【Kotlin】集合 Iterable/Collection/List/Set/Map

出處：https://blog.csdn.net/vitaviva/article/details/107587134

下標操作類

- contains —— 判斷是否有指定元素
- elementAt —— 傳回對應的元素，越界會拋IndexOutOfBoundsException
- firstOrNull —— 傳回符合條件的第一個元素，沒有回傳null
- lastOrNull —— 傳回符合條件的最後一個元素，沒有回傳null
- indexOf —— 傳回指定元素的下標，沒有回傳-1
- singleOrNull —— 傳回符合條件的單一元素，如有沒有符合或超過一個，傳回null

判斷類

- any —— 判斷集合中是否有滿足條件的元素
- all —— 判斷集合中的元素是否都符合條件
- none —— 判斷集合中是否都不符合條件，是則回傳true
- count —— 查詢集合中符合條件的元素個數
- reduce —— 從第一項到最後一項累計

過濾類

- filter —— 過濾掉所有滿足條件的元素
- filterNot —— 過濾所有不符合條件的元素
- filterNotNull —— 過濾NULL
- take —— 傳回前n 個元素

轉換類別

- map —— 轉換成另一個集合（與上面我們實現的convert 方法作用一樣）；
- mapIndexed —— 除了轉換成另一個集合，還可以拿到Index(下標);
- mapNotNull —— 執行轉換前過濾掉為NULL 的元素
- flatMap —— 自訂邏輯合併兩個集合；
- groupBy —— 依照某個條件分組，返回Map；

排序類

- reversed —— 反序
- sorted —— 升序
- sortedBy —— 自訂排序
- SortedDescending —— 降序

建立集合

不可變更集合 immutable

```
import java.util.*
import java.text.SimpleDateFormat
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */
fun main() {
    val list = listOf(1,2,3, null)
    // 不為 null List
    val notNullList = listOfNotNull(null,1,2,3,null)
    val emptyList = emptyList<Int>()
    val map = mapOf("foo" to "FOO", "bar" to "BAR", "bar" to "BB")
    val set = setOf(4,5,6,6)
    println(list)    //  [1, 2, 3, null]
    println(notNullList)  // [1, 2, 3]
    println(emptyList)  //  []
    println(map)   //  {foo=FOO, bar=BB}
    println(set)   //  [4, 5, 6]
}
```

可變更集合 mutable

```
import java.util.*
import java.text.SimpleDateFormat
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */
fun main() {
  val list = mutableListOf(1,2,3,3)
  val map = mutableMapOf("foo" to "FOO", "bar" to "BAR", "bar" to "BB")
  val set = mutableSetOf(1,2,3,3)
  println(list) // [1, 2, 3, 3]
  println(map) // {foo=FOO, bar=BB}
  println(set) // [1, 2, 3]
}
```

addAll(), list 相加

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {
 val list1 = mutableListOf(1,2,3,4)
    val list2 = mutableListOf(5,6,7,8)
   println(list1.addAll(list2)) // true (注意當下有返回值)
    println(list1)     // [1, 2, 3, 4, 5, 6, 7, 8

}
```

remove()，list 刪除

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {

    val numberList = mutableListOf(1,2,3)
```

```
    numberList.remove(2)

    println(numberList)  // [1, 3]
}
```

索引

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {
 val list = listOf(1,2,3)
    val indices: IntRange = list.indices
    println(indices)

    for(i in list.indices){
        println(list[i])
    }
    /*
     0..2
    1
    2
    3
    */
    println(list.first())  // 1
    println(list.last()) // 3
    println(list.lastIndex) // 2 = ( size - 1 )
    println(list.size)  // 3
}
```

Stream 與 kotlin 比較

Stream.allMatch()	可迭代.all()、Map.all()
Stream.anyMatch()	Iterable.any()、Map.any()
Stream.count()	Iterable.count()、Map.count()
Stream.distinct()	可迭代.distinct()
Stream.filter()	可迭代.filter(), Map.filter()
Stream.findFirst()	Iterable.first(), Iterable.firstOrNull()
Stream.flatMap()	Iterable.flatMap(), Map.flatMap()
Stream.forEach()	Iterable.forEach()、Map.forEach()
Stream.limit()	iterable.take()
Stream.map()	Iterable.map(), Map.map()
Stream.max()	Iterable.max()、Map.maxBy()
Stream.min()	Iterable.min()、Map.minBy()
Stream.noneMatch()	可迭代.none(), Map.none()
Stream.peek()	–
Stream.reduce()	可迭代.reduce()
Stream.sorted()	可迭代.sorted()
Stream.skip()	–
Stream.collect(toList())	Iterable.toList()、Map.toList()
Stream.collect(toMap())	可迭代.toMap()
Stream.collect(toSet())	可迭代.toSet()
Stream.collect(加入())	Iterable.joinToString()
Stream.collect(partitioningBy())	可迭代.partition()
Stream.collect(groupingBy())	可迭代.groupBy()
Stream.collect(減少())	可迭代.fold()
IntStream.sum()	可迭代.sum()
IntStream.average()	可迭代.average()

List 相關

list.map{}

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {
 val nameList = listOf("Sam","John",null,"Mary")
    val resultList = nameList.map{ "name:" + it }
    println(resultList)  // [name:Sam, name:John, name:null, name:Mary]
```

```
    // 過濾null
    val resultList2 = nameList.mapNotNull{ it }
    println(resultList2)  // [Sam, John, Mary]
}
```

list 保留或排除(removeAll(), retainAll())

