



Alunas:
Flávia Avelino dos Santos
Priscila Maria Franca Da Silva

Um pouco da história

- ❖ Criada em 2003 a partir da publicação de um artigo de James Strachan publicou em seu blog
- ❖ Ideia inicial: era fazer uma linguagem dinâmica, que fosse compilada diretamente em classes Java, e que tivesse toda a produtividade e elegância encontrada em Ruby e Python”.
- ❖ Em janeiro de 2007 houve sua primeira versão 1.0. Em 2 de julho de 2012, o Groovy 2.0 foi lançado apresentando novos recursos tais como a adição de compilação estática e inferência de tipos
- ❖ Até hoje tem a versão 2.4 como o principal lançamento.

Principais características

- Linguagem Orientada a Objeto
- Suporte para tipagem estática e dinâmica
- Possui recursos inspirados em Linguagens como Ruby, SmallTalk e Python.
- Sintaxe similar ao do Java
- Interpretada ou compilada para bytecode.
- Fornece várias simplificações comparadas ao padrão da linguagem Java, além de recursos avançados como *closures* e suporte nativo a *listas* e *mapas*.
- MetaProgramação

O Groovy integrado ao Java

- Groovy é complementar ao Java
- Sua integração ocorre dentro da Virtual Machine
- Faz uso das bibliotecas Java



Groovy vs Java

- Os tipos primitivos do Java são convertidos para suas respectivas classes encapsuladoras de forma transparente para o programador.
- Simplicidade nas expressões

```
if (stringExemplo != null && !stringExemplo.isEmpty()) {...} //código Java
if (stringExemplo()) {...} //código Groovy
```
- Importações default. Os seguintes pacotes e classes são importados por default, sendo desnecessário usar *import* explícito para utilizá-los:
- Na linguagem Java você é obrigado a digitar *private* para declarar atributos privados. No Groovy todos os atributos de uma classe são por padrão *private*.
- Na linguagem Java você é obrigado a digitar *public* para declarar a classe pública. No Groovy não, pois todas as classes são por padrão pública
- Na linguagem Java você é obrigado a digitar os *get's* e *set's* para expor os atributos. No Groovy não, eles serão automaticamente e dinamicamente gerados para você.

Em Java

```
public class Greeter {  
    private String owner;  
    public String getOwner(){  
        return owner;  
    }  
    public void setOwner(String owner){  
        this.owner = owner;  
    }  
    public String greet (String name){  
        return "Ola" +name+ ", eu sou" +owner;  
    }  
}  
Greeter greeter = new Greeter ();  
greeter.setOwner("Pedro");  
System.out.println(greeter.greet("Maria"));
```

Em Groovy

```
class Greeter{  
    String owner  
    String greet (String name){  
        "Ola ${name}, eu sou ${owner}"  
    }  
}  
  
def greeter = new Greeter(owner: "Pedro")  
println greeter.greet("Maria")
```

```
groovy> class Greeter{  
groovy>     String owner  
groovy>     String greet (String name){  
groovy>         "Ola ${name}, eu sou ${owner}"  
groovy>     }  
groovy> }  
groovy> def greeter = new Greeter(owner:"Pedro")  
groovy> println greeter.greet("Maria")
```

Ola Maria, eu sou Pedro

Execution complete. Result was null.

