pdfkit / pdfkit

Watch 34 Star 2,343 Fork 334

Code Issues 24 Pull requests 16 Projects 0 Wiki Pulse Graphs

A Ruby gem to transform HTML + CSS into PDFs using the command-line utility wkhtmltopdf

pdfkit ruby wkhtmltopdf html-to-pdf middleware

404 commits 6 branches 26 releases 59 contributors MIT

Branch: master New pull request Find file Clone or download

sigmavirus24 committed on GitHub Merge pull request #375 from guilhermesimoes/master … Latest commit 36da50b Aug 11, 2016

lib	Use `Array()` instead of `[].flatten` in `middleware#conditions_as_re…	Mar 14, 2016
spec	Preprocess HTML in PDFKit, not PDFKit::Middleware.	Jan 8, 2016
.document	Initial commit to PDFKit.	May 21, 2010
.gitignore	Ensure Gemfile.lock is not accidentally added back	Aug 10, 2014
.rspec	Actually run specs under RSpec 2	Jun 18, 2010
.ruby-gemset	Add .ruby-version, .ruby-gemset	Oct 5, 2013
.ruby-version	Add .ruby-version, .ruby-gemset	Oct 5, 2013
.travis.yml	Address Travis issues with Bundler failures	Jan 17, 2016
CHANGELOG.md	Bump to 0.8.2	Aug 26, 2015
Gemfile	Add SimpleCov & Pry as development dependencies	May 8, 2015
LICENSE	Initial commit to PDFKit.	May 21, 2010
POST_INSTALL	fixed a link to wiki in POST_INSTALL message	Mar 13, 2012
README.md	Explain :root_url and :protocol options in README	Jan 8, 2016
Rakefile	Use rdoc/task	Apr 29, 2013
pdfkit.gemspec	Add gemspec requirements attribute	Aug 11, 2016

README.md

PDFKit

Create PDFs using plain old HTML+CSS. Useswkhtmltopdf on the back-end which renders HTML using Webkit.

Install

PDFKit

```
gem install pdfkit
```

wkhtmltopdf

1. Install by hand (recommended):

   https://github.com/pdfkit/pdfkit/wiki/Installing-WKHTMLTOPDF

2. Try using the `wkhtmltopdf-binary` gem (mac + linux i386)

```
gem install wkhtmltopdf-binary
```

Note: The automated installer has been removed.

Usage

```
# PDFKit.new takes the HTML and any options for wkhtmltopdf
# run `wkhtmltopdf --extended-help` for a full list of options
kit = PDFKit.new(html, :page_size => 'Letter')
kit.stylesheets << '/path/to/css/file'

# Get an inline PDF
pdf = kit.to_pdf

# Save the PDF to a file
file = kit.to_file('/path/to/save/pdf')

# PDFKit.new can optionally accept a URL or a File.
# Stylesheets can not be added when source is provided as a URL of File.
kit = PDFKit.new('http://google.com')
kit = PDFKit.new(File.new('/path/to/html'))

# Add any kind of option through meta tags
PDFKit.new('<html><head><meta name="pdfkit-page_size" content="Letter"')
PDFKit.new('<html><head><meta name="pdfkit-cookie cookie_name1" content="cookie_value1"')
PDFKit.new('<html><head><meta name="pdfkit-cookie cookie_name2" content="cookie_value2"')
```

Resolving relative URLs and protocols

If the source HTML has relative URLs (`/images/cat.png`) or protocols (`//example.com/site.css`) that need to be resolved, you can pass `:root_url` and `:protocol` options to PDFKit:

```
PDFKit.new(html, root_url: 'http://mysite.com/').to_file
# or:
PDFKit.new(html, protocol: 'https').to_file
```

Using cookies in scraping

If you want to pass a cookie to cookie to pdfkit to scrape a website, you can pass it in a hash:

```
kit = PDFKit.new(url, cookie: {cookie_name: :cookie_value})
kit = PDFKit.new(url, [:cookie, :cookie_name1] => :cookie_val1, [:cookie, :cookie_name2] => :cookie_val2
```

