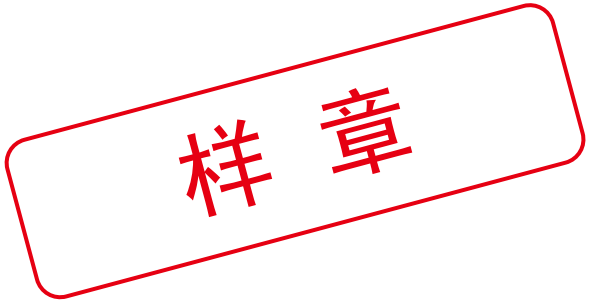
Jekyll小书

安道

Jekyll 小书

安道 著



rev 3.2.1.p1, 2016-08-04T12:00:00+08:00

目录

第 1 章 简介

这是一本关于 Jekyll（发音 /'dʒi:kil/）的书。我希望通过这本书告诉你 Jekyll 是什么，以及如何使用 Jekyll。首先，我们要弄明白 Jekyll 是什么。

1.1 Jekyll 是什么

1.1.1 小说中的人物

Jekyll 是英国新浪漫主义小说家罗伯特·路易斯·史蒂文森所著小说《化身博士》中的角色。《化身博士》讲述了体面绅士 Jekyll 博士喝了自己配制的药剂后化身邪恶的 Hyde 先生的故事。Jekyll 因抵挡不了潜藏在天性中邪恶、狂野因子的耸动，发明了一种药水，将平时被压抑在虚伪表相下的心性，毫无保留地展露出来，同时随着人格心性的转变，身材样貌也会随之改变。原本一个社会公众认为行善不遗余力的温文儒雅之士，一旦喝下药水，即转身一变，成为邪恶、毫无人性且人人憎恶的猥亵男子 Hyde。Jekyll 是善的代表，Hyde 则是恶魔的化身。后来，Jekyll 因控制不了自己内心的恶魔，以自尽的方法来终止自己以 Hyde 的身分作恶。

《化身博士》被多次改编为影视作品，最近一次是由 BBC 出品的 6 集迷你剧。

1.1.2 一个小岛

Jekyll 是美国东南沿海一带群岛中的一个小岛，隶属佐治亚州格林县。

1.1.3 静态网站生成工具

当然，本书要讲的并不是前面两种 Jekyll。我们要讲的 Jekyll 是一个静态网站生成工具（Static Site Generator）。Jekyll 这个名字取自《化身博士》，这一点从 Jekyll 以前的网站中可以窥见，如图 1.1 所示。

图 1.1：Jekyll 以前的网站

Jekyll 使用 Ruby 编程语言开发，用来生成纯静态网站。Jekyll 按照一定规则，把使用 Markdown、Textile、Asciidoc 等书写语言编写的文章转换成 HTML，最终组成一个完整的网站。

那么问题来了，为什么要搭建静态网站呢？

1.2 静态网站的好处

接触过博客的读者可能知道，搭建网站有很多知名的程序，例如 WordPress 和 Moveable Type，还有很多 CMS（Content Managerment System，内容管理系统）。为什么要搭建静态网站呢？

其实原因很多，我觉得比较重要的有两点。

历史诟病

WordPress 等程序出现的时间都很久了，甚至超过了十年。在这么长时间的开发中，架构再好的程序也难免有被人诟病的地方。就拿 WordPress 来说，其强大功能的背后，是性能的低下，为人所诟病。

名人效应

Jekyll 最初的开发者是 Tom Preston-Werner。他是 Gravatar（后来卖给运营 WordPress 的公司 Automattic）的创始人，而后又参与创建了 GitHub。这两个网站可以说在一定程度上改变了网络。很多人都是奔着 Tom 的名气才使用 Jekyll 的，这在一定程度上导致了静态网站工具的流行。Jekyll 出现之前，已经有一些静态网站生成工具，例如 nanoc。但在 Jekyll 之后，各种静态网站生成工具如雨后春笋般兴起，使用各种编程语言编写的都有。具体有多少，可以看一下 Static Site Generators 网站的统计。

