README

April 28, 2016

Contents

1 TODO

- ☒ input as xls file
- ☐ gui?
  - ☐ shoes!

2 Shoes

```
require 'cj-parser'

cj_file = ARGV[0].to_s
days = ARGV[1]

Shoes.app do
  background "#efc"
```

```
  border("#be8",
         strokewidth: 6)

  stack(margin: 12) do
    para "Number Days"

    flow do
      @days = edit_line
      @button = button "Submit"
      @button.click do
        set_variables(@days.text.to_i)
      end
    end

    flow do
      @file = edit_line
      @button = button "Submit"
      @button.click do
        #Sheets.make_sheets(@file)
        #Sheets.make_sheets(cj_file)
      end
    end
  end
end
```

3 Main

./cj-parser.rb

```
require 'json'
require 'chronic'
require 'rubyXL'
require 'docx'
require 'roo'

require './lib/sheets.rb'
require './lib/label.rb'

# write to take in xml file
def set_variables(days)
```

```
  $days = days.to_f

  $cj_path = Dir.pwd()
  $pdf_path = "#{$cj_path}/pdfs"
  $templates_path = "#{$cj_path}/lib/templates"
  $template_top = File.open("#{$templates_path}/template-top.txt").readlines
  $template_bottom = File.open("#{$templates_path}/template-bottom.txt").readlines

  $sheets_dir = "#{$pdf_path}/sheets"
  $labels_dir = "#{$pdf_path}/labels"
  $sheet_top = File.open("#{$templates_path}/labelsTemplate-top.txt").readlines

end

def strip(s)
  s.gsub(/"/, '').
    gsub(/g/, '').
    gsub(/G/, '').
    gsub(/,/, '').
    split(' ')
end

def nil_convert(c)
  if c.nil?
    ""
  else
    c
  end
end

def get_labels(file)
  puts "getting labels"

  labels = []

  xls_file = Roo::Spreadsheet.open(file)

  xls_file.sheets.each do |sheet|
```

```
  sheet = xls_file.sheet(sheet)

  sheet.parse[4..-1].each do |row|

    zero,one,two,four,five,ten = nil_convert(row[0]),
    nil_convert(row[1]),
    nil_convert(row[2]),
    nil_convert(row[4]),
    nil_convert(row[5]),
    nil_convert(row[10])

    sizes = strip(five.to_s)
    gauge = "#{sizes[0]}g"
    size = "#{sizes[1]}\""
    desc = two.gsub("&", "and")
    id = one.to_s.split(/-/)[0]
    price = "$#{four.to_s.split(".")[0]}"
    supply = five
    updated = Chronic.parse(ten).to_f

    label = Label.new(gauge,
                      size,
                      desc,
                      id,
                      price,
                      supply,
                      updated
                     )

    seconds = 60*60*24*$days

    if (Time.now.to_f - updated.to_f) < seconds
      puts label.id
      labels.push label
    end

  end
end

# old csv code, keeping around for a rainy day
```

```
  # CSV.foreach(
  #   file,
  #   headers: false,
  #   skip_blanks: true,
  #   skip_lines: Regexp.union([ /^(?:,\s*)+$/, /^(?:Product)/ ]) ) do |row|

  #   size = row[5].to_s.gsub(/"/, '').gsub(/g/, '').gsub(/G/, '').gsub(/,/, '').split

  #   updated = Chronic.parse(row[10])

  #   label = Label.new("#{size[0]}g",
  #                         "#{size[1]}\"",
  #                         row[2].gsub("&", "and"),
  #                         row[1].to_s.split(/-/)[0],
  #                         row[4].to_s.split(".")[0],
  #                         row[5],
  #                         updated.to_f
  #                        )

  #   unless row[1] == "CASE JEWELRY-CJ"
  #     unless row[1] == "Product ID"
  #       if (Time.now.to_f - updated.to_f) < 60*60*24*$days
  #         puts label.id
  #         labels.push label
  #       end
  #     end
  #   end
  # end

  return labels

end

def rows_to_json(file)

  puts "converting rows to javascript object notation"

  json_file = "cj_db.json"
  count = get_labels(file).size

  File.open(json_file, "w") do |file|
```

```
    file.puts '{ "products": ['
  end

  get_labels(file).each_with_index do |row, index|
    File.open(json_file, "a") do |json|
      json.puts row.to_json

      unless index == count - 1
        json.puts ","
      end
    end
  end

  File.open(json_file, "a") do |file|
    file.puts '] }'
  end
end

def labels_to_tex(file)

  get_labels(file).each do |row|

    puts row.id

    tex_file = "#{row.id}.tex"
    pdf_file = "#{row.id}.pdf"

    if row.size == "\""
      size = row.gauge
    elsif row.gauge == ""
      size = row.size
    else
      size = "#{row.gauge} #{row.size}"
    end

    type = row.desc
    id = row.id
    price = row.price

    File.open(tex_file, "w") do |file|
```

```
      pre_script = "{\\scriptsize\\textit{"
      pre_lg = "{\\large"
      pre_LG = "{\\Large"
      post = "}}\n\n"

      file.puts $template_top

      file.puts "\\begin{center}" +
                "#{pre_lg}{" +
                "#{type}#{post}" +
                "\\end{center}"

      file.puts "\\begin{center}" +
                "#{pre_LG}" + "\\textit{" +
                "#{size}#{post}" +
                "\\end{center}"

      file.puts "\\begin{center}" +
                "#{pre_lg}{" +
                "#{id}\\hspace{25mm}  \\#{price}#{post}" +
                "\\end{center}"

      file.puts $template_bottom
    end

    `pdflatex #{tex_file} && mv *.tex *.aux *.log *.out tmp && mv *.pdf #{$pdf_path}`
  end
end
```

