README

May 2, 2016

Contents

Traveling Ruby Win32 package for CJ-parser

1 Move Core

`./app.rb`

```
require 'cj-parser'

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

```
Shoes.app(title: "Case Jewelry Label Maker", width: 800, height: 1000, resizable: true
  #background "#c3f4f8".."#fff"
  #background "#000"
  background "img/sheet.png"

  @days_display = stack(margin: 12) do
    @days_para = para strong("0")

    @days_para
  end

  stack(margin: 12) do
    para "Number of Days"
    flow {
      #@days = edit_line :width => 50

      button "Choose File" do
        @file = ask_open_file
      end

      @days_button = button "Days" do
        @days = ask("How many days back?")
        @days_display.clear do
          @days_para = para strong(@days)

          @days_para
        end
      end

      button "Submit" do
        #alert "Making sheets from #{@file} from last #{@days.text} days"
        #alert "Making sheets from #{@file} from last #{@days} days"
        #set_variables(@days.text)

        if confirm("Make sheets from #{@file} from last #{@days} days?")
          set_variables(@days)
          Sheets.make_sheets(@file)
        end
      end
      button "X" do
        exit()
      end
    }

    # @sheet = image(
    #   #"img/label.png",
    #   "img/sheet.png",
    #   width: 850,
    #   height: 1100
    # )

  end
}
```

1.1 Main

./cj-parser.rb

```
#!/usr/bin/env ruby

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

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

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
        $labelID = 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
```

1.2 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
```

1.3 Modules

1.3.1 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 tex
    ‘pdflatex #{filename} && mv *.aux *.log *.out *.tex texfiles‘

   end

  end

  Dir.chdir($cj_path)

 end


end
```

2 REQUIREMENTS

- ☐ deprecate old files
  - ☐ remove tangle from src block headers

3 Preparation

```
rbenv local 2.1.9
```

```
mkdir packaging
```

```
 ./Gemfile
```

```
source 'https://rubygems.org'

gem 'faker'
gem 'json'
gem 'chronic'
gem 'roo', '~> 2.3.2'

group :development do
  gem 'rake'
end
```

./packaging/wrapper.sh

```
#!/bin/bash
set -e

# Figure out where this script is located
SELFDIR="`dirname \"$0\"`"
SELFDIR="`cd \"$SELFDIR\" && pwd`"

# Tell Bundler where the Gemfile and gems are.
export BUNDLE_GEMFILE="$SELFDIR/lib/vendor/Gemfile"
unset BUNDLE_IGNORE_CONFIG

# Run the actual app using the bundled Ruby interpreter, with Bundler activated.
exec "$SELFDIR/lib/ruby/bin/ruby" -r bundler/setup "$SELFDIR/lib/app/cj-parser.rb"
```

```
chmod +x packaging/wrapper.sh
```

./packaging/bundler-config

```
BUNDLE_PATH: .
BUNDLE_WITHOUT: development
BUNDLE_DISABLE_SHARED_GEMS: '1'
```

```
bundle install
```

4 Creating a batch file

./packaging/wrapper.bat

```
@echo off

:: Tell Bundler where the Gemfile and gems are.
set "BUNDLE_GEMFILE=%~dp0\lib\vendor\Gemfile"
set BUNDLE_IGNORE_CONFIG=

:: Run the actual app using the bundled Ruby Interpreter, with Bundler activated.
@"%~dp0\lib\ruby\bin\ruby.bat" -rbundler/setup "%~dp0\lib\app\cj-parser.rb"
```

5 Modifying the Rakefile

./Rakefile

```
# For Bundler.with_clean_env
require 'bundler/setup'

PACKAGE_NAME = "cj_parser"
VERSION = "1.0.0"
TRAVELING_RUBY_VERSION = "20150210-2.1.5"

desc "Package your app"
task :package => ['package:linux:x86', 'package:linux:x86_64', 'package:osx', 'package

namespace :package do
  namespace :linux do
    desc "Package your app for Linux x86"
    task :x86 => [:bundle_install,
                  "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-linux-x86.tar.gz
    ] do
      create_package("linux-x86")
    end

    desc "Package your app for Linux x86_64"
    task :x86_64 => [:bundle_install,
      "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-linux-x86_64.tar.gz"
    ] do
      create_package("linux-x86_64")
    end
  end
  desc "Package your app for OS X"
  task :osx => [:bundle_install,
                "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-osx.tar.gz"
               ] do
    create_package("osx")
  end

