



MANNING

覆盖 Apache Lucene 3.0

Lucene

实战

(第2版)

Lucene IN ACTION

SECOND EDITION

[美] Michael McCandless

[美] Erik Hatcher

[美] Otis Gospodnetic

牛长流 肖宇 译

著



人民邮电出版社
POSTS & TELECOM PRESS

Lucene

实战

(第2版)

Lucene
IN ACTION
SECOND EDITION

[美] Michael McCandless
[美] Erik Hatcher
[美] Otis Gospodnetic

著

牛长流 肖宇 译

人民邮电出版社
北京

图书在版编目 (C I P) 数据

Lucene实战 / (美) 麦肯德利斯 (McCandless, M.),
(美) 哈彻 (Hatcher, E.), (美) 高斯波纳提克
(Gospodnetic, O.) 著; 牛长流, 肖宇译. — 2版. —
北京: 人民邮电出版社, 2011.6
ISBN 978-7-115-25177-0

I. ①L… II. ①麦… ②哈… ③高… ④牛… ⑤肖…
III. ①互联网络—程序设计 IV. ①TP393.4

中国版本图书馆CIP数据核字(2011)第053293号

版 权 声 明

Original English language edition, entitled *Lucene in Action (Second Edition)* by Michael McCandless, Erik Hatcher, Otis Gospodnetic, published by Manning Publications Co., 209 Bruce Park Avenue, Greenwich, CT 06830. Copyright © 2010 by Manning Publications Co.

Simplified Chinese-language edition copyright ©2011 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由 Manning Publications Co.授权人民邮电出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

版权所有, 侵权必究。

Lucene 实战 (第 2 版)

-
- ◆ 著 [美] Michael McCandless Erik Hatcher
Otis Gospodnetic
译 牛长流 肖宇
责任编辑 杜洁
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
三河市潮河印业有限公司印刷
- ◆ 开本: 800×1000 1/16
印张: 30.5
字数: 669 千字 2011 年 6 月第 1 版
印数: 1—3 000 册 2011 年 6 月河北第 1 次印刷

著作权合同登记号 图字: 01-2009-3793 号

ISBN 978-7-115-25177-0

定价: 69.00 元

读者服务热线: (010)67132705 印装质量热线: (010)67129223
反盗版热线: (010)67171154

目录

第 1 部分 Lucene 核心

第 1 章 初识 Lucene 3

- 1.1 应对信息爆炸 4
- 1.2 Lucene 是什么 5
 - 1.2.1 Lucene 能做什么 6
 - 1.2.2 Lucene 的历史 7
- 1.3 Lucene 和搜索程序组件 9
 - 1.3.1 索引组件 10
 - 1.3.2 搜索组件 13
 - 1.3.3 搜索程序的其他模块 16
 - 1.3.4 Lucene 与应用程序的整合点 18
- 1.4 Lucene 实战: 程序示例 18
 - 1.4.1 建立索引 19
 - 1.4.2 搜索索引 22
- 1.5 理解索引过程的核心类 25
 - 1.5.1 IndexWriter 25
 - 1.5.2 Directory 25
 - 1.5.3 Analyzer 26
 - 1.5.4 Document 26
 - 1.5.5 Field 27
- 1.6 理解搜索过程的核心类 27
 - 1.6.1 IndexSearcher 27
 - 1.6.2 Term 28
 - 1.6.3 Query 28
 - 1.6.4 TermQuery 28
 - 1.6.5 TopDocs 29
- 1.7 小结 29

第 2 章 构建索引 30

- 2.1 Lucene 如何对搜索内容进行建模 31
 - 2.1.1 文档和域 31
 - 2.1.2 灵活的架构 32
 - 2.1.3 反向规格化 (Denormalization) 32
- 2.2 理解索引过程 33
 - 2.2.1 提取文本和创建文档 33
 - 2.2.2 分析文档 34
 - 2.2.3 向索引添加文档 34
- 2.3 基本索引操作 35
 - 2.3.1 向索引添加文档 35
 - 2.3.2 删除索引中的文档 38
 - 2.3.3 更新索引中的文档 39
- 2.4 域选项 41
 - 2.4.1 域索引选项 41
 - 2.4.2 域存储选项 42
 - 2.4.3 域的项向量选项 42
 - 2.4.4 Reader、TokenStream 和 byte[] 域值 42
 - 2.4.5 域选项组合 43
 - 2.4.6 域排序选项 44
 - 2.4.7 多值域 44
- 2.5 对文档和域进行加权操作 45
 - 2.5.1 文档加权操作 45

