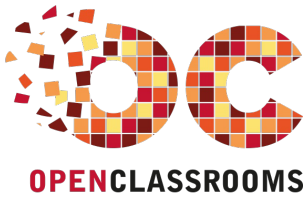
Faire une image anti-robots (ou captcha) facilement

Par Francois GAY (fecgay)
et Benoît CRAUET (Kermi)



www.openclassrooms.com

Dernière mise à jour le 1/07/2010

Sommaire

Faire une image anti-robots (ou captcha) facilement

Par Benoît CRAUET (Kermi) et Francois GAY (fecgay)

Mise à jour : 01/07/2010
Difficulté : Facile Durée d'étude : 30 minutes

Bien le bonjour !

Dans ce tutoriel, nous allons réaliser un générateur de captchas en PHP. Tout d'abord, je réponds à tous ceux qui se posent encore la question :

Qu'est-ce qu'un captcha ?

Un captcha est une image contenant un code que doit composer l'utilisateur lors d'une inscription, dans un formulaire de contact, etc. Ce concept n'est pas sans utilité, évidemment. En effet, cela empêche les robots de poster des messages de spam : sur votre site, sur vos forums, dans votre boîte mail, etc.

C'est compliqué à réaliser ?

Comment dire… Prenez le module GD, une session, une condition, et le compte devrait être bon !
Plus sérieusement, il est extrêmement simple de réaliser une image de ce type.

Commençons !

Sommaire du tutoriel :

Caractéristiques et explication du code

Bien, nous allons voir que ce script ne requiert pas beaucoup de choses en soi, étant donné qu'il ne nécessite que GD…

Euh… c'est bien tout ça, mais c'est quoi GD ?

Vous, vous n'avez pas lu le tutoriel PHP / MySQL de M@teo 21 en entier !

GD, si vous ne savez pas ce que c'est (et c'est bien dommage !), est un **module PHP** qui permet de programmer des images dites ***dynamiques***.

Par exemple, une image de cette sorte peut afficher la date et l'heure actuelles, et tout ce qui est affichable (donc à peu près tout

) sur une page XHTML.
La différence est que vous affichez ces informations sur… une image !

Pour plus d'informations sur GD, allez voir le tuto PHP GD de M@teo21 qui explique parfaitement toutes les bases.
Le module GD est présent chez tous les hébergeurs aujourd'hui. Très rares sont ceux qui ne le prennent pas en charge.

C'est pas tout, mais codons un peu !

Ce petit script va s'occuper de créer une variable qui va nous être fort utile : c'est une session, et elle se nomme **aleat_nbr**.
Nous comparerons ensuite celle-ci avec une autre variable, comme **$_POST['code_verif']** par exemple…

Alors sortez tous vos éditeurs, bande de bachi-bouzouks !

Création du script principal

Nous allons créer, ici, le script principal de notre image !
En effet, nous allons, avec quelques lignes, créer une image dynamique et aléatoire.

Pour commencer, créez un fichier nommé **verif_code_gen.php** dans un répertoire vierge.
Ensuite, enregistrez dans le même répertoire cette image, que vous appellerez **fond_verif_img.png** (Clic droit > *Enregistrer la cible sous…*).

Nous allons tout d'abord ouvrir le fichier verif_code_gen.php, que vous avez normalement créé tout à l'heure.
Il nous faut donc maintenant démarrer la session **avant tout**. Nous devrons ensuite créer une image depuis **fond_verif_img.png**.

Je vous ajoute en prime dans ce code le header(), pour afficher une image, et la variable $nbr_chiffres, qui contient le nombre de chiffres aléatoires à afficher.

Voici ce début de code :

Code : PHP

```php
<?php
// On crée la session avant tout
session_start();

// On définit la configuration :
$nbr_chiffres = 6; // Nombre de chiffres qui formeront le nombre

// Là, on définit le header de la page pour la transformer en image
header ("Content-type: image/png");
// Là, on crée notre image
$_img = imagecreatefrompng('fond_verif_img.png');
```

Il nous faut aussi définir les couleurs de l'image :

Code : PHP

```php
<?php
// On définit maintenant les couleurs
// Couleur de fond :
$arriere_plan = imagecolorallocate($_img, 0, 0, 0); // Au cas où on
n'utiliserait pas d'image de fond, on utilise cette couleur-là.
// Autres couleurs :
$avant_plan = imagecolorallocate($_img, 255, 255, 255); // Couleur
des chiffres
?>
```

Maintenant, nous allons ajouter le module de "montage" du nombre aléatoire, qui fonctionne grâce à un array et à un foreach().
Le nombre généré sera ensuite enregistré dans **$_SESSION['aleat_nbr']**.

