Hexo中文版

极客学院出版

前言

Hexo 是一个快速、简洁且高效的博客框架。Hexo 使用 Markdown（或其他渲染引擎）解析文章，在几秒内，即可利用靓丽的主题生成静态网页。

欢迎使用 Hexo，本文档将帮助您快速上手。如果您在使用过程中遇到问题，请查看 问题解答 (#9525aaf3a3ec144eb0263e71113a03e0) 中的解答，或者在 GitHub (https://github.com/hexojs/hexo/issues) 、Google Group (https://groups.google.com/forum/#!forum/hexo) 上提问。

安装

安装 Hexo 只需几分钟时间，若您在安装过程中遇到问题或无法找到解决方式，请提交问题 (https://github.com/hexojs/hexo/issues) ，我会尽力解决您的问题。

安装前提

安装 Hexo 相当简单。然而在安装前，您必须检查电脑中是否已安装下列应用程序：

- Node.js (https://nodejs.org/)
- Git (http://git-scm.com/)

如果您的电脑中已经安装上述必备程序，那么恭喜您！接下来只需要使用 npm 即可完成 Hexo 的安装。

```
$ npm install -g hexo-cli
```

如果您的电脑中尚未安装所需要的程序，请根据以下安装指示完成安装。

> Mac 用户
>
> 您在编译时可能会遇到问题，请先到 App Store 安装 Xcode，Xcode 完成后，启动并进入 Preferences -> Download -> Command Line Tools -> Install 安装命令行工具。

安装 Git

- Windows：下载并安装 msysgit (https://code.google.com/hosting/moved?project=msysgit) .
- Mac：使用 Homebrew (http://brew.sh/) , MacPorts (http://www.macports.org/) 或下载 安装程序 (https://code.google.com/p/git-osx-installer/) 安装。
- Linux (Ubuntu, Debian)：`sudo apt-get install git-core`
- Linux (Fedora, Red Hat, CentOS)：`sudo yum install git-core`

安装 Node.js

安装 Node.js 的最佳方式是使用 nvm (https://github.com/creationix/nvm) 。

cURL:

```
$ curl https://raw.github.com/creationix/nvm/master/install.sh | sh
```

Wget:

```
$ wget -qO- https://raw.github.com/creationix/nvm/master/install.sh | sh
```

安装完成后，重启终端并执行下列命令即可安装 Node.js。

```
$ nvm install 0.10
```

或者您也可以下载 应用程序 (https://nodejs.org/) 来安装。

安装 Hexo

所有必备的应用程序安装完成后，即可使用 npm 安装 Hexo。

```
$ npm install -g hexo-cli
```

> 致谢
>
> 内容撰写：http://hexo.io/zh-cn/

更新日期	更新内容
2015-05-21	Hexo 中文版

目录

1

开始使用

建站

安装 Hexo 完成后，请执行下列命令，Hexo 将会在指定文件夹中新建所需要的文件。

```
$ hexo init
$ cd
$ npm install
```

新建完成后，指定文件夹的目录如下：

```
.
├── _config.yml
├── package.json
├── scaffolds
├── scripts
├── source
|   ├── _drafts
|   └── _posts
└── themes
```

_config.yml

网站的 配置 (http://hexo.io/zh-cn/docs/configuration.html) 信息，您可以在此配置大部分的参数。

package.json

应用程序的信息。EJS (http://www.embeddedjs.com/) , Stylus (http://learnboost.github.io/stylus/) 和 Markdown (http://daringfireball.net/projects/markdown/) renderer 已默认安装，您可以自由移除。

package.json

```
{
  "name": "hexo-site",
  "version": "",
  "private": true,
  "hexo": {
    "version": ""
  },
  "dependencies": {
    "hexo-renderer-ejs": "*",
    "hexo-renderer-stylus": "*",
```

```
    "hexo-renderer-marked": "*"
  }
}
```

scaffolds

模版 (页 18) 文件夹。当您新建文章时，Hexo 会根据 scaffold 来建立文件。

scripts

脚本 (页 63) 文件夹。脚本是扩展 Hexo 最简易的方式，在此文件夹内的 JavaScript 文件会被自动执行。

source

资源文件夹是存放用户资源的地方。除 `_posts` 文件夹之外，开头命名为 `_` (下划线)的文件 / 文件夹和隐藏的文件将会被忽略。Markdown 和 HTML 文件会被解析并放到 `public` 文件夹，而其他文件会被拷贝过去。

themes

主题 (页 39) 文件夹。Hexo 会根据主题来生成静态页面。

配置

您可以在 `_config.yml` 中修改大部份的配置。

网站

参数	描述
title	网站标题
subtitle	网站副标题
description	网站描述
author	您的名字
language	网站使用的语言
timezone	网站时区。Hexo 预设使用您电脑的时区。时区列表 (https://en.wikipedia.org/wiki/List_of_tz_database_time_zones)

网址

参数	描述	默认值
url	网址	
root	网站根目录	
permalink	文章的 永久链接 (页 0) 格式	:year/:month/:day/:title/
permalink_default	永久链接中各部分的默认值	

> 网站存放在子目录
>
> 如果您的网站存放在子目录中，例如 `http://yoursite.com/blog` ，则请将您的 `url` 设为 `http://yoursite.com/blog` 并把 `root` 设为 `/blog/` 。

目录

参数	描述	默认值
source_dir	资源文件夹，这个文件夹用来存放内容。	source
public_dir	公共文件夹，这个文件夹用于存放生成的站点文件。	public

参数	描述	默认值
tag_dir	标签文件夹	tags
archive_dir	归档文件夹	archives
category_dir	分类文件夹	categories
code_dir	Include code 文件夹	downloads/code
i18n_dir	国际化（i18n）文件夹	:lang
skip_render	跳过指定文件的渲染，您可使用 glob 来配置路径。	

文章

参数	描述	默认值
new_post_name	新文章的文件名称	:title.md
default_layout	预设布局	post
auto_spacing	在中文和英文之间加入空格	false
titlecase	把标题转换为 title case	false
external_link	在新标签中打开链接	true
filename_case	把文件名称转换为 (1) 小写或 (2) 大写	0
render_drafts	显示草稿	false
post_asset_folder	启动 Asset 文件夹 (页 27)	false
relative_link	把链接改为与根目录的相对位址	false
future	显示未来的文章	true
highlight	代码块的设置	

