olOwOlo

主页 关

Hugo 从入门到会用

2017-08-15 · 约 4517 字 · 预计阅读 10 分钟 · 806 次阅读

为什么选择 Hugo？

- 快！世界上最快的静态网站生成工具！5秒生成6000个页面！
- 超详细的文档，~~虽然是英文的~~
- 活跃的社区
- 更加自由的内容组织方式
- 丰富的主题，目前主题数量已经超过 Hexo

Hugo 有哪些劣势？

- 你喜欢的某个 Jekyll 或者 Hexo 主题是无法直接用在 Hugo 上的┐(￣Д￣)┌

那么...

翻滚吧！奈亚子！

安装 Hugo

macOS

```
Code
1  brew install hugo
```

Windows

```
Code
1  # 前往 https://github.com/gohugoio/hugo/releases
2  # 下载 hugo_X.XX_Windows-64bit.zip
3  # 解压得到 hugo.exe
4  # 将 hugo.exe 所在目录添加至系统环境变量
5  # 安装完成！
```

验证安装是否成功

```
Code
1  hugo version
```

其他系统请参考官方文档——安装

建站

Code

```
hugo new site quickstart
```

添加一个主题

Code

```
cd quickstart;
git init;
git submodule add https://github.com/budparr/gohugo-theme-ananke.git themes/ananke;

# 编辑你的 config.toml 配置文件使用该主题
echo 'theme = "ananke"' >> config.toml
```

开始写作

Code

```
hugo new post/my-first-post.md
```

编辑你新建的markdown文件。现在，在启用drafts参数的条件下开启Hugo内置的服务器。

Code

```
hugo server -D
```

你的网站已经在http://localhost:1313/开启了。

基本使用

最常用的命令一定是在你的站点目录下使用 `hugo` 命令，该命令将会生成静态文件。~~（可以看到生成速的很快ᕦ(ò_óˇ)ᕤ）~~

Code

```
hugo
0 draft content
0 future content
99 pages created
0 paginator pages created
16 tags created
0 groups created
in 90 ms
```

这里节选出 `new` 与 `server` 这两个比较常用的命令和 `-D`, `-E`, `-F` 这三个参数的使用方法。更多的hugo以参考官方文档。

Code

```
使用方法:
  hugo
  hugo [flags]
```

```
  hugo [command]
  hugo [command] [flags]

节选的 command:
  new          为你的站点创建新的内容
  server       一个高性能的web服务器

节选的 flags:
  -D, --buildDrafts               包括被标记为draft的文章
  -E, --buildExpired              包括已过期的文章
  -F, --buildFuture               包括将在未来发布的文章

举几个栗子:
  hugo -D                         生成静态文件并包括draft为true的文章
  hugo new post/new-content.md    新建一篇文章
  hugo new site mysite            新建一个称为mysite的站点
  hugo server --buildExpired      启动服务器并包括已过期的文章
```

PS: hugo server 会自动监听你的原始文稿，你在编辑原始 `.md` 文件时的变化都会实时的反映在网站上，如果你不希望启用这个功能你可以使用 `hugo server --watch=false` 命令。

目录结构

在执行完 `hugo new site` 命令后你会得到一个包含以下文件的目录。

Code

```
.
├── archetypes
├── config.toml
├── content
├── data
├── layouts
├── static
└── themes
```

- **archetypes**: 储存 `.md` 的模板文件，类似于 `hexo` 的 `scaffolds`，该文件夹的优先级高于主题下的 `/archetypes` 文件夹
- **config.toml**: 配置文件
- **content**: 储存网站的所有内容，类似于 `hexo` 的 `source`
- **data**: 储存数据文件供模板调用，类似于 `hexo` 的 `source/_data`
- **layouts**: 储存 `.html` 模板，该文件夹的优先级高于主题下的 `/layouts` 文件夹
- **static**: 储存图片,css,js等静态文件，该目录下的文件会直接拷贝到 `/public`，该文件夹的优先级主题下的 `/static` 文件夹
- **themes**: 储存主题
- **public**: 执行 `hugo` 命令后，储存生成的静态文件

配置 Hugo

编辑你的 `config.toml` 文件以配置你的站点。以下节选了部分有趣的参数，完整的配置文件可以参考档。

TOML

```
+++
```

```
# 主机名 例如: http://spf13.com/
baseURL =                       ""
# 语言编码(中文: zh-CN)
languageCode =                  ""
# 默认的内容语言
defaultContentLanguage =        "zh-CN"

# 自动检测是否包含中文/日文/韩文，该参数会影响摘要和字数统计功能，建议设置为true
hasCJKLanguage =                false

# 若为 false，`Getting Started` 这样的英文标签将会被转换为 `getting-started`
preserveTaxonomyNames =         true

# 设置使用的主题名称 (默认储存在 /themes/THEMENAME/)
theme =

# 分页
paginate =                      10
paginatePath =                  "page"

# 启用 Emoji; see emoji-cheat-sheet.com
enableEmoji =                   false

# 创建robots.txt，建议设置为true
enableRobotsTXT =               false

# 定义文章访问路径，详细见下文"URL管理" See "content-management/urls/"
permalinks =                    ""
+++
```

