

Welcome to OpenClassrooms! By continuing on the site, you are agreeing to our use of cookies. Read more

OK

Sign up

Sign in

[Home](#) ► [Course](#) ► [Concevez votre site web avec PHP et MySQL](#) ► [Cr  er des images en PHP](#)

Concevez votre site web avec PHP et MySQL



30 hours



Medium

License



Cr  er des images en PHP

[Log in](#) or [subscribe](#) to enjoy all this course has to offer!

Vous savez quoi ? Il y a des gens qui croient que le PHP n'est fait que pour g  n  rer des pages web !

Si, si, je vous jure !

Quoi, vous aussi ?

Bon... remarquez, je ne peux pas vous en vouloir non plus : tout au long de ce cours, on n'a fait « que » g  n  rer des pages HTML avec PHP. Difficile de croire que l'on pourrait faire autre chose...

En fait,    la base, PHP a bien   t   cr     pour r  aliser des pages web. Mais au fur et    mesure, on s'est rendu compte qu'il serait dommage de le limiter      a. On a donc pr  vu de pouvoir lui rajouter des « extensions ». Ainsi, en rajoutant certains fichiers (des DLL sous Windows), PHP peut se mettre    g  n  rer des images, ou m  me des PDF !

Dans ce chapitre, nous allons parler de l'extension sp  cialis  e dans la g  n  ration d'images : la biblioth  que GD.

Activer la biblioth  que GD



On a d  j   un probl  me (  a commence fort ;-)).

En effet, la biblioth  que GD (qui vous permet de cr  er des images) est livr  e avec PHP, mais **elle n'est pas activ  e**.   a veut dire quoi ? Qu'il va falloir demander    l'activer tout simplement.

La proc  dure    suivre est exactement la m  me que celle qu'on avait vue pour activer PDO lorsque nous avons d  couvert les bases de donn  es.

Sous WAMP par exemple, faites un clic gauche sur l'icône de WAMP dans la barre des tâches, puis allez dans le menu `PHP / Extensions PHP` et cochez `php_gd2`.



Et sur Internet, avec mon hébergeur ? Est-ce que je peux utiliser GD ?

Cela dépend des hébergeurs. Une grande partie des hébergeurs gratuits désactivent GD parce que ça consomme beaucoup de ressources du processeur.

Si des dizaines de sites se mettent à générer des images en même temps, ça risquerait de faire ramer toute la machine et donc de ralentir tous les autres sites.

Ne désespérez pas pour autant, il existe certainement des hébergeurs gratuits qui acceptent la bibliothèque GD... Sinon, il faudra peut-être trouver un hébergement payant (on peut en trouver des pas chers qui ont activé GD !).

Les bases de la création d'image



Voici le plan que nous allons suivre pour créer une image :

1. nous allons découvrir ce qu'est un **header** ;
2. ensuite, nous allons créer l'image de base ;
3. enfin, nous verrons comment on affiche l'image concrètement.

Au boulot !

Le header

Il y a deux façons de générer une image en PHP.

- Soit on fait en sorte que notre script PHP renvoie une image (au lieu d'une page web, comme on en avait l'habitude). Dans ce cas, si on va sur la page `http://www.monsite.com/testgd.php`, une image sera affichée et non pas une page web !
- Soit on demande à PHP d'enregistrer l'image dans un fichier.

Dans les deux cas, on utilisera exactement les mêmes fonctions.

On va commencer par la première façon de générer l'image, c'est-à-dire qu'on va faire en sorte que notre script « renvoie » une image au lieu d'une page web.



Mais comment faire pour que le navigateur sache que c'est une image et non pas une page HTML qu'il doit afficher ?

Il va falloir envoyer ce qu'on appelle un **header** (un en-tête). Grâce à la fonction `header`, on va « dire » au navigateur que l'on est en train d'envoyer une image.

Je vous rappelle les types d'images les plus courants sur le web :

- - **JPEG** : c'est un format très adapté pour les photos par exemple, car on peut utiliser beaucoup de couleurs ;

- **PNG** : c'est le format le plus récent, très adapté dans la plupart des cas. En fait, à moins d'avoir affaire à une photo, le mieux est d'utiliser le PNG.

Le PNG est en quelque sorte le « remplaçant » du format GIF.

Donc pour faire simple : si c'est une photo, vous faites un JPEG, sinon dans tous les autres cas vous faites un PNG.

Voici le code PHP qu'il faut mettre pour « annoncer » au navigateur que l'on va renvoyer une image PNG :

php

```
1 <?php
2 header ("Content-type: image/png");
3 ?>
```

Voilà, c'est assez simple. Ce code signifiera pour le navigateur que l'on envoie une image PNG, et non pas une page HTML.

Si vous envoyez un JPEG, c'est presque pareil, mais vous remplacez le « png » par « jpeg ».