分类 & 标签

参数	描述	默认值
default_category	默认分类	uncategorized
category_map	分类别名	
tag_map	标签别名	

日期 / 时间格式

Hexo 使用 Moment.js (http://momentjs.com/) 来解析和显示时间。

参数	描述	默认值
date_format	日期格式	MMM D YYYY
time_format	时间格式	H:mm:ss

分页

参数	描述	默认值
per_page	每页显示的文章量 (0 = 关闭分页功能)	10
pagination_dir	分页目录	page

扩展

参数	描述
theme	当前主题名称
deploy	部署

指令

init

```
$ hexo init [folder]
```

新建一个网站。如果没有设置 folder ， Hexo 默认在目前的文件夹建立网站。

new

```
$ hexo new [layout] <title>
```

新建一篇文章。如果没有设置 layout 的话，默认使用 _config.yml 中的 default_layout 参数代替。如果标题包含空格的话，请使用引号括起来。

generate

```
$ hexo generate
```

生成静态文件。

选项	描述
-d , -- deploy	文件生成后立即部署网站
-w, --watch	监视文件变动

publish

```
$ hexo publish [layout] <filename>
```

发表草稿。

server

```
$ hexo server
```

启动服务器。

选项	描述
-p , --port	重设端口
-s, --static	只使用静态文件
-l , --log	启动日记记录，或覆盖记录格式

deploy

```
$ hexo deploy
```

部署网站。

参数	描述
-g , --generate	部署网站前，需要预先生成静态文件

render

```
$ hexo render <file> ...
```

渲染文件。

参数	描述
-o , --output	设置输出路径

migrate

```
$ hexo migrate <type>
```

从其他系统 迁移内容 ()

clean

```
$ hexo clean
```

清除缓存文件 (db.json) 和已生成的静态文件 (public)。

list

```
$ hexo list <type>
```

列出网站资料。

version

```
$ hexo version
```

显示 Hexo 版本。

选项

安全模式

```
$ hexo --safe
```

在安全模式下，不会载入插件和脚本。当您在安装新插件遭遇问题时，可以尝试以安全模式重新执行。

调试模式

```
$ hexo --debug
```

在终端中显示调试信息并记录到 debug.log。当您碰到问题时，可以尝试用调试模式重新执行一次，并 提交调试信息到 GitHub ()

简洁模式

```
$ hexo --silent
```

隐藏终端信息。

自定义配置文件的路径

```
$ hexo --config custom.yml
```

自定义配置文件的路径，执行后将不再使用 _config.yml。

显示草稿

```
$ hexo --draft
```

显示 source/_drafts 文件夹中的草稿文章。

自定义 CWD

```
$ hexo --cwd /path/to/cwd
```

自定义当前工作目录（Current working directory）的路径。

迁移

RSS

首先，安装 `hexo-migrator-rss` 插件。

```
$ npm install hexo-migrator-rss --save
```

插件安装完成后，执行下列命令，从 RSS 迁移所有文章。`source` 可以是文件路径或网址。

```
$ hexo migrate rss
```

Jekyll

把 `_posts` 文件夹内的所有文件复制到 `source/_posts` 文件夹，并在 `_config.yml` 中修改 `new_post_name` 参数。

```
new_post_name: :year-:month-:day-:title.md
```

Octopress

把 Octopress `source/_posts` 文件夹内的所有文件转移到 Hexo 的 `source/_posts` 文件夹，并修改 `_config.yml` 中的 `new_post_name` 参数。

```
new_post_name: :year-:month-:day-:title.md
```

WordPress

首先，安装 `hexo-migrator-wordpress` 插件。

```
$ npm install hexo-migrator-wordpress --save
```

在 WordPress 仪表盘中导出数据("Tools" → "Export" → "WordPress")（详情参考[WP支持页面][1]）。

插件安装完成后，执行下列命令来迁移所有文章。`source` 可以是 WordPress 导出的文件路径或网址。

```
$ hexo migrate wordpress
```

Joomla

首先，安装 `hexo-migrator-joomla` 插件。

```
$ npm install hexo-migrator-joomla --save
```

使用 J2XML (http://extensions.joomla.org/extensions/extension/migration-a-conversion/data-import-a-export/j2xml) 组件导出 Joomla 文章。

插件安装完成后，执行下列命令来迁移所有文章。`source` 可以是 Joomla 导出的文件路径或网址。

```
$ hexo migrate joomla
```

2

基本操作

写作

接下来，我们要在网站中建立第一篇文章，您可以直接从现有的示例文章「Hello World」改写，但我们更建议您学习 new 指令。

```
$ hexo new [layout] <title>
```

您可以在命令中指定文章的布局（layout），默认为 `post`，可以通过修改 `_config.yml` 中的 `default_layout` 参数来指定默认布局。

布局（Layout）

Hexo 有三种默认布局：`post`、`page` 和 `draft`，它们分别对应不同的路径，而您自定义的其他布局和 `post` 相同，都将储存到 `source/_posts` 文件夹。

布局	路径
post	source/_posts
page	source
draft	source/_drafts

文件名称

Hexo 默认以标题做为文件名称，但您可编辑 new_post_name 参数来改变默认的文件名称，举例来说，设为 `:year-:month-:day-:title.md` 可让您更方便的通过日期来管理文章。

变量	描述
:title	标题
:year	建立的年份（4 位数）
:month	建立的月份（2 位数）
:i_month	建立的月份（去掉开头的零）
:day	建立的日期（2 位数）
:i_day	建立的日期（去掉开头的零）

草稿

刚刚提到了 Hexo 的一种特殊布局：draft，这种布局在建立时会被保存到 source/_drafts 文件夹，您可通过 publish 命令将草稿移动到 source/_posts 文件夹，该命令的使用方式与 new 十分类似，您也可在命令中指定 layout 来指定布局。

```
$ hexo publish [layout] <title>
```

草稿默认不会显示在页面中，您可在执行时加上 `--draft` 参数，或是把 `render_drafts` 参数设为 `true` 来预览草稿。

模版（Scaffold）

在新建文章时，Hexo 会根据 scaffolds 文件夹内相对应的文件来建立文件，例如：

```
$ hexo new photo "My Gallery"
```

在执行这行指令时，`Hexo` 会尝试在 `scaffolds` 文件夹中寻找 photo.md，并根据其内容建立文章，以下是您可以在模版中使用的变量：

变量	描述
`layout`	布局
`title`	标题
`date`	文件建立日期

Front-matter

Front-matter 是文件最上方以 `\---` 分隔的区域，用于指定个别文件的变量，举例来说：

```
title: Hello World
date: 2013/7/13 20:46:25
---
```

以下是预先定义的参数，您可在模板中使用这些参数值并加以利用。

参数	描述	默认值
layout	布局	
title	标题	
date	建立日期	文件建立日期
updated	更新日期	文件更新日期
comments	开启文章的评论功能	true
tags	标签（不适用于分页）	
categories	分类（不适用于分页）	
permalink	覆盖文章网址	

分类和标签

只有文章支持分类和标签，您可以在 Front-matter 中设置。在其他系统中，分类和标签听起来很接近，但是在 Hexo 中两者有着明显的差别：分类具有顺序性和层次性，也就是说 `Foo, Bar` 不等于 `Bar, Foo`；而标签没有顺序和层次。

```
categories:
- Diary
tags:
- PS3
- Games
```

JSON Front-matter

除了 YAML 外，你也可以使用 JSON 来编写 Front-matter，只要将 `\---` 代换成 `;;;` 即可。

```
"title": "Hello World",
"date": "2013/7/13 20:46:25"
;;;
```

标签插件（Tag Plugins）

标签插件和 Front-matter 中的标签不同，它们是用于在文章中快速插入特定内容的插件。

Block Quote

在文章中插入引言，可包含作者、来源和标题。

捷径： quote

```
{% blockquote [author[, source]] [link] [source_link_title] %}
content
{% endblockquote %}
```

没有提供参数，则只输出普通的 blockquote

```
{% blockquote %}
Lo
{% endblockquote %}
```

> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque hendrerit lacus ut purus iaculis feugiat. Sed nec tempor elit, quis aliquam neque. Curabitur sed diam eget dolor fermentum semper at eu lorem.