- 2.5.2 域加权操作 46
- 2.5.3 加权基准 (Norms) 47
- 2.6 索引数字、日期和时间 48
 - 2.6.1 索引数字 48
 - 2.6.2 索引日期和时间 49
- 2.7 域截取 (Field truncation) 50
- 2.8 近实时搜索 (Near-real-time search) 51
- 2.9 优化索引 51
- 2.10 其他 Directory 子类 52
- 2.11 并发、线程安全及锁机制 55
 - 2.11.1 线程安全和多虚拟机安全 55
 - 2.11.2 通过远程文件系统访问索引 56
 - 2.11.3 索引锁机制 57
- 2.12 调试索引 59
- 2.13 高级索引概念 60
 - 2.13.1 用 IndexReader 删除文档 61
 - 2.13.2 回收被删除文档所使用过的磁盘空间 62
 - 2.13.3 缓冲和刷新 62
 - 2.13.4 索引提交 63
 - 2.13.5 ACID 事务和索引连续性 65
 - 2.13.6 合并段 66
- 2.14 小结 68

第3章 为应用程序添加搜索功能 70

- 3.1 实现简单的搜索功能 71
 - 3.1.1 对特定项的搜索 72
 - 3.1.2 解析用户输入的查询表达式: QueryParser 73
- 3.2 使用 IndexSearcher 类 76
 - 3.2.1 创建 IndexSearcher 类 76
 - 3.2.2 实现搜索功能 78
 - 3.2.3 使用 TopDocs 类 78
 - 3.2.4 搜索结果分页 79
 - 3.2.5 近实时搜索 79

- 3.3 理解 Lucene 的评分机制 81
 - 3.3.1 Lucene 如何评分 81
 - 3.3.2 使用 explain() 理解搜索结果评分 83
- 3.4 Lucene 的多样化查询 84
 - 3.4.1 通过项进行搜索: TermQuery 类 85
 - 3.4.2 在指定的项范围内搜索: TermRangeQuery 类 86
 - 3.4.3 在指定的数字范围内搜索: NumericRangeQuery 类 87
 - 3.4.4 通过字符串搜索: PrefixQuery 类 88
 - 3.4.5 组合查询: BooleanQuery 类 88
 - 3.4.6 通过短语搜索: PhraseQuery 类 91
 - 3.4.7 通配符查询: WildcardQuery 类 93
 - 3.4.8 搜索类似项: FuzzyQuery 类 94
 - 3.4.9 匹配所有文档: MatchAllDocsQuery 类 95
- 3.5 解析查询表达式: QueryParser 96
 - 3.5.1 Query.toString 方法 97
 - 3.5.2 TermQuery 97
 - 3.5.3 项范围查询 98
 - 3.5.4 数值范围搜索和日期范围搜索 99
 - 3.5.5 前缀查询和通配符查询 99
 - 3.5.6 布尔操作符 100
 - 3.5.7 短语查询 100
 - 3.5.8 模糊查询 101
 - 3.5.9 MatchAllDocsQuery 102
 - 3.5.10 分组查询 102
 - 3.5.11 域选择 103
 - 3.5.12 为子查询设置加权 103
 - 3.5.13 是否一定要使用 QueryParser 103
- 3.6 小结 104

