

编程红宝书（珍藏版）



Ruby

完全自学手册

Mastering Ruby Step by Step

邓蔚 等编著

喜讯：把当当网卓越网超星读秀指针的书做成高清PDF电子书

aizihiv.taobao.com/技术传授

完整版请与店主联系！超值

本书特色

- ◎ 涵盖基础知识、核心技术、典型示例等内容
- ◎ 按照“基本概念→核心语法→典型示例”的模式讲解，容易上手
- ◎ 提供180余个典型实例、4个项目案例

超值光盘内容

- ◎ 本书源代码 + 本书视频演示 + 本书电子教案（PPT）
- ◎ 1200余页编程技术文档（免费赠送）+ 45个编程专题视频讲座（免费赠送）



机械工业出版社
China Machine Press

书山有路勤为径，学海无涯苦作舟！

Ruby

完全自学手册

本书涵盖主题

- ◎ 搭建Ruby开发环境
- ◎ 了解Ruby常用工具
- ◎ Ruby语法、基本类型及使用
- ◎ Ruby的编码处理、I/O操作及文件处理
- ◎ 使用Ruby访问各种数据库
- ◎ 使用Ruby进行网络编程
- ◎ 使用Ruby进行系统管理及图形界面程序开发
- ◎ Ruby的测试框架
- ◎ Ruby on Rails及MVC框架
- ◎ 安装Ruby on Rails
- ◎ 使用脚手架进行快速开发
- ◎ 深入介绍Rails的框架结构
- ◎ Rails中的迁移技术及Ajax应用
- ◎ 进行测试、调试和部署Rails网站
- ◎ 生命游戏模拟
- ◎ 自制Ruby编辑器
- ◎ Rails开发Digger掘客网实战
- ◎ 用Rails开发留言簿系统



投稿热线: (010) 88379604
购书热线: (010) 68995259, 68995264
读者信箱: hzjsj@hzbook.com

华章网站 <http://www.hzbook.com>

 网上购书: www.china-pub.com

封面设计: 李阳 制

上架指导: 计算机 / 程序设计

ISBN 978-7-111-25569-7



9 787111 255697

定价: 58.00 元 (附光盘)

编程红宝书（珍藏版）

Ruby

完全自学手册

Mastering Ruby Step by Step

邓蔚 等编著



机械工业出版社
China Machine Press



本书是一本完全覆盖Ruby和Ruby on Rails的完全自学手册。本书的特色是由浅入深、循序渐进，注重理论和实践的结合。虽然定位为入门手册，但是依然涉及许多高级技术和应用，覆盖到的应用领域包括系统管理、网络编程、图形处理、图形界面开发以及Web开发等开发热门领域。希望借助于平易的讲解，让读者在学习的过程中，理解Ruby的编程思想，充分享受编程的乐趣，通过本书进入Ruby开发的殿堂。同时也希望能够与各位读者分享多年来积累的Ruby程序和网站开发的经验。

本书适合准备学习或了解Ruby语言和Rails框架的各类读者阅读，并可作为开发人员的参考手册。

版权所有，侵权必究。

本书法律顾问 北京市展达律师事务所

图书在版编目 (CIP) 数据

Ruby完全自学手册 / 邓蔚等编著. —北京：机械工业出版社，2009.1
(编程红宝书)

ISBN 978-7-111-25569-7

I. R… II. 邓… 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177335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：李华君

三河市明辉印装有限公司印刷·新华书店北京发行所发行

2009年1月第1版第1次印刷

203mm × 260mm · 27.5印张

标准书号：ISBN 978-7-111-25569-7

ISBN 978-7-89482-863-7 (光盘)

定价：58.00元 (附光盘)

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



写给自学编程的人员

书山有路勤为径，学海无涯苦作舟。

——韩愈

选择一本好书，少走很多弯路。这是我们给刚开始学习程序设计的自学人员的一个忠告。当前，各类程序设计的图书琳琅满目，但在如此众多的图书中，却并不容易找到非常适合自学人员阅读的图书。究其原因，编排不科学，没有注意到自学人员的学习需求和规律是最大的问题所在。这导致很多图书都不适合这类人员阅读和学习。

为了让自学程序设计的人员能够比较容易入门和提高，我们策划了这套“编程红宝书”丛书，希望能对那些想要自学程序设计或者正在为此感到迷茫的人们有所帮助。为了让读者对这套丛书有个大体的了解，我们从丛书特色、丛书书目、给自学人员的建议等几个方面进行一个大体的介绍。

丛书特色

本丛书充分考虑了自学人员学习程序设计的需求和规律，在编写上着重体现以下特色。

1. 编排科学，讲解细致，易于学习

本丛书对内容的讲解都遵循从“基本概念→语法讲解→示例讲解”的模式，每本书安排综合案例，这非常符合自学人员的学习规律。而且，无论是对理论知识，还是实例，讲解都非常详细，很容易让读者掌握。

2. 概念准确，容易理解

概念是每个自学人员理解上的难点，也是掌握每个技术点的关键，所以必须准确。本丛书所涉及的概念都以简单的语言进行描述，必要的时候还进行类比，让人容易理解。

3. 实例丰富，强调实践

本丛书在讲解过程中穿插了大量的实例，比较实用，为以后程序开发奠定了基础。

4. 代码规范，注释丰富

为了让自学人员更加容易读懂源代码，在编排时特别注意到了代码的规范性，而且对代码进行了丰富的注释，从而让读者阅读起来没有障碍。

5. 光盘内容实用、超值

本书配套光盘提供了书中所涉及的源代码及相关操作的多媒体视频演示，以方便读者使用。除此之外，还特别免费赠送了一些相关的编程入门视频、技术文档和每本书的电子教案（PPT），以方便相

关人员学习和教学使用。

6. 提供技术支持

本丛书提供了论坛：<http://www.rzchina.net>，读者可以在上面提问交流。另外，论坛上还有一些小的教程、视频动画和各种技术文章，可帮助读者提高开发水平。

丛书书目

- | | |
|--|----------------------|
| 《ASP.NET 3.5完全自学手册》 | 《Java完全自学手册》 |
| 《Flex完全自学手册》 | 《Visual C++完全自学手册》 |
| 《ActionScript 3.0完全自学手册》 | 《Visual Basic完全自学手册》 |
| 《PHP完全自学手册》 | 《C# 3.0完全自学手册》 |
| 《PHP+Ajax完全自学手册》 | 《Ruby完全自学手册》 |
| 《Java Web 整合开发完全自学手册
——Struts+Hibernate+Spring+Eclipse》 | 《Python完全自学手册》 |
| 《Ajax完全自学手册》 | 《SQL Server完全自学手册》 |
| 《JavaScript完全自学手册》 | 《Excel VBA完全自学手册》 |
| 《CSS完全自学手册》 | 《Perl完全自学手册》 |
| 《FORTRAN完全自学手册》 | 《PHP+MySQL完全自学手册》 |

给自学编程人员的建议

- ☐ 选择一本适合自己阅读的书，这样可以让你少走很多弯路。
- ☐ 有条件的话，可以找一些志同道合的人一起学习和分享。
- ☐ 不要忽视对概念的理解。只有真正理解了概念，才能深入学习。当然，如果实在理解不了，可以放一放，先学习相关实例，再回头理解概念，可能会有好的效果。
- ☐ 不要死记语法，语法的东西用到的时候查阅即可。
- ☐ 多动手，亲自去上机实践，这样更加容易理解所学的知识。
- ☐ 遇到问题时，学会利用网络资源解决。例如，利用Google和Baidu搜索相关资料，或者在相关论坛上发帖提问，会有热心人给你答复。
- ☐ 经常阅读别人的源代码，还要养成良好的编码习惯，这会让你大大受益。

最后要说的是，自学程序设计是一个既辛苦，但又很开心的事。你会在一次次调试程序未能通过时备受煎熬，但你能享受到完成一个程序后的喜悦。无论如何，只有那些肯下功夫的人才能最后到达成功的对岸。让我们以韩愈的那句“书山有路勤为径，学海无涯苦作舟”自勉吧！

丛书策划编辑

前 言

随着科学技术的不断更新,企业IT系统也一直向着越来越复杂的趋势演进。复杂的J2EE开发模式为复杂的企业应用提供了各种各样的解决方式,但由于较高的学习成本和实现方式的复杂性,面向企业的技术无疑已成为开发者身上沉重的枷锁,同时也将一些没有很好规划的企业应用开发项目拖入泥潭。

伴随着动态语言的不断发展,动态语言体现出来的众多特性已经震撼了传统的语言和开发模式。Ruby on Rails的出现,在Java社区引起了巨大的反响,虽然争议不断,但是在Web 2.0互联网开发中硕果累累的Ruby on Rails无疑让面向企业应用的开发者看到了一线曙光。Ruby on Rails的介入简化了企业应用日益复杂化的趋势,项目复杂度的简化同时也意味着开发效率和个体生产力的提高。

同时Ruby on Rails将Ruby引入了公众的视野,让大家见识了Ruby的强大功能和独特魅力。Ruby已经成为近年发展势头最强的语言,在某种意义上代表了未来的一个发展方向。但是国内对于Ruby或Ruby on Rails的发展起步较晚,至今只有寥寥几本原创书籍,大部分都是引入的译作,与国内读者的知识水平还有一定的偏差。可以说适合国内读者的比较全面介绍Ruby的入门书籍,至今还是一个空缺。因此笔者精心编写了本书,希望借助于平易的讲解,让读者在学习的过程中,能够理解Ruby的编程思想,充分享受编程的乐趣,通过本书进入Ruby开发的殿堂。同时也希望能够与各位读者分享多年来积累的Ruby程序和网站开发的经验。

本书特色

本书基本涵盖了Ruby语言以及Rails开发框架的各个方面的知识,从Ruby语言的基本语法到各种高级特性,从Ruby开发中的字符编码问题到网络编程、I/O处理、数据库相关应用以及系统管理,从Rails开发框架的模型、控制器和视图的介绍到Ajax应用、数据库迁移技术、网站测试等都有涉及。本书由浅入深、循序渐进地介绍了Ruby和Rails的知识体系、开发思想以及常见应用。

本书的特点主要体现在以下几个方面。

- ☐ 采用了大量的实例,覆盖了Ruby中所有常用的知识和应用。
- ☐ 所有实例都采用代码、图示相结合的方式,做到了明确直观。
- ☐ 针对语言发展的趋势,本书在实际应用的基础上较为详细地介绍了Ruby的编程思想,这种思想能够让读者更好地理解Ruby并引导读者在实际应用中采用正确的思考和开发方式。
- ☐ 对每个知识点都指明了使用时的注意事项,使阅读者能够明确重点。
- ☐ 通过一个整体的实例关联了所有的Ruby相关知识,使阅读者能够从应用的角度重新理解Ruby和Ruby on Rails的各个知识点的关系,而不是单纯地对每个知识的记忆和掌握。
- ☐ 注重实际应用,所有的知识讲解都为应用服务,达到让读者阅读每一页就有实际收获的目的。
- ☐ 紧扣最新的Ruby技术,注重知识点的实用性、趣味性和扩展性。

本书的内容

本书共分为四篇，共22章，从Ruby语言的基本语法等概念讲起，再进一步介绍Ruby语言的高级特性以及在实际开发或研究中的应用。随后基于Ruby语言的知识，介绍网站开发框架Ruby on Rails的基本概念以及应用。在本书的最后一篇开发实战中通过3个开发实战，分别针对Ruby的基础知识、各种扩展库的应用、图形界面程序开发以及Rails开发等相关知识点进行了回顾和拓展。

第一篇（第1章~第4章）Ruby语言。

主要讲述Ruby语言，内容涵盖Ruby开发环境、语法、高级特性以及各种常见的问题解答。

第二篇（第5章~第10章）Ruby常用库介绍及高级应用。

主要讲述基于Ruby的常用库以及各种应用，包含中文处理、网络应用、数据库应用、系统操作以及图形、动画处理等热点应用。

第三篇（第11章~第18章）网站开发框架Ruby on Rails。

主要讲述基于Ruby语言的网站开发框架Ruby on Rails。全面覆盖了Ruby on Rails设计原理、核心技术、Ajax实现、Rails调试和部署等Ruby on Rails的应用知识。

第四篇（第19章~第22章）开发实战。

分别以模拟生命游戏演示、自制Ruby编辑器以及开发一个Digger掘客网站为例，介绍了Ruby在科研学习、程序开发和Web开发领域中的应用。在开发实战中再次复习了Ruby的相关基础知识以及各种扩展库的应用，同时介绍一些Ruby on Rails的实战技巧以及网站部署方案。

本书由浅入深，由理论到实践，采取实例和理论相结合的方式，讲解细致直观。通过一个整站制作的实例，让读者通过阅读本书，可以快速掌握Ruby语言以及使用Ruby on Rails开发网站的技术。

