photo, publier, web, piwigo

Avant de commencer

Quand j'écris “avant de commencer” c'est sérieux, très sérieux. Je n'ai pas trouvé comment relire les tags *IPTC* des photos d'un album chargé dans Piwigo directement via le formulaire prévu avant ces manipulations et j'ai donc dû supprimer et réimporter les photos concernées.

D'une manière générale, si vous avez quelques idées précises en tête de ce que vous voulez obtenir de *Piwigo*, je recommande de tester à fond avec seulement quelques photos jusqu'à obtenir le résultat voulu en étant prêt à recommencer à 0 avant de ranger vos clichés pour de bon.

Une base de données

Sans base de données MySQL (ou équivalent), *Piwigo* ne fonctionnera pas. Il faut donc créer une base de donnée vierge (en interclassement *utf8_general_ci* et basée sur le moteur *MyISAM*) et définir l'utilisateur qui sera utilisé par *Piwigo* pour y accéder. Le plus simple avec par exemple *phpMyAdmin* est de demander la création d'un utilisateur `piwigo` avec sa base de données dédiée.

Read Metadata : un plugin indispensable

D'après ce que j'ai compris (mais je peut me tromper), sans cette extenion, la notion de tag de *Piwigo* est indépendante des photos : on crée une liste de *tags* dans la base de données *Piwigo* et on y lie les photos mais l'info n'est stockée que dans la base de données et pas dans les photos (à moins que le plugin Write Metadata permette d'écrire proprement ces tags créés dans *Piwigo* mais je n'ai pas testé).

Une fois que Read Metadata est activé, lorsque l'on importe une photo, les *tags* contenus dans la photo sont lus et intégrés à la base de données. Il permet par ailleurs (dans `Administration` ⇒ `Plugins` ⇒ `Lire les métadonnées`) d'afficher tous les champs *EXIF* et *IPTC* contenu dans une image dont on connait l'*ID*.

D'après mon expérience, il doit impérativement être installé avant d'importer les photos, mais il existe peut-être une astuce pour les autres (j'ai tenter de resynchroniser des albums existants en cochant la case [Écraser les données existantes] mais sans succès en ce qui concerne les albums virtuels).

Configuration nécessaire

Toujours pour permettre à *Piwigo* d'utiliser les tags contenus dans les fichiers (entre autres

métadonnées *EXIF* et *IPTC*), il faut ajouter des options au fichiers de configuration local (qu'il faut créer car il n'existe pas à l'installation) :

<cli>root@muffin:/# vi /var/www/geekitude.photos/local/config/config.inc.php</cli>

config.inc.php

```
#<?php
$conf['use_exif'] = true;
$conf['use_iptc'] = true;
?>
```

Initialiser la galerie

Pour cela, il suffit de naviguer jusqu'à *Piwigo* qui demandera les paramètres de la base de données qu'il doit utiliser ainsi que le nom et le mot de passe de l'administrateur.

Choisir une méthode de stockage des photos

ATTENTION : la méthode utilisée ici s'applique à un serveur hébergé chez moi. Certains hébergeur considèrent qu'une utilisation importante (et la notion est subjective) des ressources du serveur est un abus des Conditions Générales du contrat passible d'une rupture de contrat sans préavis. Autrement dit il peut être important de générer les miniatures et autres tailles d'images avant de les envoyer sur le serveur (afin que leur génération ne provoque pas une utilisation “importante” de ressources). Si votre site est hébergé chez un fournisseur, je vous recommande de lire attentivement cette rubrique de la FAQ du logiciel.

