Ruby trunk - Feature #12931

Add support for Binding#instance_eval

11/14/2016 03:30 AM - ioquatix (Samuel Williams)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	

Description

Many people would probably like to use binding.instance_eval when executing templates. The reason for this is because binding.eval is slow.

The use case is template rendering, e.g. systems like ERB.

In my template renderer, I did have

```
            if Binding === scope
                # Slow code path, evaluate the code string in the given binding (scope).
                scope.eval(code, @buffer.path)
            else
                # Faster code path, use instance_eval on a compiled Proc.
                scope.instance_eval(&to_proc)
            end
```

The binding path is several orders of magnitude slower because the code is a string and must be parsed, compiled, etc. But, to_proc path can be cached.

In terms of duck typing, it would be nice if binding implemented #instance_eval and would allow passing a block.

History

#1 - 11/14/2016 04:58 AM - nobu (Nobuyoshi Nakada)

- Description updated

Since local variables and constants depend on the context, it won't be able to be "cached," as a Proc compiled other place.
So I think it isn't worth.

#2 - 11/15/2016 05:55 AM - ioquatix (Samuel Williams)

Nobuyoshi Nakada wrote:

> Since local variables and constants depend on the context, it won't be able to be "cached," as a Proc compiled other place.
> So I think it isn't worth.

That makes sense.

So, to clarify, you are saying that there is no way to reuse the compiled proc? It would need to be recompiled?

From my POV, I guess the benefit of this is duck typing.

If performance was the same of eval, it would be acceptable, but if there is some way it could be better, that would be awesome.

#3 - 01/19/2017 07:21 AM - nobu (Nobuyoshi Nakada)

Correct, there is no way to reuse the compiled proc.

#4 - 01/19/2017 07:22 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Rejected