oj.org

Andrew Caird

2013-05-14 Tue

Contents

1 Introduction

Jekyll is a static web site generator written in Ruby. It can transform various text markups, using a templating language, into static html. The resulting site

can be served by almost any web server without requiring additional components such as php. Jekyll is the tool used to produce Github's pages.

This article discusses how to produce both a static site and a blog using Jekyll and org. Rather than writing a markup processor for org files, I have relied on org's html export features to generate files that can be processed by Jekyll.

Org already has an excellent html export engine. However, it lacks built in support for blogging. Using Jekyll also gives more control over the final appearance of your site.

Publishing your site with org and Jekyll involves three steps:

1. write your page content using org.
2. use org to export your pages to html in the Jekyll project directory.
3. run Jekyll to convert your html pages exported from org into your final site.

By default Jekyll produces its output in the `_site` directory of Jekyll's working directory. This is a self contained version of your site, which can be deployed to your web server. The files in `_site` are completely self contained, so all you need to do is to copy them to your web server. Methods include using ftp, rsync or a git post commit hook. You can configure where Jekyll puts its published files in `_config.yml`.

Essentially, I am using org to produce everything between the `<body>` tags on the page and Jekyll to produce the rest. Note that you can easily embed html content in your org pages using the `+BEGIN_HTML` tag.

2 Install Jekyll

Installation is described at the Jekyll web site.

3 Project Directory Structure

Jekyll expects a certain directory structure. In the example below my Jekyll project is in a directory called `jekyll`. Blog posts are in `_posts` and the layout templates in `_layouts`. The `_includes` directory is for files containing code you want to include in other pages e.g. a header or sidebar.

The file `_config.yml` is a YAML file that contains Jekyll's configuration for the site.

In addition to the `_posts` directory you can create other directories to hold different non blog parts of your site.

```
'|myproject
'|   |org
'|      |_posts
```

```
'|      |-- 2009-11-26-my-first-post.org
'|   |index.org
'|   |jekyll
'|   -- _config.yml
'|   -- _layouts
'|      |-- default.html
'|      `-- post.html
'|   -- _posts
'|      |-- 2009-11-26-my-first-post.html
'|
'|   -- |_site
'|   -- |_includes
`    -- index.html
```

You should setup the directory structure of your org files to mirror that of the Jekyll project. Then when you export your org files as html the files will end up in the correct place in your Jekyll project. I usually place the directory containing my org files in the directory about the Jekyll project directory to make sure that Jekyll doesn't consider .org files to be part of its project.

4 Configuring org html Export

The fundamentals of publishing html are described in the HTML publishing tutorial on worg. I am assuming that you have a basic working org publishing setup. By default org produces complete web pages. However, as I am using Jekyll I am only really interested in the section of the page between the `<body>` tags, as Jekyll produces the rest. Most things in org are configurable and it's possible to tell org to export only the bits of the page between the `<body>` tags. Here is the relevant section of my `.emacs` file:

```
(setq org-publish-project-alist
      '(

  ("org-ianbarton"
          ;; Path to your org files.
          :base-directory "~/devel/ianbarton/org/"
          :base-extension "org"

          ;; Path to your Jekyll project.
          :publishing-directory "~/devel/ianbarton/jekyll/"
          :recursive t
          :publishing-function org-publish-org-to-html
          :headline-levels 4
          :html-extension "html"
          :body-only t ;; Only export section between <body> </body>
    )
```

```
    ("org-static-ian"
          :base-directory "~/devel/ianbarton/org/"
          :base-extension "css\\|js\\|png\\|jpg\\|gif\\|pdf\\|mp3\\|ogg\\|swf\\|php"
          :publishing-directory "~/devel/ianbarton/"
          :recursive t
          :publishing-function org-publish-attachment)

    ("ian" :components ("org-ianbarton" "org-static-ian"))

))
```

To export my site I just run `C-c e X ian`.

You need to set the destination of your exported files to your Jekyll project directory. Assuming you have set up your org directory structure to mirror that of your Jekyll project everything should end up in the correct place.

5 Creating an org File to be Published with Jekyll

When you run Jekyll it processes the source files for your site and any files with YAML Front Matter are subject to special processing. The Front Matter is used to tell Jekyll how to format your page.

Bear in mind that Jekyll doesn't process your `.org` files, but the `.html` files produced by exporting. So when writing an org file it should be formatted in such a way that when exported it produces html suitable for processing by Jekyll.