4 Classes

./lib/label.rb

```
class Label
  #include Sheets

  def initialize(gauge, size, desc, id, price, supply, updated)
    @gauge = gauge
    @size = size
    @desc = desc
```

```
    @id = id
    @price = price
    @supply = supply
    @updated = updated
  end

  attr_reader :gauge, :size, :desc, :id, :price, :supply, :updated

end
```

5 Modules

5.1 Sheets

```
./lib/sheets.rb

module Sheets

  def Sheets.get_sheet_rows()
    Dir.chdir($pdf_path)

    files = Dir.entries(".").reject { |entry| File.directory?(entry) }
    pdfs = files.select { |file| file.end_with? '.pdf' }
    label_count = pdfs.count

    fboxs = []

    pdfs.each do |pdf|
      fboxs.push "\\framebox[1.0\\width]{\\includegraphics{#{$labels_dir}/#{pdf}}}"
    end

    rows = fboxs.each_slice(4).to_a
    return rows
  end

  def Sheets.get_sheets()

    pages = []

    get_sheet_rows.each do |row|
```

```
    pages.push row
  end

  sheets = pages.each_slice(8).to_a

  return sheets
end

def Sheets.make_sheets(file)


  rows_to_json(file)
  labels_to_tex(file)

  sheet_count = get_sheets.count

  if sheet_count >= 1

    puts "creating sheets"

    sheets = get_sheets

    i = 0

    puts "entering sheets directory"
    Dir.chdir($sheets_dir)
    `mv *.pdf bak`

    sheets.each do |page|

      name = "sheet_000#{i}"
      filename = "#{name}.tex"

      puts "making #{name} sheet"
      File.open(filename, "w") do |file|
        file.puts $sheet_top
        file.puts "\\begin{center}"
        file.puts "\\setlength{\\fboxsep}{1pt}"
        file.puts "\\setlength{\\fboxrule}{0.1pt}"
      end
```

```
        page.each do |row|
          File.open(filename, "a") do |file|

            file.puts row
            file.puts "\\newline"

            row.each do |box|
              pdf = box.split("{").last.split("}").first.split("/").last
              `mv ../#{pdf} #{$labels_dir}`
            end
          end
        end

        File.open(filename, "a") do |file|
          file.puts "\\end{center}"
          file.puts "\\end{document}"
        end

        i += 1

        #`pdflatex #{filename} && evince #{name}.pdf && mv *.aux *.log *.out *.tex texi
        `pdflatex #{filename} && mv *.aux *.log *.out *.tex texfiles`

      end

    end

    Dir.chdir($cj_path)

  end


end
```