Regularni izrazi
(JMBG)

Regularni izraz se sastoji od skupa karaktera, od kojih neki imaju specijalno značenje. Ovaj skup karaktera se koristi kao šablon (pattern) za pronalaženje delova teksta koji se sa njim poklapaju ili za validaciju niza drugih karaktera. Regularni izrazi mogu da budu predstavljeni i korišćenjem konačnog automata. Način na koji se vrši poklapanje šablona sa uzorcima teksta je primer korišćenja konačnog automata.

Primena regularnih izraza je ogromna. Od najprostijih, kao što su posebna vrsta pretraživanja u teksualnim editorima, preko različitih validacija, kao što smo mogli da vidimo na primeru validacije u ASP.NET Core-u, ali i u JavaScript-u, pa do primene prilikom sintaksnog bojenja i parsiranja sintakse gotovo svih programskih jezika od strane IDE okruženja.

U najkraćim crtama, osnovni karakteri koji imaju specijalno značenje u regularnim izrazima su predstavljeni kroz sledeće kategorije:

Više na: https://www.rexegg.com/regex-quickstart.html

Karakteri:

Karakter	Značenje	Primer	Poklapanje
\d	Jedna cifra (Isto značenje kao [0-9])	Broj_\d\d\d	Broj_230
\w	"Word" karakter: ASCII slovo, broj ili podcrta	\w\w\w\w	Ab_3
\s	"Whitespace" karakter: razmak, tabulator, nova linija, carriage return, vertikalni tabulator	a\sb\sc	a b c
\D	Suprotno značenje od \d (sve što nije cifra: [^0-9])	\D\D\D	A#b
\W	Karakter koji nije "Word" karakter (\w)	\W\W\W	=*+
\S	Karakter koji nije „Whitespace" karakter (\s)	\S\S\S	Pr4

Kvantifikatori:

Kvantifikator	Značenje	Primer	Poklapanje
+	Jednom ili više puta	\w+	Ab_3
{3}	Tačno 3 puta	\D{3}	Ab3
{2,4}	Od 2 do 4 puta	\d{2,4}	156
{3,}	3 ili više puta	\w{3,}	Primer
*	Nula ili više puta	A*B*C*	AAAC
?	Jednom ili nijednom	Tog?	To
+?	"Lazy" + kvantifikator (hvata svaki broj pojedinačno)	\d+?	**1**2**3** Grupe: 1, 2, 3
*?	"Lazy" * kvantifikator (hvata prvo prazan string, pa onda jedan broj i tako do kraja)	\d*?	123 Grupe : null, 1, null, 2, null, 3
{2,4}?	"Lazy range" kvantifikator	\d{2,4}?	1234 Grupe: 12, 34

Specijalni karakteri:

Karakter	Značenje	Primer	Poklapanje
.	Bilo koji karakter, ne računajući kraj linije	Tek.t	Tekst
\.	Tačka (mora da bude nakon karaktera \, escaped)	Tek\.t	Tek.t
\	Bilo koji specijalni karakter se korišćenjem \ karaktera pretvara u običan karakter koji sledi	\.\\\/\[\+*\$	.\/[+*$

Logički operatori:

Operator	Značenje	Primer	Poklapanje
|	OR logička operacija	21|22	21
(…)	Grupa	Sta(klo|vka)	Staklo
\1	Kopira sadržaj prve grupe	(\w{2})la\1k	Zalazak
\2	Kopira sadržaj druge grupe	(\d\d)\+(\d\d)=\2\+\1	12+34=34+12
(?: …)	Grupa koja ne ostaje upamćena	Sta(?:klo|vka)	Staklo

White space karakteri:

Karakter	Značenje
\t	Tabulator
\r	"Carriage return" karakter
\n	"Line feed" karakter
\r\n	Kombinacija karaktera koji predstavljaju novu liniju u Windows-u
\N	Karakter koji nije kraj linije
\h	Horizontalni "whitespace" karakter
\H	Karakter koji nije horizontalni "whitespace" karakter
\v	Vertikalni tabulator
\V	Karakter koji nije vertikalni tabulator

Klase:

Karakteri	Značenje	Primer	Poklapanje
[…]	Sva slova u uglastim zagradama se poklapaju	[AEIOU]	AE
[x-y]	Opseg od x do y	[A-Z]+	VELIKO
[^x]	Jedno slovo koje nije x	[^a-z]{3}	S9F!
[^x-y]	Više slova koja **nisu** od x do y	[^0-3]+	456
[\d\D]	Jedno slovo koje je iz grupe \d ili \D	[\d\D]+	Sva slova i 5
[\x41]	Heksadecimalni karakter	[\x33-\x35]{3}	345

Regularni izrazi sadrže i neke druge specijalne karaktere. Neki od njih su dostupni samo u određenim jezicima (neki u JavaScript-u, većina u PHP-u...). Od korisnijih izvdajaju se još:

Karakter	Značenje
^	Početak linije
$	Kraj linije
\A	Početak string-a
\z	Kraj string-a
Modifikatori pretrage (samo u nekim jezicima)	
(?i)	Case insensitive
(?s)	Single line mod (. poklapa svaki karakter osim kraja linije)
(?m)	Multiline mod (. poklapa i krajeve linija)

Posebna kategorija karaktera koja se izdvaja su lookaround operatori. U njih spadaju **lookbehind** i **lookahead**, dok dalje postoje **pozitivni** i **negativni**.

