Ruby trunk - Bug #11929

No programatic way to check ability to dup/clone an object

12/30/2015 04:30 PM - lkdjiin (xavier nayrac)

Status:	Closed		
Priority:	Normal		
Assignee:	matz (Yukihiro Matsumoto)		
Target version:			
ruby -v:	ruby 2.3.0dev (2015-12-06 trunk 52904) [i686-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

We can't dup a Fixnum, and it's ok. But I'm wondering «why is Fixnum saying it can dup?»

```
1.respond_to?(:dup) #=> true
```

Don't you think that the dup method should be undefined in the class Fixnum?

Currently I can do class Fixnum; undef :dup; end, but that should be in the core Ruby, isn't it?

Related issues:

Related to Ruby trunk - Feature #12979: Avoid exception for #dup on Integer (...	**Closed**

History

#1 - 12/31/2015 04:24 AM - 0x0dea (D.E. Akers)

This demonstration should clarify the observed behavior. Symbol, Fixnum, and indeed every other numeric class inherit their #dup from Kernel, whose implementation does a sanity check before proceeding, thus obviating the need to remove the method from the classes for which it might otherwise make sense to do so.

#2 - 10/20/2016 04:03 AM - MikeVastola (Mike Vastola)

- Subject changed from dup should be undefined in Fixnum to No programatic way to check ability to dup/clone an object

D.E., while you're technically not wrong, IMHO, the need to **undef** the method is only abated in the strictest of senses: the sanity check averts what might otherwise be a segfault or memory leak or some other aberration, and makes things 'safe', but it's neither straightforward nor intuitive for the programmer, nor consistent with the implementation of similar functions in ruby.

For instance, **freeze** doesn't throw any sort of similar error when you call it on an already frozen object, or even for the immediate (and therefore inherently immutable) objects you used in your demonstration.

I think the core issue Xavier is getting at here (and I took the liberty of updating the subject thusly; Xavier, if you don't feel it's a fair representation of the issue, please feel free to change it back) is that a select few object types in Ruby are immediate variables and therefore cannot be **dup**ed/**clone**d, yet there is no graceful/robust method of averting the error thrown by this sanity check when you attempt to **dup**/**clone** them. (In the source, both **rb_obj_dup** and **rb_obj_clone** if the object to be **dup**ed/**clone**d is a **rb_special_const_p**. As far as I can tell, this method nor anything equivalent is exposed in the ruby language, however.)

I literally can't imagine any scenario in which a dev, when, say, coding a class with the line:

```
return val.dup.freeze
```

.. really wants an **Exception** thrown when **val** happens to be de-facto un-**dup**-able. What they really want is:

```
return val.dup.freeze rescue val
```

To me it seems there are three possible solutions to this (of which any two can be implemented):

- As Xavier suggested, only define the functions for objects that will actually **dup**/**clone**. (This will implicitly allow **respond_to?** checks as well as not interfere with existing code that may attempt to **rescue** failed attempts -- that is unless **TypeError** was specified, and even that could be worked around with a bit more code.)
- Add additional methods to all objects named **dupable?** and **clonable?** that return values consistent with exactly what their names suggest.
- Have these methods function exactly the way **freeze** does: fail silently. If the object can't be **dup**ed/**clone**d because it's an immediate, **dup**/**clone** should return the object itself. (There shouldn't be any harm in doing so since nothing about the object can be changed in the first place.)

#3 - 10/20/2016 09:11 AM - duerst (Martin Dürst)

- Assignee set to matz (Yukihiro Matsumoto)

I clearly prefer the last proposal (fail silently).

That would make for a much more unified, streamlined protocol, avoiding needless exposition of internals. It would do exactly what dup (and clone) are described to do, namely (pretend to) return a shallow copy.

#4 - 10/20/2016 09:08 PM - Eregon (Benoit Daloze)

Martin Dürst wrote:

> I clearly prefer the last proposal (fail silently).
>
> That would make for a much more unified, streamlined protocol, avoiding needless exposition of internals. It would do exactly what dup (and clone) are described to do, namely (pretend to) return a shallow copy.

I very much agree.

#5 - 10/21/2016 09:04 AM - lkdjiin (xavier nayrac)

Mike, your last proposal is simple, beautiful and smart.

```
1.dup #=> 1
```

#6 - 11/25/2016 06:35 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Feedback

1. ability to be duped cannot be checked by respond_to?. It's simply wrong.
2. the "fail silently" proposal is attractive, but it should be a separate proposal (for the sake of bookkeeping).

Matz.

#7 - 11/25/2016 07:12 AM - duerst (Martin Dürst)

- Related to Feature #12979: Avoid exception for #dup on Integer (and similar cases) added

#8 - 11/25/2016 07:24 AM - matz (Yukihiro Matsumoto)

- Status changed from Feedback to Closed

The issue closed by #12979

Matz.