Nous allons donc voir dans ce tutorial comment faire des **images dynamiques** (c'est-à-dire qui changent suivant certains paramètres) grâce à PHP, qui les générera.

La **librairie GD** peut être schématisée par "un fichier" qui comporte de nombreuses fonctions permettant de travailler les images. Grâce à cette librairie, vous allez pouvoir modifier vos images, récupérer des informations sur ces images, voir même, créer vous-même vos propres images.

Avant de nous lancer dans la création de superbes images, une petite vérification s'impose.
En effet, si la libraire GD n'est pas installée sur votre serveur (votre hébergeur) cela ne sert à rien de poursuivre ce tutorial, vu que les fonctions que nous allons utiliser ne seront pas activées.

Pour savoir si la librairie GD est installée sur votre serveur, vous devez faire un **phpinfo**. Pour cela, créer un fichier, par exemple **phpinfo.php**, dans lequel vous allez placer le code suivant :

exemple1

```php
<?php
phpinfo();
?>
```

Ensuite, dans votre navigateur favori, affichez la page ainsi créée.

Faites défiler cette page jusqu'au moment où vous voyez quelque chose qui ressemble à ceci :

Là, si comme ici, le **GD Support** est **enabled**, c'est tout bon, vous pouvez continuer la lecture de ce tutorial.
Enfin sachez également, que depuis la version **2.0** de la librairie, le format **GIF** n'est plus supporté, et il a été avantageusement remplacé par le format **PNG**.

Pour ceux qui n'ont pas la librairie installée, le seul remède est de demander gentiment à leur hébergeur de l'installer :)

Passons de suite à la création de notre première image dynamique.

Prenons par exemple le code suivant, très basique, permettant l'affichage d'une image.
On a par exemple une page que l'on nommera **index.php**, et qui contiendra :

exemple2

```php
<html>
<head>
<title>Notre page de test</title>
</head>

<body>
<img src="./mon_image.php">
</body>

</html>
```

Et vous notez que bien le lien vers notre image désigne un fichier PHP. C'est tout à fait normal, vu que c'est PHP qui va générer notre image.

Voyons à présent le code de la page **mon_image.php**.
Imaginons que l'on désire dessiner un rectangle rouge (en fait nous n'allons pas dessiner un rectangle, nous allons en fait créer une image vide remplie avec de la couleur rouge).
On aura alors (pour la page **mon_image.php**), le code suivant :

exemple3

```php
<?php
// on spécifie le type de document que l'on va créer (ici une image au format PNG
```

```php
header ("Content-type: image/png");

// on dessine une image vide de 200 pixels sur 100
$image = @ImageCreate (200, 100) or die ("Erreur lors de la création de l'image");

// on applique à cette image une couleur de fond, les couleurs étant au format RVB, on aura donc ici une couleur rouge
$couleur_fond = ImageColorAllocate ($image, 255, 0, 0);

// on dessine notre image PNG
ImagePng ($image);
?>
```

Et voila notre première image générée par PHP.

Cependant, quelques explications sont nécessaires et extrêmement importantes :

- lorsque l'on crée une page WEB avec PHP, on n'est pas obligé de dire à PHP que l'on crée une page WEB, en revanche, lorsque l'on crée une image, **on est obligé de le spécifier** avec l'instruction :

exemple4

```php
<?php
header ("Content-type: image/png");
?>
```

- naturellement, si l'on désire créer une image de type **JPG**, on prendra soin d'écrire :

exemple5

```php
<?php
header ("Content-type: image/jpeg");
?>
```

- l'appel à la fonction **ImageCreate** nous retourne en fait une ressource (**$image**) correspondant à l'image que nous sommes en train de créer.

- nous faisons ensuite appel à la fonction **ImageColorAllocate** qui a deux fonctions principales : premièrement, elle crée une couleur stockée dans une variable pouvant être réutilisée ultérieurement (ici **$couleur_fond**), et deuxièmement, elle enregistre cette couleur dans la palette de l'image **$image**.
Notez aussi, et c'est extrêmement important, que **cette couleur étant la première couleur enregistrée dans la palette, elle correspondra en fait à la couleur de fond de notre image** (en fait, la première couleur enregistrée dans la palette de couleur correspond à la couleur de fond de l'image).

