CommonRuby - Feature #8275

Add Module#public_const_get

04/16/2013 06:28 PM - rkh (Konstantin Haase)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Right now, const_get will always return a constant, no matter the visibility, which is probably what we want most of the time. But if you for instance have code that does some automatic name to constant resolution, you might now want to leak private constants.

module Foo
Bar = 42
private_constant :Bar
end

currently:

Foo::Bar # raises NameError
Foo.const_get("Bar") # => 42
Object.const_get("Foo::Bar") # => 42

proposed:

Foo.public_const_get("Bar") # raises NameError
Object.public_const_get("Foo::Bar") # raises NameError

History

#1 - 04/16/2013 08:02 PM - vipulnsward (Vipul Amler)

=begin

And vice-versa for set too maybe

(({Module#public_const_set}))/
(({Module#private_const_set}))

=end

#2 - 04/16/2013 11:58 PM - headius (Charles Nutter)

What about an optional boolean for the existing const_get/const_set that indicates whether private constants should be included?

I guess having separate methods does align with public_instance_method and friends, though.

#3 - 04/17/2013 03:02 AM - rkh (Konstantin Haase)

I was considering a boolean. However, const_get already takes a boolean argument indicating whether or not to include inherited constants, and two boolean arguments would not make a readable API, I guess.

Maybe this would be a good use case for keyword arguments if the API were to be extended?

For instance with the following signature:

```
const_get(name, inherited = true, private: true)
```

Or even overload so that the old signature still works

```
const_get(name, inherited = true)
```

but it might also support

```
const_get(name, inherited: true, private: true)
```

What do you think?

#4 - 04/17/2013 03:12 AM - headius (Charles Nutter)

rkh (Konstantin Haase) wrote:

> I was considering a boolean. However, const_get already takes a boolean argument indicating whether or not to include inherited constants, and two boolean arguments would not make a readable API, I guess.

Ahh, good point. I agree.

> Maybe this would be a good use case for keyword arguments if the API were to be extended?
>
> For instance with the following signature:
>
> `const_get(name, inherited = true, private: true)`
>
> Or even overload so that the old signature still works
>
> `const_get(name, inherited = true)`
>
> but it might also support
>
> `const_get(name, inherited: true, private: true)`
>
> What do you think?

Interesting ideas. I'm starting to lean back toward having a separate method rather than pushing too much logic into this one. Keyword args definitely help readability if we wanted to overload, but maybe we really don't need to overload it more than it is overloaded already?