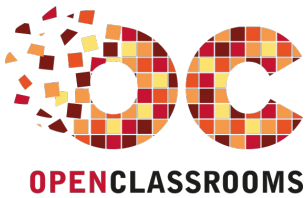
Un système de cache simple

Par Benoît CRAUET (Kermi)



www.openclassrooms.com

Dernière mise à jour le 18/10/2010

Sommaire

Un système de cache simple

Par Benoît CRAUET (Kermi)

Mise à jour : 18/10/2010

Beaucoup de sites ne sont malheureusement pas assez optimisés, et cela pour plusieurs raisons :

- les jointures ou les unions SQL ne sont pas utilisées ;
- les " sont utilisés à la place des ' ;
- des inclusions de fichiers inutiles ;
- l'absence d'un système de cache.

Je vais vous apprendre dans ce tutoriel à réaliser un système de cache simple, basé sur un ou plusieurs fichiers (tout dépend si vous souhaitez installer un ou plusieurs caches).

Nous prendrons l'exemple de l'ancien menu « **Livre d'or** » du Site du Zéro : chaque message est mis en cache pendant une minute pour être ensuite mis à jour en puisant un autre message dans la base de données.
Sommaire du tutoriel :

Préparation du script

Nous allons commencer par préparer notre script.

Création du fichier .cache

Pour cela, il vous faut créer un fichier *.cache.

Par souci d'organisation, je vous conseille de mettre tous vos fichiers *.cache dans des dossiers différents, tous stockés dans le répertoire /cache.

Pour notre script, nous allons créer un fichier cache contenant un message du livre d'or, qui changera toutes les minutes et qui s'affichera dans un block du menu. Cette information apparaîtra sur toutes les pages (puisque le menu figure sur toutes les pages). Nous allons donc mettre ça directement dans le dossier /cache :

/cache/livre_or.cache

Créez donc ce fichier et mettez-le en CHMOD 777 (tous les droits).

Et ce fichier va me servir à quoi ?

En fait, je vais être franc avec vous : il ne vous servira à rien… Il sera en revanche très utile à PHP !

PHP va s'en servir pour stocker des informations temporaires (ici, le message du livre d'or).

Sur votre site ne s'affichera que le texte contenu dans le fichier, et non le texte issu de la requête SQL qui récupère les informations dans la base de données.

Cette requête ne s'effectuera que ponctuellement. Dans notre exemple, la requête ne s'exécutera qu'une seule fois toutes les minutes grâce à notre système de cache. C'est un gain énorme de performance, et là réside tout son intérêt !

Mais, mais ! Comment je fais, moi, si je veux que le message soit affiché en permanence ?

Un système de cache est justement conçu pour cela.
La requête ne s'effectue qu'une seule fois par minute et son contenu, lui, est affiché en permanence !

Au final, tout le monde est gagnant :

- **le serveur MySQL :** il respire un peu et reçoit moins de requêtes par page ;
- **les visiteurs :** votre site est plus rapide sans pour autant que son contenu diminue.

Il y a donc tout à gagner en intégrant des caches sur votre site !

Même si vous utilisez des jointures, certaines grosses jointures peuvent être lourdes pour le serveur MySQL, et nécessitent aussi un système de cache.
Ne vous dites donc pas que c'est parce que votre site utilise des jointures que le serveur MySQL ne souffre pas.

Réalisation du script

Nous avons tout préparé comme il faut !

Nous avons créé notre fichier .cache et nous allons maintenant nous en servir par le biais de PHP.

Nous allons donc commencer par vérifier l'ancienneté du fichier. Pour cela, nous allons utiliser la fonction filemtime(). C'est une fonction qui renvoie le timestamp de la date de dernière modification d'un fichier.

Nous allons ensuite, à partir du timestamp actuel (récupéré grâce à la fonction time()) et à partir de celui de la dernière modification du fichier cache, calculer le nombre de secondes écoulées depuis la dernière modification de notre fichier. Si ça fait plus de 60 secondes, on mettra à jour le cache.

Pour cela, une simple soustraction suffit :

Code : PHP

```php
<?php
// On soustrait du timestamp actuel celui de la dernière
modification pour obtenir le nombre de secondes écoulées depuis la
dernière modification
$modif_ago = time() - filemtime('cache/livre_or.cache');
?>
```

Maintenant, nous allons vérifier si notre fichier est assez ancien.
Sachant que le timestamp est une valeur en secondes, nous allons vérifier si le fichier a été modifié il y a plus de 60 secondes :

Code : PHP

```php
<?php
if($modif_ago > 60) { // SI le fichier a été modifié il y a plus
d'une minute
     // ON MET LE FICHIER À JOUR \\
}
?>
```

Nous n'avons pas besoin de else, car nous n'avons rien d'autre à faire si le fichier est assez récent.

