Ruby - Feature #6802

String#scan should have equivalent yielding MatchData

07/27/2012 04:17 PM - prijutme4ty (Ilya Vorontsov)

Status:	Assigned	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

Ruby should have method to obtain not an array of arrays but of MatchData objects. It can help in obtaining named groups:

pattern = /x: (?\d+) y:(?\d+)/
polygon = []
text.scan_for_pattern(pattern){|m| polygon << Point.new(m[:x], m[:y]) }

Not to break existing code we need unique name. Ideas? May be #each_match

Related issues:

Related to Ruby - Feature #5749: new method String#match_all needed	**Assigned**
Related to Ruby - Feature #5606: String#each_match(regexp)	**Feedback**
Related to Ruby - Feature #12745: String#(g)sub(!) should pass a MatchData to...	**Feedback**

History

#1 - 07/27/2012 04:30 PM - prijutme4ty (Ilya Vorontsov)

Simple implementation:

class String
def each_match(pattern, &block)
return Enumerator.new(self, :each_match, pattern) unless block_given?
text = self
m = text.match(pattern)
while m
yield m
text = text[m.end(0)..-1]
m = text.match(pattern)
end
end
end

#2 - 07/27/2012 06:54 PM - Eregon (Benoit Daloze)

=begin
You can use (({String#scan})) with the block form and (({$~})) (as well as other Regexp-related globals) for this:

```
> text="x:1 y:12 ; x:33 y:2"
> text.scan(/x:(?<x>\d+) y:(?<y>\d+)/) { p [$~[:x],$~[:y]] }
["1", "12"]
["33", "2"]
```

Please check your Regexp and give an example of (({text})) next time.
=end

#3 - 07/29/2012 08:13 PM - prijutme4ty (Ilya Vorontsov)

Thank you for a solution! I always forgot about regexp global vars. Though I suggest that using a special method here is more clear. So what'd you say about String#each_match and Regexp#each_match
Yes, implementation is as simple as
class String
def each_match(pat)
scan(pat){ yield $~ }
end
end

and similar for Regexp.

Eregon (Benoit Daloze) wrote:

=begin
You can use (({String#scan})) with the block form and (({$~})) (as well as other Regexp-related globals) for this:

```
> text="x:1 y:12 ; x:33 y:2"
> text.scan(/x:(?<x>\d+) y:(?<y>\d+)/) { p [$~[:x],$~[:y]] }
["1", "12"]
["33", "2"]
```

Please check your Regexp and give an example of (({text})) next time.
=end

#4 - 07/29/2012 10:28 PM - trans (Thomas Sawyer)

+1 I have definitely used this before (as Facets' #mscan).

#5 - 07/29/2012 11:52 PM - Eregon (Benoit Daloze)

prijutme4ty (Ilya Vorontsov) wrote:

> Though I suggest that using a special method here is more clear.
> So what'd you say about String#each_match and Regexp#each_match

I did indeed somewhat expected String#scan to yield a MatchData object, instead of $~.captures.
I'm in favor of String#each_match, it might be a nice addition and the name is clear, but the naming is different from the usual regexp methods on String, and it might not be worth to add a method (I agree $~ is not the prettiest thing around).

I think Regexp#each_match does not convey well what it does though.

#6 - 08/08/2012 12:51 AM - tomoakin (Tomoaki Nishiyama)

+1 to have a method to return MatchData.
This is related to (or duplicate of) #5749 and #5606.

Even with the simple implementation I think to establish a standard
name and specification.

#7 - 11/20/2012 11:31 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

#8 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#9 - 11/08/2018 07:31 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #12745: String#(g)sub(!) should pass a MatchData to the block, not a String added