

Regular Expressions

13. 10. 2014

Agenda

- Introduction
- Special Characters ...
- Sets
- Character Classes
- Simple examples
- Multipliers
- Number Quantifiers
- Subexpressions
- Sed and Regex
- Exercise
- Read more...
- Q/A

Introduction

- A regular expression is a pattern that either matches or doesn't match a given string or substring. When comparing this pattern against a string, it will return either true or false.
- Use and syntax of regex is the same across many Unix programs (vi, sed, awk etc.) and programming / scripting languages(Perl , Java, PHP etc.) & is supported in all major development environments.
- Uses:
 - Search for the existence of a pattern
 - Validate User Input data in web forms
 - Bulk Search and replace at ease.
 - String manipulation

Special Characters

- **Caret ^** Matches the beginning of lines.
- **\$** Matches the end of lines.
- **Period .** Matches any single character.
- ***** zero or more occurrences of the previous char
- **[chars]** - any one of the characters in chars
- **range [a-m]** - any one of the range of a-m chars.
- **pipe (|)** - either what comes before or after it.
- All regex are case sensitive unless told not to be so. – with the use of 'i'

Special Characters ...

<code>/a.c/</code>	Matches lines that contain strings such as <code>a+c</code> , <code>a-c</code> , <code>abc</code> , <code>match</code> , and <code>a3c</code> , whereas the pattern
<code>/a*c/</code>	Matches the same strings along with strings such as <code>ace</code> , <code>yacc</code> , and <code>arctic</code> .
<code>/[tT]he/</code>	Matches the string <code>The</code> and <code>the</code> :
<code>/^\$/</code>	Matches Blank lines
<code>/^.*\$/</code>	Matches an entire line whatever it is.
<code>/ */</code>	Matches one or more spaces
<code>[a-z]</code>	Matches a single lowercase letter
<code>[A-Z]</code>	Matches a single uppercase letter
<code>[a-zA-Z]</code>	Matches a single letter any case
<code>[0-9]</code>	Matches a single number
<code>[a-zA-Z0-9]</code>	Matches a single letter or number

Special Characters (...)

- A backslash (\) means escape the next character if it is a special one. Few e.g.
 - Match a question mark – “\?” ; Match a forward slash - “\” ; Match a backslash - “\\” ...
- If the character after the backslash is not a special one, then it may be an escape sequence. Few eg.
 - \l - Lowercase next character ; \n - newline character; \r - Return character ; \s - Character class for white space ; \S - Character class for non white space ; \t - Tab character ; ...

Sets

- A character set is a group of characters from which only one is desired.

[0123456789] – matches any single number

Sets can use ranges of characters

[0-9] – matches any single digit

A dash can be represented in a set by placing it first
(i.e. not in a range)

[-aeiou] – matches a dash or a vowel

A Caret (^) at the beginning of a set negates.

[^1-4] – matches any character which isn't 1,2,3 or 4

Character classes

- A character class lets you represent a bunch of characters as a single item
- Alpha :: Matches any letter, same as [A-Za-z].
- Upper :: Matches any upper-case letter; same as [A-Z].
- Lower :: Matches any lower-case letter; same as [a-z].
- Digit :: Matches any digit; same as [0-9].
- Alnum :: Matches any alphanumeric character; same as [A-Za-z0-9].
- Xdigit :: Matches any hexadecimal digit; same as [0-9A-Fa-f].
- Negated character class:: matches any character that is not in the class. e.g [^ab]

Simple Examples

- `art` :: Matches `art` in `'art'`, `'article'`, `'artifact'`, `'martial'`, `'cart'`, `'mart'`
- `^art` :: Matches `'article'` & `'artifact'`
- `art$` :: Matches `'cart'` & `'mart'`
- `^art$` :: Matches `'art'`
- `(jpg|png)` :: Matches `'jpg'` or `'png'`
- `([wx])([yz])` :: Matches `'wy'`, `'wz'`, `'xy'` or `'xz'`
- `([A-Z]{3}|[0-9]{4})` :: Matches three cap letters or 4 numbers

Simple examples

- `www.ibm.com`
 - Matches patterns like “www1ibmacom”, “wwwaibmscom” ...
- `"\d\d.\d\d.\d\d\d\d"`
 - Matches patterns like “01.01.2000”
- `"\w\w\w, \d\d \w\w\w \d\d\d\d"`
 - Matches patterns like “Wed, 21 Jul 2000”
- `".. \[[0-9]\]:"`
 - Matches patterns like SL [9]: , IQ [5]:
- `"[a-zA-Z]99"`
 - Matches patterns like s99, K99

Multipliers

- Any character or character class can be assigned a multiplier - say whether a character must exist, is optional, may exist for a certain minimum or maximum ...
- Plus (+) :: One or more
 - ✕ A^+ - A followed by any no. of additional A's
- Asterisk (*) :: anything
 - ✕ A^* - A followed by anything
- Question Mark (?) :: Zero or more occurrences
 - ✕ $A?$ - Either A or no As
- Curly Brackets({}) :: A specific range of occurrences
 - ✕ $A\{2,4\}$ - 2 As or more but no more than 4.
 - ✕ $[[\text{digit:}]]\{1,6\}$ - 1 number (0-9) or more, but no more than 6.

Number Quantifiers

- Specify number of occurrences, how many times previous character should occur.
- * * - 0 or more
- * + - 1 or more
- * ? - 0 or 1
- * {5} - Exactly 5 times
- * {5,} - 5 or more ; at least 5
- * {5,10} - from 5 to 10 times

SubExpressions

- A way of grouping characters - Reference the group at once. To group characters, place them within '()'.
(Name) = name ;; (Name)+ = name, namename

- A pipe within a subExp means either I grp or II (or more)
(Na|me) = Na or me ;; (Name|Date) = Name or date
- SubExp allow us to do back referencing: The ability to reference one or more groups directly. Use the backslash (\) followed by a number that specifies which subexp we want.

Example:

(name)\1 = namename

(Name|Date)\1 = namename or datedate

Sed and Regex

```
$ cat testing
```

```
root:x:0:0:root user:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

```
$ cat testing | sed '/daemon/d'
```

```
root:x:0:0:root user:/root:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

```
$ cat testing | sed '/sh/d'
```

```
sync:x:4:65534:sync:/bin:/bin/sync
```


Exercise – Try Out

- `/3.14159/` matches 3.14159, 3214159, 3=14159...
What 's the RE to match 3.14159 exactly ?
- `/TIFR*/` matches TIF, TIFR, TIFRRRR. Modify the RE to search for exact string 'TIFR'
- `[a-zA-Z]` matches any letter `[0-9]` matches any number.
What is the RE for matching SKS919 or this **exact** pattern of 3 letters followed by exactly 3 numbers

Read more...

➤ Books

- Mastering Regular expressions by Jeffrey E. F. Friedl (O'Reilly)
- Sams Teach Yourself Regular Expressions in 10 Minutes by Ben Forta
- Regular Expressions Cookbook by Jan Goyvaerts (O'Reilly)

➤ Web references

- <http://www.phpf1.com/tutorial/php-regular-expression.html>
- <http://weblogtoolscollection.com/regex/regex.php>
-lot many references

(Q/A) / Discussion