第4章 Lucene 的分析过程 105

- 4.1 使用分析器 106
 - 4.1.1 索引过程中的分析 107
 - 4.1.2 QueryParser 分析 109

- 4.1.3 解析 vs 分析: 分析器
何时不再适用 109
 - 4.2 剖析分析器 110
 - 4.2.1 语汇单元的组成 111
 - 4.2.2 语汇单元流揭秘 112
 - 4.2.3 观察分析器 115
 - 4.2.4 语汇单元过滤器: 过滤
顺序的重要性 119
 - 4.3 使用内置分析器 121
 - 4.3.1 StopAnalyzer 122
 - 4.3.2 StandardAnalyzer 122
 - 4.3.3 应当采用哪种核心
分析器 123
 - 4.4 近音词查询 123
 - 4.5 同义词、别名和其他
表示相同意义的词 126
 - 4.5.1 创建 Synonym
Analyzer 127
 - 4.5.2 显示语汇单元的位置 131
 - 4.6 词干分析 132
 - 4.6.1 StopFilter 保留空位 133
 - 4.6.2 合并词干操作和停用
词移除操作 134
 - 4.7 域分析 134
 - 4.7.1 多值域分析 135
 - 4.7.2 特定域分析 135
 - 4.7.3 搜索未被分析的域 136
 - 4.8 语言分析 139
 - 4.8.1 Unicode 与字符编码 139
 - 4.8.2 非英语语种分析 140
 - 4.8.3 字符规范化处理 140
 - 4.8.4 亚洲语种分析 141
 - 4.8.5 有关非英语语种分析的
其他问题 143
 - 4.9 Nutch 分析 144
 - 4.10 小结 146
- 第 5 章 高级搜索技术 147**
- 5.1 Lucene 域缓存 148
 - 5.1.1 为所有文档加载域值 149
 - 5.1.2 段对应的 reader 149
 - 5.2 对搜索结果进行排序 150
 - 5.2.1 根据域值进行排序 150
 - 5.2.2 按照相关性进行排序 153
 - 5.2.3 按照索引顺序进行
排序 154
 - 5.2.4 通过域进行排序 154
 - 5.2.5 倒排序 155
 - 5.2.6 通过多个域进行排序 156
 - 5.2.7 为排序域选择类型 157
 - 5.2.8 使用非默认的 locale 方式
进行排序 157
 - 5.3 使用 MultiPhraseQuery 158
 - 5.4 针对多个域的一次性
查询 160
 - 5.5 跨度查询 162
 - 5.5.1 跨度查询的构建模块:
SpanTermQuery 165
 - 5.5.2 在域的起点查找跨度 166
 - 5.5.3 彼此相邻的跨度 167
 - 5.5.4 在匹配结果中排除重叠的
跨度 169
 - 5.5.5 SpanOrQuery 类 170
 - 5.5.6 SpanQuery 类和
QueryParser 类 171
 - 5.6 搜索过滤 172
 - 5.6.1 TermRangeFilter 173
 - 5.6.2 NumericRangeFilter 174
 - 5.6.3 FieldCacheRangeFilter 174
 - 5.6.4 特定项过滤 174
 - 5.6.5 使用 QueryWrapper
Filter 类 175
 - 5.6.6 使用 SpanQuery
Filter 类 175
 - 5.6.7 安全过滤器 176
 - 5.6.8 使用 BooleanQuery 类
进行过滤 177
 - 5.6.9 PrefixFilter 178
 - 5.6.10 缓存过滤结果 178
 - 5.6.11 将 filter 封装成 query 179
 - 5.6.12 对过滤器进行过滤 179
 - 5.6.13 非 Lucene 内置的
过滤器 180
 - 5.7 使用功能查询实现自定义
评分 180
 - 5.7.1 功能查询的相关类 180
 - 5.7.2 使用功能查询对最近修改
过的文档进行加权 182
 - 5.8 针对多索引的搜索 184

- 5.8.1 使用 MultiSearch 类 184
- 5.8.2 使用 ParallelMultiSearcher 进行多线程搜索 186
- 5.9 使用项向量 186
 - 5.9.1 查找相似书籍 187
 - 5.9.2 它属于哪个类别 190
 - 5.9.3 TermVectorMapper 类 193
- 5.10 使用 FieldSelector 加载域 194
- 5.11 停止较慢的搜索 195
- 5.12 小结 196
- 第 6 章 扩展搜索 198**
 - 6.1 使用自定义排序方法 199
 - 6.1.1 针对地理位置排序方式进行文档索引 199
 - 6.1.2 实现自定义的地理位置排序方式 200
 - 6.1.3 访问自定义排序中的值 203
 - 6.2 开发自定义的 Collector 204
 - 6.2.1 Collector 基类 205
 - 6.2.2 自定义 Collector: BookLinkCollector 206
 - 6.2.3 AllDocCollector 类 207
 - 6.3 扩展 QueryParser 类 208
 - 6.3.1 自定义 QueryParser 的行为 208
 - 6.3.2 禁用模糊查询和通配符查询 209
 - 6.3.3 处理数值域的范围查询 210
 - 6.3.4 处理日期范围 211
 - 6.3.5 对已排序短语进行查询 213
 - 6.4 自定义过滤器 215
 - 6.4.1 实现自定义过滤器 215
 - 6.4.2 搜索期间使用自定义过滤器 216
 - 6.4.3 另一种选择: FilterQuery 类 217
 - 6.5 有效载荷 (Payloads) 218
 - 6.5.1 分析期间生成有效载荷 219
 - 6.5.2 搜索期间使用有效载荷 220
 - 6.5.3 有效载荷和跨度查询 223
 - 6.5.4 通过 TermPositions 来检索有效载荷 223
 - 6.6 小结 223

第 2 部分 Lucene 应用

- 第 7 章 使用 Tika 提取文本 227**
 - 7.1 Tika 是什么 228
 - 7.2 Tika 的逻辑设计和 API 230
 - 7.3 安装 Tika 231
 - 7.4 Tika 的内置文本提取工具 232
 - 7.5 编程实现文本提取 234
 - 7.5.1 索引 Lucene 文档 234
 - 7.5.2 Tika 工具类 237
 - 7.5.3 选择自定义分析器 238
 - 7.6 Tika 的局限 238
 - 7.7 索引自定义的 XML 文件 239
 - 7.7.1 使用 SAX 进行解析 239
 - 7.7.2 使用 Apache Commons Digester 进行解析和索引 242
 - 7.8 其他选择 244
 - 7.9 小结 245
- 第 8 章 Lucene 基本扩展 246**
 - 8.1 Luke: Lucene 的索引工具箱 247

