Customize Plugin

Manual

Revision 0.0.1

iSOLveIT

SPHINX
PDF GENERATE

2023, iSOLveIT

Disclaimer: Content can change at anytime and best to refer to website for latest information.

Contents

1. Customisation 3
   1.1. Custom cover page 3
      1.1.1. Using custom cover template 4
   1.2. Adjusting the output 5
      1.2.1. Changing the orientation of a page 6
         1.2.1.1. Example 7
   1.3. Roles 9
      1.3.1. pagebreak role 9

1. Customisation

1.1. Custom cover page

You can create a custom cover page using the jinja2 templating engine or HTML and saving the content in a file with these file extensions `.html.j2`, `.html.jinja2`, `.html`, or `.htm`.

> **Note**
>
> You should store the custom cover template file in the directory used as the value for pdfgen_custom_template_path option.

The plugin provides the following variables which you can use in your custom Jinja template:

- cover_title
- cover_subtitle
- cover_image
- author
- author_logo
- copyright
- disclaimer
- site_url
- revision
- project - (variable from **conf.py** file)
- version - (variable from **conf.py** file)
- release - (variable from **conf.py** file)
- and all the options you provided as the local pdf options in a particular RST file.

Using jinja2 syntax, you can access all the data above. Example: use *{{ author }}* to get the value for the pdfgen_author option:

1.1.1. Using custom cover template

You can specify the cover page to use for your PDF by following these steps:

Step 1

Set the pdfgen_custom_template_path option to the directory you want to store the cover template files at.

```
pdfgen_custom_template_path = "my_cover_templates_dir"
```

Step 2

Under the directory you set as `pdfgen_custom_template_path`, create a custom template file.
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`.

In the cover template file, write your preferred template syntax into it.

Example of a cover template file using Jinja2 syntax:

```
<article id="doc-cover">
    {% if cover_image is defined %}
        <div class="wrapper upper">
            <div class="logo" style="background-image: url('{{ cover_image | to_url }}');"></
div>
        </div>
    {% else %}
        <div class="wrapper"></div>
    {% endif %}
    <div class="wrapper">
        <h1>{{ cover_title | e }}</h1>
        <h2>{{ cover_subtitle | e }}</h2>
        {% if revision %}
            <h3>Revision {{ revision | e }}</h3>
        {% endif %}
    </div>
    <div class="properties">
        <address>
            {% if author %}
                <p id="author">{{ author | e }}</p>
            {% endif %}
        </address>
    </div>
    <div class="reserved_rights">
        <address>
            {% if copyright %}
                <p id="copyright">{{ copyright | e }}</p>
            {% endif %}
        </address>
    </div>
</article>
```

Step 3

Save the file changes and rebuild your Sphinx project.

1.2. Adjusting the output

The resulting PDF can be customized easily by adding a custom stylesheet such as the following:

```
@page {
    size: a4 portrait;
    margin: 25mm 10mm 25mm 10mm;
    counter-increment: page;
    font-family: "Roboto","Helvetica Neue",Helvetica,Arial,sans-serif;
    white-space: pre;
    color: grey;
    @top-left {
        content: '© 2018 My Company';
    }
    @top-center {
        content: string(chapter);
    }
    @top-right {
        content: 'Page ' counter(page);
    }
}
```

To implement the custom CSS, you need to create a `pdf_custom.css` file and save the custom CSS rules in it.

> **Note**
>
> You should store the `pdf_custom.css` file under the directory you set as pdfgen_custom_css_path

The plugin provides the following CSS variables and named strings which you can use in your `pdf_custom.css` file:

- –title
- –subtitle
- –author
- –author-logo
- –copyright
- –type
- –site_url
- –revision
- –filename
- chapter

Using the `var()` CSS function, you can access all the CSS variables provided by the plugin. E.g. use `var(--author)` to get the value for the pdfgen_author option.

You can also use the `string()` function to access the value of a named string. E.g. use `string(chapter)` to get the value for a chapter.

The custom CSS is appended to the Sphinx stylesheets so, you can choose to override rules by using the `! important` CSS keyword or not.