除了预设的参数外，你也可以通过下面的方式自定义自己的参数。你可以在模板中获取到自定义的参数，常情况下你使用的主题都会要求你这么做，具体信息可以阅读相关主题的文档。

TOML

```
[params]
  author = "olOwOlo"
  github = "olOwOlo"
```

至此，你已经掌握了足够的知识，你只需要从主题列表中挑选一个你喜欢的主题，阅读主题的文档并按题所需的信息，便可以开始使用 `hugo` 来写作了。

阅读下面的内容可以帮你更好的了解到 `hugo` 如何管理你的内容，而你又该如何更好的与 `hugo` 协作。

> 以上是对Getting Started的总结，下面是对Content Management的总结

千反田同学很好奇！

组织content/文件夹

`hugo` 会将 `content/` 目录下的结构反映到生成的静态网站中，如下：

Code

```
.
└── content
    └── about
    |   └── _index.md  // <- https://example.com/about/
    ├── post
    |   ├── _index.md   // <- https://example.com/post/
    |   ├── firstpost.md   // <- https://example.com/post/firstpost/
    |   ├── happy
    |   |   └── ness.md  // <- https://example.com/post/happy/ness/
    |   └── secondpost.md  // <- https://example.com/post/secondpost/
    └── quote
        ├── first.md       // <- https://example.com/quote/first/
        └── second.md      // <- https://example.com/quote/second/
```

需要注意的是 `_index.md` 是一个比较特殊的文件，如果你只是需要一个about页面，你只需要 `hugo ne about.md` 即可。关于 `_index.md` 的相关信息可以参阅官方文档。

Hugo 中的路径分解

Code

```
                    section
                    ├--^--┤
                              url
                    ├-------------^------------┤

      baseURL             path        slug
├---------^---------┤ ├-------^------┤├----^-------┤
                   permalink
├-----------------------^------------------------┤
https://example.com/events/chicago/lollapalooza/
```

Hugo 中的类型

`content/` 下的内容可能不仅仅只有文章，还可能有照片甚至其他不同形式的内容。 `hugo` 通过 `type` 属标记不同形式的内容，根据不同的 `type` 值， `hugo` 会选择合适的模板来渲染内容。

默认情况下 `type` 被赋值为 `section` 。例如：使用 `hugo new posts/new-post.md` 命令得到的新文件会为 `posts` 类型。当然你也可以在下文提到的front matter中显式的将文章的 `type` 指定为其他值。

Front Matter

Hugo 支持TOML、YAML、JSON格式的Front Matter。

以下列出一份完整的YAML格式的Front Matter，除了必要的 `data` 与 `title` 参数外，你可以有选择性的其他参数。

YAML

```yaml
---
title: "文章标题"
description: "文章的描述信息"
tags: [ "标签1", "标签2", "标签3" ]
categories: [ "分类1", "分类2" ]
keywords: [ "Hugo", "keyword" ]
date: 2012-04-06
```

```yaml
lastmod: 2015-12-23
# CJKLanguage: Chinese, Japanese, Korean
isCJKLanguage: true

# 如果draft为true，除非使用 --buildDrafts 参数，否则不会发布文章
draft: false

# 设置文章的过期时间，如果是已过期的文章则不会发布，除非使用 --buildExpired 参数
expiryDate: 2020-01-01

# 设置文章的发布时间，如果是未来的时间则不会发布，除非使用 --buildFuture 参数
publishDate: 2020-01-01

# 排序你的文章
weight: 40

# 使用这两个参数将会重置permalink，默认使用文件名
url:
slug:

# 别名将通过重定向实现
aliases:
  - 别名1
  - /posts/my-original-url/
  - /2010/01/01/another-url.html

# type 与 layout 参数将会改变 Hugo 寻找该文章模板的顺序，将在下一节细述
type: review
layout: reviewarticle

# 三个比较复杂的参数
taxonomies:
linkTitle:
outputs:
# 实验性的参数
markup: "md"

# 你还可以自定义任何参数，这些参数可以在模板中使用
include_toc: true
show_comments: false
---
```

模板选择顺序

这是一篇 `content/posts/` 下的文章寻找模板的顺序（此时我们未在文章的Front Matter指定 `type` 与 `l` 属性）：

1. /layouts/UNSPECIFIED/UNSPECIFIED.html
2. /layouts/posts/UNSPECIFIED.html
3. /layouts/UNSPECIFIED/single.html
4. /layouts/posts/single.html
5. /layouts/_default/single.html
6. /themes/<THEME>/layouts/UNSPECIFIED/UNSPECIFIED.html
7. /themes/<THEME>/layouts/posts/UNSPECIFIED.html
8. /themes/<THEME>/layouts/UNSPECIFIED/single.html
9. /themes/<THEME>/layouts/posts/single.html
10. /themes/<THEME>/layouts/_default/single.html

