# 0x00 > 写在开始

这里有已经翻译好的内容：

[www.jekyll.com.cn](http://www.jekyll.com.cn)

<http://jekyllcn.com/>

# 0x01 > 欢迎

本站的目标是成为综合 Jekyll 使用指南。我们的话题覆盖了：生成并运行你的站点，创建和管理内容，自定义构建和分发方式……

## Jekyll的本质是什么？

Jekyll 是一个简易的，自助的，静态博客生成器。

你通过文件夹来组织你创建的文本文件(Markdown)。然后使用增强的Liquid 模板来构建网站外壳。Jekyll 自动讲文本内容和模板组织在一起生成静态页面，可以方便的发布到任何服务器。Jekyll 恰好是GitHub Pages 的引擎，所以你可以免费将自己的jekyll项目(page/blog/website)部署到GitHub服务器。

## 阅读指南

**ProTips**提示和技巧帮助你成为Jekyll魔法师

**Notes** 有时必要的额外信息可以帮助理解 Jekyll

**Warnings** 避免常见陷阱

**Unreleased** Jekyll正在开发中的功能，当前不可用

如果你发现任何没覆盖到的问题，或者想分享其他有帮助的提示，可以创建一个issue,我们看到后会添加到本指南中。

# 0x02 > 快速开始指南

如果你有一个包含所有头文件和已经安装了RubyGems的完整的Ruby开发环境，你可以通过如下命令创建一个jekyll站点：

|  |
| --- |
| treminal |
| *# 通过RubyGems安装 Jekyll 和 Bundler*  gem install jekyll bundler  *# 在 ./myblog 创建一个新的 Jekyll 站点。*  jekyll new myblog  *# 进入新建的文件夹*  cd myblog  *# 在预览服务器上构建站点*  bundle exec jekyll serve  *# 现在可以浏览 http://localhost:4000* |

如果你在上述操作中碰到任何意想不到的错误，请参考Troubleshooting(https://jekyllrb.com/docs/troubleshooting/#configuration-problems)页面或者已经提及的requirements(https://jekyllrb.com/docs/installation/#requirements)页面，因为你可能缺少开发头文件或者其他先决条件。

## 关于Bundler

**gem install bundler** 通过RubyGems安装bundler gem。你只需要进行一次安装(而不是在每次创建Jekyll的时候都安装一次)。一些额外的细节如下：

* bundler 是一个管理者其他 Ruby gems 的gem。它确保你的gems和gem版本相兼容并且每一个gem都有其必要的依赖。
* Gemfile和Gemfile.lock文件供 Bundler 了解站点中 gem 的需求。如果你的站点没有这些文件，你可以忽略 bundle exec 仅执行 jekyll serve。
* 当你执行 bundle exec jekyll server，Bundler使用Gemfile.lock 中制定的gem的版本，确保你的Jekyll站点没有兼容性问题和依赖冲突。

更多关于如何在Jekyll项目中使用 Bundler 的问题，这个教程(https://jekyllrb.com/tutorials/using-jekyll-with-bundler/)提供了常见问题的答案，并解释了如何快速启动和运行。

## 使用 Jekyll 创建一个新站点的操作

jekyll new <PATH> 在（相对于当前文件夹）指定的路径创建一个新的Jekyll站点。在当前教程中的示例中Jekyll站点被安装在myblog文件夹中。还有一些额外的细节：

* 在当前文件夹创建Jekyll站点的命令为 jekyll new . 如果当前文件夹非空，可以通过命令强制创建：jekyll new . --force 。
* jekyll new 命令会自动执行 bundle install 安装依赖。（如果你不希望Bundler安装gems。可以：jekyll new myblog --skip-bundle）
* 默认情况下，通过jekyll new Jekyll安装的站点使用一个基于gem的主题：Minima。使用基于gem的主题，一些文件夹和文件存储在gem安装目录，于jekyll 站点目录中不可见。（Ubuntu下为：/var/lib/gems/2.3.0/gems/minima-2.5.0）
* 我们通常设置基于gem的主题，但是如果你希望设置空白模板可以：jekyll new myblog --blank。
* 学习更多关于 jekyll new 的命令：jekyll new --help 。

