Reguläre Ausdrücke ... ein langwieriger Zeitvertreib

Erfasst am : 7. November 2010 20:21 | Erfasst von : Martin
Verknüpfte Kategorie(n): Arbeit, Internet

Reguläre Ausdrücke, das sind die Programmregeln, die man aufbauen kann, um mit einer sie verstehenden Final State Machine beliebige Daten auf Muster abzugrasen. Woran erkennen wir Menschen denn, dass eine Telefonnummer eine solche ist und keine IBAN, oder umgekehrt?

Am Format: Eine Telefonnummer folgt in der Schweiz der Regel aaa-bbb cc dd, also 043-123 45 67. Schriebe ich sie einfach so 0431234567, so wäre es wohl nur irgendeine Nummer. Erst der Hinweis, dass es eine Telefonnummer wäre, würde einen sie erkennen lassen.

Eine IBAN ist eine 20 bis 34-stellige Nummer, die formal mit dem Landescode beginnt, danach folgend nur noch Zahlen, gruppiert in Vierergruppen, CH12 3456 7890 1234 5678. Der Teil ab der 5. Stelle ist eine Kontoidentifikation, die von Land zu Land verschieden sein kann.

Viele Nummern folgen also rein formal darstellerisch Regeln. Nicht weil diese Darstellungsart nötig wären, aber vor dem Computerzeitalter mussten die Menschen sich diese Nummern ja irgendwie merken können - und das geht fürs Gehirn einfacher, wenn es Zusatzinformationen mit der Zahl verlinkt speichern kann. Da ein Format eine Art Bild und das Gehirn seit Äonen auf Bilderkennung getrimmt ist, können wir uns formatierte Zahlen merken.

Wie dem auch sei: Für ein kleines Projekt musste ich mich wiedermal damit auseinandersetzen, was denn eine URL ausmacht. Eine URL, wie sie im Internet andauernd vorkommt, ist oft einfach, aber sie kann auch komplex sein. In jedem Fall folgt sie dennoch einem Regelset, auch wenn das nicht immer leicht zu erkennen ist.

Wenn ich also einen Regulären Ausdruck finden möchte, der aus einem beliebigen Text URLs entdecken kann, so muss ich den Aufbau der URLs kennen.

Nun, bin ja nicht der erste, der sowas macht / machen muss. Und ich kenne Reguläre Ausdrücke sehr gut. Aber es ist schon immer wieder knackig. Gerade weil URIs so komplex sein können, gestaltet sich das Finden des auf sie alle passenden Regulären Ausdruckes als knifflig.

Natürlich schaue ich auch im Internet, was dort vorhanden ist. Doch die Lösungen haben alle Haken. Meine Lösung zeige ich mal, danach folgt eine recht gute Lösung, die allerdings Syntaxregeln für URLs nicht befolgt.

```
r_proto  = "(([a-z]+:\/\/)|mailto:|javascript:)";           //   https://, mailto:, javascript:
r_user   = "([\w\.-]+(:\w[\w\.-]+)?@)?";                    //   user@, user:pass@
r_host   = "([\w\.-]+)";                                    //   www.domain.com
r_port   = "(:\d{1,5})?";                                   //   :80
r_uri    = "([^\?\b\s:]+";                                  //   /path/to/file.ext
r_params = "(\?[^\b\s:]+)?";                                //   ?param1=val1&param2=val2
r_anchor = "(#[^\b\s:]+)?\b";                               //   #anchor
```

Damit kann man nun Regeln für URLs zusammenstellen. Denn im Internet gibt es absolute und relative URIs. Absolute müssen mit dem Protokoll und einer Hostadresse beginnen, gefolgt von dem, was man eine relative URI nennt. So kann man also drei verschiedene Arten von Regulären Ausdrücken zimmern, die je eine oder beide URIs zusammen finden können.

