What are good regular expressions?

Asked 16 years, 4 months ago Modified 15 years, 3 months ago

Viewed 3k times

11

I have worked for 5 years mainly in java desktop applications accessing Oracle databases and I have never used regular expressions. Now I enter Stack Overflow and I see a lot of questions about them; I feel like I missed something.

For what do you use regular expressions?

P.S. sorry for my bad english

regex

Share

Improve this question

Follow

edited Oct 23, 2008 at 16:06

Adam Bellaire

110k ● 19 ● 152 ● 165

asked Aug 7, 2008 at 16:48

Telcontar

4,870 ● 7 ● 32 ● 41

Don't forget to read the Javadocs for java.util.regex.Pattern. It's a good reference. Also perldoc.perl.org/perlre.html
– Adrian Pronk Sep 17, 2009 at 9:21

9 Answers

Sorted by: Highest score (default)

7

Consider an example in Ruby:

```
puts "Matched!" unless /\d{3}-\d{4}/.match("555-
1234").nil?
puts "Didn't match!" if /\d{3}-\d{4}/.match("Not
phone number").nil?
```

The "/\d{3}-\d{4}/" is the regular expression, and as you can see it is a VERY concise way of finding a match in a string.

Furthermore, using groups you can extract information, as such:

```
match =
/([^@]*)@(.*)/.match("myaddress@domain.com")
name = match[1]
domain = match[2]
```

Here, the parenthesis in the regular expression mark a capturing group, so you can see exactly WHAT the data is that you matched, so you can do further processing.

This is just the tip of the iceberg... there are many many different things you can do in a regular expression that makes processing text REALLY easy.

Share Improve this answer Follow

answered Aug 7, 2008 at 17:02

Mike Stone
44.6k ● 30 ● 114 ● 140

7

Regular Expressions (or Regex) are used to pattern match in strings. You can thus pull out all email addresses from a piece of text because it follows a specific pattern.

In some cases regular expressions are enclosed in forward-slashes and after the second slash are placed options such as case-insensitivity. Here's a good one :)

```
/(bb|[^b]{2})/i
```

Spoken it can read "2 be or not 2 be".

The first part are the (brackets), they are split by the pipe | character which equates to an or statement so (a|b) matches "a" or "b". The first half of the piped area matches "bb". The second half's name I don't know but it's the square brackets, they match anything that is **not** "b", that's why there is a roof symbol thingie (technical term) there. The squiggly brackets match a count of the things before them, in this case two characters that are not "b".

After the second / is an "i" which makes it case insensitive. Use of the start and end slashes is environment specific, sometimes you do and sometimes you do not.

Two links that I think you will find handy for this are

1. regular-expressions.info

2. Wikipedia - Regular expression

Share Improve this answer Follow

edited Aug 7, 2008 at 17:02

answered Aug 7, 2008 at 16:51

Teifion
111k ● 76 ● 164 ● 196

It's a good description, but Mike's real-world example is preferable to the punning '2b' one. Would be nice to combine the two. – Bobby Jack Oct 13, 2008 at 10:19

`squiggly` brackets regarding `2` is not that common, they are `curly` .. – Timo Dec 11, 2017 at 8:03

6

Coolest regular expression *ever*:

```
/^1?$|^(11+?)\1+$/
```

It tests if a number is prime. And it works!!

N.B.: to make it work, a bit of set-up is needed; the number that we want to test has to be converted into a string of “`1`”s first, *then* we can apply the expression to test if the string does *not* contain a prime number of “`1`”s:

```
def is_prime(n)
  str = "1" * n
  return str !~ /^1?$|^(11+?)\1+$/
end
```

There's a detailled and very approachable explanation over at Avinash Meetoo's blog.

Share Improve this answer Follow

edited Sep 17, 2009 at 8:56

answered Aug 25, 2008 at 10:48
Konrad Rudolph
545k ● 138 ● 956 ● 1.2k

2 It's clever, but hardly appropriate for a beginner! :) – Bobby Jack Oct 13, 2008 at 10:23

@Copas: It absolutely works. Have you read the how-to and explanation that I've linked? – Konrad Rudolph Sep 17, 2009 at 8:45

2

If you want to learn about regular expressions, I recommend Mastering Regular Expressions. It goes all the way from the very basic concepts, all the way up to talking about how different engines work underneath. The last 4 chapters also gives a dedicated chapter to each of PHP, .Net, Perl, and Java. I learned a lot from it, and still use it as a reference.

Share Improve this answer Follow

answered Aug 19, 2008 at 0:00
Kibbee
66.1k ● 28 ● 144 ● 184

0

If you're just starting out with regular expressions, I heartily recommend a tool like The Regex Coach:

http://www.weitz.de/regex-coach/

also heard good things about RegexBuddy:

http://www.regexbuddy.com/

Share Improve this answer Follow

answered Aug 7, 2008 at 17:10

Marcel Levy
3,437 ● 1 ● 31 ● 40

0

As you may know, Oracle now has regular expressions: http://www.oracle.com/technology/oramag/webcolumns/2003/techarticles/rischert_regexp_pt1.html. I have used the new functionality in a few queries, but it hasn't been as useful as in other contexts. The reason, I believe, is that regular expressions are best suited for finding structured data buried within unstructured data.

For instance, I might use a regex to find Oracle messages that are stuffed in log file. It isn't possible to know where the messages are--only what they look like. So a regex is the best solution to that problem. When you work with a relational database, the data is usually pre-structured, so a regex doesn't shine in that context.

Share Improve this answer Follow

answered Aug 18, 2008 at 23:54

Jon Ericson
21.5k ● 12 ● 102 ● 151

0

> A regular expression (regex or regexp for short) is a special text string for describing a search pattern. You can think of regular expressions as wildcards on steroids. You are probably familiar with wildcard notations such as `*.txt` to find all text files in a file manager. The regex equivalent is `.*\.txt$`.

A great resource for regular expressions: http://www.regular-expressions.info

Share Improve this answer Follow

edited Aug 25, 2008 at 11:25
Thomas Owens
116k ● 99 ● 317 ● 436

answered Aug 7, 2008 at 16:50
Matt MacLean
19.6k ● 7 ● 51 ● 53

0

These RE's are specific to Visual Studio and C++ but I've found them helpful at times:

Find all occurrences of "routineName" with non-default params passed:

routineName\(:a+\)

Conversely to find all occurrences of "routineName" with only defaults: *routineName\(\)*

To find code enabled (or disabled) in a debug build:

\#*if.*_DEBUG*

Note that this will catch all the variants: ifdef, if defined, ifndef, if !defined

Share Improve this answer Follow

edited Aug 25, 2008 at 12:46

answered Aug 25, 2008 at 11:37

Onorio Catenacci
15.3k ● 16 ● 84 ● 134

0

Validating strong passwords:

This one will validate a password with a length of 5 to 10 alphanumerical characters, with at least one upper case, one lower case and one digit:

```
^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])[a-zA-Z0-9]
{5,10}$
```

Share Improve this answer Follow

answered Sep 17, 2009 at 9:00

Philippe Leybaert
172k ● 33 ● 212 ● 224