

Expressões Regulares Cookbook

**Jan Goyvaerts
Steven Levithan**

Authorized Portuguese translation of the English edition of *Regular Expressions Cookbook* ISBN 9780596520687
© 2009, Jan Goyvaerts and Steve Levithan. This translation is published and sold by permission of O'Reilly
Media, Inc., the owner of all rights to publish and sell the same.

Tradução em português autorizada da edição em inglês da obra *Regular Expressions Cookbook* ISBN
9780596520687 © 2009, Jan Goyvaerts e Steve Levithan. Esta tradução é publicada e vendida com a permissão
da O'Reilly Media, Inc., detentora de todos os direitos para publicação e venda desta obra.

© Novatec Editora Ltda. 2011.

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/1998.
É proibida a reprodução desta obra, mesmo parcial, por qualquer processo, sem prévia autorização, por escrito,
do autor e da Editora.

Editor: Rubens Prates
Tradução: Rafael Contatori e Edgard Damiani
Revisão gramatical: Jeferson Ferreira
Editoração eletrônica: Camila Kuwabata e Carolina Kuwabata

ISBN: 978-85-7522-279-9

Histórico de impressões:

Abril/2011 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
E-mail: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Goyvaerts, Jan
Expressões regulares Cookbook / Jan Goyvaerts,
Steven Levithan ; [tradução Rafael Contatori
/ Edgard Damiani]. -- São Paulo : Novatec Editora ;
Cambridge : O'Reilly Media, 2011.

Título original: Regular expressions Cookbook.
ISBN 978-85-7522-279-9

1. Cookbook (Linguagem de programação)
2. Expressões regulares I. Levithan, Steven.
II. Título.

10-08695

CDD-005.115

Índices para catálogo sistemático:

1. Expressões regulares Cookbook : Ciência da
computação 005.115
OGF20110406

Introdução às expressões regulares

Ao abrir este livro de receitas, você provavelmente ficará ansioso para inserir diretamente em seu código as deselegantes strings de parênteses e pontos de interrogação que encontrará nos próximos capítulos. Se estiver pronto para começar, fique à vontade: as expressões regulares práticas estão listadas e descritas nos capítulos 4 a 8.

Porém, os capítulos iniciais deste livro irão economizar muito tempo a longo prazo. Por exemplo, este capítulo introduz uma série de utilitários – alguns deles criados por um dos autores, Jan – que permitem testar e depurar uma expressão regular antes de embuti-la no código, onde os erros ficam mais difíceis de serem achados. Estes capítulos iniciais também mostram como utilizar as várias características e opções das expressões regulares para facilitar sua vida, ajudam a entender as expressões regulares, a fim de melhorar sua performance, e ensinam as diferenças sutis de tratamento das expressões regulares pelas diferentes linguagens de programação – e até mesmo pelas diferentes versões de sua linguagem favorita.

Dedicamos muitos esforços para estas questões de bastidores, confiantes de que você as lerá antes de começar a trabalhar, ou quando se sentir frustrado com a utilização de expressões regulares e, por isso, desejar reforçar seu nível de compreensão.

Definição de expressões regulares

No contexto deste livro, uma *expressão regular* é um tipo específico de texto-padrão que você pode utilizar em muitos aplicativos modernos e em linguagens de programação. Você pode usá-las para verificar se a entrada de dados encaixa-se no padrão de texto, para encontrar um texto que corresponda a um padrão dentro de um conjunto maior de textos, para substituir o texto padrão por outro ou reorganizar bits de texto correspondentes, para dividir um bloco de texto em uma lista de subtópicos e para dar um tiro no pé. Este livro o ajudará a entender exatamente o que você está fazendo, evitando, assim, desastres.

História do termo ‘Expressão Regular’

O termo *expressão regular* vem da matemática e da teoria da ciência da computação. Ele reflete uma peculiaridade das expressões matemáticas, chamada *regularidade*. Essa expressão pode ser implementada em programas usando um autômato finito determinístico (DFA). Um DFA é uma máquina de estados finitos que não usa retrocesso (backtracking).

Os padrões de texto utilizados pelas primeiras ferramentas *grep* (Global Regular Expression and Print) eram expressões regulares no sentido matemático. Embora o nome tenha pegado, as expressões regulares modernas, no estilo Perl, não são expressões regulares no sentido matemático. Elas são implementadas com um autômato finito não determinístico (NFA). Você aprenderá tudo sobre retrocesso em breve. Tudo de que um programador prático precisa se lembrar é que alguns cientistas elitistas irritam-se com sua bem definida terminologia sendo sobrecarregada com usos e noções inerentes à tecnologia, que é muito mais útil no mundo real.

Se você utilizar expressões regulares com habilidade, elas podem simplificar muitas tarefas de programação e processamento de texto, além de permitir outras que não seriam possíveis sem elas. Você precisaria de dezenas, se não centenas de linhas de código procedural para extrair todos os endereços de e-mail de um documento – código tedioso de escrever e difícil de manter. Mas, com a expressão regular correta, como mostrado na receita 4.1, são necessárias apenas algumas linhas de código, ou talvez até uma única linha.

Porém, se você tentar fazer muita coisa com apenas uma expressão regular, ou utilizar expressões regulares onde não for realmente necessário, irá descobrir o sentido da seguinte citação¹:

Algumas pessoas, quando confrontadas com um problema, pensam: “Já sei, vou utilizar expressões regulares”. Agora, elas têm dois problemas.

O segundo problema destas pessoas é o fato de não terem lido o manual do proprietário; este mesmo que você está segurando. Leia-o agora. A expressão regular é uma ferramenta poderosa. Se seu trabalho envolve manipulação ou extração de texto em um computador, o domínio das expressões regulares irá lhe poupar muitas horas de esforço.

Muitos sabores de expressões regulares

Tudo bem, o título da seção anterior era uma mentira. Nós não definimos o que são expressões regulares. Não podemos. Não existe nenhuma norma oficial que defina

¹ Jeffrey Friedl narra a história desta citação em seu blog em <http://regex.info/blog/2006-09-15/247>.

exatamente quais padrões de texto são expressões regulares e quais não são. Como você pode imaginar, cada programador de linguagens e cada desenvolvedor de aplicações de processamento de texto tem uma ideia diferente do que, exatamente, uma expressão regular deva ser. Por isso, agora estamos amarrados a um leque enorme de *sabores* de expressões regulares.

Felizmente, a maioria dos designers e programadores é preguiçosa. Por que criar algo totalmente novo, quando você pode copiar o que já foi feito? Como resultado, todos os sabores modernos de expressões regulares, incluindo os discutidos neste livro, podem traçar sua história a partir da linguagem de programação Perl. Chamamos esses sabores de *expressões regulares no estilo Perl*. A sintaxe desses sabores de expressões regulares é muito semelhante, e geralmente compatível, mas não totalmente.

Os escritores são preguiçosos, também. Geralmente digitamos *regex*, ou *regex*, para designar uma única expressão regular, e *regexes* para indicar o plural.

Os sabores *regex* não correspondem, um-para-um, às linguagens de programação. Linguagens de script tendem a ter seus próprios sabores de expressão regular embutidos. Outras linguagens de programação contam com bibliotecas para suporte a expressão regular. Algumas bibliotecas estão disponíveis para várias linguagens, enquanto algumas linguagens podem recorrer a diferentes bibliotecas.

Este capítulo introdutório trata apenas dos sabores de expressões regulares, e ignora completamente quaisquer considerações a respeito de programação. O capítulo 3 inicia as listagens de código, então você pode dar uma espiada no tópico “Linguagens de programação e sabores da expressão regular” daquele capítulo, para descobrir com quais sabores irá trabalhar. Por enquanto, ignore todas as coisas relacionadas à programação. As ferramentas listadas na próxima seção facilitam a exploração da sintaxe da expressão regular por meio do “aprenda fazendo”.

Sabores *regex* abordados neste livro

Para este livro, foram selecionados os sabores *regex* mais populares em uso, atualmente. Todos são sabores *regex* no *estilo Perl*. Alguns sabores têm mais recursos do que outros. Mas, se dois sabores têm uma mesma característica, eles tendem a utilizar a mesma sintaxe. Vamos apontar as poucas incoerências irritantes, conforme formos nos deparando com elas.

