

《Java 8实战》

图书基本信息

书名：《Java 8实战》

13位ISBN编号：9787115419345

出版时间：2016-4-1

作者：厄马(Raoul-Gabriel Urma),弗斯科(Mario Fusco),米克罗夫特(Alan Mycroft)

页数：349

译者：陆明刚,劳佳

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu111.com

《Java 8 实战》

内容概要

本书全面介绍了Java 8 这个里程碑版本的新特性，包括Lambdas、流和函数式编程。有了函数式的编程特性，可以让代码更简洁，同时也能自动化地利用多核硬件。全书分四个部分：基础知识、函数式数据处理、高效Java 8 编程和超越Java 8，清晰明了地向读者展现了一幅Java 与时俱进的现代化画卷。

作者简介

作者简介：

Raoul-Gabriel Urma

剑桥大学计算机科学博士，软件工程师，演讲者，培训师，Cambridge Coding Academy联合创始人、CEO。曾与谷歌、eBay、甲骨文和高盛集团等大公司合作，并参与过多个创业项目。撰写过十余篇经同行审阅的技术文章，并在国际会议上发表过40多篇演讲。

Mario Fusco

Red Hat高级软件工程师，负责JBoss规则引擎Drools的核心开发。拥有丰富的Java开发经验，曾领导媒体公司、金融部门等多个行业的企业级项目开发。对函数式编程和领域特定语言等有浓厚兴趣，并创建了开放源码库lambdaj。

Alan Mycroft

剑桥大学计算机实验室计算学教授，剑桥大学罗宾逊学院研究员，欧洲编程语言和系统协会联合创始人，树莓派基金会联合创始人和理事。发表过大约100篇研究论文，指导过20多篇博士论文。他的研究主要关注编程语言及其语义、优化和实施。他与业界联系紧密，曾于学术休假期间在AT&T实验室和英特尔工作，还创立了Codemist公司，该公司设计了最初的ARM C编译器Norcroft。

译者简介：

陆明刚

毕业于四川大学，目前在EMC中国卓越研发集团任首席工程师，曾任趋势科技中国软件研发中心技术经理，在信息科学和工程领域有十余年的实践和研究经验，拥有多项中国及美国专利。关注JVM性能调优和大数据及其实践，喜欢挖掘技术背后的内幕并乐此不疲。

劳佳

硕士毕业于上海交通大学，现在SAP美国任高级软件支持顾问。业余爱好语言、数学、设计，近年翻译出版了《咨询的奥秘》《卓越程序员密码》等书。

书籍目录

第一部分 基础知识

第1章 为什么要关心Java 8	2
1.1 Java怎么还在变	4
1.1.1 Java在编程语言生态系统中的位置	4
1.1.2 流处理	6
1.1.3 用行为参数化把代码传递给方法	7
1.1.4 并行与共享的可变数据	7
1.1.5 Java需要演变	8
1.2 Java中的函数	8
1.2.1 方法和Lambda作为一等公民	9
1.2.2 传递代码：一个例子	11
1.2.3 从传递方法到Lambda	12
1.3 流	13
1.4 默认方法	17
1.5 来自函数式编程的其他好思想	18
1.6 小结	19
第2章 通过行为参数化传递代码	20
2.1 应对不断变化的需求	21
2.1.1 初试牛刀：筛选绿苹果	21
2.1.2 再展身手：把颜色作为参数	21
2.1.3 第三次尝试：对你能想到的每个属性做筛选	22
2.2 行为参数化	23
2.3 对付啰嗦	27
2.3.1 匿名类	28
2.3.2 第五次尝试：使用匿名类	28
2.3.3 第六次尝试：使用Lambda表达式	30
2.3.4 第七次尝试：将List类型抽象化	31
2.4 真实的例子	31
2.4.1 用Comparator来排序	31
2.4.2 用Runnable执行代码块	32
2.4.3 GUI事件处理	32
2.5 小结	33
第3章 Lambda表达式	34
3.1 Lambda管中窥豹	35
3.2 在哪里以及如何使用Lambda	37
3.2.1 函数式接口	37
3.2.2 函数描述符	39
3.3 把Lambda付诸实践：环绕执行模式	41
3.3.1 第1步记得行为参数化	41
3.3.2 第2步：使用函数式接口来传递行为	42
3.3.3 第3步：执行一个行为	42
3.3.4 第4步：传递Lambda	42
3.4 使用函数式接口	43
3.4.1 Predicate	44
3.4.2 Consumer	44
3.4.3 Function	45
3.5 类型检查、类型推断以及限制	49

