Broadview
www.broadview.com.cn

Regular Expression
yet another introduction

正则指引（第2版）

余晟 著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Regular Expression
yet another introduction

正则指引（第2版）

余晟 著

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

内 容 简 介

本书综合作者自己遇到的实际问题，以及其他开发人员咨询的问题，总结出一套巧妙运用正则表达式的办法，并通过具体的例子指导读者拆解、分析问题。全书分为三部分：第一部分主要讲解正则表达式的基础知识，涵盖了正则表达式中常见的各种功能和结构；第二部分主要讲解关于正则表达式的更深入的知识，详细探讨了编码问题、匹配原理、解题思路；第三部分将之前介绍的各种知识落实到常用语言.NET、Java、JavaScript、PHP、Python、Ruby、Objective-C、Golang 中，在详细介绍了在这些语言中正则表达式的具体用法之外，还辨析了版本之间的细微差异。本书既可以作为专门的学习用书，也可以作为备查的参考手册。

本书适合经常需要进行文本处理（比如日志分析或网络运维）的技术人员、熟悉常用开发语言的程序员，以及已经对正则表达式有一定了解的读者阅读。

图书在版编目（CIP）数据

正则指引/余晟著. —2 版. —北京：电子工业出版社，2018.11
ISBN 978-7-121-35130-3

Ⅰ. ①正… Ⅱ. ①余… Ⅲ. ①正则表达式 Ⅳ.①TP301.2

中国版本图书馆 CIP 数据核字（2018）第 224858 号

策划编辑：张春雨
责任编辑：刘　舫
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
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
开　　本：787×980　1/16　印张：24.5　字数：577 千字
版　　次：2012 年 5 月第 1 版
　　2018 年 11 月第 2 版
印　　次：2018 年 11 月第 1 次印刷
定　　价：89.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

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。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

引子：关于正则表达式……

正则表达式这个名字看起来总有点古怪，概念似乎也不简单，甚至需要用一整本书来讲解。可是，它到底是什么呢？

同为技术人员，我相信你总会与字符串打交道，相应的，各种语言也都提供了与字符串有关的函数。我们先看下面几个问题，用字符串函数是如何解决的（下面的代码使用 Python 语言，它很直观，正文里有基础的介绍。现在，你只需要知道 def 是定义函数的关键词即可）。

引入正则表达式

1. 判断字符 ch 是否是数字字符

```
def isDigit(ch) :
    return ch == "0" or ch == "1" …… or ch == "9"
```

2. 判断字符串 str 是否是电话号码（为简单起见，现在只考虑固定电话号码，也就是长度在 7~8 位之间的数字字符串，且第一位不为 0）

```
def isPhoneNum (str) :
    if len(str) >= 7 and len(str) <= 8 and str[0] != "0" :
        for ch in str :
            if not isDigit(ch) :
                return false
        return true
    return false
```

任务的复杂度并没有增加太多，程序的复杂度增加了很多倍；如果你不同意，那么，来一个更复杂的。

3. 找出一段文本中所有的电话号码

最直接的办法是，在字符串中的每个位置截取 7~8 个字符，调用之前的 `isPhoneNum()`。这么做看起来没问题，只是效率太低。

当然，做点改进也不难，加上一个前置条件，只在“当前字符为数字字符”的情况下调用

isPhoneNum()。这样效率倒是改进了，但是还有问题没有解决：要求找到的是长度大于等于 7 个字符、小于等于 8 个字符的“数字字符串”，而不是“子字符串”——也就是说，假如数字字符串是 64240000，需要将它找出来；如果数字字符串是 13800138000，则需要忽略它，以及其中的任何子串（比如 13800138、00138000）。

所以，用 isPhoneNum()找出字符串之后，还需要保证它之前的字符不是数字字符，之后的字符也不是数字字符。看起来很简单，但合格的程序员一定要考虑边界问题，避免越界错误：如果当前字符是整段文本的第一个字符，则不需要判断之前的字符，因为它不存在；同样，如果找出的字符串在整段文本的末尾，则不需要判断之后的字符，因为它同样不存在……