本书配套光盘内容

- ☐ 本书源代码；
- ☐ 本书视频演示；
- ☐ 本书电子教案（PPT）；
- ☐ 1200余页编程技术文档（免费赠送）；
- ☐ 45个编程专题视频讲座（免费赠送）。

本书适合的读者

- ☐ Ruby语言爱好者。
- ☐ 希望了解Ruby语言进行系统脚本管理的技术人员。
- ☐ Ruby开发人员。
- ☐ 使用Rails框架开发Web应用程序的开发人员。
- ☐ 希望了解敏捷开发的Web开发人员。

本书的作者

本书主要由邓蔚编著，其他参与编著和资料整理的人员有冯华君、刘博、刘燕、叶青、张军、张立娟、张艺、彭涛、徐磊、戎伟、朱毅、李佳、李玉涵、杨利润、杨春娇、武鹏、潘中强、王丹、王宁、王西莉、石淑珍、程彩虹、邵毅、郑丹丹、郑海平、顾旭光。在此对他们的辛勤工作一并表示感谢！

编者

目 录

写给自学编程的人员
前言

第一篇 Ruby语言

第1章 Ruby介绍	1
1.1 Ruby是什么	1
1.1.1 Ruby概述	1
1.1.2 初识Ruby	1
1.2 为什么选择Ruby	2
1.2.1 Ruby特性	2
1.2.2 应用领域	3
1.2.3 与其他语言的比较	3
1.3 与Ruby相关的资源	4
1.3.1 Ruby社区和网站资源	4
1.3.2 JRuby、XRuby和IronRuby介绍	5
1.3.3 Hackety Hack介绍	6
1.4 如何学习Ruby语言	8
1.5 小结	8
第2章 搭建Ruby开发环境	9
2.1 安装Ruby	9
2.2 编辑工具IDE介绍	10
2.2.1 SciTE介绍	10
2.2.2 FreeRIDE介绍	10
2.2.3 Aptana RadRails介绍	10
2.2.4 使用EditPlus / UltraEditor编辑Ruby代码	11
2.2.5 NetBeans介绍	13
2.3 相关工具介绍	14
2.3.1 irb (交互式Ruby Shell)	14
2.3.2 rdoc、ri、fxri (Ruby文档工具)	15
2.3.3 gem (Ruby包管理)	17

2.3.4 RubyScript2Exe、Exerb (Ruby代码编译成EXE)	18
2.4 小结	19
第3章 Ruby语法入门	20
3.1 概述	20
3.2 面向对象OO介绍	20
3.3 关键字、标识符和注释	21
3.4 数据和运算	22
3.4.1 常量和变量	23
3.4.2 基本类型	23
3.4.3 运算符及其优先级	23
3.5 流程控制	24
3.5.1 条件语句 (if, unless, case)	24
3.5.2 循环控制 (while, until, for, each)	26
3.6 Ruby中的OOP简介	27
3.6.1 定义类和构造函数	27
3.6.2 定义类方法和属性	28
3.6.3 访问控制	30
3.6.4 类变量和实例变量	31
3.6.5 类的扩展和继承	32
3.6.6 别名	34
3.6.7 复制、冻结对象	35
3.6.8 对象序列化	37
3.7 模块和Mixin	37
3.7.1 模块介绍	37
3.7.2 加载和混入模块 (require, load, include, extend)	39
3.8 内置类和Ruby标准库	40
3.9 动态特性	41
3.9.1 会变魔法的动态特性	41
3.9.2 动态执行代码	41
3.9.3 动态获得模块或类中的方法、常量和变量值	42
3.9.4 动态定义方法	43
3.9.5 const_missing、method_missing介绍	43
3.9.6 动态删除定义	46
3.10 代码块和迭代	47
3.10.1 代码块	47
3.10.2 yield和迭代	48
3.10.3 Proc和lambda介绍	49
3.11 异常处理	50
3.11.1 异常处理结构	50



3.11.2 抛出异常	51
3.11.3 恢复执行	51
3.12 小结	52
第4章 Ruby基本类型	53
4.1 数字	53
4.1.1 整数 (Fixnum, Bignum)	53
4.1.2 浮点数Float	54
4.1.3 Math模块	54
4.2 字符串String	55
4.2.1 字符串表示	55
4.2.2 转义符、嵌入字符串	55
4.2.3 基本操作	56
4.2.4 常用方法	57
4.2.5 字符串加密	57
4.3 符号Symbol	58
4.4 可枚举模块Enumerable	59
4.4.1 概述	59
4.4.2 遍历、搜索方法	60
4.4.3 比较、排序方法	61
4.5 区间Range	62
4.5.1 概述	62
4.5.2 常用方法	62
4.5.3 自定义区间	62
4.6 数组Array	64
4.6.1 概述	64
4.6.2 定义数组、并行赋值	64
4.6.3 运算和常用方法	65
4.6.4 查找、修改、排序以及迭代	66
4.7 散列Hash	69
4.7.1 概述	69
4.7.2 定义散列	69
4.7.3 散列常用方法	69
4.7.4 查找和迭代	70
4.8 正则表达式Regexp	71
4.8.1 正则表达式简介	71
4.8.2 定义正则表达式	72
4.8.3 查找、替换	73
4.8.4 常见示例	74
4.9 时间Time	75



4.10 小结	76
---------------	----

第二篇 Ruby常用库介绍及高级应用

第5章 Ruby的编码处理	77
5.1 乱码的缘由	77
5.2 jcode库和\$KCODE	78
5.3 编码的相互转换	79
5.4 小结	80
第6章 I/O操作和文件处理	81
6.1 简单的输入/输出	81
6.2 文件和目录操作	82
6.2.1 File和Dir类	82
6.2.2 读写文件示例	84
6.2.3 目录操作示例	86
6.3 操作XML	87
6.3.1 生成XML	88
6.3.2 解析XML	88
6.3.3 访问RSS聚合	89
6.4 YMAL库	90
6.5 操作PDF	91
6.5.1 Ruby FPDF介绍	91
6.5.2 示例：使用Ruby FPDF创建PDF文件	92
6.6 读写ZIP文件	93
6.7 图像文件处理	95
6.7.1 RMagick介绍	95
6.7.2 获取图像文件信息	96
6.7.3 生成缩略图	96
6.7.4 在图片上添加文字	97
6.7.5 特效和变换	99
6.8 小结	100
第7章 访问数据库	101
7.1 与MySQL数据库直接对话	101
7.1.1 Ruby/MySQL简介	101
7.1.2 连接MySQL数据库	101
7.1.3 更多数据库操作	102
7.1.4 Ruby/MySQL API参考	103
7.2 直接连接Oracle数据库	104
7.2.1 安装Ruby/OCI8	104



7.2.2 Oracle数据库使用示例	104
7.2.3 Ruby/OCI8 API参考	104
7.3 通用数据库接口库DBI	105
7.3.1 DBI概述	105
7.3.2 安装DBI	106
7.3.3 使用DBI访问MySQL数据库	106
7.3.4 使用DBI访问Oracle数据库	107
7.3.5 事务处理	108
7.3.6 DBI API介绍	108
7.4 小结	109
第8章 网络编程	110
8.1 NET库介绍	110
8.2 抓取网页	110
8.3 使用POP3收取邮件	113
8.4 使用SMTP发送邮件	114
8.5 访问FTP	115
8.6 小结	117
第9章 系统相关技术和创建图形界面	118
9.1 运行外部程序	118
9.2 编写命令行程序和使用ARGV变量	119
9.3 Win32 API和Win32 OLE	120
9.3.1 Win32API简介	120
9.3.2 Win32OLE简介	121
9.3.3 使用Win32OLE操作Word和Excel	121
9.3.4 使用Win32OLE控制IE浏览器	123
9.4 使用Ruby构建图形界面	124
9.5 Ruby/Tk应用介绍	125
9.5.1 下载并安装Tcl/Tk	125
9.5.2 编写Hello World	125
9.5.3 Tk部件	126
9.5.4 事件处理	129
9.5.5 Canvas画布	130
9.6 WxRuby应用介绍	131
9.6.1 下载并安装WxRuby	131
9.6.2 第一个WxRuby程序	133
9.6.3 基本控件介绍	134
9.6.4 事件处理	136
9.7 Shoes应用介绍	138
9.7.1 下载并安装Shoes	138





9.7.2 第一个Shoes程序	140
9.7.3 Stacks和Flows介绍	140
9.7.4 基本控件	142
9.7.5 绘图和动画	143
9.8 小结	145
第10章 单元测试	146
10.1 单元测试概述	146
10.2 Ruby测试框架	146
10.2.1 简介	146
10.2.2 测试流程	147
10.2.3 断言 Assert	149
10.3 测试示例: 计算两点距离	149
10.4 小结	152

第三篇 网站开发框架Ruby on Rails

第11章 Rails概述	153
11.1 什么是Rails	153
11.2 MVC简介	153
11.3 Rails的特点	154
11.4 Rails核心模块	155
11.5 Rails相关资源	156
11.6 如何学习Rails	158
11.6.1 注意Rails的特殊URL表达方法	158
11.6.2 了解Rails命名约定惯例	158
11.6.3 勤于检索Rails的文档	158
11.6.4 站在Ruby的角度学习和了解Rails	159
11.7 小结	159
第12章 应用Rails	160
12.1 安装Rails	160
12.2 配置数据库	161
12.2.1 安装并配置MySQL数据库	161
12.2.2 MySQL基本操作	162
12.3 创建Rails应用程序	164
12.4 配置Rails数据库连接	166
12.5 运行Rails应用程序	167
12.6 生成MyTest控制器	168
12.7 显示多种语言的“你好”	170
12.7.1 在数据库中创建一个表	170

12.7.2 试试scaffold	171
12.7.3 显示各种语言	174
12.8 Instant Rails介绍	175
12.8.1 安装Instant Rails	175
12.8.2 Instant Rails界面介绍	175
12.8.3 Rails程序运行测试	176
12.8.4 创建Rails应用程序	177
12.9 小结	178
第13章 深入学习Rails	179
13.1 Rails框架结构	179
13.1.1 目录结构	179
13.1.2 常用命令	181
13.1.3 命名约定	182
13.1.4 开发、测试、生产模式	183
13.1.5 Rails配置	183
13.1.6 ActiveSupport	184
13.2 ActiveRecord介绍	186
13.2.1 数据库与对象的映射关系	186
13.2.2 CRUD操作	188
13.2.3 表间关联	192
13.2.4 校验	195
13.2.5 回调	196
13.3 ActionController介绍	197
13.3.1 访问处理的流程	197
13.3.2 自定义路由	198
13.3.3 控制器中的变量	199
13.3.4 控制器的应答	201
13.3.5 过滤器	202
13.4 ActionView介绍	203
13.4.1 视图模板简介	203
13.4.2 rhtml模板	204
13.4.3 辅助方法Helper	204
13.4.4 布局模板和局部页面模板	209
13.5 Rails的插件	210
13.5.1 了解Rails的插件	210
13.5.2 可视化的HTML编辑器FCKEditor for Rails	210
13.5.3 自动发送错误邮件 Exception Notifier	211
13.5.4 活动脚手架ActiveScaffold插件	212
13.6 小结	221

第14章 迁移Migration技术	222
14.1 迁移技术概述	222
14.2 创建迁移任务	222
14.3 结构迁移	224
14.3.1 创建、重命名和删除表	224
14.3.2 添加、修改和删除字段	225
14.3.3 管理索引	226
14.4 数据迁移	227
14.5 其他迁移	228
14.6 执行迁移操作	229
14.7 小结	229
第15章 Rails中Ajax的应用	231
15.1 Ajax介绍	231
15.2 Prototype介绍	232
15.2.1 Prototype概述	232
15.2.2 实用方法介绍	232
15.2.3 Element类	234
15.2.4 Enumerable类	235
15.2.5 Ajax类	236
15.2.6 Prototype应用示例	237
15.3 Rails的辅助方法	240
15.3.1 link_to_remote方法	240
15.3.2 form_remote_tag方法	242
15.3.3 observe_field方法	244
15.3.4 periodically_call_remote方法	244
15.4 小结	245
第16章 使用ActionMailer发送电子邮件	246
16.1 ActionMailer概述	246
16.2 创建邮件发送器	247
16.3 修改发送模型和视图	249
16.4 发送邮件	250
16.5 发送HTML格式邮件	251
16.6 为邮件添加附件	253
16.7 小结	255
第17章 网站测试	256
17.1 Rails测试结构	256
17.2 模型的测试	257
17.2.1 测试模型HelloText类的CRUD方法	258
17.2.2 测试模型校验方法	260