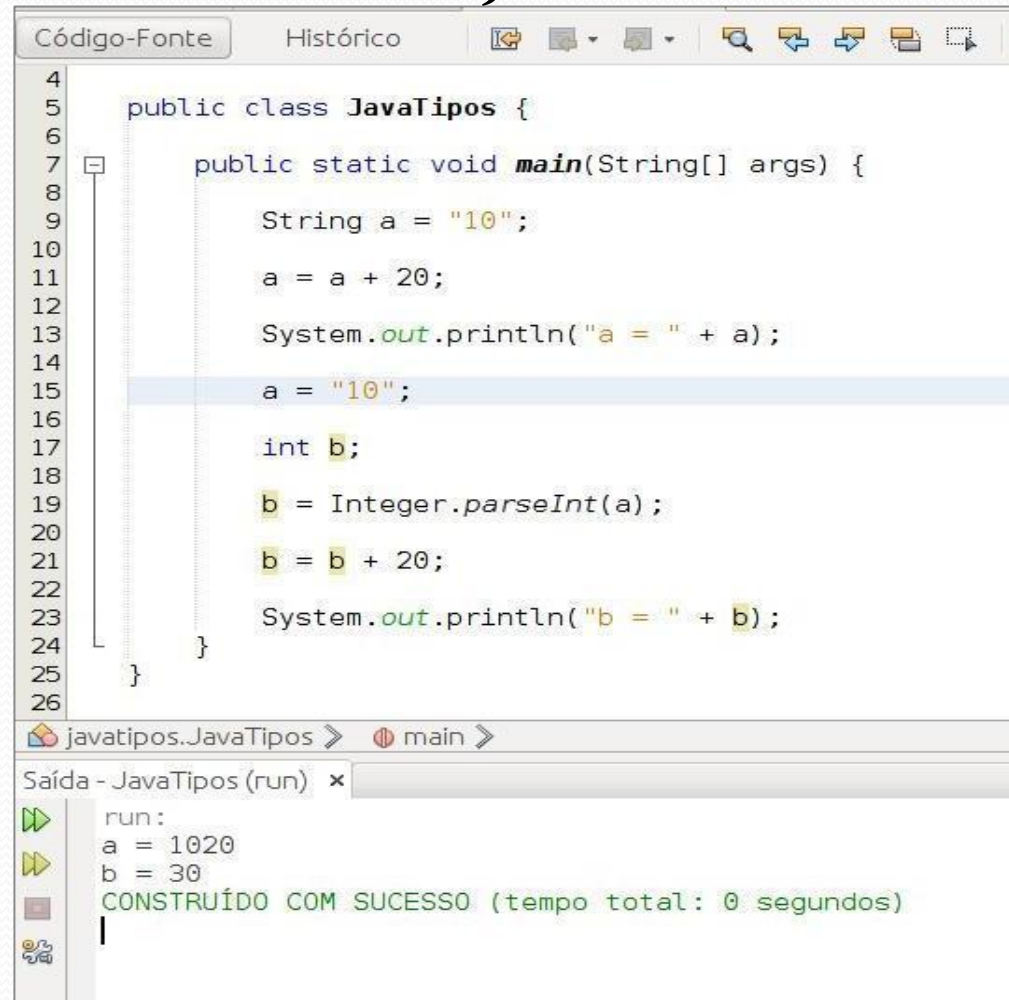
4:10

Conversão de Tipo

Groovy

```
linux@Computer: ~  
linux@Computer:~$ groovysh  
Groovy Shell (2.4.12, JVM: 1.8.0_131)  
Type ':help' or ':h' for help.  
-----  
groovy:000> a = "10"  
==> 10  
groovy:000> a + 20  
==> 1020  
groovy:000> a = 10  
==> 10  
groovy:000> a + 20  
==> 30  
groovy:000> 
```

Java



The screenshot shows an IDE window with a code editor and a console. The code editor displays a Java class named `JavaTipos` with a `main` method. The code demonstrates type conversion by assigning a string to a variable, adding an integer to it, and then parsing the result back to an integer. The console shows the output of the program, which is `a = 1020` and `b = 30`, followed by a success message.

```
Código-Fonte  Histórico  
4  
5 public class JavaTipos {  
6  
7     public static void main(String[] args) {  
8  
9         String a = "10";  
10  
11         a = a + 20;  
12  
13         System.out.println("a = " + a);  
14  
15         a = "10";  
16  
17         int b;  
18  
19         b = Integer.parseInt(a);  
20  
21         b = b + 20;  
22  
23         System.out.println("b = " + b);  
24     }  
25 }  
26  
javatipos.JavaTipos > main >  
Saída - JavaTipos (run) x  
run:  
a = 1020  
b = 30  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```


Reflexão

- Recurso da API Java que permite acesso e a modificação do comportamento de aplicações que estão rodando na Java Virtual Machine.
- Uso do pacote `java.lang.reflect`
- Permite exibir o nome de atributos e métodos pertencentes a uma classe, em tempo de execução.

Java

