RSS Generation

How to generate RSS feed via command line

Yann Esposito

[2019-09-30 Mon]

TL;DR: To generate an RSS file you need to provide many metadatas. Those metadata are not part of all HTML files. So generating RSS from a tree of HTML file is not straightforward. Here is the script I use.

RSS Problem

RSS feed is meant to declare updates and new articles for a website. Each RSS entry must therefore have a date, an unique id, a title, maybe some categories, etc...

For most blog platform or even static website generator, those meta infos are clearly put in the sources or in some DB.

I use `org-mode` for generating my website, and the `ox-rss` is quite slow when generating an RSS with the full content of each item. Mainly, the way to achieve full content of my articles inside an RSS with `ox-rss` is by first creating a very big org file containing all the articles, and then transforming it in RSS. And this is very slow (several minutes).

So a simpler idea inspired by lb[1] is to generate the RSS directly from the generated HTML files. The only difficulty is to find the metadata inside those HTML. Unfortunately there is no real standard for all those metas.

So to use the HTML as source you'll need to "parse" the HTML file. For that purpose I use `html-xml-utils`[2].

I wrote a simple zsh script; it starts with lot of variables to fill:

```
# Directory
webdir="_site" # directory containing your website html files
postsdir="$webdir/posts" # directory containing the articles
rssfile="$webdir/rss.xml" # the RSS file to generate
```

[1] https://github.com/LukeSmithxyz/lb
[2] https://www.w3.org/Tools/HTML-XML-utils/

```
# maximal number of articles to put in the RSS file
maxarticles=10

# RSS Metas
rsstitle="her.esy.fun"
rssurl="https://her.esy.fun/rss.xml"
websiteurl="https://her.esy.fun"
rssdescription="her.esy.fun articles, mostly random personal thoughts"
rsslang="en"
rssauthor="yann@esposito.host (Yann Esposito)"
rssimgtitle="yogsototh"
rssimgurl="https://her.esy.fun/img/FlatAvatar.png"
```

Then I set the accessor to extract the information I want from HTML files. It is quite unfortunate that there is no really strong convention for where to put article dates, article author email. There are metas for title and keywords thought.

```
# HTML Accessors (similar to CSS accessors)
dateaccessor='.article-date'
contentaccessor='#content'
# title and keyword shouldn't be changed
titleaccessor='title'
keywordsaccessor='meta[name=keywords]::attr(content)'
```

A few helper functions:

```
formatdate() {
    # format the date for RSS
    local d=$1
    LC_TIME=en_US date --date $d +'%a, %d %b %Y %H:%M:%S %z'
}

finddate(){ < $1 hxselect -c $dateaccessor }
findtitle(){ < $1 hxselect -c $titleaccessor }
# retrieve the content, take care of using absolute URL
getcontent(){
    < $1 hxselect $contentaccessor | \
                perl -pe 'use URI; $base="'$2'"; s# (href|src)="((?!https?://)[^"]*)"#" ".$1."=\"".URI->new_abs(
findkeywords(){ < $1 hxselect -c $keywordsaccessor | sed 's/,//g' }
mkcategories(){
    for keyword in $*; do
        printf "\\n<category>%s</category>" $keyword
    done
}
```

