Lenguaje de Programación Groovy

http://groovy-lang.org

https://groovyconsole.appspot.com

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Introducción

- Groovy es un lenguaje potente, con tipado opcional y dinámico, con capacidad de tipado estático y compilación estática, para la plataforma Java.

- Destinado a mejorar la productividad del desarrollador gracias a una sintaxis concisa, familiar y fácil de aprender.

- Se integra sin problemas con cualquier programa Java.

- Provee potentes características, como poder incluir funciones de scripting, soporte para la creación de DSL (lenguaje específico de dominio), y capacidades de meta-programación en tiempo de ejecución y en tiempo de compilación.

- Capacidad de programación funcional.

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Instalación

- Descargar desde http://groovy-lang.org/download.html
- Setear la variable de ambiente GROOVY_HOME con el directorio donde se instalo GROOVY
- Setear la variable de ambiente PATH con el directorio bin dentro de GROOVY_HOME
- Tener seteada la variable de ambiente JAVA_HOME

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Comentarios

- #! Comentario para scripting. Ejemplo #!/usr/bin/groovy
- Comentario de linea simple //
- Comentario de múltiples lineas /* */
- Comentario para JavaDoc /** */

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Imports

- Groovy importa algunos paquetes por defecto:
    - import java.lang.*
    - import java.util.*
    - import java.net.*
    - import java.io.*
    - import java.math.BigInteger
    - import java.math.BigDecimal
    - import groovy.lang.*
    - import groovy.util.*

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

```
public class HelloWorld {
    String name;

public void setName(String name)
    { this.name = name; }

public String getName(){ return name; }

public String greet()
    { return “Hello “+ name; }

public static void main(String args[]){
    HelloWorld helloWorld = new HelloWorld()
    helloWorld.setName(“Groovy”)
    System.err.println( helloWorld.greet() )
  }
}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

Todo es publico en Groovy a no ser que se le indique lo contrario.
Los puntos y coma (;) son opcionales.

```
class HelloWorld {
    String name
    void setName(String name)
        { this.name = name }
    String getName(){ return name }
    String greet()
        { return "Hello "+ name }
    static void main(String args[]){
        HelloWorld helloWorld = new HelloWorld()
        helloWorld.setName("Groovy")
        System.err.println( helloWorld.greet() )
    }
}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

No es necesario escribir los get/set, groovy los agrega automáticamente.
En los métodos main no es necesario indicar el parametro String[]
Imprimir por consola esta simplificado.

```
class HelloWorld {
    String name
    String greet()
        { return "Hello "+ name }
    static void main( args ){
        HelloWorld helloWorld = new HelloWorld()
        helloWorld.setName("Groovy")
        println( helloWorld.greet() )
    }
}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

Podemos usar la palabra reservada def cuando no nos interesa especificar el tipo de dato.
Groovy se da cuenta cual es le tipo de dato correcto. Esto se llama duck typing.

```
class HelloWorld {
    String name
    def greet()
        { return "Hello "+ name }
    static def main( args ){
        def helloWorld = new HelloWorld()
        helloWorld.setName("Groovy")
        println( helloWorld.greet() )
    }
}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

En groovy se puede intepolar las variables a través de Gstring. Permite encerrar cualquier expresión Groovy con ${} dentro de un String.

```
class HelloWorld {
    String name
    def greet(){ return "Hello ${name}" }
    static def main( args ){
        def helloWorld = new HelloWorld()
        helloWorld.setName("Groovy")
        println( helloWorld.greet() )
  }
}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

La palabra return es opcional, el valor de retorno va a ser la ultima expresión evaluada.
No es necesario utilizar def en métodos estáticos.

```
class HelloWorld {
    String name
    def greet(){ "Hello ${name}" }
    static main( args ){
        def helloWorld = new HelloWorld()
        helloWorld.setName("Groovy")
        println( helloWorld.greet() )
    }
}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

Provee constructor por defecto con parámetros nombrados.
POGOs soportan acceso a propiedades como arreglos (bean[prop])
y con notación (.) (bean.prop)

```
class HelloWorld {
    String name
    def greet(){ "Hello ${name}" }
    static main( args ){
        def helloWorld = new HelloWorld()
        helloWorld.setName("Groovy")
        println( helloWorld.greet() )
    }
}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Java a Groovy

Groovy soporta scripts que también igual que las clases son compiladas en bytecode. Los script permiten definir clases dentro de ellos.

