YAG-OSDL Homepage

Organisation: Copyright (C) 2004-2021 Olivier Boudeville

Contact: about (dash) yag-osdl (at) esperide (dot) com

Creation date: Sunday, August 17, 2008

Lastly updated: Sunday, March 14, 2021

Status: Stable

Version: 0.8.0

Dedication: Users of the `YAG-OSDL` tool.

Abstract: The role of YAG-OSDL is to generate static HTML galleries out of a filesystem tree containing snapshots (camera pictures).

Browsing this content can be done according to the tree structure (per gallery, through which one can navigate and that can be nested, each with subgalleries containing thumbnails) or based on themes that themselves form a separate hierarchy (any theme may be further split into subthemes recursively).

Besides themes, a comment may be attached to a snapshot, and will be displayed accordingly.

The next level of information is to read the corresponding source files, which are intensely commented and generally straightforward.

Overview

YAG-OSDL is a GPL'd cross-platform tool whose purpose is to generate a static website from a set of image files stored in a directory tree, in order to build a full web gallery allowing to browse the pictures pleasantly afterwards, based on navigation options and thumbnails.

Hierarchical thematical sort and comments are supported (see an older example of result).

Purpose of YAG-OSDL

YAG-OSDL is made of a set of Python scripts that scan a directory tree containing images (ex: JPEG files), and construct out of them a set of HTML pages with thumbnails, offering the possibility to skim through the pictures thanks a standard browser, and according to two different ways:

- in a **tree-based fashion**: the user can browse the images according to their directory structure
- in a **theme-based fashion**: the user selects a theme and sees which are its sub-themes and which resources belong to that theme (of course it implies that the gallery author filled the theme information thanks to our tools beforehand)

Gallery authors are indeed provided with a way of adding their comments for any picture, and also with a way of describing to which theme(s) a given picture belongs.

YAG-OSDL can be easily customised, thanks to its configuration file, whose default name is `yag-osdl.conf`. Everything can be generated, the portal page included, with comments and terms of use. Web themes ("skins" for galleries, not to be confused with content themes) are supported, and can be customised thanks to CSS (*Cascading Style Sheets*).

We developed YAG-OSDL as we needed a tool that would archive our graphical resources so that we could browse quickly through them, either for holiday pictures or to select the assets suitable for, say, a video game: another way of describing YAG-OSDL is indeed: "a media content browser". We plan to add archiving support for audio content, as soon as we feel enough need for such a feature.

Installation

Current version of YAG-OSDL is 0.8. It relies on Python 3 (ex: 3.9.2 at the time of this writing) and uses a pretty standard `requirements.txt` file in order to secure its third-party prerequisites thanks to `pip`. The only extra prerequisite is Ceylan-Snake.

Inspiration

YAG-OSDL derives from YAG, a previous work from Stas Z (`linuxisbeter (at) yahoo (dot) com`); thanks!

Step-by-step Mini User's Guide

Objective

You have a directory tree (possibly only a directory, preferably a pre-sorted tree) of image files, such as:

```
MySnapshots
|-- GoldWashing
|   |-- 200407-Chilhac-0096.jpeg
|   |-- 200407-Chilhac-0099.jpeg
|   |-- 200407-Chilhac-0103.jpeg
|   |-- 200407-Chilhac-0113.jpeg
|   |-- 200407-Chilhac-0114.jpeg
|   |-- 200407-Chilhac-0115.jpeg
|   |-- 200407-Chilhac-0116.jpeg
|   `-- 200407-Chilhac-0117.jpeg
`-- InsideVillage
    |-- 200407-Chilhac-0087.jpeg
    |-- 200407-Chilhac-0088.jpeg
    |-- 200407-Chilhac-0089.jpeg
    |-- 200407-Chilhac-0090.jpeg
    `-- 200407-Chilhac-0091.jpeg
```

You want to generate a full web gallery out of it, and you are using GNU/Linux (the Windows platform cannot benefit from helper shell scripts, which will ease the work of the gallery author).

You might have relevant, overall (general) gallery information to associate to this content; let's suppose that you wrote them down in a text file (ex: `MyInfos.txt`). If your gallery is to be put online, you might have thought to a license and terms of use, that you would have written in another text file (ex: `MyLicence.txt`).

Step zero (optional): prepare the snapshots

One may take advantage of the relevant scripts in Ceylan-Hull in order to better manage pictures, namely to fix their filenames as a whole (see `rename-snapshots.sh` for that) and to remove any associated metadata (typically EXIF information) before publishing (not to disclose timestamps, locations, etc.), thanks to `remove-snapshot-metadata.sh`.

One may then, once these metadata have been applied (images having been made upright) and cleared, correct the orientation (as some camera mess with them) of the remaining pictures that need it.

For that, one may use `ImageMagick` for that, precisely:

```
# Counter-clockwise, in-place rotation:
$ mogrify -rotate "-90" foobar.jpeg
```

Step one: install the YAG-OSDL tooling

Python First, retrieve and install prerequisites, first of which is Python, precisely `Python 3` (we use `3.9.2` at the time of this writing).

Most probably that your distribution allows to take care of it (example for Arch Linux : `pacman -Sy python3`; users of Debian-based distributions can use `apt-get install python3`).

Check your actual version with `python -V`.

Ceylan-Snake `YAG-OSDL` belongs to `OSDL-Snake`, which relies on `Ceylan-Snake`.