引用书上的句子

```
{% blockquote David Levithan, Wide Awake %}
Do not just seek happiness for yourself. Seek happiness for all. Through kindness. Through mercy.
{% endblockquote %}
```

> Do not just seek happiness for yourself. Seek happiness for all. Through kindness. Through mercy.
>
> David LevithanWide Awake

引用 Twitter

```
{% blockquote @DevDocs https:
NEW: DevDocs now comes with syntax highlighting. http:
{% endblockquote %}
```

> NEW: DevDocs now comes with syntax highlighting. http://devdocs.io

> **@DevDocs**twitter.com/devdocs/status/356095192085962752 (https://twitter.com/devdocs/status/356095192085962752)

引用网路上的文章

```
{% blockquote Seth Godin http://sethgodin.typepad.com/seths_blog/2009/07/welcome-to-island-marketing.html Welc
Every interaction is both precious and an opportunity to delight.
{% endblockquote %}
```

> Every interaction is both precious and an opportunity to delight.
>
> **Seth Godin**Welcome to Island Marketing (http://sethgodin.typepad.com/seths_blog/2009/07/welcome-to-island-marketing.html)

Code Block

在文章中插入代码。

快捷方式： code

```
{% codeblock [title] [lang:language] [url] [link text] %}
code snippet
{% endcodeblock %}
```

普通的代码块

```
{% codeblock %}
alert('Hello World!');
{% endcodeblock %}
```

```
alert('Hello World!');
```

指定语言

```
{% codeblock lang:objc %}
[rectangle setX: 10 y: 10 width: 20 height: 20];
{% endcodeblock %}
```

```
[rectangle setX: 10 y: 10 width: 20 height: 20];
```

附加说明

```
{% codeblock Array.map %}
array.map(callback[, thisArg])
{% endcodeblock %}
```

Array.map

```
array.map(callback[, thisArg])
```

附加说明和网址

```
{% codeblock _.compact http://underscorejs.org/
_.compact([0, 1, false, 2, '', 3]);
=> [1, 2, 3]
{% endcodeblock %}
```

_.compact[Underscore.js][3]

```
_.compact([0, 1, false, 2, '', 3]);
=> [1, 2, 3]
```

Backtick Code Block

另一种形式的代码块。

```
<!-- 10-->
```

Pull Quote

在文章中插入 Pull quote。

```
{% pullquote [class] %}
content
{% endpullquote %}
```

jsFiddle

在文章中嵌入 jsFiddle。

```
{% jsfiddle shorttag [tabs] [skin] [width] [height] %}
```

Gist

在文章中嵌入 Gist。

```
{% gist gist_id [filename] %}
```

iframe

在文章中插入 iframe。

```
{% iframe url [width] [height] %}
```

Image

在文章中插入指定大小的图片。

```
{% img [class names] /path/to/image [width] [height] [title text [alt text]] %}
```

Link

在文章中插入链接，并自动给外部链接添加 `target="_blank"` 属性。

```
{% link text url [external] [title] %}
```

Include Code

插入 `source` 文件夹内的代码文件。

```
{% include_code [title] [lang:language] path/to/file %}
```

Youtube

在文章中插入 Youtube 视频。

```
{% youtube video_id %}
```

Vimeo

在文章中插入 Vimeo 视频。

```
{% vimeo video_id %}
```

引用文章

引用其他文章的链接。

```
{% post_path slug %}
{% post_link slug [title] %}
```

引用资源

引用文章的资源。

```
{% asset_path slug %}
{% asset_img slug [title] %}
{% asset_link slug [title] %}
```

Raw

如果您想在文章中插入 Swig 标签，可以尝试使用 Raw 标签，以免发生解析异常。

```
{% raw %}
content
{% endraw %}
```

资源文件夹

资源（Asset）代表 `source` 文件夹中除了文章以外的所有文件，例如图片、CSS、JS 文件等。Hexo 提供了一种更方便管理 Asset 的设定：`post_asset_folder` 。

```
post_asset_folder: true
```

当您设置 `post_asset_folder` 参数后，在建立文件时，Hexo 会自动建立一个与文章同名的文件夹，您可以把与该文章相关的所有资源都放到那个文件夹，如此一来，您便可以更方便的使用资源。

标签插件

Hexo 3.0 新增了几个插件，让您更方便的在文章内引用资源。

```
{% asset_path slug %}
{% asset_img slug [title] %}
{% asset_link slug [title] %}
```

数据文件

有时您可能需要在主题中使用某些资料，而这些资料并不在文章内，并且是需要重复使用的，那么您可以考虑使用 Hexo 3.0 新增的「数据文件」功能。此功能会载入 `source/_data` 内的 YAML 或 JSON 文件，如此一来您便能在网站中复用这些文件了。

举例来说，在 `source/_data` 文件夹中新建 `menu.yml` 文件：

```
Home: /
Gallery: /gallery/
Archives: /archives/
```

您就能在模板中使用这些资料：

```
{% for link in site.data.menu %}
  {{ loop.key }}
{% endfor %}
```

服务器

hexo-server (https://github.com/hexojs/hexo-server)

Hexo 3.0 把服务器独立成了个别模块，您必须先安装 hexo-server (https://github.com/hexojs/hexo-server) 才能使用。

```
$ npm install hexo-server --save
```

安装完成后，输入以下命令以启动服务器，您的网站会在 `http://localhost:4000` 下启动。在服务器启动期间，Hexo 会监视文件变动并自动更新，您无须重启服务器。

```
$ hexo server
```

如果您想要更改端口，或是在执行时遇到了 `EADDRINUSE` 错误，可以在执行时使用 `-p` 选项指定其他端口，如下：

```
$ hexo server -p 5000
```

静态模式

在静态模式下，服务器只处理 `public` 文件夹内的文件，而不会处理文件变动，在执行时，您应该先自行执行 `hexo generate`，此模式通常用于生产环境（production mode）下。

```
$ hexo server -s
```

自定义 IP

服务器默认运行在 `0.0.0.0`，您可以覆盖默认的 IP 设置，如下：

```
$ hexo server -i 192.168.1.1
```

Pow

Pow (http://pow.cx/) 是一个 Mac 系统上的零配置 Rack 服务器，它也可以作为一个简单易用的静态文件服务器来使用。

安装

```
$ curl get.pow.cx | sh
```

设置

在 `~/.pow` 文件夹建立链接（symlink）。

```
$ cd ~/.pow
$ ln -s /path/to/myapp
```

您的网站将会在 `http://myapp.dev` 下运行，网址根据链接名称而定。

Forever / PM2

为了让 Hexo 服务保持链接，你可以使用 Forever (https://github.com/foreverjs/forever) 或 PM2 (https://github.com/Unitech/pm2)

Hexo 从 2.5 版本开始，就可以运行在编程模式下，所以你可以在 JavaScript 中调用 Hexo，而不是使用 CLI。

1. 在你的站点文件夹中安装 Hexo。

```
$ npm install hexo --save
```

2. 新建一个 JavaScript 文件并编写以下代码。

app.js

```
require('hexo').init({command: 'server'});
```

3. 使用刚刚创建的 [Forever][3] 或 [PM2][4] 运行这个 JavaScript 文件。

PM2 的一个 [已知问题] 是，当停止运行脚本后，除非中断 PM2，否则端口不能自动释放。你必须在 fork 模式下运行脚本。

```
$ forever start app.js
$ pm2 start app.js -x
```