- on utilise alors la fonction **ImagePng** pour afficher notre ressource, soit donc notre image (**$image**).

- notez également que si nous avions créé une image de type **JPG**, nous aurions utilisé ici l'instruction **ImageJpeg**.

Après ce premier exemple tout simple, vous devez sûrement vous dire qu'il n'y a rien de dynamique dans tout cela, et que votre **Photoshop** ferait bien mieux.
A première vue, oui.
Mais vu que notre image est générée par PHP, nous pouvons lui fournir des paramètres. Et l'avantage, il est là.
C'est-à-dire que vous allez pouvoir créer des images différentes en fonction de certains paramètres.

Reprenons alors le code de notre **index.php**, auquel nous allons greffer un petit formulaire permettant de choisir la couleur de notre image.

On aura alors, par exemple (pour la page **index.php**) :

exemple6

```php
<html>
<head>
<title>Notre page de test</title>
```

```
</head>

<body>
Sélectionner l'intensité des différentes teintes :<br />
<form action="./mon_image.php" method="post">
Rouge (un nombre entre 0 et 255) : <input type="text" name="rouge"><br />
Vert (un nombre entre 0 et 255) : <input type="text" name="vert"><br />
Bleu (un nombre entre 0 et 255) : <input type="text" name="bleu"><br />
<input type="submit" value="Voir">
</form>
</body>

</html>
```

Et prenons par exemple le code suivant pour la page **mon_image.php** (nous ne ferons pas ici la vérification des champs du formulaire : en effet, on supposera que l'utilisateur saisi bien à chaque fois un nombre entre 0 et 255) :

exemple7

```php
<?php
// on teste nos 3 variables pour nos couleurs
if (isset($_POST['rouge']) && isset($_POST['vert']) && isset($_POST['bleu'])) {

      // on spécifie le type de document que l'on va créer (ici une image au format PNG
      header ("Content-type: image/png");

      // on dessine une image vide de 200 pixels sur 100
      $image = @ImageCreate (200, 100) or die ("Erreur lors de la création de l'image");

      // on applique à cette image une couleur de fond, les couleurs étant       au
format RVB, on obtiendra ici la couleur que l'utilisateur aura spécifié en paramètre du
formulaire
      $couleur_fond = ImageColorAllocate ($image, $_POST['rouge'], $_POST['vert'],
$_POST['bleu']);

      // on dessine notre image PNG
      ImagePng ($image);
}
else {
      echo 'Les variables du formulaire ne sont pas déclarées.';
}
?>
```

Pas mal, non ?

Maintenant que vous avez vu une première approche de l'utilisation de cette librairie, un petit rappel est nécessaire.
En effet, en règle générale, lorsque l'on dessine en programmation, **l'origine du repère représentant la position des pixels de notre image est située dans le coin supérieur gauche de l'image**.
Cela diffère donc des repères classiques mathématiques où l'origine du repère est toujours situé dans le coin inférieur gauche.

Et ceci est très important.
En effet, pratiquement toutes les fonctions de cette librairie vous demanderont les coordonnées de différents points.
Autant se mettre tout de suite en accord avec le système.

Conseil :

Vu que l'axe des x est orienté de la "gauche" vers la "droite", lorsque vous allez utiliser des fonctions permettant de dessiner des rectangles, des droites, etc, l'idéal est de toujours fournir en arguments les coordonnées du point le plus à gauche de votre figure, puis les coordonnées du point le plus à droite.

Après cette petite précision, voici un autre exemple intéressant de l'utilisation de la librairie GD.

En effet, vous affichez parfois des images sur vos sites WEB, et beaucoup de vos photos partent dans la nature (pillées

la plupart du temps par des visiteurs peu attentionnés).
Ne serait-il pas sympathique de pouvoir protéger vos photos à l'aide d'une petite incrustation permettant de "copyrighter" votre image ?
Et bien c'est tout à fait possible.

