Programmation Internet
PHP - base de la programmation

Jean-Michel Ilié

IUT Paris 5 - département informatique

Variables et typage

- Création : a priori implicite, fait à la demande (*pas de mot 'var'*)
- Typage défini par affectation ou typage explicite
- Visibilité (niveau du bloc contenant la déclaration)
  - ✓ *le script au maximum*
  - ✓ *voir aussi la notion de variables super-globales*

Typage implicite

```
$a=1       //entier (int)
$a=1.2     //double (double ou float)
$a="abc"   //chaîne (string)
```

int *double* *string* *unknown type*
array
object *class*

Typage explicite

```
$bool=settype ($v, $typ) ;    // type $typ pour la variable $v, rend true si réussi
$typ= gettype ($v);           // rend le type de $v
```

Gestion explicite des types de données

- Test sur les types des variables

  *$bool=* ***is_int****($v); **is_double****($v); **is_long****($v);*
  is_float*($v); **is_real****($v);*
  is_string*($v)*

- transtypage explicite

  *$i=**intval** ('ch', [int base]);*
  *$f=**doubleval**('ch');*
  $ch=**strval**($d);

- Le transtypage par 'cast' existe aussi

  $v=(typ)$w; //exple $v=(int)$w

Quelques exploitations du type bool

true, **false**
(insensible à la casse)

Evaluation à FALSE par un transtypage booléen

- chaîne vide, ou '0' ou '0.0'
- tableau vide
- objet vide
- valeur NULLs

trim() élimine les espaces en début et fin de chaine.

```
$ch=trim($_GET['nom']);
if ((bool) $_GET['nom'])
        echo "il faut saisir un nom";
```

null pour variable non définie
(insensible à la casse)

Test d'existence d'une variable avec '**== null**'
ou fonction booléenne **isset($v)**

```
$bool= isset($v); //rend false si la variable $v n'est pas définie
```

Fonctions

- Passage des paramètres
  - par valeur (défaut)
  - par référence (symbole &)
  - valeur défaut possible
- Contexte d'exécution
  - variable locale à la session d'interprétation
  - accès au variables globales au sein des fonctions : (mot clé **global**)
- Retour de fonction
  - par valeur
  - par référence

Passage et retour par valeur, par défaut
(du fait des applications internet).

!! les références sont des alias
(avec le même statut que la variable d'origine)

Exemples pour les fonctions

Déclarations

```
function f($b=3)  //passage par valeur, valeur défaut  possible
{global $a;              //pour utiliser la var globale $a
  ...$a=2; return $a;}  //retour : un tableau
function g(&$c)     // paramètre par référence
{... $c .= "<br>"; ...}
function &h() // !! retour d'une référence
{...return $uneRef; }
```

Appels

```
$a= 2;    //déclaration globale
$a1="...";   $j = f($a1);              //passage par valeur de $a1 dans f
$c="debut";  g($c);  echo ($c);        //passage par référence dans g
$r &= h();              //retour d'une référence (encore le &)
```

Clarifier la présentation du texte php

Constante

- **define** ("nomConst","val"); //définit une constante
  //les prédéfinies : TRUE(1), FALSE(0 ou ch vide), var PHP : PHP_OS...
- **defined**("nomConst"); // rend TRUE si la valeur existe

Inclusion de fichier

- **require** ("fic.php"); //remplacement de la ligne require par le texte du fichier
- **include**("fic.php"); //ajout du texte contenu (sans élmination de la ligne)
  //-> 'include' itéré possible

Préciser le chemin ou
se baser sur l'emplacement par défaut pour les includes du serveur

```
require("./commun.php"); //par exemple les constantes de l'ensemble des pages
for ($i=0; $i<3; $i++)  include ('fic_' . $i . '.php') //inclusions dynamiques
```

Gestion des erreurs et deboguage

Déboggage par affichage du contenu des variables

- **echo ("msg" . $v)**
- **printf ("msg %s", $v); …**
  - affichage éventuellement formaté, ..

