Programmation Web Serveur

Fonctions et fichiers

D'après les cours de Philippe Renevier

Fabien Givors

Université de Nice Sophia Antipolis

Département Informatique

fabien.givors@unice.fr

Comment écrire nos propres fonctions en php

FONCTION

PARTAGE DE CODE

- Exemple de code répété dans le tp02

```
$nomarticle = explode("/",
$fichiers[$indice]);
$nomarticle =
$nomarticle[count($nomarticle)-1];
```

- Besoin de mettre en commun...

FONCTIONS

- Définition : Les fonctions en PHP doivent être définies avant leur appel. Le nom d'une fonction ne commence pas par '$', et n'est pas sensible à l'utilisation des majuscules/minuscules. Voici la syntaxe :

```
function nom($chemin)
{
  $fichier = explode("/",$chemin);
  $fichier = $fichier[count($fichier)-1];

  return $fichier;
}
```

- Toutes les fonctions en PHP ont une portée globale
- **Les variables à l'intérieur des fonctions ont une portée locale**
- La récursivité est possible

FONCTIONS : PASSAGE DES ARGUMENTS

- par valeur :
  - Les arguments sont passés généralement *par valeur*,
  - ce qui signifie qu'une copie des variables est faite au moment de l'appel de la fonction,
  - et que les éventuelles modifications faites dans le corps de la fonction sur les arguments n'ont qu'un effet local.
  - i.e., une fois la fonction terminée, les modifications sont « oubliées »
- par adresse (non abordé ici)

FONCTIONS : PASSAGE DES ARGUMENTS

- Valeurs par défaut pour un ou plusieurs arguments d'une fonction.
  - La valeur par défaut d'un argument doit obligatoirement être une constante, et ne peut être ni une variable, ni un membre de classe, ni un appel de fonction.
  - Il est à noter que si vous utilisez des arguments avec valeur par défaut avec d'autres sans valeur par défaut, ceux avec une valeur par défaut doivent être à la fin.

```
function Connexion ($pNom, $pMotPass, $pBase =
  "clubvideo", $pServeur = "toto")
{  // Ici le code de la fonction }
// on peut donc appeler cette fonction sans citer les
  deux derniers arguments
$connexion1 = Connexion ("dupond", "passdupond");
$connexion2 = Connexion ("dupont", "passdupont",
  "Films");
```

FONCTIONS

- les fonctions sont à nombre d'arguments variable.
  - Possible avec les fonctions suivantes :
    - `func_num_args()` [pas de paramètre, retourne le nombre d'arguments passés à la fonction],
    - `func_get_arg()` [un paremètre i : retourne le (i+1)ième élément de la liste des arguments]
    - `et func_get_args()` [pas de paramètre, retourne les arguments d'une fonction sous forme de tableau]

- Fonctions et variables : PHP propose trois types de variables (la même terminologie du C) :
  - Variables automatiques (pas de portée à l'extérieur)
  - *Variables statiques (persistantes entre les appels)*
  - *Variables globales (mot clef « global »)*

STRUCTURATION DU CODE

- `require()` et `include()` incluent et exécutent un fichier PHP.
  - La commande `require()` se remplace elle-même par le contenu du fichier spécifié
  - `require()` et `include()` sont identiques, sauf dans leur façon de gérer les erreurs. `include()` produite une Alerte (warning) tandis que `require()` génère une erreur fatale. Notamment lorsque le fichier manque.
- `require_once()` et `include_once()`
  - La principale différence est qu'avec `require_once()`, vous êtes assurés que ce code ne sera ajouté qu'une seule fois, évitant de ce fait les redéfinitions de variables ou de fonctions, génératrices d'alertes.
- Structuration du code
- Partage de code

FONCTIONS

- Les valeurs de retour
  - Les valeurs sont renvoyées en utilisant une instruction de retour optionnelle. Tous les types de variables peuvent être renvoyés, tableaux et objets compris. Cela fait que la fonction finit son exécution immédiatement et passe le contrôle à la ligne appelante.
  - c.f. `return()`

```
function maFonction()
{
   // ….
      return $unResultat;
}


$resultat = maFonction();
```

RETURN

- Si appelée depuis une fonction, la **commande `return`** termine immédiatement la fonction, et retourne l'argument qui lui est passé.
- Si appelée depuis l'environnement global, l'exécution du script est interrompue.
  - Si le script courant était `include()` ou `require()`, alors le contrôle est rendu au script appelant, et la valeur retournée sera utilisée comme résultat de la fonction `include()`.
  - Si `return()` est appelée depuis le script principal, alors l'exécution du script s'arrête.

