17. Access-Entwickler-Konferenz
Nürnberg 27./28.09.2014 und Hannover 11./12.10.2014

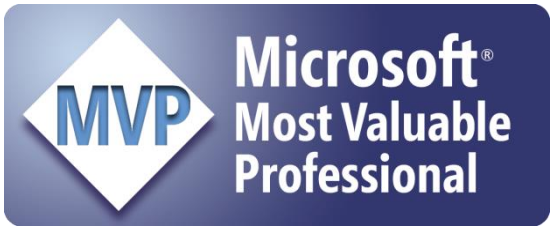
REGULÄRE AUSDRÜCKE

In Theorie und Praxis

Thomas Möller, www.Team-Moeller.de

Vorstellung

- Thomas Möller
- dipl. Sparkassenbetriebswirt
- Arbeite mit Access seit 1997
- Seit 2000 hauptberuflich als Entwickler tätig
- Schwerpunkte
  - Access mit DB/2 als BackEnd
  - Web-Entwicklung mit ASP.Net
  - Aktuell ApEx mit Oracle
- Nebenberuflich: Team-Moeller.de
  - Add-Ins
- Seit 1.1.2007 MVP für Access

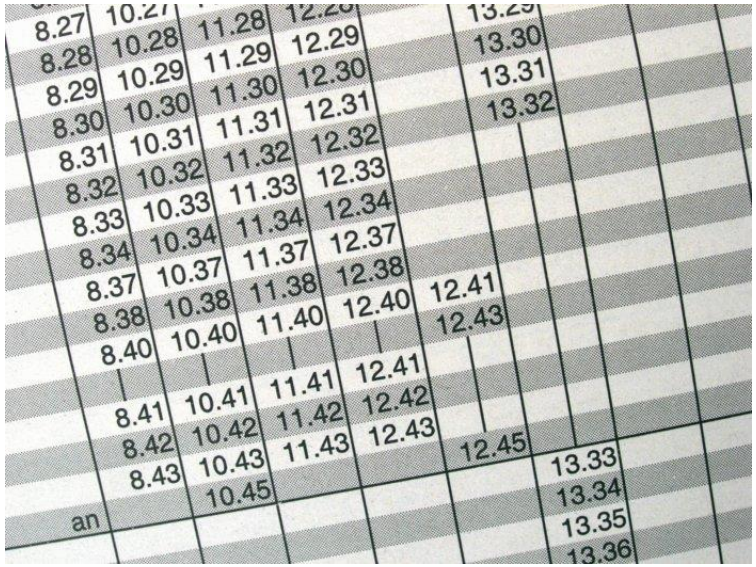


Kontaktdaten

- Webseite www.Team-Moeller.de
- Blog Blog.Team-Moeller.de
- E-Mail Thomas@Team-Moeller.de
- Twitter https://twitter.com/ThomasMoeller
- XING https://www.xing.com/profile/Thomas_Moeller40
- Facebook https://www.facebook.com/TeamMoeller
- Google Plus https://plus.google.com/101082101965686277484
- MVP http://mvp.microsoft.com/de-de/mvp/Thomas%20Moeller-37548

Fahrplan

- Grundlagen, Umsetzung in VBA
- Einführung in RegEx
- Komplexe Beispiele
- Beispiel: Access
- Beispiel: Java Script
- Lessons Learned
- Ressourcen



Umfrage

- Wer hat schon mal von Regular Expression gehört?
- Wer hat schon einmal versucht,
  eine Regular Expression zu erstellen?
- Wer hat schon mal eine Regular Expression
  kopiert und angepasst?
- Wer programmiert regelmäßig mit Regular Expressions?

GRUNDLAGEN, UMSETZUNG VBA

Grundlagen

- Zeichenkette, die der Beschreibung von Mengen von Zeichenketten mit Hilfe bestimmter syntaktischer Regeln dient.

Quelle: Wikipedia

- Programmiersprache für Texte
- Häufig in „Suchen und Ersetzen“
- Einfachster Anwendungsfall: Wildcards

Grundlagen (II)

- In den 60er Jahren erstmals in UNIX implementiert
- In vielen Programmiersprachen vorhanden
- Einheitlicher Standard,
  mit sprachspezifischen Abweichungen
- Einmal „richtig“ lernen, überall verwenden

Learn once, use everywhere

Wildcards

- Datei-Dialog
  - `*.txt`
- Like in Abfragen
  - `Like "*Access*"`
- * für beliebig viele Zeichen
- ? für **ein** beliebiges Zeichen
- Like kann noch mehr!

Feld:	ID	Veranstaltung: Demo
Tabelle:	tbl_Veranstaltung	tbl_Veranstaltung
Sortierung:		
Anzeigen:	☑	☑
Kriterien:		Wie "*AEK*"
oder:		

Unterschiede bei „Like“

Access (*.mdb) DAO und VBA	Access Projekt (*.adp) ADO und SQL Server	Erklärung
*	%	Ein oder mehrere beliebige Zeichen
?	_	Ein beliebiges Zeichen
[Liste]	[Liste]	Jedes Zeichen aus der Liste
[!Liste]	[^Liste]	Schließt alle Zeichen aus der Liste aus
#	N/A	Ziffern 0 - 9

Like in VBA

- Prüfen, ob Wort in Text enthalten
  - `If (InStr(strText, „Wort")) Then`
  - `If (strText Like "*Wort*“) Then`
- Prüfen ob Text mit Wort beginnt
  - `If (Left$(strText, 4) = “Wort") Then`
  - `If (strText Like “Wort*") Then`
- Prüfen, ob Text mit Wort endet
  - `If (Right$(strText, 4) = “Wort") Then`
  - `If (strText Like “*Wort") Then`

