Table of Contents

Quickstart 5

Installation 6

- From PyPi 6
- From source 6
- Requirements 6
  - macOS installation 6
  - Windows installation 6
- Using Sphinx-SimplePDF directives 7
- ReadTheDocs configuration 7

Building PDF 8

Configuration 8

- simplepdf_vars 8
  - Config vars 10
  - Examples 11
    - Color selection 11
    - File references 11
    - SimplePDF docs 11
- simplepdf_file_name 11
- simplepdf_debug 12
- simplepdf_use_weasyprint_api 12
- simplepdf_weasyprint_flags 13
- simplepdf_weasyprint_timeout 13
- simplepdf_theme 13
- simplepdf_theme_options 13

Directives 14

- if-builder 14
- if-include 15

PDF only 16

- pdf-include 16
  - Options 16

- width / height 16
- page 17
- toolbar 17

CSS voodoo 17

- Predefined css classes 18
    - Page breaks 18
    - Page Orientation 18
    - Table content wrap 19
    - Table size 19
    - Sphinx-Needs elements 19
- Customizing theme 20
    - config() functions 20

Technical details 20

- Workflow 20
- DEMO project 21

Changelog 21

- Release 1.6 21
- Release 1.5 22
- Release 1.4 22
- Release 1.3 23
- Release 1.2 23
- Release 1.1 23
- Release 1.0 23

License 23

Sphinx-SimplePDF

This Sphinx extension provides an easy way to build beautiful PDFs based on CSS rules.

It contains:

- A PDF specific, CSS based Sphinx theme: `sphinx_simplepdf` .
- A Sphinx builder, called `simplepdf`
- Directives to
    - control different structures and content for `HTML` and `PDF` builds.
    - embed PDF inside HTML views.

It is using weasyprint as PDF generator.

Note

This extension is in a beta phase.

It is not bug free and documentation is also missing some minor stuff. You can help us to make it better by reporting bugs or by providing code/docs changes via a PR. The code is available on github: useblocks/sphinx-simplepdf

Showcase

Sphinx-SimplePDF Documentation

The PDF is based on the current HTML documentation.

`Download PDF`

Sphinx-SimplePDF Demo

A PDF containing different content types to check the handling of them by Sphinx-SimplePDF.

`Download PDF`

Quickstart

Install via `pip install sphinx-simplepdf` .

Then inside your Sphinx documentation folder run `make simplepdf` . Your PDF is available under `_build/simplepdf` .

Color and images can be changed by setting `simplepdf_vars` inside your `conf.py` file:

```
simplepdf_vars = {
    'primary': '#333333',
    'links': '#FF3333',
}
```

For more configuration options take a look into Configuration .

For PDF/HTML specific content, use the `if-builder` directive.

```
# conf.py
extensions = ['sphinx_simplepdf']
```

```
# rst file
.. if-builder:: simplepdf

   .. toctree::

      my_files
      specific_pdf_file
```

```
.. if-builder:: html

   .. toctree::

      my_files

   Other HTML specific content, which will not be part of the PDF.
```

Installation

From PyPi

```
pip install sphinx-simplepdf
```

From source

```
git clone git@github.com:useblocks/sphinx-simplepdf.git
cd sphinx-simplepdf
pip install .
```

Requirements

Sphinx-SimplePDF requires **Sphinx version >= 4.4.4** to properly render the Table of Content with page counts.

macOS installation

If you are using **macOS** as operating system, the chance is high that the package **pango** gets not automatically installed when installing **Sphinx-SimplePDF** .

In this case please run also `brew install pango` .

Windows installation

Sphinx-SimplePDF is based on WeasypPrint, which is not so easy to get installed on Windows.

Please follow their instructions about how to install WeasyPrint on Windows .

Using Sphinx-SimplePDF directives

Sphinx-SimplePDF can be called directly after the installation.

However, if you want to use the included directives, like if-builder , you need to add Sphinx-SimplePDF to the list of extensions in your `conf.py` file:

```
extensions = [
    'sphinx_simplepdf',
    # additional extensions
]
```

ReadTheDocs configuration

Sphinx-SimplePDF can be also used on Read The Docs (RTD) to generate your PDF. As it is not supported by RTD by default, you need to create a `.readthedocs.yaml` configuration file on the root level of our project.

