Ruby - Feature #12374

SingletonClass

05/12/2016 06:28 PM - sawa (Tsuyoshi Sawada)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

I propose to have a class SingletonClass, a subclass of the class Class, to which all singleton classes belong. It should be the owner of all the properties that are specific to singleton classes. Also, the methods defined on Singleton module should be moved to this class.

Reasons are as follows:

1. I was thinking that the reason #12084 hasn't been seen positively may be because the developers do not want to define a method only on limited instances of a class. If we have SingletonClass, the method #instance proposed in #12084 could be defined as an instance method of SingletonClass.

2. The way to introduce the singleton pattern using the Singleton module (http://ruby-doc.org/stdlib-2.3.0/libdoc/singleton/rdoc/Singleton.html):

```
class A
   include Singleton
   # ...
end
```

is a bit unnatural and verbose. If we have SingletonClass, then we can define a singleton class more naturally:

```
A = SingletonClass.new
```

History

#1 - 05/12/2016 06:30 PM - sawa (Tsuyoshi Sawada)

Sorry, for the last example, I meant:

```
A = SingletonClass.new
```

#2 - 05/20/2016 03:45 PM - dsferreira (Daniel Ferreira)

Typo corrected

#3 - 05/20/2016 03:56 PM - dsferreira (Daniel Ferreira)

- Description updated

#4 - 05/20/2016 04:00 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

What is SingletonClass?
A wrapper object?

```
def (SingletonClass = Object.new).new(*args)
  Class.new(*args) {include Singleton}
end
```

#5 - 05/21/2016 02:39 AM - sawa (Tsuyoshi Sawada)

Nobuyoshi Nakada wrote:

> What is SingletonClass?
> A wrapper object?

No, it should be a subclass of Class. It would perhaps need C-level modification. For any singleton class, I want SingletonClass to be its class.

Present:

```
"".singleton_class.class
# => Class
"".singleton_class.class.ancestors
# => [Class, Module, Object, Kernel, BasicObject]
```

I want it to be:

```
"".singleton_class.class
# => SingletonClass
"".singleton_class.class.ancestors
# => [SingletonClass, Class, Module, Object, Kernel, BasicObject]
```

#6 - 05/21/2016 03:03 AM - nobu (Nobuyoshi Nakada)

If a singleton class is a subclass of SingletonClass, no singleton classed inheriting other classes cannot be made.
I think it's a bad idea.

#7 - 05/21/2016 03:05 AM - sawa (Tsuyoshi Sawada)

Nobuyoshi Nakada wrote:

> If a singleton class is a subclass of SingletonClass, no singleton classed inheriting other classes cannot be made.
> I think it's a bad idea.

Sorry, my previous example was inappropriate. I fixed it. I want a singleton class to be an **instance** of SingletonClass.

#8 - 08/03/2016 04:07 AM - trans (Thomas Sawyer)

So...

```
Object..singleton_class.instance_of?(SingletonClass)  #=> true
```

#9 - 08/09/2016 06:13 AM - matz (Yukihiro Matsumoto)

- Status changed from Feedback to Closed

I don't think the idea itself is meaningless.
But its use-case is not clear to us yet.

(For the record, Singleton module is totally different story.)

If you have any real-world use-case, please reopen the issue.

Matz.