
EDUCACIÓN
EN LÍNEA

# Computación Paralela y Distribuida

Groovy

Nombres: Diego Vaca, Anderson Lucero, Cristian Quilumbaquin

# La historia de Groovy


- En Groovy One 2004, una reunión de desarrolladores de Groovy en Londres, James Strachan pronunció un discurso de apertura contando la historia de cómo llegó a la idea de inventar Groovy. Él y su esposa estaban esperando un avión con retraso. Mientras ella iba de compras, visitó un cibercafé y espontáneamente decidió ir al sitio web de Python y estudiar el lenguaje.

# La historia de Groovy


- En el transcurso de esta actividad, el se volvió más y más intrigado. Siendo un gran desarollador de java, él reconoció que su lenguaje carecía de muchas de las muy útiles características de Python tales como el muy importante comportamiento dinámico. Esta idea lo llevó a desarrollar Groovy.

# La historia de Groovy


- Principios fundamentales que guiaron el desarrollo de groovy:
  - Tener muchas funciones.
  - Ser un lengauje amigable con java.
  - Traer los atractivos beneficios de los lenguajes dinámicos a una plataforma robusta y con buen soporte.

# ¿Qué es Groovy?


- Es un lenguaje dinámico, opcionalmente escrito para la plataforma java con muchas características que son inspiradas por lenguajes como Python, Ruby, entre otros, poniéndolos a disposición de desarrolladores java utilizando una sintaxis similar a la de java.
- En contraste de otros lenguajes alternativos, este no es diseñado como un remplazo para java, sino como un ayudante.

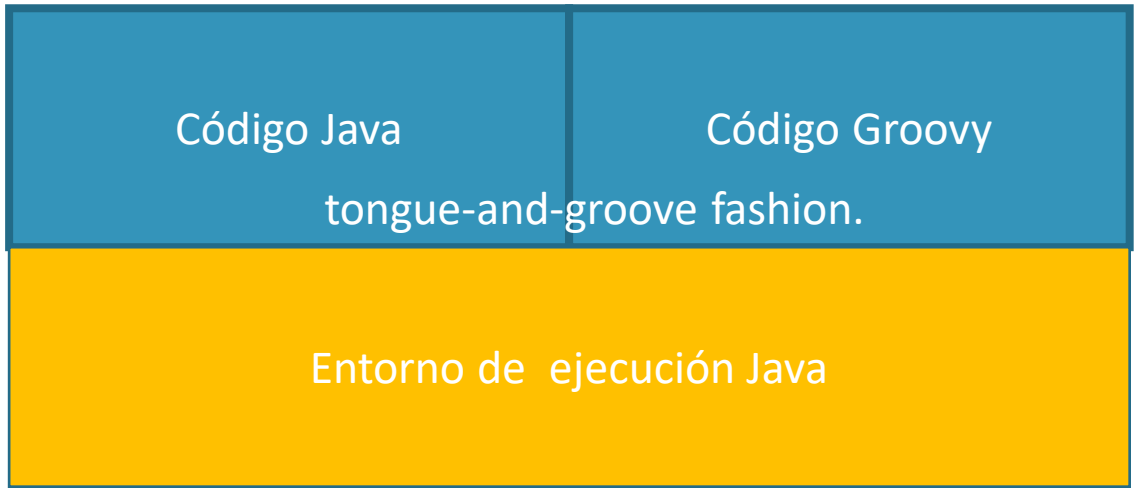
# ¿Qué es Groovy?


- Groovy es comúnmente conocido como un lenguaje de escripting y trabaja muy bien para esto.

- Groovy es muy cercanamente unido a la plataforma java (muchas partes de groovy son escritas en java y el resto son escritas en el mismo groovy).

# La integración Perfecta:

- Ser amigable con java significa dos cosas:
  - Una perfecta integración con el entorno de ejecución de java.
  - Tener una sintaxis que esté alineada con java.