17.2.3 测试自定义方法	261
17.3 控制器的测试	262
17.4 运行测试	265
17.5 小结	266
第18章 调试和部署	267
18.1 调试	267
18.1.1 使用console控制台	267
18.1.2 使用Logger类	269
18.1.3 在NetBeans里面进行调试	270
18.1.4 使用Firebug进行客户端调试	271
18.2 发布和部署Rails网站	273
18.2.1 常见的部署方式	273
18.2.2 安装Apache	274
18.2.3 安装Mongrel	274
18.2.4 部署配置	275
18.3 小结	277

第四篇 开发实战

第19章 模拟生命游戏	279
19.1 生命游戏介绍	279
19.2 初步设计	280
19.2.1 功能分析	280
19.2.2 界面设计	280
19.3 编写代码	281
19.3.1 界面代码	281
19.3.2 细胞演化代码	286
19.3.3 绘图及事件响应代码	288
19.3.4 完成项目代码	290
19.4 小结	299
第20章 自制Ruby编辑器	300
20.1 界面和功能分析	300
20.2 界面实现	301
20.2.1 菜单实现	301
20.2.2 文件选择Tab和编辑区域的实现	302
20.3 创建Ruby Editor项目实战	306
20.3.1 创建Ruby Editor项目	306
20.3.2 创建主程序ruby_editor.rb	307
20.3.3 创建文本编辑控件代码文件scintilla.rb	307



20.3.4 修改主程序ruby_editor.rb, 完善程序界面	309
20.3.5 运行测试	312
20.4 事件响应处理	312
20.5 功能实现	313
20.5.1 实现新建、打开和保存文件功能	313
20.5.2 查找和替换功能的实现	322
20.6 代码完善	325
20.6.1 主程序ruby_editor.rb	325
20.6.2 主界面aui_frame.rb	325
20.6.3 文本编辑器控件类scintilla.rb	330
20.7 小结	332
第21章 Digger掘客网站	333
21.1 功能及用例分析	333
21.1.1 功能分析	333
21.1.2 用例分析	335
21.2 界面与数据库设计	335
21.2.1 界面设计	335
21.2.2 数据库设计	336
21.3 开始编码	337
21.3.1 创建digger项目	337
21.3.2 数据库配置	338
21.3.3 使用acts_as_authenticated插件	339
21.3.4 创建模型	341
21.3.5 创建控制器和视图	344
21.3.6 完善界面	351
21.4 加入分类标记功能	354
21.4.1 安装acts_as_taggable插件	354
21.4.2 修改Content模型	355
21.4.3 修改相关动作和视图	356
21.5 小结	359
第22章 用Rails开发留言簿系统	360
22.1 系统分析和设计流程	360
22.1.1 浏览留言簿主界面	360
22.1.2 浏览用户留言	361
22.1.3 新建留言	361
22.1.4 管理员查看留言列表	362
22.1.5 管理员修改留言	363
22.1.6 管理员新建留言	363
22.1.7 整体设计思路	363



22.2 分析并设计数据库	364
22.3 Rails开发步骤	364
22.3.1 创建Rail Web应用项目	365
22.3.2 创建数据库和配置数据库连接	365
22.3.3 创建模型类	365
22.3.4 给模型类添加业务逻辑处理	366
22.3.5 给模型类添加验证逻辑处理	366
22.3.6 创建普通用户访问Web控制器guestbook	367
22.3.7 定义普通用户请求页面统一布局	367
22.3.8 添加逻辑处理并修改相应的视图模板完成页面输出	368
22.3.9 创建管理员访问Web控制器guestbookadmin	373
22.3.10 定义管理员请求页面统一布局	373
22.3.11 逻辑处理并修改相应的视图模板完成页面输出	373
22.3.12 测试运行	378
22.4 小结	378
附录 常用类函数	379

第一篇

Ruby语言

第1章 Ruby介绍

2004年Ruby on Rails的出现，在Web开发领域掀起了一阵敏捷开发的风暴。在人们惊讶于Ruby on Rails的简洁高效的同时，Ruby也迅速被大家所认识，一举成为了最受欢迎的十大程序设计语言之一。

ThoughtWorks的首席科学家、敏捷开发的鼻祖Martin Fowler在博客中曾写到“我的任何一个合格读者都应该知道这几年我已成了一个Ruby爱好者，简洁的语法及优雅的框架使我喜欢上Ruby，它已成为我开发个人项目的首选语言；Ruby社区（比Python社区）似乎更加友好和令人愉快——特别是对新手。”

究竟Ruby蕴含着什么样的魔力，使其能够在开发领域引起强烈的反响，甚至获得了众多世界级大师的推崇，本书将从零开始，逐步向读者展现Ruby的世界。

1.1 Ruby是什么

本小节将介绍什么是Ruby，Ruby的发展历史以及Ruby作为一门语言所具有的一些特性，并且通过一段简单的Ruby代码让读者对Ruby具有一个初步的印象。

1.1.1 Ruby概述

简单来说，Ruby是一种跨平台、面向对象的动态类型编程语言。

1995年12月，松本行弘 Yukihiko Matsumoto，混合了他喜欢的语言（Perl、Smalltalk、Eiffel、Ada和Lisp），发布了一种具有函数式及指令程序设计特性的新语言，并以发布的月份7月的诞生石（红宝石）为名，将其命名为Ruby。

Ruby是一种注重均衡的语言，它体现了表达的一致性和简单性，它不仅是一门编程语言，更是表达想法的一种简练方式。松本行弘说到“要试着让Ruby更自然，而不是简单，就像生活一样”。

虽然Ruby提倡自然简洁，却依然拥有强大的数据分析和处理能力，同时应用覆盖极其广泛。在Ruby官方网站的成功案例里面，不难发现无论在一般的网络开发、系统管理，还是在商业应用，甚至人工智能领域都能发现Ruby的身影。

1.1.2 初识Ruby

了解一门语言，最直截了当的方法莫过于直接看代码。代码1.1即为一段极其简单的Ruby代码，一



眼扫过，如果具有一点编程基础，不难理解代码所实现的功能：定义一个Robot（机器人）的类，然后初始化一个Robot的实例，调用Robot的say_hello方法，输出“Hello”，让机器人向大家问声好。

代码1.1 Ruby代码示例

```
#Robot类
#可以说话的机器人
class Robot
  #说话
  def say(words)
    puts words
  end

  #说你好!
  def say_hello
    say "你好!"
  end
end

my_robot = Robot.new
my_robot.say_hello
```

说明

如果无法理解示例中的代码也没有关系，这里只需要对Ruby的代码有个直观的印象，在看完本书后回过头来，就会发现这个示例是多么的简单了。

1.2 为什么选择Ruby

在选择学习某一门语言之前，肯定会有这样的疑问：为什么要选择这门语言和其他语言相比它有什么优势？

虽然Ruby发展的时间并不久，但它吸收了不少语言的优良特性，在很多领域中都极具竞争力。同时Ruby的简洁、动态特性以及开发的效率，是Java或者C++等主流语言不能相提并论的。

1.2.1 Ruby特性

互联网上到处都可以看到Ruby爱好者们编写的喜欢Ruby的n个理由，理由虽因人而异各有侧重，却也从侧面反映了Ruby不失为一门优秀的语言。

下面列出Ruby最具吸引力的一些特性：

- ☐ 解释性语言：无需编译即可直接运行，甚至可以进行类似于对话方式的交互执行。
- ☐ 纯面向对象语言：Ruby中所有的东西都是对象，所有的信息和代码都可以具有属性和方法，甚至数字都不例外。Ruby的这种思想贯穿于整个语言的设计之中，按照松本行弘的话来说就是要一个比 Perl 更强大、比 Python 更面向对象的语言。
- ☐ 语法简单，贴近自然语言：Ruby的语法十分简单，不需要声明变量，关键字也很少。Ruby的语句编写出来和说话用语言描述差不多，甚至有人把Ruby视为新世纪的Basic语言，应该作为青少年接触计算机的入门语言。另外Ruby编写代码的一个极为重要的原则就是自然化，让程序员的代码专注于逻辑的表述，而不是语法。

- ❑ 跨平台：在Windows、Linux、Mac系统中均可以使用。在最新版本的Red Hat Linux或Mac OS X中都已经内置了Ruby的支持。
- ❑ 动态绑定（dynamic binding）：支持动态为对象和实例添加、删除方法。这个特性为程序的扩展性、多样性提供了支持，特别是在人工智能领域具有重要作用。
- ❑ 没有多重继承：在很多语言中，多重继承往往会使程序变得更加复杂和混乱。在Ruby中采用的是模块糅合（Mix-in）的方法，使得对象能够进行功能拼装。
- ❑ 迭代器和闭包：程序的流程结构过程也可以是对象，可以重复迭代调用，大大提高了代码的复用率。
- ❑ 强大的科学计算支持：Ruby内置超长整数的支持，计算阶乘、数幂轻而易举。
- ❑ 强大的字符串处理能力：Ruby内建众多字符串的处理方法，以及强大的正则表达式支持，相比Perl而言，数据处理方面是青出于蓝而胜于蓝。

说明 Ruby还有不少特性例如智能的内存管理功能、内置多线程支持等，在这里就不一一列举了。

其实对于开发人员而言，功能并不是最重要的，Ruby最吸引、最让人称道的地方在于其编程的“乐趣”：能够让你轻松描述心中所想，编写起来自由挥洒，快速找到解决问题的方法，在改变想法的时候又能很快地调整你的程序。如果能够在编写Ruby代码的时候体会到这种乐趣，才能算是了解了这门语言的真谛。

1.2.2 应用领域

作为一种完备的基础性语言，Ruby的应用领域也非常广泛。无论编程爱好者、普通的程序开发人员、系统管理员还是科研学者，都适合学习Ruby语言。

目前Ruby常见的应用有：

- ❑ 网络应用开发，成为类似于ASP、PHP那样的Web开发语言。
- ❑ 数据处理和数据挖掘。
- ❑ 系统管理，成为类似于Perl那样的系统脚本语言。
- ❑ 人工智能和机器人领域。
- ❑ 创建领域特定语言（DSL）——专为某特定领域考虑而设计的简单、目标明确的语言。

当然Ruby也可以作为一门语言像Visual Basic一样用来开发Windows应用程序，微软已经发布了一个名为IronRuby的项目，为Ruby提供完整的.NET支持。可以预见Ruby几乎可以应用于任何常见领域。

1.2.3 与其他语言的比较

随着计算机技术的发展，计算机语言也是日新月异，层出不穷。最终能够流行并延续下来的少数语言，都有其特殊的优势或特定的应用领域。Ruby作为一门新兴的语言，汲取了前辈的精华，青出于蓝而胜于蓝，但并不意味着它就能替代一切。下面将对Ruby和其他主流的语言进行一些比较和探讨，同时也是对计算机语言发展的历史进行一次回顾，对于学习Ruby或其他语言都是极为有益的。

以前大多数人学习程序设计都是从Basic语言开始的，因为Basic比较简单，换句话说就是比较贴近人的思维，不用太多去了解计算机内在的东西。但随后有些人便把目光投向了C语言，因为它能帮助理解计算机的底层系统，提升使用算法和数据结构的能力，而且执行效率是Basic远远无法比拟的。不幸的是，在钻研完C语言后想进行一些实际的开发时，便发现开发的效率并不如想象的那么快，语言（以及



其体现的计算机系统)和现实的世界相差甚远,而且编写起来很容易犯错。

于是后来出现了面向对象(Object Oriented, OO)的语言C++和Java。OO的思想有效地将现实世界中的结构和计算机系统结构对应起来,将语言变得更加人性化,大大提升了人们开发的效率而系统架构的能力也得到了提高。C++提供了大量的强大的语言机制,同时还维持了C语言的灵巧,功能极其强大,但也带来了不少开发的陷阱,程序的好坏相当依赖于程序员水平的高低。而Java走的是另外一条路线,提倡严肃规范,在程序的开发过程中设定了严整的结构和规范,进行严格的编译检查,甚至代码编写规范。Java的这种特性有效地避免了程序员的部分错误,弥补了编程水平的差异带来的程序差异。也正因为这些特性,在工业化的开发中得到了充分的认可和应用。当然,Java必然大大束缚了开发人员的创造能力,而且大量的戒律规则也会导致系统的臃肿和开发效率的低下。

追求性能和效率的统一,看似很矛盾的需求就这样不断在推动着计算机语言的进步。有人说C语言已经在发挥机器的性能方面达到了极致,于是类似于推出OO的思想,很多语言便在人性化、发挥人脑的效率方向进行着不断的尝试。

Perl和Python这种动态语言的出现,开创了一种新的设计方式,崇尚自由灵活,也追求简单直接。抛开那些无关紧要的东西,不用声明,不用编译,拿过来就能运行就能干活。动态语言在许多领域得到了广泛的使用,例如Perl基本上已经成为UNIX下系统管理的标准脚本语言,而Python是Google内部使用最为广泛的语言之一。特别是在互联网时代,动态语言的特性更是决定了它会获得前所未有的发展。

