Broadview
WWW.BROADVIEW.COM.CN

Java技术大系

Java 优化编程

林胜利　王坤茹　孟海利　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
PHEI
http://www.phei.com.cn

Java 技术大系

Java优化编程

林胜利　王坤茹　孟海利　编著

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

内 容 简 介

本书通过丰富、完整、富有代表性的实例，展示了如何提升 Java 应用性能，并且给出了优化前与优化后的 Java 应用程序的性能差别，以实际的实例与数字告诉你，为什么不可以这么做，应该怎么做，深入分析了影响 Java 应用程序性能的根本原因。本书不是教你怎样使用 Java 语言开发应用程序，而是教你怎样才能开发出更高效、更优秀的 Java 应用程序。书中每一个例子都经过了作者严格的验证。

本书适合于所有想编写更高效、完美的 Java 应用程序的开发人员阅读。

图书在版编目（CIP）数据

Java 优化编程 / 林胜利，王坤茹，孟海利编著. —北京：电子工业出版社，2005.5
（Java 技术大系）
ISBN 7-121-01018-6

I. J… II. ①林… ②王… ③孟… III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2005）第 018215 号

责任编辑：孙学瑛
印　　刷：北京智力达印刷有限公司
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
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：24.25　字数：589 千字
印　　次：2005 年 5 月第 1 次印刷
印　　数：5000 册　　定价：36.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。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。

前　言

Java 语言一经问世，就打破以往 C++在面向对象开发界一统天下的格局。Java 语言是一门非常纯正的面向对象语言，它已经被广泛地应用到电信应用系统与企业级系统软件的开发领域中。在目前从事软件开发的程序员中，应用 Java 语言进行开发的占了非常大的比例，这主要是因为 Java 语言的跨平台性与强大的功能。但是大部分开发人员对 Java 语言只是知道如何应用，至于怎样才能应用好，怎样才能提高 Java 语言开发应用程序的效率，却知之甚少。一门优秀的语言只有知道如何更好地运用它，才能充分发挥它的潜在功用，开发出优质的软件。一门语言好比一个工具，如果使用者只知道如何简单地使用它，即使这个工具非常优秀，那么这个使用者也只能用这个工具创造出平庸的作品。对 Java 语言来讲亦是如此。

正是考虑到以上原因，编写本书的目的不是教读者如何使用 Java 语言，也不是向读者介绍 Java 虚拟机的深奥理论，而是通过大量的知识点与开发实例结合作者多年的开发经验，向读者介绍如何提升用 Java 语言所开发的应用程序的效率与性能。本书的目的是告诉那些已经具备 Java 开发基础知识的开发人员与涉猎 Java 开发领域不久的开发人员，怎样才能写出优秀的大师级的 Java 代码，帮助他们快速成为优秀的 Java 程序员。

本书虽然重点讲解了如何提升 Java 应用程序性能的相关理论，但是本书不是一本侧重理论的图书。过于强调理论的图书，通常内容晦涩，容易让读者读后有种云里雾里的感觉，很难体会理解。本书通过丰富、完整、更具代表性的实例，展示了如何使提升应用程序性能。本书不但告诉读者如何提升应用程序的性能，并且给出了这样做与不这样做开发的 Java 应用程序的性能差别，以实际的实例与数字告诉读者，为什么不可以这么做，应该怎么做，并且告诉读者影响 Java 应用程序性能的根本原因是什么，这样可以使读者能更深刻地理解书中所涉及的知识点，从而达到深刻理解、熟练运用的目的。

本书没有教读者怎样使用 Java 语言进行开发，而是教读者怎样才能开发出更高效、更优秀的 Java 应用程序，这是本书的最大特点。举个例子来说，本书没有具体讲解如何运用 Java 核心类来进行开发，因为这些知识很容易找到相关的技术资料，也容易被开发人员掌握。本书重点讲解的是影响 Java 应用程序性能的核心类的特点，以及在开发应用程序时，它们所造成的性能瓶颈，如何可以摆脱这个性能瓶颈，或者给出打破这个瓶颈的可行性方法，引导读者掌握合理使用这些核心类，得到性能最优化的应用程序。书中所涉及的每一个知识点几乎都是 Java 开发中的技术要点，并在讲解这些知识点时，给出了如何避免进入某些技术误区的方法与措施。