抛开这两点，我们来看一下静态网站有什么好处：

- **安全性高**。因为静态网站中全是静态内容，服务器端不涉及任何动态语言，所以很少会被攻击。就算被攻击，网站本身也没什么损失。
- **性能高**。静态网站都是静态的文件，如 HTML 文件、CSS 样式表、JavaScript 脚本和各种图片，无需在服务器端处理，也不用连接数据库。
- **对服务器的要求低**。可以说，静态网站对服务器没有任何要求，不要求安装哪个具体版本的 PHP，不要求安装 MySQL 等数据库——只要服务器能伺服静态网页即可。而且，你甚至不需要自己购买服务器，可以把网站托管到 GitHub Pages 等平台中。
- **入门容易**。说得简单点儿，静态网站就是把数据套入模板后生成 HTML 文件，没有复杂的插件机制，模板系统也容易理解，对新手而言，入门非常容易。
- **便于做版本控制**。因为静态网站中所有内容都是以文件的形式存储，所以特别易于纳入版本控制系统，如 Git。
- **有极客范儿**。不知从何时起，“极客”（geek）这个词开始流行了。如果网站没使用静态网站生成工具搭建，“出门都不好意思跟人打招呼”。

1.3 我该不该使用静态网站生成工具

“动态”还是“静态”完全是个人喜好。如果你觉得自己更适合使用 WordPress，那就继续使用。如果你想体验一下新鲜事物，可以试一下静态网站生成工具，说不定会爱上她。

如果你恰好是 Ruby 程序员，或者对 Ruby 感兴趣，那么 Jekyll 定是不二选择。

决定之后，下面开始安装 Jekyll 吧。

1.4 预备知识

这是一本面向初学者的书。我说的“初学者”是指刚接触 Jekyll。Jekyll 并不是“零基础”，你需要具备一些基础知识。具体而言，你要掌握：

- 基本的命令行知识，至少要知道什么是命令行，怎么在命令行中执行命令；
- 基本的 HTML 和 CSS 知识，如果懂一点 JavaScript 更好；
- 如果想自己编写插件，或者想深入理解 Jekyll 的运作机制，需要一定的 Ruby 知识；

1.5 排版约定

本书没有使用特殊的排版方式，相信多数情况下你都能理解，不过还是要做些说明。

等宽字体

表示代码，文件和文件夹名称。

斜体字

表示 URL。

$ 符号

表示命令行提示符。如果你在代码中看到 `$` 符号，说明随后的内容是在命令行中执行的命令。如果 `$` 符号之前有内容，例如 `~/blog`，说明随后的命令要在指定的文件夹中执行。

1.6 随书源码

书中用到的代码托管在 GitHub 中，地址为 https://github.com/jekyll-book。各章的代码放在单独的仓库中，例如第 3 章的代码在 jekyll-book/chapter03 仓库中。

1.7 反馈

如果你在阅读本书的过程中发现了错误，或者觉得有些地方没讲清楚，或者想知道其他的用法和功能，或者有好的建议，都可以反馈给我。你可以到本书的讨论页面发帖，我会尽快回复。也可以直接发电子邮件给我，我的邮箱地址是 andor.chen.27@gmail.com。

本书的官方网站：http://jekyll-china.com。

1.8 致谢

感谢 Tom 和 Jekyll 开发团队开发了这个灵活好用的静态网站生成工具。感谢我的妻子忍受我在写这本书的过程中敲击键盘发出的声音。感谢给我提供反馈的各位读者。感谢所有购买这本书的读者。

1.9 版权

第 2 章 安装 Jekyll

Jekyll 是使用 Ruby 编程语言开发的程序，以 gem [1] 的形式分发，所以在安装 Jekyll 之前，要先在系统中安装 Ruby。

2.1 安装 Ruby

Mac OS X 和某些基于 Linux 的系统中都已经预装了 Ruby。如果不确定自己的系统中是否安装有 Ruby，可以执行下面的命令检查。在系统的命令行中输入以下命令：

```
$ ruby -v
```

如果看到 Ruby 的版本信息，说明系统中已经安装了 Ruby。如果提示找不到命令，说明系统中没有安装 Ruby，需要自己动手安装。

就算系统中已经安装 Ruby，版本也可能太旧。如果看到的版本号小于 2.0.0，需要安装最新版。

具体怎么安装 Ruby 已经超出了本书范畴，你可以参考 InstallRails 网站中的说明。这个网站说明了如何在 Windows、Mac OS X，以及流行的 Linux 发行版中安装 Ruby。后面介绍如何安装 Rails 的步骤可以忽略，不过也可以看一下，因为 Jekyll 的安装方式和 Rails 基本一样。