Pour ce faire, voyons le code pour notre fichier PHP contenant le code générant cette image "copyrightée" (fichier **mon_image.php** par exemple) :

exemple8

```php
<?php
// on spécifie le type de fichier créer (ici une image de type jpeg)
header ("Content-type: image/jpeg");

// on crée deux variables contenant les chemins d'accès à nos deux fichiers : $fichier_source contenant le lien vers l'image à "copyrighter", $fichier_copyright contenant le lien vers la petite vignette contenant le copyright (bien sur, on prendra soin de placer les images sources dans un répertoire "caché" sinon le copyright ne sert à rien si les visiteurs ont accès aux images sources)
$fichier_source = "./gd.jpg";
$fichier_copyright = "./copyright.jpg";

// on crée nos deux ressources de type image (par le biais de la fonction ImageCreateFromJpeg)
$im_source = ImageCreateFromJpeg ($fichier_source);
$im_copyright = ImageCreateFromJpeg ($fichier_copyright);

// on calcule la largeur de l'image qui va être copyrightée
$larg_destination = imagesx ($im_source);

// on calcule la largeur de l'image correspondant à la vignette de copyright
$larg_copyright = imagesx ($im_copyright);
// on calcule la hauteur de l'image correspondant à la vignette de copyright
$haut_copyright = imagesy ($im_copyright);

// on calcule la position sur l'axe des abscisses de la vignette
$x_destination_copyright = $larg_destination - $larg_copyright;

// on réalise la superposition, le dernier paramètre étant le degré de transparence de la vignette (cependant, allez voir la fin de ce même tutorial pour une définition complète de tous les arguments de cette fonction)
@imageCopyMerge ($im_source, $im_copyright,
      $x_destination_copyright, 0, 0, 0, $larg_copyright,
      $haut_copyright, 70);

// on affiche notre image copyrightée
Imagejpeg ($im_source);
?>
```

Ensuite, afin de voir le résultat, il suffit d'écrire une page toute simple affichant l'image **mon_image.php**, avec une insertion d'image du genre (par exemple) :

exemple9

```
<img src="./mon_image.php">
```

Bien sur, pour rendre ce script vraiment dynamique, il suffit de passer à l'image en paramètre le nom du fichier image à "copyrighter".
En effet, comme dans le premier exemple avec l'image basique, on peut faire un lien ressemblant à :

exemple10

```
<img src="mon_image.php?fichier=toto.jpg">
```

Et ensuite, dans le script **mon_image.php**, on récupère la variable $_GET['fichier'] qui sera en fait le nom du fichier de l'image à "copyrighter".

Je vous fais confiance pour cette partie :)

Voyons maintenant un autre exemple de l'utilisation de cette librairie : la création de **miniatures**.
En effet, vous voulez mettre vos photos en ligne, et jusqu'ici, vous faisiez vos miniatures à la main :
- c'était long
- pas toujours réussi

Et pourtant les miniatures sont extrêmement pratiques pour afficher plus rapidement vos grandes images.
En effet, il est toujours agréable de voir une liste de petites images (les miniatures) munies d'un lien permettant d'afficher ces mêmes images dans leur taille originelle.

Détaillons maintenant le code permettant de créer vos miniatures.
En revanche, ici, **à la différence des exemples précédents, nous n'allons pas afficher à la volée une image générée par PHP**.
En effet, nous allons en fait écrire un script PHP, qui, avec le lien vers une image de type **JPG**, va **créer une miniature de cette même image qui sera sauvegardée sur votre espace disque**.
Le nom de la miniature sera en fait le nom du fichier original, précédé de la mention **mini**_.

On aura alors le code suivant pour ce script de création de miniatures :