La fonction `header` est particulière. Comme `setcookie`, elle doit être utilisée avant d'avoir écrit le moindre code HTML.

En clair, mettez cette ligne au tout début de votre code, et vous n'aurez pas de problèmes.

Créer l'image de base

Il faut savoir qu'il y a deux façons de créer une image : soit vous créez une nouvelle image vide, soit vous chargez une image qui existe déjà et qui servira de fond à votre nouvelle image.

À partir d'une image vide

On va commencer par créer une image vide.

Pour créer une image vide en PHP, on utilise la fonction `imagecreate`.

Cette fonction est simple. Elle prend deux paramètres : la largeur et la hauteur de l'image que vous voulez créer. Elle renvoie une information que vous devez mettre dans une variable (par exemple `$image`). Ce qui nous donne :

php

```
1 <?php
2 header ("Content-type: image/png");
3 $image = imagecreate(200,50);
4 ?>
```

Ici, nous sommes en train de créer une image de **200 pixels de large** et **50 pixels de haut**.

`$image` ne contient ni un nombre, ni du texte. Cette variable contient une « image ». C'est assez difficile à imaginer qu'une variable puisse « contenir » une image, mais c'est comme ça.



On dit que `$image` est une « ressource ». Une ressource est une variable un peu spéciale qui contient toutes les informations sur un objet. Ici, il s'agit d'une image, mais il pourrait très bien s'agir d'un PDF ou même d'un fichier que vous avez ouvert avec `fopen`. Tiens, tiens, ça vous rappelle quelque chose ?

À partir d'une image existante

Maintenant, l'autre possibilité : créer une image à partir d'une image déjà existante.

Cette fois, il y a deux fonctions à connaître. Laquelle choisir ? Ça dépend du format de l'image que vous voulez charger :

- **JPEG** : il faut utiliser la fonction `imagecreatefromjpeg` ;
- **PNG** : il faut utiliser la fonction `imagecreatefrompng` .

Par exemple, j'ai une jolie photo de coucher de soleil qui s'appelle `couchersoleil.jpg` (figure suivante).



Ma photo couchersoleil.jpg

Pour créer une nouvelle image en se basant sur celle-là, je dois utiliser la fonction `imagecreatefromjpeg` . Ça nous donnerait le code suivant :

```
1 <?php
2 header ("Content-type: image/jpeg");
3 $image = imagecreatefromjpeg("couchersoleil.jpg");
4 ?>
```

php

Voilà, vous savez créer une nouvelle image.

Nous allons maintenant voir comment afficher l'image que vous venez de créer.

Quand on a terminé : on affiche l'image

Une fois que vous avez chargé l'image, vous pouvez vous amuser à y écrire du texte, à dessiner des cercles, des carrés, etc. Nous allons apprendre tout cela juste après.

Je souhaite vous montrer ici comment faire pour dire à PHP qu'on a fini et qu'on veut afficher l'image.

La fonction à utiliser dépend du type de l'image que vous êtes en train de créer :

- **JPEG** : il faut utiliser la fonction `imagejpeg` ;
- **PNG** : il faut utiliser la fonction `imagepng` .

Ces deux fonctions s'utilisent de la même manière : vous avez juste besoin d'indiquer quelle image vous voulez afficher.

Comme je vous le disais, il y a deux façons d'utiliser les images en PHP : vous pouvez les afficher directement après les avoir créées, ou les enregistrer sur le disque pour pouvoir les ré-afficher plus tard, sans avoir à refaire tous les calculs.

Afficher directement l'image

C'est la méthode que l'on va utiliser dans ce chapitre. Quand la page PHP est exécutée, elle vous affiche l'image que vous lui avez demandé de créer.

Vous avez toujours votre variable `$image` sous la main ? Parfait. ;-)

Alors voici le code complet que j'utilise pour créer une nouvelle image PNG de taille et l'afficher directement :

php

```
1 <?php
2 header ("Content-type: image/png"); // 1 : on indique qu'on va envoyer une image PNG
3 $image = imagecreate(200,50); // 2 : on crée une nouvelle image de taille 200 x 50
4 // 3 : on s'amuse avec notre image (on va apprendre à le faire)
5 imagepng($image); // 4 : on a fini de faire joujou, on demande à afficher l'image
6 ?>
```



C'est bien joli, mais là on n'a qu'une image sous les yeux. Et si je veux mettre du texte autour ? Les menus de mon site ?

En fait, on utilise une technique qui, j'en suis sûr, va vous surprendre. On va demander à **afficher la page PHP comme une image**.

Donc, si la page PHP s'appelle « image.php », vous mettrez ce code HTML pour l'afficher depuis une autre page :

php

```
1 
```

Incredible, isn't it?