À la fin de la génération du code, on détruit toutes les variables inutiles avec la fonction unset().

Code : PHP

```php
<?php
##### Ici on crée la variable qui contiendra le nombre aléatoire
#####
$i = 0;
while($i < $nbr_chiffres) {
        $chiffre = mt_rand(0, 9); // On génère le nombre aléatoire
        $chiffres[$i] = $chiffre;
        $i++;
}
$nombre = null;
// On explore le tableau $chiffres afin d'y afficher toutes les
entrées qui s'y trouvent
foreach ($chiffres as $caractere) {
        $nombre .= $caractere;
}
##### On a fini de créer le nombre aléatoire, on le rentre
maintenant dans une variable de session #####
$_SESSION['aleat_nbr'] = $nombre;
// On détruit les variables inutiles :
unset($chiffre);
unset($i);
unset($caractere);
unset($chiffres);
?>
```

Mais ? Pourquoi s'embêter à faire un monstre pareil pour afficher un nombre aléatoire de 6 chiffres ? Pourquoi ne pas utiliser tout simplement un mt_rand(0, 999999); ?

Ah oui !
Cette question est souvent posée, mais cette technique a pourtant un but.

En fait, cette méthode n'est pas très fiable. Si le nombre sélectionné est 007865, le code s'affichera comme suit : 7865.
Cela fait donc 2 caractères en moins ! Imaginez que notre mt_rand() sélectionne le nombre 0… Nous nous retrouverions avec un code à 1 caractère… Ça ferait fouillis !

Donc, pour garder le nombre de caractères, nous devons inscrire 6 entrées dans un array, qui contiendront chacune un nombre aléatoire entre 0 et 9. Or, le chiffre 0 est affiché s'il est seul.

Comment va se dépatouiller notre script dans ce cas ?

Pour résumer, dans ce code, PHP va générer ça différemment.

1. Je crée l'entrée 0 dans l'array, et je sélectionne aléatoirement un chiffre entre 0 et 9.
2. Je crée l'entrée 1 dans l'array, et je sélectionne aléatoirement un chiffre entre 0 et 9.
3. Je crée l'entrée 2 dans l'array, et je sélectionne aléatoirement un chiffre entre 0 et 9.
4. Je crée l'entrée 3 dans l'array, et je sélectionne aléatoirement un chiffre entre 0 et 9.
5. etc.

Tout simplement !
Désormais, notre code affichera **007865** au lieu de **7865**.

Pour terminer notre code, nous allons afficher la variable $nombre (qui je vous le rappelle contient le code généré) dans notre image. Ensuite, nous afficherons notre image, comme ceci :

Code : PHP

```php
<?php
imagestring($_img, 5, 18, 8, $nombre, $avant_plan);

imagepng($_img);
?>
```

Et je vous informe désormais que notre script est… terminé ! 

Si vous n'avez grillé aucune étape, et bien écouté en classe, vous devriez avoir un script ressemblant à celui-ci :

Code : PHP

```php
<?php
// On crée la session avant tout
session_start();

// On définit la configuration :
$nbr_chiffres = 6; // Nombre de chiffres qui formeront le nombre

// Là, on définit le header de la page pour la transformer en image
header ("Content-type: image/png");
// Là, on crée notre image
$_img = imagecreatefrompng('fond_verif_img.png');

// On définit maintenant les couleurs
// Couleur de fond :
$arriere_plan = imagecolorallocate($_img, 0, 0, 0); // Au cas où on
n'utiliserait pas d'image de fond, on utilise cette couleur-là.
// Autres couleurs :
$avant_plan = imagecolorallocate($_img, 255, 255, 255); // Couleur
des chiffres

##### Ici on crée la variable qui contiendra le nombre aléatoire
#####
$i = 0;
while($i < $nbr_chiffres) {
        $chiffre = mt_rand(0, 9); // On génère le nombre aléatoire
        $chiffres[$i] = $chiffre;
        $i++;
}
$nombre = null;
// On explore le tableau $chiffres afin d'y afficher toutes les
entrées qui s'y trouvent
foreach ($chiffres as $caractere) {
        $nombre .= $caractere;
}
##### On a fini de créer le nombre aléatoire, on le rentre
maintenant dans une variable de session #####
$_SESSION['aleat_nbr'] = $nombre;
// On détruit les variables inutiles :
unset($chiffre);
unset($i);
unset($caractere);
unset($chiffres);

imagestring($_img, 5, 18, 8, $nombre, $avant_plan);

imagepng($_img);
?>
```

Ce simple script vous permet de protéger vos formulaires des vilains robots en soif de spam et autres.  Avouez que ça fait

peur, non ?