当你怀疑使用 help 命令提醒自己可用的选项的时候要知道，new，build，server这些子命令可以这样：jekyll help new 或者 jekyll help build 。

## 下一步

使用默认的模板构建一个 Jekyll 站点只是第一步。当你使用Jekyll中可用的配置设置好了模板和布局并且应用于blog的时候才真正感觉到有魔力。

# 0x03 > 安装

Jekyll是一个 Ruby Gem，可以被安装到大多数操作系统。

## 依赖

开始之前，确保你的系统有如下内容：

* Ruby 版本 >= 2.25 （使用 ruby -v 检查）
* RubyGems已经安装（私用 gem -v 检查）
* GCC 和 Make 已安装（使用 gcc -v , g++ -v 和 make -v 检查）

# 0x04 > Jekyll on Windows

# 0x05 > 基本用法

Jekyll gem 可以在命令行中执行，常用的命令如下：

|  |
| --- |
| terminal |
| jekyll build  # => 当前文件夹会被生成到 ./\_site 中  jekyll build --destination <destination>  # => 当前文件夹会被生成到 <destination> 中  jekyll build --source <source> --destination <destination>  # => <source> 文件夹会被生成到 <destination> 中  jekyll build --watch  # => 当前文件夹会被生成到 ./\_site 中，监听变化并自动重新生成。 |

## 覆盖默认的开发设置

在开发环境中默认的 URL 被设置为 <http://localhost:4000> （3.3.0）

如果你希望为你的产品设置环境变量：

* 在 \_config.yml 设置你的产品URL例如：url:https://example.com。
* 运行 JEKYLL\_ENV=projuction bundle exec jekyll build

**不在自动再生期间修改 \_config.yml 。**

\_config.yml 配置文件中的全局配置信息在执行的时候会被读取一次。在自动再生期间修改了其内容，只有到下次加载的时候修改才会被执行。

注意在重新自动生成期间 Data Files 被包含在重新加载的范围。

**在构建站点的时候目标文件家会进行清理**

默认情况下当站点构建的时候<destination>中的内容会被自动清理。不是通过站点生成的文件或者文件夹将会被清理。如果有希望继续保留的文件或者文件夹可以通过<keep\_files>进行配置。

不要将<destination>设置为重要的路径；而是应该设置已改中转区用于从这里复制文件到你的站点。

Jekyll包含一个内置的开发服务器，这可以让你在本地浏览器预览自己即将生成的站点。

|  |
| --- |
| terminal |
| jekyll server  *# => 一个开发服务器将运行于 <http://localhost:4000/>*  *# => 可以通过 ‘--no-watch’ 取消自动重新生成*  jekyll server --livereload  *# => 实时重新（LiveReload）加载会在修改后刷新浏览器。*  jekyll server--incremental  *# => 增量（incremental）将执行部分构建以减少自动生成的时间*  jekyll server--detach  *# => 作用和 ‘jekyll server’ 相同，但是会和终端分离。*  *# => 如果你需要杀死 server 可以 ‘kill -9 1234’ , 1234是 jekyll PID*  *# => 查看 PID ‘ps aux | grep jekyll’*  jekyll server --no-watch  *# => 作用和 ‘jekyll server’ 相同，但是没有自动监听变化。* |

这里只有部分可用的配置选项。在命令行中指定的配置选项也可以写在站点根目录 \_comfig.yml ,具有相同的作用。写在这个文件中的配置选项会在Jekyll启动的时候自动加载，例如：

|  |
| --- |
| source : \_source  destination: \_deploy |

在上述配置下，下述两个命令其本质是相同的。

|  |
| --- |
| terminal |
| jekyll build  jekyll build --source \_source --destination \_deploy |

更多的配置选项请查看配置(configuration)页面。

**需要帮助**

help命令可以提示你所有可用的命令和用法，其作为 build,server,new 的子命令用法如下：jekyll help new 或者 jekyll help build