D
E
M
O

1

Groß- und Kleinschreibung

- Machen InStr und Like keinen Unterschied?
- Unterschied abhängig von Option Compare

Einstellung	Auswirkung
Compare Database	Nicht Case-sensitiv
Compare Text	Nicht Case-sensitiv
Compare Binary	Case-sensitiv

- Funktion ggf. in gesondertes Modul auslagern

DEMO 2

Verwendung von Wertelisten

- Werden in eckigen Klammern angegeben
  - `Like "*[a-z]*"`
  - `Like "Jede[nmrs]"`
- Negation mittels Ausrufezeichen („!“)
  - `Like "*[!a-z]"`
  - `Like "*[!0-9]*"`
- Beispiel Passwortprüfung
  - 8 Zeichen, mindesten je ein Klein- u. Großbuchstabe, eine Ziffer und ein Sonderzeichen

DEMO 3

VBScript-Bibliothek

- Implementiert gemäß JavaScript für IE 5.5

- Early Binding
  - Verweis auf "MS VBScript Regular Expressions 5.5"
- Late Binding
  - Instanz mit CreateObject("VBScript.RegExp")

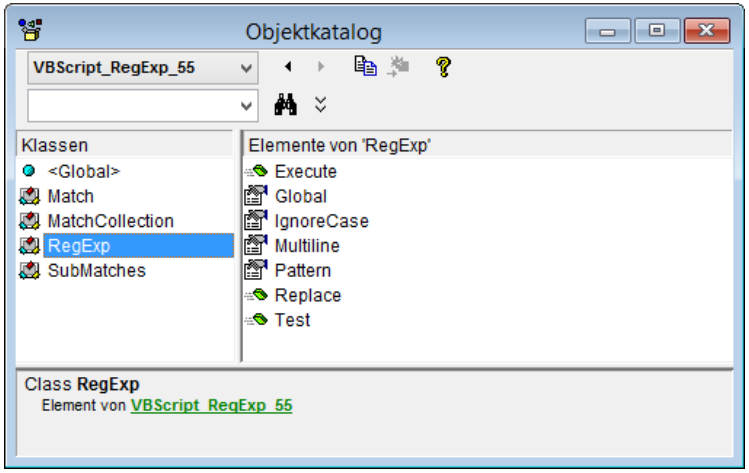
- Detaillierte Beschreibung hier:
  - Microsoft Beefs Up VBScript with Regular Expressions

Eigenschaften

- Global (= True)
  - Nach dem ersten Treffer wird weiter gesucht
  - Alle Treffer werden gefunden
- Multiline (= True)
  - Mehrzeilenmodus aktiv
  - Zeilenwechsel wird als Textbeginn bzw. -ende interpretiert
- IgnoreCase (= True)
  - Kein Unterschied zwischen Groß- und Kleinschreibung
- Pattern
  - Suchmuster

Methoden

- Test
  - Zeigt an, ob Muster gefunden wurde



- Execute
  - Liefert Match bzw. MatchCollection

- Replace
  - Führt Suchen und Ersetzen durch

Beispiele

```
Dim regEx As RegExp
Set regEx = New RegExp

regEx.Global = True
regEx.IgnoreCase = True
regEx.Multiline = True

regEx.Pattern = "Liebe"

MsgBox regEx.Test("Ich liebe Access")
```

D
E
M
O

4

EINFÜHRUNG IN REGEX

RegexBuddy - Steckbrief

- Name des Tools
  - RegexBuddy
- Autor
  - Jan Goyvaerts
- Download
  - http://www.regexbuddy.com/
- Lizenz / Preis
  - Kommerziell / 29,95 € 7-Tage Testversion verfügbar
- Zweck / Funktion
  - Editor zum Erstellen und Testen regulärer Ausdrücke

RegexBuddy - Funktionen

- Regular Expressions
  - erstellen und testen
- Erläuterung bestehender Ausdrücke
- Diverse Dialekte verfügbar
  - Übersetzung zwischen den Dialekten
- Debug ermöglicht Analyse/Diagnose
- Bibliothek mit fertigen Ausdrücken
- Sehr ausführliche Hilfe / Tutorium
- Zugriff auf Online-Forum

Literale und Metazeichen

- Zeichen stellen sich selbst dar
  - RegEx: H
  - Match: **H**allo

- Metazeichen haben besondere Bedeutung
- Müssen mit \ maskiert werden
- Metazeichen sind .+*?(){}[]\^$
  - RegEx: \?
  - Match: Wer**?**

Literale und Metazeichen (II)

- Backslash selbst ist auch ein Metazeichen
  - RegEx: \\
  - Match: C:\Windows

- Punkt („.“) steht für jedes beliebige Zeichen
  - RegEx: W.
  - Match: **We**r? Was?
  - Weitere: Wer? **Wa**s?

  - RegEx: \.
  - Match: Ich liebe Access.

Zeichenklassen

- Eingefasst in eckige Klammern []
- Stehen für **ein** Zeichen aus der Gruppe
- Z.B. [a-z] oder [0-9]

  - RegEx: [aeiou]
  - Match: Ich liebe Access.
  - Weitere: Ich liebe Access.

  - RegEx: [0-9]
  - Match: 17. AEK
  - Weitere: 17. AEK

Zeichenklassen (II)

- Negierte Zeichenklassen
- Beginnen mit ^
- Stehen für KEIN Zeichen aus der Gruppe
- Z.B. [^a-z] oder [^0-9]

  - RegEx: W[^0-9]
  - Match: **We**r?

