Ruby master - Feature #7274

UnboundMethods should be bindable to any object that is_a?(owner of the UnboundMethod)

11/04/2012 07:55 PM - rits (First Last)

Status:	Rejected	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	2.6	

Description

=begin
as a corollary, ((({UnboundMethod})))s referencing the same method name on the same owner, should be equal

currently ((({UnboundMethod})))s binding is determined by the class via which they were retrieved, not the owner

class Base; def foo; end end
class Sub < Base; end

base_foo = Base.instance_method :foo
sub_foo = Sub.instance_method :foo
sub_foo.bind(Base.new).call

((({sub_foo.owner}))) is ((({Base}))) so there does not seem to be any reason why it's not safe for it to bind to an instance of ((({Base}))).

and there does not seem to be any reason for ((({sub_foo}))) and ((({base_foo}))) to be unequal, they both refer to the same method, ((({foo}))) on ((({Base}))).
=end

History

#1 - 11/05/2012 09:37 AM - marcandre (Marc-Andre Lafortune)

- Tracker changed from Bug to Feature

Reposted as original text was not posted on ruby-core mailing list.

#2 - 11/05/2012 10:00 AM - marcandre (Marc-Andre Lafortune)

- Category set to core

I agree.

Quite simple: https://github.com/marcandre/ruby/compare/marcandre:trunk...marcandre:bind_with_owner
Diff: https://github.com/marcandre/ruby/compare/marcandre:trunk...marcandre:bind_with_owner.diff

note that make test and test-all already pass with your proposed behavior. I see no reason why this could cause an incompatibility either.

Moving to "feature", as the current behavior is intended, as the current doc shows.

If this behavior is accepted, we could also revisit the == (and the inspect method too), as where the method was taken would become irrelevant, only the point of definition would be important. I believe this should be the case too.

Any objection for bind to use the actual owner?

#3 - 11/24/2012 10:11 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

#4 - 08/31/2013 10:42 AM - marcandre (Marc-Andre Lafortune)

- File bind.pdf added

#5 - 08/31/2013 02:46 PM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Rejected

It is due to implementation limitation of CRuby.

The structure of instances of subclass (TT_XXX) may be different from superclasses.
In that case, the C implemented methods would crash.

So we prohibit them conservatively.

Matz.

#6 - 09/01/2013 03:41 AM - marcandre (Marc-Andre Lafortune)

matz (Yukihiro Matsumoto) wrote:

> The structure of instances of subclass (TT_XXX) may be different from superclasses.
> In that case, the C implemented methods would crash.

I'm not sure I understand exactly what you are saying, but I am sure that the proposed patch would not cause crashes. Do you had a concrete example in mind?

Note that the request is not about binding methods across unrelated classes or going from a subclass up to a superclass (both of which could create crashes). It is only about using the actual owner instead of the class used to access it.

Can you explain why, for example, Kernel#dup can be bound to a String, but not if it was accessed using Array, even though the exact same code would be executed?

```
Kernel.instance_method(:dup).bind("hello") # => accepted
Array.instance_method(:dup).bind("hello")  # => raises error, but really this is the same method.
```

#7 - 09/16/2013 08:24 AM - rits (First Last)

marcandre (Marc-Andre Lafortune) wrote:

> matz (Yukihiro Matsumoto) wrote:
>
> > The structure of instances of subclass (TT_XXX) may be different from superclasses.
> > In that case, the C implemented methods would crash.
>
> I'm not sure I understand exactly what you are saying, but I am sure that the proposed patch would not cause crashes. Do you had a concrete example in mind?
>
> Note that the request is not about binding methods across unrelated classes or going from a subclass up to a superclass (both of which could create crashes). It is only about using the actual owner instead of the class used to access it.
>
> Can you explain why, for example, Kernel#dup can be bound to a String, but not if it was accessed using Array, even though the exact same code would be executed?
>
> ```
> Kernel.instance_method(:dup).bind("hello") # => accepted
> Array.instance_method(:dup).bind("hello")  # => raises error, but really this is the same method.
> ```

It's also not clear to me what the concern is, please explain.

#8 - 09/18/2013 12:03 PM - nobu (Nobuyoshi Nakada)

- Description updated

#9 - 09/18/2013 12:22 PM - matz (Yukihiro Matsumoto)

I am not sure the intention of the patch in [ruby-core:48864], but

- if the owner is a module
- if the binding object is an instance of the owner

we can allow bind(), but I am not sure how much useful this relaxing is, especially the latter one.

Matz.

#10 - 09/19/2013 08:17 PM - nobu (Nobuyoshi Nakada)

Instance methods of modules can be bound on any objects, already.
It's a part of method transplanting.

#11 - 11/25/2013 04:02 AM - rits (First Last)

is this rejected?

