Ruby - Bug #4537

Incorrectly creating private method via attr_accessor

03/29/2011 08:04 PM - ryanlecompte (Ryan LeCompte)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-darwin15]	**Backport:**	2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

The following fails with a failure to call "x=" private method

```
String.send(:attr_accessor, :x)
s = ""
s.x = 100
```

The following works:

```
class String
  self.send(:attr_accessor, :x)
end
s = ""
s.x = 100
```

Associated revisions

Revision ef45a57801a2ae8621b0cde59f11159f89f0a8dc - 08/01/2019 03:52 PM - jeremyevans (Jeremy Evans)

Make attr* methods define public methods if self in caller is not same as receiver

Previously, attr* methods could be private even if not in the
private section of a class/module block.

This uses the same approach that ruby started using for define_method
in 1fc33199736f316dd71d0c551edbf514528ddde6.

Fixes [Bug #4537]

Revision ef45a57801a2ae8621b0cde59f11159f89f0a8dc - 08/01/2019 03:52 PM - jeremyevans (Jeremy Evans)

Make attr* methods define public methods if self in caller is not same as receiver

Previously, attr* methods could be private even if not in the
private section of a class/module block.

This uses the same approach that ruby started using for define_method
in 1fc33199736f316dd71d0c551edbf514528ddde6.

Fixes [Bug #4537]

Revision ef45a578 - 08/01/2019 03:52 PM - jeremyevans (Jeremy Evans)

Make attr* methods define public methods if self in caller is not same as receiver

Previously, attr* methods could be private even if not in the
private section of a class/module block.

This uses the same approach that ruby started using for define_method
in 1fc33199736f316dd71d0c551edbf514528ddde6.

Fixes [Bug #4537]

History

#1 - 03/29/2011 08:06 PM - ryanlecompte (Ryan LeCompte)

The following fails with a failure to call "x=" private method

```
String.send(:attr_accessor, :x)
s = ""
s.x = 100
```

The following works:

```
class String
  self.send(:attr_accessor, :x)
end
s = ""
s.x = 100
```

#2 - 06/26/2011 06:45 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to ko1 (Koichi Sasada)

#3 - 06/26/2011 06:59 PM - ko1 (Koichi Sasada)

- Category set to core

- Target version set to 2.0.0

Behavior: It inherits current visibility (visibility of top-level is "private"). 1.8 also causes an exception. It seems to be a spec.

However, the following code:

```
::String.send(:define_method, :x=){|v|}
s = ''
s.x = 100
```

doesn't cause an exception. It seems to be an inconsistency.

The following code doesn't cause an exception on 1.9 and 1.8:

```
class C
  private
  ::String.send(:define_method, :x=){|v|}
end
s = ''
s.x = 100
```

However, the following code causes an exception only on 1.8:

```
class C
  private
  define_method(:x=){|v|}
end
C.new.x = 10
```

I can't understand how should it be.

Possible solutions:

(1) Remain it as spec.
(2) All "attr_*" methods define all methods in public.
(3) others?

#4 - 11/26/2012 09:19 AM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 2.6

No discussion.

#5 - 08/23/2016 10:50 PM - bughit (bug hit)

why should top level visibility (which applies to methods defined in the Object class) have any effect on other classes?

this also applies to any other module which you might be in

```
class Class1
end

module SomeUnrelatedModule
```

```
  Class1.send(:attr_accessor, :public_attr)
  private
  Class1.send(:attr_accessor, :private_attr)
end

c1 = Class1.new
c1.public_attr
c1.private_attr
```

what does visibility in SomeUnrelatedModule have to do with methods defined in Class1?

methods defined on a given target module when the default definee at the point of definition is not the target module, should be public, since that's the default visibility

```
Class1.send(:define_method,  :foo) {}
```

defines a public method

```
Class1.send(:attr_accessor, :foo)
```

should too

#6 - 08/23/2016 11:10 PM - bughit (bug hit)

define_method used to have the same (or similar) problem
https://bugs.ruby-lang.org/issues/9005

which was fixed, so why shouldn't this be?