Mais en fait, c'est logique quand on y pense ! La page PHP que l'on vient de créer EST une image (puisque l'on a modifié le `header`). On peut donc afficher l'image que l'on vient de créer depuis n'importe quelle page de votre site en utilisant simplement la balise `` .

Le gros avantage de cette technique, c'est que l'image affichée pourra changer à chaque fois !

Enregistrer l'image sur le disque

Si, au lieu d'afficher directement l'image, vous préférez l'enregistrer sur le disque, alors il faut ajouter un paramètre à la fonction `imagepng` : le nom de l'image et éventuellement son dossier. Par contre, dans ce cas, votre script PHP ne va plus renvoyer une image (il va juste en enregistrer une sur le disque). Vous pouvez donc supprimer la fonction `header` qui ne sert plus à rien.

Ce qui nous donne :

php

```
1 <?php
2 $image = imagecreate(200,50);
3 // on fait joujou avec notre image
4 imagepng($image, "images/monimage.png"); // on enregistre l'image dans le dossier "images"
5 ?>
```

Cette fois, l'image a été enregistrée sur le disque avec le nom `monimage.png`. Pour l'afficher depuis une autre page web, vous ferez donc comme ceci :

php

```
1 
```

Ça, vous avez un peu plus l'habitude, j'imagine.

Cette technique a l'avantage de ne pas nécessiter de recalculer l'image à chaque fois (votre serveur aura moins de travail), mais le défaut c'est qu'une fois qu'elle est enregistrée, l'image ne change plus.



Mais... mais ? Si je teste ces codes, ça crée une image toute blanche ! C'est nul, il ne s'est rien passé de bien !

Oui, je sais. Vous avez été patients et c'est bien, parce que c'est maintenant que ça va devenir intéressant. Allez donc chercher votre baguette magique, je vous attends.

Texte et couleur



C'est bon, vous avez votre baguette magique ?

Alors voici ce que nous allons apprendre à faire maintenant :

- manipuler les couleurs ;
- écrire du texte.

Vous allez commencer à voir un peu ce qu'il est possible de faire grâce à la bibliothèque GD, mais vous verrez plus loin qu'on peut faire bien plus.

Manipuler les couleurs

Un ordinateur — il faut le savoir — décompose chaque couleur en Rouge-Vert-Bleu. En mélangeant les quantités de rouge, de vert et de bleu, ça nous donne une couleur parmi les millions de possibilités !

On indique la « quantité » de rouge, de vert et de bleu par un nombre compris entre 0 et 255.

- Par exemple, si je dis que je mets 255 de bleu, ça veut dire qu'on met tout le bleu.
- Si je mets 100 de bleu, il y a un peu moins de bleu.
- Si je mets 0, alors là il n'y a plus du tout de bleu.

On doit écrire les trois quantités dans l'ordre RVB (Rouge Vert Bleu). Par exemple :

```
( 255 0 0 ).
```

Ça, c'est une couleur qui contient plein de rouge, et pas du tout de vert ni de bleu. C'est donc la couleur... rouge !
Bravo !

Maintenant, si je mets plein de rouge et de vert :

```
( 255 255 0 ).
```

Ça nous donne la couleur : jaune !

Allez, un dernier essai pour la route et on arrête là :

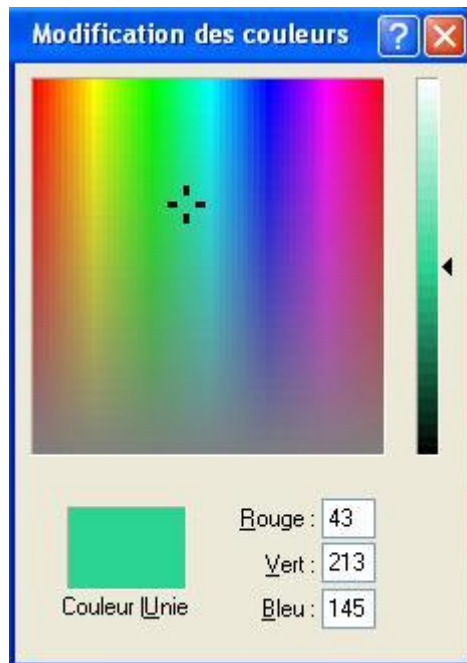
(255 128 0).

Ça, c'est la couleur orange !



Pour info, la couleur blanche correspond à (255 255 255), et la couleur noire à (0 0 0).

Si vous avez un logiciel de dessin comme Paint et que vous allez dans le menu `Couleur / Modifier les couleurs`, vous pouvez choisir la couleur que vous voulez, comme sur la figure suivante.



Sélection d'une couleur

Comme vous pouvez le voir, en cliquant sur la couleur qui vous intéresse on vous donne les quantités de Rouge Vert Bleu.

