卒業論文

コマンドラインツール作成ライブラリ Thor による hikiutils の書き換え

関西学院大学　理工学部　情報科学科

27013554 山根亮太

2017 年 3 月

指導教員　　西谷　滋人　教授

目次

コマンドラインツール作成ライブラリ Thor による hikiutils の書き換え関西学院大学理工学部情報科学科 27013554 山根亮太

目次

1　概要

2 序論

2.1 目的

本研究では hiki の編集作業をより容易にするためのツールの開発を行った．hiki は通常 web 上で編集を行っているが,GUI と CUI が混在しており, 操作に不便な点がある．そこで, 編集操作が CUI で完結するために開発をされたのが hikiutils である．しかし, そのユーザインタフェースにはコマンドが直感的でないという問題点がある．そこで，Thor というコマンドラインツール作成ライブラリを用いる．optparse というコマンドライン解析ライブラリを使用している hikiutils を新たなコマンドライン解析ライブラリを使用することコマンドを書き換え，より直感的なコマンドにすることが可能である．

2.2 既存システムの背景

2.2.1 hiki

hiki とはプログラミング言語 Ruby を用いられることで作られた wiki クローンの 1 つである．hiki の主な特徴として

- オリジナル wiki に似たシンプルな書式
- プラグインによる機能拡張
- 出力する HTML を柔軟に変更可能
- ページにカテゴリ付けできる
- CSS を使ったテーマ機能
- 携帯端末可能
- InterWiki のサポート
- HikiFarm に対応
- ページの追加，編集がしやすい

等がある [1].

2.2.2 hikiutils

hikiutils は hiki の編集作業を容易に行うことができるよう にするツール群であり, プログラミング言語 Ruby のライブ ラリである gem フォーマットに従って提供されている [2]. hikiutils は CLI(Command Line Interface) で操作するため, オプション解析をお

こなう必要がある. gem には, この用途 に適合したライブラリがいくつも提供されている [3]. この中 で, あまり利用頻度は高くないが古くから開発され, 使用例 が広く紹介されている optparse を利用している.

3 結果

3.1 開発の結果

開発の結果コマンドを以下のように書き換えることができた

表 1

変更前	変更後	コマンド一覧
add		add sources info
checkdb		check database file
datebase FILE		read datebase file
display FILE		display converted hikifile
edit FILE	open	open file
euc FILE		translate file to euc
help [COMMAND]		Describe available commands or one specific command
list [FILE]	ls	list files
move [FILE]	mv— move file	
remove [FILE]	rm	remove files
rsync		rsync files
show		ls とかぶる.
target VAL	cd	set target id
update FILE	touch	update file
version		show program version

hikiutils のコマンドライン解析ツールを optparse から thor に換えることでコマンドの書き換えを行うことができた．また，プログラムのコードも optparse より thor のほうが短く書け，簡単にコマンドを作成することができた．

3.2 thor と optparse のコードの比較

3.2.1 Thor とは

Thor とは，コマンドラインツールの作成を支援するライブラリのことである．git や bundler のようにサブコマンドを含むコマンドラインツールを簡単に作成することができる [4].

■Thor の基本的な流れ

1. Thor を継承したクラスのパブリックメソッドがコマンドになる
2. クラス.start(ARGV) でコマンドラインの処理をスタートする

という流れである [4].

3.2.2 optparse とは

optparse モジュールとは，getopt よりも簡便で，柔軟性に富み，かつ強力なコマンドライン解析ライブラリである．optparse では，より宣言的なスタイルのコマンドライン解析手法，すなわち OptionParser のインスタンスでコマンドラインを解析するという手法をとっている．これを使うと，GNU/POSIX 構文でオプションを指定できるだけでなく，使用法やヘルプメッセージの生成も行える [5].

■optparse の基本的な流れ

1. OptionParser オブジェクト opt を生成する
2. オプションを取り扱うブロックを opt に登録する
3. opt.parse(ARGV) でコマンドラインを実際に parse する

という流れである [6].

3.2.3 コードの解説

■Thor の定義

1. Hikithor::CLI.start(ARGV) が呼ばれる
2. initialize メソッドが呼ばれる
3. これでは Thor の initialize メソッドが呼ばれない
4. super を書くことで Thor の initialize メソッドが呼ばれる

図 1

optparse では require で optparse を呼ぶだけでいいが，Thor では require で Thor を呼び CLI クラスで継承し initialize メソッドに super を書くことで Thor の initialize を呼ぶ必要がある．

