Migrating Wordpress Blogs to Octopress

2012-09-16, Sun
Weekly Challenge
Comments

The task: liberating my site from Wordpress' clumsy grasp.

About the challenge

This has been said many times before: it's fantastic how Wordpress allowed so many people the ability to publish their content on the internet for free. For all the criticism, let's not forget how many people gave their free time to developing Wordpress. But the world has moved on and I can't stand the platform anymore. So I am moving to Octopress.

Installing Octopress

I simply followed the instructions on the Octopress setup page, using rvm to upgrade ruby, and it all went well.

Configuring Octopress

Next I edited the _config.yml file - again, no major surprises there. I used "$F, %a"" as the date format (2004-12-25, Mon) and /:categories/:title/ as the permalink structure.

Creating test blogs

I created the first test file:

```
1 $ rake new_post["Test 1, I will probably delete this"]
2 Creating new post: source/_posts/2012-09-15-test-1.markdown
3 subl source/_posts/2012-09-15-test-1.markdown
```

which opeened the file in Sublime Text 2. I added a single category and a some sample text.

```
1  ---
2  layout: post
3  title: "Test 1"
4  date: 2012-09-15 17:34
5  comments: true
6  categories: geekery
7  ---
8  This is my first test.
9  <!--more-->
10 And this is what it is all about.
```

I generated the site with

```
1  $rake generate
2  ## Generating Site with Jekyll
3  directory source/stylesheets/
4     create source/stylesheets/screen.css
5  /Users/ME/.rvm/gems/ruby-1.9.3-p194/gems/maruku-0.6.0/lib/maruku/input/parse_doc.rb:22:in `<top (required)>': iconv will be deprecated in
6  Configuration from /Users/ME/work/octopress/_config.yml
7  Building site: source -> public
8  Successfully generated site: source -> public
9
10 $open public/index.html
```

It launched the static file into a browser. It looked like all the bits are there, but of course the images etc aren't because the app needs a web server. I could just deploy everything to Apache, but there is a more convenient way.

Previewing Octopress with POW

Octopress is a rack app, and can be viewed wtih Pow, the simplest of rack servers. I installed it as per the instructions, then started rake to automatically deploy to it when I save.

```
curl get.pow.cx | sh
cd ~/.pow
ln -s ~/work/octopress octopress
rake watch
```

Changing Octopress theme

Wanted to try a different theme, so went for the Slash Octopress theme. The instructions are pretty simple, but had to remember to stop watching the octopress folder before running the commands.

```
$ cd octopress
$ git clone git://github.com/tommy351/Octopress-Theme-Slash.git .themes/slash
$ rake install['slash']
$ rake generate
$ rake watch
```

Liberating data from Wordpress blogs

Now the laborious parts. First of all, got the Exitwp plugin from github

```
git clone https://github.com/thomasf/exitwp.git
```

Then I logged onto the Wordpress admin console (for the last time!) and in `wp-admin/export.php` I clicked on “Download Export File”. I saved the xml files into the wordpress-xml directory I just cloned from github.

I run the XML through xmlint as suggested, although without a DTD I am not sure what I was looking for

```
$ xmllint --noout wordpress-xml/*.xml
```

Installed dependencies - but first had to install Pip.

```
$ curl -O http://pypi.python.org/packages/source/p/pip/pip-0.7.2.tar.gz
$ tar xzf pip-0.7.2.tar.gz
$ cd pip-0.7.2
$ python setup.py install
$ cd exitwp/
$ sudo pip install --upgrade  -r pip_requirements.txt
```

Edited the config.yaml file - changed download_images to true, and added a few filters:

```
# Replace certain patterns in body
# Simply replace the key with its value
body_replace: {
    '[python]': '{% codeblock lang:python %}',
    '[/python]': '{% endcodeblock %}',
    '[bash]': '{% codeblock lang:bash %}',
    '[/bash]': '{% endcodeblock %}',
    '[js]': '{% codeblock lang:js %}',
    '[/js]': '{% endcodeblock %}',
    '[ruby]': '{% codeblock lang:ruby %}',
    '[/ruby]': '{% endcodeblock %}',
    '[xml]': '{% codeblock lang:xml %}',
    '[/xml]': '{% endcodeblock %}',
    '[css]': '{% codeblock lang:css %}',
    '[/css]': '{% endcodeblock %}',
    '[html]': '{% codeblock lang:html %}',
    '[/html]': '{% endcodeblock %}',
    '[yaml]': '{% codeblock lang:yaml %}',
    '[/yaml]': '{% endcodeblock %}',
    '[php]': '{% codeblock lang:php %}',
    '[/php]': '{% endcodeblock %}'
}
```

Finally, run the converter command.

```
python exitwp.py
```

It is quite good - it mostly does a good job, and the lists the files it couldn’t parse at the end. I only had 6 out of 400 posts, which is quite something.