本书非常注重内容的充实性与实用性，希望读者通过阅读本书，能获取更多实用的 Java 程序设计技术知识。书中的每一章都包含较多的知识点，通常情况下每一个知识点是一小节，在每一章的结束部分都有一个小结，小结的目的是总结回顾这一章中最重要的知识点，以加深读者对本章所学知识的印象。本书在对每一个知识点的讲解中都附有详细的例子，每个例子都是非常有代表性的并且都经过笔者严格的验证，相信读者在掌

握本书所涵盖的知识之后一定能开发出具备最优性能的 Java 应用程序。

本书大部分知识点都是作者多年开发经验的结晶，衷心地希望它们能帮助读者在 Java 开发的职业生涯中少走弯路，养成良好的编程习惯，尽早成为 Java 开发专家。

本书的结构

本书共分为 11 章。由于本书并不是系统讲解 Java 语言的程序设计，而是着重介绍如何优化 Java 应用开发的关键技术，因此，本书的内容是并列的，但是在写作过程中，笔者还是做了必要的调整与统筹，尽量做到由浅入深地编排本书的内容。下面简单介绍一下本书各章的内容安排。

第 1 章　Java 程序设计风格。本章详细地讲解了有关 Java 程序设计风格的技术知识，其中包括：Java 文件名与文件组织结构，Java 文件头，包的声明与引用，类（class）与接口（interface）的声明，Java 源文件编排格式，Java 程序注释，变量的声明初始化与放置，Java 程序语句编写规则，空格与空行的合理应用，以及 Java 程序设计中方法，变量与常量的命名规则等相关知识，本章最后还给出了在 Java 编程实践中的三个规则（① 访问实例与类中变量的规则；② 引用类中的静态变量与方法的规则；③ 变量赋值规则），通过对上述知识的学习、掌握，只要读者能够坚持上面的编写规则，仅从代码风格上来讲，读者已经具备了编写大师级 Java 代码的能力了。

第 2 章　内存管理。这一章介绍了 Java 内存管理的相关知识，虽然在 Java 程序设计中的内存是由 JVM 控制管理的，但并不是说开发人员无法通过改善应用程序或通过其他途径对内存的使用进行优化，本章通过对下面知识点的讲解，使读者对 Java 内存的使用了如指掌，运用自如：垃圾回收，堆内存（heap），JVM 中的对象生命周期（创建阶段、应用阶段、不可视阶段、不可到达阶段、可收集阶段、终结阶段与释放阶段 7 个阶段），以及 Java 中的析构方法 finalize 调用策略，数组的创建，共享静态变量存储空间，对象重用与 GC，瞬间值（transient），JVM 内存参数调优与 Java 程序设计中有关内存管理的经验等。通过对本章的学习，读者对 Java 的内存管理机制就会有一个系统的认识。在对知识点的讲解过程中，笔者给出了选择与放弃的规则，这样读者就可以在实际的开发中，避免由于不合理地使用 Java 内存而导致系统性能下降或者系统崩溃等情况的发生，并且可以权衡考虑在特定的场景下选用哪种合理的使用内存的技术。

第 3 章　表达式、语句与保留字。这一章讲解了 Java 语言中有关表达式、语句与保留字的相关知识，其中包括：① 表达式的相关规则的知识，如括号应用规则、简单规则、单一意图规则、方法返回值比较规则、字符串比较规则；② Java 语言中的保留字，关于 Java 语言中的关键字我们选出了较为重要的关键字做了详细的讲解，包括：静态的（static）、超类（super）、最终的（final）、同步（synchronized）、实例识别（instanceof）；③ 判断语句与循环语句的相关知识，并且对 if-esle，switch 等判断语句及 for,while 循环语句做了详细的讲解，同时给出了应用经验及建议。在本章的最后介绍了 Java 语言中有关正则表达式的相关知识。通过本章的学习读者会发现在日常的应用程序开发中有很多值得注意的地方，我们却往往忽略了，结合本章所讲解的知识，相信读者能在很多方面

