Internet et bases de données

Clara Bertolissi

SIL-NTI

Université de Provence

Plan

- Formulaires
- Les objets en php
- Gestion des fichiers
- Fonction Date
- Sessions et cookies

Les formulaires

- Un formulaire est l'élément Html déclaré par les balises <FORM></FORM>
- Un formulaire contient un ou plusieurs éléments
  - Ces contrôles sont notés par exemple par la balise <INPUT TYPE= ...>
  - Parmi les types on a le

- Bouton radio
- Bouton case a`cocher
- Liste de selection
- Zone de texte
- Bouton reset
- Bouton de soumission

Les formulaires

<FORM NAME= METHOD= ACTION= >

…

</FORM>

- METHOD = on a deux méthodes d'envoie possibles :

  - GET: les informations des différents champs du formulaire apparaitrons dans la barre d'adresse (taille limitée)

  - POST : les informations n'apparaissent plus (taille illimitée des champs)

- ACTION = l'url de la page qui traitera les données du formulaires (page html avec du javascript, cgi, php, etc...)

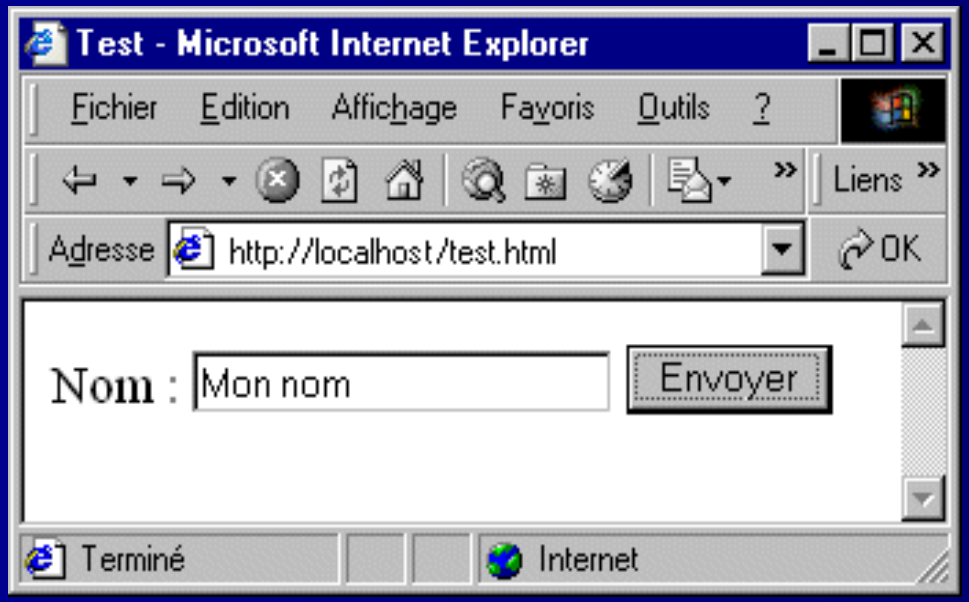
Variables php issues de formulaires

- Les variables php issues de formulaires HTML correspondent aux différents champs positionnés entre les balises <FORM> et </FORM>.
- La page qui reçoit ces variables est celle qui est désignée par l'attribut ACTION de la balise <FORM>.

exemple (fichier *test.html*) :

```
<HTML>
<FORM ACTION="test.php" METHOD=POST>
Nom : <INPUT TYPE=text NAME="nom">
<INPUT TYPE=submit VALUE="Envoyer">
</FORM>
</HTML>
```

Variables php issues de formulaires



Variables php issues de formulaires

- Dans le fichier test.php on pourra faire référence aux entrées du formulaire qui sont automatiquement stockées dans le tableau correspondant à la méthode d'envoie utilisée. Les variables du formulaire seront connues comme variables globales du script php destination.

