadeira, ou seja, a mesma coisa que o comando while, porém ele testa a condição no depois de executar o bloco de código pelo menos uma vez.

object Teste{

def main(args: Array[String]){

var x=1;

do{

println(“O valor de x é: ” + x);

x = x +1;

}

while(x<5)

}

}

O comando for executa uma sequência de comandos várias vezes e abrevia o código que conduz quantas vezes esse código será repetido.

object Teste{

def main(args: Array[String]){

var x=0;

for (x<- 1 to 5){

println(“O valor de x é: ” + x);

}

}

}

A linguagem Scala não possui comandos de desvio.

# 6. Escopo (regras de visibilidade)

As variáveis em Scala podem estar em três diferentes tipos de escopo.

Campos são variáveis que pertencem a um objeto. Os campos são acessáveis de dentro de cada método do objeto. Podem ser acessados por fora dependendo do que o acesso modifica no campo declarado. Podem ser mutáveis ou imutáveis, os imutáveis podem ser definidos usando var ou val. Exemplo abaixo, as variáveis estão destacadas em vermelho.

class Janela {

var altura = 3.2f

var largura = 5.6f

def Tamanho() {

println("Altura é:"+altura)

}

println("Largura é :"+largura);

}

Parâmetros são variáveis usadas para passar um valor para um método, quando este é chamado. Só podem ser acessados de dentro, mas os objetos passados podem ser acessados por fora se houver uma referencia do objeto fora do método. São sempre imutáveis e definidos pela palavra val. Exemplo das variáveis abaixo destacado em vermelho.

class Subtrair {

def sub(s1:Int,s2:Int){

var res = s1-s2

println("Resultado é:"+res);

}

}

object Teste {

def main(args:Array[String]){

val su = new Subtract()

su.sub(40,10)

}

}

Variáveis locais são declaradas dentro de um método. São acessadas somente dentro do método, elas só são usadas fora se forem o retorno do método. Podem ser mutáveis ou imutáveis, são declaradas usando var ou val. Abaixo o exemplo com as variáveis destacadas em vermelho.

class Adicao {

def adc() {

var(a,b) = (10,20);

var c = a+b;

println("Resultado é:"+c)

}

}

# 7. Exemplo prático de uso da linguagem de programação

### O exemplo prático escolhido para o trabalho foi um aplicativo que quando conectado a conta do Twitter de um usuário verifica as hashtags da timeline do usuário e as organiza de acordo com a quantidade que elas foram utilizadas. O ambiente de desenvolvimento foi o Scala IDE for Eclipse, que o proprio site do Scala indica.

O paradigma utilizado no aplicativo foi a orientação a objetos. Foi necessário importar uma biblioteca (Twitter4j) para acessar a conta de usuário do Twitter e suas ferramentas para fazer uma listagem das hashtags, essa biblioteca possui características da orientação a objetos.

# 8. Conclusão

Conclui-se que a Scala é uma linguagem de programação na qual tem como característica tipagem segura e uma forma elegante de expressar padrões comuns de programação. Essa linguagem suporta dois tipos de paradigmas: orientada a objeto e funcional. A maioria das pessoas que utilizam essa linguagem, já tem conhecimento prévio de Java, pois a linguagem Scala é derivada do Java e utiliza sua máquina virtual.

A Scala é uma linguagem pouco conhecida e consequentemente pouco uso. No entanto, há grandes expectativas de ser uma linguagem mais utilizada no futuro, uma vez que algumas das maiores empresas tecnológicas consideram-na uma substituta direta do Java. Os motivos são as vantagens que a linguagem apresenta, entre eles produtividade e simplicidade maiores que o Java.

# 9. Referências

Scala Variables, Variable Scopes, Field Variables, Method Parameters Example. Disponível em: <<http://www.journaldev.com/7581/scala-variables-variable-scopes-field-variables-method-parameters-example>>. Acesso em: 2 de agosto de 2016.

Scala IDE. Disponível em:< <http://scala-ide.org/>>. Acesso em: 1 de agosto de 2016.

Scala Tutorial. Disponível em:< <http://www.tutorialspoint.com/scala/index.htm>>. Acesso em: 2 de agosto de 2016.

Blog Caelum Disponível em : < <http://blog.caelum.com.br/scala-sua-proxima-linguagem/>> Acesso em: 5 de agosto de 2016