Vous pouvez donc choisir la couleur que vous voulez. Allez-y, servez-vous.

Mais revenons à ce qui nous intéresse : PHP.

Pour définir une couleur en PHP, on doit utiliser la fonction `imagecolorallocate` .

On lui donne quatre paramètres : l'image sur laquelle on travaille, la quantité de rouge, la quantité de vert, et la quantité de bleu.

Cette fonction nous renvoie la couleur dans une variable. Grâce à cette fonction, on va pouvoir se créer des « variables-couleur » qui vont nous être utiles pour indiquer ensuite la couleur.

Voici quelques exemples de création de couleur :

```
1 <?php
2 header ("Content-type: image/png");
3 $image = imagecreate(200,50);
4
5 $orange = imagecolorallocate($image, 255, 128, 0);
6 $bleu = imagecolorallocate($image, 0, 0, 255);
7 $bleuclair = imagecolorallocate($image, 156, 227, 254);
```

php

```
8 $noir = imagecolorallocate($image, 0, 0, 0);
9 $blanc = imagecolorallocate($image, 255, 255, 255);
10
11 imagepng($image);
12 ?>
```

Une chose très importante à noter : la première fois que vous faites un appel à la fonction `imagecolorallocate`, cette couleur devient la couleur de fond de votre image.

Donc, si vous avez bien compris, ce code doit créer une image... toute orange !

Essayez !



Si j'avais voulu que le fond soit blanc et non orange, il aurait fallu placer `$blanc` en premier.

Voilà, vous savez maintenant créer toutes les couleurs de l'arc-en-ciel en PHP.

Écrire du texte

Nous voici enfin dans le vif du sujet (ouf !). Nous avons une belle image avec un magnifique fond orange, et nous voulons y écrire du texte.

Avec la fonction `imagestring`, c'est facile !

Cette fonction prend beaucoup de paramètres. Elle s'utilise comme ceci :

```
1 <?php
2 imagestring($image, $police, $x, $y, $texte_a_ecrire, $couleur);
3 ?>
```

php



Il existe aussi la fonction `imagestringup` qui fonctionne exactement de la même manière, sauf qu'elle écrit le texte verticalement et non horizontalement !

Je vous détaille ci-dessous les paramètres dans l'ordre, c'est important que vous compreniez bien.

- `$image` : c'est notre fameuse variable qui contient l'image.
- `$police` : c'est la police de caractères que vous voulez utiliser. Vous devez mettre un nombre de 1 à 5 ; 1 = petit, 5 = grand. Il est aussi possible d'utiliser une police de caractères personnalisée, mais il faut avoir des polices dans un format spécial qu'il serait trop long de détailler ici. On va donc se contenter des polices par défaut. ;-)
- `$x` et `$y` : ce sont les coordonnées auxquelles vous voulez placer votre texte sur l'image. Et là vous vous dites : « Aïe, ça sent les maths » (comme quoi les maths, ça sert). Vous devez savoir que l'origine se trouve en haut à gauche de votre image. Le point de coordonnées (0, 0) représente donc le point tout en haut à gauche de l'image.

Voici, en figure suivante, le schéma de notre image orange de tout à l'heure, qui est de taille 200x50 :



Coordonnées de l'image



Notez que les coordonnées de l'image s'arrêtent à (199, 49) et non à (200, 50) comme on pourrait s'y attendre. En effet, il ne faut pas oublier qu'on commence à compter à partir de 0 ! De 0 à 199 il y a bien 200 points.

Comme vous pouvez le voir, j'ai marqué les quatre points des côtés de l'image. (0, 0) se trouve tout en haut à gauche, et (199, 49) se trouve tout en bas à droite.

- `$texte_a_ecrire`, c'est le... texte que vous voulez écrire. Non, non, il n'y a pas de piège.
- `$couleur`, c'est une couleur que vous avez créée tout à l'heure avec `imagecolorallocate`.

Voici un exemple concret de ce qu'on peut faire :

php

```

1 <?php
2 header ("Content-type: image/png");
3 $image = imagecreate(200,50);
4
5 $orange = imagecolorallocate($image, 255, 128, 0);
6 $bleu = imagecolorallocate($image, 0, 0, 255);
7 $bleuclair = imagecolorallocate($image, 156, 227, 254);
8 $noir = imagecolorallocate($image, 0, 0, 0);
9 $blanc = imagecolorallocate($image, 255, 255, 255);
10
11 imagestring($image, 4, 35, 15, "Salut les Zéros !", $blanc);
12
13 imagepng($image);
14 ?>

```



Résultat

La ligne contenant `imagestring` peut se traduire par : **Mets dans l'image \$image, avec la police de taille 4, aux coordonnées (35, 15), le texte « Salut les Zéros ! », de couleur blanche.**

Dessiner une forme



Dessiner du texte c'est bien, mais ça serait bête d'être limité à ça. Heureusement, PHP a pensé à tout !

