十个惊人的Scala集合操作函数

当我操作 Scala 集合时，我一般会进行两类操作：转换操作（transformation ）和行动操作（actions）（有些人喜欢叫他为聚合操作）。第一种操作类型将集合转换为另一个集合，第二种操作类型返回某些类型的值。

本文我将集中介绍几个日常工作必备的 Scala
集合函数，如转换函数和聚合函数。文章最后，我会展示如何结合这些函数以解决具体问题。

最大值和最小值

我们先从动作函数（action function）开始。在序列中查找最大或最小值是一个极常见的需求，较常用于面试问题和算法。还记得 Java 中的代码行吗？如下：

```
int[] arr = {11, 2, 5, 1, 6, 3, 9};

int to = arr.length - 1;
int max = arr[0];

for (int i = 0; i < to; i++) {
   if (max < arr[i+1])
      max = arr[i+1];
}

System.out.println(max);
```

问题：怎么在 List 中找到最大/最小值呢？

Scala 推荐了一个很赞的解决方案：

```
val numbers = Seq(11, 2, 5, 1, 6, 3, 9)

numbers.max //11
numbers.min //1
```

但实际操作的数据更加复杂。下面我们介绍一个更高级的例子，其中包含一个书的序列（查看源代码案例）。

```
case class Book(title: String, pages: Int)

val books = Seq(
  Book("Future of Scala developers", 85),
  Book("Parallel algorithms", 240),
  Book("Object Oriented Programming", 130),
  Book("Mobile Development", 495)
)

//Book(Mobile Development,495)
books.maxBy(book => book.pages)

//Book(Future of Scala developers,85)
books.minBy(book => book.pages)
```

如上所示，minBy &
maxBy方法解决了复杂数据的问题。你只需选择决定数据最大或最小的属性。

Filter

你过滤过集合吗？比如，筛选价格大于10美元的条目，或挑选年龄在24岁以下员工等，所有这些操作属于过滤。

让我们举例说明：过滤一个数字 List，只获取奇数的元素。

```
val numbers = Seq(1,2,3,4,5,6,7,8,9,10)

numbers.filter(n => n % 2 == 0)
```

然后加大难度，我想获取页数大于120页的书。

```
val books = Seq(
  Book("Future of Scala developers", 85),
  Book("Parallel algorithms", 240),
  Book("Object Oriented Programming", 130),
  Book("Mobile Development", 495)
)

books.filter(book => book.pages >= 120)
```

实际上，过滤是一个转换类型的方法，但是比运用 min和 max方法简单。

还有一个与 filter类似的方法是 filterNot。它的名字就体现了它的作用。如果你还是不了解它的实际用途，你可以在一个示例中，用 filterNot替换 filter 方法。

Flatten O_o

我想大多数朋友都没听说过这个功能。其实它很好理解，我们来举例说明：

```
val abcd = Seq('a', 'b', 'c', 'd')
val efgj = Seq('e', 'f', 'g', 'h')
val ijkl = Seq('i', 'j', 'k', 'l')
val mnop = Seq('m', 'n', 'o', 'p')
val qrst = Seq('q', 'r', 's', 't')
val uvwx = Seq('u', 'v', 'w', 'x')
val yz   = Seq('y', 'z')

val alphabet = Seq(abcd, efgj, ijkl, mnop, qrst, uvwx, yz)

// List(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z)
alphabet.flatten
```

当有一个集合的集合，然后你想对这些集合的所有元素进行操作时，就会用到 flatten。

欧拉图函数（Euler Diagram函数）

不要紧张！接下来的操作大家都熟知：差集、交集和并集。以下示例能很好地解释 Euler Diagram 函数：

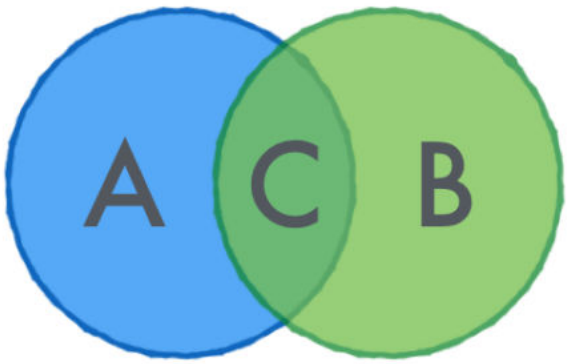
```
val num1 = Seq(1, 2, 3, 4, 5, 6)
val num2 = Seq(4, 5, 6, 7, 8, 9)

//List(1, 2, 3)
num1.diff(num2)

//List(4, 5, 6)
num1.intersect(num2)
```

```
//List(1, 2, 3, 4, 5, 6, 4, 5, 6, 7, 8, 9)
num1.union(num2)
```

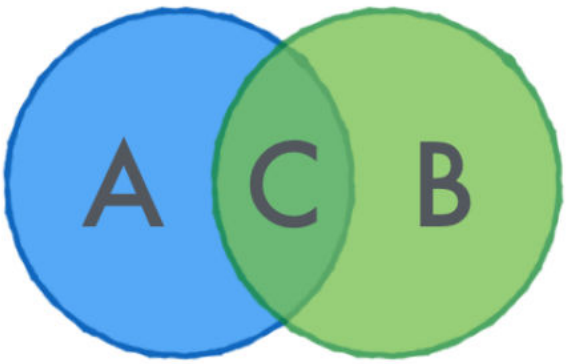
上述示例中的 union保留了重复的元素。如果我们不需要重复怎么办？这时可以使用distinct函数：

```
//List(1, 2, 3, 4, 5, 6, 7, 8, 9)
num1.union(num2).distinct
```

