

File List

1. [./lib/github_exporter.rb.xhtml](#)
2. [./lib/github_exporter/cli.rb.xhtml](#)
3. [./lib/github_exporter/exporter.rb.xhtml](#)
4. [./lib/github_exporter/github_exporter.rb.xhtml](#)
5. [./lib/github_exporter/logger.rb.xhtml](#)
6. [./lib/github_exporter/version.rb.xhtml](#)
7. [./test/lib/github_exporter/test_github_exporter.rb.xhtml](#)
8. [./test/test_helper.rb.xhtml](#)

```
1 require_relative "github_exporter/version"
2 require_relative "github_exporter/logger"
3 require_relative "github_exporter/github_exporter"
4 require_relative "github_exporter/exporter"
5 require_relative "github_exporter/cli"
6 include GithubExporter
```

```

1 require "thor"
2 require "vim_printer"
3 require "html2pdf"
4 require "pdfs2pdf"
5 require_relative "github_exporter"
6 module GithubExporter
7   class CLI < Thor
8     desc "export", "Export a given URL or project to a single pdf file "
9     method_option "url",
10       aliases: "-u",
11       desc: "The full url of the github project to be cloned",
12       required: true
13     method_option "exts",
14       type: :array,
15       aliases: "-e",
16       desc: "The list of file extension to be exported",
17       required: true
18     method_option "non_exts",
19       type: :array,
20       aliases: "-f",
21       desc: "The list of file without extension to be exported",
22       default: []
23     method_option "theme",
24       type: :string,
25       aliases: "-t",
26       desc: "The theme to be used with vim_printer",
27       default: "default"
28     def export
29       exporter = GithubExporter::Exporter.new options[:url],
30                                               exts: options[:exts],
31                                               non_exts: options[:non_exts],
32                                               theme: options[:theme]
33       exporter.export
34     end
35
36     desc "usage", "Display help screen"
37     def usage
38       puts <<-EOS
39 Usage:
40
41 $github_exporter -e, --exts=EXT1 EXT2 EXT3 -u, --url=URL -theme=theme_name
42
43 Example:
44
45 # Export the *.rb from the given repository
46 $github_exporter -e rb -u https://github.com/agilecreativity/filename\_cleaner.git
47
48 # Export the *.rb and also 'Gemfile' from a given directory 'filename_cleaner'
49 # Note: must be one directory directly relative to current directory
50
51 $github_exporter -e rb -f Gemfile -u filename_cleaner
52
53 # Export the *.rb and also 'Gemfile' from a given directory 'filename_cleaner'
54 # using 'solarized' theme
55 # Note: must be one directory directly relative to current directory
56
57 $github_exporter -e rb -f Gemfile -u filename_cleaner -t solarized
58
59 Options:
60
61 -u, --url=URL # The full url of the github project to be cloned
62
63 -e, --exts=EXT1 EXT2 EXT3 .. # The list of extension names to be exported
64 # e.g. -e md rb java
65
66 -f, [--non-exts=one two three] # The list of file without extension to be exported
67 # e.g. -f Gemfile LICENSE
68
69 -t, [--theme=theme_name] # The theme to be used with vim_printer see :h
70 :colorscheme from Vim

