

Conception d'une galerie d'images générée à la volée

par [Pierre-Baptiste Naigeon](#)

Date de publication : 15 Septembre 2006

Dernière mise à jour : 15 Septembre 2006

Comment construire une application capable de lister toutes les images (gif, jpg, png) d'un répertoire (et de ses sous-répertoires), de générer des miniatures de ces images si elles n'existent pas encore, puis d'afficher les miniatures avec un lien sur l'image originale.

- I - Introduction
 - I-A - Précisions et pré-requis
 - I-B - Détail des opérations à effectuer
- II - Travail préparatoire
 - II-A - Parcours du répertoire
 - II-B - Test du format de fichier
- III - Génération de la galerie
 - III-A - Création des miniatures
 - III-B - Création du tableau de liens
 - III-C - Affichage des images
- IV - Conclusion

I - Introduction

I-A - Précisions et pré-requis

Dans ce tutorial, je n'ai pas souhaité passer par une base de données, afin que le code soit facilement exploitable. Cela aurait pu permettre une gestion plus fine des miniatures (tests de mise à jour, etc.), mais je n'ai pas retenu cette solution.

Nous n'utiliserons pas non plus de système de template, toujours dans un but de simplicité et de facilité de déploiement.

Mes fonctions et styles seront tous dans un seul et unique fichier, afin de faciliter encore plus le déploiement.

Je veux simplement déposer mon fichier PHP à la racine d'un répertoire, et qu'il se débrouille tout seul :)

Pour aborder ce tutorial, un minimum de connaissances en PHP vous sera demandé.

Les différents traitements de l'image s'effectuent à l'aide de la **librairie GD2**. Il faut donc penser à l'activer sur son serveur web.

I-B - Détail des opérations à effectuer

Essayons de penser en terme de programmation : quelles seront les étapes à effectuer pour en arriver au résultat souhaité ?

Voici celles que j'ai retenues :

- Lister tous les fichiers du répertoire et des sous-répertoires.
- Pour chaque fichier, vérifier s'il est de type image et correspondant à l'un des trois formats retenus (gif, jpg et png).
- Pour chaque image, vérifier si la miniature existe déjà ou pas (si elle existe, elle portera le même nom que l'original, mais sera située dans un répertoire 'miniature' situé immédiatement sous son original).
- Si la miniature n'existe pas, il va falloir la générer (et créer le répertoire 'miniature' s'il n'existe pas, mais c'est du détail).
- Générer le lien de la miniature vers son original dans un tableau global.
- Et enfin, une fois tous les liens générés, afficher la galerie (je serai très sommaire sur cette partie, chacun l'accordera à sa sauce...).

II - Travail préparatoire

Passons maintenant au code de l'application, étape par étape, en suivant l'ordre donné ci-dessus.

Définissons simplement la structure de notre page avant toute chose :

Structure générale de la page

```
<?php
    // Active tout les warning. Utile en phase de développement
    // En phase de production, remplacer E_ALL | E_STRICT par 0
    error_reporting(E_ALL | E_STRICT);
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Affichage images</title>
<style type="text/css">
<!--
/* Style défini pour ne pas avoir l'immonde bordure bleue autour des images. */
a img {
    border-color:transparent;
}
-->
</style>
</head>
<body>
<?php
    // Fonction chargée de lister le répertoire en cours
    // Et ses sous répertoires, ainsi que récupérer toutes
    // Les informations nécessaires.
    liste_repertoire('.');

    // Fonction chargée de l'affichage des données.
    affichage();
?>
</body>
</html>
```

Comme vous le voyez, nous avons fait appel à deux fonctions seulement dans le corps de notre page (liste_repertoire et affichage). Il ne nous reste plus maintenant qu'à les définir petit à petit.

A partir de maintenant, tout le code sera à rajouter tout en haut du fichier précédemment créé, entre les balises <?php et ?>, sous la commande "error_reporting(E_ALL | E_STRICT);".

