Studienprogramm Sachbearbeiter:in Digitale Forensik
Praktikum Digitale Forensik Grundlagen
Dozenten: Tim Wetterau | Leander Hossfeld
wetterau@hs-mittweida.de | hossfeld@hs-mittweida.de
Stand: 23.03.2023

HOCHSCHULE MITTWEIDA
University of Applied Sciences

Referenz Reguläre Ausdrücke

Dieses Dokument soll Ihnen eine kleine Referenz für die Gestaltung von regulären Ausdrücken bieten. Es sei darauf hingewiesen, dass dieses Dokument keine vollständige Referenz zum Thema reguläre Ausdrücke ist. Sie ist aber ausreichend zur Lösung der Aufgaben in den Praktikumsveranstaltungen.

Reguläre Ausdrücke

... sind Zeichenketten, die der Beschreibung von Mengen beziehungsweise Untermengen von Zeichenketten dient. Sie stellen Syntaktische Regeln und Suchmuster zum Abgleich von Zeichenketten zur Verfügung und beschreiben den Aufbau der gesuchten Wörter.

Symbole

Werden zur Steuerung des Verhaltens der Regex verwendet. Damit werden bestimmte Zeichen in den Ausdrücken markiert. Eine besondere Bedeutung hat der Punkt (.). Dieser gilt als JOKER, er kann also jedes beliebige Zeichen darstellen. Die Entwertung dieser Sonderzeichen erfolgt mit dem Backslash(\).

	Erklärung	**Beispiel**	
`^`	Zeilenanfang	`^G`	→ G am Zeilenanfang
`$`	Zeilenende	`$G`	→ G am Zeilenende
`|`	Auswahl	`G|A`	→ G oder A
`()`	Gruppierung	`(GSG)*`	→ Der Ausdruck GSG beliebig oft
`.`	Beliebiges Zeichen (Wildcard)	`G.G`	→ Ausdruck von G + bel. Zeichen + G
`\`	Entwerten des Folgezeichens	`\.`	→ Punkt ist kein bel. Zeichen (nur an der Stelle)

Charakterklassen

Charakterklassen bilden die Grundlagen für die Gruppierung von Zeichen. Sie können benutzerdefiniert erstellt werden oder eine aus den vorgefertigten ausgewählt werden.

	Erklärung
`[abc]`	Auswahl von Zeichen
`[^z]`	Negation
`[A-Z]`	Zeichenreichweite
`[:alpha:]`	Groß-/Kleinbuchstaben
`[:blank:]`	Leerzeichen und TABs
`[:digit:]`	Dezimalzahlen von 0-9
`[:alnum:]`	= [:alpha:] + [:digit:]
`[:xdigit:]`	Hexadezimalzahlen von 0-f
`[:lower:]`	Kleinbuchstaben
`[:upper:]`	Großbuchstaben
`[:punct:]`	ASCII-Sonderzeichen
`[:space:]`	Leerzeichen

Beispiel

`[abc]` → ein Zeichen aus a, b oder c
`[^z]` → Zeichen das nicht z ist
`[^BJAC]` → Zeichen außer B, J, A, und C
`[A-Z]` → Zeichen von A bis Z

Charakterklassen können auch aneinandergereiht werden. Beispiel:

➔ `[a-zA-Z0-9] = [[:lower:][:upper:][:digit:]] = [:alnum:]`

Quantifier

Dienen dazu die voransgestellten Wildcards und Charakterklassen in ihrer Vorkommensanzahl einzugrenzen. Dabei sind einseitige, beidseitige und exakte Beschränkungen möglich.

		Erklärung	Beispiel	
`{m,n}`		mindestens m, maximal n	`{1,5}`	→ min 1-mal, max 5-mal
`{m}`		genau m mal	`{5}`	→ genau 5 mal
`{m,}`		mindestens m mal	`{5,}`	→ min 5 mal
`*`	`{0,}`	nicht, einmal oder beliebig oft	`G*`	→ G beliebig oft
`+`	`{1,}`	mindestens einmal	`G+`	→ G min einmal
`?`	`{0,1}`	nicht oder einmal	`G?`	→ G gar nicht oder einmal
`*?`		Wie *, nicht greedy		
`+?`		Wie +, nicht greedy		
`??`		wie ?, nicht greedy		

- ➔ greedy = gesamte Zeile; sucht immer bis zum letzten Zeichen (So viel wie möglich)
- ➔ Nicht greedy = sucht nur bis zum nächsten passenden Zeichen

Gute Beispiele greedy:

G.*G → findet in einer Sequenz vom ersten G bis zum letzten G und beliebig viele Zeichen dazwischen
G.*?G → findet in einer Sequenz vom ersten G bis zum nächsten G beliebig viele Zeichen dazwischen

Bsp.: → GAGGAAGSGAAAGGSGAAAAGSSSSGAAA

G.*G → GAGGAAGSGAAAGGSGAAAAGSSSSG
G.*?G → GAG, GAAG, GAAAG, GSG, GSSSSG

Sollten Sie weitere Informationen zu Regex wollen, können Sie sich gern auf dieser Seiter belesen:

- https://www.replacemagic.com/help/RegularExpressionsMatchwildcards.html
- https://www.webmasterpro.de/coding/einfuehrung-in-regular-expressions/