深入理解Java 7:核心技术与最佳实践

作者：成富

深入理解Java 7

——核心技术与最佳实践

成富　著

ISBN：978-7-111-38039-9

本书纸版由机械工业出版社于2012年出版，电子版由华章分社（北京华章图文信息有限公司）全球范围内制作与发行。

客服热线：+ 86-10-68995265

客服信箱：service@bbbvip.com

官方网址：www.hzmedia.com.cn

新浪微博 @研发书局

腾讯微博 @yanfabook

目　录

前言
为什么要写这本书
读者对象及如何阅读本书
勘误和支持
致谢
Java的挑战与展望
第1章　Java 7语法新特性
1.1　Coin项目介绍
1.2　在switch语句中使用字符串
1.2.1　基本用法
1.2.2　实现原理
1.2.3　枚举类型
1.3　数值字面量的改进
1.3.1　二进制整数字面量
1.3.2　在数值字面量中使用下划线
1.4　优化的异常处理
1.4.1　异常的基础知识
1.4.2　创建自己的异常
1.4.3　处理异常
1.4.4　Java 7的异常处理新特性
1.5　try-with-resources语句
1.6　优化变长参数的方法调用
1.7　小结
第2章　Java语言的动态性
2.1　脚本语言支持API
2.1.1　脚本引擎
2.1.2　语言绑定
2.1.3　脚本执行上下文
2.1.4　脚本的编译
2.1.5　方法调用
2.1.6　使用案例
2.2　反射API
2.2.1　获取构造方法
2.2.2　获取域
2.2.3　获取方法
2.2.4　操作数组
2.2.5　访问权限与异常处理
2.3　动态代理
2.3.1　基本使用方式
2.3.2　使用案例
2.4　动态语言支持
2.4.1　Java语言与Java虚拟机
2.4.2　方法句柄
2.4.3　invokedynamic指令
2.5　小结
第3章　Java I/O
3.1　流
3.1.1　基本输入流
3.1.2　基本输出流
3.1.3　输入流的复用
3.1.4　过滤输入输出流
3.1.5　其他输入输出流
3.1.6　字符流
3.2　缓冲区
3.2.1　基本用法
3.2.2　字节缓冲区
3.2.3　缓冲区视图
3.3　通道
3.3.1　文件通道
3.3.2　套接字通道
3.4　NIO.2
3.4.1　文件系统访问
3.4.2　zip/jar文件系统
3.4.3　异步I/O通道
3.4.4　套接字通道绑定与配置
3.4.5　IP组播通道
3.5　使用案例
3.6　小结
第4章　国际化与本地化
4.1　国际化概述
4.2　Unicode

4.2.1 Unicode编码格式
4.2.2 其他字符集
4.2.3 Java与Unicode
4.3 Java中的编码实践
4.3.1 Java NIO中的编码器和解码器
4.3.2 乱码问题详解
4.4 区域设置
4.4.1 IETF BCP 47
4.4.2 资源包
4.4.3 日期和时间
4.4.4 数字和货币
4.4.5 消息文本
4.4.6 默认区域设置的类别
4.4.7 字符串比较
4.5 国际化与本地化基本实践
4.6 小结
第5章 图形用户界面
5.1 Java图形用户界面概述
5.2 AWT
5.2.1 重要组件类
5.2.2 任意形状的窗口
5.2.3 半透明窗口
5.2.4 组件混合
5.3 Swing
5.3.1 重要组件类
5.3.2 JLayer组件和LayerUI类
5.4 事件处理与线程安全性
5.4.1 事件处理
5.4.2 事件分发线程
5.4.3 SwingWorker类
5.4.4 SecondaryLoop接口
5.5 界面绘制
5.5.1 AWT中的界面绘制
5.5.2 Swing中的绘制
5.6 可插拔式外观样式
5.7 JavaFX
5.7.1 场景图
5.7.2 变换
5.7.3 动画效果
5.7.4 FXML
5.7.5 CSS外观描述
5.7.6 Web引擎与网页显示
5.8 使用案例
5.9 小结
第6章 Java 7其他重要更新
6.1 关系数据库访问
6.1.1 使用try-with-resources语句
6.1.2 数据库查询的默认模式
6.1.3 数据库连接超时时间与终止
6.1.4 语句自动关闭
6.1.5 RowSet实现提供者
6.2 java.lang包的更新
6.2.1 基本类型的包装类
6.2.2 进程使用
6.2.3 Thread类的更新
6.3 Java实用工具类
6.3.1 对象操作
6.3.2 正则表达式
6.3.3 压缩文件处理
6.4 JavaBeans组件
6.4.1 获取组件信息
6.4.2 执行语句和表达式
6.4.3 持久化
6.5 小结
第7章 Java虚拟机
7.1 虚拟机基本概念
7.2 内存管理
7.3 引用类型
7.3.1 强引用
7.3.2 引用类型基本概念
7.3.3 软引用
7.3.4 弱引用