Ruby则是诞生于众多动态语言前辈肩膀上的语言,松本行弘本身就是一个很资深的Perl程序员。相对于Perl而言,Ruby几乎吸取了Perl的所有优点,许多人甚至称Ruby是面向对象的Perl。Perl相对Ruby的优势则是在于Perl悠久的历史,强大的类库。

Ruby也吸取了不少Python的优点,有不少库和应用都直接取自Python。但是两者在设计哲学上却有着细微的不同:Python追求一种简约化,语言结构非常清晰易懂,认为简单即是美;而Ruby却站在比较实际中庸的立场,它追求自然,而不是简单。在追求简约的前提下,并不排斥对技巧的追求,往往能看到有高手用寥寥几行代码实现惊人的功能,仍不失自然,拥有犹如魔幻般的魅力。Ruby常被称为是魔幻语言,但是魔幻语言也能通过一定的约束归还简约质朴,例如Ruby on Rails。Ruby和Python代表了动态语言发展的两个方向,两者都是不可替代的,选择哪门语言,就要看自己的偏好了。

1.3 与Ruby相关的资源

本小节将介绍一些与Ruby相关的资源。这些资源包括社区网站资源,同时还有一些语言工具。了解这些资源有助于全方位了解Ruby。

1.3.1 Ruby社区和网站资源

短短数年,Ruby已经获得了迅速的发展。Ruby社区是极为活跃的群体,无论在互联网上的相关网站、论坛、博客还是在现实中的爱好者聚会,都对Ruby的传播起到了相当积极的作用。国内学习Ruby的热潮起始于2006年,在北京和上海都曾举办过相应的学习观摩的聚会。

□ Ruby语言官方网站: <http://www.ruby-lang.org/>

Ruby官方网站上提供的是关于Ruby最为权威的信息,包含了Ruby的介绍、下载、快速入门、文档、最新消息等信息。比较有特色的是里面有一个在浏览器中尝试Ruby的链接(Try Ruby in your browser),能够在浏览器里面交互执行Ruby的代码,尝试Ruby的特性。图1.1为Ruby语言官方网站的截图。

- ❑ Ruby开源项目网站: <http://www.rubyforge.org/>
- ❑ Ruby文档、教程: <http://www.ruby-doc.org/>
- ❑ 比较有名的Ruby技术介绍的网站: <http://www.rubyinside.com/>
- ❑ 国内比较有名的开发社区, 网站本身即架构于Ruby on Rails, 有非常多的文章介绍相关的应用技巧: <http://www.javaeye.com/>
- ❑ 国内比较早的Ruby和Rails社区网站: <http://chinaonrails.com/>
- ❑ Ruby on Rails中文社区论坛: <http://www.railschina.com/>



图1.1 Ruby官方网站

1.3.2 JRuby、XRuby和IronRuby介绍

Ruby的发展震撼了传统开发语言的地位, 于是乎传统的开发平台也开始接受Ruby, 希望能够将Ruby的优良特性与自身长期发展的成熟性和认可度结合起来, 获得双赢的发展。JRuby和XRuby是在Java平台下的Ruby实现, 而IronRuby则是.NET平台下的实现。

JRuby是一个100%纯Java实现的Ruby解释器。通过JRuby, 可以在JVM上直接运行Ruby程序, 调用Java的类库。目前已被Sun公司收编, 成为Java官方支持的解释器。JRuby允许现有Java开发者充分利用Ruby提供的强有力和易于使用的编程特点, 而Ruby开发者将能够自由使用庞大的曾使Java广泛地应用于各个软件开发领域的Java库来进行开发。最吸引人的应用在于Ruby on Rails正迅速成为轻量级Web应用开发方面的领头羊, 有了JRuby, Rails就有望获得Java库和JVM具有的功能、效率及业界认可度。JRuby的官方网站地址是<http://jruby.codehaus.org/>, 目前已经发布了正式的版本1.0。

XRuby是另外一个在Java平台上实现Ruby语言的项目, 功能与JRuby类似, 二者的差别主要在实现上, JRuby目前为止是一个解释器, 而XRuby是一个编译器。XRuby可以实现较高的性能, 不过目前实

展还不够成熟，还没有实际的应用。XRuby的官方网站地址是<http://xruby.com/>。

IronRuby是运行在.NET CLR之上的Ruby实现，利用它可以访问和使用.NET框架中的任何API，还可以轻松地与用任何一门.NET语言编写的代码进行互操作。譬如，可以编写一个Ruby类，在其中调用一个C#类，这个C#类进而调用一个Python类。IronRuby的网站地址是<http://www.ironruby.net/>。

1.3.3 Hackety Hack介绍

Hackety Hack是一个一站式的入门级Ruby学习软件。所谓一站式，是指Hackety Hack中不仅涵盖了Ruby的入门教程，还包含了基本的Ruby语言环境、编辑器以及一些简单的扩展库和工具，只需要安装好Hackety Hack就能够立即体验Ruby，一边学习教程一边在Hackety Hack内置的编辑器中编写和运行Ruby代码。

Hackety Hack的作者在网站上提到：在20世纪80年代，一门名为Basic的语言风靡全国，大部分的电脑中都装有它。它能够让初学者非常轻松地编写少许代码令计算机播放音乐或者画一个笑脸。现在计算机的语言虽然种类繁多，但似乎越来越专业化，很少能如此简单得让大家体会到以前的那种学习乐趣，因此Hackety Hack应运而生。

软件设计的初衷是对低龄儿童进行计算机语言的启蒙教学，因此整个软件界面都非常卡通，语言也是通俗易懂，非常适合Ruby的入门学习。可惜Hackety Hack迄今为止还没有推出中文版本，不过倒是在使用Hackety Hack学习Ruby的同时顺带练习一下自己的英语。

Hackety Hack的官方网站地址为<http://hackety-hack.net/>。安装文件的下载地址为<http://hacketyhack.net/get/HacketyHack-0.5.1.exe>，目前最新的版本为0.5.1。Hackety Hack的安装非常简单，将安装文件下载到本地计算机，运行安装文件，依次单击下一步即可完成安装。Hackety Hack安装时的界面如图1.2所示。

安装完毕后，安装程序会在开始菜单中创建Hackety Hack的程序组。运行Hackety Hack主程序，程序界面如图1.3所示。程序窗口最上方为导航按钮栏，7个按钮分别对应了不同的功能，窗口正中央显示了3个链接，分别是开始创建一个新的程序，交互式学习Ruby和获取使用帮助。单击Start a new program，就可以直接在Hackety Hack编写Ruby程序，编写完成后可以直接运行或者保存，如图1.4所示。

如果选择Learn Ruby interactively，则进入交互式学习的界面，如图1.5所示：上方会显示教程内容，而在下方则可以立即编写Ruby代码进行实践或测试。Hackety Hack中包含了7个课程，其中介绍了大量Ruby基本对象的使用方法以及示例应用。

Hackety Hack中还提供了一个对话方式交互运行Ruby语句的环境，单击工具栏中的Try Ruby按钮，即会出现如图1.6所示界面。在光标闪烁处直接输入Ruby语句并回车，语句运行的结果会立即返回并显示出来。

另外Hackety Hack中还提供了不少Ruby学习的辅助工具，单击工具栏中的See the Cheat Sheet可以打开一个简要的表格。如图1.7所示归纳了Ruby部分基本对象的基本使用方法精要，在学习的初期非常具有参考价值。图1.8显示的是Hackety Hack的帮助界面，不但包含了Hackety Hack软件的使用说明，而且附带了一个入门级的Ruby参考文档，介绍了大量的Ruby对象的使用方法和示例。如果想要稍微深入了解，不妨通读一遍。

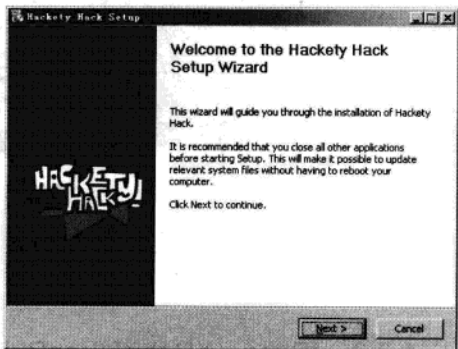


图1.2 Hackety Hack安装界面

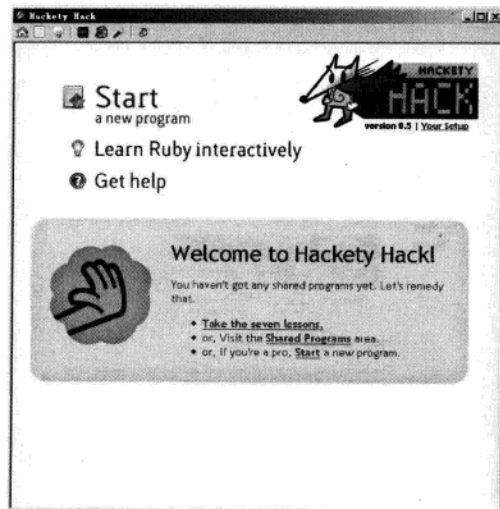


图1.3 Hackety Hack主界面



图1.4 编写Ruby代码界面

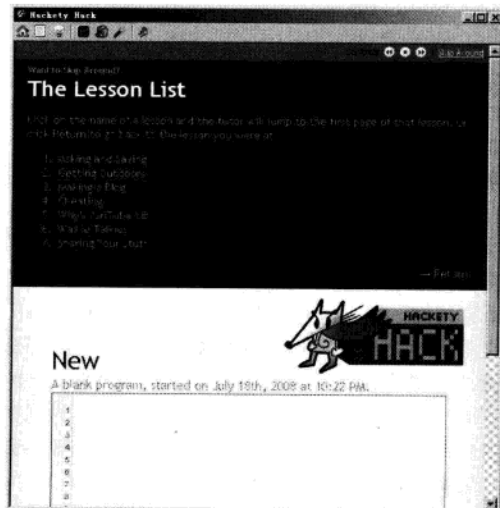


图1.5 交互式学习界面

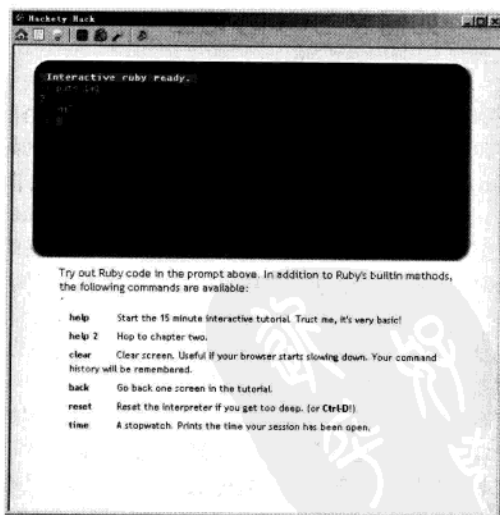


图1.6 Ruby交互运行环境

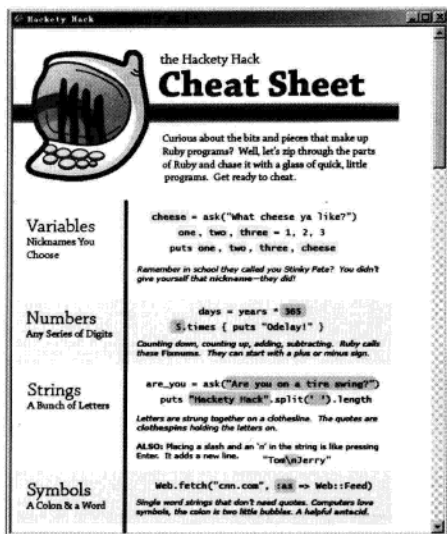


图1.7 Ruby的Cheat Sheet

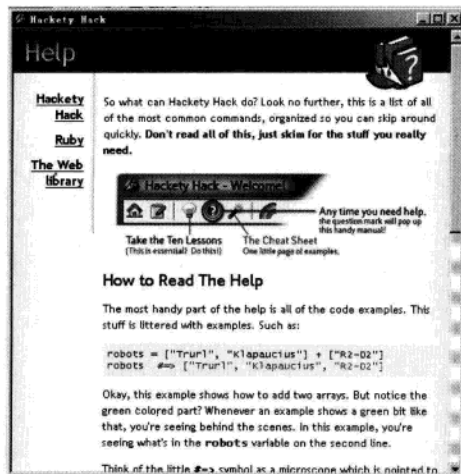


图1.8 Hackety Hack帮助界面

1.4 如何学习Ruby语言

如果具有一定的语言基础，学习Ruby的语法结构会感觉非常轻松。首先现代语言基本概念、流程控制等结构大同小异，其次Ruby本身就是一个语法非常简单的语言。唯一要注意的就是以前学习的语言和Ruby不同的地方，摆脱以前的思维惯势。例如Ruby中基本对象中的区间很多语言中都没有；同时Ruby中所有东西都是对象，数字后直接调用方法非常常见，这个在其他语言中也比较罕见。