Even the files it couldn’t convert, it still created them with the right front matter, so all I needed to take care of is the HTML to markdown conversion. I used Pandoc, a remarkable conversion tool for that. I pasted the HTML from the Wordpress window to

a file called text.html, run the command below, then pasted the text.md file into the correct jekyll post file.

```
$ pandoc -f html -t markdown text.html  > text.md
$ subl text.md
```

The only issue is that a the <!–more–> excerpt thing was missing for most posts. I bit the bullet and added it manually to all the files - it only took me half a hour, nothing too dramatic.

Combining multiple Octopress blogs

With the basics out of the way, it's time to fine tune things. I actually had two instances of Wordpress running two separate subsites - a blog and a portfolio site with a common homepage. I was hoping to be able to combine them when WP 3.0 came out, but managing the subdomains was too painful so I never did. I want to keep that setup for now, as I am planning to do a lot of reorganizing of the portfolio site, but not the blog. Octopress is not set up to manage multiple blogs, but eventually found a way.

My starting point was two separate octopress instances, Octo1/ and Octo2/, sitting side by side.

First of all I tried deploying both to a third folder octopress_deploy. That had the undesirable side effect of duplicating assets - there'll be two versions of images, css, and so on. But also, rake watch didn't work anymore. I found watch very useful so that wasn't good.

Then I tried the technique suggested on this Octopress github page. This makes the rake watch task work again, but doesn't solve the repeated assets issue. I guess I would have to edit the themes for that, something I can do later.

So the main site is the blog. It is a vanilla Octopress site, except that the links have the structure

```
root: /
permalink: /blog/:categories/:title/
```

The portfolio site is published to the SOURCE of the main site - so that when the main site is generated, it copies along the portfolio site files too.

```
root: /work
permalink: /work/:categories/:title/
source: source
```

The rakefile for the portfolio site was amended accordingly, so that the generate publishes to the correct directory (again, notice the 'source' in the path)

```
public_dir      = "~/work/octopress/source/work"    # compiled site directory
```

So that almost creates the structure I want:

```
gotofritz.net/ -> homepage, with links to blog entries and a single link to the portfolio site in the main nav
gotofritz.net/blog/ -> blog homepage - MISSING
gotofritz.net/blog/blah/blah-blah -> blog entries
....
gotofritz.net/work/ -> portfolio homepage
gotofritz.net/work/blah/blah-blah -> portfolio entries
...
```

Now I need a way to generate the missing blog summary page. I thought I could use the archive for that, since I don't use it for anyehing else. It turns out I can just move the index.html inside source/blog/archives to source/blog - that's my blog index page, there and then. Of course it uses a different template, but that's ok for now.

Finally, some tweaks to the theme files. Changed `octopress_blog/source/includes/custom/navigation.html` *removing Archives links and changing url for blog links. Did the same on* `octopress_work/source/includes/custom/navigation.html`. Updated the favicons and head.html. Changed some image paths in the scss for the buttons in the top navigation bar - removed the Rails image-url(helper and replaced it with an hard coded URL. It's good enough for now.

Added an intro message in the homepage by editing the default.html page

```
{% if "/index.html" == page.url %}
Hello my name is fritz. blah blah
{% endif %}
```

That's pretty much it for now.

Merging sitemaps

One issue with merging multiple blogs is that each comes with its own sitemap.xml file, and they'll need to be merged. After a discussion on GitHub I came up with these amends to the rakefile, which basically merge all the sitemap.xml files it finds inside the public/ directory, and runs at the end of the generate task

```
is_multiblog    = true        # runs some extra tasks for blogs
# ...

desc "Generate jekyll site"
task :generate do
  raise "### You haven't set anything up yet. First run `rake install` to set up an Octopress theme." unless File.directory?(source_dir)
  puts "## Generating Site with Jekyll"
  system "compass compile --css-dir #{source_dir}/stylesheets"
  system "jekyll"
  if( defined? is_multiblog and is_multiblog )
    Rake::Task[:merge_sitemaps].execute
  end
end

# ....

desc "merges all the sitemaps it finds inside public. Useful if you have more than one blog under the same site. Idea from https://github.com/imathis/oct
task :merge_sitemaps do
  root_dir = "public"
  howmany  = 0
  header   = []
  trailer  = []
  alllines = []
  lines    = []
  Dir.glob( root_dir + "/**/sitemap.xml" ).each{ |sitemap|
    lines    = (IO.readlines sitemap)
    header   = lines.slice!( 0..1 )
    trailer  = [ lines.slice!( -1 ) ]
    alllines = alllines + lines
    howmany  += 1
    File.delete sitemap
  }
  File.open( root_dir + "/sitemap.xml", 'w' ) do |f|
    f.write ( header + alllines + trailer ).join( "\n" )
  end
  puts "Merged #{howmany} sitemaps onto #{root_dir}/sitemap.xml"
end
```

I made a pull request for this Octopress change, should anyone be interested

Fixing bad SEO in Octopress permalinks with categories

Another problem with Jekyll / Octopress is that it doesn't do a good job of creating permalinks with categories in them. If you have a category "Café dreams" and post "My favourite Café", and your structure is /:categories/:title, then your permalink will include `/Café Dreams/my-favourite-cafe` which is bad SEO. There are two separate aspects to it, with two different solutions - fixing the legacy Wordpress apges, and ensuring all future pages do not suffer from this issue.

