Ruby trunk - Feature #12075

some container#nonempty?

02/16/2016 08:08 AM - naruse (Yui NARUSE)

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

Description

I sometimes write following code.

```
ary = some_metho_returns_nil_or_empty_container() # nil or "" or [] or {}

if ary && !ary.empty?
  #  some code
end
```

But the condition ary && !ary.empty? is too long and complex.
Though Ruby 2.3 introduces &., but this can't be written as ary&.empty?.

One idea is add nonempty? write as ary&.nonempty?.

akr: nonempty? is not good name because human is not good at handling

This discussion matches following core classes:

- String
- Array
- Hash

Related issues:

Related to Ruby trunk - Feature #13395: Add a method to check for not nil	**Open**

History

#1 - 02/16/2016 08:30 AM - mrkn (Kenta Murata)

How about ary.include_something? ?

#2 - 02/16/2016 08:31 AM - akr (Akira Tanaka)

How about "some?".

#3 - 02/16/2016 08:52 AM - sawa (Tsuyoshi Sawada)

That is a use case for Rails' blank? (or present?).

```
class Object
  def blank?
    respond_to?(:empty?) ? !!empty? : !self
  end
end

unless ary.blank?
  #  some code
end
```

What about incorporating these methods from Rails?

#4 - 02/16/2016 10:02 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

+1 for bringing blank? and present? to Ruby core. I often use (a || '').empty? checks for not having to depend on ActiveSupport directly in my Ruby code. I'd love to see them in Ruby core though.

#5 - 02/16/2016 02:17 PM - shyouhei (Shyouhei Urabe)

No, the OP wants to detect things that are not empty. This is not what blank? means. Also note that blank? in ActiveSupport has different (far more

Rails-centric) semantics than what is described in #3.

https://github.com/rails/rails/blob/b3eac823006eb6a346f88793aabef28a6d4f928c/activesupport/lib/active_support/core_ext/object/blank.rb#L115

#6 - 02/16/2016 03:28 PM - sawa (Tsuyoshi Sawada)

Shyouhei Urabe wrote:

> No, the OP wants to detect things that are not empty. This is not what blank? means.

Yes, I meant that blank? is the opposite of what OP wants. present? is what the OP wants. I replaced if in the original example with unless, so it is the same (Or, to retain if, present? can be used).

> ActiveSupport has different (far more Rails-centric) semantics than what is described in #3.

I see.

#7 - 02/16/2016 05:30 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

The implementation doesn't have to be the same as the one implemented by ActiveSupport. I think it would be fine to simply check for nil? and empty?. But I'd like to keep the names present? and blank? anyway.

#8 - 02/19/2016 11:10 PM - shevegen (Robert A. Heiler)

Perhaps the name .contains? might be good?

Although, .include? sort of is more or less synonymous with .contains? so perhaps this is not a good choice either.

.nonempty? seems a bit long, .non_empty? would be even longer :)

.empty? is a very good name already, I am not sure if "! .empty?" has a good name, though ruby uses the keyword "not" already. Could use .notempty? haha sorry, I have no good suggestion for a fitting name for negation either, but I am totally fine with the idea and functionality of the proposal itself, it's a good one.

#9 - 02/25/2016 02:47 AM - shyouhei (Shyouhei Urabe)

I have just learned that zsh(1) calls this concept being "full". http://zsh.sourceforge.net/Doc/Release/Expansion.html#Glob-Qualifiers

#10 - 02/25/2016 05:12 AM - sawa (Tsuyoshi Sawada)

What about introducing NilClass#empty?:

```
nil.empty? # => true
```

The code in question can be written simply as:

```
unless ary.empty?
  #  some code
end
```

If the original proposal is going to be realized, then a new method would have to be added to NilClass, String, Array, and Hash, but my proposal makes use of the existing method empty?, and needs to add to only NilClass, keeping the change minimal.

Even if the original proposal is going to be realized, extending empty? as above would make it the complete opposite of such method, and would introduce parallelism.

#11 - 02/25/2016 04:37 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I like this idea a lot, Tsuyoshi. I'm +1 for introducing nil.empty? as returning true.

#12 - 02/26/2016 04:33 AM - nobu (Nobuyoshi Nakada)

First, along this line, we'll need negative forms for all predicate methods.

And I think nil.empty? makes no sense.

Just an idea:

```
module Kernel
  def not(*a)
    not a.empty? ? self : __send__(*a)
  end
end
```

```
ary = nil;   ary&.not(:empty?) #=> nil
ary = [];    ary&.not(:empty?) #=> false
ary = [nil]; ary&.not(:empty?) #=> true
```

#13 - 02/26/2016 04:40 AM - nobu (Nobuyoshi Nakada)

Or

```
module Kernel
  def !(*a)
    a.empty? ? super() : !__send__(*a)
  end
end

ary = nil;   ary&.!:empty? #=> nil
ary = [];    ary&.!:empty? #=> false
ary = [nil]; ary&.!:empty? #=> true
```

#14 - 02/26/2016 05:22 AM - phluid61 (Matthew Kerwin)

Nobuyoshi Nakada wrote:

> First, along this line, we'll need negative forms for all predicate methods.
>
> And I think nil.empty? makes no sense.
>
> Just an idea:
>
> ```
> module Kernel
>   def not(*a)
>     not a.empty? ? self : __send__(*a)
>   end
> end
>
> ary = nil;   ary&.not(:empty?) #=> nil
> ary = [];    ary&.not(:empty?) #=> false
> ary = [nil]; ary&.not(:empty?) #=> true
> ```

I like this proposal. I definitely prefer the word 'not' over the symbol '!', because ary&.!:empty? has too much consecutive punctuation for my eyes.

Would there be value in extending it to accept a block?

```
module Kernel
  def not(*a, &b)
    not a.empty? ? self : __send__(*a, &b)
    # or even:
    #not a.empty? ? (b ? (yield self) : self) : __send__(*a, &b)
  end
end

ary = [];  ary&.not(:any?){|x|x>0} #=> true
ary = [1]; ary&.not(:any?){|x|x>0} #=> false
```

#15 - 02/26/2016 07:25 AM - nobu (Nobuyoshi Nakada)

https://github.com/ruby/ruby/compare/trunk...nobu:feature/12075-not

#16 - 02/26/2016 09:31 AM - Eregon (Benoit Daloze)

Nobuyoshi Nakada wrote:

> https://github.com/ruby/ruby/compare/trunk...nobu:feature/12075-not

I like it!

#17 - 05/17/2016 07:31 AM - nobu (Nobuyoshi Nakada)

- Description updated

#18 - 05/17/2016 09:26 AM - naruse (Yui NARUSE)

- Status changed from Assigned to Feedback

Array#any? seems to work usual use cases.
Feedback if another cases are discovered.

#19 - 09/26/2016 06:27 PM - sorah (Sorah Fukumori)

I know any? works on some use cases, but I'm positive to have a proposed method because using any? has a pitfall. We have to guarantee an array doesn't have only false or nil. Also I'm worrying users who started to use any? for this use case, but doesn't know this pitfall.

I'm positive on Object#not idea.

#20 - 04/12/2017 12:30 PM - nobu (Nobuyoshi Nakada)

- Related to Feature #13395: Add a method to check for not nil added