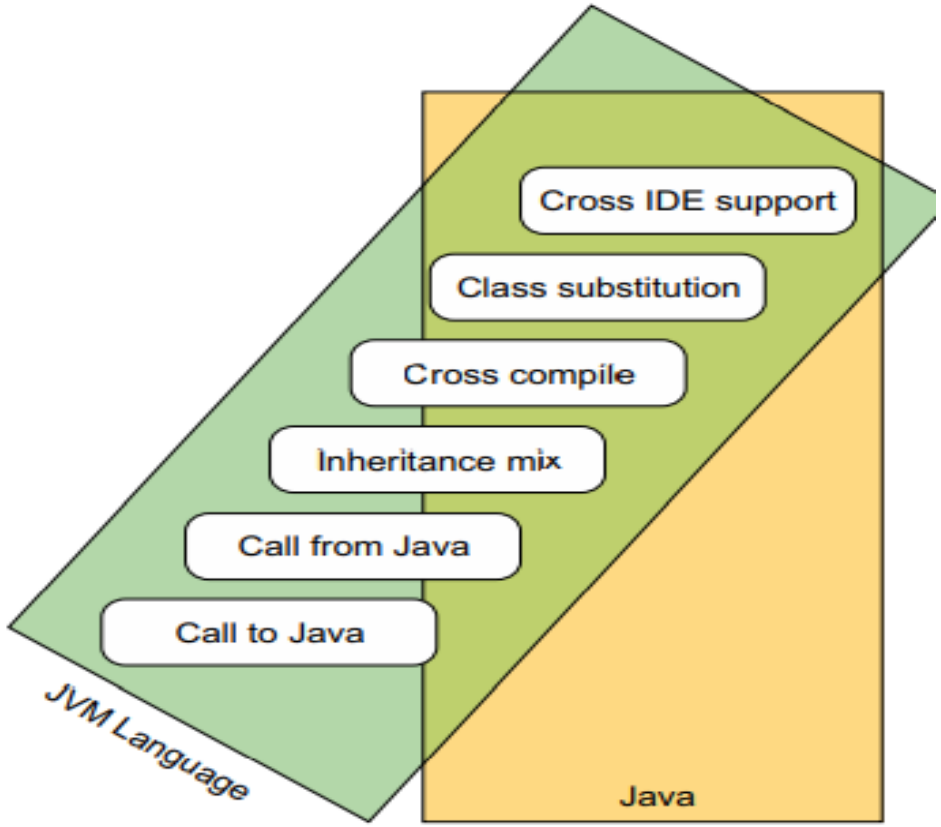


# La integración Perfecta:


- Llamar a Java es un ejercicio sencillo. Es algo que ofrecen todos los lenguajes de JVM, al menos de los que vale la pena hablar. Todos lo hacen posible, algunos quedándose dentro de sus propias abstracciones que no son de Java, y algunas proporcionando una puerta de enlace. Groovy es uno de los pocos que lo hacen a su manera y a la manera de Java al mismo tiempo, porque no hay ninguna diferencia.

# La integración Perfecta:


- java.lang.Object
- java.util.Date
- new MyGroovyClass();