不断地完善、优化读者的应用程序。

第 4 章　Java 核心类与性能优化。这一章讲解了 Java 语言中的部分核心类与开发应用软件性能的知识，本章包括以下相关知识点：散列表相关类及其性能问题，如线程同步散列表类，设置 ArrayList 初始化容量，ArrayList 与 LinkedList 适用场景等知识点；在 String 类与性能中包括，字符串累加与性能，字符串的 length()方法与性能，应用 toCharArray()方法与性能，以及字符串转化为数字等知识点；在系统 I/O 类包括 Java 语言中输入与输出流，通过系统缓冲流类提高 I/O 操作效率，以及通过自定制缓冲区提高 I/O 操作效率等知识点；最后讲解了一些与应用性能相关其他的知识，包括数据格式化与性能及获取文件信息与性能等知识点。通过对本章的学习，相信读者在应用 Java 核心类做应用开发时，有了取舍的依据及怎样提高应用性能的方法。

第 5 章　JNI 程序设计与性能优化。这一章首先讲解了 JNI 技术架构，包括：如何创建带有本地方法的 Java 应用；如何创建 C 端代码，其中又包含，如何创建 C 端代码头文件，以及如何创建 C 端代码主文件两部分内容。接着讲解了 JNI 技术中数据类型与处理方法，其中又包含：JNI 技术中的本地数据类型，如何访问 JNI 本地数据类型的方法，在 JNI 本地方法中访问数组及 JNI 中的主要方法四部分内容。在 JNI 中的重要技术一节中又包括：局部引用与全局引用，处理本地方法引起的 Java 错误，以及线程与本地方法三部分内容。最后讲解了 JNI 数学计算与性能及如何处理好 JNI 技术程序设计中的中文问题等相关技术知识。通过本章的学习，可以帮助读者对 JNI 技术建立一个完整概念，并且掌握其技术要点，知道在什么地方，什么时候应用 JNI 技术，以及怎样更好地应用 JNI 技术，开发优质应用软件。

第 6 章　类与接口。这一章讲解了 Java 语言程序设计中有关类与接口的相关知识，如类的构造器，在类的构造器中又包括：构造器编写规则，类的继承规则（单线继承规则、包内部继承规则与逻辑包含继承规则）等知识点；接着讲解了有关抽象类与接口，继承与组合的应用时机，以及接口与抽象类的应用时机，内部类（Inner Class）及与性能相关的建议与经验的相关知识。通过本章的学习使读者对面向对象程序设计中的两个重要概念——类与接口——有了更高层次上的认识。经历了从过去的知道到现在的精通的这一过程，希望能有一个质的飞越。

第 7 章　JSP 与 Servlet 性能优化。这一章分为两部分内容：第一部分讲解如何提升 JSP 应用性能，其中包括，优化 jspInit()方法，通过优化_jspService()方法提高 JSP 应用性能，JSP 高级知识；第二部分讲解如何提升 Servlet 应用性能，其中包括，提高 Servlet 应用性能的 7 个方法，如何合理缓冲静态数据与动态数据，如何改善 Servlet 应用性能的方法，以及 Filter Servlet 与 Listener Servlet 等相关技术知识。通过本章的学习，可使读者进一步认识到开发良好的 JSP 与 Servlet 应用软件需要多方面综合考虑，掌握提高 JSP 与 Servlet 应用性能的技术要点。

第 8 章　开发高性能的 EJB 应用。这一章首先讲解了有关如何提高 EJB 应用性能的相关技术知识，其中包括，采用 EJB 技术的必要性，EJB 的类型，EJB 的生命周期及 3 种 EJB 的特点与适用场合两部分内容。紧接着讲解了，如何优化 Session Bean 性能，其中包括：HttpSession 与无状态会话（Stateless Session Bean），如何缓冲数据提高性能，如何释

