Ruby 1.8 - Bug #1962

Kernel#Array shouldn't call #to_a if it has been undef'd

08/20/2009 02:51 PM - oggy (George Ogata)

Status:	Assigned
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	
ruby -v:	ruby 1.8.6 (2009-08-19 patchlevel 384) [i686-darwin9.7.0]

Description

=begin
Applies to 1.8.*.

Kernel#Array is supposed to behave as follows:

- if the object has a #to_ary, call it and return the result
- else if the object has a #to_a, call it and return the result
- else return the object in a 1-element array

If #to_a has been undefined (with #undef_method), however, #Array will see the NULL entry in the method table and try to call it. This is silly. Practically speaking, this leads to spurious warnings ("default `to_a' will be obsolete") when calling #Array on an ActiveRecord AssociationProxy. This happens because AssociationProxy undefs all its methods (including #to_a, since Object#to_a is defined), in order to delegate everything to the object it wraps via #method_missing. #Array will thus find #to_a to be mapped to an undef'd method, and proceed to call it, which gets delegated to the wrapped object, which doesn't implement #to_a, so it falls back to Object#to_a -- hence the warning.

Aside from finding this behavior strange, I don't really see how ActiveRecord can work around this properly. I've attached a patch which makes #Array check if the #to_a it found is an undef'd method, and not call it if it is. This is consistent with 1.9.
=end

History

#1 - 10/28/2009 02:39 AM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

=begin

=end

Files

Array.patch	609 Bytes	08/20/2009	oggy (George Ogata)