Sexy development with ClassX

Keiji Yoshimi (Akakaka.rb)
on TokyoRubyKaigi01

注意

- 東京Ruby会議01当時(08/08/21)はClassXはclassでした。(gem ver 0.0.2) 0.0.3からはmoduleになりますので注意してください。

ClassXとは?

- perlの**Moose**っぽいインターフェース
  - Class::MOP(= CLOSの実装)のラッパー
  - Role
  - **Attribute**

簡単な例

```
require ‘classx’
class Point
  include ClassX
  has :x
end

Point.new({})
#=>
./lib/classx.rb:37:in `initialize':
param: :x is required to {}
(ClassX::AttrRequiredError)
```

簡単な例(続き)

```
require ‘classx’
class Point
  include ClassX
  has :x
end

point = Point.new({ :x => 10 })
point.x #=> 10
point.x = 10
 #=> NoMethodError: private method
`x=' called for #<Point:0x31f114>
```

Q. 書きかえ可能にしたい!!

A. writableを有効に

```
require ‘classx’
class Point
  include ClassX
  has :x, :writable => true
end

point = Point.new({ :x => 10 })
point.x = 20
point.x
  #=> 20
```

Q. newしたときに
データなくてもよい
ようにしたい!!

A. optionalを有効に

```
require ‘classx’
class Point
  include ClassX
  has :x,
    :optional => true,
    :writable => true
end

point = Point.new #=> not error!!
point.x           #=> nil
point.x = 20
point.x           #=> 20
```

Q. デフォルトで初期値が欲しい!!

A. :defaultを使う

```
require ‘classx’
class Point
  include ClassX
  has :x,
    :optional => true,
    :default  => proc { 10 }
end

point = Point.new
point.x  #=> 10
```

Q. なぜProcを渡すのか?

```
require ‘classx’
class Point
  include ClassX
  has :x,
    :optional => true,
    :default  => proc { 10 }
end

point = Point.new
point.x  #=> 10
```

A. インスタンスごとに別なオブジェクトを保持させたいから。

```
require ‘classx’
class Stack
  include ClassX
  has :data,
    :optional => true,
    :default  => proc { [] }
end

Stack.new.data.object_id  #=> 1592010
Stack.new.data.object_id  #=> 1586750
```

Q. 値をvalidationしたい

A. :validateを使う

```
require ‘classx’
class Point
  include ClassX
  has :x,
    :validate => proc {|val|
val.is_a? Fixnum },
    :writable => true
end

point = Point.new({ :x => ‘hoge’ })
  #=> raise
ClassX::InvalidAttrArgument
```

A. :validateを使う

```
require ‘classx’
class Point
  include ClassX
  has :x,
    :validate => proc {|val|
val.is_a? Fixnum },
    :writable => true
end
point = Point.new({ :x => 10 })
point.x = "str"
 #=> raise
ClassX::InvalidAttrArgument
```

A. also :kind_of (:isa)

```
require ‘classx’
class Point
  include ClassX
  has :x,
    :kind_of => Fixnum,
    :writable => true
end
point = Point.new({ :x => 10 })
point.x = "str"
 #=> raise
ClassX::InvalidAttrArgument
```

A. Duck typing?

```
require ‘classx’
class Point
  include ClassX
  has :x,
    :respond_to => :to_int,
    :writable => true
end
point = Point.new({ :x => 10 })
point.x = "str"
 #=> raise
ClassX::InvalidAttrArgument
```

ClassXの書き方のメリット

```
# old style
class YourClass
  def
initialize( a,b,c,d )
  @a, @b, @c, @d =
      a, b, c, d
end
```

```
# with ClassX
class YourClass
  inlucde ClassX
  has :a
  has :b
  has :c
  has :d
end
```

ClassXの書き方のメリット

```
# old style
class YourClass
  def initialize h
    @config = h
  end
end
```

```
# with ClassX
class YourClass
  include ClassX
  has :host,
    :default => proc
{ ... }
  has :port,
    :default => 8080
end
```

Q. コンストラクタ以外の
Hashをとるメソッドで
使えないの?

A. use ClassX::Validate

```
require ‘classx’
require ‘classx/validate’
class YourClass
  include ClassX::Validate
  def run opts={}
     valid_opts = validate opts do
       has :host
       has :port
     end
     valid_opts.host #=> ‘wassr.jp’
  end
end
```

A. CLIアプリが簡単に

```
require ‘classx’
require ‘classx/commandable’
class YourApp
  include ClassX
  extend ClassX::Commandable
  has :file,
    :kind_of => String,
    :desc => ‘filename’
end
YourApp.from_argv
#=>
bin/your_app.rb [options]
        --file VAL      filename
    -h, --help          show this document
```

Thanks!!

Any Question?

site: http://github.com/walf443/classx/

install: gem install classx

git clone git://github.com/walf443/classx.git && cd classx && rake install

Q. ClassXの速度

- A. すごく…遅いです。ただしまだまだ最適化の余地はあるかも。
- ClassXで作ったクラスを1000回newさせるベンチマークだと~~0.4~~s(0.15s)(real)で通常のClassをnewするより~~100~~倍(約35倍)遅い

Q. attribute情報はメタプログラミングできますか?

- A. できます。(割と新しめの機能なので、インターフェースは変わるかもしれません)

```
p = Point.new
p.attribute_of
p.attribute_of[‘x’].class.optional?
p.attribute_of[‘x’].set(10)
p.attribute_of[‘x’].get #=> 10
Point.attribute_of[‘x’].optional?
```

Q. attributeの属性は拡張できますか?

- まだ仕組みを施行錯誤中。
- どういう時にどういう風に拡張したいかの具体的な例が今のところあまり思いついていない。