生成文件

使用 Hexo 生成静态文件快速而且简单。

```
$ hexo generate
```

监视文件变动

Hexo 能够监视文件变动并立即重新生成静态文件，在生成时会比对文件的 SHA1 checksum，只有变动的文件才会写入。

```
$ hexo generate --watch
```

完成后部署

您可执行下列的其中一个命令，让 Hexo 在生成完毕后自动部署网站，两个命令的作用是相同的。

```
$ hexo generate --deploy
$ hexo deploy --generate
```

部署

Hexo 提供了快速方便的一键部署功能，让您只需一条命令就能将网站部署到服务器上。

```
$ hexo deploy
```

在开始之前，您必须先在 `_config.yml` 中修改参数，一个正确的部署配置中至少要有 `type` 参数，例如：

```
deploy:
  type: git
```

您可同时使用多个 deployer，Hexo 会依照顺序执行每个 deployer。

```
deploy:
- type: git
  repo:
- type: heroku
  repo:
```

Git

安装 hexo-deployer-git (https://github.com/hexojs/hexo-deployer-git)

```
$ npm install hexo-deployer-git --save
```

修改配置。

```
deploy:
  type: git
  repo:
  branch: [branch]
  message: [message]
```

参数	描述
`repo`	库（Repository）地址
`branch`	分支名称。如果您使用的是 GitHub 或 GitCafe 的话，程序会尝试自动检测。
`message`	自定提交信息 (默认为 `Site updated: {{ now("YYYY-MM-DD HH:mm:ss") }}`)

Heroku

安装 hexo-deployer-heroku (https://github.com/hexojs/hexo-deployer-heroku)

```
$ npm install hexo-deployer-heroku --save
```

修改配置。

```
deploy:
  type: heroku
  repo:
  message: [message]
```

参数	描述
repo	Heroku 库（Repository）地址
message	自定提交信息 (默认为 Site updated: {{ now("YYYY-MM-DD HH:mm:ss") }})

Rsync

安装 hexo-deployer-rsync (https://github.com/hexojs/hexo-deployer-rsync)

```
$ npm install hexo-deployer-rsync --save
```

修改配置。

```
deploy:
  type: rsync
  host:
  user:
  root:
  port: [port]
  delete: [true|false]
  verbose: [true|false]
  ignore_errors: [true|false]
```

参数	描述	默认值
host	远程主机的地址	
user	使用者名称	
root	远程主机的根目录	
port	端口	22

参数	描述	默认值
delete	删除远程主机上的旧文件	true
verbose	显示调试信息	true
ignore_errors	忽略错误	false

OpenShift

安装 hexo-deployer-openshift (https://github.com/hexojs/hexo-deployer-openshift)

```
$ npm install hexo-deployer-openshift --save
```

修改配置。

```
deploy:
  type: openshift
  repo:
  message: [message]
```

参数	描述
repo	OpenShift 库 (Repository) 地址
message	自定提交信息 (默认为 Site updated: {{ now("YYYY-MM-DD HH:mm:ss") }})

FTPSync

安装 hexo-deployer-ftpsync (https://github.com/hexojs/hexo-deployer-ftpsync)

```
$ npm install hexo-deployer-ftpsync --save
```

修改配置。

```
deploy:
  type: ftpsync
  host:
  user:
  pass:
  remote: [remote]
  port: [port]
  ignore: [ignore]
  connections: [connections]
  verbose: [true|false]
```

参数	描述	默认值
host	远程主机的地址	
user	使用者名称	
pass	密码	
remote	远程主机的根目录	/
port	端口	21
ignore	忽略的文件或目录	
connections	使用的连接数	1
verbose	显示调试信息	false

其他方法

Hexo 生成的所有文件都放在 public 文件夹中，您可以将它们复制到您喜欢的地方。

3
自定义

永久链接(Permalinks)

您可以在配置中调整网站的永久链接。

变量

除了下列变量外，您还可使用 Front-matter 中的所有属性。

变量	描述
:year	文章的发表年份（4 位数）
:month	文章的发表月份（2 位数）
:i_month	文章的发表月份（去掉开头的零）
:day	文章的发表日期 (2 位数)
:i_day	文章的发表日期（去掉开头的零）
:title	文件名称
:id	文章 ID
:category	分类。如果文章没有分类，则是 default_category 配置信息。

您可在 permalink_defaults 参数下调整永久链接中各变量的默认值：

```
permalink_defaults:
  lang: en
```

示例

假设 source/_posts 文件夹中有个 hello-world.md，包含以下内容：

```
title: Hello World
date: 2013-07-14 17:01:34
categories:
- foo
- bar
```

参数	结果
:year/:month/:day/:title/	2013/07/14/hello-world
:year-:month-:day-:title.html	2013-07-14-hello-world.html
:category/:title	foo/bar/hello-world

多语种支持

若要建立一个多语种的网站，您可修改 `new_post_name` 和 `permalink` 参数，如下：

```
new_post_name: :lang/:title.md
permalink: :lang/:title/
```

当您建立新文章时，文章会被储存到：

```
$ hexo new "Hello World" --lang tw
 # => source/_posts/tw/Hello-World.md
```

而网址会是：

```
http://localhost:4000/tw/hello-world/
```

主题

创建 Hexo 主题非常容易，您只要在 `themes` 文件夹内，新增一个任意名称的文件夹，并修改 `_config.yml` 内的 `theme` 设定，即可切换主题。一个主题可能会有以下的结构：

```
.
├── _config.yml
├── languages
├── layout
├── scripts
└── source
```

_config.yml

主题的配置文件。修改时会自动更新，无需重启服务器。

languages

语言文件夹。请参见 本地化 (i18n) (页 0)

layout

布局文件夹。用于存放主题的模板文件，决定了网站内容的呈现方式，Hexo 内建 Swig (http://paularmstrong.github.io/swig/) 模板引擎，您可以另外安装插件来获得 EJS (https://github.com/hexojs/hexo-renderer-ejs) 、Haml (https://github.com/hexojs/hexo-renderer-haml) 或 Jade (https://github.com/hexojs/hexo-renderer-jade) 支持，Hexo 根据模板文件的扩展名来决定所使用的模板引擎，例如：

```
EJS: layout.ejs
Swig: layout.swig
```

您可参考 模板 (页 41) 以获得更多信息。

scripts

脚本文件夹。在启动时，Hexo 会载入此文件夹内的 JavaScript 文件，请参见 插件 (页 63) 以获得更多信息。

source

资源文件夹，除了模板以外的 Asset，例如 CSS、JavaScript 文件等，都应该放在这个文件夹中。文件或文件夹开头名称为 `_`（下划线线）或隐藏的文件会被忽略。

如果文件可以被渲染的话，会经过解析然后储存到 `public` 文件夹，否则会直接拷贝到 `public` 文件夹。

发布

当您完成主题后，可以考虑将它发布到 主题列表 (http://hexo.io/themes/)，让更多人能够使用您的主题。在发布前建议先进行 主题单元测试 (https://github.com/hexojs/hexo-theme-unit-test)，确保每一项功能都能正常使用。发布主题的步骤和更新文件 非常类似。

