Source Code (makesite.py)

Sunaina Pai

PyCon UK 2018, Cardiff, UK

GitHub URL:
https://github.com/sunainapai/makesite
Short URL:
https://git.io/makesite

```python
"""Make static website/blog with Python."""


import os
import shutil
import re
import glob
import sys
import json
import datetime


def fread(filename):
    """Read file and close the file."""
    with open(filename, 'r') as f:
        return f.read()


def fwrite(filename, text):
    """Write content to file and close the file."""
    basedir = os.path.dirname(filename)
    if not os.path.isdir(basedir):
        os.makedirs(basedir)

    with open(filename, 'w') as f:
        f.write(text)


def log(msg, *args):
    """Log message with specified arguments."""
    sys.stderr.write(msg.format(*args) + '\n')


def truncate(text, words=25):
    """Remove tags and truncate text to the specified number of words."""
    return ' '.join(re.sub('(?s)<.*?>', ' ', text).split()[:words])


def read_headers(text):
    """Parse headers in text and yield (key, value, end-index) tuples."""
    for match in re.finditer('\s*<!--\s*(.+?)\s*:\s*(.+?)\s*-->\s*|.+', text):
        if not match.group(1):
            break
        yield match.group(1), match.group(2), match.end()


def rfc_2822_format(date_str):
    """Convert yyyy-mm-dd date string to RFC 2822 format date string."""
    d = datetime.datetime.strptime(date_str, '%Y-%m-%d')
    return d.strftime('%a, %d %b %Y %H:%M:%S +0000')


def read_content(filename):
    """Read content and metadata from file into a dictionary."""
    # Read file content.
    text = fread(filename)

    # Read metadata and save it in a dictionary.
    date_slug = os.path.basename(filename).split('.')[0]
    match = re.search('^(?:(\d\d\d\d-\d\d-\d\d)-)?(.+)$', date_slug)
    content = {
        'date': match.group(1) or '1970-01-01',
        'slug': match.group(2),
    }

    # Read headers.
    end = 0
    for key, val, end in read_headers(text):
        content[key] = val

    # Separate content from headers.
    text = text[end:]

    # Convert Markdown content to HTML.
    if filename.endswith(('.md', '.mkd', '.mkdn', '.mdown', '.markdown')):
        try:
            if _test == 'ImportError':
                raise ImportError('Error forced by test')
            import CommonMark
            text = CommonMark.commonmark(text)
        except ImportError as e:
            log('WARNING: Cannot render Markdown in {}: {}', filename, str(e))

    # Update the dictionary with content and RFC 2822 date.
    content.update({
        'content': text,
        'rfc_2822_date': rfc_2822_format(content['date'])
    })

    return content


def render(template, **params):
    """Replace placeholders in template with values from params."""
    return re.sub(r'{{\s*([^}\s]+)\s*}}',
                  lambda match: str(params.get(match.group(1), match.group(0))),
                  template)


def make_pages(src, dst, layout, **params):
    """Generate pages from page content."""
    items = []

    for src_path in glob.glob(src):
        content = read_content(src_path)

        page_params = dict(params, **content)

        # Populate placeholders in content if content-rendering is enabled.
        if page_params.get('render') == 'yes':
            rendered_content = render(page_params['content'], **page_params)
            page_params['content'] = rendered_content
            content['content'] = rendered_content

        items.append(content)

        dst_path = render(dst, **page_params)
        output = render(layout, **page_params)

        log('Rendering {} => {} ...', src_path, dst_path)
        fwrite(dst_path, output)

    return sorted(items, key=lambda x: x['date'], reverse=True)


def make_list(posts, dst, list_layout, item_layout, **params):
    """Generate list page for a blog."""
    items = []
    for post in posts:
        item_params = dict(params, **post)
        item_params['summary'] = truncate(post['content'])
        item = render(item_layout, **item_params)
        items.append(item)

    params['content'] = ''.join(items)
    dst_path = render(dst, **params)
    output = render(list_layout, **params)

    log('Rendering list => {} ...', dst_path)
    fwrite(dst_path, output)


def main():
    # Create a new _site directory from scratch.
    if os.path.isdir('_site'):
        shutil.rmtree('_site')
    shutil.copytree('static', '_site')

    # Default parameters.
    params = {
        'base_path': '',
        'subtitle': 'Lorem Ipsum',
        'author': 'Admin',
        'site_url': 'http://localhost:8000',
        'current_year': datetime.datetime.now().year
    }

    # If params.json exists, load it.
    if os.path.isfile('params.json'):
        params.update(json.loads(fread('params.json')))

    # Load layouts.
    page_layout = fread('layout/page.html')
    post_layout = fread('layout/post.html')
    list_layout = fread('layout/list.html')
    item_layout = fread('layout/item.html')
    feed_xml = fread('layout/feed.xml')
    item_xml = fread('layout/item.xml')

    # Combine layouts to form final layouts.
    post_layout = render(page_layout, content=post_layout)
    list_layout = render(page_layout, content=list_layout)

    # Create site pages.
    make_pages('content/_index.html', '_site/index.html',
               page_layout, **params)
    make_pages('content/[!_]*.html', '_site/{{ slug }}/index.html',
               page_layout, **params)

    # Create blogs.
    blog_posts = make_pages('content/blog/*.md',
                            '_site/blog/{{ slug }}/index.html',
                            post_layout, blog='blog', **params)
    news_posts = make_pages('content/news/*.html',
                            '_site/news/{{ slug }}/index.html',
                            post_layout, blog='news', **params)

    # Create blog list pages.
    make_list(blog_posts, '_site/blog/index.html',
              list_layout, item_layout, blog='blog', title='Blog', **params)
    make_list(news_posts, '_site/news/index.html',
              list_layout, item_layout, blog='news', title='News', **params)

    # Create RSS feeds.
    make_list(blog_posts, '_site/blog/rss.xml',
              feed_xml, item_xml, blog='blog', title='Blog', **params)
    make_list(news_posts, '_site/news/rss.xml',
              feed_xml, item_xml, blog='news', title='News', **params)


# Test parameter to be set temporarily by unit tests.
_test = None


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