define_method define_method

大林一平/Ippei Obayashi

京都大学数学教室, 京大マイコンクラブ/Dept. Math. Kyoto U., KMC

2011/7/18(RubyKaigi2011)

自己紹介

- 大林一平
- 京都在住
  - Ruby 関西
- `http://www.kmc.gr.jp/~ohai/`
- Ruby/SDL (Ruby でゲーム)
- RRSE(Emacs で Ruby のドキュメントを見るツール)
- るりまプロジェクト
- 数学者

メタプログラミング

- プログラムの情報を問い合わせたり，プログラムを変更するようなプログラム
- Ruby の場合，実行時にいろんなことができる
- class Module, class Class, class Object

Module#define_method

- メソッドを定義するメソッド
- メソッド名の文字列 or シンボルと定義されるメソッドのコードのブロックを渡す
- このブロックは外側のローカル変数にアクセスできる
  - これが重要

これでクラス変数もどきを実装できることは良く知られている

```
module A
  def define_class_attr(name)
    v = nil
    define_method(name){ v }
    define_method("#{name}="){|x| v = x}
  end
end
```

```
class X
  extend A
  define_class_attr :y
end

a = X.new
b = X.new
a.y = 2
a.y # => 2
b.y # => 2
b.y = 3
a.y # => 3
```

では，インスタンス変数もどきは作れるか？

```
class Y
  extend B
  define_attr :z
end

a = Y.new
b = Y.new
a.z = 2
p a.z # => 2
p b.z # => nil
b.z = 3
p a.z # => 2
p b.z # => 3
```

- 当然インスタンス変数は使わないとする
  - 値は適当な環境に保持させる
- define_attr の実行は一度しか行なわれない
  - すなわち各オブジェクトごとに環境を作ることはできない

- つまりオブジェクト毎に確実に呼ばれるメソッドを考える

- つまりオブジェクト毎に確実に呼ばれるメソッドを考える
- new と initialize
  - initialize は何かと面倒
  - new を hook する

- つまりオブジェクト毎に確実に呼ばれるメソッドを考える
- new と initialize
  - initialize は何かと面倒
  - new を hook する
- もう一つ存在する

- つまりオブジェクト毎に確実に呼ばれるメソッドを考える
- new と initialize
  - initialize は何かと面倒
  - new を hook する
- もう一つ存在する
  - 定義しようとしているメソッドそのもの

というわけでこうなる

```
module B
  def define_attr(name)
    define_method(name) {
      nil
    }
    define_method("#{name}="){|v|
      singleton_class.module_eval {
        define_method(name){ v }
        define_method("#{name}="){|w| v = w}
      }
      v
    }
  end
end
```

- 真面目にするんだったらスレッドについて考えましょう
  - 要排他

- 真面目にするんだったらスレッドについて考えましょう
  - 要排他
- そこまで真面目にするようなことか，という話も
  - marshaling で保存されない

- 真面目にするんだったらスレッドについて考えましょう
  - 要排他
- そこまで真面目にするようなことか，という話も
  - marshaling で保存されない
- define_method で define_method
  - 共有される環境を好きに作れる
  - 他の使い道は何だろう？

完

ご静聴ありがとうございました

おまけ

new を hook.

```
module C
 def define_attr(name)
  orig_new = method(:new)
  singleton_class.module_eval {
   define_method(:new){|*args, &b|
    new_obj = orig_new[*args, &b]
    v = nil
    singleton_class.module_eval{
     define_method(name){v}
     define_method("#{name}="){|w| v=w}
    }
   new_obj
   } }
 end; end
```

代入するたびに再定義.

```
module D
  def define_attr(name)
    define_method(name) {
      nil
    }
    define_method("#{name}="){|v|
      singleton_class.module_eval {
        define_method(name){ v }
      }
      v
    }
  end
end
```