■実際のコード

```
1  # -*- coding: utf-8 -*-
2  require 'thor'
3  require 'kconv'
4  require 'hikidoc'
5  require 'erb'
6  require "hikiutils/version"
7  require "hikiutils/tmarshal"
8  require "hikiutils/infodb"
9  require 'systemu'
10 require 'fileutils'
11 require 'yaml'
12 require 'pp'
13
14 module Hikithor
15
16   DATA_FILE=File.join(ENV['HOME'],'.hikirc')
17   attr_accessor :src, :target, :editor_command, :browser, :
         data_name, :l_dir
18
19   class CLI < Thor
```

```
20      def initia
21          super
22          @data_n
                glob
23          data_pa
24          DataFil
25
26          file =
27          @src =
28          file.cl
29          @target
30          @l_dir=
31          browser
32          @browse
33          p edito
34          @editor
                : e
35      end
```

■hikiutils の実行

- Thor

図 2

1. hiki_thor の Hiki
   ぶ{{br}}
2. hikiutils_thor.rb

- optparse

図 3

1. Hiki の HikiUtils::Command.run(ARGV) で hikiutils.rb の run メソッドを呼ぶ
2. new(argv).execute で execute メソッドが実行される

このように optparse では実行を行うためのメソッドが必要であるが，Thor ではクラスのメソッドを順に実行していくため run メソッドと execute メソッドは必要ない．

■実際のコード

- Thor

```
1 #!/usr/bin/env ruby
2
3 require "hikiutils_thor"
4
5 Hikithor::CLI.start(ARGV)
```

```
# -*- coding: utf-8 -*-
require 'thor'
require 'kconv'
require 'hikidoc'
require 'erb'
require "hikiutils/version"
require "hikiutils/tmarshal"
require "hikiutils/infodb"
require 'systemu'
require 'fileutils'
require 'yaml'
require 'pp'

module Hikithor

  DATA_FILE=File.join(ENV['HOME'],'.hikirc')
  attr_accessor :src, :target, :editor_command, :browser, :
      data_name, :l_dir

  class CLI < Thor
   def initialize(*args)
      super
      @data_name=['nick_name','local_dir','local_uri','
          global_dir','global_uri']
      data_path = File.join(ENV['HOME'], '.hikirc')
      DataFiles.prepare(data_path)

      file = File.open(DATA_FILE,'r')
      @src = YAML.load(file.read)
      file.close
      @target = @src[:target]
      @l_dir=@src[:srcs][@target][:local_dir]
      browser = @src[:browser]
      @browser = (browser==nil) ? 'firefox' : browser
```

- optparse

```
#!/usr/bin/env ruby

require "hikiutils"

HikiUtils::Command.run(ARGV)
```

```
    def self.run(argv=[])
```

```
2         print "hikiutils: provide utilities for helping hiki
              editing.\n"
3         new(argv).execute
4       end
5
6       def execute
7         @argv << '--help' if @argv.size==0
8         command_parser = OptionParser.new do |opt|
9           opt.on('-v', '--version','show program Version.') {
                |v|
10            opt.version = HikiUtils::VERSION
11            puts opt.ver
12          }
13          opt.on('-s', '--show','show sources') {show_sources}
14          opt.on('-a', '--add','add sources info') {
                add_sources }
15          opt.on('-t', '--target VAL','set target id') {|val|
                set_target(val) }
16          opt.on('-e', '--edit FILE','open file') {|file|
                edit_file(file) }
17          opt.on('-l', '--list [FILE]','list files') {|file|
                list_files(file) }
18          opt.on('-u', '--update FILE','update file') {|file|
                update_file(file) }
19          opt.on('-r', '--rsync','rsync files') {rsync_files}
20          opt.on('--database FILE','read database file') {|
                file| db_file(file)}
21          opt.on('--display FILE','display converted hikifile'
                ) {|file| display(f\
22 ile)}
23          opt.on('-c', '--checkdb','check database file') {
                check_db}
24          opt.on('--remove FILE','remove file') {|file|
                remove_file(file)}
25          opt.on('--move FILES','move file1,file2',Array) {|
                files| move_file(file\
26 s)}
27          opt.on('--euc FILE','translate file to euc') {|file|
                 euc_file(file) }
28          opt.on('--initialize','initialize source directory')
                 {dir_init() }
29        end
30        begin
31          command_parser.parse!(@argv)
32        rescue=> eval
33          p eval
34        end
```

```
35          dump_so
36          exit
37        end
```

■コマンドの表示と実行

- Thor

図 4

1. コマンド名，コマンドの説明を一覧に表示させる
2. パブリックメソッドのコマンドを別のコマンド名でも実行できるようにする
3. コマンドの命令の実行コード

- optparse

{{attach_view(hikiutils_yamane.005.jpg)}} よって，optparse では OptionParser オブジェクト opt を生成を行い，コマンドを opt に登録して一覧に表示するメソッドとそれぞれコマンドの実行処理が書かれたメソッドがあるが，thor ではそれぞれの desc で一覧を表示し map とパブリックメソッドでコマンドの実行処理を行うためコードが短くなる．