```
public class Pessoa {  
  
    public String nome;  
    public int idade;  
    private String telefone;  
    private String endereco;  
  
    public String getNome() { return nome; }  
  
    public void setNome(String nome) { this.nome = nome;}  
  
    public int getIdade() { return idade; }  
  
    public void setIdade(int idade) { this.idade = idade;}  
  
    public String getEndereco() { return endereco; }  
  
    public void setEndereco(String endereco) { this.endereco = endereco; }  
  
    public String getTelefone() { return telefone;}  
  
    public void setTelefone(String telefone) { this.telefone = telefone; }  
}
```

```
import java.lang.reflect.Field;  
import java.lang.reflect.Method;  
import java.lang.reflect.Modifier;  
import java.lang.reflect.Parameter;  
import projetoexemplo.Pessoa;
```

```
public class ProjetoExemplo {  
  
    public static void main(String[] args) {  
  
        Class classe = Pessoa.class;  
        Pessoa p1 = new Pessoa();  
        p1.setNome("Ana Paula");  
        p1.setIdade(30);  
        p1.setTelefone("25698571");  
    }  
}
```

//Recuperando informações da classe pessoa

```
for (Field field : classe.getDeclaredFields()){  
    String campo = field.getName();  
    Class tipo = field.getType();  
    String nomeTipo = tipo.getTypeName();  
    System.out.println("Campo: " + campo + " , Tipo: " + nomeTipo );  
    System.out.println("Acessibilidade: " + Modifier.toString(field.getModifiers()));  
    System.out.println("Tem acesso: " + field.isAccessible());  
}
```

Resultados

Campo: nome , Tipo: java.lang.String
Acessibilidade: public
Tem acesso: false
valor = Ana Paula

Campo: idade , Tipo: int
Acessibilidade: public
Tem acesso: false
valor = 30

Campo: telefone , Tipo: java.lang.String
Acessibilidade: private
Tem acesso: false
valor = não pode acessar valor

Campo: endereco , Tipo: java.lang.String
Acessibilidade: private
Tem acesso: false
valor = não pode acessar valor

Método : setName Tipo de retorno: void
Acessibilidade: public
Parâmetros:
java.lang.String

Método : getName Tipo de retorno: class java.lang.String
Acessibilidade: public
Parâmetros:

Método : getIdade Tipo de retorno: int
Acessibilidade: public
Parâmetros:

Método : getEndereco Tipo de retorno: class java.lang.String
Acessibilidade: public
Parâmetros:

Método : setEndereco Tipo de retorno: void
Acessibilidade: public
Parâmetros:
java.lang.String

Método : getTelefone Tipo de retorno: class java.lang.String
Acessibilidade: public
Parâmetros:

Método : setIdade Tipo de retorno: void
Acessibilidade: public
Parâmetros:
int

Método : setTelefone Tipo de retorno: void
Acessibilidade: public
Parâmetros:
java.lang.String

```

try {
    //field.setAccessible(true);
    System.out.println("valor = " + field.get(p1) + "\n");
} catch (Exception e) {
    System.out.println("valor = não pode acessar valor\n");
}
}
System.out.println("");

```

//Recuperando os metodos da classe pessoa

```

for (Method method : classe.getDeclaredMethods()){
    String nome = method.getName();
    Class tipoRetorno = method.getReturnType();

