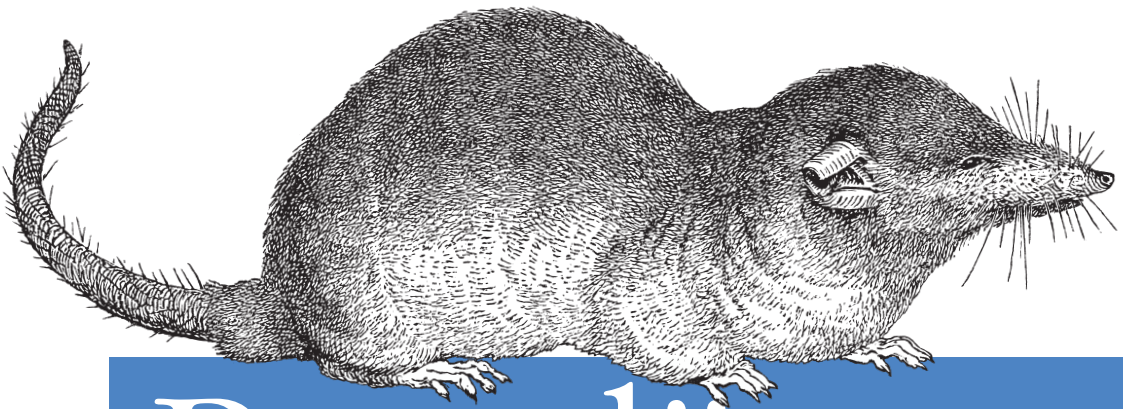
Detaillierte Lösungen
für acht Programmiersprachen

Mit Einstiegs-Tutorial



Reguläre Ausdrücke Kochbuch

Jan Goyvaerts & Steven Levithan
Deutsche Übersetzung von Thomas Demmig

O'REILLY®

Reguläre Ausdrücke Kochbuch

Jan Goyvaerts & Steven Levithan

Deutsche Übersetzung von Thomas Demmig

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Taipei · Tokyo

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
Tel.: 0221/9731600
Fax: 0221/9731608
E-Mail: kommentar@oreilly.de

Die Originalausgabe erschien 2009 unter dem Titel
Regular Expressions Cookbook im Verlag O'Reilly Media, Inc.

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte bibliografische Daten
sind im Internet über *http://dnb.ddb.de* abrufbar.

Lektorat: Alexandra Follenius & Susanne Gerbert, Köln
Korrektorat: Sibylle Feldmann, Düsseldorf
Satz: Tim Mergemeier, Reemers Publishing Services GmbH, Krefeld, www.reemers.de
Umschlaggestaltung: Michael Oreal, Köln
Produktion: Karin Driesen & Andrea Miß, Köln
Belichtung, Druck und buchbinderische Verarbeitung:
Druckerei Kösel, Krugzell; www.koeselbuch.de

ISBN 978-3-89721-957-1

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Vorwort

Im letzten Jahrzehnt ist die Beliebtheit regulärer Ausdrücke deutlich angestiegen. Heutzutage gibt es in allen verbreiteten Programmiersprachen mächtige Bibliotheken zur Verarbeitung regulärer Ausdrücke. Zum Teil bietet die Sprache sogar selbst die entsprechenden Möglichkeiten. Viele Entwickler nutzen diese Features, um den Anwendern ihrer Applikationen das Suchen und Filtern der Daten mithilfe regulärer Ausdrücke zu ermöglichen. Reguläre Ausdrücke sind überall.

Es gibt viele Bücher, die sich mit regulären Ausdrücken befassen. Die meisten machen ihren Job ganz gut – sie erklären die Syntax und enthalten ein paar Beispiele sowie eine Referenz. Aber es gibt keine Bücher, die Lösungen vorstellen. Lösungen, die auf regulären Ausdrücken basieren und für eine ganze Reihe von praktischen Problemen aus der realen Welt genutzt werden können. Bei solchen Problemen geht es vor allem um Fragen zu Texten auf einem Computer und um Internetanwendungen. Wir, Steve und Jan, haben uns dazu entschieden, diese Lücke mit diesem Buch zu füllen.

Wir wollten vor allem zeigen, wie Sie reguläre Ausdrücke in Situationen verwenden können, in denen weniger Erfahrene im Umgang mit regulären Ausdrücken sagen würden, das sei nicht möglich, oder in denen Softwarepuristen der Meinung sind, ein regulärer Ausdruck sei nicht das richtige Tool für diese Aufgabe. Da reguläre Ausdrücke heute überall zu finden sind, sind sie oft auch als Tool verfügbar, das von Endanwendern genutzt werden kann. Auch Programmierer können durch die Verwendung einiger weniger regulärer Ausdrücke viel Zeit sparen, etwa wenn sie Informationen suchen und verändern müssen. Die Alternative ist oft, Stunden oder Tage mit dem Umsetzen in prozeduralen Code zu verbringen oder eine Bibliothek von dritter Seite zu nutzen.

Gefangen im Gewirr der verschiedenen Versionen

Vergleichbar mit anderen beliebten Dingen der IT-Branche, gibt es auch reguläre Ausdrücke in vielen unterschiedlichen Ausprägungen mit unterschiedlicher Kompatibilität. Das hat dazu geführt, dass es diverse *Varianten* eines regulären Ausdrucks gibt, die sich nicht immer gleich verhalten oder die teilweise gar nicht funktionieren.

Viele Bücher erwähnen, dass es unterschiedliche Varianten gibt, und führen auch einige der Unterschiede auf. Aber immer wieder lassen sie bestimmte Varianten unerwähnt – vor allem wenn in diesen Varianten Features fehlen –, statt auf alternative Lösungen und Workarounds hinzuweisen. Das ist frustrierend, wenn Sie mit unterschiedlichen Varianten regulärer Ausdrücke in den verschiedenen Anwendungen oder Programmiersprachen arbeiten müssen.

Saloppe Bemerkungen in der Literatur wie „jeder nutzt mittlerweile reguläre Ausdrücke im Perl-Stil" bagatellisieren leider eine ganze Reihe von Inkompatibilitäten. Selbst Pakete im „Perl-Stil" besitzen entscheidende Unterschiede, und Perl entwickelt sich ja auch noch weiter. Solche oberflächlichen Äußerungen können für Programmierer trostlose Folgen haben und zum Beispiel dazu führen, dass sie eine halbe Stunde oder mehr damit verbringen, nutzlos im Debugger herumzustochern, statt die Details ihrer Implementierung für reguläre Ausdrücke zu kontrollieren. Selbst wenn sie herausfinden, dass ein Feature, auf dem sie aufbauen, nicht vorhanden ist, wissen sie nicht immer, wie sie stattdessen vorgehen können.

Dieses Buch ist das erste, das die am meisten verbreiteten und umfangreichsten Varianten regulärer Ausdrücke nebeneinander aufführt – und zwar durchgängig im ganzen Buch.

Für wen dieses Buch gedacht ist

Sie sollten dieses Buch lesen, wenn Sie regelmäßig am Computer mit Text zu tun haben – egal ob Sie einen Stapel Dokumente durchsuchen, Text in einem Texteditor bearbeiten oder Software entwickeln, die Text durchsuchen oder verändern soll. Reguläre Ausdrücke sind für diese Aufgaben exzellente Hilfsmittel. Das *Reguläre Ausdrücke Kochbuch* erklärt Ihnen alles, was Sie über reguläre Ausdrücke wissen müssen. Sie brauchen kein Vorwissen, da wir selbst einfachste Aspekte regulärer Ausdrücke erklären werden.

