MkDocs plugins with additional features

New features can be added to MkDocs engine by installing additional plugins. These packages can modify the navigation behavior, or render new content types, or export the site to PDF documents. Plugins also can be modified easily as they are written in Python.

#mkdocs

Last update: 2021-06-02 19:10:48

Table of Content

1. Awesome Pages

MkDocs Awesome Pages plugin simplifies configuring page titles and their entries order.

Install the plugin:

```
pip install -U mkdocs-awesome-pages-plugin
```

Enable it in the config file:

```
plugins:
    - search # built-in search must be always activated
    - awesome-pages
```

It overrides the `nav` sections in the site config file `mkdocs.yml`, and provides some more extra configs:

1. Create a YAML file named `.pages` in a directory and use the `nav` attribute to customize the navigation on that level. List the files and subdirectories in the order that they should appear in the navigation.
2. A 3-dots `...` entry is used to specify where all remaining items should be inserted. It can filter the remaining items using glob patterns or regular expressions. For example:

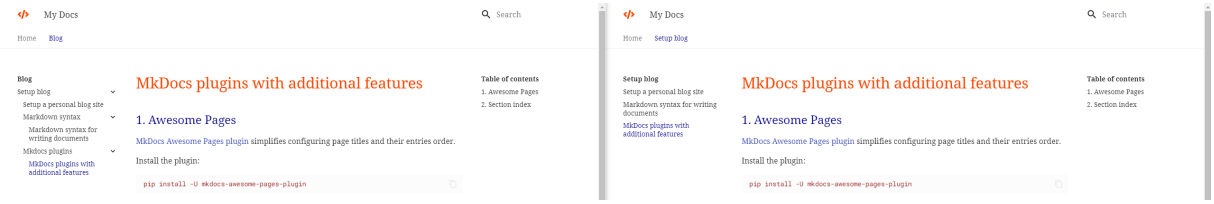
   .pages
   ```
   nav:
       - ... | introduction-*.md
       - ...
       - summary.md
   ```

   ⚠ The pattern is checked against the basename of remaining items - not their whole path, so it can be used to filter files in sub-folders.

3. Hide directory by setting the `hide` attribute to `true`.
4. Optionally set the directory title using the `title` attribute.
5. Optionally specify a title for the navigation entry before its document path. For example:

   .pages
   ```
   title: New section
   nav:
       - First page: page1.md
       - Link Title: https://example.com
   ```

6. Collapse single nested pages by setting `collapse_single_pages` attribute to `true`.



Using `collapse_single_pages` only, before and after applying

2. Section index

MkDocs Section Index is a plug that change the navigation sidebar to turn section name to a link that show the index page that section.

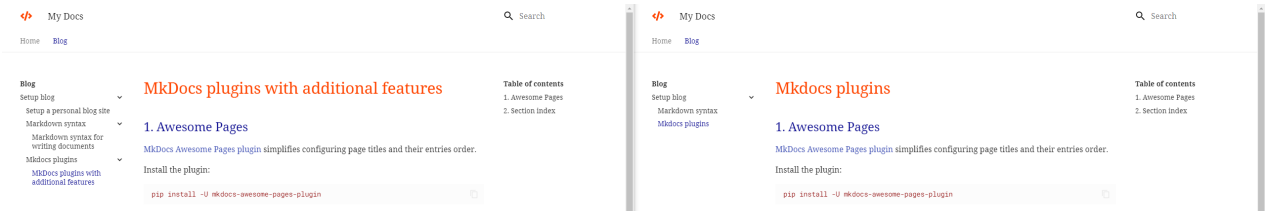
Install the plugin:

```
pip install -U mkdocs-section-index
```

Enable it in the config file:

```
plugins:
    - search # built-in search must be always activated
    - awesome-pages
    - section-index # must be after awesome-pages
```

In MkDocs, each directory will become a section, and by default, section only contains its children. There is no page associated to a section. This plugin will merge the `index.md` page in the directory to its section label. However, the merge section page show the section's title /* from directory name */, not the page's title.



Using the Section Index plugin only, before and after applying

This result is different from using `collapse_single_pages: true` option in the Awesome Pages plugin. If use single page collapsing, Awesome Pages plugin replace the section which has only one child by its child page. Therefore, a directory with `index.md` and some sub-folders will not be processed to merge the `index.md` to the section label of that directory.

Use both Awesome Pages `collapse_single_pages` option and Section Index will make a better result because both page and title are merged to the section.

3. Revision date

To keep tracking the last modified date of a post, git-revision-date plugin can be used. A better alternative plugin is git-revision-date-localized which provides more types of date format (even in time-ago format), and the creation date.

Install the plugin:

```
pip install -U mkdocs-git-revision-date-localized-plugin
```

Enable it in the config file:

```
plugins:
    - search # built-in search must be always activated
    - git-revision-date-localized:
          enable_creation_date: true
          type: iso_date
```

This plugin create new field in the post's meta-data which content the creation and update date. This information is used to sort the posts by revision date to get recently updated items, as shown in the Blog page. Read more in the Customize theme.

4. Print to PDF

To export the posts on this blog, there are plugins which can do it. However, most of them depend on Weasy Print which in turn depends on many other packages. There is one plugin that does printing in an easy and simple way: use browser to print page by sending print command (like press `Ctrl + S`).

More detail of installation and configuration the MkDocs PDF with JS plugin for printing to PDF can be read in Print to PDF.

