BeiYuu +

使用Github Pages建独立博客

2012-02-22

Github很好的将代码和社区联系在了一起，于是发生了很多有趣的事情，世界也因为他美好了一点点。Github作为现在最流行的代码仓库，已经得到很多大公司和项目的青睐，比如jQuery、Twitter等。为使项目更方便的被人理解，介绍页面少不了，甚至会需要完整的文档站，Github替你想到了这一点，他提供了Github Pages的服务，不仅可以方便的为项目建立介绍站点，也可以用来建立个人博客。

Github Pages有以下几个优点：

- 轻量级的博客系统，没有麻烦的配置
- 使用标记语言，比如Markdown
- 无需自己搭建服务器
- 根据Github的限制，对应的每个站有300MB空间
- 可以绑定自己的域名

当然他也有缺点：

- 使用Jekyll模板系统，相当于静态页发布，适合博客，文档介绍等。
- 动态程序的部分相当局限，比如没有评论，不过还好我们有解决方案。
- 基于Git，很多东西需要动手，不像Wordpress有强大的后台

大致介绍到此，作为个人博客来说，简洁清爽的表达自己的工作、心得，就已达目标，所以Github Pages是我认为此需求最完美的解决方案了。

购买、绑定独立域名

虽说Godaddy曾支持过SOPA，并且首页放着极其不专业的大胸美女，但是作为域名服务商他做的还不赖，选择它最重要的原因是他支持支付宝，没有信用卡有时真的很难过。

域名的购买不用多讲，注册、选域名、支付，有网购经验的都毫无压力，优惠码也遍地皆是。域名的配置需要提醒一下，因为伟大英明的GFW的存在，我们必须多做些事情。

流传Godaddy的域名解析服务器被墙掉，导致域名无法访问，后来这个事情在BeiYuu也发生了，不得已需要把域名解析服务迁移到国内比较稳定的服务商处，这个迁移对于域名来说没有什么风险，最终的控制权还是在Godaddy那里，你随时都可以改回去。

我们选择DNSPod的服务，他们的产品做得不错，易用、免费，收费版有更高端的功能，暂不需要。注册登录之后，按照DNSPod的说法，只需三步（我们插入一步）：

- 首先添加域名记录，可参考DNSPod的帮助文档：https://www.dnspod.cn/Support
- 在DNSPod自己的域名下添加一条A记录，地址就是Github Pages的服务IP地址：207.97.227.245
- 在域名注册商处修改DNS服务:去Godaddy修改Nameservers为这两个地址：f1g1ns1.dnspod.net、f1g1ns2.dnspod.net。如果你不明白在哪里修改，可以参考这里：Godaddy注册的域名如何使用DNSPod
- 等待域名解析生效