使用国内镜像安装 Ruby

由于国内特殊的网络环境，安装 Ruby 时可能发现速度很慢，或者根本无法安装。这时，可以选择使用国内的镜像。

从源码编译安装 Ruby 的用户，可以直接从 Ruby China 的镜像中下载相应的版本。使用 rbenv 管理 Ruby 的用户，可以使用笔者开发的 rbenv-china-mirror 插件。

---

1. gem 是 Ruby 语言的包管理和包分发工具，详情参见 http://rubygems.org。

2.2 安装 Jekyll

Jekyll 以 gem 的形式分发，安装起来十分简单。确认系统中有 Ruby 之后，在命令行中执行以下命令即可：

```
$ gem install jekyll
```

在某些系统中，执行这个命令时可能需要使用 `sudo`。上述命令会在系统中安装最新版 Jekyll，目前，最新版是 3.2.1。安装完成后，在命令行中执行以下命令，检查 Jekyll 是否正确安装：

```
$ jekyll -v
jekyll 3.2.1
```

如果输出了 Jekyll 的版本号，说明安装是成功的。

使用国内镜像安装 gem

由于国内特殊的网络环境，安装 gem 时可能发现速度很慢，或者根本无法安装。这时，可以选择使用国内的镜像。设置方法如下：

```
$ gem sources --add https://gems.ruby-china.org/ --remove
https://rubygems.org/
$ gem sources -l
https://gems.ruby-china.org
# 确保只有 gems.ruby-china.org
```

第 3 章 创建第一个网站

安装好 Ruby 和 Jekyll 之后，我们该创建网站了。不过在此之前，我们要知道如何使用 Jekyll 的命令行界面。

3.1 命令行界面

Jekyll 是命令行工具，几乎所有操作都在命令行中完成，没有图形化界面（GUI），也没有网页界面（例如 WordPress 的后台）。所以你至少要知道怎么在命令行中运行命令。

在命令行中执行 `jekyll help` 命令，查看帮助信息：

```
$ jekyll _3.2.1_ help

jekyll 3.2.1 -- Jekyll is a blog-aware, static site generator in Ruby

Usage:

  jekyll <subcommand> [options]

Options:
        ...
        ...
        ...

Subcommands:
  docs
  import
  build, b              Build your site
  clean                 Clean the site ...
  doctor, hyde          Search site and print specific deprecation warnings
  help                  Show the help message, optionally for a given subcommand.
  new                   Creates a new Jekyll site scaffold in PATH
```

```
new-theme             Creates a new Jekyll theme scaffold
serve, server, s      Serve your site locally
```

现在我们只关注“Subcommands”（子命令）部分，其他内容先不看。

Jekyll 默认提供了 9 个子命令，各个命令的作用如表 3.1 所示。

表 3.1：Jekyll 子命令及其作用

子命令	作用
`docs`	在本地查看 Jekyll 文档，必须先安装 `jekyll-docs` 插件
`import`	把其他博客程序中的内容导入 Jekyll 网站，必须先安装 `jekyll-import` 插件
`build` 或 `b`	构建网站
`clean`	清理网站（删除生成的网站和元信息文件）
`doctor` 或 `hyde`	扫描整个网站，查找是否用到了弃用的功能，如果有，显示提醒消息
`help`	显示帮助信息
`new`	新建一个 Jekyll 网站
`new-theme`	新建一个 Jekyll 主题（3.2 版新增）
`serve`，`server` 或 `s`	启动本地服务器，预览网站

现在，系统中已经安装好了 Jekyll，下面我们来新建一个网站。根据上表，我们需要使用的子命令是 `new`。

3.2 新建网站

在命令行中执行以下命令，新建一个 Jekyll 网站：

```
$ jekyll _3.2.1_ new /path/to/site
```

传给 `new` 命令的参数是保存网站的文件夹路径。假如我们要在家目录中的 `blog` 文件夹中新建一个网站，可以执行下述命令：

```
$ cd ~                          # 进入家目录
~ $ jekyll _3.2.1_ new blog   # 新建网站
New jekyll site installed in ~/blog.
```