到现在为止，即便只是找到最简单的固定电话号码，程序也非常复杂，难以维护。如果要查找的是形式更多变的文本，比如带区号的电话号码（021-64240000 或者 03718888888）、手机号码（13800138000、+8613800138000 或者 013800138000），程序更是不可想象，更不用说文件路径名、URL 地址、电子邮件地址了！然而，日常开发中我们又确实经常需要面对这类任务，有什么更好的办法呢？

正则表达式就是解决这类问题的万能药。虽然许多人有点看不起它，觉得不入流，一些科班教材里也不会花太多篇幅来介绍它，但它确实是解决问题的利器——之前提到的三个例子，用正则表达式都可以轻松解决。

引入正则表达式之后

1. 判断字符 ch 是否是数字字符

```
def isDigit(ch) :
    return re.search(ch, "[0-9]") != None
```

看起来很复杂，其实并不复杂：这里真正要关心的就是正则表达式[0-9]，它表示“从 0 到 9 之间的任意字符”，很形象吧？re.search()是正则表达式运算函数，它判断 ch 能否由正则表达式[0-9]匹配，可以则返回一个结果，否则返回 None（这些细节正文中会讲到）。

2. 判断字符串 str 是否是电话号码

```
def isPhoneNum(str) :
    return re.search(str, "[1-9][0-9]{6,7}") != None
```

这个正则表达式最开始是[1-9]，表示第一个字符必须是 1~9 之间的数字字符；之后是[0-9]{6,7}，表示长度在 6 和 7 之间，由 0~9 之间的数字字符组成的字符串（两部分加起来，整个字符串的长度在 7 和 8 之间）。要解决的问题复杂了，正则表达式仍然直观形象。

3. 找出一段文本中所有的固定电话号码

```
def findNumStr(str) :
    return re.findall(str, '(?<![0-9])[1-9][0-9]{6,7}(?![0-9])')
```

这个正则表达式之前多出了`(?<![0-9])`，表示“之前不能是[0-9]”；之后多出了`(?![0-9])`，表示“之后不能是[0-9]”。虽然稍微复杂点，但意思明确，而且不难理解。`re.findall()`的意思也很明显：找到所有这样的字符串。

可以想象，循着这种思路，查找更复杂的电话号码、手机号码等任务都不难解决。更重要的是，之前需要许多行语句才能完成的任务，现在基本上只需要一个正则表达式、一条语句就可以完成。正因为如此，不少人虽然认为正则表达式不够花哨、漂亮，却不得不承认它是一种“匕首应用”——匕首，没有十八般兵刃那么气派，关键时候却不可或缺，所以值得花时间练练。同样，正则表达式虽然不能用来显摆，但总有派得上用场的地方，花时间练练绝不是坏事。即便你的工作不是纯粹的文本处理（比如日志分析），也总会有用到正则表达式的地方（比如查找和修改源代码），所以我希望，这本书能陪伴你练出一身正则表达式的好功夫，在关键场合能亮出称手的工具。

最后，为了传承经典教科书的良好习惯，附上正则表达式的“科班史”。

正则表达式发源于与计算机密切相关的两个领域：计算理论和形式语言。20 世纪 40 年代，两位神经生理学家 Warren McCulloch 和 Walter Pitts 发明了一种用数学方式来描述神经网络的办法，他们把神经系统中的神经元描述成小而简单的自动控制单元。1956 年，数学家 Stephen Cole Kleene 在他们研究的基础上，发表了一篇名为《神经网事件的表示法》的论文，在其中，他采用了一些称之为“正则集合（regular set）”的数学符号来描述神经网络模型。

之后，UNIX 的主要发明人 Ken Thompson 将这个符号系统引入了文本编辑器 QED（意思是“在文本中搜索某种模式”），正则表达式由此也进入了计算机世界。随后 Ken Thompson 又将正则表达式引入了 UNIX 下的文本编辑器 ed，ed 最终演化为大家熟悉的 grep（grep 得名自 ed 编辑器中的正则表达式搜索命令 g/re/p，其中的 re 表示“正则表达式”）。

