Add Binding.from_hash

07/16/2013 07:11 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Status: Rejected
Priority: Normal
Assignee: ko1 (Koichi Sasada)
Target version:

Description
Binding.from_hash would work like:

class Binding
  def self.from_hash(hash)
    OpenStruct.new(hash)( binding )
  end
end

It would simplify things like:

ERB.new(IO.read 'template.erb').result Binding.from_hash(template_local: 'example')

Or if you need to eval some code in another process (JRuby, for instance) and need to pass some arguments to the eval code in a hash form.

I didn't want to pollute Hash by adding Hash#to_binding. I believe Binding.from_hash is more appropriate.

Related issues:
Related to Ruby master - Feature #8631: Add a new method to ERB to allow assi...

Closed

History
#1 - 07/16/2013 07:29 PM - rosenfeld (Rodrigo Rosenfeld Rosas)
Whoops. I can't update the description. The implementation should be:

OpenStruct.new(hash).instance_eval { binding }

#2 - 07/17/2013 04:09 AM - charliesome (Charlie Somerville)
Personally I think hash keys should be local variables in the binding, not method calls against self.

It's hard to express this in Ruby, but this can easily be done from the C side.

#3 - 07/17/2013 04:10 AM - charliesome (Charlie Somerville)
PS: I'm neutral towards this feature. I've got no strong feelings that it should or shouldn't be part of Ruby.

#4 - 07/18/2013 06:26 AM - rosenfeld (Rodrigo Rosenfeld Rosas)
I don't mind on it being a local var in the binding since it should work either way, just an implementation detail I'd say... Maybe other Ruby implementations might prefer to use them as methods?

#5 - 08/09/2013 06:56 PM - ko1 (Koichi Sasada)
- Assignee set to ko1 (Koichi Sasada)

What do you think about [Feature #8761]?

Usage:

def get_empty_binding
  binding
end
...

b = get_empty_binding
hash.each{|k, v|
  b.local_variable_set(k, v)
I think that `Binding#local_variable_set()` can be extended to accept one hash parameter (pairs of local variable name and value).

```ruby
b = get_empty_binding
b.local_variable_set(hash)
b.local_variable_set(a: 1, b: 2)
```

#6 - 08/09/2013 10:08 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I don't quite understand how that would help me, Koichi.

How could I use `#8761` to get the same result as the example in the description?

```ruby
ERB.new(IO.read 'template.erb').result Binding.from_hash(template_local: 'example')
```

My current alternative is:

```ruby
ERB.new(IO.read 'template.erb').result OpenStruct.new(template_local: 'example')[:binding]
```

How would `#8761` make it easier for me?

#7 - 08/09/2013 10:10 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

In other words, I want an easier way to convert a hash to a binding. I first thought about `Hash#to_binding` but it didn't feel right to me...

#8 - 08/09/2013 11:53 PM - ko1 (Koichi Sasada)

(2013/08/09 22:10), rosenfeld (Rodrigo Rosenfeld Rosas) wrote:

In other words, I want an easier way to convert a hash to a binding. I first thought about `Hash#to_binding` but it didn't feel right to me...

Ok. Maybe I misunderstood your proposal.

What is conversion from Hash to Binding?
I think it is Binding has a local variables which specified pairs in Hash.

```ruby
hash = {a: 1, b: 2}
b = Binding.to_hash(hash)
eval("p [a, b]", b) #=> [1, 2]
```

Could you explain what do you want?

--
// SASADA Koichi at atdot dot net

#9 - 08/10/2013 07:29 PM - nobu (Nobuyoshi Nakada)

(13/08/09 23:34), SASADA Koichi wrote:

What is conversion from Hash to Binding?
I think it is Binding has a local variables which specified pairs in Hash.

```ruby
hash = {a: 1, b: 2}
b = Binding.to_hash(hash)
```

`Binding.from_hash`?

#10 - 08/12/2013 12:55 AM - ko1 (Koichi Sasada)

nobu (Nobuyoshi Nakada) wrote:

`Binding.from_hash`?

Yes.

