Ruby master - Feature #5781

Query attributes (attribute methods ending in `?` mark)

12/20/2011 04:57 AM - trans (Thomas Sawyer)

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

Description

Pretty sure this has come up before, but I'd like to revisit b/c I don't understand why it isn't allowed.

Sometimes I define "query" attributes, and in those cases I'd like the reader method to end in a ? mark. Currently I have to do:

```
# @attribute
def foo?
  @foo
end
```

or, if I don't mind a shadowing bare method,

```
attr :foo
alias_method :foo?, :foo
```

So why not just allow:

```
attr :foo?
```

Currently this causes an error. But why? It just seems like a waste of potentially cleaner code.

Related issues:

Related to Ruby master - Feature #12046: Allow attr_reader :foo? to define in...	**Rejected**
Related to Ruby master - Feature #15991: Allow questionmarks in variable names	**Open**

History

#1 - 12/20/2011 09:52 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version changed from 1.9.4 to 2.0.0

#2 - 12/20/2011 11:03 PM - matz (Yukihiro Matsumoto)

It's mostly because semantics. attr :a creates a method corresponding to an instance variable @a. So naively, attr :a? tries to create an instance variables @a? which is not a valid name for a instance variable.

I don't want to allow instance variable names ending '?', just because ? is for predicates, not for variables.
The other option is removing '?' from instance variables. But as far as I remember no one seriously proposed the idea before, and we haven't got consensus.

Matz.

#3 - 12/20/2011 11:53 PM - javanthropus (Jeremy Bopp)

On 12/20/2011 08:03 AM, Yukihiro Matsumoto wrote:

> Issue #5781 has been updated by Yukihiro Matsumoto.
>
> It's mostly because semantics. attr :a creates a method corresponding to an instance variable @a. So naively, attr :a? tries to create an instance variables @a? which is not a valid name for a instance variable.
>
> I don't want to allow instance variable names ending '?', just because ? is for predicates, not for variables.
> The other option is removing '?' from instance variables. But as far as I remember no one seriously proposed the idea before, and we haven't got consensus.

How about adding an attr_query method that otherwise works identically
to attr_reader but creates a method with a '?' on the end of the given
name instead?

For example, you could use it like this:

```
class Foo
attr_query :bar
attr_writer :bar
end

foo = Foo.new
foo.bar = true
foo.bar?        # => true
foo.bar = false
foo.bar?        # => false
```

-Jeremy

#4 - 12/21/2011 01:23 AM - yeban (Anurag Priyam)

On Tue, Dec 20, 2011 at 8:17 PM, Jeremy Bopp jeremy@bopp.net wrote:

> How about adding an attr_query method that otherwise works identically
> to attr_reader but creates a method with a '?' on the end of the given
> name instead?

I would suggest the name predicate instead.

```
class Foo
attr :bar
predicate :bar
end
```

Alternatively,

On Tue, Dec 20, 2011 at 7:33 PM, Yukihiro Matsumoto matz@ruby-lang.org wrote:

> The other option is removing '?' from instance variables. Â But as far as I remember no one seriously proposed the idea before, and we haven't got consensus.

Use attr, but if the variable name passed to attr contains a '?'
define a predicate corresponding to it too.

```
class Foo
attr :bar?
attr :baz
end
```

f

#5 - 12/21/2011 03:51 AM - trans (Thomas Sawyer)

@jeremy Been down that road, and it's not as clean as you might expect. You end up needing two methods e.g. attr_query_reader and attr_query_accessor (attr_query_writer would be essentially meaningless). Moreover, adding additional attr methods tends to be one of those "Cambrian explosion" deals --there are vast variations people have devised. Check out Rails for examples. So I don't think it's a good precedence for core Ruby. In fact I've always thought it a bit unfortunate that #attr alone wasn't all we needed.

#6 - 12/21/2011 03:59 AM - trans (Thomas Sawyer)

matz (Yukihiro Matsumoto) In that case I would propose:

```
attr :foo?

#=> def foo?; @foo; end

attr_reader :foo?
#=>  def foo?; @foo; end

attr_writer :foo?
#=>  def foo=(x); @foo=(x); end

attr_accessor :foo?
```

#=> attr_reader :foo?; attr_writer :foo?

It's an open question as to whether #attr and/or #attr_reader should define the plan method too. Or if only true/false should be the return value. For the former, I do not think it matters much; maybe #attr just defines the "predicate" method and #attr_reader can define both? As to that later, I think it's better not to do boolean conversion. I believe Ara (or was it Austin?) made good arguments to that effect some years ago, reminding us that conditionals would function the same either way.

