Issue list for mrblog.nl

Because everything is orgmode now, I might as well keep an issue list for the site and publish it. Managing todo items is pretty well covered in org-mode ;-)

Current issues

TODO CSS changes to do BUILD

- make use of the icon font (calendar and home icon)

TODO This would be useful to have when writing longer documents CHECK

https: martignoni/hugo-notice: A Hugo theme component to display nice notices

TODO Scrolling on the map component starts zoom when scrolling page FIX

Not sure if I want to change it, but it is annoying.

Remnants from earlier iterations

1. TODO Create a better way for handling remote images BUILD

   Similar to the gmg plugin, but more generic. Goal is to have a quick way to insert a properly styled image, based on a class or keyword.

2. TODO Improve search functionality FIX

   Search functionality exists but needs improvements:

   - Move search from home page to header for better accessibility
   - Add debouncing to typing to avoid slow response (immediate result display)
   - Improve result rendering for better presentation

3. TODO Pullquote needs to be reimplemented BUILD

   Perhaps use a functionField for this, this allows for the syntax

   ```
   $somefunction("String argument")$
   ```

   in templates. This function is defined in a context like

   ```
   postCtx :: Context String
   postCtx = functionField "somefunction" (\args _ -> error $ show args) <> defaultContext
   ```

which will just print the arguments while compiling the templates.

An example for pullquote might be:

```
$pullquote("right", "Paragraph with {marker that signals begin and} marker that signals

Resulting rendering mockup:
"Paragraph with marker that signals beging and maker that signals end"
                                 "...marker that signals begin and..."
```

In html this needs to happen:

```
<span class="pullquote-right" data-pullquote="...marker that signals begin and...">

  Paragraph with marker that signals begin and marker that signals end
</span>
```

And the CSS can take care of the actual rendering.

4. TODO Show post titles in prev/next links BUILD

5. TODO Gitactivity needs to be reimplemented BUILD

6. TODO Generate multiple versions for each post BUILD

   This is now relatively easy. Generate multiple output formats for blog postings. With all source in valid org-mode, there are several format options:

   Minimal set: source+html

   - ☒ source (deliver original org-mode format)
   - ☒ html (this is the normal rendering)
   - ☒ pdf (for printing)
   - ☐ epub
   - ☐ openoffice format (for additional editing)
   - ☐ plain text format (for email)
   - ☐ activitystreams for listings (connect to mastodon)

   And provide some icon after the title or on the same line to indicate available formats

7. TODO Replace feed with my own template FIX

   Problem:

   - the updated field is a bit special, but i can copy code from

8. TODO Implement the projects or subblog functionality BUILD

   - http://ur1.ca/pj76y has an example implementation
   - using categories ?

9. TODO Implement automatic linkchecker for mrblog.nl BUILD

   Perhaps a cron job which does this.

   - only report what can be fixed by me;
   - report only if there is something to be fixed;
   - check daily

10. TODO Implement subsites with subdomains while keeping the main site the same BUILD

    Usecase: cobra.mrblog.nl content is now also on mrblog.nl tagged with 'cobra'

11. TODO Expose subdomain structure on front page BUILD

12. TODO Create an anonymous blog, as some sort of game. "Find out who I am" BUILD

    Needs:

    - reliable TOR network access
    - anonymous email
    - encrypted connections to everywhere
    - damage controle system (assume new identity quickly when exposure signs are there)
    - think about how
    - http://ur1.ca/ecl5w has some good information
    - the idea of findingout who I am is for play, it is not enough that people "know" who I am, but they need a record of how they find out. (I even consider spilling my identity upfront but that would make it much less interesting)
    - researching the HOWTO is already leaking information, so the first and foremost action is to get access to the TOR network completely.

    Recently acquired book from Mitnick may help with this.

INFO Thoughts on having a commenting system

There is currently no commenting facility on the site.

1. Requirements

   - based on open source solutions only
   - multiple interfaces (web, mail, xmpp etc.)
   - static rendering of pages must be preserved (TODO: clarify this in relation to on-site rendering with javascript)
   - authentication not needed, anyone may comment –> thus moderation
   - plain text storage (git based would be nice)

   Note: staticman.net more or less satisfies all criteria.

2. Workflow

Someone visits a page, wants to make a comment. There is **one** thing that user wants to do and that is fill a field with their comments. Extensive login and identification procedures are just getting in the way. What they will be prepared to do is easy identification, so they are credited with the comment or are able to follow the thread of discussion.

