SebastiaanKlippert / **go-wkhtmltopdf**

Watch 4 Star 12 Fork 2

Code Issues 1 Pull requests 0 Projects 0 Pulse Graphs

Golang commandline wrapper for wkhtmltopdf

44 commits **2** branches **0** releases Fetching contributors MIT

Branch: **master** New pull request Find file Clone or download

SebastiaanKlippert update readme for setpath Latest commit ec43762 Sep 1, 2016

testfiles	add a simpeler test case	Jul 26, 2016
.gitignore	update readme for setpath	Sep 1, 2016
.travis.yml	allow travis osx fails	Sep 1, 2016
LICENSE	Initial commit	Nov 21, 2015
README.md	update readme for setpath	Sep 1, 2016
options.go	check if uint and float options are set using a bool variable, closes #3	Jul 27, 2016
simplesample_test.go	enable tests and linux builds again	Jul 26, 2016
wkhtmltopdf.go	fix a bug.	Sep 1, 2016
wkhtmltopdf_test.go	check if uint and float options are set using a bool variable, closes #3	Jul 27, 2016

README.md

godoc reference build passing go report A+

go-wkhtmltopdf

Golang commandline wrapper for wkhtmltopdf

~~Work in progress, used internally only at this point.~~
~~No guarantees and **everything may change**.~~
Update 17-07-2016:
This package is now used in our production environment after a long test period so changes are unlikely.
Bugs wil be fixed as soon as they are found, but don't expect major changes from now on.

See http://wkhtmltopdf.org/index.html for wkhtmltopdf docs.

What and why

We needed a way to generate PDF documents from Go. These vary from invoices with highly customizable lay-outs to reports with tables, graphs and images. In our opinion the best way to do this was by using HTML/CSS templates as source for our PDFs. Using CSS print media types and millimeters instead of pixel units we can generate very acurate PDF documents using wkhtmltopdf.

go-wkhtmltopdf is a pure Golang wrapper around the wkhtmltopdf command line utility.

It has all options typed out as struct members which makes it very easy to use if you use an IDE with code completion and it has type safety for all options. For example you can set general options like

```
pdfg.Dpi.Set(600)
pdfg.NoCollate.Set(false)
pdfg.PageSize.Set(PageSizeA4)
pdfg.MarginBottom.Set(40)
```

The same goes for adding pages, settings page options, TOC options per page etc.

It takes care of setting the correct order of options as these can become very long with muliple pages where you have page and TOC options for each page.

Secondly it makes usage in server-type applications easier, every instance (PDF process) has its own output buffer which contains the PDF output and you can feed one input document from an io.Reader (using stdin in wkhtmltopdf). You can combine any number or external HTML documents (HTTP(S) links) with at most one HTML document from stdin and set options for each input document.

Note: You can also ignore the internal buffer and let wkhtmltopdf write directly to disk if required for large files.

For us this is one of the easiest ways to genere PDF documents from Go(lang) and performance is very acceptable.

Installation

go get or use a Go dependency manager of your liking.

```
go get -u github.com/SebastiaanKlippert/go-wkhtmltopdf
```

go-wkhtmltopdf finds the path to wkhtmltopdf by

- first looking in the current dir
- looking in the PATH and PATHEXT environment dirs
- using the WKHTMLTOPDF_PATH environment dir

If you need to set your own wkhtmltopdf path or want to change it during execution, you can call SetPath().

Usage

See testfile `wkhtmltopdf_test.go` for more complex options, the most simple test is in `simplesample_test.go`

```go
package wkhtmltopdf

import (
	"fmt"
	"log"
)

func ExampleNewPDFGenerator() {

	// Create new PDF generator
	pdfg, err := NewPDFGenerator()
	if err != nil {
		log.Fatal(err)
	}

	// Add one page from an URL
	pdfg.AddPage(NewPage("https://github.com/SebastiaanKlippert/go-wkhtmltopdf"))

	// Create PDF document in internal buffer
	err = pdfg.Create()
	if err != nil {
		log.Fatal(err)
	}

	// Write buffer contents to file on disk
	err = pdfg.WriteFile("./simplesample.pdf")
	if err != nil {
		log.Fatal(err)
	}

	fmt.Println("Done")
	// Output: Done
}
```

Speed

The speed if pretty much determined by wkhtmltopdf itself, or if you use extrnal source URLs, the time it takes to get the source HTML.

The go wrapper time is negligible with around 0.04ms for parsing an above average number of commandline options.

Benchmarks are included.