Ex : test.php

```
<?php
$nom_choisi = $_POST["nom"];
  echo "$nom_choisi";
?>
```

Formulaires : passage de paramètres

- Les variables passées en paramètres seront considérées comme des chaînes des caractères. Mais les casts implicites permettront de les récupérer directement dans des variables d'autre type (entier, réel…).
- *Exemple :*

```
if ($_POST[“login”]==‘’user12’’)
echo ‘’Ok, valid user.’’;
/* … + données */
else
echo ‘’Acces forbiden.’’;
```

- Les données saisies n'apparaîtront pas dans l'URL et ne seront donc pas stockées dans les fichiers de log du serveur, contrairement à la méthode **GET**

Formulaires : passage de paramètres

- ***Méthode des ancres.***
- Les variables peuvent directement être passées par l'intermédiaire des URL.
- *Exemple :*

```
$id = 20;
echo ''<a href=\''fichier.php?action=buy&id=$id\''>Acheter</a>'';
```

- Cet exemple va créer dans le script destination les variables globales **$action** et **$id** avec les valeurs respectives **''buy''** et **''20''**.
- La barre d'adresse affichera l'URL suivante :

Upoload de fichiers

- Les formulaires peuvent aussi permettre à un internaute de transmettre un fichier vers le serveur.
- C'est la balise HTML suivante : **<input type="file" />** qui permet le chargement de fichiers. La balise FORM doit posséder l'attribut **ENCTYPE** de valeur **"multipart/form-data"**.
- La méthode utilisée sera **POST**. Il est utile d'imposer au navigateur une taille maximal de fichier **MAX_FILE_SIZE** en octets.

```
<form action="script.php" method="POST" ENCTYPE="multipart/form-data">
<input type="hidden" name="MAX_FILE_SIZE" value="1024000" />
<input type="file" name="mon_fichier" /><br />
<input type="submit" value="envoyer" />
</form>
```

Upload de fichiers

- Pour récupérer le fichier, il faut utiliser la variable d'environnement **$_FILES** qui est un tableau associatif.
- La variable **$_FILES['mon_fichier']** est elle aussi un tableau associatif possédant les champs suivants : name, type, size, tmp_name (nom temporaire sur le serveur)
- Si aucun fichier n'a été envoyé par le client, la variable **mon_fichier** vaudra la chaîne de caractères : **''none''** ou bien **''''** (chaîne vide).
- La durée de vie du fichier temporaire sur le serveur est limitée au temps d'exécution du script. Pour pouvoir exploiter ultérieurement le fichier sur le serveur, il faut le sauvegarder dans un répertoire et lui donner un vrai nom.

Upload de fichiers

- A la réception, la variable **$_FILES** sera un tableau de tableaux associatifs.

- *Exemple :*

```
$file_tab = $_FILES['mon_fichier'];
foreach($files_tab as $file) {
        /* faire le traitement prevu :
        - extraire le nom du fichier temporaire sur le serveur
        - vérifier la taille et le type
        - donner un nouveau nom, emplacement et extension du fichier
        - copier le fichier temporaire dans son nouvel emplacement */
}
```

- Les fichiers temporaires sont généralement placés dans le répertoire **/tmp** du serveur. C'est la directive de configuration **upload_tmp_dir** du fichier **php.ini** qui détermine l'emplacement des fichiers chargés par la méthode POST.

Exemple

Exemple du cas du chargement de ce qui doit être une image GIF de moins de 1024.000 octets :

```
// création d'une variable contenant toutes les infos utiles
$file = $_FILES['mon_fichier'];
// si un fichier a bel et bien été envoyé :
if ($file != ''none'') {
    // extraction du nom du fichier temporaire sur le serveur :
    $file_tmp = basename($file['tmp_name']);
    // vérification de la taille et du type MIME
    if(($file['size'] <= 1024000) || ereg(''gif$'',$file['type']))
        // nouveau nom, emplacement et extension du fichier :
        $file_def = $dir.'/'.$name.'.'.$ext;
        // copie du fichier temporaire dans son nouvel emplacement :
        copy($file_tmp, $file_def);
    }
}
```

Upoload de fichiers

- Pour charger simultanément plusieurs fichiers, il suffit de rajouter des crochets au nom du champ HTML **file**, et de mettre autant de champs **file** que désiré.

- *Exemple :*
- **<input type="file" name="mes_fichiers[]" />**
- **<input type="file" name="mes_fichiers[]" />**
- **<input type="file" name="mes_fichiers[]" />**
- **<input type="file" name="mes_fichiers[]" />**

- Dans cet exemple, l'internaute pourra charger jusqu'à quatre fichiers.

Upoload de fichiers

On aurait pu utiliser, a` la place des crochets, des champs **file** de noms complètement différents.

