Ruby trunk - Feature #708

Lazy Enumerator#select, Enumerator#map etc.

11/03/2008 06:54 PM - candlerb (Brian Candler)

Status:	Rejected	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	2.0.0	

Description

=begin
There are a number of methods in Enumerable which build an Array of results from the entire collection - e.g. map, select, take etc.

I propose that the Enumerator class have its own implementations of these methods, which return another Enumerator. Enumerators can then be chained:

seq.to_enum.map { ... }.select { ... }.take(...).each { |x| puts x }

This runs "horizontally": that is, each element is processed left to right. No intermediate arrays are created, and it works happily with infinite sequences.

There are precendents for SomeClass#select behaving differently to Enumerable#select. For example, Hash#select now returns a Hash. So I believe it would be reasonable for Enumerator to return another Enumerator.

You can then choose between array-building or lazy evaluation, depending on whether there is an Enumerator in the chain. Of course, the last Enumerator has to be turned into something useful, e.g. by calling to_a or each { ... }.

```
# Normal
res = (1..1_000_000).map { |x| x * 2 }.take(100)
# Lazy
res = (1..1_000_000).to_enum.map { |x| x * 2 }.take(100).to_a
```

I have attached a simple implementation of this for select, map, take and a new method skip. There are further methods like take_while, zip and so on which would also need to be implemented.
=end

Related issues:

Related to Ruby trunk - Feature #4653: [PATCH 1/1] new method Enumerable#rude...	**Rejected**	**05/08/2011**
Related to Ruby trunk - Feature #4890: Enumerable#lazy	**Closed**	**06/16/2011**

History

#1 - 11/29/2008 04:29 PM - ko1 (Koichi Sasada)

- Assignee set to matz (Yukihiro Matsumoto)

=begin

=end

#2 - 09/14/2010 04:51 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#3 - 11/02/2010 12:07 AM - rogerdpack (Roger Pack)

=begin
another option might be to add new methods called "e_select" or what not, to avoid changing current functionality.
=end

#4 - 02/08/2012 03:09 AM - kosaki (Motohiro KOSAKI)

Can anyone take a feedback? If nothing, I have to close this ticket sadly.

#5 - 02/08/2012 03:37 AM - trans (Thomas Sawyer)

There may be easy "first edition" solution to this. Facets has Denunumerable/Denumerator and defer.

- https://github.com/rubyworks/facets/blob/master/lib/core/facets/denumerable.rb
- https://github.com/rubyworks/facets/blob/master/lib/core/facets/enumerable/defer.rb

Use this as starting point. Modify API as needed. Could start out as standard library until someone has time to translate to C.

#6 - 02/08/2012 04:22 AM - mame (Yusuke Endoh)

- Priority changed from 3 to Normal

Hello,

I think no one doubt if this feature is useful and actually needed.
In fact, there are some proposals for the same (or similar) motivation.

- #4653 (focuses only map ?)
- #4890 (a new class Enumerable::Lazy)
- #5663 (focuses only select+map ?)

Especially, #5663 is recently discussed. So it is too early to close
this ticket.

Once, I also created the similar proof-of-concept library [1].
It provides some methods like Enumerable#mapper, #selector, etc., which
are Enumerator-version of corresponding Enumerable methods.
Matz said in [2] that it can be accepted except the name convention
(*er).

[1] http://mamememo.blogspot.com/2009/11/enumerablerrb-enumerable-lazy-version.html
[2] http://d.hatena.ne.jp/ku-ma-me/20091111/p2 (sorry, in Japanese)

So it might be good to suggest another name convention. In #4653, some
convention is proposed.

- mapper
- mapping
- mapL
- map_lz
- lazy_map
- enum_map
- map_enum

Matz himself suggested enum_* (enum_map, enum_select, ...), though I
don't like it because it is too long. I don't know if matz still like
it.

--
Yusuke Endoh mame@tsg.ne.jp

#7 - 02/08/2012 11:23 AM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:42411] [ruby-trunk - Feature #708] Lazy Enumerator#select, Enumerator#map etc."
on Wed, 8 Feb 2012 04:22:24 +0900, Yusuke Endoh mame@tsg.ne.jp writes:

|I think no one doubt if this feature is useful and actually needed.
|In fact, there are some proposals for the same (or similar) motivation.
|
| - #4653 (focuses only map ?)
| - #4890 (a new class Enumerable::Lazy)
| - #5663 (focuses only select+map ?)
|
|Especially, #5663 is recently discussed. So it is too early to close
|this ticket.

I vote for #4890.

```
                        matz.
```

#8 - 02/08/2012 11:53 AM - trans (Thomas Sawyer)

Konnichiwa matz,

Which term do you prefer for the method #lazy or #defer ?

Also organization (regardless of actual names), there is Enumerable namespace, e.g.

```
module Enumerable
  class Deferred < Enumerator
```

Or toplevel namespace, e.g.

```
class Denumerator < Enumerator
```

And note that previously mentioned Denumerable module allows option of mixin independent of Enumerable. If is organized like:

```
module Denumerable
   ...
end
```

```
class Denumerator
  include Denumerable
end
```

If you prefer "lazy" term I supposed the names for these would be "LazyEnumerable" and "LazyEnumerator" -- b/c I don't think "Lazierable" and "Lazierator" are going to cut it ;-)

#9 - 02/13/2012 08:11 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Rejected

2012/2/8 Yukihiro Matsumoto matz@ruby-lang.org:

> I vote for #4890.

Thank you for your opinion.
I'll close those tickets except #4890.

--
Yusuke Endoh mame@tsg.ne.jp

Files

lazy_enum.rb	1.75 KB	11/03/2008	candlerb (Brian Candler)