上述命令在指定的文件夹中创建网站的骨架结构。如果 `blog` 文件夹已经存在，而且其中有内容，这个命令会提醒“`Conflict: ~/blog exists and is not empty`”。如果一定要在已经有内容的文件夹中创建 Jekyll 项目，可以使用 `-f` 旗标：

```
~ $ jekyll _3.2.1_ new blog -f
New jekyll site installed in ~/blog.
```

注意

执行 `jekyll help` 和 `jekyll new` 命令时，我们在命令行中指定了 Jekyll 的版本号。这么做是为了使用指定版本的 Jekyll。因为执行这两个命令时没有 Bundler 环境（3.3 节），无法限定 Jekyll 的版本，所以我们直接在命令行中指定版本号。

3.3 网站的文件结构

现在，进入 `~/blog` 文件夹，我们来了解一下 Jekyll 网站的文件结构。

```
~/blog
├── _posts/
├── css/
├── .gitignore
├── _config.yml
├── about.md
├── feed.xml
├── Gemfile
└── index.html
```

各个文件夹和文件的作用如表 3.2 所示。

表 3.2：Jekyll 网站的文件结构

文件（夹）	作用
`_posts/`	存放博客文章文件
`css/`	存放样式表

`.gitignore`	把指定的文件排除在 Git 仓库之外
`_config.yml`	网站的配置文件
`about.md`	“关于”页面
`feed.xml`	网站的订阅源文件
`Gemfile`	列出网站使用的 gem 和 Jekyll 插件（即依赖）
`index.html`	网站的入口文件，即首页

你可以在文本编辑器中打开各个文件，看看里面的内容。本书后文会分别讲解各个文件（夹）。

其中需要特别说明的是 `Gemfile` 文件。这个文件用于列出网站使用的 gem 和 Jekyll 插件（也以 gem 的形式分发），供 Bundler 使用。Bundler 是依赖管理工具，在 Ruby 社区中广泛使用。Bundler、`Gemfile` 文件（以及即将提到的 `Gemfile.lock` 文件）组合在一起能确保不同人使用的项目依赖始终相同，从而避免因版本不同而导致的问题。

Bundler 本身也是一个 gem，因此要执行下述命令安装：

```
$ gem install bundler
```

执行 `jekyll new` 命令生成网站后，还要运行 Bundler，让它读取 `Gemfile` 文件中声明的依赖，安装它们，然后生成 `Gemfile.lock` 文件。注意，我们执行的命令是 `bundle`，而不是 `bundler`，后面没有“r”。

```
~ $ cd blog
~/blog $ bundle install
Fetching gem metadata from https://gems.ruby-china.org/.............
Fetching version metadata from https://gems.ruby-china.org/...
Fetching dependency metadata from https://gems.ruby-china.org/..
Resolving dependencies...
...
...
...
Bundle complete! 2 Gemfile dependencies, 18 gems now installed.
Use `bundle show [gemname]` to see where a bundled gem is installed.
```

让 Bundler 使用国内镜像

由于国内特殊的网络环境，安装 gem 时可能发现速度很慢，或者根本无法安装。这时，可以选择使用国内的镜像。使用 Bundler 安装 gem 时，可以执行下述命令，设为 Ruby China 的镜像：

```
$ bundle config mirror.https://rubygems.org https://gems.ruby-china.org
```

当然，也可以直接把 `Gemfile` 文件中的 `source` 改成镜像地址。

3.4 查看网站

新建网站项目后，你可能很心急，想看一下网站是什么样子。没问题，在命令行中执行下述命令，启动本地服务器：

```
~/blog $ bundle exec jekyll server
Configuration file: ~/blog/_config.yml
            Source: ~/blog
       Destination: ~/blog/_site
 Incremental build: disabled. Enable with --incremental
      Generating...
                    done in 0.672 seconds.
 Auto-regeneration: enabled for '~/blog'
Configuration file: ~/blog/_config.yml
    Server address: http://127.0.0.1:4000/
  Server running... press ctrl-c to stop.
```