Wenn Sie schon Erfahrung mit regulären Ausdrücken haben, werden Sie eine Menge Details kennenlernen, die in anderen Büchern oder Onlineartikeln häufig einfach übergangen werden. Sind Sie jemals über eine Regex gestolpert, die in einer Anwendung funktioniert hat, in einer anderen aber nicht, werden Sie die in diesem Buch detailliert und gleichwertig behandelten Beschreibungen zu sieben der verbreitetsten Varianten regulärer Ausdrücke sehr hilfreich finden. Wir haben das ganze Buch als Kochbuch aufgebaut, sodass Sie direkt zu den Themen springen können, die Sie interessieren. Wenn Sie dieses Buch von vorne bis hinten durchlesen, werden Sie am Ende Meister regulärer Ausdrücke sein.

Mit diesem Buch erfahren Sie alles, was Sie über reguläre Ausdrücke wissen müssen, und noch ein bisschen mehr – unabhängig davon, ob Sie Programmierer sind oder nicht. Wenn Sie reguläre Ausdrücke in einem Texteditor nutzen wollen, in einem Suchtool oder in irgendeiner Anwendung, die ein Eingabefeld „Regex" enthält, können Sie dieses Buch auch ganz ohne Programmiererfahrung lesen. Die meisten Rezepte bieten Lösungen an, die allein auf einem oder mehreren regulären Ausdrücken basieren.

Die Programmierer unter Ihnen erhalten in Kapitel 3 alle notwendigen Informationen zum Implementieren regulärer Ausdrücke in ihrem Quellcode. Dieses Kapitel geht davon aus, dass Sie mit den grundlegenden Features der Programmiersprache Ihrer Wahl vertraut sind, aber Sie müssen keine Erfahrung mit dem Einsatz regulärer Ausdrücke in Ihrem Quellcode mitbringen.

Behandelte Technologien

.NET, Java, JavaScript, PCRE, Perl, Python und Ruby kommen nicht ohne Grund auf dem Rückseitentext vor. Vielmehr stehen diese Begriffe für die sieben Varianten regulärer Ausdrücke, die in diesem Buch behandelt werden, wobei alle sieben gleichermaßen umfassend beschrieben werden. Insbesondere haben wir versucht, alle Uneinheitlichkeiten zu beschreiben, die wir in diesen verschiedenen Varianten finden konnten.

Das Kapitel zur Programmierung (Kapitel 3) enthält Code-Listings in C#, Java, JavaScript, PHP, Perl, Python, Ruby und VB.NET. Auch hier gibt es zu jedem Rezept Lösungen und Erläuterungen für alle acht Sprachen. Damit gibt es in diesem Kapitel zwar einige Wiederholungen, aber Sie können die Abhandlungen über Sprachen, an denen Sie nicht interessiert sind, gern überspringen, ohne etwas in der Sprache zu verpassen, die Sie selbst anwenden.

Aufbau des Buchs

In den ersten drei Kapiteln dieses Buchs geht es um nützliche Tools und grundlegende Informationen, die eine Basis für die Verwendung regulärer Ausdrücke bilden. Jedes der folgenden Kapitel stellt dann eine Reihe von regulären Ausdrücken vor, die bestimmte Bereiche der Textbearbeitung behandeln.

Kapitel 1, *Einführung in reguläre Ausdrücke*, erläutert die Rolle regulärer Ausdrücke und präsentiert eine Reihe von Tools, die das Erlernen, Aufbauen und Debuggen erleichtern.

Kapitel 2, *Grundlagen regulärer Ausdrücke*, beschreibt alle Elemente und Features regulärer Ausdrücke zusammen mit wichtigen Hinweisen zu einer effektiven Nutzung.

Kapitel 3, *Mit regulären Ausdrücken programmieren*, stellt Coding-Techniken vor und enthält Codebeispiele für die Verwendung regulärer Ausdrücke in jeder der in diesem Buch behandelten Programmiersprachen.

Kapitel 4, *Validierung und Formatierung*, enthält Rezepte für den Umgang mit typischen Benutzereingaben, wie zum Beispiel Datumswerten, Telefonnummern und Postleitzahlen in den verschiedenen Staaten.

Kapitel 5, *Wörter, Zeilen und Sonderzeichen*, behandelt häufig auftretende Textbearbeitungsaufgaben, wie zum Beispiel das Testen von Zeilen auf die An- oder Abwesenheit bestimmter Wörter.

Kapitel 6, *Zahlen*, zeigt, wie man Integer-Werte, Gleitkommazahlen und viele andere Formate in diesem Bereich aufspürt.

Kapitel 7, *URLs, Pfade und Internetadressen*, zeigt Ihnen, wie Sie mit den Strings umgehen, die im Internet und in Windows-Systemen für das Auffinden von Inhalten genutzt werden.

Kapitel 8, *Markup und Datenaustausch*, dreht sich um das Bearbeiten von HTML, XML, Comma-Separated Values (CSV) und Konfigurationsdateien im INI-Stil.

Konventionen in diesem Buch

Die folgenden typografischen Konventionen werden in diesem Buch genutzt:

Kursiv
: Steht für neue Begriffe, URLs, E-Mail-Adressen, Dateinamen und Dateierweiterungen.

`Feste Breite`
: Wird genutzt für Programme, Programmelemente wie Variablen oder Funktionsnamen, Werte, die das Ergebnis einer Ersetzung mithilfe eines regulären Ausdrucks sind, und für Elemente oder Eingabetexte, die einem regulären Ausdruck übergeben werden. Dabei kann es sich um den Inhalt eines Textfelds in einer Anwendung handeln, um eine Datei auf der Festplatte oder um den Inhalt einer String-Variablen.

`Feste Breite, kursiv`
: Zeigt Text, der vom Anwender oder durch den Kontext bestimmt werden sollte.

`‹Regulärer●Ausdruck›`
: Steht für einen regulären Ausdruck, entweder allein oder so, wie Sie ihn in das Suchfeld einer Anwendung eingeben würden. Leerzeichen in regulären Ausdrücken werden durch graue Kreise wiedergegeben, außer im Free-Spacing-Modus.

`«Text●zu●ersetzen»`
: Steht für den Text, der bei einer Suchen-und-Ersetzen-Operation durch den regulären Ausdruck gefunden wird und dann ersetzt werden soll. Leerzeichen im zu ersetzenden Text werden mithilfe grauer Kreise dargestellt.

`Gefundener Text`
: Steht für den Teil des Texts, der zu einem regulären Ausdruck passt.

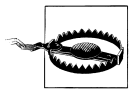
`···`
: Graue Punkte in einem regulären Ausdruck weisen darauf hin, dass Sie diesen Bereich erst mit Leben füllen müssen, bevor Sie den regulären Ausdruck nutzen können. Der Begleittext erklärt, was Sie dort eintragen können.

`CR`, `LF` *und* `CRLF`
: CR, LF und CRLF in Rahmen stehen für die echten Zeichen zum Zeilenumbruch in Strings und nicht für die Escape-Zeichen `\r`, `\n` und `\r\n`. Solche Strings können entstehen, wenn man in einem mehrzeiligen Eingabefeld die Eingabetaste drückt oder im Quellcode mehrzeilige String-Konstanten genutzt werden, wie zum Beispiel die Verbatim-Strings in C# oder Strings mit dreifachen Anführungszeichen in Python.

↵

Der „Wagenrücklauf“-Pfeil, den Sie vielleicht auf Ihrer Tastatur auf der Eingabetaste sehen, wird genutzt, wenn wir eine Zeile auftrennen müssen, damit sie auf die Druckseite passt. Geben Sie den Text in Ihrem Quellcode ein, sollten Sie hier nicht die Eingabetaste drücken, sondern alles auf einer Zeile belassen.

Dieses Icon steht für einen Tipp, einen Vorschlag oder eine allgemeine Anmerkung.



Dieses Icon steht für einen Warnhinweis.

