

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
	Verdoppelte Wörter, nochmals	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	95
	Integrierter Ansatz	96
	Prozeduraler und objektorientierter Ansatz	97
	Beispiele mit »Suchen-und-Ersetzen«	100
	Suchen und Ersetzen in anderen Sprachen	102
	Wartung und Pflege: Zusammenfassung	103
	Strings, Zeichencodierungen und Modi	103
	Strings als reguläre Ausdrücke	104
	Zeichencodierungen	108
	Regex-Modi	112
	Übliche Metazeichen	115
	Zeichendarstellung	117
	Zeichenklassen und ähnliche Konstrukte	120
	Anker und andere »Zusicherungen der Länge Null«	131
	Kommentare und Modus-Modifikatoren	136
	Gruppierende und einfangende Klammern, logische und gierige Konstrukte	138
	Führer durch die Kapitel für Fortgeschrittene	144
4	Wie Regex-Maschinen arbeiten	145
	Motor anlassen!	145
	Zwei Arten von Motoren	145
	Kalifornische Abgasvorschriften	146

Typen von Regex-Maschinen	147
Aus der Abteilung für Redundanz-Abteilung	148
Den Typ der Regex-Maschine bestimmen	148
Grundlegendes zum Vorgang der Mustersuche	149
Zu den Beispielen	149
Regel 1: Der am weitesten links beginnende Treffer gewinnt	150
Bestandteile der Regex-Maschine	151
Regel 2: Die normalen Quantoren sind gierig	153
Regex-gesteuerte und textgesteuerte Maschinen	155
NFA-Maschine: Regex-gesteuert	156
DFA-Maschine: Textgesteuert	157
Ein erster Vergleich von NFA- und DFA-Maschinen	158
Backtracking	160
Backtracking wie Hänsel und Gretel	160
Zwei wichtige Regeln beim Backtracking	161
Gespeicherte Zustände	161
Backtracking und Gier	164
Mehr Gieriges	166
Probleme durch gieriges Verhalten	167
Mehrbuchstabile »Anführungszeichen«	168
Nicht-gierige Quantoren	168
Gierig oder genügsam – der Treffer geht vor	170
Gier, Genügsamkeit, Backtracking: Die Essenz	171
Possessive Quantoren und atomare Gruppen	172
Backtracking bei Lookaround	175
Ist die Alternation gierig?	177
Ausnutzen der geordneten Alternation	178
NFA, DFA und POSIX	180
Der »längste früheste Treffer«	180
POSIX und der »längste früheste Treffer«	182
Geschwindigkeit und Effizienz	182
DFA und NFA im Vergleich	184
Zusammenfassung	187
5 Regex-Methoden aus der Praxis	189
Die ausgewogene Regex	190
Einige kleine Beispiele	190
Fortsetzungszeilen	190
Eine IP-Adresse erkennen	191
Umgang mit Dateinamen	194
Verschachtelte Klammerpaare	197
Ungewollte Treffer vermeiden	198

201	Eingefassten Text erkennen	200
202	... Erwartete Daten und Annahmen	202
203	... Leerzeichen am Anfang und am Ende entfernen	203
204	... Beispiele zu HTML	204
204	HTML-Tags erkennen	204
205	Einen HTML-Link erkennen	205
207	Eine HTTP-URL auseinandernehmen	207
209	Einen Hostnamen auf syntaktische Korrektheit prüfen	209
210	Eine URL in der Praxis erkennen	210
214	... Ausführliche Beispiele	214
214	Mit den Daten im Takt bleiben	214
217	CSV-Dateien verarbeiten	217
225	6 Die Kunst, reguläre Ausdrücke zu schreiben	225
226	... Ein ernüchterndes Beispiel	226
227	... Eine einfache Änderung – die Schokoladenseite zuerst	227
227	... Effizient oder korrekt?	227
229	... Gieriges Verhalten nur lokal zulassen	229
230	... Zurück zur Realität	230
232	... Backtracking global betrachtet	232
233	Überstunden für den POSIX-NFA	233
234	Mehr Arbeit bei einem Fehlschlag	234
235	Einschränkendere Formulierung	235
236	Alternationen können teuer sein	236
236	... Benchmarking	236
238	Was genau wird da gemessen?	238
239	Benchmarks mit Java	239
241	Benchmarks mit VB.NET	241
242	Benchmarks mit Python	242
243	Benchmarks mit Ruby	243
244	Benchmarks mit Tcl	244
245	... Übliche Optimierungen	245
245	Umsonst gibt's nichts!	245
246	Jeder macht's ein bisschen anders	246
246	Wie ein regulärer Ausdruck angewandt wird	246
247	Optimierungen vor der eigentlichen Suche	247
251	Optimierungen mit dem Getriebe	251
253	Optimierungen der Regex-Maschine	253
257	... Programmiermethoden für schnellere Ausdrücke	257
259	Gestunden Menschenverstand anwenden	259
261	Literalen Text herausstellen	261
261	Anker herausstellen	261