注意，我们在 `jekyll` 命令前加上了 `bundle exec`，这么做是为了使用 `Gemfile` 文件中指定的 Jekyll 版本。

然后，打开浏览器，在地址栏中输入 *http://localhost:4000*，就能看到网站的首页了，如图 3.1 所示。

如果想关闭本地服务器，在命令行中按 `Ctrl-C` 键。

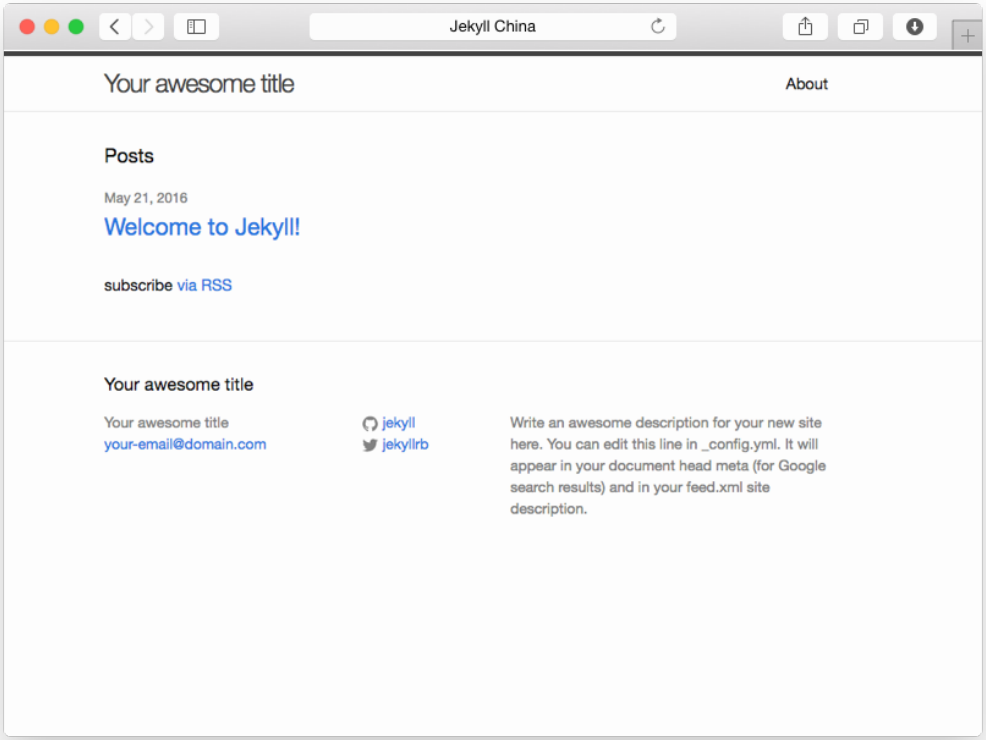


图 3.1：网站的首页

3.5 设置网站

仔细观察首页，我们发现有很多信息是占位用的，例如网站的标题和描述信息等，需要根据需求修改。首页的右下角有一段话，从中得知，这些占位信息在 `_config.yml` 文件中。

在文本编辑器中打开 `_config.yml` 文件。这是个 YAML 文件，网站所有的配置都保存在这里。文件中以 `#` 号开头的行是注释，通过 Jekyll 生成的这些注释我们可以初步了解这个文件的作用。

你可以根据自己的需求修改网站的标题（`title`）、电子邮件地址（`email`）、Twitter 用户名（`twitter_username`）和 GitHub 用户名（`github_username`）等。例如，我修改后的设置如下：

```
title: Andor Chen
email: andor.chen.27@gmail.com
...
...
url: "http://jekyll-china.com"
twitter_username: andor_chen
github_username: andorchen
```

修改完这些设置之后，我们来看一下生成的网站。进入 `_site` 文件夹，你会看到很多文件（夹），其中有两个文件并不是网站的一部分，即 `Gemfile` 和 `Gemfile.lock`。为了不让这两个文件出现在生成的网站中，我们要使用 `exclude` 设置。在 `_config.yml` 文件中添加下述设置：

```
exclude: ["Gemfile", "Gemfile.lock"]
```

修改设置后，要重启本地服务器才能生效：先按 `Ctrl-C` 键关闭本地服务器，然后执行 `bundle exec jekyll server` 命令重启。

第 4 章 文章

在本地预览网站时可以看到，新建的网站中已经有了一篇文章，就像 WordPress 网站中的第一篇文章“Hello, world!”一样。点击“Welcome to Jekyll!”，打开这篇文章，阅读其中的内容。

但是，这篇文章在哪儿？如果想发布新文章该怎么做呢？本章为你解答这些疑问。

4.1 文章文件的组成

你应该还记得网站的文件结构（3.3 节）中有一个名为 `_posts` 的文件夹，打开它，可以看到第一篇文章对应的文件。我的文件名是 `2016-05-21-welcome-to-jekyll.markdown`，你的文件名可能和我不一样，具体来说，是数字部分不一样。

