HZ BOOKS

Apress®

在系统管理中应用Ruby强大而优雅的功能

Ruby 系统管理实战

Practical Ruby for System Administration

André Ben Hamou 著

仲田 等译

Ruby和Rails技术系列

机械工业出版社
China Machine Press

Ruby 系统管理实战

Practical Ruby for System Administration

André Ben Hamou 著
仲田 等译

机械工业出版社
China Machine Press

本书主要讲述用 Ruby 来进行系统管理和维护。本书主要内容包括用 Ruby 来构建文件，存储和检索对象，接入数据云团，构建领域专用语言，处理企业数据，监控网络，执行网络流量分析，测试与编写文档等。本书帮助你掌握 Ruby 编码风格的实用技巧，学会分析和改进脚本的性能，并把关于脚本开发流程的实用建议加以运用。

本书适合系统管理人员和系统维护人员参考。

André Ben Hamou: *Practical Ruby for System Administration*(ISBN: 1-59059-821-0).

Original English language edition published by Apress L. P., 2560 Ninth Street, Suite 219, Berkeley, CA 94710 USA. Copyright © 2007 by Apress L. P.

Simplified Chinese-language edition copyright © 2008 by China Machine Press. All rights reserved.

This edition is licensed for distribution and sale in the People's Republic of China only, excluding Hong Kong, Taiwan and Macao and may not be distributed and sold elsewhere.

本书原版由 Apress 出版社出版。

本书简体字中文版由 Apress 出版社授权机械工业出版社独家出版。未经出版者预先书面许可，不得以任何方式复制或抄袭本书的任何部分。

此版本仅限在中华人民共和国境内(不包括中国香港、台湾、澳门地区)销售发行，未经授权的本书出口将被视为违反版权法的行为。

版权所有，侵权必究。

本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2008-1860

图书在版编目（CIP）数据

Ruby 系统管理实战/汉默(Hamou, A. B.)著；仲田等译. —北京：机械工业出版社，2008.12

(Ruby 和 Rails 技术系列)

书名原文：Practical Ruby for System Administration

ISBN 978-7-111-25083-8

Ⅰ.R…　Ⅱ.①汉…　②仲…　Ⅲ.计算机网络－程序设计　Ⅳ.TP393.09

中国版本图书馆 CIP 数据核字(2008)第 138076 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：周茂辉

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2008 年 12 月第 1 版第 1 次印刷

186mm×240mm · 14 印张

标准书号：ISBN　978-7-111-25083-8

定价：32.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：(010) 68326294

译者序

几年前刚通过 Python 接触到 Ruby 时，我的第一感觉就是，和传统语言相比，Python 已经够好了，可 Ruby 比 Python 还好！它的语法简单易懂、灵活多变，而且实现了真正纯粹的面向对象——一切都是对象（例如 `1.upto(10)` 这种在传统语言中不敢想像的语法）。它在几十年来程序语言设计的前人经验积累基础上，进行了大胆的组合与创新，从量变到质变，达到了一个全新的高度。一直以来，我有一种观点：编程语言应该面向程序员，尽量为程序员提供便利，而不是为了方便机器编译，而让程序员背上不必要的学习和使用负担。Ruby 正是这样一门语言，它让我有一种感觉：我找到了！

Ruby 语言自从诞生以来，由于没有“杀手级”应用，一直默默无闻地在小范围传播，未得到广泛注意。直到 2005 年 Ruby on Rails 横空出世，世人才惊觉在耀眼夺目的 Rails 背后，有如此强大的 Ruby。原来 Ruby 可以这么用，原来 Ruby 可以这么强！可以说，没有强大灵活的 Ruby，就没有一鸣惊人的 Rails！

本书让我们看到，Ruby 在系统管理员手中，还可以得到进一步的发挥，让系统管理维护的日常繁杂事务变得如此井井有条，而且利用 Ruby 灵活而强大的语法，只需几行代码，即可快速解决问题。

值得一提的是，教授语言的大师 Bruce Eckel（写过《Thinking in C ++》、《Thinking in Java》、《Thinking in Python》等获奖名著）和面向对象设计大师 Martin Fowler（写过《Patterns of Enterprise Application Architecture》、《Refactoring》、《UML Distilled》、《Planning Extreme Programming》等获奖名著）都对 Ruby 推崇备至。自从 Ruby 出现后，他们也像常人一样从 Python 移情别恋了。