Die Codebeispiele verwenden

Dieses Buch ist dazu da, Ihnen bei Ihrer Arbeit zu helfen. Sie können den Code dieses Buchs in Ihren Programmen und Dokumentationen verwenden. Sie brauchen uns nicht um Erlaubnis zu fragen, solange Sie nicht einen beachtlichen Teil des Codes wiedergeben. Beispielsweise benötigen Sie keine Erlaubnis, um ein Programm zu schreiben, das einige Codeteile aus diesem Buch verwendet. Für den Verkauf oder die Verbreitung einer CD-ROM mit Beispielen aus O’Reilly-Büchern brauchen Sie auf jeden Fall unsere Erlaubnis. Die Beantwortung einer Frage durch das Zitieren dieses Buchs und seiner Codebeispiele benötigt wiederum keine Erlaubnis. Wenn Sie einen erheblichen Teil der Codebeispiele dieses Buchs in die Dokumentation Ihres Produkts einfügen, brauchen Sie eine Erlaubnis.

Wir freuen uns über einen Herkunftsnachweis, bestehen aber nicht darauf. Eine Referenz enthält i.d.R. Titel, Autor, Verlag und ISBN, zum Beispiel: „*Reguläre Ausdrücke Kochbuch* von Jan Goyvaerts & Steven Levithan, Copyright 2010, O’Reilly Verlag, ISBN 978-3-89721-957-1.“

Wenn Sie denken, Ihre Verwendung unserer Codebeispiele könnte den angemessenen Gebrauch oder die hier erteilte Erlaubnis überschreiten, nehmen Sie einfach mit uns über *kommentar@oreilly.de* Kontakt auf.

Danksagung

Wir danken Andy Oram, unserem Lektor bei O’Reilly Media, Inc., für seine Begleitung bei diesem Projekt vom Anfang bis zum Ende. Ebenso danken wir Jeffrey Friedl, Zak Greant, Nikolaj Lindberg und Ian Morse für ihre sorgfältigen fachlichen Korrekturen, durch die dieses Buch umfassender und genauer wurde.

KAPITEL 1

Einführung in reguläre Ausdrücke

Wenn Sie dieses Kochbuch aufgeschlagen haben, sind Sie wahrscheinlich schon ganz erpicht darauf, ein paar der hier beschriebenen seltsamen Strings mit Klammern und Fragezeichen in Ihren Code einzubauen. Falls Sie schon so weit sind: nur zu! Die verwendbaren regulären Ausdrücke sind in den Kapiteln 4 bis 8 aufgeführt und beschrieben.

Aber wenn Sie zunächst die ersten Kapitel dieses Buchs lesen, spart Ihnen das langfristig möglicherweise eine Menge Zeit. So finden Sie in diesem Kapitel zum Beispiel eine Reihe von Hilfsmitteln – einige wurden von Jan, einem der beiden Autoren, erstellt –, mit denen Sie einen regulären Ausdruck testen und debuggen können, bevor Sie ihn in Ihrem Code vergraben, wo Fehler viel schwieriger zu finden sind. Außerdem zeigen Ihnen diese ersten Kapitel, wie Sie die verschiedenen Features und Optionen regulärer Ausdrücke nutzen können, um Ihr Leben leichter zu machen. Sie werden verstehen, wie reguläre Ausdrücke funktionieren, um deren Performance zu verbessern, und Sie lernen die subtilen Unterschiede kennen, die in den verschiedenen Programmiersprachen existieren – selbst in unterschiedlichen Versionen Ihrer bevorzugten Programmiersprache.

Wir haben uns also mit diesen Grundlageninformationen viel Mühe gegeben und sind recht zuversichtlich, dass Sie sie lesen werden, bevor Sie mit der Anwendung regulärer Ausdrücke beginnen – oder spätestens dann, wenn Sie nicht mehr weiterkommen und Ihr Wissen aufstocken wollen.

Definition regulärer Ausdrücke

Im Rahmen dieses Buchs ist ein *regulärer Ausdruck* eine bestimmte Art von Textmuster, das Sie in vielen modernen Anwendungen und Programmiersprachen nutzen können. Mit ihm können Sie prüfen, ob eine Eingabe zu einem Textmuster passt, Sie können Texte eines bestimmten Musters in einer größeren Datei finden, durch anderen Text oder durch eine veränderte Version des bisherigen Texts ersetzen, einen Textabschnitt in eine Reihe von Unterabschnitten aufteilen – oder auch sich selbst ins Knie schießen. Dieses Buch hilft Ihnen dabei, genau zu verstehen, was Sie tun, um Katastrophen zu vermeiden.

Geschichte des Begriffs „regulärer Ausdruck“

Der Begriff *regulärer Ausdruck* kommt aus der Mathematik und der theoretischen Informatik. Dort steht er für eine Eigenschaft mathematischer Ausdrücke namens *Regularität*. Solch ein Ausdruck kann als Software mithilfe eines deterministischen endlichen Automaten (DEA) implementiert werden. Ein DEA ist ein endlicher Automat, der kein Backtracking nutzt.

Die Textmuster, die von den ersten *grep*-Tools genutzt wurden, waren reguläre Ausdrücke im mathematischen Sinn. Auch wenn der Name geblieben ist, sind aktuelle reguläre Ausdrücke im Perl-Stil keine regulären Ausdrücke im mathematischen Sinn. Sie sind mit einem nicht deterministischen endlichen Automaten (NEA) implementiert. Später werden Sie noch mehr über Backtracking erfahren. Alles, was ein normaler Entwickler aus diesem Textkasten mitnehmen muss, ist, dass ein paar Informatiker in ihren Elfenbeintürmen sehr verärgert darüber sind, dass ihr wohldefinierter Begriff durch eine Technologie überlagert wurde, die in der realen Welt viel nützlicher ist.

Wenn Sie reguläre Ausdrücke sinnvoll einsetzen, vereinfachen sie viele Programmier- und Textbearbeitungsaufgaben oder ermöglichen gar erst deren Umsetzung. Ohne sie bräuchten Sie Dutzende, wenn nicht Hunderte von Zeilen prozeduralen Codes, um zum Beispiel alle E-Mail-Adressen aus einem Dokument zu ziehen – Code, der nicht besonders spannend zu schreiben ist und der sich auch nur schwer warten lässt. Mit dem passenden regulären Ausdruck, wie er in Rezept 4.1, zu finden ist, braucht man nur ein paar Zeilen Code, wenn nicht sogar nur eine einzige.

Aber wenn Sie versuchen, mit einem einzelnen regulären Ausdruck zu viel auf einmal zu machen, oder wenn Sie Regexes auch dort nutzen, wo sie eigentlich nicht sinnvoll sind, werden Sie folgendes Statement nachvollziehen können:[1]

> Wenn sich manche Menschen einem Problem gegenübersehen, denken sie: „Ah, ich werde reguläre Ausdrücke nutzen.“ Jetzt haben sie zwei Probleme.

Dieses zweite Problem taucht jedoch nur auf, wenn diese Menschen die Anleitung nicht gelesen haben, die Sie gerade in den Händen halten. Lesen Sie weiter. Reguläre Ausdrücke sind ein mächtiges Tool. Wenn es bei Ihrer Arbeit darum geht, Text auf einem Computer zu bearbeiten oder zu extrahieren, wird Ihnen ein solides Grundwissen über reguläre Ausdrücke viele Überstunden ersparen.

Viele Varianten regulärer Ausdrücke

Okay, der Titel des vorigen Abschnitts war gelogen. Wir haben gar nicht definiert, was reguläre Ausdrücke sind. Das können wir auch nicht. Es gibt keinen offiziellen Standard, der genau definiert, welche Textmuster reguläre Ausdrücke sind und welche nicht. Wie

1 Jeffrey Friedl hat die Geschichte dieses Zitats in seinem Blog verfolgt: *http://regex.info/blog/2006-09-15/247*.

Sie sich vorstellen können, hat jeder Designer einer Programmiersprache und jeder Entwickler einer textverarbeitenden Anwendung eine andere Vorstellung davon, was genau ein regulärer Ausdruck tun sollte. Daher sehen wir uns einem ganzen Reigen von *Varianten* regulärer Ausdrücke gegenüber.

