Programmation Web Avancée php

Compléments

Formulaire, Fichier

Détruire une session

```
// il faut détruire la session : copier coller de php.net...
// Initialisation de la session.
// Si vous utilisez un autre nom
// session_name("autrenom")

session_start();

// Détruit toutes les variables de session
$_SESSION = array();
// Si vous voulez détruire complètement la session, effacez également
// le cookie de session.
// cela détruira la session et pas seulement les données de session !

if (isset($_COOKIE[session_name()]))  {
    setcookie(session_name(), '', time()-42000, '/');
}
// Finalement, on détruit la session.
session_destroy();
```

Détection de mobile

- Principe : test sur le client web utilisé
  - Appelé user agent
  - $_SERVER['HTTP_USER_AGENT']…
  - $_SERVER['ALL_HTTP'] : concaténation de plusieurs, non standard
- Quasiment que des cas particuliers…

- Exemple de valeur à partir d'un Android

$_SERVER['HTTP_X_WAP_PROFILE'] = http://www.htcmms.com.tw/Android/Common/Legend/ua-profile.xml

$_SERVER['HTTP_USER_AGENT'] = Mozilla/5.0 (Linux; U; Android 2.2; fr-fr; HTC Legend 3.14.163.1 Build/FRF91) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1

$_SERVER['HTTP_ACCEPT'] = application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

Code php (à faire évoluer)

```
public function isMobile()
{
		// source : http://mobiforge.com/developing/story/lightweight-device-detection-php?dm_switcher=true
		$mobile_browser = '0';
		$user_agent = strtolower($_SERVER['HTTP_USER_AGENT']);

		// test sur l'OS
		if (preg_match('/(up.browser|up.link|mmp|symbian|smartphone|midp|wap|phone|android)/i',
$user_agent)) {
			$mobile_browser++;
		}

		// test sur les types mimes acceptes en retour
		if ((strpos(strtolower($_SERVER['HTTP_ACCEPT']),'application/vnd.wap.xhtml+xml') > 0) or
((isset($_SERVER['HTTP_X_WAP_PROFILE']) or isset($_SERVER['HTTP_PROFILE'])))) {
			$mobile_browser++;
		}

		/* … suite sur le transparent suivant … */
```

Code php (à faire évoluer)

```
/* … suite  du precedent … */

// test  sur le debut du clien web
$mobile_ua = strtolower(substr($_SERVER['HTTP_USER_AGENT'], 0, 4));
$mobile_agents = array(
                'w3c ','acs-','alav','alca','amoi','audi','avan','benq','bird','blac',
                'blaz','brew','cell','cldc','cmd-','dang','doco','eric','hipt','inno',
                'ipaq','java','jigs','kddi','keji','leno','lg-c','lg-d','lg-g','lge-',
                'maui','maxo','midp','mits','mmef','mobi','mot-','moto','mwbp','nec-',
                'newt','noki','oper','palm','pana','pant','phil','play','port','prox',
                'qwap','sage','sams','sany','sch-','sec-','send','seri','sgh-','shar',
                'sie-','siem','smal','smar','sony','sph-','symb','t-mo','teli','tim-',
                'tosh','tsm-','upg1','upsi','vk-v','voda','wap-','wapa','wapi','wapp',
                'wapr','webc','winw','winw','xda ','xda-');

if (in_array($mobile_ua,$mobile_agents)) {
                $mobile_browser++;
}

/* … suite sur le transparent suivant … */
```

Code php (à faire évoluer)

```
/* … suite  du precedent … */

//  cas particulier d'opera pour mobile
if (strpos(strtolower($_SERVER['ALL_HTTP']),'OperaMini') > 0) {
                $mobile_browser++;
}

// cas particulier pour windows et Windows Mobile…
if ((strpos($user_agent,'windows') > 0) && (! strpos($user_agent,'windows ce'))
                                        && (! strpos($user_agent,'Windows  Mobile') ) {
                $mobile_browser = 0;
}

return ($mobile_browser > 0);
}
```

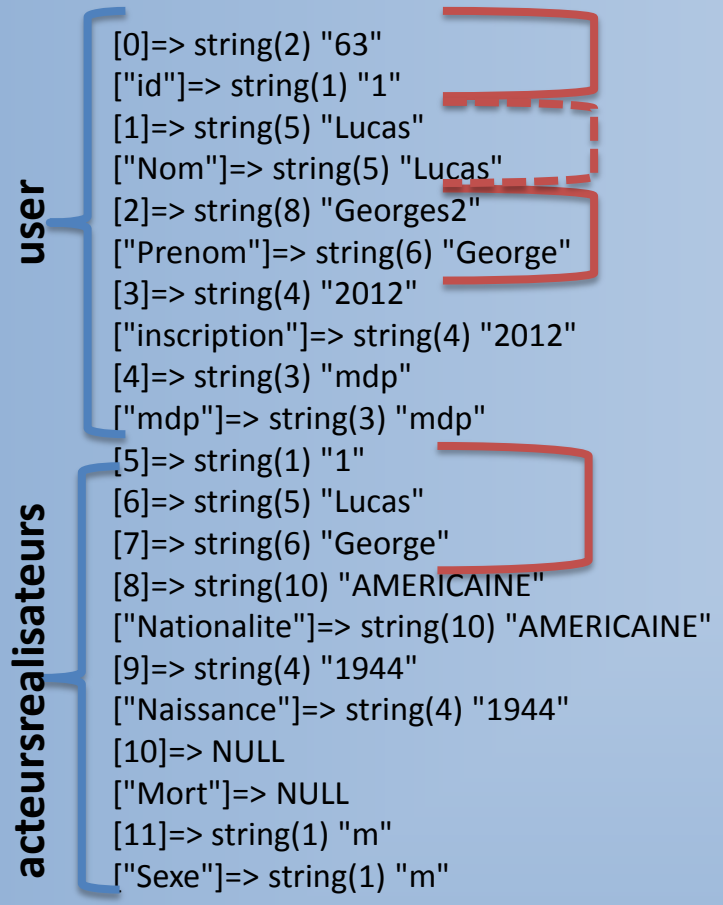
Jointure et « doublons »

