Regular expression to stop at first match [duplicate]

440

111

This question already has an answer here:
My regex is matching too much. How do I make it stop? *5 answers*

My regex pattern looks something like

```
<xxxx location="file path/level1/level2" xxxx some="xxx">
```

I am only interested in the part in quotes assigned to location. Shouldn't it be as easy as below without the greedy switch?

```
/.*location="(.*)".*/
```

Does not seem to work.

regex

edited Dec 1 '12 at 18:43

community wiki
3 revs, 2 users 75%
publicRavi

marked as duplicate by Wiktor Stribiżew regex Feb 18 at 9:45

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

What's your source, is it HTML or xml or something? – Oskar Kjellin Mar 23 '10 at 20:39

18 Why is this a community wiki? It's a real question. Too late now. – Ahmad Mageed Mar 23 '10 at 20:41

3 Not if all you want is to scan for simple attributes. Regex is appropriate and faster. – codenheim Mar 23 '10 at 20:44

6 Answers

921

You need to make your regular expression non-greedy, because by default, `"(.*)"` will match all of `"file path/level1/level2" xxx some="xxx" .`

Instead you can make your dot-star non-greedy, which will make it match as few characters as possible:

```
/location="(.*?)"/
```

Adding a `?` on a quantifier (`?` , `*` or `+`) makes it non-greedy.

answered Mar 23 '10 at 20:40 community wiki Daniel Vandersluis

28 FWIW, incase your using VIM, this regex needs to be a little different: instead of `.*?` it's `.\{-}` for a non-greedy match. – SooDesuNe Mar 24 '11 at 0:21

39 Thanks Daniel. **"Adding a ? on a quantifier (?, * or +) makes it non-greedy."** is helpful tip for me. – PhatHV Aug 20 '14 at 2:30

9 The ? describes my confusion in trying to figure this out. How appropriate. – Robbie Smith Apr 18 '16 at 17:38

I believe you can say 'lazy' instead of 'non-greedy' – Manticore Oct 19 '16 at 20:15

0

Because you are using *quantified subpattern* and as descried in Perl Doc,

> By default, a quantified subpattern is "**greedy**", that is, it will match as **many times** as possible (given a particular starting location) while still allowing the rest of the pattern to match. If you want it **to match the minimum number of times** possible, follow the quantifier **with** a "?" . Note that the meanings don't change, just the "greediness":

Thus, to allow your *quantified* pattern to make minimum match, follow it by `?` :

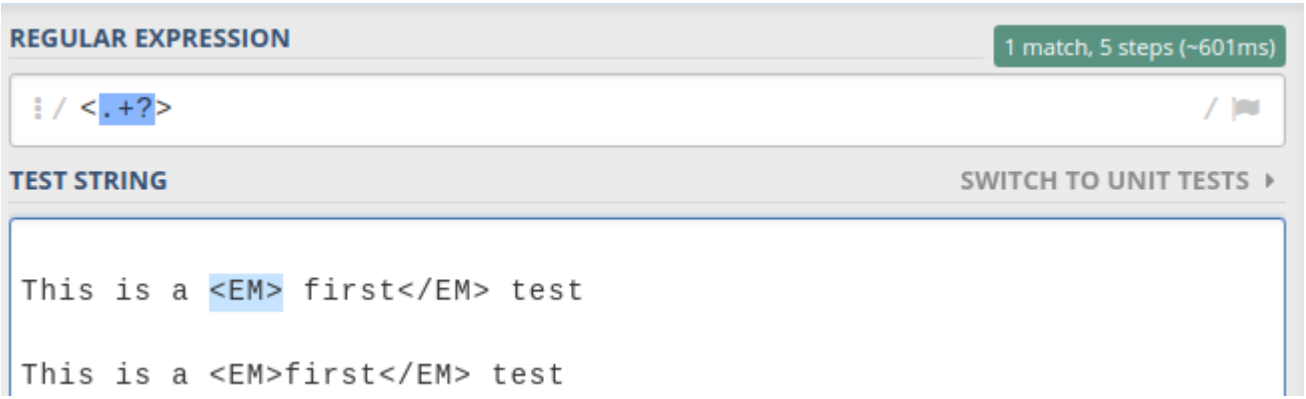
```
/location="(.*?)"/
```

answered Nov 20 '18 at 10:26 community wiki Mohammad Kanan

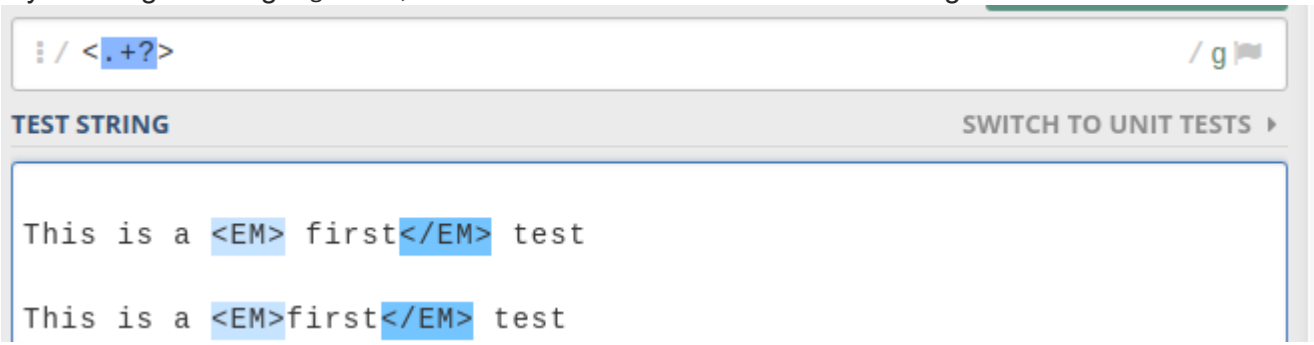
8

Use of Lazy quantifiers `?` with no global flag is the answer.

Eg,



If you had global flag `/g` then, it would have matched all the lowest length matches as below.



How about

29

```
.*location="([^"]*)".*
```

This avoids the unlimited search with .* and will match exactly to the first quote.

edited Feb 21 '13 at 8:55

community wiki
2 revs, 2 users 89%
user193690

Due to discrepancies in grep the above should be the preferred pattern if portability is a concern. – Josh Habdas Aug 13 '18 at 5:44

47

`location="(.*)"` will match from the " after `location=` until the " after `some="xxx` unless you make it non-greedy. So you either need `.*?` (i.e. make it non-greedy) or better replace `.*` with `[^"]*` .

answered Mar 23 '10 at 20:40

community wiki
sepp2k

7 +1, `[^"]*"` is clearer than `.*?"` any day – Kip Mar 23 '10 at 20:47

3 [^"]* is also probably faster with most regex engines because it does not need to lookup the pattern after the current pattern. – Jean Vincent Jul 21 '12 at 10:34

1 @Kip: You're probably right, but the `.*?` notation is more general than `[^"]*` – Bondax Sep 2 '15 at 7:45

how about if I want to include the delimiter character using [^"]* – Frohlich Nov 28 '16 at 11:46

not at all, if you don't know what ^ and [ ] mean here. Most people will understand .* – Vincent Gerris Jun 4 at 14:37

Use non-greedy matching, if your engine supports it. Add the ? inside the capture

answered Mar 23 '10 at 20:40

community wiki
codenheim