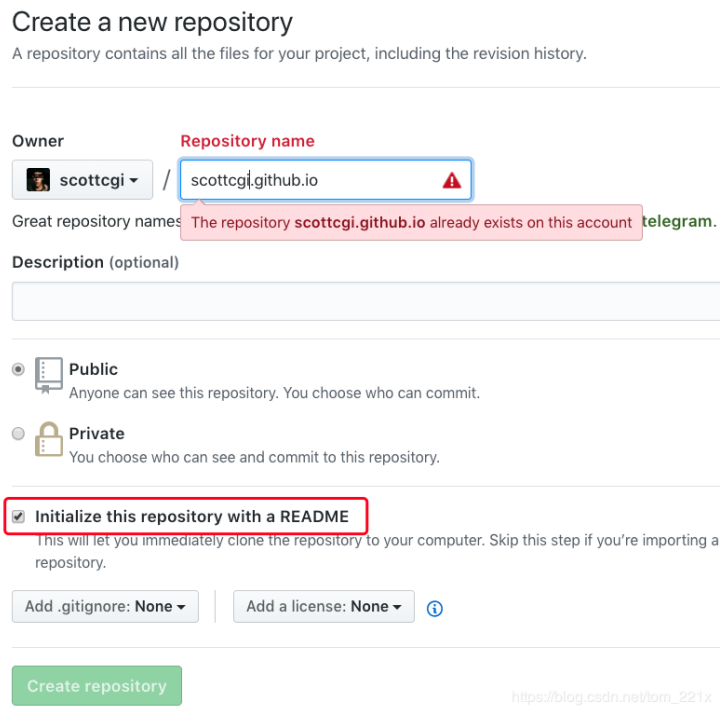
(1条消息)Github Pages + jekyll 全面介绍极简搭建个人网站和博客 - scott.cgi

本文将会全面介绍一下如何使用Github Pages + jekyll搭建个人站点，所谓极简的意思就是不用使用git和本地构建jekll服务，直接在Github网站上编辑设置即可，但会涉及到jekll的一些配置和编程控制。可以参看我的网站模板：https://scottcgi.github.io 使用了github内置的主题修改而成。

文章目录

第一步，建立Github仓库

首先到这里Github，创建一个仓库。

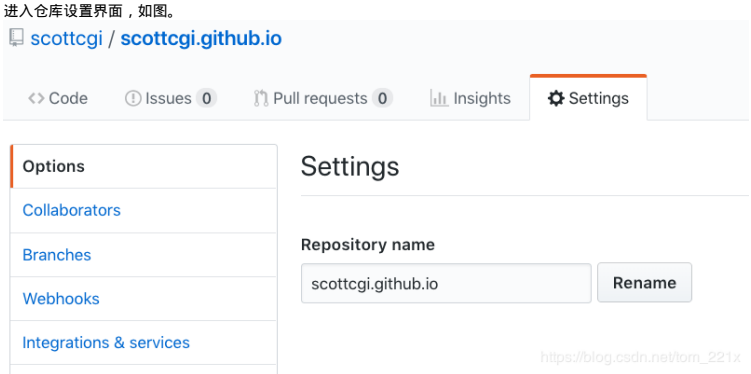


仓库名称有固定的格式： `username.github.io`，其中**username**必须是Github账户的用户名（我的是scottcgi），github.io是固定的，这个地址将会成为个人站点的网站地址。另外，我们可以勾选**Initialize this repository with a README**，让仓库自动创建一个README.md文件，我们用它来做站点的首页（当然也可以不创建，后面自行创建，或是建立index.html也行）。

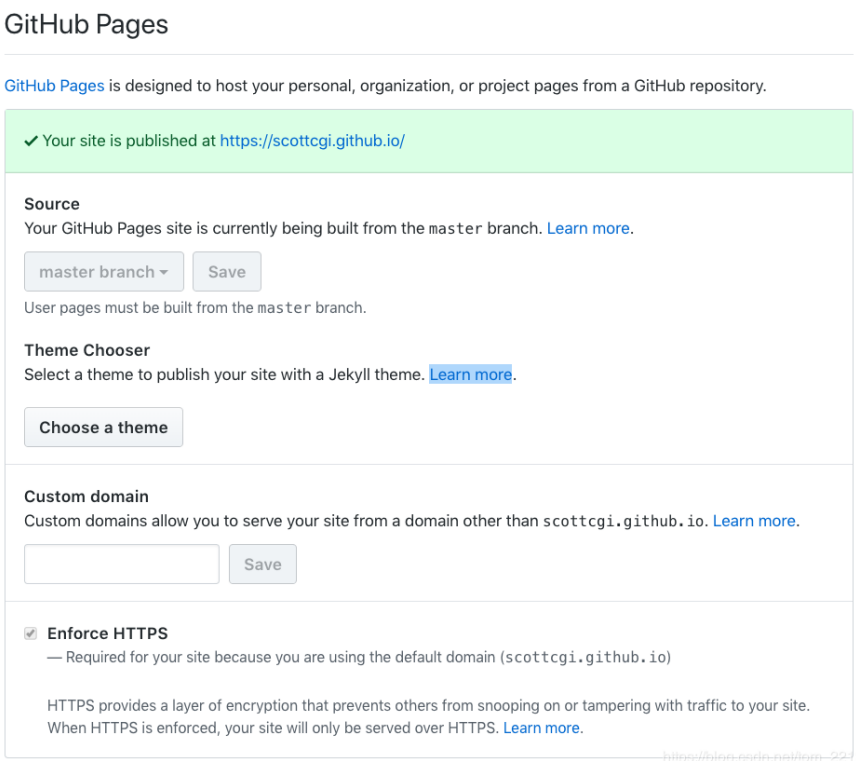
注意： username如果不是Github账户名，这个仓库就会成为`username.github.io`的子站点，比如访问地址会是：`username.github.io/aaa.github.io`。可见，`username.github.io`是github默认分配给你的域名，同名仓库即代表着默认网站内容。而`username.github.io/仓库名称`，是用来访问你的其它仓库的地址。

第二步，设置仓库开启Github Pages

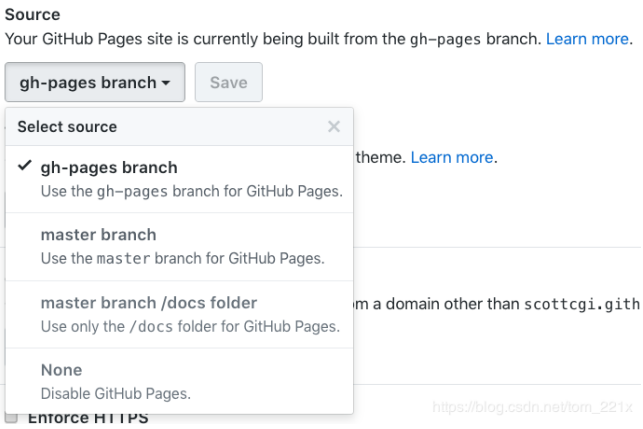
进入仓库设置界面，如图。



这里能够重新修改仓库的名称，比如这个仓库内容是fork别人的，就可以在这里修改成自己的`username.github.io`名称。



在Setting页面下有Github Pages的设置选项。绿色表示部署成功，每次修改仓库内容，都会出发Github jekyll重新编译部署，需要1-2分钟的时间，更新才能体现。如果有编译错误，包括js，css，html，markdown语法问题，都会显示红色以及错误页面和行号，同时会发邮件通知。其中，Source有以下几个选项：

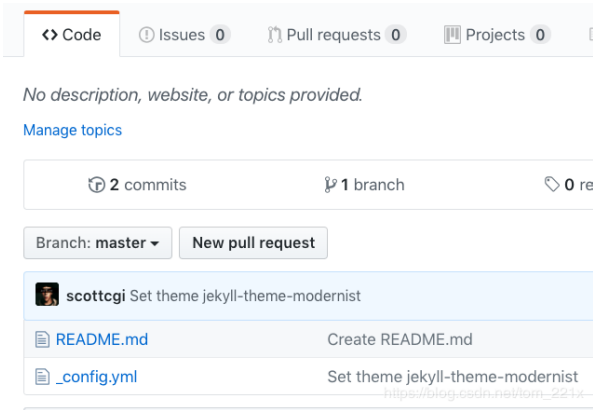


- gh-pages branch 是项目新建一个分支命名为这个，使用这个分支来做站点内容。
- master branch 是使用主分支也是默认的，来作为站点内容。
- master branch/docs folder 是使用主分支的docs文件夹来作为站点内容。
- None 就是禁用Github Pages。

如果是username.github.io只能使用主分支，其它仓库项目可以选择其它两个。接下来Choose a theme是Github提供的内置的网站主题，选择即可应用无需其它设置。Custom domain是自定义域名，本文暂不讨论。

第三步，使用Github内置主题

选择好主题，过一会刷新网站地址就已经能看到效果了，而在Code页面仅有两个文件。

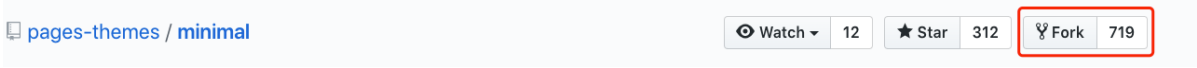


编辑README.md文件的内容，就会默认显示在网站首页，`_config.yml` 是jekyll的全局配置文件，现在里面只有一句话，`theme: jekyll-theme-modernist`。我们可以手动修改这个theme主题配置，网站就会应用不同的主题。

Github内置支持的几个主题，官方的仓库在这里：https://pages.github.com/themes，每个README.md里都有介绍如何设置。

那么我们现在就有两种方法来使用这些主题：

- 第一种，就是直接fork一个主题仓库，然后修改仓库名称为我们自己的，然后修改我们需要的部分。



- 第二种，只是简单的Choose (Change) theme（或在_config.yml设置theme），然后我们对照着官方仓库的主题目录，需要改什么文件就按照同样的路径拷贝单独一个文件到自己的仓库来修改（保持路径一致），这样就可以保持自己仓库的简洁。（如果使用了github内置的主题，github就会把你仓库的内容和内置主题内容合并到一起编译成静态网页。）

另外，更多主题可以参看这两个地址（不要挑花眼了）：jekyll themes 和 jekyll wiki site。

第四步，jekyll的目录结构

我们只需要关注几个核心的目录结构如下（可以自己创建）：

- `_layouts`（存放页面模板，md或html文件的内容会填充模板）
- `_sass`（存放样式表）
- `_includes`（可以复用在其它页面被include的html页面）
- `_posts`（博客文章页面）
- `assets`（原生的资源文件）
  - `js`
  - `css`
  - `image`
- `_config.yml`（全局配置文件）
- `index.html, index.md, README.md`（首页index.html优先级最高，如果没有index，默认启用README.md文件）
- 自定义文件和目录

更多更详细的目录结构参看jekyll官网：https://jekyllrb.com/docs/structure

第五步，jekyll的模板编程语言Liquid的使用

- 变量 `{{ variable }}` 被嵌入在页面中，会在静态页面生成的时候被替换成具体的数值。常用的全局变量对象有：`site` 和 `page`。这两个对象有很多默认自带的属性，比如：`{{ site.time }}`，`{{ page.url }}`。更多的默认值参看：https://jekyllrb.com/docs/variables。
  - `site`对象对应的就是网站范围，自定义变量放在`_config.yml`中，比如`title:标题`使用`{{ site.title }}`访问。
  - `page`对象对应的是单个页面，自定义变量放在每个页面的最开头，比如：

```
---
myNum:100
myStr:我是字符串
---
```

