Array#select!

12/22/2009 11:52 PM - rogerdpack (Roger Pack)

Status: Closed
Priority: Normal
Assignee: 
Target version: 2.0.0

Description

=end

Though there is no enumerable equivalent, this method would be a good complement to the existing 

Array#select
Array#reject
Array#reject!

Much thanks.
-r
=end

History

#1 - 12/23/2009 12:19 AM - shyouhei (Shyouhei Urabe)

=begin
Not a bug at least... Moved to the Feature tracker.
=end

#2 - 02/28/2010 09:24 PM - mame (Yusuke Endoh)

- Status changed from Open to Rejected

=begin

Hi,

The feature was discussed in the thread from [ruby-core:27242].
In the thread, many English speaking people disagree this. So I close this ticket as “rejected”.

If you still want this feature, please reopen with "some real use of select!" [ruby-core:27323], or another convincing reason that beats English nuance.

--
Yusuke Endoh mame@tsg.ne.jp
=end

#3 - 03/02/2010 04:34 AM - rogerdpack (Roger Pack)

- Status changed from Rejected to Open

=begin
Hmmm.

I personally have had uses for select! in the past, as others have as well.
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/27316
http://www.ruby-forum.com/topic/200640#875371

Currently we are forced to use reject! which is surprising...
Think of select! as a destructive version of select, which currently doesn't exist in the std lib.

For example if you have this (albeit trivial) example :

a = some_large_array
b = a.select{|b| b%2 == 1}

c = a.select{|b| b%2 == 1}  # or 'c = a.select{|b| b%2 == 1}'}
a.select!{|b| b % 2 == 1}  
But it surprisingly doesn't exist (reject has its destructive counterpart, map does too, but select doesn’t), so you are forced to rewrite your boolean logic to use reject!

a.reject!{|b| (b%2) != 3}

which can be less readable, but more importantly--it is surprising to have to do so, at least to me.

It would be more convenient and less surprising to have a select!

Reopening.
Thanks.
-rp
=end

#4 - 03/02/2010 05:04 AM - hgs (Hugh Sasse)
=begin
On Tue, 2 Mar 2010, Roger Pack wrote:

   Issue #2515 has been updated by Roger Pack.
   [...]  
   Currently we are forced to use reject! which is surprising...  
   [...]  
   But it surprisingly doesn't exist (reject has its destructive
   [...]  
   which can be less readable, but more importantly--it is surprising to have to do so, at least to me.

   It would be more convenient and less surprising to have a select!

You can use the "surprising" argument 4 or more times if you like, but
since about 2005 Matz has said he will not accept changes
based on the Principle Of Least Surprise, because it is his surprise
that matters.
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/144077

However, if you can build the case based on orthogonality, you may
stand more of a chance. Possibly. I can't be certain.

Hugh
=end

#5 - 03/02/2010 12:22 PM - mame (Yusuke Endoh)
=begin
Hi, Roger

Hugh Sasse wrote:

   On Tue, 2 Mar 2010, Roger Pack wrote:
   
   It would be more convenient and less surprising to have a select!

   You can use the "surprising" argument 4 or more times if you like, but
since about 2005 Matz has said he will not accept changes
based on the Principle Of Least Surprise, because it is his surprise
that matters.

I agree with Hugh.

In Ruby design, POLS is very unconvincing reason currently. Instead,
it may make an enemy of committer.

# Anyone should know that many committers are bored with suggestion
# based on POLS.

In addition, consistency rarely beats name argument, AFAIK. I propose
you find another reason.

I leave this ticket open for a couple of days. But I'll close again
if discussion will seem to be still stalemated.

---

Yusuke ENDOH mame@tsg.ne.jp
=end

#6 - 03/02/2010 02:39 PM - marcandre (Marc-Andre Lafortune)
- Category set to core
- Target version set to 2.0.0

=begin
I would suggest that this feature request be extended to also introduce Array#keep_if.

Indeed, Enumerable#reject has two corresponding in-place Array methods: Array#delete_if and Array#reject!

I feel that Enumerable#select would benefit from also having two in-place Array methods: Array#keep_if and Array#select!

