Ruby 初級者向けレッスン第 27 回

okkez @ Ruby 関西

2009 年 04 月 25 日

今回の内容

- Ruby を用いてコマンドラインアプリケーションを作成する
- 作り方を説明します

今回のゴール

- Ruby で簡単なコマンドラインアプリケーションを作成する

コマンドラインアプリケーションって？

- コンソールでコマンドを叩いて操作するアプリケーションです
- 慣れると GUI よりも便利

Ruby で作成されたコマンドラインアプリケーションの例

- RubyGems
  - パッケージ管理
  - Ruby1.9.1 から標準添付
- Rake
  - ビルドツール
  - Ruby1.9.1 から標準添付
- BitClust
  - るりま
  - 色々なツールが含まれている
- rak

  - grep の Ruby 実装
- Ruby on Rails
  - 特に rails コマンド
- 他にも色々

使い捨てスクリプトではない理由

- 自分以外の人に使ってもらう
- 未来の自分に使ってもらう
- 長く使う
- 自己満足

コマンドラインアプリケーションにするといい場合

- 何回も繰り返す必要がある場合
- cron などで自動実行することがある場合
- テストなどの自動化の一部になっている場合
- マウスに触りたくない場合

コマンドラインアプリケーションの作り方

- やりたいことをだらだらとべた書きする
- main だけ作ってその中にやりたいことをべた書きする
- やりたいことがある程度、複雑な場合はメソッドをいくつか作成してそれらを main から呼び出す
- もっと複雑な場合はクラスやモジュールを作成する
- さらに汎用的にする場合はやりたいことをライブラリ化する

べた書きの例

どこかの Redmine にアクセスしてあるプロジェクトのチケットの ID とタイトル一覧を取得して表示する。

```
01: #!/usr/bin/ruby
02: # -*- coding: utf-8 -*-
03:
04: require 'rubygems'
05: require 'mechanize'
06:
07: agent = WWW::Mechanize.new
08: agent.user_agent_alias = 'Mac Safari'
09: page = agent.get('http://redmine.ruby-lang.org/projects/rurema/issues')
10:
11: page.root.search('table.list.issues').search('tr').each do |e|
12:   next unless e.search('th').empty?
13:   puts e.search('td > a').map{|a| a.text }.join(' : ')
14: end
```

解説

- 一行目は shebang
- 二行目は magic comment

メリット

- 上から下に読めば何をしているかわかる
- 書くのが簡単

デメリット

- やることを上から順にしか書けない
- テストしにくい

main だけ作ってみた例

さっきの例と同じことをやっている。

```
01: #!/usr/bin/ruby
02: # -*- coding: utf-8 -*-
03:
```

```
04: require 'rubygems'
05: require 'mechanize'
06:
07: def main
08:   agent = WWW::Mechanize.new
09:   agent.user_agent_alias = 'Mac Safari'
10:   page = agent.get('http://redmine.ruby-lang.org/projects/rurema/issues')
11:
12:   page.root.search('table.list.issues').search('tr').each do |e|
13:     next unless e.search('th').empty?
14:     puts e.search('td > a').map{|a| a.text }.join(' : ')
15:   end
16: end
17:
18: main
```

メリットとデメリットもさっきの例と変わらない。

いくつかメソッドに分けてみた例

priority が Normal 以外のチケットの URL リストを表示する。

```
01: #!/usr/bin/ruby
02: # -*- coding: utf-8 -*-
03:
04: require 'rubygems'
05: require 'mechanize'
06:
07: def main
08:   agent = WWW::Mechanize.new
09:   agent.user_agent_alias = 'Mac Safari'
10:   page = agent.get('http://redmine.ruby-lang.org/projects/rurema/issues')
11:
12:   page.root.search('table.list.issues').search('tr').each do |e|
13:     next unless e.search('th').empty?
14:     next unless /Normal/ =~ status(e)
15:     puts ticket_url(e.search('td > a').first.text)
16:   end
17: end
18:
19: def ticket_url(id)
20:   %Q!http://redmine.ruby-lang.org/issues/show/#{id}!
21: end
22:
```

```
23: def status(tr)
24:   tr.search('td.priority').text
25: end
26:
27: if __FILE__ == $0
28:   main
29: end
```

ちょっとだけわかりやすくなった。

メリット

- メソッドごとにやりたいことが明確になる
- メソッドごとに比較的テストがしやすくなる
- 読んでわかりやすくなる

デメリット

- トップレベルの名前空間を使ってしまう

クラスやモジュールを作ってみた例

チケットをクラスにしてみた例

```
01: #!/usr/bin/ruby
02: # -*- coding: utf-8 -*-
03:
04: require 'rubygems'
05: require 'mechanize'
06:
07: def main
08:   agent = WWW::Mechanize.new
09:   agent.user_agent_alias = 'Mac Safari'
10:   page = agent.get('http://redmine.ruby-lang.org/projects/rurema/issues')
11:   tickets = []
12:   page.root.search('table.list.issues').search('tr').each do |e|
13:     next unless e.search('th').empty?
14:     tickets << Ticket.new(e.search('td > a').first.text)
15:     tickets.last.page = agent.get(tickets.last.url)
16:     break # すべて実行するのは自重する
17:     sleep 2
18:   end
19:   puts tickets.first.title
```

```
20:   puts tickets.first.description
21: end
22:
23: class Ticket
24:
25:   attr_accessor :id, :page
26:
27:   def initialize(id, page = nil)
28:     @id = id
29:     @page = page
30:   end
31:
32:   def url
33:     %Q!http://redmine.ruby-lang.org/issues/show/#{@id}!
34:   end
35:
36:   def title
37:     @page.search('div#content > div.issue > h3').text
38:   end
39:
40:   def description
41:     @page.search('div#content > div.issue > div.issue-description').text
42:   end
43:
44:   def comments
45:     # TODO
46:   end
47:
48: end
49:
50: if __FILE__ == $0
51:   main
52: end
```