<input type="file" name="mon_fichier" />
<input type="file" name="mon_autre_fichier" />

Par contre, à la réception, on ne peut plus utiliser de boucle !

Exemple :

```
$file = $_FILES['mon_fichier'];
// puis faire le traitement
$file = $_FILES['mon_autre_fichier'];
// puis refaire encore le même traitement
```

L'approche utilisant des crochets convient au cas où le nombre de fichiers à charger est dynamique (non connu à l'avance, dépendant de paramètres divers).

Upoload de fichiers

Pour les versions PHP4 supérieures à la 4.0.2, il existe une autre méthode, plus simple :

Exemple 1 :

```
// vérification que le fichier a bien été envoyé par la méthode POST
if (is_uploaded_file($mon_fichier)) {
    // déplace le fichier dans le répertoire de sauvegarde
    copy($mon_fichier, $dest_dir);
}
```

Exemple 2 :

```
/* déplace directement le fichier dans le répertoire de sauvegarde en
    faisant les vérifications nécessaires */
move_uploaded_file($mon_fichier, $dest_dir);
```

Les classes en php

- Php gère la programmation orientée objet à l'aide de classes.
- Mot clef « class »
- Les variables de classes (propriétés) sont déclarées par « var » (Php4)
- Element courant : mot clef « $this »
- Constructeur: méthode du même nom que la classe

```
<?
  class Carre {
     var $nombre;
      function Execute () {
        echo «  <br> ».$this->nombre * $this->nombre;
      }
  }
?>
```

Les objets

```
<?
   $ObjCarre = new Carre();
   $ObjCarre->nombre = 10;
   $ObjCarre->Execute (); // affiche 100.

   $AutreCarre = $ObjCarre; // ici copie en Php4
   $AutreCarre->nombre = 2;
   $AutreCarre->Execute (); // affiche 4.
   $ObjCarre->Execute (); // Affiche 100 en Php4
?>
```

Les objets sont implicitements passés en copie (Php < 5).
Pour copier un objet en Php5 on utilise :
$copie-objet = ***clone*** $objet

Héritage

- Mot clef « extends »
- Pas d'héritage multiple
- Possibilité de surcharge

```
<?
  class Chef extends Personne {
    public $secretaire;              // Personne

     function maFonction () {
      Personne::maFonction();
      echo "prérogatives en plus";
    }
  }
?>
```

Exemple

```
class Voiture {                    // déclaration de la classe
     public $couleur;                          // déclaration d'un attribut
     public $belle = TRUE;      // initialisation d'un attribut
      function voiture() {          // constructeur
              $this->couleur = ''noire'';
     }    // le mot clé $this faisant référence à l'objet est obligatoire
      function Set_Couleur($couleur) {
              $this->couleur = $couleur;
     }
}

$mavoiture = new Voiture();                // création d'une instance
$mavoiture->Set_Couleur(''blanche'');// appel d'une méthode
$coul = $mavoiture->couleur;                              // appel d'un attribut
```

Exemple d'héritage

```
class A {
   function example() {
       echo "I am A::example() and provide basic functionality.\n";
   }
}
class B extends A {
   function example() {
       echo "I am B::example() and provide additional functionality.\n";
       parent::example();
   }
}
$b = new B;
// This will call B::example(), which will in turn call A::example().
$b->example();
```

Quelques fonctions :

- **get_declared_classes()** : retourne un tableau listant toutes les classes définies
- **class_exists($str)** : vérifie qu'une classe dont le nom est passé en argument a été définie
- **get_class($obj), get_parent_class** : retournent le nom de la classe (parent) de l'objet **$obj**
- **get_class_methods($str)** : retourne les noms des méthodes de la classe **$str** dans un tableau
- **get_class_vars($str)** : retourne les valeurs par défaut des attributs de la classe **$str** dans un tableau associatif
- **get_object_vars($obj)** : retourne un tableau associatif des attributs de l'objet **$obj** les clés sont les noms des attributs et les valeurs, celles des attributs si elles existent
- **is_subclass_of($obj,$str)** : détermine si l'objet **$obj** est une instanciation d'une sous-classe de **$str**, retourne VRAI ou FAUX
- **method_exists($obj,$str)** : vérifie que la méthode **$str** existe pour une classe dont **$obj** est une instance, retourne VRAI ou FAUX

En bref

- Tout objet instancié est une variable et peut à se titre être passé en argument à une fonction ou bien être un retour de fonction ou encore être sauvegardée en donnée de session.