7.3.5　幽灵引用
7.3.6　引用队列
7.4　Java本地接口
7.4.1　JNI基本用法
7.4.2　Java程序中集成C/C++代码
7.4.3　在C/C++程序中启动Java虚拟机
7.5　HotSpot虚拟机
7.5.1　字节代码执行
7.5.2　垃圾回收
7.5.3　启动参数
7.5.4　分析工具
7.5.5　Java虚拟机工具接口
7.6　小结
第8章　Java源代码和字节代码操作
8.1　Java字节代码格式
8.1.1　基本格式
8.1.2　常量池的结构
8.1.3　属性
8.2　动态编译Java源代码
8.2.1　使用javac工具
8.2.2　Java编译器API
8.2.3　使用Eclipse JDT编译器
8.3　字节代码增强
8.3.1　使用ASM
8.3.2　增强代理
8.4　注解
8.4.1　注解类型
8.4.2　创建注解类型
8.4.3　使用注解类型
8.4.4　处理注解
8.5　使用案例
8.6　小结
第9章　Java类加载器
9.1　类加载器概述
9.2　类加载器的层次结构与代理模式
9.3　创建类加载器
9.4　类加载器的隔离作用
9.5　线程上下文类加载器
9.6　Class.forName方法
9.7　加载资源
9.8　Web应用中的类加载器
9.9　OSGi中的类加载器
9.9.1　OSGi基本的类加载器机制
9.9.2　Equinox框架的类加载实现机制
9.9.3　Equinox框架嵌入到Web容器中
9.10　小结
第10章　对象生命周期
10.1　Java类的链接
10.2　Java类的初始化
10.3　对象的创建与初始化
10.4　对象终止
10.5　对象复制
10.6　对象序列化
10.6.1　默认的对象序列化
10.6.2　自定义对象序列化
10.6.3　对象替换
10.6.4　版本更新
10.6.5　安全性
10.6.6　使用Externalizable接口
10.7　小结
第11章　多线程与并发编程实践
11.1　多线程
11.1.1　可见性
11.1.2　Java内存模型
11.1.3　volatile关键词
11.1.4　final关键词
11.1.5　原子操作
11.2　基本线程同步方式
11.2.1　synchronized关键词
11.2.2　Object类的wait、notify和notifyAll方法
11.3　使用Thread类
11.3.1　线程状态

