正则表达式 *Cookbook*（影印版）

Includes a Regular Expressions Tutorial

Regular Expressions Cookbook

O'REILLY®
東南大學出版社

Jan Goyvaerts
& Steven Levithan 著

O'REILLY®

正则表达式Cookbook（影印版）

本书提供了超过100条的锦囊妙计，帮助你利用正则表达式处理数据、操纵文本。每位程序员都能找到正则表达式的用武之地，但想要充分发挥它的威力却未必容易。纵使经验丰富的用户也常会遇到性能不佳、误判、漏判或者令人费解的错误。《正则表达式Cookbook》对涉及此工具的最常见任务做了逐步讲解，此外还包括在C#、Java、JavaScript、Perl、PHP、Python、Ruby和VB.NET语言中使用正则表达式的诀窍。

阅读本书，你将：

- 通过简洁的教程理解正则表达式的基本原理
- 在多种编程和脚本语言中高效地应用正则表达式
- 学习如何验证和格式化输入
- 操纵单词、行、特殊字符和数值
- 找到在URL、路径、标记和数据交换中使用正则表达式的方法
- 学习更高级的正则表达式特性
- 理解在不同语言中正则表达式的应用程序接口、语法和行为的不同
- 针对特定需要，编写更加优化的正则表达式

无论你是初学者还是经验丰富的用户，《正则表达式Cookbook》都将有助于你对这一独特而不可替代的工具的理解。你将学到功能强大的新技巧，避免和语言相关的陷阱，利用这一经过实践检验的方法解决现实世界中的难题，从而节省宝贵的时间。

“这是一本精心打造的著作，内容翔实。我仅仅读了介绍的章节就学会了一些新技巧。”

——Nikolaj Lindberg，
计算机语言学家，
STTS Speech
Technology Services

“《正则表达式Cookbook》对当前面临的问题做出了优雅的解答。总之，我对这些技巧之详尽感到震惊。”

——Zak Greant，
开放技术倡导者
和战略家

Jan Goyvaerts经营Just Great Software软件公司，在这家公司他负责设计和开发一些最流行的正则表达式软件。

Steven Levithan是一位JavaScript正则表达式权威专家，同时他还负责一个以正则表达式内容为中心的流行博客。

www.oreilly.com

ISBN 978-7-5641-1931-7
9 787564 119317
定价：72.00元

正则表达式Cookbook（影印版）

Regular Expressions Cookbook

Jan Goyvaerts & Steven Levithan

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

东南大学出版社

图书在版编目（CIP）数据

正则表达式Cookbook：英文/（美）高瓦特斯(Goyvaerts, J.)，（美）莱文森（Levithan, S.）著. —影印本. —南京：东南大学出版社，2010.1

书名原文：Regular Expressions Cookbook

ISBN 978-7-5641-1931-7

Ⅰ. 正… Ⅱ. ①高… ②莱… Ⅲ. 正则表达式－英文 Ⅳ. TP301.2

中国版本图书馆CIP数据核字（2009）第205641号

江苏省版权局著作权合同登记

图字：10-2009-247号

正则表达式Cookbook（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼2号　　邮编：210096
出 版 人：江　汉
网　　址：http://press.seu.edu.cn
电子邮件：press@seu.edu.cn
印　　刷：扬中市印刷有限公司
开　　本：787毫米×980毫米　16开本
印　　张：32印张
字　　数：538千字
版　　次：2010年1月第1版
印　　次：2010年1月第1次印刷
书　　号：ISBN 978-7-5641-1931-7
印　　数：1~1600册
定　　价：72.00元（册）

本社图书若有印装质量问题，请直接与读者服务部联系。电话（传真）：025-83792328

正则表达式Cookbook（影印版）

Regular Expressions Cookbook

Preface

Over the past decade, regular expressions have experienced a remarkable rise in popularity. Today, all the popular programming languages include a powerful regular expression library, or even have regular expression support built right into the language. Many developers have taken advantage of these regular expression features to provide the users of their applications the ability to search or filter through their data using a regular expression. Regular expressions are everywhere.

Many books have been published to ride the wave of regular expression adoption. Most do a good job of explaining the regular expression syntax along with some examples and a reference. But there aren't any books that present solutions based on regular expressions to a wide range of real-world practical problems dealing with text on a computer and in a range of Internet applications. We, Steve and Jan, decided to fill that need with this book.

We particularly wanted to show how you can use regular expressions in situations where people with limited regular expression experience would say it can't be done, or where software purists would say a regular expression isn't the right tool for the job. Because regular expressions are everywhere these days, they are often a readily available tool that can be used by end users, without the need to involve a team of programmers. Even programmers can often save time by using a few regular expressions for information retrieval and alteration tasks that would take hours or days to code in procedural code, or that would otherwise require a third-party library that needs prior review and management approval.