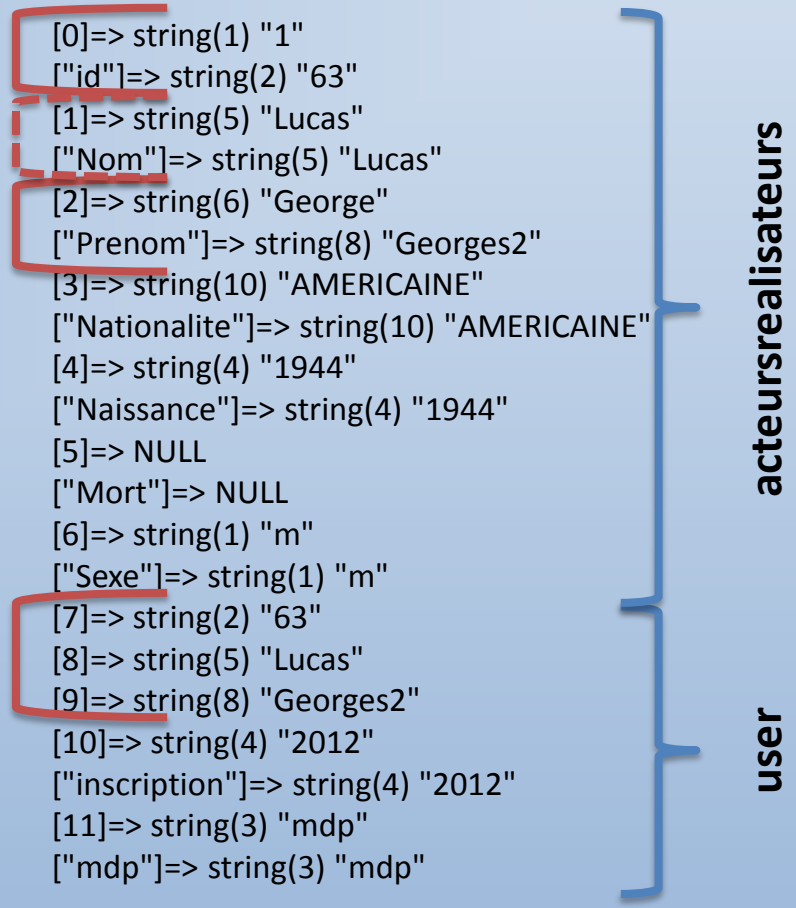
- Indexation numérique
  - Tout les champs y sont
  - Dans l'ordre de la jointure, dans l'ordre des tabes
- Indexation associative
  - Écrasement par la dernière valeur (dernière table de la jointure)

Jointure et « doublons » : illustration, code php

```
// definition d'une constante
define("CONNEXION", mysql_connect("localhost", "root", ""));
mysql_select_db("ihm2011-dev", constant("CONNEXION"));

// les deux requetes
$requeteSens1 = "SELECT * FROM user JOIN acteursrealisateurs ON acteursrealisateurs.nom=
user.nom";
$requeteSens2 = "SELECT * FROM acteursrealisateurs JOIN user ON acteursrealisateurs.nom=
user.nom";

/// la requete est executee
$repSens1 = mysql_query($requeteSens1,  constant("CONNEXION"));
$repSens2 = mysql_query($requeteSens2,  constant("CONNEXION"));

// "affichage" de la 1er reponse dans le sens 1
$ligne = mysql_fetch_array($repSens1) ;
echo "<pre>\n";
var_dump($ligne);
echo "</pre>\n";

// "affichage" de la 1er reponse dans le sens 2
$ligne = mysql_fetch_array($repSens2) ;
echo "<pre>\n";
var_dump($ligne);
echo "</pre>\n";
```

