Manipulation d'image

Avant de commencer, vérifier que la librairie GD (ou GD2) est activée sur PHP.

Upload de fichier

Rapide rappel sur l'upload de fichier :

```
<form enctype="multipart/form-data" action="upload.php" method="post">
   Nom : <input type="text" name="nom"><br />
   Photo : <input type="file" name="photo"><br />
   <input type="submit" value="Envoyer">
</form>
```

```
if(isset($_FILES['photo'])){
   if($_FILES['photo']['error'] == UPLOAD_ERR_OK){
      $dossier = 'upload/photo/';
      $fichier = $_FILES['photo']['name'];//ou on peut mettre le nom de fichier que l'on veut pour être certain d'éviter les doublons
      if(move_uploaded_file($_FILES['photo']['tmp_name'], $dossier.$fichier)){
         //la fonction renvoie true, le fichier a bien été enregistré
      }else{
         echo 'echec de l\'upload.';
      }
   }
}
```

Nous allons voir comment manipuler une image qui a été uploadée avec succès.

Récupération de l'image source

Les fonctions à utiliser ne sont pas les mêmes en fonction du type d'image (jpeg, PNG et GIF).

Pour récupérer la source d'un image, on utilise la fonction imagecreatefromgif (ou PNG/jpeg) :

```
$file = '/mon/chemin/vers/limage/image.jpg';
//on récupère l'extension de l'image:
$ext = explode('.', $file);
$ext = strtolower($ext[count($ext)-1]);
switch ($ext) {
   case 'GIF':
      $source_gd_image = imagecreatefromgif($file);
      break;
   case 'jpeg':
   case 'jpg':
      $source_gd_image = imagecreatefromjpeg($file);
      break;
   case 'PNG':
      $source_gd_image = imagecreatefrompng($file);
      break;
}
if($source_gd_image === false){
```

```
    echo 'erreur lors de la récupération de la source de l\'image';
    die();
}
```

Si la fonction renvoie false, vérifier que le fichier existe, que vous appeler bien la bonne fonction pour le bon format.

Création d'une miniature

Lors d'un upload d'image d'un visiteur, nous ne pouvons pas être certains que l'image ait la taille que l'on souhaite, il faut donc redimensionner l'image pour correspondre à nos standards d'affichage en ligne (miniature, moyenne et large).

On va, dans l'ordre :

1. récupérer la taille de l'image d'origine et calculer la taille de la miniature
2. créer une image « vide » pour accueillir notre miniature
3. créer une copie et redimensionner l'image d'un coup
4. enregistrer l'image
5. libérer la mémoire

1 La taille

La fonction getmagesize nous retourne pas mal de chose :

```
//on récupère la taille de notre image
$imgsize = getimagesize($file);
if($imgsize === false){
    echo 'erreur lors de la récupération de la source de l\'image';
    die();
}
var_dump($imgsize);
```

```
array(7) {
  [0]=>
  int(1024)
  [1]=>
  int(768)
  [2]=>
  int(2)
  [3]=>
  string(25) "width="1024" height="768""
  ["bits"]=>
  int(8)
  ["channels"]=>
  int(3)
  ["mime"]=>
  string(10) "image/jpeg"
}
```

Détail du tableau :

0/ Largeur de l'image

1/ Hauteur de l'image

2/ Constante du type de l'image (IMG_GIF, IMG_JPG, etc) sous la forme d'un INT

3/ Une chaîne de caractères à afficher directement dans le code html de l'image à l'affichage : **`<img src="monfichier.jpg" width="1024" height="768">`**

bits/ Le nombre d'octect pour chaque couleur

channels/ sera 3 pour des images RGB et 4 pour des images CMYK

mime/ le type mime de l'image, à utiliser dans un header() par exemple : `header("Content-type: image/jpeg");`

2 Création de l'image vide

On crée notre image vide en indiquant la taille de notre miniature :

```
$thumbnail = imagecreatetruecolor($thumbnailWitdh, $thumbnailHeight);
```

3 Création et redimensionnent

```
imagecopyresampled($thumbnail, $source_gd_image, 0, 0, 0, 0, $thumbnailWitdh,
$thumbnailHeight, $imgsize[0], $imgsize[1]);
```

Détail des paramètres :

`$thumbnail` => La source de notre miniature (notre image noire)

`$source_gd_image` => La source de l'image d'origine

`0, 0, 0, 0` => Le décalage souhaité par rapport à l'image d'origine, pour un redimensionnement on laisse à 0

`$thumbnailWitdh` => La largeur de la miniature

`$thumbnailHeight` => La hauteur de la miniature

`$imgsize[0]` => La largeur de l'image source

`$imgsize[1]` => La hauteur de l'image source

4 Enregistrement

Toutes les manipulations que l'on a faites avant sont sur des ressources, pas sûres des fichiers sur le serveur. Une fois toutes nos manipulations terminées, on enregistre l'image pour pouvoir l'afficher :

```
imagejpeg($thumbnail,  $dossier.'thumb_'.$fichier, 90);
```

Ici on peut enregistrer l'image sous n'importe quel format, car nous manipulons des ressources (pour PNG et GIF utilisez les fonctions imagepng (qui a un 4e paramètre filtre) et imagegif (qui n'a pas de 3e paramètre qualité)).

On donne dans l'ordre : la ressource de notre image, la destination et la qualité de l'image.

5 Libérer la mémoire

Toutes ces ressources sont stockées en mémoire et peuvent prendre de la place, il faut donc libérer la mémoire !

```
imagedestroy($source_gd_image);
imagedestroy($thumbnail);
```

Script complet

```php
//on récupère la taille de notre image
$imgsize = getimagesize($file);
if($imgsize === false){
   echo 'erreur lors de la récupération de la source de l\'image';
   die();
}
//création de la miniature, en concervant le ratio.
//on fixe une largeur (width)
$thumbnailWitdh = 150;
//on calcul la hauteur
$thumbnailHeight = floor($thumbnailWitdh*$imgsize[1]/$imgsize[0]);
$thumbnailWitdh;
$thumbnailHeight;
//on créé une image "vide" (une image noire)
$thumbnail = imagecreatetruecolor($thumbnailWitdh, $thumbnailHeight);
//on créé une copie de notre image source
imagecopyresampled($thumbnail, $source_gd_image, 0, 0, 0, 0, $thumbnailWitdh,
$thumbnailHeight, $imgsize[0], $imgsize[1]);
//et on en fait un fichier jpeg avec une qualité de 90%
imagejpeg($thumbnail,  $dossier.'thumb_'.$fichier, 90);
//on oublie pas de libérer la mémoire, car nos images sources sont stocké et
prennent de la place!
imagedestroy($source_gd_image);
imagedestroy($thumbnail);
echo '<img src="'. $dossier.'thumb_'.$fichier.'">';
echo '<img src="'. $dossier.$fichier.'">';
```