# La integración Perfecta:

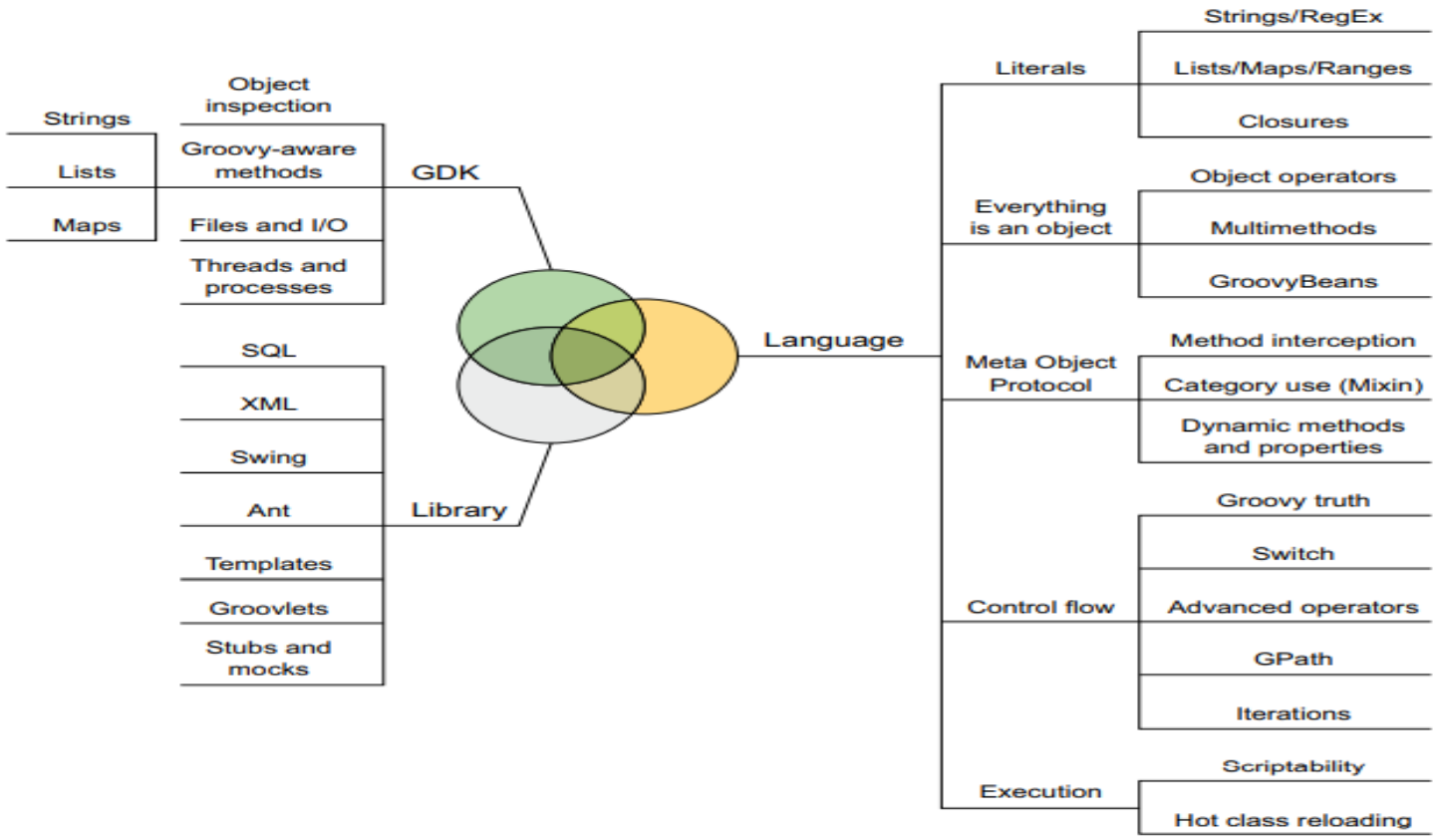

- Lineamientos de sintaxis:

```
import java.util.*;          // Java
Date today = new Date();     // Java

today = new Date()           // Groovy

require 'date'                # Ruby
today = Date.new              # Ruby

import java.util._           // Scala
var today = new Date         // Scala

(import '(java.util Date))  ; Clojure
(def today (new Date))      ; Clojure
(def today (Date.))         ; Clojure alternative
```

# Un lenguaje rico en funciones:


- Dar una lista de características de Groovy es un poco como enumerar los movimientos que puede realizar un bailarín. Aunque cada función es importante en sí misma, es lo bien que funcionan juntas lo que hace que brille.

# Un lenguaje rico en funciones:



# Tipos de datos simples en Groovy


- Groovy admite un conjunto limitado de tipos de datos a nivel de lenguaje; es decir, ofrece constructores para declaraciones literales y operadores especializados.

- En Groovy, todo es un objeto. Después de todo, es un lenguaje orientado a objetos. Groovy no es como Java, que está orientado a objetos y aparte algunos tipos integrados.

# Sistema de tipos en JAVA: primitivos y referencias


- Java distingue entre tipos primitivos tales como( boolean, short, int, double, float, char y byte) y tipos de referencia como (object y String). Hay un conjunto fijo de datos primitivos, y estos son los únicos datos que tienen semántica de valor, donde el valor de una variable de ese tipo es el número real, carácter o valor de verdadero o falso. NO se puede crear tipos de datos propios en Java.

# En groovy todo es un objeto:


- Para hacer de groovy un completo lenguaje orientado a objetos y porque a nivel de JVm java no soporta operaciones orientadas a objetos tales como llamadas a los métodos en tipos primitivos, los diseñadores de groovy decidieron desechar los datos primitivos. Cuando Groovy necesita almacenar valores que habrían usado tipos primitivos en java. Groovy utiliza el contenedor de clases que son proveeidas por la plataforma de java.