II-A - Parcours du répertoire

Tâchons dès maintenant d'éviter les écueils. Lors du listing d'un répertoire, PHP nous retournera, outre les fichiers/dossiers visibles via l'explorateur, les dossiers "." et "..". Il faudra prendre garde à ne pas les traiter.

Rappelons-nous que les miniatures seront stockées dans un sous-répertoire 'miniature' du dossier courant. Il faudrait veiller à ne pas le traiter non plus, afin d'éviter de se retrouver avec des miniatures de miniatures....

Définissons donc un tableau contenant ces restrictions : \$tabl_exclus.

Voici une fonction basique permettant de lister les fichiers d'un répertoire et de ses sous répertoires, et tenant compte des remarques ci-dessus.

Lister les fichiers d'un répertoire

```

$tabl_exclus = array ('.', '..', 'miniature');

// Parcours le répertoire courant et tout ses sous-répertoires récursivement.
function liste_repertoire($dir) {
    if ($handle = opendir($dir)) {
        while (($file = readdir($handle)) !== false) {
            $chemin_fichier = $dir.'/'.$file;
            if (is_dir($chemin_fichier)) {
                if (!in_array($file, $GLOBALS['tabl_exclus'])) {
                    liste_repertoire($dir.'/'.$file);
                }
            } else {
                // Si mon fichier est une image
                $chemin_miniature = $dir.'/miniature/'.$file;
                // Génère miniature
                // Génère le lien
            }
        }
    }
    closedir($handle);
}
}

```

Voici la traduction en langage naturel du code ci-dessus :

```

Si on arrive à ouvrir le répertoire passé en paramètre
  Tant qu'il y a des choses à lire dans ce répertoire
    On génère le chemin complet du fichier
    Si le fichier est un répertoire
      Si le répertoire n'est pas exclus
        On rappelle la fonction avec le nouveau répertoire en paramètre
      Sinon
        Si le fichier est une image
          On crée le chemin de la miniature
          Si la miniature n'existe pas
            On génère la miniature
          Fin si
          On crée le lien
        Fin si
      Fin si
    Fin si
  Fin tant que
Fin si

```

Comme vous le voyez, rien de bien sorcier ici.

Il nous reste donc à définir une fonction de test du format de fichier, une de test d'existence de la miniature, une de génération, et une de création du lien.

II-B - Test du format de fichier

Concentrons-nous maintenant sur le test du format de fichier.

J'ai déclaré que seuls trois types de fichiers nous intéressent : gif, jpg et png. Nous allons donc définir un tableau qui contiendra les types MIME de ces fichiers.

De plus, comme dans cette fonction nous allons récupérer des informations qui nous serviront plus tard, nous allons définir trois variables globales : largeur, hauteur, et extension.

Vérification du format de fichier

```

// Teste si le fichier passé en paramètre correspond à l'un des trois type d'image défini
function est_image($chemin_fichier) {
    if (list($GLOBALS['largeur'], $GLOBALS['hauteur'], $type) = getimagesize($chemin_fichier)) {
        $type = image_type_to_mime_type($type);
        if (in_array($type, $GLOBALS['types_ok'])) {

```

Vérification du format de fichier

```
        $ext = explode("/", $type);
        $GLOBALS['extension'] = $ext[1];
        return true;
    }
}
return false;
}
```

Cette fonction renvoie TRUE si le fichier est une image qui correspondant à nos critères, FALSE sinon.

Traduit en langage naturel, voici ce que fait cette fonction :

```
Si le fichier est une image
    On récupère son type MIME
    Si son type correspond à nos attentes
        On récupère son extension
        On renvoie TRUE
    Fin si
Fin si
On renvoie FALSE
```

Ca a l'air tellement plus facile vu comme ça ;) . Mais ne vous inquiétez pas, ce n'est pas si compliqué que ça y parait.

