Plugin Options

Manual

Revision 0.0.1

iSOLveIT

Contents

1. Options

The plugin allows users to pass in both global and local options. Some of these global options are used as default options when local options are not set or the plugin's default values are used by the PDF plugin.

> **Note**
>
> Local options have higher precedence than global options.

1.1. Global Options

The global options are passed to the plugin through the **conf.py** file and they are:

```
# Sphinx-PDF-Generate configurations
pdfgen_verbose = False
pdfgen_site_url = "https://isolveit.github.io/sphinx-pdf-generate/"
pdfgen_debug = True
pdfgen_debug_target = "index.rst"
pdfgen_author = "Sphinx-PDF"
pdfgen_author_logo = "_static/img/author-logo.png"
pdfgen_copyright = "2023, Sphinx-PDF"
pdfgen_disclaimer = "Disclaimer: Content can change at anytime and best to refer to website
for latest information."
pdfgen_cover = True
pdfgen_cover_title = ""
pdfgen_cover_subtitle = ""
pdfgen_custom_template_path = ""
pdfgen_custom_css_path = ""
pdfgen_plugin_handler_path = "custom_code.py"
pdfgen_toc = True
pdfgen_toc_numbering = True
pdfgen_toc_title = "Contents"
pdfgen_toc_level = 6
pdfgen_cover_images = {
    "default": "https://example.com/cover.svg",
    "type1": "_static/img/type1.png",
}
```

1.1.1. for Cover

1.1.1.1. pdfgen_cover

Set the value to `False` if you don't need a cover page.
default: `True`

1.1.1.2. pdfgen_cover_title

Set the title text on the cover page.
default: use `project` variable in your project's **conf.py**

> **Note**
>
> The following rule is applied when setting the value for cover title text in cover page.
>
> Cover Title precedence:
>
> 1. pdf-title (local pdf metadata option)
> 2. H1 heading for page
> 3. pdfgen_cover_title (global option)
> 4. `project` variable in your project's **conf.py**

1.1.1.3. pdfgen_cover_subtitle

Set the subtitle text in cover page.
default: `None`

> **Note**
>
> The following rule is applied when setting the value for cover subtitle text in cover page.
>
> Cover Subtitle precedence:
>
> 1. pdf-subtitle (local pdf metadata option)
> 2. pdf-type (local pdf metadata option)
> 3. pdfgen_cover_subtitle (global option)

1.1.1.4. pdfgen_author

Set the author text.
default: use `author` variable in your project's **conf.py**

1.1.1.5. pdfgen_author_logo

Provide a logo image which you can use in the cover page.
default: use `html_logo` variable in your project's **conf.py**

> **Tip**
>
> Using an SVG image as the value for author logo is recommended because it is easier to adjust the image size without losing the quality of the image.

1.1.1.6. pdfgen_copyright

Set the copyright text.
default: use `copyright` variable in your project's **conf.py**

1.1.1.7. pdfgen_disclaimer

Set the disclaimer text.
default: `None`

1.1.1.8. pdfgen_cover_images

Set the cover image for specific document types.

The option takes a Python dictionary where the `key` must be the one of the values you specified for the pdf-type local pdf metadata option.

The `value` for a `key` must be the path to the image.
default: `None`

Example:

```
pdfgen_cover_images = {
    "default": "https://example.com/cover.svg",
    "home": "_static/img/home-cv-img.png",
    "options": "_static/img/options-cv-img.png",
    "customise": "_static/img/custom-cv-img.png",
}
```

> **Note**
>
> Apart from the `default` key-value pair under the pdfgen_cover_images option, the other key-value pairs can have user-defined values.
>
> Recommended: You must specify an image path for the `default` key-value pair. We will use the image as the cover image for any document that does not specify the pdf-type local pdf metadata option.

