Ruby - Feature #6684

Object#do

07/02/2012 11:07 AM - merborne (kyo endo)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		

Description

=begin
#Object#do
This is my first post.

初めての投稿です。よろしくお願いします。

Let me propose a new method Object#do.

次のような実装のObject#doを提案します。

```
class Object
  def do(*args, &blk)
    yield(self, *args)
  end
end
```

do encapsulate a sequencial procedure into a block. It makes some temporal variables to be block local and enhance code readability. I might say that do is an Object#tap with block result, or is Array#map for one item.

doはオブジェクトに対する一連の処理をブロック内に閉じ込めます。それによって、一時変数のブロック内への局所化、コードの可読性向上を実現します。結果を返すObject#tap、単項版Array#mapと見ることができます。

##Usage

```
# calculate average and standard deviation for list
#
# without `do`
scores = [56, 87, 49, 75, 90, 63, 65]
scores.inject(:+) / scores.size # => 69

avg = scores.inject(:+) / scores.size
sigmas = scores.map { |n| (avg - n)**2 }
sd = Math.sqrt(sigmas.inject(:+) / scores.size) # => 14.247806848775006

# with `do`
avg = [56, 87, 49, 75, 90, 63, 65].do { |s| s.inject(:+) / s.size } # => 69

sd = scores.do { |s|
  avg = s.inject(:+) / s.size
  sigmas = s.map { |n| (avg - n)**2 }
  Math.sqrt(sigmas.inject(:+) / s.size)
}
sd # => 14.247806848775006

# create a hash from a set of lists
#
# without `do`
h = Hash[ [:a, :b, :c].zip([1, 2, 3]) ] # => {:a=>1, :b=>2, :c=>3}

# with `do`
h = [:a, :b, :c].zip([1,2,3]).do { |arr| Hash[arr] } # => {:a=>1, :b=>2, :c=>3}


# sum of array using recursion
```

```
#
# without `do`
def sum(lst, mem=0)
  return mem if lst.empty?
  sum(lst.drop(1), mem+lst.first)
end

sum [*1..5], 5 # => 20

      # or

def sum(lst, mem=0)
  return mem if lst.empty?
  fst, *tail = lst
  sum(tail, mem+fst)
end

# with `do`
def sum(lst, mem=0)
  return mem if lst.empty?
  lst.do { |fst, *tail| sum(tail, mem+fst) }
end

sum2 [*1..5], 5 # => 20
```

BasicObject#instance_eval works for the above, but it not appropriate for them.

同様の目的はBasicObject#instance_evalでも実現できますが、この種の一般的演算には適さないと考えます。

Thank you for your consideration.
=end

Related issues:

Related to Ruby - Feature #6721: Object#yield_self	**Closed**
Has duplicate Ruby - Feature #12760: Optional block argument for `itself`	**Closed**

History

#1 - 07/04/2012 04:17 PM - nobu (Nobuyoshi Nakada)

something.do do end seems messy a little.

something.do do endというのはどうにも…°

#2 - 07/04/2012 04:32 PM - knu (Akinori MUSHA)

I'm afraid do might be a bit too bold choice for a Kernel method. (DBI has a do method for example)

You can use tap for now, like result = object.tap { |o| break f(o) }.

Speaking of which, I've always felt that it would be nice if object.{ ... } was a shorthand for object.instance_eval { ... }, but I can't think of a do-end counterpart for that.

#3 - 07/04/2012 09:33 PM - merborne (kyo endo)

nobu (Nobuyoshi Nakada) wrote:

> something.do do end seems messy a little.

Yes. something.do do end is messy. I'm happy if something.do end works.. I agree the name is not good.

#4 - 07/04/2012 09:43 PM - merborne (kyo endo)

knu (Akinori MUSHA) wrote:

> You can use tap for now, like result = object.tap { |o| break f(o) }.

great alternative!
I'm satisfied with this. Thank you.

#5 - 07/08/2012 09:47 AM - sorah (Sorah Fukumori)

- Status changed from Open to Rejected

No problem to reject?

#6 - 07/08/2012 05:16 PM - merborne (kyo endo)

sorah (Shota Fukumori) wrote:

> No problem to reject?

no problem. thank you.

#7 - 11/20/2015 06:04 PM - nobu (Nobuyoshi Nakada)

- Related to Feature #6721: Object#yield_self added

#8 - 09/20/2016 12:55 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Feature #12760: Optional block argument for `itself` added