YAML Front Matter must be the first thing in the file, with no blank lines above the Front Matter Section. A typical Front Matter Section would look like:

```
---
layout: default
title: My Page Title.
---
```

So you should ensure that any Front Matter directives come first in your org file.

Note that the three hyphens `---` are part of the markup and are required. The layout tag tells Jekyll which layout from its `_layouts` directory should be used to format your page. You can include any other keys in the Front Matter section (e.g. `title:`), which you can use in your page. See the Jekyll wiki for more details on Front Matter.

Below is a short extract from one of my org files showing my setup:

```
#+STARTUP: showall indent
#+STARTUP: hidestars
It was early January when six of us travelled up to ....
```

The Front Matter section is wrapped in `#+BEGIN_HTML` so it is exported literally to the final html file. You may need to upgrade your org version as older versions produced two blank lines before the Front Matter section when exported. You can define your own Front Matter keys and use them within your generated page. In the above example I use the "excerpt" key to display "teasers" for a blog post.

Note that the current git version of org removes the first `---` if the directory containing the file start with an underscore. The workaround is to start your file with `---` in both the first two lines.

Carsten has also provided two hooks that are run after exporting is complete, which can also be used to tidy up the output:

```
org-export-html-final-hook      (always)
org-publish-after-export-hook   (when going through org-publish)
```

Once you have exported your org project to html it's simply a matter of running jekyll to produce the final output. By default Jekyll puts its output in the `_site` directory of your project, but you can customize this in your `_config.yml` file.

6 Blogging with Jekyll and Org

Jekyll has built-in support for blogging. Anything you place in the `_posts` directory of your Jekyll project is considered as a blog post. However, the file names of your posts must adhere to the following format:

```
yyyy-mm-dd-post_name.html
```

To write a post just create a new file with the correct filename in your `org/_posts` directory. You may find that Yasnippet is useful for inserting Front Matter and other directives in your org file. When you have finished just run `C-c e X` project_name to export your org project as html and then run jekyll to generate your site.

You can use Jekyll's template markup to decide how your blog posts are displayed. On the Jekyll sites page there are many sites with source listed, so you can study how other people use the markup to create their blog. You can also view my site http://www.ian-barton.com and see a snapshot of the source at http://github.com/geekinthesticks/ianbarton.

You can assign categories to your posts either by placing posts inside folders like:

```
_posts/org/jekyll/howto.html
```

This would assign your post to the *org* and *jekyll* categories.
or by using YAML markup in your org file:

```
categories:
    - org
    - linux
```

6.1 Showing Blog Posts on the Front Page

Most blogs show the latest posts on their front page. The example below shows the title and an excerpt for the five latest posts:

```
<ul class="posts">
{% for post in site.posts limit: 5 %}
  <div class="post_info">
    <li>
            <a href="{{ post.url }}">{{ post.title }}</a>
            <span>({{ post.date | date:"%Y-%m-%d" }})</span>
    </li>
    </br> <em>{{ post.excerpt }} </em>
    </div>
  {% endfor %}
</ul>
```

6.2 Creating Archive Pages

You will probably only want to display a limited number of blog posts on your front page. However, you will also want to make older pages available. You can create a simple list of all blog posts using the following markup:

```
<ul>
  {% for post in site.posts %}
  <li>
    <a href="{{ post.url }}" title="{{ post.title }}">
      <span class="date">
        <span class="day">{{ post.date | date: '%d' }}</span>
        <span class="month"><abbr>{{ post.date | date: '%b' }}</abbr></span>
        <span class="year">{{ post.date | date: '%Y' }}</span>
      </span>
      <span class="title">{{ post.title }}</span>
    </a>
  </li>
  {% endfor %}
</ul>
```

7 Inserting Image

You will probably want to insert some images into your blog posts. I use the following method:

```
<img src ="/images/skiddaw.jpg"
alt="John and Ella on Skiddaw" align="left" width="300" height="250"
title="John and Ella on Skiddaw" class="img"</img>
```

Note that the class attribute refers to the class used to style the image tag in your css. My css contains:

```
img {
    margin: 15px;
    border: 1px solid blue;
}
```

Note that if you wish to have some space between your image and the text, using padding in your css doesn't seem to work. I use margin, which gives the same effect.

Whilst this works, it won't display captions for your images. Unfortunately, after years of development xhtml doesn't seem to provide an easy way to display image captions. I decided to use the method described here. An example from of floating a picture to the right of the text is shown below.