3. Viable options

   - use mastodon (non-static, ActivityPub)
   - use gotosocial (non-static, ActivityPub)
   - homebrew js only solution
   - staticman.net :: automated solution to post files to git repository, which then can be picked up by the site compiler

4. Scenario: using ActivityPub (Mastodon/GoToSocial)

   The first assumption I am going to make is that there is an ActivityPub note/post somewhere on a federated instance, i will be using a post URL as the example. The comments/replies to that post are considered comments to the posting on the website. Using the assumption the task can be split into the following sub-parts:

   (a) Displaying the comments in the relevant page

   (the next section contains api keys, so is encrypted by default, in case this gets published somewhere)

   (a) Ad 1. Displaying comments for a specific note

   —–BEGIN PGP MESSAGE—–

hQEMA79Ps5L+qnrdAQgAyjV6+jk8SoGTKzf8+SuTUIN9M1Z4lvUWvkyF2SU+W8iR
OH62xh0DldMlluvb191fsA8S81Qkrl6FecvSamnJNGOpXL9vsXdkKzXpn109zkoX
+IjskILrOzcmt/Vjt6bMrPj3HE3TaGv1FY71sGF/n4XIlxSngbZvx3b0z8RPmGP6
1mC08vGIIzXbrKJGQbKGA+f4vUUAg0GsUy+QZ03N722G9Se+QPhyQ87GIFzuCm78
Jn0r2FxHgqQgaP1Q3P5rWDnPy8oT1cshZccw9NYWYj7TDpXVdmSMPu2VZ46fXY3q
ygsIVyHPuL9fJWv2tz8u3fbcDgJZ5nrhJoDGZt7cLdLpAZYAtcELZJAPdC4oXhWR
HVWjC/G/+KjdhS4Xv2vzUcTZZwnVSJQmvl2/WJdjyW6Poa5z9onGztMLTORlgKr8
CFQlw1TtDTwBmC8kGoVpGIAyazHlg55/Aj3eNODPStQJBOz7vsOp8fCpGa8TreL1
5KdEUt9DHVL3EenedQXKY1Sp1hAJFH0jg/XqZJbG+MJjz/18yXs6kVAxgXZiJo1Z
AVlS6Aa+UcHju3fO9mBHwUFNaQ/oPFQLmrOV2AH5UpBfNu+k3sHyaM/5Ntmo7Sh0
/1nebKsMbJpJsMXZTNAPpIYusqOZJ2e2lgehMoq4po4+2uzRaqwloImmqOurmezt
jQccNECGEDwi2Al14w1V3sm6xoa8f1pbkVxW7/GV/ayIBL2IDr6mS2zoEidDyelx
GPj66PNw5P9JKF2zHP/FZtWzfyi0bzwOoNb3A4fIEb9EgLTi1eCqePML4TgFwhAn
9rvk3fzE3i5pUkfMj0FOLMDk2igK918SATH6byfgAVGBgNPerLfHCGZotYiULUqD
ShZUz/tKa3GVtniluts+xJAObtl7xKjwtiVMbU0N4tfsqOxjgmJYCVk9qoK2a/XU
BSpfCVzXkxyH6vJzD3wSXWWX2HEtAGBPGUdapH2Zx6I0Y0up7VEKmmGEn0WJIu8x
AoFGEDrD+Ng0r0HK2ZjBcVzAjnysFBT4j9DJKTDxjp5UjKiPfFnmAP8U9XBolYT3
aPrHhSNCCguLDqQ1KrLxx6Lk2WjcommfGOhZ8/it1LzJE9MomNhY8FHn3YZB5cbe
zWRYyiDahxseRli4sHaGGTkWX/DWfDV3qI61BT6sKgNJ1Q+838PhZlHIuKqoZeia

c5w9EBd3nSdC8K9aF5bE/eQKG0ZK77ThYw2G+rIWAVyRSWwmMR3sUoA+YPl1DPIL
vCMzx/FYiti8w1qihrI6xJD360V+TO9A3xIdeEBSiQwhgyaW+Hf8a/GcqftMj8iQ
ihtLcIMD8hkPaIJDjDttk4+7DY7O1uCNWWvM5/WdurFOH+iheVeNp4EqqJvic/wU
6Y7sAjhx8IQ8gNBsgiJXhEVUcUxSsZ8cRh4B+6KSmvWWtoX5YFXjOAeqB7opkISI
VCBYPrar7YqrDuZnNJ5rUn8= =5atE —–END PGP MESSAGE—–

5. Scenario: using staticman.net

   I've begun implementing staticman.net usage. This is a log of that effort

   First step: getting the data

   The current commenting system is disquss and they offer an export which dumps all comments into a proprietary xml file.

   I converted this xml file to a representation in json with xml2json to get the data in json format. Reason is that the staticman implementation directly produces json too, so I'm trying to mimick that. The reason I chose for staticman.net to produce json over yaml is that I already use json data for the site search *engine* on the blog.