- 8.1.1 Overview 标签页: 索引的
全局视图 248
- 8.1.2 浏览文档 249
- 8.1.3 使用 QueryParser 进行
搜索 251
- 8.1.4 Files and Plugins
标签页 252
- 8.2 分析器、语汇单元器和
语汇单元过滤器 253
 - 8.2.1 SnowballAnalyzer 255
 - 8.2.2 Ngram 过滤器 256
 - 8.2.3 Shingle 过滤器 258
 - 8.2.4 获取捐赠分析器 258
- 8.3 高亮显示查询项 259
 - 8.3.1 高亮显示模块 259
 - 8.3.2 独立的高亮显示示例 262
 - 8.3.3 使用 CSS 进行高亮显示
处理 263
 - 8.3.4 高亮显示搜索结果 264
- 8.4 FastVector
Highlighter 类 266
- 8.5 拼写检查 269
 - 8.5.1 生成提示列表 269
 - 8.5.2 选择最佳提示 271
 - 8.5.3 向用户展示搜索结果 272
 - 8.5.4 一些加强拼写检查的
考虑 273
- 8.6 引人注目的查询扩展
功能 274
 - 8.6.1 MoreLikeThis 274
 - 8.6.2 FuzzyLikeThisQuery 275
 - 8.6.3 BoostingQuery 275
 - 8.6.4 TermsFilter 276
 - 8.6.5 DuplicateFilter 276
 - 8.6.6 RegexQuery 276
- 8.7 构建软件捐赠模块
(contrib module) 277
 - 8.7.1 源代码获取方式 277
 - 8.7.2 contrib 目录的 Ant
插件 277
- 8.8 小结 278

第 9 章 Lucene 高级扩展 279

- 9.1 链式过滤器 280

- 9.2 使用 Berkeley DB
存储索引 282
- 9.3 WordNet 同义词 284
 - 9.3.1 建立同义词索引 285
 - 9.3.2 将 WordNet 同义词链
接到分析器中 287
- 9.4 基于内存的快速索引 289
- 9.5 XML QueryParser: 超出“one
box” 的搜索接口 289
 - 9.5.1 使用 XmlQueryParser 291
 - 9.5.2 扩展 XML 查询语法 295
- 9.6 外围查询语言 296
- 9.7 Spatial Lucene 298
 - 9.7.1 索引空间数据 299
 - 9.7.2 搜索空间数据 302
 - 9.7.3 Spatial Lucene 的性能
特点 304
- 9.8 远程进行多索引搜索 306
- 9.9 灵活的 QueryParser 309
- 9.10 其他内容 312
- 9.11 小结 313

第 10 章 其他编程语言

使用 Lucene 314

- 10.1 移植入门 315
 - 10.1.1 移植取舍 316
 - 10.1.2 选择合适的移植版本 317
- 10.2 CLucene (C++) 317
 - 10.2.1 移植目的 318
 - 10.2.2 API 和索引兼容 319
 - 10.2.3 支持的平台 321
 - 10.2.4 当前情况以及未来
展望 321
- 10.3 Lucene.Net (C#和其他
.NET 编程语言) 321
 - 10.3.1 API 兼容 323
 - 10.3.2 索引兼容 324
- 10.4 KinoSearch 和
Lucy (Perl) 324
 - 10.4.1 KinoSearch 325
 - 10.4.2 Lucy 327

- 10.4.3 其他 Perl 选项 327
- 10.5 Ferret (Ruby) 328
- 10.6 PHP 329
 - 10.6.1 Zend Framework 329
 - 10.6.2 PHP Bridge 330
- 10.7 PyLucene (Python) 330
 - 10.7.1 API 兼容 332
 - 10.7.2 其他 Python 选项 332
- 10.8 Solr (包含多种编程语言) 332
- 10.9 小结 334

第 11 章 Lucene 管理和性能调优 335

- 11.1 性能调优 336
 - 11.1.1 简单的性能调优步骤 337
 - 11.1.2 测试方法 338
 - 11.1.3 索引-搜索时延调优 339

- 11.1.4 索引操作吞吐量调优 340
- 11.1.5 搜索时延和搜索吞吐量调优 344
- 11.2 多线程和并行处理 346
 - 11.2.1 使用多线程进行索引操作 347
 - 11.2.2 使用多线程进行搜索操作 351
- 11.3 资源消耗管理 354
 - 11.3.1 磁盘空间管理 354
 - 11.3.2 文件描述符管理 357
 - 11.3.3 内存管理 361
- 11.4 热备份索引 364
 - 11.4.1 创建索引备份 365
 - 11.4.2 恢复索引 366
- 11.5 常见错误 367
 - 11.5.1 索引损坏 367
 - 11.5.2 修复索引 369
- 11.6 小结 369

