Ruby master - Bug #8297

extend & inherited class variable issue

04/19/2013 09:02 PM - dunric (David Unric)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.0.0p0	**Backport:**	

Description

=begin
By the current documentation, (({Object#extend})) method has to (only) add instance methods of a module given as a parameter.
In the following example, the class ((|C|)) is extended with module ((|M|)).
By (({class_variables})) method sent to singleton class of C also did inherit class variable ((|@@xyz|)).
However when inherited ((|@@xyz|)) is accessed, (({NameError})) exception is raised as it is was not initialized.

```
module M
@@xyz = 123
end
M.class_variables    # [:@@xyz]
M.class_variable_get :@@xyz    # 123 , so far so good

class C
extend M
end
p C.singleton_class.class_variables    # [:@@xyz]
p C.singleton_class.class_variable_get :@@xyz # NameError exception
```

Either (({class_variables})) returns invalid array - ie. ((|@@xyz|)) was not inherited at all or (({class_variable_get})) ignores class variables inherited from module (when sent to a singleton).
=end
Prior Ruby versions like 1.9.3p392 does not suffer this issue as return with Module#class_variables returns an empty array.

Associated revisions

Revision 7470f965 - 09/21/2019 11:10 PM - jeremyevans (Jeremy Evans)

Fix Module#class_variables for singleton classes of classes/modules

Module#class_variables should reflect class variable lookup. For singleton classes of classes/modules, this means the lookup should be:

- Singleton Class
- Class
- All Ancestors of Class

Note that this doesn't include modules included in the singleton class, because class variable lookup doesn't include those.

Singleton classes of other objects do not have this behavior and always just search all ancestors of the singleton class, so do not change the behavior for them.

Fixes [Bug #8297]

History

#1 - 04/19/2013 09:12 PM - dunric (David Unric)

=begin
By the current documentation, Object#extend method has to (only) add instance methods of a module given as a parameter.
In the following example, the class ((*C*)) is extended with module ((*M*)).
By ((*class_variables*)) method sent to singleton class of C also did inherit class variable ((*@@xyz*)).
However when inherited ((*@@xyz*)) is accessed, ((*NameError*)) exception is raised as it is was not initialized:

```
module M
```

@@xyz = 123
end

M.class_variables   # [:@@xyz]
M.class_variable_get :@@xyz   # 123 , so far so good

class C
extend M
end
p C.singleton_class.class_variables   # [:@@xyz]
p C.singleton_class.class_variable_get :@@xyz # NameError exception

Either ((*class_variables*)) returns invalid array - ie. ((*@@xyz*)) was not inherited at all or ((*class_variable_get*)) ignores class variables inherited from module (when sent to a singleton).

Prior Ruby versions like 1.9.3p392 does not suffer this issue as Module#class_variables returns an empty array.
=end

#2 - 04/19/2013 11:53 PM - nobu (Nobuyoshi Nakada)

- Description updated

#3 - 01/14/2019 05:37 PM - jsc (Justin Collins)

This appears to still be an issue with Ruby 2.6.0 (ruby 2.6.0p0 (2018-12-25 revision 66547) [x86_64-linux]):

Example:

```
module M
  @@xyz = 123
end

puts "M.class_variables: #{M.class_variables.inspect}"
puts "M.class_variable_get :@@xyz: #{M.class_variable_get :@@xyz}"

class C
  extend M
end

puts "C.class_variables: #{C.class_variables.inspect}"
puts "C.class_variable_get :@@xyz: #{C.class_variable_get :@@xyz}"
```

Output:

```
M.class_variables: [:@@xyz]
M.class_variable_get :@@xyz: 123
C.class_variables: []
Traceback (most recent call last):
    1: from 8297.rb:13:in `<main>'
8297.rb:13:in `class_variable_get': uninitialized class variable @@xyz in C (NameError)
```

#4 - 08/10/2019 04:33 AM - jeremyevans0 (Jeremy Evans)

- Backport deleted (1.9.3: UNKNOWN, 2.0.0: UNKNOWN)

- File singleton-class-class-variable-lookup-8297.patch added

This bug is still present in the master branch. I think the best way to fix it is to modify Module#class_variables for singleton classes of classes/modules to use the same lookup order as Module#class_variable_get:

- Singleton Class
- Class
- All Ancestors of Class

Note that this does not include modules included in the singleton class.

Attached is a patch that implements this behavior.

#5 - 09/21/2019 11:10 PM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|7470f965650bf17875632f0c5f9e5a4d9de9fc3f.

Fix Module#class_variables for singleton classes of classes/modules

Module#class_variables should reflect class variable lookup. For singleton classes of classes/modules, this means the lookup should be:

- Singleton Class
- Class
- All Ancestors of Class

Note that this doesn't include modules included in the singleton class, because class variable lookup doesn't include those.

Singleton classes of other objects do not have this behavior and always just search all ancestors of the singleton class, so do not change the behavior for them.

Fixes [Bug #8297]

Files

singleton-class-class-variable-lookup-8297.patch	2.75 KB	08/10/2019	jeremyevans0 (Jeremy Evans)