Ruby 1.8 - Bug #4273

Inherited hook called after module_eval for Class.new

01/12/2011 08:46 PM - banister (john mair)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	Ruby 1.8.7
ruby -v:	ruby 1.8.7 (2010-06-23 patchlevel 299) [i386-mingw32]

Description

=begin
class Base
def self.inherited(klass)
klass.instance_variable_set(:@x, :base)
end
end

Derived = Class.new(Base) do
@x = :derived
end

BUG HERE!
Derived.instance_variable_get(:@x) #=> :base

In Ruby 1.9 the line above returns :derived. Ruby 1.9 has the expected behaviour as the inherited hook should be invoked prior to running the block -- so that the block can override any ivar defaults defined by the inherited hook.

The problem lies in the following code, from object.c:

static VALUE
rb_class_initialize(argc, argv, klass)
int argc;
VALUE *argv;
VALUE klass;
{
VALUE super;

```
 if (RCLASS(klass)->super != 0) {
rb_raise(rb_eTypeError, "already initialized class");
 }
 if (rb_scan_args(argc, argv, "01", &super) == 0) {
super = rb_cObject;
 }
 else {
rb_check_inheritable(super);
 }
 RCLASS(klass)->super = super;
 rb_make_metaclass(klass, RBASIC(super)->klass);
```

- rb_mod_initialize(klass);

- rb_class_inherited(super, klass);

  return klass;
  }

The fix would be simple, just swap the two starred lines above.

thanks,

John
=end

History

#1 - 01/13/2011 07:39 AM - zimbatm (zimba tm)

=begin
Hi,

unfortunately, changing these two lines would break backward-compatibility, which would bring the same situation between the 1.8.7 release and the next-one. The only solutions I see for you is to either not rely on that feature, either discard 1.8 backward-compatibility.

I added this problem to this page that I just created : http://redmine.ruby-lang.org/wiki/ruby-19/MigrationIssuesFrom18

Cheers,
zimbatm
=end