Jointure et « doublons » : illustration, résultat





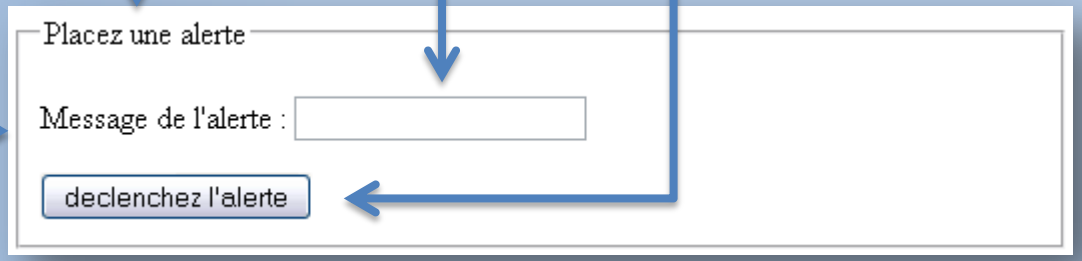
Formulaires et Php

Le côté html…

Le côté Php

Exemple: côté client

```
<form action="page.php" method="post">
<fieldset>
        <legend>Placez une alerte</legend>
        <p>Message de l'alerte : <input type="text" name="alerte" /></p>
        <input type="submit" name="setalerte" value="declenchez l'alerte"
/>
</fieldset>
</form>
```



Côté Serveur

```
<form action="page.php" method="post">
<fieldset>
        <legend>Placez une alerte</legend>
        <p>Message de l'alerte : <input type="text" name="alerte" /></p>
        <input type="submit" name="setalerte" value="declenchez l'alerte"
/>
</fieldset>
</form>
```

```
if (isset($_POST["setalerte"]))
        {
        $fichier = fopen("alerte.txt", "w");
        $alerte =
rawurlencode(trim($_POST["alerte"]));
        fwrite($fichier, $alerte);
        fclose($fichier);
        }
```

Balises de formulaire : form

- Contient des éléments de contrôle de formulaire (bouton, champs, etc.)
  - « block » (sauf form) ou script
  - Attributs
    - action (uri)
    - method ("get" ou "post")
      - get : envoi dans l'url des paires key/value : ?toto=val&titi=val2&…
      - post : envoi
    - enctype (pour une méthode "post")
      - Par défaut : **application/x-www-form-urlencoded -** encodage : espace devient + et les autres non alphanumériques %HH et les retours à la ligne : "CR LF" (i.e., `%0D%0A')
      - **multipart/form-data -** envoi en différentes parties (types à préciser à la source)
    - accept-charset (liste - , - d'encodage possible pour les caractères acceptés par le server)
    - Accept (liste - , - de types de contenu acceptés par le server)
    - events : onsubmit et onreset

Balises de formulaire : type de input

- text : champs d'entrée de texte.
- password : l'écho sont des '*'. sécutité pauvre.
- checkbox
- radio (radiobutton)
- submit : un bouton pour envoyer
- image : un bouton submit graphique. Attribut src donne l'URI de l'image. Utiliser l'attribut alt. Les coordonnées du clic sont passés au server sous la forme name.x et name.y
  - problème d'accéssibilité : navigateur non graphique, clic difficile, etc.
  - à remplacer par plusieurs boutons submit ou par des scripts côtés client.
- reset (bouton).
- button : bouton sans comportement prédéfini (script)
- hidden : champs caché (parfois utile pour passer une valeur masquée)
- file : sélection d'un fichier

Balises de formulaire : input

- Balise vide
- Attributs
  - type
  - name : nom de contrôle (très important)
  - value (valeur initiale ou libellé) : optionnel sauf pour radio et checkbox
  - size (en pixel sauf pour text et password où c'est un nombre de caractère)
  - maxlength : pour text ou password : nombre de caractères maximum
  - checked : pour radio et checkbox
  - src : pour image : la source (ne pas oublier alt)

Balises de formulaire : select

- select : menu
  - (optgroup | option)+
  - attributs
    - name : nom de contrôle
    - size (nombre) : nombre d'éléments visibles pour une scroll list
    - multiple (pas de valeur) : permet la sélection multiple
- option
  - #pcdata (texte)
  - attributs
    - selected : pour présélectionner l'élément
    - value (texte) : pour donner une valeur autre que le texte (#pcdata)
    - label (texte) : pour faire apparaître un autre nom (plus court) à la charge du navigateur !! (pas sûr que cela fonctionne !!)
- optgroup
  - regrouper les options : (option)+
  - attribut : label (texte) : libellé

Balises de formulaire : textarea

- Champs d'entrée sur plusieurs lignes
- textarea
  - #PCDATA : texte initiale
  - Attributs
    - name : nom de contrôle
    - cols : nombre de colonne
    - rows : nombre de ligne

Balises de formulaire : label

- Permet d'associer un texte à un élément de formulaire sans texte
  - inline
- Attaché par l'attribut for
  - Valeur = id d'un champ de contrôle

Balises de formulaire : navigation

- Transfert de focus
  - Souris
  - Clavier : tabulation ou touche raccourcie
- Tabulation : attribut tabindex (numéro)
  - a, area, button, input, object, select et textarea.
  - 1) Ceux qui ont l'attribut : de la plus petite à la plus grande valeur. Pas forcément consécutifs. En cas d'égalité, ordre d'apparition (flots de caractère)
  - 2) Ceux qui ne n'ont pas (impossible ou non donné) : ordre d'apparition
  - 3) Les désactivés (c.f. transparent suivant) ne participent pas.
- Touche raccourcie : attribut accesskey (="<un caractère>")
  - a, area, button, input, label, legend et textarea.
  - sous windows : besoin de la touche alt en plus…

