This feature request stems from dealing with nested hashes, like the params from a request often dealt with in web frameworks.

Conditional code often needs to be written with multiple logical ANDs in order to achieve what this simple function can:

```ruby
class Hash
  def deep_value(*ks)
    if ks.size == 1
      return self[ks.shift]
    else
      val = ks.shift
      return (self[val].is_a?(Hash) ? self[val].deep_value(*ks) : nil)
    end
  end
end
```

`deep_value (dv)` will simply recurse over a hash given a set of indexes and return the value at the end.

Example:

```ruby
foo = {:bar => {:baz => 'blah'}}
foo.dv(:bar, :baz)  # 'blah'
foo.dv(:cats)       # nil
```

---

**Related issues:**

Has duplicate Ruby master - Feature #8246: Hash#traverse

**History**

**#1 - 11/01/2011 09:29 AM - ko1 (Koichi Sasada)**

(2011/11/01 8:52), Kyle Peyton wrote:

Example:

```ruby
foo = {:bar => {:baz => 'blah'}}
foo.dv(:bar, :baz)  # 'blah'
foo.dv(:cats)       # nil
```

Just idea.
How about to extend Hash[] for it?

--
// SASADA Koichi at atdot dot net

**#2 - 11/01/2011 09:53 AM - rkh (Konstantin Haase)**

What's the difference (usability wise) between
hash[:foo][:bar]

and

hash.dv(:foo, :bar)

Konstantin

On Oct 31, 2011, at 16:52 , Kyle Peyton wrote:

Issue #5531 has been reported by Kyle Peyton.

---

Feature #5531: deep_value for dealing with nested hashes
http://redmine.ruby-lang.org/issues/5531

Author: Kyle Peyton
Status: Open
Priority: Normal
Assignee:
Category:
Target version:

This feature request stems from dealing with nested hashes, like the params from a request often dealt with in web frameworks.

Conditional code often needs to be written with multiple logical ANDs in order to achieve what this simple function can:

class Hash
def deep_value(*ks)
  if ks.size

---

#3 - 11/01/2011 09:53 AM - rkh (Konstantin Haase)

What's the difference (usability wise) between

hash[:foo][:bar]

and

hash.dv(:foo, :bar)

Konstantin

On Oct 31, 2011, at 16:52 , Kyle Peyton wrote:

Issue #5531 has been reported by Kyle Peyton.

---

Feature #5531: deep_value for dealing with nested hashes
http://redmine.ruby-lang.org/issues/5531

Author: Kyle Peyton
Status: Open
Priority: Normal
Assignee:
Category:
Target version:

This feature request stems from dealing with nested hashes, like the params from a request often dealt with in web frameworks.

Conditional code often needs to be written with multiple logical ANDs in order to achieve what this simple function can:

class Hash
def deep_value(*ks)
  if ks.size

---

#4 - 11/01/2011 09:53 AM - rkh (Konstantin Haase)

Never mind, got it.

On Oct 31, 2011, at 17:32 , Haase, Konstantin wrote:
What's the difference (usability wise) between

hash[:foo][:bar]

and

hash.dv(:foo, :bar)

Konstantin

On Oct 31, 2011, at 16:52, Kyle Peyton wrote:

Issue #5531 has been reported by Kyle Peyton.

Feature #5531: deep_value for dealing with nested hashes
http://redmine.ruby-lang.org/issues/5531

Author: Kyle Peyton
Status: Open
Priority: Normal
Assignee:
Category:
Target version:

This feature request stems from dealing with nested hashes, like the params from a request often dealt with in web frameworks.

Conditional code often needs to be written with multiple logical ANDs in order to achieve what this simple function can:

class Hash
  def deep_value(*ks)
    if ks.size
      #5 - 11/01/2011 08:23 PM - Eregon (Benoit Daloze)
On 1 November 2011 01:26, SASADA Koichi ko1@atdot.net wrote:

(2011/11/01 8:52), Kyle Peyton wrote:

Example:

  foo = {:bar => {:baz => 'blah'}}
  foo.dv(:bar, :baz)
  -> 'blah'
  foo.dv(:cats)
  -> nil

Just idea.
How about to extend Hash#[] for it?

  ...
  // SASADA Koichi at atdot dot net

That would be nice.

#6 - 11/01/2011 08:34 PM - alexeymuranov (Alexey Muranov)

Konstantin Haase wrote:

Never mind, got it.

On Oct 31, 2011, at 17:32, Haase, Konstantin wrote:

What's the difference (usability wise) between

hash[:foo][:bar]

and
I'll answer anyway if someone else didn't get it :).
{ :foo => 1 }[2][3] raises NoMethodError, and { :foo => 1 }.dv(2,3) or { :foo => 1 }[2,3] should return nil.

Update: also it is possible to keep the list of all arguments in a single variable and call { :foo => 1 }.dv(*args)

#7 - 11/01/2011 09:45 PM - alexeymuranov (Alexey Muranov)
I can think of the following questions/objections to the suggested method definition:

1. is { 1 => 2}.dv(1,1) # => nil the desired result?
2. this method examines the (super)class name of an object, rather than the behavior of an object, so does not allow to mix nested hashes and arrays,
3. this method calls itself recursively, while a loop would suffice.

