Ruby master - Feature #13560

Module#attr_ methods return reasonable values

05/12/2017 08:16 PM - dunrix (Damon Unrix)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Hi,

I'm suggesting all Module#attr_ methods should return array of generated method names, instead of useless nil.
Make them more concise across Module API, corresponding to logic of Module#define_method, allow pass as arguments to visibility public/protected/private methods etc.

Concerned method names:

- attr_accessor
- attr_reader
- attr_writer

Demonstration example:

```
class LookBeforeYouLeap
  api_meths = attr_accessor :ruby_api
  # api_meths == [:ruby_api, :ruby_api=]
  private *api_meths

  # Or you can pass generated methods directly
  # protected *attr_reader(:ruby_papi, :ruby_cext)
end

# Assignment to temporary local variable `api_meths` does not pollute class'es
# namespace.
```

I'm aware *attr_** methods also create corresponding instance variables, but Module API has no use for them, unlike created accessor methods.

In current state, where *attr_** methods return nil, it makes class definition more prone to errors, especially at code refactoring:

```
class ExistingWay
  attr_accessor :ruby_api  # result is nil

  # Need write method names manually
  private :ruby_api, :rby_api=  # throws NameError exception when executed

  # Any change of generated accessor methods names require manual rewrite
  # at each visibility method call also.
end
```

Related issues:

Related to CommonRuby - Feature #11541: Let attr_accessor, _reader & _writer ...	**Open**
Is duplicate of Ruby master - Feature #9453: Return symbols of defined method...	**Rejected**

History

#1 - 05/14/2017 04:50 AM - shyouhei (Shyouhei Urabe)

- Is duplicate of Feature #9453: Return symbols of defined methods for `attr` and friends added

#2 - 05/14/2017 06:12 PM - shevegen (Robert A. Heiler)

What would perhaps be nice would be to have a way to initialize these to nil, upon
attr_* definition, but I guess that would require another name - it would probably

be bad in general to change the attr_* default behaviour at this point.

> In current state, where attr_* methods return nil, it makes class definition more prone to errors, especially at code refactoring

I do not know whether it makes anything more prone to errors.

The interesting thing is that I myself only rarely use the attr_* methods these days. One reason is that I also want reader methods that have a trailing '?' (I love them!), but I do not want to carry my custom modifications into my gem-projects (I use them locally, but I don't want to require of people to go "off default ruby" - it is much better to stay with the idioms of main ruby in my opinion). But another reason also is that I have more control over methods (readers, setters, getters, accessors) with a custom definition.

It means that I have to write more code though, so sometimes I'd wish that there would be more ways to automatically define such accessor methods. The current attr_* ways are fine but I believe that active* (activerecord?) uses some custom modifications too, so I may assume that there could be a need for some more of these. I digress though, feel free to disregard my comment. :D

#3 - 05/27/2017 03:29 PM - dunrix (Damon Unrix)

> What would perhaps be nice would be to have a way to initialize these to nil, upon attr_* definition

Why nice ? This is definitely a purpose of initialize method. Belongs to object initialization, not class definition. In addition, there is no general rule for nil as a default value.

> do not know whether it makes anything more prone to errors.

If you rename an attribute, you shouldn't forget change an argument name passed to visibility method call. Suggested feature allows single point of change.

#4 - 08/19/2019 01:55 AM - znz (Kazuhiro NISHIYAMA)

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