So kann ich dann im Werkzeug meiner Wahl, hier also ColdFusion, folgende drei Variablen zusammenbauen.

```
uri_any  =  "(?i)\b(" & r_proto & r_user & r_host & r_port & ")?" & r_uri & r_params & r_anchor;
uri_abs  =  "(?i)\b" & r_proto & r_user & r_host & r_port & r_uri & r_params & r_anchor;
uri_rel  =  "(?i)\b" & r_uri & r_params & r_anchor;
```

Das klappt auch sehr gut so. Damit kann ich nun das Projekt abschliessen.

Weshalb ich das veröffentliche? Eben, es gibt im Internet Tausende von solchen Lösungen. Wie eingangs erwähnt, hat diese auch Schwächen: Sie unterstützt keine Notationen mit **`Ӓ`**. Auch der Begriff **`Localhost`** wird nicht erkannt, beides brauche ich nicht und sie kommen auch selten vor. Beides wäre natürlich zu machen, aber ich möchte die einzelnen Blöcke nicht komplizieren.

Der ebenfalls im Internet auffindbare Reguläre Ausdruck

```
(?i)\b((?:[a-z][\w-]+:(?:/{1,3}|[a-z0-9%])|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)
(?:[^\s()]+|\(([^\s()]+|(\([^\s()]+\)))*\))+(?:\(([^\s()]+|(\([^\s()]+\)))*\)|
(\([^\s()]+\))*\)|[^\s`!()\[\]{};:'".,<>?«»“”‘’]))
```

Ist auch mächtig, aber er eignet sich nicht für relative URIs. Zudem: Reguläre Ausdrücke haben den Ruf von Unverständlichkeit. Äh ja, der hier gehört definitiv dazu und ist ein wunderbarer Förderer dieses Klisches ... :-)

Wichtig: Meine RegExs (wie man die bei Programmieren halt nennt - hat nichts mit Ex-Königin aka Ex-Freundin zu tun) beachten die Syntaxregeln so gut es geht. Der URI-Parameterblock wird nur dann erkannt wenn er wirklich mit einem ? beginnt. Der obige Krypto-RegEx erlaubt auch syntaktisch falsche URIs.

Der Kryptische findet auch Dinge, die ich nicht zu einer URI gehörig empfinde: so werden dort auch CSS-Dinge wie **`z-index:110`** oder **`position:absolute`** gefunden.

Beide lassen übrigens die Parent-Directory-Notation .. durch, aber auch ... und ..... und Doppelslashes wie // oder gar ////. Wie gesagt, RegEx können beliebig kompliziert sein, meine und der Kryptische sind hier pragmatisch und schlucken das. Denn leider kann man in einem RegEx nicht eben mal temporär ein Symbol definieren, so dass man .. als Parent-Directory im Ausdruck als atomares Symbol verwenden könnte.

Wieso geht das denn eigentlich so lange, einen passenden regulären Ausdruck zu finden? Nun, das hängt eben von der Komplexität der Zielmuster (hier die URIs) ab. Und da wir Menschen keine Maschinen sind, wissen wir auch nicht sofort, ob ein RegEx wirklich nur das findet, was wir wollen. Das Feintuning eines RegEx geschieht daher meistens anhand realer umfangreicher Daten. Sie sollen ja so genau wie möglich nur das Gewünschte finden, doch sie können auch mal was in den Daten erwischen, woran keiner im voraus dachte.

Dies macht RegEx zu den zwar einfach zu erlernenden Programmierhilfen, doch sie zu meistern, das ist eine harte Knacknuss. Wie gesagt: Zeitvertreib, ich habe doch einige Stunden gekniffelt, bis das Set nun steht. Anhand einiger Testdaten mit heiklen Test-URIs habe ich ein gutes Gefühl für ihre Korrektheit entwickelt.

Aber natürlich gilt: Ich übernehme keinerlei Haftung für das Zeug ... :-) ... genauso wie es im Text der GPL steht.

PS: Verbesserungen werden natürlich dankend entgegengenommen ...