

Java实战（第2版）



[Java实战（第2版）_下载链接1](#)

著者:[英] 拉乌尔-加布里埃尔 · 乌尔玛

出版者:人民邮电出版社

出版时间:2019-11

装帧:平装

isbn:9787115521484

现代Java应用充分利用了微服务、反应式架构以及流式数据等创新设计。现代Java特性，譬如Lambda、流以及大家期待已久的Java模块系统让这些设计的实现极其便利。是时候更新技能工具箱了，只有这样，你才能从容应对迎面而来的种种挑战！

本书通过透彻的示例和通俗的语言讲解了Java语言这些最激动人心的特性，作者注重细节，努力降低了学习难度，节省你宝贵的时间。依照本书边学边练，你可以很快掌握流应用程序接口、Java模块系统等现代Java新特性，再进一步去探寻实现并发的新方法，了解函数式编程如何帮你编写可读性好又容易维护的代码。潜心修炼，你的编程实力必能提高到新的层次。

本书特色：

- 对上一版（《Java 8实战》）做了全新改版
- Java 8、9、10及后续版本新特性介绍
- 流数据处理以及反应式编程
- Java模块系统

作者介绍：

作者简介：

拉乌尔-加布里埃尔·乌尔玛（Raoul-Gabriel Urma）

剑桥大学计算机科学博士，软件工程师，培训师，现任Cambridge Spark公司CEO。在谷歌、eBay、甲骨文和高盛等大公司工作过，并参与过多个创业项目。活跃在技术社区，经常撰写技术文章，多次受邀在国际会议上做技术讲座。

马里奥·富斯科（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。

艾伦·米克罗夫特（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Dell EMC中国卓越研发集团任高级主管工程师，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

