Ruby - Bug #12437

Is it "legal" to call collect! in class initializer?

05/30/2016 10:43 AM - vo.x (Vit Ondruch)

Status:	Closed		
Priority:	Normal		
Assignee:	knu (Akinori MUSHA)		
Target version:			
ruby -v:	ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-linux]	**Backport:**	2.1: DONTNEED, 2.2: DONTNEED, 2.3: REQUIRED

Description

Is there any reason the following script should not work?

```
#! /usr/bin/ruby

require 'set'

class Categories < Set

  def initialize(categories=[])
    categories.collect! { |category| category } if categories
    super categories
  end

end

categories = Categories.new()
categories += [1, 2, 3]
p categories

categories2 = Categories.new(categories)
p categories2
```

It fails with stack level too deep (SystemStackError) error and this regression seems to be introduced by r52591.

For details, please take a look at original issue reported here: https://bugzilla.redhat.com/show_bug.cgi?id=1308057

Associated revisions

Revision 0fcac8f1a35b73713849db323c206fb4f9aeda5a - 10/21/2017 05:03 PM - Akinori MUSHA

Avoid use of self.class.new(self) in Set#collect!

That prevents infinite recursion when a subclass of Set uses
collect! in its constructor.

This should fix [Bug #12437].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60317 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 0fcac8f1 - 10/21/2017 05:03 PM - Akinori MUSHA

Avoid use of self.class.new(self) in Set#collect!

That prevents infinite recursion when a subclass of Set uses
collect! in its constructor.

This should fix [Bug #12437].

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@60317 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 05/30/2016 02:16 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Assigned

- Assignee set to knu (Akinori MUSHA)

- Backport changed from 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN to 2.1: DONTNEED, 2.2: DONTNEED, 2.3: REQUIRED

#2 - 06/01/2016 12:05 PM - prijutme4ty (Ilya Vorontsov)

Second call to categories.collect! works over Categories object. Set#collect! tries to create a new set. Categories.collect! in turn tries to create new Categories.

```
s=Set.new([1,2,3])
s.object_id # => 7618880
s.collect!{|x| x**2 }
s.object_id # => 7618880 (the same)
s # => #<Set: {1, 4, 9}>

class Categories < Set; end
c = Categories.new([1,2,3])
c.object_id # => 7508680
c.collect!{|x| x**2 }
c.object_id # => 7508680 (the same)
c # => #<Categories: {1, 4, 9}>
```

Problem is that Set#collect! calls #initialize (probably to maintain some Set properties):

```
class Categories < Set
  def initialize(*args, &block)
    puts "call to #initialize"
    super
  end
end

c = Categories.new([1,2,3])
# => call to #initalize
c.collect!{|x| x ** 2 }
# => call to #initalize
```

More precisely, Set#collect! creates new Set and then replaces its content with Set#replace. So each time you call a #collect! etc, you create a new object with your customized constructor which recursively calls #collect and .new. Not a simply-to-fix sort of bug.

#3 - 06/19/2016 06:54 PM - valtri (František Dvořák)

- File collect-workaround.rb added

Actually it could be fixed by reverting this patch:

https://svn.ruby-lang.org/cgi-bin/viewvc.cgi?view=revision&revision=52591

Or it is possible to worked around that in the "Category" class by using custom collect method with the old code.

So the question is, how it is from ruby language point of view?

The revision 52591 is announced as micro-optimization, so it could be safe to remove it...

#4 - 10/21/2017 05:03 PM - knu (Akinori MUSHA)

- Status changed from Assigned to Closed

Applied in changeset trunk|r60317.

Avoid use of self.class.new(self) in Set#collect!

That prevents infinite recursion when a subclass of Set uses
collect! in its constructor.

This should fix [Bug #12437].

Files

collect-workaround.rb	452 Bytes	06/19/2016	valtri (František Dvořák)