  - RegEx: Diese[^r]
  - Match: Dieser nicht, **dieses** doch

Zeichenklassen (III)

- Minus als erstes
  - RegEx: [-+.,E]
  - Match: -2,5
  - Weitere: -2,5
- Carret (^) irgendwo, nur nicht am Anfang
  - RegEx: [-+^#]
  - Match: 2^3

Alternativen / Alternations

- Alternativen werden durch „|“ getrennt
  - RegEx: Del(f|ph)in
  - Match: Der **Delphin** schwimmt.
  - Match: Der **Delfin** auch.
- RegEx startet links und sucht ersten Treffer
  - RegEx: Hund|Katze|Maus
  - Match: Ich habe eine **Katze** und einen Hund.
  - Weitere: Ich habe eine Katze und einen **Hund**.

Alternativen / Alternations (II)

- Reihenfolge der Alternativen kann wichtig sein
  - RegEx: Auto|Autobus
  - Match: Der **Auto**bus fährt auf der Autobahn.
  - RegEx: Autobus|Auto
  - Match: Der **Autobus** fährt auf der Autobahn.

Quantifizierer / Wiederholungen

Token	kein Mal	ein Mal	mehr-mals	unend-lich	Effekt
?	J	J			Optional
*	J	J	J	J	Beliebig oft
+		J	J	J	Ein oder mehr Mal
{n}		(J)	J		Exakt n Mal
{min,max}	(J)	(J)	J		min bis max Mal
{min,}	(J)	(J)	J	J	Mindestens min Mal

- Wirken jeweils auf den vorherigen Ausdruck

Quantifizierer (I)

- „?“ macht Ausdruck optional
  - RegEx: Hall?o
  - Match: **Halo**genscheinwerfer
- „*“ beliebig oft, auch kein mal
  - RegEx: 10*
  - Match: **1**, 10, 100, 1000
  - Weitere: 1, **10**, **100**, **1000**

Quantifizierer (II)

- „+“ mindestens einmal oder mehrmals
  - RegEx: 10+
  - Match: 1, **10**, 100, 1000
  - Weitere: 1, 10, **100**, **1000**
- „{n}“ exakt n mal
  - RegEx: 10{2}
  - Match: 1, 10, **100**, 1000
  - Weitere: 1, 10, 100, **100**0

Quantifizierer (III)

- „{min, max}“ min bis max mal
  - RegEx: 10{2,3}
  - Match: 1, 10, **100**, 1000, 10000
  - Weitere: 1, 10, 100, **1000**, **1000**0
- „{min,}“ mindestens min mal
  - RegEx: 10{2,}
  - Match: 1, 10, **100**, 1000, 10000
  - Weitere: 1, 10, 100, **1000**, **10000**

Gierige Quantifizierer / Greedyness

- Quantifizierer versuchen immer so viel wie möglich zu erfassen
- Quantifizierer sind „gierig“ / „greedy“

- Gier kann „ausgeschaltet“ werden

- Lösung: Fragezeichen („?“) an Quantifizierer anhängen

Gierige Ausdrücke (II)

- Beispiel HTML-Tag
  - RegEx: <.+>
  - Match: **<B>Test</B>**
- Nicht-gierige Lösung
  - RegEx: <.+?>
  - Match: **<B>**Test</B>
  - Weitere: <B>Test**</B>**

Der Punkt

- Steht für jedes beliebige Zeichen
- Matched alles außer Zeilenende

- Kann in vielen Dialekten per Option umgestellt werden
- Keine Option in VBScript und Javascript

- Alternative Zeichenklasse:
  - [\s\S]
  - \s => Whitespace
  - \S => Alles außer Whitespace

Besondere Zeichen

Token	Bedeutung
^	Textanfang / Zeilenanfang
$	Textende / Zeilenende
\b	Wortgrenze (Wortanfang oder –ende)
\d	Ziffer
\w	Wortzeichen
\s	Whitespace

Text- bzw. Zeilenanfang und -ende

- Einschränkung auf gesamten Text bzw. gesamte Zeile
- Unterschied abhängig von Optionen
  - RegEx: ^Hallo Welt$
  - Match: **Hallo Welt**

Wortgrenze (Anfang o. Ende)

- Nicht sichtbares Zeichen direkt vor bzw. nach dem Wort
- Hier dargestellt durch Pipe-Zeichen („|“)
  - |Ich| |liebe| |Access|
  - RegEx: \bAuto\b
  - Match: Automat, **Auto**

Ziffern / nicht Ziffern

- \d => jede beliebige Ziffer
  (Synonym für [0-9])
- \D => jedes Zeichen, das keine Ziffer ist

  - RegEx: \d*
  - Match: **17**. AEK

  - RegEx: \D*
  - Match: 17**. AEK**

Wortzeichen / nicht Wortzeichen

- \w => jedes alphanumerische Zeichen
  (Synonym für [A-Za-z0-9_])
- \W => jedes nicht-alphanumerische Zeichen

  - RegEx: \w*
  - Match: **17**. AEK

  - RegEx: \W*
  - Match: 17**.** AEK

Whitespace / nicht Whitespace

- \s => jedes nicht sichtbare Zeichen
  (Z.B. Leerzeichen, Tabulator, Zeilenwechsel)
- \S => jedes sichtbare Zeichen

  - RegEx: \s
  - Match: 17. AEK

  - RegEx: \S*
  - Match: **17.** AEK
  - Weitere: 17. **AEK**

Gruppierungen

- Fasst Teile einer RegEx zusammen
- Hilfreich bei Quantifizierern
  - RegEx: \d+(,\d+)?
  - Match: **17**
  - Match: **3,1415**

Gruppierungen (II)

- Werden intern gespeichert
- Referenz über \1, \2, etc.
  - RegEx: (\w+) \1
  - Match: Das Haus, **das das** ich meine.

- Speicherung als Referenz kann verhindert werden
- Nach öffnender Klammer „?:“ einfügen

  - RegEx: (?:\w+) \1
  - Match: Das Haus, das das ich meine.

