Ruby trunk - Feature #8631

Add a new method to ERB to allow assigning the local variables from a hash

07/13/2013 04:42 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

Status:	Closed	
Priority:	Normal	
Assignee:	k0kubun (Takashi Kokubun)	
Target version:	2.5	

Description

It would be interesting if ERB could allow a hash instead of a binding for processing the template.

We wouldn't have to do hacks like:

b = OpenStruct.new(hash).instance_eval{ binding }

Related issues:

Related to CommonRuby - Feature #8643: Add Binding.from_hash	**Rejected**

Associated revisions

Revision eb1652b5 - 05/25/2017 03:38 PM - k0kubun (Takashi Kokubun)

erb.rb: Add ERB#result_with_hash

[ruby-core:55985] [Feature #8631] [fix GH-1623]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58891 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 58891 - 05/25/2017 03:38 PM - k0kubun (Takashi Kokubun)

erb.rb: Add ERB#result_with_hash

[ruby-core:55985] [Feature #8631] [fix GH-1623]

Revision 58891 - 05/25/2017 03:38 PM - k0kubun (Takashi Kokubun)

erb.rb: Add ERB#result_with_hash

[ruby-core:55985] [Feature #8631] [fix GH-1623]

Revision 58891 - 05/25/2017 03:38 PM - k0kubun (Takashi Kokubun)

erb.rb: Add ERB#result_with_hash

[ruby-core:55985] [Feature #8631] [fix GH-1623]

History

#1 - 07/14/2013 05:28 PM - sorah (Sorah Fukumori)

- Status changed from Open to Assigned

- Assignee set to seki (Masatoshi Seki)

Assigning to erb maintainer;

IMO, I recommend you to show example mock-code that uses your proposal, to show what interface (API) do you want :)

#2 - 07/15/2013 09:39 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I didn't propose an API because I don't really care about the API as long as it allows us to provide a hash instead of a binding.

If you want an example API that would satisfy me, I'd be happy if we used the same API (ERB#result). If the argument is a hash, use the keys as the local variables and the values as the variable values.

#3 - 07/15/2013 09:43 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

You can see here how often people want to use a hash to render an ERB template:

https://github.com/search?q=erb+result+openstruct&type=Code&ref=searchresults

And there's no quick way for performing this common procedure. The quickest one seems to be using OpenStruct for that...

#4 - 06/26/2014 12:49 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

- File feature-8631.pdf added

Attached slide for proposal

#5 - 06/30/2014 08:07 AM - naruse (Yui NARUSE)

received, thanks!

#6 - 07/26/2014 05:36 AM - ko1 (Koichi Sasada)

I think it is easy to implement, but not clear what is "self" on ERB evaluation context.

If you provide binding, self will be binding's self.

#7 - 07/26/2014 05:39 AM - matz (Yukihiro Matsumoto)

I like the idea, but I have one concern.

The binding also has information about the receiver (which defaults to top-level self).
I am not sure whether it's critical or not.

I hope Seki will address this.

Matz.

#8 - 07/26/2014 06:07 AM - matsuda (Akira Matsuda)

FYI here's an already existing implementation by Seki-san: https://gist.github.com/seki/6449467
Gemified version is here: https://github.com/takkanm/erb_with_hash

#9 - 08/10/2016 02:54 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #8643: Add Binding.from_hash added

#10 - 02/06/2017 01:45 PM - nobu (Nobuyoshi Nakada)

Another implementation:

```
diff --git a/lib/erb.rb b/lib/erb.rb
index 9483711024..9813b4dc71 100644
--- a/lib/erb.rb
+++ b/lib/erb.rb
@@ -887,6 +887,11 @@
    # code evaluation.
    #
    def result(b=new_toplevel)
+     if b.respond_to?(:each_pair)
+       x = new_toplevel
+       b.each_pair {|k, v| x.local_variable_set(k, v)}
+       b = x
+     end
      if @safe_level
        proc {
          $SAFE = @safe_level
```

#11 - 05/16/2017 01:22 AM - hsbt (Hiroshi SHIBATA)

- Target version set to 2.5

- Assignee changed from seki (Masatoshi Seki) to k0kubun (Takashi Kokubun)

#12 - 05/18/2017 02:04 PM - k0kubun (Takashi Kokubun)

There's a problem that a receiver is unclear and not configurable if argument is only a Hash object. To address the problem, we need to have both anything for a receiver (of course Binding is okay) and a Hash object in arguments.

Even in that case, most users of this feature will want to pass only a Hash object as argument. So we want to pass a Hash object with default option for its receiver, without breaking backward compatibility.