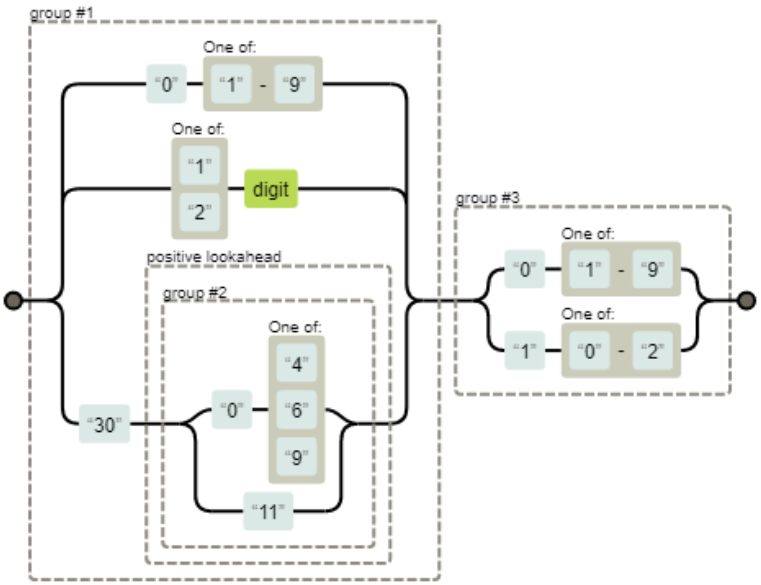
Positive lookahead:

Positive lookadhead operator predstavlja sledeći skup karaktera: **(?=...)**

Primer ovog operatora možemo da vidimo na sledećem primeru:

(0[1-9]|[12]\d|30(?=(0[469]|11)))(0[1-9]|1[0-2])

Da bi smo bolje razumeli šta ovaj regularni izraz predstavlja, prikazaćemo ga grafički:



Na slici možemo da vidimo kako prvi karakter koji se pretražuje je karakter „0“. Ukoliko njega pronađemo, sledeći su karakteri između „1“ i „9“ (One of). Osim ovog puta, postoji i 2 alternativna. Jedan počinje karakterom „1“

ili „2“, gde zatim sledi bilo koji broj, a drugi pretražuje „30“. Najzanimljiviji deo sledi u bloku koji je nakon „30“. Ovaj „positive lookahead“ blok predstavlja proveru karaktera koji su nakon „30“. Ukoliko se nakon „30“ nalazi „0“,, praćena „4“, „6“ ili „9“, kao i druga opcija „11“, onda će „30“ biti uzet u razmatranje. U ostalim slučajevima biće ignorisan. To znači da engine koji parsira regularni izraz „odlazi korak u budućnost“ i pretražuje deo string-a koji nije još došao na red, a od rezultata tog pretraživanja zavisi da li će trenutni rezultat da bude poklopljen ili ne.

Konačno, vidimo da postoji još jedan blok, koji počinje ili „0“ ili „1“, a ukoliko se radi o „0“, onda bilo koji karakter između „1“ i „9“ sme da se nađe nakon njega, dok u slučaju jedinice, to mogu da budu samo karakteri „0“, „1“ ili „2“.

Sada već možemo da vidimo da se radi o delu parsiranja datuma (dan i mesec), koji obuhvata mesece koji imaju 30, ali ne i one koji imaju 31 dan.

Positive lookbehind:

Slično lookahead operatoru, lookbehind proverava da li se karakter ili niz karaktera nalazi ispred našeg poklapanja. **(?<=Broj\:)\d{2}** znači da pokušavamo da pronađemo niz karaktera „Broj:“, a zatim naš niz od dva broja. Ovaj izraz možemo (kao i prethodni) da napišemo na drugi način. Svaki lookaround operator može da se piše ispred ili iza našeg poklapanja. Ukoliko ga u ovom slučaju napišemo nakon poklapanja, to bi izgledalo ovako: **\d{2}(?<=Broj\:\d{2})**. Razlika između ova dva pristupa je ta što se u drugom slučaju izraz nalazi nakon poklapanja, pa zato, kada prosleđujemo lookbehind operator moramo da mu prosledimo i taj broj koji želimo da preklopimo ponovo. Zato je ovaj metod manje efikasan od prethodnog.

Negative lookahead:

Za razliku od positive varijanti lookaround operatora, negative lookaround operatori pokušavaju da zaključe kada se poklapanje ne nalazi nakon ili ispred odgovarajuće vrednosti. Zato je sintaksa za ove dve varijante veoma slična prethodnoj.

(?!\d+kg)\d+ Ovaj izraz može da pronađe bilo koji niz cifara, ukoliko se nakon njega ne nalazi jedinica *kg*.

Negative lookbehind:

Ono što preostaje je negative verzija lookbehind operatora. Kao što smo već rekli, to znači da ispred našeg poklapanja ne sme da bude odgovarajući niz karaktera, i pišemo ga na sledeći način: **(?<!N\=)\d+** što u prevodu znači da neće biti uhvaćen broj u tekstu N=100, ali hoće u npr. K=100.

Ukoliko sada pređemo na problem matičnog broja, ovaj problem može da bude rešen na veliki broj načina. Neki od njih su naravno trivijalni, ali više netačni. Drugi su komplikovaniji, ali bolje reprezentuju realnost datuma i ostalih cifara u matičnom broju.

Najprostiji način

Najjednostavniji način, ukoliko pročitamo specijalne vrednosti iz tabela na vrhu bi bio korišćenje kvantifikatora za brojeve.

