Ruby master - Feature #7226

Add Set#join method as a shortcut for to_a.join

10/28/2012 05:38 AM - nathan.f77 (Nathan Broadbent)

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

Description

I was surprised that Set.new.join gives me a NoMethodError.
This patch that adds a #join method to Set, which is a shortcut for to_a.join.

Related issues:

Related to Ruby master - Bug #1893: Recursive Enumerable#join is surprising	**Closed**	**08/06/2009**
Is duplicate of Ruby master - Feature #5970: Add Enumerable#join with same se...	**Assigned**	

History

#1 - 10/28/2012 05:45 AM - nathan.f77 (Nathan Broadbent)

- File add_join_to_set.patch added

Sorry, previous patch contained a typo. Here's an updated patch.

#2 - 11/07/2012 06:28 AM - ayumin (Ayumu AIZAWA)

- Assignee set to matz (Yukihiro Matsumoto)

- Target version set to 2.6

#3 - 11/07/2012 09:56 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

+1. I was about to create the same feature request some weeks ago when I tried to join a set but was too lazy to do so :)

#4 - 05/27/2014 03:18 AM - naruse (Yui NARUSE)

- Related to Bug #1893: Recursive Enumerable#join is surprising added

#5 - 05/27/2014 03:30 AM - nobu (Nobuyoshi Nakada)

- Subject changed from Added #join method as a shortcut for to_a.join to Add Set#join method as a shortcut for to_a.join

- Status changed from Open to Assigned

- Assignee changed from matz (Yukihiro Matsumoto) to knu (Akinori MUSHA)

#6 - 05/27/2014 04:08 AM - knu (Akinori MUSHA)

- Status changed from Assigned to Rejected

The proposed implementation is far from efficient, collecting all elements into a temporary array only for calling Array#join.

It wouldn't be worth having it unless it went something like this: inject(nil) { |s, o| s.nil? ? "#{o}" : s << "#{sep}#{o}" } || ''

In any case, the method would better be implemented in Enumerable rather than in Set, because it would not be specific to Set at all but apply to any class that implements each, and Set is an unordered collection you shouldn't expect to have #join in the first place.

#7 - 05/27/2014 10:34 AM - duerst (Martin Dürst)

Akinori MUSHA wrote:

> The proposed implementation is far from efficient, collecting all elements into a temporary array only for calling Array#join.
>
> It wouldn't be worth having it unless it went something like this: inject(nil) { |s, o| s.nil? ? "#{o}" : s << "#{sep}#{o}" } || ''

I tried some very simple cases, and didn't see much of a difference. It's often better to start with a simple implementation and make it more complicated if additional performance is really needed.

> In any case, the method would better be implemented in Enumerable rather than in Set,

I agree.

#8 - 05/27/2014 12:28 PM - knu (Akinori MUSHA)

Martin Dürst wrote:

> I tried some very simple cases, and didn't see much of a difference. It's often better to start with a simple implementation and make it more complicated if additional performance is really needed.

If .to_a.join is what you want then you can and should live with it.

I implied by the phrase "far from efficient" that it would be against our expectation for #join to consume unnecessary time and space. What .to_a.join would do is to iterate to accumulate to iterate to accumulate, which certainly involves unnecessary complexity.

> > In any case, the method would better be implemented in Enumerable rather than in Set,
>
> I agree.

Enumerable represents a stream, #join is a stream friendly operation by nature, and I believe no .to_a should take place there.

With all being said, Enumerable#join has a history of once being added and then removed later as seen in #1893.
We must work out a way to add it again, resolving the compatibility issue with Array#join somehow.

Files

add_join_to_set.patch	788 Bytes	10/28/2012	nathan.f77 (Nathan Broadbent)
add_join_to_set.patch	756 Bytes	10/28/2012	nathan.f77 (Nathan Broadbent)