Caught in the Snarls of Different Versions

As with anything that becomes popular in the IT industry, regular expressions come in many different implementations, with varying degrees of compatibility. This has resulted in many different regular expression *flavors* that don't always act the same way, or work at all, on a particular regular expression.

Many books do mention that there are different flavors and point out some of the differences. But they often leave out certain flavors here and there—particularly when a flavor lacks certain features—instead of providing alternative solutions or workarounds. This is frustrating when you have to work with different regular expression flavors in different applications or programming languages.

Casual statements in the literature, such as "everybody uses Perl-style regular expressions now," unfortunately trivialize a wide range of incompatibilities. Even "Perl-style" packages have important differences, and meanwhile Perl continues to evolve. Oversimplified impressions can lead programmers to spend half an hour or so fruitlessly running the debugger instead of checking the details of their regular expression implementation. Even when they discover that some feature they were depending on is not present, they don't always know how to work around it.

This book is the first book on the market that discusses the most popular and feature-rich regular expression flavors side by side, and does so consistently throughout the book.

Intended Audience

You should read this book if you regularly work with text on a computer, whether that's searching through a pile of documents, manipulating text in a text editor, or developing software that needs to search through or manipulate text. Regular expressions are an excellent tool for the job. *Regular Expressions Cookbook* teaches you everything you need to know about regular expressions. You don't need any prior experience whatsoever, because we explain even the most basic aspects of regular expressions.

If you do have experience with regular expressions, you'll find a wealth of detail that other books and online articles often gloss over. If you've ever been stumped by a regex that works in one application but not another, you'll find this book's detailed and equal coverage of seven of the world's most popular regular expression flavors very valuable. We organized the whole book as a cookbook, so you can jump right to the topics you want to read up on. If you read the book cover to cover, you'll become a world-class chef of regular expressions.

This book teaches you everything you need to know about regular expressions and then some, regardless of whether you are a programmer. If you want to use regular expressions with a text editor, search tool, or any application with an input box labeled "regex," you can read this book with no programming experience at all. Most of the recipes in this book have solutions purely based on one or more regular expressions.

If you are a programmer, Chapter 3 provides all the information you need to implement regular expressions in your source code. This chapter assumes you're familiar with the basic language features of the programming language of your choice, but it does not assume you have ever used a regular expression in your source code.

Technology Covered

.NET, Java, JavaScript, PCRE, Perl, Python, and Ruby aren't just back-cover buzzwords. These are the seven regular expression flavors covered by this book. We cover all seven flavors equally. We've particularly taken care to point out all the inconsistencies that we could find between those regular expression flavors.

The programming chapter (Chapter 3) has code listings in C#, Java, JavaScript, PHP, Perl, Python, Ruby, and VB.NET. Again, every recipe has solutions and explanations for all eight languages. While this makes the chapter somewhat repetitive, you can easily skip discussions on languages you aren't interested in without missing anything you should know about your language of choice.

Organization of This Book

The first three chapters of this book cover useful tools and basic information that give you a basis for using regular expressions; each of the subsequent chapters presents a variety of regular expressions while investigating one area of text processing in depth.

Chapter 1, *Introduction to Regular Expressions*, explains the role of regular expressions and introduces a number of tools that will make it easier to learn, create, and debug them.

Chapter 2, *Basic Regular Expression Skills*, covers each element and feature of regular expressions, along with important guidelines for effective use.

Chapter 3, *Programming with Regular Expressions*, specifies coding techniques and includes code listings for using regular expressions in each of the programming languages covered by this book.

Chapter 4, *Validation and Formatting*, contains recipes for handling typical user input, such as dates, phone numbers, and postal codes in various countries.

Chapter 5, *Words, Lines, and Special Characters*, explores common text processing tasks, such as checking for lines that contain or fail to contain certain words.

Chapter 6, *Numbers*, shows how to detect integers, floating-point numbers, and several other formats for this kind of input.

Chapter 7, *URLs, Paths, and Internet Addresses*, shows you how to take apart and manipulate the strings commonly used on the Internet and Windows systems to find things.

Chapter 8, *Markup and Data Interchange*, covers the manipulation of HTML, XML, comma-separated values (CSV), and INI-style configuration files.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, program elements such as variable or function names, values returned as the result of a regular expression replacement, and subject or input text that is applied to a regular expression. This could be the contents of a text box in an application, a file on disk, or the contents of a string variable.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

`‹Regular•expression›`
: Represents a regular expression, standing alone or as you would type it into the search box of an application. Spaces in regular expressions are indicated with gray circles, except when spaces are used in free-spacing mode.

`«Replacement•text»`
: Represents the text that regular expression matches will be replaced with in a search-and-replace operation. Spaces in replacement text are indicated with gray circles.

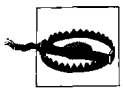
`Matched text`
: Represents the part of the subject text that matches a regular expression.

