TD5 : PHP « avancé »
V3.1.1

Cette œuvre de Mickaël Martin Nevot est mise à disposition sous licence Creative Commons Attribution - Utilisation non commerciale - Partage dans les mêmes conditions.

Document en ligne : www.mickael-martin-nevot.com

1 Généralités

Vous visualiserez systématiquement votre travail dans différents navigateurs Web.

N'oubliez pas de faire des recherches sur le Web à chaque fois que cela est nécessaire en prenant soin de vérifier que les informations trouvées soient correctes.

Vous trouverez la boîte à outils ainsi que l'ensemble des documents nécessaires à la réalisation de ce TD sur le site Web de l'enseignant.

2 Système de cache en PHP

2.1 Présentation

Le principe du cache est de mettre en mémoire dans un fichier sur le serveur le code source généré d'une page Web. Puis, lorsqu'un internaute demande l'affichage de la page concernée, le système de cache vérifie, avant toute chose, si le contenu précédemment généré est toujours d'actualité. Si tel est le cas, le système de cache économisera l'envoi des requêtes MySQL et les algorithmes potentiellement gourmands en ressources : le résultat est à la place récupéré depuis le fichier du cache.

Si un site Web a de nombreux visiteurs, l'économie en ressources pour le serveur est conséquent et la vitesse de navigation accélérée grâce à un système de cache.

2.2 Classe Cache

Cette classe contient :

- les variables d'instance : `$page`, `$expiration` et `$buffer` ;
- un constructeur avec un paramètre `$page` ;
- les accesseurs et mutateurs associés à ces variables d'instance ;
- une méthode `cacheView()` qui, si la page en cache existe et a été modifiée il y a moins d'une heure, lit un fichier et l'envoie dans le buffer de sortie avec la fonction `readfile($file)`, avant de terminer le script ;

(cc) BY-NC-SA

- une méthode `startBuffer()` qui enclenche une temporisation de sortie avec la fonction `ob_start()` ;
- une méthode `endBuffer()` qui enregistre le contenu du tampon de sortie dans `$buffer`, détruit les données du tampon de sortie, écrit le contenu de `$buffer` dans la page du cache avant d'afficher `$buffer`, en utilisant respectivement les fonctions `ob_get_clean()` et `file_put_contents(…)`.

2.3 Utilisation du système de cache

Utilisez le système de cache :

```
< ?php
    require 'class/cache.php';

    // On instancie un nouvel objet cache et on lui passe en paramètre
    // le nom de la page. Ce nom doit être unique pour chaque page car il va
    // générer un fichier du même nom.
    $cache = new Cache('index');

    // Si le cache est à jour, la méthode cacheView() l'affiche et le reste du
    // code est ignoré.
    $cache->cacheView();

    // La méthode startBuffer(), enregistre tout ce qui suit en mémoire tampon.
    $cache->startBuffer();

    // Ici on affiche le code de la page, les requêtes SQL, etc.
    echo 'Le corps de ma page';

    // La page est finie, on enregistre tout le contenu qu'elle génère.
    $cache->endBuffer();

    echo 'Version sans cache !';
?>
```

3 Fil d'Ariane

3.1 Présentation

Un fil d'Ariane sur un site Web est une aide à la navigation sous forme de signalisation de la localisation d'un internaute dans une page Web.

3.2 Mise en place du fil d'Ariane

Éditez un fichier `breadcrumbs.php`.

À l'intérieur, grâce à `$_SERVER['PHP_SELF']` et aux fonctions `explode($delimiter, $str)` et `count($arr)` (et éventuellement de la fonction `str_replace($search, $replace, $str)`), mettez en place un fil d'Ariane en HTML en utilisant les balises `<a></a>` et `<span></span>` (par exemple, utilisez > comme séparateur d'arborescence).

3.3 Utilisation du fil d'Ariane

Intégrez le fil d'Ariane dans différentes pages Web existantes ; constatez le résultat.

(cc) BY-NC-SA

4 Fichier .htaccess (Apache HTTP Server)

4.1 Présentation

Les fichiers `.htaccess` (écriture et casse très précises) sont des fichiers de configuration (permettant de surcharger certains éléments de configuration) **d'Apache HTTP Server**. Ils doivent être positionnés dans les répertoires du site Web, pas dans le répertoire de configuration d'Apache HTTP Server. La portée de leur configuration est limitée au contenu du répertoire (et des sous-répertoires) où ils se trouvent.

Cette particularité apporte deux principaux avantages :

- leur gestion peut être déléguée à des utilisateurs n'ayant pas le droit de gérer le serveur HTTP lui-même ;
- les modifications sont prises en compte sans qu'il soit nécessaire de redémarrer le serveur HTTP.

L'utilisation des fichiers `.htaccess` a un coût en matière de performances car le serveur HTTP doit vérifier la présence d'un tel fichier avant de traiter chaque requête. L'administrateur d'un tel serveur restreint donc en général l'utilisation des fichiers `.htaccess` à quelques directives particulières grâce à la directive `AllowOverride`.

Les fichiers `.htaccess` sont notamment utilisés pour configurer des droits d'accès, des redirections d'URL, des messages d'erreur personnalisés, etc.

4.2 Exemple de mise en place

Voici un exemple de fichier `.htaccess` :

```
# Fichier(s) par défaut (si le premier n'existe pas, on prend le 2e, etc.).
DirectoryIndex index.html index.htm /error/404.html
# Chemin du fichier .htpasswd.
AuthUserFile /path/.htpasswd
# Chemin pour les groupes.
#AuthGroupFile /dev/null
AuthGroupFile /path/.htgroup
# Message d'authentification.
AuthName "Authenticated access required for this directory"
# Type d'authentification (Basic signifie que l'accès est réservé aux
# utilisateurs listés dans le fichier spécifié par la directive AuthUserFile)
AuthType Basic
# Indique que l'on autorise uniquement les utilisateurs authentifiés.
Require valid-user
```

L'authentification `Basic` est, en principe, toujours disponible chez tous les hébergeurs. Il existe d'autres types d'authentification beaucoup plus sûrs, utilisés notamment en matière de sécurité bancaire ou pour la lecture des cartes d'identités électroniques, mais ces authentifications nécessitent des modules spécifiques.

4.3 Fichier .htpasswd

Chaque ligne d'un fichier `.htpasswd` doit contenir le login suivi de « : » puis l'empreinte du mot de passe. Le `$` indique que le mot de passe est crypté (il est aussi possible de le mettre en clair, même si cela est déconseillé). Ensuite vient le sel (ou *salt*), puis le mot de passe crypté (après le

(cc) BY-NC-SA

second `$`). Pour crypter le mot de passe, vous pouvez utiliser la commande `htpasswd` (ou la fonction PHP `crypt($str, …)`).

Un fichier `.htpasswd` peut donc ressembler à ceci :

```
user1:$apr1$TVFhC/..$rRph2WN9n1DeW6Cs89So2.
user2:$apr1$7ALHn/..$aV8IuW3jqdQWaStyX2Izg.
…
```

4.4 Fichier .htgroup

Les groupes d'utilisateurs sont définis dans le fichier `.htgroup` de la façon suivante :

```
group1:user1 user2 user3
group2:user4
```

4.5 Utilisation de fichiers .htaccess

Configurez et placez plusieurs fichiers `.htaccess` (avec les fichiers `.htpasswd` et `.htgroup` correspondants) dans différents répertoires Web ; constatez le résultat.