如果没有接触过任何语言，从零开始学习Ruby语言未尝不是一件好事，本书会从基础开始，循序渐进接触各种编程的概念，再逐步涉及一些高级的技术。只要一步步踏踏实实，在阅读之后还能跟着示例进行操作，多动手多思考。其实Ruby比起其他语言而言学起来更容易。

有点特别的是：在学习Ruby过程中，除了要学习语言结构，学习如何使用Ruby实现各种功能以外，还要学习的一样东西就是Ruby的精神思想，也可以称之为Ruby之道。Ruby之道是Ruby语言设计的理念精髓所在，也是使用Ruby进行编程的使用哲学。可能称之为道有点玄乎，但这是Ruby语言贯彻始终的东西。随着时代的发展，语言会与时俱进，使用的方法会发生巨大的变化，但是设计和使用的思想在一定的范围内是不会变的。了解了这些，语言才会真正成为解决问题得心应手的工具，真正让你体会到编程的乐趣所在。关于Ruby之道，在对Ruby具有了一定的了解后，本书中还会多次涉及。

1.5 小结

本章对Ruby语言进行了一个简单的介绍，介绍了Ruby语言的特点、应用领域，简单引用了一段计算机语言的发展历程，同时将Ruby和其他常见语言进行了比较。最后列举了一些Ruby相关的网络资源和产品，并对如何较好地学习Ruby语言提出了一些建议。这一章基本上是关于语言概念方面知识的介绍，而在下一章中，将进入实战阶段，介绍如何在计算机中搭建Ruby开发环境。

第2章 搭建Ruby开发环境

在开始学习Ruby语言之前,首先需要在计算机上搭建好Ruby的开发环境。2.1节中将介绍如何安装Ruby语言支持,让计算机能够运行Ruby代码。2.2节中将介绍一些常用的编辑工具,选择合适的编辑工具往往有助于提高编写代码的效率。2.3节中将介绍Ruby的一些相关实用工具,它们也是Ruby环境中不可缺少的一部分。

2.1 安装Ruby

在Windows下安装Ruby非常简单,在Ruby的官方网站上下载Ruby的一键安装程序(One-Click Installer),最新的稳定版本是1.8.6,本书中也将使用该版本进行介绍。然后运行下载的ruby186-26.exe,安装界面如图2.1所示。

一直单击Next按钮,选择安装目录,接受默认的设置,安装即可完成。

注意 默认的安装目录是c:\ruby,不建议修改。

安装完毕以后,打开命令行窗口,在里面输入:

```
ruby -v
```

如果结果如图2.2所示,则表示Ruby安装成功,命令返回的是当前Ruby语言的版本号。



图2.1 Ruby的一键安装程序

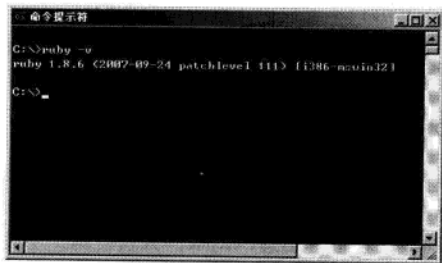


图2.2 显示已安装Ruby的版本

提示

ruby.exe是Ruby在Windows平台下的解释器,主要负责解释执行Ruby代码。它的用法通常很简单,在后面加上Ruby代码的文件名即可执行该Ruby代码文件。同时可以将一些命令行参数传递给它实现一些附加的功能,例如ruby -v或者ruby -version都会让它输出当前系统中安装的Ruby语言的版本号,使用ruby -h可以列出它的所有用法。



2.2 编辑工具IDE介绍

工欲善其事，必先利其器。使用Ruby进行开发，挑选一个适合于自己的编辑工具非常重要。下面介绍的编辑工具各有其特点，适合不同的场合使用。

2.2.1 SciTE介绍

SciTE是Ruby安装包中自带的IDE，非常小巧，具有语法加亮、自动补完（默认快捷键是Ctrl+Enter）、区块折叠等功能，非常适合编写Ruby代码以及小规模的开发。

在SciTE中，可以直接按F5键运行Ruby代码，同时在右边会显示输出的结果。图2.3为SciTE的界面截图。

SciTE的默认设置对中文的支持不是很好，把汉字当成两个字符，因此在删除和选择汉字的时候都会出现问题。解决的方法是修改SciTE的全局配置文件SciTEGlobal.properties。在菜单Options选择Open Global Options File打开全局配置文件，在文件中添加如下两行：

```
code.page=936
character.set=134
```

然后保存，重启SciTE即可。

2.2.2 FreeRIDE介绍

FreeRIDE是一个由Ruby编写的编辑器，对Ruby有着非常完善的支持，除了一般的代码着色、区块折叠等功能以外，它还内置了Ruby Doc（Ruby文档）、irb（交互式运行）以及程序调试的功能。FreeRIDE的特点是提供了较为完备的Ruby开发支持，缺点在于发展还不够成熟，对中文的支持也欠佳。

FreeRIDE的下载地址为<https://rubyforge.org/projects/freeride/>，最新的版本为0.96。FreeRIDE的界面如图2.4所示。

2.2.3 Aptana RadRails介绍

Aptana是一个基于Eclipse的定位于网络开发的IDE。早期的RadRails是一个独立的专门针对Ruby和Ruby on Rails开发的Eclipse插件，后来被包含在Aptana中，形成了一个Aptana RadRails的版本。RadRails比较适合熟悉Eclipse的人员使用。另外它是一个同时提供HTML、JavaScript、CSS格式支持，能够不断利用插件扩展的集成式IDE，因此也特别适合进行基于Ruby on Rails的网站开发。

Aptana现在分为专业版和共享版两种，专业版售价99美元，共享版是免费的。共享版下载地址为<http://www.aptna.com/studio/download/>。Aptana的界面如图2.5所示。

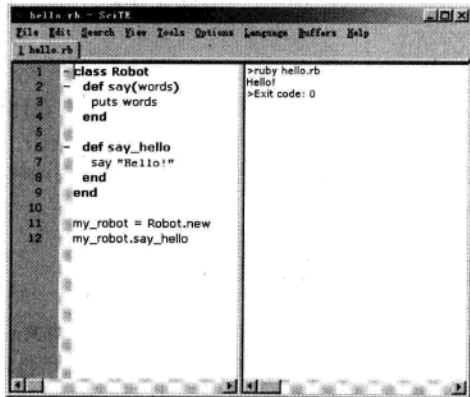


图2.3 SciTE界面

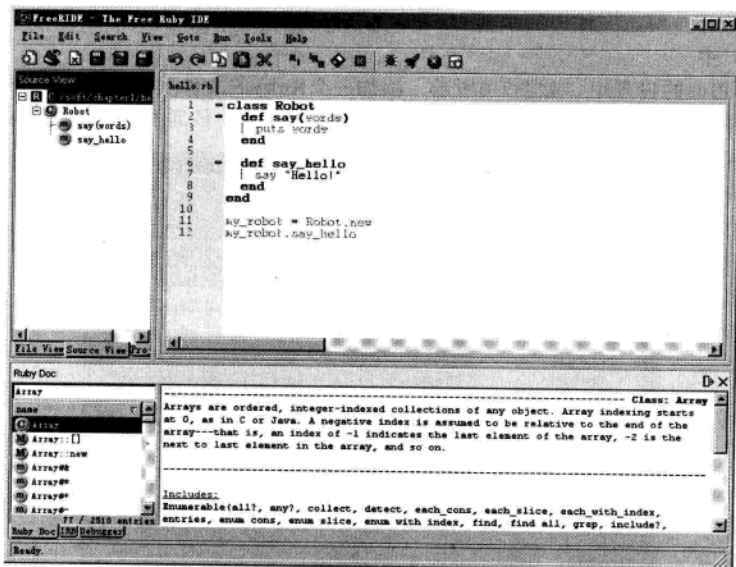


图2.4 FreeRIDE界面

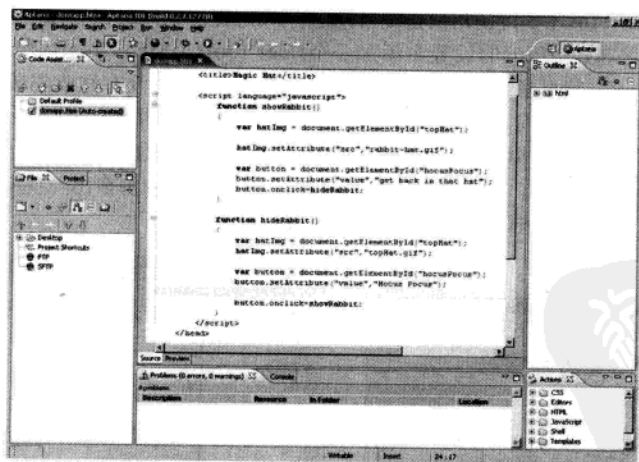


图2.5 Aptana界面截图

2.2.4 使用EditPlus / UltraEditor编辑Ruby代码

很多人都比较喜欢使用通用编辑器EditPlus或UltraEditor来编写代码，因为它们足够简单清爽，占用内存比较少，而且支持常用的代码着色、自动补完等功能。下面以EditPlus为例介绍使用EditPlus编辑Ruby代码的方法。UltraEditor的设置方法与EditPlus大同小异，这里就不做详细介绍了。

要让EditPlus提供Ruby代码的支持，首先需要在<http://www.editplus.com/others.html>上下载Ruby的

支持插件ruby2.zip, 解压文件中的ruby.stx和ruby.acp到EditPlus的安装文件夹(例如C:\Program Files\EditPlus 2)中。

然后在EditPlus的设置中, 如图2.6所示, 添加一个Ruby的文件类型支持, Syntax file和Auto completion选项中分别选择刚刚解压出来的ruby.stx和ruby.acp文件。最后单击OK按钮保存设置即可。

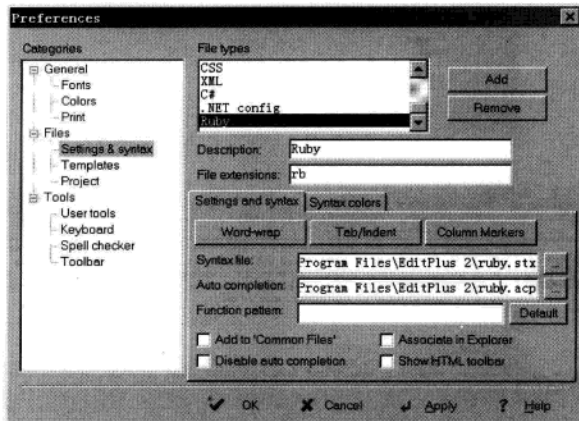


图2.6 EditPlus中Ruby支持设置

如图2.8所示, EditPlus还支持使用快捷键自动运行Ruby程序, 同时在界面中返回运行的结果。配置的方法参见图2.7: 在EditPlus的设置中添加一个User tools, Menu text中填写Run Ruby file, Command中选择Ruby安装目录中的ruby.exe, 如果在安装Ruby时没有修改安装路径应该为c:\ruby\bin\ruby.exe; Argument中填写“\$(FileName)”, 表示将当前的文件名传递给要运行的ruby.exe; 在Initial directory中填写“\$(FileDir)”, 再勾选下面的Capture output复选框, 表示需要捕获运行的输出结果, 最后单击OK按钮保存即可。设置成功后在EditPlus的Tools菜单中会多出一项Run Ruby File, 打开此菜单或者直接按下快捷键Ctrl+L, 即可运行当前编辑的Ruby程序。

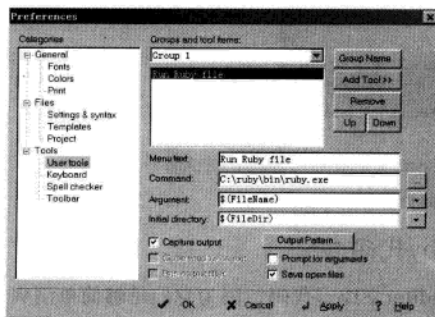


图2.7 EditPlus运行Ruby的设置

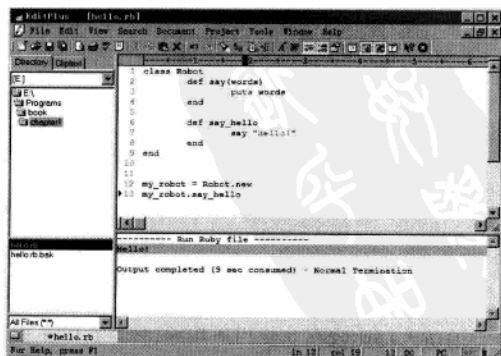


图2.8 在EditPlus中运行Ruby程序并显示运行结果

2.2.5 NetBeans介绍

NetBeans是一个广受欢迎且功能强大的Java IDE，由Sun公司在2000年开放源码。随着用户社区的增长，如今已成为行业中一个主要的IDE。在2007年，NetBeans开始将功能扩大到支持许多其他语言，特别是在Sun公司引入JRuby之后，NetBeans在最新的6.0版本中开始全面支持Ruby和Ruby on Rails的开发。

NetBeans针对Ruby有一个专门的版本。在Sun的不断努力之下，最新的6.0.1版本只有23MB，运行速度以及内存占用方面相比以前的版本有了质的提升，甚至最近还推出了官方的中文版本。相对于其他IDE，NetBeans不仅免费，而且有着Sun公司的强大后盾支持，对于Ruby的支持也是最为完美的，因此推荐下载使用，在后面也都将以NetBeans为IDE进行介绍。

在NetBeans的官方网站<http://www.netbeans.org>上下载NetBeans的最新版本，注意要选择针对Ruby的版本。下载的文件名格式类似于netbeans-6.0.1-ml-ruby-windows.exe，6.0.1为NetBeans的版本号，ruby和windows分别表示Ruby和Windows平台版本。

安装NetBeans也非常简单，照着安装程序的提示即可完成。不过安装NetBeans IDE之前必须先安装JDK 6或者JDK 5.0，这里推荐使用JDK 6以获得最佳的性能。在<http://java.sun.com/javase/downloads/index.jsp>上可以找到JDK 6的安装包下载。

图2.9所示为NetBeans界面截图，图2.10和图2.11所示分别为NetBeans的一些特殊功能的截图介绍。

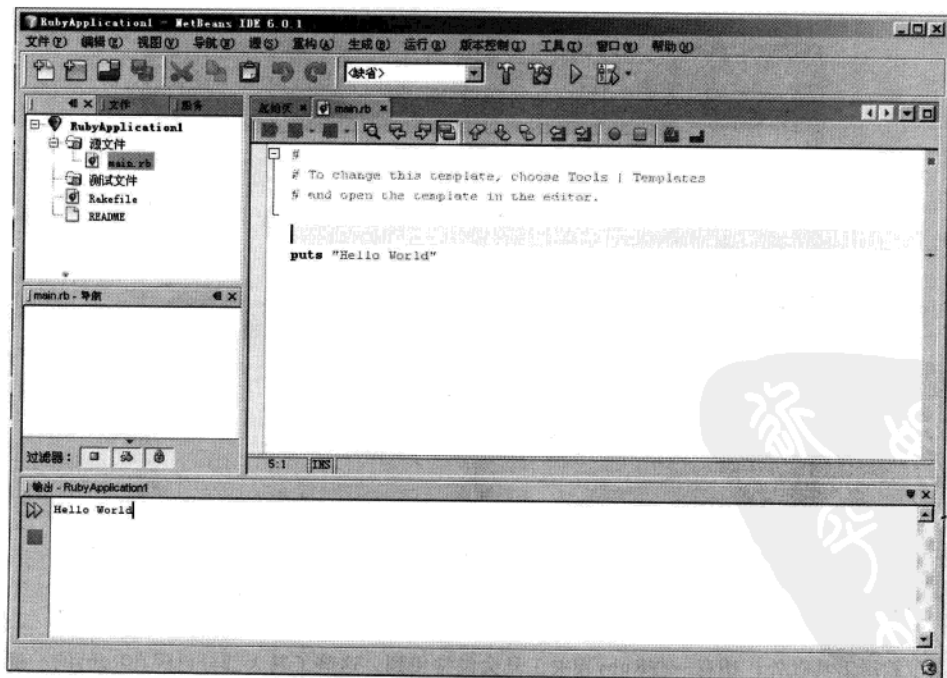
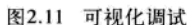
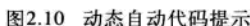
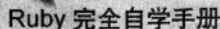


图2.9 NetBeans界面



除了编辑工具以外，还有一些Ruby相关工具会经常用到。这些工具大部分已经在Ruby的一键安装包中包含，即不需要再去下载安装了。

如果想练习使用或者简单测试Ruby，交互式Ruby程序——简称irb是最好的帮手。它提供一个命令

行式的环境，可以实时接收语句的输入并反馈显示执行的结果。

说明

完成Ruby一键安装程序后，irb就已经成功安装在计算机中了。打开命令行窗口，直接输入irb，然后按Enter键即可进入交互式的环境。

如图2.12所示，此时会出现“irb(main):001:0>”的字样，然后光标提示符在闪烁等待输入。一旦输入一个表达式，再按Enter键执行，irb就会返回并显示它的值。尝试输入1+1，然后按Enter键，irb会即时返回“=> 2”。当然输入并不仅限于表达式，任意的Ruby代码都可以输入，irb都会返回执行的结果。在图2.12中，还定义了一个say_hello方法，并调用它。

如果想要退出irb，只需要在光标处输入exit或quit然后按Enter键即可。irb是非常实用的工具，通常脑中有一个想法时，就可以打开irb来尝试验证是否可行。



图2.12 irb交互运行Ruby代码

提示

irb提供了一些参数支持定制化的配置，在irb中可以使用conf对象来进行设置。例如如果觉得提示符中的“irb(main):001:0>”字样太长，输入“conf.prompt_mode = :SIMPLE”，即可将提示符改变为简单模式；输入“conf”，可以将所有支持的参数列出。

2.3.2 rdoc、ri、fxri (Ruby文档工具)

Ruby是一个可以利用各种库不断扩充功能的语言，光Ruby安装版自带的标准库含有的方法就超过了9000个，因此Ruby社区采用了一个称为RDoc的标准体系来进行代码的文档化，描述各种库中的类和方法，以及介绍它们的使用方法。RDoc支持通过源代码自动生成文档，只需要在编写代码的过程中按照一定的格式来添加注释即可。现在，基本上所有的Ruby项目都是采用RDoc来发布文档。<http://www.ruby-doc.org>上就包含RDoc格式的完整Ruby标准库文档。

rdoc是自动生成RDoc格式文档的工具，自动分析Ruby代码生成文档，支持HTML或chm格式的输出。下面以第1章中的代码1.1为例，介绍如何为Ruby代码生成文档。

首先打开命令行窗口，将当前目录改为代码所在目录（这里假设为c:\book\chapter1，代码文件名为hello.rb），如图2.13所示，输入rdoc hello.rb，然后按Enter键，rdoc便开始生成HTML的文档。执行完后会发现rdoc创建了一个名为doc的文件夹，打开文件夹中的index.html，即可在浏览器中查看生成的文档。文档格式如图2.14所示。

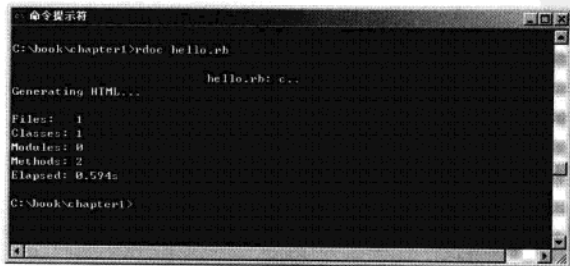


图2.13 rdoc自动生成文档

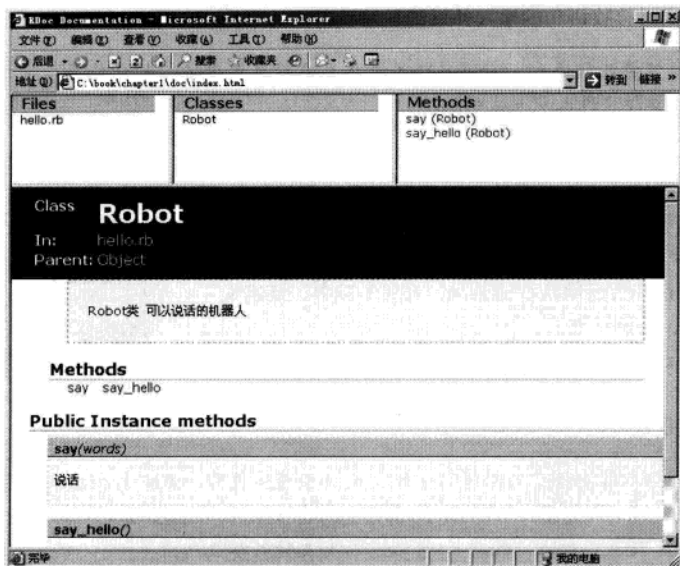


图2.14 HTML格式的RDoc文档

提示 如果需要生成chm格式的文档，可以在执行rdoc时加上-f chm的参数。chm格式的文档便于携带而且可以进行搜索，不过生成chm需要计算机中已安装好微软的HTML Help Workshop工具，没有的话需要到微软的网站上去下载。另外执行rdoc -help可以显示rdoc的所有使用方式。

ri是用于阅读RDoc格式文档的工具。它是一个命令行工具，因此能够很容易集成在其他的工具中。ri使用的方法也很简单，将需要查询的类名或者方法名作为参数传递给ri运行即可。例如在命令行窗口中输入ri String，运行部分结果如下所示，显示了String对象的介绍以及其所包含的方法：

```
C:\>ri String

----- Class: String

A +String+ object holds and manipulates an arbitrary sequence of
bytes, typically representing characters. String objects may be
created using +String::new+ or as literals.

Because of aliasing issues, users of strings should be aware of the
methods that modify the contents of a +String+ object. Typically,
methods with names ending in ``!'`` modify their receiver, while
those without a ``!'`` return a new +String+. However, there are
exceptions, such as +String#[]+=.

-----

User defined methods to be added to String.

-----
```

```
Includes:
```

```
-----
Comparable(<, <=, ==, >, >=, between?), Enumerable(all?, any?,
collect, detect, each_cons, each_slice, each_with_index, entries,
enum_cons, enum_slice, enum_with_index, find, find_all, grep,
include?, inject, inject, map, max, member?, min, partition,
reject, select, sort, sort_by, to_a, to_set, zip)
...

```

命令行形式的ri使用起来可能不大方便，因此Ruby还自带了一个名为fxri的工具。fxri是基于ri的图形化界面的文档查询工具。如图2.15所示在左上角的文本框中可以输入关键字进行搜索查询，右上部分是详细文档，右下部分是集成的irb工具，便于一边查询一边测试。

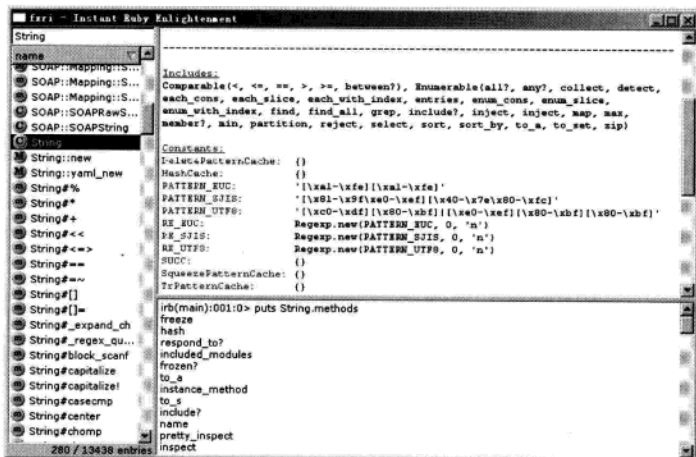


图2.15 fxri界面

2.3.3 gem (Ruby包管理)

RubyGems是一个库和程序的标准化打包以及安装框架。正如前面所述，Ruby程序可以通过各种类型的库来扩展功能，对于库的标准化也是极其重要的。RubyGems即是这样的一个框架，使得Ruby包的查找、安装、升级和卸载都变得非常容易。

通常而言，开发人员将要发布的程序或库打包到一个gem文件中，然后通过互联网或其他方式将gem文件公开。需要使用这个程序或库的人，则可以使用一个称为gem的工具在互联网上查到这个包，然后可以自动下载并安装这个包，在不需要的时候也可以随时将它卸载。gem工具同样已经包含在Ruby的一键安装包中，无需再去下载了。

下面就将以安装Ruby的zip格式支持包rubyzip为例介绍如何使用gem。首先，假设并不清楚包的名称，在命令行窗口中输入gem query -rn zip，在互联网中查找包含zip的包，返回的部分结果如下：

```
C:\book\chapter1>gem query -rn zip
```



```
*** REMOTE GEMS ***
Bulk updating Gem source index for: http://gems.rubyforge.org

gzip_filter (0.2)
  The GzipFilter GemPlugin

rubyzip (0.9.1, 0.5.12, 0.5.11, 0.5.9, 0.5.8, 0.5.7, 0.5.5)
  rubyzip is a ruby module for reading and writing zip files

rwdziparubyslippers (0.99)
  rubyslippers is frontend for the RubyGems system using rwdtinker and
  RubyWebDialogs. Requires rwdtinker >1.51
...
```

