Ruby - Feature #18564

Add Exception#detailed_message

02/01/2022 07:49 PM - mame (Yusuke Endoh)

Status:	Closed
Priority:	Normal
Assignee:	mame (Yusuke Endoh)
Target version:	

Description

(This ticket is for recording the final spec of #18438)

Proposal

I would introduce a method Exception#detailed_message, and let the default error printer use it instead of Exception#message to create a error output.

```
class MyClass < StandardError
  def message = "my error!"
  def detailed_message(highlight: false, **opt)
    super + "\nThis is\nan additional\nmessage"
  end
end

raise MyClass

$ ./miniruby test.rb
test.rb:8:in `<main>': my error! (MyClass)
This is
an additional
message
```

Here is the implementation: https://github.com/ruby/ruby/pull/5516

Spec

Exception#detailed_message(highlight: false) calls #message and decorates the returned string. It may add the class name of exception and, when highlight keyword is true, some escape sequences for highlights.

```
e = RuntimeError.new("my error!")
p e.detailed_message                  #=> "my error! (RuntimeError)"
p e.detailed_message(highlight: true) #=> "\e[1mmy error! (\e[1;4mRuntimeError\e[m\e[1m)\e[m"
```

Previously, the default error printer and Exception#full_message called #message to get the error message, applied some processing (adding the error class name and adding escape sequences) to the string, and added backtrace. Now, they now use #detailed_message(highlight: Exception.to_tty?) instead of #message.

All keyword arguments passed to Exception#full_message are delegated to detailed_message.

Motivation

The primary motivation is a clean integration of did_you_mean and error_highlight gems.

At the present time, they overrides Exception#to_s to add their suggestions. However, there are some known problems in this approach:

- It may break some tests to check the result of Exception#to_s depending on whether the gems add suggestions or not.
- Some Ruby scripts re-raise an exception by raise e.class, e.message, e.backtrace, which makes the gems add their suggestion multiple times (currently, the gems ad-hocly check and avoid multiple addition).
- Sometimes a user needs to get the original message without their addition. For the sake, did_you_mean provides Exception#original_message, but the workaround is not very well known.

This proposal allows the gems to override Exception#detailed_message. Exception#to_s is kept as-is, so the above problems will no longer occur.

Also, the proposal allows a user to get a full_message without the suggestions by err.full_message(did_you_mean: false, error_highlight: false).

Here is a proof-of-concept patch for did_you_mean and error_highlight: https://gist.github.com/mame/2c34230f11237dc4af64510cb98acdd8 I'll create PRs for the gems after Exception#detailed_message is merged.

Cooperation needed

This change requires application monitoring services such as Sentry, DataDog, ScoutAPM, etc. They need to use Exception#detailed_message(highlight: false) instead of Exception#message to log the error messages after Ruby 3.2. Thankfully, @st0012 (Stan Lo) (the maintainer of Sentry's Ruby SDK) and @ivoanjo and @marcotc (the maintainers of Datadog's application monitoring gem) have agreed with this change.

https://bugs.ruby-lang.org/issues/18438#note-1
https://bugs.ruby-lang.org/issues/18438#note-9

@matz (Yukihiro Matsumoto) has already approved this proposal in #18438 . I'll merge my PR in a few days after some reviews.

History

#1 - 02/01/2022 08:06 PM - mame (Yusuke Endoh)

- Description updated

#2 - 11/14/2022 03:11 AM - mame (Yusuke Endoh)

- Status changed from Open to Closed

I think I have already merged the change. Closing.