上海交通大学硕士，现任SAP（美国）高级软件支持顾问。业余爱好语言、数学、设计

，英、法双语译者，近年翻译出版了《咨询的奥秘》《卓越程序员密码》《计算进化史：改变数学的命运》等书。

目录: 第一部分 基础知识

第1章 Java 8、9、10以及11的变化 2

1.1 为什么要关心Java的变化 2

1.2 Java怎么还在变 4

1.2.1 Java在编程语言生态系统中的位置 5

1.2.2 流处理 6

1.2.3 用行为参数化把代码传递给方法 7

1.2.4 并行与共享的可变数据 8

1.2.5 Java需要演变 9

1.3 Java中的函数 9

1.3.1 方法和Lambda作为一等值 10

1.3.2 传递代码：一个例子 11

1.3.3 从传递方法到Lambda 13

1.4 流 14

1.5 默认方法及Java模块 17

1.6 来自函数式编程的其他好思想 19

1.7 小结 20

第2章 通过行为参数化传递代码 22

2.1 应对不断变化的需求 23

2.1.1 初试牛刀：筛选绿苹果 23

2.1.2 再展身手：把颜色作为参数 23

2.1.3 第三次尝试：对你能想到的每个属性做筛选 24

2.2 行为参数化 25

2.3 对付啰唆 30

2.3.1 匿名类 30

2.3.2 第五次尝试：使用匿名类 31

2.3.3 第六次尝试：使用Lambda表达式 32

2.3.4 第七次尝试：将List类型抽象化 33

2.4 真实的例子 33

2.4.1 用Comparator来排序 33

2.4.2 用Runnable执行代码块 34

2.4.3 通过Callable返回结果 35

2.4.4 GUI事件处理 35

2.5 小结 36

第3章 Lambda表达式 37

3.1 Lambda管中窥豹 37

3.2 在哪里以及如何使用Lambda 40

3.2.1 函数式接口 40

3.2.2 函数描述符 42

3.3 把Lambda付诸实践：环绕执行模式 44

3.3.1 第1步：记得行为参数化 44

3.3.2 第2步：使用函数式接口来传递行为 45

3.3.3 第3步：执行一个行为 45

3.3.4 第4步：传递Lambda 46

3.4 使用函数式接口 47

3.4.1 Predicate 47

3.4.2 Consumer 47

3.4.3 Function 48

3.5 类型检查、类型推断以及限制 52

3.5.1 类型检查 52

3.5.2 同样的Lambda，不同的函数式接口 53

3.5.3 类型推断 55

3.5.4 使用局部变量 56

3.6 方法引用 57

3.6.1 管中窥豹 57

3.6.2 构造函数引用 60

3.7 Lambda和方法引用实战 62

3.7.1 第1步：传递代码 62

3.7.2 第2步：使用匿名类 62

3.7.3 第3步：使用Lambda表达式 62

3.7.4 第4步：使用方法引用 63

3.8 复合Lambda表达式的有用方法 63

3.8.1 比较器复合 64

3.8.2 谓词复合 64

3.8.3 函数复合 65

3.9 数学中的类似思想 66

3.9.1 积分 66

3.9.2 与Java 8的Lambda联系起来 68

3.10 小结 68

第二部分 使用流进行函数式数据处理

第4章 引入流 72

4.1 流是什么 72

4.2 流简介 76

4.3 流与集合 78

4.3.1 只能遍历一次 79

4.3.2 外部迭代与内部迭代 80

4.4 流操作 82

4.4.1 中间操作 83

4.4.2 终端操作 84

4.4.3 使用流 84

4.5 路线图 85

4.6 小结 85

第5章 使用流 86

5.1 筛选 87

5.1.1 用谓词筛选 87

5.1.2 筛选各异的元素 87

5.2 流的切片 88

5.2.1 使用谓词对流进行切片 88

5.2.2 截短流 90

5.2.3 跳过元素 90

5.3 映射 91

5.3.1 对流中每一个元素应用函数 91

5.3.2 流的扁平化 92

5.4 查找和匹配 95

5.4.1 检查谓词是否至少匹配一个元素 95

5.4.2 检查谓词是否匹配所有元素 96

5.4.3 查找元素 96

5.4.4 查找第一个元素 97

5.5 归约 98

5.5.1 元素求和 98

5.5.2 最大值和最小值 100

5.6 付诸实践 103

5.6.1 领域：交易员和交易 103

5.6.2 解答 104

- 5.7 数值流 106
 - 5.7.1 原始类型流特化 107
 - 5.7.2 数值范围 108
 - 5.7.3 数值流应用：勾股数 108
- 5.8 构建流 111
 - 5.8.1 由值创建流 111
 - 5.8.2 由可空对象创建流 111
 - 5.8.3 由数组创建流 112
 - 5.8.4 由文件生成流 112
 - 5.8.5 由函数生成流：创建无限流 113
- 5.9 概述 116
- 5.10 小结 116
- 第6章 用流收集数据 118
 - 6.1 收集器简介 119
 - 6.1.1 收集器用作高级归约 119
 - 6.1.2 预定义收集器 120
 - 6.2 归约和汇总 121
 - 6.2.1 查找流中的最大值和最小值 121
 - 6.2.2 汇总 122
 - 6.2.3 连接字符串 123
 - 6.2.4 广义的归约汇总 124
 - 6.3 分组 127
 - 6.3.1 操作分组的元素 128
 - 6.3.2 多级分组 130
 - 6.3.3 按子组收集数据 131
 - 6.4 分区 134
 - 6.4.1 分区的优势 135
 - 6.4.2 将数字按质数和非质数分区 136
 - 6.5 收集器接口 138
 - 6.5.1 理解Collector接口声明的方法 139
 - 6.5.2 全部融合到一起 143
 - 6.6 开发你自己的收集器以获得更好的性能 144
 - 6.6.1 仅用质数做除数 145
 - 6.6.2 比较收集器的性能 148
 - 6.7 小结 150
- 第7章 并行数据处理与性能 151
 - 7.1 并行流 152
 - 7.1.1 将顺序流转换为并行流 52
 - 7.1.2 测量流性能 154
 - 7.1.3 正确使用并行流 158
 - 7.1.4 高效使用并行流 159
 - 7.2 分支/合并框架 161
 - 7.2.1 使用RecursiveTask 161
 - 7.2.2 使用分支/合并框架的最佳做法 164
 - 7.2.3 工作窃取 165
 - 7.3 Spliterator 166
 - 7.3.1 拆分过程 167
 - 7.3.2 实现你自己的Spliterator 168
 - 7.4 小结 173
- 第三部分 使用流和Lambda进行高效编程
- 第8章 Collection API的增强功能 176
 - 8.1 集合工厂 176
 - 8.1.1 List工厂 177
 - 8.1.2 Set工厂 178