返璞归真——评《正则指引》

第一次接触正则表达式，是 2000 年我在西安一家公司使用 Perl 做网站开发时。之前我在工作中只使用过标准的 C 语言，Perl 这门编程语言的强大表达能力，令我印象极为深刻。Perl 的力量，除了语言本身的设计之外，很大程度上来自它对正则表达式的完美支持。当时我们开发了一个网上商城的应用，允许很多商家在这里开店，可以选择一些不同的样式模板。我很快发现，使用 Perl+正则表达式是开发这类应用的利器。我们只花了大约一个月的时间，就完成了网站核心功能的开发。那时候我意识到，使用正则表达式是聪明人写程序的方法（没说我是聪明人，但是我非常希望与那些聪明人为伍），可以极大地提高代码的重用度和执行效率。如果完全不使用正则表达式，代码量会增加数倍甚至数十倍。

后来因为一些原因，我告别了 Perl。在之后的工作中，我使用过 Java、JavaScript、Ruby 等编程语言。我发现这些语言对于正则表达式的支持，没有一个能够超越 Perl。Java 这种所谓的“工业主流编程语言”，一直到 2002 年 JDK 1.4 推出时，才正式把对正则表达式的支持加入核心类库。因为长期缺乏对正则表达式的原生支持，以及语言本身表达能力欠缺，使用 Java 来做大量的文本处理，感觉非常笨拙，完全没有使用 Perl 那种指哪打哪的快感。直到 2007 年我发现了另一个更好的 Perl 语言——Ruby，才重新找回了 2000 年 Perl 带给我的编程快感。

因为我的工作主要是做 Web 开发，大量的时间花在与 HTML/CSS/JavaScript 以及关系数据库打交道上。在这里并没有很高深的算法，只有大量繁重的文本处理。难以想象，如果没有正则表达式，我们的开发将会是何等原始。

除了 Web 开发领域，需要实现大量自动化功能的一些领域，例如运维领域和自动化测试领域，也是正则表达式大显身手的地方。无论使用稍显简陋的 sed/awk 还是更高级的 Perl/Python/Ruby，实现自动化功能，都必须依赖大量的正则表达式。

自从面向对象的编辑方式时髦起来之后，甚至一度出现了面向对象万能论，有人试图用 MDA 和可执行的 UML 来解决一切编程问题。但是我一直认为面向对象只解决了软件开发的一小部分问题，而且是宏观方面的问题。正则表达式解决的问题，是面向对象无能为力的一些微观方面的问题。在这里不需要坐而论道的方法论争论，需要的是刺刀见红的肉搏战。这些问题即使使用完全面向对象的方式能够解决，也会是很笨拙的。如果用物理学来比喻，面向对象是“广义相对论”，

而正则表达式则是“量子力学”。

正则表达式已经成为现代编程语言的基础模块，现在很难找到一种不支持正则表达式的编程语言。除了编程语言外，在很多工具软件，例如文本编辑器（Vi、Emacs、UltraEdit）、Web 服务器（Apache、Nginx）中都能找到正则表达式的身影。

余晟老师是我的朋友，我对他印象最为深刻的是他对于技术工作的严谨态度。“格物致知”是中国传统儒家学派所追求的一种道德修养，也是一种境界。余老师是我的朋友中最接近“格物致知”这种境界的一位。我虽然从未精通过任何一门技术，但是很喜欢结交余老师这样的朋友。

余老师潜心编著的这本《正则指引》深入浅出，将正则表达式的由来和分支娓娓道来。阅读这本书，我仿佛回到了 11 年前做 Perl 程序员时的快乐时光。国内很多程序员的一个通病是好高骛远，像《正则指引》这样一本详细讲解基础知识的书未必会有很好的销路，但是等你做过很多年开发之后，你会发现，对你最有价值的，正是这些基础知识和工具。软件开发的“道”，正是隐藏在这些看起来不起眼的基础知识和工具之中的。

李锟

2011 年 11 月 25 日

克制我们内心的冲动

《正则指引》（第 2 版）就要出版了，按说这是一条好消息。在这条好消息面前，我更想做的是克制自己内心的冲动，静下心来讲讲这本书初版以来的故事。

《正则指引》初级刚出版的时候，我一度认为，自己和正则表达式的缘分到此为止了。如果说翻译《精通正则表达式》之后还有许多遗憾，比如某些讲解方式不符合中国程序员的思维，以及过份关注英文，所以关于东亚文字处理的知识无从寻找，经验无从分享。那么写作《正则指引》，就是弥补这种缺憾的绝好机会。《正则指引》面世之后，这些缺憾已经悉数补上了。