Nous allons maintenant nous occuper de la mise à jour de notre fichier.
Pour cela, nous devons tout d'abord récupérer un message aléatoirement dans la base de données et exporter les données :

Code : PHP

```php
<?php
$select_message_aleatoire = mysql_query('SELECT * FROM livre_or
ORDER BY RAND() LIMIT 1'); // On sélectionne un message
aléatoirement
$infos_message = mysql_fetch_array($select_message_aleatoire);
?>
```

Dans ce code, les mots clés ORDER BY RAND() permettent d'organiser tous les messages aléatoirement, et le mot clé LIMIT 1 permet la sélection d'un seul message dans cette liste.

Étape suivante : nous devons rédiger un code qui met à jour le fichier.

Nous utiliserons fopen() avec l'option w+.
Cette option permet de réaliser plusieurs choses.

- Elle positionne le pointeur de fichier à 0 automatiquement (comme si elle remettait votre souris au début d'un fichier texte). Nous n'avons donc pas besoin d'utiliser la fonction fseek().
- Autre avantage : nous n'avons pas besoin d'effacer le contenu du fichier avant d'y écrire quoi que ce soit, car cette option réduit automatiquement la taille du fichier à 0 (elle efface donc son contenu).
- Enfin, par mesure de prévention, le fichier sera créé automatiquement si jamais un jour il est supprimé par mégarde. Cette option est représentée par le + de w+.

Nous allons également utiliser la fonction fwrite(), qui écrit dans le fichier ; et pour finir, fclose() qui ferme la « connexion » au fichier.

Code : PHP

```php
<?php
$select_message_aleatoire = mysql_query('SELECT * FROM livre_or
ORDER BY RAND() LIMIT 1'); // On sélectionne un message
aléatoirement
$infos_message = mysql_fetch_array($select_message_aleatoire);

// On fait un stripslashes sur toutes les entrées de l'array
$infos_message, la fonction array_map() permettant d'effectuer une
fonction sur toutes les entrées d'un array. Ici, la fonction
stripslashes() sera donc effectuée sur toutes les informations que
l'on a récupérées grâce à la requête SQL
array_map('stripslashes', $infos_message);
// On crée notre code xHTML
$xHTML = 'Par <strong>' . $infos_message['pseudo'] . '</strong> :<br
/>
&amp;quot;<em>' . nl2br($infos_message['message']) . '</em>&amp;quot;';
// On enregistre notre code dans le fichier...
        // On va commencer par ouvrir le fichier en w+
        $fichier = fopen('cache/livre_or.cache', 'w+');
        /* Rappel : l'option w+ ne nécessite pas le replacement du
pointeur
ni l'effacement du fichier. */

        // On écrit le code xHTML dans le fichier
        fwrite($fichier, $xHTML);
```

```
        // Pour finir, on coupe la communication avec le fichier
        fclose($fichier);
?>
```

Votre code devrait maintenant ressembler à ce qui suit.

Code : PHP

```
<?php
// On soustrait du timestamp actuel celui de la dernière
modification pour obtenir le nombre de secondes écoulées depuis la
dernière modification
$modif_ago = time() - filemtime('cache/livre_or.cache');

if($modif_ago > 60) { // SI le fichier a été modifié il y a plus
d'une minute

$select_message_aleatoire = mysql_query('SELECT * FROM livre_or
ORDER BY RAND() LIMIT 1'); // On sélectionne un message
aléatoirement
$infos_message = mysql_fetch_array($select_message_aleatoire);

        // On fait un stripslashes sur toutes les entrées de
l'array $infos_message, la fonction array_map() permettant
d'effectuer une fonction sur toutes les entrées d'un array. Ici, la
fonction stripslashes() sera donc effectuée sur toutes les
informations que l'on a récupérées grâce à la requête SQL
        array_map('stripslashes', $infos_message);
        // On crée notre code xHTML
        $xHTML = 'Par <strong>' . $infos_message['pseudo'] .
'</strong> :<br />
&amp;quot;<em>' . nl2br($infos_message['message']) . '</em>&amp;quot;';
        // On enregistre notre code dans le fichier...
                // On va commencer par ouvrir le fichier en w+
                $fichier = fopen('cache/livre_or.cache', 'w+');
                /* Rappel : l'option w+ ne nécessite pas le
replacement du pointeur
ni l'effacement du fichier. */

                // On écrit le code xHTML dans le fichier
                fwrite($fichier, $xHTML);

                // Pour finir, on coupe la communication avec le
fichier
                fclose($fichier);
}
?>
```

Afficher le contenu du fichier

Nous avons maintenant terminé le système de mise en cache !