Todos estes sabores *regex* fazem parte de linguagens de programação e bibliotecas que estão em desenvolvimento ativo. A lista de sabores lhe diz quais versões este livro cobre. Mais adiante no livro, mencionamos o sabor sem quaisquer versões, caso a expressão regular atual funcione da mesma forma com todos os sabores. Quase sempre esse será o caso. Salvo as correções de bugs que afetam casos isolados, sabores *regex*

não tendem a mudar, exceto pela adição de recursos que dão um novo significado à sintaxe, tratada anteriormente como um erro:

Perl

O suporte embutido do Perl para expressões regulares é a principal razão pela qual as regexes são populares hoje. Este livro cobre Perl 5.6, 5.8 e 5.10.

Muitos aplicativos e bibliotecas regex que dizem utilizar o Perl, ou expressões regulares compatíveis com o Perl, na realidade, meramente utilizam expressões regulares no estilo Perl. Elas utilizam uma sintaxe de expressão regular similar à do Perl, mas não suportam o mesmo conjunto de características. Muito provavelmente, elas usam um dos sabores regex logo adiante. Esses sabores são todos no estilo Perl.

PCRE

PCRE é a biblioteca C das “Expressões regulares compatíveis com Perl”, desenvolvida por Philip Hazel. Você pode baixar esta biblioteca de fonte aberta em <http://www.pcre.org>. Este livro cobre as versões 4 a 7 da PCRE.

Embora PCRE afirme ser compatível com Perl, e provavelmente o é mais do que qualquer outro sabor neste livro, trata-se, realmente, de apenas um estilo Perl. Algumas funções, tais como o suporte a Unicode, são ligeiramente diferentes, e você não pode misturar código Perl em sua expressão regular, como o próprio Perl permite.

Devido a sua licença de código aberto e programação sólida, o PCRE encontrou seu caminho em muitas linguagens de programação e aplicações. Ele está embutido no PHP e empacotado em numerosos componentes Delphi. Se a aplicação diz suportar expressões regulares “compatíveis com Perl”, sem listar especificamente o sabor regex usado, ela provavelmente está usando o PCRE.

.NET

O Framework Microsoft .NET fornece um conjunto completo de sabores de expressões regulares no estilo Perl por meio do pacote `System.Text.RegularExpressions`. Este livro cobre do .NET 1.0 até o 3.5. Estritamente falando, só existem duas versões do `System.Text.RegularExpressions`: 1.0 e 2.0. Nenhuma alteração foi feita para as classes `Regex` nas versões de .NET 1.1, 3.0 e 3.5.

Qualquer linguagem de programação .NET, incluindo o C#, VB.NET, Delphi para .NET e até mesmo COBOL.NET, tem acesso completo aos sabores das expressões regulares .NET. Se um aplicativo desenvolvido com .NET lhe oferecer suporte às expressões regulares, pode estar certo de que ele usa o sabor .NET, mesmo que diga utilizar “expressões regulares de Perl”. A única exceção

é o Visual Studio (VS) em si. O ambiente de desenvolvimento integrado do VS (IDE) ainda usa o velho sabor de expressão regular utilizado desde o início, que não é o estilo Perl.

Java

Java 4 é a primeira versão Java a fornecer suporte interno a expressões regulares por meio do pacote `java.util.regex`. Ele ofuscou rapidamente as várias bibliotecas Java de expressões regulares de terceiros. Além de ser padrão e integrada, oferece um sabor de expressões regulares no estilo Perl, completo e com excelente desempenho, mesmo quando comparado com aplicações escritas em C. Este livro aborda o pacote `java.util.regex` em Java 4, 5 e 6.

Se você vem utilizando programas desenvolvidos em Java ao longo dos últimos anos, qualquer suporte a expressão regular oferecido por eles provavelmente utiliza o sabor Java.

JavaScript

Neste livro, usamos o termo *JavaScript* para indicar o sabor da expressão regular definida na versão 3 do padrão ECMA-262. Esta norma define a linguagem de programação ECMAScript, mais conhecida por suas implementações de JavaScript e JScript em diferentes navegadores. O Internet Explorer, do 5.5 até o 8.0, o Firefox, o Opera e o Safari, todos implementam a 3ª edição da ECMA-262. No entanto, todos os navegadores apresentam vários bugs, levando-os a desviarem-se da norma. Apontamos essas questões em situações em que elas importam.

Se um site permite que você busque ou filtre usando uma expressão regular sem esperar por uma resposta do servidor web, ele usa o sabor regex JavaScript, o único sabor de expressão regular que trabalha com todos os navegadores no lado do cliente. Mesmo o Microsoft VBScript e o Adobe ActionScript 3 utilizam-no.

Python

Python suporta expressões regulares por meio do seu módulo `re`. Este livro cobre o Python 2.4 e 2.5. O suporte às expressões regulares do Python mantém-se inalterado há muitos anos.

Ruby

O suporte às expressões regulares do Ruby é parte da própria linguagem Ruby, similar ao Perl. Este livro cobre o Ruby 1.8 e 1.9. Uma compilação padrão do Ruby 1.8 utiliza o sabor de expressões regulares fornecido diretamente pelo código-fonte Ruby. Uma compilação padrão do Ruby 1.9 utiliza a biblioteca Oniguruma de expressões regulares. O Ruby 1.8 pode ser compilado para utilizar a Oniguruma, e o Ruby 1.9 pode ser compilado para utilizar expressões

regulares Ruby mais antigas. Neste livro, identificamos o sabor Ruby nativo como sendo o Ruby 1.8, e o sabor Oniguruma como o Ruby 1.9.

Para testar qual tipo de expressão regular Ruby seu site utiliza, tente utilizar a expressão regular `<a++>`. O Ruby 1.8 dirá que a expressão regular é inválida, porque não suporta quantificadores possessivos, enquanto que no Ruby 1.9 ela corresponderá a uma sequência de um ou mais caracteres.

A biblioteca Oniguruma foi projetada para ser compatível com o Ruby 1.8, adicionando novas funcionalidades que não irão conflitar com as expressões regulares existentes. Os implementadores até deixaram funcionalidades que provavelmente deveriam ter sido alteradas, como o uso do `(?m)` para significar “o ponto corresponde a quebras de linha”, onde outros sabores de expressões regulares utilizam `(?s)`.

Pesquisa e substituição com expressões regulares

Pesquisa-e-substituição é uma tarefa comum para as expressões regulares. Uma função de pesquisa-e-substituição envolve, como entrada, uma string de assunto, uma expressão regular e uma string de substituição. A saída é a string de assunto com todas as correspondências da expressão regular trocadas pelo texto de substituição.

Embora o texto de substituição não seja uma expressão regular, você pode utilizar certas sintaxes especiais para construir textos de substituição dinâmicos. Todos os sabores permitem que você reinsira o texto correspondido pela expressão regular ou por um grupo de captura dentro do texto de substituição. As receitas 2.20 e 2.21 explicam isso. Alguns sabores também suportam a inserção do texto correspondido no texto de substituição, como mostra a receita 2.22. No capítulo 3, a receita 3.16 ensina como gerar um texto de substituição diferente para cada correspondência no código.

Muitos sabores de textos de substituição

Ideias diferentes, de diferentes desenvolvedores de programas de expressões regulares, levaram a uma ampla gama de sabores, cada um com uma sintaxe e conjuntos de recursos diferentes. A história, no caso dos textos de substituição, não é diferente. Na verdade, há mais sabores de textos de substituição do que sabores de expressões regulares. Construir um mecanismo de expressão regular não é fácil. A maioria dos programadores prefere reutilizar um já existente, e a aplicação de uma função de pesquisa-e-substituição em um mecanismo de expressão regular existente é bem fácil. O resultado é que existem muitos sabores de textos de substituição, no caso das bibliotecas de expressões regulares, que não têm recursos nativos de pesquisa-e-substituição.

Felizmente, todos os sabores de expressões regulares neste livro têm sabores de textos de substituição correspondentes, exceto no PCRE. Esta lacuna no PCRE complica a vida dos programadores que utilizam sabores nele baseados. A biblioteca de código aberto do PCRE não inclui as funções necessárias para fazer substituições. Assim, todos os aplicativos e linguagens de programação que se baseiam no PCRE precisam providenciar sua própria função de pesquisa-e-substituição. A maioria dos programadores tenta copiar sintaxes existentes, mas nunca as fazem exatamente da mesma maneira.

Este livro cobre os seguintes sabores de textos de substituição. Consulte “Muitos sabores de expressões regulares”, na página 16, para mais detalhes sobre os sabores de expressões regulares que correspondem aos sabores de textos de substituição:

Perl

Perl tem o suporte embutido para substituição de expressões regulares por meio do operador `s/regex/replace/`. O sabor de texto de substituição do Perl corresponde ao sabor de expressão regular do Perl. Este livro cobre do Perl 5.6 ao Pearl 5.10. Esta última versão adiciona suporte a retrorreferências (backreferences) no texto de substituição, ao acrescentar capturas nomeadas à sintaxe da expressão regular.

