

Ruby trunk - Feature #6470

Make attr_accessor return the list of generated method

05/20/2012 08:29 AM - rupert (Robert Pankoweki)

Status:	Open	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		
Description		
<p>attr_accessor currently returns nil. It would be more helpful if it return list of generated methods so that it can become an argument to other methods like :private or :protected. That way private accessors can still be defined at top of the class and be private without changing the visibility of next methods.</p>		
<pre>class Something private *attr_accessor :user, :action # IMHO This is nice # private attr_accessor :user, :action # <-- would be even better if :private method accepted ar rays def initialize(user, action) self.user = user self.action = action end def public_method user.do_something(action) end end</pre>		
VS		
<pre>class Something private; attr_accessor :user, :action; public # IMHO Hack!! def initialize(user, action) self.user = user self.action = action end def public_method user.do_something(action) end end</pre>		
VS		
<pre>class Something def initialize(user, action) self.user = user self.action = action end def public_method user.do_something(action) end private attr_accessor :user, :action # IMHO Does not look nice at bottom of the class definition end</pre>		
Related issues:		
Related to Ruby trunk - Feature #11539: Support explicit declaration of volat...		Open
Related to CommonRuby - Feature #11541: Let attr_accessor, _reader & _writer ...		Open

History

#1 - 05/20/2012 01:54 PM - henry.maddocks (Henry Maddocks)

Aren't accessors public by definition? If you want them to be private use attr.

#2 - 05/20/2012 04:16 PM - shevegen (Robert A. Heiler)

Yes, they are public.

If I understood them correctly, they are the same as this in pure ruby code:

```
attr_accessor :foo
```

```
def foo
  @foo
end
```

```
def foo=(i)
  @foo = i
end
```

I found that I personally only need attr_reader, attr_writer and attr_accessor. I have not found a use case for attr alone yet.

#3 - 05/21/2012 08:34 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiko Matsumoto)

This duplicates [#6198](#).

BTW, why don't you use instance variables directly? That is:

```
class Something
  def initialize(user, action)
    @user = user
    @action = action
  end

  def public_method
    @user.do_something(@action)
  end
end
```

--

Yusuke Endoh mame@tsg.ne.jp

#4 - 05/22/2012 04:05 AM - rupert (Robert Pankowceki)

I want to access my private fields also via methods instead of directly via instance variables so refactoring in future is easier. For example instead of finding in class all occurrences of @var = something and changing them into either "@var = something.strip" or extracting it into setter, I already use the private setter defined with attr_accessor everywhere. That way I only need to change the setter implementation in one place and don't need to look for code setting instance variable because there is non such, only calls for the accessor.

#5 - 11/20/2012 10:45 PM - mame (Yusuke Endoh)

- Target version set to 2.6

#6 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#7 - 01/10/2019 08:28 AM - matz (Yukihiko Matsumoto)

- Status changed from Assigned to Rejected

There's no use for private attr_reader, attr_writer, etc. And protected is not encouraged enough for new features. So I reject this.

Matz.

#8 - 01/10/2019 09:15 AM - Eregon (Benoit Daloze)

- Status changed from Rejected to Open

[matz \(Yukihiro Matsumoto\)](#) There are use cases, see <https://bugs.ruby-lang.org/issues/11539> and <https://bugs.ruby-lang.org/issues/11541>.

Also, one case which has been IIRC frequently requested (mentioned just above in this issue, <https://bugs.ruby-lang.org/issues/6470#note-4>) is:

```
attr_reader :foo
private attr_writer :foo
```

So one can use the symmetric `foo` and `foo=` in the class, but only the getter would be public.

This is also useful to evolve `foo=` (e.g., to invalidate some caches if set) and add extra logic in it, without having to change all places from `@foo = to self.foo =`.

I reopen because I think not all relevant issues have been considered.

In general, I support this feature as it is a general and composable extension which enables many more things such as for `def` (decorators, debugging, concurrency, etc).

#9 - 01/10/2019 09:16 AM - Eregon (Benoit Daloze)

- Related to Feature #11539: Support explicit declaration of volatile instance variables added

#10 - 01/10/2019 09:16 AM - Eregon (Benoit Daloze)

- Related to Feature #11541: Let `attr_accessor`, `_reader` & `_writer` return symbols of the defined methods added

#11 - 01/10/2019 11:26 AM - phluid61 (Matthew Kerwin)

In general I support this request, but in this proposed use-case ...

```
attr_reader :foo
private attr_writer :foo
```

So one can use the symmetric `foo` and `foo=` in the class, but only the getter would be public.

This is also useful to evolve `foo=` (e.g., to invalidate some caches if set) and add extra logic in it, without having to change all places from `@foo = to self.foo =`.

`foo=` without `@` or `self.` will assign a local variable. You'd have to change it to `foo=(...)` or `send :foo=, ...` anyway, no?

#12 - 01/10/2019 02:40 PM - rupert (Robert Pankowecki)

There's no use for `private attr_reader`, `attr_writer`, etc.

The intended usage is to ease future refactorings. If you always start with a method then later you can easily redefine just the method.

Initial code

```
class Something
  private attr_accessor :x, :y

  def something(a)
    self.x = a + y
  end
end
```

Code after refactoring:

```
class Something
  private attr_accessor :y
  private attr_reader :x

  def something(a)
    self.x = a + y
  end

  private

  def x=(new_value)
    @x_set_at = Time.now
  end
end
```

```
@x = new_value  
end
```

Notice that nothing setting @x had to be refactored because @x variable was always changed via the self.x= setter.

So when the time comes and cache expiration or additional logic needs to be added, makes it easy to just redefine the setter or getter with additional logic.

That's why I always prefer to use private accessors instead of instance variables. They are more flexible.

#13 - 02/07/2019 04:44 AM - hsbt (Hiroshi SHIBATA)

- Description updated