# Groovy 概览

作者: songtianyi create@2020-11-24

## 前言

和 Go 的火热相比，Groovy 显得名不见经传。但 Groovy 目前在 tiobe 的排名比 Go 是要高的。Groovy 的成功案例有 Gradle, Grails, Spock testing等等。我们在用静态语言做数据处理的时候是非常痛苦的，而且在客户环境测试的时候，我们需要经过反复的修改，发布，上传，更新等操作，对客户体验影响很大。Groovy作为脚本语言，既能提供灵活性，又能复用 Java 的代码遗产，是一个不错的选项。

## Groovy 是什么

- 运行在 jvm 上，设计之初就考虑从 Java 过渡到 Groovy 的平滑性，能够load jar包，执行 java 函数，你甚至可以同时使用 java 语法和 Groovy 语法来编程；可以编译成 java 字节码，集成进 java 程序里运行；既对 java 友好，又具有比 java 更丰富的语法特性
- 提供闭包，元编程能力，还有强大的 DSL
- Gradual typing, 既支持静态类型(static typing)，也支持动态类型(dynamic typing)
- Groovy 的类型都属于 `java.lang.Object` 的子类型
- 多范式，IP, SP, PP, OOP, FP, MP

## 安装

下载并安装 groovy

## 类型系统

| Lang | Typed | Static and dynamic checks | Strongly checked | Weakly or strongly typed | Dynamically or statically typed | Type theories | Paradigms |
|---|---|---|---|---|---|---|---|
| Groovy | ☑ | optional | ☑ | weakly | optional | generic, overloading, subtype | IP, SP, PP, OOP, FP, MP |

- optional typing(Gradual typing): optional typing 是指类型系统的检查既可在编译时，也可以在运行时，Groovy 将选择权留给使用者。
- weakly typed: 类型是可变的

```
groovy:000> a = '1'
===> 1
groovy:000> a.class
===> class java.lang.String
groovy:000> a = 1
===> 1
```

```
groovy:000> a.class
===> class java.lang.Integer
groovy:000>
```

- type inference: 支持类型推断

```
groovy:000> 1.class
===> class java.lang.Integer
groovy:000>
```

## 语法规范

虽然大部分 Groovy 代码和 Java 一致，但是 Groovy 语法并不是 Java 语法的超集(superset)。
比如，Java 不支持下面的 `for` 表达式:

```
for(init1,init2;test;inc1,inc2)
```

Groovy 里的 `==` 是判断相等性(equality)，而不像 java 是判断同一性(identity)， `==` 在 Groovy 里被解释为 java 里的 `equals`

除了这些细微的差别，大部分的 java 语法都是可以被 Groovy 兼容的。Groovy 相对 Java 新增的语法包括:

- 通过新的表达式和运算符访问 java 对象
- 更丰富的对象创建方式

```
def http = [
 100 : 'CONTINUE',
 200 : 'OK',
 400 : 'BAD REQUEST'
]
```

- 提供了高级的流控制语法
- 新的数据类型，及附带的运算符和表达式
- 支持通过 `\` 符号分割代码
- 支持将 `""`里的内容解释成表达式
- `null` 在条件表达式里会被当作 `false` 来对待, 而在 java 里则会出现异常。

```
groovy:000>
groovy:000> a="1+1"
===> 1+1
groovy:000> evaluate(a)
```

```
===> 2
groovy:000>
```

除了新增的语法，Groovy 相对 Java 做了大量的语法优化，使我们写起来更美观和简洁。

- 当没有冲突的时候可以省略包的前缀，可以省略一些括号, 分号等

```
// java
java.net.URLEncoder.encode("a b");
```

```
// Groovy
URLEncoder.encode 'a b'
```

虽然 Groovy 提供了更多的灵活写法，但是我们需要斟酌使用，为自己的代码可读性负责。

- 包自动导入，无需指定包名。虽然 java 会自动导入 `java.lang.*`, 当然仅此而已。
- 不需要显式的类型转换
- 和 rust 一样，return 是可选的。

```
def sum(a, b) {
    a + b
    // return a + b
}
```

- 不需要显式地 throw 捕捉到的 exception

## Types

### range

range 使用 `left..right` 形式表达。range 是一个对象，可以调用该对象的方法, 可以作为 `witch case`

```
print 0..1
print 0..<2 // 不包含2
```

```
def result = ''
(5..9).each { element -> result += element }
assert result == '56789'

assert 5 in 0..10

assert (0..10).isCase(5)
```

```
def age = 36
switch(age){
 case 16..20 : insuranceRate = 0.05 ; break
 case 21..50 : insuranceRate = 0.06 ; break
 case 51..65 : insuranceRate = 0.07 ; break
 default: throw new IllegalArgumentException()
}
assert insuranceRate == 0.06

def ages = [20, 36, 42, 56]
def midage = 21..50
assert ages.grep(midage) == [36, 42]
```

**lists**

# Java 和 Groovy 的亲和性

## Java 调用 Groovy

编写一个 Groovy 类 `MyGroovyClass` , 然后编译成 `MyGroovyClass.class` , 将其放进 classpath , 然后可以在 java 中使用该类

```
new MyGroovyClass(); // create from Java
```