PHP

Neste livro, o sabor PHP de texto de substituição refere-se à função `preg_replace`. Essa função utiliza o sabor das expressões regulares PCRE e o sabor de texto de substituição do PHP.

Outras linguagens de programação que utilizam o PCRE não utilizam o mesmo sabor de texto de substituição que o PHP. Dependendo de onde o designer de sua linguagem de programação obteve inspiração, a sintaxe de texto de substituição pode ser semelhante à do PHP, ou a qualquer outro sabor de textos de substituição deste livro.

O PHP também possui uma função `ereg_replace`. Essa função utiliza um sabor diferente de expressão regular (POSIX ERE), além de usar um sabor diferente de texto de substituição. As funções `ereg` do PHP não são discutidas neste livro.

.NET

O pacote `System.Text.RegularExpressions` fornece várias funções de pesquisa-e-substituição. O sabor de texto de substituição do .NET corresponde ao sabor de expressão regular .NET. Todas as versões .NET utilizam o mesmo sabor de texto de substituição. Os novos recursos de expressão regular no .NET 2.0 não afetam a sintaxe de texto de substituição.

Java

O pacote `java.util.regex` tem funções internas de pesquisa-e-substituição. Este livro aborda o Java 4, 5 e 6. Todos utilizam a mesma sintaxe de texto de substituição.

JavaScript

Neste livro, utilizamos o termo *JavaScript* para indicar tanto o sabor de texto de substituição, quanto o sabor de expressão regular, definidos na 3ª edição do padrão ECMA-262.

Python

O módulo `re` do Python oferece a função `sub` para pesquisa-e-substituição. O sabor de texto de substituição do Python corresponde ao sabor de expressão regular Python. Este livro cobre o Python 2.4 e 2.5. O suporte às expressões regulares do Python tem se mantido estável por muitos anos.

Ruby

O suporte às expressões regulares do Ruby é parte constituinte da linguagem em si, incluindo a função de pesquisa-e-substituição. Este livro aborda o Ruby 1.8 e 1.9. Uma compilação padrão do Ruby 1.8 utiliza expressões regulares fornecidas diretamente pelo código-fonte, enquanto uma compilação padrão do Ruby 1.9 utiliza a biblioteca Oniguruma de expressões regulares. O Ruby 1.8 pode ser compilado para utilizar a biblioteca Oniguruma, e o Ruby 1.9 pode ser compilado para utilizar uma expressão regular Ruby mais antiga. Neste livro, identificamos o sabor nativo Ruby como sendo o Ruby 1.8 e o sabor Oniguruma como sendo o Ruby 1.9.

A sintaxe de texto de substituição das versões 1.8 e 1.9 do Ruby é a mesma, exceto pelo fato de que o Ruby 1.9 adiciona retrorreferências ao texto de substituição. Captura nomeada é um recurso novo nas expressões regulares do Ruby 1.9.

Ferramentas para se trabalhar com expressões regulares

A menos que você já venha programando com expressões regulares por algum tempo, recomendamos, em primeiro lugar, a manipulação das expressões regulares em uma ferramenta, e não diretamente no código-fonte. O exemplo das expressões regulares, neste capítulo e no capítulo 2, são expressões regulares simples que não contêm o escape extra que uma linguagem de programação (mesmo em um Unix shell) exige. Você pode digitar estas expressões regulares diretamente na caixa de busca de um aplicativo.

O capítulo 3 explica como corresponder expressões regulares ao seu código-fonte. Citar uma expressão regular literal como uma string torna-a ainda mais difícil de ler, porque

as regras de escape da string se misturam com as regras de escape das expressões regulares. Evitaremos isso até a receita 3.1. Depois de entender os conceitos básicos das expressões regulares, você será capaz de enxergar em meio à floresta de barras invertidas.

As ferramentas descritas nesta seção também fornecem depuração, verificação de sintaxe e outros feedbacks, elementos que você não teria na maioria dos ambientes de programação. Portanto, ao desenvolver expressões regulares para seus aplicativos, você pode considerar útil construir expressões regulares complicadas em uma dessas ferramentas, antes de inseri-las em seu programa.

RegexBuddy

No momento em que escrevemos isso, RegexBuddy (Figura 1.1) é a ferramenta mais completa disponível para criação, teste e implementação de expressões regulares. Ela tem a habilidade única de emular todos os sabores de expressões regulares discutidos neste livro, e até mesmo de fazer conversões entre os diferentes sabores.

RegexBuddy foi projetado e desenvolvido por Jan Goyvaerts, um dos autores deste livro. Projetar e desenvolver RegexBuddy fez de Jan um especialista em expressões regulares, e a utilização do RegexBuddy ajudou a tornar o coautor Steven em um viciado em expressões regulares, a ponto de produzir este livro para a O'Reilly.

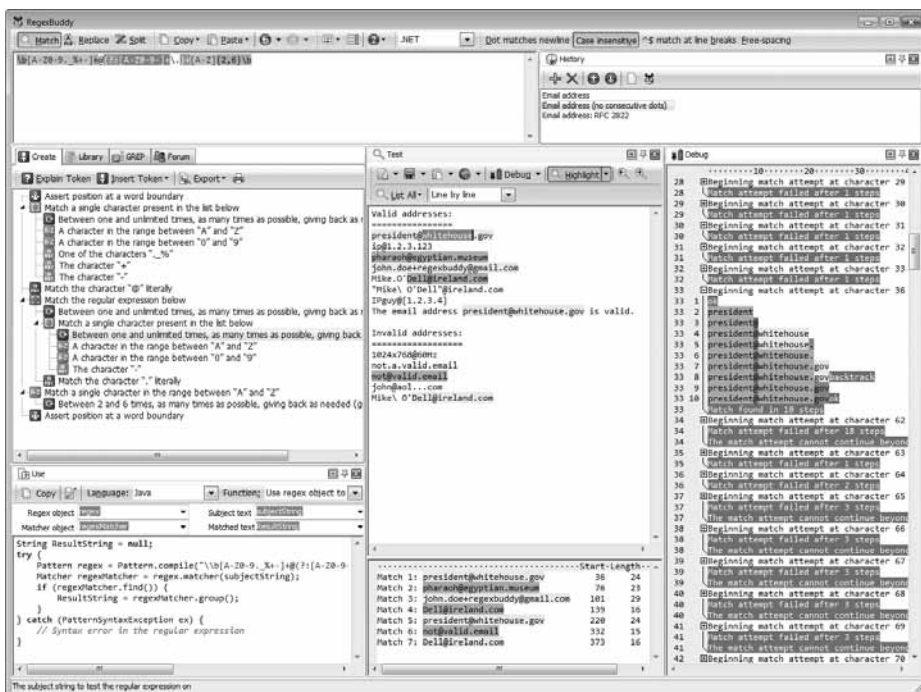


Figura 1.1 – RegexBuddy.

Se a captura de tela (Figura 1.1) parece um pouco carregada é porque quisemos dispor a maioria dos painéis lado a lado, para mostrar a extensa funcionalidade do RegexBuddy. A exibição padrão acopla todos os painéis perfeitamente em uma linha de guias. Você também pode arrastar os painéis para um monitor secundário.

Para experimentar uma das expressões regulares mostradas neste livro, simplesmente digite-a na caixa de edição, no topo da janela do RegexBuddy. O RegexBuddy aplica automaticamente a sintaxe, destacando a sua expressão regular e fazendo com que os erros e parênteses incompatíveis fiquem óbvios.

O painel **Create** cria automaticamente uma análise detalhada, em inglês, enquanto digitamos a regex. Dê um duplo clique em qualquer descrição, na árvore da expressão regular, para editar essa parte de sua expressão regular. Você pode inserir novas partes manualmente em sua expressão regular, ou clicar no botão **Insert Token** e selecionar o que quiser em um menu. Por exemplo, se você não se lembra da complexa sintaxe para um procedimento de antecipação positiva (positive lookahead), pode solicitar ao RegexBuddy que insira os caracteres apropriados.

Digite ou cole algum texto de exemplo no painel **Test**. Quando o botão **Highlight** estiver ativo, o RegexBuddy automaticamente sublinha o texto correspondido pela regex.

Alguns dos botões mais utilizados são:

List All

Exibe uma lista de todas as correspondências.

Replace

O botão **Replace**, na parte superior, exibe uma nova janela que permite a digitação do texto de substituição. O botão **Replace**, na caixa **Test**, permite que você visualize o texto de assunto após as substituições feitas.