Look around

- Erlaubt auf nachfolgende (Lookahead) o. vorhergehende (Lookbehind) Zeichen zu prüfen
- VBScript unterstützt nur Lookahead

- Positives Lookahead: (?=)
- Etwas gefolgt von etwas
  - RegEx: Auto(?=bahn)
  - Match: Der Autobus fährt auf der **Auto**bahn

Look around (II)

- Negatives Lookahead: (?!)
- Etwas nicht gefolgt von etwas
  - RegEx: Auto(?!bus)
  - Match: Der Autobus fährt auf der **Auto**bahn
- Text aus Lookahead ist nicht in Match enthalten

Vorrangregeln

- Die eckigen Klammern binden stärker als die runden.
- Stern, Pluszeichen und Fragezeichen binden stärker als Hintereinanderschreiben, d.h. ab* bedeutet a(b*) und nicht (ab)*.
- Der senkrechte Strich bindet schwächer als Hintereinanderschreiben, d.h. ab|c bedeutet (ab)|c und nicht a(b|c).
- Mehrere Sterne, Pluszeichen oder Fragezeichen oder aus diesen Zeichen zusammengesetzte Kombinationen dürfen nicht vorkommen,
  Ausnahme: *? und +?.

Backtracking

- RegEx: ^ab*bc
- Starte am Anfang des Strings, suche ein a gefolgt von möglichst vielen b's gefolgt von einen b gefolgt von einem c.
- Suche in 'abbbbbbc'
  01234567

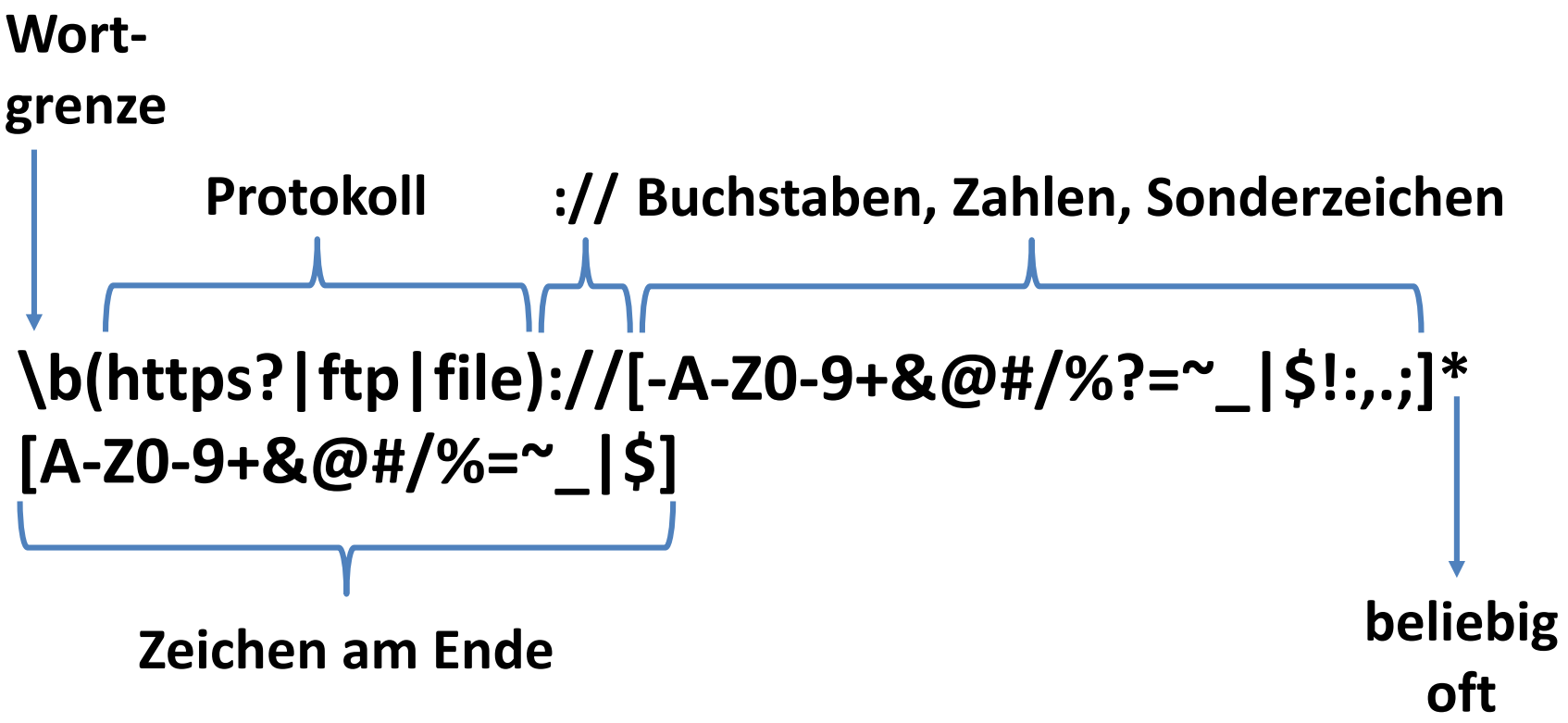
1) ok!		pos = 0. (zw)
2) ok!	Found an 'a' at pos 0.	pos = 1.
3) ok!	Found 6 'b's at pos 1 through 6.	pos = 7.
4) fail!	Did not find a 'b' at pos 7.	Backtrack!
3) ok!	Found 5 'b's at pos 1 through 5.	pos = 6.
4) ok!	Found a 'b' at pos 6.	pos = 7.
5) ok!	Found a 'c' at pos 7.	pos = 8.
Match!		