#11 - 08/12/2013 09:18 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Koichi-san, that's correct:

```ruby
eval 'p a, b', Binding.from_hash(a: 1, b: 2) #=> 1, 2
```
I didn't notice I replied only to Koichi Sasada when replying to the ruby-core list. Is it possible to set it up so that the reply-to field is set to ruby-core?

Here is some discussion from us from those e-mails so that everyone could have access to it:

(2013/08/10 20:47), Rodrigo Rosenfeld Rosas wrote:

I'm not sure how else to explain it

The only API example I know that requires a binding is the ERB one

It's designed to be used this way:

```ruby
a = 1
b = 2
erb.result binding # both a and b are available inside the template as
well as other methods and variables, like erb
```

Koichi wrote:

Please try:

```ruby
require 'erb'
bind = binding
bind.local_variable_set(:a, 1)
bind.local_variable_set(:b, 2)
puts ERB.new("<%= a %> and <%= b %>").result(bind)
#=> 1 and 2
```

That works, but it is too much trouble for a simple requirements and
besides that more methods and variables may leak when using the current
binding.

That's why people will often use the "OpenStruct.new(hash)(binding)"
trick, since it's much shorter, but still a hack in my opinion.

This is often used in automation tools like Chef or Puppet where the
recipe settings (stored as a hash, usually) should be available for some
templates.

Suppose you have the NewRelic settings under settings[:new_relic] hash
(eg: { application_name: 'My App', enable_rum: true })

In such cases, when generating the newrelic.yml from a newrelic.yml.erb
template, those tools would process it as:

```ruby
File.write 'newrelic/location/newrelic.yml', ERB.new(File.read 'path/to/newrelic.yml.erb').result(OpenStruct.new(settings[:new_relic]).binding)
```

That's why I've created two tickets based on this common requirement. On
this specific one I'm suggesting a Binding.from_hash(hash) to make it
easier to get a binding from a hash for usage in such API's. The other
one suggested ERB to accept also a hash, instead of a binding for #result.

Koichi then replied with:

I'm not sure what methods and variables are leaks.

For example, only "make_binding" mathod is leaked.

```ruby
def make_binding(hash)
  __b = binding
  hash.each{|k, v|
    __b.local_variable_set(k, v)
  }
  __b
end

created_binding = make_binding(a: 1, b: 2)
```
For that tiny script, this is true, Koichi, but usually we get a binding from some class, so all methods would be available as well as other intermediary
local variables in the method calling erb#results.

Like this:

```ruby
class A
def a
  1
end
def b
  eval 'p a', binding
end
end
A.new.b # a has leaked
```

Rodrigo Rosenfeld Rosas wrote:
The other one suggested ERB to accept also a hash, instead of a binding for #result.

It feels better to me.

Rodrigo Rosenfeld Rosas wrote:
The other one suggested ERB to accept also a hash, instead of a binding for #result.

It feels better to me.

Either one is fine to me as long as I can easily pass locals to ERB from a hash using a proper API :) Since this is the only use case I have in mind for
Binding.from_hash I agree with you that passing a hash to ERB constructor feels better.

It seems this feature is not related at all to #8439, could you please review it and remove the related feature? Please relate this ticket to #8631 instead.

Related to deleted (Feature #8430: Rational number literal)

Related to Feature #8631: Add a new method to ERB to allow assigning the local variables from a hash added

Ticket links changed.

Thanks!

Thanks!
Can I close this issue?

#25 - 02/06/2017 12:21 PM - rosenfeld (Rodrigo Rosenfeld Rosas)
If #8631 was accepted, I think it could be okay to close this one as this is the only use-case for this I've needed so far. But before #8631 is accepted, this one would be also a good alternative...

#26 - 05/25/2017 03:47 PM - k0kubun (Takashi Kokubun)
- Status changed from Feedback to Rejected

Since [Feature #8631] is accepted, closing this ticket.

#27 - 05/25/2017 06:30 PM - rosenfeld (Rodrigo Rosenfeld Rosas)
Yes, it makes sense. Thanks a lot! :)

#28 - 12/23/2021 11:40 PM - hsbt (Hiroshi SHIBATA)
- Project changed from 14 to Ruby master

Files

| feature-8643.pdf | 19 KB | 06/26/2014 | rosenfeld (Rodrigo Rosenfeld Rosas) |