11.3.2　线程中断
11.3.3　线程等待、睡眠和让步
11.4　非阻塞方式
11.5　高级实用工具
11.5.1　高级同步机制
11.5.2　底层同步器
11.5.3　高级同步对象
11.5.4　数据结构
11.5.5　任务执行
11.6　Java SE 7新特性
11.6.1　轻量级任务执行框架fork/join
11.6.2　多阶段线程同步工具
11.7　ThreadLocal类
11.8　小结
第12章　Java泛型
12.1　泛型基本概念
12.2　类型擦除
12.3　上界和下界
12.4　通配符
12.5　泛型与数组
12.6　类型系统
12.7　覆写与重载
12.7.1　覆写对方法类型签名的要求
12.7.2　覆写对返回值类型的要求
12.7.3　覆写对异常声明的要求
12.7.4　重载
12.8　类型推断和<>操作符
12.9　泛型与反射API
12.10　使用案例
12.11　小结
第13章　Java安全
13.1　Java安全概述
13.2　用户认证
13.2.1　主体、身份标识与凭证
13.2.2　登录
13.3　权限控制
13.3.1　权限、策略与保护域
13.3.2　访问控制权限
13.3.3　特权动作
13.3.4　访问控制上下文
13.3.5　守卫对象
13.4　加密与解密、报文摘要和数字签名
13.4.1　Java密码框架
13.4.2　加密与解密
13.4.3　报文摘要
13.4.4　数字签名
13.5　安全套接字连接
13.5.1　SSL协议
13.5.2　HTTPS
13.6　使用案例
13.7　小结
第14章　超越Java 7
14.1　lambda表达式
14.1.1　函数式接口
14.1.2　lambda表达式的语法
14.1.3　目标类型
14.1.4　词法作用域
14.1.5　方法引用
14.1.6　接口的默认方法
14.2　Java平台模块化
14.3　Java SE 8的其他更新
14.4　小结
附录A　OpenJDK
附录B　Java简史

前言

为什么要写这本书

我最早开始接触Java语言是在大学的时候。当时除了用Java开发一些小程序之外，就是用Struts框架开发Web应用。在后来的实习和工作中，我对Java的使用和理解更加深入，逐渐涉及Java相关的各种不同技术。使用Java语言的一个深刻体会是：Java语言虽然上手容易，但是要真正掌握并不容易。

Java语言对开发人员屏蔽了一些与底层实现相关的细节，但是仍然有很多内容对开发人员来说是很复杂的，这些内容恰好是容易出现错误的地方。我在工作中就经常遇到与类加载器和垃圾回收相关的问题。在解决这些问题的过程中，我积累了一些经验，遇到类似的问题可以很快地找到问题的根源。同时，在解决这些实际问题的过程中，我意识到虽然可以解决某些具体的问题，但是并没有真正理解这些问题的解决办法背后所蕴含的基本原理，仍然还只是处于一个“知其然，不知其所以然”的状态。于是我开始阅读Java相关的基础资料，包括Java语言规范、Java虚拟机规范、Java类库的源代码和其他在线资料等。在阅读的基础上，编写小程序进行测试和试验。通过阅读和实践，我对Java平台中的一些基本概念有了更加深入的理解。从2010年开始，我对积累的相关知识进行了整理，在InfoQ中文站的“Java深度历险”专栏上发表出来，受到了一定的关注。

2011年7月，在时隔数年之后，Java的一个重大版本Java SE 7发布了。在这个新的版本中，Java平台增加了很多新的特性。在Java虚拟机方面，invokedynamic指令的加入使虚拟机上的动态语言的性能得到很大的提升。这使得开发人员可以享受到动态语言带来的在提高生产效率方面的好处。在Java语言方面，语言本身的进一步简化，使开发人员编写代码的效率更高。在Java类库方面，新的IO库和同步实用工具类为开发人员提供了更多实用的功能。从另外一个角度来说，Java SE 7是Oracle公司收购Sun公司之后发布的第一个Java版本，从侧面反映出了Oracle公司对Java社区的领导力，可以继续推动Java平台向前发展。这可以打消企业和社区对于Oracle公司领导力的顾虑。Java SE 7的发布也证明了基于JCP和OpenJDK的社区驱动模式可以很好地推动Java向前发展。

随着新版本的发布，肯定会有越来越多的开发人员想尝试使用Java SE 7中的新特性，毕竟开发者社区对这个新版本期待了太长的时间。在Java程序中使用这些新特性，可以提高代码质量，提升工作效率。Java平台的每个版本都致力于提高Java程序的运行性能。随着新版本的发布，企业都应该考虑把Java程序的运行平台升级到最新的Java SE 7，这样可以享受到性能提升所带来的好处。对于新的Java程序开发，推荐使用Java SE 7作为标准的运行平台。本书将Java SE 7中的新特性介绍和对Java平台的深入探讨结合起来，让读者既可以了解最新版本的Java平台的新特性，又可以对Java平台的底层细节有更加深入的理解。