Generating Octopress permalinks from legacy Wordpress slugs

This is a semi-manual batch job, but there isn't really an easy way to do it. The good thing is that all the posts have a Wordpress slug: field, so I can use that to create the title from. I create a temporary rake task for this. It worked ok, bar a couple of files which I fixed manually.

```
desc "fix permalinks"
task :fix do

  howmany = 0

  #get all files
  Dir.glob( "#{source_dir}/#{posts_dir}/**" ).each{ |post|

      #get frontmatter
      stream = File.open( post )
      frontmatter = YAML::load( stream )
      stream.close

      #only create permalink if not there
      if( frontmatter["permalink"] or !frontmatter.has_key?("slug") )
        next
      end

      #generate permalink - regex is from category_generator.rb
      catSlug = frontmatter["categories"][0].gsub(/_|\P{Word}/, '-').gsub(/-{2,}/, '-').downcase
      frontmatter["permalink"] = "blog/" + catSlug + "/" + frontmatter["slug"]

      #gets rest of file
      content = File.read( post ).gsub( /^---.+:?---/m, "" )

      #updates file
      File.open( post, "w") { |file|
        file.puts frontmatter.to_yaml + "---" + content
```

```
      }

      howmany += 1
  }
  puts "Update #{howmany} files"

end
```

Ensuring new Octopress pages have an SEO friendly link with category

In order to ensure all new Octopress posts do not suffer from the same bad permalink problem, I amended the rake new_post task to take an optonial second parameter, category. So you can call it like this

```
rake new_post["is coffee passé?","Café Dreams"]
```

and it will generate this front matter below. Notice that rake will complain if there are any blank spaces between the two square brackets. While I was at it I added an 'editor' variable (in my case, "subl") to open the newly created file with.

```
---
layout: post
title: "is coffee passé?"
date: 2012-09-20 17:14
comments: true
categories:
- Café Dreams
permalink: /blog/cafe-dreams/is-coffee-passe/
---
```

Note that I had to the encoding as the first line, to avoid the regular expression choking on umlauts etc

```
# encoding: utf-8
```

The code for the rake is below.

```
# usage rake new_post[my-new-post] or rake new_post['my new post'] or rake new_post (defaults to "new-post")
desc "Begin a new post in #{source_dir}/#{posts_dir}"
task :new_post, :title, :category do |t, args|
  raise "### You haven't set anything up yet. First run `rake install` to set up an Octopress theme." unless File.directory?(source_dir)
  mkdir_p "#{source_dir}/#{posts_dir}"
  args.with_defaults(:title => 'new-post', :category => "")
  title = args.title
  filename = "#{source_dir}/#{posts_dir}/#{Time.now.strftime('%Y-%m-%d')}-#{title.to_url}.#{new_post_ext}"
  if File.exist?(filename)
    abort("rake aborted!") if ask("#{filename} already exists. Do you want to overwrite?", ['y', 'n']) == 'n'
  end

  permalink = ""

  #get config
  stream = File.open( "_config.yml" )
  configYml = YAML::load( stream )
  stream.close

  #does it need to fix the permalink?
  if /:categories\b/.match( configYml["permalink"] ) and !/:((year)|(date)|(month)|(day)|(pretty))\b/.match( configYml["permalink"] )
    permalink = "permalink: " + configYml["permalink"]
                    .gsub(/:categories/, args.category.to_url )
                    .gsub(/:title/, title.to_url )
  end


  puts "Creating new post: #{filename}"
  open(filename, 'w') do |post|
    post.puts "---"
    post.puts "layout: post"
    post.puts "title: \"#{title.gsub(/&/,'&amp;')}\""
    post.puts "date: #{Time.now.strftime('%Y-%m-%d %H:%M')}"
    post.puts "comments: true"
    post.puts "categories: "
    if "" != args["category"]
      post.puts "- #{args.category}"
    end
    if permalink
      post.puts permalink
    end
    post.puts "---"
  end

  #open straight away
  if defined? editor
    system "#{editor} #{filename}"
  end
end
```

There are plenty more small adjustements to do, but this is it for now. Now it’s time for part 2: depoly to an Nginx server

Related Posts

Adding Related Post to Octopress

Geekery 2012-09-16

Hosting Octopress on nginx

Weekly Challenge 2012-09-16

Comments