本书作者是 Ruby“狂热分子”，他在自己就职的公司，尤其是本职岗位（系统管理员）推行 Ruby，取得了良好的成果。他根据自己的亲身经历和丰富经验写成本书，其中涵盖 UNIX（和 Linux）管理员经常碰到的问题和日常例行工作，包括：

- 用单行脚本代码快速解决问题。
- 脚本的性能分析和优化。
- 利用 Ruby 强大的元编程实现领域专用语言。
- 用程序驱动和模板驱动等方式创建文件。
- 在文件和网络中实现对象的存储和读取。

- 利用 XML 和各种网络协议解析企业数据。
- 网络传输数据收集、事件监控和统计结果展示。
- 用 GEM 包扩展 Ruby 的功能。
- 用 RAKE 进行测试和文档生成。

本书译者分别为仲田、顾娟、吴畏、汪燕。其中仲田负责第 1、2、3、5、8、10 章和附录，顾娟负责第 4、6 章，吴畏负责第 11、12 章，汪燕负责第 7、9 章，最后由仲田统稿。

翻译的过程也是一种学习，通过翻译本书我才发觉自己的眼界非常狭窄，原来系统管理维护也是这么丰富多彩，其中的趣味和困难丝毫不亚于程序开发。本书让我大开眼界，希望也能让你感觉耳目一新，这样也不枉作者和译者的一番辛苦，当然，更对得起你为本书花费的人民币了。

祝你读得愉快！

此为试读，需要完整PDF请访问：www.ertongbook.com

前 言

写书应该是件相当简单的事，例如写一本与别人有关、与自己无关的书——现在我才发现，这远比我想像的困难得多。我热爱 Ruby 的优雅、简洁和强大，而且在我工作的 ISP 公司，我每天都用它来开发系统功能。你可能会想，根据我的经验和能力，提炼出几章重点突出的文字岂不是轻而易举。但并非如此。事实上，构思本书的结构框架就花了我将近一个月的时间，我这才发现，这与问题环境有关。

你瞧，本书的目标读者明摆着是系统管理员，但知道这点没什么用处，就好像为了缩小搜索范围，叫电话公司帮你接通委内瑞拉的 Bob 一样。因为，除了几个共同的特点，我们系统管理员的情况五花八门（假如能以 Slashdot 网站作为评价标准的话）。我们是“奇客”，这意味着，我们热爱技术和架构本身，并从解决问题中获得快感。我们总有太多的活，而没有足够的时间来处理。人们要求我们做各种各样的事，从找回丢失的电子邮件，到从零开始构建一个定制的 CMS 系统，而且这种事总是“昨天就要完成”，例如下面这种经常发生的事：

星期一早上 8 点 52 分，Jo 正要冲进房间，还没跑到半路她就大叫起来：“今天交易结束之前，MD 要在我们的邮件服务器上实现内容镜像，否则我们都要被起诉啦！”

在此类情景中，一个个问题冲击着你：以理智之名的 Jo 到底是谁？她怎么总是能通过安全门禁？

从工程角度审视我们的工作，快速部署的想法如此深地渗透到日常工作中，导致许多（虽然不是全部）系统管理员都草草学习了一门解释型语言。问题是，你应该选择哪一门语言？

有好几年时间，我一直使用并信任 Perl 语言，直到我转向 Ruby。我转投 Ruby 的原因可以简单地用莎士比亚的话来概括（谢谢你，威廉）。当 Bard 在谈论生活，评述一个傻瓜讲的故事时，他仿佛就在评论我编写的复杂 Perl 脚本：“充满声音和规则，但却毫无意义”。

程序应该写得很美，不会让眼睛紧张到几乎视网膜脱落。正如 Eric Raymond[⊖] 所说：“丑陋的程序犹如烂尾桥梁：它比漂亮的桥梁更容易倒塌，因为人们（尤其是工程师）对美的感知

