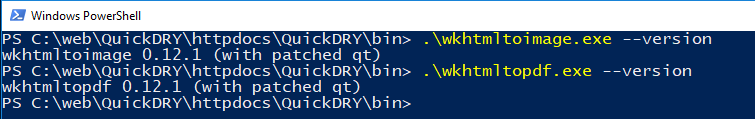
Webkit HTML to PDF

# The PDF Settings in class Web

There are currently 6 settings available in QuickDRY for rendering a PDF.

* RenderPDF – tells the framework to render the current page to a PDF
* PDFPageOrientation - either PDF\_PAGE\_ORIENTATION\_LANDSCAPE or PDF\_PAGE\_ORIENTATION\_PORTRAIT (these are defined in BasePage)
* PDFFileName – the name of the resulting file
* PDFPostRedirect – if set, will redirect the user rather than sending the PDF direct to the browser
* PDFHeader – the header that will be applied to each page
* PDFFooter – the footer that will be applied to each page

# About the version of wkhtmltopdf included with QuickDRY



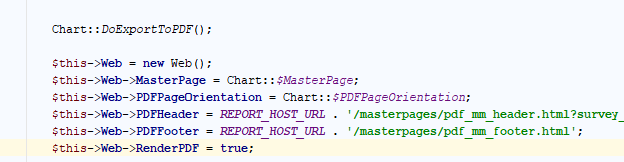
You may note that this is not the latest version of wkhtmltopdf. This is the 64-bit Windows version that seems to work best for the purposes QuickDRY has been developed for. When 0.12.4 was attempted, table headers were rendered over table data among other issues. 0.12.1 so far renders correctly for everything thrown at it. Your results may vary. Wkhtmltopdf has not had a release since 2016 and 0.12.1 is from 2014.

If you find a scenario where the page is not rendering correctly to PDF, consider upgrading or testing other versions of the tool. Different versions have different quirks. And in the case of QuickDRY, it’s more important to have something that works ready to go than to endlessly test and work around bugs. It is ultimately up to the end user to find the version that works best for their needs.

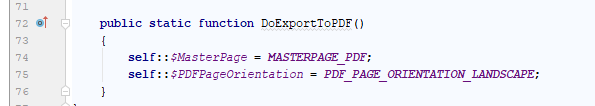
# RenderPDF

By default, this value is to set to false when the Web object is initialized. It is set to true when “export=pdf” is set in the HTTP request.

When creating cron jobs that export pages to PDF, you will need to manually set RenderPDF = true.



Note that the code for DoExportToPDF is simply:



Which sets the master page to be used of the body of the PDF (which does not impact the header or footer) and the orientation of the page.

When it comes time to render the page and export it to a file, you would use something like

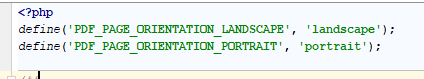


Note that including webkit.php is enough to render the PDF to the file. Webkit.php will detect that it was called from a command line script and save the file to the location PDFFileName rather than try to output it to a browser.

# PDFPageOrientation

This parameter has only two settings: “Portrait” and “Landscape”

For legacy purposes, setting it to “letter” will switch the setting to “Portrait”



These settings are defined in QuickDRY/web/BasePage.php so you can pull them up quickly.

This really should be its own class. The reason it isn’t is because originally DomPDF was used and was expected to be an option in QuickDRY. So many of the design choices are based on supporting legacy code.

# PDFFileName

When running from the command line, you will want to include the destination folder in the filename as the resulting PDF will be saved to that location.

When rendering a web-page as a PDF, only the name of the file should be in this setting and that will be the name the PDF downloads as for the end user.

If PDFPostRedirect is set, then destination folder also needs to be set, as the PDF will be stored in that location and the user will be redirected to the specified location.

# PDFPostRedirect

Occasionally you may want the user to trigger a PDF to be generated but for it to be stored on the server.

Using PDFPostRedirect in conjunction with a PDFFileName set with a folder location, it is possible to keep the PDF stored on the server and then send the user to another page to view and download files.

# PDFHeader / PDFFooter

When PDFHeader is set, the file will be put on the top of every page. When PDFFooter is set, the file will be on the bottom of every page.

They both work the exact same way.

This is where wkhtmltopdf documentation is very lacking so I will cover in detail some of the features that will probably be used most commonly.

The first is page numbers. Note again, that both the header and the footer work identically but do not communicate with each other. Putting something in the header has no impact on the footer. There are no global settings that can be passed from one to the other.



Wkhtmltopdf will render any HTML you throw at in the header/footer file. In this particular instance I am including bootstrap and font awesome so that the text in the header will match the style of the text in the body of the PDF.

Note the contents of the script tag.

It is easy enough to find this block of code to copy and paste into your own project.

<https://github.com/wkhtmltopdf/wkhtmltopdf/issues/3270>

Let’s say you want to put the name of the survey which you are reporting on in the header. There are in theory, three ways to do this.

1. Use the --replace <name> <value> parameter that wkhtmltopdf makes available
2. Reference a dynamically generated page
3. Use GET parameters when referencing your header / footer HTML file

If you can get method 1 to work, please let me know how with a complete example. It may be useful.

In theory you put some templatized variable in the HTML and then it will be replaced with the given value. That has not been directly implemented in QuickDRY because it is too poorly documented to create a working scenario.

Methods 2 and 3 are related. In order to use GET parameters you must reference the header / footer using a fully qualified HTTP request.

You cannot use

/masterpages/pdf\_mm\_header.html

you must use something like

http://reports.localhost.com/masterpages/pdf\_mm\_header.html

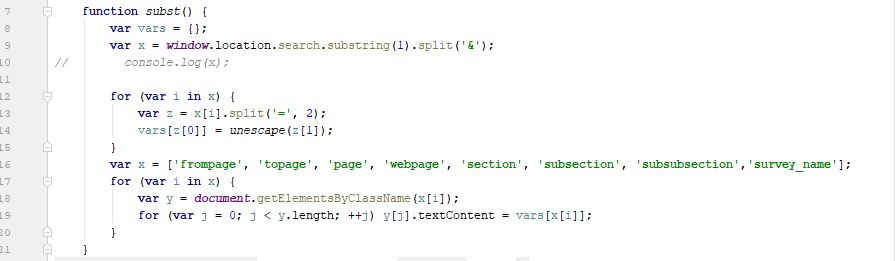
You can then add ?custom\_var=blah and have those values available to your header / footer

NOTE:

* frompage
* topage
* page
* webpage
* section
* subsection
* subsubsection

All appear to be reserved for wkhtmltopdf. If you attempt to modify them manually using HTTP GET parameters, your submitted values will be ignored. Because header and footer files are intended to be simple, if you want to handle these requests with a server-side language, you’re on your own.

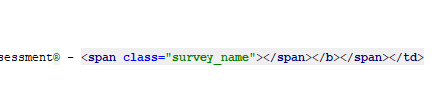
Note that you cannot just use whatever custom GET variables you want, you must configure the given JavaScript to handle them.



Notice that I have added “survey\_name” at the end of the list. Now this code will look for

?survey\_name=<something>

and set the HTML of the element with the class “survey\_name” to the value of survey\_name



That span will contain the text passed in through the HTTP GET parameter.