如果你对阅读本文档感兴趣，可以安装 jekyll-docs 这个 gem。然后运行 jekyll docs 就可以在本地浏览器查看。

# 0x06 > 文件结构

Jekyll 的核心是一个文本转换引擎。这个概念通俗的说就是，你提供使用文本标记语言(Markdown/Textile/HTML)写的文本，Jekyll 负责将文本文件和一系列布局文件(按照预设配置)整合在一起。从始至终你都可以调整 URL 的风格，和所展示的文本布局和其他任何你希望改变的内容。这些事都通过修改文本文件来做到，静态网站就是最终的产品。

一个 Jekyll 的目录结构通常如下所示：

|  |
| --- |
| terminal |
| .  ├── \_config.yml  ├── \_data  | └── members.yml  ├── \_drafts  | ├── begin-with-the-crazy-ideas.md  | └── on-simplicity-in-technology.md  ├── \_includes  | ├── footer.html  | └── header.html  ├── \_layouts  | ├── default.html  | └── post.html  ├── \_posts  | ├── 2007-10-29-why-every-programmer-should-play-nethack.md  | └── 2009-04-26-barcamp-boston-4-roundup.md  ├── \_sass  | ├── \_base.scss  | └── \_layout.scss  ├── \_site  ├── .jekyll-metadata  └── index.html # can also be an 'index.md' with valid YAML Frontmatter |

**jekyll站点文件结构使用基于 gem 的主题。**

从 Jekyll3.2 开始 ‘jekyll new’ 命令自动使用基于 gem 的主题作为站点外观。这样做的结果是文件结构：\_layout , \_include , \_sass 都默认存储在gem主题文件夹中。

minima是当前默认的主题，bundle show minima 可以显示minima主题文件在计算机中存储的位置。

|  |
| --- |
| tom@lover:/media/tom/document/test\_jekyll/test\_site\_1$ bundle show minima  /var/lib/gems/2.3.0/gems/minima-2.5.0 |

文件概览：

|  |  |
| --- | --- |
| **文件/文件夹** | **描述** |
| \_config.yml | 存储配置信息。很多配置选项可以再命令行执行的时候制定，写在这个配置文件中的好处就是不用担心记不住。 |
| \_drafts | 草稿（Drafts）是未公开的文章。这些文件的格式没有title.MARKUP数据。学习如何使用草稿。 |
| \_includes | 这里的内容可以通过布局和文章进行混合和匹配今儿促进重用。liquid标签{% include file.ext %} 可以使用 \_include/file.ext文件。 |
| \_layouts | 这里是包裹文章的布局。布局是在每一篇文章中通过 YAML 头信息单独指定的(关于 YAML 头信息在后续会讲解)。Liquid标签{{ content }} 用于注入文章内容。 |
| \_posts | 这里是你的动态内容。这里文章的命名格式为YEAR-MONTH-DATA-title.MARKUP 。每篇文章的访问链接都可以被定制但是日志和标记语言都是通过文件名指定的。 |
| \_data | 格式化的站点数据放在这里。Jekyll引擎讲自动加载所有的文件(使用.yml，.yaml，.json或者.csv格式的扩展名)在这个文件夹中，它们将通过 ‘site.data’ 访问。如果这个文件夹中有 members.yml 文件，你可以通过 site.data.members 访问。 |
| \_sass | 这部分能被导入main.scss，main.scss将被执行进入main.css ，main.css定义了站点的风格。 |
| \_site | 默认情况下Jekyll讲转换后的生成的站点文件放到这个文件夹。将这个文件夹加入到.gitignore文件是个不错的实践。 |
| .jekyll-metadata | 这个文件帮助追踪最后一次build之后尚未被修改的文件，以及那个文件需要在下次build中重新生成。这个文件将不会被包含到生成的站点。将之加入到.gitignore是不错的实践。 |
| index.html/index.md和其他HTML，Markdown文件 | 文件中具有YAML头信息的小节，这个文件将会被Jekyll转换。根目录中上述未列出的文件（.html，.markdown，.md，.textile）或者文件夹都会被Jekyll转换。 |
| 其他文件/文件夹 | 任何除上述列出的文件或者文件夹，例如 css 和 image 文件件，favicon.icon 文件，等等将会被逐个复制到到生成的站点。如果你好奇别人是如何配置的，这里有大量基于Jekyll站点：<https://jekyllrb.com/docs/sites/> |