文章对应的文件，其文件名由三部分组成：日期，别名（slug）和扩展名。对默认生成的第一篇文章来说，日期是“2016-05-21”，别名是“welcome-to-jekyll”，扩展名是“.markdown”。日期要满足特定的条件：年必须使用四位数字表示，例如 `2016`；月必须使用两位数字表示，例如 `02`，`11`；日也必须使用两位数字表示，例如 `03`，`27`。别名就是在浏览器中显示的这篇文章的 URL，一般使用英文，单词之间使用“-”（连字符）连接。扩展名表示这篇文章使用什么书写语言，“`.markdown`”表示 Markdown。

图 4.1：文章文件名的构成

在文本编辑器中打开这个文件，看一下里面的内容。文章对应的文件由两部分组成：头部元信息（frontmatter）和正文。

元信息包围在两行 `---`（三个连字符）之间，使用 YAML 格式定义。在默认生成的第一篇文章中，元信息为：

```
---
layout: post
```

```
title:  "Welcome to Jekyll!"
date:   2016-05-21 15:32:06 +0800
categories: jekyll update
---
```

其中：

- `layout` 指定文章使用哪个布局渲染；
- `title` 是文章的标题；
- `date` 是文章的发布日期、时间和时区；
- `categories` 是文章所属的分类，多个分类之间使用空格或英文逗号分开。

默认生成的第一篇文章使用 Markdown 句法编写。Markdown 句法的简介参见附录 B。

4.2 发布新文章

了解文章的基本结构之后，我们来发布一篇新文章。如果你觉得默认生成的第一篇文章没用，可以把它删掉。在 `_posts` 文件夹中新建一个文件，文件名随意取，能表明文章的主旨即可，但要符合前面所讲的格式。例如，`2016-05-21-first-post.markdown`。

在文本编辑器中打开新建的文件，写入元信息。例如：

```
---
layout: post
title:  "我的第一篇文章"
date:   2016-05-21 16:45:27 +0800
categories: news
---
```

然后，把你想发布的内容用 Markdown 句法写出来。写完后，在命令行中执行下述命令，启动本地服务器（同时也会生成网站）：

```
~/blog $ bundle exec jekyll server
```

提示

你可能觉得每次启动本地服务器都输入 `server` 子命令有点麻烦，想知道有没有快捷方式。当然有，`server` 的简写是 `s`，如表 3.1 所示。因此，启动本地服务器时，为了少输入几个字符，可以执行 `jekyll s` 命令。

然后在浏览器中打开 *http://localhost:4000*，就能看到这篇文章出现在首页中。点击首页中这篇文章的标题，即可打开这篇文章。

这就是日常发布新文章的流程：在 `_posts` 文件夹中新建一个文件，写入元信息和内容，启动本地服务器预览。当然，后面还有一步——部署，不过现在先不管，[deploying#deploying]再介绍。

4.3 新建文章的 Rake 命令

如果经常发布文章，你会觉得每次都自己动手新建文件，再写入元信息有点麻烦。有没有办法可以把这个操作交给电脑自动完成呢？[1] 答案是肯定的，我们可以把这个操作交给 rake 任务完成。

rake 是什么

简单来说，rake [2] 是 Ruby 界的 make，一种构件和自动化工具。rake 定义了一套简单的领域特定语言（Domain-Specific Language，简称 DSL），简化了任务的编写。使用 rake 任务可以自动完成一些操作。这里，我们要让 rake 帮我们新建文件，并在文件中写入一些默认的内容。

rake 也是以 gem 的形式分发的，我们将其添加到 `Gemfile` 文件中：

```
...
gem "jekyll", "3.2.1"
```

1. 人和电脑都有自己存在的意义。电脑出现的目的之一是，帮助人类完成重复的操作。不要心存疑虑，这就是电脑的使命。能交给电脑完成的操作绝不要自己动手。

2. 2014 年 2 月，rake 的核心开发者 Jim Weirich 不幸离世，他是很活跃的 Ruby 开发者，开发了很多 Ruby 程序员常用的程序。R.I.P

```
gem "rake"
...
```