域名的配置部分完成，跪谢方校长。

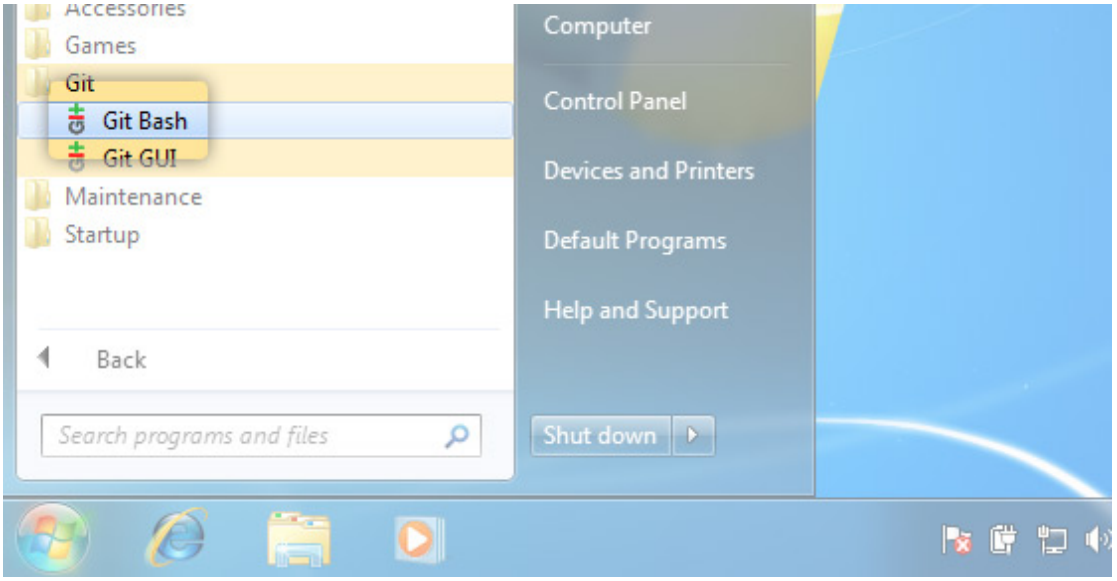
配置和使用Github

Git是版本管理的未来，他的优点我不再赘述，相关资料很多。推荐这本Git中文教程。

要使用Git，需要安装它的客户端，推荐在Linux下使用Git，会比较方便。Windows版的下载地址在这里：http://code.google.com/p/msysgit/downloads/list。其他系统的安装也可以参考官方的安装教程。

下载安装客户端之后，各个系统的配置就类似了，我们使用windows作为例子，Linux和Mac与此类似。

在Windows下，打开Git Bash，其他系统下面则打开终端（Terminal）：



1、检查SSH keys的设置

首先我们需要检查你电脑上现有的ssh key：

```
$ cd ~/.ssh
```

如果显示“No such file or directory”，跳到第三步，否则继续。

2、备份和移除原来的ssh key设置：

因为已经存在key文件，所以需要备份旧的数据并删除：

```
$ ls
config  id_rsa  id_rsa.pub  known_hosts
$ mkdir key_backup
$ cp id_rsa* key_backup
$ rm id_rsa*
```

3、生成新的SSH Key：

输入下面的代码，就可以生成新的key文件，我们只需要默认设置就好，所以当需要输入文件名的时候，回车就好。

```
$ ssh-keygen -t rsa -C "邮件地址@youremail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/your_user_directory/.ssh/id_rsa
```

然后系统会要你输入加密串（Passphrase）：

```
Enter passphrase (empty for no passphrase):<输入加密串>
Enter same passphrase again:<再次输入加密串>
```

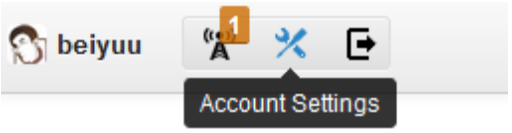
最后看到这样的界面，就成功设置ssh key了：

4、添加SSH Key到GitHub：

在本机设置SSH Key之后，需要添加到GitHub上，以完成SSH链接的设置。

用文本编辑工具打开id_rsa.pub文件，如果看不到这个文件，你需要设置显示隐藏文件。准确的复制这个文件的内容，才能保证设置的成功。

在GitHub的主页上点击设置按钮：



选择SSH Keys项，把复制的内容粘贴进去，然后点击Add Key按钮即可：

PS：如果需要配置多个GitHub账号，可以参看这个多个github帐号的SSH key切换，不过需要提醒一下的是，如果你只是通过这篇文章中所述配置了Host，那么你多个账号下面的提交用户会是一个人，所以需要通过命令`git config --global --unset user.email`删除用户账户设置，在每一个repo下面使用`git config --local user.email '你的github邮箱@mail.com'` 命令单独设置用户账户信息

5、测试一下

可以输入下面的命令，看看设置是否成功，`git@github.com`的部分不要修改：

```
$ ssh -T git@github.com
```

```
$ ssh -T git@github.com
```

如果是下面的反应：

```
The authenticity of host 'github.com (207.97.227.239)' can't be established
RSA key fingerprint is 16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)?
```

不要紧张，输入`yes`就好，然后会看到：

```
Hi <em>username</em>! You've successfully authenticated, but GitHub does no
```

6、设置你的账号信息

现在你已经可以通过SSH链接到GitHub了，还有一些个人信息需要完善的。

Git会根据用户的名字和邮箱来记录提交。GitHub也是用这些信息来做权限的处理，输入下面的代码进行个人信息的设置，把名称和邮箱替换成你自己的，名字必须是你的真名，而不是GitHub的昵称。