    1. 1
    2. 2
    3. 3
    4. 4

    使用`{{ page.myNum }}`和`{{ page.myStr }}`访问。

- 条件判断语句，更多详见：https://shopify.github.io/liquid/tags/control-flow

```
{% if site.title == 'Awesome Shoes' %}
    These shoes are awesome!
{% endif %}

{% if site.name == 'kevin' %}
    Hey Kevin!
{% elsif site.name == 'anonymous' %}
    Hey Anonymous!
{% else %}
    Hi Stranger!
{% endif %}
```

  1. 1
  2. 2
  3. 3
  4. 4
  5. 5
  6. 6
  7. 7
  8. 8
  9. 9
  10. 10
  11. 11

- 循环迭代，更多详见：https://shopify.github.io/liquid/tags/iteration

```
{% for product in collection.products %}
    {{ product.title }}
{% endfor %}
```

```
    1
    2
    3
```

- 默认函数，可以对变量进行一些处理，比如大小写转化、数学运算、格式化、排序等等，在Liquid中叫做Filters。比如`{{ "Hello World!" | downcase }}`转换字符串为小写。更多内置函数详见：https://jekyllrb.com/docs/liquid/filters

第六步，使用_config.yml文件设置jekyll

如果不是fork别人的仓库，就需要自己创建一个这个文件。然后，我们就可以配置一些默认的属性来控制jekyll的编译过程。更多详细的内置属性详见：https://jekyllrb.com/docs/configuration/default

同时我们可以自定变量，会自动绑定到`site`对象上，比如我们可以把导航配置到_config.yml中：

```
nav:
- name: Home
  link: /
- name: About
  link: /about.html
```

```
1
2
3
4
5
```

```
// 以下嵌入页面，page.url以 "/" 开头
<nav>
  {% for item in site.nav %}
    <a href="{{ item.link }}"
      {% if page.url == item.link %} style="color: red;" {% endif %}
    >
      {{ item.name }}
    </a>
  {% endfor %}
</nav>
```

```
1
2
3
4
5
6
7
8
9
10
```

当然，我们也可以把一些数据单独放入一个`yml`文件，然后放在固定的数据文件夹`_data`下，比如`_data/navigation.yml`，这样访问这个文件的数据对象就是`site.data.navigation`。

第七步，_layouts模板配置

`_layouts`文件夹存放的是页面模板，默认需要一个`default.html`，什么意思？就是说，layout提供一个页面的布局框架，这是固定的模式，包括样式、结构、布局、脚本控制等等。然后，我们在用其它md或html文件去动态填充这个框架，这样就形成了一个完整的页面。比如我的`default.html`页面如下：

```
<!doctype html>
<html lang="{{ page.lang | default: site.lang }}">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    {% seo %}

    <link rel="icon" href="https://avatars0.githubusercontent.com/u/1797320" type="image/x-icon" title="scottcgi">
    <link rel="stylesheet" href="{{ '/assets/css/style.css?v=' | append: site.github.build_revision | relative_url }}">

    <script src="{{ '/assets/js/scale.fix.js' | relative_url }}"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">

  </head>
  <body>
    <div class="wrapper">
      <header {% unless site.description or site.github.project_tagline %} class="without-description" {% endunless %}>
        <h1>{{ site.title | default: site.github.repository_name }}</h1>
        <p>{{ page.description | default: site.description }}</p>

        <ul>
        {% for item in site.nav %}
          {% if page.url == item.link %}
            <li style="background-color:#069">
              <a href="{{ item.link }}">
                <strong>{{ item.name }}</strong>
              </a>
            </li>
          {% else %}
            <li><a href="{{ item.link }}">{{ item.name }}</a></li>
          {% endif %}
        {% endfor %}
        </ul>
      </header>
      <section>

      {{ content }}

      </section>
    </div>

    <footer>
      <p>{{ site.copyright | default: "copyright not found in _config.yml" }}</p>
    </footer>
  </body>
</html>
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

- {% seo %} 是jekyll的一个插件提供的seo优化，详情在这里：https://github.com/jekyll/jekyll-seo-tag
- 核心在于 {{ content }} 这个变量是内置的，会用我们的md或html页面填充这部分内容。
- 其它的，我们看到会大量使用变量和流程控制代码，来填充模板的方方面面。
- 于是，填充模板的内容，一方面是来自读取配置文件的变量，一方面是来自_includes的页面，还有就是来自 {{ content }} 对应的页面。

当然，我们也可以不使用 {{ content }} 来填充模板，而是使用_includes的页面来代替{{ content }} ，但这样不够灵活，因为使用 {{ content }}，我们可以在每个页面单独设置对应的layout模板。

第八步，md和html页面编写

站点内容页面，可以使用markdown或html来编写，但markdown编写的md文件，在浏览器地址访问的时候依然使用html文件后缀。推荐使用markdown来书写内容，语法参见：Github md 示例 和 Github md 教程。比如下面这个About.md页面：