然后，在命令行中执行 `bundle install` 命令，安装 rake。

rake 任务在一个特殊的文件中定义，名为 `Rakefile`。下面，在网站的根目录中新建文件 `Rakefile`，然后在文本编辑器中打开，写入下述内容：

代码清单 4.1：新建文章的 rake 任务

```
require 'time'

# Usage: rake post title="A Title" [date="2014-04-14"]
desc "Create a new post"
task :post do
  unless FileTest.directory?('./_posts')
    abort("rake aborted: '_posts' directory not found.")
  end
  title = ENV["title"] || "new-post"
  slug = title.downcase.strip.gsub(' ', '-').gsub(/[^\w-]/, '')
  begin
    datetime = (ENV['date'] ? Time.parse(ENV['date']) : Time.now)
                                  .strftime('%Y-%m-%d %H:%M:%S %z')
    date = datetime.split.first
  rescue Exception => e
    puts "Error: date format must be YYYY-MM-DD!"
    exit -1
  end
  filename = File.join('.', '_posts', "#{date}-#{slug}.md")
  if File.exist?(filename)
    abort("rake aborted: #{filename} already exists.")
  end

  puts "Creating new post: #{filename}"
  open(filename, 'w') do |post|
    post.puts "---"
    post.puts "layout: post"
    post.puts "title: \"#{title.gsub(/-/,' ')}\""
    post.puts "date: #{datetime}"
```

```
    post.puts "categories:"
    post.puts "---"
  end
end
```

Rakefile 使用 Ruby 语言编写。上述代码定义了一个 rake 任务，名为 post。这个任务的作用是，在 _posts 文件夹中新建一个文件，写入一些默认内容。我们来看一下如何执行这个任务。

```
~/blog $ bundle exec rake post title="Hello, again"
Creating new post: ./_posts/2016-05-21-hello-again.md

# date 选项是可选的
~/blog $ bundle exec rake post title="Hello, there" date="2016-05-21"
Creating new post: ./_posts/2016-05-21-hello-there.md
```

你可以自己执行试一下，然后进入 _posts 文件夹确认有没有生成文件，再打开新建的文件，看看其中是不是有元信息。

细心的读者可能发现了，新建的文件扩展名为 .md，和默认生成的第一篇文章不一样。这没关系，Jekyll 也会把扩展名为 .md 的文件识别为使用 Markdown 句法编写的文章。除此之外，Jekyll 默认能识别的 Markdown 文件扩展名还有 .mkdown、.mkdn 和 .mkd。你可以根据自己的喜好选择使用某个扩展名。不过简单起见，一般都使用 .md。

提示

如 3.5 节所述，为了不让 Rakefile 文件出现在生成的网站中，我们要把它添加到 exclude 设置中：

```
exclude: ["Gemfile", "Gemfile.lock", "Rakefile"]
```

4.4 文章元信息的默认值

有了前面的 rake 任务，创建新文章的过程大大简化了，不过还有一个问题。如果你是程序员，一定知道 DRY 原则，即不要重复表述相同的信息。在文章的元信息中就有信息重复——布局。一般情况下，所有文章都会使用相同的布局渲染，偶尔可能会使用特别的布局渲染一两篇特殊的文章。既然如此，为什么每篇文章都指定相同的布局信息呢？！Jekyll 允许我们为元信息设置默认值，从而避免这个问题。

打开 _config.yml 文件，写入下述设置：

```
defaults:
  -
    scope:
      path: ""
      type: "posts"
    values:
      layout: "post"
```

保存文件，然后重启本地服务器。打开一篇文章，去掉元信息中的 `layout` 设置。你会发现，那篇文章仍会使用 `post` 布局渲染。

4.5 文章的固定链接

在浏览器中打开一篇文章后，注意地址栏中的地址。可以看出，Jekyll 默认使用的固定链接格式是“分类-日期-别名”，这种格式的代号是 `date`。`date` 只是一个代号，具体的内容是：`/:categories/:year/:month/:day/:title.html`。除了 `date` 之外，Jekyll 还内置了另外两种固定链接格式：

- `pretty` → `/:categories/:year/:month/:day/:title/`
- `none` → `/:categories/:title.html`

固定链接的格式在 `_config.yml` 文件中设置。如果不想使用默认的固定链接格式，可以在这个文件中设置。在文本编辑器中打开 `_config.yml`，新起一行写入如下内容：

```
permalink: 'pretty'
```