   The first idea was to extract a json file for every comment from the xml file and use that directly on site with a javascript piece to bridge it to rendering.

   Extracting the json file per comment is done with `jq` in a number of distinct steps (basically for loops). While tedious to do, it's scripted and now a reproduceable step.

   (a) TODO The messages sometimes hold html, do we want to keep this? CHECK

   It seems to be only <p> tags mostly, so I guess we can clean it a bit.

   (b) END

   Having separate files for each comment is less suitable for web exposure, because they can only be accessed by their URI. This means that their filenames must be known for them to be rendered by the browser. Gathering these filenames by hakyll or combining them into one 'well-known' file is then the next step.

   Choosing them to be gathered into one well-known filename is attractive because it fits nicely with the techniques already in use on my blog. The output of hakyll is a directory per post/page which contains, so far, 1 file per format of the content. (.html and .org so far) With that one file in place, a piece of client side code can then be written to render the page based on that file.

   So, first goal is to have in the directory:

blog-slug.html the html page that is rendered
blog-slug.org the original content
blog-slug.json data related to the page (comments, other meta-data?)

It's fairly trivial to also include the blog contents into the `.json` file also. In fact this might be a good starting point.

Multisites

I want to host multiple blogs below mrblog.nl, like:

(me.)mrblog.nl main blog, having all content
cobra.mrblog.nl the cobra build
hbx360.mrblog.nl cazeneuve hbx360 related stuff
sacia.mrblog.nl sacia shaper related stuff
photo.mrblog.nl photoblog (sparse subset of media.mrblog.nl)
tools.mrblog.nl tools in general
media.mrblog.nl mediastorage for all mrblog.nl sites

The hosting of sub-blogs is probably quite easy to do by defining to publish them to subdirectories and make sure they are self-sufficient, meaning they have an index.html file there.

One way to help with this is to use categories which are a bit special:

- /work/code/$_{\text{posts}}$/blahblah.md -> post is in cats work and code and if the permalink variable contains :categories the live site will contain folders for the categories.
- categories variable in front matter: YAML list or space separated string; this just registers the category and makes the variables work.

I like the specification of categories and blogs to be separate really, but file organisation above $_{\text{posts}}$ is unavoidable to get them into categories so might as well use that to my advantage

Another option could be to have a data file below $_{\text{data}}$ which contains information on the sub-blogs.

In URI terms this would be:

1. https://mrblog.nl :: main blog, containing everything
2. https://mrblog.nl/cobra :: everything in https://cobra.mrblog.nl
3. https://mrblog.nl/hbx360 :: everything in https://hbx360.mrblog.nl
4. These should all be the same:
   (a) https://cobra.mrblog.nl/2013/11/15/the-title-of-the-post.html
   (b) https://mrblog.nl/cobra/2013/11/15/the-title-of-the-post.html
   (c) https://mrblog.nl/2013/11/15/the-title-of-the-post.html

Item 4.3 is now in place. 4.2 is not really necessary

Items 2. and 3. more or less dicate there should be folders in the sources named `cobra` and `hbx360` which have their own `index.html`

Using different config files per site in Hugo is possible but may be sensitive for conflicts. So, trying to avoid that is worth something.

Some things (like having cobra.mrblog.nl serve up from a cobra (virtual) subdirectory can be arranged from nginx.

Hugo has a similar concept with *sections* which are like pages (not time based content like posts) organized into directories, exposed in their respective url part. They can be rendered via custom layouts.

[2017-01-14 za] As I'm progressing I'm more and more leaning towards one base domain (https://mrblog.nl in this case) and have features in the blog system to define my tags categories and what have we. This makes everything a lot simpler and probably more flexible.

1. Requirements

   - [x] each subsite has its own feeds under its own domain
   - [ ] links in the feed point to the proper subdomain
   - [ ] main site has all content of all subsites, properly tagged and least, preferably categorized too
   - [ ] everything can be generated and previewed as if it was one site
   - [ ] if resources are duplicated, the process to manage them should be automated.
   - [ ] if not a separate theme, show the use that he is on a sub-blog.
   - [ ] https://sub.mrblog.nl/YYYY/MM/DD/title.html point to proper contents
   - [ ] https://mrblog.nl/sub/YYYY/MM/DD/title.thml point to proper contents
   - [x] https://mrblog.nl/YYY/MM/DD/title.html points to proper contents
   - [ ] a simple mechanism is in place for local testing.
   -

2. References

   (Previously contained Jekyll references - now using Hugo)

3. Implementation

   Taking photo.mrblog.nl as example, the steps to implement it are as follows.