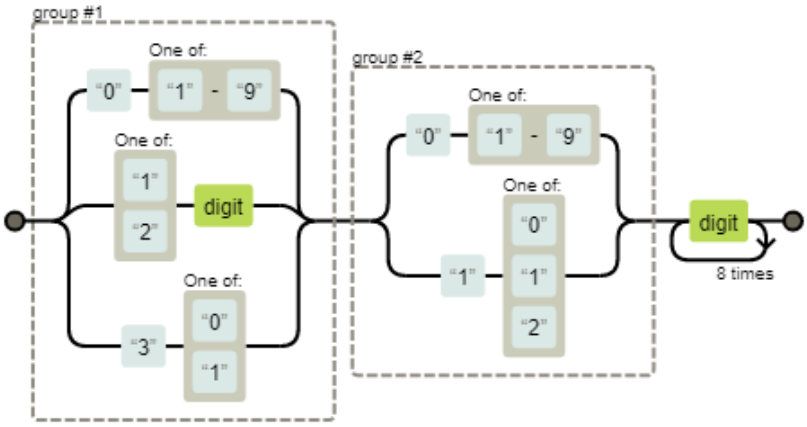
\d{13}

Na ovaj način možemo da uhvatimo broj koji se ponavlja 13 puta. Problem sa ovim pristupom je očigledan. Nije svaki broj dozvoljen u datumu, ali isto tako ni u narednim ciframa. Zato, da bi smo rešili neke od ovih problema, prelazimo na sledeći pristup: Naivni pristup

Naivni pristup

Ukoliko malo usložimo prethodni regularni izraz, možemo da se pozabavimo brojevima iz dela zaduženog za datum.

(0[1-9]|[12]\d|3[01])(0[1-9]|1[012])\d{9}



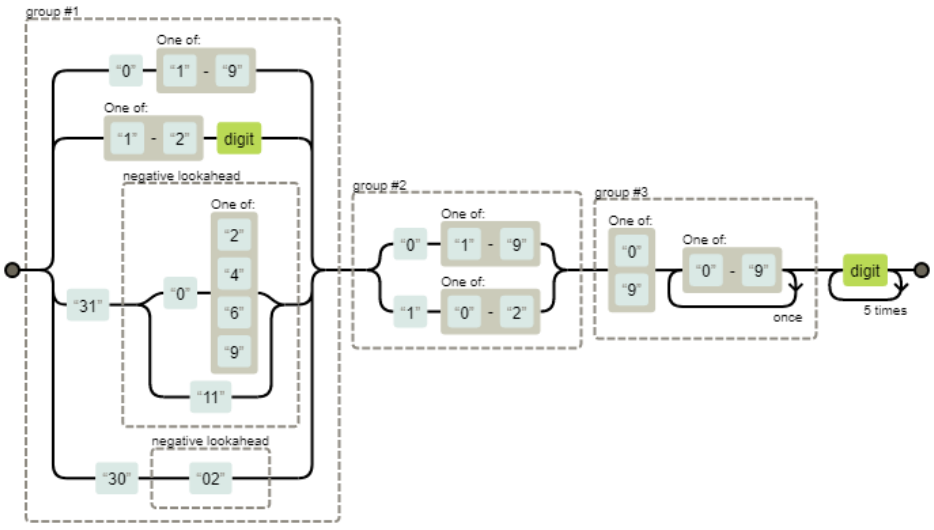
Razlika nije prevelika, tako da ni težina realizacije ovog regularnog izraza nije previeš komplikovana. U ovom slučaju smo koristili veoma sličan regularni izraz gore korišćenom za datum. Proverava se da li broj počinje sa 0, ukoliko je to slučaj, ograničavamo sledeću cifru na brojeve od 1-9. „|“ operator se koristi između svake grupe da bi se uzele u obzir i druge mogućnosti. Tako je druga mogućnost da broj počinje ciframa 1 ili 2, a bilo koji broj može da sledi. Treća mogućnost je broj 3, gde sledeća cifra može da bude samo 0 ili 1. Time smo brojeve ograničili na brojeve od 01-31. Sledeća grupa predstavlja mesec, pa je zato prva cifra takođe 0. Ukoliko je to slučaj, sledeća cifra može biti bilo koja od 1-9. Ukoliko je broj 1, onda možemo da koristimo karaktere 0, 1 ili 2. Nakon ovoga, sledi još 9 brojeva koji moraju da budu deo matičnog broja.

Iako smo ograničili brojeve koji predstavljaju datum na 01-31, ovo i dalje nije dovoljno za većinu meseci, ni one koji traju 30 dana, a ni februar, koji traje 28 ili 29 dana. Zato prelazimo na sledeći pokušaj: Početnički pristup

Početnički pristup

I dalje smo na problemu datuma, tako da ćemo da pokušamo da budemo još precizniji i ograničimo datume na 30 dana ukoliko toliko dana imaju, dok ćemo kod februara u ovom pokušaju da se zadržimo na 29 dana.

(0[1-9]|[1-2]\d|31(?!(?:0[2469]|11))|30(?!02))(0[1-9]|1[0-2])([09][0-9]{2})\d{6}



Prvi put koristimo lookaround operatore. Ovaj regularni izraz već počinje da podseća na nešto jako komplikovano, pa sada slika može da nam olakša posao razumevanja drastično. Kada pogledamo sliku, situacija je mnogo jednostavnija.