放系统资源，设置 Session Bean 实例池尺寸，以及如何设置与无状态会话 Bean 的生存期等技术知识。在如何优化 Entity Bean 性能一节中又包含如何缓冲数据提高性能，如何释放系统资源，如何设置 Session Bean 实例池尺寸，Entity Bean 中的事务及 Entity Bean 与数据存取等技术知识。在如何优化 Message Driven Bean 性能一节中，连带讲解了如何缓冲数据提高性能，如何释放系统资源，以及如何设置 Message Driven Bean 实例池尺寸等技术知识。最后讲解了 EJB 的结合应用法则与提高 EJB 应用性能的其他途径等内容。通过对本章的学习，使读者进一步加深对 EJB 技术的认识，同时成为一名精通 EJB 技术的高手。

第 9 章　JMS 性能优化。这一章讲解了以下内容，首先讲解了有关 JMS 类型及其各自适用场合，在 JMS 的会话过程管理与优化一节中又包含：如何设置合适的应答模式，事务处理与如何销毁会话对象等技术知识。接下来讲解了如何优化 JMS 虚拟通道及如何优化消息发送与接收过程的相关技术知识，在如何优化消息发送与接收过程一节中又包含，如何选择非持久消息提高系统性能，如何设置最佳消息生命周期，如何选择合适的消息接收方式与完成消息发送与接收后必要的善后工作等技术知识。最后讲解了，如何选择合适的消息类型，JMS 服务器的选择及 JMS 技术的相关经验等内容。通过对本章的学习，使读者进一步加深对 JMS 技术的了解，在实际的应用开发中，通过本章讲解的技术知识，开发更加高效的 JMS 应用。

第 10 章　JDBC 与性能优化。这一章讲解了有关 JDBC 性能优化的技术知识，其中包括 JDBC 编程的步骤，选择最优的驱动程序优化应用性能，如何优化 JDBC 连接，在如何优化 JDBC 连接一节中又包括：如何设置连接的数据行预取属性，如何采用连接池技术提高性能，如何采用合理的事务管理规则，如何选择最优的隔离层及怎样合理、及时关闭连接的相关知识。接着讲解了关于如何优化 Statement 对象，以及如何优化 ResultSet 对象等相关知识。在如何优化 Statement 对象一节中，又包含：如何选用正确的 Statement 接口，如何进行批量数据更新，如何进行批量数据获取，以及为何要及时关闭 Statement 对象的相关技术知识。在如何优化 ResultSet 对象一节中，又包含：如何通过 ResultSet 对象获取批量数据，如何设置恰当的行方向及如何采用适当的 getXXX() 方法等相关技术知识。最后讲解了如何优化 SQL 查询操作大数据字段的读取方法及 JDBC 编程的相关经验等内容。通过对本章的学习，使读者能够更好地应用 JDBC 技术，掌握提升 JDBC 应用的方法，开发高性能的 JDBC 应用。

第 11 章　模式与软件结构、性能优化。这一章首先对设计模式做了简要的介绍，接着讲解了常用模式的 Java 实现与结构优化的技术课题，其中包括：单态/单件（Singleton）模式的结构优化，抽象工厂（Abstract Factory）模式的结构优化，适配器（Adapter）模式的结构优化，代理（Proxy）模式的结构优化，桥接（Bridge）模式的结构优化，命令（Command）模式的结构优化，观察者（Observer）模式的结构优化，职责链（Chain of Responsibility）模式的结构优化，中介者（Mediator）模式的结构优化与访问者（Visitor）模式的结构优化等相关内容。接下来讲解了 J2EE 中的模式与性能的技术知识，其中包括：服务定位（Service Locator）模式与性能，会话门面（Session Facade）模式与性能，消息门面（Message Facade）模式与性能，返回值打包（Ret-Value Packing）模式与性能，返回值包装工厂（Ret Value Packing Factory）模式与性能，值列表处理器（Value List

Handler）模式与性能及复合实体（Entity Composite）模式与性能等内容。

通过对本章的学习，读者应该可以在实际的应用开发中，熟练地使用上述模式来解决实际的问题。并且根据实际的需要创建新的模式。

附录 A　Together 工具的使用简介。在附录 A 中，介绍软件开发、建摸、发布工具 Together，这是一款非常不错的软件工具，尤其是在 UML 建模、文档生成等方面更是强于其他的同类产品。本附录主要介绍如何通过 Together 创建工程，如何完成工程建模，如何引用标准模式，创建工程模块等内容。