- 8.1.3 Map工厂 179
- 8.2 使用List和Set 180
 - 8.2.1 removeIf方法 180
 - 8.2.2 replaceAll方法 181
- 8.3 使用Map 181
 - 8.3.1 forEach方法 182
 - 8.3.2 排序 182
 - 8.3.3 getOrDefault方法 183
 - 8.3.4 计算模式 183
 - 8.3.5 删除模式 184
 - 8.3.6 替换模式 185
 - 8.3.7 merge方法 185
- 8.4 改进的ConcurrentHashMap 187
 - 8.4.1 归约和搜索 187
 - 8.4.2 计数 188
 - 8.4.3 Set视图 188
- 8.5 小结 188
- 第9章 重构、测试和调试 189
 - 9.1 为改善可读性和灵活性重构代码 189
 - 9.1.1 改善代码的可读性 190
 - 9.1.2 从匿名类到Lambda表达式的转换 190
 - 9.1.3 从Lambda表达式到方法引用的转换 191
 - 9.1.4 从命令式的数据处理切换到Stream 193
 - 9.1.5 增加代码的灵活性 193
 - 9.2 使用Lambda重构面向对象的设计模式 195
 - 9.2.1 策略模式 196
 - 9.2.2 模板方法 197
 - 9.2.3 观察者模式 198
 - 9.2.4 责任链模式 201
 - 9.2.5 工厂模式 202
 - 9.3 测试Lambda表达式 204
 - 9.3.1 测试可见Lambda函数的行为 204
 - 9.3.2 测试使用Lambda的方法的行为 205
 - 9.3.3 将复杂的Lambda表达式分为不同的方法 205
 - 9.3.4 高阶函数的测试 206
 - 9.4 调试 206
 - 9.4.1 查看栈跟踪 206
 - 9.4.2 使用日志调试 208
 - 9.5 小结 209
- 第10章 基于Lambda的领域特定语言 210
 - 10.1 领域特定语言 212
 - 10.1.1 DSL的优点和弊端 212
 - 10.1.2 JVM中已提供的DSL解决方案 214
 - 10.2 现代Java API中的小型DSL 217
 - 10.2.1 把Stream API当成DSL去操作集合 219
 - 10.2.2 将Collectors作为DSL汇总数据 220
 - 10.3 使用Java创建DSL的模式与技巧 221
 - 10.3.1 方法链接 224
 - 10.3.2 使用嵌套函数 226
 - 10.3.3 使用Lambda表达式的函数序列 228
 - 10.3.4 把它们都放到一起 230
 - 10.3.5 在DSL中使用方法引用 232
 - 10.4 Java 8 DSL的实际应用 234
 - 10.4.1 jOOQ 235