Formulaires : réception en php

- Page qui reçoit le formulaire : attribut « action » du form
- Valeur(s) accessible(s) par :
  - tableau associatif : index est l'attribut « name » de l'input
    - **$_REQUEST**
    - **$_POST** ou **$_GET**
- Un peu de sécurité
  - 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 champs caché MAX_FILE_SIZE (mesuré en octets) doit précédé 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'] : Le nom original du fichier, tel que sur la machine du client web.
  - $_FILES['userfile']['type'] : Le type MIME du fichier, si le navigateur a fourni cette information. Par exemple, cela pourra être "image/gif".
  - $_FILES['userfile']['size'] : La taille, en octets, du fichier téléchargé.
  - $_FILES['userfile']['tmp_name'] : Le nom temporaire du fichier qui sera chargé sur la machine serveur.
  - $_FILES['userfile']['error'] : Le code d'erreur error code associé au téléchargement de fichier. Cet élément a été introduit en PHP 4.2.0

Les codes erreurs

- UPLOAD_ERR_OK
  - Valeur : 0. Aucune erreur, le téléchargement est correct.
- UPLOAD_ERR_INI_SIZE
  - Valeur : 1. Le fichier téléchargé excède la taille de upload_max_filesize, configuré dans le php.ini.
- UPLOAD_ERR_FORM_SIZE
  - Valeur : 2. Le fichier téléchargé excède la taille de MAX_FILE_SIZE, qui a été spécifiée dans le formulaire HTML.
- UPLOAD_ERR_PARTIAL
  - Valeur : 3. Le fichier n'a été que partiellement téléchargé.
- UPLOAD_ERR_NO_FILE
  - Valeur : 4. Aucun fichier n'a été téléchargé.
- UPLOAD_ERR_NO_TMP_DIR
  - Valeur : 6. Un dossier temporaire est manquant. Introduit en PHP 4.3.10 et PHP 5.0.3.
- UPLOAD_ERR_CANT_WRITE
  - Valeur : 7. Échec de l'écriture du fichier sur le disque. Introduit en PHP 5.1.0.
- Note : Ces constantes sont apparues en PHP 4.3.0.

Téléchargement d'un fichier

- is_uploaded_file ($filename)
  - retourne TRUE si le fichier filename a été téléchargé par HTTP POST. Cela est 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).
  - Pour un fonctionnement correct, la fonction is_uploaded_file( ) nécessite un argument comme $_FILES['userfile']['tmp_name']
  - le nom du fichier téléchargé sur la machine cliente $_FILES['userfile']['name'] ne fonctionne pas.
- move_uploaded_file ($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 retourne TRUE
  - Si filename n'est pas valide, rien ne se passe, et move_uploaded_file() retournera FALSE.
  - Si filename est un fichier téléchargé, mais que pour une raison quelconque, il ne peut être déplacé, rien ne se passe, et move_uploaded_file() retourne FALSE. De plus, une alerte sera affichée.

Exemple côté html

```
<form action="acceptefichiers.php" method="post"
   enctype="multipart/form-data">
   <div>
   <input type="submit" value="Envoi!" /> <br /> <div
   id="fichiers">
<!-- MAX_FILE_SIZE doit preceder le champs input de type
   file -->
   input type="hidden" name="MAX_FILE_SIZE" value="30000"
   />

   fichier &agrave; t&eacute;l&eacute;charger :
   type="file" name="fichiers[]" />
   </div>
</form>
```

Exemple côté php

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

Fichiers et BD

- Stockage d'une « url » (ou chemin local)
- Ecriture d'un BLOB
  - Binary large object
  - file_get_contents pour convertir les fichiers en String

Ajout d'une image dans une BD

```
/**
 * ajoute une image (affiche) du film.
 * Le type mime (image/png, image/gif, image/jpeg) est déterminé automatiquement
 * en fonction du nom de l'image (.png, .gif, .jpg ou .jpeg)
 * @param Data_Film $f le film
 * @param string image : le nom de fichier ou l'url
 * @return boolean vrai si cela a fonctionné
 */
public function addAfficheAuFilm(Data_Film $f, $image)   {
                $retour = false;

                $fid = $this->filmToId($f);

                $ext3 = substr($file, -4);
                $ext4 = substr($file, -5);
                $type = "image/png";
                if (($ext3 == ".jpg") || ($ext4 == ".jpeg")) $type = "image/jpg";
                else if ($ext3 == ".gif") $type = "image/jpg";

                $content = addslashes(file_get_contents($image));

               if ($content)   {
                $query = "insert into affiche (afficheId, film, image, type) values ('','$fid','$content', '$type')";
                $retour = mysql_query($query, $this->connexion);
               }

                return $retour;
}
```