```
class HelloWorld {
    String name
    def greet() { "Hello $name" }
}

def helloWorld = new HelloWorld(name:"Groovy")
println helloWorld.greet()
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Listas

- Definición por medio de corchetes [elem,..]
- Indexado en 0
- lista[posicion]=valor
- assert lista[posicion]==valor
- assert lista.size()==tamaño

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Listas

Agregar Elementos

```
myList = []
myList += 'a'
assert myList == ['a']

myList += ['b','c']
assert myList == ['a','b','c']
```

```
myList = []
myList << 'a' << 'b'
assert myList == ['a','b']
assert myList - ['b'] == ['a']
assert myList * 2 == ['a','b','a','b']
```

-

Concatenar

```
def list = [1,2,3]
assert list.join('-') == '1-2-3‘
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Listas

Eliminar Elementos

```
list = ['a','b','c']
list.remove(2) // elimina el último elemento
assert list == ['a','b']

list.remove('b') // elimina el elemento ‘a’
assert list == ['a']

list = ['a','b','b','c']
list.removeAll(['b','c'])
assert list == ['a']
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Listas

Manipulando Listas

Aplanando

```
assert [1,[2,3]].flatten() == [1,2,3]
```

Intersección y disyunción

```
assert [1,2,3].intersect([4,3,1])== [3,1]
assert [1,2,3].disjoint([4,5,6])
```

Ultimo elemento

-

```
assert [1,2,3]. pop() == 3
```

Orden e inversión

```
assert [1,2].reverse() == [2,1]
assert [3,1,2].sort() == [1,2,3]
assert [3,1,2].sort {a,b-> b.compareTo a} == [3,2,1]
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Mapas

- Se define utilizando corchetes ([]) listando pares de clave y valor separado por coma (,). El par de clave valor se define uniendo la clave con el valor con los dos puntos (:)

- Mapa vacío [:]

- def http=[100:’CONTINUE’,200:’OK’,400:’BAD REQUEST’]

- assert http[200]==‘OK’

- http[500]=‘INTERNAL SERVER ERROR’

- assert http.size()==4

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Mapas

Manipulando Mapas

Tamaño

```
def myMap = [a:1, b:2, c:3]
assert myMap instanceof HashMap
assert myMap.size() == 3
assert myMap['a'] == 1
def emptyMap = [:]
assert emptyMap.size() == 0
```

- ### Se puede evitar las comillas

```
assert ['a':1] == [a:1]
```

Elementos existentes

```
def myMap = [a:1, b:2, c:3]
assert myMap['a'] == 1
assert myMap.a == 1
assert myMap.get('a') == 1
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