    System.out.println("Método : " + nome + " Tipo de retorno: " + tipoRetorno);
    System.out.println("Acessibilidade: " + Modifier.toString(method.getModifiers()));
}

```

//Recuperandos os parâmetros do método

```

Class parametros[] = method.getParameterTypes();

```

```

System.out.println("Parâmetros: ");

```

```

for (Class c : parametros){
    System.out.println("\t" + c.getName());
}

```

```

System.out.println("");

```

```

}

```

```

}

```

```

}

```




```

1 import java.lang.reflect.Modifier
2
3 class Pessoa {
4     String nome
5     String endereco
6     String telefone
7
8     Pessoa(String Nome, String Endereco, String Telefone){
9         this.nome = Nome
10        this.endereco = Endereco
11        this.telefone = Telefone
12    }
13
14    def digaOla(){ return 'Ola Pessoa!!!' }
15 }
16
17 Pessoa.metaClass.idade = 0
18
19 def p1 = new Pessoa("Ana Paula", "Avenida Maracana", "25368547")
20
21 Pessoa.metaClass.setIdade(){int idade -> delegate.idade = idade}
22
23 Pessoa.metaClass.Mudar_Telefone(){String num -> delegate.telefone = num}
24
25 p1.setIdade(30)
26
27 println "Nome: " + p1.nome
28 p1.Mudar_Telefone("123568875")
29 println "Telefone: " + p1.telefone
30 println "Idade: " + p1.idade
31 println p1.digaOla()
32

```

Groovy

```

println "\n METODOS \n"
println Pessoa.metaClass.getMethods().name.sort().unique() + "\n\n"

List<MetaMethod> listaMetodos = Pessoa.metaClass.getMethods()

for (MetaMethod m : listaMetodos){
    println "Nome='" + m.name + "'    retorno= " + m.getReturnType()
    println "Assinatura : " + m.getSignature()
    println "Parametros : \n\t" + m.getParameterTypes()
    println "Acesso= " + Modifier.toString(m.getModifiers()) + "\n"
}

println "\n ATRIBUTOS\n"
println Pessoa.metaClass.getProperties().name.unique()
println " "

List<MetaProperty> listaPropriedade = Pessoa.metaClass.getProperties()

