Sphinx-SimplePDF

Version 1.7.0

The simple PDF builder for Sphinx.

Build: 18.03.2026

Maintained by team useblocks

Table of Contents

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-lts-latest
  apt_packages:
    - default-jre # This seems to be ignored
  tools:
    python: "3.12"
  # 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"}

# use this to force using the weasyprint python API instead of
building via the binary
```

simplepdf_file_name

Added 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

Added 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

Added 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

Added 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_weasyprint_retries

Added in version 1.6.

In rare cases **weasyprint** seems to run into infinite loops during processing of the input file. In case a `subprocess.TimeoutExpired` exception occured and retries are configured **weasyprint** is started again.

```
simplepdf_weasyprint_retries = 1
```

simplepdf_theme

Added in version 1.5.

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

simplepdf_theme_options

Added 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)

simplepdf_weasyprint_filter

Added in version 1.6.

If **weasyprint** is used as executable the output contains warnings and errors from **weasyprint**. To reduce output noise the output can be filtered by a list of regular expressions given in this configuration option.

`simplepdf_weasyprint_filter = ["WARNING: Ignored"]`

To suppress all output, the quite flag *-q* should be used.

simplepdf_html_hook

Added in version 1.7.

Path to a Python script that will be called to manipulate the HTML before PDF generation. The script must define a function named `html_hook`. This allows custom transformations using BeautifulSoup.

Format: `"path/to/script.py"`

The path can be absolute or relative to the `conf.py` directory.

Example conf.py:

```
simplepdf_html_hook = "./hooks/pdf_hook.py"
```

Example hook script (hooks/pdf_hook.py):

```
from bs4 import BeautifulSoup

def html_hook(soup, app):
    """
    Customize HTML before PDF generation.

    Args:
        soup: BeautifulSoup object with parsed HTML
        app: Sphinx application instance
    Returns:
        Modified BeautifulSoup object
    """
    # Example: Remove navigation elements
    for nav in soup.find_all("nav"):
        nav.decompose()
```

```
    # Example: Add watermark
    watermark = soup.new_tag("div", attrs={"class": "watermark"})
    watermark.string = "DRAFT"
    body = soup.find("body")
    if body:
        body.insert(0, watermark)

    return soup
```

Function signature:

The `html_hook` function must accept two arguments:

soup:

A `BeautifulSoup` object containing the parsed HTML

app:

The Sphinx application instance (provides access to `config`, `srcdir`, `outdir`, etc.)

The function must return a `BeautifulSoup` object.

Error handling:

- If the script file is not found, a `ConfigError` is raised
- If the function is not found in the script, a `ConfigError` is raised
- If the hook returns `None`, a warning is logged and the original HTML is used
- If the hook returns a non-BeautifulSoup type, an error is raised
- If the hook raises an exception, it is wrapped in an `ExtensionError`

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

orphan:

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.

```
test = ["nox", "pytest", "pytest-xdist", "pytest-cov"]
docs = [
  "sphinx-copybutton",
  "sphinx-design",
  "sphinx-immaterial",
```

Workflow

1. User calls `make simplepdf`.
2. `simplepdf` builder overwrites theme to use `sphinx-simplepdf`.
3. Builder generates `main.css` from `main.scss` files. Injects also config-vars from `simplepdf_vars`.
4. Builder starts a **SingleFileHTML** based build.
5. Sphinx creates one single `index.html`.
6. Builder manipulates created `index.html`:
    - Fixes toc-tree links
7. Builders starts **weasyprint** with `index.html` as input
8. Done, PDF file exists under `_build/simplepdf`.

DEMO project

The DEMO project is stored under */demo/* and provides a common way for all developers and users to test everything on a common base.

It can be build by the following steps:

- `git clone git@github.com:useblocks/sphinx-simplepdf.git`
- `cd sphinx-simplepdf`
- `pip install .`
- `cd demo`
- `pip install -r doc-requirements.txt`
- `sphinx-build -a -E -b simplepdf . _build/`

Changelog

Release 1.8

- **Enhancement**: Add the ability to configure an simplepdf_html_hook to modify the HTML before PDF generation.

Release 1.7

released:

02.12.2025

- **Bugfix**: Replace deprecated *pkg_resources* with *importlib.metadata* for Python 3.12+ compatibility.
- **Bugfix** [#116] Prettify breaks HTML formatting.
- **Bugfix** [#82] TOC Tree Fix and Page Break Handling for document handling.
- **Bugfix** [#70] Added config option to build PDFs retries times if weasyprint binary fails to build.
- **Bugfix** [#66] Remove 'display: inline-block' from inline code blocks.

- **Enhancement** [#40] Added config option to filter warnings from weasyprint.
- **Enhancement** [#8] Latex equations rendered in demo.
- **Enhancement**: Add support for section and figure numbers.
- **Enhancement**: Add duplicate headline fix for toctree.
- **Enhancement**: Add duplicate check to toctree_fix function.

Release 1.6

released:

20.01.2023

- **Bugfix** [#60] Fix TOC hrefs for sections that use file title anchors.
- **Enhancement** [#62] Added config option to build PDFs with the weasyprint Python API instead of the binary.

Release 1.5

released:

26.10.2022

- **Enhancement**: `nocover` option for simplepdf_theme_options.
- **Enhancement**: New config options to directly configue the weasyprint build.
- **Enhancement**: New config option simplepdf_file_name allows to specify the output file name.
- **Enhancement**: Provides pdf-include directive to embed PDF files in HTML views.

Release 1.4

released:

22.09.2022

- **Enhancement**: Adds `simplepdf_debug` option, to collect and print some environment specific details.
- **Enhancement**: Adds *demo* PDF to test various layout & style elements.
- **Enhancement**: Replace not-open fonts with open-source fonts.
- **Enhancement**: All fonts are provided by this package. No pre-installed fonts are needed.
- **Enhancement**: [#19] Simple PDF customization with header and bottom page content
- **Enhancement**: Add class wrapper to switch paper orientation, e.g. for large tables
- **Enhancement**: Tables font-size and padding can be reduced by using `ssp-tinier` or `ssp-tiny`.
- **Bugfix**: [#34] handling word wrap in tables
- **Bugfix**: Image handling is done much better.
- **Bugfix**: Font location fixes -> No fonts warnings anymore.
- **Bugfix**: `html_theme_options` gets overwritten to suppress Sphinx warnings.
- **Bugfix**: HTML file operations are using hard-coded *utf-8* de/encoding.

Release 1.3

released:

25.08.2022

- **Improvement**: file-path in *url()* css configs are now relative to the Sphinx `_static` folder.
- **Improvement**: Toctree is working and page numbers are added.
- **Improvement**: Introducing if-builder to control content based on builder.

Release 1.2

released:

19.08.2022

- **Improvement**: Overwriting some html_config vars, which are not supported by simplepdf during a PDF build.

Release 1.1

released:

19.08.2022

- **Bugfix**: Adds missing scss files to installation package.
- **Bugfix**: Better cover image handling.

Release 1.0

released:

18.08.2022

License

```
MIT License

Copyright (c) 2026 useblocks GmbH

Permission is hereby granted, free of charge, to any person
obtaining a copy
of this software and associated documentation files (the
"Software"), to deal
in the Software without restriction, including without limitation
the rights
```

```
to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE
SOFTWARE.
```