1.2.1. Changing the orientation of a page

The plugin allows you to change the orientation of a page to fit the content on that page.

For example, if you have a table on a page, and it is too wide to fit the current orientation used by the page, you can change the page orientation of the individual page by doing the following:

- Wrap the RST content in a raw HTML `div` element. The `div` element should have its `class` attribute set to `"rotated-page"`.
  Example:

```
.. raw:: html

    <div class="rotated-page">

PLACE CONTENT HERE

.. raw:: html

    </div>
```

- Create a `pdf_custom.css` file and set these CSS variables under the `:root {}` CSS rule:
  - `--base-page-orientation` - default page orientation to use and
  - `--rotated-page-orientation` - page orientation to use for rotated pages.
    Example:

```
:root {
    --base-page-orientation: a4 portrait;
    --rotated-page-orientation: a4 landscape;
}
```

1.2.1.1. Example

In this example, we are going to change the page orientation for this subsection.

Note

Download the generated pdf to see the result.

GPIO Table			
Header / Pin	**Symbol**	**Type**	**Description**
Header1 - 1	GND	Power	Module / System GND
Header1 - 2	IO3	I/O	GPIO – Capabilities are Module Dependent
Header1 - 3	IO2	I/O	GPIO – Capabilities are Module Dependent
Header1 - 4	IO1	I/O	GPIO – Capabilities are Module Dependent
Header1 - 5	3V3 OUT	Power	3.3V Power Output for User
Header2 - 1	RESET	I	System Reset, Active Low

In the example above, the example section is wrapped inside a `div` like below:

```
.. raw:: html

    <div class="rotated-page">

Example
*******

In this example, we are going to change the page orientation for `this subsection <example>`_.

.. note::

    Download the generated pdf to see the result.

.. table::
    :widths: auto

    +-------------------------------------------------------------------------------------+
    | GPIO Table                                                                          |
    +==================+=============+===========+=========================================+
    | **Header / Pin** | **Symbol**  | **Type**  | **Description**                         |
    +------------------+-------------+-----------+-----------------------------------------+
    | Header1 - 1      |   GND       | Power     | Module / System GND                     |
    +------------------+-------------+-----------+-----------------------------------------+
    | Header1 - 2      |   IO3       |  I/O      | GPIO – Capabilities are Module Dependent |
    +------------------+-------------+-----------+-----------------------------------------+
    | Header1 - 3      |   IO2       |  I/O      | GPIO – Capabilities are Module Dependent |
    +------------------+-------------+-----------+-----------------------------------------+
    | Header1 - 4      |   IO1       |  I/O      | GPIO – Capabilities are Module Dependent |
    +------------------+-------------+-----------+-----------------------------------------+
    | Header1 - 5      | 3V3 OUT     | Power     | 3.3V Power Output for User              |
    +------------------+-------------+-----------+-----------------------------------------+
    | Header2 - 1      |  RESET      |   I       | System Reset, Active Low                |
    +------------------+-------------+-----------+-----------------------------------------+

.. raw:: html

    </div>
```

and the `pdf_custom.css` file contains this code:

```
:root {
    --base-page-orientation: a4 portrait;
    --rotated-page-orientation: a4 landscape;
}
```

Note

You can write your own custom CSS to handle page orientation but you must use the **named page** CSS approach like below:

```
/* Named page ↓ */
@page rotated {
  size: A3 landscape;
}

.rotated-page {
  page: rotated;
  page-break-before: always;
  page-break-after: always;
}
```

1.3. Roles

1.3.1. pagebreak role

Adds a page break in the PDF document at the position it was called in the RST document. You can use the role by the calling the role in the RST document as `:pagebreak:`True``.

For example, you have the following RST document:

```
Title 1
=======

This is the first page content.

Title 2
=======

This is the next page content.
```

But if you want to move the section **Title 2** and its contents to the next page, you can do the following:

```
Title 1
=======

This is the first page content.

:pagebreak:`True`

Title 2
=======

This is the next page content.
```

Tip

You can click on the edit button at the beginning of each page to view how we used the `pagebreak` role in the RST source code used in generating the page.