Glücklicherweise sind die meisten Designer und Entwickler faul. Warum sollte man etwas total Neues aufbauen, wenn man kopieren kann, was schon jemand anderer gemacht hat? Im Ergebnis lassen sich alle modernen Varianten regulärer Ausdrücke, einschließlich derer, die in diesem Buch behandelt werden, auf die Programmiersprache Perl zurückverfolgen. Wir nennen diese Varianten *reguläre Ausdrücke im Perl-Stil*. Ihre Syntax ist sehr ähnlich und meist auch kompatibel, aber eben nicht immer.

Autoren sind ebenfalls faul. Meistens verwenden wir den Ausdruck *Regex* oder *Regexp*, um einen einzelnen regulären Ausdruck zu bezeichnen, und *Regexes* für den Plural.

Regex-Varianten entsprechen nicht eins zu eins den Programmiersprachen. Skriptsprachen haben meist ihre eigene, eingebaute Regex-Variante. Andere Programmiersprachen nutzen Bibliotheken, um eine Unterstützung regulärer Ausdrücke anzubieten. Manche Bibliotheken gibt es für mehrere Sprachen, während man bei anderen Sprachen auch aus unterschiedlichen Bibliotheken wählen kann.

Dieses einführende Kapitel kümmert sich nur um die Varianten regulärer Ausdrücke und ignoriert vollständig irgendwelche Programmierüberlegungen. Kapitel 3 beginnt dann mit den Codebeispielen, sodass Sie schon mal in Kapitel 3, *Mit regulären Ausdrücken programmieren*, spicken könnten, um herauszufinden, mit welchen Varianten Sie arbeiten werden. Aber ignorieren Sie erst einmal den ganzen Programmierkram. Die im nächsten Abschnitt vorgestellten Tools ermöglichen einen viel einfacheren Weg, die Regex-Syntax durch „Learning by Doing“ kennenzulernen.

Regex-Varianten in diesem Buch

In diesem Buch haben wir die Regex-Varianten ausgewählt, die heutzutage am verbreitetsten sind. Es handelt sich bei allen um Regex-Varianten im *Perl-Stil*. Manche der Varianten besitzen mehr Features als andere. Aber wenn zwei Varianten das gleiche Feature besitzen, haben sie meist auch die gleiche Syntax dafür. Wir werden auf die wenigen, aber nervigen Unregelmäßigkeiten hinweisen, wenn wir ihnen begegnen.

All diese Regex-Varianten sind Teile von Programmiersprachen und Bibliotheken, die aktiv entwickelt werden. Aus der Liste der Varianten können Sie ersehen, welche Versionen in diesem Buch behandelt werden. Im weiteren Verlauf des Buchs führen wir die Varianten ohne Versionen auf, wenn die vorgestellte Regex überall gleich funktioniert. Das ist nahezu immer der Fall. Abgesehen von Fehlerkorrekturen, die Grenzfälle betreffen, ändern sich Regex-Varianten im Allgemeinen nicht, es sei denn, es werden neue Features ergänzt, die einer Syntax Bedeutung verleihen, die vorher als Fehler angesehen wurde:

Perl

Perls eingebaute Unterstützung für reguläre Ausdrücke ist der Hauptgrund dafür, dass Regexes heute so beliebt sind. Dieses Buch behandelt Perl 5.6, 5.8 und 5.10.

Viele Anwendungen und Regex-Bibliotheken, die behaupten, reguläre Ausdrücke im Perl- oder einem zu Perl kompatiblen Stil zu nutzen, tun das meist gar nicht. In Wirklichkeit nutzen sie eine Regex-Syntax, die der von Perl ähnlich ist, aber nicht die gleichen Features unterstützt. Sehr wahrscheinlich verwenden sie eine der Regex-Varianten aus dem Rest der Liste. Diese Varianten nutzen alle den Perl-Stil.

PCRE

PCRE ist die C-Bibliothek „Perl-Compatible Regular Expressions", die von Philip Hazel entwickelt wurde. Sie können diese Open Source-Bibliothek unter *http://www.pcre.org* herunterladen. Dieses Buch behandelt die Versionen 4 bis 7.

Obwohl die PCRE behauptet, zu Perl kompatibel zu sein, und dies vermutlich mehr als alle anderen Varianten in diesem Buch auch ist, setzt sie eigentlich nur den Perl-Stil um. Manche Features, wie zum Beispiel die Unicode-Unterstützung, sind etwas anders, und Sie können keinen Perl-Code mit Ihrer Regex mischen, wie es bei Perl selbst möglich ist.

Aufgrund der Open Source-Lizenz und der ordentlichen Programmierung hat die PCRE Eingang in viele Programmiersprachen und Anwendungen gefunden. Sie ist in PHP eingebaut und in vielen Delphi-Komponenten verpackt. Wenn eine Anwendung behauptet, reguläre Ausdrücke zu nutzen, die „zu Perl kompatibel" sind, ohne aufzuführen, welche Regex-Variante genau genutzt wird, ist es sehr wahrscheinlich PCRE.

.NET

Das .NET Framework von Microsoft stellt über das Paket `System.Text.RegularExpressions` eine vollständige Regex-Variante im Perl-Stil bereit. Dieses Buch behandelt die .NET-Versionen 1.0 bis 3.5. Im Prinzip gibt es aber nur zwei Versionen von `System.Text.RegularExpressions`: 1.0 und 2.0. In .NET 1.1, 3.0 und 3.5 gab es keine Änderungen an den Regex-Klassen.

Jede .NET-Programmiersprache, unter anderem C#, VB.NET, Delphi for .NET und sogar COBOL.NET, hat einen vollständigen Zugriff auf die .NET-Variante der Regexes. Wenn eine Anwendung, die mit .NET entwickelt wurde, eine Regex-Unterstützung anbietet, können Sie ziemlich sicher sein, dass sie die .NET-Variante nutzt, selbst wenn sie behauptet, „reguläre Ausdrücke von Perl" zu nutzen. Eine wichtige Ausnahme bildet das Visual Studio (VS) selbst. Die in VS integrierte Entwicklungsumgebung (IDE) nutzt immer noch die gleiche alte Regex-Variante, die sie von Anfang an unterstützt hat. Und die ist überhaupt nicht im Perl-Stil nutzbar.

Java

Java 4 ist das erste Java-Release, das durch das Paket `java.util.regex` eine eingebaute Unterstützung regulärer Ausdrücke anbietet. Schnell wurden dadurch die verschiedenen Regex-Bibliotheken von dritter Seite verdrängt. Abgesehen davon, dass es sich um ein Standardpaket handelt, bietet es auch einen vollständige Regex-Variante im Perl-Stil an und hat eine ausgezeichnete Performance, selbst im Vergleich zu

Anwendungen, die in C geschrieben wurden. Dieses Buch behandelt das Paket `java.util.regex` in Java 4, 5 und 6.

Wenn Sie Software verwenden, die in den letzten paar Jahren mit Java entwickelt wurden, wird jede Unterstützung regulärer Ausdrücke vermutlich auf die Java-Variante zurückgreifen.

JavaScript

In diesem Buch nutzen wir den Begriff *JavaScript*, um damit die Variante regulärer Ausdrücke zu beschreiben, die in Version 3 des ECMA-262-Standards definiert ist. Dieser Standard ist die Basis der Programmiersprache ECMAScript. Diese ist besser bekannt durch ihre Implementierungen JavaScript und JScript in den verschiedenen Webbrowsern. Internet Explorer 5.5 bis 8.0, Firefox, Opera und Safari implementieren alle Version 3 von ECMA-262. Trotzdem gibt es in allen Browsern unterschiedliche Grenzfälle, in denen sie vom Standard abweichen. Wir weisen auf solche Probleme hin, wenn sie eine Rolle spielen.

Ist es möglich, auf einer Website mit einem regulären Ausdruck suchen oder filtern zu können, ohne auf eine Antwort vom Webserver warten zu müssen, wird die Regex-Variante von JavaScript genutzt, die die einzige browserübergreifende Regex-Variante auf Clientseite ist. Selbst VBScript von Microsoft und ActionScript 3 von Adobe nutzen sie.