The `mkcategories` will be used to add an RSS category for each keyword. And

finally the real loop doing the work:

```
tmpdir=$(mktemp -d) # create a temporary work dir
typeset -a dates    # an array to save dates of all articles
dates=( )

# for each HTML file we generate the XML for the item in a file
# named ${d}-$(basename $fic).rss that naming convention will be useful to
# sort article by date
for fic in $postsdir/**/*.html; do
    postfile="$(echo "$fic"|sed 's#^'$postsdir'/##')"
    blogfile="$(echo "$fic"|sed 's#^'$webdir'/##')"
    printf "%-30s" $postfile
    xfic="$tmpdir/$fic.xml"
    mkdir -p $(dirname $xfic)
    hxclean $fic > $xfic # create a cleaner HTML file to help hxselect work
    d=$(finddate $xfic)
    echo -n " [$d]"
    rssdate=$(formatdate $d)
    title=$(findtitle $xfic)
    keywords=( $(findkeywords $xfic) )
    printf ": %-55s" "$title ($keywords)"
    # up until here, we extracted the informations we need for the item
    categories=$(mkcategories $keywords)
    absoluteurl="${websiteurl}/${blogfile}"
    { printf "\\n<item>"
      printf "\\n<title>%s</title>" "$title"
      printf "\\n<guid>%s</guid>" "$absoluteurl"
      printf "\\n<pubDate>%s</pubDate>%s" "$rssdate"
      printf "%s" "$categories"
      printf "\\n<description><![CDATA[\\n%s\\n]]></description>" "$(getcontent "$xfic" "$absoluteurl")"
      printf "\\n</item>\\n\\n"
    } >>  "$tmpdir/${d}-$(basename $fic).rss"
    # we append the date to the list of dates
    dates=( $d $dates )
    echo " [${fg[green]}OK${reset_color}]"
done

# Now we publish the items in reverse newer articles first
echo "Publishing"
for fic in $(ls $tmpdir/*.rss | sort -r | head -n $maxarticles ); do
    echo "${fic:t}"
    cat $fic >> $tmpdir/rss
done

# we get the latest publish date
```

```
rssmaxdate=$(formatdate $(for d in $dates; do echo $d; done | sort -r | head -n 1))
# we put the current date for the latest build date
rssbuilddate=$(formatdate $(date))

# we generate the RSS file
{
# Write the preamble of the RSS file
cat <<END
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0"
       xmlns:content="http://purl.org/rss/1.0/modules/content/"
       xmlns:wfw="http://wellformedweb.org/CommentAPI/"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:atom="http://www.w3.org/2005/Atom"
       xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
       xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
       xmlns:georss="http://www.georss.org/georss"
     xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
     xmlns:media="http://search.yahoo.com/mrss/"><channel>
  <title>${rsstitle}</title>
  <atom:link href="${rssurl}" rel="self" type="application/rss+xml" />
  <link>${websiteurl}</link>
  <description><![CDATA[${rssdescription}]]></description>
  <language>${rsslang}</language>
  <pubDate>${rssmaxdate}</pubDate>
  <lastBuildDate>$rssbuilddate</lastBuildDate>
  <generator>mkrss.sh</generator>
  <webMaster>${rssauthor}</webMaster>
  <image>
    <url>${rssimgurl}</url>
    <title>${rssimgtitle}</title>
    <link>${websiteurl}</link>
  </image>
END

# write all items
cat $tmpdir/rss

# close the RSS file
cat <<END
</channel>
</rss>
END
} > "$rssfile"

# cleanup temporary directory
```

```
rm -rf $tmpdir
echo "RSS Generated"
```

Full script

Here is the full script I use:

mkrss.sh

You can notice I start my script with:

```
#!/usr/bin/env nix-shell
#!nix-shell -i zsh
```

The `nix-shell` bang pattern is a neat trick to have all the dependencies I need when running my script. It takes care that `zsh`, `coreutils` and `html-xml-utils` are installed before running my script. For example my script uses `date` from GNU coreutils and not the `BSD` date from my OS, which makes the script more portable. This also take care of using the URI perl package.

Along my script I have a `shell.nix` file containing:

```
# { pkgs ? import <nixpkgs> {} }:
{ pkgs ? import (fetchTarball https://github.com/NixOS/nixpkgs/archive/19.09.tar.gz) {} }:
  pkgs.mkShell {
    buildInputs = [ pkgs.coreutils pkgs.html-xml-utils pkgs.zsh pkgs.perl pkgs.perlPackages.URI ];
  }
```

Mainly it *pins* a package version and the list in `buildInputs` contains the packages to install locally.

If you are not already using nix[3] you should really take a look. That `shell.nix` will work on Linux and MacOS.

[3] https://nixos.org/nix