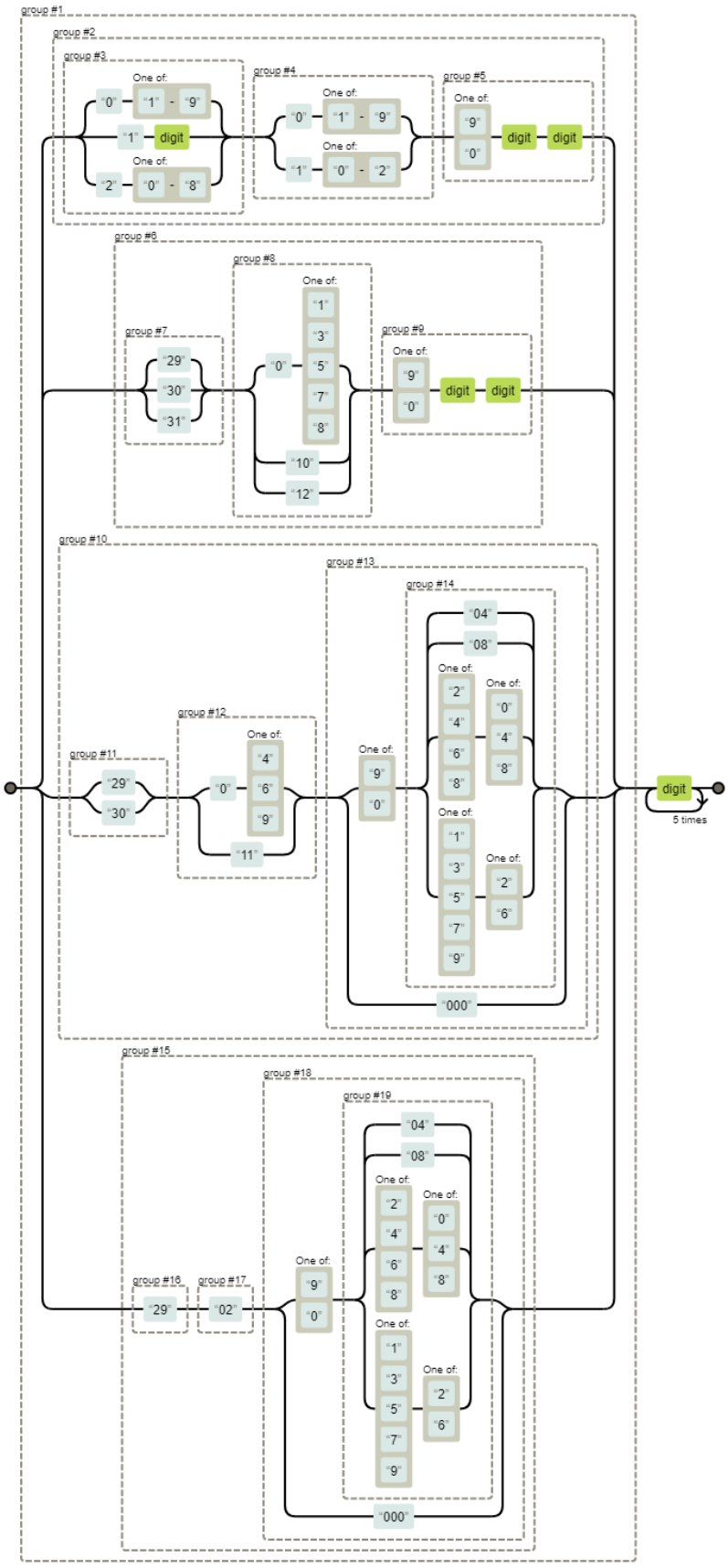
Postiji 2 razlike u odnosu na prethodni primer. Jedna je prepoznavanje meseci uz pomoć negative lookahead operatora, koje nam omogućava da ne dozvolimo poklapanje ukoliko imamo dan 31, ukoliko meseci nisu 02, 04, 06, 09 ili 11, a ne dozvoljavamo da dan bude 30 ukoliko je mesec 02. To nam omogućava da imamo ispravne sve mesece osim februara, za koji koristimo vrednost 29 uvek.

Druga razlika je godina. Godina može da bude u opsegu od 900-099 (iako možda ne deluje tako iz primera). Prvi broj može da bude ili 0 ili 9 (pošto je prva cifra godine preskočena, to znači da druga cifra, cifra koja označava vek, može da bude 0 ili 9, za 1900-te i 2000-te godine). Problema ima još puno, zato prelazimo na sledeće rešenje: kompetentni pristup.

Kompetentni pristup

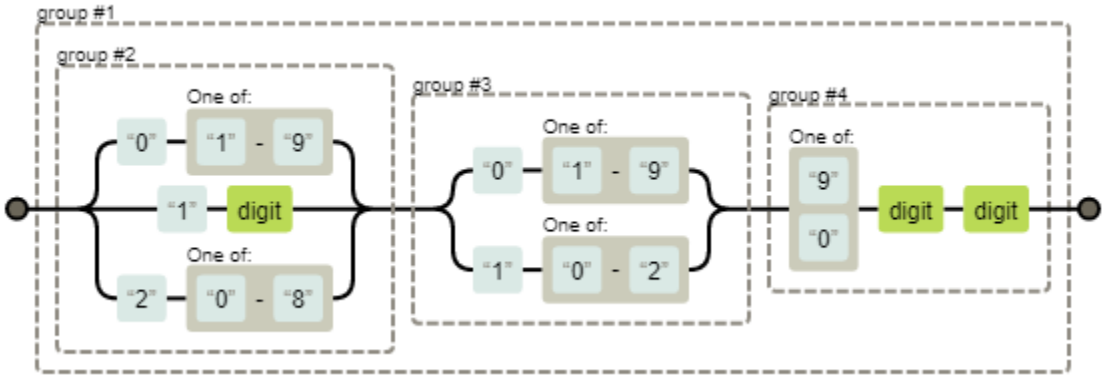
Sada već počinjemo da se bavimo najtežim problemom u obradi datuma, a to je februar mesec. Ovaj problem ćemo da rešimo na 2 načina. U jednom ćemo da problem rešimo bez korišćenja lookaround operatora, a u drugom ćemo da uprostimo taj izraz njihovim korišćenjem.

(((0[1-9]|1\d|2[0-8])(0[1-9]|1[0-2])([90]\d\d))|((29|30|31)(0[13578]|10|12)([90]\d\d))|((29|30)(0[469]|11)([90](04|08|[2468][048]|[13579][26])|000))|((29)(02)([90](04|08|[2468][048]|[13579][26])|000)))


group #1
group #2
group #3
One of:
"0"
"1" - "9"
"1"
digit
"2"
One of:
"0" - "8"
group #4
One of:
"0"
"1" - "9"
"1"
One of:
"0" - "2"
group #5
One of:
"9"
"0"
digit
digit
group #6
group #7
"29"
"30"
"31"
group #8
One of:
"0"
"1"
"3"
"5"
"7"
"8"
"10"
"12"
group #9
One of:
"9"
"0"
digit
digit
group #10
group #11
"29"
"30"
group #12
One of:
"0"
"4"
"6"
"9"
"11"
group #13
One of:
"9"
"0"
group #14
"04"
"08"
One of:
"2"
"4"
"6"
"8"
One of:
"0"
"4"
"8"
One of:
"1"
"3"
"5"
"7"
"9"
One of:
"2"
"6"
"000"
digit
5 times
group #15
group #16
"29"
group #17
"02"
group #18
One of:
"9"
"0"
group #19
"04"
"08"
One of:
"2"
"4"
"6"
"8"
One of:
"0"
"4"
"8"
One of:
"1"
"3"
"5"
"7"
"9"
One of:
"2"
"6"
"000"

