Ruby - Feature #615

"with" operator

10/06/2008 06:26 AM - Lavir_the_Whiolet (Lavir the Whiolet)

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

Description

=begin
"with" operator is required. It must work like an ordinary method which gets one arguemnt and a block. All expressions in the block are not required to point the argument explicitly; all method calls are related to the argument by default.

Example:

```
x = "Sample"
with x do
puts class
puts reverse
end
```

would produce:

```
String
elpmaS
```

=end

Associated revisions

Revision 3b0f952ec810c08eac01ce2377dfbb252026760b - 07/26/2019 09:29 AM - aycabta (aycabta .)

[ruby/rdoc] Support nesting text page URL

RDoc::Servlet#documentation_page replaces "/" in URL with "::" for class or module but it's also used for the replaced name on text pages. This causes a bug when text pages are in nesting directory.

This commit fixes #615.

https://github.com/ruby/rdoc/commit/d73b915b1e

Revision 3b0f952ec810c08eac01ce2377dfbb252026760b - 07/26/2019 09:29 AM - aycabta (aycabta .)

[ruby/rdoc] Support nesting text page URL

RDoc::Servlet#documentation_page replaces "/" in URL with "::" for class or module but it's also used for the replaced name on text pages. This causes a bug when text pages are in nesting directory.

This commit fixes #615.

https://github.com/ruby/rdoc/commit/d73b915b1e

Revision 3b0f952e - 07/26/2019 09:29 AM - aycabta (aycabta .)

[ruby/rdoc] Support nesting text page URL

RDoc::Servlet#documentation_page replaces "/" in URL with "::" for class or module but it's also used for the replaced name on text pages. This causes a bug when text pages are in nesting directory.

This commit fixes #615.

https://github.com/ruby/rdoc/commit/d73b915b1e

History

#1 - 10/06/2008 07:37 AM - pragdave (Dave Thomas)

=begin
irb(main):001:0> x = "Sample"
=> "Sample"
irb(main):002:0> x.instance_eval do
irb(main):003:1*   puts self.class
irb(main):004:1>   puts reverse
irb(main):005:1> end
String
elpmaS

=end

#2 - 10/06/2008 10:48 AM - nobu (Nobuyoshi Nakada)

=begin
Hi,

At Mon, 6 Oct 2008 06:24:46 +0900,
Lavir the Whiolet wrote in [ruby-core:19132]:

> "with" operator is required. It must work like an ordinary
> method which gets one arguemnt and a block. All expressions
> in the block are not required to point the argument
> explicitly; all method calls are related to the argument by
> default.

No. I'd implemented and tested it once but found it's just
problematic rather than useful. For instance, how do you
consider about instance variables?

--
Nobu Nakada

=end

#3 - 10/07/2008 01:48 AM - zenspider (Ryan Davis)

=begin

On Oct 5, 2008, at 15:35 , Dave Thomas wrote:

> Issue #615 has been updated by Dave Thomas.
>
> irb(main):001:0> x = "Sample"
> => "Sample"
> irb(main):002:0> x.instance_eval do
> irb(main):003:1*   puts self.class
> irb(main):004:1>   puts reverse
> irb(main):005:1> end
> String
> elpmaS

to take the idea one step further to the OP's request:

def with o, &block
o.instance_eval(&block)
end

x = "Sample"
with x do
puts self.class # needs self because class is part of the syntax
puts reverse
end

To me, this seems simultaneously trivial and unhelpful. I wouldn't
want to see it part of the language itself.

=end

#4 - 10/07/2008 04:36 AM - trans (Thomas Sawyer)

=begin

On Oct 6, 2:11 pm, _why w...@ruby-lang.org wrote:

> While investigating Guy Decoux's old messages, I've recently
> discovered a way to do exactly this. It involves inserting a mixin
> into the inheritance chain and then enabling and disabling it as
> needed.
>
> http://hackety.org/2008/10/06/mixingOurWayOutOfInstanceEval.html

Jimmy Crickets! That code is so straight forward. Er... Why isn't this
core Ruby?

T.

=end

#5 - 10/07/2008 05:22 AM - austin (Austin Ziegler)

=begin
On Mon, Oct 6, 2008 at 3:34 PM, Trans transfire@gmail.com wrote:

> On Oct 6, 2:11 pm, _why w...@ruby-lang.org wrote:
>
> > While investigating Guy Decoux's old messages, I've recently
> > discovered a way to do exactly this. It involves inserting a mixin
> > into the inheritance chain and then enabling and disabling it as
> > needed.
> >
> > http://hackety.org/2008/10/06/mixingOurWayOutOfInstanceEval.html
>
> Jimmy Crickets! That code is so straight forward. Er... Why isn't this
> core Ruby?