Pour simplifier, il y a deux méthodes possibles pour ajouter des photos :

- utiliser le formulaire prévu dans [Administration ⇒ [Photos] ⇒ [Ajouter] : les photos sont alors analysées et chargées vers le dossier `upload` de *Piwigo* dans une arborescence qui dépends de la date (par exemple `…/upload/2016/08/25` pour des images chargées le 25 août 2016), renommées selon une nomenclature propre à *Piwigo* (par exemple `20160825205809-c6a109e6.jpg`) et enfin liées à l'album choisi dans le formulaire d'ajout (on parle d'***albums virtuels*** dans le sens où les photos ne sont pas stockées dans une arboresence en lien direct avec l'album et n'y sont reliées que par la base de données)
- on peut aussi charger une arborescence complète de photos directement dans le répertoire `galleries` du serveur *Piwigo* **à condition de respecter quelques règles** : les noms des répertoires ne doivent contenir que des caractères alpha-numériques non accentués et pas d'espace(s) (on parle d'***albums physiques*** et les photos ne seront ni déplacées ni renommées)

puis manuellement utiliser la synchronisation ([Administration] ⇒ [Outils] ⇒ [Synchroniser]) pour qu'elles soient analysées et rendues accessibles

La première méthode est celle qui est recommandée car elle est plus souple (déplacement des albums, etc) mais j'y vois un énorme défaut : personnellement je n'aime pas (mais alors vraiment pas ) le fait que les images soient renommées. Il est bien trop compliqué de récupérer les images si l'on souhaite par exemple changer de méthode de publication ou tout simplement en cas de crash de la base de données (oui bien sûr il “suffit” de la sauvegarder mais les histoires de sauvegarde trop ancienne ou simplement corrompue sont légion)

Noter que le plugin Virtualize permet théoriquement de transformer un album physique en album virtuel (mais je ne l'ai pas testé).

Voir le wiki de Piwigo pour plus de détails.

Mon choix

Les deux méthodes à la fois ! 

Pour commencer, je crée le ou les albums virtuels qui constitueront la galerie proprement dite en une organisation par lieux ou pas événements, par exemple un album `Tanzanie` avec les sous albums `01-Arusha` et `02-Manyara` ou un album `Noëls` avec un sous-album par année.

Je charge ensuite les photos dans des *albums physiques* (donc « par arborescence ») dans des sous-dossiers du répertoire `…/galleries` et organisés de manière historique (par date de prise de vue) et en respectant les limitations de caractères citées plus haut (par exemple `…/galleries/2016/07/10/2016-07-10_06-09-00_tza_arusha.jpg`). Les *albums physiques* seront cachés pour n'être accessible qu'à l'administrateur.

Cette méthode permet de profiter de la souplesse des *albums virtuels* (quand à l'organisation de la galerie) tout en conservant les photos avec leur nom d'origine.

Dans la pratique, telle quelle, cette méthode s'avère peu pratique car d'une part il faudra cacher chaque album physique (avec le risque d'oubli que cela représente) et d'autre part la liste des albums de l'administrateur risque de devenir très conséquente. Il vaut donc mieux créer dans le répertoire `…/galleries` un premier dossier (à nommer selon vos goûts) qui sera caché et qui recevra les futurs dossiers créés pour chaque année (si un *album* est caché, ses *sous-albums* le seront aussi). Donc dans la suite de ce document, tous mes *albums physiques* correspondant chacun à une année seront des sous-dossiers de

…/galleries/CHRONOLOGIE.