ダメなところ

- 色々と即値な部分がある (今回は少ないけど)
- 他のプロジェクトや他の Redmine を参照したい場合にスクリプトを書き換える必要がある

ちょこっと汎用化してみた例

本来はもっとやるべきことがあるけど、説明用なので簡易にしてみました。

```
01: #!/usr/bin/ruby
02: # -*- coding: utf-8 -*-
03:
04: require 'optparse'
05:
06: require 'rubygems'
07: require 'mechanize'
08:
09: def main
10:   base_url = nil
11:   project_name = nil
12:   user = nil
13:   pass = nil
14:   parser = OptionParser.new
15:   parser.on_tail('-h', '--help', 'Print this message and quit.') do
16:     puts parser.help
17:     exit 0
18:   end
19:   parser.on('-b', '--base-url URL', 'Specify base url of Redmine') do |url|
20:     base_url = url
21:   end
22:   parser.on('-p', '--project NAME', 'Specify project name') do |name|
23:     project_name = name
24:   end
25:   parser.on('--user USER', 'Login as USER') do |u|
26:     user = u
27:   end
28:   parser.on('--password PASS', 'Use this password when login.') do |pw|
29:     pass = pw
30:   end
31:   begin
32:     parser.parse!
33:   rescue OptionParser::ParseError => err
34:     $stderr.puts err.message
35:     $stderr.puts parser.help
36:     exit 1
37:   end
38:
39:   agent = Redmine::Agent.new(base_url, project_name, user, pass)
40:   p agent.all_tickets
41:
42: end
43:
```

```
44: module Redmine
45: end
46:
47: class Redmine::Agent
48:
49:   attr_reader :agent
50:
51:   def initialize(base_url, project_name = nil, user = nil, pass = nil)
52:     @base_url = base_url
53:     @project_name = project_name
54:     @user = user
55:     @pass = pass
56:     @agent = WWW::Mechanize.new
57:   end
58:
59:   def login
60:     page = File.join(@base_url, 'login')
61:     form = page.forms[1]
62:     form.username = @user
63:     form.password = @pass
64:     @agent.submit(form, form.buttons.first)
65:   end
66:
67:   def issues_url
68:     File.join(@base_url, 'projects', @project_name, issues)
69:   end
70:
71:   def issues_page
72:     @agent.get(issue_url)
73:   end
74:
75:   # FIXME 一ページに収まらない場合がある
76:   def all_tickets
77:     result = []
78:     issues_page.search('table.list.issues > tbody > tr').each do |tr|
79:       result << Ticket.new(@base_url, tr.search('td > a').first.text)
80:     end
81:     result
82:   end
83: end
84:
85: class Redmine::Ticket
86:
```

```
87:   attr_accessor :id, :page
88:
89:   def initialize(base_url, id, page = nil)
90:     @base_url = base_url
91:     @id = id
92:     @page = page
93:   end
94:
95:   def url
96:     File.join(@base_url, '/issues/show/', @id)
97:   end
98:
99:   def title
100:     @page.search('div#content > div.issue > h3').text
101:   end
102:
103:   def description
104:     @page.search('div#content > div.issue > div.issue-description').text
105:   end
106:
107:   def comments
108:     # TODO
109:   end
110:
111:   def update
112:     page = @page.link_with(:text => /更新/)
113:     form = page.forms[1]
114:     yield form
115:     @agent.submit(form, form.buttons.first)
116:   end
117:
118: end
119:
120:
121: if __FILE__ == $0
122:   main
123: end
```

よいところ

- 汎用的になった
  - いろいろなプロジェクトを参照することができる

- いろいろな Redmine を参照することができる

悪いところ

- すべてが一つのファイルに収まっている
- ここまでやるならきちんとライブラリ化するべき

演習

自分でコマンドラインアプリケーションを作ってみましょう。使用するライブラリは自由です。
初級者の人は、演習 0 からやってください。そうでない方はどこからはじめても自由です。

演習 0

引数を受け取るコマンドラインアプリケーションを作ってください。

- 指定された二つの数値を用いて四則演算の実行した結果を表示するスクリプトを書いてください。
- ゼロから指定した数値までの和を計算した結果を表示するスクリプトを書いてください。
- optparse ライブラリを使って上のスクリプトを書いて見てください。

演習 1

引数をいくつか取るコマンドラインアプリケーションを作ってください。以下の仕様を満たすようにしてください。

- CSV ファイルを分割します。分割数 n を指定できます。
- 読み込むファイル名を指定できます。
- 出力先ディレクトリを指定できます。
- 出力ファイル名のプレフィクスを指定できます。

演習 2

cat コマンドを実装してください。実装するオプションは、お任せします。
see also cat(1)

まとめ

- コマンドラインは便利。
- GUI を作るのはちょっと面倒。
- 今日の話題は仕事で使えます。

今後の情報源

公式 Web サイト
: http://www.ruby-lang.org/

Ruby リファレンスマニュアル刷新計画
: http://doc.loveruby.net/

日本 Ruby の会
: http://jp.rubyist.net/

Rubyist Magazine
: http://jp.rubyist.net/magazine/

okkez weblog
: http://typo.okkez.net/

Ruby Reference Manual (beta)
: http://doc.okkez.net/