Création de notre formulaire de vérification

Stop !
Voici une partie XHTML que je ne traiterai pas ici. J'estime que si vous en êtes au PHP, vous savez déjà faire du XHTML.

Si ce n'est pas chose faite, je vous invite à aller lire sans plus attendre le tutoriel sur le XHTML/CSS de M@teo21 ou, si vous le préférez, le livre du même auteur : Réussir son site web avec XHTML et CSS.

Créez tout d'abord le fichier **inscription.html** que vous mettrez dans le même répertoire que tous les autres fichiers créés.

Ensuite, mettez-y ce code HTML :

Code : HTML

```
<html>
<head>
<title>Inscription</title>
</head>
<body>
<form action="verif_insc.php" method="post">
<p><label>Pseudo</label> : <input type="text" name="pseudo" /></p>
<p><label>Mot de passe</label> : <input type="password" name="mdp"
/></p>
<p><label>Retapez votre MdP</label> : <input type="password"
name="mdpv" /></p>
<p><label>E-Mail</label> : <input type="text" name="email" /></p>
<p><label>Retapez votre E-Mail</label> : <input type="text"
name="emailv" /></p>
<!-- On affiche l'image générée par notre script -->
<p><img src="verif_code_gen.php" alt="Code de vérification" /></p>

<p><label>Merci de retaper le code de l'image ci-dessus</label> :
<input type="text" name="verif_code" /></p>
<p><input type="submit" value="Inscription" /></p>
</form>
</body>
</html>
```

Rien ne vous empêche de personnaliser ce code comme bon vous semble, ce n'est évidemment qu'un exemple parmi tant d'autres !

Création de la page de confirmation

Nous voici face à notre visiteur, qui a gentiment lu notre image, et a tout aussi gentiment rempli le champ correspondant, avant de poster le formulaire.
Nous nous intéressons à présent au traitement de notre image de vérification.

Vous devez créer le fichier **verif_insc.php** dans le même répertoire que inscription.html.

Maintenant, entrez-y ce code :

Code : PHP - Script de vérification

```
<?php
session_start(); // L'image utilise les sessions. On les active
donc ici, car on a besoin de ces informations.
$debut_html = '<html>
<head>
<title>Inscription</title>
</head>
```

```
<body>
<p>';
$milieu_html = NULL;
$fin_html = '</p>
</body>
</html>';

if(IsSet($_POST['verif_code']) AND !Empty($_POST['verif_code'])) {
// Le champ du code de confirmation a été rempli
if($ POST['verif code']==$ SESSION['aleat_nbr']) { // Si le champ
est égal au code généré par l'image
          $milieu_html = 'Vous avez entré le bon code de
confirmation !';
     }
     else {
          $milieu_html = 'Votre code de confirmation n\'est pas bon
! Merci de réessayer.<br /><a href="#" onclick="history.go(-
1);">Retour</a>';
     }
}
else {
    $milieu_html = 'Vous devez remplir le champ du code de
confirmation !';
}

// Là, on affiche toute la source générée :
echo $debut_html . $milieu_html . $fin_html;

?>
```

La ligne 15, qui a été surlignée, nous intéresse tout particulièrement :

Code : PHP

```
<?php
if($ POST['verif code']==$ SESSION['aleat_nbr']) { // Si le champ
est égal au code généré par l'image
?>
```

Cette condition, aussi simple soit-elle, nous permet tout simplement de vérifier si le code qu'a entré l'utilisateur (**$_POST['verif_code']**) est identique à celui de notre image (**$_SESSION['aleat_nbr']**).

Vous pouvez modifier ce script comme bon vous semble, selon vos besoins.
Vous avez donc vu la facilité de créer une image captcha pour votre site.
Il suffit juste de savoir manier les sessions et le GD !

Cependant, sachez que les robots sont de plus en plus sophistiqués dans l'analyse des images de protection.
Ce tuto n'est qu'une première approche dans le domaine.

Faites-en bon usage.

Ce tutoriel a été corrigé par les zCorrecteurs.