Ruby trunk - Feature #12698

Method to delete a substring by regex match

08/24/2016 04:37 AM - sawa (Tsuyoshi Sawada)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

There is frequent need to delete a substring from a string. There already are methods String#delete and String#delete!, but their feature is a little bit different from the use cases that I am mentioning here.

I request methods that take a string or a regexp as an argument, and delete the matches from the receiver string. I am not sure of the method name, and I will use the term remove here. It can be named in some other better way. I request all combinations of global vs. local, and non-destructive vs. destructive. The expected feature is something like the following. First, the non-destructive ones:

```
"abcabc".remove("c") # => "ababc"
"abcabc".remove(/\zc/) # => "abcab"
"abcabc".gremove("c") # => "abab"
"abcabc".gremove(/c/) # => "abab"
```

Then, the destructive ones:

```
s = "abcabc"
s.remove!("c") # => "ababc"
s # => "ababc"

s = "abcabc"
s.gremove!("d") # => nil
s # => "abcabc"
```

Using this, cases like https://bugs.ruby-lang.org/issues/12694 would be just special cases. They can be handled like this:

```
"abcdef".remove(/\Aabc/) # => "def"
```

Related issues:

Related to Ruby trunk - Feature #13890: Allow a regexp as an argument to 'cou...	**Open**

History

#1 - 08/24/2016 04:42 AM - sawa (Tsuyoshi Sawada)

Perhaps my examples were not clear enough. My point is that, unlike String#delete, the given string argument should not be interpreted as character classes, but rather as a substring. So, the following should also hold:

```
"abc".remove("cba") # => "abc"
"abc".remove("abc") # => ""
```

#2 - 08/24/2016 01:28 PM - znz (Kazuhiro NISHIYAMA)

Active Support already has String#remove.

```
$ irb -r irb/completion --simple-prompt
>> require 'active_support/all'
=> true
>> "abcabc".remove("c")
=> "abab"
>> "abcabc".remove(/\zc/)
=> "abcabc"
>> "abcabc".remove(/c\z/)
=> "abcab"
>> "abcabc".remove(/\Aabc/)
=> "abc"
```

#3 - 08/27/2016 07:14 PM - shevegen (Robert A. Heiler)

I assume that this makes sense; my only concern is that ruby people may

be confused as to what to use. We have lots of ways :)

```
.gsub
.sub
.delete
.tr
[] =
```

Actually, I think my biggest complaint is that .delete() would sound so similar to .remove() - .delete(/regex_here/) might be nice too.

But anyway, to conclude, I concur with the threadstarter in principle, I think that ruby should know what to do when either a string is given as argument or a regex so I am in principle in favour of the suggestion.

#4 - 11/25/2016 09:08 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

I don't think it's worth adding which is easily done by sub/gsub.

Matz.

#5 - 03/08/2017 05:03 PM - uchagani (Umair Chagani)

Yukihiro Matsumoto wrote:

> I don't think it's worth adding which is easily done by sub/gsub.
>
> Matz.

I think the problem goes beyond than that. String#delete takes a string as a parameter and natural thought progression would indicate that this acts as a substring delete. If String#delete took an array then its current behavior would make more sense.

#6 - 11/13/2018 11:23 AM - svoop (Sven Schwyn)

matz (Yukihiro Matsumoto) wrote:

> I don't think it's worth adding which is easily done by sub/gsub.

Easily done, yes, but Ruby being a very expressive language, the following two are not equally readable:

- "foo test bar".gsub(/ test/, '')
- "foo test bar".delete(/ test/)

Personally, I've never used String#delete in almost 10 years of Ruby coding. It will be much more useful a method (and still downwards compatible) if it shifts from String#delete([other_str]+) to String#delete([pattern|other_str]+).

#7 - 11/19/2018 04:50 AM - duerst (Martin Dürst)

- Related to Feature #13890: Allow a regexp as an argument to 'count', to count more interesting things than single characters added

#8 - 11/19/2018 04:59 AM - duerst (Martin Dürst)

matz (Yukihiro Matsumoto) wrote:

> I don't think it's worth adding which is easily done by sub/gsub.

If put this way, it's easy to agree. But looking at it starting from String#delete, it feels annoying that String#delete doesn't accept a regular expression. If the above argument were taken to its full conclusion, it would mean that we could depreciate String#delete, because str.delete chars can be easily rewritten as str.gsub /[chars]/, ''.

String#count is a quite similar case, and even stronger, because in that case, it's not easy to replace it by sub/gsub. See feature #13890.

The only other method on String that has a list of characters as an argument is String#tr. For that, I don't see how to add a regular expression as a parameter.

#9 - 11/22/2018 09:00 AM - knu (Akinori MUSHA)

What about making the replacement of sub/gsub optional? (str.gsub(/re/, repl=''))
sub here might be considered as short for "subtract". □□

#10 - 04/25/2019 01:05 PM - svoop (Sven Schwyn)

The suggestion by knu (Akinori MUSHA) is pretty cool if you think about it: Since both sub and gsub exist, the uncertainty whether only one or all occurrences are deleted is gone. Also, with this in place, delete could be deprecated IMO (and removed on Ruby 3 which will most likely break existing code in other places as well).