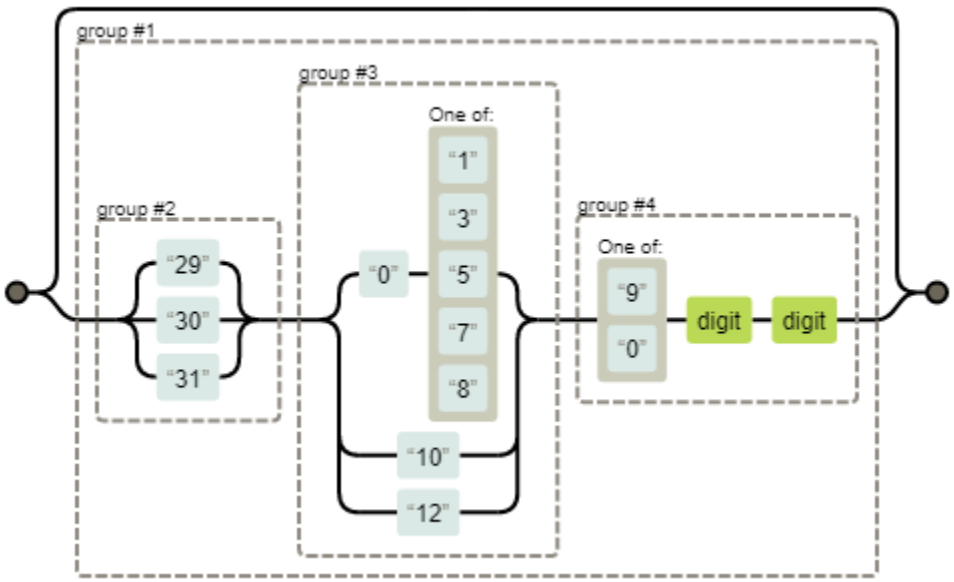
Celokupan izraz koji se nalazi gore, nije više pregledan, pa će zbog toga biti podeljen na celine koje će biti objašnjene.

((0[1-9]|1\d|2[0-8])(0[1-9]|1[0-2])([90]\d\d))



Prvi deo izraza treba da poklopi datume u svim mesecima koji su između 01-28. Nakon što pronalazi datum, kao i u prethodnom primeru pronalazi i mesec (01-12), a zatim i godinu (900-099). Princip je vrlo sličan primeru iznad, zato nećemo da se zadržavamo na njemu.

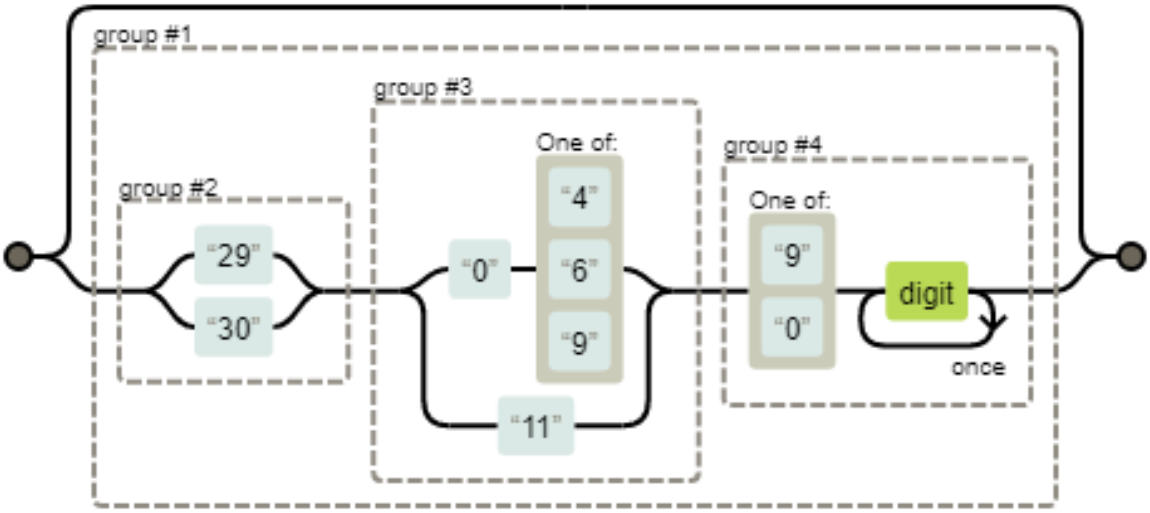
|((29|30|31)(0[13578]|10|12)([90]\d\d))



Nakon logičkog operatora |, kojim govorimo engine-u da ukoliko nije pronašao prvi deo regularnog izraza, nastavi na ovaj, u njemu pokušavamo da pronađemo mesece

koji imaju 31 dan. To su 01, 03, 05, 07, 08, 10, 12 mesec. Zato prvo pronalazimo dane 29, 30 i 31, zato što su svi ostali već poklopljeni prethodnim slučajem i proveravamo da li nakon ovih datuma sledi odgovarajući mesec. Ukoliko je odgovor pozitivan i nakon toga detektujemo i godinu kao i prošli put, između 900-099, onda smo uspešno pronašli mesec koji ima 29, 30 ili 31 dan. Ukoliko ne, prelazimo na sledeći deo izraza.

