LinuxAcademy.com

Ruby 101

Intermediate Ruby:Regular Expressions

How can take a line of a log file and separate it out into different categories of data?

We could try separating the data out by:

spaces
quotation marks

But that gets pretty messy. Sometimes spaces don't separate data.

Regular Expressions:

Powerful pattern-matching sequences of code
Can match multiple patterns at once
Can subsititute

Can be tested live: rubular.com

Building blocks

\d	-> a digit
\D	-> a non-digit
\s	-> a whitespace
\S	-> a non-whitespace
\w	-> a word
\W	-> a non-word
\b	-> a word boundary
.	-> any character

Repeated Chars or Patterns:

?	-> 0 or 1 time
+	-> 0 or more times
*	-> 1 or more times
{#}	-> # of times
{#, }	-> # or more times
{,#}	-> 0 to # times
{#, #}	-> # to # of times

Specific possible Chars:

[xyz]	-> either x, y or z
[^xyz]	-> Any character besides x, y, or z
[a-zA-Z]	-> ranges of characters

Location of Chars

^	-> beginning of line
$	-> end of line
\A	-> beginning of string
\z	-> end of string

Capturing:

(…)	-> capture the pattern enclosed
(…|…)	-> capture either pattern enclosed

One line of an apache access log:

```
10.0.1.144 - - [04/Jan/2015:03:25:02 +0000] "GET /maintenance/index/timeout HTTP/1.1" 200 623 "-" "Wget/1.13.4 (linux-gnu)"
```

IP Address (IPv4)

```
(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})
```

Computer and User

```
(\S*) (\S*)
```

Date

```
\[(\d\d)\/([^\/]*)\/(\d{4}):(\d\d):(\d\d):(\d\d) [\+-]\d{4}\]
```

One line of an apache access log:

```
10.0.1.144 - - [04/Jan/2015:03:25:02 +0000] "GET /maintenance/index/timeout HTTP/1.1" 200 623 "-" "Wget/1.13.4 (linux-gnu)"
```

HTTP Action:

```
"([^"]*)"
```

Response Code and File Size:

```
(\S+) (\S+)
```

HTTP Referer and User Agent:

```
"([^"]*)" "([^"]*)"
```

One line of an apache access log:

```
10.0.1.144 - - [04/Jan/2015:03:25:02 +0000] "GET /maintenance/index/timeout HTTP/1.1" 200 623 "-" "Wget/1.13.4 (linux-gnu)"
```

Putting it all together:

```
(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) (\S*) (\S*) \[(\d\d)\/([^\/]*)\/(\d{4}):(\d\d):(\d\d):(\d\d) [\+-]\d{4}\] "([^"]*)" (\S+) (\S+) "([^"]*)" "([^"]*)"
```

Using a Regular Expression in Ruby:

/reguar_expression/ =~ *test_string* -> return location of 1st match
test_string =~ */regular_expression/*

/regular_expression/.match *string* -> return MatchData object
string.match */regular_expression/*

my_string.sub *regular_expression, replacement* ->Sub first occurence
my_string.gsub *regular_expression, replacement* -> All occurences
/regular_expression/options

options:

i	->	ignore case
m	->	'.' matches '\n' too
x	->	ignore whitespace and comments
o	->	only do interpolation #{} once

```
def parse_row row
	regex = /(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) (\S*) (\S*) \[(\d\d)\/([^\/]*)
\/(\d{4}):(\d\d):(\d\d):(\d\d) [\+-]\d{4}\] "([^"]*)" (\d+) (\d+) "([^"]*)"
"([^"]*)"/

	regex.match row
end
```