



squareroots

Workshop “Reguläre Ausdrücke”

06. November 2008



Alexander Pfister / alex@squareroots.de



PCRE – Perl kompatible reguläre Ausdrücke
(Programm-bibliothek die dem Stand von Perl 5.0
entspricht aber auch Elemente des POSIX
Standards enthält – momentan aktuelle
Programmversion 7.8) <http://www.pcre.org/>

PCRE wird verwendet von Ruby, Python, PHP...

Perl heute hat eine wesentlich komplexere eigene
Bibliothek (PCRE: „*re::engine::PCRE*“)





RegEx – Aufbau von Pattern

Pattern haben immer die Gestalt:

`/pattern/i`

„/“ : Delimiter (jedes beliebige nicht-alphanumerische Zeichen außer „\“)

„`pattern`“ : Eigentliche Pattern

„`i`“ : Modifier (beeinflussen das gesamte Pattern)





- **i** : Ausschalten der Groß- und Kleinschreibung
- **s** : Zeilenweise Suche
- **m** : Komplement von **s** – keine Auftrennung in Zeilen

- (Javascript: **g** - Alle Vorkommen suchen)

Modifier sind beliebig kombinierbar





Vorkommen der voranstehenden Ausdrücke

- ? : Optional / Null oder ein mal ($\{0,1\}$)
- + : mindestens ein mal ($\{1,\}$)
- * : Null oder beliebig oft ($\{0,\}$)
- {*n*} : exakt *n*-mal
- {*min*,} : mindestens *min*-mal
- {,*max*} : maximal *max*-mal
- {*min*,*max*} : mindestens *min*-mal und maximal *max*-mal

Durch ein „?“ welches hinter einem Quantifier steht, wird vom Standard-Greedy Verfahren abgewichen.





RegEx Suchverfahren

Greedy Verhalten

Regulärer Ausdruck: **/D.*t/**

Dieser Satz wird durchsucht.

non-Greedy Verhalten

Regulärer Ausdruck: **/D.*?t/**

Dieser Satz wird durchsucht.





RegEx Zeichen einer Auswahl

- `[]` : zwischen diesen eckigen Klammern stehen normalerweise die erlaubten Zeichen
- `[A-Z]` : Bindestriche symbolisieren einen Bereich (,-' Teil der Auswahl, wenn erstes oder letztes Zeichen oder durch ,\' aufgehoben wird)
- `[^A-Z]` : ,^' Negiert die gesamte Auswahl (,^' ist Teil der Auswahl, wenn es irgendwo in der Mitte steht, oder durch ,\' aufgehoben wird)
- `()` : Gruppierung (Bsp: „(abc)+“ ermöglicht „abc“ oder „abcabc“)
- `|` : ODER (Bsp: (ABC|abc) entweder ABC oder abc aber nicht Abc)





RegEx Benutzung von Klassen

(nicht von allen Interpretern unterstützt)

- `[aInum:]` : alphanumerische Zeichen
- `[alpha:]` : Buchstaben
- `[blank:]` : Leerzeichen und Tabulator
- `[cntrl:]` : Steuerzeichen (Im ASCII sind das die Zeichen 00 bis 1F, und 7F (DEL))
- `[digit:]` : Ziffern: 0, 1, 2,... bis 9 ($\backslash d - \mathbb{D} = [^\backslash d]$)
- `[graph:]` : Graphische Zeichen (`[aInum:]` und `[punct:]`)
- `[lower:]` : Kleinbuchstaben (nicht nur von a bis z)
- `[print:]` : Druckbare Zeichen (`[aInum:]`, `[punct:]` und Leerzeichen)
- `[punct:]` : Zeichen wie: ! " # \$ % & ' () * + , - . / : ; < = > ? @ [\] ^ _ ` { | } ~
- `[space:]` : **Whitespace** (Horizontaler und vertikaler Tabulator, Zeilen- und Seitenvorschub, Wagenrücklauf und Leerzeichen)
- `[upper:]` : Großbuchstaben (nicht nur von A bis Z)
- `[xdigit:]` : Hexadezimale Ziffern: 0 bis 9, A bis F, a bis f
- $\backslash d$ entspricht `[0-9]` $\Leftrightarrow \mathbb{D}$ entspricht `[^\backslash d]`
- $\backslash w$ entspricht `[a-zA-Z_0-9]` (+ Umlaute) $\Leftrightarrow \mathbb{W}$ entspricht `[^\backslash w]`
- $\backslash s$ entspricht Whitespace `[\\f\\n\\r\\t\\v]` $\Leftrightarrow \mathbb{S}$ entspricht `[^\backslash s]`





weitere Zeichen mit Sonderbedeutung

- `.` : steht für ein beliebiges Zeichen (manchmal auch für „Newline“)
- `^` : steht für den Zeilenanfang (`<> [^...]`)
- `$` : kann je nach Kontext für das Zeilen- oder „String“ - Ende stehen, wobei noch ein „`\n`“ folgen darf
- `\z` : tatsächliches Ende
- `\` hebt die Metabedeutung des nächsten Zeichens auf (Bsp: `(A*)+` oder `\.`)
- `\b` steht für die leere Zeichenkette am Wortanfang oder am Wortende
- `\B` steht für die leere Zeichenkette, die *nicht* den Anfang oder das Ende eines Wortes bildet
- `\<` steht für die leere Zeichenkette am Wortanfang
- `\>` steht für die leere Zeichenkette am Wortende
- `\n` steht für einen Zeilenumbruch





RegEx Rückwärtsreferenz

- Mit „\n“ („\$n“ – neue Syntax) kann rückwärtsreferenziert werden. Wobei n für die Anzahl der geöffneten Klammern steht.

Bsp: „A(a|b)B\1“ : Damit gilt nur „AaBa“ oder „AbBb“
aber nicht „AbBa“

- (?:xxx) : verhindert die Rückwärtsreferenz
- (?!xxx) : Negiert den Ausdruck „xxx“ und wirkt sich auf das **vorangegangene** Pattern aus
- (?<!xxx) : Negiert den Ausdruck „xxx“ und wirkt sich auf das **nachfolgende** Pattern aus





RegEx

Rückwärtsreferenz Beispiele

Bsp1: `/(?:Hallo)(Welt)/`
=> \$1 enthält hier „Welt“

Bsp2: `/(lange)(?!Nacht)/`
=> Findet nur eine Übereinstimmung, wenn auf „lange“ **KEIN** „Nacht“ folgt.

Bsp3: `/(?<!CTF)(2008)/`
=> Findet nur eine Übereinstimmung, wenn vor „2008“ **nicht** „CTF“ steht.





RegEx Programme und Webseiten

Programme:

- RegEx- Coach (<http://weitz.de/regex-coach/>)

Webseiten:

- <http://regexp-evaluator.de/evaluator/>
(Auswerter für PCRE + ein paar PHP eigene Sachen) - <http://regexp-evaluator.de/tutorial/>
(Tutorial dazu)
- http://www.devmag.net/webprog/regulaere_aus/
(Interessantes & kurzes Tutorial)