- 10.4.2 Cucumber 236
- 10.4.3 Spring Integration 238
- 10.5 小结 239
- 第四部分 无所不在的Java
- 第11章 用Optional取代null 242
 - 11.1 如何为缺失的值建模 243
 - 11.1.1 采用防御式检查减少NullPointerException 243
 - 11.1.2 null带来的种种问题 245
 - 11.1.3 其他语言中null的替代品 245
 - 11.2 Optional类入门 246
 - 11.3 应用Optional的几种模式 248
 - 11.3.1 创建Optional对象 248
 - 11.3.2 使用map从Optional对象中提取和转换值 248
 - 11.3.3 使用flatMap链接Optional对象 249
 - 11.3.4 操纵由Optional对象构成的Stream 253
 - 11.3.5 默认行为及解引用Optional对象 254
 - 11.3.6 两个Optional对象的组合 255
 - 11.3.7 使用filter剔除特定的值 257
 - 11.4 使用Optional的实战示例 258
 - 11.4.1 用Optional 封装可能为null的值 259
 - 11.4.2 异常与Optional的对比 259
 - 11.4.3 基础类型的Optional对象，以及为什么应该避免使用它们 260
 - 11.4.4 把所有内容整合起来 260
 - 11.5 小结 262
- 第12章 新的日期和时间API 263
 - 12.1 LocalDate、LocalTime、LocalDateTime、Instant、Duration以及Period 264
 - 12.1.1 使用LocalDate和LocalTime 264
 - 12.1.2 合并日期和时间 265
 - 12.1.3 机器的日期和时间格式 266
 - 12.1.4 定义Duration或Period 267
 - 12.2 操纵、解析和格式化日期 268
 - 12.2.1 使用TemporalAdjuster 270
 - 12.2.2 打印输出及解析日期-时间对象 272
 - 12.3 处理不同的时区和历法 274
 - 12.3.1 使用时区 274
 - 12.3.2 利用和UTC/格林尼治时间的固定偏差计算时区 275
 - 12.3.3 使用别的日历系统 276
 - 12.4 小结 277
- 第13章 默认方法 278
 - 13.1 不断演进的API 280
 - 13.1.1 初始版本的API 281
 - 13.1.2 第二版API 281
 - 13.2 概述默认方法 283
 - 13.3 默认方法的使用模式 285
 - 13.3.1 可选方法 285
 - 13.3.2 行为的多继承 286
 - 13.4 解决冲突的规则 289
 - 13.4.1 解决问题的三条规则 289
 - 13.4.2 选择提供了最具体实现的默认方法的接口 290
 - 13.4.3 冲突及如何显式地消除歧义 291
 - 13.4.4 菱形继承问题 293
 - 13.5 小结 294
- 第14章 Java模块系统 295
 - 14.1 模块化的驱动力：软件的推理 295

- 14.1.1 关注点分离 295
- 14.1.2 信息隐藏 296
- 14.1.3 Java软件 296
- 14.2 为什么要设计Java模块系统 297
 - 14.2.1 模块化的局限性 297
 - 14.2.2 单体型的JDK 298
 - 14.2.3 与OSGi的比较 299
- 14.3 Java模块：全局视图 300
- 14.4 使用Java模块系统开发应用 301
 - 14.4.1 从头开始搭建一个应用 302
 - 14.4.2 细粒度和粗粒度的模块化 303
 - 14.4.3 Java模块系统基础 303
- 14.5 使用多个模块 304
 - 14.5.1 exports子句 304
 - 14.5.2 requires子句 305
 - 14.5.3 命名 306
- 14.6 编译及打包 306
- 14.7 自动模块 310
- 14.8 模块声明及子句 311
 - 14.8.1 requires 311
 - 14.8.2 exports 311
 - 14.8.3 requires的传递 311
 - 14.8.4 exports to 312
 - 14.8.5 open和opens 312
 - 14.8.6 uses和provides 313
- 14.9 通过一个更复杂的例子了解更多 313
- 14.10 小结 314
- 第五部分 提升Java的并发性
- 第15章 CompletableFuture及反应式编程背后的概念 316
 - 15.1 为支持并发而不断演进的Java 318
 - 15.1.1 线程以及更高层的抽象 319
 - 15.1.2 执行器和线程池 320
 - 15.1.3 其他的线程抽象：非嵌套方法调用 322
 - 15.1.4 你希望线程为你带来什么 324
 - 15.2 同步及异步API 324
 - 15.2.1 Future风格的API 326
 - 15.2.2 反应式风格的API 327
 - 15.2.3 有害的睡眠及其他阻塞式操作 328
 - 15.2.4 实战验证 329
 - 15.2.5 如何使用异步API进行异常处理 330
 - 15.3 “线框-管道”模型 331
 - 15.4 为并发而生的CompletableFuture和结合器 332
 - 15.5 “发布-订阅”以及反应式编程 335
 - 15.5.1 示例：对两个流求和 337
 - 15.5.2 背压 341
 - 15.5.3 一种简单的真实背压 341
 - 15.6 反应式系统和反应式编程 342
 - 15.7 路线图 342
 - 15.8 小结 343
- 第16章 CompletableFuture：组合式异步编程 344
 - 16.1 Future接口 344
 - 16.1.1 Future接口的局限性 346
 - 16.1.2 使用CompletableFuture构建异步应用 346
 - 16.2 实现异步API 347

