File List

```ruby
require_relative "github_exporter/version"
require_relative "github_exporter/logger"
require_relative "github_exporter/github_exporter"
require_relative "github_exporter/exporter"
require_relative "github_exporter/cli"
include GithubExporter
```

```ruby
require "thor"
require "vim_printer"
require "html2pdf"
require "pdfs2pdf"
require_relative "github_exporter"
module GithubExporter
  class CLI < Thor
    desc "export", "Export a given URL or project to a single pdf file"
    method_option "url",
                  aliases:  "-u",
                  desc:     "The full url of the github project to be cloned",
                  required: true
    method_option "exts",
                  type:     :array,
                  aliases:  "-e",
                  desc:     "The list of file extension to be exported",
                  required: true
    method_option "non_exts",
                  type:     :array,
                  aliases:  "-f",
                  desc:     "The list of file without extension to be exported",
                  default: []
    method_option "theme",
                  type:     :string,
                  aliases:  "-t",
                  desc:     "The theme to be used with vim_printer",
                  default:  "default"
    def export
      exporter = GithubExporter::Exporter.new options[:url],
                                              exts:     options[:exts],
                                              non_exts: options[:non_exts],
                                              theme:    options[:theme]
      exporter.export
    end

    desc "usage", "Display help screen"
    def usage
      puts <<-EOS
Usage:

  $github_exporter -e, --exts=EXT1 EXT2 EXT3 -u, --url=URL -theme=theme_name

Example:

  # Export the *.rb from the given repository

  $github_exporter -e rb -u https://github.com/agilecreativity/filename_cleaner.git

  # Export the *.rb and also 'Gemfile' from a given directory 'filename_cleaner'
  # Note: must be one directory directly relative to current directory

  $github_exporter -e rb -f Gemfile -u filename_cleaner

  # Export the *.rb and also 'Gemfile' from a given directory 'filename_cleaner'
  # using 'solarized' theme
  # Note: must be one directory directly relative to current directory

  $github_exporter -e rb -f Gemfile -u filename_cleaner -t solarized

Options:

  -u, --url=URL                   # The full url of the github project to be cloned

  -e, --exts=EXT1 EXT2 EXT3 ..    # The list of extension names to be exported
                                  # e.g. -e md rb java

  -f, [--non-exts=one two three]  # The list of file without extension to be exported
                                  # e.g. -f Gemfile LICENSE

  -t, [--theme=theme_name]        # The theme to be used with vim_printer see :h
:colorscheme from Vim
```

```
                                    # default: 'default'
                                    # e.g. -t solarized

Export a given URL or project to a single pdf file

      EOS
    end

    default_task :usage
  end
end
```

```ruby
#!/usr/bin/env ruby
require "uri"
require "agile_utils"
require_relative "../github_exporter"
module GithubExporter
  class Exporter
    attr_reader :url,
                :exts,
                :non_exts,
                :theme
    attr_reader :base_dir,
                :repo_name,
                :output_path
    # Constructor for Executor
    #
    # @param [String] url the input URL like
    #        https://github.com/opal/opal.git or just the immediat folder name
    # @param [Hash<Symbol,Object>] opts the option hash
    #
    # @option opts [Array<String>] :exts the list of file extension to be used
    # @option opts [Array<String>] :non_exts the list of file without extension to be
used
    # @option opts [String]        :theme the theme to use for `vim_printer`
    def initialize(url, opts = {})
      @url         = url
      @base_dir    = Dir.pwd
      @exts        = opts[:exts]     || []
      @non_exts    = opts[:non_exts] || []
      @theme       = opts[:theme]    || "default"
      @repo_name   = project_name(url)
      @output_path = File.expand_path([base_dir, repo_name].join(File::SEPARATOR))
    end

    # Print and export the source from a given URL to a pdf
    def export
      clone
      puts "FYI: list of extensions: #{all_extensions}"
      puts "FYI: list of all files : #{all_files}"
      files2htmls
      htmls2pdfs
      pdfs2pdf
      cleanup
    end

    def to_s
     <<-EOT
      url         : #{url}
      base_dir    : #{base_dir}
      exts        : #{exts}
      non_exts    : #{non_exts}
      repo_name   : #{repo_name}
      theme       : #{theme}
      output_path : #{output_path}
     EOT
    end

  private

    def clone
      if File.exist?(output_path)
        puts "The project #{output_path} already exist, no git clone needed!"