This is just a matter of:

```
$ cd ~/my-projects
$ git clone https://github.com/Olivier-Boudeville/Ceylan-Snake
```

Set in your environment a `CEYLAN_SNAKE` variable pointing to that clone; ex:

```
$ export CEYLAN_SNAKE="${HOME}/my-projects/Ceylan-Snake"
```

YAG-OSDL itself Then obtain the YAG-OSDL codebase thanks to, for example:

```
$ cd ~/my-projects
$ git clone https://github.com/Olivier-Boudeville/OSDL-Snake
$ cd OSDL-Snake
```

Then the needed Python3 packages, which are listed in requirements.txt, can be installed with:

```
$ python3 -m pip install -r requirements.txt
```

This only involves installing actually `Pillow` (which supersedes `PIL`).

Step two: edit your YAG-OSDL configuration file

Simplest solution is to derive it from the sample one:

```
$ cp yag-osdl.conf.sample yag-osdl-for-foobar.conf
```

We hope that the sample is self-describing enough for most uses; yours could be edited that way:

```
[Options]

# Now can be mostly any string:
project_name          = Moon Photos

# One may prefer absolute paths for simpler management.

content_directory     = /var/my-encrypted-storage/www/Moon-sources
#resource_directory    = /home/dalton/Projects/Tools/yag/yag-osdl-latest/resources

output_in_content     = False
output_directory      = /var/my-encrypted-storage/www/Moon

#language = English
language = French

#theme                 = OSDL-english-theme
theme                 = OSDL-french-theme

thumbsize             = 120
images_by_row         = 4
images_by_column      = 4
dash_is_space_in_menu = True

author                = William Dalton
author_mail           = william.dalton@maverick.org
```

Step three: annotate the images, thanks to our helper scripts

To Trigger the Annotation Process If we launch YAG-OSDL now, a full gallery will be generated, yet no comment nor theme information will be available.

If one wants to provide them, just enter, for example, from the OSDL-Snake root:

```
$ MY_CONTENT_ROOT="/var/my-encrypted-storage/www/Moon-sources"
$ ./annotate-images.sh ${MY_CONTENT_ROOT}
```

We hereby suppose that two really common tools are available on your computer:

- a text editor, `emacs` or any that you like; just define the `EDITOR` environment variable accordingly (ex: `export EDITOR=vim`) to override
- an image viewer; various ones will be looked-up; to select a particular one, just define the `IMAGE_VIEWER` environment variable accordingly

Commenting Galleries & Images The annotate-images.sh script (which uses handle-image.sh) will guide you and will trigger the appropriate tools.

That is, it will first fire a text editor for each encountered *gallery* when scanning your content tree.

On the example above, it would be first `MySnapshots`, then `GoldWashing`, and then `InsideVillage`. In each of the corresponding files, feel free to give overall information relative to the content in that gallery directory.

After this gallery-commenting phase, the next phase will be the *per-content* commenting. Indeed, for each image in the content tree, `annotate-images.sh` will launch the specified image viewer to display that picture (so that you are reminded of it), and then a first text editor is fired, where you may enter any *comments*[1] you may have regarding that content.

When this comment writing is over, closing the text editor will trigger a new instance of it, as explained in the section.

Adding Thematical Information This time, the *theme information*[2] (if any) associated to this image is to be specified.

Per-Content Themes How to tell that your image belongs to themes `Surf` and `Hawaï`, and that the theme `Surf` is a sub-theme of the `Awful activities` theme? Simply, in the spawned theme file, enter on the first line: `Awful activities: Surf` and in the second `Hawaii`. That's it, YAG-OSDL has all the information that it needs in order to perform its sorting work.

You can therefore close the image window (you have finished with that one) and this theme file, and repeat the process:

1. view the current image
2. enter its comment (if any) and close the text editor
3. enter its theme information (if any), close the image viewer and the text editor (preferably in that order)
4. repeat from point 1 until `annotate-images.sh` stops with `Annotations finished !`, which means that you went successfully through the whole comment and thematical enrichment process

Note that you can at any time stop the annotation script, as no entered information will be lost: by re-launching the script the same way that you launched it, you will be able first to validate or update every already available information you gave, until you reach the point where you stopped. You will then just have to continue the enrichment process from then on.

[1] The files dedicated to comments bear the `.txt` extension.

[2] The files dedicated to themes bear the `.thm` extension.

Another Way of Defining the Theme Hierarchy There is an alternative or complementary method in order to define your theme tree: instead of specifying the link between themes at the level of a given image[3], a standalone theme file whose name is `yag-overall-themes.thm` can be used. This file should be placed at the root of your content, and might have the usual structure:

```
father-theme: a-child-theme
```

as shown here:

```
Awful activities: sailing
Awful activities: surf
Awful activities: yoga

surf: Hawaii-2003
surf: Hawaii-2004
surf: Snapshots Of The Shark That Ate My Board
```

This standalone theme file proved useful, since, that way, all the theme tree can be defined and understood in one place.

However, any combination of theme specifications will work: one can both use image theme files and the standalone theme file.

Step four: launch YAG-OSDL

Still following the example, one just have to run:

```
$ run-yag-osdl.sh --config yag-osdl-for-foobar.conf
```

Step five: admire the result

Use your browser to inspect your stunning new gallery, improve comments and themes if necessary and, if you feel like it should be put online, copy the whole content directory to a webserver, add a link to the gallery main page and tell all your friends about it!

Licence

The one of the original work applies, namely the GNU General Public License (GPL), version 3.0 or later.

Support

Bugs, questions, remarks, patches, requests for enhancements, etc. are to be reported (please specify it relates to YAG-OSDL) to the project interface (typically issues) or directly at the email address mentioned at the beginning of this document.

[3] Knowing that a given father/child theme relationship can be specified only once, at the level of any related content.

Please React!

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, follow the Support guidelines).

Have fun with YAG-OSDL!