Gierig oder genügsam?	262
Aufteilen in mehrere reguläre Ausdrücke	262
Die »Erstes-Zeichen«-Optimierung imitieren	264
Atomare Gruppen und possessive Quantoren verwenden	265
Die Regex-Maschine zum Treffer hinführen	265
Die Schleife aufbrechen	267
Methode 1: Eine Regex anhand früherer Erfahrungen aufbauen	268
Ein Rezept zum Aufbrechen von Schleifen	269
Methode 2: Die kritische Schleife im größeren Zusammenhang betrachten ..	272
Methode 3: Ein Internet-Hostname in Anführungszeichen	273
Beobachtungen	274
Atomare Gruppen und possessive Quantoren verwenden	274
Beispiele zum Aufbrechen von Schleifen	276
C-Kommentare aufbrechen	278
Die frei fließende Regex	283
Eine helfende Hand führt die Maschine	283
Eine gut geführte Regex ist eine schnelle Regex	285
Zusammenfassung	287
Denken!	287
7 Perl	289
Reguläre Ausdrücke als Teil der Programmiersprache	291
Perls größte Stärken	292
Perls größte Schwächen	292
Perls Regex-Dialekt	292
Regex-Operanden und Regex-Literale	294
Wie Regex-Literale geparkt werden	298
Regex-Modifikatoren	299
Perlges über reguläre Ausdrücke	300
Kontext bei Ausdrücken	300
Dynamische Geltungsbereiche: Auswirkungen auf die Mustererkennung ...	301
Durch das Matching gesteuerte Spezialvariablen	306
Der qr/.../-Operator und Regex-Objekte	310
Regex-Objekte aufbauen und verwenden	310
Regex-Objekte anschauen	312
Regex-Objekte zur Effizienzsteigerung	313
Der Match-Operator	313
Der Regex-Operand	314
Der Suchtext-Operand	315
Verschiedene Einsatzmöglichkeiten des Match-Operators	317
Iterative Mustersuche – Skalarer Kontext mit dem /g-Modifikator	320
Beziehungen des Match-Operators zum Umfeld	324

Der Substitutionsoperator	326
Der Ersatztext-Operand	327
Der /e-Modifikator	327
Kontext und Rückgabewert	329
Der Split-Operator	329
Grundlegendes zu split	330
Split kann leere Elemente zurückgeben	332
Spezielle Formen des Regex-Operanden bei split	333
Der Regex-Operand mit einfangenden Klammern	334
Verrückte Dinge mit den Regex-Erweiterungen in Perl	335
Verschachtelte Paare mit dynamischen regulären Ausdrücken erkennen	337
Perl-Code in der Regex	340
local in einem Codemuster	345
Vorsicht bei my-Variablen in Codemustern	347
Mit Codemustern verschachtelte Konstrukte erkennen	349
Überladen von Regex-Literalen	351
Probleme mit dem Überladen von Regex-Literalen	354
Benannte Unterausdrücke imitieren	354
Effizienz in Perl	357
»There's More Than One Way To Do It«	358
Regex-Kompilierung, der /o-Modifikator, qr/.../ und Effizienz	358
Perl kopiert den Suchstring vor der Mustersuche	364
Die Funktion study	369
Benchmarks	370
Debug-Informationen zu regulären Ausdrücken	371
Abschließende Betrachtungen	373

8 Java 375

Ein Regex-Package einschätzen	376
Technische Merkmale	376
Soziale und politische Merkmale	377
Objekt-Modelle	378
Einige abstrakte Objekt-Modelle	378
Zunehmende Komplexität	382
Packages, Packages, Packages	382
Warum sind fast alles »Perl5«-Dialekte?	385
Traue keinem Benchmark, den du nicht selber gefälscht hast	385
Empfehlungen	388

Das Regex-Package von Sun	388
Der Regex-Dialekt	388
Verwendung von <code>java.util.regex</code>	393
Die <code>Pattern.compile()</code> -Factory	394
Das <code>Matcher</code> -Objekt	395
Weitere <code>Pattern</code> -Methoden	400
Ein kurzer Blick auf Jakarta-ORO	403
<code>Perl5Util</code> von ORO	403
Eine kleine Zusammenstellung der <code>Perl5Util</code> -Methoden	404
Die einzelnen ORO-Klassen	408
9 .NET	411
Der Regex-Dialekt in .NET	412
Weitere Anmerkungen zum .NET-Dialekt	414
Gebrauch von regulären Ausdrücken in .NET	419
Ganz kurz: Reguläre Ausdrücke in .NET	419
Überblick	421
Das Objekt-Modell in .NET – Überblick	422
Die Objekte im Detail	424
Regex-Objekte erzeugen	425
Regex-Objekte verwenden	427
Match-Objekte verwenden	436
Group-Objekte verwenden	437
Statische »Komfort-Funktionen«	438
Caching von regulären Ausdrücken	439
Hilfsfunktionen	440
Fortgeschrittenes mit regulären Ausdrücken in .NET	441
Regex-Assemblies	441
Verschachtelte Konstrukte erkennen	443
Capture-Objekte	445
Index	447