Ruby master - Bug #7554
TracePoint#defined_class doesn't return Class or Module
12/13/2012 05:29 PM - ko1 (Koichi Sasada)

Status: Closed
Priority: Normal
Assignee: ko1 (Koichi Sasada)
Target version: 2.0.0

ruby -v:
ruby 2.0.0dev (2012-12-01 trunk 38127) [i386-mswin32_100]

Description
In some case (using singleton method), TracePoint#defined_class doesn't return Class or Module.

example

def self.foo
  end

  obj = Object.new

  def obj.foo
    end

  module M
    def baz
      end
    end

  class C
    include M
    def self.bar
      end
    end

  TracePoint.trace(:call){|tp| p [tp.defined_class, tp.defined_class.kind_of?(Module)]}

  foo
  obj.foo
  C.bar
  C.new.baz

  #=>

  ruby 2.0.0dev (2012-12-01 trunk 38127) [i386-mswin32_100]

  [main, false]
  [#Object:0x2d8267c, false]
  [C, true]
  [M, true]

  ###

  The name `defined_class' expected to return object of Class or Module.
  So it should be return Class or Module.

  Current code returns modified object by singleton class.
  I propose to return singleton class directly.

08/07/2021
The following patch fixes this issue.

--- vm_trace.c  (revision 38362)
+++ vm_trace.c  (working copy)
@@ -712,9 +712,6 @@
     if (RB_TYPE_P(trace_arg->klass, T_ICLASS)) {
         trace_arg->klass = RBASIC(trace_arg->klass)->klass;
     }
-  else if (FL_TEST(trace_arg->klass, FL_SINGLETON)) {
-      trace_arg->klass = rb_iv_get(trace_arg->klass, "attached");
-  } else {
-      trace_arg->klass = Qnil;
+
After that, that script shows:

###
[C: <Object:0x874b244>, true]
[C: <Object:0x873f9a8>, true]
[C: <Object:C>, true]
[M, true]
###

The current behavior is from `set_trace_func` (6th parameter of block).
I'm not sure why it returns modified object instead of singleton class.

I believe TracePoint#defined_class should return singleton class directly.
TracePoint is introduced from 2.0, so no compatibility issue.

Associated revisions
Revision 6247099f - 12/17/2012 08:28 PM - ko1 (Koichi Sasada)

- vm_trace.c (fill_id_and_klass): TracePoint#defined_class returns singleton class. set_trace_func' passed attached class (which is
  attached/modified by singleton class) by 6th block parameter if it is singleton class. Previous behavior follows this spec. However, this method
  nameddefined_class' should return singleton class directly because singleton methods are defined in singleton class. There are no compatible
  issue because TracePoint is introduced after 2.0. But compatibility with set_trace_func' is broken. This means that you can not replace
  allset_trace_func' code with TracePoint without consideration of this behavior. [Bug #7554]
- test/ruby/test_settracefunc.rb: change a test to catch up an above change.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@38430 b2dd03c3-39d4-4d8f-98ff-823fe69b080e

Revision 38430 - 12/17/2012 08:28 PM - ko1 (Koichi Sasada)

- vm_trace.c (fill_id_and_klass): TracePoint#defined_class returns singleton class. set_trace_func' passed attached class (which is
  attached/modified by singleton class) by 6th block parameter if it is singleton class. Previous behavior follows this spec. However, this method
  nameddefined_class' should return singleton class directly because singleton methods are defined in singleton class. There are no compatible
  issue because TracePoint is introduced after 2.0. But compatibility with set_trace_func' is broken. This means that you can not replace
  allset_trace_func' code with TracePoint without consideration of this behavior. [Bug #7554]
- test/ruby/test_settracefunc.rb: change a test to catch up an above change.