第 3 部分 案例分析

第 12 章 案例分析 1: Krugle 373

- 12.1 Krugle 介绍 374
- 12.2 应用架构 375
- 12.3 搜索性能 376
- 12.4 源代码解析 377
- 12.5 子串搜索 378
- 12.6 查询 VS 搜索 381
- 12.7 改进空间 382
 - 12.7.1 FieldCache 内存使用 382
 - 12.7.2 合并索引 382
- 12.8 小结 383

第 13 章 案例分析 2: SIREn 384

- 13.1 SIREn 介绍 385
- 13.2 SIREn 优势 385

- 13.2.1 通过所有域进行搜索 387
- 13.2.2 一种高效词典 388
- 13.2.3 可变域 388
- 13.2.4 对多值域的高效处理 388
- 13.3 使用 SIREn 索引实体 388
 - 13.3.1 数据模型 389
 - 13.3.2 实现问题 389
 - 13.3.3 索引概要 390
 - 13.3.4 索引前的数据准备 390
- 13.4 使用 SIREn 搜索实体 392
 - 13.4.1 搜索内容 392
 - 13.4.2 根据单元限制搜索范围 393
 - 13.4.3 将单元合并成元组 393
 - 13.4.4 针对实体描述进行查询 394
- 13.5 在 Solr 中集成 SIREn 394
- 13.6 Benchmark 395
- 13.7 小结 397

第 14 章 案例分析 3:

LinkedIn 398

14.1 使用 Bobo Browse 进行 分组搜索 398

14.1.1 Bobo Browse 的设计 400

14.1.2 深层次分组搜索 403

14.2 使用 Zoie 进行实时 搜索 405

14.2.1 Zoie 架构 406

14.2.2 实时 VS 近实时 409

14.2.3 文档与索引请求 411

14.2.4 自定义 IndexReaders 411

14.2.5 与 Lucene 的近实时搜索 进行比较 412

14.2.6 分布式搜索 413

14.3 小结 415

附录 A 安装 Lucene 416

A.1 二进制文件安装 416

A.2 运行命令行演示程序 417

A.3 运行 Web 应用演示 程序 418

A.4 编译源代码 419

A.5 排错 420

附录 B Lucene 索引格式 421

B.1 逻辑索引视图 421

B.2 关于索引结构 422

B.2.1 理解多文件索引结构 422

B.2.2 理解复合索引结构 425

B.2.3 转换索引结构 426

B.3 倒排索引 427

B.4 小结 430

附录 C Lucene/contrib

benchmark 431

C.1 运行测试脚本 432

C.2 测试脚本的组成部分 435

C.2.1 内容源和文档生成器 438

C.2.2 查询生成器 439

C.3 控制结构 439

C.4 内置任务 441

C.4.1 建立和使用行文件 445

C.4.2 内置报表任务 446

C.5 评估搜索质量 446

C.6 出错处理 449

C.7 小结 449

附录 D 资源 450

D.1 Lucene 知识库 450

D.2 国际化 450

D.3 语言探测 451

D.4 项向量 451

D.5 Lucene 移植版本 451

D.6 案例分析 452

D.7 其他 452

D.8 信息检索软件 452

D.9 Doug Cutting 的著作 453

D.9.1 会议论文 453

D.9.2 美国专利 454

第 1 部分

Lucene 核心

本书前半部分涵盖了 Lucene 的 Java 程序包相关内容。第 1 章“初识 Lucene”是对 Lucene 的总体概述，你可据此开发一个完整的索引和搜索程序。每个后续章节都深入阐述了一个具体部分。第 2 章“构建索引”和第 3 章“为应用程序添加搜索功能”均为使用 Lucene 的第一步。第 4 章“Lucene 的分析过程”将深入到索引过程，帮助我们理解 Lucene 是如何对文本进行索引的。

通过对前 4 章的学习，你会较好地理解 Lucene 的一些基本功能。但 Lucene 真正的亮点在于搜索，所以我们将本书前半部分结束前用两章的篇幅来介绍它，分别是：第 5 章“高级搜索技术”，这些高级搜索技术都基于 Lucene 的内置特性；第 6 章“扩展搜索”，展示了 Lucene 针对定制目的而提供的可扩展性。