exemple11

```php
<?php
// on donne à PHP le lien vers notre image à miniaturiser
$Image = "metaeffect_001.jpg";

// on impose la taille de la largeur ou de la hauteur de la photo (le choix entre la largeur ou la hauteur se fait automatiquement, suivant que la photo est "horizontale" ou "verticale")
$ratio = 150;

// on crée une ressource représentant en fait l'image à miniaturiser
$src=imagecreatefromjpeg($Image);

// on récupère les paramètres de notre image (getimagesize est une fonction qui retourne un tableau contenant les paramètres d'une image : sa largeur, son hauteur, son type, etc...)
$size = getimagesize($Image);

// on test si la largeur de l'image est supérieur à sa longueur
if ($size[0] > $size[1]) {
   // on crée une ressource pour notre miniature
   $im=imagecreate(round(($ratio/$size[1])*$size[0]), $ratio);
   // on place dans la ressource que nous venons de créer une copie de l'image originelle, redimensionnée et réechantillonée
   imagecopyresampled($im, $src, 0, 0, 0, 0, round(($ratio/$size[1])*$size[0]),$ratio, $size[0], $size[1]);
}
else {
   // si la largeur est inférieure ou égale à la hauteur, on entre dans ce cas
   // on crée une ressource pour notre miniature
   $im=imagecreate($ratio, round(($ratio/$size[0])*$size[1]));
   // on place dans la ressource que nous venons de créer une copie de l'image originelle, redimensionnée et réechantillonée
   imagecopyresampled($im, $src, 0, 0, 0, 0, $ratio, round($size[1]*($ratio/$size[0])), $size[0], $size[1]);
}

// on définit le nom de notre miniature
```

```php
$miniature = "mini_$Image";

// on crée notre miniature
ImageJpeg ($im, $miniature);
?>
```

Et surtout, remarquez bien que nous n'avons pas utilisé la fonction **header**.
En effet, ici, **on ne cherche pas à "afficher" une image générée à la volée**.
Ce script inspecte en fait une image, et à partir de celle-ci, il en crée une nouvelle.
Pour voir le résultat de cette opération, et donc voir votre miniature, vous devrez créer une page html contenant la balise suivante :

exemple12

```
<img src="mini_metaeffect_001.jpg ">
```

Enfin, afin d'accroître vos talents de Picasso des temps modernes, voyons un dernier exemple : la **création d'un histogramme** pour un script de statistiques par exemple.
En effet, bon nombre de scripts de statistiques affichent la popularité de vos pages sur de jolis dessins.
Et bien sachez que c'est possible de faire quelque chose d'équivalent à l'aide de la librairie GD.
Pour ce faire, je vous propose la création d'un histogramme permettant d'afficher des "bâtons" représentants le nombre de pages vues de votre site sur une année (un bâton représentant un mois de l'année).

Pour ce faire, voici par exemple le code du script **stats_year.php** (**il s'agira d'une image dynamique, crée à la volée, l'utilisation d'un header sera donc nécessaire**) :