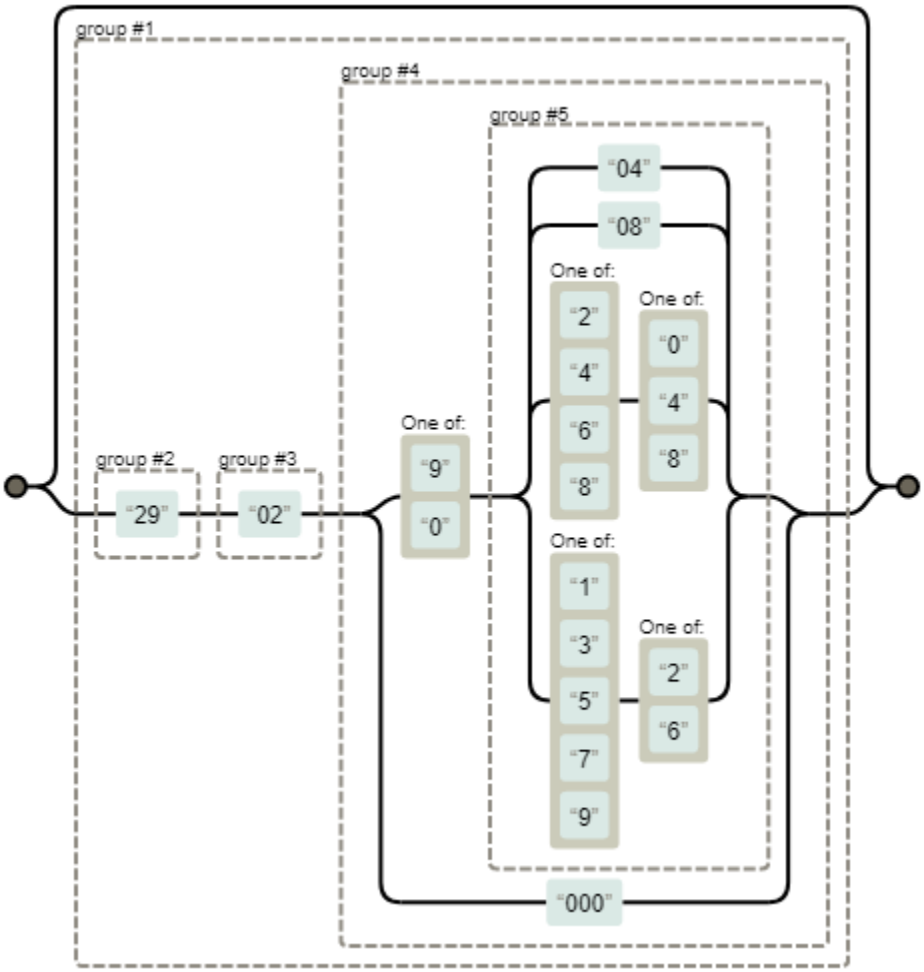
|((29|30)(0[469]|11)([90]\d{2}))



Ovaj deo izraza zadužen je da pronađe mesece koji imaju 29 ili 30 dana. To su 04, 06, 09, 11 mesec. Kod za obradu godine je isti kao i u prethodnom slučaju (uz izmenu da koristi tačno 2 cifre).

I konačno, dolazimo do najsloženijeg dela poklapanja. 29 februara. To ćemo da uradimo tako što ćemo da pronađemo 2902, ali ćemo nakon toga da tražimo i odgovarajuću godinu koja će da bude prestupna. Od veka takođe zavisi da li je godina prestupna (u gregorijanskom kalendaru, u julijanskom ne zavisi, ali nažalost ne koristimo njega u ovom slučaju), pa zbog toga ne možemo da napišemo izraz koji će da odredi prestupnu godinu korišćenjem 00 na kraju, zato što svaka 100-ta godina nije prestupna, osim ako nije 400-ta. Znači da u našem primeru, 000 hoće da bude prestupna, ali 900 neće, pa zato ćemo 000 da izdvojimo kao poseban primer.