You can take the one from **Sphinx-SimplePDF** as a good example:

```
# .readthedocs.yaml
# Read the Docs configuration file
# See https://docs.readthedocs.io/en/stable/config-file/v2.html for
details

# Required
version: 2

# Set the version of Python and other tools you might need
build:
  os: ubuntu-22.04
  apt_packages:
    - default-jre  # This seems to be ignored
  tools:
    python: "3.9"
  # We can only define the content of the final deployment, if we do
the complete build on our own.apt_packages:
  # So we need to handle package installation, build start and
copying the right files for deployment.
  commands:
    - pip install .
    - pip install -r demo/doc-requirements.txt
    - sphinx-build -M simplepdf demo demo/_build  # Use a different
build-folder for a really clean build
    - pip install -r docs/doc-requirements.txt
    - sphinx-build -M simplepdf docs docs/_build2  # Use a different
```

```
build-folder for a really clean build
    - cp demo/_build/simplepdf/Sphinx-SimplePDF-DEMO.pdf docs/
_static/Sphinx-SimplePDF-DEMO.pdf
    - cp docs/_build2/simplepdf/Sphinx-SimplePDF.pdf docs/_static/
Sphinx-SimplePDF.pdf
    - sphinx-build -M html docs docs/_build  # HTML latest, because
it needs the built PDF files
    - mkdir -p _readthedocs/html/
    - cp -r docs/_build/html/* _readthedocs/html/
```

Building PDF

Inside your `docs` folder, run:

```
make simplepdf
```

or for more control:

```
sphinx-build -M simplepdf . _build
```

Configuration

simplepdf_vars

Sphinx-SimplePDF provides the config variable `simplepdf_vars` , which must be a dictionary. The key is used as identifier inside scss-files and the value must be a css/scss compatible string.

Example conf.py

```
simplepdf_vars = {
    'primary': '#FA2323',
    'secondary': '#379683',
    'cover': '#ffffff',
    'white': '#ffffff',
    'links': 'FA2323',
    'cover-bg': 'url(cover-bg.jpg) no-repeat center',
    'cover-overlay': 'rgba(250, 35, 35, 0.5)',
    'top-left-content': 'counter(page)',
    'bottom-center-content': '"Custom footer content"',
}
```

This values are used then inside the scss files, which define the PDF layout.

Config vars

primary :	Primary color
primary_opaque :	Primary color with opaqueness. Example `rgba(150, 26, 26, .5)`
secondary :	Secondary color
cover :	Text color on the cover
white :	A color representing white
links :	Color for links
cover-bg :	Cover background image. Can be a single color or even an image path.
cover-overlay :	RBG based color overlay for the cover-image. Example: `rgba(250, 35, 35, 0.5)`
top-left-content :	Text or css function to display on pdf output. Example: `counter(page)`
top-center-content :	Text or css function to display on pdf output.
top-right-content :	Text or css function to display on pdf output.
bottom-left-content :	Text or css function to display on pdf output.
bottom-center-content :	Text or css function to display on pdf output.
bottom-right-content :	Text or css function to display on pdf output.

All variables are defined inside `/themes/sphinx_simplepdf/sttuc/stles/sources/_variables.scss` .

Hint

If a content-string shall be set, please make sure to use extra “ around the string. Example: *‘bottom-center-content’: ‘“Custom footer content”’* .

Examples

The values from the configuration are taken as they are and injected into `scss` files, which are used to generate the css files. So each value or command, which is supported by `scss` , can be set.

Color selection

```
simplepdf_vars = {
    'primary': '#FA2323',
    'cover-overlay': 'rgba(250, 35, 35, 0.5)',
}
```

File references

```
simplepdf_vars = {
    'cover-bg': 'url(cover-bg.jpg) no-repeat center'
}
```

The file path must be relative to the Sphinx _static folder. So in the above example the image is stored under `/_static/cover-bg-jpg` .

