

Ruby - Feature #7376

Proposal for new syntax construct to define errors

11/17/2012 09:05 AM - kenn (Kenn Ejima)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		
Description =begin As discussed here - https://gist.github.com/4091803 When we define an error class in a module, we do one of the following two ways. module App class Error < StandardError; end class ServerError < Error; end class ClientError < Error; end end module App Error = Class.new(StandardError) ServerError = Class.new(Error) ClientError = Class.new(Error) end IMO, the ugliness of the syntax is partly responsible that not many libraries have custom errors of their own, even when it makes sense. It would be great if we could write this way instead: module App define_error Error # inherits StandardError by default define_error ServerError, ClientError < Error # inherits App::Error end Which would encourage define errors. I realized that the same could apply to empty class inheritance in general, but errors are much more likely to inherit without adding any features - thus naming specifically ((define_error)) here. Or, as Matz suggested in the comment: module App define_error :Error define_error :ServerError, :ClientError, super: Error end this one looks good too. =end		

History

#1 - 11/17/2012 07:23 PM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

Put the following code to your program:

```
class Module
  def define_error(*errors, superclass: StandardError)
    errors.each do |e|
      self.const_set(e, Class.new(superclass))
    end
  end
end
```

```
end  
end
```

```
def define_error(*errors, **k)  
  Object.define_error(*errors, **k)  
end
```

Matz.