# Ruby trunk - Feature #11643

## A new method on Hash to grab values out of nested hashes, failing gracefully

11/02/2015 02:04 AM - gkop (Gabe Kopley)

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

**Description**

(I posted this to the mailing list last year [0] and received no response, but am inspired to file an issue here based on the positive reception to https://bugs.ruby-lang.org/issues/11537 )

This comes up sometimes in Rails programming [1]:

params[:order] && params[:order][:shipping_info] && params[:order][:shipping_info][:country]

or

params[:order][:shipping_info][:country] rescue nil

or

params.fetch(:order, {}).fetch(:shipping_info, {}).fetch(:country, nil)

What if Hash gave us a method to accomplish this more concisely and semantically?

Eg.

params.traverse_nested_hashes_and_return_nil_if_a_key_isnt_found(:order, :shipping_info, :country)

Or to take a nice method name suggestion [2]:

params.dig(:order, :shipping_info, :country)

Another example solution is https://github.com/intridea/hashie#deepfetch (Although I don't like "fetch" in this method name since it doesn't and can't take a default value as an argument like Hash#fetch does)

What do you all think?

[0] https://groups.google.com/forum/#!topic/ruby-core-google/guleNgEJWcM

[1]
https://groups.google.com/d/msg/rubyonrails-core/bOkvcvS3t_A/QXLEXwt9ivAJ
https://stackoverflow.com/questions/1820451/ruby-style-how-to-check-whether-a-nested-hash-element-exists
https://stackoverflow.com/questions/19115838/how-do-i-use-the-fetch-method-for-nested-hash
https://stackoverflow.com/questions/10130726/ruby-access-multidimensional-hash-and-avoid-access-nil-object

[2] http://stackoverflow.com/a/1820492/283398

### Associated revisions

**Revision 29862685 - 11/09/2015 12:27 PM - nobu (Nobuyoshi Nakada)**

dig

- array.c (rb_ary_dig): new method Array#dig.
- hash.c (rb_hash_dig): new method Hash#dig.
- object.c (rb_obj_dig): dig in nested arrays/hashes. [Feature #11643]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@52504 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 52504 - 11/09/2015 12:27 PM - nobu (Nobuyoshi Nakada)**

dig

- array.c (rb_ary_dig): new method Array#dig.
- hash.c (rb_hash_dig): new method Hash#dig.
- object.c (rb_obj_dig): dig in nested arrays/hashes. [Feature #11643]

#### Revision 52504 - 11/09/2015 12:27 PM - nobu (Nobuyoshi Nakada)

dig

- array.c (rb_ary_dig): new method Array#dig.
- hash.c (rb_hash_dig): new method Hash#dig.
- object.c (rb_obj_dig): dig in nested arrays/hashes. [Feature #11643]

#### Revision 52504 - 11/09/2015 12:27 PM - nobu (Nobuyoshi Nakada)

dig

- array.c (rb_ary_dig): new method Array#dig.
- hash.c (rb_hash_dig): new method Hash#dig.
- object.c (rb_obj_dig): dig in nested arrays/hashes. [Feature #11643]

#### Revision 52504 - 11/09/2015 12:27 PM - nobu (Nobuyoshi Nakada)

dig

- array.c (rb_ary_dig): new method Array#dig.
- hash.c (rb_hash_dig): new method Hash#dig.
- object.c (rb_obj_dig): dig in nested arrays/hashes. [Feature #11643]

#### Revision 52504 - 11/09/2015 12:27 PM - nobu (Nobuyoshi Nakada)

dig

- array.c (rb_ary_dig): new method Array#dig.
- hash.c (rb_hash_dig): new method Hash#dig.
- object.c (rb_obj_dig): dig in nested arrays/hashes. [Feature #11643]

#### Revision 748abedf - 11/09/2015 12:48 PM - nobu (Nobuyoshi Nakada)

vm_eval.c: rb_check_funcall_default

- vm_eval.c (rb_check_funcall_default): split from rb_check_funcall to return the given fallback value.
- object.c (rb_obj_dig): use rb_check_funcall_default so that tail call optimization will be possible. [Feature #11643]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@52505 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

#### Revision 52505 - 11/09/2015 12:48 PM - nobu (Nobuyoshi Nakada)

vm_eval.c: rb_check_funcall_default

- vm_eval.c (rb_check_funcall_default): split from rb_check_funcall to return the given fallback value.
- object.c (rb_obj_dig): use rb_check_funcall_default so that tail call optimization will be possible. [Feature #11643]

#### Revision 52505 - 11/09/2015 12:48 PM - nobu (Nobuyoshi Nakada)

vm_eval.c: rb_check_funcall_default

- vm_eval.c (rb_check_funcall_default): split from rb_check_funcall to return the given fallback value.
- object.c (rb_obj_dig): use rb_check_funcall_default so that tail call optimization will be possible. [Feature #11643]

#### Revision 52505 - 11/09/2015 12:48 PM - nobu (Nobuyoshi Nakada)

vm_eval.c: rb_check_funcall_default

- vm_eval.c (rb_check_funcall_default): split from rb_check_funcall to return the given fallback value.
- object.c (rb_obj_dig): use rb_check_funcall_default so that tail call optimization will be possible. [Feature #11643]

