### Ruby master - Feature #14132

**Module#attr{|_reader|_writer} should be public**

11/28/2017 02:40 AM - marcandre (Marc-Andre Lafortune)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>matz (Yukihiro Matsumoto)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

#### Description

Modules and classes can be reopened and changed (unless frozen).

This is used in many meta programming techniques.

Currently, attr_accessor, attr_writer and attr_reader are private, so we need to either do a class_eval, reopen the class somehow, or resort to :send

As I previously stated in #6539, I feel that the use of send should be reserved for incorrect usage of actually private methods that might change of interface or aren't meant to be called this way (e.g. respond_to_missing?)

Matz has stated before that "class/module operations should be done in the scope.". Nevertheless, common usage shows that there are many cases where Rubyists prefer using a single line for this, even if it means having to call send.

Here are 15k+ examples of send :attr_accessor in the wild:

https://github.com/search?utf8=%E2%9C%93&q=language%3Aruby+%22send+%3Aattr_accessor%22&type=Code

15k+ examples of send :attr_writer in the wild:

https://github.com/search?utf8=%E2%9C%93&q=language%3Aruby+%22send+%3Aattr_writer%22&type=Code

15k+ examples of send :attr_reader in the wild:

https://github.com/search?utf8=%E2%9C%93&q=language%3Aruby+%22send+%3Aattr_reader%22&type=Code


#### Related issues:

- Related to Ruby master - Feature #6539: public and private for core methods
- Related to Ruby master - Feature #14133: Module#{define|alias|undef}_method should be made public

#### Associated revisions

- Revision 60940 - 11/29/2017 05:47 PM - marcandre (Marc-Andre Lafortune)
  Make Module#attr{accessor|reader|writer} public [#14132]

- Revision 60940 - 11/29/2017 05:47 PM - marcandre (Marc-Andre Lafortune)
  Make Module#attr{accessor|reader|writer} public [#14132]

- Revision 60940 - 11/29/2017 05:47 PM - marcandre (Marc-Andre Lafortune)
  Make Module#attr{accessor|reader|writer} public [#14132]

#### History

- **#1 - 11/28/2017 02:41 AM - marcandre (Marc-Andre Lafortune)**
  - Related to Feature #6539: public and private for core methods added

- **#2 - 11/28/2017 03:26 AM - marcandre (Marc-Andre Lafortune)**
  - Related to Feature #14133: Module#{define|alias|undef}_method should be made public added

- **#3 - 11/28/2017 05:48 AM - shevegen (Robert A. Heiler)**
  
  [...] I feel that the use of send should be reserved for incorrect usage of actually private methods that might change of interface or aren't meant to be called this way (e.g. respond_to_missing?)
But you are here conveying an extra meaning onto .send().

I like .send() because it gives me a simple way to call whatever I want to, at any moment in time. So when you write "aren't meant to be called this way" then you are contradicting what .send() allows ruby hackers to do. I'd hate to lose that.

There is more than one way to do things in ruby and that includes .send().

Matz has stated before that "class/module operations should be done in the scope.". Nevertheless, common usage shows that there are many cases where Rubyists prefer using a single line for this, even if it means having to call send.

I have no problem at all whatsoever with .send(). The reason I use it is because it is so simple. It is also shorter than .public_send() which I do not use myself and do not need (as long as .send() is available).

By the way note that I am in no way against your suggestion at all. I am somewhat in support of your suggestion. I myself, however had, have largely stopped using the attr variants; not completely but largely. They are still useful to me to dynamically batch-define methods, but other than that for my own classes, I tend to go "the extra mile" and manually define methods; also because I tend to define a lot of "def foo?: @foo; end"-like methods, and as far as I know, there is no default attr. (I could specify my own attr but carrying this modification in distributed ruby code is rather cumbersome. I default to MRI usually when I distribute ruby code, it makes my life easier.)

I do, however had, also have to adm it that I never used:

..send attr_reader

and so forth. That looks really weird. But I think I can understand what you mean so I agree, this is really strange. Or at the least it looks strange to my eyes. I never used that so far. :)

By the way, I think you also have some false positives in the above search query such as:

alias :read_attribute_for_serialization :send
attr_reader :results

https://github.com/MrAaronOlsen/fair_bnb/blob/f1fe7c663bda085191df07836307170a668c9096/app/presenters/state_presenter.rb

People do write weird ruby code indeed ...

As for alternatives, I guess one can use one of the eval variants and define_method. But .send() has the edge here still - because it is so simple! :D

#4 - 11/29/2017 06:54 AM - matz (Yukihiro Matsumoto)

OK, I understand. They will be public in 2.5

Matz.

#5 - 11/29/2017 05:48 PM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Closed

Great, thank you :-)