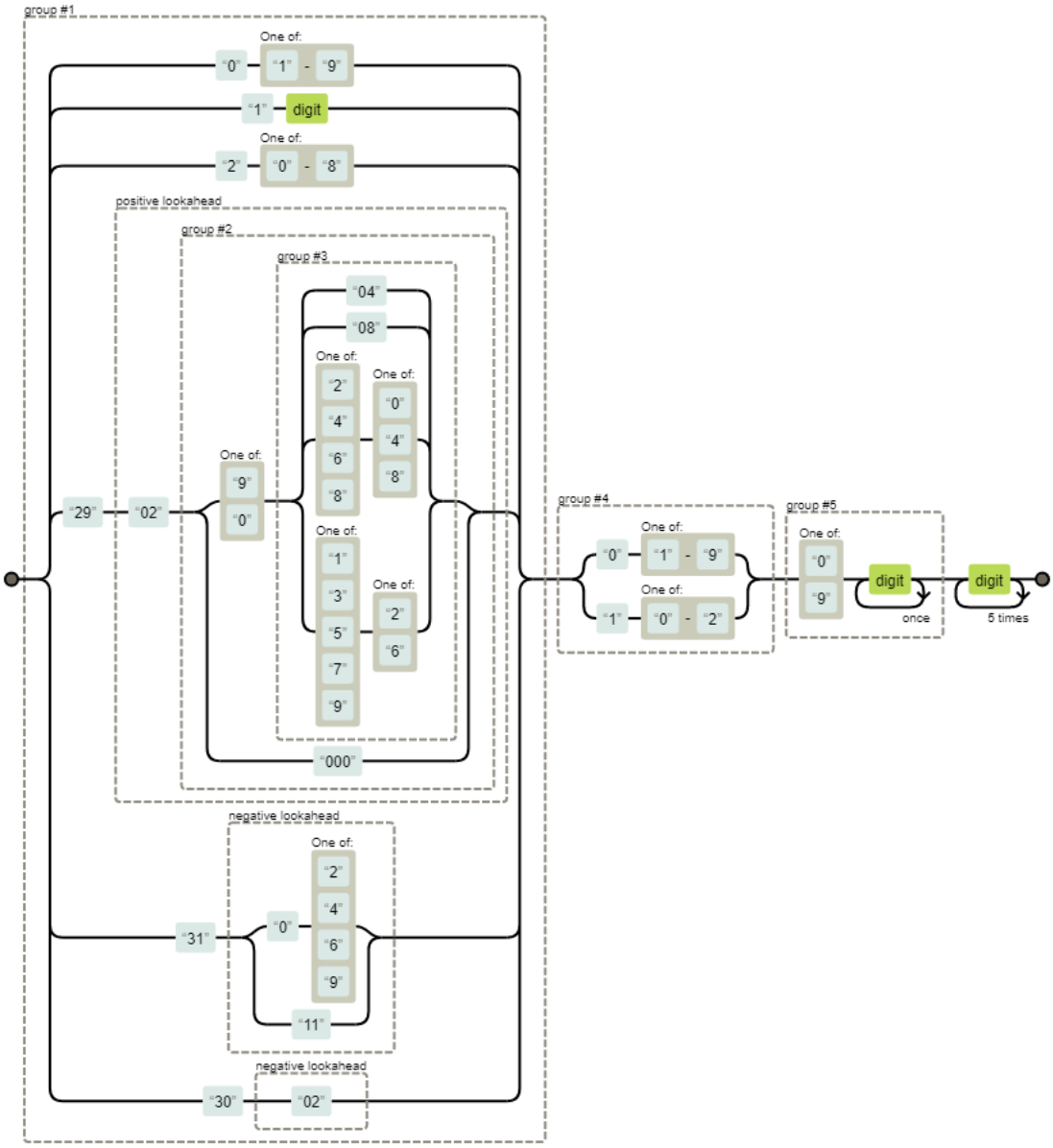
|((29)(02)([90](04|08|[2468][048]|[13579][26])|000))



Uz to objašnjenje, ostatak regularnog izraza bi trebalo da je poprilično jednostavan. Prestupne godine su one koje se završavaju sa 04, 08 ili su u parnoj deceniji i završavaju se sa 0, 4 ili 8, dok se one koje su u neparnoj deceniji završavaju sa 2 ili 6. Kao poseban primer smo izdvojili 000, zato što, kao što smo rekli ona je specijalan slučaj, zato što nije 400-ta godina. Na kraju naravno imamo i **\d{6}** koji treba da uhvati poslednjih 6 cifara matičnog broja.

Iako smo rešili problem na ovaj način, pogledajmo sada kako bi izgleda lookaround postupak.

(0[1-9]|1\d|2[0-8]|29(?=02([90](04|08|[2468][048]|[13579][26])|000))|31(?!(?:0[2469]|11))|30(?!02))(0[1-9]|1[0-2])([09]\d{2})\d{6}



U ovom primeru nećemo da delimo izraz na delove, već ćemo samo da primetimo da korišćenje positive lookahead i negative lookahead operatora drastično pojednostavljava regularni izraz. Kod je i smisleniji, zato što prvo, kao i u prethodnom slučaju obrađujemo primer svakog meseca koji ima do 28 dana, ali nakon toga, proveravamo 2902 na način što koristimo positive lookahead. To znači da pokušavamo da nakon 2902 pronađemo i godinu koja je prestupna, što radimo na isti način kao i u prethodnom primeru. 31. i 30. su takođe identični, razlika je jedino u tome što sada ne proveravamo koji meseci imaju 31 dan, nego koji nemaju (negative lookahead). Kada se uverimo da mesec nije na listi onih koji nemaju 31 dan, onda možemo da vratimo poklapanje. Slično, koristimo negative lookahead i

kod ostalih meseci (svi koji su preostali dolaze do dela koji proverava 30.) pa tada možemo da proverimo samo da mesec nije 02.

Problem matičnog broja ne može da se kompletno reši regularnim izrazom. Razlog za to je kontrolna cifra na kraju, koja se dobija aritmetikom između ostalih brojeva u matičnom broju. Regularni izrazi nisu namenjeni tome. Ali ukoliko zanemarimo tu činjenicu i ipak želimo da koristimo regularni izraz za validaciju, da li je matični broj uopšte ispravan, ta cifra svakako može da bude bilo koji broj od 0-9. Zato jedini deo koji još možemo da unapredimo su brojevi koji slede nakon datuma.

DDMMGGGRRBBBL

DD - datum - već smo obradili ovaj problem

MM - mesec - takođe smo obradili i ovaj problem

GGG - godina - kao i ovaj

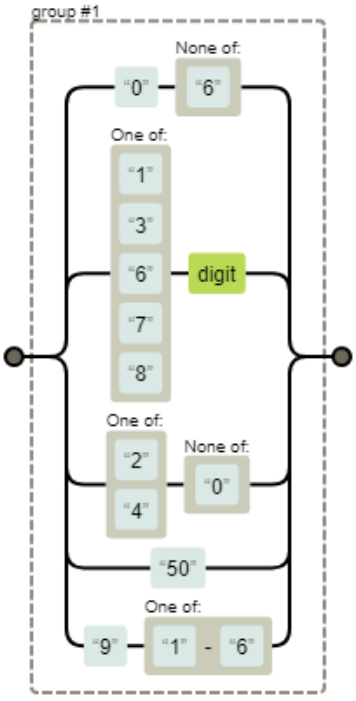
RR - region - moguće je ograničiti ga dodatno

BBB - 000-499 - muške osobe, 500-999 - ženske, znači svi brojevi su obuhvaćeni

L - kontrolna cifra, takođe bilo koji broj 0-9

Na sajtu https://sr.wikipedia.org/sr-ec/Јединствени_матични_број_грађанина možemo da pronađemo informacije o tome kako se RR cifre koriste (wikipedia nije baš najbolji izbor za referencu, ali u nedostatku bolje...)