Python

Python unterstützt reguläre Ausdrücke durch sein Modul `re`. Dieses Buch behandelt Python 2.4 und 2.5. Die Unterstützung regulärer Ausdrücke in Python hat sich seit vielen Jahren nicht geändert.

Ruby

Die Unterstützung regulärer Ausdrücke in Ruby ist wie bei Perl Teil der Sprache selbst. Dieses Buch behandelt Ruby 1.8 und 1.9. Eine Standardkompilierung von Ruby 1.8 verwendet die Variante regulärer Ausdrücke, die direkt im Quellcode von Ruby implementiert ist. Eine Standardkompilierung von Ruby 1.9 nutzt die Oniguruma-Bibliothek für reguläre Ausdrücke. Ruby 1.8 kann so kompiliert werden, dass es Oniguruma verwendet, und Ruby 1.9 so, dass es die ältere Ruby-Regex-Variante nutzt. In diesem Buch bezeichnen wir die native Ruby-Variante als Ruby 1.8 und die Oniguruma-Variante als Ruby 1.9.

Um herauszufinden, welche Ruby-Regex-Variante Ihre Site nutzt, versuchen Sie, den regulären Ausdruck `‹a++›` zu verwenden. Ruby 1.8 wird Ihnen mitteilen, dass der reguläre Ausdruck ungültig ist, da es keine possessiven Quantoren unterstützt, während Ruby 1.9 damit einen String finden wird, der aus einem oder mehreren Zeichen besteht.

Die Oniguruma-Bibliothek ist so entworfen, dass sie abwärtskompatibel zu Ruby 1.8 ist und neue Features so ergänzt, dass keine bestehenden Regexes ungültig werden. Die Implementatoren haben sogar Features darin gelassen, die man wohl besser entfernt hätte, wie zum Beispiel die Verwendung von `(?m)`, mit der der Punkt auch Zeilenumbrüche findet, wofür andere Varianten `(?s)` nutzen.

Suchen und Ersetzen mit regulären Ausdrücken

Suchen und Ersetzen ist eine Aufgabe, für die reguläre Ausdrücke häufig eingesetzt werden. Solch eine Funktion übernimmt einen Ausgangstext, einen regulären Ausdruck und einen Ersetzungsstring als Eingabe. Die Ausgabe ist der Ausgangstext, in dem alle Übereinstimmungen mit dem regulären Ausdruck durch den Ersetzungstext ausgetauscht wurden.

Obwohl der Ersetzungstext kein regulärer Ausdruck ist, können Sie bestimmte Syntaxelemente nutzen, um einen dynamischen Ersetzungstext aufzubauen. Bei allen Varianten können Sie den Text einfügen, der durch den regulären Ausdruck gefunden wurde, oder einen Teil davon. In den Rezepten 2.20 und 2.21 wird das beschrieben. Manche Varianten unterstützen auch noch das Einfügen von passendem Kontext im Ersetzungstext, wie in Rezept 2.22 zu lesen ist. In Kapitel 3, Rezept 3.16 erfahren Sie, wie man für jede Übereinstimmung einen anderen Ersetzungstext erzeugen kann.

Viele Varianten des Ersetzungstexts

Unterschiedliche Ideen von unterschiedlichen Softwareentwicklern haben dazu geführt, dass es viele verschiedene Varianten regulärer Ausdrücke gibt. Jede davon hat eine andere Syntax und unterstützt andere Features. Bei den Ersetzungstexten ist das nicht anders. Tatsächlich gibt es sogar noch mehr Varianten als bei den regulären Ausdrücken selbst. Es ist schwer, eine Engine für reguläre Ausdrücke aufzubauen. Die meisten Programmierer bevorzugen es, eine bestehende zu nutzen, aber es ist recht einfach, eine Funktion zum Suchen und Ersetzen auf einer bestehenden Regex-Engine aufzubauen. So gibt es viele Varianten für Ersetzungstexte in Regex-Bibliotheken, die nicht von sich aus bereits Funktionen zum Ersetzen anbieten.

Glücklicherweise haben alle Varianten regulärer Ausdrücke in diesem Buch entsprechende Varianten für den Ersetzungstext, mit Ausnahme der PCRE. Diese Lücke in PCRE macht den Programmierern, die damit arbeiten, das Leben schwer. Die Open Source-PCRE-Bibliothek enthält keinerlei Funktionen für Ersetzungen. Das heißt, alle Anwendungen und Programmiersprachen, die auf PCRE aufbauen, müssen ihre eigenen Funktionen bereitstellen. Die meisten Programmierer versuchen, eine bestehende Syntax zu kopieren, aber sie machen es nie exakt gleich.

Dieses Buch behandelt die folgenden Ersetzungstextvarianten. In „Viele Varianten regulärer Ausdrücke“ auf Seite 2, finden Sie weitere Informationen über die Regex-Varianten, die zu den entsprechenden Ersetzungstextvarianten gehören:

Perl

Perl besitzt eine eingebaute Unterstützung für Ersetzungen mit regulären Ausdrücken. Dazu nutzt es den Operator `s/regex/replace/`. Die Perl-Variante für Ersetzungen gehört zu der Perl-Variante für reguläre Ausdrücke. Dieses Buch behandelt Perl 5.6 bis Perl 5.10. In der letzten Version ist eine Unterstützung für benannte Rückwärtsreferenzen im Ersetzungstext hinzugekommen, so wie auch bei den regulären Ausdrücken nun benannte einfangende Gruppen enthalten sind.

PHP

In diesem Buch bezieht sich die PHP-Variante für den Ersetzungstext auf die Funktion `preg_replace`. Diese Funktion nutzt die PCRE-Variante für die regulären Ausdrücke und die PHP-Variante für den Ersetzungstext.

Andere Programmiersprachen, die PCRE nutzen, verwenden nicht die gleiche Ersetzungstextvariante wie PHP. Abhängig davon, woher die Designer Ihrer Programmiersprache ihre Inspiration beziehen, kann die Ersetzungstextsyntax der von PHP gleichen oder eine beliebige andere Variante aus diesem Buch nutzen.

PHP bietet zudem noch die Funktion `ereg_replace`. Diese Funktion nutzt eine andere Variante für reguläre Ausdrücke (POSIX ERE) und auch eine andere Variante für die Ersetzungstexte. PHPs `ereg`-Funktionen werden in diesem Buch nicht behandelt.

.NET

Das Paket `System.Text.RegularExpressions` stellt eine ganze Reihe von Funktionen zum Suchen und Ersetzen bereit. Die .NET-Variante für den Ersetzungstext gehört zur .NET-Variante für die regulären Ausdrücke. Alle Versionen von .NET verwenden die gleiche Ersetzungstextvariante. Die neuen Features in .NET 2.0 für die regulären Ausdrücke haben keinen Einfluss auf die Syntax für den Ersetzungstext.

Java

Das Paket `java.util.regex` enthält Funktionen zum Suchen und Ersetzen. Dieses Buch behandelt Java 4, 5 und 6. Alle Versionen greifen auf die gleiche Ersetzungstextsyntax zurück.

JavaScript

In diesem Buch nutzen wir den Begriff *JavaScript*, um sowohl auf die Variante für den Ersetzungstext zu verweisen als auch auf die für die regulären Ausdrücke. Beides ist in Edition 3 des ECMA-262-Standards definiert.

Python

Pythons Modul `re` stellt eine Funktion `sub` bereit, mit der gesucht und ersetzt werden kann. Die Python-Variante für den Ersetzungstext gehört zur Python-Variante für reguläre Ausdrücke. Dieses Buch behandelt Python 2.4 und 2.5. Die Regex-Unterstützung ist bei Python in den letzten Jahren sehr stabil gewesen.