   - The source documents are in folders sites/blog1,

   sites/blog2... etc. This makes sure the documents are in categories automatically which we can filter on. Example: ./sites/photo/*.org contains documents for http://photo.mrblog.nl The index.html loops over all content which is in category 'photo'. In addition, cross-blog posting can be done by setting the category in the frontmatter manually.

- nginx points photo.mrblog.nl to ./photo as root dir
-

Given the amount of time I already spent on this, perhaps a separate subblog with syndication approach is simpler (it is).

Blog article ideas

1. TODO Consider a blog series on yubikey applications WRITE

   Series: Authenticating:

   - OTP usage (with and without running own validation server)
   - FIDO U2F (only google uses this?)
   - OATH (use android as example, compare with google auth app)
   - OpenPGP usage (gnupg, ssh, firefox, openkeychain, pass, password store)
   - PIV (no usecase found yet)

   Encrypting

   - OpenPGP
   - PIV

2. TODO the conflict of being open source and having 'next new features' WRITE

3. TODO Project: cobra WRITE

4. TODO Project: Cazeneuve HBX WRITE

5. TODO Project: Sacia Shaper WRITE

6. TODO Project: Power9 machine WRITE

7. TODO Project: Cars (for each one?) WRITE

TODO Perhaps put mastodon shortcode into my blog to use check

https: Mastodon Embed Shortcode for hugo — KevOps — DevOps, Cooking, and everything else I want to write about

TODO consider a 'short note' like publication type build

This would not be posted in the main line, but appear in a footer, sidebar or something else. It's a microblog like thing.

TODO Look into having an automated list of updates on my blog check

https: Maintenance | Hugo

After first deployment

TODO add search functionality BUILD

https://gohugo.io/tools/search/

https://gist.github.com/eddiewebb/735feb48f50f0ddd65ae5606a1cb41ae looks ok-ish

The basic method is the same, generate some data file to query with client side code. Typically a json file. While this is fine for small sites, this doesn't scale very well, the json file gets loaded completely for every search. At some point a smarter solution is needed.

In terms of implementation, typically some template is constructed, which is then generated alongside the pages, uploaded so the client side code can access it.

fuse.js gets the most votes

The theme I am using is planning an implementation for search in their 2.0 version which may be fine to wait for.

TODO integrate existing comments BUILD

They are not on site right now. But I have json exports from the disqus days.

No need to do this directly, but I don want to save the data.

I like the idea of staticman although i do not want to rely on github for the comments, I think it is possible to self host, but this may be quite a bit of work. Probably postponing.

I also like the idea of having the comments on a social site, and not having them directly on the blog itself, but that has a few nicknacks as well. I haven't seen an implementation yet which I liked.

TODO When viewing a post, show related content in the sidebar BUILD

TODO look at automatically minifying CHECK

I think there is a section in the documentation about the hugo pipe which may be related to this? Ideally this should be an option in the staging/production environment to set

TODO After deploy: reorganize content files BUILD

There are a couple of folders below content which shouldn't really be there. The files folder should be examined and most of the files should probably move to its relevant document / posting where the post is converted to a page bundle. The downside is that the filesystem will then be littered with all these files and it will be hard to optimize for them or move them to a CDN if needed.

TODO Extend the deployment script WRITE

hugo deploy does only supports some cloud services, so i just need some sort of script to rsync the whole shebang.

Options:

1. rsync script to sync .site dir

   This is basically what the old site did, manual action, no frills. Good way to start I guess. I did this now using a local git alias `git deploy` which runs an rsync script from the root of the repository. I would have liked a post-push-hook, but apparently that is not the proper way to use git according to the bosses.

2. Use ansible for deployment

   Another option is to use an ansible script which can do the rsync, but also help configure the webserver and other things that may help manage the deployment. The latter part is unquestioned, but I'd hesitatie to use ansible to publish content, if only for speed reasons.

3. Post receive hook on server

   An alternative deploy option could be a post-receive hook on the server side, this means only git and editor is enough (locally) to publish content, instead of hugo/git/rsync. This requires hugo on the server though, so it's not much better in general.

   The site would then run a script after each push, taking care of updating the site on the right location on the server. An extra pro is that the amount of content that needs to go over the connection is probably a lot less (although rsync should be fairly efficient in just transferring the differences to what has already been published)

TODO Make the gpx shortcode into a hugo module WRITE

Or even better: use https://github.com/altrdev/hugo-leaflet

This requires fontawesome which is also helpful for some other items such as using a calendar icon and share icons.

TODO Look at https://fed.brid.gy CHECK

It's a procedure to bring *normal* websites into the fediverse without having a framework, it's a modular setup which can be implemented in different levels, depending on the need.

Not sure where I would use this though.