Ruby master - Feature #14801

New method 'show_stack' to show Ruby stack(s) when program is running

06/01/2018 02:27 AM - duerst (Martin Dürst)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

It would be great to have a method to show the Ruby stack(s) (there are actually two of these) while a Ruby program is running. This would help people to understand how Ruby works internally. There is functionality in Ruby already to print out the stack(s), but it is only triggered when there is an internal error in the interpreter.

The attached patch is just a proof of concept. At least some more cleanup will be needed.

There's how it's supposed to work:

```
ruby -e '3 + 4 * 5 ** 6.show_stack'
```

will produce:

```
-- Control Frames -----------------------------------------------
c:0001 p:0000 s:0003 E:001c90 (none) [FINISH]
c:0002 p:0011 s:0009 e:000005 EVAL   -e:1 [FINISH]

-- Internal Stack ------------
0009 (0x6ffffef0058): 000000000000000d Integer:    6 <- Stack Pointer (2)
0008 (0x6ffffef0050): 000000000000000b Integer:    5
0007 (0x6ffffef0048): 0000000000000009 Integer:    4
0006 (0x6ffffef0040): 0000000000000007 Integer:    3
0005 (0x6ffffef0038): 0000000077770021 (flags) <- Environment Pointer (2)
0004 (0x6ffffef0030): 000000060013e3f1 (specval)
0003 (0x6ffffef0028): 0000000000000000 (me_cref) <- Stack Pointer (1)
```

In the above example, the various operands (3, 4, 5, 6) on the stack can easily be seen.

Because the stack grows e.g. when expressions are evaluated (see above), it's very handy to have the Object#show_stack, so that show_stack can be applied at any very specific point during execution. I implemented RubyVM.show_stack before I realized that Object#show_stack would be needed, so it's still there, but may now be superfluous.

The format of the output is based on the existing functions in vm_dump.c, but changed in a few aspects:
1) Overall order and terminology adapted to match the "Ruby under the Microscope" book.
2) Added more user-friendly output for frequent classes such as Integer, String, Symbol, nil, false, true,...
3) Cut off output of the actual call to the show_stack method.

Issues on which I'd like to get feedback:

- Builtin/stdlib or gem? (the functionality is not needed in production, only for educational purposes, but it would be more convenient to have it in the standard library because of the code overlap).
- Method name
- Implementation details (I'm sure there's lots of little issues to fix)

History

#1 - 06/01/2018 09:45 AM - shevegen (Robert A. Heiler)

A bit of feedback as asked by Martin; I'll skip the implementation detail part since
I do not know C so others have to comment on that part.

- The name: I think the name is ok in "itself", that is, if we see "show_stack",
  we can understand what is meant. That it can work on e. g. an Integer is a bit
  weird though, such as 6.show_stack. Perhaps it may be better to have it a bit
  more verbose and reside on Kernel.show_stack(6) or some other default ruby
  module (a bit like RbConfig or RubyVM so... perhaps a new one.) But anyway, I

think the name part is separate from the functionality, and I think the functionality is fine. I love introspection so just about every change to provide us with more information is good, IMO.

- As for built-in or gem, I agree that it would be better to have it within ruby core itself. There were other discussions about other changes and people providing reasons as to why something should be a gem or part of the stdlib/core. For example, since ruby is so flexible (duck typing and duck patching and doing all about the ducklings), additional functionality could be easily put into a gem and loaded /or removed) at will. For me, personally, having something part of core/stdlib, is good for two reasons:

- Just about everyone will have this functionality, which can be really helpful when distributing code. This is, for me, also a reason why I think the bundler integration was good - I myself do not use bundler, but I understand the rubygem team not wanting to duplicate functionality when bundler provides it, and vice versa. So from this point of view, having functionality within ruby core/stdlib is very, very useful.

- The other reason is a bit more from a design point of view. When changes are made, we can know that matz has approved of them. So that is a good thing IMO since it is a part of quality control too; in other words, to be ruby ruby.

I implemented RubyVM.show_stack before I realized that Object#show_stack would be needed, so it's still there, but may now be superfluous.

Strange. I would think that .show_stack fits better to RubyVM than Object.
But I guess matz can comment at the next developer meeting perhaps.

Files

show_stack.patch	4.88 KB	06/01/2018	duerst (Martin Dürst)