Quelle: PerlMonks

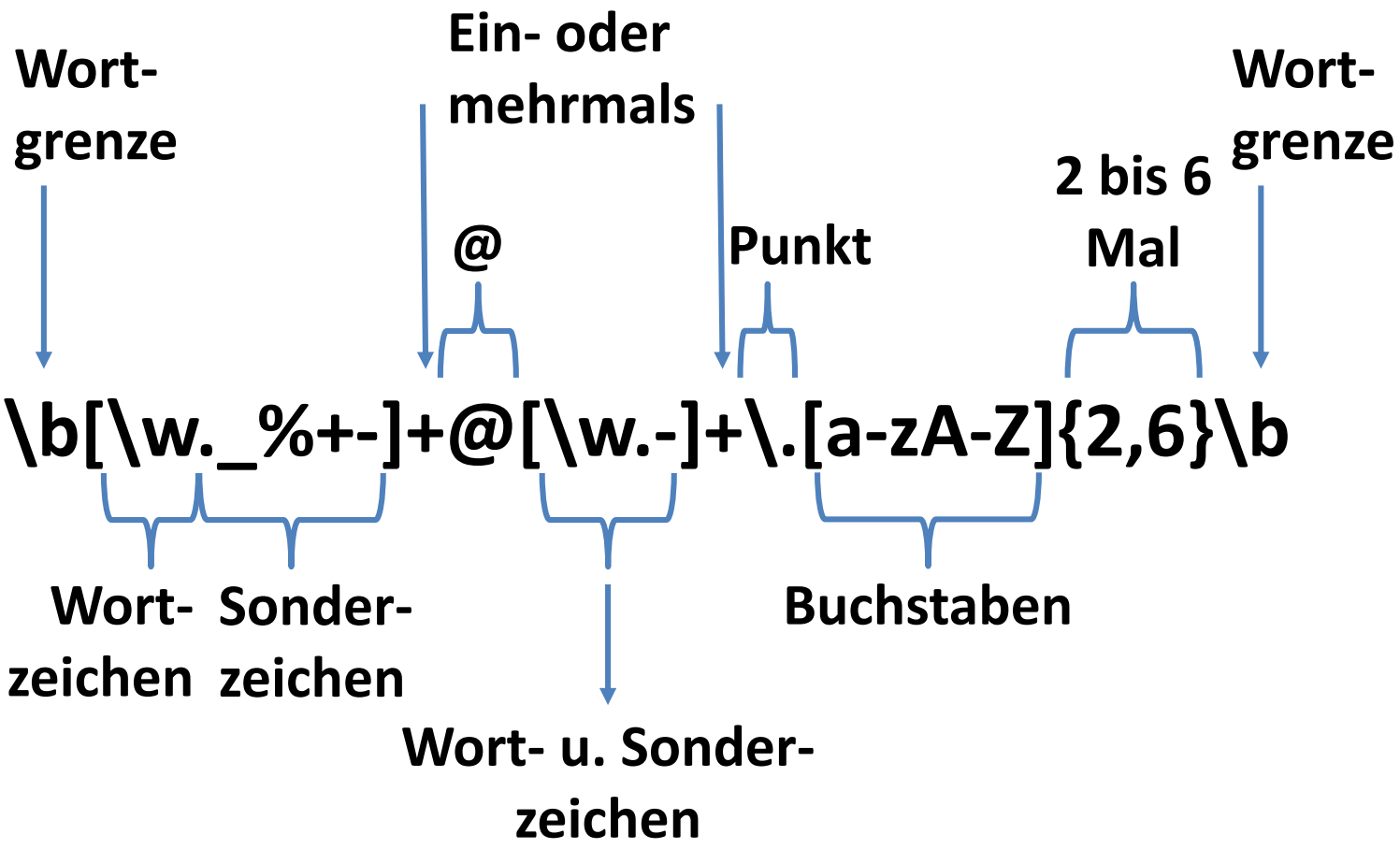
KOMPLEXE BEISPIELE

URL

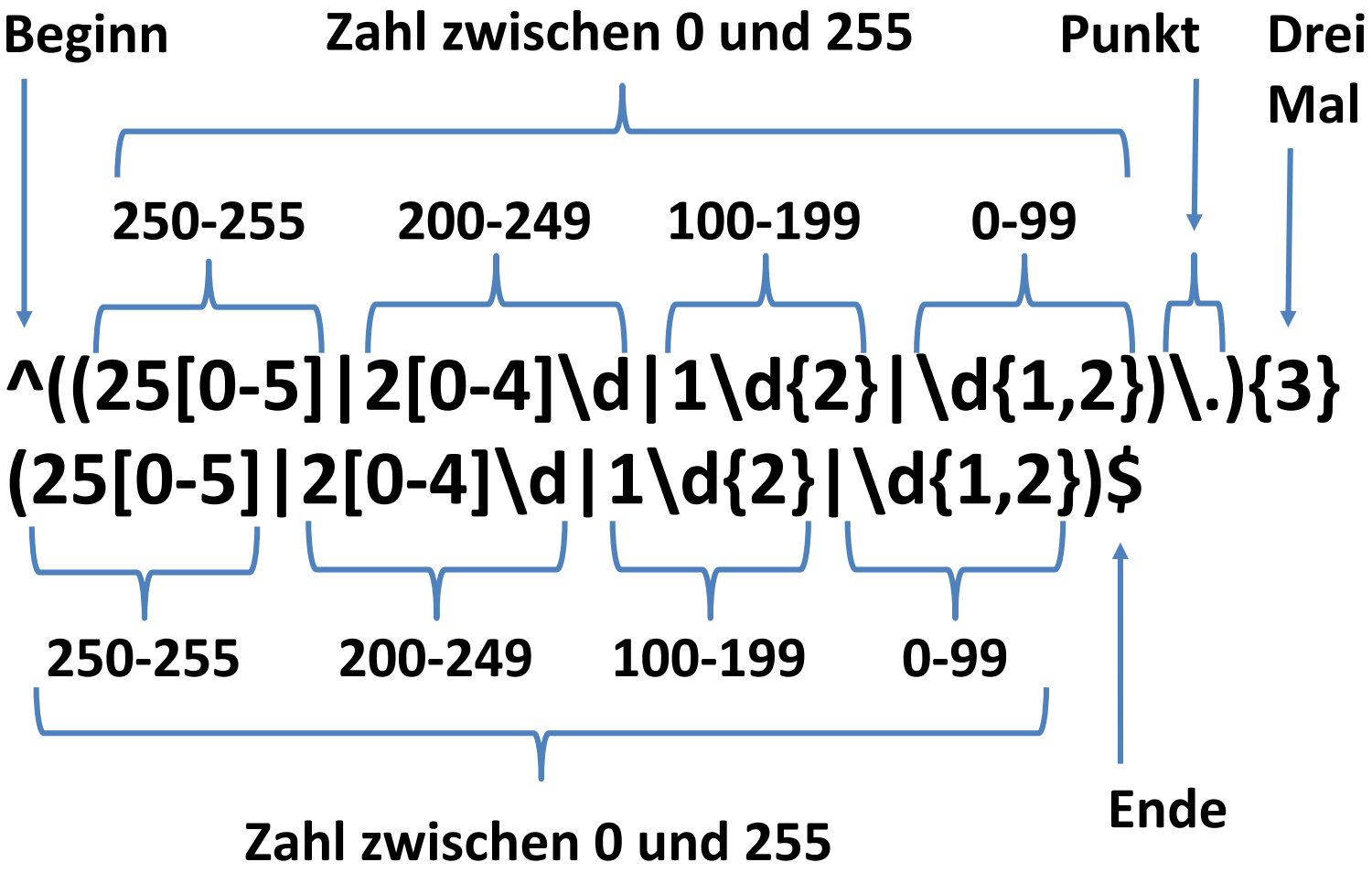
Case-Insensitiv



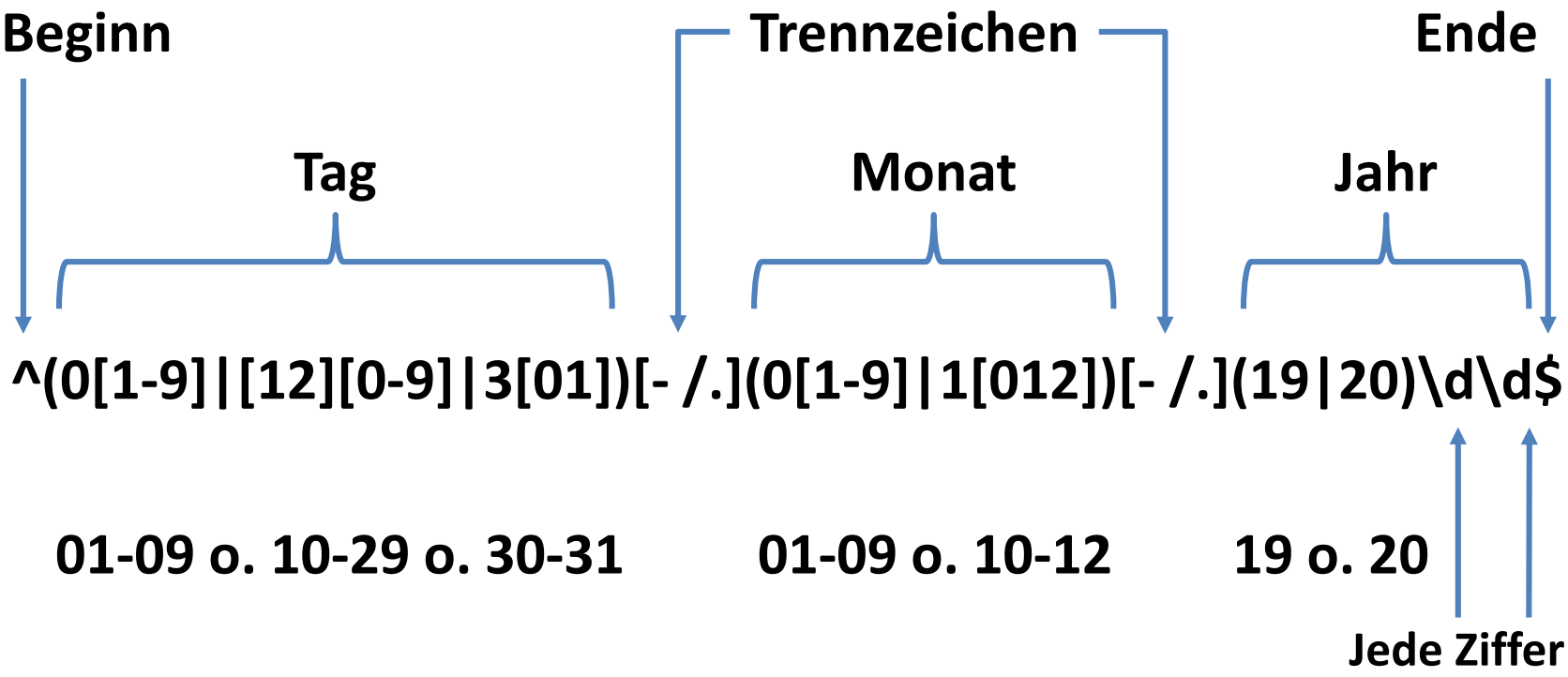
E-Mail Adresse



IPv4 - Adresse

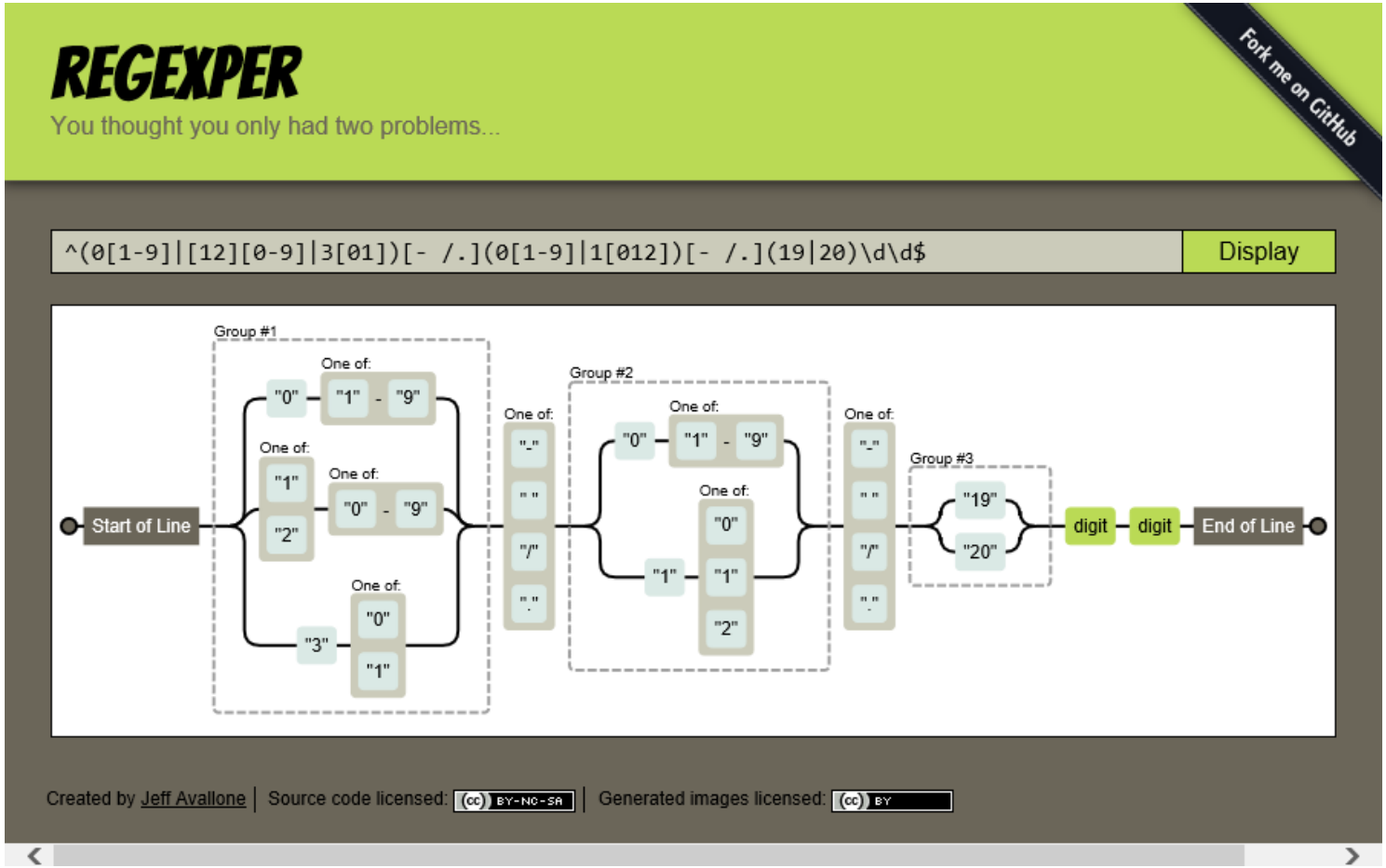


Beispiel: Datum



REGEXPER

- Online-Tool, stellt RegEx grafisch dar:



BEISPIEL ACCESS

Aufgabenstellung

- Einlesen einer Textdatei mit Tarifinformationen
- Tarifschlüssel als Zahlen-Buchstaben-Kombination
- Bedeutung der Zeichen abhängig von Position und vorhergehenden und nachfolgenden Buchstaben
- Datenlieferant konnte/wollte Tarife nicht als eindeutige Zahl liefern
- Eindeutige Tarifinfo wird für Mapping auf interne Informationen benötigt

Lösung / Umsetzung

- Verknüpfung mittels Importspezifikation
- Einlesen der Information aus verknüpfter Tabelle mittels Abfrage
- Abfrage ruft VBA-Funktion auf
- VBA-Funktion ermittelt gewünschten Tarif mit Hilfe von Regular Expressions
- Absicherung mit Acc-Unit

DEMO

BEISPIEL JAVASCRIPT

Aufgabenstellung

- ApEx-Anwendung zur Speicherung von „Prüfregeln“
- Prüfregeln enthalten SQL-Statement
- SQL-Statement muss in Bestandteile (Select, From, Where, GroupBy, Having und OrderBy) zerlegt werden
- SQL-Statement liegt in unformatierter Form vor

- Aufgabe:
  - SQL in Bestandteile zerlegen
  - Ansprechende Formatierung

Lösung / Umsetzung

- Modales Eingabeformular für „rohes“ SQL-Statement
- Suchen und Ersetzen-Funktion für sprechende Aliase
- Beim Klick auf „Übernehmen“ wird SQL-Statement zerlegt und formatiert
- Lösung durch JavaScript-Funktionen
- Dort können RegEx eingesetzt werden
- Tests erfolgen (leider) manuell

DEMO

Auszug JavaScript-Code

```
function RemoveEndingSemicolon(SQL)
{
   SQL = SQL.replace(/;\s*$/,'');
   return SQL;
}

function RemoveWhitespaceAtBoundaries(String)
{
   String = String.replace(/^\s+/,'');
   String = String.replace(/\s+$/,'');
   return String;
}
```

