

Java 9 并发编程实战



[Java 9 并发编程实战 下载链接1](#)

著者:Javier Fernández González

出版者:人民邮电出版社

出版时间:2019-8

装帧:

isbn:9787115505866

作者介绍:

Javier Fernández González 是一名有着超过 10 年 Java 技术经验的软件架构师。他曾担任过教师，研究员，程序员和分析员，现在是 Java 项目、特别是 J2EE 相关项目的架构师。在担任教师期间，他在 Java、J2EE 和 Struts 框架上有超过 1,000 个小时的教学时间。当研究员时，他曾在信息检索领域，用 Java 开发应用程序来处理大量的数据，并且是一些期刊文章及和会议演示的合作者。近些年来，他在不同的领域（比如公共行政，保险，医疗保健，交通，等等）为不同的客户开

发 J2EE Web

应用程序。目前，他在欧洲最大的咨询公司（Capgemini，凯捷）担任软件架构师，为保险公司开发和维护应用程序。

目录: 第 1章 线程管理 1

1.1 简介 1

1.2 线程的创建、运行和设置 2

1.3 线程中断 8

1.4 控制线程中断 11

1.5 线程的休眠和唤醒 14

1.6 等待线程执行结束 16

1.7 守护线程的创建与运行 19

1.8 处理线程中的不可控异常 23

1.9 使用线程本地变量 26

1.10 线程分组及线程组中不可控异常的处理 29

1.11 使用工厂创建线程 33

第 2章 线程同步基础 37

2.1 简介 37

2.2 方法同步 38

2.3 在同步代码块中使用条件 46

2.4 在同步代码块中使用锁机制 51

2.5 用读/写锁保护同步代码块 57

2.6 在一个锁中使用多个条件 62

2.7 高阶知识：StampedLock的使用 70

第3章 线程同步工具 78

3.1 简介 78

3.2 控制对资源的一个或多个副本的并发访问 79

3.3 等待多个并发事件 85

3.4 在指定状态点同步任务 90

3.5 运行阶段性并发任务 98

3.6 阶段性并发任务中阶段转变的控制 108

3.7 两个并发任务间的数据交换 114

3.8 异步地完成和关联任务 118

第4章 线程执行器 128

4.1 简介 128

4.2 创建一个线程执行器并实现其拒绝策略 129

4.3 在一个执行器里执行任务并返回结果 136

4.4 运行多个任务并处理第一个返回结果 140

4.5 运行多个任务并处理全部返回结果 146

4.6 在执行器内延迟运行任务 150

4.7 在执行器内周期性地运行任务 154

4.8 在执行器内取消任务 157

4.9 在执行器内控制任务的完成 160

4.10 在执行器内分离任务的启动并处理返回结果 164

第5章 fork/join框架 171

5.1 简介 171

5.2 创建一个fork/join池 173

5.3 合并任务的执行结果 180

5.4 异步地运行任务 189

5.5 在任务中抛出异常 196

5.6 取消一个任务 200

第6章 并行反应式流 208

6.1 简介 208

- 6.2 使用不同的源创建流 210
- 6.3 归约一个流的元素 217
- 6.4 收集流中的元素 224
- 6.5 把一个动作应用到流的每个元素上 231
- 6.6 过滤流中的元素 234
- 6.7 转换流中的元素 237
- 6.8 排序流中的元素 241
- 6.9 在流中的元素上验证条件 244
- 6.10 在反应式流上反应式编程 248
- 第7章 并发集合 256
 - 7.1 简介 256
 - 7.2 运用非阻塞线程安全的双端队列 257
 - 7.3 运用阻塞线程安全的双端队列 262
 - 7.4 运用按优先级排序的阻塞线程安全队列 265
 - 7.5 运用带延迟元素的线程安全列表 271
 - 7.6 运用线程安全的跳表 276
 - 7.7 运用线程安全的HashMap 281
 - 7.8 运用原子性变量 287
 - 7.9 运用原子性数组 294
 - 7.10 运用volatile关键字 298
 - 7.11 运用变量句柄 302
- 第8章 自定义并发类 307
 - 8.1 简介 307
 - 8.2 自定义ThreadPoolExecutor类 308
 - 8.3 实现一个基于优先级的Executor类 313
 - 8.4 实现ThreadFactory接口来生成自定义线程 317
 - 8.5 在一个Executor对象中使用ThreadFactory 322
 - 8.6 自定义在一个周期调度性线程池中运行的任务 324
