/Users/rambo/code/ruby/code_rippa/lib/code_rippa.rb

```ruby
require 'uv'
require 'find'
require 'linguist'
# require 'uv_overrides'


YAML::ENGINE.yamler= 'syck'

# Change to module later on
module CodeRippa

   def self.rip_file(path, theme, syntax, excluded_exts = [])
      outfile = File.open('out.tex', 'w')

      begin
         srcfile = File.read(path)
         src_ext = File.extname(path)[1..-1]

         if not excluded_exts.include? src_ext
            outfile.write preamble theme
            outfile.write "\\textcolor{headingcolor}{\\textbf{\\texttt{#{path.gsub('_','\_').gsub('%','\%')}}}}\\\\\n"
            outfile.write "\\textcolor{headingcolor}{\\rule{\\linewidth}{1.0mm}}\\\\\n"
            outfile.write Uv.parse(srcfile, 'latex', syntax, true, theme)
            outfile.write endtag
            outfile.close
         else
            puts "Warning: #{path} not processed. Check arguments.".background(:red).foreground(:yellow)
         end

      rescue Exception => e
         print e.backtrace.join('\n').background(:red).foreground(:white)
      end
   end


   def self.rip(dir_path, theme, syntax, excluded_exts = [])

      outfile = File.open('out.tex', 'w')
      counter = 0

      outfile.write preamble(theme)

      Find.find(dir_path) do |path|
         depth = path.to_s.count('/')
         if File.basename(path)[0] == ?.
            Find.prune
         else
```

```
48            begin
49               unless FileTest.directory?(path) or Linguist::FileBlob.new(path).binary?
50                  outfile.write "\\textcolor{white}{\\textbf{\\texttt{#{path.gsub('_','\_').gsub('%','\%')}}}}\\\\\n"
51                  outfile.write "\\textcolor{white}{\\rule{\\linewidth}{1.0mm}}\\\\\n"
52               end
53
54               f.write "\\pdfbookmark[#{depth-2}]{#{File.basename(path).gsub('_','\_').gsub('%','\%')}}{#{counter}}\n"
55
56               unless FileTest.directory? path or Linguist::FileBlob.new(path).binary?
57                  if Linguist::FileBlob.new(path).language.name == 'Ruby'
58                     # lang = Linguist::FileBlob.new(path).language.name.downcase
59                     outfile.write Uv.parse(File.read(path), 'latex', 'ruby', theme)
60                  end
61               end
62               outfile.write "\\clearpage\n"
63            rescue Exception => e
64               # ignore if something nasty happens
65            end
66            counter += 1
67         end
68      end
69      outfile.write endtag
70      outfile.close
71   end
72
73   private
74
75      def self.usage
76       "Usage: code_rippa [options] file_or_directory"
77      end
78
79      def self.syntax_path
80         Uv.syntax_path
81      end
82
83      def self.supported_syntax
84         syntax = []
85
86         Dir.foreach(syntax_path) do |f|
87            if File.extname(f) == ".syntax"
88               y = YAML.load(File.read "#{syntax_path}/#{f}")
89               syntax << File.basename(f, '.*')
90            end
91         end
92         syntax
93      end
94
95      def self.supported_langs
96         langs = []
```

```
97
98          Dir.foreach(syntax_path) do |f|
99             if File.extname(f) == ".syntax"
100               y = YAML.load(File.read "#{syntax_path}/#{f}")
101               langs << y["name"] if y["name"]
102            end
103         end
104         langs
105      end
106
107
108      def self.supported_exts
109         filetypes = []
110
111         Dir.foreach(syntax_path) do |f|
112            if File.extname(f) == ".syntax"
113               y = YAML.load(File.read "#{syntax_path}/#{f}")
114               filetypes += y["fileTypes"] if y["fileTypes"]
115            end
116         end
117         filetypes
118      end
119
120      def self.page_color(theme)
121         f = YAML.load(File.read("#{Uv.render_path}/latex/#{theme}.render"))
122         /([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})/.match(f['listing']['begin'].split('\\')[3]).to_s
123      end
124
125      def self.heading_color(theme)
126         f = YAML.load(File.read("#{Uv.render_path}/latex/#{theme}.render"))
127         /([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})/.match(f['listing']['begin'].split('\\')[2]).to_s
128      end
129
130      def self.preamble(theme)
131         preamble = ''
132         preamble << "\\documentclass[a4paper,landscape]{article}\n"
133         preamble << "\\pagestyle{empty}\n"
134         preamble << "\\usepackage{xcolor}\n"
135         preamble << "\\usepackage{colortbl}\n"
136         preamble << "\\usepackage{longtable}\n"
137         preamble << "\\usepackage[left=0cm,top=0.2cm,right=0cm,bottom=0.2cm,nohead,nofoot]{geometry}\n"
138         preamble << "\\usepackage[T1]{fontenc}\n"
139         preamble << "\\usepackage[scaled]{beramono}\n"
140         preamble << "\\usepackage[bookmarksopen,bookmarksdepth=20]{hyperref}\n"
141         preamble << "\\definecolor{pgcolor}{HTML}{#{page_color(theme)}}\n"
142         preamble << "\\definecolor{headingcolor}{HTML}{#{heading_color(theme)}}\n"
143         preamble << "\\pagecolor{pgcolor}\n"
144         preamble << "\\begin{document}\n"
145         preamble << "\\setlength\\LTleft\\parindent\n"
```

```
146           preamble « "\\setlength\\LTright\\fill\n"
147           preamble « "\\setlength{\\LTpre}{-10pt}\n"
148           preamble
149         end
150
151         def self.endtag
152           "\\end{document}\n"
153         end
154
155   end
156
157
158
159
160
161
162
163   # infile = File.new 'out.tex', 'w'
164
165   # CodeRippa.rip_file infile, options[:theme],
166
167   # CodeRippa.rip_file infile, ARGV[0]
168   # infile.close
169
```