Broadview®
www.broadview.com.cn

Scala集合技术手册

晁岳攀 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Scala集合技术手册

晁岳攀 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

《Scala 集合技术手册》是第一本全面介绍 Scala 集合框架的图书，它基于最新的 Scala 2.11 编写，深入全面地介绍了 Scala 集合框架的集合类和方法，通过图例、代码示例、表格等多种方式多方位地介绍集合类的方法和实现，并且对相关的类型的性能进行分析和比较，总结了各个集合类的特点，帮助读者快速地掌握 Scala 集合框架，并且可以作为日常 Scala 开发的参考书。

本书适合架构师、软件开发工程师、测试人员以及其他对 Scala 集合感兴趣的相关人员阅读。

图书在版编目（CIP）数据

Scala 集合技术手册 / 晁岳攀著 . —北京：电子工业出版社，2016.6

ISBN 978-7-121-28776-3

Ⅰ . ① S… Ⅱ . ①晁… Ⅲ . ① JAVA 语言－程序设计 Ⅳ . ① TP312

中国版本图书馆 CIP 数据核字 (2016) 第 099005 号

策划编辑：张春雨
责任编辑：安　娜
印　　刷：北京中新伟业印刷有限公司
装　　订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：12　　字数：242 千字
版　　次：2016 年 6 月第 1 版
印　　次：2016 年 6 月第 1 次印刷
定　　价：58.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819　faq@phei.com.cn。

推荐序

非常高兴国内有第一本用中文原创的 Scala 书籍。Scala 俨然成为了大数据领域的明星语言，它强大的抽象和并发能力，以及高效的集合操作让它抓住了大数据的这波浪潮。

提起 Scala 很多人可能会先想到多范式、高并发等特点，其实在集合库的设计上 Scala 也是非常大胆进取的。比如引入了 Tuple、不可变集合，对很多集合内置了 map/reduce/filter/fold 等操作，支持模式匹配，等等，让数据处理得到了极大的简化。

另外，Scala 在设计上对 Java 的一些“不一致性”或“直观性”做了修正，比如用 Any 类型统一了 Java 里的引用类型和基础类型，用统一的 Int 替代了 Java 里的 Integer 和 int 两种类型，并针对这种值类型 (value type) 的运行时 Boxing/Unboxing 做了优化。还有 Java 的数组类型在设计上出于计算的方便被设计为了天然支持协变特性，这在 Scala 2.X 版本得到了修正，去掉了协变。对于相等性的比较让直观的“==”等价于“equals”。

Scala 里的不可变集合主要从函数式语言的集合库里演化而来，一方面在并发情况下这些数据避免了竞争所需要的锁的开销，另一方面在底层则通过共享数据降低了集合在更新时生成新对象的成本。配合惰性计算 (lazy evaluation) 可以发挥更大的威力。当然 Scala 也不反对使用可变集合，在追求性能的地方你可以根据自己的具体情况做出选择。

在学习 Scala 的集合库时，建议深入地了解一些“关键”集合的实现，比如 List，它跟 Java 里的 List 有着完全不同的设计，深入体会这个源自 Lisp 的递归结构的数据类型所表达的含义，对学习 Scala 有非常大的帮助。

这本书非常务实，提供了很多例子，由浅入深地带领你了解 Scala 的集合领域。不管是对大数据领域还是对日常工作都有很大的裨益。

王宏江　挖财架构师《Scala 函数式编程》译者

序

在我十六年多的编程生涯中，大部分项目中我都使用 Java 开发。Java 的生态圈非常的庞大而丰富，可以帮助我快速地实现项目的架构和开发。虽然感觉 Java 的语法有很多啰唆的地方，也调研过 Groovy、Python 等编程语言，但还是觉得 Java 比较适合开发工作。但是当我尝试使用 Scala 实现了一个高性能的消息系统的时候，我彻底地喜欢上了 Scala，它简洁的语法、丰富的特性、面向对象和函数式编程的美妙结合都给我留下了深刻的印象。因为 Scala 是基于 JVM 的编程语言，它可以使用 Java 实现的库，充分利用 Java 生态圈中已有的库，而性能上又不会有多少损失，所以我经常使用 Scala 做一些编程的工作。通过阅读 Kafka 等 Scala 实现框架，也促使我更加地坚信 Scala 的简洁高效。

