Ruby - Feature #6298

Proc#+

04/15/2012 05:31 PM - trans (Thomas Sawyer)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	

Description

=begin
Maybe there is another way to do this, and if so please enlighten me.

I have a case where collection of blocks need to be handled as if a single block, e.g.

class BlockCollection
def initialize(*procs)
@procs = procs
end
def to_proc
procs = @procs
Proc.new{ |*a| procs.each{ |p| p.call(*a) } }
end
end

The issue with this is with #to_proc. It's not going to do the right thing if a BlockCollection instance is passed to #instance_eval b/c it would not actually be evaluating each internal block via #instance_eval.

But if we change it to:

```
def to_proc
  Proc.new{ |*a| procs.each{ |p| instance_exec(*a, &p) } }
end
```

It would do the right thing with #instance_eval, but it would no longer do the right thing for #call, b/c would it evaluate in the context of BlockCollection instance instead of where the blocks weer defined.

So, unless there is some way to do this that I do not see, to handle this Ruby would have to provide some means for it. To this end Proc#+ is a possible candidate which could truly combine two procs into one.
=end

Related issues:

Related to Ruby - Feature #5007: Proc#call_under: Unifying instance_eval and ...	**Assigned**

History

#1 - 04/16/2012 12:37 PM - mame (Yusuke Endoh)

- Status changed from Open to Rejected

Hello,

I think you have valid concern. AFAIK, there is no way to do this.
But #5007 (Proc#call_under) is apparently a more general solution
for this issue.
You will be able to write BlockCollection with Proc#call_under:

def to_proc
Proc.new{ |*a| procs.each{ |p| p.call_under(self, *a) } }
end

So, let's discuss the feature in that thread.

--
Yusuke Endoh mame@tsg.ne.jp