Ruby master - Feature #12790

Better inspect for stdlib classes

09/26/2016 02:17 PM - zverok (Victor Shepelev)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

#inspect is important for understanding "what I have" in irb/pry, and in puts-debugging, and in ton of other cases.

Sadly, some of important stdlib classes (in my opinion) fail to provide concise and readable representation for #inspect.

Some examples below:

BigDecimal (important for representing money values, for example)

Current behavior:

```
#<BigDecimal:128d34f4,'0.25E2',9(18)>
#   ^           ^         ^      ^
#   1           2         3      4
```

1. OK, this is reasonable
2. Do we really need object id here? As far as I can understand, developer is typically concerned only about numeric value identity, not object identity for bigdecimals.
3. OK, I understand about scientific representation, but it is hard to read (at least for me), and also single quotes around add to a confusion.
4. I'm not sure. Number of significant digits it is. So, what are the situation when you need to look at it constantly?..

So, ideal behavior:

```
#<BigDecimal: 250>
# or, preserving num. of sig.dig.
#<BigDecimal: 250 digits: 9(18)>
# ...or something like this

# But for really large numbers it is still
#<BigDecimal: 1.5E35>
```

Side note: try to guess what BigDecimal.new(2)**10_000 looks like?.. And whether this look is really useful for anything.

Date and DateTime

```
Date.today
# => #<Date: 2016-09-26 ((2457658j,0s,0n),+0s,2299161j)>
DateTime.now
# => #<DateTime: 2016-09-26T16:40:17+03:00 ((2457658j,49217s,186886101n),+10800s,2299161j)>
```

Maybe it is just me, but it does not look like part in parenthises (and double parenthises!) contain information of such real importance that it can change our perception of "what's going on"?.. If you work with current dates, it is just unnecessary; if you work with some really complicated historical dates, it is not enough, being too concise and enigmatic to give some understanding of epochs and calendars.

(And to add, it is awfully inconsistent with Time's #inspect, which does provide just 2016-09-26 17:03:59 +0300.)

History

#1 - 09/26/2016 06:37 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I agree, I'd also love to have a better inspect representation for big decimals and dates.

#2 - 09/27/2016 02:13 AM - nobu (Nobuyoshi Nakada)

- Description updated

- Tracker changed from Misc to Feature

I agree about BigDecimal almost.
Prec may be useful but MaxPrec would be rarely.
And for huge BigDecimal, I prefer _ separated representation like Integer.
https://github.com/ruby/ruby/compare/trunk...nobu:feature/12790-bigdecimal%23inspect

As for Date and DateTime, do you still need/use them?

#3 - 09/27/2016 08:37 AM - zverok (Victor Shepelev)

> As for Date and DateTime, do you still need/use them?

Why not? It is a common pattern seen everywhere to use Date when you need, just, err, date (e.g. emphasize the fact that "this variable is granulated to days").

About DateTime matters are, of course, more confusing (as far as I can understand, we had both Time and DateTime initially, because just Time had a limited range, but it became better, so DateTime is, kinda, unnecessary now?..)

Also, I believe I saw somewhere in this tracker statements that date library have no maintainers currently: does it planned to be retired completely?..

#4 - 09/27/2016 09:00 AM - zverok (Victor Shepelev)

> https://github.com/ruby/ruby/compare/trunk...nobu:feature/12790-bigdecimal%23inspect

So, you decided to leave scientific notation for any amounts? Could we talk about that?.. It is really hard not to make small mistakes comparing values like 0.25E2 and 0.248E3 and so on by eyes, on tests and in console.

#5 - 09/27/2016 09:54 AM - Eregon (Benoit Daloze)

Maybe it would be nice to use a formatting like sprintf's %g?
So numbers without a huge exponent would print just like floats.
Actually, following how Float are printed would be consistent (1.234e+57, 12.34, etc),
while keeping the extra info it's a BigDecimal:

```
#<BigDecimal:1234.5678,#{prec}>
```

#6 - 09/30/2016 05:47 AM - shevegen (Robert A. Heiler)

That the object id is displayed also used to confuse me. I adjusted to it
so it is fine but I don't think I ever really needed to know the object
id. Perhaps it visually looks cuter if the object id is shown. :)

On a slightly related note, I think that Aaron Petterson wrote the "I
am a puts debugger" blog entry.

```
https://tenderlovemaking.com/2016/02/05/i-am-a-puts-debuggerer.html
```

I am also a puts debugger, or actually, I am a pp debugger. I usually
end up doing

```
require 'pp'
```

Would be nice if pp would be available all the time without a require
statement. But anyway, sorry for commenting this here on an unrelated
entry, I just wanted to mention it because debug-inspect output was
the topic.

Back to date and time - I actually never noticed this, this is
interesting. But to be honest, I don't think I really needed that
information. I think I have had customized inspect output for my own
code only two or three times in 10 years or so. The only use case I
really once had was for a MUD engine started in ruby, where I needed
the objects (like a sword) to show some more meaningful output in
general. Other than that, I don't think I actually paid that much

attention to the inspect output - providing a to_str (or was it
to_s, I always mix them up) method was then usually more important.

#7 - 02/27/2017 07:16 PM - zverok (Victor Shepelev)

```
$ rvm use 2.4
$ ruby -rbigdecimal -e "p BigDecimal.new(150)"
0.15e3
```

So, the boilerplate dropped yet scientific notation remains? Could I provide patch to get rid of it for "reasonably small" numbers?..

#8 - 02/27/2017 08:36 PM - stomar (Marcus Stollsteimer)

Two thoughts regarding BigDecimal and Float:

1. INHO it's desirable that the inspect output should be different; it should be possible to distinguish between a BigDecimal and a Float by inspecting it.

2. BigDecimal doesn't use scientific notation in standard form (with a leading digit between 1 and 9), and there are also differences in the formatting of the exponent, see examples below.

Maybe BigDecimal could adopt the style used by Float, with an additional marker/tag(?) to distinguish between them?

```
require "bigdecimal"
require "bigdecimal/util"

 12300000000000000.0          # => 1.23e+16
"12300000000000000.0".to_d    # => 0.123e17

 0.000000123                  # => 1.23e-07
"0.000000123".to_d            # => 0.123e-6
```