令我没有想到的是，《正则指引》自 2012 年出版以来，不断有读者向我反馈问题。除去最早一两年密集热烈的反馈，后续的反馈如涓涓细流绵绵不绝。而我一度想当然地认为勘误已经完整了，所以一直没检查勘误邮箱。直到一年前读者在微信公众号后台给我留言，详细列明勘误意见之外，毫不留情地指责“对自己的作品不负责，长期不回复读者意见”。这封信让我惭愧不已。在软件开发中，“发布了就不管”是很不负责的，在技术书籍的写作中，“出版了两年就不回复读者意见”，同样是很不负责的。

所以，我必须克制自己内心“年代久远，不值得继续打理”的偷懒冲动。文责自负，完整的说法应该是“文责终身自负”。

本次《正则指引》（第 2 版）的出版，对我而言是全新的补过机会，可以“一次性”回复迄今为止所有的读者意见。当然，新增的 Objective-C、Golang 等章节，尽管已经找熟悉的朋友审读过，但我可以肯定，它们必定不是完美无瑕的，未来仍然需要坦然面对广大读者持续的批评指正。哪怕有些批评指正的语气不那么让人舒服，甚至“看不到几分善意”，我仍然需要克制自己内心“反唇相讥”的冲动，认清事实，撇开情绪，虚心面对。

同时，我也希望读者在阅读这本书时，能克制自己内心的冲动。

我希望大家克制的第一重冲动，是浅尝辄止——“正则表达式这玩意儿，要用时翻翻就好，没必要深究”。正则表达式已经诞生很多年了，以今天的标准来看，它的语法和结构相当粗陋，不幸的是，它的内部逻辑又相当复杂。有些朋友会问我一些“怎么看也看不懂”的正则表达式，坦白地说，我也要反复琢磨才能看懂。所以，尽管这本书提供了若干“速查”资料，但我还是建

议读者能耐下心来，至少通读一遍。正则表达式有点像游泳，学会了就不会忘，用的时候自然能想起来。否则，你永远只能在岸边扑腾，离开了其他人的协助，一步都不敢往深处去。虽然很多时候，与你要的东西就只有一步之遥。

我希望大家克制的第二重冲动，是玩弄正则表达式的快感。前面说过，正则表达式的语法和结构相当粗陋，内部逻辑又相当复杂，所以不少人学会之后，产生了“掌握神奇魔咒”的快感。凡是和字符串相关的处理必亮出神奇的正则表达式，能用一个正则表达式的绝不用两个，能用高级特性的绝不用简单特性……随之而来的是其他人查错时层出不穷的抱怨，更不用提更新时胆战心惊的烦恼。要知道，熟练使用正则表达式，却不滥用正则表达式，同时考虑合作同事的感受和效率，才能真正赢得大家的尊敬。

武学大师说：武功不是用来伤害，而是用来制止伤害的。哲学大师说：没有审慎思考，不懂得克制的人生，是不值得过的。这些道理听起来有悖常理，我花了不少时间才终于弄懂。我相信，正在阅读这本书的你，也应当懂得这些道理。

特别感谢两位女士，西乔和刘舫。西乔为这本书设计的封面，丝毫不受岁月的影响。刘舫编辑细致严谨的工作态度，支撑着我完成《正则指引》的第 2 版，再写下这篇序。

余晟

2018年9月3日

前言

提到正则表达式，许多人很有点不屑一顾：这东西，不登大雅之堂，再说也不是总要用到，何必专门花时间学习？

没错，正则表达式并不“总要用到”，但如果到了需要的场合不会用，往往面临“一分钱难倒英雄汉”的困境。经常需要处理文本的程序员自然知道正则表达式的价值，其他的程序员如果不会正则表达式，即便开发的领域与文本处理没什么关系，也难以躲过“躺着中枪”的命运——前几天我遇到一个问题，将一行长长的地址拆分成多行，负责这部分的程序员的日常工作只是制作 PDF 而已，拆分地址是很“边缘”的功能，但不会正则表达式就无法准确折行（一般需要在标点符号出现的地方折行，而不能只在空白字符处折行，但是不同语言中的标点符号各有不同），结果一筹莫展；相反，如果了解正则表达式，就可以很容易地处理各种语言中的标点字符。