SimplePDF docs

This is `simplepdf_vars` as it is used inside the **Sphinx-SimplePDF** `conf.py` file:

```
simplepdf_vars = {
    'cover-overlay': 'rgba(150, 26, 26, 0.7)',
    'cover-bg': 'url(cover-bg.jpg) no-repeat center'
}
```

simplepdf_file_name

New in version 1.5.

File name of the resulting PDF file in the `simplepdf` build folder. If not set, the project name is used.

File name and extension can be set. But it should not be used to manipulate the output path.

Example:

```
simplepdf_file_name = "my_cool.pdf"
```

Default: project name

simplepdf_debug

A boolean value. If set to `True` , **Sphinx-SimplePDF** will add some debug information add the end of the PDF.

This contains data about the used Python Environment and the Sphinx project. It is mainly used if any problems occur and extra information is needed.

`simplepdf_debug = True`

You can see an example in our `PDF Demo` at the end of the file.

Warning

The debug output contains absolute file paths and maybe other critical information. Do not use for official PDF releases.

simplepdf_use_weasyprint_api

New in version 1.6.

This forces simplepdf to use the weasyprint python API instead of calling the binary via subproces.

`simplepdf_use_weasyprint_api = True`

Warning

Other variables like simplepdf_weasyprint_flags will not work when using the API.

simplepdf_weasyprint_flags

New in version 1.5.

List of flags to pass to **weasyprint** subprocess. This may be helpfull in debugging the pdf creation

`simplepdf_weasyprint_flags = ['-v']`

Warning

The flags should only pass switches to **weasyprint** , input and output file names are appended by Sphinx-SimplePDF

simplepdf_weasyprint_timeout

New in version 1.5.

In rare cases **weasyprint** seems to run into infinite loops during processing of the input file. To avoid blocking CI jobs a timeout can be configured. The build is aborted with a `subprocess.TimeoutExpired` exception.

`simplepdf_weasyprint_timeout = 300`

simplepdf_theme

New in version 1.5.

Add custom theme for simplepdf. This overrides the default theme `simplepdf_theme`

simplepdf_theme_options

New in version 1.5.

Additional options for the theme. The default theme `simplepdf_theme` inherits all options from the **Sphinx Basic Theme** .

`simplepdf_theme` options:

nocover :	Do not display cover pages (front and back cover)

Directives

Warning

To use the directives, your `conf.py` file must contain Sphinx-SimplePDF in the extension list:

```
extensions = [
    'sphinx_simplepdf',
    # additional extensions
]
```

if-builder

`if-builder` can be used to define builder specific content.

The content of the directive gets added to the documentation only, if the provided directive argument matches the chosen builder name. The argument is case-insensitive.

Example

```
.. if-builder:: simplepdf

   .. toctree::

      my_files
      specific_pdf_file

.. if-builder:: html

   .. toctree::
      my_files

   Other HTML specific content, which will not be part of the PDF.
```

```
# Call examples
make simplepdf
sphinx-build -M html . _build
```

Warning

`if-builder` may not be taken into account, if a Sphinx incremental build is performed. Be sure to always use a clean first build, after a builder switch.

Note

Why not using the `.. only::` directive?

The `only` directive works differently and does not support for instance `toctree` and other mechanism for controlling the documentation structure.

if-include

`if-include` can be used to include files only when the specified builder is used. This is the same as using a include nested in a if-builder statement. You can list multiple files and use different builders.

```
.. if-builder:: simplepdf

   .. include:: ./path/to/my/file.xy

   .. include:: ./path/to/my/other/file.xy
```

is the same as

```
.. if-include:: simplepdf
```

```
    ./path/to/my/file.xy
    ./path/to/my/other/file.xy
```

Warning

in some cases content meant for html only builds will get included in the PDF if you build the html documentation and do not delete the build files.

Always make sure to use `make clean` or similar to delete build files before building the PDF.

The following chapter should only be visible in the PDF version of this documentation

PDF only

only visible in PDF

pdf-include

Includes a PDF file inside the the HTML code. The browser decides mostly, what kind of PDF-viewer.

```
.. pdf-include:: _static/SimplePDF_test.pdf
```

Options

For more options of how to configure the PDF viewer, take a look into the documentation of sphinxcontrib-pdfembed

width / height

Provide a number and unit for width/height.