 - 8.7 实现一个ThreadFactory以生成fork/join框架的自定义线程 331
 - 8.8 自定义运行于fork/join框架中的任务 338
 - 8.9 实现一个自定义Lock类 342
 - 8.10 实现一个基于优先级的传递队列 348
 - 8.11 实现自己的原子性对象 359
 - 8.12 实现自己的流生成器 363
 - 8.13 实现自己的异步流 369
- 第9章 并发程序的测试 378
 - 9.1 简介 378
 - 9.2 监测Lock接口 379
 - 9.3 监测Phaser类 383
 - 9.4 监测Executor框架 387
 - 9.5 监测fork/join任务池 390
 - 9.6 监测流 395
 - 9.7 输出有效日志信息 397
 - 9.8 利用FindBugs分析并发程序代码 402
 - 9.9 使用Eclipse调试并发程序代码 406
 - 9.10 使用NetBeans调试并发程序代码 408
 - 9.11 使用MultithreadedTC调试并发程序代码 413
 - 9.12 使用JConsole监测 416
- 第 10章 附加信息 421
 - 10.1 简介 421
 - 10.2 在Executor框架中处理Runnable对象的结果 421
 - 10.3 在ForkJoinPool类中处理未控制的异常 427
 - 10.4 使用线程安全的阻塞队列在生产者和消费者之间进行交互 431
 - 10.5 监测Thread类 436

10.6 监测Semaphore类 440
10.7 生成并发随机数 443
第 11 章 并发编程设计 445
11.1 简介 445
11.2 尽可能使用不可变对象 446
11.3 对锁排序以避免死锁 449
11.4 使用原子变量替代同步 451
11.5 尽可能短地持有锁 455
11.6 委托执行器管理线程 459
11.7 使用并发数据结构替代手动编程 462
11.8 使用延迟初始化预防问题 464
11.9 使用fork/join框架替代执行器 466
11.10 避免在锁中使用阻塞操作 470
11.11 避免使用已弃用的方法 472
11.12 使用执行器替代线程组 473
11.13 使用流处理大数据集 474
11.14 其他提示和技巧 479
• • • • • ([收起](#))

[Java 9 并发编程实战_下载链接1](#)

标签

并发

评论

[Java 9 并发编程实战_下载链接1](#)

书评

上手很简单，给的例子由浅入深，很容易理解。虽然没有中文版，还是挺简单的。
一上来看《Java Concurrency in Practice》觉得有些难度的话，可以从这本开始，这两本书再配上源代码，Java并发入门应该没啥问题了。

P151 Finally, you can configure the behavior of the ScheduledThreadPoolExecutor class when you call the shutdown() method and there are pending tasks waiting for the end of their delay time. The default behavior is that those tasks will be executed despite ...

英文很简单，可能作者母语非英语的原因。
这种一个方法一小节的cookbook形式的介绍，也很容易接受。
有一点不好的是：每一节的Getting Ready都是完全重复的文字，难道是为了凑字，哈哈～

我看的是英文原版，中文版翻译的如何不知道。与评分很高的 Java并发编程实战相比，这本书 并没有讲 并发的理论。而是通过一个个的示例 来告诉你怎样 照葫芦画瓢 用多线程/Executors/ForkJoin/ 写出一个可Run的多线程处理程序。书中的示例程序讲解的很仔细（解释每部分代码用。
..

后悔啊 应该看原版后悔啊 应该看原版后悔啊 应该看原版后悔啊 应该看原版后悔啊
应该看原版后悔啊 应该看原版后悔啊 应该看原版后悔啊 应该看原版后悔啊
应该看原版后悔啊 应该看原版后悔啊 应该看原版后悔啊 应该看原版后悔啊
应该看原版后悔啊 应该看原版后悔啊 应该看原版后...

没有讲并发原理，实实在在的实例讲学，脱离了低级趣味的纯粹的cookbook。。
对多线程和并发的理论不是很扎实的可以看看这本书，看完之后，你对并发的理论可能也没啥进步。。 有些书根本没必须要写那么多字，你搞这个要求干嘛。。。
有些书根本没必须要写那么多字，你搞这个要...

[Java 9 并发编程实战_下载链接1](#)