Graphistes en herbe, vous allez certainement trouver votre bonheur dans toutes ces fonctions : vous pouvez créer des lignes, des rectangles, des cercles, des polygones...

Je vais vous présenter la plupart de ces fonctions ci-dessous, et je vous montrerai ensuite ce que ça donne dans une image de taille 200×200 , histoire d'avoir un aperçu.

ImageSetPixel

Son rôle : dessiner un pixel aux coordonnées (x, y) .

php

```
1 ImageSetPixel ($image, $x, $y, $couleur);
```

Illustration en figure suivante, grâce au code : `ImageSetPixel ($image, 100, 100, $noir);`



x, y

ImageSetPixel

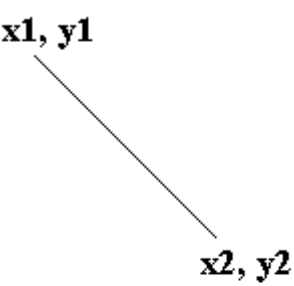
ImageLine

Celle-là sert à dessiner une ligne entre deux points de coordonnées (x_1, y_1) et (x_2, y_2) .

php

```
1 ImageLine ($image, $x1, $y1, $x2, $y2, $couleur);
```

La preuve en image : figure suivante. Le code utilisé est le suivant : `ImageLine ($image, 30, 30, 120, 120, $noir);`



x1, y1

x2, y2

ImageLine

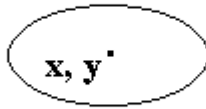
ImageEllipse

Celle-ci dessine une ellipse dont le centre est aux coordonnées (x, y) , de largeur `$largeur` et de hauteur `$hauteur`.

php

```
1 ImageEllipse ($image, $x, $y, $largeur, $hauteur, $couleur);
```

Voici une illustration en figure suivante. Et voici son code : `ImageEllipse ($image, 100, 100, 100, 50, $noir);`



ImageEllipse

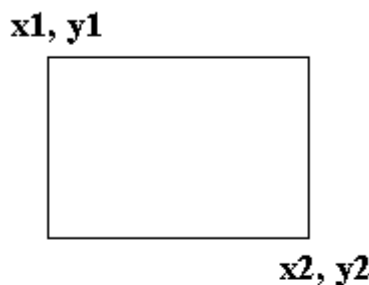
ImageRectangle

Elle, elle dessine un rectangle, dont le coin en haut à gauche est de coordonnées (x_1, y_1) et celui en bas à droite (x_2, y_2) .

```
1 ImageRectangle ($image, $x1, $y1, $x2, $y2, $couleur);
```

php

Figure suivante, vous avez le résultat produit par le code suivant : `ImageRectangle ($image, 30, 30, 160, 120, $noir);`



ImageRectangle

ImagePolygon

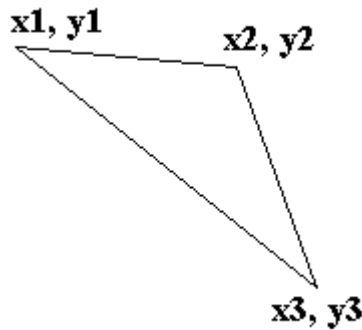
Elle dessine un polygone ayant un nombre de points égal à `$nombre_de_points` (s'il y a trois points, c'est donc un triangle). L'array `$array_points` contient les coordonnées de tous les points du polygone dans l'ordre : $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4, \dots$

```
1 ImagePolygon ($image, $array_points, $nombre_de_points, $couleur);
```

php

Allez, la figure suivante pour vous donner un exemple, dont le code est :

```
$points = array(10, 40, 120, 50, 160, 160); ImagePolygon ($image, $points, 3, $noir);
```



ImagePolygon

Des fonctions encore plus puissantes



Des rectangles, des ellipses, des lignes... Ouais, bof. C'est tout ce qu'on peut faire ?

Bien sûr que non ! Il y a d'autres fonctions que je veux absolument vous montrer parce qu'elles permettent de réaliser des opérations bien plus complexes !

Nous allons apprendre à :

- rendre une image transparente ;
- mélanger deux images ;
- redimensionner une image, pour créer une miniature par exemple.

J'espère que vous êtes encore en forme, ce serait dommage de s'endormir sur les fonctions les plus intéressantes.

Rendre une image transparente

Tout d'abord, il faut savoir que seul le PNG peut être rendu transparent. En effet, un des gros défauts du JPEG est qu'il ne supporte pas la transparence. Nous allons donc ici travailler sur un PNG.