1. Fork hexojs/site (https://github.com/hexojs/site)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

```
$ git clone https://github.com//site.git
$ cd site
$ npm install
```

1. 编辑 `source/_data/themes.yml`，在文件中新增您的主题，例如：

```
- name: landscape
  description: A brand new default theme for Hexo.
  link: https://github.com/hexojs/hexo-theme-landscape
  preview: http://hexo.io/hexo-theme-landscape
  tags:
    - official
    - responsive
    - widget
    - two_column
    - one_column
```

1. 在 `source/themes/screenshots` 新增同名的截图档案，图片必须为 800x500 的 PNG 文件。
2. 推送（push）分支。
3. 建立一个新的合并申请（pull request）。

模版

模板决定了网站内容的呈现方式，每个主题至少都应包含一个 `index` 模板，以下是各页面相对应的模板名称：

模板	用途	回调
index	首页	
post	文章	index
page	分页	index
archive	归档	index
category	分类归档	archive
tag	标签归档	archive

布局（Layout）

如果页面结构类似，例如两个模板都有页首（Header）和页脚（Footer），您可考虑通过「布局」让两个模板共享相同的结构。一个布局文件必须要能显示 `body` 变量的内容，如此一来模板的内容才会被显示，举例来说：

index.ejs

```
index
```

layout.ejs

```
<!DOCTYPE html>
<html>
  <body><%- body %></body>
</html>
```

生成：

```
<!DOCTYPE html>
<html>
  <body>index</body>
</html>
```

每个模板都默认使用 `layout` 布局，您可在 front-matter 指定其他布局，或是设为 `false` 来关闭布局功能，您甚至可在布局中再使用其他布局来建立嵌套布局。

局部模版（Partial）

局部模板让您在不同模板之间共享相同的组件，例如页首（Header）、页脚（Footer）或侧边栏（Sidebar）等，可利用局部模板功能分割为个别文件，让维护更加便利。举例来说：

partial/header.ejs

```
<h1 id="logo"><%= config.title %></h1>
```

index.ejs

```
<%- partial('partial/header') %>
<div id="content">Home page</div>
```

生成：

```
<h1 id="logo">My Site</h1>
<div id="content">Home page</div>
```

局部变量

您可以在局部模板中指定局部变量并使用。

partial/header.ejs

```
<h1 id="logo"><%= title></h1>
```

index.ejs

```
<%- partial('partial/header', {title: 'Hello World'}) %>
<div id="content">Home page</div>
```

生成：

```
<h1 id="logo">Hello World</h1>
<div id="content">Home page</div>
```

最佳化

如果您的主题太过于复杂，或是需要生成的文件量太过于庞大，可能会大幅降低性能，除了简化主题外，您可以考虑 Hexo 2.7 新增的局部缓存（Fragment Caching）功能。

本功能借鉴于 Ruby on Rails (http://guides.rubyonrails.org/caching_with_rails.html#fragment-caching)，它储存局部内容，下次便能直接使用缓存内容，可以减少文件夹查询并使生成速度更快。

它可用于页首、页脚、侧边栏等文件不常变动的位置，举例来说：

```
<%- fragment_cache('header', function(){
  return '';
});
```

如果您使用局部模板的话，可以更简单：

```
<%- partial('header', {}, {cache: true});
```

但是，如果您开启了 `relative_link` 参数的话，请勿使用局部缓存功能，因为相对链接在每个页面可能不同。

变量 | Hexo

全局变量

变量	描述
site	网站变量
page	针对该页面的内容以及 front-matter 所设定的变量。
config	网站配置
theme	主题配置。继承自网站配置。
_ (单下划线)	Lodash (https://lodash.com/) 函数库
path	当前页面的路径（不含根路径）
url	当前页面的完整网址
env	环境变量

网站变量

变量	描述
site.posts	所有文章
site.pages	所有分页
site.categories	所有分类
site.tags	所有标签

页面变量

文章（post, page, …）

变量	描述
page.title	文章标题
page.date	文章建立日期（Moment.js (http://momentjs.com/) 物件）
page.updated	文章更新日期（Moment.js (http://momentjs.com/) 物件）
page.categories	文章分类
page.tags	文章标签
page.comments	留言是否开启
page.layout	布局名称
page.content	文章的完整内容

变量	描述
page.excerpt	文章摘要
page.more	除了文章摘要的其余内容
page.source	文章原始路径
page.full_source	文章的完整原始路径
page.path	文章网址（不含根路径）。我们通常在主题中使用 url_for(page.path) 。
page.permalink	文章的完整网址
page.prev	上一篇文章。如果此为第一篇文章则为 null 。
page.next	下一篇文章。如果此为最后一篇文章则为 null 。
page.raw	文章的原始内容
page.photos	文章的照片（用于相簿）
page.link	文章的外部链接（用于链接文章）

首页（index）

变量	描述
page.per_page	每页显示的文章数量
page.total	总页数
page.current	目前页数
page.current_url	目前分页的网址
page.posts	本页文章
page.prev	上一页的页数。如果此页是第一页的话则为 0 。
page.prev_link	上一页的连结。如果此页是第一页的话则为 '' 。
page.next	下一页的页数。如果此页是最后一页的话则为 0 。
page.next_link	下一页的网址。如果此页是最后一页的话则为 '' 。
page.path	当前页面的路径（不含根目录）。我们通常在主题中使用 url_for(page.path) 。

归档 (archive)：与 index 布局相同，但新增以下变量。

变量	描述
archive	等于 true
year	归档年份（4 位数）
month	归档月份（不含开头的零）

分类 (category)：与 index 布局相同，但新增以下变量。

变量	描述
`category`	分类名称

标籤 (tag)：与 `index` 布局相同，但新增以下变量。

变量	描述
`tag`	标签名称

辅助函数（Helpers）

辅助函数帮助您在模版中快速插入内容。

网址

url_for

在路径前加上根路径，从 Hexo 2.7 开始您应该使用此函数，避免使用 `config.root + path` 。

```
<%- url_for(path) %>
```

relative_url

取得与 `from` 相对的 `to` 路径。

```
<%- relative_url(from, to) %>
```

gravatar

插入 Gravatar 图片。

```
<%- gravatar(email, [size]);
```

示例：

```
<%- gravatar('a@abc.com') %>
// http://www.gravatar.com/avatar/b9b00e66c6b8a70f88c73cb6bdb06787

<%- gravatar('a@abc.com', 40) %>
// http://www.gravatar.com/avatar/b9b00e66c6b8a70f88c73cb6bdb06787?s=40
```

HTML 标签

css

载入 CSS 文件。`path` 可以是数组或字符串，如果 `path` 开头不是 `/` 或任何协议，则会自动加上根路径；如果后面没有加上 `.css` 扩展名的话，也会自动加上。

```
<%- css(path, ...) %>
```

示例：

```
<%- css('style.css') %>
// <link rel="stylesheet" href="/style.css" type="text/css">

<%- css(['style.css', 'screen.css']) %>
// <link rel="stylesheet" href="/style.css" type="text/css">
// <link rel="stylesheet" href="/screen.css" type="text/css">
```

js

载入 JavaScript 文件。`path` 可以是数组或字符串，如果 `path` 开头不是 `/` 或任何协议，则会自动加上根路径；如果后面没有加上 `.js` 扩展名的话，也会自动加上。

```
<%- js(path, ...) %>
```

