#deep_clone and #deep_dup for Objects

03/21/2011 07:23 PM - wardrop (Tom Wardrop)

Status: Assigned
Priority: Normal
Assignee: matz (Yukihiro Matsumoto)
Target version:

Description

```ruby
=end
There's often a need to do a deep clone of an object, especially of Hash/Array trees. The typical work around to the lack of this functionality is to Marshall and then Unmarshall (e.g. Marshal::load(Marshal::dump(self)) ), which incurs more overhead than it probably should, and is not very semantic. My suggestion is to either provide #deep_clone and #deep_dup methods on the Object class, or to at least provide equivalent functionality for Hashes and Arrays, such as possibly a #deep_merge method for Hash. The exact implantation is not a large concern of mine; I'll let the experts determine the best method of achieving the desired outcome.
=end
```

History

#1 - 03/08/2012 06:35 PM - wardrop (Tom Wardrop)
No one wants to add anything?

#2 - 03/08/2012 07:09 PM - shyouhei (Shyouhei Urabe)
Some thoughts

- #deep_merge is a Rails method. If you only need Arrays + Hashes to be deep_dup'able, chances are they also should go into Rails.
- It is not always obvious what a "deep" copy is. For instance it is very hard to define one for a Proc instance.
- Recursive duplication may not be that simple to implement than you imagine. For instance an Array can contain itself: r = [].tap {|r| r << r } How do you copy it deeply?

#3 - 03/09/2012 02:14 AM - marcandre (Marc-Andre Lafortune)
Hi,

Shyouhei Urabe wrote:

- It is not always obvious what a "deep" copy is. For instance it is very hard to define one for a Proc instance.

Agreed, but since Procs do no "contain" anything, I'd say Proc#deep_dup == Proc#dup (except that it would call deep_dup on any instance variables, I imagine).

- Recursive duplication may not be that simple to implement than you imagine. For instance an Array can contain itself: r = [].tap {|r| r << r } How do you copy it deeply?

That shouldn't be too hard, we simply maintain a hash with the ids of objects being cloned as keys and with the corresponding new copies as values. I would have fun implementing it. Note that Marshal::load(Marshal::dump(r)) works for recursive arrays and so does YAML serialization.

I can attest that the question comes up from time to time on StackOverflow, so it is something quite a bit of people wonder about.

#4 - 03/10/2012 12:02 AM - kstephens (Kurt Stephens)
Marc-Andre Lafortune wrote:

Shyouhei Urabe wrote:

- It is not always obvious what a "deep" copy is. For instance it is very hard to define one for a Proc instance.
- Recursive duplication may not be that simple to implement than you imagine. For instance an Array can contain itself: r = [].tap {|r| r << r } How do you copy it deeply?

That shouldn't be too hard, we simply maintain a hash with the ids of objects being cloned as keys and with the corresponding new copies as values. I would have fun implementing it. Note that Marshal::load(Marshal::dump(r)) works for recursive arrays and so does YAML serialization.

See [https://github.com/kstephens/red_steak/blob/master/lib/red_steak/copier.rb](https://github.com/kstephens/red_steak/blob/master/lib/red_steak/copier.rb)
The problem is how to control how "deep" a copier should go, which objects need to have identity maintained and which objects are simply containers. The programmer needs control via some protocol with the copier.

#5 - 03/25/2012 03:15 PM - mame (Yusuke Endoh)
- Status changed from Open to Assigned
- Assignee set to matz (Yukihiro Matsumoto)

#6 - 11/20/2012 09:35 PM - mame (Yusuke Endoh)
- Target version set to 2.6

#7 - 12/25/2017 06:14 PM - naruse (Yui NARUSE)
- Target version deleted (2.6)