Ruby - Feature #7702

Remove Proc#binding

01/16/2013 09:09 AM - jballanc (Joshua Ballanco)

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

Description

=begin
As discussed in the most recent Ruby implementer's meeting and elsewhere over the years, being able to get a handle to the binding of any block/proc has a number of problems:

- Code execution after passing a block/proc in unpredictable, as the binding of said proc can be used to redefine any local, instance variable, method, class, module, etc.
- Potentially sensitive data can leak out of the scope of a method via a proc binding
- Large object graphs may need to be retained for the lifetime of a proc, since any identifier in scope at the time of proc creation must remain live in the event that the binding of the proc is used to evaluate code

Additionally, removal of Proc#binding would enable a number of very useful optimizations and performance improvements.
=end

History

#1 - 01/18/2013 04:23 PM - Anonymous

I suspect it would have a negative impact on the ability to write a debugger. I seem to recall noting (I think it was from the pry person) that the equivalent of ruby-debug's binding_n was impossible in 2.0 and perhaps a patch was added to include it back.

However, given the gap between what is desirable in writing a good debugger and where things have been going in MRI, my current view is that there should be two separate interpreters. In this way, the concerns of good debugging don't have to run up against the concerns of good security or performance. And so far, the trend has been that debuggers have been losing. What can be done in 1.9 with respect to debugging is not as good comprehensive as what was available in 1.8. And in 2.0, I am not certain things will change that much although it appears that things will be different.

On the other hand, with respect to the two interpreter approach I also do believe that one should be able debug optimized code -- even if that means putting more burden on the programmer to understand more of the transformations, optimizations or run-time behavior of the interpreter. While one might not guarantee that Proc#binding is available because some kind of optimization took place, I think it a mistake to remove it in those cases where it could have been provided by using the same mechanisms that were in place previously.

On Tue, Jan 15, 2013 at 7:09 PM, jballanc (Joshua Ballanco) <jballanc@gmail.com> wrote:

> Issue #7702 has been reported by jballanc (Joshua Ballanco).
>
> ---
>
> Bug #7702: Remove Proc#binding
> https://bugs.ruby-lang.org/issues/7702
>
> Author: jballanc (Joshua Ballanco)
> Status: Open
> Priority: Normal
> Assignee:
> Category:
> Target version:
> ruby -v: 2.0.0-preview1
>
> =begin

As discussed in the most recent Ruby implementer's meeting and elsewhere over the years, being able to get a handle to the binding of any block/proc has a number of problems:

- Code execution after passing a block/proc in unpredictable, as the binding of said proc can be used to redefine any local, instance variable, method, class, module, etc.
- Potentially sensitive data can leak out of the scope of a method via a proc binding
- Large object graphs may need to be retained for the lifetime of a proc, since any identifier in scope at the time of proc creation must remain live in the event that the binding of the proc is used to evaluate code

Additionally, removal of Proc#binding would enable a number of very useful optimizations and performance improvements.
=end

--
http://bugs.ruby-lang.org/

#2 - 01/25/2013 12:11 PM - ko1 (Koichi Sasada)

- Category set to core

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

#3 - 01/28/2013 08:11 AM - jballanc (Joshua Ballanco)

I should note that this will obviously require much thought and some more research. First big question: how much would break if Proc#binding was removed? How severely, for example, would debugging be effected?

Also, an alternative to complete removal would be to limit the semantics of Proc#binding, such that only identifiers referenced in the body of the proc would be available from the proc's binding. If this alternate solution is considered, there may be overlap with feature request #7747 (Expanded API for Binding Semantics). If the binding returned by Proc#binding was pre-frozen, that may address the concerns listed above, but still allow for (mostly) unhindered debugging.

#4 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#5 - 07/15/2019 07:39 PM - jeremyevans0 (Jeremy Evans)

- Tracker changed from Bug to Feature

- ruby -v deleted (2.0.0-preview1)

- Backport deleted (2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN)

#6 - 04/03/2024 03:50 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned