

搭建一个免费的，无限流量的Blog---github Pages

和Jekyll入门

作者： 阮一峰

日期： 2012年8月25日

喜欢写Blog的人，会经历三个阶段。

第一阶段，刚接触Blog，觉得很新鲜，试着选择一个免费空间来写。

第二阶段，发现免费空间限制太多，就自己购买域名和空间，搭建独立博客。

第三阶段，觉得独立博客的管理太麻烦，最好在保留控制权的前提下，让别人来管，自己只负责写文章。

大多数Blog作者，都停留在第一和第二阶段，因为第三阶段不太容易到达：你很难找到俯首听命、愿意为你管理服务的人。



但是两年前，情况出现变化，一些程序员开始在[github](https://github.com)网站上搭建blog。他们既拥有绝对管理权，又享受github带来的便利----不管何时何地，只要向主机提交commit，就能发布新文章。更妙的是，这一切还是免费的，github提供无限流量，世界各地都有理想的访问速度。

今天，我就来示范如何在github上搭建Blog，你可以从中掌握github的Pages功能，以及Jekyll软件的基本用法。更重要的是，你会体会到一种建立网站的全新思路。



github
SOCIAL CODING

一、Github Pages 是什么？

如果你对编程有所了解，就一定听说过[github](https://github.com)。它号称程序员的Facebook，有着极高的人气，许多重要的项目都托管在上面。

简单说，它是一个具有版本管理功能的代码仓库，每个项目都有一个主页，列出项目的源文件。

🕒 Latest commit to the **master** branch

Update Sizzle: passing null to \$.contains should not throw an error. ...



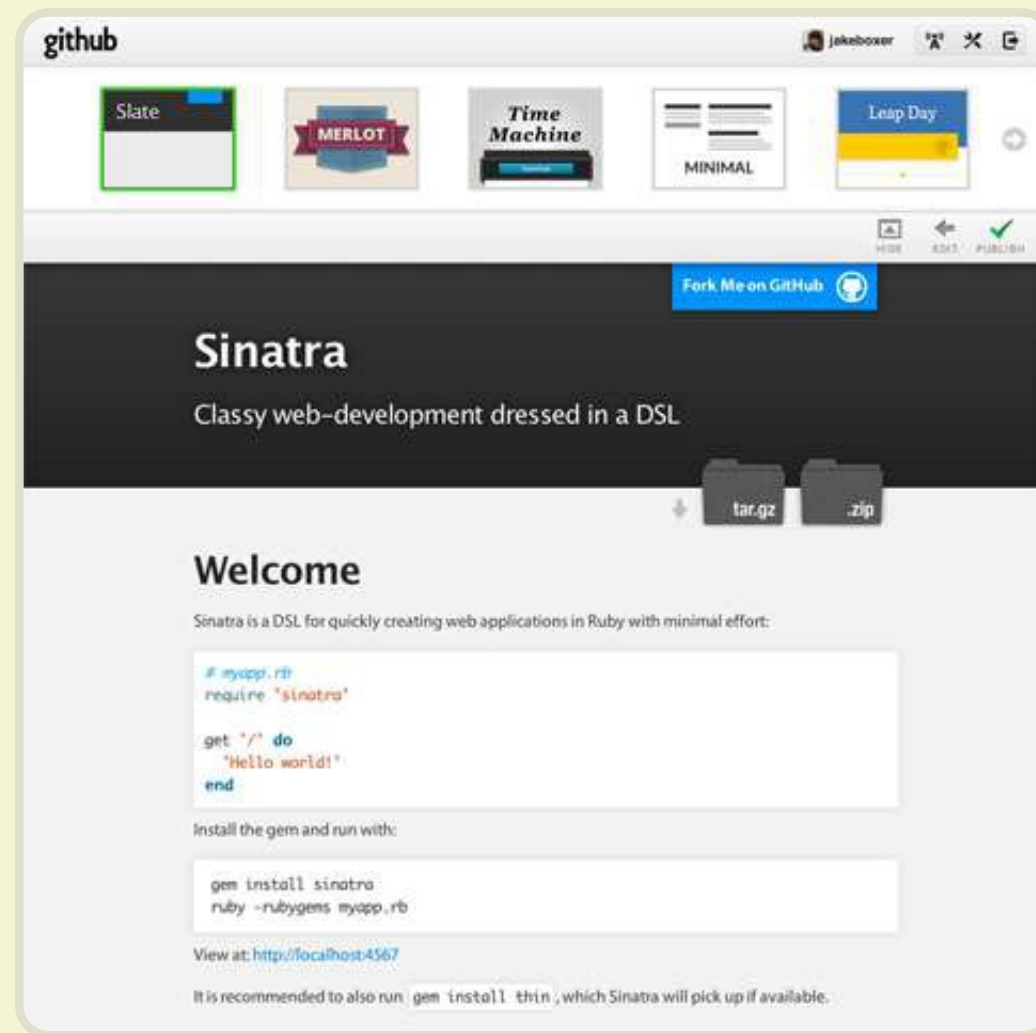
timmywil authored 10 hours ago

jquery /

name	age	message
📁 build	15 days ago	Don't let the release notes script truncate!
📁 speed	a month ago	Fixes a variety of typographical problems.
📁 src	10 hours ago	Update Sizzle: passing null to \$.contains s
📁 test	16 hours ago	Revert "Fix #11586. Ajax DELETE ain't go
📄 .editorconfig	3 months ago	Fix #11777. Add EditorConfig support, clo
📄 .gitattributes	2 years ago	Mark *.jar files as binary. [hwiechers]
📄 .gitignore	3 months ago	Allow users to store custom dist destinatio
📄 .gitmodules	a year ago	Update the Sizzle location to point to the n

但是对于一个新手来说，看到一大堆源码，只会让人头晕脑涨，不知何处入手。他希望看到的是，一个简明易懂的网页，说明每一步应该怎么做。因此，**github**就设计了 [Pages功能](#)，允许用户自定义项目首页，用来替代默认的源码列表。所以，**github**

Pages可以被认为是用户编写的、托管在**github**上的静态网页。



github提供模板，允许站内生成网页，但也允许用户自己编写网页，然后上传。有意思的是，这种上传并不是单纯的上传，而是会经过Jekyll程序的再处理。

二、**Jekyll**是什么？

Jekyll（发音/'dʒi:k əl/，"杰克尔"）是一个静态站点生成器，它会根据网页源码生成静态文件。它提供了模板、变量、插件等功能，所以实际上可以用来编写整个网站。



整个思路到这里就很明显了。你先在本地编写符合**Jekyll**规范的网站源码，然后上传到**github**，由**github**生成并托管整个网站。

这种做法的好处是：

- * 免费，无限流量。
- * 享受git的版本管理功能，不用担心文章遗失。
- * 你只要用自己喜欢的编辑器写文章就可以了，其他事情一概不用操心，都由github处理。

它的缺点是：

- * 有一定技术门槛，你必须懂一点git和网页开发。
- * 它生成的是静态网页，添加动态功能必须使用外部服务，比如评论功能就只能用[disqus](#)。
- * 它不适合大型网站，因为没有用到数据库，每运行一次都必须遍历全部的文本文件，网站越大，生成时间越长。

但是，综合来看，它不失为搭建中小型Blog或项目主页的最佳选项之一。

三、一个实例

下面，我举一个实例，演示如何在github上搭建blog，你可以跟着一步步做。为了便于理解，这个blog只有最基本的功能。

在搭建之前，你必须已经安装了[git](#)，并且有[github](#)账户。

第一步，创建项目。

在你的电脑上，建立一个目录，作为项目的主目录。我们假定，它的名称为 `jeekyll_demo`。

```
$ mkdir jeekyll_demo
```

对该目录进行git初始化。

```
$ cd jeekyll_demo
```

```
$ git init
```

然后，创建一个没有父节点的分支`gh-pages`。因为[github](#)规定，只有该分支中的页面，才会生成网页文件。

```
$ git checkout --orphan gh-pages
```

以下所有动作，都在该分支下完成。

第二步，创建设置文件。

在项目根目录下，建立一个名为`_config.yml`的文本文件。它是`jeekyll`的设置文件，我们在里面填入如下内容，其他设置都可以用默认选项，具体解释参见[官方网页](#)。

```
baseurl: /jeekyll_demo
```

目录结构变成：

```
/jeekyll_demo  
|-- _config.yml
```

第三步，创建模板文件。

在项目根目录下，创建一个`_layouts`目录，用于存放模板文件。

```
$ mkdir _layouts
```

进入该目录，创建一个`default.html`文件，作为Blog的默认模板。并在该文件中填入以下内容。

```
<!DOCTYPE html>

<html>

<head>

    <meta http-equiv="content-type" content="text/html;
charset=utf-8" />

    <title>{{ page.title }}</title>

</head>

<body>

    {{ content }}

</body>

</html>
```

Jekyll使用[Liquid模板语言](#)，`{{ page.title }}`表示文章标题，`{{ content }}`表示文章内容，更多模板变量请参考[官方文档](#)。

目录结构变成：

```
/jekyll_demo
|-- _config.yml
|-- _layouts
|   |-- default.html
```

第四步，创建文章。

回到项目根目录，创建一个_posts目录，用于存放blog文章。

```
$ mkdir _posts
```

进入该目录，创建第一篇文章。文章就是普通的文本文件，文件名假定为**2012-08-25-hello-world.html**。（注意，文件名必须为"年-月-日-文章标题.后缀名"的格式。如果网页代码采用html格式，后缀名为html；如果采用[markdown](#)格式，后缀名为md。）

在该文件中，填入以下内容：（注意，行首不能有空格）

```
---
layout: default
title: 你好，世界
---
```

```
<h2>{{ page.title }}</h2>

<p>我的第一篇文章</p>

<p>{{ page.date | date_to_string }}</p>
```

每篇文章的头部，必须有一个[yaml文件头](#)，用来设置一些元数据。它用三根短划线 "---"，标记开始和结束，里面每一行设置一种元数据。"layout: default"，表示该文章的模板使用 `_layouts` 目录下的 `default.html` 文件；"title: 你好，世界"，表示该文章的标题是"你好，世界"，如果不设置这个值，默认使用嵌入文件名的标题，即"hello world"。

在yaml文件头后面，就是文章的正式内容，里面可以使用模板变量。{{ page.title }}就是文件头中设置的"你好，世界"，{{ page.date }}则是嵌入文件名的日期（也可以在文件头重新定义date变量），"| date_to_string"表示将page.date变量转化成人类可读的格式。

目录结构变成：

```
/jekyll_demo
|-- _config.yml
|-- _layouts
|   |-- default.html
|-- _posts
|   |-- 2012-08-25-hello-world.html
```

第五步，创建首页。

有了文章以后，还需要有一个首页。

回到根目录，创建一个index.html文件，填入以下内容。

```
---
layout: default
title: 我的Blog
---

<h2>{{ page.title }}</h2>

<p>最新文章</p>

<ul>

    {% for post in site.posts %}

        <li>{{ post.date | date_to_string }} <a href="
{{ site.baseurl }}{{ post.url }}">{{ post.title }}</a>
</li>

    {% endfor %}
```

```
</ul>
```

它的Yaml文件头表示，首页使用**default**模板，标题为"我的Blog"。然后，首页使用了`{% for post in site.posts %}`，表示对所有帖子进行一个遍历。这里要注意的是，Liquid模板语言规定，输出内容使用两层大括号，单纯的命令使用一层大括号。至于`{{site.baseurl}}`就是`_config.yml`中设置的`baseurl`变量。

目录结构变成：

```
/jekyll_demo
|-- _config.yml
|-- _layouts
|   |-- default.html
|-- _posts
|   |-- 2012-08-25-hello-world.html
|-- index.html
```

第六步，发布内容。

现在，这个简单的Blog就可以发布了。先把所有内容加入本地git库。

```
$ git add .
```

```
$ git commit -m "first post"
```

然后，前往github的网站，在网站上创建一个名为jekyll_demo的库。接着，再将本地内容推送到github上你刚创建的库。注意，下面命令中的username，要替换成你的username。

```
$ git remote add origin  
https://github.com/username/jekyll_demo.git  
  
$ git push origin gh-pages
```

上传成功之后，等10分钟左右，访问

http://username.github.com/jekyll_demo/就可以看到Blog已经生成了（将username换成你的用户名）。

首页：

我的Blog

最新文章

- 25 Aug 2012 [你好，世界](#)

文章页面：

你好，世界

我的第一篇文章

25 Aug 2012

第七步，绑定域名。

如果你不想用http://username.github.com/jekyll_demo/这个域名，可以换成自己的域名。





具体方法是在repo的根目录下面，新建一个名为CNAME的文本文件，里面写入你要绑定的域名，比如example.com或者xxx.example.com。

如果绑定的是顶级域名，则DNS要新建一条A记录，指向204.232.175.78。如果绑定的是二级域名，则DNS要新建一条CNAME记录，指向username.github.com（请将username换成你的用户名）。此外，别忘了将_config.yml文件中的baseurl改成根目录"/"。

至此，最简单的Blog就算搭建完成了。进一步的完善，请参考Jekyll创始人的[示例库](#)，以及其他用Jekyll搭建的[blog](#)。

（完）

文档信息

- 版权声明： 自由转载-非商用-非衍生-保持署名（[创意共享3.0许可证](#)）
- 发表日期： 2012年8月25日
- 更多内容： [档案](#) » [开发者手册](#)
- 购买文集：  《如何变得有思想》
- 社交媒体：  twitter,  weibo
- Feed订阅： 

相关文章

- **2015.12.09:** [常用 Git 命令清单](#)

我每天使用 Git ， 但是很多命令记不住。

- **2015.09.30:** [Github 的清点对象算法](#)

使用 Github 的时候，你有没有见过下面的提示？

- **2015.09.23:** [持续集成是什么？](#)

互联网软件的开发和发布，已经形成了一套标准流程，最重要的组成部分就是持续集成（Continuous integration，简称CI）。

- **2015.09.17:** [网页性能管理详解](#)

你遇到过性能很差的网页吗？