```

```
71             # default: 'default'
72             # e.g. -t solarized
73
74 Export a given URL or project to a single pdf file
75
76     EOS
77   end
78
79   default_task :usage
80 end
81 end
```

```

1  #!/usr/bin/env ruby
2  require "uri"
3  require "agile_utils"
4  require_relative "../github_exporter"
5  module GithubExporter
6    class Exporter
7      attr_reader :url,
8                  :exts,
9                  :non_exts,
10                 :theme
11      attr_reader :base_dir,
12                  :repo_name,
13                  :output_path
14      # Constructor for Executor
15      #
16      # @param [String] url the input URL like
17      #           https://github.com/opal/opal.git or just the immediat folder name
18      # @param [Hash<Symbol,Object>] opts the option hash
19      #
20      # @option opts [Array<String>] :exts the list of file extension to be used
21      # @option opts [Array<String>] :non_exts the list of file without extension to be
22      # @option opts [String] :theme the theme to use for `vim_printer`
23      def initialize(url, opts = {})
24        @url = url
25        @base_dir = Dir.pwd
26        @exts = opts[:exts] || []
27        @non_exts = opts[:non_exts] || []
28        @theme = opts[:theme] || "default"
29        @repo_name = project_name(url)
30        @output_path = File.expand_path([base_dir, repo_name].join(File::SEPARATOR))
31      end
32
33      # Print and export the source from a given URL to a pdf
34      def export
35        clone
36        puts "FYI: list of extensions: #{all_extensions}"
37        puts "FYI: list of all files : #{all_files}"
38        files2htmls
39        htmls2pdfs
40        pdfs2pdf
41        cleanup
42      end
43
44      def to_s
45        <<-EOT
46          url      : #{url}
47          base_dir : #{base_dir}
48          exts     : #{exts}
49          non_exts : #{non_exts}
50          repo_name : #{repo_name}
51          theme    : #{theme}
52          output_path : #{output_path}
53        EOT
54      end
55
56      private
57
58      def clone
59        if File.exist?(output_path)
60          puts "The project #{output_path} already exist, no git clone needed!"
61          return
62        end
63        GithubExporter.clone_repository(url, repo_name, base_dir)
64      end
65
66      # List all extensions
67      def all_extensions
68        all_exts = GithubExporter.list_extensions(output_path)
69        # Strip off the '.' in the output if any.
70        all_exts.map! { |e| e.gsub(/^\.\/, "") }

```

```

71     all_exts
72 end
73
74 # List all files base on simple criteria
75 def all_files
76     files = []
77     if input_available?
78         files = GithubExporter.list_files base_dir: output_path,
79                                           exts: exts,
80                                           non_exts: non_exts,
81                                           recursive: true
82
83     end
84     files
85 end
86
87 # Convert files to htmls
88 def files2htmls
89     if input_available?
90         GithubExporter.files_to_htmls base_dir: output_path,
91                                       exts: exts,
92                                       non_exts: non_exts,
93                                       theme: theme
94     end
95 end
96
97 # Convert list of html to list of pdf files
98 def htmls2pdfs
99     input_file = File.expand_path("#{output_path}/vim_printer_#{repo_name}.tar.gz")
100    if File.exist?(input_file)
101        FileUtils.mkdir_p output_dir
102        # input_file = File.expand_path("#{output_path}/vim_printer_#{repo_name}.tar.gz")
103        AgileUtils::FileUtil.gunzip input_file, output_dir
104        GithubExporter.htmls_to_pdfs(base_dir: output_dir)
105    end
106 end
107
108 # Merge/join multiple pdf files into single pdf
109 def pdfs2pdf
110     input_file = File.expand_path("#{output_dir}/html2pdf_#{repo_name}.tar.gz")
111     if File.exist?(input_file)
112         AgileUtils::FileUtil.gunzip input_file, output_dir
113         GithubExporter.pdfs_to_pdf base_dir: output_dir,
114                                    recursive: true
115     end
116 end
117
118 # TODO: use random name instead of 'tmp' for intermediate directory name
119 def cleanup
120     generated_file = "#{output_dir}/pdfs2pdf_#{repo_name}.pdf"
121     if File.exist?(generated_file)
122         destination_file = File.expand_path(File.dirname(output_dir) + "../../#{
123 {repo_name}.pdf")
124         FileUtils.mv generated_file, destination_file
125         puts "Your final output is #{File.expand_path(destination_file)}"
126
127         # Now cleanup the generated files
128         FileUtils.rm_rf File.expand_path(File.dirname(output_dir) + "../../tmp")
129
130         # Also remove the 'vim_printer_#{repo_name}.tar.gz' if we have one
131         FileUtils.rm_rf File.expand_path(File.dirname(output_dir) + "../../#{
132 {repo_name}/vim_printer_#{repo_name}.tar.gz")
133     end
134 end
135
136 private
137
138 def output_dir
139     File.expand_path("#{base_dir}/tmp/#{repo_name}")
140 end

