Ruby trunk - Feature #6478

BasicObject#__class__

05/22/2012 10:05 PM - trans (Thomas Sawyer)

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

Description

How else is one supposed to get the class of a subclass of BasicObject?

History

#1 - 05/23/2012 12:03 AM - mame (Yusuke Endoh)

- Status changed from Open to Feedback

I don't understand you.

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 05/23/2012 12:12 AM - trans (Thomas Sawyer)

=begin
Sorry, I'll be more specific via example:

```
class Foo < BasicObject
end

foo = Foo.new

foo.class  #=> raises NoMethodError
```

How to get class?

I suggest adding **#class** feature if there is no current means.
=end

#3 - 05/23/2012 12:20 AM - mame (Yusuke Endoh)

- Status changed from Feedback to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

Okay, thanks. I assign this to matz.

--
Yusuke Endoh mame@tsg.ne.jp

#4 - 05/23/2012 02:17 PM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Feedback

- Target version changed from 1.9.3 to 2.0.0

=begin
((*Why*)) do you need it?

BTW, it's possible with pure-ruby.

```
class Foo < BasicObject
include ::Kernel.dup.module_eval {
alias_method(:class, :class)
undef_method *(instance_methods - [:class, :object_id])
self
}
end
```

p Foo.new.**class**
=end

#5 - 05/23/2012 02:34 PM - trans (Thomas Sawyer)

=begin
To ensure proper functionality when creating new instances from subclasses.

class Foo < BasicObject
def initialize(stuff)
@stuff = stuff
end
def dup
class.new(@stuff)
end
end

class Bar < Foo
end

We can't use (({Foo})) in dup, otherwise Bar would not be right.
=end

#6 - 05/23/2012 02:38 PM - trans (Thomas Sawyer)

> "BTW, it's possible with pure-ruby."

That's a rather nasty implementation. Is there no better way than that? I tried binding Kernel method but that didn't work, obviously, b/c BasicObject isn't "an instance of Kernel".

#7 - 05/24/2012 02:56 PM - nobu (Nobuyoshi Nakada)

=begin
Seems what you want is (({dup})), not (({**class**})).

class Foo < BasicObject
mix ::Kernel, dup: :dup, clone: :clone
end
=end

#8 - 05/24/2012 08:10 PM - trans (Thomas Sawyer)

That was just one example. Here, you can look at this for more cases:

https://github.com/rubyworks/ostruct2/blob/master/lib/ostruct2.rb

Just ctrl-f for **class**.

But what's this about "mix"? What Ruby are you running!? This is interesting, b/c I was thinking that I could use #respond_to? and I don't see anyway to add it to my BasicObject subclass except the "nasty" approach you demonstrated earlier.

#9 - 05/25/2012 01:14 AM - nobu (Nobuyoshi Nakada)

=begin
(({Module#mix})) is a feature introduced last year, but may be removed from 2.0.
=end

#10 - 05/25/2012 04:01 AM - Eregon (Benoit Daloze)

nobu (Nobuyoshi Nakada) wrote:

> Seems what you want is (({dup})), not (({**class**})).
>
> class Foo < BasicObject
> mix ::Kernel, dup: :dup, clone: :clone
> end

But that would include all methods from Kernel with the current behavior of #mix, as mix ::Kernel would do.
So you need to opt-out all methods:

```
class Foo < BasicObject
```

```
  meths = (::Kernel.instance_methods - [:dup])
  mix ::Kernel, meths.each_with_object(dup: :dup) { |m,h| h[m] = nil }
end
```

(And Foo.new.dup fails with "undefined method `initialize_dup'")

That behavior of #mix is not very intuitive I think, what do you think about:

diff --git a/class.c b/class.c
index 8e637c0..e9d7a7e 100644
--- a/class.c
+++ b/class.c
@@ -769,8 +769,9 @@ do_mix_method_i(st_data_t key, st_data_t value, st_data_t arg)
st_table *aliasing = argp->aliasing;
st_data_t old, alias;

- if (aliasing && st_lookup(aliasing, ID2SYM(id), &alias)) {
- if (NIL_P(alias)) return ST_CONTINUE;
- if (aliasing) {
- if (!st_lookup(aliasing, ID2SYM(id), &alias) || NIL_P(alias))
- return ST_CONTINUE; id = rb_to_id(alias); } if (st_lookup(argp->mtbl, id, &old)) {

(and corresponding changes for the three other functions).
That is, if a Hash of methods is given, only import these methods.

#11 - 10/25/2012 09:34 PM - yhara (Yutaka HARA)

- Target version changed from 2.0.0 to 2.6

#12 - 10/26/2012 12:10 PM - nobu (Nobuyoshi Nakada)

=begin
"Method transplanting" is introduced into 2.0, so you can write:
class Foo < BasicObject
include ::Module.new {
[:dup, :initialize_dup, :initialize_copy].each {|m|
define_method(m, ::Kernel.instance_method(m))
}
}
end

I expect someone would make such method in (({Module})) as an external library.
=end

#13 - 11/11/2012 12:21 AM - alexeymuranov (Alexey Muranov)

Maybe BasicObject is not intended to be subclassed directly? Why not to subclass Object instead? I do not think it is wrong that basic objects do not know who their class is, after all they are basic.

#14 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)