

Ruby - Bug #12671

Hash#to_proc result is not a lambda, but enforces arity

08/12/2016 01:25 AM - headius (Charles Nutter)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		
ruby -v:	ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-darwin14]	Backport: 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN
Description <pre>\$ ruby23 -e 'pr = {foo:1}.to_proc; puts pr.lambda?; pr.call rescue puts \$!; pr.call(1, 2) rescue puts \$!' false wrong number of arguments (given 0, expected 1) wrong number of arguments (given 2, expected 1)</pre> I believe it should be marked as a lambda, since it enforces arity.		

Associated revisions

Revision d514ba8e17106c6d159c3902ac5456d6269731f8 - 03/16/2020 02:38 PM - mame (Yusuke Endoh)

Proc made by Hash#to_proc should be a lambda [Bug #12671]

Like Symbol#to_proc (f0b815dc670b61eba1daaa67a8613ac431d32b16)

Revision d514ba8e17106c6d159c3902ac5456d6269731f8 - 03/16/2020 02:38 PM - mame (Yusuke Endoh)

Proc made by Hash#to_proc should be a lambda [Bug #12671]

Like Symbol#to_proc (f0b815dc670b61eba1daaa67a8613ac431d32b16)

Revision d514ba8e - 03/16/2020 02:38 PM - mame (Yusuke Endoh)

Proc made by Hash#to_proc should be a lambda [Bug #12671]

Like Symbol#to_proc (f0b815dc670b61eba1daaa67a8613ac431d32b16)

Revision 241244739f2b721ac7aa0961bd90d904c5e3fff6 - 07/29/2020 04:09 PM - Eregon (Benoit Daloze)

Fix arity of Hash#to_proc [Bug #12671]

Revision 241244739f2b721ac7aa0961bd90d904c5e3fff6 - 07/29/2020 04:09 PM - Eregon (Benoit Daloze)

Fix arity of Hash#to_proc [Bug #12671]

Revision 24124473 - 07/29/2020 04:09 PM - Eregon (Benoit Daloze)

Fix arity of Hash#to_proc [Bug #12671]

History

#1 - 08/12/2016 05:47 AM - nobu (Nobuyoshi Nakada)

None-lambda doesn't mean that it never checks its arguments.
And if it's a lambda, it doesn't match the arity value.

As for the implementation detail, there is no room for arity in ifuncs.

#2 - 08/12/2016 06:46 AM - headius (Charles Nutter)

JRuby implements Hash#to_proc as:

```
class Hash
  def to_proc
    method(:[]).to_proc
  end
end
```

```
end
```

This allows us to present the proc as a lambda with correct arity:

```
$ jruby -e "pr = {}.to_proc; puts pr.arity; puts pr.lambda?"
1
true
```

It works for MRI too:

```
$ ruby23 -e "class Hash; def to_proc; method(:[]).to_proc; end; end; pr = {}.to_proc; puts pr.arity; puts pr.lambda?"
1
true
```

I think this is more representative of this proc's behavior. Can MRI do it this way?

#3 - 03/16/2020 01:45 PM - mame (Yusuke Endoh)

This ticket was briefly discussed at today's dev meeting, and matz said that the result of Hash#to_proc should be a lambda.

#4 - 03/16/2020 02:38 PM - mame (Yusuke Endoh)

- Status changed from Open to Closed

Applied in changeset [git|d514ba8e17106c6d159c3902ac5456d6269731f8](https://github.com/ruby/ruby/commit/d514ba8e17106c6d159c3902ac5456d6269731f8).

Proc made by Hash#to_proc should be a lambda [Bug [#12671](https://github.com/ruby/ruby/issues/12671)]

Like Symbol#to_proc (f0b815dc670b61eba1daaa67a8613ac431d32b16)

#5 - 07/27/2020 08:14 PM - marcandre (Marc-Andre Lafortune)

- Status changed from Closed to Open

Is there a reason why the arity was not fixed to 1 (instead of -1)?

#6 - 07/28/2020 06:30 PM - Eregon (Benoit Daloze)

Looking at the commit, should Hash#to_proc use rb_func_lambda_new() instead of rb_func_proc_new()? (and then rb_func_proc_new is unused) Also it seems confusing that rb_func_proc_new() creates a lambda, so using rb_func_lambda_new() seems better here. Code changed quite a bit though around there.

#7 - 07/28/2020 06:42 PM - Eregon (Benoit Daloze)

Proposed fix (simple as it gets): <https://github.com/ruby/ruby/pull/3370>

#8 - 07/29/2020 04:10 PM - Eregon (Benoit Daloze)

- Status changed from Open to Closed

Applied in changeset [git|241244739f2b721ac7aa0961bd90d904c5e3fff6](https://github.com/ruby/ruby/commit/241244739f2b721ac7aa0961bd90d904c5e3fff6).

Fix arity of Hash#to_proc [Bug [#12671](https://github.com/ruby/ruby/issues/12671)]