⊖ 开源运动的创始人，世界上最伟大的黑客之一。——译者注

能力，是与处理和理解复杂性的能力紧密相关的。如果用一门语言很难写出优雅的代码，那用它也很难写出好的代码”。

简而言之，管理员需要这样一门语言，用它来思考越容易越好，简明而不晦涩，语法设计使得“正确的”做法与“快速的”做法没有两样，阅读起来好似可执行的元代码（metacode）。让我们直面事实吧——符合以上条件的只有两种主流语言：Ruby 和 Python。但对我的钱来说，虽然 Python 非常接近，但只有 Ruby 正中目标。

当我刚开始用 Ruby 时，我猜我的做法和以前不少人一样，用 Ruby 编程，而用 Perl 思考。这样可写不出好的脚本（就好比我只把香蕉顶在头上，就让你相信我是 Carmen Miranda，这是不可能的事）。在改掉因 Perl 养成的各种针对内部结构编码的坏习惯时，我真希望自己早就用上 Ruby，这样就能早点享用它的好处。

综上所述，我得在此解释一下在本书中我采取什么样的叙述方式。本书是我 6 年前就希望能得到的一本书，那时我第一次在别人那里见识了 Ruby 代码，并决定改弦易辙。本书不是权威的 Ruby 语言参考书（尽管第 1 章将假设你几乎没有编程经验，并简要介绍 Ruby 以便让你跟得上进度），它也不是一本提供 101 种方法创建 LDAP 客户端的“菜谱”书，本书没有整章描述“如何在 Linux、Windows、Solaris 和 Mac OS X 中创建用户、删除用户”之类的内容，它不“可用于微波炉”，也不能当成“救生圈”来用。

我想做的，是在非常概念化、纯理论的讨论与精心选择的示例之间取得平衡，关注重点在于管理员用 Ruby 编程所能获知的核心技术与方法。之所以采取这种方式，是因为我相信，作为奇客，我们从不读 DVD 播放机的用户手册。因为我们更喜欢在头脑中建立一个抽象播放器的通用模型，并根据经验构想某些常用按钮的样子。通过这样的组织思考，我们在处理不熟悉的系统时就比较容易适应——这也就是 Scott Adams 所谓的“诀窍”。

在这个系统管理员被索求无度的世界，你得有快速读写代码的能力和打开套接字（socket）、锁定文件、强制限定文件格式的能力。基本上，你要能迅速打开无所不包、满满当当的工具包，才能应对纷至沓来的、无法预期的挑战。我希望，本书能为你的工具包提供几个额外的扳手工具。

在我们开始之前，对于不熟悉 Ruby 的读者，下面介绍一下完全精确的 Ruby 历史。

完全精确的 Ruby 历史

在古代，生活比现在更艰辛。那时候男人是真正的男人，女人也是真正的男人，甚至寄生虫也是真正的男人。到处都是战场，庞然大物在远古的沼泽里相对嘶吼。那是一个个性的时代，是中流砥柱、钢筋铁骨、卧薪尝胆的英雄时代。

在这样的艰难困苦之中，有某种颠覆性的事物在人类心中闪耀。黑暗中有红色的光芒在闪烁，其中弥漫着大胆的承诺和欣喜的丢弃。一种变化正在到来，仿佛熟透裂开的栗子花，在风中可以嗅到它的味道。

在北极圈冻土边缘的一个小村庄，有个男孩出生在注册会计师和绳圈舞者的家庭。某个智慧老者预见此事，他出席了出生仪式，心中充满狂喜。随着 Matz 渐渐长大，一道深红色的光辉开始照耀他特别的生命。

运用从前辈那里继承的深厚魔力，他埋头苦干。算法语言被他重新改造，兼具枯燥琐碎之实与敏捷可靠之形，最终打磨出一块深红色宝石。Ruby 诞生了。

人们急匆匆地召开了充满火药味的紧急会议，讨论对付这个“暴发户”的最佳方式。很明显，它能让许多人的生活变得轻松和享受。如果工作和娱乐的界限模糊了，整个宇宙都将不再有意义。

那个时代的高级魔法师一个接一个地倒在 Ruby 语法的迷人魅力面前，开头是小心翼翼地接触，然后是兴致盎然地把玩，最终是兴高采烈地上瘾。正统阵营用铺天盖地的反宣传活动予以反击，向信徒提醒传统的重要性。Ruby 人士则回敬以反－反宣传的万炮齐发，威胁对方将一败涂地彻底崩溃。

