
Usage¶

Using with Python standalone¶

```
from xhtml2pdf import pisa             # import python module

# Define your data
source_html = "<html><body><p>To PDF or not to PDF</p></body></html>"
output_filename = "test.pdf"

# Utility function
def convert_html_to_pdf(source_html, output_filename):
   # open output file for writing (truncated binary)
   result_file = open(output_filename, "w+b")

   # convert HTML to PDF
   pisa_status = pisa.CreatePDF(
           source_html,                # the HTML to convert
           dest=result_file)           # file handle to recieve result

   # close output file
   result_file.close()                 # close output file

   # return False on success and True on errors
   return pisa_status.err

# Main program
if __name__ == "__main__":
   pisa.showLogging()
   convert_html_to_pdf(source_html, output_filename)
```

This basic Python example will generate a test.pdf file with the text ‘To PDF or not to PDF’ in the top left of the page. In-memory files can be generated by using StringIO or cStringIO instead of the file open. Advanced options will be discussed later in this document.

Using xhtml2pdf in Django¶

To allow URL references to be resolved using Django’s STATIC_URL and MEDIA_URL settings, xhtml2pdf allows users to specify a `link_callback` parameter to point to a function that converts relative URLs to absolute system paths.

```
import os
from django.conf import settings
from django.http import HttpResponse
from django.template.loader import get_template
```

```
from xhtml2pdf import pisa
from django.contrib.staticfiles import finders


   def link_callback(uri, rel):
           """
           Convert HTML URIs to absolute system paths so xhtml2pdf can access those
           resources
           """
           result = finders.find(uri)
           if result:
                   if not isinstance(result, (list, tuple)):
                           result = [result]
                   result = list(os.path.realpath(path) for path in result)
                   path=result[0]
           else:
                   sUrl = settings.STATIC_URL        # Typically /static/
                   sRoot = settings.STATIC_ROOT      # Typically /home/userX/project_static/
                   mUrl = settings.MEDIA_URL         # Typically /media/
                   mRoot = settings.MEDIA_ROOT       # Typically /home/userX/project_static/media/

                   if uri.startswith(mUrl):
                           path = os.path.join(mRoot, uri.replace(mUrl, ""))
                   elif uri.startswith(sUrl):
                           path = os.path.join(sRoot, uri.replace(sUrl, ""))
                   else:
                           return uri

           # make sure that file exists
           if not os.path.isfile(path):
                   raise Exception(
                           'media URI must start with %s or %s' % (sUrl, mUrl)
                   )
           return path

def render_pdf_view(request):
   template_path = 'user_printer.html'
   context = {'myvar': 'this is your template context'}
   # Create a Django response object, and specify content_type as pdf
   response = HttpResponse(content_type='application/pdf')
   response['Content-Disposition'] = 'attachment; filename="report.pdf"'
   # find the template and render it.
   template = get_template(template_path)
   html = template.render(context)

   # create a pdf
   pisa_status = pisa.CreatePDF(
      html, dest=response, link_callback=link_callback)
   # if error then show some funy view
   if pisa_status.err:
      return HttpResponse('We had some errors <pre>' + html + '</pre>')
   return response
```

You can see in action in `demo/djangoproject` folder

Using in Command line¶

xhtml2pdf also provides a convenient command line tool which you can use to convert HTML files to PDF documents using the command line.

```
$ xhtml2pdf test.html
```

This basic command will convert the content of test.html to PDF and save it to test.pdf. Advanced options will be described later in this document.

The `-s` option can be used to start the default PDF viewer after the conversion:

```
$ xhtml2pdf -s test.html
```

Advanced Command line tool options¶

Use `xhtml2pdf --help` to get started.

Converting HTML data¶

To generate a PDF from an HTML file called `test.html` call:

```
$ xhtml2pdf -s test.html
```

The resulting PDF will be called `test.pdf` (if this file is locked e.g. by the Adobe Reader it will be called `test-0.pdf` and so on). The `-s` option takes care that the PDF will be opened directly in the Operating Systems default viewer.

To convert more than one file you may use wildcard patterns like `*` and `?`:

```
$ xhtml2pdf "test/test-*.html"
```

You may also directly access pages from the internet:

```
$ xhtml2pdf -s http://www.xhtml2pdf.com/
```

Using special properties¶

If the conversion doesn't work as expected some more informations may be usefull. You may turn on the output of warnings adding `-w` or even the debugging output by using `-d`.

Another reason could be, that the parsing failed. Consider trying the `-xhtml` and `-html` options. `xhtml2pdf` uses the HTMLT5lib parser that offers two internal parsing modes: one for HTML and one for XHTML.

When generating the HTML output `xhtml2pdf` uses an internal default CSS definition (otherwise all tags would appear with no diffences). To get an impression of how this one looks like start `xhtml2pdf` like this:

```
$ xhtml2pdf --css-dump > xhtml2pdf-default.css
```

The CSS will be dumped into the file `xhtml2pdf-default.css`. You may modify this or even take a totaly self defined one and hand it in by using the `-css` option, e.g.:

```
$ xhtml2pdf --css=xhtml2pdf-default.css test.html
```

Next Previous
