Ruby - Feature #8572

Fiber should be a Enumerable

06/28/2013 09:16 AM - mattn (Yasuhiro Matsumoto)

Status:	Closed	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	
Target version:	2.6	

Description

I'm thinking that Fiber should be a Enumerable. it's easy and reasonable to implement.

https://gist.github.com/mattn/5874949

History

#1 - 08/06/2013 11:48 AM - matz (Yukihiro Matsumoto)

I think this is a good idea, and since it's a mere addition, we don't have to worry about compatibility.

Matz.

#2 - 08/09/2013 07:45 PM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)

Use case?

#3 - 09/02/2013 02:35 PM - mattn (Yasuhiro Matsumoto)

ko1 (Koichi Sasada) wrote:

> Use case?

```
f = Fiber.new {
  Fiber.yield 3
  Fiber.yield 2
  Fiber.yield 1
  Fiber.yield 0
}

f.each do |x|
  puts x
end
```

For example, I want to do each for Fiber. But it can't

#4 - 09/03/2013 02:13 PM - zzak (zzak _)

You could always included Enumerable in your class that implements Fiber and define #each

#5 - 09/03/2013 02:35 PM - zzak (zzak _)

Sorry, I missed your gist orz

#6 - 09/03/2013 03:52 PM - knu (Akinori MUSHA)

We have Generator in 1.8 and Enumerator in 1.9+ that work exactly the
same as the given example.

```
    # In ruby 1.8, require 'generator' and call Generator.new instead
    enum = Enumerator.new { |g|
      g.yield 3
      g.yield 2
      g.yield 1
      g.yield 0
    }

    # internal style
```

```
    enum.each do |x|
      puts x
    end

    # external style
    while enum.next?
      puts enum.next
    end
```

One of the reasons why Fiber was originally introduced was to reimplement Generator of ruby 1.8 in better and faster way. Using Fiber as implementation technique, Generator was successfully integrated into Enumerator in Ruby 1.9.

So, Fiber is there as a low-level API for implementing Enumerator in the first place. It does not make much sense to me to add a non-primitive feature to Fiber just to make it work like Enumerator.

I think we need a real use case that would explain why Fiber should be used that way.

#7 - 09/09/2013 05:29 PM - mattn (Yasuhiro Matsumoto)

I think most of use case of using Fiber is something like getting stream data or catching-up draining entities. Maybe we won't use Fiber for getting limited resources. For example, If Fiber has each, I guess that we will be possible to write twitter client elegant. :)

```
    f = Fiber.new {
      # getting twitter status
      loop {
        Fiber.yield tweet_status
      }
    }

    f.each do |x|
      puts "#{x.screen_name}: #{x.text}"
    end
```

#8 - 09/09/2013 05:31 PM - matz (Yukihiro Matsumoto)

+1

Matz.

#9 - 09/09/2013 05:53 PM - ko1 (Koichi Sasada)

Why you don't use Enumerator?

```
class TS
  attr_accessor :screen_name, :text
  def initialize
    @screen_name = "foo"
    @text = "sample"
  end
end

f = Enumerator.new {|g|
  # getting twitter status
  loop {
    g.yield TS.new
  }
}

f.each do |x|
  puts "#{x.screen_name}: #{x.text}"
end
```

// SASADA Koichi at atdot dot net

#10 - 09/09/2013 08:30 PM - mattn (Yasuhiro Matsumoto)

> Why you don't use Enumerator?

Ah, make sense.

#11 - 09/30/2013 08:25 PM - ko1 (Koichi Sasada)

- Category set to core

- Status changed from Open to Feedback

- Target version set to 2.6

Can I close it?

#12 - 03/14/2014 12:08 AM - mattn (Yasuhiro Matsumoto)

Koichi Sasada wrote:

> Can I close it?

Sorry for delay. Close this please.

#13 - 03/14/2014 12:44 AM - nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Closed