Ruby master - Bug #14266

Set#clone(freeze: false) makes frozen internal hash

01/02/2018 01:24 AM - znz (Kazuhiro NISHIYAMA)

Status: Closed
Priority: Normal
Assignee: knu (Akinori MUSHA)

Description

% irb -r irb/completion --simple-prompt
>> require 'set'
=> true
>> set=Set[].freeze.clone(freeze: false)
=> #<Set: {}>
>> set.frozen?
=> false
>> set.instance_variable_get(:@hash).frozen?
=> true

In Set#initialize_clone, clone hash without freeze keyword argument.
But I think there is no easy way how to know freeze keyword argument value in initialize_clone.

    # Clone internal hash.
    def initialize_clone(orig)
      super
      @hash = orig.instance_variable_get(:@hash).clone
    end

Related issues:
Related to Ruby master - Feature #16129: Call initialize_clone with freeze: f... Closed

Associated revisions

Revision 04eb7c7e - 01/04/2020 04:13 AM - jeremyevans (Jeremy Evans)
Call initialize_clone with freeze: false if clone called with freeze: false
This makes it possible to initialize_clone to correctly not freeze internal state if the freeze: false keyword is passed to clone.

If clone is called with freeze: true or no keyword, do not pass a second argument to initialize_clone to keep backwards compatibility.

This makes it so that external libraries that override initialize_clone but do not support the freeze keyword will fail with ArgumentError if passing freeze: false to clone. I think that is better than the current behavior, which succeeds but results in an unfrozen object with frozen internals.

Fix related issues in set and delegate in stdlib.

Fixes [Bug #14266]

History

#1 - 01/04/2018 03:52 AM - jeremyevans0 (Jeremy Evans)
I see two possible ways to fix this:

1) Switch to overriding clone instead of initialize_clone in such cases.

2) Make clone(freeze: false) call initialize_clone(freeze: false), but have clone otherwise call initialize_clone without a keyword argument. Make Object#initialize_clone accept and ignore the freeze keyword. This way, if you override initialize_clone and don't have it accept the freeze keyword, clone(freeze: false) will raise an ArgumentError. That's probably better than returning an unfrozen object with frozen instance variables.
Call initialize_clone with freeze: false if clone called with freeze: false

This makes it possible to initialize_clone to correctly not freeze internal state if the freeze: false keyword is passed to clone.

If clone is called with freeze: true or no keyword, do not pass a second argument to initialize_clone to keep backwards compatibility.

This makes it so that external libraries that override initialize_clone but do not support the freeze keyword will fail with ArgumentError if passing freeze: false to clone. I think that is better than the current behavior, which succeeds but results in an unfrozen object with frozen internals.

Fix related issues in set and delegate in stdlib.

Fixes [Bug #14266]