Suche

TextureSync

Version	1.1.0
Datum	02.05.2019
Autor	Robin Willmann
Projektmitglieder	Hendrik Schutter, Lukas Fürderer, Robin Willmann, Jannik Seiler

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument befasst sich mit dem Syntax und der Semantik der Suchfunktion.

Die Suchfunktion kann nach folgenden Kriterien filtern:

- Vorhandensein Tags
- *Vollständige Namen*
- *Mindest-Auflösung (inklusive)*
- *Einstelldatum liegt vor einem Datum (inklusive)*
- *Einstelldatum liegt nach einem Datum (exklusive)*
- Die Negation eines der Oberen.

Es können mehrere der obigen Kriterien als „logisches Und“ kombiniert werden. Ein „logisches Oder“ ist nicht möglich. Diese Entscheidung wurde getroffen, da durch verschalte logische Operatoren der Syntax zu kompliziert würde.

Die kursiven Kriterien werden unten als Spezialfilter benannt.

2 Eingabe

2.1 Syntaktische Elemente

Er Syntax sei folgendermaßen definiert:

```
Suche := Filter " " ( Suche | ε )

Filter := Negation | Filter1
Filter1  := Spezialfilter | Tagfilter

Negation := "!" Filter1

Spezialfilter := Spezialfiltername ":" Spezialfilterargument
Tagfilter    := Tagname
```

Name	Definition
Spezialfiltername	Siehe unten für mögliche Spezialfilternamen. Groß- und Kleinschreibung bleibt unbeachtet.
Spezialfilterargument	Ist abhängig vom Spezialfiltername, siehe unten.
Tagname	Zeichenkette aus Buchstaben, Zahlen, Bindestrichen und Unterstrichen. Die Groß- und Kleinschreibung wird hierbei nicht berücksichtigt.

2.2 Spezialfilter

Vollständiger Name

Dieser Filter trifft zu, falls der angegebene Name Teil des Namens einer Textur ist. Groß- und Kleinschreibung bleibt unbeachtet.

Spezialfiltername:
„n“ oder „Name“
Spezialfilterargument:
Zeichenkette aus Buchstaben, Zahlen, Bindestrichen und Unterstrichen.
Beispiele:

- `n:Holz`
- `!name:Papier`
- `!nAmE:Papier`

Mindest-Auflösung

Dieser Filter trifft zu, falls Minimum (Breite in Pixeln, Höhe in Pixeln) der Textur größer oder gleich dem als Argument angegebenen Pixel-Anzahl ist.

Die Entscheidung nicht nach Breite und Höhe separat zu filtern wurde getroffen, weil Texturen in der Regel quadratisch sind.

Hinweis: Nach einer Maximal-Auflösung kann gesucht werden, indem dieser Filter negiert wird.

Spezialfiltername:

„r“ oder „Res“ oder „Resolution“

Spezialfilterargument:

Wird durch folgende Regex beschrieben: `[0-9]+k?`

Befindet sich ein „k“ am Ende wird die vorausgehende Zahl mit 1024 multipliziert.

Beispiele:

- `r:4k`
- `!res:8k`
- `resolution:1337`
- `!res:25600`

Einstell-Datum (vorher)

Dieser Filter trifft zu, falls das Einstelldatum der Textur kleiner oder gleich dem als Argument angegebenen Datum ist.

Spezialfiltername:
„b“ oder „Bef“ oder „Before“
Spezialfilterargument:
Wird durch folgende Regex beschrieben: `[0-9]{1,4}\-[0-1]?[1-9]\-[0-3]?[1-9]`
Dies stellt das Format "yyyy-MM-dd" dar, siehe Javadoc unter java.text.SimpleDateFormat für mehr Informationen.
Beispiele:

- b:2018-09-05
- !before:2019-1-3
- bef:2020-9-03

Einstell-Datum (nachher)

Dieser Filter trifft zu, falls das Einstelldatum der Textur größer als Argument angegebenen Datum ist.

Spezialfiltername:
„a“ oder „After“
Spezialfilterargument:
Wird durch folgende Regex beschrieben: `[0-9]{1,4}\-[0-1]?[1-9]\-[0-3]?[1-9]`
Dies stellt das Format "yyyy-MM-dd" dar, siehe Javadoc unter java.text.SimpleDateFormat für mehr Informationen.
Beispiele:

- a:2018-09-05
- !after:2019-1-3

3 Ausgabe

Die anzeigten Elemente bei der Suche werden folgendermaßen bestimmt:

Zunächst werden alle Textur-Metadaten sequenziell durchgegangen.

> Texturen, auf die nicht alle Spezialfilter zutreffen werden übersprungen.
>
> Für jeden zutreffenden Tags (bzw. nicht zutreffenden Tag, bei einer Negation) gibt es $1+\frac{0.1}{\sqrt{(p)}}$ Punkte, wobei p die Postion des Tags im Query ist (beginnend bei 1).
> Somit wird ein Score berechnet.
> *(Anmerkung: Die Position hauptsächlich für die Sortierung eine Rolle. Nutzer geben oft wichtiges zuerst ein.)*
>
> Ist die Anzahl größer als oder gleich die Hälfte (aufgerundet) der angeben Filter, dann wird diese ausgegeben.