保存 `_config.yml` 文件。如果已经启动本地服务，在命令行中按 `Ctrl-C` 键关闭服务器，然后再执行 `jekyll s` 命令重启服务器。在浏览器中打开网站首页，点击某篇文章的标题，留意地址栏中显示的地址，会发现格式变了：后面的 `.html` 没了。

如果 Jekyll 内置的固定链接格式无法满足你的需求，还可以自定义。方法一样，只不过不再使用内置的代号，我们要使用固定链接变量。

如果只想显示标题，可以这样设置：

```
permalink: '/:title.html'
```

如果还想包括年份，可以这样设置：

```
permalink: '/:year/:title.html'
```

如果不想要后面的 .html，可以这样设置：

```
permalink: '/:title/'
```

> **注意**
>
> 修改 _config.yml 文件之后必须重启本地服务器，设置才能生效。还记得方法吗？先按 Ctrl-C 键，然后执行 bundle exec jekyll s 命令。

Jekyll 支持很多固定链接变量，如表 4.1 所示，你可以根据自己的需求灵活使用。

表 4.1：Jekyll 支持的固定链接变量

变量	说明	举例
year	从文章文件名中读取的年份，四位数字	1999, 2014
short_year	从文章文件名中读取的年份，只有后两位数字	99, 14
month	从文章文件名中读取的月份，两位数字	03, 11
i_month	从文章文件名中读取的月份，没有前导零	3, 11
day	从文章文件名中读取的日，两位数字	05, 27
i_day	从文章文件名中读取的日，没有前导零	5, 27
title	从文章文件名中读取的标题	first-post
slug	与 title 类似，不过字母和数字之外的字符会被替换成连字符	first-post
categories	文章所属的分类	jekyll update

注意，设置固定链接时，变量前面一定要加上 :（英文冒号），否则会被识别为纯文本，不会被 Jekyll 代换。

4.6 草稿

如果想记录一件事，但一时半会儿写不完，不想发布，那么可以先把文章保存为草稿。Jekyll 支持草稿功能。草稿保存在网站根目录中的 `_drafts` 文件夹里。执行 `jekyll new` 命令新建项目时，没有生成这个文件夹，因此需要自己动手创建。

打开命令行，进入网站所在的根目录，执行下述命令，创建 `_drafts` 文件夹：

```
~/blog $ mkdir _drafts
```

注意

新建文件夹的操作也可以在 GUI（图形界面）中完成，但是为了熟悉命令行操作，建议你在命令行中完成。书中的操作，请尽量都在命令行中完成。用得多了，终有一天你会克服对命令行的恐惧。

草稿和文章相比，除了存放位置不一样之外，文件的标题也不一样——草稿文件的文件名中没有日期。其他都和文章一样。

如果草稿写好了，决定发布，可以把草稿文件移到 `_posts` 文件夹中，然后在文件名中加入发布的日期。

在本地预览时，如果想查看草稿，要指定 `--drafts` 选项，例如 `bundle exec jekyll s --drafts`。

如果你感兴趣，可以参照前面新建文章的 rake 命令，自己试着编写一个 rake 任务，用来新建草稿。如果你对 Ruby 语言不熟悉，可以直接把下面的代码添加到 `Rakefile` 文件中。

代码清单 4.2：新建草稿的 rake 任务

```
# Usage: rake draft title="A Title"
desc "Create a new draft"
task :draft do
  unless FileTest.directory?('./_drafts')
    abort("rake aborted: '_drafts' directory not found.")
  end
  title = ENV["title"] || "new-post"
  slug = title.downcase.strip.gsub(' ', '-').gsub(/[^\w-]/, '')
  filename = File.join('.', '_drafts', "#{slug}.md")
  if File.exist?(filename)
    abort("rake aborted: #{filename} already exists.")
```

```
    end

    puts "Creating new draft: #{filename}"
    open(filename, 'w') do |post|
      post.puts "---"
      post.puts "layout: post"
      post.puts "title: \"#{title.gsub(/-/,' ')}\""
      post.puts "date:"
      post.puts "categories:"
      post.puts "---"
    end
  end
```

这个 rake 任务的用法和新建文章的任务类似，但不用指定 `date` 参数：

```
~/blog $ bundle exec rake draft title="A fake draft"
```