

3. AUFLAGE

Reguläre Ausdrücke

Jeffrey E. F. Friedl

*Deutsche Übersetzung von
Andreas Karrer*

O'REILLY®

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

Vorwort	XV
1 Einführung in reguläre Ausdrücke	1
Probleme aus der Praxis lösen	2
Reguläre Ausdrücke als Programmiersprache	4
Die Analogie zu Dateinamen	4
Die Analogie zu natürlichen Sprachen	5
Reguläre Ausdrücke als Denkweise	6
Wenn Sie Erfahrungen mit regulären Ausdrücken haben	6
Textdateien durchsuchen: egrep	6
Metazeichen bei egrep	8
Zeilenanfang und Zeilenende	8
Zeichenklassen	9
Auf irgendein Zeichen prüfen: der Punkt	11
Alternation	13
Groß- und Kleinschreibung ignorieren	15
Wortgrenzen	15
Kurze Rekapitulation	16
Optionale Elemente	17
Andere Quantoren: Repetition	18
Klammern und Rückwärtsreferenzen	20
Ausbrecher!	23
Erweiterung der Fundamente	23
Linguistisches Divertissement	23
Das Ziel eines regulären Ausdrucks	24
Weitere Beispiele	24
Reguläre Ausdrücke: Terminologie	27
Den Status quo verbessern	30
Zusammenfassung	32
Persönliche Einsprengsel	34

2 Erweiterte einführende Beispiele	35
Zu den Beispielen	36
Eine kleine Einführung in Perl	37
Mustererkennung mit regulären Ausdrücken	38
Mehr Praxisnähe	40
Nebeneffekte bei erfolgreicher Mustererkennung	40
Verschachtelte reguläre Ausdrücke	43
Rekapitulation	49
Mit regulären Ausdrücken Text verändern	50
Programmbeispiel: Serienbrief	51
Programmbeispiel: Aktienkurse	52
Automatisiertes Editieren von Dateien	53
Ein kleines Mail-Programm	54
Große Zahlen in Dreiergruppen aufteilen: Lookaround	60
Nackten Text in HTML verwandeln	69
Ein Wiedersehen mit verdoppelten Wörtern	79
3 Features und Dialekte	85
Ein Spaziergang durch die Welt der regulären Ausdrücke	87
Die Ursprünge regulärer Ausdrücke	87
Kurzer Überblick	93
Wartung und Pflege von regulären Ausdrücken	96
Integrierter Ansatz	96
Prozeduraler und objektorientierter Ansatz	97
Beispiele mit »Suchen und Ersetzen«	100
Suchen und Ersetzen in anderen Sprachen	103
Wartung und Pflege: Zusammenfassung	104
Strings, Zeichencodierungen und Modi	104
Strings als reguläre Ausdrücke	105
Zeichencodierungen	108
Unicode	110
Regex-Modi	113
Übliche Metazeichen	116
Zeichendarstellung	118
Zeichenklassen und ähnliche Konstrukte	121
Anker und andere »Zusicherungen der Länge null«	132
Kommentare und Modus-Modifikatoren	138
Gruppierende und einfangende Klammern, logische und gierige Konstrukte	140
Führer durch die Kapitel für Fortgeschrittene	146
4 Wie Regex-Maschinen arbeiten	147
Motor anlassen!	147
Zwei Arten von Motoren	147

Kalifornische Abgasvorschriften	148
Typen von Regex-Maschinen	149
Aus der Abteilung für Redundanz-Abteilung	150
Den Typ der Regex-Maschine bestimmen	150
Grundlegendes zum Vorgang der Mustersuche	151
Zu den Beispielen	151
Regel 1: Der am weitesten links beginnende Treffer gewinnt	152
Bestandteile der Regex-Maschine	153
Regel 2: Die normalen Quantoren sind gierig	155
Regex-gesteuerte und textgesteuerte Maschinen	157
NFA-Maschine: Regex-gesteuert	158
DFA-Maschine: Textgesteuert	159
Ein erster Vergleich von NFA- und DFA-Maschinen	160
Backtracking	162
Backtracking wie Hänsel und Gretel	162
Zwei wichtige Regeln beim Backtracking	163
Gespeicherte Zustände	163
Backtracking und Gier	166
Mehr Gieriges	168
Probleme durch gieriges Verhalten	169
Mehrbuchstellige »Anführungszeichen«	170
Nicht-gierige Quantoren	170
Gierig oder genügsam – der Treffer geht vor	172
Gier, Genügsamkeit, Backtracking: Die Essenz	173
Possessive Quantoren und atomare Gruppen	174
Backtracking bei Lookaround	177
Ist die Alternation gierig?	179
Ausnutzen der geordneten Alternation	180
NFA, DFA und POSIX	182
Der »längste früheste Treffer«	182
POSIX und der »längste früheste Treffer«	184
Geschwindigkeit und Effizienz	184
DFA und NFA im Vergleich	186
Zusammenfassung	189
5 Regex-Methoden aus der Praxis	191
Die ausgewogene Regex	192
Einige kleine Beispiele	192
Fortsetzungszeilen	192
Eine IP-Adresse erkennen	193
Umgang mit Dateinamen	195
Verschachtelte Klammerpaare	199
Ungewollte Treffer vermeiden	200