# 0x07 > 配置

Jekyll允许你按照任何你希望的方式控制你的站点，这得益于灵活切强大的配置玄子昂提供支持。这些选项既可以在站点根目录的 \_config.yml 文件中进行设置也可以在执行jekyll命令的时候使用标签指定。

## 配置设置

### 全局配置

下表罗列了Jekyll可用的设置，变量options用于在配置文件中制定配置参数，flags用于在命令行中指定配置参数。

|  |  |  |
| --- | --- | --- |
| **设置** | **option** | **flags** |
| **Site Source（站点源码）**  修改Jekyll读取文件的文件夹 | source: DIR | -s , --source DIR |
| **Site Destination（站点目的）**  设置Jekyll输出文件的文件夹 | destination:DIR | -d , --destination DIR |
| **Safe**  取消自定义插件忽略符号链接(https://jekyllrb.com/docs/plugins/) | safe:BOOL | --safe |
| **Exclude（排除）**  从待转换的文件中排除文件或者文件夹。排除的内容相对于站点源码（Site Source）不能是其他文件目录。 | exclude:[DIR , File , …] |  |
| **Include（包含）**  强制包含一些文件或者文件夹到待转换子集中。.htaccess 是一个好例子因为点文件是默认被排除（exclude）的。 | Include:[DIR , File , …] |  |
| **Keep files（保留文件）**  当阻塞站点目标（destination）时，保留所选文件。对于不是有Jekyll生成的文件是有用的；例如由你的构建工具生成的文件或者资源，他们的路径是相对于destination的。 | Keep\_file:[DIR , File , …] |  |
| **Time Zone（时区）**  为生成的站点设置时区。修改Ruby用于创建和操作日期和时间的环境变量 TZ 。[IANA 时区数据库](https://en.wikipedia.org/wiki/Tz_database)中的任何实体都是有效的，例如 America/New\_York 。[一个完整的实体列表](https://en.wikipedia.org/wiki/List_of_tz_database_time_zones)。当运行于本地机器的时候默认的时区被设置为操作系统的时区。但是当你运行于远程服务器时区依赖于配置参数。 | timezone:TIMEZONE |  |
| **Encoding（编码）**  通过名字设置文件的编码（只在Ruby version >=1.9可行）。在Jekyll>=2.0.0默认值是 utf-8在这之前默认值是nil，Ruby的默认值是ASCII-8BIT。编码可以通过如下命令查看：ruby -e ‘puts Encoding::list.join(“\n”)’ | encoding:ENCODING |  |
| **Defaults（默认）**  为 YAML头设置默认值 | 查看下文 |  |

**在构建的时候目标（Destination）文件夹被清空**

默认的，当站点被构建的时候<destination>文件夹中的内容会自动被清空。管理者创建的任何文件或者文件夹都会被移除。一些设置了<keep\_files>的文件会被保留下来。

