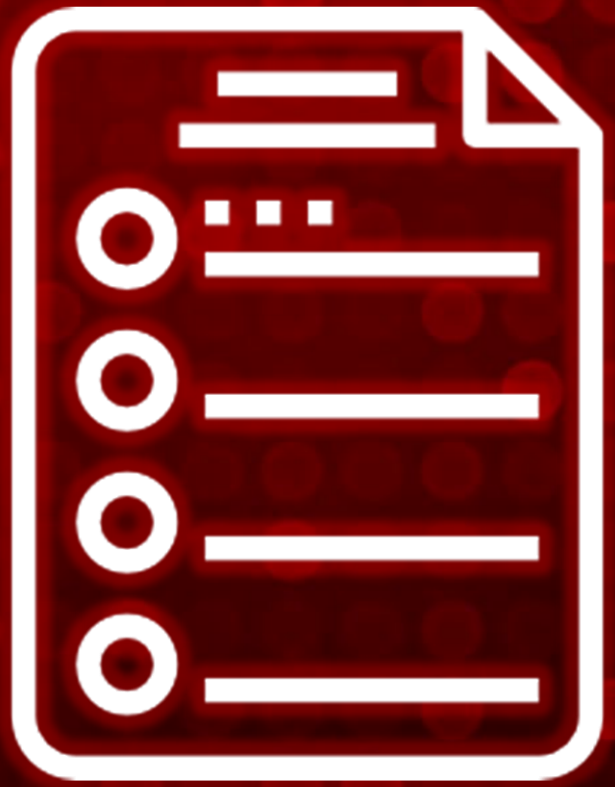

Scala

Cronograma Principal

- [x] Introdução ao Scala
- [x] Aplicabilidade e Mercado
- [x] Vantagens
- [x] Desvantagens
- [x] Sintaxe



Introdução ao Scala – Sua criação

- ☑ Criada em 2001 pelo ex integrante do Java Generics* Martin Odersky;
- ☑ Origem no Instituto Federal Suíço de Tecnologia de Lausanne;
- ☑ Publicada oficialmente na plataforma Java em 2004.

Introdução ao Scala – Características Principais

- Baseada no Java;
- Orientada a Objetos e Funcional;
- Compilada pela JVM – Java Virtual Machine;
- Permite métodos e classes Java.

Aplicabilidade e Mercado

Empresas do mercado que utilizam a linguagem: Twitter, Linkedin, FourSquare, Blizzard, Apache Spark (projeto open source)...

Aplicabilidade e Mercado

Scala é inevitavelmente uma forte candidata para BigData.

"Caso você esteja fazendo análise de dados pesados com cálculos estatísticos obscuros, então você seria louco de não favorecer o R. Caso esteja fazendo NLP ou processamentos intensos de rede neural entre GPUs, então o Python é uma boa aposta. E para uma solução de transmissão de produção rígida com todas as ferramentas operacionais importantes, Java ou Scala são definitivamente boas escolhas."

By Computer World – computerworld.com.br

Aplicabilidade e Mercado

Também boa escolha para análise de dados contendo bibliotecas para:

- ☑ Modelagem e programação probabilística;
- ☑ Análise preditiva;
- ☑ Machine Learning;
- ☑ Análise dimensional;
- ☑ Álgebra abstrata.

Vantagens

- Sintaxe simples e direta;
- Objetos imutáveis;
- Funcional e Puramente Orientado à Objetos;
- Rápida implementação;
- Fácil correção com problemas de concorrência;
- Suporte a XML.

Desvantagens

- Comunidade pouco ativa/colaborativa;
- Compilação lenta;
- Difícil aprendizado;
- Difícil adoção da Linguagem;
- Compatibilidade limitada.

Sintaxe

Primeiramente analisaremos as seguintes estruturas:

- ☑ Declarações e impressões em tela;
- ☑ Operadores e estruturas de condição;
- ☑ Laços de Repetição (For, While);
- ☑ Definição de Classes e Funções;

Declarações e impressões em tela

```
val num1 = 20
var num2 = 13
```

```
println(“Num1: ”+num1)
println(s“Num2: $num1”)
```

Val -> Valor Imutável
Var -> Valor Mutável

Obs.: Scala é *case sensitive*.