最终，双方达成了平衡。“暴发户”登堂入室得到合法承认，其无处不在的积极影响也将得到发现。全新一代的后来者奔驰在强大的铁道上，编织着高度结构化的数据和计算能力之网。魔法学徒发现自己可以超越想像极限，做得更多、走得更远。他们再次感觉到，未来令人兴奋，一切都将不同。

作译者简介

作者简介

André Ben Hamou 1999 年进入英国帝国学院学习物理，但这事没有阻碍他对计算机技术的喜爱，他加入计算机系的系统支持组，维护麻烦不断的苹果电脑网络。在接下来的 5 年时间里，他学到了一些东西，例如怎样使用几种语言编程，怎样进行复杂性分析，为什么磁铁有两极（红极和蓝级），为什么松耦合是系统设计的重要因素，伦敦最好的鸡尾酒吧大概是哪一家，苹果电脑在多少种层面上是无与伦比的，为什么黄瓜和甩干机不能混用，怎样用磁带载入器才不会丢失附件，怎样不必大嚷大叫即可部署数据库，怎样为一个无限深势阱建立量子力学模型，以及为什么在使用榨汁机之前，必须把盖子拧紧。

当然，在所有这些发现中，最突出的是对绝妙的、令人醍醐灌顶的 Ruby 语言的喜爱，直到现在他一直沉迷其中，并在工作中运用。他目前就职于 Freedom 255 公司（英国主要的互联网服务商之一）。

André 喜欢散步、谈话，以及使用英语谈论“暴力解放”。目前他和他的想像之猫住在英国南海岸，如想和他联系，请发邮件到 andre@ bluetheta. com。

译者简介

仲田 南京某软件公司项目经理，高级程序员、系统分析员，有多年软件开发与管理经验，从事过 Delphi、J2EE、Rails 应用开发，应用领域主要是企业管理应用，包括财务、审计、商务进销存、业务管理等。目前正在研究 Ruby 语言和 Rails 框架。

电子邮件：ztian@ 163. com

技术评审者简介

■ **Dee Zsombor** 于2004年逃离了令人痛苦的花括号式编程语言的追捕，成为令人愉快的 Ruby 程序员。作为开源软件的长期倡导者，为了对自己的信念进行终极测试，他与人合伙创建了 PrimalGrasp LLC(`http://primalgrasp.com`)，一家“够辣”的软件工作坊。事实证明 PrimalGrasp 是一次美妙的体验，它遵循了“小就是美”的精神。目前 Dee 用 Ruby、JavaScript 和 Erlang 语言编程，偶尔也参与 Rails 框架开发。他喜欢爬山、读书、做图形试验，以及游泳(如果时间允许的话)。

致 谢

在帝国学院的生活改变了我的一生。我得感谢 Duncan White，他给许多管理员新手提供了依靠的肩膀，还供应英国菜园最怪异的番茄。作为向我第一时间介绍 Ruby 的引路人，我觉得自己欠 Mike Wyer 的人情，他是我见过最聪明的系统管理员之一，也是面对用户最胸有成竹的管理员之一。我还要感谢 Tim Southerwood，不仅因为他展示了作为系统管理员，即使经过多年的服务工作煎熬，还能保持幽默感和心智平衡（希望我们都能做到），还因为他爆料能把核潜艇外壳都炸掉的怪味面条。

我要感谢的其他大学伙伴包括 Adam Langley，为了那些数学题、“奇客”马拉松和他闪耀的智慧火花，以及他的巧克力蛋糕；Mark Thomas，他的正直、友谊和果断，永远值得信赖；Nick Maynard，感谢他的好心，对我说的笑话（特别是那些不好笑的）总是开怀大笑，以及帮我做了那么多次的主机状态监控而没有掐断我的脖子；Phil Willoughby，他能在上千个步骤中指出某个设计缺陷，他关于人性本恶的幽默感总是让我暗自发笑；Paul Jolly，感谢他激烈的争论和他提供的充满内部笑话的工作环境，让人从不感到压力；Ian McCubbin，他是我知道的唯一可以用身体模拟安非他明药效的人，而且一次几周，从无间断；还有 James Moody，感谢他直截了当、不拐弯抹角的做事方法，以及保护我们免受作为水平高超的 Windows 管理员（一种罕见动物）无法言说的痛苦。

