Ruby master - Feature #18145
Rescue by nested exception
09/02/2021 10:18 PM - svoop (Sven Schwyn)

Status: Closed
Priority: Normal
Assignee:
Target version:

Description
The introduction of Exception#cause helps a lot when debugging nested errors.

Same goes for wrapped errors. I'm not really sure whether such wrapped errors are an advisable pattern to begin with, feel free to comment on this, but I've used it in a couple of vendored gems dealing with payment providers.

Here's some simplified code to illustrate. The Payment class deals with all the gory things such as authorization, coercion or API quirks. A simplified version might look like this:

```ruby
require 'rest_client'

module Provider
  class FindError < StandardError; end
  class Payment
    attr_reader :id, :amount
    private_class_method :new
    def initialize(id, amount)
      @id, @amount = id, amount
    end

    def self.find(id)
      response = RestClient.get('https://api.provider.com/payments', params: { id: id })
      body = JSON.parse(response.body)
      new(id, body.fetch('amount'))
      rescue FindError
        raise FindError
      end
    end
  end
end

You can easily rescue from anything going wrong when loading a payment:

```begin
  Provider::Payment.find(123)
rescue FindError
  ...
end```

However, you might want to rescue differently for some specific causes (e.g. not found) but not for others (e.g. timeout):

```begin
  Provider::Payment.find(123)
rescue FindError => error
  if error.cause.instance_of? RestClient::NotFound
    ...
  else
    ...
end```

How about allowing to rescue by nested exception with a syntax like?
begin
  Provider::Payment.find(123)
rescue FindError & RestClient::NotFound
    ...
rescue FindError
    ...
end

History
#1 - 09/02/2021 10:39 PM - jeremyevans0 (Jeremy Evans)
svoop (Sven Schwyn) wrote:

    How about allowing to rescue by nested exception with a syntax like?

begin
  Provider::Payment.find(123)
rescue FindError & RestClient::NotFound
    ...
rescue FindError
    ...
end

That particular syntax is already valid. It just depends on FindError.&(RestClient::NotFound) returning a valid exception class.

You can already implement this support yourself:

def Exception.&(cause)
  klass = self
  Class.new do
    define_singleton_method(:===) do |exc|
      klass === exc && cause === exc.cause
    end
  end
end

#2 - 09/02/2021 11:04 PM - svoop (Sven Schwyn)

jeremyevans0 (Jeremy Evans) Nice one, thanks for the hint! I'll add this to a ProviderError and inherit from there. I guess this use case is too specific to add this self.& to StandardError or even Exception upstream?

#3 - 09/03/2021 01:39 AM - jeremyevans0 (Jeremy Evans)
svoop (Sven Schwyn) wrote in #note-2:

jeremyevans0 (Jeremy Evans) Nice one, thanks for the hint! I'll add this to a ProviderError and inherit from there. I guess this use case is too specific to add this self.& to StandardError or even Exception upstream?

That's not for me to decide, but I would be against it. The need seems very limited to me, so the benefit seems low. Also, the approach requires allocating a new class for each exception where it needs to be tested, so it's suboptimal from a performance standpoint. A better performing approach would cache the equivalent of the FindError & RestClient::NotFound expression into a constant, and use that in the rescue clause.

#4 - 09/03/2021 09:27 AM - byroot (Jean Boussier)

I'll add that this isn't really a good design because it leaks the underlying implementation. The caller of Provider::Payment shouldn't have to know that RestClient was used.

The more elegant way to do this would be:

module Provider
  class FindError < StandardError; end
  class NotFoundError < FindError; end
end

class Payment
  attr_reader :id, :amount

private_class_method :new

  def initialize(id, amount)
    @id, @amount = id, amount
  end
end
def self.find(id)
    response = RestClient.get('https://api.provider.com/payments', params: { id: id })
    body = JSON.parse(response.body)
    new(id, body.fetch('amount'))
rescue RestClient::NotFound
    raiseNotFoundError
rescue
    raise FindError
end
end

begin
  Provider::Payment.find(123)
rescue NotFoundError
  ...
rescue FindError
  ...
end

#5 - 09/03/2021 09:03 PM - swoop (Sven Schwyn)
Both valid points, thanks for your suggestions... this issue can be closed IMO.

#6 - 09/05/2021 08:30 AM - duerst (Martin Dürst)
- Status changed from Open to Closed

swoop (Sven Schwyn) wrote in #note-5:
  Both valid points, thanks for your suggestions... this issue can be closed IMO.

Closed at request of OP.