Introduce super! for calling old definition when reopening classes

07/09/2012 10:23 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Status: Rejected
Priority: Normal
Assignee: matz (Yukihiro Matsumoto)
Target version: 2.0.0

Description
ActiveSupport adds support for alias_method_chain which is a hack for being able to call the original method being overridden when reopening a class:

Extracted from documentation:
http://apidock.com/rails/ActiveSupport/CoreExtensions/Module/alias_method_chain

"Encapsulates the common pattern of:

alias_method :foo_without_feature, :foo
alias_method :foo, :foo_with_feature"

I'd prefer to have an official non-hacking way of achieving the same with just Ruby. Something simpler like:

class A
def a
  2
end

class A
def a
  super! * 3
end

A.new.a == 6

This way we wouldn't need to polute A with a_with_feature and a_without_feature methods if we're not interested on them.

History
#1 - 07/09/2012 10:40 PM - shyouhei (Shyouhei Urabe)

(1) I don't like the name
(2) It's not obvious what'd happen. What if A includes other module, which also define a method? alias_method_chain is clear about it because alias_method_chain is evaluated at the time of class loading. Your synatx delays it to actual calling of the method.

#2 - 07/09/2012 10:45 PM - marcandre (Marc-Andre Lafortune)

Hi,

I'd prefer to have an official non-hacking way of achieving the same with just Ruby. Something simpler like:

class A
def a
  2
end

class A
def a
  super! * 3
end

A.new.a == 6
This way we wouldn't need to pollute A with a_with_feature and a_without_feature methods if we're not interested on them.

The alias_method_chain pattern is now obsolete with Module#prepend:

```ruby
module Triple
  def a
    super * 3
  end
end

class A
  prepend Triple
end
A.new.a # => 6
```

#3 - 07/09/2012 10:59 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

shyouhei (Shyouhei Urabe), I think the behavior should be the same as the "prepend" behavior described by marcandre (Marc-Andre Lafortune).

```ruby
def a
  super! * 3
end
```

would be a short syntax for

```ruby
prepend Module.new {
  def a
    super * 3
  end
}
```

I'm not really worried about what name to use, but I guess most other names would only be possible for 3.0.

Another options could be "old_def", "old_definition", "old_method_definition", "old_super" or anything else you might prefer, I don't really care about the name.

#4 - 07/10/2012 01:04 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

remember eval.

```ruby
def a
  eval("super!") * 3
end
```

should what happen?

#5 - 07/10/2012 01:46 AM - rosenfeld (Rodrigo Rosenfeld Rosas)

good catch :) Have no idea :P Just raise an exception "super! can't be used with eval" :)

#6 - 07/10/2012 02:05 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Rejected

Rejecting because of unworkability and no longer any use cases.