按照我的开发经验，专门花点时间学习一下正则表达式，确实很有必要。目前可以见到的关于正则表达式的书籍和资料有不少，但又各有不足。

在互联网上，流传着一些编程语言的正则文档和《30 分钟教会你正则表达式》之类的帖子。这类资料的好处是简单直接，如果有现成的例子，而且适用于自己的语言，则可以直接抄来用。然而，其坏处也是简单直接，因为缺乏背后原理的讲解，如果找不到现成的例子，或者找不到能在自己所使用语言中行得通的例子（要知道，一种语言下的正则表达式往往并不能直接套用到另一种语言中），则束手无策。

在正式的出版领域，已经有《精通正则表达式》、《正则表达式必知必会》之类的书籍出版，尤其是前者，堪称关于正则表达式的经典著作，如果想认真学习正则表达式，这类书籍是必须阅读的。但这类书籍的弱点也很明显，即都是由英文版本翻译而来的，更多侧重英文文本的处理，身为中文世界的开发人员，我们经常需要处理中文文本——英文之外的字符。其实对于非英文字符的处理，正则表达式已经提供了足够丰富的功能，可惜资料相当匮乏。

为解决这些问题，我花了很多时间研习各种资料，然后经常给人讲解正则表达式的相关知识。我发现，很多人并不是不努力学，实在是合适的资料太少了。所以，我斗胆写作这本书。

相对于正则文档和速成教学帖子，本书深入讲解了匹配背后的原理，而且往往会举一反三，

告诉读者，这里为何这样写，如果改成其他形式，会造成什么结构差异；同时集中讲解和比较了多种语言中正则表达式用法的异同，方便读者把现成的正则表达式“移植”到自己的工作环境中。

相对于《精通正则表达式》等“正式”的书籍，本书辟出专门的章节讲解语言和编码，告诉读者如何设定编码，如何正确处理中文字符等。另外，本书还涵盖了.NET、Java、JavaScript、PHP、Python、Ruby、Objective-C、Golang 等常用语言，为每种语言专门撰写相关内容，不但详细介绍了语言中正则表达式的用法，更辨析了版本之间的细微差异，既可以作为专门学习的教材，也可以成为有用的参考手册。

本书结构

本书分为三部分。

第一部分主要讲解正则表达式的基础知识，覆盖常见正则表达式中的各种功能和结构。看完前 3 章，就可以基本弄明白现在流行的各种正则表达式；如果你之前有一些经验，会觉得阅读起来并不困难。但是我也希望读者不要忽略其他的内容，断言和匹配模式现在已经是正则表达式的“标准配置”了，而且确实可以派上大用场，所以第 4 章和第 5 章的内容，即便不是很熟悉，阅读起来可能有一些麻烦，但也不应该忽略。最后的第 6 章，则厘清了正则表达式在使用中的若干疑惑，了解它们，你就可以相对自如地穿行于正则表达式的世界了。

第二部分主要讲解关于正则表达式的深层次知识，这一部分用 3 章的内容，详细探讨了编码问题、匹配原理、解题思路。这部分内容更抽象，需要多花一点时间来阅读和理解，但是它们确实可以帮你在正则表达式的世界里登堂入室，脱离“术”的层面，掌握万变不离其宗的“道”。

第三部分的作用是接地气，将之前介绍的各种知识落实到常用语言.NET、Java、JavaScript、PHP、Python、Ruby、Objective-C、Golang 中来。每一章的开头有正则功能列表，其中的功能对应着前面部分的讲解，这些功能的具体应用实例以及不同版本之间的差异，则在章节中详细讲解，每一章的最后还给出了常见任务的示例代码，方便日后查询。第 18 章简要介绍了正则表达式在 Linux 下常用工具 vi、grep、awk、sed 中的使用，并通过一个实际的例子将这几种工具串起来，对比说明了它们适合解决的问题。