第 1 章 初识 Lucene

本章要点

- 了解 Lucene
- 理解典型的搜索程序结构
- 使用基本的索引 API
- 使用搜索 API

Lucene 其实是一类强大的 Java 搜索库，它能让你很轻易地将搜索功能加入到任何程序中。近年来 Lucene 变得非常流行，同时它也是使用最为广泛的信息搜索库：它能够增强很多 Web 站点和桌面应用程序的搜索能力。尽管当初它是用 Java 编写的，由于使用太广泛，以及热心开发人员的努力，目前你已经可以自由获取大量的针对其他编程语言的 Lucene 移植版本（其中包含 C/C++、C#、Ruby、Perl、Python 以及 PHP 版本）。

简单易用是 Lucene 广受欢迎的关键因素之一，但是不要被这点所迷惑：后台复杂、设计先进的信息检索技术其实一直在不为人知地运行着。Lucene 是一款设计非常优秀的软件，它向用户提供了简单易用的索引和搜索 API，并屏蔽了复杂的内部实现过程。当开始使用 Lucene 时，你不必深入了解它的信息索引及检索的工作原理。同时由于 Lucene API 简单直接，你只需要学会如何使用它提供的类就可以了。再者，对于早已厌倦了臃肿软件的你而言，会惊奇地发现 Lucene 的核心 JAR 包是如此短小精悍——仅有 1MB 大小——并且不需要其他任何依赖的 JAR 包！

在本章中，我们将分析一款典型搜索程序的总体架构，以及 Lucene 在其中的使用场合。

需要指出的是, Lucene 仅仅是一个提供搜索功能的类库, 所以你还需要根据实际情况自行完成搜索程序的其他模块 (例如网页抓取、文档处理、服务器运行、用户界面和管理等)。在具体分析过程中, 我们将首先通过一些现成的代码示例, 为你展示如何使用 Lucene 进行基本的索引和搜索实现, 然后简要地介绍在索引和搜索过程中需要了解的全部核心知识点。我们处在一个信息爆炸的时代, 需要解决的首要问题就是具备强大的搜索能力。

NOTE Lucene 是一个快速发展的开源项目。当你读到此处时, 很可能 Lucene 的很多 API 和特性都已经发生改变。本书内容基于该项目的 3.0.1 版本, 由于 Lucene 版本是向后兼容的, 本书所有代码示例都可以在后续 3.x 版本中编译和运行。如果在这过程中你遇到任何问题, 请发送邮件至 java-user@lucene.apache.org。Lucene 社区规模巨大, 他们热情并且反馈迅速, 一定能够为你提供帮助。

1.1 应对信息爆炸

为了认识我们这个复杂的世界, 人们发明了各种各样的方案来对信息进行分类和组织。在图书馆使用的杜威十进制分类法 (Dewey Decimal System) 就是层次分类方法的一个经典案例。

随着互联网的普及以及数字信息的爆炸式增长, 人们已经可以足不出户地接触到海量信息。随着数据量的日益剧增, 我们迫切需要采用全新的、更为动态化的方法来查找所需要的信息 (如图 1.1 所示)。尽管我们可以对数据进行分门别类, 但从成千上万的类别或者子类别中查找信息已经不再是一种行之有效的方法了。

如今人们需要在浩如烟海的数据中快速查找所需信息, 这不仅仅体现在互联网领域中——因为台式计算机的数据存储量也随着硬盘存储能力的提高而激增。通过改变目录, 展开或收起文件夹层次结构, 已经不再是一种访问存储文档的有效方法。此外, 人们不仅要使用计算机的原始计算功能, 而且要用它进行通信交流、多媒体播放和存储等。这类应用需要计算机能够快速查找某个特定的数据片段; 同时, 我们还需要能够方便地查到诸如图片、视频和音频文件等各式各样的多媒体文件 (Rich Media)。

我们一边要面对如此大量的数据信息, 一边又在吝惜自己宝贵的时间资源, 为了解决这个矛盾, 我们必须找到一种灵活、自由和及时的数据查询方法, 以便能够在花费最小精力代价的情况下, 用这种方法快速穿越各种严格的分类界限, 准确找到我们需要的信息。

为了说明在互联网和台式计算机中使用广泛的搜索应用, 如图 1.1 所示是在 Google 中搜索 Lucene 的情形。图 1.2 展示了 Apple Mac OS X Finder (类似于 Microsoft Windows 资源管理器) 以及它右上方显示的内嵌搜索功能。Mac OS X 音乐播放器 iTunes, 同样具有内嵌搜索功能, 如图 1.3 所示。