不要讲<Destination>设置在一个重要的路径，应该使用一个中转路径用于从这里复制文件到服务器。

### 构建命令选项

|  |  |  |
| --- | --- | --- |
| **设置** | **option** | **flag** |
| **Regeneration（再生）**  当文件被修改是否允许站点自动再生。 |  | -w , --[no-]watch |
| **Configuration（配置）**  制定取代 \_config.yml 的配置文件。设置用后面的文件覆盖前面的文件。 |  | --config FILE1[,FILE2,…] |
| **Drafts（草稿）**  执行和渲染草稿文章。 | show\_drafts:BOOL | --drafts |
| **Environment（环境）**  在构建的时候使用指定的环境变量。 | JEKYLL\_EVN=production |  |
| **Future（将来）**  用将来的日期发布文章。 | future:BOOL | --future |
| **Unpublished（未发布）**  渲染标记为未发布的文章。 | unpublished:BOOL | --unpublished |
| LSI  为发布的文章建立索引。需要[classifier-reborn](http://www.classifier-reborn.com/)插件。 | lsi:BOOL | --lsi |
| **Limit Posts（限制文章）**  限制发布文章的数量。 | limit\_posts:NUM | --limit\_posts NUM |
| **Force polling（强制轮循）**  使用强制轮循。 |  | --force\_polling |
| **Verbose output（详细输出）**  打印详细输出。 |  | -V , --verbose |
| **Silence Output（静默输出）**  在构建期间不显示正常输出。 |  | -q , --quite |
| **Incremental build（增量构建）**  允许执行增量构建特性增量构建。增量构建仅仅重新构建那些被修改过的文章和页面，这可以提高大型网站的构建效率，也可能在特定情况下停止网站生成。 | incremental:BOOL | -I , --incremental |
| **Liquid profiler（Liquid分析）**  生成 Liquid 渲染分析帮助你识别性能瓶颈。 | profile:BOOL | --profile |
| Strict From Matter  如果页面中的YAML语法有误，构建失败。 | strict\_front\_matter:BOOL | --strict\_front\_matter |

### 服务命令选项

除下面附加的选项，serve 子命令可以是任何 build 的子命令。如果build的子命令确实接在server之后使用，server将要启动的服务将在启动前使用制定的命令进行构建。

|  |  |  |
| --- | --- | --- |
| **设置** | **options** | **flags** |
| **Local Server Port**  监听指定端口 | port:PORT | --port PORT |
| **Local Server Hostname**  监听给定的主机名 | host:HOSTNAME | --host HOSTNAME |
| **Base URL**  网站根路径 | baseurl:URL | --baseurl URL |
| **Detach（分离）**  将服务和命令行分离 | detach:BOOL | -B , --detach |
| **Skips the initial site build**  当服务部署的时候跳过部署前的自动构建 |  | --skip-initial-build |
| **X.509(SSL)Private Key**  SSL 私钥 |  | --ssl-key |
| **X.509(SSL)Certificate**  SSL 公共证书 |  | --ssl-cert |

**在配置文件中不要使用 tab 键**

这么做将导致解析错误，并且Jekyll会将设置信息重置为默认值。在需要使用tab的地方用空格吧。

## 自定义 WEBrick 标题

你可以通过在\_config.yml文件中增加如下内容自定义你站点的标题。

|  |
| --- |
| # File:\_config.yml |
| webrick:  headers:  My-Header:My-Value  My-Other-Header:My-Other-Value |

### 默认

默认的我们提供内容类型（Content-Type）和缓存控制（Cache-Control）响应头：当处于开发模式时，动态可以用于指定被服务的数据，静态可以用于取消缓存从而不必于 Chrome 的积极缓存作斗争。

## 在构建（build）时指定Jekyll环境

在使用 build 或则 server 命令的时候可以制定一个Jekyll环境变量。这个环境变量会被应用于你的任何条件变量中。

例如，你设置一个条件变量如下：

|  |
| --- |
| {% if jekyll.environment == "production" %}  {% include disqus.html %}  {% endif %} |

当你 build Jekyll站点的时候 if 中的条件变量之后在你做如下设置之后，该条件才会返回true。

|  |
| --- |
| terminal |
| JEKYLL\_ENV=production jekyll build |

指定一个环境变量允许你在特定的环境变量下执行指定的内容。

JEKYLL\_ENV 默认值是 development。因此如果你在build阶段忽略JEKYLL\_ENV其值为JEKYLL\_ENV=development。任何 {% if jekyll.environment == "development" %} 标签将自动允许。

除了 development / production 两个值外，你的环境变量的值随便设置。或许你希望隐藏一些在开发环境下的元素例如： Disqus或者Google Analytics。也或者你希望货站一个 Edit me in GitHub 的按钮，但是在生产环境又不显示它。

通过将 JEKYLL\_ENV 在命令行设置一个特殊的值，你不必在环境迁移时修改配置文件。

## 头信息default字段

YAML Front Matter 为用户提供了一种在文章中制定文章路径和风格的途径。诸如默认布局，自定义标题，为文章指定更精确的时间等都可以在YAML Front Matter中设置。