Split (O botão no painel Test, e não o do topo)

Trata a expressão regular como um separador e divide o assunto em tokens, de acordo com os locais em que as correspondências foram encontradas em seu texto de assunto, usando a sua expressão regular.

Clique em qualquer um desses botões e selecione **Update Automatically** para fazer o RegexBuddy manter os resultados em sincronia, dinamicamente, ao editar sua regex ou texto de assunto.

Para ver exatamente como funcionará sua regex (ou não), clique em uma correspondência destacada, ou no ponto onde a regex falha no painel **Test**, e clique no botão **Debug**. O RegexBuddy mudará para o painel **Debug**, mostrando passo a passo todos os

processos de correspondências. Clique em qualquer lugar no texto de saída do depurador para ver qual símbolo regex corresponde ao texto em que você clicou. Clique em sua expressão regular para destacar esta parte da regex no depurador.

No painel **Use**, escolha sua linguagem de programação favorita. Em seguida, selecione uma função para gerar instantaneamente o código-fonte que implementa sua regex. Os modelos de código-fonte do **RegexBuddy** são totalmente editáveis por meio de seu editor interno. Você pode adicionar novas funções e até mesmo novas linguagens, ou alterar as fornecidas.

Para testar a sua regex em um conjunto maior de dados, alterne para o painel **GREP**, para pesquisar (e substituir) em qualquer número de arquivos e pastas.

Quando você encontrar uma regex em seu código-fonte, copie-a para a área de transferência, incluindo as aspas delimitadoras ou barras. No **RegexBuddy**, clique no botão **Paste**, na parte superior, e selecione o estilo de string de sua linguagem de programação. Sua regex aparecerá, então, no **RegexBuddy** como uma regex pura, sem as aspas extras e escapes necessários para literais strings. Utilize o botão **Copy**, na parte superior, para criar uma string na sintaxe desejada, e cole-a de volta em seu código-fonte.

Conforme sua experiência aumentar, é possível construir uma biblioteca de expressões regulares no painel **Library**. Certifique-se de adicionar uma descrição detalhada e um assunto de teste ao armazenar uma regex. As expressões regulares podem ser enigmáticas, mesmo para especialistas.

Se você realmente não consegue entender uma regex, clique no painel **Forum** e, em seguida, no botão **Login**. Se você comprou o **RegexBuddy**, a tela de login aparecerá. Clique em **OK**, e você estará imediatamente conectado ao Forum do Usuário **RegexBuddy**. Steven e Jan às vezes estão por lá.

RegexBuddy roda em Windows 98, ME, 2000, XP e Vista. Para os fãs do Linux e da Apple, o **RegexBuddy** também vai bem no VMware, Parallels, CrossOver Office e, com algumas restrições, no WINE. Você pode baixar uma cópia de avaliação gratuita do **RegexBuddy** em <http://www.regexbuddy.com/RegexBuddyCookbook.exe>. Exceto pelo fórum de usuários, a cópia é totalmente funcional para sete dias de uso.

RegexPal

RegexPal (Figura 1.2) é um testador on-line de expressões regulares criado por Steven Levithan, um dos autores deste livro. Tudo de que você precisa para usá-lo é um navegador moderno. **RegexPal** é inteiramente escrito em JavaScript, portanto ele suporta apenas o sabor regex JavaScript implementado no navegador que estiver usando para acessá-lo.

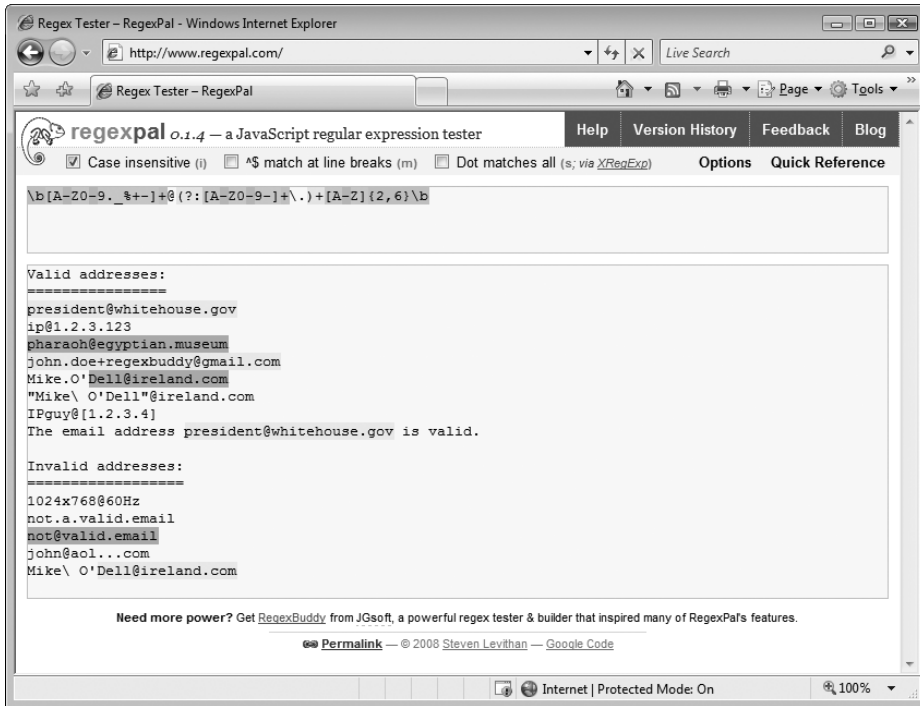


Figura 1.2 – Regexpal.

Para experimentar uma das expressões regulares mostradas neste livro, vá a <http://www.regexpal.com>. Digite a regex dentro da caixa que diz: “Enter regex here”. O Regexpal aplica automaticamente o destaque de sintaxe na sua expressão regular, revelando imediatamente qualquer erro de sintaxe. O Regexpal tem consciência das diferenças entre navegadores, e dos problemas, oriundos destas diferenças, que podem arruinar seu trabalho ao lidar com expressões regulares em JavaScript. Se uma determinada sintaxe não funcionar corretamente em alguns navegadores, o Regexpal vai destacá-la como um erro.

Agora, digite ou cole algum exemplo de texto na caixa que diz: “Enter test data here.” O Regexpal destaca automaticamente o texto correspondido por sua regex.

Não há botões nos quais clicar, fazendo do Regexpal um dos mais convenientes testadores on-line para expressões regulares.

Mais testadores regex online

Criar um simples testador de expressão regular on-line é fácil. Se você tem algumas habilidades básicas de desenvolvimento web, a informação no capítulo 3 é tudo de que precisa para criar seu próprio testador. Centenas de pessoas já o fizeram, mas algumas delas acrescentaram recursos extras, tornando-os dignos de menção.

regex.larsolavtorvik.com

Lars Olav Torvik disponibilizou um excelente testador de expressão regular on-line em <http://regex.larsolavtorvik.com> (ver figura 1.3).

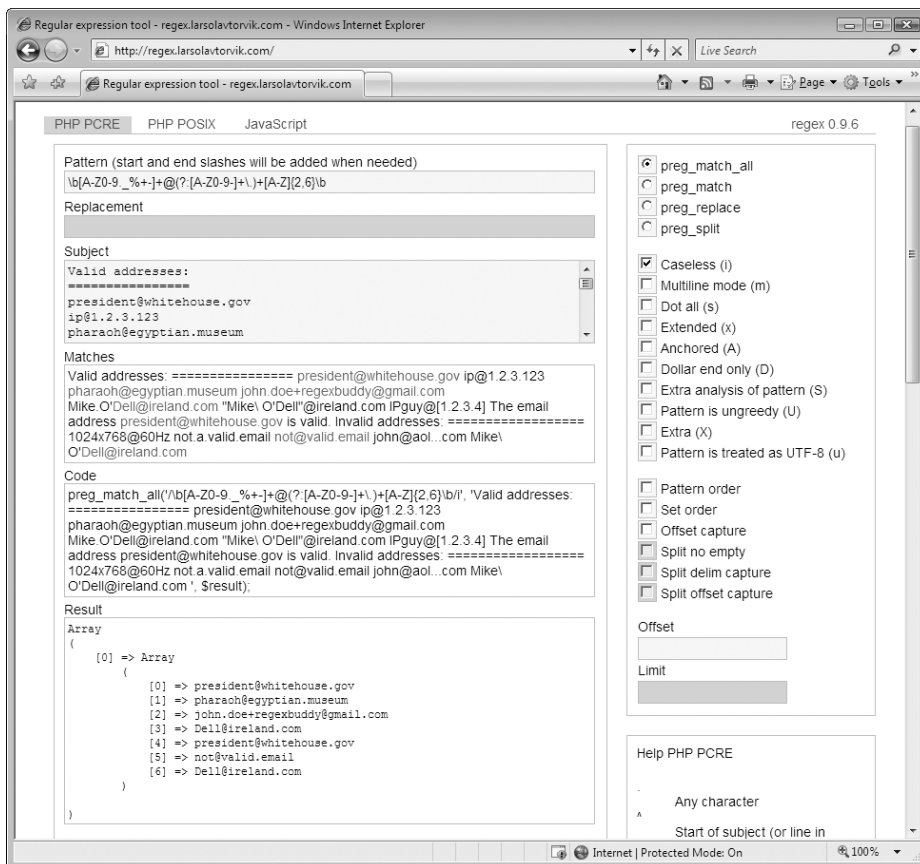


Figura 1.3 – regex.larsolavtorvik.com.

