Ruby - Feature #3714

Add getters for Enumerator

08/19/2010 04:51 AM - marcandre (Marc-Andre Lafortune)

Status:	Closed
Priority:	Normal
Assignee:	knu (Akinori MUSHA)
Target version:	2.6

Description

=begin
Given an enumerator, there is no way to know what receiver, method and arguments it is based upon (although one could use Enumerator#inspect and parse it to get the method).

For sake of completeness and introspection, it could be useful to have access to them.

This patch adds Enumerator#receiver, #method and #arguments: http://github.com/marcandre/ruby/commit/fd4c17f7fd

Note that for Enumerator created with a block like Enumerator.new{|y| y << 1 << 2}, then receiver is a Enumerator::Generator, method is :each and arguments is []. This is consistent with inspect.

Thanks.
=end

Related issues:

Related to Ruby - Feature #3715: Enumerator#size and #size=	**Rejected**	**08/19/2010**
Related to Ruby - Feature #14044: Introduce a new attribute `step` in Range	**Rejected**	
Has duplicate Ruby - Feature #13904: getter for original information of Enume...	**Closed**	

History

#1 - 08/21/2010 11:30 PM - Eregon (Benoit Daloze)

=begin
Hi,

On 18 August 2010 21:51, Marc-Andre Lafortune redmine@ruby-lang.org wrote:

> For sake of completeness and introspection, it could be useful to have access to them.

Do you have any idea for a use case ?

Would it allow to interact better with chained Enumerator ?

Regards,
B.D.

=end

#2 - 08/24/2010 02:56 AM - runpaint (Run Paint Run Run)

=begin
The information's presence in #inspect output implies its utility. It is a nod to symmetry to allow the arguments with which the enumerator was instantiated to be retrieved subsequently. However, the selector named 'method' is problematic because it shadows Object#method. Perhaps it could be named #name?
=end

#3 - 08/27/2010 12:55 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

- Assignee set to knu (Akinori MUSHA)

=begin

=end

#4 - 08/29/2010 04:37 PM - knu (Akinori MUSHA)

- Status changed from Assigned to Feedback

- Priority changed from Normal to 3

=begin
Enumerator was designed as a means to providing a handy and safe way to generate and pass (or return) an Enumerable object without exposing internal details.

So, it was intentional for me as the original API designer not to have provided those accessor methods. For a generator of an Enumerator object, the ingredients are known without a need for such methods. For a consumer, it should only matter that it is an Enumerable object.

Actually, Enumerator#inspect originally did not show anything internal but I changed it to show some just to help debugging. So if the symmetry really matters and you insist on it I'd rather change it back.

These are my points of view after all and you can of course make yours to persuade me. First of alll, can you tell me what use cases you are thinking of?
=end

#5 - 08/30/2010 10:37 AM - runpaint (Run Paint Run Run)

=begin

> Actually, Enumerator#inspect originally did not show anything internal but I changed it to show some just to help debugging. So if the symmetry really matters and you insist on it I'd rather change it back.

I certainly don't insist. :-) Your explanation was much appreciated; I concede the argument.
=end

#6 - 09/02/2010 12:26 AM - mame (Yusuke Endoh)

=begin
Hi, knu

2010/8/29 Akinori MUSHA redmine@ruby-lang.org:

> So, it was intentional for me as the original API designer not to have provided those accessor methods. ?For a generator of an Enumerator object, the ingredients are known without a need for such methods. ?For a consumer, it should only matter that it is an Enumerable object.

It is reasonable (for me), but sometimes too strict.

For example, when I want to define my custom Enumerator#inspect, I must parse the result of original inspect. It is cumbersome.

In general, Ruby does not stop us to shoot our foot. How about providing methods that Marc-Andre suggested, as private methods?

> Actually, Enumerator#inspect originally did not show anything internal but I changed it to show some just to help debugging. ?So if the symmetry really matters and you insist on it I'd rather change it back.

Nooo! The current Enumerator#inspect is really helpful. Do not revert it ;-(

--
Yusuke Endoh mame@tsg.ne.jp

=end

#7 - 10/25/2012 09:57 PM - yhara (Yutaka HARA)

- Description updated

- Target version changed from 2.0.0 to 2.6

#8 - 01/15/2013 11:13 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Feedback to Open

I can think of two use cases.

I would like to backport some Ruby 2.0 features, and the lack of introspection of Enumerator makes it quite difficult.

1. In Ruby 1.9, Enumerator#each does not accept arguments. This has been fixed in 2.0, but there is no reasonable way to monkeypatch Enumerator#each in 1.9. If these getters existed, that would be easy.

2. Ruby 2.0 introduces Enumerable#size. If the getters existed, it would have been possible to backport Enumerator#size in Ruby for 1.9 without monkeypatching all methods that produce enumerators.

#9 - 10/22/2017 05:30 AM - knu (Akinori MUSHA)

- Has duplicate Feature #13904: getter for original information of Enumerator added

#10 - 11/22/2017 10:06 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Closed

#11 - 03/15/2018 12:24 AM - mrkn (Kenta Murata)

- Related to Feature #14044: Introduce a new attribute `step` in Range added