Rendre une image transparente est d'une facilité déconcertante. Il suffit d'utiliser la fonction `imagecolortransparent` et de lui indiquer quelle couleur on veut rendre transparente. Cette fonction s'utilise comme ceci :

php

```
1 <?php
2 imagecolortransparent($image, $couleur);
3 ?>
```

Je vais reprendre l'exemple de l'image où j'ai écrit « Salut les Zéros ! » sur un vieux fond orange, et je vais y rajouter la fonction `imagecolortransparent` pour rendre ce fond transparent :

php

```
1 <?php
2 header ("Content-type: image/png");
3 $image = imagecreate(200,50);
4
5 $orange = imagecolorallocate($image, 255, 128, 0); // Le fond est orange (car c'est la première couleur)
6 $bleu = imagecolorallocate($image, 0, 0, 255);
```

```
7 $bleuclair = imagecolorallocate($image, 156, 227, 254);
8 $noir = imagecolorallocate($image, 0, 0, 0);
9 $blanc = imagecolorallocate($image, 255, 255, 255);
10
11 imagestring($image, 4, 35, 15, "Salut les Zéros !", $noir);
12 imagecolortransparent($image, $orange); // On rend le fond orange transparent
13
14 imagepng($image);
15 ?>
```

Et voilà, sur la figure suivante, le PNG transparent que ça nous donne.

Salut les Zéros !

Un texte dans un PNG

Sympa, non ? ;-)

Mélanger deux images

Ça, c'est un tout petit peu plus compliqué que de rendre une image transparente, mais bon je vous rassure : c'est loin d'être insurmontable quand même et ça en vaut la peine.

La fonction que je vais vous présenter permet de « fusionner » deux images en jouant sur un effet de transparence. Ça a l'air tordu comme ça, mais c'est en fait quelque chose de vraiment génial !

On peut s'en servir par exemple pour afficher le logo de son site sur une image. Voyez le logo en figure suivante.



Logo

Et en figure suivante, l'image en question.



Image à marquer par le logo

La fonction qui permet de réaliser la fusion entre deux images est `imagecopymerge`.

Ce script est un peu plus gros que les autres, alors je préfère vous le donner tout de suite. Je vous expliquerai juste après comment il fonctionne.

php

```
1 <?php
2 header ("Content-type: image/jpeg"); // L'image que l'on va créer est un jpeg
3
4 // On charge d'abord les images
5 $source = imagecreatefrompng("logo.png"); // Le logo est la source
6 $destination = imagecreatefromjpeg("couchersoleil.jpg"); // La photo est la destination
7
8 // Les fonctions imagesx et imagesy renvoient la largeur et la hauteur d'une image
9 $largeur_source = imagesx($source);
10 $hauteur_source = imagesy($source);
11 $largeur_destination = imagesx($destination);
12 $hauteur_destination = imagesy($destination);
13
14 // On veut placer le logo en bas à droite, on calcule les coordonnées où on doit placer le logo sur la
  photo
15 $destination_x = $largeur_destination - $largeur_source;
16 $destination_y = $hauteur_destination - $hauteur_source;
17
18 // On met le logo (source) dans l'image de destination (la photo)
19 imagecopymerge($destination, $source, $destination_x, $destination_y, 0, 0, $largeur_source,
  $hauteur_source, 60);
20
21 // On affiche l'image de destination qui a été fusionnée avec le logo
22 imagejpeg($destination);
23 ?>
```

Vous trouverez en figure suivante le résultat que donne ce script.



Le logo a été intégré dans l'image

`imagecopymerge` est une fonction vraiment sympa, parce que vous allez maintenant pouvoir « copyrighter » automatiquement toutes les images de votre site si vous le voulez.

Cependant, le script utilisé ici est un petit peu plus complexe, et je crois que quelques explications ne seraient pas de refus.

Voici donc les points à bien comprendre.

- Dans ce script, on manipule deux images : `$source` (le logo) et `$destination` (la photo). Les deux sont créées à l'aide de la fonction `imagecreatefrompng` (et `fromjpeg` pour la photo).
- Il y a ensuite toute une série de calculs à partir des coordonnées et de la largeur et hauteur des images. J'imagine que ça a dû vous faire peur, mais c'est en fait très simple du moment qu'on sait faire une soustraction.

Notre but est de savoir à quelles coordonnées placer le logo sur la photo. Moi, je veux le mettre tout en bas à droite. Pour ça, j'ai besoin de connaître les dimensions des images. J'utilise les fonctions `imagesx` et `imagesy` pour récupérer les dimensions du logo et de la photo. Ensuite, pour placer le logo tout en bas, il faut le mettre à la position `$hauteur_de_la_photo - $hauteur_du_logo`. On fait de même pour placer le logo à droite : `$largeur_de_la_photo - $largeur_du_logo`. Si j'avais voulu mettre le logo tout en haut à gauche, là, ça aurait été beaucoup plus simple : pas besoin de faire de calculs, vu qu'en haut à gauche les coordonnées sont (0, 0) !