#7 - 12/21/2011 07:44 AM - Eregon (Benoit Daloze)

> So why not just allow: attr :foo?

I agree. I know many people wish for that too.

> matz: The other option is removing '?' from instance variables. But as far as I remember no one seriously proposed the idea before, and we haven't got consensus.

What do you mean by removing '?' from instance variables ?
As you said, '?' is already forbidden in instance variable names.

> Thomas Sawyer: It's an open question as to whether #attr and/or #attr_reader should define the plan method too.

I think the plain (bare) method should not be defined (to keep it as clean and simple as possible), and there should not be any conversion (which could lose information).

Also, since attr* :foo? does not conflict with current uses, I think it's fine to use the usual attr* methods.

#8 - 12/21/2011 08:19 AM - shevegen (Robert A. Heiler)

attr_query :foo

Would be nice to add a query method ending in foo? to return the instance variable @foo.

I agree that changing current behaviour of attr or attr_reader or attr_writer would be bad.

But a new method "attr_query :foo" would be nice to have. (I don't agree with "predicate :foo",
but I could agree to attr_predicate :foo - I think it would be nice and consistent to include
the attr_ part in such a method name.)

I also agree with matz on instance variables.

@foo? = "hi"

Does not read elegantly and thus should not be possible and not be changed.

But I also agree that there should be a very easy way to generate a
query method ending in "?".

It would make some code shorter.

#9 - 12/22/2011 04:27 AM - jballanc (Joshua Ballanco)

Perhaps one option to consider is to allow extra parameters specifying alternate names for the getters and setters (Obj-C does this for synthesized properties). Something like:

attr_accessor :foo, { :var => :bar, :getter => :is_barable? }, :baz

#10 - 10/27/2012 06:31 AM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.6

I changed target to next minor because no discussion on it.

#11 - 12/24/2012 03:43 AM - trans (Thomas Sawyer)

=begin
I revisited this b/c in one of my projects it is much needed. To compensate, I created a special extension called (({attr_switch})),

```
def attr_switch(name)
  attr_writer name
  module_eval %{
```

```
    def #{name}?
      @#{name}
    end
  }
end
```

But it has the problem that the (({#source_location})) for the method created is in attr_switch's definition and not where attr_switch is called. And in my case that is a problem. Does anyone know if there is a way to tell it the (({source_location})) should be at caller[0]? I tried adding that info to (({#module_eval})) call, i.e.

file, line = *caller[0].split(':')[0..1]
module_eval %{...}, file, line.to_i

But it didn't work.

So bringing this back to this feature request. I, for one, still would very much like this feature. Sometimes it's just much more convenient. And I'd much rather it just worked out-of-box then me having to fuss with creating a custom attr method (and as I point out above, I can't even get it to work exactly the same).

I took a look at the relevant C code, (({rb_attr()})) in (({vm_method.c})), but I simply do not understand that code enough to adjust it myself. If I did, I would have submitted a patch for this already.

Given what I understand about the new ((*process*)) for changing Ruby, I guess I need a sponsor from core team or a core member of another implementation. Is that correct? If so, is anyone willing to back this?
=end

#12 - 12/24/2012 11:10 AM - trans (Thomas Sawyer)

=begin
FYI,

file, line = *caller[0].split(':')[0..1]
module_eval %{...}, file, line.to_i

Actually this does work. My problem with #source_location stemmed from getting it from the (({attr_writer})) defined method. What I had to do was:

def attr_switch(name)
file, line = *caller[0].split(':')[0..1]
module_eval %{
def #{name}=(x)
@#{name}=x
end
def #{name}?
@#{name}
end
}, file, line.to_i
end

In any case, still would be better to have attr_accessor :x? work.
=end

#13 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#14 - 07/09/2019 12:08 AM - mame (Yusuke Endoh)

- Related to Feature #12046: Allow attr_reader :foo? to define instance method foo? for accessing @foo added

#15 - 07/13/2019 10:31 AM - Eregon (Benoit Daloze)

FWIW, a long time ago I tried to implement https://bugs.ruby-lang.org/issues/5781#note-7 in MRI but it turned out to be not so easy and I gave up.

#16 - 07/15/2019 09:42 AM - sudo (Sudo Nice)

How about implementing it similarly to Crystal?

```
attr_accessor? :foo
```

#17 - 08/28/2019 12:52 AM - mrkn (Kenta Murata)

- Related to Feature #15991: Allow questionmarks in variable names added

#18 - 01/10/2020 06:34 AM - anders (Anders Bälter)

sudo (Sudo Nice) wrote:

> How about implementing it similarly to Crystal?
>
> ```
> attr_accessor? :foo
> ```

+1