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {
 val peopleList = mutableListOf(People("sam",18),People("John",23),People("Mary",30))
    // 移除 age > 25
    peopleList.removeAll{ it.age > 25 } // [People(name=sam, age=18), People(name=John, age=23)]
    println(peopleList)

    // 保留 name == sam
    val peopleList2 = mutableListOf(People("sam",18),People("John",23),People("Mary",30))
    peopleList2.retainAll{ it.name == "sam" } // [People(name=sam, age=18)]
    println(peopleList2)
}
```

list 轉 map

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {
 val peopleList = listOf(People("Sam",12),People("Mary",18), People("Sam",13))
    // 指定 key : value
    val map = peopleList.associate { it.name to it }
    // 指定 key , value => 固定為物件
    val map2 = peopleList.associateBy { "name:" + it.name }
    println(map)  // {Sam=People(name=Sam, age=13), Mary=People(name=Mary, age=18)}
    println(map2) // {name:Sam=People(name=Sam, age=13), name:Mary=People(name=Mary, age=18)}

}

data class People(
 val name:String,
    val age: Int
)
```

isEmpty()

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {
 val list1 = listOf(1,2,3,4)
    val list2 = emptyList<Int>()
    var list3:List<Int>? = null

    println(list1.isEmpty())  // flase
    println(list2.isEmpty())  // true
    println(list3?.isEmpty())  // null

    println(list1.orEmpty())  // [1, 2, 3, 4]
    println(list2.orEmpty())  // []
```

```
    println(list3?.orEmpty())  // null
}
```

take(), takeLast() 取出list 數量

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {

   val numberList = mutableListOf(1,2,3,4,5,6)
   // 從前面取兩個
   println(numberList.take(2))  // [1, 2]
   // 從後面取兩個
   println(numberList.takeLast(2)) // [5, 6]

   val noneList :List<Int>? = emptyList()
   println(noneList?.take(1))  // []

    val nullList :List<Int>? = null
   println(nullList?.take(1))  // null
}
```

getOrElse() ,getOrNull()

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {
 val list = listOf(1,2,3,4)

   println(list.getOrElse(1,{9})) // 1
    println(list.getOrElse(5,{9})) // 9

    println(list.getOrNull(1)) // 2
    println(list.getOrNull(5)) // null
}
```

Map 相關

containsKey()，包含 key

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {

    val numberMap = mapOf("one" to 1,"two" to 2)

    println(numberMap.containsKey("one")) // true

    val worldMap = mapOf("one" to "ONE","two" to "TWO")
    println(worldMap.containsValue("one")) // false
    println(worldMap.containsValue("ONE")) // true
}
```

filter{}, filterNot{}, 過濾

filterKeys{}, filterValues{}

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {

    val worldMap = mapOf("one" to "ONE","two" to "TWO", "three" to "THREE")

    println(worldMap.filter {it.key.contains("t")})   // {two=TWO, three=THREE}
    println(worldMap.filterNot {it.key.contains("t")})  // {one=ONE}

    println(worldMap.filterKeys {key -> key.contains("o")})   // {one=ONE, two=TWO}
    println(worldMap.filterValues {value -> value.contains("T")}) // {two=TWO, three=THREE}

}
```

forEach{}

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {

    val map = mapOf("one" to "ONE","two" to "TWO", "three" to "THREE")

    for( (key, value) in map ){
        println("key:${key}, value:${value}")
    }
    // key=one, value=ONE
    // key=two, value=TWO
    // packagekey=three, value=THREE

    map.forEach{
        println("key:${it.key},value:${it.value}")
    }
    // key=one, value=ONE
    // key=two, value=TWO
    // packagekey=three, value=THREE
}
```

getOrElse()

isEmpty(), isNotEmpty()

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {

    val map = mapOf("one" to "ONE","two" to "TWO", "three" to "THREE")

   println(map.getOrElse("one",{"Default"}))  // ONE
    println(map.getOrElse("four",{"Default"}))  // Default
```

```
    val map2 = mapOf<String,String>()
    val map3 :MutableMap<String,String>? = null

    println(map.isEmpty())   // false
    println(map.isNotEmpty())  // true
    println(map.orEmpty())   // {one=ONE, two=TWO, three=THREE}

    println(map2.isEmpty())   // true
    println(map2.isNotEmpty())  // false
    println(map2.orEmpty())   // {}

    println(map3?.isEmpty())  // null
    println(map3?.isNotEmpty())  // null
    println(map3?.orEmpty())  // null

}
```

count(), 數量, 有條件的數量

```
import java.util.*
import java.text.SimpleDateFormat

fun main() {

    val worldMap = mapOf("one" to "ONE","two" to "TWO", "three" to "THREE")
    println(worldMap.count()) // 3
    println(worldMap.count{ it.key.contains("t")}) // 2

}
```

修訂版本 #3
★由 treeman 建立於 24 ﷺ 2024 10:27:04
由 treeman 更新於 26 ﷺ 2024 11:45:02