Configuration

If you're on Windows or you would like to use a specific wkhtmltopdf you installed, you will need to tell PDFKit where the binary is. PDFKit will try to intelligently guess at the location of wkhtmltopdf by running the command `which wkhtmltopdf`. If you are on Windows, want to point PDFKit to a different binary, or are having trouble with getting PDFKit to find your binary, please manually configure the wkhtmltopdf location. You can configure PDFKit like so:

```ruby
# config/initializers/pdfkit.rb
PDFKit.configure do |config|
  config.wkhtmltopdf = '/path/to/wkhtmltopdf'
  config.default_options = {
    :page_size => 'Legal',
    :print_media_type => true
  }
  # Use only if your external hostname is unavailable on the server.
  config.root_url = "http://localhost"
  config.protocol = 'http'
  config.verbose = false
end
```

Middleware

PDFKit comes with a middleware that allows users to get a PDF view of any page on your site by appending .pdf to the URL.

Middleware Setup

Non-Rails Rack apps

```ruby
# in config.ru
require 'pdfkit'
use PDFKit::Middleware
```

Rails apps

```ruby
# in application.rb(Rails3) or environment.rb(Rails2)
require 'pdfkit'
config.middleware.use PDFKit::Middleware
```

With PDFKit options

```ruby
# options will be passed to PDFKit.new
config.middleware.use PDFKit::Middleware, :print_media_type => true
```

With conditions to limit routes that can be generated in pdf

```ruby
# conditions can be regexps (either one or an array)
config.middleware.use PDFKit::Middleware, {}, :only => %r[^/public]
config.middleware.use PDFKit::Middleware, {}, :only => [%r[^/invoice], %r[^/public]]

# conditions can be strings (either one or an array)
config.middleware.use PDFKit::Middleware, {}, :only => '/public'
config.middleware.use PDFKit::Middleware, {}, :only => ['/invoice', '/public']

# conditions can be regexps (either one or an array)
config.middleware.use PDFKit::Middleware, {}, :except => [%r[^/prawn], %r[^/secret]]

# conditions can be strings (either one or an array)
config.middleware.use PDFKit::Middleware, {}, :except => ['/secret']
```

Saving the generated .pdf to disk

Setting the `PDFKit-save-pdf` header will cause PDFKit to write the generated .pdf to the file indicated by the value of the header.

For example:

```
headers['PDFKit-save-pdf'] = 'path/to/saved.pdf'
```

Will cause the .pdf to be saved to `path/to/saved.pdf` in addition to being sent back to the client. If the path is not writable/non-existant the write will fail silently. The `PDFKit-save-pdf` header is never sent back to the client.

Troubleshooting

- Single thread issue: In development environments it is common to run a single server process. This can cause issues when rendering your pdf requires wkhtmltopdf to hit your server again (for images, js, css). This is because the resource requests will get blocked by the initial request and the initial request will be waiting on the resource requests causing a deadlock.

  This is usually not an issue in a production environment. To get around this issue you may want to run a server with multiple workers like Passenger or try to embed your resources within your HTML to avoid extra HTTP requests.

  Example solution (rails / bundler), add unicorn to the development group in your Gemfile `gem 'unicorn'` then run `bundle`. Next, add a file `config/unicorn.conf` with

  ```
  worker_processes 3
  ```

  Then to run the app `unicorn_rails -c config/unicorn.conf` (from rails_root)

- Resources aren't included in the PDF:Images, CSS, or JavaScript does not seem to be downloading correctly in the PDF. This is due to the fact that wkhtmltopdf does not know where to find those files. Make sure you are using absolute paths (start with forward slash) to your resources. If you are using PDFKit to generate PDFs from a raw HTML source make sure you use complete paths (either file paths or urls including the domain). In restrictive server environments the root_url configuration may be what you are looking for change your asset host.

- Mangled output in the browser:Be sure that your HTTP response headers specify "Content-Type: application/pdf"

Note on Patches/Pull Requests

- Fork the project.
- Setup your development environment with: `gem install bundler`; `bundle install`
- Make your feature addition or bug fix.
- Add tests for it. This is important so I don't break it in a future version unintentionally.
- Commit, do not mess with rakefile, version, or history. (if you want to have your own version, that is fine but bump version in a commit by itself I can ignore when I pull)
- Send me a pull request. Bonus points for topic branches.

Copyright

Copyright (c) 2010 Jared Pace. See LICENSE for details.