Web (Persistance)

Andrea G. B. Tettamanzi
Université de Nice Sophia Antipolis
Département Informatique
andrea.tettamanzi@unice.fr

CM - Séance 5

Lecture / écriture de fichiers en PHP : persistance sur fichier

Plan

- Manipulation des fichiers en PHP
- Entrée / Sortie avec les fichiers
- Fichiers et persistance
- Formulaires
- *Upload* (téléchargement) de fichiers

Pourquoi utiliser des fichiers en PHP ?

- Savegarder et récupérer des données sur un disque dur
  - Persistance au-delà d'une session
  - Partage (ou pas)
    - Configuration du serveur HTTP (ex., Apache : allow, deny)
    - Sauvegarde en dehors du Web
- Souplesse de programmation
  - Aucun format imposé
- Désavantage : code « bas niveau »

Manipulation des fichiers

- Équivalent des opérations sur les fichiers fournies par les systèmes d'exploitation
  - glob($pattern) : recherche de chemins vérifiant $pattern
  - scandir($dirname) : retourne tableau de fichiers
  - is_dir($fname) – $fname est-il un dossier ?
  - is_file($fname), is_executable($fname), is_link($fname)
  - is_readable($fname), is_writable($fname)
  - mkdir($nom [, $mode]) – créer dossier avec droits d'accès
  - rmdir($dir), rename($vieux, $nouv), touch($f), unlink($f)
  - filesize($fname)
- Attention : les résultats de certaines fonctions sont mis en cache

Informations sur un fichier

- Fonctions : **array** stat($fname) / **array** fstat(**resource** $handle)

Num	**Assoc**	**Description**
0	dev	device number
1	ino	inode number *
2	mode	inode protection mode
3	nlink	number of links
4	uid	userid of owner *
5	gid	groupid of owner *
6	rdev	device type, if inode device
7	size	size in bytes
8	atime	time of last access (Unix timestamp)
9	mtime	time of last modification (Unix timestamp)
10	ctime	time of last inode change (Unix timestamp)
11	blksize	blocksize of filesystem IO **
12	blocks	number of 512-byte blocks allocated **

Entrée / Sortie avec les fichiers

- Les fonctions fournies par PHP sont inspirées à la « libc »
- Principe de fonctionnement
  - On ouvre un fichier et on obtient une « ressource » (handle)
  - On passe cette ressource aux différentes fonctions
  - On peut acquérir un verrou en lecture ou écriture
  - On doit lever le verrou éventuel
  - On ferme le fichier après utilisation
- En général, les fonctions renvoient un résultat qui doit être testé pour vérifier si une erreur a eu lieu

Ouverture d'un fichier

- $ressource = fopen ($filename, $mode)
  - crée une ressource nommée, spécifiée par le paramètre filename, sous la forme d'un flux
  - URL possible
  - Retourne faux en cas de problème (et warning)
  - Mode : ‘r’, ‘r+’, ‘w’, ‘w+’, ‘a’, ‘a+’, ‘x’, ‘x+’
    - + : signifie : lecture et écriture
    - r : lecture à partir du début du fichier
    - w : écriture (et écrasement du fichier)
    - a : concaténation (ajout à la fin)
    - Pour w, a : si le fichier n’existe pas, on le crée
    - x : en écriture seule. Si le fichier existe, retourne faux

Navigation dans un fichier

- feof ($ressource) – Fin du fichier ?
  - Attention à ne pas se tromper de $ressource... boucle infinie…
- fseek($file, $position) – Positionne le pointeur à $position octets
  - Ne fonctionne pas si le fichier est ouvert en "a" ou "a+"
  - Taille d'un caractère : 1 ou 2 ou 4 octets (dépend du système, de l'encodage et du caractère)
    - 'a' en utf-8 prend un octet, 'ñ' en prend deux
- ftell($file) – Retourne la position du pointeur en octet
- rewind($file) – Replace le pointeur au début
- fclose ($ressource) – Ferme la ressource
  - Retourne vrai en cas de succès, faux sinon