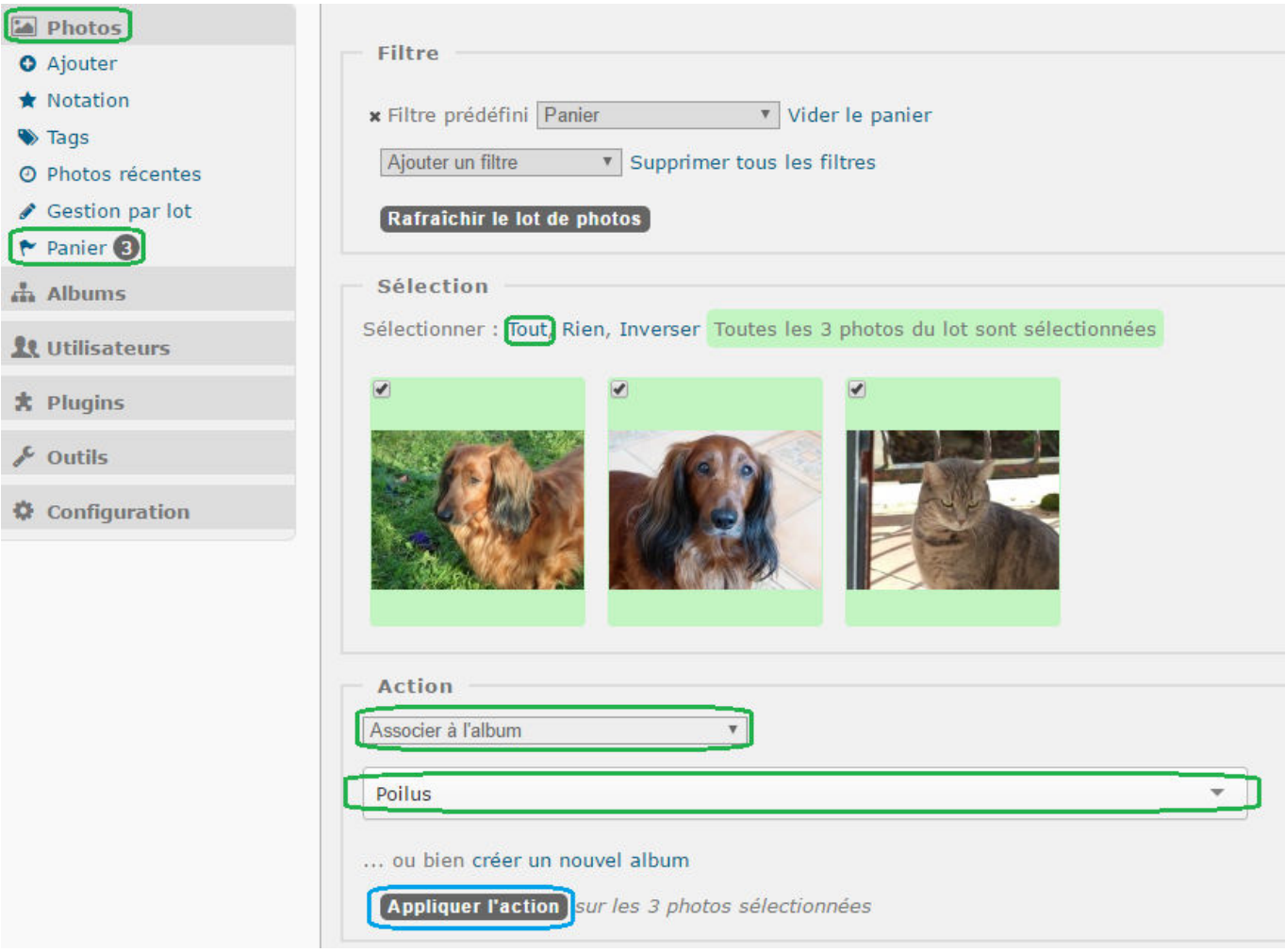
Premier album

Une fois les images placées dans l'*album physique* correspondant, il faut synchroniser l'arborescence et la base de données ([Administration] ⇒ [Outils] ⇒ [Synchroniser]) avec les options suivantes :

L'étape de simulation est importante car elle permettra de détecter d'éventuelles erreurs qui pourraient provoquer des problèmes de synchronisation (comme un nom de dossier ou de photo non valide).

Il ne reste plus qu'à recommencer après avoir désactiver la simulation pour obtenir quelque chose qui ressemble à ceci :

L'option [Ajouter les nouvelles photos au panier] permet de les retrouver facilement dans le menu [Photos] à la rubrique [Panier], qui n'est visible que lorsqu'il contient quelque chose. L'étape suivante consiste donc à lier ces photos qui sont donc dans un *album physique* à un *album virtuel* (notez qu'il suffit de cliquer sur [Accueil administration] pour accéder au *panier* ou retrouver les dernières photos dans la rubrique [Photos récentes]) :



Il faudra par contre malheureusement penser à cliquer en haut de la page sur le lien [Vider le panier]

😕.

Si l'*album physique* concerné n'est pas lui-même le sous-album d'un autre album déjà caché, ne pas oublier de le “verrouiller” pour le cacher sauf aux administrateurs en passant par la gestion des albums.