Para começar, selecione o sabor da expressão regular com a qual esteja trabalhando, clicando sobre o nome do sabor no topo da página. Lars oferece PHP PCRE, PHP POSIX e JavaScript. PHP PCRE, sabor regex PCRE discutido neste livro, é usado para funções `preg` PHP. O POSIX é um sabor regex antigo e limitado, usado pelas funções `ereg` do PHP, não discutidas neste livro. Se você selecionar o JavaScript, estará trabalhando com a implementação JavaScript do seu navegador.

Digite sua expressão regular no campo **Pattern** (padrão) e seu texto de assunto no campo **Subject** (assunto). Em seguida, o campo **Matches** (correspondências) exibirá o texto de assunto com as correspondências da regex em destaque. O campo **Code** exibe uma única linha de código-fonte, que aplica sua regex a seu texto de assunto. Copiar e colar

este código em seu editor de código lhe poupará o trabalho enfadonho de converter manualmente sua regex em uma string literal. Qualquer string ou array retornado pelo código é exibido no campo **Result**. Como o Lars utilizou a tecnologia Ajax para construir seu site, os resultados são atualizados em apenas alguns momentos, para todos os sabores. Para utilizar a ferramenta você tem que estar on-line, pois o PHP é processado no servidor, e não no navegador.

A segunda coluna exibe uma lista de comandos e opções regex. Estas dependem do sabor regex. Os comandos regex, tipicamente, incluem operações de correspondência, substituição e divisão. As opções da regex são comuns, tais como não-diferenciação entre letras maiúsculas e minúsculas (case insensitivity), bem como opções específicas a cada implementação. Estes comandos e opções são descritos no capítulo 3.

Nregex

<http://www.nregex.com> (Figura 1.4) é um testador simples, construído com a tecnologia .NET por David Seruyange. Embora o site não diga qual sabor ele implementa, o .NET 1.x era usado no momento da redação deste texto.

O layout da página é um pouco confuso. Digite sua expressão regular no campo, sob o rótulo **Regular Expression**, e defina as opções da regex utilizando as caixas de seleção abaixo do campo. Digite o texto de assunto, na caixa grande que aparece na parte inferior, substituindo o texto padrão `If I just had $5,00 then "she" wouldn't be so @#$$! mad..`. Se o assunto for uma página da Web, digite a URL no campo **Load Target From URL** e clique no botão **Load**, abaixo do campo de entrada. Se o seu assunto for um arquivo em seu disco rígido, clique no botão **Browse**, localize o arquivo desejado e, em seguida, clique no botão **Load**, abaixo do campo de entrada.

Seu texto de assunto aparecerá duplicado no campo **Matches & Replacements** (Correspondências & Substituições) no centro da página web, com as correspondências da regex em destaque. Se você digitar algo no campo **Replacement String** (String de Substituição), o resultado da pesquisa-e-substituição será apresentado. Se sua expressão regular for inválida, aparecerá

A correspondência da regex é feita em um código .NET rodando no servidor, então você precisa estar on-line para funcionar. Se as atualizações automáticas estiverem lentas, talvez seja porque o texto de amostra seja muito longo, então assinale a opção **Manually Evaluate Regex** (Avaliar a Expressão Regular Manualmente), na caixa de seleção acima do campo da sua expressão regular, para mostrar o botão **Evaluate**. Clique nesse botão para atualizar a visualização em **Matches & Replacements**.

Rubular

Michael Lovitt disponibilizou um testador de regex minimalista on-line em <http://www.rubular.com> (Figura 1.5), usando o sabor regex Ruby 1.8.



Figura 1.4 – Nregex.

Digite a sua expressão regular na caixa entre as duas barras, abaixo de “Your Regular Expression”. Você pode ativar a não-diferenciação entre maiúsculas e minúsculas, digitando um `i` na pequena caixa posicionada após a segunda barra. Da mesma forma, se você preferir, ative a opção “a dot matches a line break” (“um ponto corresponde a uma quebra de linha”), digitando um `m` nessa mesma caixa. `im` ativa as duas opções. Embora estas convenções possam parecer confusas se você for iniciante no Ruby, elas satisfazem a sintaxe `/regex/im` usada para especificar uma regex no código-fonte do Ruby.



Figura 1.5 – Rubular.

Digite ou cole seu texto de assunto na caixa “Your test string” e aguarde um instante. Uma nova caixa “Match result” aparecerá à direita, mostrando seu texto de assunto com todas as correspondências da expressão regular em destaque.

myregexp.com

Sergey Evdokimov criou vários testadores de expressão regular para desenvolvedores Java. A página em <http://www.myregexp.com> (Figura 1.6) oferece um testador regex on-line. É um applet Java que roda no seu navegador. O Java 4 (ou posterior) precisa estar instalado em seu computador. O applet usa o pacote `java.util.regex` para avaliar suas expressões regulares, uma funcionalidade nova do Java 4. Neste livro, o sabor regex “Java” refere-se a este pacote.

Digite sua expressão regular na caixa Regular Expression. Use o menu **Flags** para definir as opções da regex que você desejar. Três dessas opções também possuem caixas de seleção diretas.

Se você quiser testar uma regex já existente como uma string de código Java, copie toda a sequência para a área de transferência. No testador `myregexp.com`, clique no menu **Edit** e, em seguida, “Paste Regex from Java String” (“Cole a expressão regular a partir de uma String Java”). No mesmo menu, escolha “Copy Regex for Java Source” (“Copie a expressão regular para um fonte Java”) quando terminar de editar a expressão regular. O menu **Edit** também possui comandos semelhantes para JavaScript e XML.

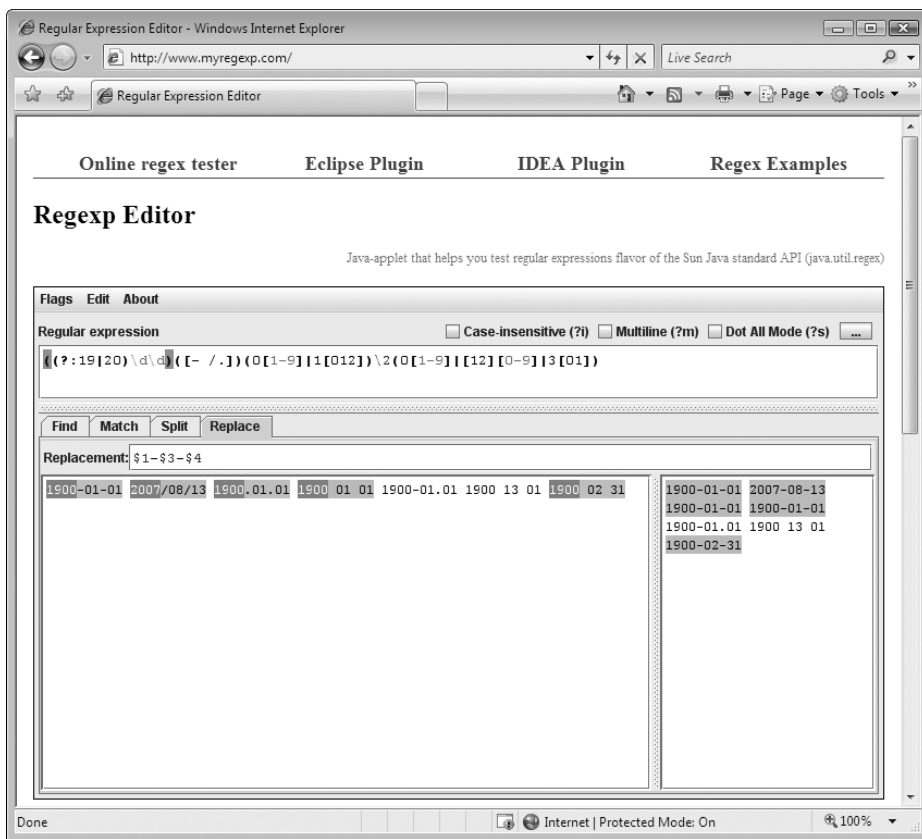


Figura 1.6 – myregex.com

Abaixo da expressão regular, existem quatro guias que rodam quatro testes diferentes:

Find

Destaca todas as correspondências da expressão regular no texto de assunto. Estas são as correspondências encontradas pelo método `Matcher.find()` do Java.