#7 - 08/29/2016 04:29 PM - bughit (bug hit)

- Assignee changed from ko1 (Koichi Sasada) to nobu (Nobuyoshi Nakada)

@nobu (Nobuyoshi Nakada) shouldn't this get the same treatment as #9005

#8 - 10/11/2016 08:39 AM - shyouhei (Shyouhei Urabe)

- Status changed from Assigned to Closed

- ruby -v changed from ruby 1.9.2p180 (2011-02-18 revision 30907) [x86_64-darwin10.7.0] to ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-darwin15]

#9 - 10/11/2016 08:40 AM - shyouhei (Shyouhei Urabe)

- Status changed from Closed to Assigned

- Assignee changed from nobu (Nobuyoshi Nakada) to ko1 (Koichi Sasada)

#10 - 10/11/2016 11:11 AM - shyouhei (Shyouhei Urabe)

Just FYI we looked at it in developer meeting today and agreed this is a bug that have yet to be fixed.

#11 - 01/31/2017 09:04 AM - ko1 (Koichi Sasada)

- Description updated

#12 - 01/31/2017 09:10 AM - ko1 (Koichi Sasada)

So we should choose

> (2) All "attr_*" methods define all methods in public.

(on #3), right?

#13 - 08/01/2019 12:16 AM - jeremyevans0 (Jeremy Evans)

- File attr-visibility-4537.patch added

Attached is a patch that makes attr* methods handle visibility the same as define_method, using the approach nobu developed for #9005. I think this behavior makes the most sense. Always defining attr* methods as public (approach (2)) breaks backwards compatibility in more cases, and could lead to security issues in cases where public methods are treated differently than private methods in regards to how user input is handled.

#14 - 08/01/2019 08:21 AM - nobu (Nobuyoshi Nakada)

- Description updated

Seems fine.

BTW, it doesn't need to be the global String.

```
diff --git a/test/ruby/test_module.rb b/test/ruby/test_module.rb
index cdc084c8bc..9e57692ca0 100644
--- a/test/ruby/test_module.rb
+++ b/test/ruby/test_module.rb
@@ -706,13 +706,16 @@
   end

   def test_attr_public_at_toplevel
-    eval(<<-END, TOPLEVEL_BINDING)
-      String.send(:attr_accessor, :x)
-      String.send(:attr, :y)
-      String.send(:attr_reader, :z)
-      String.send(:attr_writer, :w)
+    s = Object.new
+    TOPLEVEL_BINDING.eval(<<-END).call(s.singleton_class)
+      proc do |c|
+        c.send(:attr_accessor, :x)
+        c.send(:attr, :y)
+        c.send(:attr_reader, :z)
+        c.send(:attr_writer, :w)
+      end
     END
-    s = ""
+
     assert_nil s.x
     s.x = 1
     assert_equal 1, s.x
@@ -727,10 +730,6 @@

     s.w = 4
     assert_equal 4, s.instance_variable_get(:@w)
-  ensure
-    [:x, :x=, :y, :z, :w=].each do |meth|
-      String.undef_method(meth) rescue nil
-    end
   end

   def test_const_get_evaled
```

#15 - 08/01/2019 03:55 PM - jeremyevans (Jeremy Evans)

- Status changed from Assigned to Closed

Applied in changeset git|ef45a57801a2ae8621b0cde59f11159f89f0a8dc.

Make attr* methods define public methods if self in caller is not same as receiver

Previously, attr* methods could be private even if not in the
private section of a class/module block.

This uses the same approach that ruby started using for define_method
in 1fc33199736f316dd71d0c551edbf514528ddde6.

Fixes [Bug #4537]

Files

attr-visibility-4537.patch	2.75 KB	08/01/2019	jeremyevans0 (Jeremy Evans)