Pour aller plus loin

Plugins

Voici quelques autres plugins pratiques et/ou sympas…

- EXIF View qui améliore un peu la lisibilité des infos EXIF en traduisant certains champs
- Extended Description permet quand à lui d'élargir énormément les possibilités des descriptions que l'on peut ajouter aux albums ou aux photos
- Fotorama qui rends plus jolis les diaporama (la transition entre les photos en particulier)
- GThumb+ ❤️ qui permet d'afficher toutes les photos d'un album sur la même page sans que les performances n'en souffrent trop car par défaut, lorsqu'un album contient plus de 20 photos, *Piwigo* les affiche par pages mais surtout n'affiche plus que les *tags* liés aux photos de la page en cours (au lieu d'afficher tous les *tags* de l'album courant, ce qui, à mon avis, n'est vraiment pas intuitif)
- LocalFiles Editor ❤️ qui permet d'accéder facilement aux principaux fichiers de configuration locaux (options cachées, CSS, …)
- Media Icon qui n'a toutefois d'intérêt que si l'on prévois de gérer des fichiers autres que des images
- No Stats For Robots qui évite de comptabiliser les visites des robots du web (comme les moteurs de recherche) dans les statistiques
- Piwigo OpenStreetMap ❤️ qui ajoute des cartes *open source* à l'accueil, à chaque album (en gérant au passage automatiquement les éventuels fichiers gpx) ou encore à la page de chaque image
- PWG Stuffs qui, malgré son nom pourri, apporte quelques possibilités intéressantes pour personnaliser sa galerie en ajoutant des blocs *HTML* supplémentaires
- rightClick qui apporte un peu de sécurité (entre autres en désactivant le clic-droit sur les images)
- RV Autocomplete qui étends les possibilités de la recherche intégrée à *Piwigo*
- RV Thumbnail Scroller ❤️ qui, comme *GThumb+*, permet de remplacer le système de pages des albums longs par un *infinite scroll* bien plus pratique (les deux plugins ne sont évidement pas prévus pour être activés simultanément)
- RV DB Integrity parce qu'un outil de vérification de l'intégrité de la base de donnée ne peut pas faire de mal (la vérification n'est toutefois pas automatique, il faut passer par [Outils] ⇒ [Maintenance] et déclencher la vérification à la main)

Configuration avancée

Dans ce qui suit, je considère que le plugin LocalFiles Editor est installé mais j'indique tout de même le nom du fichier *Piwigo* à modifier à la main pour obtenir le même résultat. Avec le plugin, il suffit, sur la page d'Administration, de cliquer sur [Plugins] puis [LocalFiles Editor] et de se promener dans les onglets.

Pour le fichiers contenant des tags comme <?php et ?> (correspondant aux rubriques Configuration locale, Langues et Plugin Personnel ci-dessous), veillez à ne pas les supprimer ou les oublier.

Configuration locale

Fichier à modifier: **…**`/local/config/config.inc.php`

Pour que le plugin Read Metadata fonctionne

<cli><?php $conf['show_iptc'] = true; $conf['use_iptc'] = true; ?></cli>

Pour pouvoir uploader des fichiers autres que des images

<cli><?php $conf['upload_form_all_types'] = true; ?></cli>

Pour intégrer une liste de liens au menu principal

<cli><?php $conf['links'] = array('http://phpwebgallery.net' ⇒ 'L\'homepage PWG', 'http://forum.phpwebgallery.net' ⇒ 'Son forum', 'http://phpwebgallery.net/doc' ⇒ 'Sa documentation'); ?></cli>

Afficher tous les tags

En tout cas dépasser la limite initiale de 20 tags (la seconde ligne ne concerne évidemment que le plugin Menu Tags).

<cli><?php $conf['full_tag_cloud_items_number'] = 2000; $conf['menubar_tag_cloud_items_number'] = 2000; ?></cli>

URL Rewriting

Par défaut, les URLs *Piwigo* sont franchement moches (par exemple : *mon.site.fr/index.php?/category/10*).

Il est heureusement possible d'améliorer un petit peu les choses...