Lecture / Écriture – Fonctions de base

- $msg = fread ($ressource, $length)
  - lit jusqu'à $length octets dans le fichier référencé par $ ressource.
  - La lecture s'arrête lorsque length octets ont été lus ou que l'on a atteint la fin du fichier
  - Renvoie la chaîne lue ou FALSE si une erreur survient.
- fwrite ($ressource, $chaine, $length) // $length optionnel
  - écrit le contenu de la chaîne $chaine dans le fichier pointé par $ressource.
  - Si la longueur $length est fournie, l'écriture s'arrêtera après $length octets ou à la fin de la chaîne (le premier des deux).
  - renvoie le nombre d'octets écrits ou FALSE en cas d'erreur.

Lecture – fonctions fgetX

- fgets($ressource, $length) // $length optionnel depuis PHP 4.2.0
  - renvoie la chaîne lue jusqu'à la longueur $length - 1 octets depuis le pointeur de fichier $ressource, ou bien la fin du fichier, ou une nouvelle ligne (qui est inclue dans la valeur retournée). Si aucune longueur n'est fournie, la longueur par défaut est de 1 ko ou 1024 octets.
  - fgets = LECTURE D'UNE LIGNE
- fgetss ($ressource, $length, $tag_ok)
  - $length, $tag_ok optionnel ($length depuis PHP 5)
  - Idem que fgets, mais en supprimant les tags HTML, sauf ceux dans $tag_ok
- fgetc ($ressource) – lecture d'un seul caractère (FALSE = EOF)

Lecture – fonctions haut niveau

- file($fname) – lecture d'un fichier en un tableau
  - Renvoie un tableau de string, 1 ligne = une case
  - Fin de ligne présent
    - rtrim($str) : enlève les « espaces » à la fin de $str
  - url possible
- file_get_contents($fname [, bool $use_include_path = false [, resource $context [, int $offset = -1 [, int $maxlen]]]])
  - Idem file, mais le résultat est dans une chaine
  - Possibilité de préciser une sous partie (par des octets)
  - Utilisation possible de include_path de php.ini pour rechercher le fichier

Écriture

- fput = fwrite
- fseek + fwrite = écraser ce qu'il y avait
- Réécrire la fin pour ajouter
- file_put_contents(string $filename , mixed $data [, int $flags = 0 [, resource $context]])
  - Pour écrire dans un fichier
  - $data : string ou tableau (ou stream resource)
  - $flags : FILE_USE_INCLUDE_PATH ou FILE_APPEND ou LOCK_EX
- fflush($f) – forcer synchronisation cache (mémoire tampon)

Verrou en écriture / lecture

- flock ($ressource, $operation)
  - $operation est une des valeurs suivantes :
    - Acquisition d'un verrou en lecture : operation = LOCK_SH
    - Acquisition d'un verrou exclusif en écriture : operation = LOCK_EX
    - Libération d'un verrou partagé ou exclusif, operation = LOCK_UN
  - Si vous voulez que flock( ) ne se bloque pas durant le verrouillage, ajoutez (&) LOCK_NB à operation.
  - Cette fonction retourne TRUE en cas de succès, FALSE en cas d'échec.
  - Le verrou doit être levé (par programmation)

Persistance

- Mécanisme général de sauvegarde et restauration de
  - Données
  - État de la computation
- Faire en sorte qu'un programme puisse se terminer sans que ses données et son état d'exécution ne soient perdus
- Sujet très important dans la programmation Web
  - Les protocoles du Web ne prévoient pas d'état !
- Sauvegarde
  - Localement dans un fichier
  - Sur un serveur distant
    - Ex., un serveur de bases de données relationnelles

Persistance et fichiers - Problèmes

- Code « bas niveau »
  - Refaire toujours les mêmes morceaux de code
  - Pas de structure « commune », non partageable
- Cas du CSV (comma separeted values)
  - format d'export textuel des tableurs
  - $chaine = fgets($fichier, 4096) – lire une ligne;
  - $donnees = explode(";", $chaine) – séparer les champs ;
- Manque de sémantiques
  - Ordre des colonnes
  - Contenu des colonnes

Persistance et fichiers - Solutions

- Faire une librairie pour un format
  - Un format = adaptation à un « métier »
  - Ex: je peux faire une lib pour lire des « personnes »
    - Recherche de titre de colonne (sinon valeurs par défauts)
  - Mais elle ne pourra pas lire des « stages »...
- Faire une libraire / un système pour n'importe quel « format »
  - soit dans des fichiers text (xml)
  - soit dans un système de BD
    - utilise des fichiers aussi, mais gérés par un SGBD !

Formulaires

- Balises HTML
  - FORM action= URL method=GET | POST …
  - LABEL, FIELDSET, LEGEND
  - INPUT type=…
  - BUTTON type=submit | button | reset
  - SELECT, OPTGROUP, OPTION
  - TEXTAREA
- Envoi d'un formulaire :
  - GET : les données sont ajoutées à l'URL comme paramètres
  - POST : les données sont envoyées dans le corps du fichier
  - Utiliser POST si le formulaire provoque des effets secondaires

Réception d'un formulaire

- Valeur(s) accessible(s) par des variables « superglobales » :
  - tableaux associatifs : index est l'attribut « name » de l'input
    - $_POST / $_GET – données du POST ou du GET
    - $_REQUEST = $GET + $POST + $COOKIE
- Un peu de sécurité (attention aux attaques!)
  - trim(htmlspecialchars(addslashes(…)))
  - str_replace pour remplacer des caractères « spéciaux »
  - Tests complémentaires...
- Les valeurs peuvent être des tableaux (si le name de l'input est du style nom[])

Téléchargement d'un fichier

- Méthode post
  - Le champ caché MAX_FILE_SIZE (mesuré en octets) doit précéder le champ input de type file et sa valeur représente la taille maximale acceptée du fichier.
- Variable globale $_FILES ('userfile‘ est le nom donné en HTML à la balise input)
  - $_FILES['userfile']['name'] : nom original du fichier
  - $_FILES['userfile']['type'] : type MIME, ex. : "image/gif"
  - $_FILES['userfile']['size'] : taille en octets
  - $_FILES['userfile']['tmp_name'] : nom temporaire du fichier
  - $_FILES['userfile']['error'] : code d'erreur (> PHP 4.2.0)

Téléchargement d'un fichier

- is_uploaded_file ($filename)
  - très utile pour vous assurer qu'un utilisateur n'essaie pas d'accéder intentionnellement à un fichier auquel il n'a pas droit (comme /etc/passwd).
  - Utiliser $_FILES['userfile']['tmp_name'] comme $filename !
- move_uploaded_file ($filename, $destination)
  - vérifie que le fichier $filename est un fichier téléchargé par HTTP POST. Si le fichier est valide, il est déplacé jusqu'à destination et TRUE est renvoyé
  - Si $filename n'est pas valide, rien ne se passe → FALSE
  - Si $filename ne peut pas être déplacé → FALSE.
    De plus, une alerte sera affichée.

Emploi côté client (HTML)

```
<form action="acceptefichiers.php" method="post"
 enctype="multipart/form-data">
<input type="submit" value="Envoi!" /> <br />
<div id="fichiers">
  <!-- MAX_FILE_SIZE doit précéder le champ input
de type file -->
  <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
  Fichier &agrave; t&eacute;l&eacute;charger :
  <input type="file" name="fichiers[]" />
</div>
</form>
```

Traitement côté serveur (PHP)

```
foreach($_FILES['fichiers']['tmp_name'] as $key => $tmp_file)
{
  $uploadfile = $uploaddir .
    basename($_FILES['fichiers']['name'][$key]);
  if(move_uploaded_file($tmp_file, $uploadfile))
  {
    // telechargement ok,
    // placement du fichier à l'endroit voulu
  }
  else
  {
    // échec dans le téléchargement
  }
}
```

Merci de votre attention