3.5.1	类型检查	49
3.5.2	同样的Lambda，不同的函数式接口	50
3.5.3	类型推断	51
3.5.4	使用局部变量	52
3.6	方法引用	53
3.6.1	管中窥豹	53
3.6.2	构造函数引用	55
3.7	Lambda和方法引用实战	57
3.7.1	第1步：传递代码	58
3.7.2	第2步：使用匿名类	58
3.7.3	第3步：使用Lambda表达式	58
3.7.4	第4步：使用方法引用	59
3.8	复合Lambda表达式的有用方法	59
3.8.1	比较器复合	60
3.8.2	谓词复合	60
3.8.3	函数复合	61
3.9	数学中的类似思想	62
3.9.1	积分	62
3.9.2	与Java 8的Lambda联系起来	63
3.10	小结	64
第二部分 函数式数据处理		
第4章 引入流 68		
4.1	流是什么	68
4.2	流简介	72
4.3	流与集合	74
4.3.1	只能遍历一次	75
4.3.2	外部迭代与内部迭代	76
4.4	流操作	78
4.4.1	中间操作	78
4.4.2	终端操作	79
4.4.3	使用流	80
4.5	小结	81
第5章 使用流 82		
5.1	筛选和切片	83
5.1.1	用谓词筛选	83
5.1.2	筛选各异元素	83
5.1.3	截短流	84
5.1.4	跳过元素	85
5.2	映射	86
5.2.1	对流中每一个元素应用函数	86
5.2.2	流的扁平化	87
5.3	查找和匹配	90
5.3.1	检查谓词是否至少匹配一个元素	90
5.3.2	检查谓词是否匹配所有元素	90
5.3.3	查找元素	91
5.3.4	查找第一个元素	92
5.4	归约	92
5.4.1	元素求和	93
5.4.2	最大值和最小值	94

5.5	付诸实践	97
5.5.1	领域：交易员和交易	98
5.5.2	解答	99
5.6	数值流	101
5.6.1	原始类型流特化	101
5.6.2	数值范围	102
5.6.3	数值流应用：勾股数	103
5.7	构建流	105
5.7.1	由值创建流	106
5.7.2	由数组创建流	106
5.7.3	由文件生成流	106
5.7.4	由函数生成流：创建无限流	107
5.8	小结	110
第6章	用流收集数据	111
6.1	收集器简介	112
6.1.1	收集器用作高级归约	112
6.1.2	预定义收集器	113
6.2	归约和汇总	114
6.2.1	查找流中的最大值和最小值	114
6.2.2	汇总	115
6.2.3	连接字符串	116
6.2.4	广义的归约汇总	117
6.3	分组	120
6.3.1	多级分组	121
6.3.2	按子组收集数据	122
6.4	分区	126
6.4.1	分区的优势	126
6.4.2	将数字按质数和非质数分区	128
6.5	收集器接口	129
6.5.1	理解Collector接口声明的方法	130
6.5.2	全部融合到一起	134
6.6	开发你自己的收集器以获得更好的性能	135
6.6.1	仅用质数做除数	136
6.6.2	比较收集器的性能	139
6.7	小结	140
第7章	并行数据处理与性能	141
7.1	并行流	141
7.1.1	将顺序流转换为并行流	142
7.1.2	测量流性能	144
7.1.3	正确使用并行流	147
7.1.4	高效使用并行流	148
7.2	分支/合并框架	149
7.2.1	使用RecursiveTask	149
7.2.2	使用分支/合并框架的最佳做法	153
7.2.3	工作窃取	154
7.3	Splititerator	155
7.3.1	拆分过程	155
7.3.2	实现你自己的Splititerator	157
7.4	小结	162