在私人层面，我要感谢 Aidan Bowen，他是我见过的那种最好的老板和朋友（而且他好心地允许我在本书中引用我的日常工作为例）。我很高兴认识了 Alexi Tingey、Andrew Smith 和 Michael Allan，我对此表示感谢，因为这么好的朋友是自然界最持久的奇迹。我还得感谢我的母亲，她对我意味着太多太多，让我又敬又爱。

还要隆重感谢 Apress 出版社的杰出人士，由于你们的能力、经验和耐心，本书才得以面世。你们真是太棒了。

最后，我要向松本行弘（Yukihiro Matsumoto）表达我的谢意和崇敬，感谢他创造和培育了 Ruby 语言。Matz[⊖]给了我一个职业，一个爱好，一件投入激情的事物，还有一个杰出而执着的社区。为此，我永远心怀感激。

⊖ 松本行弘在 Ruby 社区的昵称。——译者注

此为试读，需要完整PDF请访问：www.ertongbook.com

目　录

第1章 Ruby能为你做什么

正如在前言中所说，Ruby是我的首选语言，它是我解决系统管理问题时的本能选择。再强调一次，它在我心目中有如此高的评价，是因为它：

- 易于写出可读性极高的代码。
- 强调惯例优于配置，因此一小段代码可以完成很多事情。
- 提供与C程序库的无缝接口机制。
- 其语法、扩展和执行的惯例使得“正确的”做法与“快速的”做法没有两样。
- 彻底吸收了面向对象原则，从而使极其强大的元编程（metaprogramming）技术唾手可得。

当然，这种狂热也可能是某种只说好话的大量不实赞誉的结果（对我来说，有时一千克巧克力松饼能达到这种效果）。因此，我用第1章来介绍Ruby，但愿能证明我对Ruby的所有评价都是有根有据的。另外，因为我假定绝大多数读者都有Perl背景，因此我用Perl来引起注意，让你看到与许多此类传统语言相比，Ruby在语法和逻辑上的改进之处。

另外，如果你感觉本章好像是用200mph（miles per hour，每小时英里数）的速度对Ruby语言作旋风式浏览，请不要担心——这正是我的本意。下一章将稍微回退一点，让你落到实处，看看Ruby的日常基本用法。

1.1 Hello World程序

为什么我们不直接跳到大家熟知并喜爱的例子呢？下面是在屏幕终端输出“hello world”这句话的Ruby代码：

```
$ ruby -e 'puts "hello world"'
hello world
```

正如你猜想的那样，-e标志是用来告诉Ruby解释器程序：请执行后续的任何脚本（在本例中，是指单引号中的内容）。如果想了解更多的Ruby解释器程序的命令行选项，请执行man ruby命令。用C语言编过程序的人都认识puts命令，它是“put string（输出字符串）”的简写（这里的字符串是指双引号内的所有字符）。

ri：Ruby语法参考检索工具

在我们深入探寻Ruby内置的方法、类和其他好东西之前，你有必要熟悉命令行式的Ruby语法参考检索工具：ri，它包含在Ruby标准发行版本中，是学习探索Ruby

核心库和标准库的最好方法之一。想知道 String 类有哪些功能吗？只需输入 ri String 命令即可。如果你想检索 puts，但想不起来它属于哪个模块 / 类该怎么办？只需输入 ri puts 命令，即可列出所有可能包含 puts 的模块 / 类。

如果你在找某个类，它既有某个名字的类方法(class method)，也有同名的对象方法(object method)，用 ri SomeClass. some_method 命令是不够清晰的。如果你这么做，ri 会显示警告信息，并显示所有的可选项。可以通过以下标点符号惯用法来区分：

- 对于类方法，用 SomeClass::some_ method。
- 对于对象方法，用 SomeClass#some_ method。

请务必养成这种随时查找的习惯，当我提到某个你不熟悉的新类或方法，或你对已知类和方法想了解更多时，请用 ri 检索。

现在只有少数人，才使用单行编码来完成脚本。更常见的方式是把脚本单独保存在文本文件中。因此请用你喜欢的编辑器创建一个纯文本文件，命名为 hello. rb，文件内容如下：

```
#!/usr/bin/env ruby -w
puts "hello world"
```

