File List

```ruby
require_relative "code2pdf/version"
require_relative "code2pdf/logger"
require_relative "code2pdf/code2pdf"
require_relative "code2pdf/exporter"
require_relative "code2pdf/cli"
include Code2Pdf
```

```ruby
require "thor"
require "vim_printer"
require "html2pdf"
require "pdfs2pdf"
require "agile_utils"
require_relative "code2pdf"
module Code2Pdf
  class CLI < Thor
    using AgileUtils::HashExt

    desc "export", "Export a given Git's project or a local project to a single pdf file "
    method_option "url",
                  aliases:  "-u",
                  desc:     "The full url of the git project to be cloned or local project
directory (mandatory)",
                  required: true
    method_option "exts",
                  type:     :array,
                  aliases:  "-e",
                  desc:     "The list of file extension to be exported (mandatory)",
                  required: true
    method_option "non_exts",
                  type:     :array,
                  aliases:  "-f",
                  desc:     "The list of file without extension to be exported (optional)",
                  default: []
    method_option "theme",
                  type:     :string,
                  aliases:  "-t",
                  desc:     "The theme to be used with vim_printer (optional)",
                  default:  "default"
    method_option "output_name",
                  type:     :string,
                  aliases:  "-o",
                  desc:     "The output filename (optional)"
    def export
      #opts = options.symbolize_keys(options)
      opts = options
      exporter = Code2Pdf::Exporter.new opts[:url],
                                        exts:        opts[:exts],
                                        non_exts:    opts[:non_exts],
                                        theme:       opts[:theme],
                                        output_name: opts[:output_name]

      exporter.export
    end

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

  $code2pdf -u, --url=URL -e, --exts=EXT1 EXT2 EXT3 -t, --theme=theme_name -o, --output-
name=output_file.pdf

Example:

  # Export the *.rb from the given repository

  $code2pdf -u https://github.com/agilecreativity/code2pdf.git -e rb

  # Export the *.rb and also 'Gemfile' from a 'code2pdf' directory
  # Note: this directory must be directly below the current directory

  $code2pdf -u code2pdf -e rb -f Gemfile

  # Same as previous command with the 'solarized' instead of 'default' colorscheme
  $code2pdf -u code2pdf -e rb -f Gemfile -t solarized

Options:

  -u, --url=URL                   # The full url of the git repository to be cloned OR local
directory name

  -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/colorscheme to be used with vim_printer see
```

```
:help :colorscheme from inside Vim
                                    # default: 'default'
                                    # e.g. -t solarized

  -o, [--output-name=output.pdf]  # The output pdf filename (will default to
'repository_name'.pdf)
                                    # e.g. -o repository_name.pdf

Export a given Git's repository or a local project directory to a single pdf file

      EOS
    end

    default_task :usage
  end
end
```

```ruby
require "tmpdir"
require "git"
require "code_lister"
module Code2Pdf
  CustomError = Class.new(StandardError)
  class << self
    # Clone the given repository from git repository
    #
    # @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"
      ]
      args.concat(["--non-exts"]).concat(non_exts) unless non_exts.empty?

      puts "FYI: 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"
      ]
    end

    private

    # Always expand the directory name so that '~' or '.' is expanded correctly
    def base_dir(dir_name)
      File.expand_path(dir_name)
    end
  end
end
```

```ruby
require "uri"
require "agile_utils"
require_relative "../code2pdf"
module Code2Pdf
  TMP_DIR = "code2pdf_tmp"
  # rubocop:disable ClassLength, MethodLength
  class Exporter
    attr_reader :url,
                :exts,
                :non_exts,
                :theme,
                :output_name

    attr_reader :base_dir,
                :repo_name,
                :output_path

    # The initializer for Exporter
    #
    # @param [String] url the input URL like
    #        https://github.com/opal/opal.git or just the immediate 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 colorscheme to use with `vim_printer`
    # @option opts [String]        :output_name the output filename if any
    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))
      @output_name = pdf_filename(opts[:output_name]) || "#{@repo_name}.pdf"
    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
      copy_output
      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}
        output_name : #{output_name}
       EOT
    end

    private

    def clone
      if File.exist?(output_path)
        puts "The project #{output_path} already exist, no git clone needed!"
        return
      end
      Code2Pdf.clone_repository(url, repo_name, base_dir)
    end

    # List all extensions
    def all_extensions
      all_exts = Code2Pdf.list_extensions(output_path)
      # Strip off the '.' in the output if any.
      all_exts.map! { |e| e.gsub(/^\./, "") }
      all_exts
    end
```

```ruby
    # List all files base on simple criteria
    def all_files
      files = []
      if input_available?
        files = Code2Pdf.list_files base_dir:  output_path,
                                    exts:      exts,
                                    non_exts:  non_exts,
                                    recursive: true
      end
      files
    end

    # Convert files to htmls
    def files2htmls
      Code2Pdf.files_to_htmls base_dir: output_path,
                              exts:     exts,
                              non_exts: non_exts,
                              theme:    theme if input_available?
    end

    # Convert list of html to list of pdf files
    def htmls2pdfs
      input_file = File.expand_path("#{output_path}/vim_printer_#{repo_name}.tar.gz")
      FileUtils.mkdir_p output_dir
      AgileUtils::FileUtil.gunzip input_file, output_dir if File.exist?(input_file)
      Code2Pdf.htmls_to_pdfs base_dir: output_dir
    end

    # Merge/join multiple pdf files into single pdf
    def pdfs2pdf
      input_file = File.expand_path("#{output_dir}/html2pdf_#{repo_name}.tar.gz")
      AgileUtils::FileUtil.gunzip input_file, output_dir if File.exist?(input_file)
      Code2Pdf.pdfs_to_pdf base_dir:  output_dir,
                           recursive: true
    end

    def copy_output
      generated_file = "#{output_dir}/pdfs2pdf_#{repo_name}.pdf"
      destination_file = File.expand_path(File.dirname(output_dir) + "../../#{output_name}")
      FileUtils.mv generated_file, destination_file if File.exist?(generated_file)
      puts "Your final output is #{File.expand_path(destination_file)}"
    end

    def cleanup
      FileUtils.rm_rf File.expand_path(File.dirname(output_dir) + "../../#
{Code2Pdf::TMP_DIR}")
      FileUtils.rm_rf File.expand_path(File.dirname(output_dir) + "../../#
{repo_name}/vim_printer_#{repo_name}.tar.gz")
    end

    def output_dir
      File.expand_path("#{base_dir}/#{Code2Pdf::TMP_DIR}/#{repo_name}")
    end

    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)
      name = URI(uri).path.split(File::SEPARATOR).last if uri
      File.basename(name, ".*") if name
    end

    # Add/rename the file extension to pdf
    #
    # @param [String] filename input file
    # @return [String,NilClass] output file with .pdf as extension or nil
    def pdf_filename(filename)
      return nil unless filename
      extname  = File.extname(filename)
      basename = File.basename(filename, ".*")
      if extname == ""
        "#{filename}.pdf"
```

```
      elsif extname != ".pdf"
        "#{basename}.pdf"
      else
        filename
      end
    end
  end
  # robocop:enable All
end
```

```ruby
require "logger"
module Code2Pdf
  class << self
    attr_writer :logger
    # @return [Logger] the logger for the project
    def logger
      @logger ||= Logger.new STDOUT
    end
  end
end
```

```ruby
module Code2Pdf
  VERSION = "0.2.0"
end
```

```ruby
require_relative "../../test_helper"
describe Code2Pdf 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/code2pdf"
include Code2Pdf
```