Revision 38430 - 12/17/2012 08:28 PM - ko1 (Koichi Sasada)

- vm_trace.c (fill_id_and_klass): TracePoint#defined_class returns singleton class. set_trace_func' passed attached class (which is
  attached/modified by singleton class) by 6th block parameter if it is singleton class. Previous behavior follows this spec. However, this method
  nameddefined_class' should return singleton class directly because singleton methods are defined in singleton class. There are no compatible
  issue because TracePoint is introduced after 2.0. But compatibility with set_trace_func' is broken. This means that you can not replace
  allset_trace_func' code with TracePoint without consideration of this behavior. [Bug #7554]
- test/ruby/test_settracefunc.rb: change a test to catch up an above change.

Revision 38430 - 12/17/2012 08:28 PM - ko1 (Koichi Sasada)

- vm_trace.c (fill_id_and_klass): TracePoint#defined_class returns singleton class. set_trace_func' passed attached class (which is
  attached/modified by singleton class) by 6th block parameter if it is singleton class. Previous behavior follows this spec. However, this method


named\texttt{defined\_class}' should return singleton class directly because singleton methods are defined in singleton class. There are no compatible issue because \texttt{TracePoint} is introduced after 2.0. But compatibility with \texttt{set\_trace\_func}' is broke. This means that you can not replace \texttt{allset\_trace\_func}' code with \texttt{TracePoint} without consideration of this behavior. [Bug #7554]

- \texttt{test/ruby/test\_settracefunc.rb}: change a test to catch up an above change.

Revision 38430 - 12/17/2012 08:28 PM - ko1 (Koichi Sasada)

- \texttt{vm\_trace.c (fill\_id\_and\_klass)}: \texttt{TracePoint\#defined\_class} returns singleton class. \texttt{set\_trace\_func} passed attached class (which is attached/modified by singleton class) by 6th block parameter if it is singleton class. Previous behavior follows this spec. However, this method \texttt{named\texttt{defined\_class}} should return singleton class directly because singleton methods are defined in singleton class. There are no compatible issue because \texttt{TracePoint} is introduced after 2.0. But compatibility with \texttt{set\_trace\_func}' is broke. This means that you can not replace \texttt{allset\_trace\_func}' code with \texttt{TracePoint} without consideration of this behavior. [Bug #7554]

- \texttt{test/ruby/test\_settracefunc.rb}: change a test to catch up an above change.

Revision 38430 - 12/17/2012 08:28 PM - ko1 (Koichi Sasada)

- \texttt{vm\_trace.c (fill\_id\_and\_klass)}: \texttt{TracePoint\#defined\_class} returns singleton class. \texttt{set\_trace\_func} passed attached class (which is attached/modified by singleton class) by 6th block parameter if it is singleton class. Previous behavior follows this spec. However, this method \texttt{named\texttt{defined\_class}} should return singleton class directly because singleton methods are defined in singleton class. There are no compatible issue because \texttt{TracePoint} is introduced after 2.0. But compatibility with \texttt{set\_trace\_func}' is broke. This means that you can not replace \texttt{allset\_trace\_func}' code with \texttt{TracePoint} without consideration of this behavior. [Bug #7554]

- \texttt{test/ruby/test\_settracefunc.rb}: change a test to catch up an above change.

Revision 38430 - 12/17/2012 08:28 PM - ko1 (Koichi Sasada)

- \texttt{vm\_trace.c (fill\_id\_and\_klass)}: \texttt{TracePoint\#defined\_class} returns singleton class. \texttt{set\_trace\_func} passed attached class (which is attached/modified by singleton class) by 6th block parameter if it is singleton class. Previous behavior follows this spec. However, this method \texttt{named\texttt{defined\_class}} should return singleton class directly because singleton methods are defined in singleton class. There are no compatible issue because \texttt{TracePoint} is introduced after 2.0. But compatibility with \texttt{set\_trace\_func}' is broke. This means that you can not replace \texttt{allset\_trace\_func}' code with \texttt{TracePoint} without consideration of this behavior. [Bug #7554]

- \texttt{test/ruby/test\_settracefunc.rb}: change a test to catch up an above change.

History

#1 - 12/13/2012 05:34 PM - ko1 (Koichi Sasada)
- Description updated

#2 - 12/18/2012 05:28 AM - ko1 (Koichi Sasada)
- Status changed from Open to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r38430.
Koichi, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.