但是，完全掌握 Scala 的特性、熟练使用它的库和编程方式进行开发却不是一件容易的事情。如果有 Java 编程经验，则 Scala 入门会是一件很容易的事情，但是到专家级别却是一个曲折的过程。这是因为，一、Scala 的语法相比其他语言，如 Java、Python、Go 等要复杂很多，要想完全理解和掌握需要较长的时间。二、Scala 本身提供了非常多的库，尤其是占重要部分的集合库，只有熟练掌握它们才能轻松地应用到开发中。

对于第一个问题，现在已经有了多本介绍和学习 Scala 语法的书，尤其是刚刚出版的 Scala 之父 Martin Odersky 编写的《Programming in Scala》第三版，都是很好的学习和深入了解 Scala 语法的图书。通过编程实践以及阅读其他 Scala 开发的项目，也会很快地提高 Scala 语法的掌握程度。

对于第二个问题，尤其是 Scala 集合框架的全面介绍，并没有太好的学习资料，相关的 Scala 书籍中会有一两章关于 Scala 集合的介绍，官方的网站上也有多篇关于 Scala 集合的教程，但是总体来说都是粗略地或者说局部地介绍了Scala集合框架的类和方法，看过之后若有所得，但并不踏实，总是感觉没有全面地掌握 Scala 集合。所以针对这个需求，我动手编写了这本书，主要目的就是编写一本全面、专门、深入地介绍 Scala 集合的书与大家分享。

作为一本专门介绍 Scala 集合的图书，它有一些其他技术图书没有的特色：

- **图文并茂**。本书不仅提供了多幅插图，列出了并发 / 非并发集合类和 Trait 的继承关系，还为最主要的集合的方法提供了直观的图例，可以实现见图识意。

- **归类**。将类似的方法放在一起介绍，比如 map 和 flatmap，scan、scanLeft 和 scanRight，size 和 length 等。
- **对比**。将容易混淆的方法进行对比，剖析它们之间的区别，比如 head、last、init、tail，reduce 和 fold 方法的区别等。
- **代码实例**。通过最简单的代码类和方法的功能。代码简短而功能直接。
- **表格**。通过表格进行汇总和知识整理。
- **全面**。本书详细介绍了 Scala 集合绝大部分的类和方法，对于一些类还介绍了它的历史优化过程和当前的问题。
- **深入**。对于一些典型的集合类，本书还深入介绍了它们的底层实现，通过底层实现了解它们的特性和性能。

我期望能通过多种方式搭建起本书的架构，多方位地帮助读者学习和掌握 Scala 集合库。而且本书还能作为案头的一本常用查找手册，当对 Scala 集合的类或者方法有疑惑时，可以通过翻看本书找到答案。

联系方式

尽管我尽量保持本书内容的准确性，每一个方法和例子都编写测试过，但是因为 Scala 集合框架的复杂和知识点比较多，加之作者水平有限，疏漏和错误在所难免，恳请广大读者批评指正。如果你发现了错误或好的建议，也可以联系我。我的联系方式如下：

邮箱：oldnest@gmail.com

新浪微博：colobu

致谢

特别感谢电子工业出版社的编辑张春雨，没有你的努力和辛苦工作，就不会有本书顺利的出版。没有你的激励，可能本书的内容只能三三两两地散落在我的笔记或者博客中，不能集结出版。

感谢我的老婆和儿子。写书占用了我大部分的业余时间，不能花更多的时间陪伴你们。春节的时候也只能让你们自己回娘家，给我独自的时间写稿和修改，没有你们的理解和支持，我也不能安心地写作。

谢谢阅读本书的朋友，如果没有你们对相关知识的渴求，也就不会有本书的编写。你们对 Scala 的喜爱，对 Scala 集合框架的知识的渴求，就是我写本书最大的动力。

目录

第1章

Scala 集合库简介

Scala 编程语言

Scala 是一门多范式的编程语言，同时支持面向对象和函数式编程风格。它以一种优雅的方式解决现实问题。虽然它是强静态类型的编程语言，但是它的强大的类型推断能力，使其看起来就像一个动态编程语言一样。

