Ruby master - Bug #9005

object.send(:define_method, ...){...} creates private method

10/09/2013 07:40 AM - jeremyevans0 (Jeremy Evans)

Status:	Closed		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:	2.1.0		
ruby -v:	ruby 2.1.0dev (2013-09-22 trunk 43011) [i386-openbsd]	**Backport:**	1.9.3: DONTNEED, 2.0.0: DONTNEED, 2.1.0: REQUIRED

Description

I assume this is caused by r40022, which made define_method consider visibility. However, visibility should only be considered if define_method is called normally, not via send. When called via send, it should define a public method. Here's example code showing the error:

$ ruby21 -ve "Object.send(:define_method, :foo){|*a| 1}.foo"
-e:1:in <main>': private methodfoo' called for :foo:Symbol (NoMethodError)

I apologize in advance if this has already been fixed (I tested 2.1.0-preview1, not trunk), but from the commit logs it doesn't appear to have been.

Related issues:

Related to Backport21 - Backport #9296: please backport r44380 (visibility of...	**Closed**	**12/25/2013**
Has duplicate Ruby master - Bug #9141: define_singleton_method creates privat...	**Closed**	**11/22/2013**

Associated revisions

Revision 1fc33199 - 12/24/2013 07:28 AM - nobu (Nobuyoshi Nakada)

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]
- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@44380 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 44380 - 12/24/2013 07:28 AM - nobu (Nobuyoshi Nakada)

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]
- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

Revision 44380 - 12/24/2013 07:28 AM - nobu (Nobuyoshi Nakada)

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]
- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

Revision 44380 - 12/24/2013 07:28 AM - nobu (Nobuyoshi Nakada)

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]
- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

Revision 44380 - 12/24/2013 07:28 AM - nobu (Nobuyoshi Nakada)

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]

- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

Revision 44380 - 12/24/2013 07:28 AM - nobu (Nobuyoshi Nakada)

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]
- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

Revision 44380 - 12/24/2013 07:28 AM - nobu (Nobuyoshi Nakada)

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]
- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

Revision 18f605a2 - 12/25/2013 07:58 AM - naruse (Yui NARUSE)

merge revision(s) 44380: [Backport #9296]

```
    * proc.c (rb_mod_define_method): consider visibility only if self
      in the caller is same as the receiver, otherwise make public as
      well as old behavior.  [ruby-core:57747] [Bug #9005]
      [ruby-core:58497] [Bug #9141]

    * vm.c (rb_vm_cref_in_context): return ruby level cref if self is
      same.
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_1@44410 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 12/02/2013 04:13 AM - elight (Evan Light)

Ran against trunk:

```
-e:1:in `<main>': private method `foo' called for :foo:Symbol (NoMethodError)
```

#2 - 12/14/2013 07:50 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Assigned

- Assignee set to nobu (Nobuyoshi Nakada)

#3 - 12/23/2013 03:54 PM - jeremyevans0 (Jeremy Evans)

There's only two days until the release of 2.1.0, and this still hasn't been fixed. This is a serious regression that breaks existing code (among other things, it causes Sequel's tests to freeze). If this can't be fixed before 2.1.0, I think r40022 should be reverted on the 2.1 branch, and the feature should be moved to 2.2.0. Assuming this is fixed on trunk before 2.1.0, it should be immediately backported to the 2.1 branch.

#4 - 12/23/2013 08:52 PM - nobu (Nobuyoshi Nakada)

send is irrelevant here.
If you make define_method public, it isn't needed.

```
c = Class.new {
  class << self
    public :define_method
  end
}
c.define_method(:foo) {}
c.new.foo #=> NoMethodError
```

Your code just reflects the default visibility at the top level.

#5 - 12/24/2013 04:28 PM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r44380.
Jeremy, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.

---

proc.c: make method by define_method public

- proc.c (rb_mod_define_method): consider visibility only if self in the caller is same as the receiver, otherwise make public as well as old behavior. [ruby-core:57747] [Bug #9005] [ruby-core:58497] [Bug #9141]
- vm.c (rb_vm_cref_in_context): return ruby level cref if self is same.

#6 - 12/24/2013 04:34 PM - nobu (Nobuyoshi Nakada)

- Backport set to 1.9.3: DONTNEED, 2.0.0: DONTNEED, 2.1.0: REQUIRED

#7 - 10/22/2014 05:25 PM - scotttam (Scott Tamosunas)

This isn't fixed on 2.1.3. Has that changeset not made it into a release?

```
2.1.3 :001 > RUBY_VERSION
 => "2.1.3"
2.1.3 :002 > class Foo
2.1.3 :003?>
2.1.3 :004 >       private
2.1.3 :005?>
2.1.3 :006 >       define_method("bar") { puts "BAR" }
2.1.3 :007?>   end
 => :bar
2.1.3 :008 >
2.1.3 :009 >   Foo.new.bar
NoMethodError: private method `bar' called for #<Foo:0x007ff90b8e3198>
    from (irb):9
    from /Users/me/.rvm/rubies/ruby-2.1.3/bin/irb:11:in `<main>'`
```

Where this works on 2.0

```
2.0.0-p451 :001 > RUBY_VERSION
 => "2.0.0"
2.0.0-p451 :002 > class Foo
2.0.0-p451 :003?>
2.0.0-p451 :004 >       private
2.0.0-p451 :005?>
2.0.0-p451 :006 >       define_method("bar") { puts "BAR" }
2.0.0-p451 :007?>   end
 => #<Proc:0x0000010103cd00@(irb):6 (lambda)>
2.0.0-p451 :008 >
2.0.0-p451 :009 >   Foo.new.bar
BAR
```

#8 - 10/26/2014 03:58 PM - jeremyevans0 (Jeremy Evans)

Scott Tamosunas wrote:

> This isn't fixed on 2.1.3. Has that changeset not made it into a release?
>
> ```
> 2.1.3 :001 > RUBY_VERSION
>  => "2.1.3"
> 2.1.3 :002 > class Foo
> 2.1.3 :003?>
> 2.1.3 :004 >       private
> 2.1.3 :005?>
> 2.1.3 :006 >       define_method("bar") { puts "BAR" }
> 2.1.3 :007?>   end
>  => :bar
> 2.1.3 :008 >
> 2.1.3 :009 >   Foo.new.bar
> NoMethodError: private method `bar' called for #<Foo:0x007ff90b8e3198>
>   from (irb):9
>   from /Users/me/.rvm/rubies/ruby-2.1.3/bin/irb:11:in `<main>'`
> ```
>
> Where this works on 2.0
>
> ```
> 2.0.0-p451 :001 > RUBY_VERSION
>  => "2.0.0"
> 2.0.0-p451 :002 > class Foo
> 2.0.0-p451 :003?>
> 2.0.0-p451 :004 >       private
> ```

```
2.0.0-p451 :005?>
2.0.0-p451 :006 >       define_method("bar") { puts "BAR" }
2.0.0-p451 :007?>   end
 => #<Proc:0x0000010103cd00@(irb):6 (lambda)>
2.0.0-p451 :008 >
2.0.0-p451 :009 >   Foo.new.bar
BAR
```

The behavior change here is deliberate, since you are calling define_method inside the class definition after calling private. This bug was that define_method when called outside the class definition was generating private methods, which was fixed before the release of 2.1.0.