Voici les ajouts nécessaires au fichier de configuration local :

<cli><?php $conf['question_mark_in_urls'] = false; $conf['php_extension_in_urls'] = false; $conf['category_url_style'] = 'id-name'; ?></cli>

La dernière option ci-dessus ajoute juste le nom de la catégorie en cours à son identifiant numérique. Les deux premières désactivent respectivement le `?` et `.php` dans les URL mais il faut modifier la configuration du site web en lui même.

Pour permettre la suppression du `?`, il faut que la directive *AcceptPathInfo* soit activée (ce qui est plutôt simple), et, pour la suppression de `.php`, il faut activer l'*URL rewriting* proprement dit et indiquer une règle. Pour faire tout cela, voici (en vert et bleu ci-dessous) comment adapter la configuration du site :

<cli>

```
      <Directory /var/www/geekitude.photos/>
              Options Indexes FollowSymLinks <color
#00a2e8>MultiViews</color>
              AllowOverride None
              Order allow,deny
              allow from all
              <color #22b14c>AcceptPathInfo On</color>
              <color #00a2e8><IfModule mod_rewrite.c></color>
                      <color #00a2e8>Options -MultiViews</color>
                      <color #00a2e8>RewriteEngine On</color>
                      <color #00a2e8>RewriteCond %{REQUEST_FILENAME}.php -
f</color>
                      <color #00a2e8>RewriteRule ^([^/.]+)/?(.*)$ /$1.php/$2
[QSA,L]</color>
              <color #00a2e8></IfModule></color>
      </Directory>
```

</cli>

Par la suite, les URLs proposées par *Piwigo* ressembleront à ceci : **mon.site.fr/index/category/10-bretagne**.

A ce jour (16/10/2017), cacher le `?` désactive malheureusement l'affichage automatique des traces *GPS* dans les cartes affichées par le plugin *OpenStreetMap*.

CSS

Fichier à modifier: **…**`/local/config/???`

On peut appliquer ici n'importe quelle règle *CSS* qui remplaceront les valeurs initiales. Voici quelques exemples très simples pour ajouter un peu de couleurs et éviter un effet de déplacement horizontal lorsque la page dépasse la hauteur de l'écran en forçant l'affichage de l'ascenseur vertical :

<cli>html { overflow-y: scroll; } #theHeader h1 { color: #007acc }.tagLevel1 { color: #ccebff; } .tagLevel2 { color: #99d6ff; } .tagLevel3 { color: #66c2ff; } .tagLevel4 { color: #33adff; } .tagLevel5 { color: #0099ff; }</cli>

Templates

Fichier à modifier: **…**`/local/config/???`

Langues

Fichier à modifier: **…**`/local/config/???`

Je suppose qu'il s'agit ici de personnaliser les chaînes de caractères utilisées par l'interface de *Piwigo* (test à venir… un jour… peut-être…)

Plugin personnel

Fichier à modifier: **…**`/local/config/???`

La grosse bidouille du Geek

L'une de ces idées qui me trottaient en tête depuis un moment et avant même de découvrir *Piwigo* était de pouvoir choisir entre l'affichage de toutes les photos d'une galerie et seulement celles prises avec un modèle précis d'appareil photo. Pendant que je cherchait la solution, l'idée de pouvoir faire la même chose avec le nom du pays, de la province, de la ville et du lieu de prise de vue est venue d'elle-même.

Piwigo en est capable mais cela dépasse la simple configuration ou la petite bidouille.

L'objectif passe par deux étapes : collecter les informations voulues pour toutes les photos (ce qui doit se faire avant de commencer le rangement proprement dit) et comment l'utiliser. Pour le moment, je n'ai cherché et trouvé que la solution à la première étape.

Ces informations existent en tant que donnée *EXIF* (pour le modèle d'appareil photo) ou *ITPC* (si elles ont été ajoutées) incluses aux fichiers mais pas dans la base données, or, pour la suite (c'est à dire la requête qui va permettre de filtrer uniquement les clichés pris avec tel ou tel modèle d'appareil) il faut que l'information soit disponible dans la base de données.