读者对象及如何阅读本书

本书面向的主要读者是具备一定Java基础的开发人员和在校学生。本书中不涉及Java的基本语法，因此不适合Java初学者阅读。如果只对Java SE 7中的新特性感兴趣，可以阅读第1章到第6章；如果对Java中的特定主题感兴趣，可以根据目录有选择地阅读。另外，第1章到第6章虽然以Java SE 7的新特性介绍为主，但是其中也穿插了对相关内容的深入探讨。

本书可分为三大部分：

第一部分为Java SE 7新特性介绍，从第1章到第6章。这部分详细地介绍了Java SE 7中新增的重要特性。在对新特性的介绍中，也包含了对Java平台相关内容的详细介绍。

第二部分为Java SE 7的深入探讨，从第7章到第13章。这部分着重讲解了Java平台上的底层实现，并对一些重要的特性进行了深入探讨。这个部分所涉及的内容包括Java虚拟机、Java源代码和字节代码操作、Java类加载器、对象生命周期、多线程与并发编程实践、Java泛型和Java安全。

第三部分为Java SE 8的内容展望，即第14章。这部分简要介绍了Java SE 8中将要增加的新特性。

本书还通过两个附录对OpenJDK（附录A）和Java语言的历史（附录B）进行了简要的介绍。

勘误和支持

由于作者的水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。如果您有更多的宝贵意见，欢迎发送邮件至邮箱alexcheng1982@gmail.com，也可以通过微博（http://weibo.com/alexcheng1982）与我取得联系。期待能够得到您的真挚反馈。

本书官方微群：http://q.weibo.com/943166。

本书中的源代码请登录华章公司的网站（http://www.hzbook.com）本书页面进行下载。

致谢

感谢InfoQ中文站和InfoQ编辑张凯峰先生。这本书能够面世，得益于我在InfoQ中文站的“Java深度历险”专栏上发表的文章。

感谢机械工业出版社华章公司的编辑杨福川和姜影的辛勤工作，使得这本书能够最终顺利完成。

感谢家人和朋友对我的支持与帮助！

Java的挑战与展望

从Java语言出现到现在的16年间，在Java语言本身发展演化的同时，整个软件开发行业也在发生着巨大的变化。新的软件开发思想和程序设计语言层出不穷。虽然Java语言一直是最流行的程序设计语言之一，但它也面临着来自其他编程语言的冲击。这其中主要是互联网应用发展所带来的动态语言的影响。

Java是静态强类型语言。这种特性使Java编译器在编译时就可以发现非常多的类型错误，而不会让这些错误在运行时才暴露出来。对于构建一个稳定而安全的应用来说，这是一个很大的优势，但是这种静态的类型检查也限制了开发人员编写代码时的创造性和灵活性。

Web 2. 0概念的出现和互联网应用的发展，为新语言的流行创造了契机。Ruby语言凭借着杀手级应用Ruby on Rails一举蹿红，而Google的Web应用开发平台Google App Engine最初也只支持Python一种语言，甚至流行的JavaScript语言也借助于node.js和Aptana Jaxer等平台在服务器端开发中占据了一席之地。这些语言的共同特征是动态类型与灵活自由的语法。开发人员一旦掌握了这些语言，开发效率会非常高。在这一点上，Java语言繁琐的语法就显得缺乏吸引力。Java语言也受到来自同样运行在Java虚拟机上的其他语言的挑战。这些语言包括Groovy、Scala、JRuby和Jython等。任何语言，只要它生成的字节代码符合Java字节代码规范，就可以在Java虚拟机上运行。前面提到的这些Java虚拟机上的语言既具有简洁优雅的语法，又能充分利用已有的Java虚拟机资源，相对于Java语言本身来说，非常具有竞争力。

