Ruby master - Feature #13026

Public singleton methods

12/12/2016 11:03 AM - subtileos (Daniel Ferreira)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Abstract

I would like to propose the implementation of:

```
Object#public_singleton_methods
```

Object#public_singleton_methods should return a list of public singleton methods.

Background

Currently Object#singleton_methods returns a list of public and protected singleton methods.

Currently the best way I know of retrieving a list of public singleton methods is:

```
foo = Foo.new
foo.singleton_methods(false) & foo.public_methods(false)
```

I consider the definition of the public interface of any object very important.
For that reason I tend to keep it as simple as possible.

Also, the way I like to enforce the design is by using unit tests, testing the interface itself.
That way a change of the interface is very visible for the developer and code reviewer.

I also tend to use protected a lot to give private methods a higher level of relevance.

Implementation

This is a feature that I would be very interested in developing.
It would be my first contribution to ruby core.
If you understand that it is worth the effort I would be more than happy to do it.

History

#1 - 12/12/2016 11:09 AM - subtileos (Daniel Ferreira)

- Description updated

- Subject changed from Singleton public methods to Public singleton methods

#2 - 12/20/2016 03:26 AM - shyouhei (Shyouhei Urabe)

- Tracker changed from Bug to Feature

#3 - 02/22/2017 06:48 AM - matz (Yukihiro Matsumoto)

Interesting. Could you tell us expected use-case?

Matz.

#4 - 02/22/2017 06:53 PM - subtileos (Daniel Ferreira)

Yukihiro Matsumoto wrote:

> Interesting. Could you tell us expected use-case?
>
> Matz.

Hi Matz. Thank you for your question and the opportunity.

Unit tests are one of the ways I use to enforce the class/module API contract.

So for a given class:

```
class Foo
  class << self
    def foo
      :foo
    end

    protected

    def bar
      :bar
    end

    private

    def baz
      :baz
    end
  end
end
```

I usually have the following unit test:

```
require "minitest/autorun"

class TestFoo < Minitest::Test
  def test_public_singleton_interface
    public_singleton_methods = Foo.singleton_methods & Foo.public_methods(false)
    assert_equal [:foo], public_singleton_methods
  end
end
```

I also tend to inspect my module/class interfaces while in debugging mode.
Foo.public_singleton_methods it is much easier to type than the intersection I'm using currently.

I don't believe refinements are an option here.
Monkey patching is something I would like to avoid as much as possible.

This comes inline with my idea of clearly defining contract interfaces.

This is one of the aspects of it. There are many more.

Kind regards,

Daniel