如果我们在Front Matter中添加以下代码

```yaml
type: review
layout: reviewarticle
```

该文章现在的寻找模板顺序为：

1. /layouts/review/reviewarticle.html
2. /layouts/posts/reviewarticle.html
3. /layouts/review/single.html
4. /layouts/posts/single.html
5. /layouts/_default/single.html
6. /themes/<THEME>/layouts/review/reviewarticle.html
7. /themes/<THEME>/layouts/posts/reviewarticle.html
8. /themes/<THEME>/layouts/review/single.html
9. /themes/<THEME>/layouts/posts/single.html
10. /themes/<THEME>/layouts/_default/single.html

值得注意的是，**`/layouts/` 目录下模板优先级总是高于 `/themes/<THEME>/layouts/` 。同理，若根目在与 `/themes/` 文件夹下同名的文件夹，根目录下的文件优先级总是高于 `/themes/` 文件夹。**

因此，在我们只是需要别人提供的主题做一些小修改时，尤其是对于一些静态资源需要进行覆盖时，文件置于根目录的文件夹下而不是直接对主题进行修改，日后需要更新主题时就无需解决 git 冲突的了。

当然，如果是需要直接对 `/layouts/` 目录下的模板进行修改，还是建议新建一个 git 分支进行更改。

URL管理

正如前文所言，`hugo` 会将 `content/` 目录下的结构反映到生成的静态网站中，但 `config.toml` 中的 `permalinks` 参数允许你自由更改内容的URL。例如：你想从 `hexo` 迁移到 `hugo`，你可以将 `permalin` 为下面这种形式以适应之前的URL。

```toml
[permalinks]
  post = "/:year/:month/:title/"
```

上面的配置将改变 `content/post/` 文件夹下所有文章的URL。

举个栗子，`content/post/sample-entry.md` 的URL将从默认的 `https://example.com/post/sample-e` 改变为 `https://example.com/2013/11/sample-entry/` 。

所有可用的属性如下：

```toml
/:monthname/:day/:weekday/:weekdayname/:yearday/:section/:title/:slug/:filename/
```

内容摘要

Hugo会自动提取文章的前70个字符作为摘要。（注意：该功能在中文环境下需要在`config.toml`中添加`hasCJKLanguage = true`才能发挥更好的效果。）

当然你也可以在文章内使用`<!--more-->`针对文章手动进行摘要提取，在`<!--more-->`之前出现的内容作为摘要使用，且能够保持渲染后的结构而不是纯文字版本。

Shortcodes

Shortcodes帮助你在编写markdown时快捷的插入HTML代码，功能上类似于Hexo的标签插件。

Go HTML Template

```
{{< ref "blog/post.md" >}} => https://example.com/blog/post/
{{< ref "post.md#tldr" >}} => https://example.com/blog/post/#tldr:caffebad
{{< relref "post.md" >}} => /blog/post/
{{< relref "blog/post.md#tldr" >}} => /blog/post/#tldr:caffebad
{{< ref "#tldr" >}} => #tldr:badcaffe
{{< relref "#tldr" >}} => #tldr:badcaffe
```

上述代码通过内置的`rel`与`relref`帮助你快速引用站点内的其他文章。

注意: 如果你的`content/`目录下有多个同名的文件，引用该文章必须使用`blog/post.md`这样的相对路径而不是只提供`post.md`这样的文件名。

`hugo`还内置了`instagram`、`tweet`、`youtube`等Shortcodes，可以阅读官方文档了解更多信息，你的主题可能也会提供Shortcodes，当然你也可以定制你自己的Shortcodes。

分类系统

默认情况下即`tags`与`categories`，通常来说这已经足够我们使用了，但你也可以在`config.toml`文件中加下面的代码来添加更多的分类。

TOML

```
[taxonomies]
  tag = "tags"
  category = "categories"
  series = "series"
```

维包子什么都知道!

Custom Output Formats

`hugo`能够输出多种不同格式的内容，这里简单的举个**如何同时输出markdown文件**的栗子:

在你的`config.toml`文件下添加:

TOML

```
[mediaTypes]
  [mediaTypes."text/plain"]
    suffix = "md"

[outputFormats.MarkDown]
  mediaType = "text/plain"
  isPlainText = true
  isHTML = false
```

为 MarkDown 新建一个 single.md 模板，写入以下代码，并放置在 /layouts/_default/ 文件夹下

Go HTML Template

```
{{ .RawContent }}
```

针对所有内容，在 `config.toml` 文件下添加：

TOML

```
[outputs]
  home = ["HTML", "RSS"]
  page = ["HTML", "MarkDown"]
  section = ["HTML", "RSS"]
  taxonomy = ["HTML", "RSS"]
  taxonomyTerm = ["HTML"]
```

针对某篇文章，在该文章的Front Matter中添加：

YAML

```
outputs:
  - html
  - markdown
```

在模板中的合适位置添加代码链接到你的 `.md` 文件：

Go HTML Template

```
{{ with .OutputFormats.Get "markdown" -}}
<a href="{{ .Permalink }}">查看本文 Markdown 版本 »</a>
{{- end }}
```

扩展阅读

- **官方文档**

文章作者： olOwOlo

上次更新： 2018-02-05

#Hugo #Getting Started

❮ Google 涂鸦 - Kids Coding - 兔子吃萝卜解法