Restitution d'une image dans une BD

```
/**
 * @author renevier-gonin
 * @package utilities
 *
 * @abstract pour fournir une image (src de img) stockée dans la bd à partir de son id et de son type (affiche ou portrait)
 * <code> <img alt="affiche de starwars 4" src="../utilities/Image.php?id=1" /> </code>
 * le src est donnée via la classe Data_Img
 * @see Data_Img
 */
/*  la connexion à la bd est requise...  */
require "../includes/connexion.inc";

if ( isset($_GET['id']) ) {
		$id = intval ($_GET['id']);
		$table = " afficheId, image, type FROM affiche WHERE afficheId = ";
		if (isset($_GET['type']))  {
			if ($_GET['type'] == "portrait") $table = " portraitId, image, type FROM portrait WHERE portraitId =";
		}
		$req = "SELECT $table  ".$id;
		$ret = mysql_query ($req) or die (mysql_error ());
		$col = mysql_fetch_row ($ret);
		if ( !$col[0] )   {  echo "Id d'image inconnu";  }
		else  {
			header ("Content-type: {$col[2]}");
			echo $col[1];
		}
}
else {    echo "Mauvais id d'image";   }
```

Php et la manipulation de fichier

D'abord des infos générales…

puis un exemple

Manipulation des fichiers

- filemtime ($filename)
  - Renvoie la date (un entier) de dernière modification du fichier
  - Utilisez date( ) sur le résultat
  - Le résultat est mis en cache
- fileperms ($filename)
  - Renvoie les permissions affectées à un fichier sous forme d'un entier (0777)
  - Le résultat est mis en cache
  - Il existe des fonctions pour « jouer » avec les permissions…
- filesize ($filename)
  - Renvoie la taille d'un fichier (en octet)
  - Comme le type entier de PHP est signé et que de nombreuses plates-formes utilisent des entiers de 32 bits, filesize( ) peut retourner des résultats étranges pour les fichiers de taille supérieure à 2 Go
  - Le résultat est mis en cache
- filetype -- Retourne le type de fichier
  - renvoie le type du fichier *filename*. Les réponses possibles sont : *fifo*, *char*, *dir*, *block*, *link*, *file* et *unknown*.
  - Le résultat est mis en cache

Manipulation des fichiers

- glob
  - Recherche des chemins qui vérifient un masque qui sont les mêmes que celles utilisées par le Shell en général

```
<?php
$files = glob("*.php");
foreach ($files as $filename) {
   echo "$filename occupe " . filesize($filename) . " octets\n";
}
?>
```

- is_dir ($filename) -- Indique si le fichier est un dossier
- is_executable ($filename) -- Indique si le fichier est exécutable
- is_file ($filename) -- Indique si le fichier est un fichier
- is_link ($filename) -- Indique si le fichier est un lien symbolique
- is_readable ($filename) -- Indique si un fichier est autorisé en lecture
- is_writable ($filename) -- Indique si un fichier est autorisé en écriture
  - Les résultats de ces fonctions sont mis en cache

Manipulation des fichiers

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

Manipulations des dossiers en php

- getcwd
  - Retourne le dossier de travail courant en cas de réussite ou **FALSE** en cas d'échec.
- chdir($dir)
  - **chdir( )** change le dossier courant de PHP en *directory*.
  - Cette fonction retourne **TRUE** en cas de succès, **FALSE** en cas d'échec.
- $ressource = opendir ($path)
  - opendir( ) retourne un pointeur sur un dossier qui pour être utilisé avec les fonctions closedir( ), readdir( ) et rewinddir( ).
  - Retourne la ressource de dossier en cas de succès ou **FALSE** en cas d'échec.
  - Génération de warning (@ pour masquer)
- closedir ($ressource)
  - closedir() ferme le pointeur de dossier $ressource.

Manipulations des dossiers en php

- readdir ($ressource)
  - readdir( ) retourne le nom (string) du fichier suivant dans le dossier identifié par $ressource. Les noms sont retournés dans l'ordre qu'ils sont enregistrés dans le système de fichiers
  - *$a === $b (*Identique) **TRUE** si *$a* est égal à *$b* et qu'ils sont de même type (introduit en PHP 4).
  - $a !== $b (Différent) **TRUE** si *$a* est différent de *$b* ou bien qu'ils ne sont pas du même type. (introduit en PHP 4)
- rewinddir ($ressource)
  - retourne à la première entrée du dossier identifiée par $ressource.
- scandir (string directory [, int sorting_order])
  - Retourne un tableau des fichiers en cas de succès ou FALSE en cas d'échec. Si directory n'est pas un dossier, alors une valeur booléenne FALSE est retournée et un WARNING est généré.
  - Par défaut, le tri est en ordre alphabétique. Si le paramètre optionnel *sorting_order* est utilisé (mis à 1), alors le tri sera en ordre alphabétique inverse.
- **c.f. glob**

Entrées / Sorties avec les fichiers

- $ressource = fopen ($filename, $mode)
  - crée une ressource nommée, spécifiée par le paramètre filename, sous la forme d'un flux
  - 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
    - Pour w, a : si le fichier n’existe pas, on le crée
    - x : création en lecture seule. Si le fichier exite, retourne faux

Entrées / Sorties avec les fichiers

- fclose ($ressource)
  - Ferme la ressource ouverte avec fopen.
  - Retourne vrai en cas de succès, faux sinon
- feof ($ressource)
  - feof( ) retourne TRUE si le pointeur handle est à la fin du fichier ou si une erreur survient, sinon, retourne FALSE.
  - Attention à ne pas se tromper de $ressource… boucle infinie…
- $msg = fread ($ressource, $length)
  - fread( ) 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
  - Retourne la chaîne lue ou FALSE si une erreur survient.
- fwrite ($ressource, $chaine, $length) // $length optionel
  - fwrite( ) 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).
  - fwrite() retourne le nombre d'octets écrits ou FALSE en cas d'erreur.