因此, 搜索功能可以说是无处不在! 所有主流操作系统都内嵌了搜索功能。对于

Mac OS X 系统来说，它的突出特性在于，它集成的索引和搜索功能涵盖了所有文件类型，包括具有大量元数据的文件类型，比如电子邮件、通讯录等¹。

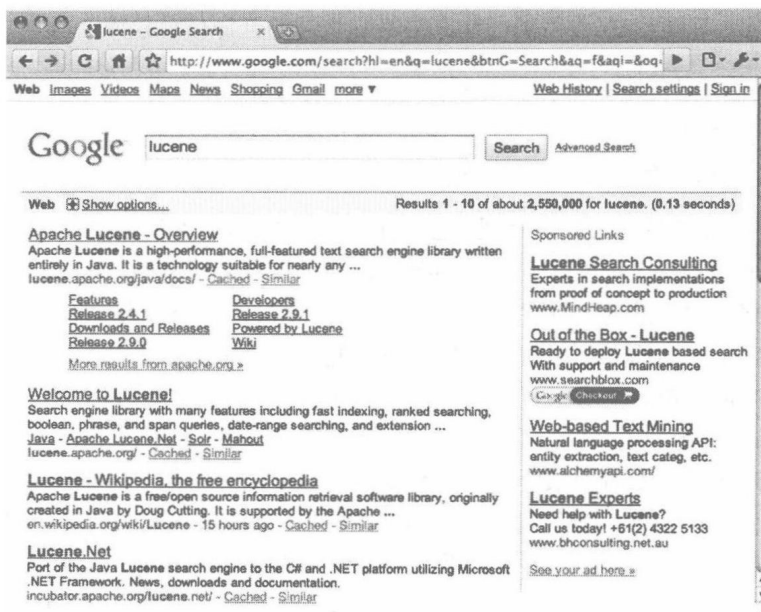


图 1.1 在互联网上利用 Google 进行搜索

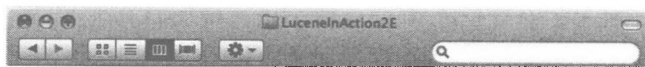


图 1.2 Mac OS X Finder 及其内嵌的搜索器

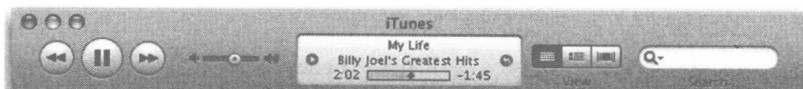


图 1.3 Apple's iTunes 嵌入直观搜索功能

不同的人都在为解决同一个问题而奋斗着——信息量太过庞大——从而采取不同的方法来处理它。一些人致力于不断推出新的用户界面，一些人则是推出智能助理，还有一些人致力于开发复杂的搜索工具或者类似于 Lucene 的搜索库来解决这个问题。在进入示例代码之前，我们先从总体上介绍一下 Lucene 是什么，Lucene 能够做什么，以及它的发展历程。

1.2 Lucene 是什么

Lucene 是一款高性能的、可扩展的信息检索 (IR) 工具库。信息检索是指文档搜索、文档内信息搜索或者文档相关的元数据搜索等操作。Lucene 能够融入到你的应用程序

¹ 本书作者 Erik 和 Mike 对 Apple 相关的技术都颇感兴趣。

中,以增加搜索功能。它是一款以 JAVA 实现的成熟、自由、开源的软件项目,也是 Apache 软件基金 (Apache Software Foundation) 中的一个项目,并且基于 Apache 软件许可协议授权。因此, Lucene 在近年来已经成为最受欢迎的开源信息检索工具库。

NOTE 本书中,我们将一直使用信息检索(或它的英文缩写 IR)这个术语来描述 Lucene 这一类型的搜索工具。人们通常认为信息检索库就是搜索引擎,其实两者是有区别的,我们不能混淆信息检索库与 Web 搜索引擎这两个概念。

使用 Lucene 后你会很快发现,它为你提供了一套简单而强大的核心 API,并且在使用它们时你不必深入理解全文索引和搜索机制。你只需要掌握 Lucene 中少数几个类就可以将它集成到你的应用程序中了。由于 Lucene 只是一个 Java 类库,对于不同的索引和搜索内容它是通用的,相对于其他大量的搜索程序来说,这是个很大的优势。Lucene 的设计紧凑而简单,能够很容易地嵌入桌面应用程序当中。

在 Lucene 的核心 JAR 包之外,有大量的扩展模块,它们提供一些附加功能。其中一些功能对于几乎所有搜索程序来说都是至关重要的,譬如 spellchecker 和 highlighter 模块。这些模块位于一个我们称之为 contrib 的单独区域,本书会多次提及这类 contrib 模块。这些模块的数量太大,所以我们将第 8 章和第 9 章分两章来详细讲解它们!

