Scala Option(选项) Scala 类和对象

# Scala Iterator（迭代器）

Scala 集合

Scala Iterator（迭代器）不是一个集合，它是一种用于访问集合的方法。

迭代器 it 的两个基本操作是 **next** 和 **hasNext**。

调用 **it.next()** 会返回迭代器的下一个元素，并且更新迭代器的状态。

调用 **it.hasNext()** 用于检测集合中是否还有元素。

让迭代器 it 逐个返回所有元素最简单的方法是使用 while 循环：

```
object Test {
   def main(args: Array[String]) {
      val it = Iterator("Baidu", "Google", "Runoob", "Taobao")

      while (it.hasNext){
         println(it.next())
      }
   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
Baidu
Google
Runoob
Taobao
```

## 查找最大与最小元素

你可以使用 **it.min** 和 **it.max** 方法从迭代器中查找最大与最小元素，实例如下:

```
object Test {
   def main(args: Array[String]) {
      val ita = Iterator(20,40,2,50,69, 90)
      val itb = Iterator(20,40,2,50,69, 90)

      println("最大元素是: " + ita.max )
      println("最小元素是: " + itb.min )
```

```
    }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
最大元素是：90
最小元素是：2
```

## 获取迭代器的长度

你可以使用 **it.size** 或 **it.length** 方法来查看迭代器中的元素个数。实例如下：

```
object Test {
   def main(args: Array[String]) {
      val ita = Iterator(20,40,2,50,69, 90)
      val itb = Iterator(20,40,2,50,69, 90)

      println("ita.size 的值: " + ita.size )
      println("itb.length 的值: " + itb.length )

   }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
ita.size 的值: 6
itb.length 的值: 6
```

## Scala Iterator 常用方法

下表列出了 Scala Iterator 常用的方法：

| 序号 | 方法及描述 |
|---|---|
| 1 | **def hasNext: Boolean**<br>如果还有可返回的元素，返回true。 |
| 2 | **def next(): A**<br>返回迭代器的下一个元素，并且更新迭代器的状态 |
| 3 | **def ++(that: => Iterator[A]): Iterator[A]** |

| | |
|---|---|
| | 合并两个迭代器 |
| 4 | **def ++[B >: A](that :=> GenTraversableOnce[B]): Iterator[B]**<br>合并两个迭代器 |
| 5 | **def addString(b: StringBuilder): StringBuilder**<br>添加一个字符串到 StringBuilder b |
| 6 | **def addString(b: StringBuilder, sep: String): StringBuilder**<br>添加一个字符串到 StringBuilder b，并指定分隔符 |
| 7 | **def buffered: BufferedIterator[A]**<br>迭代器都转换成 BufferedIterator |
| 8 | **def contains(elem: Any): Boolean**<br>检测迭代器中是否包含指定元素 |
| 9 | **def copyToArray(xs: Array[A], start: Int, len: Int): Unit**<br>将迭代器中选定的值传给数组 |
| 10 | **def count(p: (A) => Boolean): Int**<br>返回迭代器元素中满足条件p的元素总数。 |
| 11 | **def drop(n: Int): Iterator[A]**<br>返回丢弃前n个元素新集合 |
| 12 | **def dropWhile(p: (A) => Boolean): Iterator[A]**<br>从左向右丢弃元素，直到条件p不成立 |
| 13 | **def duplicate: (Iterator[A], Iterator[A])**<br>生成两个能分别返回迭代器所有元素的迭代器。 |
| 14 | **def exists(p: (A) => Boolean): Boolean**<br>返回一个布尔值，指明迭代器元素中是否存在满足p的元素。 |
| 15 | **def filter(p: (A) => Boolean): Iterator[A]**<br>返回一个新迭代器 ，指向迭代器元素中所有满足条件p的元素。 |
| 16 | **def filterNot(p: (A) => Boolean): Iterator[A]**<br>返回一个迭代器，指向迭代器元素中不满足条件p的元素。 |
| 17 | **def find(p: (A) => Boolean): Option[A]** |

