Ruby master - Feature #11034
Nil Conditional
04/04/2015 06:54 PM - grzesiek (Grzegorz Bizon)

| Status:   | Closed |
| Priority: | Normal |
| Assignee: |        |
| Target version: |        |

Description
Hi everyone!

Some time ago I was thinking about Nil Conditional Operator in Ruby (??). This would be particularly useful to avoid frequent checking for nil, and should behave and look like Null Conditional Operator introduced in C# 6.0.

I was thinking about something like this (assume var is nil or doesn't exist):

```
var?.method1.method2(123, 345).method3 { |i| i == 1 } => nil
```

When var is nil or doesn't exist, code above should return nil or NilConditionalClass object instead of raising NoMethodError or NameError.

This can also work with methods (assume var exists):

```
var.method1?.method2(a, b) => nil
```

When var exists and can receive method1, but method1 returns nil - this shouldn return nil instead of raising `NoMethodError: undefined method method2` for nil:NilClass`

When var exists and is not nil, and can receive method1, and object returned by method1 can receive method2 this, of course should behave as expected (like version without ?? operator) and return value according to implementation of method2.

When var doesn't exist - this should raise NameError.

I tried to create gem for that ([https://github.com/grzesiek/nil-conditional](https://github.com/grzesiek/nil-conditional)) but from now on, only native implementation seems reasonable.

What do you think about that feature? Maybe it is already considered, but I couldn't find anything similar at Google/this issue tracker (in that case I'm sorry for duplicate).

Thanks in advance for feedback!

Kind regards,
Grzegorz

Related issues:
Related to Ruby master - Feature #1122: request for: Object#try  Rejected  02/07/2009
Related to Ruby master - Feature #11537: Introduce "Safe navigation operator"  Closed

History
#1 - 04/05/2015 02:28 AM - nobu (Nobuyoshi Nakada)
  - Description updated

A local variable written in code always exists.
I think this is a duplicate ticket but can't find the former tickets.

#2 - 04/05/2015 02:32 AM - nobu (Nobuyoshi Nakada)
IIRC, there were some discussions like:

- the condition is only whether nil or not, or whether responds to the method or not
- the result when the condition doesn't match will be the receiver itself or nil

03/26/2021 1/3
and others.

Also, I'm afraid that ?? would conflict with the current syntax.

#3 - 04/05/2015 05:55 AM - grzesiek (Grzegorz Bizon)

Hi Nobuyoshi, thanks for reply!

Yes, this Nil Conditional Operator has to introduce new syntax. That is why this cannot be achieved via gem. Currently - of course - double question mark ?? violates syntax (in some cases) or is treated as ternary operator. But this doesn't have to be ??.

I also agree that my version of nil conditional is too permissive and shouldn't return nil if variable does not exist. This was just less difficult to implement ;)

I simply believe that Nil Conditional Operator would be useful for every rubyist. In my opinion it is worth talking over, considering.

I will try to find related issue.

Kind regards,
Grzegorz

#4 - 04/05/2015 08:40 AM - recursive-madman (Recursive Madman)

This is similar to a pattern I have seen somewhere else:

```
var = 123.4
var.try.round #=> 123
var = nil
var.try.round #=> nil
```

Which doesn't introduce a new operator and can be implemented like this:

```ruby
class Object
  def try
    self
  end
end

class NilClass
  def try
    IgnoreCall.new
  end
end

class IgnoreCall
  def method_missing(*a)
    nil
  end
end
```

By the way, fun fact:

```
???????:?? #=> '?'
```

#5 - 04/06/2015 07:06 PM - grzesiek (Grzegorz Bizon)

@Recursive Madman - try is not native function. It is Active Support Core Extension.

Moreover implementation and behavior of try is not stable (is changing depending on Rails version), it will not work when creating train wrecks and is terribly ugly.

Of course - this can be, in some specific way, achieved by using gem (like nil_conditional gem), but, in my opinion, only native implementation will be reasonable enough. Only native solution for that problem will be stable, consistent, short and concise.

What do you think?

Kind regards,
Grzegorz

#6 - 04/10/2015 04:34 AM - akr (Akira Tanaka)

- Related to Feature #1122: request for: Object#try added

#7 - 04/14/2015 06:51 AM - grzesiek (Grzegorz Bizon)
Hi again!

I've made new attempt to create nil conditional operator ([https://github.com/nil-conditional](https://github.com/nil-conditional)). This is Ruby implementation, you can check it out here: [https://github.com/grzesiek/nil-conditional/blob/master/lib/nil_conditional.rb](https://github.com/grzesiek/nil-conditional/blob/master/lib/nil_conditional.rb).

It is simple, it encapsulates objects in NilConditional instance. Sample usage:

```ruby
def nil_test(str)
  str._?.concat("456")
end

nil_test("123")
=> "123456"

nil_test(nil)
=> #<NilConditional:0x007fc602016718 @object=nil>

nil_test(nil).nil?
=> true
```

What do you think about that? It seems (for me) that this is sufficient, and native implementation is not necessary with this approach.

Anyway, thanks for feedback!

Kind regards,
Grzegorz

---

#8 - 09/18/2015 09:58 AM - akr (Akira Tanaka)
- Related to Feature #11537: Introduce "Safe navigation operator" added

#9 - 10/21/2015 08:06 AM - matz (Yukihiro Matsumoto)
- Status changed from Open to Closed

We will introduce .? [#11537](https://github.com/nil-conditional/pull/11537) which address this proposal.

Matz.