To solve all those problems, I propose to use keyword argument like https://github.com/ruby/ruby/pull/1618.

Following is example use case. I think it's fairly easy to use and understand what it does.

```
ERB.new('<%= my_local %>').result(locals: { my_local: 'value' })
```

#13 - 05/19/2017 03:56 PM - k0kubun (Takashi Kokubun)

I found that my suggestion is a little hard to implement to avoid argument modification. It got a little ugly https://github.com/ruby/ruby/pull/1618/commits/ed1c1e520eec6cb96f1eae88bef2fa4ac54a3e6c.

Anyway, currently we have 3 possible choices.

1. Proposed one (implementation is suggested by nobu): ERB.new(*).result({ foo: bar })
2. Seki-san's patch (erb_with_hash): ERB.new(*).result_with_hash({ foo: bar })
3. My suggestion: ERB.new(*).result(locals: { foo: bar })

From user's point of view, 1 is okay but a little confusing that it can take multiple types in the same argument, 2 is good but a name is long and 3 is the best.
From maintainer's point of view, 1 is okay, 2 is the best and 3 is a little hard.

So all of them have trade-offs. While I feel it's overkill to make it capable of having both Binding and Hash as arguments, I personally want to introduce 3 because it's easy to use and not confusing.

I want your opinions about this.

#14 - 05/19/2017 04:09 PM - k0kubun (Takashi Kokubun)

Umm, I noticed that actually the length of "ERB.new(*).result(locals: {a: b})" is the same as "ERB.new("aaa").result_with_hash(a: b)". So the long name of 2 seems not a problem compared to 3.

Then, my final personal preference is Seki-san's 2 (erb_with_hash) https://github.com/ruby/ruby/pull/1623. It's already used by some users of erb_with_hash.gem and I think the method is good to have in ERB.

#15 - 05/20/2017 05:41 PM - k0kubun (Takashi Kokubun)

- Status changed from Assigned to Feedback

#16 - 05/20/2017 08:42 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I would already suggest a new name when I read about the problems with overriding the current method, so I would agree that 2 would be a better approach, I'm just not sure which name would be best. I like result_with_hash, but it could be result_from_locals or result_from_hash.

Or maybe ERB.new(str, locals: {a: b}).result?

#17 - 05/22/2017 03:50 AM - k0kubun (Takashi Kokubun)

> Or maybe ERB.new(str, locals: {a: b}).result?

For this case, it would have the same problem as ERB.new(str).result(locals: {a: b}) because it can have both binding and locals. In that case, adding locals to given binding is hard to maintain. Also, if we want to avoid the situation, raising error for that case does not seem a good idea (method should be separated in that case, especially for "3").

> I'm just not sure which name would be best. I like result_with_hash, but it could be result_from_locals or result_from_hash.

Then, the remaining problem is only the name of result_with_hash counterpart. Possible choices are:

- result_with_hash
- result_from_locals
- result_from_hash

What this method does is "rendering template with a context that given local variables are set and returning the result". It's not result derived from only local variables. While "locals" is longer than "hash", it describes not type of argument but its actual content.

So I prefer "result_with_locals" ("render_with_locals" may be better but it should be unified with "result"). If you use the method with that name, I'm okay to add that.

#18 - 05/22/2017 06:35 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

I'm actually fine with whatever name you prefer. result_with_locals and render_with_locals are both fine to me. Thanks a lot!

#19 - 05/25/2017 03:37 PM - k0kubun (Takashi Kokubun)

I discussed about this with Seki-san.

For name, I rethought that "result_with_locals" sounds to use caller's local variables. Since "result_with_hash" doesn't seem to mean it and is shorter, we agreed "result_with_hash" is the best.

For receiver problem, since TOPLEVEL_BINDING.dup (and its receiver, main) is already used, we agreed it's consistent and safe to use TOPLEVEL_BINDING.dup.

As Seki-san agreed to introduce this in person, I'm going to merge https://github.com/ruby/ruby/pull/1623.

#20 - 05/25/2017 03:38 PM - k0kubun (Takashi Kokubun)

- Status changed from Feedback to Closed

Applied in changeset trunk|r58891.

---

erb.rb: Add ERB#result_with_hash

[ruby-core:55985] [Feature #8631] [fix GH-1623]

#21 - 05/25/2017 06:29 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Awesome, thanks!

Files

feature-8631.pdf	27.7 KB	06/26/2014	rosenfeld (Rodrigo Rosenfeld Rosas)