Entrées / Sorties avec les fichiers

- $ligne fgets($ressource, $length) // $length optionel depuis PHP 4.2.0
  - retourne la chaîne lue jusqu'à la longueur $length - 1 octet depuis le pointeur de fichier $ressource, ou bien la fin du fichier, ou une nouvelle ligne (qui est inclue dans la valeur retournée). Si aucune longueur n'est fournie, la longueur par défaut est de 1 ko ou 1024 octets.
- fgetss ($ressource, $length, $tag_ok)
  - $length, $tag_ok optionnel ($length depuis php 5)
  - Idem que fgets, mais en supprimant les tag html, sauf ceux dans $tag_ok
- fgetc ($ressource)
  - retourne une chaîne contenant un seul caractère, lu depuis le fichier pointé par handle.
  - fgetc( ) retourne FALSE à la fin du fichier

Entrées / Sorties avec les fichiers

- flock ($ressource, $operation)
  - flock() agit sur le fichier ($ressource qui doit avoir été ouvert au préalable).
  - 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 est également levé avec la fonction fclose() (qui est également automatiquement appelée lors de la fin du script).

Entrées / Sorties avec les fichiers

- $nb_octets_lus = readfile ($filename) -- Affiche un fichier (dans la page web)
- $chaine_lue = file_get_contents ($filename)
  - Comme readfile, sauf lit tout un fichier dans une chaîne
- $nb_octets_lus = fpassthru($ressource) -- Affiche le reste du fichier

- $result = fscanf ($ressource, $format)
  - Analyse un fichier en fonction d'un format (comme en C)
  - Le résultat peut être une chaîne ou un tableau
- fstat -- Lit les informations sur un fichier à partir d'un pointeur de fichier
- ftruncate -- Tronque un fichier

Entrées / Sorties avec les fichiers

- fseek($ressource, $offset [, $whence])
  - modifie le curseur de position dans le fichier $ressource. La nouvelle position mesurée en octets à partir du début du fichier est obtenue en additionnant la distance offset à la position $whence. Ce paramètre peut prendre les valeurs suivantes :
    - SEEK_SET - La position finale vaut offset octets.
    - SEEK_CUR - La position finale vaut la position courante ajoutée à offset octets.
    - SEEK_END - La position finale vaut la position courante par rapport à la fin du fichier, ajoutée de offset.
    - Si $whence n'est pas spécifiée, il vaut par défaut SEEK_SET.
  - **fseek( )** retourne 0 en cas de succès, et sinon -1. Notez que positionner le pointeur au-delà de la fin du fichier n'est pas une erreur.
- $position = ftell($ressource) -- Renvoie la position du pointeur du fichier
- rewind($ressource)
  - replace le pointeur du fichier $ressource au début.
  - Cette fonction retourne TRUE en cas de succès, FALSE en cas d'échec.
  - Si vous avez ouvert le fichier en mode d'ajout ("a" ou "a+"), toutes les données que vous écrirez dans ce fichier seront toujours ajoutées à la fin, sans se soucier de la position du pointeur de fichier.

Entrées / Sorties avec les fichiers

- $nb_octets_ecrit = file_put_contents($filename, $a_ecrire)
  - Ecrit une chaîne ou un tableau ($a_ecrire) dans un fichier de nom $filename
- fflush($ressource) -- Envoie tout le contenu généré dans un fichier
- tempnam ($dir, $prefix)
  - crée un fichier temporaire unique dans le dossier $dir. Si le dossier n'existe pas, tempnam( ) va générer un nom de fichier dans le dossier temporaire du système. Le nom sera préfixé par le paramètre $prefix.
  - tempnam ( ) retourne le nom du fichier temporaire ou FALSE en cas d'échec.
  - À effacer « manuellement »
- $ressource = tmpfile ( )
  - tmpfile() crée un fichier temporaire avec un nom unique, ouvert en écriture et lecture (w+), et retourne un pointeur de fichier, identique à ceux retournés par fopen(). Ce fichier sera automatiquement effacé lorsqu'il sera fermé (avec fclose()), ou lorsque le script sera terminé.

Lecture d'un fichier ligne par ligne

```
$fichierSource = fopen($filename, "r");

while (!feof($fichierSource)) {
    $buffer = fgets($fichierSource);
    // lecture jusqu'a la fin de la ligne

    /* … */
}

fclose($fichierSource);
```

Ecriture d'un fichier

```
// ouverture en concatenation (ecriture)
$fichierDestination = fopen($filename, "a");

/* instructution d'ecriture…  */
fwrite($fichierDestination, $chaineDeCaractere);

/* … */

fclose($fichierDestination);
```

Dom XML en php

Document Object Model

- Voir un document XML comme un arbre
- Parcours des nœuds, détails des attributs, etc.
- Présent dans quasiment tous les langages.

