Ruby - Feature #3916

Add flag to ruby to make warnings fatal.

10/07/2010 11:46 PM - docwhat (Christian Höltje)

Status:	Closed
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	2.6

Description

=begin
I have a feature request: Please add a flag to the ruby command line to make warnings fatal.

It would be really handy when debugging and testing to make warnings fatal.

It's not always obvious what bit of code is causing the code that has the warning to execute. Having a full exception style trace back is very handy.

In addition, if you're doing tests, it would be good to have it raise an exception if a warning is fired. See http://stackoverflow.com/questions/660737/can-you-ask-ruby-to-treat-warnings-as-errors for an example.

Thanks!
=end

History

#1 - 12/10/2010 03:58 AM - tenderlovemaking (Aaron Patterson)

=begin
I'm not sure this is necessary. For example, sprintf() will raise an exception if in debug mode:

irb(main):001:0> $-w = true
=> true
irb(main):002:0> "foo" % 10
(irb):2: warning: too many arguments for format string
=> "foo"
irb(main):003:0> $DEBUG = true
=> true
irb(main):004:0> "foo" % 10
Exception ArgumentError' at (irb):4 - too many arguments for format string Exception ArgumentError' at
/Users/apatterson/.local/lib/ruby/1.9.1/irb/workspace.rb:80 - too many arguments for format string
ArgumentError: too many arguments for format string
from (irb):4:in %' from (irb):4 from /Users/apatterson/.local/bin/irb:12:in '
irb(main):005:0>

Maybe we should just update code to raise an exception when in debug mode like sprintf() does.
=end

#2 - 03/18/2012 06:14 PM - nahi (Hiroshi Nakamura)

- Description updated

- Category set to core

- Assignee set to matz (Yukihiro Matsumoto)

#3 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#4 - 03/18/2012 11:05 PM - trans (Thomas Sawyer)

Hmm... I just wrote article related to this:

https://github.com/trans/trans.github.com/wiki/2012-02-25-setting-priorities-trumps-warnings

#5 - 03/20/2012 07:40 AM - drbrain (Eric Hodel)

Sometimes when I use Kernel#warn in my code it is a message the user may not be able to do anything about, such as a deprecation message in library "a" that is used in library "b" that has not yet been updated, but the user wishes to use. Turning these into exceptions would break this use of

warnings.

#6 - 03/20/2012 10:48 AM - trans (Thomas Sawyer)

@drbrain (Eric Hodel) Wouldn't using priorities and setting such a warning to a very low priority solve this?

#7 - 03/20/2012 12:52 PM - drbrain (Eric Hodel)

There are no priorities for warnings at present, so any code using warnings would need to be updated to take advantage of the feature, breaking backwards compatibility.

Existing uses of #warn could be set as "lowest priority" which makes the feature not very useful since only new code will opt in, and the exception will only be useful when ruby is run with -d, which is rare.

#8 - 03/21/2012 12:09 AM - trans (Thomas Sawyer)

It wouldn't break backward compatibility, but it would take time for libraries to adjust to take the most advantage of it. And yet, any library with active development will likely adjust very quickly.

I would expect existing uses of warn to be set at "nominal priority" -- just below the error threshold. Although I imagine one could make the case that we should insert one priority layer between the two which would help make them a little more useful with libraries that haven't yet adjusted.

#9 - 11/20/2012 09:31 PM - mame (Yusuke Endoh)

- Target version set to 2.6

#10 - 11/29/2017 08:41 AM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Closed

Now we can implement the feature by redefining Warning.warn.

Matz.