就像 Groovy、Clojure、JRuby 等基于 JVM 虚拟机的编程语言一样，Scala 的源代码最终被编译成 Java 字节码，所以由 Scala 编译好的程序可以无缝地运行在 JVM 虚拟机之上，甚至你还可以使用 Scala 开发 Android 程序，这也意味着 Scala 和 Java 的类库可以互相调用。但是与其他基于 JVM 的编程语言相比，Scala 优秀的地方在于，除了和 Java 一样基于类似 C 语言的面向对象的编程风格外，Scala 还拥有类似 Scheme、Standard ML 和 Haskell 一样的函数式编程的风格，比如柯里化（Currying）、类型推断、不可变类型，延迟求值（Lazy Evaluation）、模式匹配等。它还提供了其他的高级特性，如特质混入、协变（Convariance）和逆变（Contravariance）、高阶函数和类型（Higher-order）、lambda 等，其他特性如操作符重载、默认参数、命名参数、原始字符串（Raw string）、字符串插值（String Interpolation）、无 checked exception 等。

总之，Scala 编程语言的好处如下：

- **面向对象编程**：在 Scala 编程语言中，一切值都是对象，每一个操作都是方法调用，不像 Java 中还有 int、boolean 等这样的基本类型。传统的编程模式也适用于 Scala，比如单例可以通过 `Object` 实现，模式匹配可以实现 `visitor` 模式。通过隐式类，Scala 可以为其他已有的类型增加新的方法，即使这些类来自 Java（C# 很早也添加了这个功能）。隐式方法在 Scala 库和其他项目（如 Kafka）都得到了广泛的应用，极大地扩展了既有类的功能，同时还精简了代码。
- **函数式编程**：Scala 还是一种函数式编程语言，而且语法相当方便。这种将面向对象和函数式编程混合的方式真是一种天才的设计。它包括了很多你想要的函数编程的功能，而对于熟悉面向对象的开发者而言，它同样没有违和感。与其他的函数式语言不同，

Scala允许你渐进地使用函数式风格。在你不熟悉函数式编程的情况下，你可以把它看成一门没有分号分隔的Java语言。随着对Scala语言的熟悉，你会渐渐喜欢上其简洁的函数式风格。

- **无缝的Java互操作**：Scala代码被编译成Java字节码，所以理论上来说，你可以随意地在项目中混用这两种语言。比如在Maven或SBT管理的项目中，它们的代码结构都是类似的，而且代码之间可以互相引用。当然，Java使用Scala生成的类型并不总是非常的方便，比如Scala中包含实现方法的特质Trait），并不能直接编译成Java的接口。所以，有时候Scala项目也会提供专门为Java使用的API，如Apache Spark项目。
- **函数也是对象**：在Scala中，函数是第一类（first-class）的，一个函数也是一个对象，有确定的类型。

  第一类的类型要满足以下特征：

  1. 可以被存放在变量和数据结构中。
  2. 可以当作函数参数。
  3. 可以当作函数的返回值。
  4. 可以在运行期间被创建。

- **并发编程**：Scala可以使用Java的并发编程模型，比如concurrent库、线程、Fork-Join框架。Scala本身还为异步编程提供了简便的语法，比如`future`、`promise`等，还有`Actor`异步框架也可以使用。
- **趣味性**：使用Scala编程是一件享受的事情。Scala语言提供了如此丰富的特性，而且它迭代快速且不古板迂腐。

Scala还在大数据领域得到广泛的应用，如Spark、Kafka等都是使用Scala开发的。一些著名的公司如Twitter、Foursquare、Coursera、LinkedIn、Verizon等也在广泛地使用Scala。

那么，是谁创建的Scala编程语言呢？是**Martin Odersky**，瑞士洛桑联邦理工学院（EPFL）的教授，德国人，编程语言和代码分析的专家。他的另外两个知名的工作是Generic Java和javac（Oracle/Sun Java编译器）。基于Funnel的工作，他于2001年开始设计Scala，Java平台的Scala于2003年底至2004年初发布。该语言的第二个版本，v2.0，发布于2006年3月。当前的版本为2.11.x。2011年，他和Akka框架的作者Jonas Bonér一起成立了TypeSafe公司，致力于Scala的推广。

Scala集合

凡是学过一点计算机的人大概都知道“算法 + 数据结构 = 程序”这个著名的公式。提出这一公式并以此作为其一本专著书名的瑞士计算机科学家尼克劳斯·威茨（Niklaus Wirth）于1984年获得了图灵奖。数据结构，是抽象的表示数据的方式；算法，则是计算的一系列有

效、通用的步骤。算法与数据结构是程序设计中相辅相成的两个方面。因此数据结构是编程中很重要的一个方面。很多编程语言及社区都提供了数据结构的对应编程库，并称之为集合库（Collection library）。