#### Revision 52505 - 11/09/2015 12:48 PM - nobu (Nobuyoshi Nakada)

vm_eval.c: rb_check_funcall_default

- vm_eval.c (rb_check_funcall_default): split from rb_check_funcall to return the given fallback value.
- object.c (rb_obj_dig): use rb_check_funcall_default so that tail call optimization will be possible. [Feature #11643]

#### Revision 52505 - 11/09/2015 12:48 PM - nobu (Nobuyoshi Nakada)

vm_eval.c: rb_check_funcall_default

- vm_eval.c (rb_check_funcall_default): split from rb_check_funcall to return the given fallback value.
- object.c (rb_obj_dig): use rb_check_funcall_default so that tail call optimization will be possible. [Feature #11643]

## History

#### #1 - 11/02/2015 04:07 AM - phluid61 (Matthew Kerwin)

How about:

```
params.?[:order].?[shipping_info].?[country]
```

#### #2 - 11/02/2015 08:36 PM - gkop (Gabe Kopley)

Matthew Kerwin wrote:

> How about:
>
> ```
> params.?[:order].?[shipping_info].?[country]
> ```

Thanks Matthew, I'll be honest, I hadn't thought of that. There is a certain appeal in avoiding adding a new method on Hash. On the other hand, by adding a new method we can more easily and more beautifully do metaprogramming, use a potentially more concise expression, convey more rich semantics, and possibly reduce the number of method calls.

#### #3 - 11/03/2015 06:21 AM - matz (Yukihiro Matsumoto)

I prefer method way to (already reverted) params.?[:order].?[:shipping_info].?[:country].
I am not sure dig is the best name for it. It's short, concise thought.
Any other idea, anyone?

Matz.

#### #4 - 11/03/2015 07:06 AM - Hanmac (Hans Mackowiak)

dam i begun to like that "params.?[:order]" bad that it got reverted :/
i think the problem is that it might parse "?[" as a char or something?

#### #5 - 11/04/2015 12:03 AM - dsisnero (Dominic Sisneros)

Yukihiro Matsumoto wrote:

> I prefer method way to (already reverted) params.?[:order].?[:shipping_info].?[:country].
> I am not sure dig is the best name for it. It's short, concise thought.
> Any other idea, anyone?
>
> Matz.

clojure has get-in for their maps, how about fetch_in with replacement like fetch

hash.fetch_in(:order, :shipping_info, :country, 'Not found')

#### #6 - 11/07/2015 04:58 AM - austin (Austin Ziegler)

The problem with hash.fetch_in(:order, :shipping_info, :country, 'Not found') is that 'Not found' is a (possibly) valid key. You would need to implement this with a kwarg.

```
class Hash
  def fetch_in(*keys, **kwargs, &block)
    keys = keys.dup
    ckey = keys.shift

    unless self.key?(ckey)
      return kwargs[:default] if kwargs.key?(:default)
      return block.call(ckey) if block
      fail KeyError, "key not found #{ckey.inspect}"
    end

    child = self[ckey]

    if keys.empty?
      child
    elsif child.respond_to?(:fetch_in)
```

```
      child.fetch_in(*keys, **kwargs, &block)
    else
      fail ArgumentError, 'more keys than Hashes'
    end
  end
end

a = {
  a: {
    b: {
      c: :d
    }
  }
}

def y
  yield
rescue => e
  e
end

p y { a }
p y { a.fetch_in(:a) }
p y { a.fetch_in(:a, :b) }
p y { a.fetch_in(:a, :b, :c) }
p y { a.fetch_in(:a, :b, :c, :d) }
p y { a.fetch_in(:a, :b, :d) }
p y { a.fetch_in(:a, :b, :d, default: 'z') }
p y { a.fetch_in(:a, :b, :d) { 'z' } }
```

As a proposed name, I suggest locate.

--
Austin Ziegler • halostatue@gmail.com • austin@halostatue.ca
http://www.halostatue.ca/ • http://twitter.com/halostatue

#### #7 - 11/07/2015 07:53 PM - keithrbennett (Keith Bennett)

I like the 'dig' method approach for these reasons:

- it does not require any fanciness or magic that could confuse novice Rubyists
- it does not require any change to the interpreter
- the name 'dig' is concise and intention-revealing

I have been hoping for this feature for a long time.  This would be great.

#### #8 - 11/09/2015 06:56 AM - matz (Yukihiro Matsumoto)

The idea is accepted.  The name is the problem. The current candidates are 'dig' and 'fetch_in'.
I prefer 'dig'.  If you have any other idea, please propose.

Matz.

#### #9 - 11/09/2015 07:01 AM - ko1 (Koichi Sasada)

Discussion: https://docs.google.com/document/d/1D0Eo5N7NE_unlySOKG9lVj_eyXf66BQPM4PKp7NvMyQ/pub

Feel free to continue discussion on this ticket.

#### #10 - 11/09/2015 07:02 AM - Hanmac (Hans Mackowiak)

hm shortly patch idea: instead of

```
keys = keys.dup
ckey = keys.shift
```

wouldn't

```
ckey, *keys = keys
```

be better?

EDIT:
maybe a similar function does needed to add to Array too if there is a nested Array/Hash combination like from JSON

#### #11 - 11/09/2015 12:27 PM - nobu (Nobuyoshi Nakada)

*- Status changed from Open to Closed*

Applied in changeset r52504.

---

dig

- array.c (rb_ary_dig): new method Array#dig.
- hash.c (rb_hash_dig): new method Hash#dig.
- object.c (rb_obj_dig): dig in nested arrays/hashes. [Feature #11643]