

# WkHtmlToPdf for Crystal

build passing

Crystal wrapper for libwkhtmltox C library.

*wkhtmltopdf* and *wkhtmltoimage* permit to render HTML into PDF and various image formats using the Qt WebKit rendering engine - see [wkhtmltopdf.org](http://wkhtmltopdf.org)

## Requirements

- *libwkhtmltox* must be installed
- *pkg-config* must be available

## Installation

- Add this to your application's `shard.yml`:

```
dependencies:
  wkhtmltopdf-crystal:
    github: blocknotes/wkhtmltopdf-crystal
```

- If *wkhtmltox* library is installed but missing for Crystal compiler: copy *wkhtmltox.pc* (from `lib/wkhtmltopdf-crystal` folder) in a `pkg-config` folder (ex. `/usr/local/lib/pkgconfig`) or set the environment variable `PKG_CONFIG_PATH` with the path to *wkhtmltox.pc* before compiling
- Optionally edit *wkhtmltox.pc* with the correct path to *wkhtmltox* (default headers path: `/usr/local/include/wkhtmltox`)

## Usage

HTML to PDF:

```
require "wkhtmltopdf"
Wkhtmltopdf::WkPdf.new( "test.pdf" ).convert( "<h3>Just a test</h3>" )
```

Fetch URL content and convert it to JPG:

```
require "wkhtmltopdf"
img = Wkhtmltopdf::WkImage.new
img.set_url "http://www.google.com"
img.set_output "test.jpg"
img.set "quality", "90"
img.convert
```

Write to buffer (only if no output is specified):

```
require "wkhtmltopdf"
pdf = Wkhtmltopdf::WkPdf.new
pdf.convert "<h3>Just a test</h3>"
pdf.object_setting "footer.right", "[page] / [topage]" # Set page counter on footer
unless pdf.buffer.nil?
  puts "PDF buffer size: " + pdf.buffer.try( &.size ).to_s
end
```

Lib settings (available with `set` / `object_setting` methods on wrappers): [libwkhtmltox pagesettings](#)

## More examples

See [examples](#) folder. Includes a Kemal example to print an ECR view in PDF.

## Contributors

- [Mattia Roccoberton](#) - creator, maintainer

# module Wkhtmltopdf

## Defined in:

[wkhtmltopdf/version.cr](#)

[wkhtmltopdf/wk\\_image.cr](#)

[wkhtmltopdf/wk\\_pdf.cr](#)

[wkhtmltopdf-crystal.cr](#)

## Constant Summary

**VERSION** = "0.1.6"

# class Wkhtmltopdf::WkImage

- [Wkhtmltopdf::WkImage](#)
- [Reference](#)
- [Object](#)

## Defined in:

[wkhtmltopdf/wk\\_image.cr](#)

## Constant Summary

`FORMATS = [ "jpg", "png", "bmp", "svg" ]`

Output formats available

## Class Method Summary

- [.new\(path = ""\)](#)

Init default values

## Instance Method Summary

- [#buffer : Slice\(UInt8\)?](#)

Buffer used for in-memory generation (available if no output is specified)

- [#convert\(html = nil\)](#)

Convert to image

- [#set\(key : String, value : String\)](#)

Set an option

- [#set\\_output\(path : String\)](#)

Set output path

- [#set\\_url\(url : String\)](#)

Set URL to fetch content from

## Class Method Detail

```
def self.new(path = "") #
```

Init default values

- `path`: string with an output file path (extension included)

[\[View source\]](#)

## Instance Method Detail

```
def buffer : Slice(UInt8)? #
```

Buffer used for in-memory generation (available if no output is specified)

[\[View source\]](#)

```
def convert(html = nil) #
```

Convert to image

- `html`: HTML string used as content, if omitted (or nil) a URL to fetch is required (using [#set\\_url](#))

[\[View source\]](#)

```
def set(key : String, value : String) #
```

Set an option

set an option

- `key`: string with key name
- `value`: string with setting value

NOTE for available settings see [pagePdfObject](#)

[\[View source\]](#)

```
def set_output(path : String) #
```

Set output path

- `path`: string with an output file path (extension included)

[\[View source\]](#)

```
def set_url(url : String) #
```

Set URL to fetch content from

- `url`: string with a complete URL (schema included)

[\[View source\]](#)

# class Wkhtmltopdf::WkPdf

- [Wkhtmltopdf::WkPdf](#)
- [Reference](#)
- [Object](#)

## Defined in:

[wkhtmltopdf/wk\\_pdf.cr](#)

## Class Method Summary

- [.new\(path = ""\)](#)  
Init default values

## Instance Method Summary

- [#buffer : Slice\(UInt8\)?](#)  
Buffer used for in-memory generation (available if no output is specified)
- [#convert\(html = nil\)](#)  
Convert to PDF
- [#object\\_setting\(key : String, value : String\)](#)  
Pdf object settings
- [#set\(key : String, value : String\)](#)  
Pdf global settings
- [#set\\_output\(path : String\)](#)  
Set output path
- [#set\\_url\(url : String\)](#)  
Set URL to fetch content from

## Class Method Detail

```
def self.new(path = "") #
```

Init default values

- `path`: string with an output file path (extension included)

[\[View source\]](#)

## Instance Method Detail

```
def buffer : Slice(UInt8)? #
```

Buffer used for in-memory generation (available if no output is specified)

[\[View source\]](#)

```
def convert(html = nil) #
```

Convert to PDF

- `html`: HTML string used as content, if omitted (or nil) a URL to fetch is required (using [#set\\_url](#))

[\[View source\]](#)

```
def object_setting(key : String, value : String) #
```

Pdf object settings

- `key`: string with key name

- `value`: string with setting value

NOTE for available settings see [pagePdfObject](#)

[\[View source\]](#)

```
def set(key : String, value : String) #
```

Pdf global settings

- `key`: string with key name
- `value`: string with setting value

NOTE for available settings see [pagePdfGlobal](#)

[\[View source\]](#)

```
def set_output(path : String) #
```

Set output path

- `path`: string with an output file path (extension included)

[\[View source\]](#)

```
def set_url(url : String) #
```

Set URL to fetch content from

- `url`: string with a complete URL (schema included)

[\[View source\]](#)