- Une classe fille hérite de tous les attributs et méthodes de la classe parente dont elle est une extension (d'ou la syntaxe **extends**). Il est possible de surcharger les méthodes, d'en définir de nouvelles…

- PHP 5 :
- Copies des objets par réference
- Fonction de destruction explicites
- Visibilité publique, privée, protegé des attributs

Gestion de fichiers

Quelques fonctions:

- **$fp = fopen($file [,$mode])** : ouverture du fichier identifié par son nom **$file** et dans un mode **$mode** particulier, retourne un identificateur **$fp** de fichier ou **FALSE** si échec
- **fclose($fp)** : ferme le fichier identifié par le **$fp**
- **fgets($fp, $length)** : lit une ligne de **$length** caractères au maximum
- **fputs($fp, $str)** : écrit la chaîne **$str** dans le fichier identifié par **$fp**
- **fgetc($fp)** : lit un caractère
- **feof($fp)** : teste la fin du fichier
- **file_exists($file)** : indique si le fichier **$file** existe
- **filesize($file)** : retourne la taille du fichier **$file**
- **filetype($file)** : retourne le type du fichier **$file**
- **unlink($file)** : détruit le fichier **$file**
- **copy($source, $dest)** : copie le fichier **$source** vers **$dest**
- **readfile($file)** : affiche le fichier **$file**
- **rename($old, $new)** : renomme le fichier **$old** en **$new**

Fichiers

Exemple typique d'affichage du contenu d'un fichier :

```
<?php
$file = ''fichier.txt'' ;
if( $fd = fopen($file, ''r'')) {                    // ouverture du fichier en lecture
        while ( ! feof($fd) ) {                     // teste la fin de fichier
                $str .= fgets($fd, 1024);
        /* lecture jusqu'à fin de ligne où des 1024 premiers caractères */
        }
        fclose ($fd);                               // fermeture du fichier
        echo $str;                                  // affichage
} else {
        die(''Ouverture du fichier <b>$file</b> impossible.'');
}
?>
```

Fichiers

- La fonction **fopen** permet d'ouvrir des fichiers dont le chemin est relatif ou absolu. Elle permet aussi d'ouvrir des ressources avec les protocols HTTP ou FTP. Elle renvoie FALSE si l'ouverture échoue.

- *Exemples :*
- **$fp = fopen("../docs/faq.txt", "r");**
- **$fp = fopen("http://www.php.net/","r");**
- **$fp = fopen("ftp://user:password@cia.gov/", "w");**

- Les modes d'ouverture :
- '**r**' (lecture seule), '**r+**' (lecture et écriture), '**w**' (création et écriture seule), '**w+**' (création et lecture/écriture), '**a**'(création et écriture seule ; place le pointeur de fichier à la fin du fichier), '**a+**' (création et lecture/écriture ; place le pointeur de fichier à la fin du fichier)

Repertoires

Il est possible de parcourir les répertoires grâce à ces quelques fonctions :

- **chdir($str)** : Change le dossier courant en **$str**. Retourne TRUE si succès, sinon FALSE.
- **getcwd()** : Retourne le nom du dossier courant (en format chaîne de caractères).
- **opendir($str)** : Ouvre le dossier **$str**, et récupère un pointeur **$d** dessus si succès, FALSE sinon et génère alors une erreur PHP qui peut être échappée avec @.
- **closedir($d)** : Ferme le pointeur de dossier **$d**.
- **readdir($d)** : Lit une entrée du dossier identifié par **$d**. C'est-à-dire retourne un nom de fichier de la liste des fichiers du dossier pointé. Les fichiers ne sont pas triés. Ou bien retourne FALSE s'il n'y a plus de fichier.
- **rewinddir($d)** : Retourne à la première entrée du dossier identifié par **$d**.