exemple13

```php
<?php
// on définit un tableau contenant le nombre de page vues par mois : par exemple, on suppose que 1500 pages du site ont été vues en janvier, 2450 en février, etc... Bien sur, pour que ce script soit vraiment valides, vous n'allez pas déclarer ce tableau, car sinon, les bâtons seront toujours les mêmes :) Vous allez plutôt effectuer une requête SQL sur votre base de données permettant de récupérer le nombre de pages vues de votre site par mois. Ensuite, il suffira d'appeler le script avec ces 12 paramètres dans votre page html (en faisant par exemple : <img src="./images/stats_year.php?visite_par_mois[1]=1500.....">)
$visite_par_mois[1]=1500;
$visite_par_mois[2]=2450;
$visite_par_mois[3]=800;
$visite_par_mois[4]=1780;
$visite_par_mois[5]=1900;
$visite_par_mois[6]=2450;
$visite_par_mois[7]=1684;
$visite_par_mois[8]=1845;
$visite_par_mois[9]=3450;
$visite_par_mois[10]=980;
$visite_par_mois[11]=1234;
$visite_par_mois[12]=500;

// on calcule le nombre de pages vues sur l'année
$max_visite = max($visite_par_mois);

// on spécifie le type d'image que l'on va créer, ici ce sera une image au format PNG
header ("Content-type: image/png");

// on définit la largeur et la hauteur de notre image
$largeur = 550;
$hauteur = 300;

// on crée une ressource pour notre image qui aura comme largeur $largeur et $hauteur comme hauteur (on place également un or die si la création se passait mal afin d'avoir un petit message d'alerte)
```

```
$im = @ImageCreate ($largeur, $hauteur) or die ("Erreur lors de la création de l'image")
;

// on place tout d'abord la couleur blanche dans notre table des couleurs (je vous
rappelle donc que le blanc sera notre couleur de fond pour cette image).
$blanc = ImageColorAllocate ($im, 255, 255, 255);

// on place aussi le noir dans notre palette, ainsi qu'un bleu foncé et un bleu clair
$noir = ImageColorAllocate ($im, 0, 0, 0);
$bleu_fonce = ImageColorAllocate ($im, 75, 130, 195);
$bleu_clair = ImageColorAllocate ($im, 95, 160, 240);


// on dessine un trait horizontal pour représenter l'axe du temps
ImageLine ($im, 20, $hauteur-40, $largeur-15, $hauteur-40, $noir);

// on affiche le numéro des 12 mois
for ($i=1; $i<=12; $i++) {
   if ($i==1) {
      ImageString ($im, 2, 42, $hauteur-38, $i, $noir);
   }
   else {
      ImageString ($im, 2, ($i)*42, $hauteur-38, $i, $noir);
   }
}

// on dessine un trait vertical pour représenter le nombre de pages vues
ImageLine ($im, 20, 30, 20, $hauteur-40, $noir);

// on affiche les legendes sur les deux axes ainsi que différents textes (note : pour
que le script trouve la police verdana, vous devrez placer la police verdana dans un
repertoire /fonts/)
imagettftext($im, 14, 0, $largeur-70, $hauteur-10, $noir, "./fonts/verdana.ttf", "Mois")
;
imagettftext($im, 14, 0, 10, 20, $noir, "./fonts/verdana.ttf", "Nb. de pages vues");
imagettftext($im, 14, 0, $largeur-250, 20, $noir, "./fonts/verdana.ttf", "Statistiques
pour l'année 2003");

// on parcourt les douze mois de l'année
for ($mois=1; $mois <= 12; $mois++) {
   if ($visite_par_mois[$mois]!="0") {
      // on calcule la hauteur du baton
      $hauteurImageRectangle = ceil((($visite_par_mois[$mois])*($hauteur-50))/
$max_visite));
      if ($mois=="1") {
         // si le mois est janvier, on affiche notre premier baton
         // on affiche le premier baton noir
         ImageFilledRectangle ($im, 42, $hauteur-$hauteurImageRectangle, 42+14, $hauteur
-41, $noir);
         // on affiche le second baton, bleu foncé, qui sera un peu plus petit que le
noir afin de recouvrir une partie du noir
         ImageFilledRectangle ($im, 44, $hauteur-$hauteurImageRectangle+2, 42+12,
$hauteur-41-1, $bleu_fonce);
         // on affiche le dernier baton, bleu clair, qui sera un peu plus petit que le
bleu foncé afin de recouvrir une partie du bleu foncé (on obtiendra ainsi un effet de
dégradé)
         ImageFilledRectangle ($im, 48, $hauteur-$hauteurImageRectangle+2, 42+8,
$hauteur-41-1, $bleu_clair);
      }
      else {
         // si le mois est different de janvier, on affiche les autres batons
         ImageFilledRectangle ($im, ($mois)*42, $hauteur-$hauteurImageRectangle, ($mois)
*42+14, $hauteur-41, $noir);
         ImageFilledRectangle ($im, ($mois)*42+2, $hauteur-$hauteurImageRectangle+2, (
$mois)*42+12, $hauteur-41-1, $bleu_fonce);
         ImageFilledRectangle ($im, ($mois)*42+6, $hauteur-$hauteurImageRectangle+2, (
```

```
$mois)*42+8, $hauteur-41-1, $bleu_clair);
      }
   }
}

// on dessine le tout
Imagepng ($im);
?>
```

Vous noterez quelque chose de très important : en effet, pour faire les dégradés sur les bâtons, nous avons en fait dessiné plusieurs rectangles les uns sur les autres (et donc des rectangles de plus en plus petits).
En effet, on a utilisé GD comme si on utilisait un logiciel de dessin.
En clair, quand **on dessine par-dessus quelque chose, et bien ce quelque chose disparaît au profit de la nouvelle forme que l'on vient de dessiner** (tout comme des calques sous **Photoshop**).

Aller hop, à vos pinceaux maintenant, et surtout n'oubliez pas d'aller jeter un coup d'oeil sur le manuel PHP afin de découvrir toutes les descriptions des instructions utilisées dans ce cours.

Auteur : LA GLOBULE

Dernière révision du cours : le 30/05/2007 à 00:18