```

```
140 def input_available?  
141   (exts && !exts.empty?) || (non_exts && !non_exts.empty?)  
142 end  
143  
144 # Extract project name from a given URL  
145 #  
146 # @param [String] uri input uri  
147 #  
148 # example:  
149 #  
150 # project_name('https://github.com/erikhuda/thor.git') #=> 'thor'  
151 # project_name('https://github.com/erikhuda/thor')    #=> 'thor'  
152 def project_name(uri)  
153   if uri  
154     name = URI(uri).path.split(File::SEPARATOR).last  
155     # strip the '.' if any  
156     File.basename(name, ".*") if name  
157   end  
158 end  
159 end  
160 end
```

```

1 require "tmpdir"
2 require "git"
3 require "code_lister"
4 require "awesome_print"
5 module GithubExporter
6   CustomError = Class.new(StandardError)
7   class << self
8     # Clone the given repository from github
9     #
10    # @param [String] url the github repository url like 'https://github.com/schacon/ruby-
11    git.git'
12    # @param [String] name the output name to be used
13    # @param [String] path the output directory
14    def clone_repository(url, name, path)
15      puts "git clone #{url} #{File.expand_path(path)}/#{name}"
16      Git.clone url, name, path: File.expand_path(path)
17    end
18
19    def list_extensions(base_dir = ".")
20      extensions
21      = Dir.glob(File.join(File.expand_path(base_dir), "**/*")).reduce([]) do |exts, file|
22        exts << File.extname(file)
23      end
24      extensions.sort.uniq.delete_if { |e| e == "" }
25    end
26
27    def list_files(options = {})
28      CodeLister.files(options)
29    end
30
31    def files_to_htmls(opts)
32      base_dir = base_dir(opts[:base_dir])
33      exts      = opts[:exts] || []
34      non_exts  = opts[:non_exts] || []
35      args = [
36        "print",
37        "--base-dir",
38        base_dir,
39        "--exts",
40        exts,
41        "--theme",
42        opts.fetch(:theme, "default"),
43        "--recursive"
44      ]
45
46      # Add file without the extension if any
47      unless non_exts.empty?
48        args.concat(["--non-exts"]).concat(non_exts)
49      end
50      puts "Your input options for VimPrinter : #{args}"
51      VimPrinter::CLI.start(args)
52    end
53
54    # Export list of html files to pdfs using `html2pdf` gem
55    def htmls_to_pdfs(opts)
56      base_dir = base_dir(opts[:base_dir])
57      Html2Pdf::CLI.start [
58        "export",
59        "--base-dir",
60        base_dir,
61        "--recursive"
62      ]
63    end
64
65    # Merge/combine pdfs using `pdfs2pdf` gem
66    def pdfs_to_pdf(opts)
67      base_dir = base_dir(opts[:base_dir])
68      Pdfs2Pdf::CLI.start [
69        "merge",
70        "--base-dir",
71        base_dir,
72        "--recursive"

```



```
70     ]
71   end
72
73   private
74
75   # Always expand the directory name so that '~' or '.' is expanded correctly
76   def base_dir(dir_name)
77     File.expand_path(dir_name)
78   end
79 end
80 end
```

```
1 require "logger"
2 module GithubExporter
3   class << self
4     attr_writer :logger
5     # @return [Logger] the logger for the project
6     def logger
7       @logger ||= Logger.new STDOUT
8     end
9   end
10 end
```

```
1 module GithubExporter
2   VERSION = "0.1.0"
3 end
```

```
1 require_relative "../../test_helper"
2 describe GithubExporter do
3   context "#dummy_test" do
4     it "must pass the simple test" do
5       "string".wont_be_nil
6     end
7   end
8 end
```

```
1 require "minitest"
2 require "minitest/autorun"
3 require "minitest/pride"
4 require "minitest-spec-context"
5 require "pry"
6 require "awesome_print"
7 require_relative "../lib/github_exporter"
8 include GithubExporter
```