Repertoires

Exemple 1:

```
<?php
if ($dir = @opendir('.')) {                    // ouverture du dossier
  while($file = readdir($dir)) {               // lecture d'une entrée
     echo ''$file<br />";          // affichage du nom de fichier
     }
     closedir($dir);                           // fermeture du dossier
}
?>
```

$dir est un pointeur vers la ressource dossier

$file est une chaîne de caractères qui prend pour valeur chacun des noms de fichiers retournés par **readdir()**

Repertoires

Il existe un autre moyen d'accéder aux dossiers : l'utilisation de la classe **dir.**

Ses attributs :

- **handle** : valeur du pointeur
- **path** : nom du dossier

Ses méthodes :

- **read()** : équivalent à **readdir($d)**
- **close()** : équivalent à **closedir($d)**

Constructeur :

- **dir($str)** : retourne un objet **dir** et ouvre le dossier **$str**

Repertoires

Exemple 2 :

```
<?php
$d = dir('.');                          // ouverture du dossier courant
echo "Pointeur: ".$d->handle."<br />";
echo "Chemin: ".$d->path."<br />";
while($entry = $d->read()) {            // lecture d'une entrée
    echo $entry."<br />";
}
$d->close();                            // fermeture du dossier
?>
```

Cet exemple est équivalent au précédent. Ils listent tous les deux les fichiers et sous répertoires du dossier courant.

Fonction Date

- Les fonctions de dates et heures sont indispensables sur Internet et pour la conversion des dates en français.

Quelques fonctions :

- **date("$format")** : retourne une chaîne de caractères contenant la date et/ou l'heure locale au format spécifié
- **getdate()** : retourne un tableau associatif contenant la date et l'heure
- **checkdate($month, $day, $year)** : vérifie la validité d'une date
- **mktime ($hour, $minute, $second, $month,$day, $year)** : retourne le timestamp UNIX correspondant aux arguments fournis c'est-à-dire le nombre de secondes entre le début de l'époque UNIX (1er Janvier 1970) et le temps spécifié
- **time()** : retourne le timestamp UNIX de l'heure locale

Fonction Date

Exemple 1 :

```
echo date("Y-m-d H:i:s");
/* affiche la date au format MySQL : '2002-03-31 22:30:29' */
```

Exemple 2 :

```
if(checkdate(12, 31,2001))
        echo ''La St Sylvestre existe même chez les anglais !!!'';
```

Exemple 3 :

```
$aujourdhui = getdate();
$mois = $aujourdhui['mon'];
$jour = $aujourdhui['mday'];
$annee = $aujourdhui['year'];
echo "$jour/$mois/$annee";          // affiche '31/3/2002'
```

Fonction Date

Les formats pour **date** :

- **d** Jour du mois sur deux chiffres [01..31] **j** Jour du mois sans les zéros initiaux
- **l** Jour de la semaine textuel en version longue et en anglais
- **D** Jour de la semaine textuel en trois lettres et en anglais
- **w** Jour de la semaine numérique [0..6] (0: dimanche)
- **z** Jour de l'année [0..365]
- **m** Mois de l'année sur deux chiffres [01..12] **n** Mois sans les zéros initiaux
- **F** Mois textuel en version longue et en anglais
- **M** Mois textuel en trois lettres
- **Y** Année sur 4 chiffres **y** Année sur 2 chiffres
- **h** Heure au format 12h [01..12] **g** Heure au format 12h sans les zéros initiaux
- **H** Heure au format 24h [00..23] **G** Heure au format 24h sans les zéros initiaux
- **i** Minutes [00..59] **s** Secondes [00.59]
- **a** am ou pm **A** AM ou PM
- **L** Booléen pour savoir si l'année est bisextile (1) ou pas (0)
- **S** Suffixe ordinal anglais d'un nombre (ex: *nd* pour 2)
- **t** Nombre de jour dans le mois donné [28..31]
- **U** Secondes depuis une époque **Z** Décalage horaire en secondes [-43200..43200]

Fonction Date

Les clés du tableau associatif retourné par getdate :

- **seconds** : secondes
- **minutes** : minutes
- **hours** : heures
- **mday** : jour du mois
- **wday** : jour de la semaine, numérique
- **mon** : mois de l'année, numérique
- **year** : année, numérique
- **yday** : jour de l'année, numérique
- **weekday** : jour de la semaine, textuel complet en anglais
- **month** : mois, textuel complet en anglais

Les sessions

- Les sessions sont un moyen de sauvegarder et de modifier des variables sans qu'elles ne soient visibles dans l'URL et quelque soient leurs types (tableau, objet…).
- Cette méthode permet de sécuriser un site, d'espionner le visiteur, de sauvegarder son panier (e-commerce), etc.
- Les informations de sessions sont conservées en local sur le serveur tandis qu'un identifiant de session est posté sous la forme d'un cookie chez le client (ou via l'URL si le client refuse les cookies).