What I'm wondering is if this might be a way to do selector namespaces
that everyone has been asking for...

-austin

Austin Ziegler * halostatue@gmail.com * http://www.halostatue.ca/
* austin@halostatue.ca * http://www.halostatue.ca/feed/
* austin@zieglers.ca

=end

#6 - 10/07/2008 01:41 PM - duerst (Martin Dürst)

=begin
At 11:56 08/10/07, _why wrote:

> On Tue, Oct 07, 2008 at 05:47:23AM +0900, David A. Black wrote:
>
> > I've only looked at it briefly, and maybe I'm not getting it, but it
> > strikes me as kind of odd to have a situation where bare method calls
> > go to one object and instance variables belong to another. The merit
> > of instance_eval is that, though it changes context, it doesn't
> > introduce a new kind of context.
>
> Mixing in a module for the duration of a block isn't a new kind of
> context, though. You keep the current scope and get some additional
> methods. I mean would you think that require is odd because it can
> introduce a lot of methods without spelling them all out?

Well, I think it's sligthly more than that: All these methods are
essentially executed not with the current self, but with another,
hidden, self. For those cases where that's what you want, it's just
great, but it's still a bit of a strange feeling.

Regards, Martin.

#-#-# Martin J. Du"rst, Assoc. Professor, Aoyama Gakuin University

#-#-# http://www.sw.it.aoyama.ac.jp mailto:duerst@it.aoyama.ac.jp

=end

#7 - 10/08/2008 12:52 AM - trans (Thomas Sawyer)

=begin

On Oct 6, 4:20 pm, "Austin Ziegler" halosta...@gmail.com wrote:

> On Mon, Oct 6, 2008 at 3:34 PM, Trans transf...@gmail.com wrote:
>
> > On Oct 6, 2:11 pm, _why w...@ruby-lang.org wrote:
>
> > > While investigating Guy Decoux's old messages, I've recently
> > > discovered a way to do exactly this. It involves inserting a mixin
> > > into the inheritance chain and then enabling and disabling it as
> > > needed.
>
> > > http://hackety.org/2008/10/06/mixingOurWayOutOfInstanceEval.html
>
> > > Jimmy Crickets! That code is so straight forward. Er... Why isn't this
> > > core Ruby?
>
> > What I'm wondering is if this might be a way to do selector namespaces
> > that everyone has been asking for...

The idea of selector namespaces being that an extension applies
according to where (eg. the nesting) that an *invocation* of a method
takes place? That being the case, 'mixin/unmix' could be used, but
wouldn't the constant extending and unextending of objects be a huge
performance drain? Every method would effectively have to be prepended
with a procedure to check to see if the current nesting has changed
since the last invocation, and if so change the selected extensions
accordingly. Maybe the overhead isn't as great as I fret, but
certainly this could be happening thousands of times a second.

The only effective way I see for implementing selector namespaces is
via some sort of routing. Instead of tying a module into the class
hierarchy with direct reference pointers via a Proxy, it would need to
go through a Router that directed it to the appropriate extension
module according to the current nesting.

For example, instead of swapping the Proxy in and out, ocillating
between say:

```
AClass --> ASuperClass
```

then

```
AClass --> Proxy --> ASuperClass
|
'--> ModuleA
```

then

```
AClass --> Proxy --> ASuperClass
|
'--> ModuleB
```

Rather we need:

```
AClass --> Router --> ASuperClass
| |
| '--> ModuleA
|
'--> ModuleB
```

However, though I never quite understood why, but whatever causes the

well known Module Inclusion Problem, will likely be an issue with something like this too.

T.

=end

#8 - 11/29/2008 04:15 PM - ko1 (Koichi Sasada)

- Assignee set to matz (Yukihiro Matsumoto)

=begin

=end

#9 - 09/14/2010 04:53 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#10 - 10/18/2011 09:16 AM - naruse (Yui NARUSE)

- Project changed from Ruby to 14

- Category deleted (core)

- Target version deleted (3.0)

#11 - 10/23/2011 05:21 PM - naruse (Yui NARUSE)

- Project changed from 14 to Ruby

#12 - 02/07/2012 11:52 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Rejected

I'm rejecting this feature ticket because no progress has been made for a long time. See [ruby-core:42391].

--
Yusuke Endoh mame@tsg.ne.jp