# Tipos de datos primitivos y sus contenedores


| Primitive type | Wrapper type | Description |
|---|---|---|
| byte | java.lang.Byte | 8-bit signed integer |
| short | java.lang.Short | 16-bit signed integer |
| int | java.lang.Integer | 32-bit signed integer |
| long | java.lang.Long | 64-bit signed integer |
| float | java.lang.Float | Single-precision (32-bit) floating-point value |
| double | java.lang.Double | Double-precision (64-bit) floating-point value |
| char | java.lang.Character | 16-bit Unicode character |
| boolean | java.lang.Boolean | Boolean value (true or false) |

# Groovy Dinámico


- Es importante entender que incluso cuando tulizamos todo de sus capacidades dinámicas, Groovy está proporcionando seguridad de tipo completo en el tiempo de ejecución.
- Por defecto, Groovy es un lenguaje dinñamico. Se puede de forma segura dejar marcas de tipo en la mayoría de escenarios y saber que groovy va a hacer las revisiones apropiadas en el tiempo de ejecución para aseguridar la seguridad de tipos cuando sea requerida. A causa de que las “type markers” son opcionales en groovy se le llama “tipeo opcional”. Los tipos están ahí por supuesto, pero por supuesto, se puede elegir no hacerlas explicitas en el código.

# El caso del tipeo opcional:


- El omitir marcas de tipeo no es solo conveniente para los programadores vagos, sino también  es muy útil para el tipado pato

TIPADO PATO:
"Si camina como un pato y grazna
como un pato, debe ser un pato ". Los lenguajes de escritura débil generalmente le permiten llamar a cualquier método o acceder a cualquier propiedad de un objeto, incluso si no lo sabe en el momento de la compilación o incluso en tiempo de ejecución que el objeto es de un tipo conocido que contiene ese método o
propiedad. Esto significa que sabe el tipo de objetos que espera que tengan los firma o propiedad.

# A quien va dirigido Groovy

**A los programadores Java:**

Un programador Java con muchos años de experiencia conocerá todas las partes importantes del entorno Java y su API, así como otros paquetes adicionales.

Sin embargo, algo aparentemente tan sencillo como listar todos los archivos recursivamente que cuelgan de un directorio se hace pesado en Java y suponiendo que existiera la función eachFileRecurse() tendríamos algo similar a:

# A quien va dirigido Groovy

```
1  public class ListFiles {
2      public static void main(String[] args){
3          new java.io.File(".").eachFileRecurse(
4              new FileListener() {
5                  public void onFile (File file) {
6                      System.out.println(file.toString());
7                  }
8              }
9          );
10     }
11 }
```

# A quien va dirigido Groovy

**Mientras con groovy podria quedar en una sola linea:**

```
groovy -e "new File('.').eachFileRecurse { println it}"
```

# A quien va dirigido Groovy

**A los programadores Script:**

Groovy también va dirigido a aquellos programadores que han pasado muchos años trabajando con lenguajes de programación como Perl, Ruby, Python o PHP. En ocasiones, sobre todo en las grandes empresas, los clientes requieren obligatoriamente el uso de una Plataforma robusta como puede ser Java. Estos programadores deben reciclarse, ya que no es factible cualquier solución que pase por utilizar un lenguaje script como los mencionados anteriormente.

Continuamente este tipo de programadores afirman que se sienten frustrados por toda la ceremonia que conlleva un lenguaje como Java y la frase "si pudiera utilizar un lenguaje como Ruby, podría escribir todo este método en una sola línea" es su pan de cada día. Groovy puede darles un respiro y una vía de escape a este tipo de programadores.

# A quien va dirigido Groovy

**A los programadores agiles y extremos:**

Este tipo de programadores están acostumbrados a trabajar con la próxima revisión tan cercana, que pararse a pensar en utilizar un nuevo lenguaje de programación queda fuera de sus principales objetivos. Sin embargo, una de los aspectos más importantes para los programadores ágiles, consiste en tener siempre a su disposición los mejores recursos y metodologías para llevar a cabo su trabajo.

Y ahí es donde Groovy les puede aportar mucho, ya que Groovy es un lenguaje perfecto para realizar esas tareas de automatización tan habituales en sus proyectos. Estas tareas van desde la integración continua y la realización de informes, hasta la documentación automática e instalación.

# Bibliografía:


- *Groovy*. (s. f.). Groovy. Recuperado 4 de diciembre de 2020, de https://groovy-lang.org/learn.html

- König, D., King, P., Laforge, G., D'Arcy, H., Champeau, C., Pragt, E., & Skeet, J. (2015). *Groovy in Action* (2.ª ed., Vol. 1). Manning Publications.

UTE EC

# ¡GRACIAS!

TRAS CENDE MOS