# Groovy 与 Java的不同

Groovy尽可能的试着成为Java开发者意料之中的一种语言。我们在设计Groovy的时候一直在试着遵循最小惊奇原则，尤其是为了那些有着Java背景的开发者。

## 默认引入

所有的这些包和类是默认引入的。你不必用一个显式的import来引入它们：

* java.io.\*
* java.lang.\*
* java.math.BigDecimal
* java.math.BigInteger
* java.net.\*
* Java.util.\*
* groovy.lang.\*
* groovy.util.\*

## 多方法

在Groovy中，方法是在运行时被调用的。这被称作运行时分配或多方法。意思是说方法是在运行时基于参数的类型来选择的。在Java中，是相反的：方法是基于声明类型在编译时被选择的。

下面的代码，使用Java写的，可以被Java和Groovy编译，但是表现的完全不同：

int method(String arg) {

return 1;

}

int method(Object arg) {

return 2;

}

Object o = "Object";

int result = method(o);

在Java中，你会得到：

assertEquals(2, result);

然而在Groovy中：

assertEquals(1, result);

这是因为Java使用的是静态信息类型，o被声明为一个Object，而Groovy会在运行时方法确实被调用时选择。因为o是一个String，因此参数为String的方法会被调用。

## 3. 数组初始化

在Groovy中，{...}块是为闭包保留的。这就意味着你不能用这个语法来创建数组，例如：

int[] array = { 1, 2, 3}

而必须用：

int[] array = [1,2,3]

## 包空间可见性

在Groovy中，省略变量的修饰符不会像在Java中一样导致包内私有：

class Person {

String name

}

而是创建一个私有属性以及与之相关的getter和setter方法。

如果想使它变成包内私有的，可以用@PackageScope来注解：

class Person {

@PackageScope String name

}