La seule difficulté réside dans l'utilisation de différentes fonctions pas forcément connues :

- [getimagesize](#) : Renvoie des informations sur un fichier image. Si le fichier n'est pas une image, renvoie FALSE.
- [image_type_to_mime_type](#) : Récupère le type MIME d'une image.
- [explode](#) : découpe une chaîne en un tableau, suivant de séparateur défini (ici, le '/').

Maintenant que nous savons si le fichier nous intéresse ou non, penchons-nous sur la génération de la miniature.

III - Génération de la galerie

III-A - Création des miniatures

Avant toute chose, nous allons tester si la miniature existe ou non. Si elle existe déjà, nous ne prendrons même pas la peine d'effectuer la moindre vérification, vous êtes censés être organisé.

Test d'existence de la miniature

```
if (!file_exists($chemin_miniaature)) {
    // Génération de la miniature
}
```

Pas besoin d'explications pour ce test je pense.

Passons maintenant à la génération proprement dite des miniatures. Attention, ça se corse un peu :

Génération de la miniature

```
// Génère la miniature de l'image dans le sous-répertoire 'miniature' si elle n'existe pas déjà
function genere_miniaature($dir, $chemin_image, $chemin_miniaature) {
    // Calcul du ratio entre la grande image et la miniature
    $taille_max = 100;
    if ($GLOBALS['largeur'] <= $GLOBALS['hauteur']) {
        $ratio = $GLOBALS['hauteur'] / $taille_max;
    } else {
        $ratio = $GLOBALS['largeur'] / $taille_max;
    }

    // Définition des dimensions de la miniature
    $larg_miniaature = $GLOBALS['largeur'] / $ratio;
    $haut_miniaature = $GLOBALS['hauteur'] / $ratio;

    // Crée la ressource image pour la miniature
    $destination = imagecreatetruecolor($larg_miniaature, $haut_miniaature);

    // Retourne un identifiant d'image jpeg, gif ou png
    $source = call_user_func('imagecreatefrom' . $GLOBALS['extension'], $chemin_image);

    // Redimensionne la grande image
    imagecopyresampled($destination,
        $source,
        0, 0, 0, 0,
        $larg_miniaature,
        $haut_miniaature,
        $GLOBALS['largeur'],
        $GLOBALS['hauteur']);

    // Si le répertoire de miniature n'existe pas, on le crée
    if (!is_dir($dir . '/miniature')) {
        mkdir($dir . '/miniature', 0700);
    }

    // Ecriture physique de l'image
    call_user_func('image' . $GLOBALS['extension'], $destination, $chemin_miniaature);

    // Détruit les ressources temporaires créées
    imagedestroy($destination);
    imagedestroy($source);
}
```

Commençons par jeter un oeil sur deux lignes fort étranges :

```
$source = call_user_func('imagecreatefrom' . $GLOBALS['extension'], $chemin_image);
```

Qu'est-ce donc ?

Et bien cela correspond exactement au code suivant :

```
switch ($GLOBALS['extension']) {
    case 'jpeg':
        $source = imagecreatefromjpeg($chemin_image);
        break;
    case 'gif':
        $source = imagecreatefromgif($chemin_image);
        break;
    case 'png':
        $source = imagecreatefrompng($chemin_image);
        break;
}
```

Plus élégant tout de même, non ?

Il en va de même pour la ligne suivante :

```
call_user_func('image' . $GLOBALS['extension'], $destination, $chemin_miniaature);
```

Qui correspond à :

```
switch ($GLOBALS['extension']) {
    case 'jpeg':
        $source = imagejpeg($destination, $chemin_miniaature);
        break;
    case 'gif':
        $source = imagegif($destination, $chemin_miniaature);
        break;
    case 'png':
        $source = imagepng($destination, $chemin_miniaature);
        break;
}
```

Maintenant, une petite explication des différentes fonctions de manipulation d'image utilisées :

- [imagecreatetruecolor](#) : Crée la ressource image pour la miniature
- [imagecreatefromjpeg](#), [imagecreatefromgif](#), [imagecreatefrompng](#) : // Retourne un identifiant d'image jpeg, gif ou png
- [imagecopyresampled](#) : Redimensionne la grande image
- [imagejpeg](#), [imagegif](#), [imagepng](#) : Ecriture physique de l'image
- [imagedestroy](#) : Permet de détruire les ressources image créés.