Match

Testa se a expressão regular corresponde inteiramente ao texto de amostra. Caso isso aconteça, todo o texto é realçado. Isto é o que fazem os métodos `String.matches()` e `Matcher.matches()`.

Split

A segunda caixa à direita mostra o array de strings retornado por `String.split()` ou `Pattern.split()`, quando usados com sua expressão regular e o texto de assunto.

Replace

Digite o texto de substituição e a caixa à direita exibirá o texto retornado por `String.replaceAll()` ou `Matcher.replaceAll()`.

Você pode encontrar outros testadores regex de Sergey por meio dos links no topo da página em <http://www.myregex.com>. Um deles é um plug-in para o Eclipse, e o outro, um plug-in para o IntelliJ IDEA.

reAnimator

O reAnimator, de Oliver Steele, encontrado em <http://osteele.com/tools/reanimator> (Figura 1.7), não vai trazer uma regex morta de volta à vida. Pelo contrário, ele é uma pequena e divertida ferramenta que mostra uma representação gráfica das máquinas de estados finitos que um mecanismo de expressão regular utiliza para realizar uma pesquisa de expressão regular.

A sintaxe de expressão regular do reAnimator é muito limitada. É compatível com todos os sabores discutidos neste livro. Qualquer regex que você animar com o reAnimator trabalhará com qualquer um dos sabores deste livro, mas o inverso, definitivamente, não é verdade. Isso acontece porque as expressões regulares do reAnimator são regulares no sentido matemático. A barra lateral “História do termo Expressão Regular”, na página 16, explica isso brevemente.

Comece indo até a caixa **Pattern**, na parte superior da página, e pressione o botão **Edit**. Digite sua expressão regular no campo **Pattern** e clique em **Set**. Digite lentamente o texto de assunto no campo **Input**.

Ao digitar cada caractere, bolas coloridas irão mover-se através da máquina de estados para indicar o ponto final obtido, até o momento, na máquina. Bolas azuis indicam que a máquina de estados aceita a entrada, mas necessita de mais entradas para uma correspondência completa. Bolas verdes indicam que a entrada corresponde ao padrão. Nenhuma bola significa que a máquina de estados não corresponde à entrada.

O reAnimator mostrará uma correspondência apenas se a expressão regular corresponder à string de entrada completa, como se você a tivesse colocado entre âncoras `<^>` e `<$>`. Esta é outra propriedade das expressões que são regulares no sentido matemático.

Mais testadores de expressões regulares para desktop

Expresso

Expresso (não confundir com o café expresso) é um aplicativo .NET para criar e testar expressões regulares. Você pode baixá-lo em <http://www.ultrapico.com/Expresso.htm>. O Framework .NET 2.0, ou posterior, deve estar instalado em seu computador.

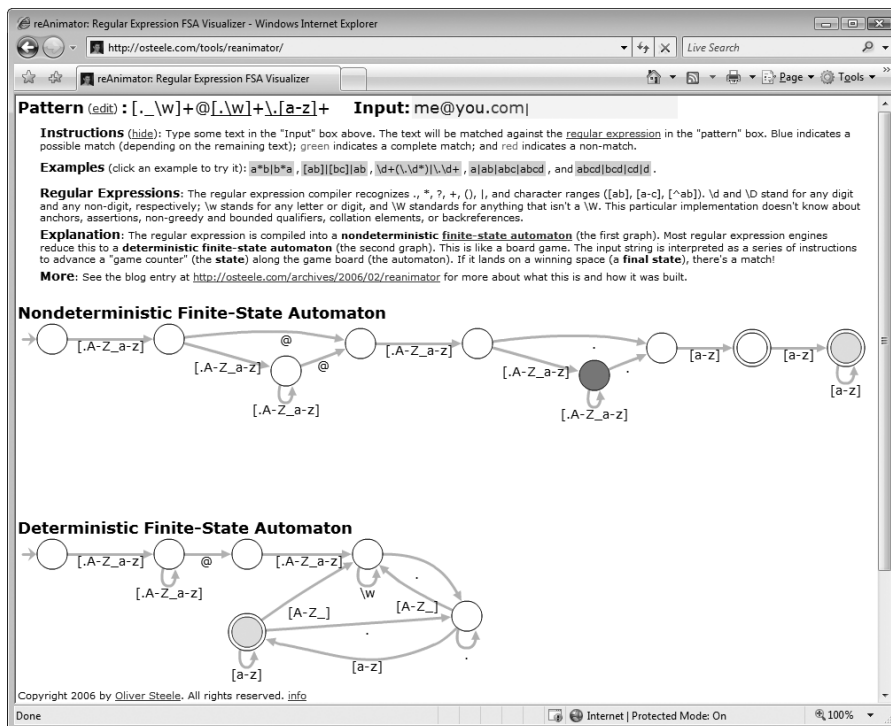


Figura 1.7 – reAnimator.

O programa é gratuito para testes por 60 dias. Após o período de experimentação, é necessário registrá-lo, ou ele irá parar de funcionar. A inscrição é gratuita, mas requer seu endereço de e-mail para a Ultrapico. A chave de registro é enviada por e-mail.

O Expresso exibe uma tela, como a mostrada na figura 1.8. A caixa **Regular Expression**, onde você digita sua expressão regular, fica sempre visível. O realce de sintaxe não está disponível. A caixa **Regex Analyzer** cria automaticamente uma breve análise, em inglês, da sua expressão regular. Ela também fica sempre visível.

No modo **Design**, você pode definir as opções de correspondências, como “**Ignore Case**”, na parte inferior da tela. A maior parte do espaço da tela é ocupado por uma fileira de guias, nas quais você pode selecionar o token de expressão regular que deseja inserir. Se tiver dois monitores, ou um monitor grande, clique no botão **Undock**, para que a fileira de guias flutue. Então, você também poderá construir sua expressão regular no outro modo (**Test Mode**, ou **Modo Teste**).

Em modo de teste, digite, ou cole o texto de assunto no canto inferior esquerdo. Em seguida, clique no botão **Run Match** (Executar Correspondência) para obter uma lista de todas as correspondências na caixa **Search Results** (Resultados da Pesquisa). Não é aplicado nenhum realce ao texto de assunto. Clique em uma correspondência, nos resultados, para selecionar a correspondência no texto de assunto.

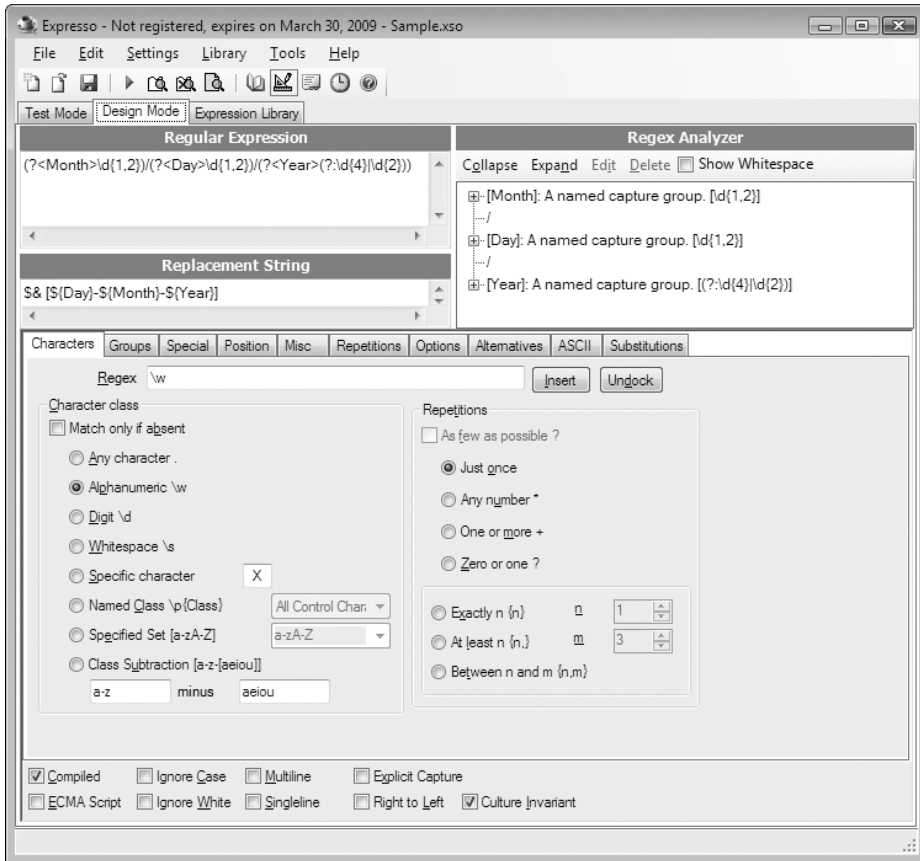


Figura 1.8 – Expresso.