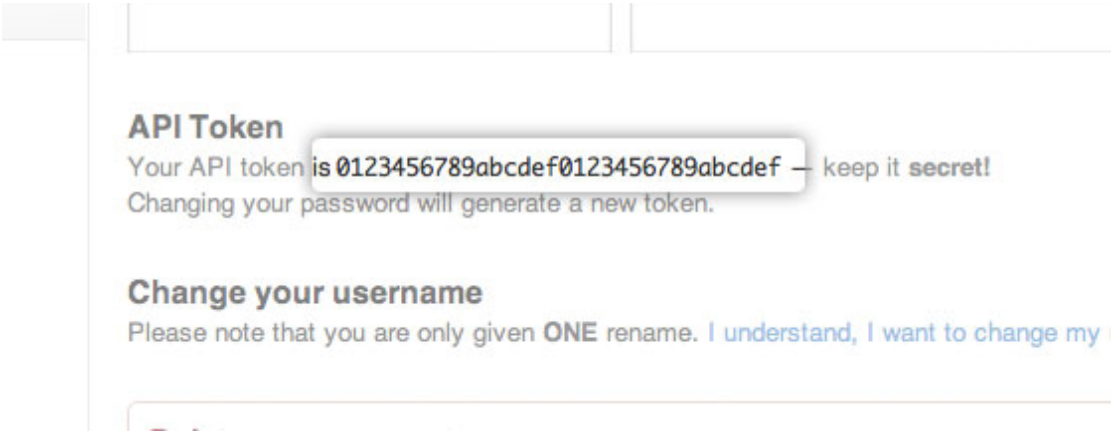
```
$ git config --global user.name "你的名字"
$ git config --global user.email "your_email@youremail.com"
```

设置GitHub的token

2012-4-28补充：新版的接口已经不需要配置token了，所以下面这段可以跳过了

有些工具没有通过SSH来链接GitHub。如果要使用这类工具，你需要找到然后设置你的API Token。

在GitHub上，你可以点击Account Setting > Account Admin：



然后在你的命令行中，输入下面的命令，把token添加进去：

```
$ git config --global user.name "你的名字"
$ git config --global user.token 0123456789your123456789token
```

如果你改了GitHub的密码，需要重新设置token。

成功了

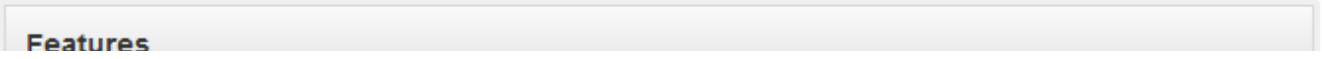
好了，你已经可以成功连接GitHub了。

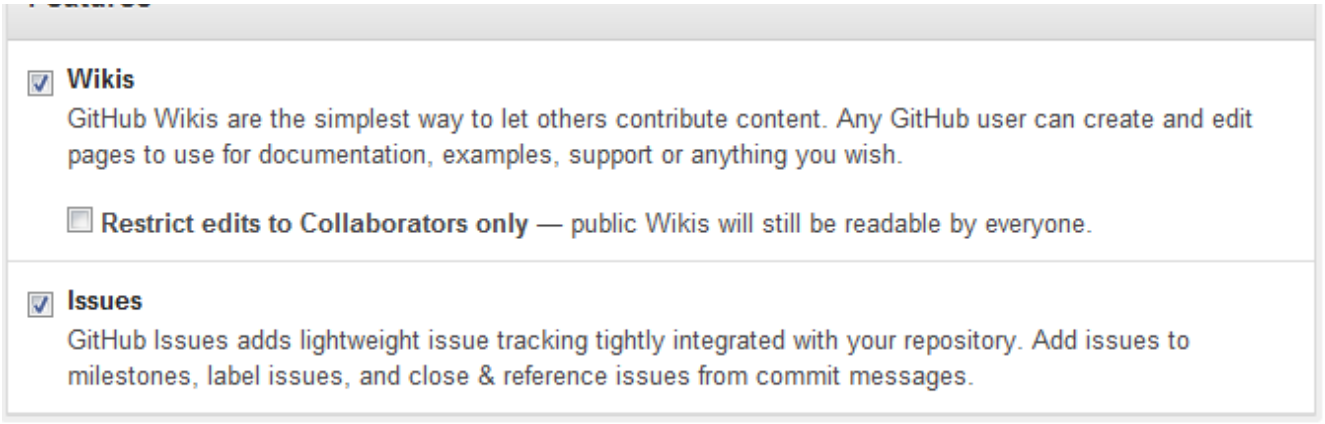
使用GitHub Pages建立博客

与GitHub建立好链接之后，就可以方便的使用它提供的Pages服务，GitHub Pages分两种，一种是你的GitHub用户名建立的`username.github.io`这样的用户&组织页（站），另一种是依附项目的pages。

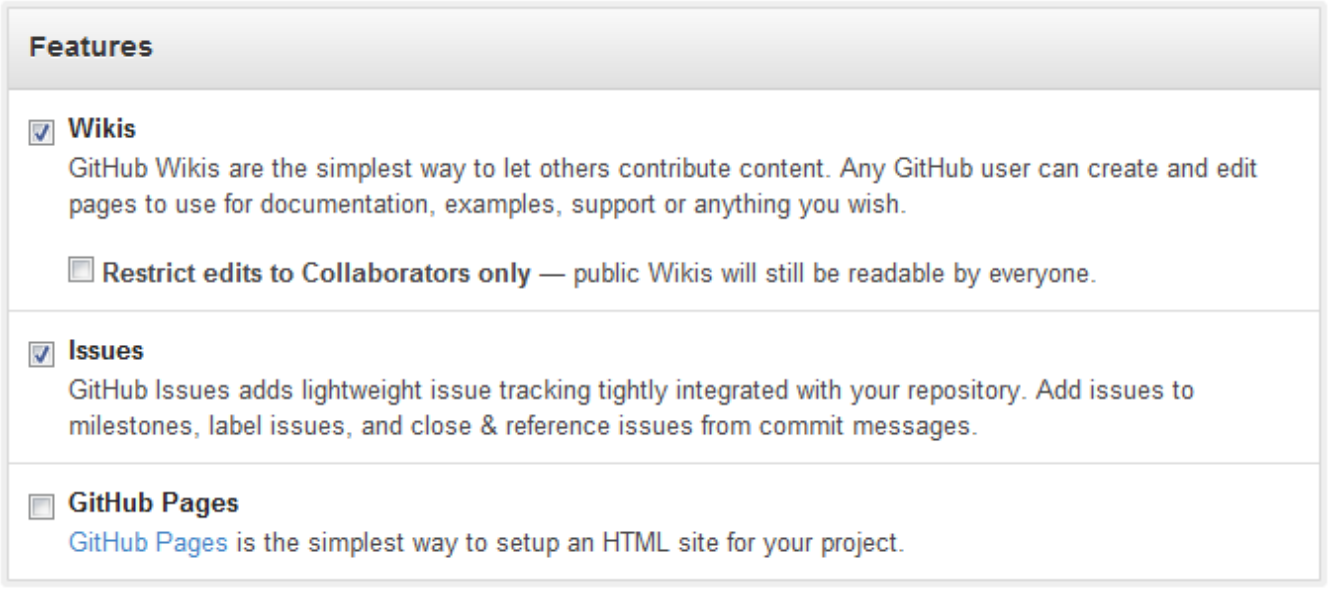
User & Organization Pages

想建立个人博客是用的第一种，形如`beiyuu.github.io`这样的可访问的站，每个用户名下面只能建立一个，创建之后点击`Admin`进入项目管理，可以看到是这样的：





而普通的项目是这样的，即使你也是用的`othername.github.io`：



创建好`username.github.io`项目之后，提交一个`index.html`文件，然后`push`到GitHub的`master`分支（也就是普通意义上的主干）。第一次页面生效需要一些时间，大概10分钟左右。