示例：

```
<%- js('script.js') %>
// <script type="text/javascript" src="/script.js"></script>

<%- js(['script.js', 'gallery.js']) %>
// <script type="text/javascript" src="/script.js"></script>
// <script type="text/javascript" src="/gallery.js"></script>
```

link_to

插入链接。

```
  <%- link_to(path, [text], [options]) %>
```

参数	描述	默认值
external	在新视窗打开链接	false
class	Class 名称	
id	ID	

示例：

```
<%- link_to('http://www.google.com') %>
// <a href="http://www.google.com" title="http://www.google.com">http://www.google.com</a>

<%- link_to('http://www.google.com', 'Google') %>
// <a href="http://www.google.com" title="Google">Google</a>

<%- link_to('http://www.google.com', 'Google', {external: true}) %>
// <a href="http://www.google.com" title="Google" target="_blank" rel="external">Google</a>
```

mail_to

插入电子邮箱链接。

```
<%- mail_to(path, [text], [options]) %>
```

参数	描述
class	Class 名称
id	ID
subject	邮件主题
cc	抄送（CC）
bcc	密送（BCC）
body	邮件内容

示例：

```
<%- mail_to('a@abc.com') %>
// <a href="mailto:a@abc.com" title="a@abc.com">a@abc.com</a>

<%- mail_to('a@abc.com', 'Email') %>
// <a href="mailto:a@abc.com" title="Email">Email</a>
```

image_tag

插入图片。

```
<%- image_tag(path, [options]) %>
```

参数	描述
alt	图片的替代文字
class	Class 名称
id	ID
width	图片宽度
height	图片高度

favicon_tag

插入 favicon。

```
<%- favicon_tag(path) %>
```

feed_tag

插入 feed 链接。

```
<%- feed_tag(path, [options]) %>
```

参数	描述	默认值
title	Feed 标题	
type	Feed 类型	atom

条件函数

is_current

检查 `path` 是否符合目前页面的网址。开启 `strict` 选项启用严格比对。

```
<%- is_current(path, [strict]) %>
```

is_home

检查目前是否为首页。

```
<%- is_home() %>
```

is_post

检查目前是否为文章。

```
<%- is_post() %>
```

is_archive

检查目前是否为存档页面。

```
<%- is_archive() %>
```

is_year

检查目前是否为年度归档页面。

```
<%- is_year() %>
```

is_month

检查目前是否为月度归档页面。

```
<%- is_month() %>
```

is_category

检查目前是否为分类归档页面。

```
<%- is_category() %>
```

is_tag

检查目前是否为标签归档页面。

```
<%- is_tag() %>
```

字符串处理

trim

清除字符串开头和结尾的空格。

```
<%- trim(string) %>
```

strip_html

清除字符串中的 HTML 标签。

```
<%- strip_html(string) %>
```

示例：

```
<%- strip_html('It's not important anymore!') %>
// It's not important anymore!
```

titlecase

把字符串转换为正确的 Title case。

```
<%- titlecase(string) %>
```

示例：

```
<%- titlecase('this is an apple') %>
# This is an Apple
```

markdown

使用 Markdown 解析字符串。

```
<%- markdown(str) %>
```

示例：

```
<%- markdown('make me **strong**') %>
// make me <strong>strong</strong>
```

render

解析字符串。

```
<%- render(str, engine, [options]) %>
```

word_wrap

使每行的字符串长度不超过 `length` 。 `length` 预设为 80。

```
<%- word_wrap(str, [length]) %>
```

示例:

```
<%- word_wrap('Once upon a time', 8) %>
// Once upon\n a time
```

truncate

移除超过 `length` 长度的字符串。

```
<%- truncate(text, length) %>
```

示例:

```
<%- truncate('Once upon a time in a world far far away', 16) %>
// Once upon a time
```

模板

partial

载入其他模板文件，您可在 `locals` 设定区域变量。

```
<%- partial(layout, [locals], [options]) %>
```

参数	描述	默认值
cache	缓存（使用 Fragment cache）	false
only	限制局部变量。在模板中只能使用 locals 中设定的变量。	false

fragment_cache

局部缓存。它储存局部内容，下次使用时就能直接使用缓存。

```
<%- fragment_cache(id, fn);
```

示例：

```
<%- fragment_cache('header', function(){
  return '';
}) %>
```

日期与时间

date

插入格式化的日期。`date` 可以是 UNIX 时间、ISO 字符串、Date 对象或 Moment.js (http://momentjs.com/) 对象。`format` 默认为 `date_format` 配置信息。

```
<%- date(date, [format]) %>
```

示例：

```
<%- date(Date.now()) %>
// Jan 1, 2013

<%- date(Date.now(), 'YYYY/M/D') %>
// 2013/1/1
```

date_xml

插入 XML 格式的日期。`date` 可以是 UNIX 时间、ISO 字符串、Date 对象或 Moment.js (http://momentjs.com/) 对象。

```
<%- date_xml(date) %>
```

示例：

```
<%- date_xml(Date.now()) %>
// 2013-01-01T00:00:00.000Z
```

time

插入格式化的时间。`date` 可以是 UNIX 时间、ISO 字符串、Date 对象或 [Moment.js][1] 对象。`format` 默认为 `time_format` 配置信息。

```
<%- time(date, [format]) %>
```

示例：

```
<%- time(Date.now()) %>
// 13:05:12

<%- time(Date.now(), 'h:mm:ss a') %>
// 1:05:12 pm
```

full_date

插入格式化的日期和时间。`date` 可以是 UNIX 时间、ISO 字符串、Date 对象或 Moment.js (http://momentjs.com/) 对象。`format` 默认为 `time_format` 配置信息。

```
<%- full_date(date, [format]) %>
```

示例：

```
<%- full_date(new Date()) %>
// Jan 1, 2013 0:00:00

<%- full_date(new Date(), 'dddd, MMMM Do YYYY, h:mm:ss a') %>
// Tuesday, January 1st 2013, 12:00:00 am
```

moment

Moment.js (http://momentjs.com/) 函数库。

列表

list_categories

插入分类列表。

```
<%- list_categories([categories], [options]) %>
```

参数	描述	默认值
orderby	分类排列方式	name
order	分类排列顺序。1，asc 升序；-1，desc 降序。	1
show_count	显示每个分类的文章总数	true
style	分类列表的显示方式。使用 list 以无序列表（unordered list）方式显示。	list
separator	分类间的分隔符号。只有在 style 不是 list 时有用。	,
depth	要显示的分类层级。0 显示所有层级的分类；-1 和 0 很类似，但是显示不分层级；1 只显示第一层的分类。	0
class	分类列表的 class 名称。	category
transform	改变分类名称显示方法的函数	

list_tags

插入标签列表。

```
<%- list_tags([tags], [options]) %>
```

选项	描述	预设值
orderby	标签排列方式	name
order	标签排列顺序。1，asc 升序；-1，desc 降序。	1
show_count	显示每个标签的文章总数	true
style	标签列表的显示方式。使用 list 以无序列表（unordered list）方式显示。	list
separator	标签间的分隔符号。只有在 style 不是 list 时有用。	,
class	标签列表的 class 名称。	tag
transform	改变标签名称显示方法的函数	
amount	要显示的标签数量（0 = 无限制）	0