A guia *Expression Library* (Biblioteca de Expressões) mostra uma lista de assuntos de expressões regulares e uma lista de expressões regulares recentes. Sua regex é adicionada à lista cada vez que você pressiona *Run Match*.

Você pode editar a biblioteca por meio do menu *Library*, na barra de menu principal.

The Regulator

The Regulator, que você pode baixar em <http://sourceforge.net/projects/regulator>, não é seguro para uso em mergulhos submarinos ou em botijões de gás de cozinha²; trata-se de outra aplicação .NET para criação e teste de expressões regulares. A versão mais recente requer .NET 2.0 ou posterior. As versões mais antigas para .NET 1.x ainda podem ser baixadas. The Regulator possui código aberto, e nenhum pagamento ou registro é necessário.

² N.T.: Trocadilho com a palavra *regulator*, que pode ser entendida como uma válvula de regulagem.

The Regulator faz tudo em uma tela (Figura 1.9). Na guia *New Document*, você digita sua expressão regular. Realce de sintaxe é aplicado automaticamente, mas os erros de sintaxe em sua regex não ficam óbvios. Clique com o botão direito do mouse para selecionar o token de regex que deseja inserir a partir de um menu. Você pode definir as opções da expressão regular por meio dos botões na barra de ferramentas principal. Os ícones são um pouco enigmáticos. Espere a dica para ver qual opção você está definindo em cada botão.

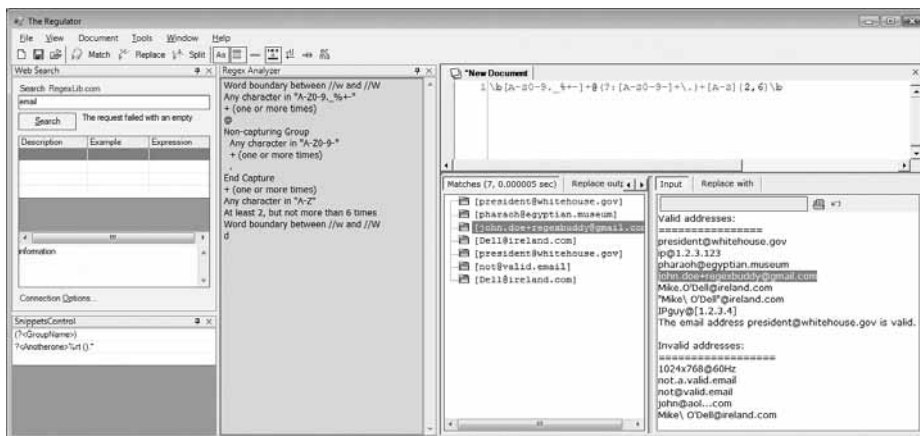


Figura 1.9 – The Regulator.

Abaixo da área para sua regex, à direita, clique no botão *Input*, para exibir o espaço para colar seu texto de assunto. Clique no botão *Replace with* para digitar o texto de substituição, caso queira fazer uma pesquisa-e-substituição. Abaixo da regex, à esquerda, você verá os resultados da sua operação regex. Os resultados não são atualizados automaticamente: você deve clicar no botão *Match*, *Replace* ou *Split*, na barra de ferramentas, para atualizar os resultados. Nenhum realce é aplicado à entrada. Clique em uma correspondência nos resultados para selecioná-la no texto de assunto.

O painel *Regex Analyzer* mostra, em inglês, uma análise simples de sua expressão regular, mas ele não é automático, ou interativo. Para atualizar a análise, selecione *Regex Analyzer* no menu *View*, mesmo que já esteja visível. Clicar sobre a análise somente move o cursor de texto.

grep

O nome *grep* é derivado do comando *g/re/p*, que realizava uma pesquisa de expressão regular no editor de texto *ed* do Unix, uma das primeiras aplicações a suportar expressões regulares. Este comando era tão popular que, atualmente, todos os sistemas Unix possuem um utilitário *grep* dedicado à pesquisa em arquivos utilizando uma

expressão regular. Se você estiver usando Unix, Linux ou Mac OS X, digite `man grep` em uma janela do terminal para aprender tudo sobre ele.

As três ferramentas seguintes são aplicativos do Windows que fazem o mesmo que o `grep`, e muito mais.

PowerGREP

PowerGREP, desenvolvido por Jan Goyvaerts, um dos autores deste livro, é provavelmente a ferramenta `grep` mais completa disponível para a plataforma Microsoft Windows (Figura 1.10). PowerGREP usa um sabor `regex` personalizado, que combina o melhor dos sabores discutidos neste livro. Este sabor é identificado como “JGsoft” dentro do RegxBuddy.

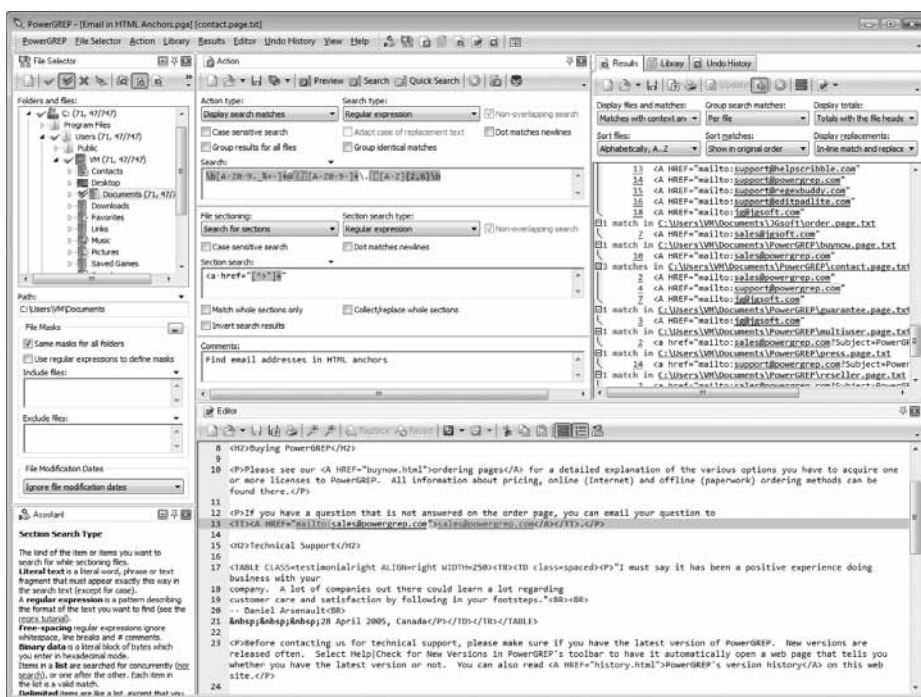


Figura 1.10 – PowerGREP.

Para executar uma busca rápida de expressão regular, basta selecionar **Clear** (Limpar), no menu **Action** (Ação), e digitar sua expressão regular na caixa de busca no painel **Action**. Clique em uma pasta no painel **File Selector** (Seletor de Arquivos), e selecione **Include File or Folder** (Incluir Arquivo ou Pasta) ou **Include Folder and Subfolders** (Incluir Pastas e Subpastas) no menu **File Selector**. Em seguida, selecione **Execute** (Executar), no menu **Action**, para executar sua pesquisa.

Para executar uma pesquisa-e-substituição, selecione **search-and-replace** (pesquisar e substituir) na lista suspensa **action type** (tipo de ação), no canto superior esquerdo do painel **Action**, depois de limpar a ação. A caixa **Replace** (Substituir) aparecerá abaixo da caixa **Search** (Pesquisa). Digite seu texto de substituição. As demais etapas são semelhantes à pesquisa.