In your `.org` file use the following html to embed the picture:

```
<div class="photofloatr">
  <p><img src="myphoto.jpg" width="300"
    height="150" alt="My Mug Shot"></p>
  <p>A photo of me</p>
</div>
```

Now you need to add some information to your style sheet:

```
div.photofloatr {
    float: right;
    border: thin silver solid;
    margin: 0.5em;
    padding: 0.5em;
}

div.photofloatr p {
  text-align: center;
  font-style: italic;
  font-size: smaller;
  text-indent: 0;
}
```

A third method, which I haven't tried myself, is to use the *jQuery EXIF* plugin to extract the caption from the image EXIF data and use Javascript to display it. See here for more details.

8 Using Text Markup in Front Matte

By default text in the Front Matter part of your file isn't processed by Jekyll's markup engine. However, you can use the Textilize filter to convert your Front Matter string into HTML, formatted using textile markup.

I use this to format my page excerpts, which I include in my org files Front Matter markup. So in my sites index.html I have:

```
<li>
  <a href="{{ post.url }}">{{ post.title }}</a>
  <span>({{ post.date | date:"%Y-%m-%d" }})</span>
</li>
</br>
<em>{{ post.excerpt | textilize}}</em>
```

This lets me use textile markup in my page excerpts, which are defined in my page's YAML Front Matter section.

9 Version Control with Jekyl

Jekyll is amenable to using version control systems. If you follow my suggested directory structure you can create a git repo to your top level directory. You can then create a post-commit script that runs the org html export and then runs Jekyll to generate your site.

10 HappyBlogger's Jekyll Modification

Bjørn Arild Mæland has created some modifications to Jekyll to provide some pre-processing to org files to allow for better integration with Jekyll. You can find his code on github.

11 Another example of Org-mode/Jekyll usage

The on-line documentation for Org-babel development is published on github which uses jekyll. The following code is used to publish one blog post for every subheading of the first to top-level headings of a org file which tracks Org-babel development. The results can be seen here, and the code used to create this site is available here.

```
(save-excursion
  ;; map over all tasks entries
  (let ((dev-file (expand-file-name
                   "development.org"
                   (file-name-directory (buffer-file-name))))
        (posts-dir (expand-file-name
                    "_posts"
                    (file-name-directory (buffer-file-name))))
        (yaml-front-matter '(("layout" . "default"))))
    ;; go through both the tasks and bugs
    (mapc
     (lambda (top-level)
       (find-file dev-file)
       (goto-char (point-min))
       (outline-next-visible-heading 1)
       (org-map-tree
        (lambda ()
          (let* ((props (org-entry-properties))
                 (todo (cdr (assoc "TODO" props)))
                 (time (cdr (assoc "TIMESTAMP_IA" props))))
            ;; each task with a state and timestamp can be exported as a
            ;; jekyll blog post
            (when (and todo time)
              (message "time=%s" time)
              (let* ((heading (org-get-heading))
                     (title (replace-regexp-in-string
                             "[:=\(\)\?]" ""
                             (replace-regexp-in-string
                              "[ \t]" "-" heading)))
                     (str-time (and (string-match "\\([[:digit:]\-]+\\) " time)
                                    (match-string 1 time)))
                     (to-file (format "%s-%s.html" str-time title))
                     (org-buffer (current-buffer))
                     (yaml-front-matter (cons (cons "title" heading) yaml-front-matter))
                     html)
                (org-narrow-to-subtree)
                (setq html (org-export-as-html nil nil nil 'string t nil))
                (set-buffer org-buffer) (widen)
                (with-temp-file (expand-file-name to-file posts-dir)
                  (when yaml-front-matter
                    (insert "---\n")
                    (mapc (lambda (pair) (insert (format "%s: %s\n" (car pair) (cdr pair))))
                          yaml-front-matter)
                    (insert "---\n\n"))
                  (insert html))
                (get-buffer org-buffer)))))))))
```

```
'(1 2))))
```

12 Other Blog Solutions for org

12.1 Blorgit

Blorgit uses org mode for markup and runs on the Sinatra mini framework. It is amenable to using git for posting and maintenance.

12.2 ikiwiki

ikiwiki is a web site compiler written in Perl. In many ways it is similar to Jekyll, but has closer integration with version control systems. It supports blogging and has many plugins.

There is an org mode plugin by Manoj, which lets you write your posts in org and converts them to html suitable for processing by ikiwiki.