Ruby master - Feature #12380

`Struct` as a subclass of `Class`

05/14/2016 06:22 AM - sawa (Tsuyoshi Sawada)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

This issue is somewhat of the same flavor as #12374.

Struct has a constructor that creates a class:

```
Struct.new(:foo)
# => #<Class:0x007f605f892cb0>
```

and this is the same situation with Class.

```
Class.new
# => #<Class:0x007f605f70c788>
```

Hence, most naturally, Struct should be a subclass of Class. But in reality, it isn't:

```
Struct.ancestors
# => [Struct, Enumerable, Object, Kernel, BasicObject]
```

The current structure around Struct is counter-intuitive to me.

I propose that either Struct should be redefined as a subclass of Class, or a new class StructClass should be introduced as a subclass of Class, and take over the functionality of Struct.

History

#1 - 05/14/2016 07:18 AM - jeremyevans0 (Jeremy Evans)

Tsuyoshi Sawada wrote:

> I propose that either Struct should be redefined as a subclass of Class, or a new class StructClass should be introduced as a subclass of Class, and take over the functionality of Struct.

You can't subclass Class in ruby:

```
$ ruby -e 'Class.new(Class)'
-e:1:in `initialize': can't make subclass of Class (TypeError)
```

I guess it is counter-intuitive that Struct.new creates a subclass of the receiver instead of an instance of the receiver, but do you think there would be any practical benefits to changing things?

If I had to guess, the reason that Struct is not a subclass of Class is that you can't subclass Class, maybe because allowing subclasses of Class would cause problems in the object model?

#2 - 05/14/2016 08:28 AM - sawa (Tsuyoshi Sawada)

Jeremy Evans wrote:

> You can't subclass Class in ruby:

I see.

> do you think there would be any practical benefits to changing things?

At the moment, my motivation is only conceptual. We are able to do things the way it is now, but from OOP point of view, it is not desirable. It is just as good as getting rid of Bignum and Fixnum in favor of Integer. Since the developers are doing this kind of thing right now, I thought maybe it is good timing to ask things like I proposed.

> the reason that Struct is not a subclass of Class is that you can't subclass Class, maybe because allowing subclasses of Class would cause problems in the object model?

I understand the difficulty of freely allowing subclassing of Class. But what about only allowing hard-wired subclasses such as Struct (and SingletonClass that I proposed in #12374), and not allowing other subclassing?