        return
      end
      GithubExporter.clone_repository(url, repo_name, base_dir)
    end

    # List all extensions
    def all_extensions
      all_exts = GithubExporter.list_extensions(output_path)
      # Strip off the '.' in the output if any.
      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) + "../../#
{repo_name}.pdf")
123         FileUtils.mv generated_file, destination_file
124         puts "Your final output is #{File.expand_path(destination_file)}"
125
126         # Now cleanup the generated files
127         FileUtils.rm_rf File.expand_path(File.dirname(output_dir) + "../../tmp")
128
129         # Also remove the 'vim_printer_#{repo_name}.tar.gz' if we have one
130         FileUtils.rm_rf File.expand_path(File.dirname(output_dir) + "../../#
{repo_name}/vim_printer_#{repo_name}.tar.gz")
131       end
132     end
133
134   private
135
136     def output_dir
137       File.expand_path("#{base_dir}/tmp/#{repo_name}")
138     end
139
```

```ruby
    def input_available?
      (exts && !exts.empty?) || (non_exts && !non_exts.empty?)
    end

    # Extract project name from a given URL
    #
    # @param [String] uri input uri
    #
    # example:
    #
    #  project_name('https://github.com/erikhuda/thor.git') #=> 'thor'
    #  project_name('https://github.com/erikhuda/thor')     #=> 'thor'
    def project_name(uri)
      if uri
        name = URI(uri).path.split(File::SEPARATOR).last
        # strip the '.' if any
        File.basename(name, ".*") if name
      end
    end
  end
end
```

```ruby
require "tmpdir"
require "git"
require "code_lister"
require "awesome_print"
module GithubExporter
  CustomError = Class.new(StandardError)
  class << self
    # Clone the given repository from github
    #
    # @param [String] url the github repository url like 'https://github.com/schacon/ruby-git.git'
    # @param [String] name the output name to be used
    # @param [String] path the output directory
    def clone_repository(url, name, path)
      puts "git clone #{url} #{File.expand_path(path)}/#{name}"
      Git.clone url, name, path: File.expand_path(path)
    end

    def list_extensions(base_dir = ".")
      extensions
= Dir.glob(File.join(File.expand_path(base_dir), "**/*")).reduce([]) do |exts, file|
        exts << File.extname(file)
      end
      extensions.sort.uniq.delete_if { |e| e == "" }
    end

    def list_files(options = {})
      CodeLister.files(options)
    end

    def files_to_htmls(opts)
      base_dir = base_dir(opts[:base_dir])
      exts     = opts[:exts]     || []
      non_exts = opts[:non_exts] || []
      args = [
        "print",
        "--base-dir",
        base_dir,
        "--exts",
        exts,
        "--theme",
        opts.fetch(:theme, "default"),
        "--recursive"
      ]

      # Add file without the extension if any
      unless non_exts.empty?
        args.concat(["--non-exts"]).concat(non_exts)
      end
      puts "Your input options for VimPrinter : #{args}"
      VimPrinter::CLI.start(args)
    end

    # Export list of html files to pdfs using `html2pdf` gem
    def htmls_to_pdfs(opts)
      base_dir = base_dir(opts[:base_dir])
      Html2Pdf::CLI.start [
        "export",
        "--base-dir",
        base_dir,
        "--recursive"]
    end

    # Merge/combine pdfs using `pdfs2pdf` gem
    def pdfs_to_pdf(opts)
      base_dir = base_dir(opts[:base_dir])
      Pdfs2Pdf::CLI.start [
        "merge",
        "--base-dir",
        base_dir,
        "--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
```

```ruby
require "logger"
module GithubExporter
  class << self
    attr_writer :logger
    # @return [Logger] the logger for the project
    def logger
      @logger ||= Logger.new STDOUT
    end
  end
end
```

```ruby
module GithubExporter
  VERSION = "0.1.0"
end
```

```ruby
require_relative "../../test_helper"
describe GithubExporter do
  context "#dummy_test" do
    it "must pass the simple test" do
      "string".wont_be_nil
    end
  end
end
```

```ruby
require "minitest"
require "minitest/autorun"
require "minitest/pride"
require "minitest-spec-context"
require "pry"
require "awesome_print"
require_relative "../lib/github_exporter"
include GithubExporter
```