Ruby trunk - Feature #10505

[PATCH 2/n] 同相条件を扱いたい./Object#eql? with block. (ja/en)

11/13/2014 12:31 AM - gogotanaka (Kazuki Tanaka)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

English follows japanese

文脈

両辺に同じ処理を施してその同値性を確かめたいという場面は往々にしてあると思います.

`a % 2 == b % 2`

`a.abs == b.abs`

`"RUBY".downcase == "Ruby".downcase`

この際、施す処理を両辺に書くのは嫌だというのが今回のチケットの主たるモチベーションになります.

特に長い処理ともなると尚更だと思います.
（そもそも長い処理を両辺に施すコンテキストがイカン！ という批判は甘んじて受け入れますが）

提案

施す処理を各オブジェクトの#eql?に対してブロックとして渡せるようにしたいです.

```
#  2を法とする合同式
a.eql?(b) { |n| n % 2 }
```

```
#  意味論的にあまり好きでないですが、ブロックをシンボルで渡す事も可能とします .
"RUBY".eql?("Ruby", &:downcase)
```

懸念点

- block処理後は#===で比較するので挙動に若干の違和感.

```
1.eql?(1.0, &:itself)
# => true
```

```
1.eql?(1.0)
# => false
```

- 既存のコードで#eql?にブロックを渡している可能性がある?

変更の範囲も広くなりそうなのと見積もれてない懸念点がある気がしたので、ひとまずNumeric#eql?だけ実装してみました
しばらく様子を見て問題がなければ他のオブジェクト群に対しても実装したいかとも思います.

よろしくお願い致します.

Motivation

We often encounter a situation where we need to compare both operands after certain function apply for these.

`a % 2 == b % 2`

`a.abs == b.abs`

```
"RUBY".downcase == "Ruby".downcase
```

I am not willing to write same function both operands.
This is main motivation.

Proposal

Now if the optional block is given, compare between results of running block for both operands.

```
# check wether `a` and `b` are congruent modulo 2 or not
a.eql?(b) { |n| n % 2 }

"RUBY".eql?("Ruby", &:downcase)
```

I've found similar issue here

For now, I've implemented for Numeric#eql?
If it looks like good for you, I'm gonna implement for other Classes

Concern

- We'll compare by using #=== after passed to block. so it may look like weird.

```
1.eql?(1.0, &:itself)
# => true

1.eql?(1.0)
# => false
```

Thanks, gogo.

History

#1 - 11/13/2014 02:21 AM - sorah (Sorah Fukumori)

This syntax helps writability but hard to read. We'll have to know that eql? takes an block and behaves like so...

IMO, I don't like such shorthand supports *only* writability.

#2 - 11/13/2014 03:04 AM - gogotanaka (Kazuki Tanaka)

@Shota Fukumori san

Thank you for your comment :)

I got your point, as you said a % 2 == b % 2 is tottaly obvious for anybody,
On the other hand, a.eql?(b) { |n| n % 2 } looks little bit strange.
(note: I belive a.eql?(b) { |n| n % 2 } is more readable if we've known eql? takes an block and behaves like so)

Shota Fukumori wrote:

> IMO, I don't like such shorthand supports *only* writability.

For this point, my motivation is come by *not only* writability, but also some point I'm gonna explain below.
As I said, it can be helpful for readable

```
obj1.eql?(obj2) { |o| func(o.method1.method2) }
```

is more readable than

```
func(obj1.method1.method2) == func(obj2.method1.method2)
```

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

And in this case, we can make every piece of knowledge a single.(you may call it DRY)
It can be helpful for our maintaining.

IMO, even if it will be helpful for only writability, there is possible we think it.
Because I believe Ruby place much value on flexible.

#3 - 11/13/2014 03:16 AM - sorah (Sorah Fukumori)

At least for me, later one: func(obj1.method1.method2) == func(obj2.method1.method2) is better.

#4 - 11/13/2014 03:35 AM - gogotanaka (Kazuki Tanaka)

@Shota Fukumori san
ok. I know you prefer later one.
And there are some people(included me) who prefer former one.

I never mention that which is minority or majority, but we can think about supporting both.
As far as I know, Ruby have done it so far.

#5 - 11/13/2014 08:16 AM - nobu (Nobuyoshi Nakada)

You can write:

```
["RUBY", "Ruby"].map(&:downcase).inject(:==)
```

#6 - 11/13/2014 08:27 AM - recursive-madman (Recursive Madman)

Nobuyoshi Nakada wrote:

> You can write:
>
> ```
> ["RUBY", "Ruby"].map(&:downcase).inject(:==)
> ```

In my opinion this is not a nice use of inject, since it won't work for arrays with > 2 elements.

#7 - 11/13/2014 08:34 AM - sorah (Sorah Fukumori)

%w(RUBY Ruby).map(&:downcase).uniq.size == 1 covers multiple elements, but note that the original proposal doesn't cover such situation.

#8 - 11/13/2014 12:04 PM - mame (Yusuke Endoh)

I think this is actually a duplicate of #10426 except the method name.
Why did you create another ticket instead of adding a comment to that ticket?

--
Yusuke Endoh mame@ruby-lang.org

#9 - 11/13/2014 05:52 PM - gogotanaka (Kazuki Tanaka)

@Nobuyoshi Nakada san, @ Recursive Madman san, @Shota Fukumori san.
I'm honored to get such nice comments, thank so much.

OK, I'm not sure I'll use ["RUBY", "Ruby"].map(&:downcase).inject(:==) or %w(RUBY Ruby).map(&:downcase).uniq.size == 1,
but I found there is no affirmative reason we support *Object#eql? with block*
I've agreed with you so far.

@ Yusuke Endoh san
I'm sure you know what I mean ;)

Thanks guys.

Files

Update_Numeric#eql_.patch	1.23 KB	11/13/2014	gogotanaka (Kazuki Tanaka)
Add_tests_for_Numeric#eql_.patch	748 Bytes	11/13/2014	gogotanaka (Kazuki Tanaka)