list_archives

插入归档列表。

```
<%- list_archives([options]) %>
```

参数	描述	默认值
type	类型。此设定可为 yearly 或 monthly 。	monthly
order	排列顺序。1 , asc 升序; −1 , desc 降序。	1
show_count	显示每个归档的文章总数	true
format	日期格式	MMMM YYYY
style	归档列表的显示方式。使用 list 以无序列表（unordered list）方式显示。	list
separator	归档间的分隔符号。只有在 style 不是 list 时有用。	,
class	归档列表的 class 名称。	archive
transform	改变归档名称显示方法的函数	

list_posts

插入文章列表。

```
<%− list_posts([options]) %>
```

参数	描述	默认值
orderby	文章排列方式	date
order	文章排列顺序。1 , asc 升序; −1 , desc 降序。	−1
style	文章列表的显示方式。使用 list 以无序列表（unordered list）方式显示。	list
separator	文章间的分隔符号。只有在 style 不是 list 时有用。	,
class	文章列表的 class 名称。	post
amount	要显示的文章数量（0 = 无限制）	6
transform	改变文章名称显示方法的函数	

tagcloud

插入标签云。

```
<%− tagcloud([tags], [options]) %>
```

参数	描述	默认值
min_font	最小字体尺寸	10

参数	描述	默认值
max_font	最大字体尺寸	20
unit	字体尺寸的单位	px
amount	标签总量	40
orderby	标签排列方式	name
order	标签排列顺序。1，sac 升序；-1，desc 降序	1
color	使用颜色	false
start_color	开始的颜色。您可使用十六进位值（#b700ff），rgba（rgba(183, 0, 255, 1)），hsla（hsla(283, 100%, 50%, 1)）或 [颜色关键字][2]。此变量仅在 color 参数开启时才有用。	
end_color	结束的颜色。您可使用十六进位值（#b700ff），rgba（rgba(183, 0, 255, 1)），hsla（hsla(283, 100%, 50%, 1)）或 [颜色关键字][2]。此变量仅在 color 参数开启时才有用。	

其他

paginator

插入分页链接。

```
<%- paginator(options) %>
```

参数	描述	默认值
base	基础网址	/
format	网址格式	page/%d/
total	分页总数	1
current	目前页数	0
prev_text	上一页链接的文字。仅在 prev_next 设定开启时才有用。	Prev
next_text	下一页链接的文字。仅在 prev_next 设定开启时才有用。	Next
space	空白文字	&hellp;

参数	描述	默认值
prev_next	显示上一页和下一页的链接	true
end_size	显示于两侧的页数	1
mid_size	显示于中间的页数	2
show_all	显示所有页数。如果开启此参数的话，end_size 和 mid_size 就没用了。	false

search_form

插入 Google 搜索框。

```
<%- search_form(options) %>
```

参数	描述	默认值
class	表单的 class name	search-form
text	搜索提示文字	Search
button	显示搜索按钮。此参数可为布尔值（boolean）或字符串，当设定是字符串的时候，即为搜索按钮的文字。	false

number_format

格式化数字。

```
<%- number_format(number, [options]) %>
```

参数	描述	默认值
precision	数字精度。此选项可为 false 或非负整数。	false
delimiter	千位数分隔符号	,
separator	整数和小数之间的分隔符号	.

示例：

```
<%- number_format(12345.67, {precision: 1}) %>
// 12,345.68

<%- number_format(12345.67, {precision: 4}) %>
// 12,345.6700

<%- number_format(12345.67, {precision: 0}) %>
// 12,345
```

```
<%- number_format(12345.67, {delimiter: ''}) %>
// 12345.67

<%- number_format(12345.67, {separator: '/'}) %>
// 12,345/67
```

open_graph

插入 open graph 资源。

```
<%- open_graph([options]) %>
```

参数	描述	默认值
title	页面标题 (og:title)	page.title
type	页面类型 (og:type)	blog
url	页面网址 (og:url)	url
image	页面图片 (og:image)	内容中的图片
site_name	网站名称 (og:site_name)	config.title
description	页面描述 (og:desription)	内容摘要或前 200 字
twitter_card	Twitter 卡片类型 (twitter:card)	summary
twitter_id	Twitter ID (twitter:creator)	
twitter_site	Twitter 网站 (twitter:site)	
google_plus	Google+ 个人资料链接	
fb_admins	Facebook 管理者 ID	
fb_app_id	Facebook 应用程序 ID	

toc

解析内容中的标题标签 (h1~h6) 并插入目录。

```
<%- toc(str, [options]) %>
```

参数	描述	默认值
class	Class 名称	toc
list_number	显示编号	true

示例:

```
<%- toc(page.content) %>
```

本地化（i18n）

若要让您的网站以不同语言呈现，您可使用本地化（localization）功能。请先在 `_config.yml` 中调整 `language` 设定，这代表的是预设语言，您也可设定多个语言来调整预设语言的顺位。

```
language: zh-tw

language:
- zh-tw
- en
```

语言文件

语言文件可以使用 YAML 或 JSON 编写，并放在主题文件夹中的 `languages` 文件夹。您可以在语言文件中使用 printf 格式 (https://github.com/alexei/sprintf.js)

模板

在模板中，透过 `__` 或 `_p` 辅助函数，即可取得翻译后的字符串，前者用于一般使用；而后者用于复数字符串。例如：

en.yml

```
index:
  title: Home
  add: Add
  video:
    zero: No videos
    one: One video
    other: %d videos
```

```
<%= __('index.title') %>
// Home

<%= _p('index.video', 3) %>
// 3 videos
```

路径

您可在 front-matter 中指定该页面的语言，也可在 `_config.yml` 中修改 `i18n_dir` 设定，让 Hexo 自动侦测。

```
i18n_dir: :lang
```

`i18n_dir` 的预设值是 `:lang`，也就是说 Hexo 会捕获网址中的第一段以检测语言，举例来说：

```
/index.html => en
/archives/index.html => en
/zh-tw/index.html => zh-tw
```

捕获到的字符串唯有在语言文件存在的情况下，才会被当作是语言，因此例二 `/archives/index.html` 中的 `archives` 就不被当成是语言。

插件

Hexo 有强大的插件系统，使您能轻松扩展功能而不用修改核心模块的源码。在 Hexo 中有两种形式的插件：

脚本（Scripts）

如果您的代码很简单，建议您编写脚本，您只需要把 JavaScript 文件放到 `scripts` 文件夹，在启动时就会自动载入。

插件（Packages）

如果您的代码较复杂，或是您想要发布到 NPM 上，建议您编写插件。首先，在 `node_modules` 文件夹中建立文件夹，文件夹名称开头必须为 `hexo-`，如此一来 Hexo 才会在启动时载入。文件夹内至少要包含 2 个文件：一个是主程序，另一个是 `package.json`，描述插件的用途和所依赖的插件。

```
.
├── index.js
└── package.json
```

`package.json` 中至少要包含 `name`，`version`，`main` 属性，例如：

package.json

```
{
  "name": "hexo-my-plugin",
  "version": "0.0.1",
  "main": "index"
}
```

