Hello,

Enumerable::Lazy#inspect is undefined right now and relies on Enumerator#inspect:

```
(1..5).lazy => #<Enumerator::Lazy: #<Enumerator::Generator:0x00000101a2f9e8>:each>
```

I think it would be nice to be similar to (direct) Enumerator#inspect:

```
(1..5).each => #<Enumerator: 1..5:each>
```

So something like: # or #>

It would also be nice to show the chaining, like Enumerator does:

```
(1..5).select.map.flat_map => #<Enumerator: #<Enumerator: #<Enumerator: 1..5:select>:map>:flat_map>
```

What do you think?

**Associated revisions**

Revision 6b885f6e - 03/24/2012 03:17 PM - shugo (Shugo Maeda)

- enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect.
  [ruby-core:43345] [Bug #6159]

- enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@35124 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 35124 - 03/24/2012 03:17 PM - shugo (Shugo Maeda)

- enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect.
  [ruby-core:43345] [Bug #6159]

- enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

Revision 35124 - 03/24/2012 03:17 PM - shugo (Shugo Maeda)
• enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect. [ruby-core:43345] [Bug #6159]

• enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

Revision 35124 - 03/24/2012 03:17 PM - shugo (Shugo Maeda)

• enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect. [ruby-core:43345] [Bug #6159]

• enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

Revision 35124 - 03/24/2012 03:17 PM - shugo (Shugo Maeda)

• enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect. [ruby-core:43345] [Bug #6159]

• enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

Revision 35124 - 03/24/2012 03:17 PM - shugo (Shugo Maeda)

• enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect. [ruby-core:43345] [Bug #6159]

• enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

Revision 35124 - 03/24/2012 03:17 PM - shugo (Shugo Maeda)

• enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect. [ruby-core:43345] [Bug #6159]

• enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

Revision 35124 - 03/24/2012 03:17 PM - shugo (Shugo Maeda)

• enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect. [ruby-core:43345] [Bug #6159]

• enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

History

#1 - 03/17/2012 12:35 AM - shugo (Shugo Maeda)
Benoit Daloze wrote:

So something like: # or #>

It would also be nice to show the chaining, like Enumerator does:

```ruby
(1..5).select.map.flat_map #=> Enumerator: #<Enumerator: #<Enumerator: 1..5:select>:map>:flat_map>
```

What do you think?

I agree that it would be nice. However, unlike Enumerator, Enumerator::Lazy doesn't have enough information, so it needs to have some information only for inspect. Is it so important to make inspect rich?

#2 - 03/23/2012 06:42 PM - shugo (Shugo Maeda)
- Status changed from Open to Feedback
- Assignee set to shugo (Shugo Maeda)

#3 - 03/24/2012 08:19 PM - Eregon (Benoit Daloze)

I agree that it would be nice. However, unlike Enumerator, Enumerator::Lazy doesn't have enough information, so it needs to have some information only for inspect.

Is it so important to make inspect rich?

I'm not sure, but it would certainly be helpful when debugging.

On the other hand, if that would impact significantly performance or code (maintainability), it's probably not worth it.

I guess I'm not used to a core Ruby class without a nice #inspect.

#4 - 03/25/2012 12:17 AM - shugo (Shugo Maeda)
- Status changed from Feedback to Closed
- % Done changed from 0 to 100

This issue was solved with changeset r35124.

Benoit, thank you for reporting this issue.

Your contribution to Ruby is greatly appreciated.

May Ruby be with you.

- enumerator (enumerator_inspect): include the original receiver and method name of Enumerator::Lazy in the result of inspect.
  [ruby-core:43345] [Bug #6159]
- enumerator (InitVM_Enumerator): don't use rb_define_alias for some methods such as collect in order to make rb_frame_this_func() return the correct method names.

#5 - 03/25/2012 12:23 AM - shugo (Shugo Maeda)

Eregon (Benoit Daloze) wrote:

I agree that it would be nice. However, unlike Enumerator, Enumerator::Lazy doesn't have enough information, so it needs to have some information only for inspect.

Is it so important to make inspect rich?

I'm not sure, but it would certainly be helpful when debugging.

On the other hand, if that would impact significantly performance or code (maintainability), it's probably not worth it.

I guess I'm not used to a core Ruby class without a nice #inspect.

I guess it wouldn't impact performance or maintainability so much, and fixed Enumerable::Lazy#inspect so that the following code:

```ruby
(1..10).lazy.select(&:odd?).map(&:to_s).inspect
```

to return the following string:
Awesome, thanks!

Your answer raises another question: Would it not be more readable if `inspect` was closer to the code?

```ruby
(1..10).lazy.select(&:odd?).map(&:to_s).cycle(2).inspect
```

Instead of
```
"#:select>:map>:cycle(2)>
```
Something like
```
"#"
```
?

Your answer raises another question: Would it not be more readable if `inspect` was closer to the code?

The former can distinguish the following lazy enumerators, but the latter can't:

```ruby
p (1..10).lazy.select(&:odd).map(&:to_s)
 #=> #:select>:map>
p (1..10).select.lazy.map(&:to_s)
 #=> #:map>
p (1..10).select.map.lazy
 #=> #:map>>
```

So I prefer the current behavior.

Right, thank you for the explanation.

Hello,

2012/3/25, shugo (Shugo Maeda)
redmine@ruby-lang.org:

I agree with you, as a core team member.
It is a simple and straightforward implementation.
But I also agree with Benoit, as a user.
The output is actually a bit verbose.

I guess the fault is the prefix "<Enumerator::Lazy::":
It is too long and unfriendly for human eyes.

--
Yusuke Endoh mame@tsq.ne.jp

So I prefer the current behavior.

Hello,

2012/3/27 Yusuke Endoh mame@tsq.ne.jp:

So I prefer the current behavior.
I agree with you, as a core team member. It is a simple and straightforward implementation. But I also agree with Benoit, as a user. The output is actually a bit verbose.

I guess the fault is the prefix "<Enumerator::Lazy: ". It is too long and unfriendly for human eyes.

Hmm... could you make a counter proposal? We may be able to choose a shorter prefix such as "<lazy", in which case, however, the class name information would be lost.

--
Shugo Maeda