```
---
layout: default
title: About
---
# About page

This page tells you a little bit about me.
```

layout: default 就是告诉jekyll这个页面使用哪个模板，即这个页面会放入哪个模板的{{ content }}。当然，我们可以在_layouts文件夹下提供多个不同的模板，然后根据需要不同的页面使用不同的layout。

页面可以放在任意位置和目录，访问的时候从站点域名开始，带上目录名称，再次注意需要使用html结尾。如果想要自定义浏览器的访问路径，可以参看详细设置：permalinks。

md和html页面的区别：

- md有自己的语法，可以使用少量的html标签，最终会编译成html，侧重于内容编写。
- html可以随意使用html标签，可以使用liquid模板语言，侧重于页面模板和功能控制。

至此，我们就可以在github上，新建md文件然后编辑提交，等待几分钟编译生成之后，就可以在浏览器里看到页面内容了。

第九步，博客文章编写和管理

我们自然可以新建目录，提交文章，然后添加一个文章列表页面。但我们也可以把这些交给jekyll的内置机制来完成，因为它提供了一些方便的内置文章管理功能。

- _posts文件夹是内置的放置文章的目录，我们可以将固定格式year-moth-day-name.md名称的md文件放到这里。比如新建一篇md的博客文章放到_posts目录下：

  ```
  ---
  layout: post
  ---
  这是一篇博客文章。
      1
      2
      3
      4
  ```

- 接下来我们需要添加一个post的模板页面到_layouts文件夹下面。

  ```
  ---
  layout: default
  ---

  <h1>{{ page.title }}</h1>
  <p>{{ page.date | date_to_string }} - {{ page.author }}</p>

  {{ content }}
      1
      2
      3
      4
      5
      6
      7
      8
  ```

  可见，模板页面本身也可以使用模板，这里post使用了default模板，而这里{{ content }} 就会填充_posts下面编写的页面（如果页面使用了layout: post模板）。

- 最后，我们还需要编写一个博客文章列表的页面，用来展示所有的文章。比如在根目录新建blog.html页面：

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

  <h1>Latest Posts</h1>

  <ul>
  ```

```
  {% for post in site.posts %}
    <li>
      <h2><a href="{{ post.url }}">{{ post.title }}</a></h2>
      <p>{{ post.excerpt }}</p>
    </li>
  {% endfor %}
</ul>
```

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15.

- `site.posts` jekyll会自动生成`_posts`目录的对象。
- `post.url` jekyll会自动会设置在`_posts`目录下的页面url。
- `post.title` 默认是md文件名称，但也可以在文章页面自定义`title: 我的文章自定义名称`。
- `post.excerpt` 默认是文章第一段的摘要文字。

第十步，Github Pages的限制

Github Pages 并不是无限存储和无限流量的静态站点服务，一些限制如下：

- 内容存储不能超过1GB。
- 每个月100GB流量带宽。
- 每小时编译构建次数不超过10次。（在线修改重新编译并未发现这个限制）
- 更多参看官方说明：usage-limits。

总结

在实际的使用过程中，我发现完全可以在Github网站上，编写md和html页面，修改js和css文件，来完成站点的设置和搭建。只不过每次修改都要触发Github jekyll的编译行为，有点慢（不知道是不是增量编译），没有在本地修改调试的速度快。

更多jekyll详细的设置和功能，参看官方网站的文档：https://jekyllrb.com/docs。