开发

Hexo 共有九种插件，您可以在 API 页面中获得更多信息：

- Generator
- Renderer
- Helper
- Deployer

- Processor
- Tag
- Console
- Migrator
- Filter

工具

您可以使用 Hexo 提供的官方工具插件来加速开发：

- hexo-fs (https://github.com/hexojs/hexo-fs)：文件 IO
- hexo-util (https://github.com/hexojs/hexo-util)：工具程式
- hexo-i18n (https://github.com/hexojs/hexo-i18n)：本地化（i18n）
- hexo-pagination (https://github.com/hexojs/hexo-pagination)：生成分页资料

发布

当您完成插件后，可以考虑将它发布到 插件列表 (http://hexo.io/plugins/)，让更多人能够使用您的插件。发布插件的步骤和 更新文件 (http://hexo.io/plugins/) 非常类似。

1. Fork hexojs/site (https://github.com/hexojs/site)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

```
$ git clone https://github.com//site.git
$ cd site
$ npm install
```

1. 编辑 source/_data/plugins.yml，在档案中新增您的插件，例如：

```
- name: hexo-server
  description: Server module for Hexo.
  link: https://github.com/hexojs/hexo-server
  tags:
    - official
    - server
    - console
```

1. 推送（push）分支。

2. 建立一个新的合并申请（pull request）。

4

其他

问题解答

在使用 Hexo 时，您可能会遇到一些问题，下列的常见问题解答可能会对您有所帮助。如果您在这里找不道解答，可以在 GitHub (https://github.com/hexojs/hexo/issues) 或 Google Group (https://groups.google.com/group/hexo) 上提问。

YAML 解析错误

```
JS-YAML: incomplete explicit mapping pair; a key node is missed at line 18, column 29:
    last_updated: Last updated: %s
```

如果 YAML 字符串中包含冒号（:）的话，请加上引号。

```
JS-YAML: bad indentation of a mapping entry at line 18, column 31:
    last_updated:"Last updated: %s"
```

请确认您使用空格进行缩进（Soft tab），并确认冒号后有加上一个空格。

您可参阅 YAML 规范 (http://www.yaml.org/spec/1.2/spec.html) 以取得更多信息。

EMFILE 错误

```
Error: EMFILE, too many open files
```

虽然 Node.js 有非阻塞 I/O，同步 I/O 的数量仍被系统所限制，在生成大量静态文件的时候，您可能会碰到 EMFILE 错误，您可以尝试提高同步 I/O 的限制数量来解决此问题。

```
$ ulimit -n 10000
```

Git 部署问题

```
fatal: 'username.github.io' does not appear to be a git repository
```

请确认您已经在电脑上 配置 git (https://help.github.com/articles/set-up-git/)，或改用 HTTPS 库（repository）地址。

服务器问题

```
Error: listen EADDRINUSE
```

您可能同时开启两个 Hexo 服务器，或者有其他应用程序正在占用相同的端口，请尝试修改 port 参数，或是在启动 Hexo 服务器时加上 -p 选项。

```
$ hexo server -p 5000
```

插件安装问题

```
npm ERR! node-waf configure build
```

当您尝试安装以 C/C++ 或其他非 JavaScript 语言所编写的插件时，可能会遇到此类问题，请确认您已经在电脑上安装相对应的编译器。

在 Jade 或 Swig 遍历资料

Hexo 使用 Warehouse (https://github.com/tommy351/warehouse) 存储资料，它不是一般数组所以必须先进行类型转型才能遍历。

```
{% for post in site.posts.toArray() %}
{% endfor %}
```

资料没有更新

有时资料可能没有被更新，或是生成的文件与修改前的相同，您可以尝试清除缓存并再执行一次。

```
$ hexo clean
```

泄露（Escape）内容

Hexo 使用 Nunjucks (http://mozilla.github.io/nunjucks/) 来解析文章（旧版本使用 Swig (http://paularmstrong.github.io/swig/)，两者语法类似），内容若包含 {{ }} 或 {% %} 可能导致解析错误，您可以用 raw 标签包裹来避免潜在问题发生。

```
{% raw %}
Hello {{ sensitive }}
{% endraw %}
```

贡献

开发

我们非常欢迎您加入 Hexo 的开发，这份文件将帮助您了解开发流程。

开始之前

请遵守以下准则：

遵守 Google JavaScript 代码风格 (http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml) 使用 2 个空格缩排。 不要把逗号放在最前面。

工作流程

1. Fork hexojs/hexo (https://github.com/hexojs/hexo)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

```
    $ git clone https://github.com//hexo.git
$ cd hexo
$ npm install
$ git submodule update --init
```

1. 新增一个功能分支。

```
    $ git checkout -b new_feature
```

1. 开始开发。
2. 推送（push）分支。

```
    $ git push origin new_feature
```

1. 对 `master` 分支建立一个新的合并申请（pull request）并描述变动。

注意事项

- 不要修改 package.json 的版本号。

- 只有在测试通过的情况下您的合并申请才会被批准，在提交前别忘了进行测试。

```
$ npm test
```

更新文件

Hexo 文件开放源代码，您可以在 hexojs/site (https://github.com/hexojs/site) 找到源代码，若要修改文件：

1. Fork hexojs/site (https://github.com/hexojs/site)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

```
    $ git clone https://github.com//site.git
$ cd site
$ npm install
```

1. 开始编辑文件，您可以通过服务器预览变动。

```
$ hexo server
```

1. 推送（push）分支。
2. 对 `master` 分支建立一个新的合并申请（pull request）并描述变动。

翻译文件

翻译文件的流程与上述的文件更新非常相似，如果要新增语言的话，请遵照以下步骤：

1. 在 `source` 资料夹中建立一个新的语言资料夹（全小写）。
2. 把 `source` 资料夹中相关的文件（Markdown 和模板文件）复制到新的语言资料夹中。
3. 在 `source/_data/language.yml` 中新增语言。
4. 在 `themes/navy/languages` 复制 `en.yml` 并命名为语言名称（全小写）。

反馈问题

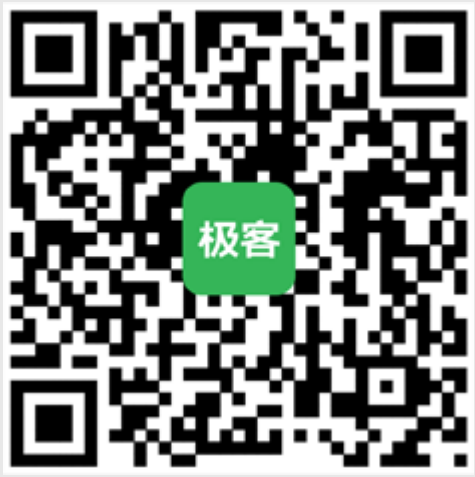
当您在使用 Hexo 时遇到问题，您可以尝试在 问题解答 (页 67) 中寻找解答，或是在 GitHub (https://github.com/hexojs/hexo/issues) 或 Google Group (https://groups.google.com/forum/#!forum/hexo) 上提问。提问时请务必附上以下信息：

1. 以 调试模式 (页 0) 再执行一次。
2. 检查版本信息。
3. 把调试信息和版本信息都贴到 GitHub。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台



更多信息请访问

http://wiki.jikexueyuan.com/project/hexo-document/