在本书的最后提供了用作参考的 3 个附录。

附录 A 是正则表达式的常用功能在不同语言中的比对，希望能给需要在多种语言中使用正则表达式或者移植正则表达式的读者提供一份有用的参考；附录 B 给出了若干常见的正则表达式，比如匹配邮政编码、身份证号、手机号、QQ 号、电子邮件地址等，希望能成为常见问题的“速查手册”；附录 C 列出了常用正则表达式的工具和资源，方便大家调试自己的正则表达式，以及继续深入学习。

本书的读者对象

本书适合以下几类读者：

经常需要进行文本处理（比如日志分析或网络运维）的技术人员。这些读者或许已经熟悉了正则表达式的基本用法，但面对日益复杂化和海量的数据，阅读本书可以帮助大家更准确、更高效地处理文本，提升自己工作的价值。

熟悉常用开发语言的程序员。虽然这些读者不需要专职进行文本处理，但源代码和许多数据其实也是文本，如果不会正则表达式，在偶然遇到处理源代码或文本数据的任务时，往往会产生无力感。本书的第三部分可以帮你快速找到有关的例子，并落实在自己的编程语言中。当然前两部分也非常有必要，因为它们可以帮你夯实基础。

对正则表达式已经有一定了解的读者。这些读者虽然能用正则表达式解决常见的问题，但未必了解正则表达式的编码问题、匹配原理、解题思路，仔细阅读本书的第二部分，可以深入完善对正则表达式的理解；而第三部分详细比较了可以使用正则表达式的各种语言，以及同一种语言中各种版本的差异。所有这一切，应该可以让你对正则表达式的掌握更上一层楼。

致谢

一本书的完成，离不开众多人的帮忙。

首先要感谢的是李笑来老师、周筠老师以及徐定翔和卢鹄翔两位编辑。在我翻译完《精通正则表达式》之后，李笑来老师三番五次地鼓励我写一本关于正则表达式的书，并且打消了我的很多顾虑；周筠老师、徐定翔和卢鹄翔两位编辑在我写作的最初阶段做了大量细致耐心的工作。可以说，没有他们，我就不会有写作这本书的念头，也不会有坚持完成的动力。

然后要感谢的是电子工业出版社的杨福平副总编、张月萍编辑、张春雨编辑和刘舫编辑，没有他们的关照和辛勤工作，这本书的出版定然会遇到更多的困难。

感谢我的朋友霍炬和韩磊，虽然我之前阅读过《精通正则表达式》，但与翻译和写作结缘，他们给了我莫大的帮助，于是今天才有了《正则指引》这本书。尤其值得一提的是，霍炬的夫人西乔，精心手绘了这本书的封面，我在这里要对她表示诚挚的谢意。

感谢我曾工作过的盛大创新院以及创新院的各位同事（李骏、郝培强、庄表伟、丁宇、许式伟、莫华枫、李道兵、赵劼、樊一鹏、张一宁等），创新院给了大家宽松自由的工作环境，与各位同事的讨论加深了我对正则表达式的理解，也为我提供了许多例子。

感谢张东亮、陆亦斌、孙勇、叶劲峰等各位朋友，愿意拨冗阅读本书的草稿，并提出了大量专业的意见。

感谢何源、陈钢、贺钧、陈驰等读者，试读本书之后提出了大量的宝贵意见，在最后关头打

消了我心中的许多忐忑。

在更早之前，我的父母从小就鼓励我研究和了解各种科学原理（“玩也要动脑筋”），我之所以有兴趣探究正则表达式背后的世界，而不满足于“够用/凑合”，都是受益于这种思维行为习惯。在中小学阶段，我的语文老师罗碧玉、易玺铭培养了我对于文字的兴趣，在大学阶段，东北师范大学文学院的王确老师给了我这个理科生非常多的帮助和指引。对各位师长，在此一并表示感谢，能遇到你们是我的幸运。

最后还需要感谢许多为这本书做出过贡献的人，你们的名字我可能暂时无法记起，或者无法一一罗列，但我会在心中保持对你们的感谢。

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误**：您对书中内容的修改意见可在 *提交勘误* 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 *读者评论* 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/35130

目录

第一部分

第二部分