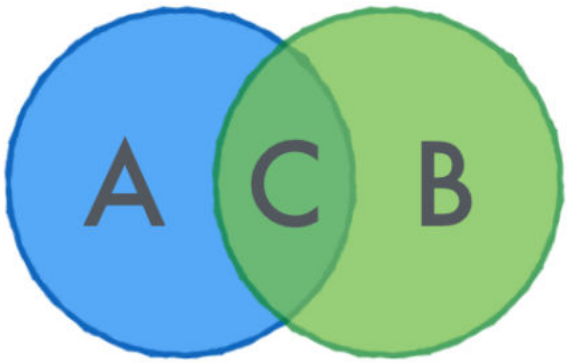
下面是上述功能的图示：



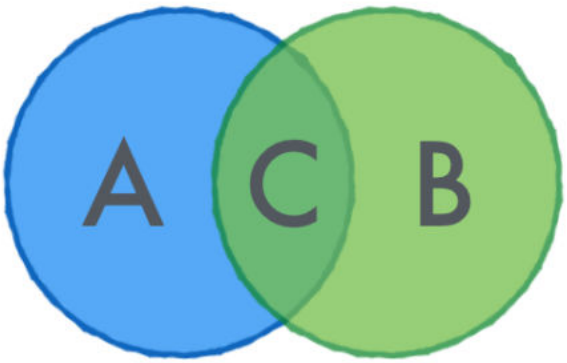
num1.diff(num2) = A



num1.intersect(num2) = C



num2.diff(num1) = B



num1.union(num2)
.distinct = A+C+B

map列表元素

map 是 Scala 集合最常用的一个函数。它的功能十分强大：

```
val numbers = Seq(1,2,3,4,5,6)

//List(2, 4, 6, 8, 10, 12)
numbers.map(n => n * 2)

val chars = Seq('a', 'b', 'c', 'd')

//List(A, B, C, D)
chars.map(ch => ch.toUpper)
```

map 函数的逻辑是遍历集合中的元素并对每个元素调用函数。你也可以不调用任何函数，保持返回元素本身，但这样 map 无法发挥作用，因为你在映射过后得到的是同样的集合。

flatMap

我很难具体说明 flatMap 的使用场合，因为很多不同的情况下都会用到
flatMap。如果大家仔细观察，就会发现 flatMap 是由下列这两个函数组成的：map & flatten

现在，假设我们想知道字母表中的大写字母和小写字母的排列情况：

```
val abcd = Seq('a', 'b', 'c', 'd')

//List(A, a, B, b, C, c, D, d)
abcd.flatMap(ch => List(ch.toUpper, ch))
```

因为这篇文章是关于集合功能的介绍，所以此处略过 Future 和 Option 的示例。

对整个集合进行条件检查

有一个场景大家都知道，即确保集合中所有元素都要符合某些要求，如果有哪怕一个元素不符合条件，就需要进行一些处理：

```
val numbers = Seq(3, 7, 2, 9, 6, 5, 1, 4, 2)

//ture
```

```
numbers.forall(n => n < 10)

//false
numbers.forall(n => n > 5)
```

而 forall 函数就是为处理这类需求而创建的。

对集合进行分组

你是否尝试过将一个集合按一定的规则拆分成两个新的集合？比如，我们把某个集合拆分成偶数集和奇数集，partition 函数可以帮我们做到这一点：

```
val numbers = Seq(3, 7, 2, 9, 6, 5, 1, 4, 2)

//(List(2, 6, 4, 2), List(3, 7, 9, 5, 1))
numbers.partition(n => n % 2 == 0)
```

Fold?

另一个流行的操作是 fold。 在 Scala 的上下文中，通常可以考虑 foldLeft 和 foldRight。他们是从不同的方面做同样的工作：

```
val numbers = Seq(1, 2, 3, 4, 5)

//15
numbers.foldLeft(0)((res, n) => res + n)
```

在第一对括号中，我们放一个起始值。
在第二对括号中，我们定义需要对数字序列的每个元素执行的操作。 第一步，n = 0，然后它根据序列元素变化。

另一个关于 foldLeft 的例子，计算字符数：

```
val words = Seq("apple", "dog", "table")

//13
```

```
words.foldLeft(0)((resultLength, word) => resultLength + word.length)
```

您最喜欢的函数

经过了上面一系列的列举，从Scala集合找到你最喜欢的函数是很酷的（cool）。请大家在评论中写下它，并提供其使用的例子。

最近我通过了一个编译测试，任务的内容是：给你一个String
S，你需要找到包含大写和小写字符，但不包含数字的最长子字符串。
比如: dP4knqw1QAp
答案: QAp

那么我们如何使用Scala集合函数来解决这个问题呢：

```
def theLongest(s: String): String = {
  s.split ("[0-9]")
    .filter (_.exists (ch => ch.isUpper))
    .filter (_.exists (ch => ch.isLower))
    .maxBy (_.length)
}
```

上面的函数解决了这个问题。如果输入字符串不包含任何合适的子字符串，将会抛出
UnsupportedOperationException。

总结

Scala具有令人难以置信的强大的集合API，你可以利用它做很多的事情。
此外，相同的事情可以以不同的方式进行，例如： 上面的欧拉函数例子。
Scala的API是很丰富的，我们需要很多时间和练习来学习它。

原文原文：10 amazing scala collection functions