Eingefassten Text erkennen	202
Erwartete Daten und Annahmen	204
Leerzeichen am Anfang und am Ende entfernen	205
Beispiele zu HTML	206
HTML-Tags erkennen	206
Einen HTML-Link erkennen	207
Eine HTTP-URL auseinandernehmen	209
Einen Hostnamen auf syntaktische Korrektheit prüfen	211
Eine URL in der Praxis erkennen	212
Ausführliche Beispiele	216
Mit den Daten im Takt bleiben	216
CSV-Dateien verarbeiten	219
6 Die Kunst, reguläre Ausdrücke zu schreiben	227
Ein ernüchterndes Beispiel	228
Eine einfache Änderung – die Schokoladenseite zuerst	229
Effizient oder korrekt?	229
Gieriges Verhalten nur lokal zulassen	231
Zurück zur Realität	232
Backtracking global betrachtet	234
Überstunden für den POSIX-NFA	235
Mehr Arbeit bei einem Fehlschlag	236
Einschränkendere Formulierung	237
Alternationen können teuer sein	238
Benchmarking	238
Was genau wird da gemessen?	240
Benchmarks mit PHP	241
Benchmarks mit Java	242
Benchmarks mit VB.NET	244
Benchmarks mit Ruby	245
Benchmarks mit Python	245
Benchmarks mit Tcl	246
Übliche Optimierungen	247
Umsonst gibt's nichts!	247
Jeder macht's ein bisschen anders	248
Wie ein regulärer Ausdruck angewendet wird	248
Optimierungen vor der eigentlichen Suche	249
Optimierungen mit dem Getriebe	253
Optimierungen der Regex-Maschine	255
Programmierungsmethoden für schnellere Ausdrücke	259
Gesunden Menschenverstand walten lassen	261
Literalen Text herausstellen	263
Anker herausstellen	263

Gierig oder genügsam?	264
Aufteilen in mehrere reguläre Ausdrücke	264
Die »Erstes Zeichen«-Optimierung imitieren	266
Atomare Gruppen und possessive Quantoren verwenden	267
Die Regex-Maschine zum Treffer hinführen	267
Die Schleife aufbrechen	269
Methode 1: Eine Regex anhand früherer Erfahrungen aufbauen	270
Ein Rezept zum Aufbrechen von Schleifen	271
Methode 2: Die kritische Schleife im größeren Zusammenhang betrachten	274
Methode 3: Ein Internet-Hostname in Anführungszeichen	275
Beobachtungen	276
Atomare Gruppen und possessive Quantoren verwenden	276
Beispiele zum Aufbrechen von Schleifen	278
C-Kommentare aufbrechen	280
Die frei fließende Regex	285
Eine helfende Hand führt die Maschine	285
Eine gut geführte Regex ist eine schnelle Regex	287
Zusammenfassung	289
Denken!	289
7 Perl	291
Reguläre Ausdrücke als Teil der Programmiersprache	293
Perls größte Stärken	294
Perls größte Schwächen	294
Perls Regex-Dialekt	294
Regex-Operanden und Regex-Literale	296
Wie Regex-Literale geparkt werden	300
Regex-Modifikatoren	301
Perlges über reguläre Ausdrücke	302
Kontext bei Ausdrücken	302
Dynamische Geltungsbereiche: Auswirkungen auf die Mustererkennung	303
Durch das Matching gesteuerte Spezialvariablen	308
Der qr/.../-Operator und Regex-Objekte	312
Regex-Objekte aufbauen und verwenden	312
Regex-Objekte anschauen	314
Regex-Objekte zur Effizienzsteigerung	315
Der Match-Operator	315
Der Regex-Operand	316
Der Suchtext-Operand	317
Verschiedene Einsatzmöglichkeiten des Match-Operators	319
Iterative Mustersuche – Skalarer Kontext mit dem /g-Modifikator	321
Beziehungen des Match-Operators zum Umfeld	326