附录 B　J2SE 5.0 的新特性与性能的提升。在附录 B 中，介绍 J2SE 的新特性，尤其是介绍 J2SE 5.0 在性能与使用简易方面有所提升的内容。该部分内容取材于 Sun 公司的 J2SE 5.0 的相关技术文档。

附录 C　编排代码的精美工具 JxBeauty。在附录 C 中，向读者介绍了一款非常精致的编排 Java 代码的工具 JxBeauty，并告诉读者如何使用该工具来编排、美化读者的 Java 代码，创建具良好风格的 Java 源程序。

本书的读者对象

如果读者是接触 Java 语言不久的程序员，通过对本书的学习读者可以缩短成为一名 Java 高手的周期；如果读者是一个熟练应用 Java 语言的程序员，但是可能对于 Java 语言开发应用程序时的性能考虑不够深入，或者没有系统、全面的认识，通过掌握本书中所讲解的知识，可以弥补读者这方面的不足，帮助读者在已有的基础上编写更高效、完美的 Java 应用程序。

致谢

作为本书的第一作者，首先感谢我的妻子给予我的极大支持与鼓励。在本书的写作过程中，赛贝斯软件（中国）公司软件研发部的吴昊、连旦晖、王健、赫虹等几位工程师，以及西门子电子（中国）公司研发中心的王立冬高级工程师，给予了许多建设性的建议，使本书所涵盖的知识更加全面、系统。另外刘华军、陈铁路、刘文彬、徐军华、毕晓冬、李艳崑、宋双锐、吕健、李乐、于永梅等同志参与了本书部分章节的校对与编排工作，进一步修正、完善了本书的内容，在此一并致谢！

与编著者联系

书中若有不当之处，敬请读者批评指正。编者的 E-mail 地址是：lsl_wkr@yeah.net。欢迎读者来信指正，谢谢。

林胜利

2005 年 2 月

目　录

第 1 章　Java 程序设计风格

很多时候，开发人员都不太注意自己的程序风格，因此总给人一种“幼稚”的感觉。他们把精力完全放在了程序功能的实现上，当然，程序设计主要目的就是实现软件需求中所描述的用户需求，这本来是无可厚非的，但是不得不提醒开发人员的是，好的程序设计风格同样也非常重要，它会使你的开发生涯焕然一新，使你的开发工作走向“成熟”，令你受益无穷。

从近期目标看是应该先着眼于功能的实现，这也符合大多数管理者的意图，但是只能解一时之渴，而不思长远之计，确不可取。一个杂乱无序的程序让人看后有一种不解其意、心绪繁乱的感觉，由此而会抱怨、怀疑开发人员的技术素质与程序设计修为的低下。如果一个庞大的系统，所有程序代码的风格均是如此，我想这个系统能转交其他开发团队的可能性，就几乎没有了，即使是原班人马近时还可以读懂程序，并可修改程序，如若加以时日，恐怕连自己都会有一种茫然无措、无从下手的感觉。造成这个结果的罪魁祸首就是开发人员没有养成良好的程序设计风格，我想这不仅对开发人员有百害而无一益，对管理人员和企业而言更是损失惨重，结果是残局难以收拾。因为在通常情况下软件公司的人员流动是较为普遍、频繁的。如果软件系统因为无法继续维护开发，而半途夭折，将会给企业带来多大的损失是不言而喻的。即使在花费巨大的精力的前提下，系统得以继续开发，企业也因此付出了极大的人力、物力。因此，通常来说优秀的开发人员一般都有良好的程序设计风格，优秀的管理人员会在自己的开发团队中灌输养成良好程序设计风格的意识，优秀的软件企业都有自己一套健全、成熟的程序设计风格规范文档，用以规范开发人员编写代码的风格。