EXEMPLE : FORMATAGE DE DONNÉES

- Fonction affichant un prix

```
affichePrix($prix);
```

- Personnalisation de l'unité

```
affichePrix($prix, $unite);
```

- Démarche :
  - Identification du code répété (conception)
  - Factorisation
  - Amélioration (partagée)
  - Partage entre fichier

PARTAGE DE FONCTIONS ENTRE PAGES

- Création d'un fichier d'inclusion
  - Par convention `.inc` ou `.php`
- Inclusion de ce fichier dans d'autres
  - `require` ou `include`
- La fonction `affichePrix` alors disponible

Élément de Configuration d'apache

```
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</FilesMatch>

#idem pour <Files *.inc> ou <
Files *~>
```

Avoir accès aux fichiers sur le serveur web

FICHIERS

FICHIER(S) ET PERSISTANCE (1/2)

- Enregistrer les données sur un disque dur
  - Pérennité
  - Partage ou pas (configuration de apache) (sauvegarde en dehors du web)
- Souplesse de programmation
  - aucun format imposé
- Mais code « bas niveau »

MANIPULATION DES FICHIERS (1/3)

- **Équivalent des opérations sur les fichiers fournies par les systèmes d'exploitation**

- `glob ($pattern)` : recherche des chemins qui vérifient une expression `$pattern`

```
<?php
$files = glob("*.php"); // que les fichiers php du dossier courant
foreach ($files as $filename) {
    echo « <a hef="$filename >".basename($filename)."</a><br />\n";
}
?>
```

- `is_dir ($nomFichier)` -- Indique si le fichier est un dossier
- `is_file ($nomFichier)` -- Indique si le fichier est un fichier

- `is_readable ($nomFichier)` -- Indique si un fichier est autorisé en lecture
- `is_writable ($nomFichier)` -- Indique si un fichier est autorisé en écriture

  - *Les résultats de ces fonctions sont mis en cache*

MANIPULATION DES FICHIERS (2/3)

- `mkdir($chemin [, int mode])`
  - Crée un dossier `$chemin` (ou retourne faux)
  - Mode : droit d'accès : `0777` par avoir les droits d'écriture (c'est `nobody` ou `www-data` ou ... qui crée...)
- `rename($oldname, $newname)`
  - Renomme un fichier ou un dossier de nom `$oldname` en `$newname`
  - Retourne `true` si cela fonctionne, `false` en cas d'échec
  - Permet de déplacer un fichier
- `rmdir($dir)`
  - Efface le dossier `$dir`, s'il est vide et si le script à les droits
  - Retourne `true` si cela fonctionne, `false` en cas d'échec
- `touch($nomFichier)`
  - Modifie la date de modification et de dernier accès d'un fichier
  - Retourne vrai ou faux
- `unlink($nomFichier)` -- Efface un fichier [retourne vrai ou faux]

MANIPULATION DES FICHIERS (3/3)

- `filesize($nomFichier)`
  - Retourne la taille en octet du fichier
  - Ou `false` (+ error `E_WARNING`)
- `filemtime($nomFichier)`
  - Retourne la date de modification du fichier sous forme d'un nombre de secondes écoulées depuis le début 1970

Elément de Configuration

- Configuration de `php.ini` pour l'affichage des erreurs)
  - error_reporting
  - display_errors
  - Etc.

ENTRÉES / SORTIES AVEC LES FICHIERS

- **`file($nomFichier)`**
  - Lecture de tout un fichier sous forme d'un tableau de string, 1 ligne = une case
  - Fin de ligne présent
    - `rtrim($str)` : enlève les « espaces » à la fin de `$str`
  - url possible

ENTRÉES / SORTIES AVEC LES FICHIERS

- **`file_get_contents($nomFichier`** `[, bool $use_include_path = false [, resource $context [, int $offset = -1 [, int $maxlen ]]]])`
  - Idem file, mais le résultat est dans une chaine
  - Possibilité de préciser une sous partie (par des octets)
  - Utilisation possible de `include_path` *de* `php.ini` pour rechercher le fichier
  - `$context` : NULL (si on n'en utilise pas)
- **`file_put_contents(`**`string $nomFichier, mixed $data [, int $flags = 0 [, resource $context ]])`
  - Pour écrire dans un fichier
  - `$date` : string ou tableau (ou stream resource)
  - `$flags` : `FILE_USE_INCLUDE_PATH` ou `FILE_APPEND` ou `LOCK_EX`

FICHIER(S) ET PERSISTANCE (2/2)

- Mais code « bas niveau »
  - Refaire toujours les mêmes morceaux de code
  - Pas de structure « commune », non partageable
- Cas du CSV (comma separated values)
  - format d’export textuel de tableur
  - `$chaine = file($fichier);`
  - `$donnees = explode(";", $chaine[$i]) ;`
  - `// $donnees` => un tableau défini par les ;
- Manque de sémantiques
  - ordre des colonnes
  - Contenu des colonnes
- Base de données…