Die Ausgabe findet mit größtem Score zuerst statt. Bei gleicher Anzahl wird alphabetisch nach Namen sortiert.

Spezialfälle

Es wird ein leerer Query angeben: Alle Texturen werden angezeigt.

Es wird kein passendes Element gefunden: Im Client wird eine Meldung statt der Texturen angezeigt. Für nähere Informationen siehe Client-Design Dokumente.

4 Client-Server-Interaktion

Die Suche selbst nimmt der Server vor. Dieser schickt dem Client das Ergebnis (Siehe Netzwerkprotokoll). **Für das Validieren der Suchanfrage ist der Client zuständig.** Sollten dennoch ungültige Eingaben den Server erreichen:

Werden diese Leise ignoriert:

- wenn nicht zugelassene Zeichen in der Suchanfrage auftauchen.

Wirft der Server einen Fehler: (siehe Netzwerkprotokoll, Fehlerhandhabung)

- bei Invalidem UTF-8 Encoding.
- Bei Unbekannten Spezialfiltern.

5 Beispiele

Beispieldaten

Nummer	Name	Tags	Eingefügt	Auflösung
#1	wood_185841	Holz, Dunkel, Rot, Edel	2019-05-15	4k
#2	wood_84846	Holz, Hell	2019-05-13	2k
#3	silk_large	Stoff, Rot, Edel	2018-01-01	8k
#4	cotton_xx	Stoff, Rot, Rau	2018-02-01	2k
#5	green_fabric	Grün, Stoff	2018-03-01	1k
#6	tin54_54	Metall, Hell	2018-03-01	4k
#7	copper4_1k	Rot, Metall	2016-03-01	1k
#8	rusty_metall	Rot, Metall, Rost, Dunkel	2015-03-01	8k

Beispiel	Erklärung	Treffer [Score]
Holz *Dunkel*	Zuerst werden Texturen angezeigt, die die Tags *Holz* **und** *Dunkel* beinhalten (Score=~2). Anschließend folgen Texturen, die den Tag *Holz* **oder** den Tag *Dunkel* enthalten (Score=~1). Bedingung: Score ≥ 1 = ceil(2/2)	#1[2,17] #2[1,10] ; #8[1,07]
*n:wood_*	Zeigt alle Texturen, deren Name w*ood_* enthält (Score=0). Bedingung: Score ≥ 0 + Name-Spezialfilter	#1[0] ; #2[0]
before:2019-05-31 *after:2019-04-30*	Zeigt alle Texturen an, die im Mai 2019 hinzugefügt wurden (Score=0). Bedingung: Score ≥ 0 + Before-Spezialfilter + After-Spezialfilter	#1[0] ; #2[0]
Stoff *Rot* *Edel*	Zuerst werden Texturen angezeigt, die die Tags *Stoff*, *Rot* **und** *Dunkel* beinhalten (Score=~3). Dann folgen Texturen mit entweder: • *Stoff* **und** *Rot* *(Score=~2)* • *Stoff* **und** *Edel (Score=~2)* • *Rot* **und** *Edel* *(Score=~2)* Bedingung: Score ≥ 2 = ceil(3/2)	#3[3,23] #4[2,15] ; #1[2,13]
Stoff *Rot* ***!****Edel*	Zuerst werden Texturen angezeigt, die die Tags *Stoff*, *Rot* **und nicht** *Edel* beinhalten (Score=~3). Dann folgen Texturen mit entweder: • *Stoff* **und** *Rot* *(Score=~2)* • *Stoff* **und *nicht*** *Edel (Score=~2)* • *Rot* **und *nicht*** *Edel* *(Score=~2)* Bedingung: Score ≥ 2 = ceil(3/2)	#4[3,23] #3[2,17] ; #5[2,15] ; #7[2,13] ; #8[2,13]
Metall *Dunkel* *res:4k*	Zuerst werden Texturen angezeigt, die eine Auflösung von ≥ 4k haben, den Tag *Metall* **und** den Tag *Dunkel* enthalten (Score=~2). Dann werden Texturen angezeigt, die eine Auflösung von ≥ 4k haben **und {** den Tag *Metall* **oder** den Tag Dunkel **}** enthalten (Score=~1). Bedingung: Score ≥ 1 = ceil(2/2) + Res-Spezialfilter	#8[2,17] #6[1,10] ; #1[1,07]

6 Changelog

Version	Änderung
0.1.0	-

1.0.0	• Füge Beispiele ein.
	• Neu: After-Spezialfilter
	• Anpassung der Auswertung:
	  ◦ (~~abgerundet~~ aufgerundet)
	  ◦ Spezialfilter sind nun erforderlich.
	  ◦ Neu: Spezialfälle
	• Zuständigkeiten jetzt Fett (Client-Server Interaktion)

1.1.0	Verbesserter Score (Einbeziehung der Postion im Query.)