软件开发所采用的语言是多种多样的，目前较流行的程序设计语言有 Java,C,C++, C#,Basic 等，每一种语言都有自己的程序设计风格与规范，我们常常会在阅读一些高手所设计的程序代码中享受到因体会、感受到良好的代码风格所带给人一种舒服的感觉，同时也能感觉到此代码作者的水平。本书是介绍 Java 程序设计技术的，因此，我们只对 Java 程序设计的风格做一下讨论。

从事 Java 程序设计已有很长的时间了，在这之前我热衷于 C 与 C++程序设计，但是自从接触 Java 语言后，有一种相识恨晚的感觉，说得有些缠绵，不过当用 Java 开发一个软件产品哪怕是编写一个小的软件模块时都会有一种极强的成就感，会像欣赏一件完美的艺术品一样品味它曼妙的格式，Java 灵活的对象引用及对象方法的调用方式可以令你极尽串联引用之能事（如：obj.getArray()[i].getName().toString()），可谓“完美、神俊”。下面我们就来看下面一段曲线玲珑的代码：

```
… …
if (elem == null) {
```

```
            for (int i = 0; i < size; i++)
                if (elementData[i]==null)
                    return i;
        } else if (elem == obj ) {

                                    if (maxIndex > 100) {
                                     for (int i = 0; i < size; i++)
                            if (elem.equals(elementData[i]))
                                return i;
                                                    else
obj.getArray()[i].getName().toString();
                                    }
        }
        return -1;
        ......
```

这段代码不长，结构也相对简单，却能体会出 Java 代码的绰约风姿。错落有致的代码确实会给人一种心旷神怡、赏心悦目的感觉。啰唆了这么多，读者可能有些不耐烦了，下面就详细地介绍在进行 Java 程序设计时应该注意的书写格式，应用好这些格式同样也能写出大师级的代码。

1.1 Java 文件名与文件组织结构

Java 文件名由实意文件名+后缀组成，后缀名因类型的不同而不同。如表 1-1 所示。

表 1-1 后缀名的意义

文件类型	后缀名
Java 源文件	.java
Java 字节码文件	.class

一般来说 Java 源文件的结构是这样的，每一节代码之间是由一个空行分割开的，并且每一节都有相应的注释，通常来讲一个 java 源文件不应该超过 2 000 行，否则就是视为类“臃肿”，这种情况在程序设计时，应该尽量避免。

在 Java 源文件中应该包含一个单一的公共类（class）或接口（interface），当私有类、私有接口与公共类有关联时，你可以把它们分到公共类的源文件中，这个公共类或公共接口，应该是这个源文件的第一个类或接口。

每一个 Java 源文件一般由下面的顺序构成：

（1）文件注释头

（2）包名（package）

（3）引入（import）声明
（4）类（class）或接口（interface）的声明部分

1.2　Java 文件注释头

Java 类文件注释头是用来描述该类功能及其特点，以及相关开发信息的，如该类的关联类（通常情况下不描述 Java 系统核心类如 java.util.Vector, java.lang.Thread 等）、开发公司或单位、版权、作者、代码审定人该类所支持的 JDK 版本、该类版本、开发日期、最后更改日期、修改人、复审人等信息，下面就是一个 Java 类文件注释头：

```
/**********************************************************************
     * 该类功能及其特点的描述（例如：该类是用来……）
     *
     * 该类未被编译测试过。
     *
     * @see（与该类相关联的类）：(AnatherClass.java)
     *
*
* 开发公司或单位：XX 软件有限公司研发中心
*
* 版权：本文件版权归属 XX 公司研发中心
*
*
     * @author（作者）：林胜利
     *
     * @since（该文件所支持的 JDK 版本）：Jdk1.3 或 JDK1.4
*
     * @version（版本）：1.0
     *
* @date （开发日期）：    1999-01-29
*
* 最后更改日期：          2003-05-16
*
* 修改人：    林胜利
*
* 复审人：    张三  李四  王五

**********************************************************************/
```

你可能已经注意到文件注释头包括@see，@author，@since，@version，@date 等标记，其实这是为后面，我们即将介绍的关于通过 jdoc 生成 Java API 标准文档做准备的，