Cifre 06, 20, 40, 5[1-9], 9[07-9] se ne koriste. Zato, da bi finalizirali ovaj regularni izraz, možemo da napišemo i jedan koji obuhvata ova pravila.



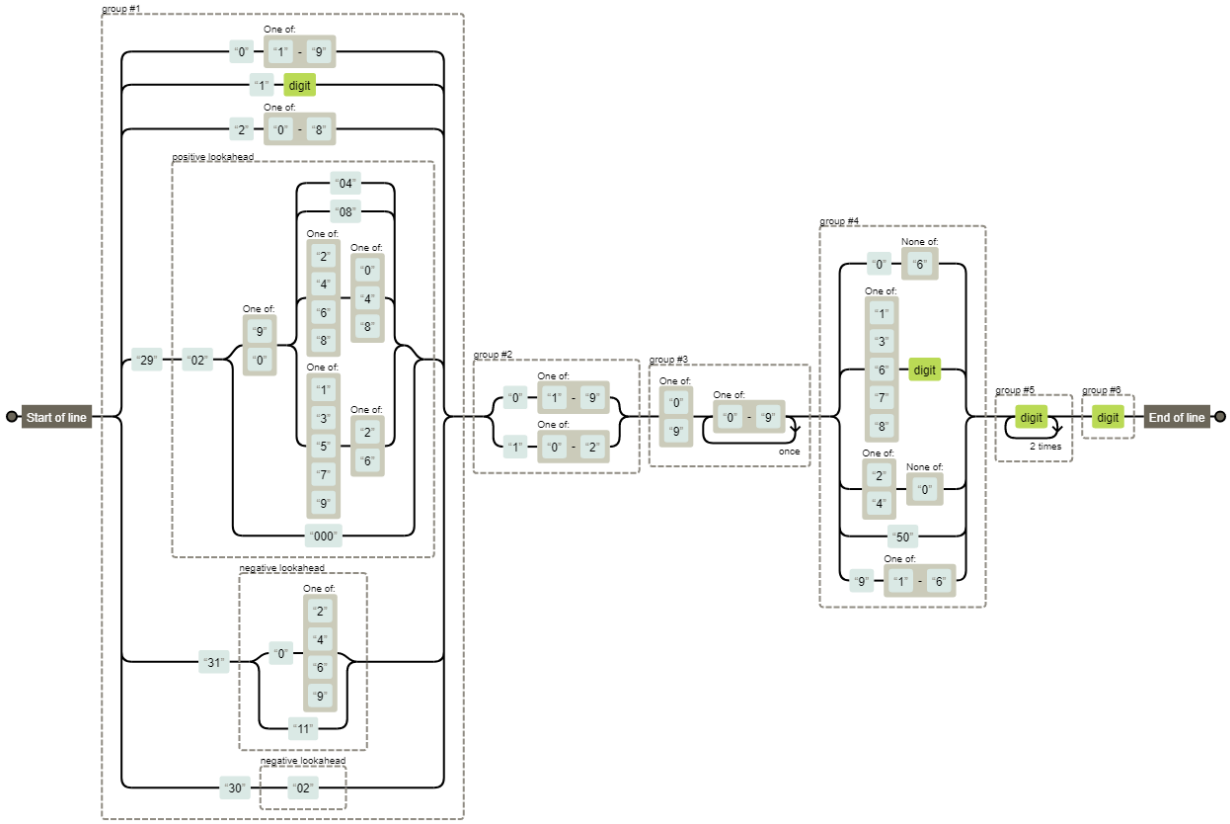
(0[^6]|[13678]\d|[24][^0]|50|9[1-6])

Ovaj izraz je jednostavan i objedinjuje logičkim operatorom nekoliko početnih cifara u kombinaciji sa nedozvoljenim (ili u nekim slučajevima dozvoljenim) brojevima koji mogu da slede.

Ekspertski pristup

Ukoliko sada kombinujemo ova dva rešenja, dobićemo validaciju koja proverava sve cifre osim poslednje. I dalje, bilo koja cifra na kraju je validna. Čak i cifra rednog broja rođenog u danu (BBB) ne može da bude sigurno validirana bez tabele svih rođenih u tom danu, tako da je kompletno pouzdan sistem jako teško napisati. Ali sa strane validacije, ovo rešenje je najbliže istini.

^(0[1-9]|1\d|2[0-8]|29(?=02(?:[90](?:04|08|[2468][048]|[13579][26])|000))|31(?!(?:0[2469]|11))|30(?!02))(0[1-9]|1[0-2])([09][0-9]{2})(0[^6]|[13678]\d|[24][^0]|50|9[1-6])(\d{3})(\d)$



Ekspertsko rešenje ima još neka poboljšanja u odnostu na prethodno. Koristi se i početak linije, kao i kraj linije (^ i **$**). Na taj način se obezbeđuje da će samo matični broj biti detektovan, a ne i druge cifre. Takođe koriste se i „non-capturing“ grupe, koje se ne pamte, pa ukoliko želimo, možemo da preuzmemo logičke celine redom iz uhvaćenih grupa: Grupa 1: DD, Grupa 2: MM, Grupa 3: GGG, Grupa 4: RR, Grupa 5: BBB, Grupa 6: L.

Korisni linkovi:

https://regexr.com/

https://www.rexegg.com/regex-quickstart.html

https://jex.im/regulex

https://regexper.com/

https://www.debuggex.com/

https://www.regular-expressions.info/

https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference

https://download.microsoft.com/download/D/2/4/D240EBF6-A9BA-4E4F-A63F-AEB6DA0B921C/Regular%20expressions%20quick%20reference.pdf

https://cheatography.com/davechild/cheat-sheets/regular-expressions/pdf/