- http://www.w3.org/TR/DOM-Level-2-Core/expanded-toc.html

DTD : (ancienne) façon pour définir un document XML

- DTD (*Document Type Definition*) : vérifier qu'un document XML est conforme à une syntaxe donnée (modèle).
  - une grammaire
  - document valide par rapport à une DTD
- Une DTD définie de 2 façons :
  - sous forme interne (inclure la grammaire dans le document)
  - sous forme externe (fichier local ou URL contenant la grammaire)

DTD

- **Définition d'un élément suivant la syntaxe : <! ELEMENT Nom Modèle >**
- **Modèle**
  - **ANY : L'élément peut contenir tout type de données**
  - **EMPTY : L'élément ne contient pas de données spécifiques**
  - **#PCDATA : L'élément doit contenir une chaîne de caractères**
    - **Le mot clé *#PCDATA* doit nécessairement être écrit entre parenthèses, sinon risque d'obtenir une erreur du parseur.**

Opérateur	Signification	Exemple
+	**L'élément doit être présent au minimum une fois**	**A+**
*	**L'élément peut être présent plusieurs fois (ou aucune)**	**A***
?	**L'élément peut être optionnellement présent**	**A?**
|	**L'élément A ou l'élément B peuvent être présents**	**A|B**
,	**L'élément A doit être présent et suivi de l'élément B**	**A,B**
( )	**Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs**	**(A,B)+**

Exemple de DTD

- <!ELEMENT personne (nom,prenom,telephone),email? >
- <!ELEMENT nom (#PCDATA) >
- <!ELEMENT prenom (#PCDATA) >
- <!ELEMENT telephone (#PCDATA) >
- <!ELEMENT email (#PCDATA) >

- <personne>
  - <nom>Renevier-Gonin</nom>
  - <prenom>Philippe</prenom>
  - <telephone>04….</telephone>
  - <email>Philippe.Renevier@unice.fr</email>
- </personne>

DTD pour un chat

```
<!DOCTYPE chat [

<!ELEMENT chat (timestamp)*>
<!ELEMENT timestamp (number , line*)>
 <!ELEMENT line (user , hour , color? , text)>
<!ELEMENT user (#PCDATA)>
<!ELEMENT hour (#PCDATA)>
<!ELEMENT color (#PCDATA)>
 <!ELEMENT text (#PCDATA)>
<!ATTLIST chat date CDATA #IMPLIED>
 <!ATTLIST timestamp date CDATA #IMPLIED>
 <!ATTLIST line date CDATA #IMPLIED>
]>
```

```
<chat>
<timestamp>
    <number>38</number>
    <line>
        <user><![CDATA[Phil]]></user>
        <hour>11:01</hour>
        <text><![CDATA[bonjour]]></text>
    </line>
    <line>
        <user><![CDATA[Jey]]></user>
        <hour>08:47</hour>
        <text><![CDATA[Yopp]]></text>
    </line>
</timestamp>
</chat>
```

DTD

- Attribut : <! ATTLIST Elément Attribut Type >
- Type représente le type de donnée de l'attribut, il en existe trois:
  - littéral: une chaîne de caractères, mot clé *CDATA*
  - l'énumération: une liste de valeurs possibles pour limiter le choix de l'utilisateur. Syntaxe
    - <! ATTLIST Elément Attribut (Valeur1 | Valeur2 | ...) >
    - <! ATTLIST Elément Attribut (Valeur1 | Valeur2) "valeur par défaut" > (valeur par défaut entre guillemets)
  - atomique: identifiant unique, mot clé *ID*.
- Caractère obligatoire d'un attribut (optionnel) : le faire suivre d'un mot clé particulier :
  - #IMPLIED : optionnel
  - #REQUIRED : obligatoire
  - #FIXED : valeur par défaut (à préciser entre guillemets) sinon défini.

- <! ATTLIST disque IDdisk ID #REQUIRED type (K7|MiniDisc|Vinyl|CD) "CD" >

DTD

- Entités : déclarer un groupe d'éléments sous un nom afin de ne pas avoir à réécrire ces derniers plusieurs fois dans la DTD
  - une meilleure lisibilité
  - un contrôle accru sur le contenu
  - une plus grande facilité de mise à jour
- On distingue plusieurs types d'entités dans XML :
  - les entités générales
    - <!ENTITY nom_de_l_entite "Contenu de l'entite">
    - <!ENTITY site "http://deptinfo.unice.fr/~renevier/L3">
    - usage : <site>&site;</site>
  - les entités paramètres
    - <!ENTITY % nom_de_l_entite definition>
  - les entités caractères
    - & : & < : < > : > ' : ‘ " : «
    - <!ENTITY nom_de_l_entite "&#xCODEHEXA;">
    - <!ENTITY ccedille "ç">

