Ruby master - Feature #15854

Tracing instance variable assignment

05/16/2019 06:34 AM - igaiga (Kuniaki IGARASHI)

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

Description

I suggest a feature "tracing instance variable assignment". It's useful for debugging.

Use case:

In Rails, we use instance variables in views and controllers. When we got a bug caused by instance variable unintentional values, if we traced instance variable assignment timing, it would be good informations.

And in Rails views, there are no source codes of self class. That's built dynamically.

Current behavior (Ruby2.6):

In Ruby 2.6, only if there is a source code file to assign instance variable, we can trace instance variable assignment by following code (check_instance_variable_assignment.rb). But it's difficult if the assignment codes are defined dynamically. For example, in Rails view.

(And in another story, global variables assignment are traced by Kernel#trace_var.)

check_instance_variable_assignment.rb

```
def trace_start
  TracePoint.trace(:line) do |tp|
    target_class_name = "Foo"
    target_instance_variable_name = "@bar"

    line = File.open(tp.path, "r"){|f| f.readlines[tp.lineno - 1] }
    node = RubyVM::AbstractSyntaxTree.parse(line).children.last

    # check instance variable assignment
    next unless node.type == :IASGN

    # check class name
    target_class = Kernel.const_get(target_class_name)
    next unless tp.self.is_a?(target_class)

    # check variable name
    instance_variable_name = node.children.first
    next unless instance_variable_name == target_instance_variable_name.to_sym

    puts "#{target_class_name} #{target_instance_variable_name} is assigned in #{tp.path}:#{tp.
lineno} #{tp.defined_class} #{tp.method_id}"
  end
end

class Foo
  def bar
    @bar = "text"
  end
end

trace_start
Foo.new.bar
#=> Foo @bar is assigned in check_instance_variable_assignment.rb:25 Foo bar
```

Suggesting feature example:

Add new arguments for TracePoint.new method like :line and :call to trace instance variables assignment.

- :iasgn (IASGN name from RubyVM::AbstractSyntaxTree::Node)
- :casgn (CVASGN (or CASGN?) name from RubyVM::AbstractSyntaxTree::Node. I think class variables tracing is useful too.)

And get informations

- class name (It might be get by trace_point.self)
- variable name ("@foo", "@@foo")

A sample code to use the feature:

tp_iasgn.rb

```
TracePoint.trace(:iasgn) do |tp|
  target_class_name = "Foo"
  target_instance_variable_name = "@bar"

  # check class name
  target_class = Kernel.const_get(target_class_name)
  next unless tp.self.is_a?(target_class)

  # check variable name
  next unless target_instance_variable_name == tp.variable_name

  puts "#{target_class_name} #{target_instance_variable_name} is assigned in #{tp.path}:#{tp.
lineno} #{tp.method_id} #{tp.defined_class}"
  puts caller # even in dynamic code case, we can get caller informations.
end
```

History

#1 - 05/16/2019 07:00 AM - k0kubun (Takashi Kokubun)

This feature would be helpful if it were integrated with byebug's tracevar, which currently supports only global variable tracing.

> :iasgn (IASGN name from RubyVM::AbstractSyntaxTree::Node)
> :casgn (CVASGN (or CASGN?) name from RubyVM::AbstractSyntaxTree::Node. I think class variables tracing is useful too.)

For me it's hard to understand what's :iasgn at a glance. Perhaps other TracePoint event names and Node's names have different naming conventions. Other TracePoint event names are like :thread_begin or :fiber_switch. So for :iasgn, I'd propose:

- :ivar_assign (close to original proposal)
- :instance_variable_set (same as official method name, maybe too long but "ivar" may not be friendly either)

> I think class variables tracing is useful too.

Cloud you describe a use case? Your reasoning behind :iasgn is Rails controllers / views and that makes sense, but I don't think it applies to :casgn.

#2 - 05/16/2019 08:37 AM - shevegen (Robert A. Heiler)

> I suggest a feature "tracing instance variable assignment"

I love introspection. The old pickaxe had an example tracing global variables, and
it would seem nice if we could extend this to other/more variables, so I am +1 for
the suggestion. See also pry acting a bit like an introspection-debugger, e. g.
"cd object; ls" what the object has (at the least as an idea).

I should, however had, also admit that I do not use the trace-var functionality
for global variables, primarily because I try to avoid global variables (and thus
use them so rarely that I don't need to trace anything), but also partially because
I am not completely sure whether I personally need that functionality. But still I
think it would be nice to have, so I agree from this point of view with igaiga.

I agree that :iasgn is a strange name but I guess other/better names can be used;
may be good to ask ko1 what he thinks about tracing (and exposing this for ruby
users) in principle (and of course matz whether he wants to expose functionality

to ruby users, but I think since we can trace global vars, it may be a related
topic to be able to trace other variables too).

#3 - 05/21/2019 08:38 AM - igaiga (Kuniaki IGARASHI)

Thanks to comments.

> :ivar_assign (close to original proposal)
> :instance_variable_set (same as official method name, maybe too long but "ivar" may not be friendly either)

Looks good. I feel they are good to read, so better than my suggested :iasgn.

> Cloud you describe a use case? ... but I don't think it applies to :casgn.

I've tried to describe the use cases, but I can't write use cases for class variables.

I thought following patterns that how to torigger their assign timings.

- global variable: Kernel#trace_var
- class variable: none
- instance variable: suggested by this issue
- local variable: none? (but we don't need)

So I thought class variable assignment trigger is useful for us. But cases to use class variables are not so often, so the trigger of :class_variable_set would be not so useful.

#4 - 07/29/2019 07:13 AM - ko1 (Koichi Sasada)

- Assignee set to ko1 (Koichi Sasada)