The same difference would apply: select! returns nil if no change is made.

I feel the name select! is as appropriate as map! , collect! and reject!
=end

#7 - 03/02/2010 06:54 PM - sdsykes (Stephen Sykes)

=begin
I agree with Marc-Andre, and also with his suggestion of keep_if.

I have sometimes wished for each of these methods, and it seems logical and reasonable that they should exist.

As for the names, I have reviewed the previous discussions, and I strongly suggest, as a native english speaker, that no-one will have any trouble understanding how select! or keep_if work.  They are good and understandable names.

The use case for select! is clear.  I quote from an example from the original discussion:

Code starts out as something like

result = collection.map { |x| ... }.select { |x| ... }

But then I might add code to the "map" part, and the code now extends over a few lines and looks poor if it's chained together. So it gets changed to:

result = collection.map { |x| ...
} result.select! { |x| ... }

+1 from me, a useful addition.
=end

#8 - 03/02/2010 07:45 PM - hgs (Hugh Sasse)

=begin
I got tripped up by redmine there: I thought that was going off list, but while the name of the addressee was there, the address was the redmine one, which I didn't notice.  I'd argue that is a misfeature.

I think the case for orthogonality is a much stronger one than POLS. select and reject are opposites, so having reject! and not select! is not orthogonal, to my mind.

http://www.artima.com/intv/dry3.html

Roger's argument, it seems to me, is essentially that to achieve an effect in one direction it is straightforward, but in the other you have to interact either with another method, or with assignment,

a = a.select(&block)
a.reject!(&complementary_block)

This is rather like the helicopter pilot in the above (artima.com) story, and there is a case for saying it makes the code less clear.  It is a circumlocution.
=end
I tend to use the assignment myself, though, because the \! signifier is a rather subtle thing, visually, for something destructive[1]. The assignment makes overwriting clearly apparent. However, that would be a case against almost all \! methods. So if destructive methods are idiomatic ruby, and if one may consider methods to be verbs, why make the language unnecessarily irregular, especially when all these reject, select, and map come from the same conceptual space of functional programming? [Of course, functional programming languages don't allow variables to change value, once bound...]

Is there a significant cost to adding this, and is it more than the cost of remembering the exceptions to the rules about which destructive methods exist? I've not looked at the source, yet, for this.

Hugh

[1] To this same vision that failed to notice my message wasn't off list. That's why I used to use (...) comments in Pascal, in preference to {...}, because the printers back then made the star into a big splat. So I'm not saying Matz was wrong to choose this, I'm saying my tastes differ.

On Tue, 2 Mar 2010, Yusuke Endoh wrote:

Issue #2515 has been updated by Yusuke Endoh.

Hi, Roger

Hugh Sasse wrote:

On Tue, 2 Mar 2010, Roger Pack wrote:

It would be more convenient and less surprising to have a select!

You can use the "surprising" argument 4 or more times if you like, but since about 2005 Matz has said he will not accept changes based on the Principle Of Least Surprise, because it is his surprise that matters.

I agree with Hugh.

In Ruby design, POLS is very unconvincing reason currently. Instead, it may make an enemy of committer.

Anyone should know that many committers are bored with suggestion based on POLS.

In addition, consistency rarely beats name argument, AFAIK. I propose you find another reason.

I leave this ticket open for a couple of days. But I'll close again if discussion will seem to be still stalemated.

--

Yusuke ENDOH mame@tsg.ne.jp

http://redmine.ruby-lang.org/issues/show/2515

http://redmine.ruby-lang.org

#9 - 03/03/2010 01:16 AM - mame (Yusuke Endoh)

=begin

Hi,

=end
I feel the name select! is as appropriate as map!, collect! and reject!

Some feel appropriate, and some inappropriate. Seems like "he said she said" scenario. Is there another reason than feeling and consistency (= orthogonality)?