Ruby

Die Unterstützung regulärer Ausdrücke in Ruby ist Bestandteil der Sprache selbst und auch die Funktion zum Suchen und Ersetzen. Dieses Buch behandelt Ruby 1.8 und 1.9. Eine Standardkompilierung von Ruby 1.8 nutzt die Variante für reguläre Ausdrücke, die direkt im Quellcode von Ruby definiert ist, während eine Standardkompilierung von Ruby 1.9 die Oniguruma-Bibliothek für reguläre Ausdrücke nutzt. Ruby 1.8 kann so kompiliert werden, dass Oniguruma verwendet wird, während Ruby 1.9 so kompiliert werden kann, dass die alte Regex-Variante von Ruby genutzt wird. In diesem Buch bezeichnen wir die native Ruby-Variante als Ruby 1.8 und die Oniguruma-Variante als Ruby 1.9.

Die Syntax für Ersetzungstexte ist in Ruby 1.8 und 1.9 die gleiche, nur dass Ruby 1.9 auch noch benannte Rückwärtsreferenzen im Ersetzungstext unterstützt. Benannte einfangende Gruppen sind ein neues Feature bei den regulären Ausdrücken von Ruby 1.9.

Tools für das Arbeiten mit regulären Ausdrücken

Sofern Sie nicht schon längere Zeit mit regulären Ausdrücken gearbeitet haben, empfehlen wir Ihnen, Ihre ersten Experimente in einem Tool durchzuführen und nicht in Quellcode. Die Beispiel-Regexes in diesem Kapitel und in Kapitel 2, sind einfache reguläre Ausdrücke, die keine zusätzliche Maskierung in einer Programmiersprache (oder sogar in einer Unix-Shell) brauchen. Sie können diese regulären Ausdrücke direkt in das Suchfeld einer Anwendung eingeben.

Kapitel 3 erklärt, wie Sie reguläre Ausdrücke in Ihren Quellcode einbauen können. Will man einen regulären Ausdruck als String mit Anführungszeichen versehen, wird er noch schwerer lesbar, weil sich dann die Maskierungsregeln für Strings mit denen für reguläre Ausdrücke vermischen. Wir heben uns das für Rezept 3.1, auf. Haben Sie einmal die Grundlagen regulärer Ausdrücke verstanden, werden Sie auch durch den Backslash-Dschungel finden.

Die in diesem Abschnitt beschriebenen Tools ermöglichen es Ihnen auch, reguläre Ausdrücke zu debuggen, die Syntax zu prüfen und andere Informationen zu bekommen, die in den meisten Programmierumgebungen nicht erhältlich sind. Wenn Sie also reguläre Ausdrücke in Ihren Anwendungen entwickeln, ist es für Sie vielleicht sinnvoll, einen komplizierten regulären Ausdruck in einem dieser Tools aufzubauen, bevor Sie ihn in Ihrem Programm einsetzen.

RegexBuddy

RegexBuddy (Abbildung 1-1) ist das Tool mit den (zum Zeitpunkt der Entstehung dieses Buchs) meisten verfügbaren Features zum Erstellen, Testen und Implementieren regulärer Ausdrücke. Es bietet die einzigartige Möglichkeit, alle Varianten regulärer Ausdrücke zu emulieren, die in diesem Buch behandelt werden, und sogar zwischen den verschiedenen Varianten zu konvertieren.

RegexBuddy wurde von Jan Goyvaerts entworfen und entwickelt, einem der Autoren dieses Buchs. Dadurch wurde Jan zu einem Experten für reguläre Ausdrücke und mithilfe von RegexBuddy war Koautor Steven in der Lage, sich so gut mit regulären Ausdrücken vertraut zu machen, dass er sich an diesem Buch beteiligen konnte.

Wenn der Screenshot (Abbildung 1-1) ein bisschen unruhig aussieht, liegt das daran, dass wir einen Großteil der Panels auf den Bildschirm gebracht haben, um zu zeigen, was RegexBuddy alles drauf hat. Die Standardansicht ordnet alle Panels hübsch in Registerkarten an. Sie können sie aber auch abreißen, um sie auf einen zweiten Monitor zu ziehen.

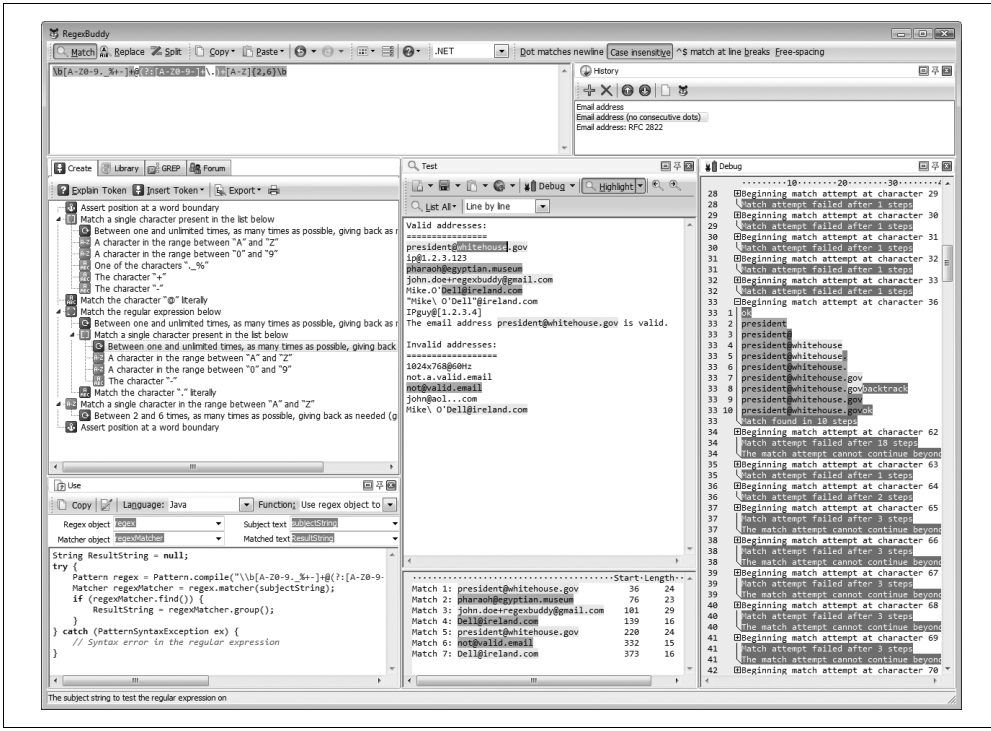


Abbildung 1-1: RegexBuddy

Um einen der regulären Ausdrücke aus diesem Buch auszuprobieren, tippen Sie ihn einfach in das Eingabefeld im oberen Bereich des RegexBuddy-Fensters ein. RegexBuddy stellt den Ausdruck dann automatisch mit Syntax-Highlighting dar und macht deutlich auf Fehler und fehlende Klammern aufmerksam.

Das *Create*-Panel baut automatisch eine detaillierte Analyse in englischer Sprache auf, während Sie die Regex eintippen. Durch einen Doppelklick auf eine Beschreibung im Baum mit der Struktur des regulären Ausdrucks können Sie diesen Teil des regulären Ausdrucks bearbeiten. Neue Teile lassen sich in Ihren regulären Ausdruck per Hand oder über den Button *Insert Token* einfügen. Bei Letzterem können Sie dann aus einem Menü auswählen, was Sie haben wollen. Wenn Sie sich zum Beispiel nicht an die komplizierte Syntax für einen positiven Lookahead erinnern, können Sie RegexBuddy bitten, die passenden Zeichen für Sie einzufügen.

Geben Sie einen Beispieltext im *Test*-Panel ein – indem Sie ihn entweder eintippen oder hineinkopieren. Wenn der Highlight-Button aktiv ist, markiert RegexBuddy automatisch den Text, der zur Regex passt.

Einige dieser Buttons werden Sie wahrscheinlich am häufigsten nutzen:

List All
: Gibt eine Liste mit allen Übereinstimmungen aus.

