Ruby master - Feature #13383

[PATCH] Module#source_location

03/29/2017 08:51 AM - koba789 (Hidekazu Kobayashi)

Status:	Open	
Priority:	Normal	
Assignee:	ioquatix (Samuel Williams)	
Target version:		

Description

Abstract

It can inspect where the module or class is defined.

Background

In debugging or development an application, I usually want to find out where the class definition of using library.
There is Method#source_location but I could not find **Class** easily.

Implementation

In Github: https://github.com/ruby/ruby/pull/1562

History

#1 - 03/29/2017 09:19 AM - nobu (Nobuyoshi Nakada)

Modules/classes can be opened again.
Your patch seems returning the last location instead of the first, is it intentional?

As for the implementation, it's not nice to keep iseq.
It would be better to save them in (hidden) instance variable(s).

#2 - 03/30/2017 04:33 AM - sorah (Sorah Fukumori)

it would be happier if we can get all of source locations where a class opened?

#3 - 03/30/2017 04:34 AM - sorah (Sorah Fukumori)

- Tracker changed from Bug to Feature

Turning this into a feature ticket.

#4 - 03/30/2017 09:26 AM - shyouhei (Shyouhei Urabe)

In modern Ruby, prior to actually requiring a library, its gemspec tends to be loaded. From what I understand, people require foo/version.rb from foo.gemspec, and this is the first time when namespace foo is opened.

Is this info that useful?

#5 - 03/30/2017 09:39 AM - Eregon (Benoit Daloze)

shyouhei (Shyouhei Urabe) wrote:

> In modern Ruby, prior to actually requiring a library, its gemspec tends to be loaded. From what I understand, people require foo/version.rb from foo.gemspec, and this is the first time when namespace foo is opened.
>
> Is this info that useful?

I think that's only the case of the development gemspec in the repository.
If installed as a gem, the gemspec is a generated file from the metadata which does not require any other file.
Example:

```
# -*- encoding: utf-8 -*-
# stub: concurrent-ruby 1.0.2 ruby lib
```

```
Gem::Specification.new do |s|
  s.name = "concurrent-ruby"
  s.version = "1.0.2"
```

I agree it would be more useful to be able to collect all source locations where a Module was opened.

#6 - 03/31/2017 06:03 AM - shyouhei (Shyouhei Urabe)

Eregon (Benoit Daloze) wrote:

> shyouhei (Shyouhei Urabe) wrote:
>
> > In modern Ruby, prior to actually requiring a library, its gemspec tends to be loaded. From what I understand, people require foo/version.rb from foo.gemspec, and this is the first time when namespace foo is opened.
> >
> > Is this info that useful?
>
> I think that's only the case of the development gemspec in the repository.
> If installed as a gem, the gemspec is a generated file from the metadata which does not require any other file.

Great. I didn't know this. Thank you.

> Example:
>
> ```
> # -*- encoding: utf-8 -*-
> # stub: concurrent-ruby 1.0.2 ruby lib
>
> Gem::Specification.new do |s|
>   s.name = "concurrent-ruby"
>   s.version = "1.0.2"
> ```
>
> I agree it would be more useful to be able to collect all source locations where a Module was opened.

Yes, I'm not against such feature.

#7 - 04/13/2017 07:10 PM - shevegen (Robert A. Heiler)

I am also for this if it is easily possible.

I remember many years ago in the ruby 1.8.x days, before pry,
I twiddled and played with the programming language Io.

I do not remember much about it, but one argument that it
said in its favour (aside from being small/lightweight)
was that it allowed introspection at run-time of pretty
much everything. (I do not remember if this was correct
or perhaps I may misremember.)

Since that, ruby got a lot more introspection support,
pry, also .source_location and being able to pull in
the method body and method comments (I think one of the
gems of pry does that).

I always found this awesome.

Anyway to come back to the suggestion by Hidekazu Kobayashi,
I agree. It would be nice to have full introspection of
everything in ruby at run-time. We got a lot more access
to all these various things these day, with the RubyVM and
objectspace showing memory even of a "namespace" like:

Like in my irb:

```
require 'objspace'; ObjectSpace.memsize_of([])      # => 20
require 'objspace'; ObjectSpace.memsize_of(Kernel) # => 2340
require 'objspace'; ObjectSpace.memsize_of(Object) # => 6484
```

I think that all of this is very cool. :)

#8 - 05/16/2017 11:19 AM - wanabe (_ wanabe)

sorah (Sorah Fukumori) wrote:

> it would be happier if we can get all of source locations where a class opened?

I hope Module#source_locations, too. Like below:

```
# Just a PoC
class Module
  attr_reader :source_locations
end

TracePoint.new(:class) do |tp|
  tp.self.module_eval do
    @source_locations ||= []
    @source_locations << [tp.path, tp.lineno]
  end
end.enable

class Foo
end
class Foo
end

p Foo.source_locations #=> [["a.rb", 13], ["a.rb", 15]]
```

#9 - 03/20/2018 05:30 PM - schneems (Richard Schneeman)

What is the status of this proposal? I think this feature would be very nice for debugging. If being able to show all the locations of where the class is defined is not possible, I think pointing to the last location would be the most useful for me.

#10 - 01/09/2020 03:59 AM - ioquatix (Samuel Williams)

- Assignee set to ioquatix (Samuel Williams)

I might have a go at this over the next month or two.

#11 - 01/09/2020 05:16 AM - sawa (Tsuyoshi Sawada)

We now have Module#const_source_location #10771, which tells us where the module was named as a constant (which is usually the same as where the module was created). If that fulfills the purpose of this issue, then this issue should be closed as duplicate.

If you mean by "open a module" to use the module Foo; ... end construction, then notice that a module can be modified without being opened in that sense. For example,

```
module Foo; end
...
Foo.define_method(:bar){}
```

Hence it might make sense and be useful to ask for a feature that tells us all locations where a module was modified rather than where a module was opened. But that would mean listing all lines within a module body where a destructive operation was called, which would result in a flood of lines.

I don't think such feature is that much useful. It would be much more fruitful to look up the source of particular constants or methods of that module.