Ruby trunk - Feature #11717

Object#trap -- pass object to block and return result

11/19/2015 08:50 PM - zverok (Victor Shepelev)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

Object#trap can be thought as useful counterpart for Object#tap: like tap, it passes object to the block; **unlike** tap, it returns results of the block, not object itself.

Rationale

#trap could allow to gracefully chain processing of objects, which isn't Enumerable, therefore enforcing "functional" style in Ruby (which considered good).

Use case

```
# Assume we grab some resource from web:
SomeWebClient.get('http://some/url', param1: 'value1', param2: 'value2').body

# And now, the body of response is JSON, and we want it parsed. How to express it?

# Option 1: wrap:
JSON.parse(SomeWebClient.get('http://some/url', param1: 'value1', param2: 'value2').body)
# Downside: messy parenthesis, and need to jump back and forth to understand the meaning

# Option 2: intermediate variable:
s = SomeWebClient.get('http://some/url', param1: 'value1', param2: 'value2').body
JSON.parse(s)
# Downside: intermediate variable is not elegant

# Option 3: monkey-patch (or refine)
SomeWebClient.get('http://some/url', param1: 'value1', param2: 'value2').body.from_json
# Downside: monkey-patching is a last resort; also, your classes should be already patched when yo
u stuck with this case

# Option 4 (proposed): trap
SomeWebClient.get('http://some/url', param1: 'value1', param2: 'value2').body.
  trap{|s| JSON.parse(s)} # => parsed JSON
```

And when you are thinking with code, experimenting with code (especially in irb, but in editor too), only last option is "natural" river of thoughts: do this, then do that (extract data from web, then parse it).

Naming

- it is similar enough to tap;
- it is specific enough to not be used widely in some popular library (or isn't it?);
- mnemonic is "do something and trap (catch) the value".

WDYT?

Related issues:

Is duplicate of Ruby trunk - Feature #6721: Object#yield_self	**Closed**
Has duplicate Ruby trunk - Feature #12760: Optional block argument for `itself`	**Closed**
Has duplicate Ruby trunk - Feature #13172: Method that yields object to block...	**Closed**

History

#1 - 11/19/2015 10:49 PM - 0x0dea (D.E. Akers)

You're looking for #instance_eval:

```
'foo'.instance_eval { |obj| obj.size } # => 3
```

#2 - 11/19/2015 11:02 PM - zverok (Victor Shepelev)

Nope.

I'm aware of #instance_eval last 10 years or so (I even can recall times when #instance_exec were external library method, not part of the core).

Primary goal/usage of #instance_eval is to "dig inside". Primary goal/usage of #trap is to "apply next transormation". Even if #trap will be implemented as a simple alias of #instance_eval, it will have some usage/popularity outside of #instance_eval's domain. On my opinion only, of course.

#3 - 11/19/2015 11:08 PM - phluid61 (Matthew Kerwin)

Some related issues and historical readings:

- #10095 Object#as
  - from #6373 Object#self
- #6721 Object#yield_seld
- #6684 Object#do
- #7388 Object#embed

Clearly this is something the Ruby community wants. Just as clearly, it's something nobody can name.

#4 - 11/19/2015 11:12 PM - phluid61 (Matthew Kerwin)

Victor Shepelev wrote:

> **Naming**
>
> - it is similar enough to tap;
> - it is specific enough to not be used widely in some popular library (or isn't it?);
> - mnemonic is "do something and trap (catch) the value".
>
> WDYT?

"trap" already means "trap a signal", it comes from long-standing Unix terminology; see Signal#trap

#5 - 11/19/2015 11:41 PM - 0x0dea (D.E. Akers)

Victor Shepelev wrote:

> Even if #trap will be implemented as a simple alias of #instance_eval...

If you did in fact know that you were essentially requesting an alias for #instance_eval, this was a remarkably roundabout way to go about it.

#6 - 11/20/2015 04:18 PM - zverok (Victor Shepelev)

> "trap" already means "trap a signal", it comes from long-standing Unix terminology

Ooops. Completely forgot about this one :(

> Clearly this is something the Ruby community wants. Just as clearly, it's something nobody can name.

Yeah, can see it now.

Thinking further, I wonder if just Object#yield could be parsed correctly...

> If you did in fact know that you were essentially requesting an alias for #instance_eval, this was a remarkably roundabout way to go about it.

But hey. It is NOT, in fact:

```
class MyClass
  def with_instance_eval(filename)
    File.read(filename).instance_eval{|s| p [s, self]; parse(s)}
  end

  def with_trap(filename)
    File.read(filename).trap{|s| p [s, self]; parse(s)}
```

```
  end

  def parse(str)
    JSON.parse(str)
  end
end

puts "#trap:"
p MyClass.new.with_trap('test.json')

puts "#instance_eval:"
p MyClass.new.with_instance_eval('test.json')
```

Output:

```
#trap:
["{\"test\": 1}\n", #<MyClass:0x911f428>]
{"test"=>1}

#instance_eval:
["{\"test\": 1}\n", "{\"test\": 1}\n"]
trap.rb:in `block in with_instance_eval': undefined method `parse' for "{\"test\": 1}\n":String (NoMethodError
)
```

#7 - 11/20/2015 06:06 PM - nobu (Nobuyoshi Nakada)

- Is duplicate of Feature #6721: Object#yield_self added

#8 - 09/20/2016 12:55 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Feature #12760: Optional block argument for `itself` added

#9 - 01/31/2017 04:16 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Feature #13172: Method that yields object to block and returns result added

#10 - 05/01/2017 07:50 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset trunk|r58528.

---

object.c: Kernel#yield_self

- object.c (rb_obj_yield_self): new method which yields the receiver and returns the result. [ruby-core:46320] [Feature #6721]