...
: A gray ellipsis in a regular expression indicates that you have to "fill in the blank" before you can use the regular expression. The accompanying text explains what you can fill in.

CR, LF, *and* CRLF
: CR, LF, and CRLF in boxes represent actual line break characters in strings, rather than character escapes such as `\r`, `\n`, and `\r\n`. Such strings can be created by pressing Enter in a multiline edit control in an application, or by using multiline string constants in source code such as verbatim strings in C# or triple-quoted strings in Python.

↵
: The return arrow, as you may see on the Return or Enter key on your keyboard, indicates that we had to break up a line to make it fit the width of the printed page. When typing the text into your source code, you should not press Enter, but instead type everything on a single line.

This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Regular Expressions Cookbook* by Jan Goyvaerts and Steven Levithan. Copyright 2009 Jan Goyvaerts and Steven Levithan, 978-0-596-2068-7."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

When you see a Safari® Books Online icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North

Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book where we list errata, examples, and any additional information. You can access this page at:

http://www.regexcookbook.com

or at:

http://oreilly.com/catalog/9780596520687

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://www.oreilly.com

Acknowledgments

We thank Andy Oram, our editor at O'Reilly Media, Inc., for helping us see this project from start to finish. We also thank Jeffrey Friedl, Zak Greant, Nikolaj Lindberg, and Ian Morse for their careful technical reviews, which made this a more comprehensive and accurate book.

Table of Contents

CHAPTER 1

Introduction to Regular Expressions

Having opened this cookbook, you are probably eager to inject some of the ungainly strings of parentheses and question marks you find in its chapters right into your code. If you are ready to plug and play, be our guest: the practical regular expressions are listed and described in Chapters 4 through 8.

But the initial chapters of this book may save you a lot of time in the long run. For instance, this chapter introduces you to a number of utilities—some of them created by one of the authors, Jan—that let you test and debug a regular expression before you bury it in code where errors are harder to find. And these initial chapters also show you how to use various features and options of regular expressions to make your life easier, help you understand regular expressions in order to improve their performance, and learn the subtle differences between how regular expressions are handled by different programming languages—and even different versions of your favorite programming language.

So we've put a lot of effort into these background matters, confident that you'll read it before you start or when you get frustrated by your use of regular expressions and want to bolster your understanding.

Regular Expressions Defined

In the context of this book, a *regular expression* is a specific kind of text pattern that you can use with many modern applications and programming languages. You can use them to verify whether input fits into the text pattern, to find text that matches the pattern within a larger body of text, to replace text matching the pattern with other text or rearranged bits of the matched text, to split a block of text into a list of subtexts, and to shoot yourself in the foot. This book helps you understand exactly what you're doing and avoid disaster.

History of the Term 'Regular Expression'

The term *regular expression* comes from mathematics and computer science theory, where it reflects a trait of mathematical expressions called *regularity*. Such an expression can be implemented in software using a deterministic finite automaton (DFA). A DFA is a finite state machine that doesn't use backtracking.

The text patterns used by the earliest *grep* tools were regular expressions in the mathematical sense. Though the name has stuck, modern-day Perl-style regular expressions are not regular expressions at all in the mathematical sense. They're implemented with a nondeterministic finite automaton (NFA). You will learn all about backtracking shortly. All a practical programmer needs to remember from this note is that some ivory tower computer scientists get upset about their well-defined terminology being overloaded with technology that's far more useful in the real world.

If you use regular expressions with skill, they simplify many programming and text processing tasks, and allow many that wouldn't be at all feasible without the regular expressions. You would need dozens if not hundreds of lines of procedural code to extract all email addresses from a document—code that is tedious to write and hard to maintain. But with the proper regular expression, as shown in Recipe 4.1, it takes just a few lines of code, or maybe even one line.

But if you try to do too much with just one regular expression, or use regexes where they're not really appropriate, you'll find out why some people say:*

> Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.

The second problem those people have is that they didn't read the owner's manual, which you are holding now. Read on. Regular expressions are a powerful tool. If your job involves manipulating or extracting text on a computer, a firm grasp of regular expressions will save you plenty of overtime.

Many Flavors of Regular Expressions

All right, the title of the previous section was a lie. We didn't define what regular expressions are. We can't. There is no official standard that defines exactly which text patterns are regular expressions and which aren't. As you can imagine, every designer of programming languages and every developer of text processing applications has a different idea of exactly what a regular expression should be. So now we're stuck with a whole palate of regular expression *flavors*.

Fortunately, most designers and developers are lazy. Why create something totally new when you can copy what has already been done? As a result, all modern regular expression flavors, including those discussed in this book, can trace their history back to

* Jeffrey Friedl traces the history of this quote in his blog at *http://regex.info/blog/2006-09-15/247*.