- 16.2.1 将同步方法转换为异步方法 348
- 16.2.2 错误处理 350
- 16.3 让你的代码免受阻塞之苦 352
 - 16.3.1 使用并行流对请求进行并行操作 353
 - 16.3.2 使用CompletableFuture发起异步请求 353
 - 16.3.3 寻找更好的方案 355
 - 16.3.4 使用定制的执行器 356
- 16.4 对多个异步任务进行流水线操作 358
 - 16.4.1 实现折扣服务 358
 - 16.4.2 使用Discount服务 359
 - 16.4.3 构造同步和异步操作 360
 - 16.4.4 将两个CompletableFuture对象整合起来，无论它们是否存在依赖 363
 - 16.4.5 对Future和CompletableFuture的回顾 364
 - 16.4.6 高效地使用超时机制 365
- 16.5 响应CompletableFuture的completion事件 366
 - 16.5.1 对最佳价格查询器应用的优化 367
 - 16.5.2 付诸实践 368
- 16.6 路线图 369
- 16.7 小结 369
- 第17章 反应式编程 370
 - 17.1 反应式宣言 371
 - 17.1.1 应用层的反应式编程 371
 - 17.1.2 反应式系统 373
 - 17.2 反应式流以及Flow API 373
 - 17.2.1 Flow类 374
 - 17.2.2 创建你的第一个反应式应用 377
 - 17.2.3 使用Processor转换数据 381
 - 17.2.4 为什么Java并未提供Flow API的实现 383
 - 17.3 使用反应式库RxJava 384
 - 17.3.1 创建和使用Observable 385
 - 17.3.2 转换及整合多个Observable 392
- 第六部分 函数式编程以及Java未来的演进
- 第18章 函数式的思考 396
 - 18.1 实现和维护系统 396
 - 18.1.1 共享的可变数据 397
 - 18.1.2 声明式编程 398
 - 18.1.3 为什么要采用函数式编程 399
 - 18.2 什么是函数式编程 399
 - 18.2.1 函数式Java编程 400
 - 18.2.2 引用透明性 402
 - 18.2.3 面向对象的编程和函数式编程的对比 402
 - 18.2.4 函数式编程实战 403
 - 18.3 递归和迭代 405
 - 18.4 小结 408
- 第19章 函数式编程的技巧 409
 - 19.1 无处不在的函数 409
 - 19.1.1 高阶函数 410
 - 19.1.2 柯里化 411
 - 19.2 持久化数据结构 412
 - 19.2.1 破坏式更新和函数式更新的比较 413
 - 19.2.2 另一个使用Tree的例子 415
 - 19.2.3 采用函数式的方法 416
 - 19.3 Stream的延迟计算 418
 - 19.3.1 自定义的Stream 418