第三部分 高效Java 8编程

第8章 重构、测试和调试	164
8.1 为改善可读性和灵活性重构代码	164
8.1.1 改善代码的可读性	165
8.1.2 从匿名类到Lambda表达式的转换	165
8.1.3 从Lambda表达式到方法引用的转换	166
8.1.4 从命令式的数据处理切换到Stream	167
8.1.5 增加代码的灵活性	168
8.2 使用Lambda重构面向对象的设计模式	170
8.2.1 策略模式	171
8.2.2 模板方法	172
8.2.3 观察者模式	173
8.2.4 责任链模式	175
8.2.5 工厂模式	177
8.3 测试Lambda表达式	178
8.3.1 测试可见Lambda函数的行为	179
8.3.2 测试使用Lambda的方法的行为	179
8.3.3 将复杂的Lambda表达式分到不同的方法	180
8.3.4 高阶函数的测试	180
8.4 调试	181
8.4.1 查看栈跟踪	181
8.4.2 使用日志调试	183
8.5 小结	184
第9章 默认方法	185
9.1 不断演进的API	187
9.1.1 初始版本的API	188
9.1.2 第二版API	188
9.2 概述默认方法	190
9.3 默认方法的使用模式	192
9.3.1 可选方法	192
9.3.2 行为的多继承	192
9.4 解决冲突的规则	196
9.4.1 解决问题的三条规则	196
9.4.2 选择提供了最具体实现的默认方法的接口	197
9.4.3 冲突及如何显式地消除歧义	198
9.4.4 菱形继承问题	200
9.5 小结	201
第10章 用Optional取代null	202
10.1 如何为缺失的值建模	203
10.1.1 采用防御式检查减少Null-PointerException	203
10.1.2 null带来的种种问题	204
10.1.3 其他语言中null的替代品	205
10.2 Optional类入门	206
10.3 应用Optional的几种模式	207
10.3.1 创建Optional对象	208
10.3.2 使用map从Optional对象中提取和转换值	208
10.3.3 使用flatMap链接Optional对象	209
10.3.4 默认行为及解引用Optional对象	213
10.3.5 两个Optional对象的组合	213

10.3.6	使用filter剔除特定的值	214
10.4	使用Optional的实战示例	216
10.4.1	用Optional封装可能为null的值	216
10.4.2	异常与Optional的对比	217
10.4.3	把所有内容整合起来	218
10.5	小结	219
第11章	CompletableFuture：组合式异步编程	220
11.1	Future接口	222
11.1.1	Future接口的局限性	223
11.1.2	使用CompletableFuture构建异步应用	223
11.2	实现异步API	224
11.2.1	将同步方法转换为异步方法	225
11.2.2	错误处理	227
11.3	让你的代码免受阻塞之苦	228
11.3.1	使用并行流对请求进行并行操作	229
11.3.2	使用CompletableFuture发起异步请求	230
11.3.3	寻找更好的方案	232
11.3.4	使用定制的执行器	233
11.4	对多个异步任务进行流水线操作	234
11.4.1	实现折扣服务	235
11.4.2	使用Discount服务	236
11.4.3	构造同步和异步操作	237
11.4.4	将两个CompletableFuture对象整合起来，无论它们是否存在依赖	239
11.4.5	对Future和CompletableFuture的回顾	241
11.5	响应CompletableFuture的completion事件	242
11.5.1	对最佳价格查询器应用的优化	243
11.5.2	付诸实践	244
11.6	小结	245
第12章	新的日期和时间API	246
12.1	LocalDate、LocalTime、Instant、Duration以及Period	247
12.1.1	使用LocalDate和LocalTime	247
12.1.2	合并日期和时间	248
12.1.3	机器的日期和时间格式	249
12.1.4	定义Duration或Period	249
12.2	操纵、解析和格式化日期	251
12.2.1	使用TemporalAdjuster	253
12.2.2	打印输出及解析日期 - 时间对象	255
12.3	处理不同的时区和历法	256
12.3.1	利用和UTC/格林尼治时间的固定偏差计算时区	257
12.3.2	使用别的日历系统	258
12.4	小结	259
第四部分	超越Java 8	
第13章	函数式的思考	262
13.1	实现和维护系统	262
13.1.1	共享的可变数据	263
13.1.2	声明式编程	264
13.1.3	为什么要采用函数式编程	265
13.2	什么是函数式编程	265
13.2.1	函数式Java编程	266