第二项就是需要的rubyzip，看它的说明是可以读写zip文件的模块，很显然就是它了。接着输入gem install rubyzip在计算机中安装rubyzip包：

```
C:\book\chapter1>gem install rubyzip
Successfully installed rubyzip-0.9.1
```

返回Successfully installed rubyzip-0.9.1，表示成功安装了rubyzip 0.9.1的版本。

输入gem list，查看计算机中已经安装的Ruby包，确实可以看到rubyzip表示安装已经成功了。

可能因为某些兼容性的原因，可能需要安装稍微低一点的版本的rubyzip，可以加上一个-v的参数，例如gem install rubyzip -v0.5.12。随后删除0.9.1的高版本：

```
C:\book\chapter1>gem uninstall rubyzip -v0.9.1
Successfully uninstalled rubyzip version 0.9.1
```

下面总结gem的常见使用方法：

- ☐ gem list：列出已经安装的包。
- ☐ gem install：安装包。
- ☐ gem uninstall：卸载安装包。
- ☐ gem query：搜索包。
- ☐ gem help：列出gem的使用方法。

2.3.4 RubyScript2Exe、Exerb（Ruby代码编译成EXE）

RubyScript2Exe和Exerb是可以把Ruby脚本转换成可脱离Ruby环境独立执行文件的工具。如果想要使用Ruby编写一些小工具发放到很多计算机上使用，就可以用上它们，不用在所有的计算机中都安装Ruby了。RubyScript2Exe同时支持创建Windows、Linux和MacOSX下可执行文件，而Exerb仅限于Windows平台的支持，不过在生成的文件大小和运行速度上略有优势。

RubyScript2Exe的网站地址为<http://www.erikveen.dds.nl/rubyscript2exe/>，最新的版本为0.5.3，下载的地址为<http://www.erikveen.dds.nl/rubyscript2exe/download/rubyscript2exe.rb>。RubyScript2Exe不需要安装，整个工具就包含在文件rubyscript2exe.rb中。使用起来也很简单，依然以代码1.1为例，将Ruby代码hello.rb编译为hello.exe。首先将下载的rubyscript2exe.rb复制到c:\book\chapter1目录下，然后执行ruby rubyscript2exe.rb hello.rb；

```
C:\book\chapter1>ruby rubyscript2exe.rb hello.rb
Tracing hello ...
你好!
Gathering files...
Copying files...
Creating hello.exe ...

C:\book\chapter1>hello.exe
你好!
```

执行hello.exe, 可以看到是和hello.rb一样的输出结果。

Exerb的网站地址为<http://exerb.sourceforge.jp/index.en.html>, 最新的版本为4.2.0, 下载的地址为<http://downloads.sourceforge.jp/exerb/25874/exerb-4.2.0.zip>。下载Exerb以后, 将压缩文件解压, 然后在命令行窗口中进入解压目录, 运行ruby setup.rb即可完成Exerb的安装。使用Exerb对Ruby代码文件编译也很简单, 如下所示输入即可:

```
C:\book\chapter1>exerb hello.rb
```

注意

唯一的不足是使用RubyScript2Exe或Exerb编译的EXE文件都比较大, 示例中RubyScript2Exe编译的EXE文件有1.5MB, 而Exerb编译的也有516KB。

迄今为止, Ruby官方还没有推出Ruby的编译器, 将Ruby代码编译成独立可执行的文件。不过Ruby的作者松本行弘已经打算在Ruby 2.0的版本中加入编译成字节码和Ruby虚拟机运行的功能, 相信到时编译的效果将会有很大的提升。

2.4 小结

本章介绍了如何在Windows平台下的计算机中搭建Ruby的开发环境(Ruby解释器的安装), 重点介绍了常见的几个Ruby编辑工具, 同时对它们的优缺点以及使用的场合进行了分析。最后一个小节还介绍了4个Ruby相关的常用小工具以及它们的使用方法。做好了这些准备工作以后, 在下面的章节将正式开始Ruby语言的学习。希望通过本章的介绍, 在后面的学习中操作和实践Ruby代码能够更加得心应手。

第3章 Ruby语法入门

刚开始学习一门新语言的时候，可能很少有人会喜欢大篇幅的语法介绍，不过这些语法的内容对于阅读或者编写代码来说又是至关重要的。因此笔者考虑再三，将语法部分压缩至一个章节，尽量使得语言的语法体系结构更加清晰，简洁明了。对于一些无法避免的琐碎细节或暂时无法理解的概念，阅读的时候可以根据情况进行略读，在实际应用中遇到再返回翻看，也许可以获得事半功倍的效果。

3.1 概述

从语法上来讲，关键字、运算符、标识符大体上构成了一门编程语言的基本定义。而数据的表现、运算以及流程控制则是算法实现的基础。本章的3.3节、3.4节、3.5节将对它们进行逐一的介绍。

Ruby是纯面向对象的语言，因此在3.2节中将首先引入对面向对象概念的介绍，带着处处是对象的观点来了解Ruby，有助于深入认识Ruby中的关键字、运算符和变量等概念。随后在3.6节、3.7节中将进一步介绍Ruby中的类、对象等技术细节。本章最后几个章节将介绍Ruby所特有的模块结构、动态特性、代码块以及迭代等高级特性。

在正式开始之前，先介绍两个Ruby的整体风格特征。

(1) Ruby是基于行的语言，一般而言遇到换行就意味着语句的结束。例外的情况一般是可以很清晰地确定语句是不是完整的。例如：行最后一个符号是操作符或者逗号，解析器则会继续阅读到下一行直到语句完成。Ruby中也可以使用分号来分割同一行中的多条语句，而在行尾加一个反斜杠表示要延续到下一行。

(2) Ruby中没有主程序，代码总是从上往下执行。在比较复杂的程序中，通常一开始会引用其他的程序模块，然后是各种类、模块定义，最后是执行的主要操作代码。

3.2 面向对象OO介绍

Ruby是纯面向对象的语言。Ruby中几乎所有的东西都是对象，代码中操作的每件东西都是对象，操作的结果也是对象。因此在全面深入介绍之前，先了解什么是对象，什么是面向对象的语言。

对象就是人们在编写代码来解决现实世界中的问题时，要进行研究的任何事物，从最简单的整数到复杂的航天飞机等均可看作对象，它不仅能表示具体的事物，还能表示抽象的规则、计划或事件。

对象具有状态，一个对象用数据值来描述它的状态（也可以称为属性）。对象还有操作，用于改变对象的状态。具有相同或相似性质的对象的抽象就是类。例如一盏台灯就可以看作是一个对象（object），灯亮、灯灭是对象的状态（state），开灯、关灯是对象的操作方法（method），而具有类似功能的物体都称作是台灯类（class），甚至灯类。

反过来，有了类，可以创建类的实例（class instance），它和前面的对象（object）是相同的概念，表示类的实体化的事物。例如特定的一盏放在卧室里的台灯就是台灯类的一个实例。

调用对象的操作方法或者对象与对象直接的沟通是通过向对象发送消息 (message) 来实现的。消息通常应该包含操作方法的名称, 以及可能需要的一些参数。

所谓的面向对象的方法其实主要是指以上面所述的方式来构建程序。在语言上来说, 至少应该提供封装 (encapsulation)、继承 (inheritance) 和多态 (polymorphism) 的功能支持。

(1) 封装表示将类的方法和属性包装成一个整体, 实现一个独立功能的模块。

(2) 继承则是一种重复使用相同代码的方法, 可以让程序员在已有的类的基础上创建新的类。例如假设已有一个灯类, 包含了灯亮灯灭的状态代码, 则可以直接创建一个台灯类继承于灯类, 自动获得灯亮灭的属性。

(3) 多态是一个比较具有争议的概念, 简单来说就是指不同的对象对相同的消息做出不同的响应的能力。很显然, 现实的世界是多样化的, 因此要求对应的程序也具有满足这种多样化的能力。还是以台灯为例, 有些台灯对应于开灯的操作是将灯的状态变为亮或者灭, 但是有些高档台灯可能会根据操作同时播放音乐。

了解了这些基本概念后, 下面回到Ruby语言中, 看看下面的代码3.1, 复习OO的基本概念。

代码3.1 OO基本概念示例

```
"I am a string".size => 13
"I am a string".index("am") => 2
-100.class => Fixnum
-100.abs => 100
my_robot.say_hello => 你好!
```

"I am a string".size返回的是字符串"I am a string"的长度。在Ruby中, "I am a string"是字符串类String的一个实例, size是一个实例方法, 调用实例方法就是在对象与方法名中间加上一个点即可。同样index也是字符串实例的一个实例方法, 它接收一个参数 (要查找的字符串), 并且返回该字符串在对象字符串中的位置。

数字在Ruby中也是对象, -100是Fixnum类的一个实例。-100.class调用了实例的class属性, 返回了实例属于的类名——Fixnum。abs则是求绝对值的方法, 返回的结果是100。最后回到第1章中的示例, 代码1.1中定义了一个Robot类, my_robot 就是Robot类的一个实例对象, my_robot.say_hello则是调用了say_hello的方法。

关于封装、继承以及多态的实现, 在后面的章节会详细介绍。

注意

代码3.1中为了表达的直观, 箭头后面直接显示的是执行的结果。前面四个语句都可以在irb中直接求值。最后一个语句来自于第1章中的代码1.1, 需要整个代码才能正确执行。

3.3 关键字、标识符和注释

所谓的关键字或称作保留字, 就是Ruby语言中有着特定意义或用处的一些字符, 不能用于变量的名字、自定义方法的名字等其他用途。Ruby的关键字非常少, 如表3.1所示。

表3.1 Ruby关键字

类 别	关 键 字
模块定义	module

类 别	关 键 字
类定义	class
方法定义	def, undef
检查类型	defined?
条件语句	if, then, else, elsif, case, when, unless
循环语句	for, in, while, until, next, break, do, redo, retry, yield
逻辑判断	not, and, or
逻辑值	true, false
空值	nil
异常处理	rescue, ensure
对象引用	super, self
块的起始结束	begin/end
嵌入模块	BEGIN, END
文件相关	__FILE__, __LINE__
方法返回	return
别名	alias

Ruby的标识符（或称为名字）用来指向常量、变量、方法、类和模块。惯例是用标识符的首字符来辅助确定标识所指向内容的作用域。基本的规则如下：

- ☐ 局部变量以小写字母或者下划线开头；
- ☐ 全局变量以美元符号\$开头；
- ☐ 实例变量以@开头；
- ☐ 类变量以@@开头；
- ☐ 常量或类名以大写字母开头。

提示

Ruby中的nil关键字非常特别。nil表示一个与其他语言中的null相类似的空集的概念。和其他语言不同的是，Ruby在逻辑判断的过程中，只有nil和false表示假，其他的所有表达式值都表示真。

Ruby的注释以#符号开始，一直到行结束为止，例如：

```
#这里是注释，下面是语句
puts "Hello word!"
```

另外，在=begin和=end两行中的代码会被解释器视为内嵌文档，内嵌文档和注释一样会被解释器忽略不进行任何处理。例如：

```
=begin
中间可以放任何文字而不会被处理
=end
```

3.4 数据和运算

数据的表示和运算是编程中最基本的操作。一般代码中直接用于当前操作的数据，都以某种结构

形式存储在内存中，然后通常使用一个标识符来引用这个数据，便于操作。换句话说讲就是给内存中的数据取一个名字，然后在代码中使用这个名字就可以使用内存中的数据，对其进行相应的操作，例如赋值（改变数据内容）、与其他数据进行运算。

3.4.1 常量和变量

常量和变量可以看作是对数据所起的名字，常量的值或变量的值指的即其所引用的数据。大多数语言中，常量意味着其引用的值是不变的，而变量则是可以改变的。

3.3节中介绍过，Ruby的惯例是常量以大写字母开头，而变量一般是以小写字母开头。这意味着和很多语言不同，Ruby中不需要任何关键字来定义常量或者变量。另外Ruby的变量和常量都只是包含数据（对象）的引用，本身没有任何的类型，所以也不需要定义变量或声明变量（所存储数据的）类型。变量或常量在第一次赋值的时候自动被声明创建，之后依然可以自由更改类型。