O PowerGREG tem a capacidade única de utilizar até três listas de expressões regulares ao mesmo tempo, com qualquer número de expressões regulares em cada lista. Enquanto os dois últimos parágrafos fornecem tudo de que você precisa para executar pesquisas simples, como em qualquer ferramenta **grep**, liberar todo o potencial do PowerGREG pode exigir a leitura da documentação detalhada da ferramenta.

PowerGREG roda em Windows 98, ME, 2000, XP e Vista. Você pode baixar uma cópia de avaliação gratuita em <http://www.powergrep.com/PowerGREPCookbook.exe>. Com exceção da gravação de resultados e bibliotecas, a cópia é totalmente funcional durante 15 dias de uso efetivo. Embora a cópia não salve os resultados mostrados no painel **Results**, ela modificará todos os seus arquivos durante as ações de pesquisa-e-substituição, tal como ocorre na versão completa.

Windows Grep

Windows Grep (<http://www.wingrep.com>) é uma das ferramentas **grep** mais antigas para Windows. Sua idade aparece um pouco na interface de usuário (Figura 1.11), mas ele faz bem o que se propõe a fazer. Ele suporta um sabor de expressão regular limitado, chamado POSIX ERE. No caso das funcionalidades suportadas, ele utiliza a mesma sintaxe que os sabores deste livro. Windows Grep é shareware, o que significa que você pode baixá-lo gratuitamente, mas o pagamento é esperado, caso queira mantê-lo em seu computador.

Para preparar uma pesquisa, selecione **Search** no menu **Search**. A tela que aparece pode ser diferente, dependendo do que você selecionou; **Beginner Mode** (Modo Iniciante) ou **Expert Mode** (Modo Especialista), no menu **Options**. Os iniciantes obtêm um passo-a-passo do assistente, enquanto os especialistas obtêm um diálogo com guias.

Ao configurar uma pesquisa, o Windows Grep imediatamente a executa, apresentando-lhe com uma lista de arquivos em que foram encontradas correspondências. Clique uma vez em um arquivo para ver suas correspondências no painel inferior, e dê um duplo clique para abrir o arquivo. Selecione “**All Matches**” (Todas as Correspondências) no menu **View** para mostrar tudo.

Para executar uma pesquisa-e-substituição, selecione **Replace** no menu **Search**.

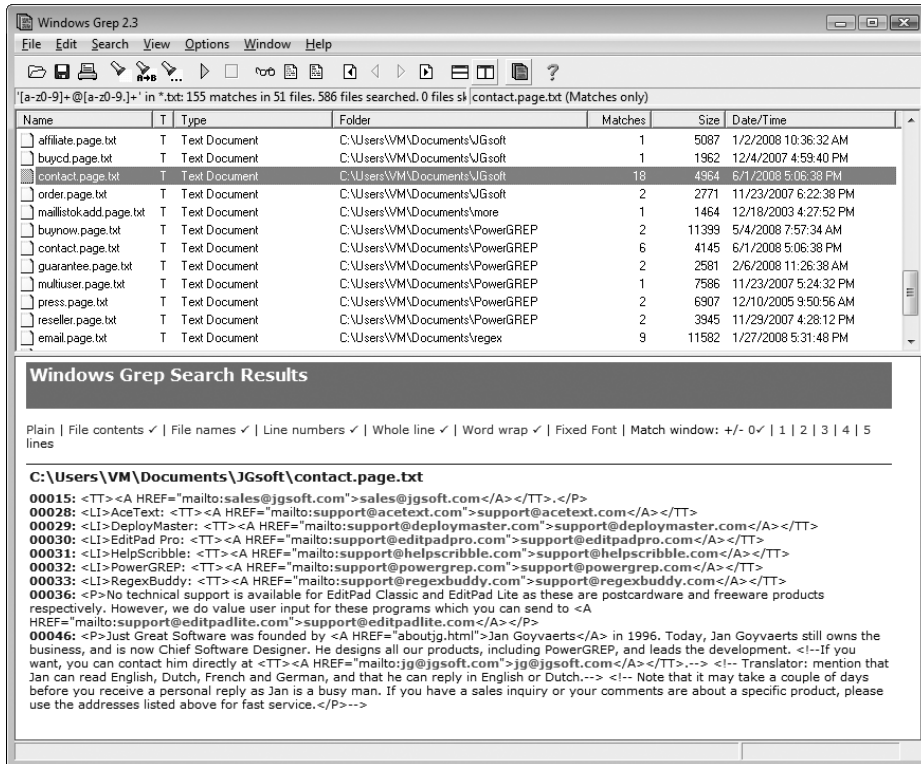


Figura 1.11 – Windows Grep.

RegexRenamer

RegexRenamer (Figura 1.12) não é realmente uma ferramenta grep. Ao invés de pesquisar o conteúdo de arquivos, ele pesquisa e substitui nomes de arquivos. Você pode baixá-lo em <http://regexrenamer.sourceforge.net>. RegexRenamer requer a versão 2.0 ou posterior do Microsoft .NET Framework.

Digite sua expressão regular na caixa Match, e o texto de substituição na caixa Replace. Clique em /i para ativar a não-diferenciação de maiúsculas e minúsculas, e em /g para substituir todas as correspondências encontradas em cada arquivo, em vez de substituir apenas a primeira. /x ativa a sintaxe de espaçamento livre, que não é muito útil, pois você tem apenas uma linha para digitar sua expressão regular.

Use a árvore do lado esquerdo para selecionar a pasta que contém os arquivos que deseja renomear. Você pode definir uma máscara de arquivo ou um filtro de expressão regular no canto superior direito. Isso restringirá a lista de arquivos nos quais sua expressão regular de procura-e-substituição será aplicada. Utilizar uma expressão regular para filtrar e outra para substituir é um procedimento muito mais prático do que tentar executar as duas funções com apenas uma regex.

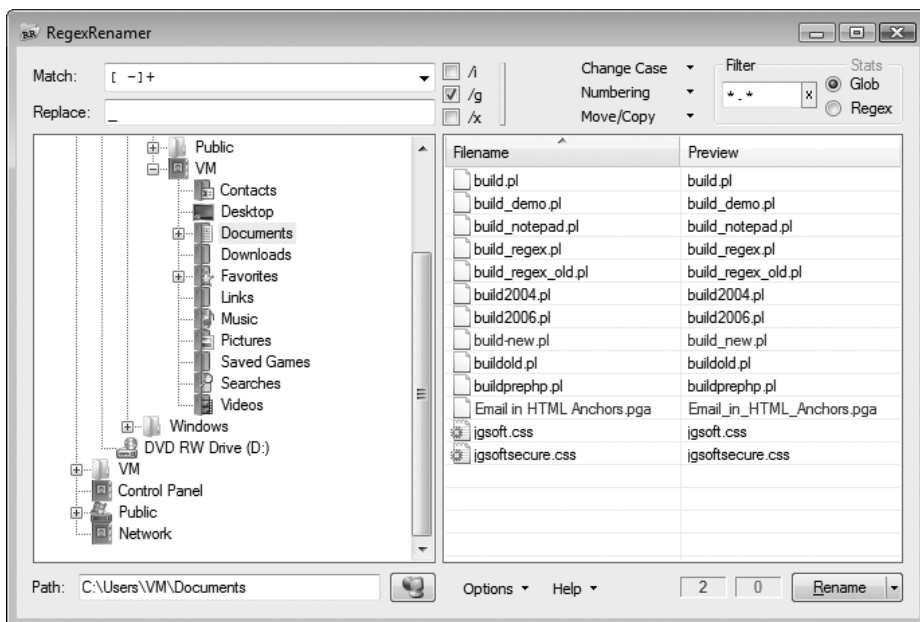


Figura 1.12 – RegExRenamer.

Editores de texto famosos

A maioria dos editores de texto modernos tem pelo menos um suporte básico a expressões regulares. No painel search (pesquisa) ou search-and-replace (pesquisar e substituir), normalmente você encontra uma caixa de seleção para ativar o modo de expressão regular. Alguns editores, como o EditPad Pro, também utilizam expressões regulares em várias funcionalidades de processamento de texto, tais como o realce de sintaxe ou de classes, e as listas de função. A documentação de cada editor explica todas essas funcionalidades. Editores de texto conhecidos, com suporte a expressões regulares, incluem:

- Boxer Text Editor (PCRE)
- Dreamweaver (JavaScript)
- EditPad Pro (sabor personalizado que combina o melhor dos sabores discutidos neste livro, chamado “JGsoft” em RegExBuddy)
- Multi-Edit (PCRE, se você selecionar a opção “Perl”)
- NoteTab (PCRE)
- UltraEdit (PCRE)
- TextMate (Ruby 1.9 [Oniguruma])