1.1.2. for Heading and TOC

1.1.2.1. pdfgen_toc

Set the value to `False` if you don't need a table of content section in the PDF document.
default: `True`

1.1.2.2. pdfgen_toc_title

Set the title text for the *Table of Contents* page.
default: `Table of Contents`

1.1.2.3. pdfgen_toc_level

Set the maximum heading level to show on the *Table of Contents* page. The option's value is between `1` to `6`.
default: `4`

1.1.2.4. pdfgen_toc_numbering

Set the value to `False` if you don't want your table of contents to be numbered in the PDF document.
default: `True`

1.1.3. … and more

1.1.3.1. pdfgen_custom_template_path

A relative path inside your documentation's directory (i.e. relative to your project's **conf.py** file). This folder is where you save the custom cover template files (e.g. cover.html).
default: `None`

> **Note**
>
> The custom template's filename can either be `cover` or any of the document types you set with any of these file extensions `.html.j2`, `.html.jinja2`, `.html`, or `.htm`. Example: `cover.html.j2`, `cover.html.jinja2`, `cover.html`, `cover.htm` OR Example: if document type is `manual` then you can create a template file called `manual.html` or `manual.html.j2` or `manual.html.jinja2` or `manual.htm`.
>
> You can refer to this example about how to use a custom cover template.

1.1.3.2. pdfgen_custom_css_path

A relative path inside your documentation's directory (i.e. relative to your project's **conf.py** file). This folder is where you save the custom CSS file (i.e. **pdf_custom.css**).
default: `None`

> **Note**
>
> - The custom CSS filename must be `pdf_custom.css`. You can refer to this example about how to use a custom CSS file.
> - We use the plugin's CSS for the supported `html_theme` chosen under **conf.py** if this option is not set.

1.1.3.3. pdfgen_theme_handler_path

This option allows you to specify a custom theme handler module for unsupported themes. The path must be **relative to your project's root** (See example below).
default: `None`

`conf.py`:

```
pdfgen_theme_handler_path = "theme_handler.py"
```

project_root folder structure:

```
project_root
├── theme_handler.py
├── docs
    ├── conf.py
    ├── index.rst
    ├── _build
.
.
```

The `theme_handler.py` file should should contain these two functions below:

```
def get_stylesheet() -> str:
    """Function which returns the custom stylesheets for the theme you want to support"""


def modify_html(html: str, href: str) -> str:
    """Function which modifies the HTML to include where you want to place the PDF download
button at

    :param html: HTML content you can modify
    :param href: URL path to the PDF document generated
    """
```

Example of the above functions being used:

```
# theme_handler.py

def get_stylesheet():
css_style = """.md-container {{
display: block;
padding-top: 0;
}}
.md-main {{
    display: block;
    height: inherit;
}}"""
return css_style


def modify_html(html: str, href: str):
    a_tag = f'<a class="pdf-icon" href="{href}" download title="Download PDF"></a>'

    # insert into HTML
    insert_point = '<div class="document">'
    html = html.replace(insert_point, insert_point + a_tag)
    return html
```

1.1.3.4. pdfgen_plugin_handler_path

This option allows you to add custom features to the PDF generator plugin. The path must be **relative to your project's root** (See example below).
default: `None`

`conf.py`:

```
pdfgen_plugin_handler_path = "custom_code.py"
```

project_root folder structure:

```
project_root
├── custom_code.py
├── docs
    ├── conf.py
    ├── index.rst
    ├── _build
```

The `custom_code.py` file should should contain the `main()` function like below:

```
def some_function():
    """Just a function doing something"""
    return ""


def main(**kwargs) -> None:
    """Function called when the custom module is loaded and executed.

    :param **kwargs: Keyword arguments you can use in the function
    """

    variable = some_function()
    return variable

if __name__ == "__main__":
    main()
```

Example of the above function being used in a custom user plugin file called `feature_addon.py`:

```
# feature_addon.py

import re
from typing import List
from bs4 import BeautifulSoup, Tag


def main(**kwargs: BeautifulSoup) -> None:
    """Finds matching thumbnails and create their PDF versions"""
    soup = kwargs['soup']
    thumbnail_images = soup.find_all("img", attrs={"class": re.compile(r"product-img|img-
thumbnail")})
    pdf_thumbnails: List[Tag] = []
    for img in thumbnail_images:
        pdf_figure = soup.new_tag("figure", attrs={"class": "align-center"})
        pdf_img: Tag = soup.new_tag("img", attrs={"src": img["src"], "alt": img["alt"]})
        pdf_figcaption = soup.new_tag("figcaption")
        figcaption_text = soup.new_tag("p", attrs={"class": "caption-text"})
        figcaption_text.append(img["alt"])
        pdf_figcaption.append(figcaption_text)

        pdf_figure.append(pdf_img)
        pdf_figure.append(pdf_figcaption)
        pdf_thumbnails.append(pdf_figure)

    # Create a div containing new PDF thumbnails.
    pdf_div: Tag = soup.new_tag("div", attrs={"class": "product-images pdf-only"})
    for thumbnail in pdf_thumbnails:
        pdf_div.append(thumbnail)

    # Insert PDF Thumbnails after div#gallery
    html_thumbnails_node = soup.find("div", attrs={"id": "gallery"})
    html_thumbnails_node.insert_after(pdf_div)


if __name__ == "__main__":
    main()
```