Les sessions

- Conserver des données d'une page à l'autre
- Très simple d'utilisation
- Pour placer un objet en session, il doit être déclaré avant son démarrage
- Les modifications aux variables de session sont enregistrées automatiquement
- Les variables de session sont accessibles comme des variables globales du script.
- Elles se trouvent dans le tableau global $_SESSION

Foctions pour les sessions

- session_start ()
  démarrage de la session
- session_register (“var”)
  enregistrement d'une variable **$var** en session
- session_unregister (“var”)
  déréférencement de la variable **$var** dans la session en cours.
- session_destroy ()
  détruit la session de l'utilisateur actuel et détruit ses données.

Exemple

Page1.php

```
<?
session_start (); // crée la session si inexistante
$valeur = 10;
session_register ("valeur"); // stocke $valeur dans la session
$valeur = 20;
?>
```

Page2.php

```
<?
session_start ();
echo $valeur; //affiche 20.
?>
```

Exemple

- Une session doit obligatoirement démarrer avant l'envoi de toute information chez le client .
- Donc pas d'affichage et pas d'envoi de header. On peut par exemple avoir une variable globale contenant le code HTML de la page et l'afficher à la fin du script.

Exemple :

```
<?
$out = ''<html><body>'';
session_start();                                  // démarrage de la session
if(! isset($client_ip)) {
    $client_ip = $_SERVER['REMOTE_ADDR'];
    session_register(''client_ip'');  // enregistrement d'une variable
}
    /* … + code de la page */
$out .= ''</body></html>'';
echo $out;
?>
```

Exemple

```
<?
session_start (); // crée la session
If (isset($_POST[“nom”]) {
     $user = $_POST[“nom”] ;
     $_SESSION[“nom”] = $user ;
}
?>


<?
session_start ();

echo “Bonjour”. $_SESSION[“nom”] ;
}
?>
```

Remarques

- Sauvegarder des variables de type objet dans une session est la méthode de sécurisation maximum des données : elles n'apparaîtront pas dans l'URL et ne pourront pas être forcées par un passage manuel d'arguments au script dans la barre d'adresse du navigateur.
- Les données de session étant sauvegardées sur le serveur, l'accès aux pages n'est pas ralenti même si des données volumineuses sont stockées.
- Une session est automatiquement fermée si aucune requête n'a été envoyée au serveur par le client durant un certain temps (2 heures par défaut dans les fichiers de configuration Apache).
- Une session est un moyen simple de suivre un internaute de page en page (sans qu'il s'en rende compte). On peut ainsi sauvegarder son parcours, établir son profil et établir des statistiques précises sur la fréquentation du site, la visibilité de certaines pages, l'efficacité du système de navigation…

Les cookies

- **setcookie** («nom_cookie », valeur_cookie);
  Définition du cookie et de sa valeur
- on retrouve le cookie dans le tableau $_COOKIE[] (ou $HTTP_COOKIE_VARS[] version < 4.1.0)
- **$_COOKIE**[«nom_cookie »] Permet de tester la valeur du cookie.
- Pour modifier le contenu d'un cookie, il faut réutiliser la fonction setcookie()
- Pour supprimer un cookie, faire un setcookie ("nom_cookie") en ne spécifiant aucune valeur.

La fonction setcookie

- **int setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])**

- **Name** : nom du cookie
- **value** : valeur stockée (vide : le cookie est détruit)
- **expire** : date d'expiration du cookie (sans date : cookie de session) la date est au format "Unix timestamp" (nombre de secondes depuis le 01/01/1970) que l'on peut obtenir avec **mktime()** ou **time()**.
- **path** : chemin du répertoire où est valide le cookie (par defaut: le chemin de la page qui fait le setcookie)
- **domaine** : nom du domaine où est valide le cookie
- **secure** : indique si le cookie est sécurisé

Exemple

- créer un cookie ayant pour nom "societe" et pour valeur "aston"

  il est valide 15 jours et est accessible sur toutes les pages du site

```
<?
setcookie("societe","aston",time()+60*60*24*15,"/");
?>
```

Exemple

```
<?
$Last = false;
if (isset ($_COOKIE['date_connexion']))
{
   $Last = $_COOKIE['date_connexion'];
}

setcookie ('date_connexion', date ("d-m-Y H:m:s"));
if ($Last)
   echo "Votre dernière connexion: ".$Last;
?>
```

Exercise : compteur de visites

- On souhaite comptabilisé le nombre de chargement d'une page (la page d'accueil par exemple). On va procéder de la façon suivante : le compteur numérique sera stocké dans un fichier, à la première ligne. Ce fichier contiendra seulement un nombre, celui des visites.