#12 - 11/26/2013 12:25 PM - matz (Yukihiro Matsumoto)

rits (First Last) Yes, basically. Method transplanting from modules is already allowed though.

Matz.

#13 - 11/26/2013 12:34 PM - rits (First Last)

matz (Yukihiro Matsumoto) wrote:

> rits (First Last) Yes, basically. Method transplanting from modules is already allowed though.
>
> Matz.

ok, but you never really explained what purpose this restriction serves, it seems pretty arbitrary and illogical

#14 - 11/26/2013 12:54 PM - matz (Yukihiro Matsumoto)

rits (First Last) You haven't read my message above, have you?

> It is due to implementation limitation of CRuby.
>
> The structure of instances of subclass (TT_XXX) may be different from superclasses.
> In that case, the C implemented methods would crash.
>
> So we prohibit them conservatively.

Matz.

#15 - 11/26/2013 01:26 PM - rits (First Last)

matz (Yukihiro Matsumoto) wrote:

> rits (First Last) You haven't read my message above, have you?
>
> > It is due to implementation limitation of CRuby.
> >
> > The structure of instances of subclass (TT_XXX) may be different from superclasses.
> > In that case, the C implemented methods would crash.
> >
> > So we prohibit them conservatively.
>
> Matz.

Neither I nor marcandre understood what you were alluding to and asked for a clarification, but you just repeated without clarifying.

You appear to be saying, and please correct if that's wrong, that a method from a subclass can not safely be bound to an instance of a superclass. If so, please note, that is not what's being suggested. sub_foo, from the original example, is a reference to a method (foo) that is defined in the superclass, not the subclass.

#16 - 11/26/2013 02:04 PM - rits (First Last)

to continue, the above:

we are binding a method from Base to an instance of Base, and it's failing. Why? How can that possibly be unsafe? What difference does it make that the method was requested from a subclass of Base? It certainly does not change the fact that the method is from Base.

#17 - 11/26/2013 02:47 PM - matz (Yukihiro Matsumoto)

OK, I misunderstood something.

In case foo is implemented in Base as in the original example, I admit that it will not cause any serious problem.

But I still have small concern.
If you are sure foo is implemented in Base, you can retrieve foo from Base (not from Sub),
so that you don't have to worry about redefinition.

If you are not, the code will be fragile. It's a sign of bad code.

Thus, I'd like to ask you why you want to relax? Consistency? Any actual use-case?
If there's actual non trivial use-case, I'd say go.

Matz.

#18 - 11/26/2013 03:41 PM - rits (First Last)

matz (Yukihiro Matsumoto) wrote:

> OK, I misunderstood something.
>
> In case foo is implemented in Base as in the original example, I admit that it will not cause any serious problem.
>
> But I still have small concern.
> If you are sure foo is implemented in Base, you can retrieve foo from Base (not from Sub),
> so that you don't have to worry about redefinition.
>
> If you are not, the code will be fragile. It's a sign of bad code.
>
> Thus, I'd like to ask you why you want to relax? Consistency? Any actual use-case?
> If there's actual non trivial use-case, I'd say go.

I discovered the current behavior while playing around, learning about method binding in ruby, not via a bug in my code. I noticed in irb (to_s) that an unbound method was remembering the class from which it was requested, which struck me as pointless (what difference could it possibly make). Then I guessed it must be used for something (for no good reason) and sure enough, two unbound method objects referencing the same method on the same class were unequal because they happened to have been requested from different classes.

This made absolutely no sense, if unbound methods are to have value equality (which they do) it should be based on owner and method name. This led to the next absurdity of a method being unbindable to an instance of its owner.

I thought such nonsense warranted a fix, so I filed a bug, I suppose you can disagree, but at least now we are on the same page.

#19 - 11/26/2013 03:53 PM - Anonymous

unsubscribe

#20 - 12/02/2013 07:12 AM - rits (First Last)

Matz, whether you reject this or not, I still would like to understand the reason for current behavior, it's not an accident, someone deliberately coded this.

#21 - 01/04/2014 02:29 PM - marcandre (Marc-Andre Lafortune)

Hi,

matz (Yukihiro Matsumoto) wrote:

> OK, I misunderstood something.
>
> In case foo is implemented in Base as in the original example, I admit that it will not cause any serious problem.
>
> But I still have small concern.
> If you are sure foo is implemented in Base, you can retrieve foo from Base (not from Sub),
> so that you don't have to worry about redefinition.
>
> If you are not, the code will be fragile. It's a sign of bad code.
>
> Thus, I'd like to ask you why you want to relax? Consistency? Any actual use-case?
> If there's actual non trivial use-case, I'd say go.

I agree. I can't really think of an actual use-case. I was asking for consistency's sake.

Files

bind.pdf	85.7 KB	08/31/2013	marcandre (Marc-Andre Lafortune)