Ruby 初級者向けレッスン 第 11 回

okkez@Ruby 関西,サカイ@小波ゼミ

2007 年 02 月 17 日

irb について

- Intaractive Ruby の略
- Ruby をインストールするともれなく付いてきます。
- Ruby の式を標準入力から簡単に入力・実行することができます。

簡単な使用例

```
$ irb
irb(main):001:0> 10+10          # 通常の計算
=> 20                           # 答えが表示される
irb(main):002:0> def add(x,y)   # メソッド定義もできる
irb(main):003:1>   x + y        # 継続行
irb(main):004:1> end
=> nil
irb(main):005:0> add(10,20)     # さっき作ったメソッドを使う
=> 30                           # メソッドの返値が表示される
irb(main):006:0> exit           # 終了する。(quit,abort でも ok)
$
```

irb でよく使うパターン

- overflow しない電卓として
- shell 代わりに
  - irbsh というものがあるらしい
- メソッドの動作確認
  - 正規表現のマッチング確認
  - フォーマット文字列の表示確認
- メソッドを探す
  - あのクラスにあるはずのメソッドを探したい
  - クラスの継承関係を調べたい
  - あのモジュールは include されているのか

irb のコマンドラインオプション

```
$ irb --help
Usage:  irb.rb [options] [programfile] [arguments]
  -f                ~/.irbrc を読み込まない.
  -m                bc モード(分数, 行列の計算ができる)
  -d                $DEBUG を true にする(ruby -d と同じ)
  -r load-module    ruby -r と同じ.
  -I path           $LOAD_PATH に path を追加する.
  --inspect         結果出力に inspect を用いる(bc モード以外はデフォルト).
  --noinspect       結果出力に inspect を用いない.
  --readline        readline ライブラリを利用する.
  --noreadline      readline ライブラリを利用しない.
  --prompt prompt-mode/--prompt-mode prompt-mode
                    プロンプトモードを切替えます. 現在定義されているプ
                    ロンプトモードは, default, simple, xmp, inf-ruby が
                    用意されています.
  --inf-ruby-mode   emacs の inf-ruby-mode 用のプロンプト表示を行なう. 特
                    に指定がない限り, readline ライブラリは使わなくなる.
  --simple-prompt   非常にシンプルなプロンプトを用いるモードです.
  --noprompt        プロンプト表示を行なわない.
  --tracer          コマンド実行時にトレースを行なう.
  --back-trace-limit n
                    バックトレース表示をバックトレースの頭から n, 後ろ
                    から n だけ行なう. デフォルトは 16
  --irb_debug n     irb のデバッグデバッグレベルを n に設定する(利用しな
                    い方が無難でしょう).
  -v, --version     irb のバージョンを表示する
```

定義済みプロンプト

- XMP
- DEFAULT
- NULL
- CLASSIC
- SIMPLE
- INF_RUBY

実際の定義はこんな感じ。

```
IRB.conf[:PROMPT] =
  {:XMP=>
    {:PROMPT_I=>nil,                    # 通常時のプロンプト
     :PROMPT_S=>nil,                    # 文字列などの継続行のプロンプト
     :PROMPT_C=>nil,                    # 式が継続しているときのプロンプト
     :RETURN=>"    ==>%s\n"},           # メソッドから戻るときのプロンプト
   :DEFAULT=>
    {:PROMPT_I=>"%N(%m):%03n:%i> ",
     :PROMPT_S=>"%N(%m):%03n:%i%l ",
     :PROMPT_C=>"%N(%m):%03n:%i* ",
     :RETURN=>"=> %s\n"},
   :NULL=>
     {:PROMPT_I=>nil,
      :PROMPT_S=>nil,
      :PROMPT_C=>nil,
      :RETURN=>"%s\n"},
   :CLASSIC=>
    {:PROMPT_I=>"%N(%m):%03n:%i> ",
     :PROMPT_S=>"%N(%m):%03n:%i%l ",
     :PROMPT_C=>"%N(%m):%03n:%i* ",
     :RETURN=>"%s\n"},
   :SIMPLE=>
    {:PROMPT_I=>">> ",
     :PROMPT_S=>nil,
     :PROMPT_C=>"?> ",
     :RETURN=>"=> %s\n"},
   :INF_RUBY=>
    {:PROMPT_I=>"%N(%m):%03n:%i> ",
     :PROMPT_S=>nil,
     :PROMPT_C=>nil,
     :AUTO_INDENT=>true,
     :RETURN=>"%s\n"}}
```