| | |
|---|---|
| | 返回第一个满足p的元素或None。注意：如果找到满足条件的元素，迭代器会被置于该元素之后；如果没有找到，会被置于终点。 |
| 18 | **def flatMap[B](f: (A) => GenTraversableOnce[B]): Iterator[B]**<br>针对迭代器的序列中的每个元素应用函数f，并返回指向结果序列的迭代器。 |
| 19 | **def forall(p: (A) => Boolean): Boolean**<br>返回一个布尔值，指明 it 所指元素是否都满足p。 |
| 20 | **def foreach(f: (A) => Unit): Unit**<br>在迭代器返回的每个元素上执行指定的程序 f |
| 21 | **def hasDefiniteSize: Boolean**<br>如果迭代器的元素个数有限则返回true（缺省等同于isEmpty） |
| 22 | **def indexOf(elem: B): Int**<br>返回迭代器的元素中index等于x的第一个元素。注意：迭代器会越过这个元素。 |
| 23 | **def indexWhere(p: (A) => Boolean): Int**<br>返回迭代器的元素中下标满足条件p的元素。注意：迭代器会越过这个元素。 |
| 24 | **def isEmpty: Boolean**<br>检查it是否为空, 为空返回 true，否则返回false（与hasNext相反）。 |
| 25 | **def isTraversableAgain: Boolean**<br>Tests whether this Iterator can be repeatedly traversed. |
| 26 | **def length: Int**<br>返回迭代器元素的数量。 |
| 27 | **def map[B](f: (A) => B): Iterator[B]**<br>将 it 中的每个元素传入函数 f 后的结果生成新的迭代器。 |
| 28 | **def max: A**<br>返回迭代器迭代器元素中最大的元素。 |
| 29 | **def min: A**<br>返回迭代器迭代器元素中最小的元素。 |
| 30 | **def mkString: String**<br>将迭代器所有元素转换成字符串。 |
| 31 | **def mkString(sep: String): String** |

| | |
|---|---|
| | 将迭代器所有元素转换成字符串，并指定分隔符。 |
| 32 | **def nonEmpty: Boolean**<br>检查容器中是否包含元素（相当于 hasNext）。 |
| 33 | **def padTo(len: Int, elem: A): Iterator[A]**<br>首先返回迭代器所有元素，追加拷贝 elem 直到长度达到 len。 |
| 34 | **def patch(from: Int, patchElems: Iterator[B], replaced: Int): Iterator[B]**<br>返回一个新迭代器，其中自第 from 个元素开始的 replaced 个元素被迭代器所指元素替换。 |
| 35 | **def product: A**<br>返回迭代器所指数值型元素的积。 |
| 36 | **def sameElements(that: Iterator[_]): Boolean**<br>判断迭代器和指定的迭代器参数是否依次返回相同元素 |
| 37 | **def seq: Iterator[A]**<br>返回集合的系列视图 |
| 38 | **def size: Int**<br>返回迭代器的元素数量 |
| 39 | **def slice(from: Int, until: Int): Iterator[A]**<br>返回一个新的迭代器，指向迭代器所指向的序列中从开始于第 from 个元素、结束于第 until 个元素的片段。 |
| 40 | **def sum: A**<br>返回迭代器所指数值型元素的和 |
| 41 | **def take(n: Int): Iterator[A]**<br>返回前 n 个元素的新迭代器。 |
| 42 | **def toArray: Array[A]**<br>将迭代器指向的所有元素归入数组并返回。 |
| 43 | **def toBuffer: Buffer[B]**<br>将迭代器指向的所有元素拷贝至缓冲区 Buffer。 |
| 44 | **def toIterable: Iterable[A]**<br>Returns an Iterable containing all elements of this traversable or iterator. This will not terminate for infinite iterators. |
| 45 | **def toIterator: Iterator[A]**<br>把迭代器的所有元素归入一个Iterator容器并返回。 |

| | |
|---|---|
| 46 | **def toList: List[A]**<br>把迭代器的所有元素归入列表并返回 |
| 47 | **def toMap[T, U]: Map[T, U]**<br>将迭代器的所有键值对归入一个Map并返回。 |
| 48 | **def toSeq: Seq[A]**<br>将代器的所有元素归入一个Seq容器并返回。 |
| 49 | **def toString(): String**<br>将迭代器转换为字符串 |
| 50 | **def zip[B](that: Iterator[B]): Iterator[(A, B)**<br>返回一个新迭代器，指向分别由迭代器和指定的迭代器 that 元素一一对应而成的二元组序列 |

更多方法可以参考 API文档

Scala 集合

Scala Option(选项) Scala 类和对象

点我分享笔记