- Vient ensuite la fonction `imagecopymerge`, la plus importante. Elle prend de nombreux paramètres. Ce qu'il faut savoir, c'est qu'elle a besoin de deux images : une source et une destination. Elle modifie l'image de destination (ici, la photo) pour y intégrer l'image source. Cela explique pourquoi c'est `$destination` que l'on affiche à la fin, et non pas `$source` (le logo) qui n'a pas changé.

Les paramètres à donner à la fonction sont, dans l'ordre, les suivants.

1. **L'image de destination** : ici `$destination`, la photo. C'est l'image qui va être modifiée et dans laquelle on va mettre notre logo.
2. **L'image source** : ici `$source`, c'est notre logo. Cette image n'est pas modifiée.
3. **L'abscisse à laquelle vous désirez placer le logo sur la photo** : il s'agit ici de l'abscisse du point située à la position `$largeur_de_la_photo - $largeur_du_logo`.
4. **L'ordonnée à laquelle vous désirez placer le logo sur la photo** : de même, il s'agit de l'ordonnée du point sur la photo (ici, `$hauteur_de_la_photo - $hauteur_du_logo`).
5. **L'abscisse de la source** : en fait, la fonction `imagecopymerge` permet aussi de ne prendre qu'une partie de l'image source. Ça peut devenir un peu compliqué, alors nous, on va dire qu'on prend tout le logo. On part donc du point situé aux coordonnées (0, 0) de la source. Mettez donc 0 pour l'abscisse.
6. **L'ordonnée de la source** : de même pour l'ordonnée. Mettez 0.
7. **La largeur de la source** : c'est la largeur qui détermine quelle partie de l'image source vous allez prendre. Nous on prend toute l'image source, ne vous prenez donc pas la tête non plus et mettez `$largeur_source`.
8. **La hauteur de la source** : de même, mettez `$hauteur_source`.
9. **Le pourcentage de transparence** : c'est un nombre entre 0 et 100 qui indique la transparence de votre logo sur la photo. Si vous mettez 0, le logo sera invisible (totalement transparent), et si vous mettez 100, il sera totalement opaque (il n'y aura pas d'effet de « fusion »). Mettez un nombre autour de 60-70, en général c'est bien. ;-)

Concrètement, on peut se servir de ce code pour faire une page `copyrighter.php`. Cette page prendra un paramètre : le nom de l'image à « copyrighter ».

Par exemple, si vous voulez « copyrighter » automatiquement `tropiques.jpg`, vous afficherez l'image comme ceci :

php

```
1 
```

À vous maintenant d'écrire la page `copyrighter.php`.

Si vous vous basez sur le script que je vous ai donné, ça ne devrait pas être bien long. Il faut juste récupérer le nom de l'image à charger (via la variable `$_GET['image']`). Arf, ça y est, je vous ai tout dit.

Redimensionner une image

C'est une des fonctionnalités les plus intéressantes de la bibliothèque GD, à mon goût. Ça permet de créer des miniatures de nos images.

Vous pouvez vous en servir par exemple pour faire une galerie de photos. Vous affichez les miniatures et si vous cliquez sur l'une d'elles, ça l'affiche dans sa taille originale.

Pour redimensionner une image, on va utiliser la fonction `imagecopyresampled`. C'est une des fonctions les plus poussées car elle fait beaucoup de calculs mathématiques pour créer une miniature de bonne qualité. Le résultat est très bon, mais cela donne énormément de travail au processeur.



Cette fonction est donc puissante mais lente. Tellement lente que certains hébergeurs désactivent la fonction pour éviter que le serveur ne rame. Il serait suicidaire d'afficher directement l'image à chaque chargement d'une page. Nous allons ici créer la miniature une fois pour toutes et l'enregistrer dans un fichier.

Nous allons donc enregistrer notre miniature dans un fichier (`mini_couchersoleil.jpg`, par exemple). Cela veut dire qu'on peut déjà retirer la première ligne (le `header`) qui ne sert plus à rien.

Comme pour `imagecopymerge`, on va avoir besoin de deux images : la source et la destination. Ici, la source c'est l'image originale et la destination, l'image miniature que l'on va créer.

La première chose à faire sera donc de créer une nouvelle image vide... Avec quelle fonction ? `imagecreate` ? Oui, c'est presque la bonne réponse.

Le problème voyez-vous, c'est que `imagecreate` crée une nouvelle image dont le nombre de couleurs est limité (256 couleurs maximum, en général). Or, notre miniature contiendra peut-être plus de couleurs que l'image originale à cause des calculs mathématiques.

On va donc devoir utiliser une autre fonction dont je ne vous ai pas encore parlé : `imagecreatetruecolor`. Elle fonctionne de la même manière que `imagecreate` mais cette fois, l'image pourra contenir beaucoup plus de couleurs.