Der Substitutionsoperator	328
Der Ersatztext-Operand	329
Der /e-Modifikator	329
Kontext und Rückgabewert	330
Der Split-Operator	331
Grundlegendes zu split	332
split kann leere Elemente zurückgeben	333
Spezielle Formen des Regex-Operanden bei split	335
Der Regex-Operand mit einfangenden Klammern	335
Verrückte Dinge mit den Regex-Erweiterungen in Perl	336
Verschachtelte Paare mit dynamischen regulären Ausdrücken erkennen	338
Perl-Code in der Regex	341
local in einem Codemuster	345
Vorsicht bei my-Variablen in Codemustern	348
Mit Codemustern verschachtelte Konstrukte erkennen	350
Überladen von Regex-Literalen	352
Probleme beim Überladen von Regex-Literalen	354
Benannte Unterausdrücke imitieren	355
Effizienz in Perl	357
»There's More Than One Way To Do It«	358
Regex-Kompilierung, der /o-Modifikator, qr/.../ und Effizienz	360
Perl kopiert den Suchstring vor der Mustersuche	365
Die Funktion study	369
Benchmarks	370
Debugging-Informationen zu regulären Ausdrücken	371
Abschließende Betrachtungen	373
8 Java	375
Der Regex-Dialekt von Java	376
Unterstützung von \p{...} und \P{...} in Java	379
Unicode-Eigenschaften	379
Verwendung von java.util.regex	381
Die Factory-Methode Pattern.compile	382
Die matcher-Methode	383
Das Matcher-Objekt	383
Die Regex anwenden	385
Resultate abfragen	386
Einfaches Suchen und Ersetzen	388
Suchen und Ersetzen im Detail	391
Direktes Suchen und Ersetzen	393
Die Region des Matchers	394
Verkettung von Methoden	399
Methoden zum Bau von Scannern	400
Weitere Matcher-Methoden	403

Weitere Pattern-Methoden	405
Die split-Methode mit einem Argument	406
Die split-Methode mit zwei Argumenten	407
Weitere Programmbeispiele	408
Hinzufügen von WIDTH- und HEIGHT-Attributen zu IMG-Tags in HTML . .	408
Ein Matcher zum Prüfen von HTML mit mehreren Pattern	409
Daten im CSV-Format verarbeiten	411
Unterschiede zwischen den Java-Versionen	411
Unterschiede zwischen Java 1.4.2 und 1.5.0	413
Unterschiede zwischen Java 1.5.0 und 1.6	414
9 .NET	415
Der Regex-Dialekt in .NET	416
Weitere Anmerkungen zum .NET-Dialekt	419
Gebrauch von regulären Ausdrücken in .NET	423
Ganz kurz: Reguläre Ausdrücke in .NET	423
Überblick	425
Das Objekt-Modell in .NET – Überblick	426
Die Objekte im Detail	428
Regex-Objekte erzeugen	429
Regex-Objekte verwenden	431
Match-Objekte verwenden	438
Group-Objekte verwenden	440
Statische »Komfort-Funktionen«	441
Caching von regulären Ausdrücken	442
Hilfsfunktionen	443
Fortgeschrittenes mit regulären Ausdrücken in .NET	444
Regex-Assemblies	444
Verschachtelte Konstrukte erkennen	446
Capture-Objekte	447
10 PHP	451
Der Regex-Dialekt von PHP	453
Die »preg«-Programmierschnittstelle	455
Der »Pattern«-Parameter	456
Die Preg-Funktionen	461
preg_match	461
preg_match_all	466
preg_replace	471
preg_replace_callback	476
preg_split	479
preg_grep	484
preg_quote	485

»Fehlende« preg-Funktionen	486
preg_regex_to_pattern	486
Ein Pattern-Argument auf korrekte Syntax prüfen	488
Eine Regex auf korrekte Syntax prüfen	489
Rekursive reguläre Ausdrücke	490
Text mit verschachtelten Klammern parsen	490
Kein Backtracking mitten in die Rekursion	492
Ein verschachteltes Klammerpaar finden	492
Überlegungen zur Effizienz	492
Der S-Modifikator	493
Ausführliche Beispiele	495
CSV-Dateien mit PHP verarbeiten	495
Tags auf korrekte Verschachtelung prüfen	496
Index	501