13.2.2	引用透明性	268
13.2.3	面向对象的编程和函数式编程的对比	268
13.2.4	函数式编程实战	269
13.3	递归和迭代	271
13.4	小结	274
第14章	函数式编程的技巧	275
14.1	无处不在的函数	275
14.1.1	高阶函数	275
14.1.2	科里化	277
14.2	持久化数据结构	278
14.2.1	破坏式更新和函数式更新的比较	279
14.2.2	另一个使用Tree的例子	281
14.2.3	采用函数式的方法	282
14.3	Stream的延迟计算	283
14.3.1	自定义的Stream	283
14.3.2	创建你自己的延迟列表	286
14.4	模式匹配	290
14.4.1	访问者设计模式	291
14.4.2	用模式匹配力挽狂澜	292
14.5	杂项	295
14.5.1	缓存或记忆表	295
14.5.2	“返回同样的对象”意味着什么	296
14.5.3	结合器	296
14.6	小结	297
第15章	面向对象和函数式编程的混合：Java 8和Scala的比较	299
15.1	Scala简介	300
15.1.1	你好，啤酒	300
15.1.2	基础数据结构：List、Set、Map、Tuple、Stream以及Option	302
15.2	函数	306
15.2.1	Scala中的一等函数	307
15.2.2	匿名函数和闭包	307
15.2.3	科里化	309
15.3	类和trait	310
15.3.1	更加简洁的Scala类	310
15.3.2	Scala的trait与Java 8的接口对比	311
15.4	小结	312
第16章	结论以及Java的未来	313
16.1	回顾Java 8的语言特性	313
16.1.1	行为参数化（Lambda 以及方法引用）	314
16.1.2	流	314
16.1.3	CompletableFuture	315
16.1.4	Optional	315
16.1.5	默认方法	316
16.2	Java 的未来	316
16.2.1	集合	316
16.2.2	类型系统的改进	317
16.2.3	模式匹配	318
16.2.4	更加丰富的泛型形式	319
16.2.5	对不变性的更深层支持	321

16.2.6	值类型	322
16.3	写在最后的话	325
附录A	其他语言特性的更新	326
附录B	类库的更新	330
附录C	如何以并发方式在同一个流上执行多种操作	338
附录D	Lambda表达式和JVM 字节码	346

精彩短评

- 1、很适合了解java8的新特性
- 2、每个点都讲得比较细，知道每个api背后都有设计者自己的考量意图
- 3、通俗易懂 读起来比think in java爽的多 - -
- 4、终于读完，很不错，写的很详细。每个知识点都有例子讲解，学到很多Java8的特性，作为函数式编程入门很不错
- 5、粗略过了一遍，后续再读。
- 6、依据摩尔定律每年新增的晶体管数量已经无法使独立CPU核的速度更快了。简单来说，要让你的代码运行得更快，需要你的代码具备并行运算的能力。Java 8中的函数式编程支持主要就是为了支持并行运算能力。
- 7、算是 Java 8 引入的语言特性及 API 的参考吧。直接看 reference 可能更全面。
- 8、从两个角度来读：SCIP的解释版本（给SCIP的程序以组合以及新的命名）和从Java的设计模式和语法到函数式的阅读；
- 9、LAMBDA从小白到入门，深层剖析JAVA语言的设计思想，感悟深刻。
- 10、了解Java8的新特性，这本书是讲解Java8的新特性最全最仔细的了。
- 11、我觉得一般，不够直白。
- 12、读过《Java 8函数式编程》一书，这本书作为补充继续阅读，加深理解。
- 13、单纯作为Java8的新特性来讲，非常详细到位的。使用了很多之前Java7的例子，来讲解Java8要解决什么问题，新手看起来很有感触，但是老手看起来有些冗余。其中lamda表达式讲解的最为详细。推荐值得一看。
- 14、终于读完了，收货很大，内容很有深度，改变上的改变是最深的。
- 15、java8入门不错的书，看完一次基础上可以上手。讲解结合实例，很通俗易懂。
- 16、不止java8函数式,等于吧java的演进也涵盖了...

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:www.tushu111.com