Ruby 初級者向けレッスン第 8 回

かずひこ@株式会社 ネットワーク応用通信研究所，コウザイ@小波ゼミ

2006 年 7 月 15 日

目次

Rubyist の基礎知識

リファレンスマニュアル

- ウェブのリファレンスマニュアル
  - http://www.ruby-lang.org/ja/man/
- ri（Ruby に同梱のリファレンス表示ツール）
  - （ヘルプ）ri –help
  - （クラス）ri Array
  - （クラスメソッド）ri Array::new
  - （インスタンスメソッド）ri Array#each
- refe（日本語）
  - http://i.loveruby.net/ja/prog/refe.html
- rbbr（日本語・英語）
  - http://ruby-gnome2.sourceforge.jp/hiki.cgi?rbbr

メソッドの表記

- リファレンスマニュアルでは、以下のような表記でメソッドを記述します

Klass#method
: Klass というクラスの method というインスタンスメソッド

Klass::method または Klass.method
: Klass というクラスの method というクラスメソッド

irb（対話型の Ruby）

- 式を入力すると、すぐに実行して式の値を表示します

```
$ irb
irb(main):001:0> 1 + 2 ←式を入力
=> 3 ←式の値を出力
irb(main):002:0> puts "Hello" ←式を入力
Hello ← puts の実行結果
=> nil ← 式の値を出力（puts の返り値は nil）
```

イテレータの使い方

- 「繰り返し」を行うためのメソッド
- Ruby の特徴的な要素の一つ

配列とイテレータ

- 配列の要素を順に処理する each メソッド

```
# each_sample.rb
["dog", "cat", "cow"].each do |elem|
  p(elem)
end

$ ruby each_sample.rb
"dog"
"cat"
"cow"
```

- [”dog”, ”cat”, ”cow”] という配列オブジェクトの each メソッドが「do |elem| p(elem); end」というブロックを引数として呼ばれています
- |〜| で囲まれた部分をブロックパラメータと言います
- p はオブジェクトを表示するメソッドです

イテレータを作ろう

- イテレータとなるメソッドは、ブロックを受け取って、各要素に対してブロックを呼び出します
- メソッド内でブロックを呼び出すには yield 式を使います
- yield は、ブロックをその場で展開するように動作します

yield の動作

- メソッド内で yield を呼ぶと、そのたびにメソッドに指定したブロックが起動されます

```
# yield_test.rb
def three_times
  yield
  yield
  yield
end

three_times do
  p("hello!")
end

$ ruby yield_test.rb
"hello!"
"hello!"
"hello!"
```

- yield の引数がブロックパラメータとなります

```
# yield_test2.rb
def one_two_three
  yield(1)
  yield(2)
  yield(3)
end

one_two_three do |arg|
  p(arg * 2)
end

$ ruby yield_test2.rb
2
```



```
4
6
```

- yield はブロック内で最後に評価した式の値を返します

```
# yield_test3.rb
def call_block
  return yield
end

p call_block{1; 2; 3}

$ ruby yield_test3.rb
3
```

Enumerable モジュール

- each を用いてさまざまなイテレータが定義されています
- each が定義されているクラスにインクルードすれば、各種イテレータが使えます
- 配列やハッシュには最初からインクルードされています

Enumerable モジュールで使えるイテレータ

- 'ri Enumerable' で見てみよう
  - all?, any?, collect, detect, each_cons, each_slice, each_with_index, entries, enum_cons, enum_slice, enum_with_index, find, find_all, grep, include?, inject, map, max, member?, min, partition, reject, select, sort, sort_by, to_a, to_set, zip
- 各メソッドの詳細は、'ri Enumerable#collect' のように見てみよう

イテレータを作ってみよう

- collect (または map)
  - 各要素に対してブロックを評価した結果を全て含む配列を返します

```
# collect_sample.rb
ary = [1, 2, 3, 4, 5].collect do |elem|
  elem * 2
end
p(ary)
```

```
$ ruby collect_sample.rb
[2, 4, 6, 8, 10] # 各要素を 2 倍した配列
```

- collect メソッド

```
class Array
  def collect
    ret = []
    self.each do |elem|
      ret << yield(elem)
    end
    return ret
  end
end
```

- まず、返り値を入れる変数 ret を空の配列で作ります
- 各要素を引数にして yield した結果を、順に ret に追加します
- 最後に ret を返します

演習問題 1

- 以下のイテレータメソッドを、自分で定義してみましょう
  - select（各要素に対してブロックを評価した値が真であった要素を全て含む配列を返します）
  - reject（各要素に対してブロックを評価し、その値が偽であった要素を集めた新しい配列を返します）
  - detect（要素に対してブロックを評価した値が真になった最初の要素を返します）
  - その他、Enumerable モジュールのメソッド
- イテレータメソッド定義プログラムの雛型（my_enumerable.rb）

```
module MyEnumerable
  def collect
    ret = []
    self.each do |elem|
      ret << yield(elem)
    end
    return ret
  end
```



```
  def select
    ...
  end
end
```

- テスト用プログラムの雛型（test_my_enumerable.rb）

```
require 'my_enumerable'
require 'test/unit'

class TestMyEnumerable < Test::Unit::TestCase
  def setup
    @array = [1, 2, 3, 4, 5]
    @my_array = [1, 2, 3, 4, 5]
    @my_array.extend(MyEnumerable)
  end

  def test_collect
    array_collect = @array.collect do |i|
      i * 2
    end
    my_array_collect = @my_array.collect do |i|
      i * 2
    end
    assert_equal(array_collect, my_array_collect)
  end

  def test_select
    ...
  end
end
```

- テスト用プログラムの setup メソッドでは、@my_array オブジェクトの各イテレータメソッドを MyEnumerable モジュールのメソッドで上書きするために、extend メソッドを使っています。

Object#extend(module ...)
引数で指定したモジュールで定義されているメソッドが self の特異メソッドとして追加されます。あるオブジェクトだけにモジュールの機能を追加したいときに使用します。

```
module Foo
  def a
    'ok'
  end
```

```
end

obj = Object.new
obj.extend Foo
p obj.a         #=> "ok"
```

extend の機能は、「特異クラスに対する include」と言い替えることもできます。

演習問題２

- 何かイテレータを提案して実装してみましょう
  - （例）指定した回数サイコロをふって、その値をブロックパラメータとしてブロックを実行するメソッド
  - （例）文字列オブジェクトからなる配列に対して、大文字小文字を区別せずにアルファベット順にブロックを実行するメソッド

参考文献

- 『たのしい Ruby』 ISBN:4797314087
- 『プログラミング Ruby』 ISBN:4894714531

今後の情報源

- 公式 Web サイト http://www.ruby-lang.org/
- リファレンスマニュアル http://www.ruby-lang.org/ja/man/
- 日本 Ruby の会 http://jp.rubyist.net/
- Rubyist Magazine http://jp.rubyist.net/magazine/
- ふぇみにん日記 http://kazuhiko.tdiary.net/