プロンプトで使用可能なフォーマット文字列

- %N
  起動しているコマンド名
- %m
  main オブジェクト（self）を to_s した文字列
- %M
  main オブジェクト（self）を inspect した文字列
- %l
  文字列中のタイプを表す（", ', /,], ']'は%w の中の時）
- %NNi
  インデントのレベル(i)を、NN 桁に右詰めした文字列。 NN は省略可能。
- %NNn
  行番号(n)を、NN 桁に右詰めした文字列。 NN は省略可能。
- %%
  文字「%」それ自体

irb の設定ファイル

- *nix ユーザーなら、~/.irbrc に記述すればよい
- Windows ユーザーなら、irb.bat のあるディレクトリに.irbrc, irb.rc, _irbrc, $irbrc のいずれかのファイルを置けばよい
- ファイルを探す順番は、上で書いた順番。

設定例

```
# TAB 補完を有効にする
requrie 'irb/completion'
# yaml ライブラリを読み込む
require 'yaml'
# pp も使えるように
require 'pp'

# ruby 1.8.3 以降なら以下でヒストリを自動保存してくれる
# 1.8.2 以前で同じ事をやろうとすると結構がりがり書かないといけない
require 'irb/completion' # TAB で補完
require 'irb/ext/save-history'
ARGV.concat [ "--readline", "--prompt-mode", "simple" ] # 無くても動く？
IRB.conf[:SAVE_HISTORY] = 100
IRB.conf[:HISTORY_FILE] = "#{ENV['HOME']}/.irb-save-history"
```

irb での exit, quit, abort の違い

- exit, quit は irb_exit に対するエイリアス
  - サブ irb で使用した場合は、その irb だけ終了する。
- abort は Kernel#abort を呼び出す。
  - サブ irb で使用した場合は、全ての irb を終了する。
  - Kernel#abort は異常終了させるためのものなので通常は irb_exit を使用すること。

load と require

- load は呼び出すたびに対象ファイルを読み直す
  - 自作ライブラリの動作確認時に便利
- require は一度読み込んだファイルは二度と読み直さない

irb 応用例

- rails の script/console
- iar（Interactive Active Record）
- irbdb
- など

bc モード（-m）

- mathn ライブラリを読み込む
- 数値計算向き
- 結果を to_s して表示する

演習

いろいろな計算をしてみよう

通常モードと bc モードで計算結果の違いを見てみましょう。

1. 1 + 1
2. 10 / 4
3. 100 ** 10000
4. sqrt(16 / 36)
5. BMI を計算してみましょう。
   BMI = 体重(kg) / 身長(m) ** 2
6. 1 年は何時間でしょう？
7. 10 年は何分でしょう？
8. わたしが生まれてから 9 億 3400 万秒経っているとしたら今何歳でしょう？

irb のオプションをいろいろといじってみよう

動作中、一時的に結果表示を消してみよう

```
irb(main):007:0> 1+1
=> 2                                # 計算結果が表示される
irb(main):008:0> **********         # 何かする
irb(main):009:0> 1+1                # もう一度計算する
irb(main):010:0>                    # 何も表示されない
```

動作中にプロンプトの一部を変更してみよう

全体を変更する場合は以下のようにしますが、一部だけ変更するにはどうするでしょう？

```
irb(main):010:0> conf.prompt_mode = :SIMPLE
>> conf.prompt_mode = :DEFAULT
irb(main):012:0>
```

メソッドの動作確認をしてみよう

各メソッドの使い方はリファレンスマニュアルなどで確認してください。

1. Array#zip
2. Kernel#rand, Kernel#srand
3. require 'yaml'して、 Kernel#y にいろいろなオブジェクトを渡してみよう

作ってみよう

irb で動作確認しながら作ってみましょう。

じゃんけんプログラムを作りましょう。

仕様

- あなたとコンピュータでじゃんけんをします。
- コンピュータの手はランダムです。
- あなたの手はキーボードから入力できるようにしてください。
- 勝ち、負け、引き分けの回数を記録します。
- 毎回、勝ち、負け、引き分けの回数を表示します。
- 終了したら、勝率を表示して勝率に応じて気の利いたメッセージを表示しましょう。

実行イメージ

```
じゃんけんゲーム開始
手を入力してください(G:ぐー,C:ちょき,P:ぱー,Q:終了)
> G [RET]
あたたの手 : ぐー
コンピュータの手 : パー
コンピュータの勝ちです。
あたたの
0 勝, 1 敗, 0 分 です。
手を入力してください(G:ぐー,C:ちょき,P:ぱー,Q:終了)
> Q [RET]
終了します。
あなたの勝率は 0 % です。弱いですね。
さようなら。
```

まとめ

- irb は便利ですね。
- 「あ」、と思ったら irb
- Ruby リファレンスマニュアル刷新計画もよろしくお願いします！

参考文献

- 初めてのプログラミング
  http://www.oreilly.co.jp/books/4873112923/
- プログラミング Ruby 第2版 言語編
  http://ssl.ohmsha.co.jp/cgi-bin/menu.cgi?ISBN=4-274-06642-8
- たのしいRuby 第2版
  http://shop.sbcr.jp/bm_detail.asp?sku=4797336617

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/
- るりま Wiki
  http://doc.loveruby.net/wiki/