Auszug JavaScript-Code (II)

```
function OptimizeList(List)
{
   List = List.replace(/\s*,\s*/g,' , \n    ');
   List = RemoveWhitespaceAtBoundaries(List);
   List = '    ' + List;
   return List;
}
```

Auszug JavaScript-Code (III)

```
function OptimizeWhere(Where)
{
  Where = Where.replace(/\s*(\b(AND|OR)\b(\s+NOT\b)?)\s*/gi,
                  ' \n    $1 \n        ');
  Where = RemoveWhitespaceAtBoundaries(Where);
  Where = '        ' + Where;
  return Where;
}
```

Auszug JavaScript-Code (IV)

```
function OptimizeSelect(Select)
{
   Select = Select.replace(/\s*\|\|';'\|\|\s*/g," ||';'|| \n    ");
   Select = Select.replace(/\s*,(?=([^\)]*\([^\(]*\))*[^\)]*$)\s*/g,
                " ||';'|| \n    ");
   Select = RemoveWhitespaceAtBoundaries(Select);
   Select = '    ' + Select;
   return Select;
}
```

LESSONS LEARNED

Lessons Learned

- Extrem mächtiges Werkzeug
- Aber: Lernkurve
- Einmal gelernt, überall einsetzbar

- In RegEx denken, sich darauf einlassen
- RegEx Stück für Stück aufbauen
- Testen, Testen, Testen
- Test auch auf unerwünschte Ergebnisse
- Dokumentation RegEx (?)

Bücher

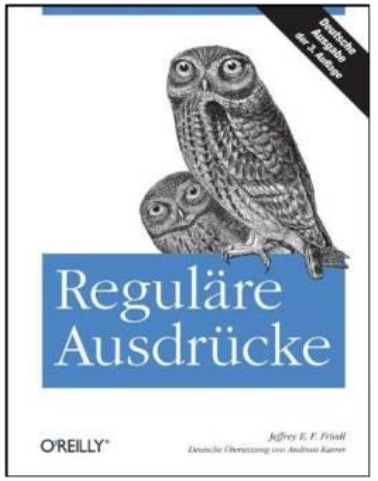
Tools

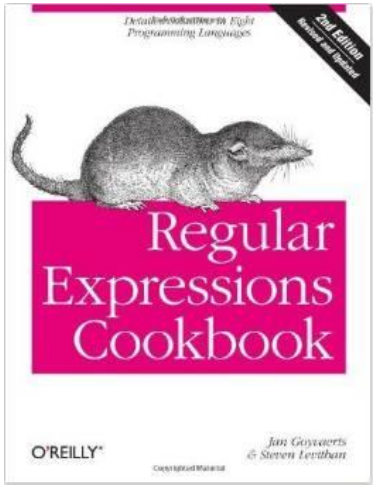
Links

RESSOURCEN

Bücher

- Titel: Reguläre Ausdrücke
- Sprache: deutsch
- Autor: Jeffrey E.F. Friedel
- Ausführliche Einführung in RegEx
- Berücksichtigung verschiedener Dialekte





- Titel: Regular Expressions Cookbook
- Sprache: englisch
- Autoren: Jan Goyvaerts, Steven Levithian
- Teil 1: Einführung in RegEx
- Teil 2: Diskussion zu konkreten RegEx

Tools

- 30 Usefull Tools and Ressources
  - http://www.hongkiat.com/blog/regular-expression-tools-resources/
- Regex Buddy
  - http://www.regexbuddy.com/
- Firefox Add-On: Regular Expressions Tester
  - https://addons.mozilla.org/de/firefox/addon/rext/
- RegExr 2.0
  - http://www.regexr.com/
- Regexpal (Javascript online regex tester)
  - http://regexpal.com/
- Regular Expression Tester
  - http://www.regular-expressions.info/vbscriptexample.html
- REGEXPER (stellt RegEx grafisch dar)
  - http://www.regexper.com/

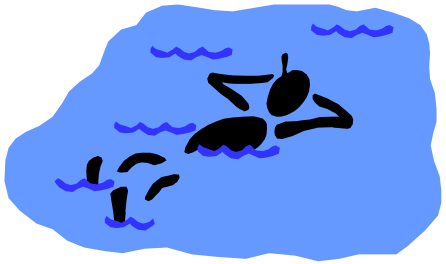
Links

- Regular-Expressions.Info (Die Seite zu RegEx)
  - http://www.regular-expressions.info/
- RegEx auf einer Seite erklärt
  - http://www.regular-expressions.info/quickstart.html
- Tutorial Regulare Ausdrücke
  - https://www.danielfett.de/internet-und-opensource,artikel,regulaere-ausdruecke
- RegexOne: Learn regular expressions
  - http://regexone.com/
- How backtracking works in regular expressions
  - http://www.perlmonks.org/?node_id=390117
- 8 Regular Expressions you should know
  - http://code.tutsplus.com/tutorials/8-regular-expressions-you-should-know--net-6149
- Regular Expressions Library
  - http://regxlib.com/DisplayPatterns.aspx
- Microsoft Beefs up VBScript with Regular Expressions
  - http://msdn.microsoft.com/en-us/library/ms974570.aspx
- 10 tips for using wildcard characters in Microsoft Access criteria expressions
  - http://www.techrepublic.com/article/10-tips-for-using-wildcard-characters-in-microsoft-access-criteria-expressions/

17. Access-Entwickler-Konferenz
Nürnberg 27./28.09.2014 und Hannover 11./12.10.2014

REGULÄRE AUSDRÜCKE

… die Königsklasse der Textverarbeitung



Thomas Möller, www.Team-Moeller.de