Записки программиста

Блог о программировании, а также электронике, радио, и всяком таком

Поднимаем статический блог на Pelican и GitHub Pages

24 апреля 2017

Генераторы статических сайтов, такие, как Octopress, Jekyll или Hakyll, не даром пользуются большой популярностью. Поскольку сайт получается статическим, для работы ему не нужно ничего, кроме какого-нибудь Nginx. Естественно, сайты получаются очень быстрыми, не требовательными к ресурсам, а также безопасными, так как никаких админок и прочего на них просто физически нет. Многим нравятся языки разметки типа Markdown, и генераторы статических сайтов используют именно их. В общем, звучит как что-то, что мне хотелось бы попробовать. А поскольку из скриптовых языков я предпочитаю Python, выбор пал на написанный на этом языке генератор статический сайтов Pelican.

Использование GitHub Pages

Если не хочется платить за хостинг, стоит обратить внимание на GitHub Pages. Просто создаем репозиторий с именем вроде «afiskon.github.io». Затем говорим:

```
git clone git@github.com:afiskon/afiskon.github.io.git
cd afiskon.github.io
echo '<h1>Hello</h1>' > index.html
git add index.html
git commit -am 'index.html created'
git push origin HEAD
```

Идем на afiskon.github.io и видим там «Hello». Помимо бесплатности, GitHub Pages интересен тем, что дает всем TLS, а также позволяет прикрутить собственное доменное имя. Правда, в последнем случае, насколько я понимаю, TLS вы потеряете.

Дополнение: В контексте TLS и собственного доменного имени вас также может заинтересовать пост Настройка HTTPS с сертификатами Let's Encrypt. В этом случае, правда, вам придется арендовать для сайта виртуальный сервер. Однако цена вопроса на момент написания этих строк составляет 5$ в месяц или даже меньше, в зависимости от VDS-провайдера.

Установка и настройка Pelican

Первым делом Pelican нужно установить. Чтобы не засорять систему пакетами, воспользуемся virtualenv:

```
cd path/to/afiskon.github.io
mkvirtualenv pelican
pip install pelican markdown
mkdir src
cd src
pelican-quickstart
```

Отвечаем на серию несложных вопросов — название блога, его URL, и так далее. Затем создаем файл content/hello.md примерно такого содержания:

```
Title: Hello, World!
Date: 2017-04-05 13:04
Modified: 2017-04-05 13:04

This is a test blog post.
```

Генерируем сайт:

```
pelican --relative-urls --ignore-cache -o .. content
chromium ../index.html
```

Шаблон по умолчанию в Pelican, прямо скажем, на любителя. Вам, вероятно, захочется выбрать себе какой-то другой шаблон. Так как шаблон вы, скорее всего, будете немного допиливать под себя, положим его в src/template:

```
git clone https://github.com/fle/pelican-simplegrey.git
rm -rf pelican-simplegrey/.git
```

Также создадим файл src/gen.sh следующего содержания:

```
#!/bin/sh

set -e

rm ../*.html 2>/dev/null || true
rm -r ../author 2>/dev/null || true
rm -r ../category 2>/dev/null || true
rm -r ../rss 2>/dev/null || true
rm -r ../theme 2>/dev/null || true

pelican --relative-urls --ignore-cache \
  --theme-path template/pelican-simplegrey \
  -o .. content

git add ../*.html
```

Приведенный скрипт не имеет проблем с кэшированием Pelican'ом шаблона, а также ряда других. В корне репозитория просто всегда будет статический код вашего сайта ровно в

том виде, в котором вы ожидаете. Поскольку и исходники сайта и генерируемый HTML-код было решено свалить в одну кучу, создадим в корне сайта файл robots.txt, запрещающий поисковым системам индексировать каталог src:

```
User-agent: *
Disallow: /src/
Crawl-delay: 10
Host: afiskon.github.io
```

Как альтернативный вариант, можно использовать два репозитория — один с исходниками сайта, второй со статикой. Такой подход в чем-то удобнее, в чем-то менее удобен. Смотрите сами, как вам больше нравится.

Для генерации RSS-лент в pelicanconf.py дописываем:

```
FEED_ALL_RSS = 'rss/all.xml'
CATEGORY_FEED_RSS = 'rss/%s.xml'
```

По большому счету, это все. Добавляете все файлы в Git, коммитите, наслаждаетесь результатом. Если нужны новые страницы, пишите их в Markdown'е по аналогии с hello.md.

Небольшая памятка по Markdown

Я лично постоянно забываю, как сделать то или иное в Markdown. Поэтому есть основания полагать, что следующая шпаргалка кому-нибудь пригодится. Следует отметить, что у Markdown есть много диалектов. В частности, GitHub и BitBucket могут один и тот же файл рендерить по-разному, так что будьте внимательны. Итак, шпаргалка.