Lucene 的站点链接为 <http://lucene.apache.org/java>, 在这里你可以更详细地了解 Lucene 的最新状况。主要有:新手教程、Lucene 最近所有版本 API 的 Java 帮助文档、问题跟踪系统、版本下载链接以及 Lucene 的维基链接 <http://wiki.apache.org/lucene-java>, 该维基链接包含了大量的由 Lucene 社区更新和维护的网页。

很可能你在使用 Lucene 的时候自己却并不知道!因为 Lucene 所提供的搜索功能目前正以令人难以置信的速度大量应用于各种场合:网飞公司 (NetFlix)、掘客 (Digg)、MySpace 社交网站、LinkedIn 专业网络社区、联邦快递 (Fedex)、苹果公司 (Apple)、特码捷票务公司 (Ticketmaster)、SalesForce.com 网站、大不列颠百科全书光盘、日蚀集成开发环境 (Eclipse IDE)、梅奥医学中心 (the Mayo Clinic)、New Scientist 杂志社、Atlassian 软件公司 (JIRA)、Epiphany 浏览器、麻省理工学院在线课件 (OpenCourseWare) 和数字空间系统 (DSpace)、Hathi Trust 数字图书馆、Akamai 公司的前端运算 (Edge Computing) 平台等。或许你的名字也即将出现在这个列表中! Lucene 维基页面的技术支持分页有更多的用户列表。

1.2.1 Lucene 能做些什么

人们初次接触到 Lucene 时,很容易将它和一些即用型程序搞混淆,比如文件搜索程序、网页搜索器以及网站搜索引擎等。其实这并不是 Lucene 的真面目: Lucene 只是一个软件类库,或者一个工具箱,而并不是一个完整的搜索程序。Lucene 专注于文本索引和搜索功能,并且运行效果非常不错。Lucene 能够让你的应用程序在不用了解复杂的

索引和搜索实现的情况下，通过调用它的一个简单易用的 API，就能够按照固定规则来进行事务处理。这时你的整个程序将围绕 Lucene 这个核心来运行。

很多完整的搜索程序其实都是建立在 Lucene 核心之上的。如果你正在寻找一些成型的网页搜索程序、文档处理程序以及搜索引擎，可以选择 Lucene 维基页面在其技术支持分页所列出的的一些现成的应用程序。

Lucene 允许你向自己的应用程序中添加搜索功能。Lucene 能够把你从文本中解析出来的数据进行索引和搜索。Lucene 并不关心数据来源、格式，甚至不关心数据的语种，只要能把它转换为文本格式即可。也就是说你可以索引和搜索存储在文件中的如下数据：远程 Web 服务器上的网页、本地文件系统中的文档、简单的文本文件、Word 文档、XML 文档、HTML 文档或者 PDF 文档，或者其他能够从中提取文本信息的数据格式。

同样，你也可以利用 Lucene 来索引存储在数据库中的数据，以给你的用户提供一些其他数据库所不具备的诸如全文搜索等功能。一旦你的应用程序集成了 Lucene，用户就可以进行诸如 +George +Rice -eat-pudding、Apple-pie+Tiger、animal:monkey AND food:banana 等有着复杂查询条件的搜索。有了 Lucene，你可以为电子邮件信息、归档邮件列表、即时聊天信息以及维基（Wiki）页面等信息进行索引和搜索。下面让我们来回顾一下 Lucene 的历史。

1.2.2 Lucene 的历史

Lucene 最初是由 Doug Cutting 编写的¹。当时其工具包可以从 SourceForge 的 Lucene 主页下载。在 2001 年 9 月，Lucene 加入 Apache 软件基金会的 Jakarta 家族并开始提供高质量 Java 开源软件产品；2005 年该项目一跃成为顶级的 Apache 项目。目前，Lucene 包含大量的子项目，具体可以通过 <http://lucene.apache.org> 了解。本书主要包含 Lucene 项目的 Java 子项目，详见 <http://lucene.apache.org/java>，但大多数人一般将这个 Java 子项目称为 Lucene 项目。

自那以后，每个 Lucene 版本的发布都使得这个项目备受关注，吸引着更多的用户和开发人员的眼球。截止 2010 年，最新的 Lucene 版本号是 3.0.1。表 1.1 列出了 Lucene 的版本发布历史。

表 1.1 Lucene 的版本发布历史

版 本	发 布 日 期	里 程 碑
0.01	2000 年 3 月	在 SourceForge 网站第一次发布开源版本
1.0	2000 年 10 月	
1.01b	2001 年 7 月	在 SourceForge 网站最后一次发布
1.2	2002 年 6 月	在 Apache Jakarta 第一次发布
1.3	2003 年 12 月	加入复合索引文件格式，增强了 QueryParser、远程搜索、Token 定位、扩展站的评分 API 等

¹ Lucene 是 Doug 妻子的中名，同时这也是她外祖母的姓。