for (MetaProperty p : listaPropriedade){
    println "Nome='" + p.name + "'    Tipo= " + p.type
    println "Acesso= " + Modifier.toString(p.getModifiers()) + "\n"
}

```

Resultados

ATRIBUTOS

[nome, class, idade, telefone, endereco]

Nome='nome' Tipo= class java.lang.String
Acesso= public

Nome='class' Tipo= class java.lang.Class
Acesso= public final native

Nome='idade' Tipo= int
Acesso= public

Nome='telefone' Tipo= class java.lang.String
Acesso= public

Nome='endereco' Tipo= class java.lang.String
Acesso= public

Nome='Mudar_Telefone' retorno= class java.lang.Object
Assinatura : java.lang.Object Mudar_Telefone(java.lang.String)
Parametros :
[class java.lang.String]
Acesso= public

Nome='equals' retorno= boolean
Assinatura : equals(Ljava/lang/Object;)Z
Parametros :
[class java.lang.Object]
Acesso= public

Nome='getClass' retorno= class java.lang.Class
Assinatura : getClass()Ljava/lang/Class;
Parametros :
[]
Acesso= public final native

Nome='hashCode' retorno= int
Assinatura : hashCode()I
Parametros :
[]
Acesso= public native

METODOS

[Mudar_Telefone, digaOla, equals, getClass, getEndereco, getIdade, getMetaClass, getNome, getProperty, getTelefone, hashCode, invokeMethod, notify, notifyAll, setEndereco, setIdade, setMetaClass, setNome, setProperty, setTelefone, toString, wait]

Nome='setIdade' retorno= class java.lang.Integer
Assinatura : java.lang.Integer setIdade(java.lang.Integer)
Parametros :
[class java.lang.Integer]
Acesso= public

Nome='getIdade' retorno= class java.lang.Integer
Assinatura : java.lang.Integer getIdade()
Parametros :
[]
Acesso= public

Nome='setIdade' retorno= class java.lang.Object
Assinatura : java.lang.Object setIdade(int)
Parametros :
[int]
Acesso= public

Nome='notify' retorno= void
Assinatura : notify()V
Parametros :
[]
Acesso= public final native

Nome='notifyAll' retorno= void
Assinatura : notifyAll()V
Parametros :
[]
Acesso= public final native

Nome='toString' retorno= class java.lang.String
Assinatura : toString()Ljava/lang/String;
Parametros :
[]
Acesso= public

Nome='wait' retorno= void
Assinatura : wait()V
Parametros :
 []
Acesso= public final

Nome='getMetaClass' retorno= interface groovy.lang.MetaClass
Assinatura : getMetaClass()Lgroovy/lang/MetaClass;
Parametros :
 []
Acesso= public

Nome='wait' retorno= void
Assinatura : wait(J)V
Parametros :
 [long]
Acesso= public final native

Nome='getNome' retorno= class java.lang.String
Assinatura : getNome()Ljava/lang/String;
Parametros :
 []
Acesso= public

Nome='wait' retorno= void
Assinatura : wait(JI)V
Parametros :
 [long, int]
Acesso= public final

Nome='getProperty' retorno= class java.lang.Object
Assinatura : getProperty(Ljava/lang/String;)Ljava/lang/Object;
Parametros :
 [class java.lang.String]
Acesso= public

Nome='getTelefone' retorno= class java.lang.String
Assinatura : getTelefone()Ljava/lang/String;
Parametros :
 []
Acesso= public

Nome='digaOla' retorno= class java.lang.Object
Assinatura : digaOla()Ljava/lang/Object;
Parametros :
 []
Acesso= public

Nome='invokeMethod' retorno= class java.lang.Object
Assinatura : invokeMethod(Ljava/lang/String;Ljava/lang/Object;)Ljava/lang/Object;
Parametros :
 [class java.lang.String, class java.lang.Object]
Acesso= public

Nome='getEndereco' retorno= class java.lang.String
Assinatura : getEndereco()Ljava/lang/String;
Parametros :
 []
Acesso= public

Nome='setEndereco' retorno= void
Assinatura : setEndereco(Ljava/lang/String;)V
Parametros :
 [class java.lang.String]
Acesso= public

Nome='setEndereco' retorno= void
Assinatura : setEndereco(Ljava/lang/String;)V
Parametros :
 [class java.lang.String]
Acesso= public

Nome='setMetaClass' retorno= void
Assinatura : setMetaClass(Lgroovy/lang/MetaClass;)V
Parametros :
 [interface groovy.lang.MetaClass]
Acesso= public

Nome='setName' retorno= void
Assinatura : setName(Ljava/lang/String;)V
Parametros :
 [class java.lang.String]
Acesso= public

Nome='setMetaClass' retorno= void
Assinatura : setMetaClass(Lgroovy/lang/MetaClass;)V
Parametros :
 [interface groovy.lang.MetaClass]
Acesso= public

Nome='setName' retorno= void
Assinatura : setName(Ljava/lang/String;)V
Parametros :
 [class java.lang.String]
Acesso= public

Nome='setProperty' retorno= void
Assinatura : setProperty(Ljava/lang/String;Ljava/lang/Object;)V
Parametros :
 [class java.lang.String, class java.lang.Object]
Acesso= public

Nome='setTelefone' retorno= void
Assinatura : setTelefone(Ljava/lang/String;)V
Parametros :
 [class java.lang.String]
Acesso= public

Prototipação e Metaprogramação

- Forma de programação em que o comportamento de um novo objeto é realizado através de um processo de expansão do comportamento de objetos já existentes
- MOP (Meta-Object Protocol) e MetaClass
- Adiciona comportamento às classes em tempo de execução;
- Adicionar métodos às classes ou à apenas 1 objeto específico;
- Adiciona propriedades aos objetos já existentes;

Metaclassse

Resultado