生效之后，访问`username.github.io`就可以看到你上传的页面了，beiyuu.github.io就是一个例子。

关于第二种项目`pages`，简单提一下，他和用户pages使用的后台程序是同一套，只不过它的目的是项目的帮助文档等跟项目绑定的内容，所以需要在项目的`gh-pages`分支上去提交相应的文件，GitHub会自动帮你生成项目pages。具体的使用帮助可以参考Github Pages的官方文档：

绑定域名

我们在第一部分就提到了在DNS部分的设置，再来看在GitHub的配置，要想让

`username.github.io`能通过你自己的域名来访问，需要在项目的根目录下新建一个名为`CNAME`的文件，文件内容形如：

```
beiyuu.com
```

你也可以绑定在二级域名上：

```
blog.beiyuu.com
```

需要提醒的一点是，如果你使用形如`beiyuu.com`这样的一级域名的话，需要在DNS处设置A记录到`207.97.227.245`（**这个地址会有变动，这里查看**），而不是在DNS处设置为CNAME的形式，否则可能会对其他服务（比如email）造成影响。

设置成功后，根据DNS的情况，最长可能需要一天才能生效，耐心等待吧。

Jekyll模板系统

GitHub Pages为了提供对HTML内容的支持，选择了Jekyll作为模板系统，Jekyll是一个强大的静态模板系统，作为个人博客使用，基本上可以满足要求，也能保持管理的方便，你可以查看Jekyll官方文档。

你可以直接fork我的项目，然后改名，就有了你自己的满足Jekyll要求的文档了，当然你也可以按照下面的介绍自己创建。

Jekyll基本结构

Jekyll的核心其实就是一个文本的转换引擎，用你最喜欢的标记语言写文档，可以是Markdown、Textile或者HTML等等，再通过`layout`将文档拼装起来，根据你设置的URL规则来展现，这些都是通过严格的配置文件来定义，最终的产出就是web页面。

基本的Jekyll结构如下：

```
|-- _config.yml
|-- _includes
|-- _layouts
|   |-- default.html
```

```
|   `-- post.html
|-- _posts
|   |-- 2007-10-29-why-every-programmer-should-play-nethack.textile
|   `-- 2009-04-26-barcamp-boston-4-roundup.textile
|-- _site
`-- index.html
```

简单介绍一下他们的作用：

_config.yml

配置文件，用来定义你想要的效果，设置之后就不用关心了。

_includes

可以用来存放一些小的可复用的模块，方便通过`{ % include file.ext %}`（去掉前两个{中或者{与%中的空格，下同）灵活的调用。这条命令会调用_includes/file.ext文件。

_layouts

这是模板文件存放的位置。模板需要通过YAML front matter来定义，后面会讲到，`{ { content }}`标记用来将数据插入到这些模板中来。

_posts

你的动态内容，一般来说就是你的博客正文存放的文件夹。他的命名有严格的规定，必须是`2012-02-22-artical-title.MARKUP`这样的形式，MARKUP是你所使用标记语言的文件后缀名，根据_config.yml中设定的链接规则，可以根据你的文件名灵活调整，文章的日期和标记语言后缀与文章的标题的独立的。

_site

这个是Jekyll生成的最终的文档，不用去关心。最好把他放在你的`.gitignore`文件中忽略它。

其他文件夹

你可以创建任何的文件夹，在根目录下面也可以创建任何文件，假设你创建了`project`文件夹，下面有一个`github-pages.md`的文件，那么你就可以通过`yoursite.com/project/github-pages`访问的到，如果你是使用一级域名的话。文件后

缀可以是`.html`或者`markdown`或者`textile`。这里还有很多的例子：https://github.com/mojombo/jekyll/wiki/Sites

Jekyll的配置

Jekyll的配置写在_config.yml文件中，可配置项有很多，我们不去一一追究了，很多配置虽有用但是一般不需要去关心，官方配置文档有很详细的说明，确实需要了可以去这里查，我们主要说两个比较重要的东西，一个是`Permalink`，还有就是自定义项。

