Ruby - Feature #12043

Add a method to NoMethodError that tells if private methods are callable at the time of

02/01/2016 12:22 PM - yuki24 (Yuki Nishijima)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

I've briefly talked about this to Sasada-san, but also wanted to hear from other committers. I would like to add a method to NoMethodError that tells whether or not private methods are callable from the line where the exception is raised. An example would be like this:

```
begin
  raies "Error" #
rescue NoMethodError => no_method_error
  no_method_error.private_method_callable? # => true
end
```

The only use case I can think of is the spell checker in the did_you_mean gem and I'm not actually sure how useful it would be for others.

Please let me know what you think, I'm open to suggestions.

Associated revisions

Revision 4d9f5482aea747e614a8d78b5e8ec114b023a768 - 02/28/2016 04:41 AM - nobu (Nobuyoshi Nakada)

NoMethodError#private_call?

- error.c (nometh_err_initialize): add private_call? parameter.
- error.c (nometh_err_private_call_p): add private_call? method,
  to tell if the exception raised in private form FCALL or VCALL.
  [Feature #12043]
- vm_eval.c (make_no_method_exception): append private_call?
  argument.
- vm_insnhelper.c (ci_missing_reason): copy FCALL flag.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@53961 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 4d9f5482 - 02/28/2016 04:41 AM - nobu (Nobuyoshi Nakada)

NoMethodError#private_call?

- error.c (nometh_err_initialize): add private_call? parameter.
- error.c (nometh_err_private_call_p): add private_call? method,
  to tell if the exception raised in private form FCALL or VCALL.
  [Feature #12043]
- vm_eval.c (make_no_method_exception): append private_call?
  argument.
- vm_insnhelper.c (ci_missing_reason): copy FCALL flag.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@53961 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 02/01/2016 01:16 PM - Eregon (Benoit Daloze)

Is respond_to?(no_method_error.name, false) not enough for this purpose?

#2 - 02/02/2016 01:54 AM - nobu (Nobuyoshi Nakada)

I think he wants to distinguish self.foo and foo().
Only public methods can be candidates in the former case, but also private methods can be in the latter case.

#3 - 02/02/2016 01:02 PM - yuki24 (Yuki Nishijima)

Nakada-san, that's exactly what I want. A code example would be something like this:

```
begin
  self.do_something
rescue NoMethodError => no_method_error
  no_method_error.private_method_callable?
# => false
end

begin
  do_something()
rescue NoMethodError => no_method_error
  no_method_error.private_method_callable?
# => true
end
```

In the first begin...end part, you call self. first, so you can't call private methods while you can in the second one.

#4 - 02/02/2016 01:14 PM - Eregon (Benoit Daloze)

So this would essentially be a way to tell if the call was a "fcall", without needing to parse the exception message?
In other terms, telling if this NoMethodError is a "private method called" error.
Maybe exc.private_method_error? or exc.private_method_called?

#5 - 02/02/2016 01:57 PM - yuki24 (Yuki Nishijima)

I think I should've been more specific and also should've mentioned that in the example above, the method you are trying to call doesn't actually exist.

```
begin
  self.method_that_does_not_exist
rescue NoMethodError => e
  e.message
# => undefined local variable or method `method_that_does_not_exist' ...

  e.private_method_callable?
# => false

end

begin
  method_that_does_not_exist()
rescue NoMethodError => e
  e.message
# => undefined local variable or method `method_that_does_not_exist' ...

  e.private_method_callable?
# => true

end
```

So, I guess what I'm proposing here is adding a way to tell if a "fcall" could be made.

#6 - 02/02/2016 02:09 PM - Eregon (Benoit Daloze)

The first error message is actually
"undefined method `method_that_does_not_exist' for main:Object".
But that does not help since anyway the error message below would be the same since it has "()" (and the same problem if it would have arguments)

"private_method_callable?" sounds like a specific private method would be callable.

Maybe e.private_call?
"Returns true if the call which generated the error was allowed to call private methods (because it had no receiver, was using self.assign= or used #send)"

#7 - 02/02/2016 02:17 PM - usa (Usaku NAKAMURA)

Does it satisfy your use case that raising another exception when private method call with self. ?

```
begin
  self.a_private_method
rescue NoMethodError => e
  e.is_a? PrivateMethodCallError #=> true
end

begin
  self.method_not_exist
rescue NoMethodError => e
```

```
  e.is_a? PrivateMethodCallError #=> false
end

begin
  method_not_exist
rescue NoMethodError => e
  e.is_a? PrivateMethodCallError #=> false
end
```

Of course, PrivateMethodCallError inherits from NoMethodError.

#8 - 02/02/2016 02:30 PM - yuki24 (Yuki Nishijima)

Benoit, you are absolutely right about the error message. I was a bad person and didn't really check after copying & pasting...
Regarding the method name, #private_call? sounds better to me than private_method_callable?.

Nakamura-san, sorry I wasn't clear enough in the first place. It would be great if you could read the further explanation I posted above.

#9 - 02/03/2016 07:40 AM - nobu (Nobuyoshi Nakada)

https://github.com/ruby/ruby/compare/trunk...nobu:feature/12043-NoMethodError%23private_call-p

#10 - 02/07/2016 12:47 PM - shevegen (Robert A. Heiler)

Good suggestion IMHO, +1

The did_you_mean gem is great. If distinguishing between "()" and no "" will
make things even greater then I am all for it. \o/

#11 - 02/19/2016 02:20 AM - nobu (Nobuyoshi Nakada)

This feature is **only** for "did_you_mean" gem, so I think that any name is OK, including implementation details.

private_call?
explicit_receiver?
fcall?
were_you_a_function?
demon_from_the_nose?
etc.

#12 - 02/26/2016 10:00 AM - Eregon (Benoit Daloze)

Nobuyoshi Nakada wrote:

> This feature is **only** for "did_you_mean" gem, so I think that any name is OK, including implementation details.
>
> private_call?
> explicit_receiver?
> fcall?
> were_you_a_function?
> demon_from_the_nose?
> etc.

It's still public API.

Let's choose private_call?, since I and the OP agree on this, unless somebody has an objection.

#13 - 02/28/2016 04:40 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r53961.

NoMethodError#private_call?

- error.c (nometh_err_initialize): add private_call? parameter.
- error.c (nometh_err_private_call_p): add private_call? method,
  to tell if the exception raised in private form FCALL or VCALL.
  [Feature #12043]
- vm_eval.c (make_no_method_exception): append private_call?
  argument.
- vm_insnhelper.c (ci_missing_reason): copy FCALL flag.