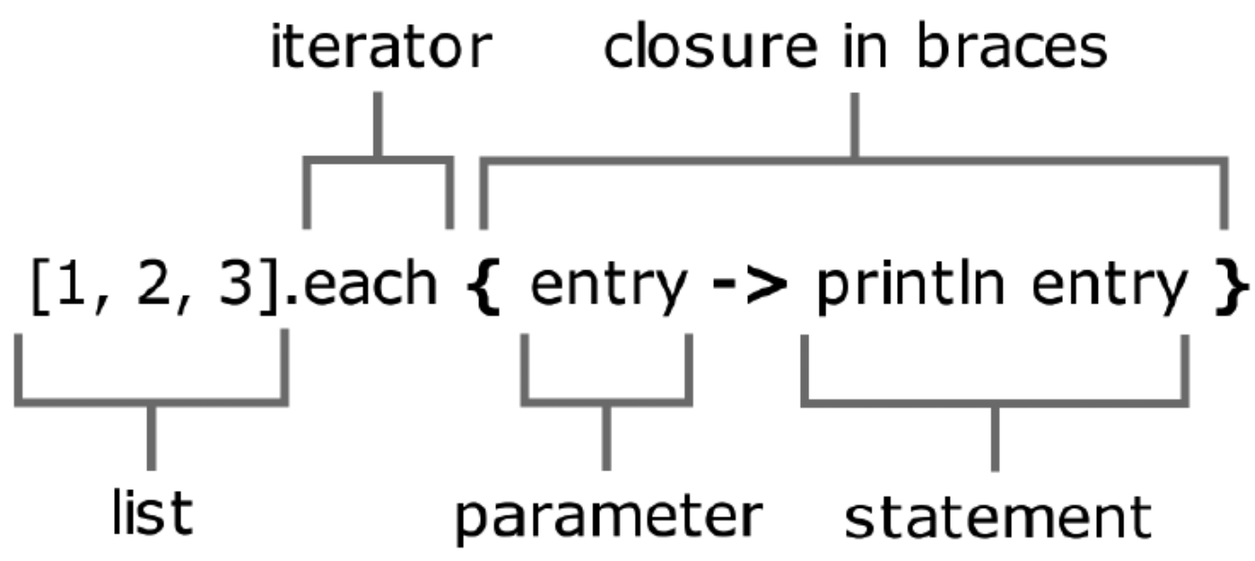
Rangos

- Un rango es un punto de inicio y un punto de fin con una noción de como moverse sobre los puntos.
- def x= 1..10
- assert x.contains(5)
- assert x.contains(15)==false
- assert x.size()==10
- assert x.from==1
- assert x.to==10
- assert x.reverse()==10..1

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Closures

- Los closures son bloques de código que permiten encapsular comportamiento que puede ser ejecutado.



- El método each recibe como parámetro un clousure. El closure posee un parámetro y una sentencia que lo utiliza

Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Closures

- Pueden ser definidos e invocados independientemente

```
def suma = {x,y -> return x+y}
assert suma(4,3)==7
assert suma.call(2,6)==8
```

- Cuando no se define ningún parámetro, se define una pseudo variable *it*

```
def log=“”
(1..10).each({log+=it})
assert log == “12345678910”
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Listas

Manipulando Listas

Colectar

```
def doubled = [1,2,3].collect{ item -> item*2 }
assert doubled == [2,4,6]
```

Selección

```
def impar = [1,2,3].findAll{ item -> item % 2 == 1}
assert impar == [1,3]
```

Recorrido

```
def store = ''
list.each { item -> store += item }
assert store == '123'
```

Recorrido invertido

```
store = ''
list.reverseEach{ item -> store += item }
assert store == '321'
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Listas

Manipulando Listas

Acumulador

```
result = list.inject(0){ acumulado, guests ->
acumulado += guests
}
assert result == 0 + 1+2+3
```

-

```
factorial = list.inject(1){ fac, item -> fac *= item}
assert factorial == 1 * 1*2*3
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar

Mapas

Metodos de consulta

```
assert myMap.any {entry -> entry.value > 2 }
assert myMap.every {entry -> entry.key < 'd'}
myMap.each {entry -> println(entry.key + entry.value)
```

```
def myMap = [a:1, b:2, c:3]
for (key in myMap.keySet()) {
println(key+’ ‘) //imprime a b c
}
```

- ```
  for (value in myMap.values()) {
  println(value+’ ‘) //imprime 1,2,3
  }
  ```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar
  ```

Mapas

Metodos de consulta

```
def myMap = [a:1, b:2, c:3]
abMap = myMap.findAll { entry -> entry.value < 3}
assert abMap.size() == 2

def found = myMap.find { entry -> entry.value < 2}
assert found.key == 'a‘
```

```
def doubled = myMap.collect { entry -> entry.value *= 2}
assert doubled.every {item -> item %2 == 0}
```

-

```
def addTo = []
myMap.collect(addTo) { entry -> entry.value *= 2}
assert addTo.every {item -> item %2 == 0}
```


Lic. Oscar Eduardo Quinteros
oequinteros@tecno.unca.edu.ar