Examples: `:width: 50%` , `height: 800px` .

Defaults:
`:width: 100%`
`:height: 400px` .

```
.. pdf-include:: _static/SimplePDF_test.pdf
   :width: 40%
   :height: 200px
```

page

Specify the page to show when PDF gets loaded.

Default: *None*

If not given the browser decides what page to open (normally page 1) and may also reuse the last seen page number.

```
.. pdf-include:: _static/SimplePDF_test.pdf
   :page: 2
```

toolbar

If set to `0` , the toolbar is hidden in most browsers (seems not to work on Firefox).

Default: `1`

```
.. pdf-include:: _static/SimplePDF_test.pdf
   :toolbar: 0
```

CSS voodoo

The PDF layout is configured via CSS files and their definitions.

You can use some Sphinx mechanism to set specific CSS class inside the rst, so that you can control the output a little bit.

Predefined css classes

Page breaks

Some Sphinx directives allow to use the option `:class:` .

If this is set to `break` , then a page break will be introduced in front of the element.

Example :

```
.. csv-table:: CSV Table
   :file: example.csv
   :class: break
```

Page Orientation

The default orientation is portrait. To change the page orientation for a side, you can add the css class `ssp-landscape` to

- directives supporting the option `:class:`
- or by using the `.. rst-class::` directive in the document with classes as arguments
- or by using the `.. container::` directive with the options for the used classes

Examples :

```
.. rst-class:: break_before, ssp-landscape, break_after

.. csv-table:: CSV Table
   :file: example.csv

.. or as alternaitve

.. container:: break_before ssp-landscape

   **Landscape page orientation**

   .. csv-table:: CSV Table
      :file: /_static/example.csv
      :header-rows: 2
```

If the default page orientation is changed to landscape, you can use `ssp-portrait` to change to portrait.

Table content wrap

By default table content is wrapped at whitespaces. If you have table content that can not be wrapped due to side limitations, the table is drawn out of the margins. This behaviour can be changed by using the css class `ssp-table-wrap` . This allows the table to break the content anywhere.

This requires a fixed table layout, so you have to set the `widths` options (or e.g. `colwidths` option in needtable) to get good results.

This option is by default added to all `Sphinx-Needs` elements or could be explicitly set by applying the `ssp-table-wrap` as `style` option to `Sphinx-Needs` directives.

Example :

```
.. list-table::
    :widths: 10,80
    :class: ssp-table-wrap
```

Table size

`ssp-tinier` and `ssp-tiny` can be set for a table to reduce the font-size and the internal padding of rows and columns. Together with Page Orientation huge tables can be presented inside a PDF.

Example :

```
.. container:: break_before ssp-landscape

    .. csv-table:: CSV Table
       :file: /_static/example.csv
       :class: ssp-tiny
```

Take a look into our Demo PDF for some examples of how tables could look like.

Sphinx-Needs elements

This works also for most Sphinx-Needs elements.

Example :

```
.. spec:: Specification Example
   :id: SPEC_001
   :style: break
```

or for `needtable` :

```
.. needtable::
   :filter: 'sphinx' in tags
   :class: break
```

Customizing theme

config() functions

Inside `scss` files you can use `config(name, default)` to get access to the values from `simplepdf_vars` .

The **default** values is used, if the **name** can not be found inside `simplepdf_vars` , which is the normal case, as `simplepdf_vars` is an empty dictionary by default.

Technical details

The sphinx-simplepdf registers the following stuff:

A sphinx builder, called `simplepdf` . Code inside `/builders/simplepdf.py` .

A sphinx theme, called `sphinx-simplepdf` . Files under `/themes/sphinx_simplepdf` .

During package installation, builder and theme get registered for Sphinx. This is done via the `enytry__points` mechanism.

```
        'static/js/*.js',
        'static/fonts/*.*'
    ]},
    include_package_data=True,
    # See http://www.sphinx-doc.org/en/stable/
theming.html#distribute-your-theme-as-a-python-package
    entry_points={
        'sphinx.html_themes': [
            'simplepdf_theme =
sphinx_simplepdf.themes.simplepdf_theme',
```

Workflow

1. User calls `make simplepdf` .