C'est cette page du wiki qui m'a mis sur la voie et voici la requête SQL utilisée : <cli>ALTER TABLE `piwigo_images` ADD `model` VARCHAR(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL , ADD `city` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL , ADD `sub_location` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_general_ci

NULL DEFAULT NULL , ADD `province_state` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL , ADD `country_name` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL ;</cli>

Ajout de nouveaux champs dans la base de données

Choisir où stocker l'information est simple, il suffit de jeter un œil à la structure de la base de données pour voir qu'il existe une table `piwigo_images` qui est toute indiquée (elle contient déjà toutes les infos comme les dates, les éventuels commentaires, l'auteur, ...).

J'ai choisit (totalement arbitrairement) d'ajouter ces champs après les autres (évidement dans la table `piwigo_images` puisque c'est elle qui contient les informations des images). Choisir le nom ou l'interclassement de chaque champs a été tout aussi simple (arbitrairement pour le nom et identique à l'existant pour l'interclassement). Pour pour le type de données j'ai choisit VARCHAR afin de pouvoir stocker une valeur `NULL` si besoin. Le nom de modèle le plus long des différents appareils dont je dispose à ce jour est constitué de 12 caractères mais impossible de savoir ce que réserve l'avenir donc j'ai choisit une longueur maximale de 20 caractères pour commencer, 50 pour les autres champs (c'est facilement modifiable par la suite, ce n'est qu'une longueur maximale).

Configurer Piwigo

Comme pour les données *IPTC*, il faut configurer *Piwigo* pour qu'il prenne en charge les données *EXIF* en lui ajoutant quelques lignes de configuration (voir ci-dessous pour la manière de procéder) en modifiant le fichier de configuration. Ajouter uniquement le champs voulu a bien fonctionné mais a aussi perturbé l'enregistrement de la date de création (la date était bien là mais plus l'heure).

Par ailleurs, quitte à remplir correctement la base de données, j'ai voulu creuser pour trouver comment remplir correctement les quelques autres champs prévus dans la base de données (à ce jour, seuls les *tags* m'intéressent mais peut-être que cela changera).

La complexité viens de trouver le lien entre les champs proposés par les logiciels d'images (*GeoSetter* ou *XnView* dans mon cas) et *Piwigo*. La tâche est d'autant plus ardue que tous les logiciels ne sont pas d'accord entre eux : ce que *XnView* appelle “auteur” et considéré comme l'auteur des libellés dans *GeoSetter* 0-8 mais *XnView* est considéré comme une référence et utilise le même champs que *Piwigo* alors que *GeoSetter* est un vieux logiciel dont le développent a cessé depuis longtemps et je n'ai pas trouvé le champs correspondant au fameux élément *ITPC* #080 supposé être réellement l'auteur.

Bref, voici le résultat obtenu après quelques tests qui m'auront coûté pas mal de cheveux blancs :

<cli><?php $conf['show_exif'] = true; $conf['use_exif'] = true; $conf['use_exif_mapping'] = array(

```
'date_creation' => 'DateTimeOriginal',
'model' => 'Model'
```

); $conf['show_iptc'] = true; $conf['use_iptc'] = true; $conf['use_iptc_mapping'] = array(

```
'keywords' => '2#025',
'city' => '2#090',
```

```
'sub_location' => '2#092',
'province_state' => '2#095',
'country_name' => '2#101',
'name' => '2#105',
'comment' => '2#120',
'author' => '2#122'
```

); ?></cli>

Chaque ligne indique quel champs de la base de données doit être utilisé pour quel élément *EXIF* ou *IPTC*.

Avec cette configuration, voici ce que *Piwigo* va récupérer dans sa base de donnée et l'élément correspondant dans chacun des deux logiciels cités (rappel : le champs `model` n'existe pas par défaut dans la base de données *Piwigo* et est directement intégré aux photos par les appareils) :

Base de données	**GeoSetter**	**XnView**
date_creation	Date de prise de vue	Date et heure de création
name	Titre	Titre
model	-	-
comment	Intitulé	Légende
author	Auteur de libellés	Auteur

C'est plus flagrant pour les éléments géographiques.