# oerforge.make

Hugo-style Markdown to HTML Static Site Generator (Python)

## Overview

oerforge.make provides functions for building static HTML sites from Markdown using Jinja2 templates, asset management, navigation, and download button generation. It is inspired by Hugo and designed for clarity, maintainability, and extensibility.

## Functions

### copy\_static\_assets\_to\_build

def copy\_static\_assets\_to\_build()

Copy static assets (CSS, JS, images) from static/ to build/. Overwrites files each time it is called.

### get\_available\_downloads\_for\_page

def get\_available\_downloads\_for\_page(rel\_path, page\_dir=None)

Scan the published output directory for a page and return a list of available download formats.

**Parameters** - rel\_path (str): Relative path to the HTML file (e.g., ‘about/index.html’). - page\_dir (str, optional): Directory containing downloadable files.

**Returns** - list[dict]: List of available downloads with label, filename, href, theme, and aria\_label.

### build\_download\_buttons\_context

def build\_download\_buttons\_context(rel\_path, page\_dir=None)

Build the download buttons context for a page.

**Parameters** - rel\_path (str): Relative path to the HTML file. - page\_dir (str, optional): Directory containing downloadable files.

**Returns** - list[dict]: List of button dictionaries for the template.

### slugify

def slugify(title: str) -> str

Convert a title to a slug suitable for folder names.

**Parameters** - title (str): Page or section title.

**Returns** - str: Slugified string.

### load\_yaml\_config

def load\_yaml\_config(config\_path: str) -> dict

Load and parse the YAML config file.

**Parameters** - config\_path (str): Path to the YAML config file.

**Returns** - dict: Parsed configuration data.

### ensure\_output\_dir

def ensure\_output\_dir(md\_path)

Ensure the output directory for the HTML file exists, mirroring build/files structure.

**Parameters** - md\_path (str): Path to the Markdown file.

### setup\_template\_env

def setup\_template\_env()

Set up the Jinja2 template environment for rendering pages.

**Returns** - jinja2.Environment: Configured Jinja2 environment.

### render\_page

def render\_page(context: dict, template\_name: str) -> str

Render a page using Hugo-style templates.

**Parameters** - context (dict): Context dictionary for the template. - template\_name (str): Name of the template file.

**Returns** - str: Rendered HTML string.

### generate\_nav\_menu

def generate\_nav\_menu(context: dict) -> list

Generate top-level navigation menu items from the content table.

**Parameters** - context (dict): Context dictionary, typically with rel\_path.

**Returns** - list[dict]: List of menu item dictionaries.

### get\_header\_partial

def get\_header\_partial(context: dict) -> str

Render the header partial using Jinja2.

**Parameters** - context (dict): Context dictionary for the template.

**Returns** - str: Rendered header HTML.

### get\_footer\_partial

def get\_footer\_partial(context: dict) -> str

Render the footer partial using Jinja2.

**Parameters** - context (dict): Context dictionary for the template.

**Returns** - str: Rendered footer HTML.

### convert\_markdown\_to\_html

def convert\_markdown\_to\_html(md\_path: str) -> str

Convert Markdown to HTML using markdown-it-py, rewriting local image paths and adding accessibility roles.

**Parameters** - md\_path (str): Path to the Markdown file.

**Returns** - str: Rendered HTML string.

### convert\_markdown\_to\_html\_text

def convert\_markdown\_to\_html\_text(md\_text: str) -> str

Convert Markdown text to HTML using markdown-it-py, rewriting local image paths and adding accessibility roles.

**Parameters** - md\_text (str): Markdown text.

**Returns** - str: Rendered HTML string.

### get\_asset\_path

def get\_asset\_path(asset\_type, asset\_name, rel\_path)

Compute the relative asset path for CSS, JS, or images based on page depth.

**Parameters** - asset\_type (str): Asset type (‘css’, ‘js’, ‘images’). - asset\_name (str): Asset filename. - rel\_path (str): Relative path to the page.

**Returns** - str: Relative asset path.

### add\_asset\_paths

def add\_asset\_paths(context, rel\_path)

Add asset paths (CSS, JS, logo) to the context for template rendering.

**Parameters** - context (dict): Context dictionary. - rel\_path (str): Relative path to the page.

**Returns** - dict: Updated context dictionary.

### get\_top\_level\_sections

def get\_top\_level\_sections(db\_path=None)

Get all top-level sections from the database for section index generation.

**Parameters** - db\_path (str, optional): Path to the SQLite database.

**Returns** - list[tuple]: List of (section\_title, output\_dir) tuples.

### build\_section\_indexes

def build\_section\_indexes()

Generate index.html files for all top-level sections using the section template.

### build\_all\_markdown\_files

def build\_all\_markdown\_files()

Build all Markdown files using Hugo-style rendering, using the first # header as the title.

### create\_section\_index\_html

def create\_section\_index\_html(section\_title: str, output\_dir: str, context: dict)

Generate section index.html using the section.html template.

**Parameters** - section\_title (str): Title of the section. - output\_dir (str): Output directory for the section index. - context (dict): Context dictionary for the template.

## Usage Example

from oerforge import make  
make.build\_section\_indexes()  
make.build\_all\_markdown\_files()

## Requirements

* Python 3.7+
* Jinja2
* markdown-it-py
* mdit-py-plugins
* PyYAML
* SQLite3

## See Also

* [Jinja2 Documentation](https://jinja.palletsprojects.com/)
* [markdown-it-py Documentation](https://markdown-it-py.readthedocs.io/)

## License

See LICENSE in the project root.