Java 同样提供了一套通用数据结构的接口和类，也称之为集合库，可以有效地存储和处理数据。

所谓集合，就是代表一组对象的数据结构，数据对象也被称为集合的元素。有些集合允许存储重复数据，有些则不可以。有些集合是有序的，有些则是乱序的。

Java 集合库所提供的丰富的集合类型，可以让我们在数据结构的设计上节省精力，专注于算法的设计上，快速地开发应用程序。Scala 扩展了 Java 的集合库，或者说重建了一个功能更强大的集合库。它为集合库提供了几十个功能强大的方法。这些方法、功能是如此的强大，组合起来简直就是一把编程的瑞士军刀。而且它还提供了非常多的特质和类，以及伴生对象类型，除 Java 的并发集合库目前还无法替代外，它完全可以独立胜任应用程序中的集合类型。

当前的 Scala 集合库是基于 Scala 2.8 重新设计的框架，它提供了一个通用的、统一的、全面的集合库。即使 Scala 到了现在的 2.11 版本，以及将要发布的 2.12 版本，还是以 2.8 版本为基础，做了一些修补、提升和补充。

Scala 集合库有以下优点：

- **易于使用**：使用集合库提供的 20 到 50 个左右的方法，灵活地组合运用，就可以解决大部分的集合问题。你不必再陷入复杂的循环和递归的一团乱麻中。可变和不可变集合以及无副作用的操作意味着，你无须担心新的数据会破坏已有的集合，迭代器和集合的更新冲突也可以避免。
- **简洁**：简单一个单词，比如 `fold`，就可以实现一个或者多个循环操作。你可以使用轻量级的语法实现功能操作和组合操作，感觉上就像代数运算一样。
- **安全**：静态类型和集合的功能特征意味着绝大部分错误都可以在编译时被发现。集合操作很好地进行了测试，而且操作的输入输出都显式地作为函数的参数和返回结果，这样就很容易进行静态类型检查。
- **快速**：集合类型的方法实现的时候都进行了调优，你可以根据需求选择合适的集合类型。Scala 还提供了并行集合，对于大数据而言可以在多核机器上提高效率。而且并行集合的方法和串行集合相同，所以不需要重新学习，代码也不必重写。它们之间还可以互换。
- **统一**：集合利用多个功能 `Trait`，为类型提供了一致的方法集。相似类型的集合拥有同样的一组方法。比如字符串可以看作是字符序列，它支持所有的序列集合方法，数组也是如此。

下面来看一个展示 Scala 集合类优势的例子：

```
val (minors, adults) = people partition (_.age < 18)
```

很明显上面的代码是将一个 people 集合分成两部分。根据年龄，小于 18 岁的为未成年（minors）集合，大于等于 18 岁的为成年人（adults）集合。因为分区方法定义在根集合类型 `TraversableLike` 特质中，所以这行代码可以应用在任意的集合上，包括可以隐式转换的数组类型。结果类型和 people 集合的类型相同。

这段代码与传统的循环处理相比要简洁得多。一旦你通过本书掌握了各种集合和集合的方法，你就会发现代码编写越来越容易，你基本不会再用 `for` 循环来处理集合。而且如果正确选择合适的集合，则操作也会非常快。对于大数据来说，你还可以在多核机器上使用并行集合，以获得更好的性能。

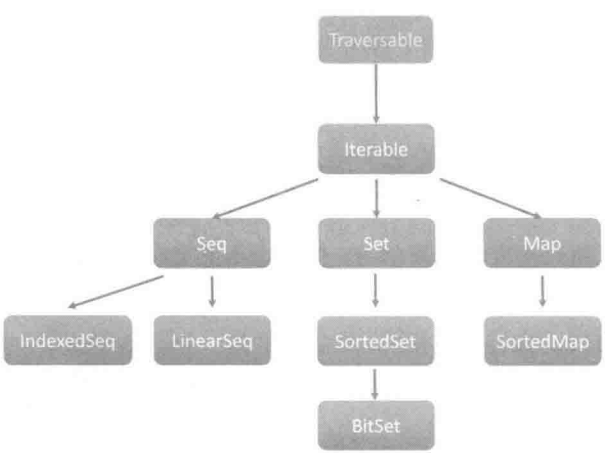
Scala 集合将类、特质及伴生对象 object 根据特性分成几个包。`scala.collection` 包为可变集合和不可变集合定义了通用的集合类型。`scala.collection.immutable` 包定义了不可变集合，而 `scala.collection.mutable` 包定义了可变集合。`scala.collection.parallel` 定义了并行集合，也包括可变集合和不可变集合。`scala.collection.convert` 定义了 Scala 集合和 Java 集合的隐式转换类。`scala.collection.generic` 定义了泛型的工厂类。