Voici le code que je vais utiliser pour générer la miniature de `couchersoleil.jpg`, ma photo :

php

```
1 <?php
2 $source = imagecreatefromjpeg("couchersoleil.jpg"); // La photo est la source
3 $destination = imagecreatetruecolor(200, 150); // On crée la miniature vide
4
5 // Les fonctions imagesx et imagesy renvoient la largeur et la hauteur d'une image
6 $largeur_source = imagesx($source);
```



```
7 $hauteur_source = imagesy($source);
8 $largeur_destination = imagesx($destination);
9 $hauteur_destination = imagesy($destination);
10
11 // On crée la miniature
12 imagecopyresampled($destination, $source, 0, 0, 0, 0, $largeur_destination, $hauteur_destination,
13 $largeur_source, $hauteur_source);
14
15 // On enregistre la miniature sous le nom "mini_couchersoleil.jpg"
16 imagejpeg($destination, "mini_couchersoleil.jpg");
17 ?>
```

Avant, l'image ressemblait à la figure suivante. Grâce à `imagecopyresampled`, on a obtenu une version miniature (figure suivante).



L'image d'origine



L'image redimensionnée

Vous pouvez afficher ensuite l'image avec le code HTML :

```

```

On a créé notre miniature vide avec `imagecreatetruecolor` en dimensions réduites (

200 × 150

). Je vous ai déjà expliqué les fonctions `imagesx` et `imagesy`, je n'y reviens pas. Voyons plutôt quels sont les paramètres de la fonction `imagecopyresampled`.

1. **L'image de destination** : c'est `$destination`, l'image qu'on a créée avec `imagecreatetruecolor`.
2. **L'image source** : l'image dont on veut créer la miniature ; ici, c'est notre `couchersoleil.jpg` qu'on a chargée avec `imagecreatefromjpeg`.
3. **L'abscisse du point à laquelle vous placez la miniature sur l'image de destination** : pour faire simple, on va dire que notre image de destination contiendra uniquement la miniature. Donc on placera la miniature aux coordonnées (0, 0), ce qui fait qu'il faut mettre 0 à cette valeur.
4. **L'ordonnée du point à laquelle vous placez la miniature sur l'image de destination** : pour les mêmes raisons, mettez 0.
5. **L'abscisse du point de la source** : ici, on prend toute l'image source et on en fait une miniature. Pour tout prendre, il faut partir du point (0, 0), ce qui fait que là encore on met 0 à cette valeur.
6. **L'ordonnée du point de la source** : encore 0.
7. **La largeur de la miniature** : un des paramètres les plus importants, qui détermine la taille de la miniature à créer. Dans notre cas notre miniature fait 200 pixels de large. On a stocké ce nombre dans la variable `$largeur_destination`.
8. **La hauteur de la miniature** : de même pour la hauteur de la miniature à créer.
9. **La largeur de la source** : il suffit d'indiquer la taille de notre image source. On a stocké cette valeur dans `$largeur_source`, donc on la réutilise ici.
10. **La hauteur de la source** : de même pour la hauteur.

Comme vous pouvez le voir, `imagecopyresampled` permet de faire beaucoup de choses, et en général on ne se servira pas de tout.

Plusieurs paramètres sont à 0, et ce n'est pas vraiment la peine de chercher à comprendre pourquoi (même si ce n'est pas bien compliqué). Basez-vous sur mon exemple pour créer vos miniatures, et le tour est joué.

En résumé

- PHP permet de faire bien plus que générer des pages web HTML. En utilisant des extensions, comme la bibliothèque GD, on peut par exemple générer des images.
- Pour qu'une page PHP renvoie une image au lieu d'une page web, il faut modifier l'en-tête HTTP à l'aide de la fonction `header()` qui indiquera alors au navigateur du visiteur l'arrivée d'une image.
- Il est possible d'enregistrer l'image sur le disque dur si on le souhaite, ce qui évite d'avoir à la générer à chaque fois qu'on appelle la page PHP.
- On peut créer des images avec GD à partir d'une image vide ou d'une image déjà existante.
- GD propose des fonctions d'écriture de texte dans une image et de dessin de formes basiques.
- Des fonctions plus avancées de GD permettent de fusionner des images ou d'en redimensionner.



[Que pensez-vous de ce cours ?](#)



Activity: Le mini-chat amélioré

Les expressions régulières (partie 1/2)



The author

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Check out this course via



eBook



Livre papier



PDF



Vidéo

OpenClassrooms

[Who we are](#)[How our courses work](#)[Jobs](#)[Contact us](#)

Partnerships

[Affiliate program](#)

Learn more

[Terms of Use](#)

Follow us

[OpenClassrooms blog](#)[Español](#)[Français](#)