Replace
: Der *Replace*-Button am oberen Fensterrand zeigt ein neues Fenster an, in dem Sie den Ersetzungstext eingeben können. Der *Replace*-Button im *Test*-Panel zeigt Ihnen dann den Ausgangstext an, nachdem die Ersetzungen vorgenommen wurden.

Split (der Button im Test-Panel, nicht der am oberen Fensterrand)
: Behandelt den regulären Ausdruck als Separator und teilt den Ausgangstext in Tokens auf, die zeigen, wo in Ihrem Text Übereinstimmungen gefunden wurden.

Klicken Sie auf einen dieser Buttons und wählen Sie *Update Automatically*, damit RegexBuddy die Ergebnisse dynamisch aktualisiert, wenn Sie Ihre Regex oder den Ausgangstext anpassen.

Um genau zu sehen, wie Ihre Regex funktioniert (oder warum nicht), klicken Sie im *Test*-Panel auf eine hervorgehobene Übereinstimmung oder auf eine Stelle, an der die Regex nicht funktioniert hat, und danach auf den Button *Debug*. RegexBuddy wechselt dadurch zum *Debug*-Panel und zeigt den gesamten Prozess zum Finden von Übereinstimmungen Schritt für Schritt. Klicken Sie in die Ausgabe des Debuggers, um herauszufinden, welches Regex-Token zu dem Text passte, den Sie angeklickt haben.

Im *Use*-Panel wählen Sie Ihre bevorzugte Programmiersprache aus. Dann wählen Sie eine Funktion, um den Quellcode für das Implementieren Ihrer Regex zu erzeugen. Die Quellcode-Templates von Regex-Buddy lassen sich mit dem eingebauten Template-Editor problemlos bearbeiten. Sie können neue Funktionen und sogar neue Sprachen ergänzen oder bestehende anpassen.

Möchten Sie Ihre Regex mit einer größeren Datenmenge testen, wechseln Sie zum *GREP*-Panel, um eine beliebige Anzahl von Dateien und Ordnern zu durchsuchen (und eventuell Ersetzungen vorzunehmen).

Wenn Sie in Code, den Sie warten, eine Regex finden, kopieren Sie sie in die Zwischenablage – einschließlich der umschließenden Anführungszeichen oder Schrägstriche. In RegexBuddy klicken Sie auf den *Paste*-Button und wählen den String-Stil Ihrer Programmiersprache aus. Ihre Regex wird dann in RegexBuddy als pure Regex erscheinen – ohne die zusätzlichen Anführungszeichen oder Maskierungen, die für String-Literale notwendig sind. Mit dem *Copy*-Button erzeugen Sie einen String in der gewünschten Syntax, sodass Sie ihn in Ihren Quellcode einfügen können.

Sobald Sie mehr Erfahrung haben, können Sie im *Library*-Panel eine praktische Bibliothek mit regulären Ausdrücke aufbauen. Ergänzen Sie die dort abgelegten Regexes auf jeden Fall um eine detaillierte Beschreibung und einen Test-String. Reguläre Ausdrücke können sehr kryptisch sein, selbst für Experten.

Haben Sie dennoch ein Problem damit, eine passende Regex aufzubauen, klicken Sie auf das *Forum*-Panel und dann auf den *Login*-Button. Falls Sie RegexBuddy gekauft haben, erscheint das Anmeldefenster. Klicken Sie auf *OK*, sind Sie direkt mit dem Benutzerforum von RegexBuddy verbunden. Steven und Jan sind dort häufig zu finden.

RegexBuddy läuft unter Windows 98, ME, 2000, XP und Vista. Falls Sie eher Linux oder Apple bevorzugen, lässt sich RegexBuddy auch gut mit VMware, Parallels, CrossOver Office und (mit ein paar Einschränkungen) auch mit WINE betreiben. Sie können eine kostenlose Testversion von RegexBuddy unter *http://www.regexbuddy.com/RegexBuddy-Cookbook.exe* herunterladen. Abgesehen vom Benutzerforum ist die Testversion sieben Tage lang uneingeschränkt nutzbar.

RegexPal

RegexPal (Abbildung 1-2) ist ein Online-Testtool für reguläre Ausdrücke. Es wurde von Steven Levithan erstellt, einem der Autoren dieses Buchs. Sie brauchen dazu lediglich einen modernen Webbrowser. RegexPal ist vollständig in JavaScript geschrieben. Daher unterstützt es nur die JavaScript-Variante für reguläre Ausdrücke, so wie sie in dem von Ihnen verwendeten Webbrowser implementiert ist.

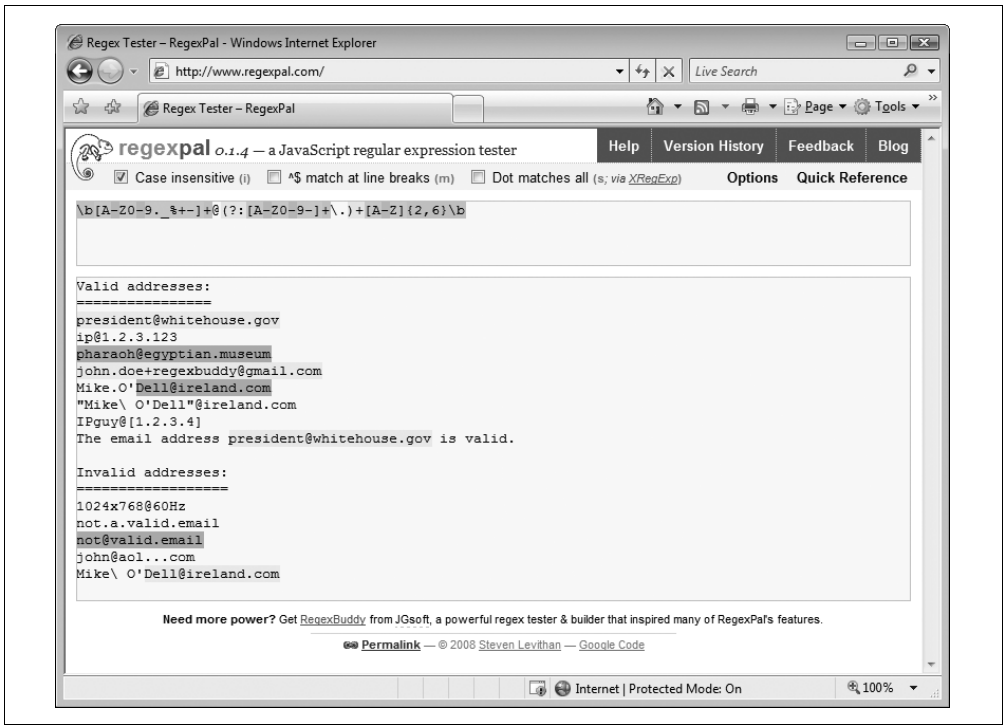


Abbildung 1-2: RegexPal

Um einen der regulären Ausdrücke auszuprobieren, die in diesem Buch vorgestellt werden, rufen Sie *http://www.regexpal.com* auf. Geben Sie die Regex in das Feld ein, in dem *Enter regex here.* steht. RegexPal zeigt Ihren regulären Ausdruck automatisch mit Syntax-Highlighting an, wodurch Sie auch Syntaxfehler in der Regex erkennen können. RegexPal ist sich der Probleme unterschiedlicher Implementierungen in den verschiedenen

Browsern bewusst, die Ihnen das Leben ganz schön schwer machen können. Wenn daher eine bestimmte Syntax in manchen Browsern nicht korrekt arbeitet, wird RegexPal diese als Fehler hervorheben.

Jetzt geben Sie einen Beispieltext in das Feld mit dem Inhalt *Enter test data here.* ein. RegexPal hebt automatisch die Übereinstimmungen zu Ihrer Regex hervor.

Es gibt keine Buttons, die man anklicken muss – das macht RegexPal zu einem der angenehmsten Onlinetools für das Testen regulärer Ausdrücke.