```
GroovyConsole
File Edit View History Script Help

1 class Pessoa {
2     String nome
3     int idade
4
5     Pessoa(String nome, int idade){
6         this.nome = nome;
7         this.idade = idade;
8     }
9 }
10
11 Pessoa.metaClass.telefone = 00000000
12
13 def p1 = new Pessoa("Ana", 22)
14
15 assert p1.telefone == 00000000
16
17 Pessoa.metaClass.Mudar_Telefone(){int num -> delegate.telefone = num}
18
19 p1.Mudar_Telefone(25709494)
20
21
22 def p2 = new Pessoa("Paulo", 30)
23
24 assert p2.telefone == 00000000
25
26 p1.metaClass.pais = "Brasil"
27 p1.metaClass.Meu_Pais(){-> println "Eu sou do pais: $delegate.pais"}
28 p1.metaClass.idioma = "Portugues"
29
30
31 println "Objeto p1 Caracteristicas"
32 println "Nome = " + p1.nome
33 println "Idade = " + p1.idade
34 println "Telefone = " + p1.telefone
35 println "Idioma = " + p1.idioma
36 p1.Meu_Pais();
37 println " "
38
39 println "Objeto p2 Caracteristicas"
40 println "Nome = " + p2.nome
41 println "Idade = " + p2.idade
42 println "p2 telefone = " + p2.telefone
43 println "Idioma = " + p2.idioma
44
```

Objeto p1 Caracteristicas
Nome = Ana
Idade = 22
Telefone = 25709494
Idioma = Portugues
Eu sou do pais: Brasil

Objeto p2 Caracteristicas
Nome = Paulo
Idade = 30
p2 telefone = 0
Exception thrown

groovy.lang.MissingPropertyException: No such property: idioma for class: Pessoa
at ConsoleScript38.run(ConsoleScript38:46)

Execution terminated with exception.

GroovyConsole

File Edit View History Script Help

```
1 class Animal {
2     String nome
3     String tipo
4     int idade
5
6     Animal(String nome, int idade, String tipo){
7         this.nome = nome;
8         this.idade = idade;
9         this.tipo = tipo;
10    }
11 }
12
13 Animal.metaClass.raca = "";
14
15 def cat = new Animal("Felix", 2, "Gato")
16 def dog = new Animal("Bob", 1, "Cachorro")
17
18 cat.raca = "siames"
19 dog.raca = "vira-lata"
20
21 cat.metaClass.miar(){-> println "miau miau"}
22
23 dog.metaClass.latir(){-> println "au au"}
24
25 println "Caracteristicas Gato"
26 println "Nome: " + cat.nome
27 println "Idade: " + cat.idade
28 println "Tipo: " + cat.tipo
29 println "Raca: " + cat.raca
30 cat.miar()
31
32 println " "
33 println "Caracteristicas Cachorro"
34 println "Nome: " + dog.nome
35 println "Idade: " + dog.idade
36 println "Tipo: " + dog.tipo
37 println "Raca: " + dog.raca
38 dog.latir()
39
40 println " "
41 assert cat.miar() == "au au"
42
```

Metaclasses

Resultado

```
Caracteristicas Gato
Nome: Felix
Idade: 2
Tipo: Gato
Raca: siames
miau miau
```

```
Caracteristicas Cachorro
Nome: Bob
Idade: 1
Tipo: Cachorro
Raca: vira-lata
au au
```

```
miau miau
Exception thrown
```

Assertion failed:

```
assert cat.miar() == "au au"
      |    |      |
      |    null   false
      Animal@7cf9bf0c
```

at ConsoleScript0.run(ConsoleScript0:42)

Execution terminated with exception.



Obrigada!