5. Macros

⚠ **This plugin is no longer used in this site!**

MkDocs Macros is a plugin/framework that makes it easy to produce richer and more beautiful pages. It can do two things:

1. Transform the markdown pages into a Jinja2 templates that can use variables, macros and filters.
2. Replace MkDocs plugins for a wide range of tasks: e.g. manipulating the navigation, adding files after the html pages have already been generated etc.

Install the plugin:

```
pip install -U mkdocs-macros-plugin
```

Enable it in the config file:

```
plugins:
    - search # built-in search must be always activated
    - macros
```

Incomplete data in marco

The macro `{{ navigation.pages }}` contains a list of all pages, but the data of each page maybe not complete, such as title or meta-data.

This issue happens when rendering a the content of the first page, but it needs to know the content of the second page which has not been parsed already as it is waiting for the first page getting done.

6. DrawIO Exporter

This plugin is no longer used in this site!

DrawIO Exporter is a great plugin that exports the `.drawio` diagrams to images at build time and insert them to the document. This plugin can replace the Mermaid plugin, and it is faster thanks to no javascript needed at runtime. It also helps to enable instant navigation mode of the Material theme.

Install the plugin:

```
pip install -U mkdocs-drawio-exporter
```

Enable it in the config file:

```
plugins:
    - search # built-in search must be always activated
    - drawio-exporter
```

To create end edit `.drawio` diagram, download and install the diagrams.net application.

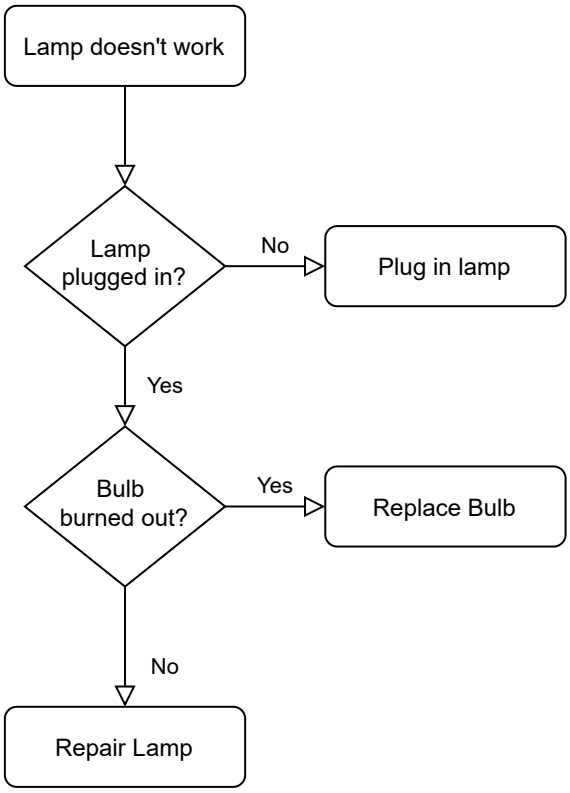
To import a diagram, just use the syntax for inserting an image:

```
![My alt text](my-diagram.drawio)
```

The plugin will generate an svg image to a cache folder (default in `docs\drawio-exporter`), and then modify the image source attribute to the generated image.

If the diagram is a multi-page documents, append the index of the page as an anchor in the URL:

```
![Page 1](my-diagram.drawio#0)
```



A draw.io diagram

A limitation

Using Draw.io Integration extension in Visual Studio Code, I can save a DrawIO diagram as a `.drawio.svg` file, then use that file directly in the page as an usual image. However this method will not support multiple pages in the drawing:

```
![My alt text](my-diagram.drawio.svg) // work
![My alt text](my-diagram.drawio.svg#1) // does not work
```

7. Mermaid

This plugin is no longer used in this site!

MkDocs Mermaid2 is a plugin to render textual graph description into Mermaid graphs (flow charts, sequence diagrams, pie charts, etc.).

Install the plugin:

```
pip install -U mkdocs-mermaid2-plugin
```

Enable it in the config file:

```
plugins:
    - search # built-in search must be always activated
    - mermaid2
```

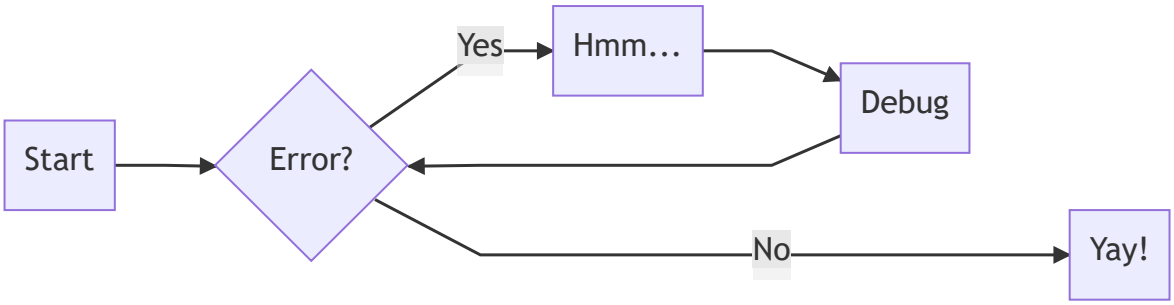
And configure the codeblock parser for mermaid2 blocks:

```
markdown_extensions:
    - pymdownx.superfences:
          custom_fences:
              - name: mermaid
                class: mermaid
                format: !!python/name:mermaid2.fence_mermaid
```

Example:

```
```mermaid
graph LR
  A[Start] --> B{Error?};
  B -->|Yes| C[Hmm...];
  C --> D[Debug];
  D --> B;
  B ---->|No| E[Yay!];
```
```

will render as:



A diagram generated by Mermaid