基于前面的这些现状，在社区中有人悲观地预言：Java已死，COBOL式的死亡。COBOL这门诞生于20世纪50年代末的编程语言，已经被诸多机构和个人论证为已经死亡的语言。实际上，COBOL语言仍然在银行、金融和会计等商业应用领域占据着主导地位。只要这些应用存在，COBOL语言就不会消亡。Java语言也是如此。只要运行在Java平台上的应用还存在，Java语言就能一直生存下去。事实上，现在仍然有许多公司和个人在向Java平台投资。这些投资既包括投入资金和人力来开发基于Java平台的应用，也包括投入时间来学习Java平台的相关技术。

当然，Java平台也有不足之处，其中最明显的是整个Java平台的复杂性。最早在JDK 1.0发布的时候，只有几百个Java类，而现在的Java 6已经包括Java SE、Java EE和Java ME等多个版本，所包含的Java类多达数千个。对于普通开发者来说，完全理解和熟悉如此庞大的类库的难度非常大。在日常的开发过程中，经常可以看到开发者在重复实现某些功能，而这些功能在Java类库中已经存在，只是不被人知道而已。除了庞大的类库之外，Java语言的语法本身也缺乏足够的灵活性，实现某些功能所需的代码量可能是其他语言的几倍。另外一个复杂性体现在Web应用开发方面。一个完整的Java EE应用程序要求程序员掌握和理解的概念太多，要使用的库也非常多。这点从市面上到处可见的以Java Web应用开发和Struts、Spring及Hibernate等框架为内容的图书上就可以看出来。虽然新出现的Grails和Play框架等都试图降低这个复杂度，但是这些新的框架的流行仍然需要足够长的时间。

对于Java语言的未来，我们有理由相信Java平台会一直发展下去。其中很重要的依据是Java平台的开放性。依托JCP和OpenJDK项目，Java平台不仅在语言规范这个层次上有健康的开放管理流程，也有与之对应的参考实现。Java语言有着人数众多的开发者社区，每年有非常多新的开发者学习和使用Java。大量的开发者使用Java语言开发各种不同类型的应用。在社区中可以看到很多提供不同功能的类库和框架。Java虚拟机已经被安装到数以十亿计的不同类型的设备上，包括服务器、个人计算机、移动设备和智能卡等。依托庞大的社区和数量众多的运行平台，Java语言的发展前景是非常乐观的。

对于Java平台来说，未来的发展将侧重于以下几个重要的方面。第一个方面是提高开发人员的生产效率。由于Java语言的静态强类型特性，使用Java语言编写的程序代码一般比较繁琐，包含了过多不必要的语法元素，这在一定程度上降低了开发人员的生产效率。大量的时间被浪费在语言本身上，而不是真正需要的业务逻辑上。从另外一个角度来说，Java语言的这种严谨性，对于复杂应用的团队开发是大有好处的，有利于构建健壮的应用。Java语言需要在这两者之间达到一个平衡。Java语言的一个发展趋势是在可能的范围内降低语言本身的语法复杂度。从J2SE 5.0中增强的for循环，到JavaSE 7中的try-with-resources语句和＜＞操作符，再到Java SE 8中引入的lambda表达式，Java正在不断地简化自身的语法。

第二个方面是提高性能。Java平台的性能一直为开发人员所诟病，这主要是因为Java虚拟机这个中间层次的存在。随着硬件技术的发展，越来越多的硬件平台采用了多核CPU和多CPU的架构。应用程序应该充分利用这些资源来提高程序的运行性能。Java平台需要帮助开发人员更好地实现这个目标。Java SE 7中的fork/join框架是一个高效的任务执行框架。Java SE 8对集合类框架和相关API做了增强，以支持对批量数据进行自动的并行处理。

第三个方面是模块化。一直以来，Java平台所包含的各种功能不同的类库是一个统一的整体。在一个程序的运行过程中，很多类库其实是不需要的。比如对于一个服务器端运行的程序来说，Swing用户界面组件库通常是不需要的。模块化的含义是把Java平台提供的类库划分成不同的相互依赖的模块，程序可以根据需要选择运行时所依赖的模块，只有被选择的模块才会在运行时被加载。模块化的实现不仅可以应用到Java平台本身，也可以应用到Java应用程序的开发中，OpenJDK中的Jigsaw项目提供了这种模块化的支持。