Génération de XML

- Une page PHP peut retourner n'importe quel type de document MIME
  - text/html par défaut
  - Mais aussi des images, du xml, n'importe quel type
- Grâce à la fonction header qui va modifier l'entête de la réponse HTTP
  - header('Content-type: text/xml');
  - Puis il faut faire générer le xml (echo de balises xml, comme pour du html)

- Autre application : les images dans les bd
  - Stockées dans des blob
  - Restituée telles quelles
  - exemple/afficheImage.ph

```
if ( isset($_GET['id']) ) {
   $id = intval ($_GET['id']);
   include ("connexion.inc");
   $req = "SELECT afficheId, image FROM affiche WHERE afficheId = $id";
   $ret = mysql_query ($req) or die (mysql_error ());
   $col = mysql_fetch_row ($ret);
   if ( !$col[0] )
                    {
                    echo "Id d'image inconnu";
                    }
   else
                    {
                    header ("Content-type: img/png");
                    echo $col[1];
                    }
}
```

Structuration DOM : Dom Document

- application de DTD
- Un Document :
  - DOMDocument
  - C'est aussi un nœud (DOMNode) => parcours selon un nœud possible
  - De nombreuses propriétés et de nombreuses méthodes
- création

```
// Create a new DOM Document to hold our document structure
$xml = new DOMDocument();
$xml->load ("fichier.xml");
```

- mixed load (string $filename [, int $options = 0])
- bool loadHTML (string $source)
- bool loadHTMLFile (string $filename)
- mixed loadXML (string $source [, int $options = 0])

Structuration DOM : Dom Document

- Sauvegarde

int save (string $filename [, int $options])
string saveHTML (void)
int saveHTMLFile (string $filename)
string saveXML ([DOMNode $node [, int $options]])

- Création de node

DOMCDATASection createCDATASection (string $data)
DOMElement createElement (string $name [, string $value])
DOMText createTextNode (string $content)

- Importation par copie

DOMNode importNode (DOMNode $importedNode [, bool $deep])

- Sélection

DOMElement getElementById (string $elementId)
DOMNodeList getElementsByTagName (string $name)

Méthodes de DomNode de manipulations des éléments enfants

DOMNode {
// pour tester si le nœud a un/des enfant(s)
bool hasChildNodes (void)

// pour ajouter un nœud / enfant enfin de listes, la valeur retournée est le nœud inséré
// un nœud ne peut être qu'à un seul endroit…
DOMNode appendChild (DOMNode $newnode)
// pour insérer un nœud parmi d'autre (devant un autre) , la valeur retournée est le nœud inséré
DOMNode insertBefore (DOMNode $newnode [, DOMNode $refnode])
// pour remplacer un nœud ($oldnode) par un autre ($newnode) , la valeur retournée est le nœud inséré
DOMNode replaceChild (DOMNode $newnode , DOMNode $oldnode)

// pour supprimer un nœud enfant, , la valeur retournée est le nœud supprimé
DOMNode removeChild (DOMNode $oldnode)

// pour dupliquer (en profondeur si $deep = true) un noeud
DOMNode cloneNode ([bool $deep])
// et encore d'autres méthodes comme bool
isSameNode (DOMNode $node)
// etc.
}

Structuration DOM: DOMNodeList

- Juste une liste de nœuds

DOMNodeList {
/* Propriétés */
readonly public int *$length* ;
/* Méthodes */
DOMNode DOMNodelist::item (int $index)
}

- exemples
  - For($i = 0; $i < $list->length; $i++) { … }
  - If ($list->item(0)->hasChildren()) { … }

Structuration DOM : DOMElement

- C'est aussi un DOMNode
  - Parcours selon un nœud
- Création : plutôt par un objet DOMDocument, sinon avec un nom de tag
- Méthodes liées aux attributs
  - string getAttribute (string $name)
  - bool hasAttribute (string $name)
  - bool removeAttribute (string $name)
  - DOMAttr setAttribute (string $name , string $value)
- sélection
  - DOMNodeList getElementsByTagName (string $name)

Structuration DOM : DOMElement

- Élément de base de l'arbre, avec parent, fratrie, enfants, etc.

```
// nom du tag
public readonly string $nodeName ;
// valeur (attention, pas toujours pertinent, dépend du type de nœud)
public string $nodeValue ;
// type : un entier prédéfini : http://php.net/manual/fr/dom.constants.php
// ex: 3 == XML_TEXT_NODE == du texte…
public readonly int $nodeType ;


// Cet attribut retourne le contenu texte de ce nœud et de ces descendants.
public string $textContent ;
```

Structuration DOM : DOMElement

- Attributs de structure

```
// nœud parent
public readonly DOMNode $parentNode ;
// listes des nœuds enfants
public readonly DOMNodeList $childNodes ;
// premier nœud inclus
public readonly DOMNode $firstChild ;
// dernier nœud inclus
public readonly DOMNode $lastChild ;
// nœud « frère » (ayant le même parent) précédent
public readonly DOMNode $previousSibling ;
// nœud « frère » (ayant le même parent) suivant
public readonly DOMNode $nextSibling ;
```

Organisation en « module »

Des morceaux de page réutilisables

Des morceaux de page plus facilement intégrables