  desc "Package your app for Windows x86"
  task :win32 => [:bundle_install,
                  "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-win32.tar.gz"#
                 ] do
    create_package("win32", :windows)
  end

  desc "Install gems to local directory"
  task :bundle_install do
    if RUBY_VERSION !~ /^2\.1\./
      abort "You can only 'bundle install' using Ruby 2.1, because that's what Travelin
    end
    sh "rm -rf packaging/tmp"
    sh "mkdir packaging/tmp"
    sh "cp Gemfile Gemfile.lock packaging/tmp/"
    Bundler.with_clean_env do
      sh "cd packaging/tmp && env BUNDLE_IGNORE_CONFIG=1 bundle install --path ../vendo
    end
    sh "rm -rf packaging/tmp"
    sh "rm -f packaging/vendor/*/*/cache/*"
    sh "rm -rf packaging/vendor/ruby/*/extensions"
    sh "find packaging/vendor/ruby/*/gems -name '*.so' | xargs rm -f"
    sh "find packaging/vendor/ruby/*/gems -name '*.bundle' | xargs rm -f"
    sh "find packaging/vendor/ruby/*/gems -name '*.o' | xargs rm -f"
  end
end

file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-linux-x86.tar.gz" do
  download_runtime("linux-x86")
end

file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-linux-x86_64.tar.gz" do
  download_runtime("linux-x86_64")
```

```
end

file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-osx.tar.gz" do
  download_runtime("osx")
end

# file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-win32-sqlite3-#{SQLITE3_VERS
#   download_runtime("win32", "sqlite3-#{SQLITE3_VERSION}")
# end

file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-win32.tar.gz" do
  download_runtime("win32")
end

file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-linux-x86.tar.gz" do
  download_native_extension("linux-x86")
end

file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-linux-x86_64.tar.gz" do
  download_native_extension("linux-x86_64")
end

file "packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-osx-sqlite3.tar.gz" do
  download_native_extension("osx")
end

def create_package(target, os_type = :unix)
  package_dir = "#{PACKAGE_NAME}-#{VERSION}-#{target}"
  sh "rm -rf #{package_dir}"
  sh "mkdir #{package_dir}"
  sh "mkdir -p #{package_dir}/lib/app"
  sh "cp cj-parser.rb #{package_dir}/lib/app/"
  sh "mkdir #{package_dir}/lib/ruby"
  sh "tar -xzf packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-#{target}.tar.gz -C #

  if os_type == :unix
    sh "cp packaging/wrapper.sh #{package_dir}/cj-parser"
  else
    sh "cp packaging/wrapper.bat #{package_dir}/cj-parser.bat"
  end
  sh "cp -pR packaging/vendor #{package_dir}/lib/"
  sh "cp Gemfile Gemfile.lock #{package_dir}/lib/vendor/"
  sh "mkdir #{package_dir}/lib/vendor/.bundle"
  sh "cp packaging/bundler-config #{package_dir}/lib/vendor/.bundle/config"
  if os_type == :unix
    sh "tar -xzf packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-#{target}.tar.gz "
  else
    sh "tar -xzf packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-#{target}.tar.gz "
  end
  if !ENV['DIR_ONLY']
    if os_type == :unix
      sh "tar -czf #{package_dir}.tar.gz #{package_dir}"
    else
      sh "zip -9r #{package_dir}.zip #{package_dir}"
    end
    sh "rm -rf #{package_dir}"
  end
end

def download_runtime(target)
  sh "cd packaging && curl -L -O --fail " +
    "https://d6r77u77i8pq3.cloudfront.net/releases/traveling-ruby-#{TRAVELING_RUBY_VERS
end

def download_native_extension(target)
  sh "curl -L --fail -o packaging/traveling-ruby-#{TRAVELING_RUBY_VERSION}-#{target}.ta
     "https://d6r77u77i8pq3.cloudfront.net/releases/traveling-ruby-gems-#{TRAVELING_RUB
end
```

6 Creating and testing the package

```
rake package:win32
```

7 NB

Tutorial 4: creating packages for Windows
https://github.com/phusion/traveling-ruby/blob/master/TUTORIAL-4.md

important Windows-specific caveats