链滴

【双语】Java9 新特性：不可变集合 Immutable Collections

作者：Polly

原文链接：https://ld246.com/article/1517902107418

来源网站：链滴

摘要： Java 9 支持了不可变的List,Set和Map集合，这里就介绍一下这几个不可变集合的使用方法，如何来简单的得到他们的交集和并集。

英文原文链接：https://www.javaspecialists.eu/archive/Issue248.html

Exciting news. We now have "immutable collections" in Java 9.
Just like Scala (not really).

这真是一个令人兴奋的消息，Java 9 像Scala一样支持了不可变集合.

For example, if we want to create lists, we can do this:

举个例子，如果我们想创建一个不可变list,我们可以像如下这么做：

```
List<Integer> list0 = List.of();  // List0
List<Integer> list1 = List.of(42);  // List1
List<Integer> list2 = List.of(42, 57);  // List2
List<Integer> list3 = List.of(42, 57, 1);  // ListN
List<Integer> list4 = List.of(42, 57, 1, 2);  // ListN
```

Someone pointed out that the lists returned by Arrays.asList() were also immutable. They are ot. Whilst you cannot add and remove elements, you can still set individual elements of the lis .

有人说数组用Arrays.asList()转换成list时不也是不可变集合吗？答案是否定的,list依然可以添加和删元素。

We can create sets in a similar way, like so:

我们可以用类似的方法创建Set集合：

```
Set<Integer> set0 = Set.of();  // List0
Set<Integer> set1 = Set.of(42);  // List1
Set<Integer> set2 = Set.of(42, 57);  // List2
Set<Integer> set3 = Set.of(42, 57, 1);  // ListN
Set<Integer> set4 = Set.of(42, 57, 1, 2);  // ListN
```

This begs the question: How can we do set operations? Let's take these sets:

这就引出了一个问题：我们怎么操作这些集合？先让我们创建两个Set不可变集合：

```
Set<Integer> one = Set.of(1, 2, 3);
Set<Integer> two = Set.of(3, 4, 5);
```

To create a union of these is a bit awkward, since there is no easy way to create an immutable Set from an ordinary Set. We could create a temporary Set and then the union from that, like o:

挺尴尬的我们没有一个简单的方法来实现得到两个不可变集合并集的效果，我们可以创建一个临时集来达到目的：

```
Set<Integer> temp = new HashSet<>(one);
temp.addAll(two);
Set<Integer> union = Set.of(temp.toArray(new Integer[0]));
System.out.println("union = " + union);
```

I do not like temporary variables, so we could wrap this in a facade somewhere. To do the inte section is similar:

我并不是很喜欢临时变量，或许我们可以写一个方法来封装这个操作，当然获得交集的方式也类似：

```
Set<Integer> temp = new HashSet<>(one);
temp.retainAll(two);
Set<Integer> intersection = Set.of(temp.toArray(new Integer[0]));
System.out.println("intersection = " + intersection);
```

We can also create Maps with Map.of():

我们创建Map不可变集合也是类似的方法：

```
Map<String, Integer> map0 = Map.of();
Map<String, Integer> map1 = Map.of("one", 1);
Map<String, Integer> map2 = Map.of("one", 1, "two", 2);
```

Map.of() works with key and value pairs up to 10 entries. Beyond that, we need to pass in a va -args of Entry instances and use Map.ofEntries(Entry...)

Map.of()最多只能放进10个键值对,如果想放入更多的键值对,我们需要用到传入实体参数的方式来创 不可变集合：

```
Map<String, Integer> mapN = Map.ofEntries(Map.entry("one", 1),
    Map.entry("two", 2),
    Map.entry("three", 3),
    Map.entry("four", 4),
    Map.entry("five", 5));
```

It is, of course, not nearly as convenient as Kotlin/Swift/Scala/Groovy/etc. It's a very small addi ion to Java which you get when you go over to Java 9.

当然了，java的不可变集合使用上还是不如Kotlin/Switf/Scala/Groovy/等等那么简单，他只是一个Ja a9所有新功能中的一小部分。

Kind regards from Crete

Heinz

英文原文链接：https://www.javaspecialists.eu/archive/Issue248.html