Ruby master - Feature #14382

Make public access of a private constant call const_missing

01/22/2018 03:50 AM - jeremyevans0 (Jeremy Evans)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

Calling obj.foo where foo is a private method of obj calls method_missing. You would expect klass::FOO where FOO is a private constant of klass to call const_missing, but currently it doesn't. This makes a small change so that const_missing will be called in such cases.

In addition to similarity to method_missing, the main reason for doing this is it offers a way to deprecate public constants. Currently, if you have a public constant and want to make it a private constant, you can't do it without breaking possible callers. With this patch, you can make it a private constant, then override const_missing in the class, and have const_missing print a deprecation warning and then return the value of the constant.

Associated revisions

Revision 63871 - 07/06/2018 01:56 PM - nobu (Nobuyoshi Nakada)

const_missing on private constants

- variable.c (rb_const_search): call #const_missing method on private constants, as well as uninitialized constants. [Feature #14328]

History

#1 - 01/22/2018 01:12 PM - Eregon (Benoit Daloze)

Isn't Module#deprecate_constant specifically for this usage?

#2 - 01/22/2018 02:41 PM - jeremyevans0 (Jeremy Evans)

Eregon (Benoit Daloze) wrote:

> Isn't Module#deprecate_constant specifically for this usage?

Module#deprecate_constant deprecates all usage of the constant, both public and private. If you want to keep a constant but change the visibility from public to private, it isn't appropriate.

#3 - 01/24/2018 05:35 AM - matz (Yukihiro Matsumoto)

The idea seems OK to me. Can you experiment?

Matz.

#4 - 01/24/2018 06:17 AM - jeremyevans0 (Jeremy Evans)

matz (Yukihiro Matsumoto) wrote:

> The idea seems OK to me. Can you experiment?

With the attached patch, both test-all and test-spec pass.

The only time I can see this causing an issue is in a class/module that overrides const_missing to do something, but also has private constants, and wants an exception raised instead of const_missing being called for public access to a private constant. That seems very unlikely to me. Usually if const_missing is used, it is designed to handle all constants to do something. For example, let's say you overrode const_missing to call a singleton method of the same name, so you could call singleton methods that start with a capital letter using klass::Method syntax without parentheses:

```
class Foo
  Bar = 1
  private_constant :Bar

  def self.const_missing(const)
    send(const)
  end
```

```
end
```

In this case, assume Foo::Bar constant is an implementation detail. It is probably desired that external use of Foo::Bar call Foo.Bar(), but currently this is not possible as an exception is raised before calling Foo.const_missing.

Unless const_missing is overridden, this will not change any behavior. I don't usually use const_missing in any of my applications/libraries, but assuming this is accepted I plan to use this in the deprecate_public gem (https://github.com/jeremyevans/ruby-deprecate_public) to add a Module#deprecate_public_constant method, which will issue a deprecation warning if a constant is accessed through the public interface (no deprecation warning if accessed through the private interface). This will allow libraries with public constants that are implementation details to deprecate public use of the constants in a way that alerts users of the libraries.

If you have specific ideas for experimentation, I will definitely try them.

#5 - 02/02/2018 02:36 PM - nobu (Nobuyoshi Nakada)

In the case it is accessed from an inherited class, the receiver class/module may differ from class/module which defines the constant as private.
So I think that the hook method will need another parameter.
And as a private constant is not accessible but not "missing", different name may be better.

#6 - 02/02/2018 03:09 PM - jeremyevans0 (Jeremy Evans)

nobu (Nobuyoshi Nakada) wrote:

> In the case it is accessed from an inherited class, the receiver class/module may differ from class/module which defines the constant as private.
> So I think that the hook method will need another parameter.

Can you please let me know which hook method needs another parameter? Also, can you provide example code showing the problem, as I'm not sure I understand?

> And as a private constant is not accessible but not "missing", different name may be better.

Are you referring to calling a method other than const_missing? Then it breaks the similarity with method_missing. Do you think we should also change ruby so that obj.foo where foo is a private method should call a method other than method_missing?

#7 - 06/06/2018 06:29 PM - jeremyevans0 (Jeremy Evans)

- File 0001-Make-public-access-of-a-private-constant-call-const_.patch added

nobu (Nobuyoshi Nakada) wrote:

> In the case it is accessed from an inherited class, the receiver class/module may differ from class/module which defines the constant as private.
> So I think that the hook method will need another parameter.

Attached is an updated patch against current trunk with an additional test showing use with nested constants, where the superclass calls private_constant and the subclass accesses the constant publicly. It shows that const_missing is still called on the subclass in that case, which I think is the most consistent behavior as that mirrors how method_missing works.

If this isn't the behavior you would expect in this case, can you please explain what behavior you expect?

#8 - 07/16/2018 05:17 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Files

0001-Make-public-access-of-a-private-constant-call-const_.patch	2.68 KB	01/22/2018	jeremyevans0 (Jeremy Evans)
0001-Make-public-access-of-a-private-constant-call-const_.patch	3.29 KB	06/06/2018	jeremyevans0 (Jeremy Evans)