Weitere Onlinetools für Regexes

Es ist einfach, ein simples Onlinetool für das Testen regulärer Ausdrücke aufzubauen. Wenn Sie ein paar grundlegende Web-Entwicklungskenntnisse besitzen, reichen die Informationen aus Kapitel 3 aus, um selbst etwas zu bauen. Hunderte von Entwicklern haben das schon getan, ein paar haben Features ergänzt, die erwähnenswert sind.

regex.larsolavtorvik.com

Lars Olav Torvik hat ein tolles kleines Testtool für reguläre Ausdrücke unter *http://regex.larsolavtorvik.com* bereitgestellt (siehe Abbildung 1-3).

Zunächst wählen Sie die Variante aus, mit der Sie arbeiten wollen, indem Sie im oberen Bereich der Seite auf deren Namen klicken. Lars bietet PHP PCRE, PHP POSIX und JavaScript an. PHP PCRE, die PCRE-Variante, die wir in diesem Buch behandeln, wird von der PHP-Funktion `preg` genutzt. POSIX ist eine alte und recht eingeschränkte Regex-Variante, die von der PHP-Funktion `ereg` verwendet wird, aber in diesem Buch keine weitere Erwähnung findet. Wenn Sie JavaScript auswählen, arbeiten Sie mit der JavaScript-Implementierung Ihres Browsers.

Geben Sie Ihren regulären Ausdruck in das *Pattern*-Feld und Ihren Ausgangstext in das *Subject*-Feld ein. Einen Moment später wird im *Matches*-Feld Ihr Ausgangstext mit den hervorgehobenen Übereinstimmungen angezeigt. Das *Code*-Feld enthält eine einzelne Codezeile, die Ihre Regex auf Ihren Ausgangstext anwendet. Kopieren Sie diese Zeile in Ihren Codeeditor, brauchen Sie die Regexp nicht selbst in ein String-Literal umzuwandeln. Strings oder Arrays, die vom Code zurückgegeben werden, finden Sie im *Result*-Feld. Da Lars Ajax-Technologie verwendet hat, um seine Site aufzubauen, sind die Ergebnisse für alle Varianten immer nach kurzer Zeit verfügbar. Um dieses Tool nutzen zu können, müssen Sie online sein, da auf dem Server PHP verarbeitet wird.

Die zweite Spalte zeigt eine Liste mit Regex-Befehlen und -Optionen an. Diese hängen davon ab, welche Variante Sie gewählt haben. Zu den Befehlen gehören üblicherweise solche zum Finden von Übereinstimmungen (*match*), zum Ersetzen von Text (*replace*) und zum Aufteilen von Texten (*split*). Die Optionen enthalten die üblichen Verdächtigen, wie zum Beispiel das Ignorieren von Groß- und Kleinschreibung, aber auch implementierungsspezifische Optionen. Diese Befehle und Optionen werden in Kapitel 3, beschrieben.

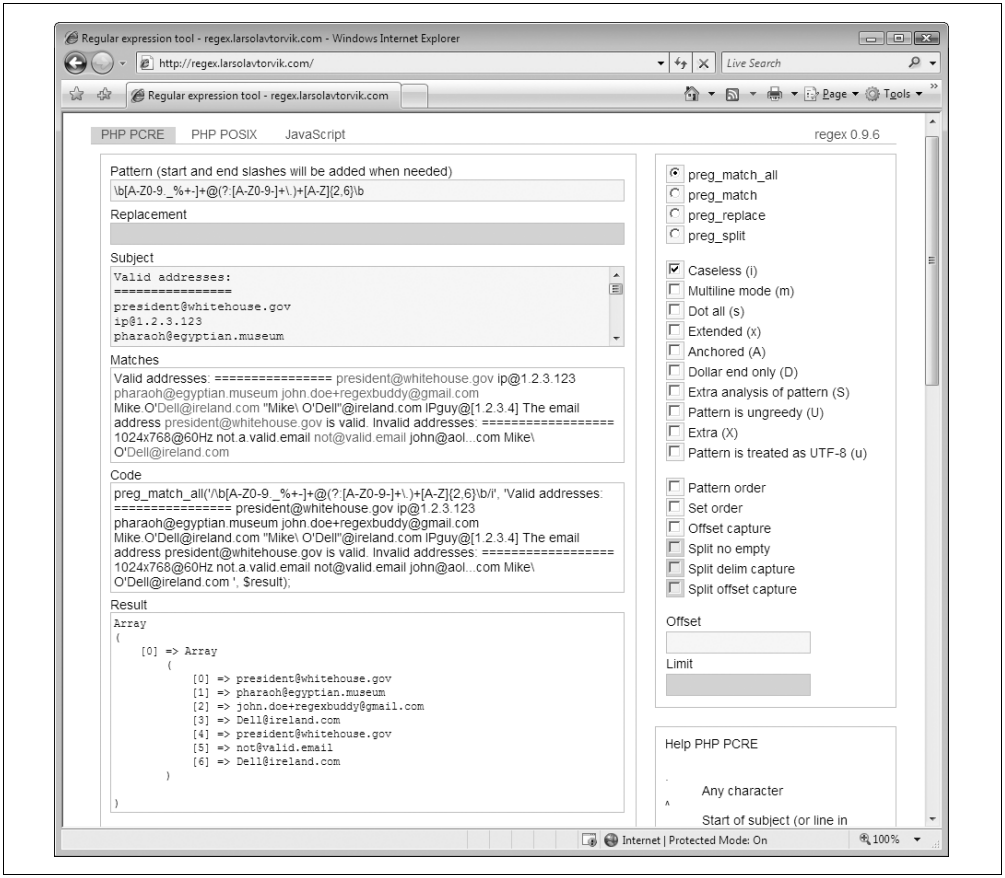


Abbildung 1-3: regex.larsolavtorvik.com

Nregex

http://www.nregex.com (Abbildung 1-4) ist ein unkompliziertes Online-Testtool für Regexes, das von David Seruyange mit .NET-Technologie gebaut wurde. Die Site erwähnt zwar nicht, welche Variante sie nutzt, aber als dieses Buch hier geschrieben wurde, war es .NET 1.x.

Das Layout der Seite ist ein bisschen verwirrend. Sie geben Ihren regulären Ausdruck in das Feld unter dem Text *Regular Expression* ein und setzen die Regex-Optionen mithilfe der Kontrollkästchen darunter. Geben Sie Ihren Ausgangstext in das große Feld am unteren Ende der Seite ein, wobei Sie den Standardtext `If I just had $5.00 then "she" wouldn't be so @#$! mad.` ersetzen. Wenn Ihr Text von einer Webseite kommt, geben Sie die URL in das Feld *Load Target From URL* ein und klicken auf den Button *Load*, der sich darunter befindet. Möchten Sie eine Datei auf Ihrer Festplatte als Ausgangsbasis nehmen, klicken Sie auf den Button *Browse*, wählen die gewünschte Datei aus und klicken dann auf den Button *Load* unter dem entsprechenden Eingabefeld.

Abbildung 1-4: Nregex

Ihr Ausgangstext wird im Feld *Matches & Replacements* in der Mitte der Webseite nochmals erscheinen, wobei die Übereinstimmungen zur Regex hervorgehoben sind. Wenn Sie etwas in das Feld *Replacement String* eingeben, wird stattdessen das Ergebnis des Ersetzungsvorgangs angezeigt. Wenn Ihr regulärer Ausdruck ungültig ist, erscheint: ...

Das Auswerten der Regex wird mit .NET-Code auf dem Server durchgeführt, daher müssen Sie für diese Site online sein. Arbeiten die automatischen Aktualisierungen langsam – vielleicht weil Ihr Ausgangstext sehr groß ist –, markieren Sie das Kontrollkästchen *Manually Evaluate Regex* über dem Eingabefeld für Ihren regulären Ausdruck, um einen *Evaluate*-Button zu erhalten. Diesen können Sie dann anklicken, um das Feld *Matches & Replacements* zu aktualisieren.