Ruby master - Bug #12588

When an exception is re-raised in the "rescue" clause, the back trace does not contain the line in that clause

07/15/2016 08:02 PM - hasari (Hiro Asari)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>ruby 2.2.4p230 (2015-12-16 revision 53155) [x86_64-darwin14]</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN</td>
</tr>
</tbody>
</table>

Description

Given:

```
$ cat -n foo.rb
  1  def foo
  2    raise StandardError
  3  rescue StandardError => e
  4    raise e
  5  end
  6
  7  foo
```

one would reasonably expect to see line 4 to be in the back trace when this file is executed, but one does not.

```
$ ruby -v foo.rb
ruby 2.2.4p230 (2015-12-16 revision 53155) [x86_64-darwin14]
foo.rb:2:in `foo': StandardError (StandardError)
  from foo.rb:7:in `<main>'
```

History

#1 - 08/20/2016 02:37 AM - nobu (Nobuyoshi Nakada)
I think it's intentional, and don't see why it should contain a new line.

#2 - 09/07/2016 08:34 AM - matz (Yukihiro Matsumoto)
- Status changed from Open to Rejected

I don't see any "reasonable expectancy". Use-case? Probably you want to use Exception#cause?

Matz.

#3 - 09/26/2016 12:56 AM - hasari (Hiro Asari)
The current stacktrace is misleading. Without line 4 in it, the implication is that the exception was found on line 2, and was not caught by rescue. In this simple case, the stack trace should contain lines [4,7], not [2,7].

I am not sure how I can use Exception#cause. I tried:

```
$ cat -n foo.rb
  1  def foo
  2    raise StandardError
  3  rescue StandardError => e
  4    puts e.cause.class
  5    raise e.cause
  6  end
  7
  8  foo
$ ruby -v foo.rb
ruby 2.4.0dev (2016-09-21 trunk 56200) [x86_64-darwin15]
NilClass
foo.rb:5:in `raise': exception object expected (TypeError)
  from foo.rb:5:in `rescue in foo'
  from foo.rb:2:in `foo'
```
Hiro Asari wrote:

I am not sure how I can use Exception#cause.

Exception#cause makes sense when you raise another exception from inside of a rescue clause. Take a look at this example:

def foo
  raise 'raised in #foo'
end

def bar
  foo
  rescue
    raise 'raised in #bar'
end

def baz
  bar
  rescue => e
    p e  # => #<RuntimeError: raised in #bar>
    p e.cause  # => #<RuntimeError: raised in #foo>
end

baz