Заголовки (h1, h2, ⋯, h6):

```
# Заголовок h1
## Заголовок h2
...
###### Заголовок h6
```

Жирный, курсив:

```
*курсив 1*
_курсив 2_
**жирный 1**
__жирный 2__
**Жирный и _курсив_**
```

Зачеркнутый текст:

```
~~текст~~
```

Цитаты:

```
> Ололо
> Трололо

Непронумерованный список (перед ним должна быть пустая строка!):

- item1
- item2

или:

* item1
* item2

Пронумерованный список:

1. item 1
2. item 2

Вложенные списки создаются при помощи отступов из четырех пробелов:

1. item 1
    1. subitem 1.1
    2. subitem 1.2

Куски кода:

```
большой кусок кода
```

```cpp
кусок кода с указанием языка
```

А так делается `пример кода в тексте`.

Ссылки:

Вот так можно [делать ссылки](https://eax.me/)

А еще [вот так][u1].

Текст, текст, текст.

[u1]: https://eax.me/

Картинки:

Пример ![alt text](https://example.ru/image.png)
```

```
TODO-списки:

- [ ] не сделано
- [x] сделано
    - [ ] тоже могут быть вложенными

Таблицы:

Заголовок 1 | Заголовок 2
------------|------------
текст 1.1   | текст 2.1
текст 1.2   | текст 2.2

Можно указывать выравнивание текста в колонке — слева, по центру или справа:

Заголовок 1 | Заголовок 2 | Заголовок 3
:-----------|:-----------:|-----------:
выравнивание слева | по центру | и справа

Заметьте, символы | выравнивать не обязательно. Наконец, чисто для красоты можно
дописать их в начале и конце:

| Заголовок 1 | Заголовок 2 |
| ------------|-------------|
| текст 1.1   | текст 2.1   |
| текст 1.2   | текст 2.2   |

Иногда поддерживаются Emoji, в частности, их поддерживает GitHub:

:smile:
:cry:
:wink:
:sleeping:
:angry:

Больше Emoji вы найдете на странице Emoji Cheat Sheet.

Наконец, иногда допускается использование HTML, в частности это поддерживает Pelican.
HTML нужен для встраивания в страницу видео с YouTube, слайдов со SlideShare, и так
далее. Кроме того, можно, к примеру, указывать выравнивание элементов страницы:

<img style="float: right;" src="/static/about.jpg" alt="Photo" />

Для предварительного просмотра Markdown можно использовать Python:

sudo pip install markdown
python -m markdown article.md
python -m markdown article.md -f ~/temp/t.html
```

Вместо python -m markdown можно использовать команду markdown_py. Если вы используете Vim, советую дописать в ~/.vimrc пару новых команд:

```
command! MarkdownPreview !python -m markdown % -f ~/temp/t.html &&
 \ chromium ~/temp/t.html
command! MarkdownUpdate !python -m markdown % -f ~/temp/t.html

au BufRead *.md set wrap tw=80
```

Последняя инструкция ограничивает ширину строки в Markdown-файлах до 80 символов. Чтобы вручную не исправлять соответствующим образом текст после редактирования, используйте команду gq}. Если же вы используете Sublime Text, для него есть плагин Markdown Preview.

Еще существует интересный проект Markdown Plus, который добавляет в Markdown графики, формулы и многое другое. Увы, на данный момент такой формат не поддерживается ни одним популярным сервисом или приложением.

Заключение

Из минусов статических сайтов на ум приходит, например, что для публикации статей по расписанию придется потанцевать с бубном и cron'ом. Чтобы на сайте появились комментарии, придется прикрутить какой-нибудь Disqus. Поиска по сайту из коробки тоже нет. Придется сделать форму поиска, отправляющую в Google, вроде такой:

```
<form method="get" id="searchform"
 action="https://encrypted.google.com/search">
<div>
<input type="hidden" name="as_sitesearch" value="afiskon.github.io" />
<input type="text" id="s" name="as_q" value="" />
<input type="submit" id="searchsubmit"
 value="  Поиск  " />
</div>
</form>
```

С другой стороны, на момент написания этих строк данный блог работал на WordPress, и все равно использовал комментарии Disqus с поиском Google. Первый хорошо режет спам и требует от пользователя один-единственный раз авторизоваться через какой-нибудь Twitter. Второй просто достаточно быстро индексирует и хорошо ищет, так зачем, спрашивается, нагружать сайт самостоятельным поиском?

Статические сайты интересны еще и тем, что позволяют писать посты оффлайн. С другой стороны, я не припомню, чтобы в последнее время внезапно оставался без интернета. Да и пост для WordPress я с тем же успехом могу написать оффлайн, а затем просто сделать Ctr+C, Ctr+V. По идее статические сайты работают быстрее и более безопасны, но и правильно настроенный WordPress не тормозит и не ломается. При этом у WordPress *намного* больше шаблонов и плагинов на все случаи жизни, чем у какого-либо другого движка. В

общем и целом, вопрос о том, какие же сайты лучше — традиционные сайты на движке вроде WordPress, или же статические, для меня остается открытым.

Метки: Python, Блогинг, Сайтостроение.

Вы можете прислать свой комментарий мне на почту, или воспользоваться комментариями в Telegram-группе.

- ## Коротко о себе

  Меня зовут Александр, позывной любительского радио R2AUK. Здесь я пишу об интересующих меня вещах и временами — просто о жизни.

  Вы можете следить за обновлениями блога с помощью RSS и Telegram. Также я являюсь одним из ведущих подкаста DevZen и выкладываю видео на YouTube.

  Мой e-mail — *afiskon@gmail.com*. Если вы хотите мне написать, прошу предварительно ознакомиться с FAQ.

- Поиск

- ## Основные рубрики

  - Антенны
  - Беспроводная связь
  - C/C++
  - Go
  - Linux
  - PostgreSQL
  - Python
  - STM32
  - СУБД
  - Электроника