`Permalink`项用来定义你最终的文章链接是什么形式，他有下面几个变量：

- `year` 文件名中的年份
- `month` 文件名中的月份
- `day` 文件名中的日期
- `title` 文件名中的文章标题
- `categories` 文章的分类，如果文章没有分类，会忽略
- `i-month` 文件名中的除去前缀0的月份
- `i-day` 文件名中的除去前缀0的日期

看看最终的配置效果：

- `permalink: pretty` /2009/04/29/slap-chop/index.html
- `permalink: /:month-:day-:year/:title.html` /04-29-2009/slap-chop.html
- `permalink: /blog/:year/:month/:day/:title` /blog/2009/04/29/slap-chop/index.html

我使用的是：

- `permalink: /:title` /github-pages

自定义项的内容，例如我们定义了`title:BeiYuu的博客`这样一项，那么你就可以在文章中使用`{ { site.title }}`来引用这个变量了，非常方便定义些全局变量。

YAML Front Matter和模板变量

对于使用YAML定义格式的文章，Jekyll会特别对待，他的格式要求比较严格，必须是这样的形式：

```
---
layout: post
title: Blogging Like a Hacker
---
```

前后的`---`不能省略，在这之间，你可以定一些你需要的变量，layout就是调用`_layouts`下面的某一个模板，他还有一些其他的变量可以使用：

- `permalink` 你可以对某一篇文章使用通用设置之外的永久链接
- `published` 可以单独设置某一篇文章是否需要发布
- `category` 设置文章的分类
- `tags` 设置文章的tag

上面的`title`就是自定义的内容，你也可以设置其他的内容，在文章中可以通过`{ { page.title }}`这样的形式调用。

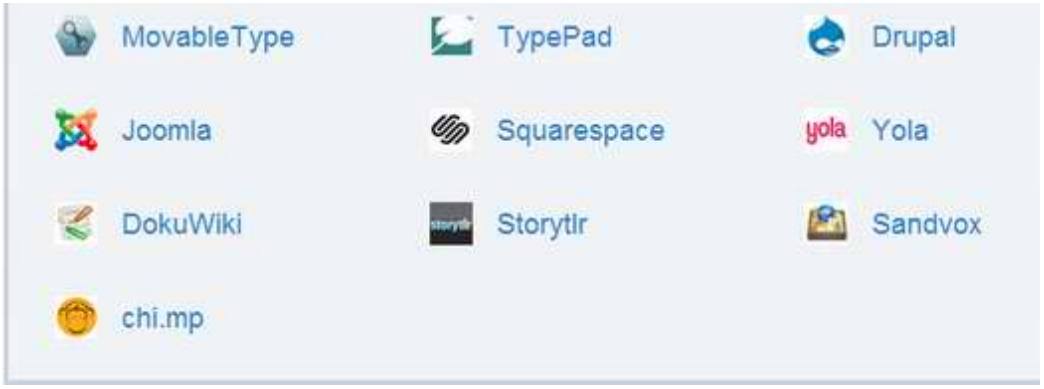
模板变量，我们之前也涉及了不少了，还有其他需要的变量，可以参考官方的文档：https://github.com/mojombo/jekyll/wiki/template-data

使用Disqus管理评论

模板部分到此就算是配置完毕了，但是Jekyll只是个静态页面的发布系统，想做到关爽场倒是很容易，如果想要评论呢？也很简单。

现在专做评论模块的产品有很多，比如Disqus，还有国产的多说，Disqus对现在各种系统的支持都比较全面，到写博客为止，多说现在仅是WordPress的一个插件，所以我这里暂时也使用不了，多说与国内的社交网络紧密结合，还是有很多亮点的，值得期待一下。我先选择了Disqus。