19.3.2 创建你自己的延迟列表	420
19.4 模式匹配	425
19.4.1 访问者模式	425
19.4.2 用模式匹配力挽狂澜	426
19.5 杂项	429
19.5.1 缓存或记忆表	429
19.5.2 “返回同样的对象”意味着什么	430
19.5.3 结合器	431
19.6 小结	432
第20章 面向对象和函数式编程的混合：Java和Scala的比较	433
20.1 Scala简介	434
20.1.1 你好，啤酒	434
20.1.2 基础数据结构：List、Set、Map、Tuple、Stream以及Option	436
20.2 函数	440
20.2.1 Scala中的一等函数	441
20.2.2 匿名函数和闭包	442
20.2.3 柯里化	443
20.3 类和trait	444
20.3.1 更加简洁的Scala类	445
20.3.2 Scala的trait与Java 8的接口对比	446
20.4 小结	447
第21章 结论以及Java的未来	448
21.1 回顾Java 8的语言特性	448
21.1.1 行为参数化（Lambda以及方法引用）	449
21.1.2 流	449
21.1.3 CompletableFuture	450
21.1.4 Optional	450
21.1.5 Flow API	451
21.1.6 默认方法	451
21.2 Java 9的模块系统	451
21.3 Java 10的局部变量类型推断	453
21.4 Java的未来	454
21.4.1 声明处型变	454
21.4.2 模式匹配	454
21.4.3 更加丰富的泛型形式	455
21.4.4 对不变性的更深层支持	457
21.4.5 值类型	458
21.5 让Java发展得更快	461
21.6 写在最后的话	462
附录A 其他语言特性的更新	463
附录B 其他类库的更新	467
附录C 如何以并发方式在同一个流上执行多种操作	475
附录D Lambda表达式和JVM字节码	483
• • • • •	(收起)

[Java实战（第2版）_下载链接1](#)

标签

Java

函数式编程

编程

计算机

程序设计

计算科学

CS

评论

第二版比第一版的内容丰富了很多，也深入了很多，特别是后面对Stream，Completable Future和响应式编程RxJava的讲解解答了困惑我很久的问题。

如果未读过第一版，那么这边书力荐阅读，如果读过第一版，这本书推荐阅读

2020.2.17买的电子版，看过第一版，第二遍主要看异步相关。

主要还是JAVA8的内容，加了一些java9的一些case。对于响应式编程有个大致了解。

[Java实战（第2版）_下载链接1](#)

书评

如果你已经很熟悉函数式编程并掌握了一门函数式编程语言（比如Scala），那你很可能对这本书中罗列的Java 8特性嗤之以鼻，但所有的一切前提是Java，对于Java而言，Java 8的改变确实令人惊叹。本书是绝佳的Java 8入门和进阶参考，引入Streaming，函数式编程和新的时间管理库...

我是先读的[《Java 8函数式编程》]再读该书，总体是两本书的质量都非常高，五星推荐！ 1.相比于《Java 8函数式编程》注重于介绍函数式编程相关，该本覆盖的Java 8特性更全面，在书中能看到对Optional、CompletableFuture、新的日期API的介绍； 2.该书对Stream的收集器介绍得更...

这本书作为学习Java8中新增加的编程知识还是非常不错的。在前面章节中，主要介绍了Java8中提供的新特性，比如lamda表达式、stream、函数式编程、CompletableFuture类，新的日期类等等。几乎覆盖了所有的新特性，同时对于原理也有深入的介绍。在介绍这些Java8中的新特性时，作...

如果一开始学习Java是从Java 8以下的版本学习的，这本Java 8实战值得一读，Java 8相对于Java 7及之前的版本代码优雅太多，结合现在web业务前后端分离的Spring Boot后端框架，通过Java 8可以写出精炼、优雅的代码。本书全面的介绍了Java 8的各种写法，第一部分、第二部分属于必...

这点东西根本不值当写成一本书。
几篇连续的博客足以完成这个任务，这个任务适合写成几篇连续的博客。
一本书的内容应该比这个多比这个深。比这个多比这个深的内容才适合写成一本书。
一定要多写文字凑够评论字数要求，为了凑够评论字数要求多写了这些无用的话。
其实我想说的...

[Java实战（第2版）_下载链接1](#)