整体来讲，一个具体的 Scala 集合会是 `Seq`、`Set` 和 `Map` 特质的一个实现类。也就是说，Scala 集合大致分为三种，江湖上称之为三大门派。不同类别的集合即使所包含的元素相同，它们的相等性比较也是 `false`，比如：

```
Seq(1,2,3) != Set(1,2,3)
```

下面列出了主要的三个包下的特质和类。

package scala.collection 这个包为可变集合和不可变集合定义了共用的特质，如图 1.1 所示。



图例

注意，上图中的箭头方向和 *UML* 类图中的箭头方向相反

图 1.1　package scala.collection

事实上，在 Scala 2.11.x 版本中，包 scala.collection 下还有个 PagedSeq 类，它是 IndexedSeq 的子类，但是几乎没有人用它。它将在 2.12 版本中被弃用。

package scala.collection.immutable 这个包下定义了不可变集合的特质和具体实现类，如图 1.2 所示。

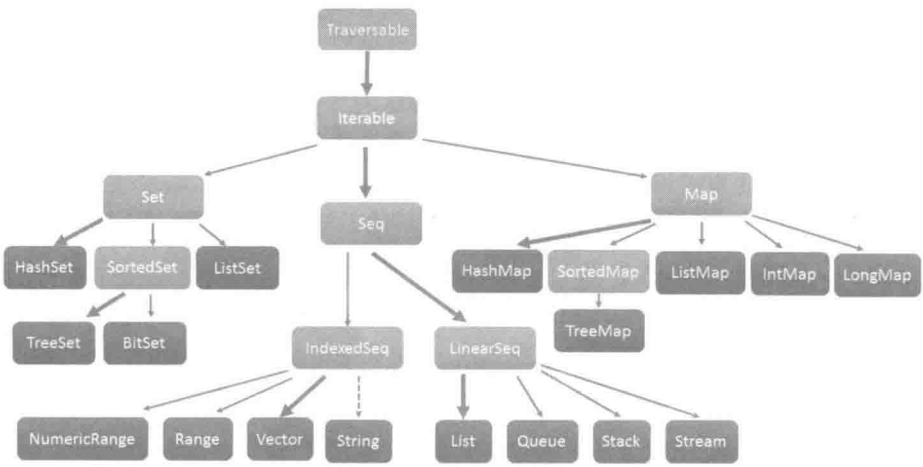


图 1.2 package scala.collection.immutable

package scala.collection.mutable 这个包下定义了可变集合的特质和具体实现类，如图 1.3 所示。

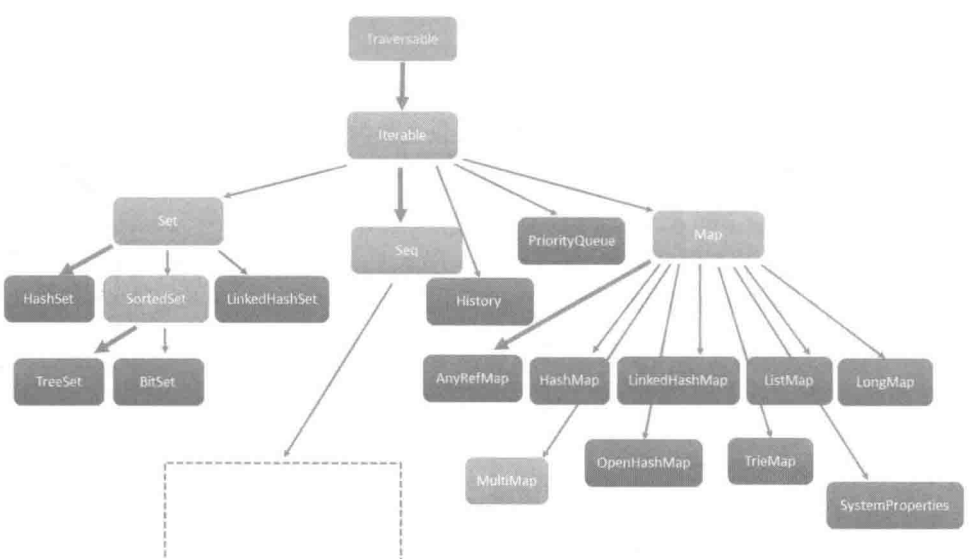


图 1.3 package scala.collection.mutable（Set & Map）