多数情况先，你会发现及重复了大量的配置选项。为文件设置相同的布局，增加相同的分类或者目录等。当然你也可以增加自定义作者标签，当然这对你的博客来说绝大多数都是相同的值。

Jekyll提供了一种方式设置站点的默认配置可以替代每次创建文章的时候都需要重复制定的配置信息。通过设置站点根目录中的\_config.yml文件中的defaults节点你可以制定站点默认风格》。

defaults 关键字对应一个元素为scope/values的数组，该数组定义了每个文件被指定的默认路径，或者在该路径下指定可选的文件类型。

假设你希望讲所有的文件都添加到生成的站点中，你的配置应该是这样的：

|  |
| --- |
| defaults:  -  scope:  path: "" # an empty string here means all files in the project  values:  layout: "default" |

**请停止 Jekyll 并且重新执行 ‘jekyll serve’ 命令。**

包含全局配置信息的\_config.yml文件仅在 ’jekyll server’ 的时候被读取和生效。修改 \_config.yml 文件的内容并不会在自动重新生成阶段生效，需要先停止 jekyll 再次重新执行 ‘jekyll server’。

注意：Data Files 在自动重新生成期间会被重新加载。

上面代码表示所有的文件都会应用 values 中的值。因为 path 所设置的值为空字符串，这意味着文件的范围为整个项目中的所有文件。你或许不想将项目中所有的文件都设置为同一个布局例如 css 文件，所以你可以在scope关键字节点下制定一个 type 关键字。

|  |
| --- |
| defaults:  -  scope:  path: "" # an empty string here means all files in the project  type: "posts" # previously `post` in Jekyll 2.2.  values:  layout: "default" |

上面的代码表示仅将type值为 posts 的文件设置为默认布局。可供选择的type的值有 pages,posts,drafts或者自己网站中定义的。当type被设置的时候你必须为path指定一个值。

如早前提到的在defaults节点中，你可以设置个 scope/values 相对应的节点。

|  |
| --- |
| defaults:  -  scope:  path: ""  type: "pages"  values:  layout: "my-site"  -  scope:  path: "projects"  type: "pages" # previously `page` in Jekyll 2.2.  values:  layout: "project" # overrides previous default layout  author: "Mr. Hyde" |

在上述代码的设置下，所有的布局都设置为 my-site 。任何 projects/ 目录下的html文件都会使用 project 布局。这些文件也会将 liquid 变量 page.author 设置为 Mr.Hyde。

|  |
| --- |
| collections:  my\_collection:  output: true  defaults:  -  scope:  path: ""  type: "my\_collection" # a collection in your site, in plural form  values:  layout: "default" |

在上面的例子中，文章中 type 设置为 my\_collection 的情况下其布局layout被设置为default。

## 头部信息defalut字段使用glob模式

在default节点中设置字段进行信息匹配的时候也可以使用glob模式（当前仅限使用 “\*”）。例如，将以section作为根目录的所有目录中的 special-page.html 文件都设置为制定布局可以使用如下代码（从 3.7.0 开始）：

|  |
| --- |
| collections:  my\_collection:  output: true  defaults:  -  scope:  path: "section/\*/special-page.html"  values:  layout: "specific-layout" |

**glog和性能**

请注意 glob 在路径中使用的时候会对性能产生负面影响，当前尚未修复，在Windows 中尤为明显。Glob之后构建的时间和所关联的路径的内容多少正相关。

## 优先权

Jekyll 将会应用你在 \_config.yml 中defaults节点指定的所有设置。然而你可以在其他的 scope/values 中指定具体的路径从而覆盖之前的值。

从本段向上倒数第三段示例代码中。首先，我们设置所有页面的默认布局都是 my-site。然后，使用了另一个 scope/values ，设置 projects/ 路径下的所有文件布局为 project。