例如：

```
MY_CONST = 1
my_variable = 2
my_variable = "string"
```

根据规则，MY_CONST首字符大写即为一常量，它的值为1，my_variable为一变量，它的值为2，此时类型为数值，之后再赋值为一字符串，这都是允许的。

提示 Ruby中常量的值其实也可以修改，不过会有already initialized constant（已经初始化的常数）的警告提示。

3.4.2 基本类型

前面提到数据通常会以某种结构形式存储在内存中，这种结构形式就是数据类型。例如数字和字符串都是常用的数据类型之一。

Ruby的基本类型包括数字、字符串、数组、符号、散列表、区间和正则表达式。Ruby的基本类型本质上都是以类的形式来定义的。Ruby中的整数是Fixnum类或者Bignum类的对象实例，字符串是String类的对象、数组是Array类的对象等。这也是Ruby中所有东西都是对象的体现。

Ruby的基本类型基本上覆盖了常见的数据类型，字符串类和数组类提供了非常丰富的操作方法支持，符号、散列表和区间则是具有比较特殊用途的类型。能够得心应手地应用这些基本类型，是使用Ruby语言的基本功，也是编写高效率代码的基础。关于这些基本类的使用方法会在第4章中详细介绍。

3.4.3 运算符及其优先级

Ruby的运算符列举如表3.2所示（按照优先级从高到低的顺序）：

表3.2 Ruby运算符

运 算 符	说 明
[]、[]=	数组下标、数组元素赋值
**	乘幂
!、~、+、-	非、位非、一元加和一元减

运 算 符	说 明
*, /, %	乘、除、模
+, -	加和减
>>, <<	右移、左移
&	位与
^,	位异或、位或
<=, <, >, >=	比较运算符小于等于、小于、大于、大于等于
<=>, ==, ===, =~, !=, !~	各种相等判断
&&	逻辑与
	逻辑或
..., ...	区间的开始点到结束点
? :	三元条件运算符
=, %=, ~=, /=, -=, +=, =, &=, >>=, <<=, *=, &&=, =, **=	各种赋值运算符
defined?	检查是否定义
not	逻辑非
or, and	逻辑或、逻辑与
if unless while until	表达式修饰符
begin/end	Block表达式

注意

Ruby中许多运算符是由对象的方法调用来实现的。例如`1*2+3`, 1是Fixnum类的对象实例, 运算实际上是先请求1的`*`方法, 传入的参数是2这个对象, 然后方法返回的结果再调用方法`+`, 传入参数3。`1*2+3`可以表示为`(1.*(2)).+(3)`。因为任何东西都是对象, 在实际的应用中可以定义字符串类的乘法或者甚至改写数字类型的加法方法。

3.5 流程控制

程序的流程控制是编写复杂程序的基础。基本上流程控制分为两类: 一类是条件执行, 通过逻辑判断来选择要执行的特定代码; 另一类是循环执行, 依照一定的规则来重复执行一段代码。

3.5.1 条件语句 (if, unless, case)

Ruby的条件控制有3种方式: if、unless和case。Ruby中的if和其他语言中的if相似, 它的使用方法如下:

```
if 条件表达式 [then]
  ...
[elsif 条件表达式 [then]
  ... ]
[else
  ...]
end
```


大致的流程是：如果if后的条件表达式为真，则执行第一个then后面的代码，随后跳出if，执行end后面的代码；如果if后的条件表达式为假，则再判断elsif后的条件表达式，如果为真，执行then后面的代码，跳出if，如果为假，继续向下直到else，如果之前的条件表达式都为假，则执行else后的代码。代码3.2是一个简单示例。

代码3.2 if示例代码

```
if a < 10
  puts "a小于10"
elsif a < 20
  puts "a大于等于10，但小于20"
elsif a < 30
  puts "a大于等于20，但小于30"
else
  puts "a小于30"
end
```

if还有一种修饰用法，比较符合英文文法的习惯：

表达式 if 条件表达式

如果条件表达式为真，则返回前面的表达式的值，否则返回nil。

```
puts "a小于10" if a < 10
```

unless与if的使用方法类似：

```
unless 条件表达式 [then]
  ...
else
  ...
end
```

执行的流程是，如果条件表达式为假，则执行then后面的语句，否则执行else后的语句。逻辑上说，unless执行的效果和if刚好相反。unless也有类似于if的修饰用法，例如：

```
puts "a小于10" if a < 10
```

case语句是通过值的匹配判断来选择执行，使用方法如下：

```
case [表达式]
[when 表达式1 [, 表达式2] ... [then]
...]
[when 表达式3 [, 表达式4] ... [then]
...]
[else
...]
end
```

执行中，用第一个表达式的值与when后面的表达式依次进行匹配，如果匹配则执行相应的代码，执行完毕后直接跳出case结构，如果没有找到相匹配的表达式，则执行else后的代码。代码3.3是case的一个示例，判断并输出变量a的类型。



代码3.3 case示例代码

```
case a.class.name
when "String"
  puts "a为字符串"
when "Fixnum", "Bignum", "Float"
  puts "a为数字"
when "Array"
  puts "a数组"
else
  puts "a为#{a.class.name}"
end
```

3.5.2 循环控制 (while, until, for, each)

Ruby中提供了while、until、for 3个关键字实现循环控制，使用的方法都非常简洁。

while的用法如下：

```
while 条件表达式 [do]
...
end
```

当条件表达式为真时，重复执行包含的代码。

until刚好相反，循环执行到条件表达式为假时中止。代码3.4分别用while和until输出1~10，实现完全相同的效果。

代码3.4 while、until示例代码

```
i = 0
while i<=10
  puts i
  i+=1
end

i = 0
until i>10
  puts i
  i+=1
end
```

Ruby中的for语句和其他语言相差较大，Ruby中的for语句只有这样一种用法：

```
for 循环变量 in 范围表达式 [do]
...
end
```

for语句在范围表达式中不断迭代运行包含的代码。如果需要输出1~10的数字，可以这样：

```
for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
  puts i
end
```

是不是有点繁琐？其实如果使用后面要介绍的区间，代码会简单很多：

```
for i in (1..10)
  puts i
end
```

for语句实质上是通过调用范围表达式对象的each方法来循环执行的, [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] 是一个数组对象, (1..10)是一个区间对象, 代码3.5在实际应用中更常用。

代码3.5 each示例代码

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].each do |i|
  puts i
end

(1..10).each do |i|
  puts i
end
```

提示 代码3.5中, do和end包围的这种使用方法叫做代码块, 代码块的介绍请参见3.10节。

Ruby中很多对象都具有类似于each迭代的方法, 使用这些方法可以很方便地实现各种复杂的循环, 在后面的章节中会陆续介绍。

下面来介绍如何改变循环的过程。在循环的过程中, 有时可能因为一些特殊原因需要跳出或者重新开始循环。Ruby提供了break、redo、next和retry这样几个关键字来改变循环的处理过程。

break用于直接跳出循环体, next用于直接跳到下一个循环周期, redo用于重复当前循环周期, retry用于重复整个迭代循环操作。代码3.6综合演示了改变循环过程的操作。

代码3.6 改变循环过程

```
#只输出1至5, 当i>=5时跳出整个循环
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].each do |i|
  puts i
  break if i>=5
end

#输出1至10中的奇数, 当i为偶数时直接跳到下一个循环
(1..10).each do |i|
  next if i%2==0
  puts i
end
```

3.6 Ruby中的OOP简介

前面3.2节中已经对面向对象的基本概念进行了介绍, 下面来讨论Ruby中如何具体实现OOP。

3.6.1 定义类和构造函数

Ruby编程处处都与对象打交道, 因此定义类和方法等操作也是非常的简单便捷。Ruby中定义类只需要使用class关键字:



```
class 类名
  def 方法名
    ...
  end
  ...
end
```

说明 遵循Ruby的惯例，类名必须保持第一个字母大写，类中的方法名都是小写。

Ruby中类也是对象，是Class类的实例，因此自己创建的类都会自动获得Class类的所有实例方法。

代码3.7 类继承关系

```
class MyClass
end

puts MyClass.class
puts MyClass.class.superclass
puts MyClass.class.superclass.superclass
puts MyClass.class.superclass.superclass.superclass
```

代码3.7中，定义了一个MyClass的类，然后调用superclass方法不断输出MyClass上层所继承的类名。输出的结果为：

```
Class
Module
Object
nil
```

可以看出，其中的继承关系如下：

MyClass <= Class <= Module <= Object。Object类的superclass为nil，表示Object类之上没有任何父类了。Object类是Ruby中所有类的父类。这种继承关系有点类似于“一生二，二生三，三生万物”的道理。关于继承的问题在3.6.5节中将有专门讨论。

创建一个类的对象实例是通过继承自Class的new方法来完成的。例如：

```
my_class = MyClass.new
```

Ruby类中没有类似于C++或Java中的专门的构造函数，但是Class的new方法会尝试自动调用MyClass中的initialize方法，并将调用new方法时传递的参数传递给initialize方法。如果MyClass中定义了new方法，那么Class中的new方法会被自动覆盖，可能就无法成功创建对象了。

3.6.2 定义类方法和属性

前面已经看到，定义类方法只需要def关键字即可完成。首先看代码3.8。

代码3.8 类方法、实例方法

```
class MyClass
  #定义类方法
  def self.method1
    puts "method 1"
  end
end
```

```

end
#另外一种方式定义类方法
def MyClass.method2
  puts "method 2"
end

#实例方法
def method3
  puts "method 3"
end

my_class = MyClass.new
my_class.method3

MyClass.method1
MyClass.method2
#method3为对象实例方法，下面语句会报错
MyClass.method3

```

下面是代码运行的结果：

```

method 3
method 1
method 2
undefined method 'method3' for MyClass:Class (NoMethodError)

```

可以看出，method1和method2是MyClass的方法，而method3是my_class的方法。这两种方法在定义上和使用上都是有区别的。

再看看属性的定义方法。定义属性有4个方法：attr、attr_reader、attr_writer和attr_accessor，它们的作用分别是定义只读（或可读写）属性、只读属性、只写属性和可读写属性。注意它们并不是关键字，而是Module类提供的方法。

attr在缺省情况下与attr_reader作用相同，不过attr能通过一个附加的参数设置属性是否是可写的。attr的使用方法如下：

```
attr(symbol, writable=false)
```

其第一个参数为属性名，第二个参数则指定属性是否可写。例如attr :attr1与attr_reader :attr1等价，而attr(:attr1, true)与attr_accessor :attr1等价。

本质上那4个方法都是通过动态定义方法来定义属性，代码3.9中两个方法是等价的。

代码3.9 定义属性

```

class MyClass
  #使用attr_accessor定义attr1属性
  attr_accessor :attr1

  #使用方法定义attr2属性
  def attr2
    @attr2
  end
end

```



```
end

def attr2=(value)
  @attr2 = value
end

end

my_class = MyClass.new
my_class.attr1 = "attr1"
puts my_class.attr1

my_class.attr2 = "attr2"
puts my_class.attr2
```

3.6.3 访问控制

在设计类的方法和属性的时候，可能会有一些仅在内部使用，不想公开暴露在外界的方法。Ruby 中提供了三种级别的设置来控制对方法的访问和使用。

这三种级别分别是Public（公有）、Protected（保护）和Private（私有）。

☐ Public级别方法可以被任何人调用，没有任何的访问限制。通常默认的级别就是公有的。

☐ Protected级别的方法只有被定义该方法的类或者其子类的对象所调用。

☐ 而Private和Protected类似，区别在于Private方法不能被明确地接收者调用，它只能被self调用。

设置访问控制的级别使用的是public、protected和private这三个函数。一般将它们放置在想要设置的方法定义的前一行。使用方法参见代码3.10。

代码3.10 访问控制示例

```
class MyClass
  #默认是public
  def public_method
    puts "public_method"
  end

  protected
  def protected_method
    puts "protected_method"
  end

  private
  def private_method
    puts "private_method"
  end

  public
  def public_method1
    puts "public_method1"
  end
end
```

```

class MySubClass < MyClass
  #子类可以在内部使用父类中protected和private的方法
  def call_protected_method
    protected_method
  end

  def call_private_method
    private_method
  end
end

my_class = MyClass.new
my_class.public_method
my_class.public_method1

my_sub_class = MySubClass.new
my_sub_class.call_protected_method
my_sub_class.call_private_method
#my_class.protected_method      #出错
#my_class.private_method        #出错

```

Private和Protected级别的差异参见代码3.11。

代码3.11 Private与Protected差异示例

```

class MyClass
  def initialize(name)
    @name = name
  end

  def compare(c)
    c.name == @name
  end

  protected #换成private即会报错
  def name
    @name
  end
end

a = MyClass.new("a")
b = MyClass.new("b")
puts a.compare(b)

```

如果name方法换成了Private级别，那么在a引用的实例中，将没有权限访问b引用的实例中的name方法。

3.6.4 类变量和实例变量

与类方法和实例方法类似，类中的变量也有类变量和实例变量之分。类变量以@@开头，在类对象层次共享，而实例变量以@开头，它的生存空间仅限于实例对象中，其他相同类的实例对象无法访问。