Ruby master - Feature #11689

Add methods allow us to get visibility from Method and UnboundMethod object.

11/15/2015 04:53 AM - yui-knk (Kaneko Yuichiro)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Add Method#visibility and UnboundMethod#visibility for getting visibility from Method and UnboundMethod object.
In GitHub https://github.com/ruby/ruby/pull/1098.

History

#1 - 11/15/2015 06:51 AM - hsbt (Hiroshi SHIBATA)

What's use-case for these methods?

At least, You should describe use-case with feature request.

#2 - 11/15/2015 10:08 AM - yui-knk (Kaneko Yuichiro)

Sorry.

These methods are useful for inspecting Method object (e.g. debugging or documenting).
Currently if we need information about visibility, we should check owner.public_instance_methods, owner.protected_instance_methods, and owner.private_instance_methods:
see also pry: https://github.com/pry/pry/blob/1f3f7e7ceff27ef516536849fc44fdf010f91c93/lib/pry/method.rb#L343

#3 - 10/22/2018 09:42 PM - MikeVastola (Mike Vastola)

Just wanted to second this FR as something I'd find useful.

Also, it would be nice to have associated predicate methods (i.e. Method#public?, Method#protected? and Method#private?)

#4 - 11/04/2018 11:56 PM - yui-knk (Kaneko Yuichiro)

Thanks for your comment.
Can you show me the use cases where predicate methods are useful?
I think these predicate methods can be implemented by using #visibility method, so these predicate methods should not be needed as core features.

#5 - 11/05/2018 12:40 AM - MikeVastola (Mike Vastola)

Oh, sorry. I missed where you were looking for an example before. Here's a good one from the popular activesupport gem: here.

As for the predicate methods, yes, they can be derrived from core methods and they -- like most core predicate methods, -- would be for convenience, but I disagree that that should be the determining factor. The same could be said for #nil?, #is_a?, #respond_to? and #*_defined? -- the functionality of which could be accomplished by checking the results of other core methods.

#6 - 11/05/2018 03:43 PM - shevegen (Robert A. Heiler)

(I think it may be easier to file a separate issue for the predicate methods,
such as Method#public?, Method#protected? or Method#private?).

Kaneko Yuichiro added the issue to the next upcoming developer meeting. Let's
see to the upcoming discussion of matz and the core developers and matz.

#7 - 11/22/2018 07:59 AM - matz (Yukihiro Matsumoto)

The proposal sounds nice, but I am not fully satisfied with the term **visibility**. So how about adding public?, private? and protected? methods instead?

Matz.

#8 - 11/22/2018 05:51 PM - MikeVastola (Mike Vastola)

I mean, as seen in yui-knk's example with pry, I think it's helpful to be able to essentially do send("#{method.visibility}_instance_methods"). I'm not picky on the name though if there is something better than visibility.

But I think it makes sense to address any/all methods that are introduced in this issue.

#9 - 11/24/2018 06:43 AM - mame (Yusuke Endoh)

I don't think that send("#{method.visibility}_instance_methods") would be a frequent, strongly-recommended idiom. You can do it more explicitly as follows.

```
def visibility(method)
  case
  when method.public?    then "public_instance_methods"
  when method.protected? then "protected_instance_methods"
  when method.private?   then "private_instance_methods"
  else raise "unknown method type"
  end
end
```

This code would require work if a new visibility is introduced. But, there is no plan to add a new visibility in foreseeable future. Also, if something is actually introduced, there is no guarantee that the idiom would work as is.

#10 - 11/24/2018 12:40 PM - Eregon (Benoit Daloze)

I think visibility is the perfect, accurate and unambiguous term for this (e.g., it's even used in the documentation of #private).
Returning the corresponding Symbol also seems very intuitive.

So take my vote as +1 for Method#visibility and -1 for 3 methods which seem very inconvenient to use.
Basically, it's 1 method versus 3 and it's strictly more expressive/powerful (the result can be displayed easily for introspection, and also compared to a known visibility).

#11 - 11/24/2018 01:10 PM - mame (Yusuke Endoh)

FYI: The reason why matz does not like the term "visibility", is because the method attribute is not a visibility. In fact, all methods are visible in Ruby. Instead, the method attribute restricts how and where it can be called. We briefly discussed another name candidate at the meeting, but we couldn't find a good name of the concept. Then matz chose the three methods (public?, etc.) because we can avoid deciding the name.

#12 - 11/24/2018 03:50 PM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote:

> FYI: The reason why matz does not like the term "visibility", is because the method attribute is not a visibility. In fact, all methods are visible in Ruby. Instead, the method attribute restricts how and where it can be called. We briefly discussed another name candidate at the meeting, but we couldn't find a good name of the concept. Then matz chose the three methods (public?, etc.) because we can avoid deciding the name.

Thanks for the information :)

Right, the visibility does not affect how to call the method reflectively (which is done with Method#call), but reflects the visibility in the context of normal calls.
To be honest, I expect very few people to be confused by this.
I think it's clear it means the definition time visibility.
Metaprogramming methods in general ignore visibility (or enforce public with #public_send), so a visibility for how to call the Method object wouldn't make sense anyway.

> Instead, the method attribute restricts how and where it can be called.
> We briefly discussed another name candidate at the meeting, but we couldn't find a good name of the concept.

Visibility is the standard term to talk about public/protected/private and restrictions of how and where a method can be called, in all languages I know.
I think no other term makes more sense than visibility, and it is a well-known concept in programming languages.
Maybe in other natural languages this is confusing? I think in English it's as clear as it can be for an established concept.

> In fact, all methods are visible in Ruby.

I'm not sure what you mean here. But one could say "method m in class A is not visible to instances of class B" and that would apply to Ruby if m is private.

#13 - 11/26/2018 04:42 AM - duerst (Martin Dürst)

I agree that #visibility is the best solution. When somebody mentioned this at the recent Ruby committers' meeting, I immediately thought "that's it". Benoit provides some more background.

There are arguments that this may be misunderstood, but so might the original keywords public, protected, and private, and many other Ruby keywords and names. Ruby users already have to learn that private doesn't really mean private, that in Ruby, there's always some metaprogamming workaround. Once they now it, they will apply this to #visibility without much problems.

#14 - 07/16/2020 04:31 PM - jacobevelyn (Jacob Evelyn)

I'd also like to advocate for this. We're working on a gem that allows you to easily memoize a method while still preserving that method's visibility, and

without Method#visibility we have to awkwardly hack around it:

```
method_visibility = if private_method_defined?(method_name)
                      :private
                    elsif protected_method_defined?(method_name)
                      :protected
                    else
                      :public
                    end
```

Files

0001-Add-Method-visibility-and-UnboundMethod-visibility.patch	3.11 KB	11/15/2015	yui-knk (Kaneko Yuichiro)