Il nous suffit juste d'afficher le fichier, qui est le message aléatoire « mis en cache ».
Et cela s'effectue grâce à la fonction file_get_contents() :

Code : PHP

```
<?php
// On récupère le contenu de notre fichier
$message_aleatoire = file_get_contents('cache/livre_or.cache');
```

```
// On l'affiche
echo $message_aleatoire;
?>
```

Et voilà ! 
Votre code est entièrement fini !
Voici à quoi devrait ressembler votre code à présent :

Code : PHP

```
<?php
// On soustrait du timestamp actuel celui de la dernière
modification pour obtenir le nombre de secondes écoulées depuis la
dernière modification
$modif_ago = time() - filemtime('cache/livre_or.cache');

if($modif_ago > 60) { // SI le fichier a été modifié il y a plus
d'une minute

$select_message_aleatoire = mysql_query('SELECT * FROM livre_or
ORDER BY RAND() LIMIT 1'); // On sélectionne un message
aléatoirement
$infos_message = mysql_fetch_array($select_message_aleatoire);

        // On fait un stripslashes sur toutes les entrées de
l'array $infos_message, la fonction array_map() permettant
d'effectuer une fonction sur toutes les entrées d'un array. Ici, la
fonction stripslashes() sera donc effectuée sur toutes les
informations que l'on a récupérées grâce à la requête SQL
        array_map('stripslashes', $infos_message);
        // On crée notre code xHTML
        $xHTML = 'Par <strong>' . $infos_message['pseudo'] .
'</strong> :<br />
&amp;quot;<em>' . nl2br($infos_message['message']) . '</em>&amp;quot;';
        // On enregistre notre code dans le fichier...
                // On va commencer par ouvrir le fichier en w+
                $fichier = fopen('cache/livre_or.cache', 'w+');
                /* Rappel : l'option w+ ne nécessite pas le
replacement du pointeur
ni l'effacement du fichier. */

                // On écrit le code xHTML dans le fichier
                fwrite($fichier, $xHTML);

                // Pour finir, on coupe la communication avec le
fichier
                fclose($fichier);
}

// On récupère le contenu de notre fichier
$message_aleatoire = file_get_contents('cache/livre_or.cache');
// On l'affiche
echo $message_aleatoire;
?>
```

Ce qu'il se passera est simple.

- Le script vérifie si le fichier cache n'est pas « trop vieux ».
- S'il est trop vieux, il exécute la requête et met à jour le fichier. S'il n'est pas trop vieux, il laisse le fichier tel quel.
- Le script affiche le fichier cache dans tous les cas (le code est en dehors de la condition).

Le livre d'or est un exemple, mais sachez que vous pouvez aussi écrire du code PHP dans un fichier .php, par exemple ; ce sera

tout de même un système de cache.

Enfin, pour ceux qui se poseraient la question…

Pourquoi écrire tout ce code alors qu'une requête seule est plus rapide ?

Je vous dirais : **non, non et non !**

En effet, le fait de passer par des fichiers est plus rapide pour PHP que de communiquer avec le serveur MySQL. Notamment dans le cas de grosses requêtes qui sont assez « longues » à exécuter.

Si vous n'avez pas un site très fréquenté, il est clair qu'un tel système pour la sélection d'un message au hasard dans le livre d'or ne vous servira pratiquement à rien (juste à savoir faire marcher un système de cache, ce qui n'est pas inutile pour l'avenir).
Si en revanche votre site reçoit beaucoup de visiteurs, la moindre requête économisée peut être bénéfique.

Q.C.M.

Le premier QCM de ce cours vous est offert en libre accès.
Pour accéder aux suivants

À quoi sert un système de cache ?

- ◯ À établir un historique des messages d'un livre d'or.
- ◯ À accélérer le chargement des pages en allégeant le serveur en requêtes SQL.
- ◯ À cacher certaines parties du site.

Quelle option permet de créer un fichier s'il n'existe pas déjà, de le vider, et enfin d'y écrire ?

- ◯ r.
- ◯ w+.
- ◯ w.
- ◯ r+.

Que va faire ce code ?

Code : PHP

```php
<?php
// On ouvre le fichier
fopen('monfichier.txt', 'w+');
// On écrit dans le fichier
fwrite($monfichier, 'Hello world !');
// On ferme le fichier
fclose($monfichier);
?>
```

- ◯ Il plante.
- ◯ Rien.
- ◯ Il écrit « Hello world ! » dans le fichier monfichier.txt.

Correction !

Statistiques de réponses au QCM

Vous venez de créer un système de cache !

Cette méthode est assez simple, vous avez dû le remarquer. Il se peut que plus tard, vous conceviez un système plus poussé ; et donc, normalement, plus performant.

Faites-en bon usage.

En cas de problème, envoyez-moi un MP, ou mieux : contactez-moi par MSN.

Ce tutoriel a été corrigé par les zCorrecteurs.