- *Phase 1 : incrémenter la valeur dans le fichier*
- Ce fichier va s'appeler **compteur.cpt**.
- Cet algorithme est écrit directement dans le code source de la page d'accueil.

Compteur de visites

- *Algorithme :*
- `$fichier = "compteur.cpt";` // affectation du nom de fichier
- `if( ! file_exists($fichier)) {` // test d'existence
- `// initialisation du fichier si n'existe pas encore`
- `$fp = fopen($fichier,"w");` // ouverture en écriture
- `fputs($fp,"0");` // écriture
- `fclose($fp);` // fermeture
- `}`
- `$fp = fopen($fichier,"r+");` // ouverture
- `$hits = fgets($fp,10);` // lecture
- `$hits++;` // incrémentation
- `fseek($fp,0);` // positionnement
- `fputs($fp,$hits);` // écriture
- `fclose($fp);` // lecture

Exercise : compteur de visites

- *Phase 2: Généralisation aux autres pages*

Comme les internautes peuvent atterrir sur des pages internes à votre site sans passer par l'accueil, il peut être intéressant de pouvoir comptabilité des visites des autres pages.

Créons donc une fonction que l'on placera dans un fichier à part par exemple **compteur.php** et que l'on appellera par inclusion comme ceci :

```
<?php    include("compteur.php");
Mon_Compteur("ma_page") ;       ?>
```

Remplacez **"ma_page"** par un identifiant unique pour chaque page.

Généralisation aux autres pages

```
<?
function Mon_Compteur($page) {
  $fichier = $page.".cpt";
  if(!file_exists($fichier)) {
      $fp = fopen($fichier,"w");
      fputs($fp,"0");
      fclose($fp);
  }
  $fp = fopen($fichier,"r+");
  $hits = fgets($fp,10);
  $hits++;
  fseek($fp,0);
  fputs($fp,$hits);
  fclose($fp);
}
?>
```

Exercise : compteur de visites

- *Phase 3 : Protection contre la redondance*
- Comme un visiteur peut charger plusieurs fois la même page au cours d'une même visite, ce mode de décompte n'est pas assez précis et va surestimé le nombre réel de visiteurs.
- Solutions ?
- Il faut garder en mémoire le fait qu'un visiteur est déjà passé par la page et incrémenter le compteur seulement si ce n'est pas le cas

Protection contre la redondance

- On va créer une variable de session : un tableau contenant la liste des pages visitées.

- Principe du nouvel algorithme :
  on teste l'existence de la variable de session **$PAGES_LIST.**

Si elle n'existe pas on la crée, on y ajoute la page en cours et on appelle la fonction **Mon_Compteur**.

Si elle existe, on teste la présence de la page en cours dans ce tableau, si elle n'y est pas alors on l'y met et on appelle **Mon_Compteur**.

- L'appel est légèrement différent :

```
<?php
$page = "ma_page";              // définition du nom de la page
include("compteur.php");        // chargement de l'algorithme
?>
```

Protection contre la redondance

Code à rajouter dans le fichier **compteur.php** :

```
session_start();                    // démarrage de la session
if( ! isset($PAGES_LIST)) {  /* test de l'existence de la variable de session */

    $PAGES_LIST = array($page); // création de la variable
    session_register($PAGES_LIST); // ajout de la page en cours
    Mon_compteur($page);                        // incrémentation du compteur
} else {
    if( ! in_array($page, $PAGES_LIST)) {   // test de redondance
            $PAGES_LIST[ ] = $page;             /* ajout dans la variable de
    session pour éviter la redondance */
            Mon_compteur($page);    // incrémentation du compteur
    }
}
```

Aujourd’hui on a vu

- Passage de parametres avec les formulaires
- La notion de classe en Php
- Gestion des fichiers
- Fonction Date
- Sessions et cookies