■実際のコード

- Thor

```
1     desc 'show,--show', 'show␣sources'
2     map "--show" => "show"
3     def show
4       printf("target_no:%i\n",@src[:target])
5       printf("editor_command:%s\n",@src[:editor_command])
6       @i_size,@n_size,@l_size,@g_size=3,5,30,15 #i,g_size
            are fixed
7       n_l,l_l=0,0
8       @src[:srcs].each_with_index{|src,i|
9         n_l =(n_l= src[:nick_name].length)>@n_size? n_l:
              @n_size
10        l_l =(l_l= src[:local_dir].length)>@l_size? l_l:
              @l_size
11      }
12      @n_size,@l_size=n_l,l_l
13      command = Command.new
14      header = command.display_format('id','name','local␣
            directory','global␣uri',@i_size,@n_size,@l_size,
            @g_size)
15
16      puts header
17      puts '-' * header.size
18
19      @src[:srcs].each_with_index{|src,i|
20        target = i==@src[:target] ? '*':'␣'
21        id = target+i.to_s
22        name=src[:nick_name]
23        local=src[:local_dir]
24        global=src[:global_uri]
25        puts command.display_format(id,name,local,global,
              @i_size,@n_size,@l_size,@g_size)
26      }
27    end
```

- optparse

```
1     def execute
2       @argv << '--help' if @argv.size==0
3       command_parser = OptionParser.new do |opt|
4         opt.on('-v', '--version','show␣program␣Version.') {
              |v|
```

```
5           opt.version = HikiUtils::VERSION
6           puts opt.ver
7         }
8         opt.on('-s', '--show','show sources') {show_sources}
9         opt.on('-a', '--add','add sources info') {
              add_sources }
10        opt.on('-t', '--target VAL','set target id') {|val|
              set_target(val)}
11        opt.on('-e', '--edit FILE','open file') {|file|
              edit_file(file) }
12        opt.on('-l', '--list [FILE]','list files') {|file|
              list_files(file)}
13        opt.on('-u', '--update FILE','update file') {|file|
              update_file(file) }
14        opt.on('-r', '--rsync','rsync files') {rsync_files}
15        opt.on('--database FILE','read database file') {|
              file| db_file(file)}
16        opt.on('--display FILE','display converted hikifile'
              ) {|file| display(file)}
17        opt.on('-c', '--checkdb','check database file') {
              check_db}
18        opt.on('--remove FILE','remove file') {|file|
              remove_file(file)}
19        opt.on('--move FILES','move file1,file2',Array) {|
              files| move_file(files)}
20        opt.on('--euc FILE','translate file to euc') {|file|
               euc_file(file)}
21        opt.on('--initialize','initialize source directory')
               {dir_init() }
22      end
23      begin
24        command_parser.parse!(@argv)
25      rescue=> eval
26        p eval
27      end
28      dump_sources
29      exit
30    end
31
32    def show_sources()
33      printf("target_no:%i\n",@src[:target])
34      printf("editor_command:%s\n",@src[:editor_command])
35      check_display_size()
36      header = display_format('id','name','local directory',
            'global uri')
37
38      puts header
```

```
39        puts '-' * header.size
40
41        @src[:srcs].each_with_index{|src,i|
42          target = i==@src[:target] ? '*':' '
43          id = target+i.to_s
44          name=src[:nick_name]
45          local=src[:local_dir]
46          global=src[:global_uri]
47          puts display_format(id,name,local,global)
48        }
49      end
50
51      def add_sources
52        cont = {}
53        @data_name.each{|name|
54          printf("%s ?", name)
55          tmp = gets.chomp
56          cont[name.to_sym] = tmp
57        }
58        @src[:srcs] << cont
59        show_sources
60      end
```

[1] 「hiki」,hikiwiki.org/ja/about.html ,2017/1/30 アクセス.{{br}} [2] 「hikiutils」,https://rubygems.org/gems/hikiutils ,2017/1/30 アクセス.{{br}} [3]「The Ruby Toolbox」, CLI Option Parsers,https://www.ruby-toolbox.com/categories/CLI_Option_Pa ,2017/1/30 アクセス.{{br}} [4]「Thor の使い方まとめ」, http://qiita.com/succi0303/items/3256(,2017/1/30 アクセス.{{br}} [5] 「15.5. optparse - コマンドラインオプション解析器」, http://docs.python.jp/2/library/optparse.html ,2017/1/30 アクセス.{{br}} [6]「library optparse」, https://docs.ruby-lang.org/ja/latest/library/optparse.html ,2017/1/30 アクセス.{{br}}