Please add a thread-safe each() method to the Queue and SizedQueue classes which are provided by the "thread" standard library.

Also mix-in the Enumerable module into those classes so we can use map/inject/etc.

Thanks for your consideration.

---

History

#1 - 04/27/2011 01:06 PM - headius (Charles Nutter)

At first I thought this would be a good idea. But then I realized that #each is, in every case I can think of, a non-mutating operation. Since I assume you meant for Queue#each to pop all elements off the queue, this would be the first example I know of a mutating #each.

Also, what happens when the queue is empty? Does it wait for another element, or does it end the iteration?

The behavior of #each over a queue seems fuzzy to me, and without a clear specification of what you want I don't see a path forward.

#2 - 04/29/2011 06:50 AM - drbrain (Eric Hodel)

I don't think it is appropriate to include Enumerable. Too many methods from Enumerable seem inappropriate (or behavior would be application-specific). For example, should #find block, or only return items that match in the Queue? Should #reject remove items from the Queue, or only filter?

I can see differing needs depending on if the Queue has a single consumer or multiple consumers.

#3 - 07/27/2011 03:21 AM - sunaku (Suraj Kurapati)

Perhaps this code example can better illustrate my request:

```ruby
class Queue
  def each(&block)
    temporary_copy = @internal_queue_lock.synchronize do
      @internal_item_array.dup
    end
    temporary_copy.each(&block)
  end
end
```

Charles Nutter wrote:

At first I thought this would be a good idea. But then I realized that #each is, in every case I can think of, a non-mutating operation.

Yes, that was my intention: read-only iteration of the queue.

Since I assume you meant for Queue#each to pop all elements off the queue, this would be the first example I know of a mutating #each.
Sorry if my request was unclear, but that is not what I requested.

Also, what happens when the queue is empty? Does it wait for another element, or does it end the iteration?

It should be non-blocking. Simply iterate over the items that are currently in the queue.

If the method was blocking, it would never finish, because we would never reach the "end" of the queue.

The behavior of #each over a queue seems fuzzy to me, and without a clear specification of what you want I don't see a path forward.

I hope my responses above clarify this feature request.

Eric Hodel wrote:

I don't think it is appropriate to include Enumerable. Too many methods from Enumerable seem inappropriate (or behavior would be application-specific). For example, should #find block, or only return items that match in the Queue?

Since Enumerable relies on #each(), and the Queue#each method I'm requesting is non-blocking, all of the non-destructive Enumerable methods (like #find, #select, #map) should work as we normally expect them to.

Should #reject remove items from the Queue, or only filter?

#reject should filter (read-only). #reject! should remove items.

I can see differing needs depending on if the Queue has a single consumer or multiple consumers.

Hmm, shouldn't the thread-safe aspect of Queue take care of that?

#4 - 07/27/2011 08:34 AM - drbrain (Eric Hodel)
There are no destructive methods in Enumerable.

Most times I use a Queue I am producing and consuming items at the same time so this would not be as useful for me since it works on a copy.

If I were to finish production of items or work with partial results this would be useful.

#5 - 07/27/2011 06:53 PM - regularfry (Alex Young)
On 27/07/11 00:34, Eric Hodel wrote:

Issue #4589 has been updated by Eric Hodel.

There are no destructive methods in Enumerable.

Enumerable gets mixed into IO, and #each on a Socket is destructive (of the buffer contents). In that sense there is precedent for this sort of behaviour.

Alex

#6 - 03/25/2012 03:20 PM - mame (Yusuke Endoh)
- Status changed from Open to Assigned
- Assignee set to ko1 (Koichi Sasada)

#7 - 10/30/2012 08:43 AM - ko1 (Koichi Sasada)
- Status changed from Assigned to Feedback
Professional's comments are very welcome. I can't determine if it should be worth or not.

#8 - 01/31/2017 09:10 AM - ko1 (Koichi Sasada)
- Status changed from Feedback to Closed

No discussion.