**Groovy集合（map）**

声明map

Map的规范和上一篇博客提到的list类似，helist一样,Map保证可以通过下标来获取或者设置值，不同的是map可以使用任何类型作为下标操作符的参数，而list仅仅支持整数，list知道list中元素的顺序，而map一般不知道，特殊的map如java.util.TreeMap也可以知道他们的key的顺序。

理论上 ，任何类型都可以作为map的键（key）或者值（value）,当使用特殊类型作为key的时候，需要遵循java.util.Map在javaDoc中的规定。

通过[:]来声明一个空的map，map缺省的类型是java.util.HashMap,也可以通过调用构造方法进行显示声明，这样map仍然可以使用下标操作符进行操作。

def map = [a:1,b:2,c:3]

assert map instanceof HashMap

assert map.size() == 3

assert map['a'] == 1

emptyMap = [:]

assert map.size() == 0

explicitMap = new TreeMap()

explicitMap.putAll(map)

assert explicitMap['a'] == 1

一般情况下key的类型都是字符串，在声明map的时候可以忽略字符串标记（单引号或者双引号）：assert ["a":1] == [a:1],如果key没有包含特殊字符（需要符合有效表示规则）而且不是groovy的关键字，那么允许通过这种便利的方式进行声明。

这种方式也有不便的地方，例如，本地变量的内容被用作key。假如有一个本地变量x，它的内容为’a’，由于[x:1]等价于[‘x’:1],那么如何保证它等于[‘a’:1]呢？通过把把符号放在圆括号中，强制让groovy将符号看做一个表达式：

def x = 'a'

assert [x:1] == ['x':1]

assert ['(x)'] == ['x':1]

使用map操作符

Groovy中有三种从map中获取对象的方法：

使用下表操作符，因为map实现了getAt方法；

使用点语法像使用属性那样来获取对象；

使用get()方法，可以传递一个缺省值，在map中没有相应的key时，允许返回缺省值，如果没有缺省值放回null，如果get(key,default)被调用时，key没有找到，缺省值返回，那么key:defualt对将被增加到map中。

def map == [a:1,b:2,c:3]

assert map['a'] == 1

assert map.a == 1

assert map.get('a') == 1

assert map.get('a',0) == 1

assert map['d'] == null

assert map.d == null

assert map.get('d') == null

assert map.get('d',0) == 0

assert map.d == 0

map['d'] = 1

assert map.d == 1

map.d = 2

assert map.d == 2

将值分配给map时可以使用下标操作符或者点语法，在使用点语法时，如果key包含了特殊字符，需要使用字符串符号括起来。

map = ["a.b" : 1]

assert map.'a.b' == 1

如果仅仅写成map.a.b,这里是不会正常工作的，这其实等价于map.getA().getB().

map方法

map = [a:1,b:2,c:3]

other = [c:3,a:1,b:1]

assert map == other

assert map.isEmpty() == false //是否为空

assert map.size() == 3

assert map.containKey('a') //判断map中是否包含给定的key

assert map.containValue(1) //判断map中是否包含给定的value

//keySet()返回一个key的set集合，不包含重复的key，没有固定的顺序

assert map.keySet() == toSet(['a','b','c'])

assert toSet(map.values()) == toSet([1,2,3])

assert map.entrySet() instanceof Collection

assert map.any{entry -> entry.value > 2} //与list中同名方法类似，

assert map.every{entry -> entry.key < 'd'}

def toSet(list){

new java.util.HashSet(list)

}

遍历map

//each()

//each()接受两种形式的闭包：传递一个参数，那么这个参数代表map的一个entry；传递两个参数，那么就是key和value

def map = [a:1,b:2,c:3]

def store = ''

map.each{entry -> store += entry.key + entry.value}

assert store == "a1b2c3"

def store = ''

map.each{key,value -> store += key+ value}

assert store == "a1b2c3"

修改map

def map = [a:1,b:2,c:3]

map.clear()

assert map.isEmpty()

def map = [a:1,b:2,c:3]

map.remove('a')

assert map.size() == 2

def map = [a:1,b:2,c:3]

assert [a:1,b:2] == map.subMap(['a','b'])

assert [a:1,b:2] == map.findAll{entry -> entry.value < 3}

neWMap = map.find{entry -> entry.value < 3} //返回符合条件一个entry

//collect() 返回闭包结果组成的list（是否返回可选）

assert [2,4,6] == map.collect{entry -> entry.value\*= 2}

//collect()可以接受一个list

assert [1,2,2,4,6] == map.collect([1,2]){entry -> entry.value\*= 2}

19

map实战

统计文章中单词出现的频率

def text =

"""

When I say I believe all children can learn, people sometimes misunderstand.

Because I have been working with poor, minority children in Harlem for the last 25 years,

some people think I am talking about good kids in bad environments — that if you give a bright kid from a poor family a good educational support system,

he or she can succeed. That's absolutely true, but that's not what I mean

"""

def works = text.tokenize()

def wordFrequency = [:]

words.each{work ->

wordFrequency[work] = wordFrequency.get(work,0) + 1

}

def workList = wordFrequency.keySet().toList()

workList.sort{wordFrequency[it]}

def statistic = "\n"

workList[-1..-6].each{work ->

statistic += work.padLeft(12) + ":"

statistic += wordFrequency[work] + "\n"

}

assert statistic ==

"""

I:5

a:3

good:2

in:2

people:2

can:2

"""