I'm almost neutral for the suggestion, but take a opposite point of view for the discussion. If I ignore Array#reject! and consider only the name "select!", I can imagine the different behavior. The name seems to me to return an array whose elements meet condition, and removes the selected elements from the original array, like Array#delete_at and #shift:

```ruby
ary = [1, 2, 3, 4, 5]
p ary.select! {|x| x.even? } #=> [2, 4]
p ary #=> [1, 3, 5]
```

Yusuke ENDOH mame@tsg.ne.jp

---

**#10 - 03/03/2010 03:55 AM - rogerdpack (Roger Pack)**

=begin
Oh I see your argument now--that select! is too confusing of a name...

For me, though, it makes sense (given some background ruby knowledge), ex:

```ruby
a = [1,2,3]
a.select!{|b| b == 1}
a #=> [1]
```

This makes sense based on my knowledge of how reject and reject! work, map and map! work, etc. My thinking is that only more advanced Ruby programmers use ! methods, and they would probably already know that "! means a destructive change to self".

Is it too confusing?

Much thanks.

-rp
=end

**#11 - 03/03/2010 04:05 AM - hgs (Hugh Sasse)**

=begin
On Wed, 3 Mar 2010, Yusuke Endoh wrote:

Hi,

2010/3/2 Marc-Andre Lafortune redmine@ruby-lang.org:

I feel the name select! is as appropriate as map!, collect! and reject!

Some feel appropriate, and some inappropriate. Seems like "he said she said" scenario. Is there another reason than feeling and consistency (= orthogonality)?

I'm almost neutral for the suggestion, but take a opposite point of view for the discussion. If I ignore Array#reject! and consider only the name "select!", I can imagine the different behavior. The name seems to me to return an array whose elements meet condition, and removes the selected elements from the original array, like Array#delete_at and #shift:

```ruby
ary = [1, 2, 3, 4, 5]
p ary.select! {|x| x.even? } #=> [2, 4]
p ary #=> [1, 3, 5]
```

I've been using Reek a lot recently. This makes me think of the...
"code smell" called "Greedy Method" in which a method is trying to do more than one thing. I think it would be difficult to remember which way round the outputs are: "Does the receiver get what the block selected, or the return value get it?"

This would break my metaphor of methods as verbs, and ! as a verb ending, which I've found helpful.

I can see such a function could have its uses, but even Python's filter() doesn't work like this, even accounting for the lack of blocks.

I'd be inclined to call such a thing ifelse, and allow it to be applied to objects which don't iterate, as well. It seems to be related to unfold, because it takes one thing and returns two (i.e. a list),

Hugh

#12 - 03/03/2010 04:23 AM - coatl (caleb clausen)
=begin
I will also sign up as being in favor of adding this feature, perhaps with a different name, tho select! makes the most sense to me.
=end

#13 - 03/03/2010 04:24 AM - matz (Yukihiro Matsumoto)
=begin
Hi,

In message "Re: [ruby-core:28398] [Feature #2515] Array#select!" on Tue, 2 Mar 2010 14:39:39 +0900, Marc-Andre Lafortune redmine@ruby-lang.org writes:

|I feel the name select! is as appropriate as map! , collect! and reject! |

Okay, okay. I understand your feeling.

I would suggest that this feature request be extended to also introduce Array#keep_if.

The biggest reason (for me at least) was lack of counterpart of delete_if, so if keep_if could gain consensus, I'd agree with adding select! (and keep_if). I was thinking of delete_unless, but was not satisfied by the name.

matz.

=end

#14 - 03/03/2010 12:49 PM - mame (Yusuke Endoh)
=begin
Hi,

2010/3/3 Yukihiro Matsumoto matz@ruby-lang.org:

|I would suggest that this feature request be extended to also introduce Array#keep_if. |

The biggest reason (for me at least) was lack of counterpart of delete_if

It's a revelation... In [ruby-core:27254], you only said the reason was the inappropriate name.

Anyway, good, I'm happy since the ticket will be closed positively :-)

+1 for keep_if

--
Yusuke ENDOH mame@tsg.ne.jp

=end
#15 - 03/03/2010 02:37 PM - matz (Yukihiro Matsumoto)
- Status changed from Open to Closed
- % Done changed from 0 to 100

=begin
This issue was solved with changeset r26800.
Roger, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.
=end

#16 - 03/05/2010 04:08 AM - rogerdpack (Roger Pack)

=begin
Many thanks!
-rp
=end