The following is not a serious suggestion, but seriously, how about:

```ruby
class Object
  def deep_value(*keys)
    obj = self
    obj = obj[keys.shift] while !keys.empty? && obj.respond_to?(:[])
    return obj
  end
end
```

(For this to work well it will be important to call it #deep_value and not to redefine #[].)

Update. Another suggestion, probably a better one (at least simpler):

```ruby
class Object
  def deep_value(*keys)
    obj = self
    obj = obj[keys.shift] until keys.empty? || obj.nil?
    return obj
  end
end
```

#8 - 11/01/2011 10:33 PM - nobu (Nobuyoshi Nakada)

What about:

```ruby
class Hash
def
  keys.inject(self) {|container, key| container.fetch(key) {return}}
end
=end
```

#9 - 11/01/2011 10:42 PM - alexeymuranov (Alexey Muranov)
Nobuyoshi Nakada wrote:

```ruby
=begin
What about:

class Hash
def
  keys.inject(self) {|container, key| container.fetch(key) {return}}
end
=end
```

Just a small remark about defining this exclusively for Hash: what if some of the values is not a Hash but responds to #fetch? (will not look consistent enough to me).

#10 - 11/06/2011 05:48 PM - trans (Thomas Sawyer)
Probably best to use #[] internally too.

```ruby
class Hash
  def [](*keys)
    keys.inject(self) { |container, key| value = container[key]; value ? value : return value }
  end
end

@Alexey you may have a point. But I suspect it would need to be conditioned off of responding to #to_h or #to_hash instead of using is_a?(Hash).
```

@ - 11/07/2011 01:23 AM - neleai (Ondrej Bilka)

Do you need hash or something like multidimensional hash class that uses [], each iterates on nested...

On Sun, Nov 06, 2011 at 05:48:52PM +0900, Thomas Sawyer wrote:

Issue #5531 has been updated by Thomas Sawyer.

Probably best to use #[] internally too.

```ruby
class Hash
  def [](*keys)
    keys.inject(self) { |container, key| value = container[key]; value ? value : return value }
  end
end

@Alexey you may have a point. But I suspect it would need to be conditioned off of responding to #to_h or #to_hash instead of using is_a?(Hash).
```

Feature #5531: deep_value for dealing with nested hashes

http://redmine.ruby-lang.org/issues/5531

Author: Kyle Peyton
Status: Open
Priority: Normal
Assignee:
Category:
Target version:

This feature request stems from dealing with nested hashes, like the params from a request often dealt with in web frameworks.

Conditional code often needs to be written with multiple logical ANDs in order to achieve what this simple function can:

```ruby
class Hash
  def deep_value(*ks)
    if ks.size == 1
      return self[ks.shift]
    else
      val = ks.shift
      return (self[val].is_a?(Hash) ? self[val].deep_value(*ks) : nil)
    end
  end

  alias dv deep_value
end
```

deep_value (dv) will simply recurse over a hash given a set of indexes and return the value at the end.

Example:

```ruby
foo = {:bar => {:baz => 'blah'}}
foo.dv(:bar, :baz)
-> 'blah'
foo.dv(:cats)
-> nil
```

```
http://redmine.ruby-lang.org
```

network packets travelling uphill (use a carrier pigeon)
#12 - 03/27/2012 10:51 PM - mame (Yusuke Endoh)
- Status changed from Open to Assigned
- Assignee set to matz (Yukihiro Matsumoto)

#13 - 03/30/2012 01:04 AM - weexpectedTHIS (Kyle Peyton)
What's the status of this issue?  Good idea?  Bad idea?

#14 - 06/27/2012 03:10 AM - weexpectedTHIS (Kyle Peyton)
I'd really like to see this in the next version of Ruby, it's a really common pattern.

#15 - 10/02/2012 07:12 AM - weexpectedTHIS (Kyle Peyton)
I think there is a strong case for this logic built in to ruby.

#16 - 11/24/2012 08:45 AM - mame (Yusuke Endoh)
- Priority changed from Normal to 3
- Target version set to 2.6

matz expressed a negative opinion for similar proposal (in Japanese, #5550)
The original in Japanese:

Hashの本質はkey-valueのマッピングなので、valueが再帰的にHashであることを想定した(再帰的なHashでなければ役に立たない)メソッドを追加することには抵抗があります。

English translation:
The essence of Hash is a key-value mapping. I'm negative for adding a method that assumes that the value is a recursive hash, or a method that is useful only for a recursive hash.

--
Yusuke Endoh mame@tsg.ne.jp

#17 - 10/08/2016 02:21 PM - dan.erikson (Dan Erikson)
I believe this has recently been implemented as Hash#dig.

#18 - 10/31/2016 03:48 AM - shyouhei (Shyouhei Urabe)
- Status changed from Assigned to Closed

Dan Erikson wrote:

I believe this has recently been implemented as Hash#dig.

Indeed. Closing peacefully.