我假定，作为系统管理员，你熟悉第一行代码的目的，是指定用哪一个解释器程序来执行文件中的后续代码；并知道如何让这个文件成为可执行文件，即通过快捷的 chmod u +x hello. rb 命令。

-w 标志是非常重要的。不严格地说，它可以被视为"警告模式"；但更确切地说，它应该是冗长模式的指令，该指令的实际结果是显示大量警告信息(如果脚本代码不够严谨的话)。我总在脚本中启用该标志，除非用到某些写得很糟糕的库文件，导致涌出几百行警告信息时，才强制关闭该标志。(是的，这样的库文件的确存在，即便是在过分吹嘘的 Ruby 世界也有，这表明恶劣的程序员真的是无处不在。)

完成上文所述操作后，你就有了一个可执行文件，其中有一些合法的 Ruby 代码，以及代码开头的 shebang 行(也就是#! 语法格式)[⊖]，你可以把它当成真正的可执行文件来执行：

```
$ ./hello.rb
hello world
```

提示 如果你的操作系统平台不支持#! /usr/bin/env 格式，别忘记还可以用手工方式来执行脚本，即把脚本文件作为参数传递给 Ruby 解释器程序：

```
<path_to_ruby_interpreter> -w hello.rb.
```

⊖ shebang 行是指脚本的第一行，以#(sharp)和!(bang)开头，二者读音组合形成 shebang 一词。——译者注

开始庆祝吧！我们的第一个 Ruby 脚本已经写好，而且功能正常——只不过看起来没有一点用处。

还有一种执行 Ruby 代码的方式，我前面没有讲过，是交互式 Ruby shell 程序：irb，它是 Ruby 发行版本中的另一个标准部件。因为 Ruby 是解释型而非编译型语言，可以一次一行地执行，因此 irb 程序启动后显示命令提示符，你可以在此输入 Ruby 代码，按回车键即可激活解释器程序，它读取输入内容并执行代码，还以检查格式显示最后执行的结果。

```
$ irb
irb(main):001:0> word = "lobotomy"
=> "lobotomy"
irb(main):002:0> word.reverse
=> "ymotobol"
```

输入 exit 或按 Ctrl + D 键可退出解释器程序。另外，如果你的 Ruby 版本含有 readline 支持功能，你可以用上下箭头键在命令历史中循环，或按 Ctrl + A / Ctrl + E 键来跳到代码行的开头 / 末尾。如果你的 irb 不支持此种功能(像 Mac OS X 自带的 Ruby 版本那样)，你会很快发觉此问题，因为当你做这些操作时，会显示奇怪的转义字符。

在浏览本书的代码时，请务必打开一个运行 irb 的终端窗口，这样你可以随时练习，而不必每次创建、保存和执行脚本文件。准备就绪之后，我们就可以踏上 Ruby 语言的轻快旅程了。

1.2 Ruby 内幕

Ruby 是一种面向对象(OO)编程语言——而且**非常**面向对象。对宣称面向对象的其他语言有经验而又初次接触 Ruby 的人，常常对 Ruby 的 OO 血统如此纯正而感到惊讶。为了理解这句话的含义，以及它对我们的编程方式有何影响，首先必须花点时间了解什么是面向对象。

1.2.1 对象漫谈：面向对象理论

尽管每个人的经验各不相同，但我推断大多数程序员在编写 10 多个程序之后，就已经开始理解抽象的威力。简单地说，**抽象**是把某个事物看成黑盒而不必关心其内部机制。当你把棍子扔给狗，你可能不关心它的神经系统是如何运作的。对于这场游戏来说，它只是一条狗，在其滴着口水的宠物表象之下，不管有多少复杂生物机制都是无关紧要的。

只要它能对某些命令有反应，你就永远不必关心它的内部机制。在这种意义上，你对它的看法就符合 OO 设计的基本目标：**封装**。假如在某个超怪异的平行宇宙中，不直接控制它的神经系统(像某类玩偶大师一样)，它就不能玩叼物游戏，那么这个游戏就会复杂得有些可怕。但从另一方面来说，如果你能对如此打破封装游刃有余，就能让它表

此为试读，需要完整PDF请访问：www.ertongbook.com