Pour plus d'informations sur l'utilisation de GD et de ses différentes fonctions, reportez-vous à l'excellent tutorial de [Michaël](#) : [La manipulation d'images avec PHP : librairie GD](#).

Ca y est, le plus difficile est passé. Il ne reste maintenant plus que quelques broutilles à gérer. Créons donc maintenant le tableau de lien, avec la miniature fraîchement créée.

III-B - Création du tableau de liens

Courage, la fin est proche ...

Ajout du lien au tableau global

```
// Crée le lien dans le tableau global
function ajoute_lien($chemin_image, $chemin_miniaature, $file) {
    // Récupère la taille de la miniature sous forme HTML (width="xxx" height="yyy")
    $taille_html_miniaature = getimagesize($chemin_miniaature);
    $taille_html_miniaature = $taille_html_miniaature[3];
}
```

Ajout du lien au tableau global

```
// Rajoute le lien vers l'image au tableau global $GLOBALS['tabl_liens']
$lien = '<a href="'. $chemin_image. '>';
$lien .= '';
$lien .= '</a>'. "\n";

array_push($GLOBALS['tabl_liens'], $lien);
}
```

La simplicité même. Nous nous contentons de récupérer les attributs de taille de la miniature, puis de créer un lien de la miniature vers l'image originale, et d'insérer le tout dans un tableau global.

Passons à la dernière étape de notre voyage, l'affichage de notre galerie.

III-C - Affichage des images

Je vous avais prévenus, vous n'allez pas être déçus... Voici ma fonction simpliste d'affichage de la galerie.

Il vous est cependant possible de l'adapter à vos besoins, sans avoir à retoucher au reste des fonctions...

Affichage de la galerie

```
define ("NBRE_COLONNES", 4);

// Gère l'affichage du tableau $GLOBALS['tabl_liens']
function affichage() {
    $compteur = 1;
    foreach ($GLOBALS['tabl_liens'] as $val_lien) {
        if ($compteur % NBRE_COLONNES == 1) {
            echo '<br>';
        }
        echo $val_lien;
        $compteur++;
    }
}
```

J'ai simplement défini la constante NBRE_COLONNES pour mon affichage. Je me contente toute les quatre images d'afficher un retour à la ligne.

IV - Conclusion

Voici donc votre fichier tel qu'il devrait de présenter au final :

Fichier final

```
<?php
error_reporting(E_ALL | E_STRICT);

define ("NBRE_COLONNES", 4);

$types_ok = array ('image/jpeg', 'image/gif', 'image/png');
$tabl_exclus = array ('.', '..', 'miniature');
$tabl_liens = array();

// Parcours le répertoire courant et tout ses sous-répertoires récursivement.
function liste_repertoire($dir) {
    if ($handle = opendir($dir)) {
        while (($file = readdir($handle)) !== false) {
            $chemin_fichier = $dir.'/'.$file;
            if (is_dir($chemin_fichier)) {
                if (!in_array($file, $GLOBALS['tabl_exclus'])) {
                    liste_repertoire($dir.'/'.$file);
                }
            } else {
                if (est_image($chemin_fichier)) {
                    $chemin_miniature = $dir.'/miniature/'.$file;
                    if (!file_exists($chemin_miniature)) {
                        genere_miniature($dir, $chemin_fichier,
$chemin_miniature);
                    }
                    ajoute_lien($chemin_fichier, $chemin_miniature, $file);
                }
            }
        }
        closedir($handle);
    }
}

// Teste si le fichier passé en paramètre correspond à l'un des trois type d'image défini
function est_image($chemin_fichier) {
    if (list($GLOBALS['largeur'], $GLOBALS['hauteur'], $type) = getimagesize($chemin_fichier)) {
        $type = image_type_to_mime_type($type);
        if (in_array($type, $GLOBALS['types_ok'])) {
            $ext = explode("/", $type);
            $GLOBALS['extension'] = $ext[1];
            return true;
        }
    }
    return false;
}

// Génère la miniature de l'image dans le sous-répertoire 'miniature' si elle n'existe pas déjà
function genere_miniature($dir, $chemin_image, $chemin_miniature) {
    // Calcul du ratio entre la grande image et la miniature
    $taille_max = 100;
    if ($GLOBALS['largeur'] <= $GLOBALS['hauteur']) {
        $ratio = $GLOBALS['hauteur'] / $taille_max;
    } else {
        $ratio = $GLOBALS['largeur'] / $taille_max;
    }

    // Définition des dimensions de la miniature
    $larg_miniature = $GLOBALS['largeur'] / $ratio;
    $haut_miniature = $GLOBALS['hauteur'] / $ratio;

    // Crée la ressource image pour la miniature
    $destination = imagecreatetruecolor($larg_miniature, $haut_miniature);

    // Retourne un identifiant d'image jpeg, gif ou png
    $source = call_user_func('imagecreatefrom'.$GLOBALS['extension'], $chemin_image);

    // Redimensionne la grande image
    imagecopyresampled( $destination,
        $source,
        0, 0, 0, 0,
```