最后，如果你在 \_config.yml 文件中设置一个 defaults 节点，你依然可以在文章中通过指定 YAML Front Matter 的方式覆盖这些设置。例如：

|  |
| --- |
| # In \_config.yml |
| ...  defaults:  -  scope:  path: "projects"  type: "pages"  values:  layout: "project"  author: "Mr. Hyde"  category: "project"  ... |

|  |
| --- |
| # In projects/foo\_project.md |
| ---  author: "John Smith"  layout: "foobar"  ---  The post text goes here... |

当 projects/foo\_project.md被构建的时候，layout将被设置为 footbar 取代 project；author将被设置为John Smith取代Mr. Hyde。

## 默认配置

默认情况下 Jekyll 会执行如下配置。要替代这些默认值可以在配置文件或者命令行中指定新值。

**这里由两个kramdown选项不支持的值**

当前 remove\_block\_html\_tags 和 remove\_span\_html\_tags是不被 Jekyll 支持的，因为他们没有实际包含在kramdown HTML转换器中。

|  |
| --- |
| # Where things are  source: .  destination: ./\_site  collections\_dir: .  plugins\_dir: \_plugins  layouts\_dir: \_layouts  data\_dir: \_data  includes\_dir: \_includes  collections:  posts:  output: true  # Handling Reading  safe: false  include: [".htaccess"]  exclude: ["Gemfile", "Gemfile.lock", "node\_modules", "vendor/bundle/", "vendor/cache/", "vendor/gems/", "vendor/ruby/"]  keep\_files: [".git", ".svn"]  encoding: "utf-8"  markdown\_ext: "markdown,mkdown,mkdn,mkd,md"  strict\_front\_matter: false  # Filtering Content  show\_drafts: null  limit\_posts: 0  future: false  unpublished: false  # Plugins  whitelist: []  plugins: []  # Conversion  markdown: kramdown  highlighter: rouge  lsi: false  excerpt\_separator: "\n\n"  incremental: false  # Serving  detach: false  port: 4000  host: 127.0.0.1  baseurl: "" # does not include hostname  show\_dir\_listing: false  # Outputting  permalink: date  paginate\_path: /page:num  timezone: null  quiet: false  verbose: false  defaults: []  liquid:  error\_mode: warn  strict\_filters: false  strict\_variables: false  # Markdown Processors  rdiscount:  extensions: []  redcarpet:  extensions: []  kramdown:  auto\_ids: true  entity\_output: as\_char  toc\_levels: 1..6  smart\_quotes: lsquo,rsquo,ldquo,rdquo  input: GFM  hard\_wrap: false  footnote\_nr: 1  show\_warnings: false |

## Liquid选项

Liquid响应中的错误可以通过配置error\_mode节点指定。该节点的可选值为：

* lax — 忽略所有错误
* warn — 在控制台中为每个错误输出一个警告
* strict — 输出错误信息并且停止构建

（从3.8.0开始）你可以配置 Liquid 的渲染去捕获未分配的变量和不存在的过滤器通过设置 strict\_variables 和 strict\_filters 的值分别置为 true。

请注意，尽管error\_mode配置了Liquid的解析器，但strict\_variables和strict\_filters选项配置了Liquid的渲染器，因此是互斥的。

## Markdown 选项

Jekyll 支持的Markdown 渲染选项变量有额外的选项。

### Redcarpet

Rendcarpet 可以被 extensions 子项进行设置，extensions 的值是 string 数组。每一个字符串都应该是一个 Redcarpet::Markdown 的扩展类型；如果存在于数组中，它会将相应的扩展名设置为 true。

Jekyll持有两个特殊的Redcarpet扩展：

no\_fenced\_code\_blocks — 默认的，Jekyll设置 fenced\_code\_block 扩展（代码块通过三个反引号或者三个波浪线来界定）为true，这大概和 GitHub的积极采用有关。当使用 Jekyll 的时候Redcarpet的一般扩展无效；这时候你可以使用与之相反的扩展用于取消 fence code。

注意：你也可以制定一种语言用于高亮显示代码如下：

|  |
| --- |
| ```ruby  # ...ruby code  ``` |

略……

### 自定义Markdown程序

略……

## 增量再生

增量再生是一个商不可用的实验产品