1.1.3.5. pdfgen_verbose

Setting this to `True` will show all WeasyPrint debug messages during the build.
default: `False`

1.1.3.6. pdfgen_debug (for development purposes only)

Setting this to `True` enables the debug mode which saves the HTML content used in generating the PDF files into a folder called **pdf_html_debug**. The **pdf_html_debug** folder is relative to the documentation output (build) directory.

This option is intended to help users in debugging and writing appropriate CSS styles for the HTML content used to generate the PDF documents.

default: `False`

> **Tip**
>
> It is recommended to add the **pdf_html_debug** folder to your ignored files when using a version control system.

1.1.3.7. pdfgen_debug_target (for development purposes only)

This option helps you to generate a PDF file for a single target document so you can fine-tune the PDF output. The value for `debug_target` should be the path to the target document in your project's source directory or relative to your project's **conf.py** file. Example: `pdfgen_debug_target = "PCs/ARM/CS10600T070.rst`.

This option is intended to help reduce the time spent by users in debugging a single document used to generate a PDF file.

default: `None`

> **Note**
>
> You must set the pdfgen_debug option to `True`, if you want to use the `pdfgen_debug_target` option.

1.1.3.8. pdfgen_site_url

This option allows you to set the site url for your documentation project. Example:
`pdfgen_site_url = "https://isolveit.github.io/sphinx-pdf-generate/"`
default: `http://127.0.0.1:8000`

1.2. Local Options

The plugin allows you to set document specific options using the Sphinx File-wide metadata feature. If a local option is specified, it has higher precedence than the Global Options.

The local options are specified at the top of the RST file preceding any other markup like below:

```
:pdf-build: True
:pdf-filename: Plugin Options
:pdf-title: Options for Sphinx PDF Generate Plugin
:pdf-type: options
:pdf-revision: 0.1
```

The following options are available:

- pdf-build
- pdf-title
- pdf-subtitle
- pdf-type
- pdf-filename
- pdf-revision

1.2.1. pdf-build

Allows you to specify whether to generate a PDF file for the RST file or not. Value is `True` or `False`.
default: `True`

> **Note**
>
> The `pdf-build` option disables PDF generation for a single RST file but does not disable PDF generation for the entire project.

1.2.2. pdf-title

Set the title text on the cover page.
default: `None`

1.2.3. pdf-subtitle

Set the subtitle text on the cover page. The value of this option can be one or more text separated by pipe (|) symbol.

Example: `:pdf-subtitle: text1 | text2`
default: `None`

1.2.4. pdf-type

Set the document type.
default: `document`

> **Note**
>
> - We use the value, of this option, in selecting the document's cover image from the pdfgen_cover_images option.
> - Also, we use the value, of this option, in selecting the custom cover template from the folder set for the pdfgen_custom_template_path option.

1.2.5. pdf-filename

Set the filename to use in saving the generated PDF document.
default: `None`

> **Note**
>
> Filename precedence:
>
> 1. pdf-filename (local pdf metadata) - formatted such that all characters become valid characters
> 2. pdf-title (local pdf metadata) - formatted such that all characters become valid characters
> 3. The page's HTML title tag (i.e. `<title>`) value - formatted such that all characters become valid characters
> 4. Content of the first H1 tag (i.e. `<h1>`) in the page's HTML - formatted such that all characters become valid characters

1.2.6. pdf-revision

Set the document revision text on the cover page.
default: `None`