Fichier final

```

        $larg_miniaiture,
        $haut_miniaiture,
        $GLOBALS['largeur'],
        $GLOBALS['hauteur']);

    // Si le répertoire de miniature n'existe pas, on le crée
    if (!is_dir($dir.'/miniature')) {
        mkdir ($dir.'/miniature', 0700);
    }

    // Ecriture physique de l'image
    call_user_func('image'.$GLOBALS['extension'], $destination, $chemin_miniaiture);

    // Détruit les ressources temporaires créés
    imagedestroy($destination);
    imagedestroy($source);
}

// Crée le lien dans le tableau global
function ajoute_lien($chemin_image, $chemin_miniaiture, $file) {
    // Récupère la taille de la miniature sous forme HTML (width="xxx" height="yyy")
    $taille_html_miniaiture = getimagesize($chemin_miniaiture);
    $taille_html_miniaiture = $taille_html_miniaiture[3];

    // Rajoute le lien vers l'image au tableau global $GLOBALS['tabl_liens']
    $lien = '<a href="'. $chemin_image. ">";
    $lien .= '";
    $lien .= '</a>'. "\n";

    array_push($GLOBALS['tabl_liens'], $lien);
}

// Gère l'affichage du tableau $GLOBALS['tabl_liens']
function affichage() {
    $compteur = 1;
    foreach ($GLOBALS['tabl_liens'] as $val_lien) {
        if ($compteur % NBRE_COLONNES == 1) {
            echo '<br>';
        }
        echo $val_lien;
        $compteur++;
    }
}
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Affichage images</title>
<style type="text/css">
<!--
a img {
    border-color:transparent;
}
-->
</style>
</head>
<body>
<?php
    liste_repertoire('.');
    affichage();
?>
</body>
</html>

```

Lors des premières générations de la galerie, si vous avez beaucoup d'images dans votre répertoire (et ses sous-répertoires), il est tout à fait possible que le script tombe en timeout, à cause de la durée de génération des miniatures.

Pas d'inquiétudes ! Il vous suffit de relancer le script, et il reprendra là où il s'était arrêté.

Vous pouvez également aller modifier les paramètres de votre serveur web afin d'autoriser une exécution plus

longue (max_execution_time dans php.ini).

[Télécharger le code-source](#)

Comme vous avez pu le voir tout du long de cet article, rien de très compliqué n'a été effectué.

Il est aisé avec la connaissance des bonnes fonctions, ainsi qu'un temps de réflexion préalable, de créer de telles applications.

Si vous souhaitez aller plus avant dans la conception d'une galerie dynamique, je vous invite à aller consulter l'article suivant : [Galerie dynamique et système de templates Smarty](#)