Operadores e estrutures de condição

```
//IF Structure
var x = 30;
if( x < 20 ){
     println("This is if statement")
} else {
     println("This is else statement")
}
```

Operadores principais:

- **<, >, ==, !=, <=, >=**
- **&&, ||, !**
- **+, -, *, /, %**

Obs.: Não é necessário “ ; ” mas a Scala reconhece.

Operadores e estrutures de condição

```
//CASE Structure
var i = 6
i match {
    case 1  => println("January")
    case 2  => println("February")
    case 3  => println("March")
    case 4  => println("April")
    case 5  => println("May")
    case whoa  => println(“Valor de entrada: " + whoa.toString)
}
```

Obs.: Não é a única forma de se fazer um Case.

Laços de repetição - FOR

```
//Para um List Array
   val languages = List("Java", "Scala", "F#")
   for (language <- languages)
       println(language)
//Até um número
   for (i <- 0 to 5)
       println(i)
//Até um número antes do último
   for (i <- 100 until 105) {
       println(i)
   }
```

Laços de repetição – FOR

```
Forma functional (to, until)
//Até um número
   for (i <- 0.to(3)) {
       println("for to", i)
   }
//Até um número antes do último
   for (i <- 0.until(3)) {
       println("for until", i)
   }
```

Laços de repetição – FOR

```
//Loop entre a faixa de valores especificada(de 5 até 7).
   for (i <- Range(5, 8)) {
       println("range1", i)
   }
//Loop com passo 2.
   for (i <- Range(0, 5, 2)) {
       println("range2", i)
   }
//Loop decrementado. //...Passo negativo unitario.
   for (i <- Range(10, 5, -1)) {
       println("range3", i)
   }
```

Laços de repetição – While

```
//While simples
   var i = 3
   while (i >= 0) {
        println(i)
        i -= 1  //Decrement
   }
//While indefinido
   while (true) {
       val rand = Math.random()
       //Irá sair do while se o número gerado estiver abaixo de 0.1
       if (rand <= 0.1)
              return
       println(rand)
   }
```

Definição de Classes e Funções

```
//Função com retorno
   def functionName([list of parameters]) : [return type] = {
      //function body
      return [expr]
   }
//Função sem retorno
   def printMe( ) : Unit = {
      println("Hello, Scala!")
   }
//Chamada de função
   functionName( list of parameters )
```

Definição de Classes e Funções

```
//Exemplo de criação de classe
  class Point(var x: Int, var y: Int) {
    def move(dx: Int, dy: Int): Unit = {
      x = x + dx
      y = y + dy
    }
    override def toString: String =
      s"($x, $y)"
  }
  val point1 = new Point(2, 3)
  point1.x  // 2
  println(point1)  // prints (2, 3)
```

Definição de Classes e Funções

```
//Exemplo de criação de classe
  class Point(var x: Int, var y: Int) {
    def move(dx: Int, dy: Int): Unit = {
      x = x + dx
      y = y + dy
    }
    override def toString: String =
      s"($x, $y)"
  }
  val point1 = new Point(2, 3)
  point1.x  // 2
  println(point1)  // prints (2, 3)
```

Links Úteis

- Compilador Online
  https://scalafiddle.io
- IDE (plugin eclipse)
  http://scala-ide.org
- Aplicações do Scala
  http://www.cienciaedados.com/data-science-com-scala-scalable-language/
- Aplicações do Scala
  https://computerworld.com.br/2016/06/03/r-python-scala-ou-java-qual-e-melhor-linguagem-para-big-data/
- Introdução ao Scala e exemplos
  https://www.devmedia.com.br/conheca-a-linguagem-scala/32850
- Códigos exemplos
  https://www.ime.usp.br/~gold/cursos/2015/MAC5742/reports/scala.pdf
- Operadores
  http://www.w3big.com/pt/scala/scala-operators.html
- Switch case
  https://alvinalexander.com/scala/how-to-use-scala-match-expression-like-switch-case-statement
- Estrutura de repetição FOR
  https://www.dotnetperls.com/for-scala

Links Úteis

- ☑ Explicações e exemplos mais aprofundados:
  https://github.com/LRAbbade/Workshop-Scala
  (Instruções no Readme)

Obrigado! E vamos á prática