注册账号什么的就不提了，Disqus支持很多的博客平台，参见下图：



我们选择最下面的`Universal Code`就好，然后会看到一个介绍页面，把下面这段代码复制到你的模板里面，可以只复制到显示文章的模板中：

```
<div id="disqus_thread"></div>
<script type="text/javascript">
    /* * * CONFIGURATION VARIABLES: EDIT BEFORE PASTING INTO YOUR WEBPAGE *
    var disqus_shortname = 'example'; // required: replace example with you

    /* * * DON'T EDIT BELOW THIS LINE * * */
    (function() {
        var dsq = document.createElement('script'); dsq.type = 'text/javasc
        dsq.src = 'http://' + disqus_shortname + '.disqus.com/embed.js';
        (document.getElementsByTagName('head')[0] || document.getElementsBy
    })();
</script>
<noscript>Please enable JavaScript to view the <a href="http://disqus.com/?
<a href="http://disqus.com" class="dsq-brlink">blog comments powered by <sp
```

配置完之后，你也可以做一些异步加载的处理，提高性能，比如我就在最开始页面打开的时候不显示评论，当你想看评论的时候，点击“显示评论”再加载Disqus的模块。代码很简单，你可以参考我的写法。

```
$('#disqus_container .comment').on('click',function(){
        $(this).html('加载中...');
        var disqus_shortname = 'beiyuu';
        var that = this;
        BYB.includeScript('http://' + disqus_shortname + '.disqus.com/embed
});
```

如果你不喜欢Disqus的样式，你也可以根据他生成的HTML结构，自己改写样式覆盖它的，Disqus现在也提供每个页面的评论数接口，帮助文档在这里可以看到。

代码高亮插件

如果写技术博客，代码高亮少不了，有两个可选插件DlHightLight代码高亮组件和

如果写技术博客，代码高亮多个选，有两个可选插件DLHightLight代码高亮组件和Google Code Prettify。DLHightLight支持的语言相对较少一些，有js、css、xml和html，Google的高亮插件基本上任何语言都支持，也可以自定义语言，也支持自动识别，也有行号的特别支持。

Google的高亮插件使用也比较方便，只需要在`<pre>`的标签上加入`prettyprint`即可。所以我选择了Google Code Prettify。

搭建本地jekyll环境

这里主要介绍一下在Mac OS X下面的安装过程，其他操作系统可以参考官方的jekyll安装。

作为生活在水深火热的墙内人民，有必要进行下面一步修改gem的源，方便我们更快的下载所需组建：

```
sudo gem sources --remove http://rubygems.org/
sudo gem sources -a http://ruby.taobao.org/
```

然后用Gem安装jekyll

```
$ gem install jekyll
```

不过一般如果有出错提示，你可能需要这样安装：

```
$ sudo gem install jekyll
```

我到了这一步的时候总是提示错误`Failed to build gem native extension`，很可能的一个原因是没有安装rvm，rvm的安装可以参考这里，或者敲入下面的命令：

```
$ curl -L https://get.rvm.io | bash -s stable --ruby
```

然后还需要安装Markdown的解释器，这个需要在你的_config.yml里面设置`markdown:rdiscount`：

```
$ gem install jekyll rdiscount
```

好了，如果一切顺利的话，本地环境就基本搭建完成了，进入之前我们建立的博客目录，运行下面的命令：

```
$ jekyll --server
```

这个时候，你就可以通过`localhost:4000`来访问了。还有关于jekyll bootstrap的资料，需要自己修改调试的，可以研究一下。

我在这个过程中还遇到两个诡异的没有解决的问题，一个是我放在根目录下面的blog.md等文件，在GitHub的pages服务上一切正常，可以通过`beiyuu.com/blog`访问的到，但是在本地环境下，总是`not found`，很是让人郁闷，看生成的`_site`目录下面的文件，也是正常的`blog.html`，但就是找不到，只有当我把URL改为`localhost:4000/blog.html`的时候，才能访问的到，环境不同真糟糕。

还有一个是关于`category`的问题，根据`YAML`的语法，我们在文章头部可以定义文章所属的类别，也可以定义为`category:[blog,rss]`这样子的多类别，我在本地试一切正常，但是push到GitHub之后，就无法读取了，真让人着急，没有办法，只能采用别的办法满足我的需求了。这里还有一篇Jekyll 本地调试之若干问题，安装中如果有其他问题，也可以对照参考一下。

结语

如果你跟着这篇不那么详尽的教程，成功搭建了自己的博客，恭喜你！剩下的就是保持热情的去写自己的文章吧。

点击查看评论