- **var_dump (*$v*)**
  - affichage correct d'une variable structurée (ici $v)

  tableau, tableau de tableaux, objet, ...
- **print_r (*$v*)** affichage récursif et bien présenté

Gestion des erreurs en PHP

- **@expression**
  - Force la continuation de l'exécution même si l'expression (PHP) est erronée
  - Il n'y a donc pas de message système affichée sur la page client
  - On peut ajouter @ à toute variable, fonction, include()...
- echo **$php_errormsg** **:** variable globale indiquant l'erreur
  - cette variable à sauvegarder le msg d'erreur que l'on peut afficher
  - condition : l'option **track_errors** doit être activée (fichier configuration php.ini)
- **die** (string message)
  - alias du exit(), avec possibilité d'un message d'erreurs

Exemples de gestion d'erreurs

*$tab= **file** (nomFic [, int includePath])*
// instruction de lecture qui retourne un tableau des lignes à partir du fichier texte nomFic

pas d'alerte
si fichier 'fic'
absent

mais (suite du or),
un exit avec message explicite sur l'erreur.

```
<?php
 $my_file = @file ('fic') or
                die ("Impossible d'ouvrir le fichier : erreur" . $php_errormsg);
?>
```

```
$value = @$tab[$cle];
```

// il n'y aura pas de blocage du programme si la clé $cle n'est pas définie dans $tab.

Gestion des tableaux

- Création statique ou dynamique
- types de tableaux indexés ou associatifs
- accès et parcours standard ou spécialisés
- Tableaux et HTML
- Tableaux super-globaux de PHP

Allocation et indiçage du tableau

```
$pays= array('it', 'fr', 'al');    //indice 0..2
```

Par défaut, les indices commencent à 0

```
$pays= array();
$pays[2]="it";   //indice 2
$pays[0]="fr";   //indice 0
$pays[]="al";    //indice 3
```

indiçage implicite automatisé

```
$pays= array(0 => "it", 1 => "fr");
```

=> Spécification d'index entiers et de leur valeur

```
unset($pays[2]);
```

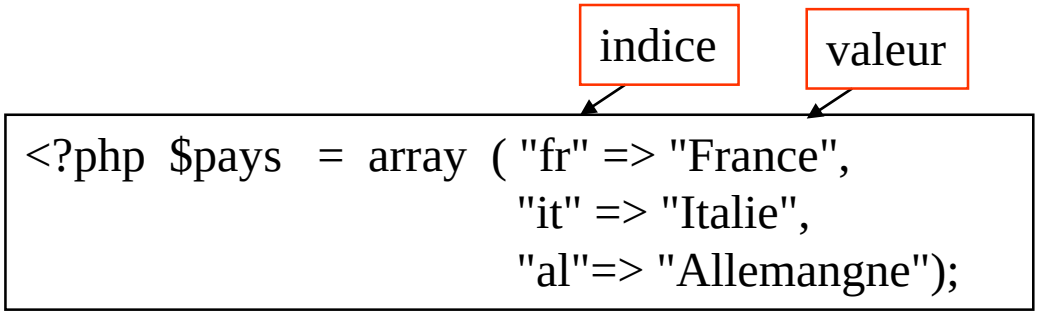
Désallocation mémoire appliquée sur un indice (2)

```
if ($pays[2]!=null) {}
```

Test d'une allocation mémoire sur un indice (2)

Cas des tableaux associatifs

Dans un tableau associatif,
les indices sont des identifiants (chaînes).
- avantage : meilleure intuition sur le contenu -



Parcours itératifs classiques

for(){}	foreach(){}	while( ){}	do{} while( )

```
<?
$pays[1]="it";
$pays[0]="fr";
$pays[]="al";
```

```
$nb= count($pays);
echo ("$nb pays : <br>\n");
for ($i=0; $i<$nb; $i++)  {echo $pays[$i]. " <br>\n" ;}
?>
```

Parcours avec la boucle foreach

le tableau
à parcourir

Pour chaque itération,
l'indice et la valeur

```
foreach ($pays as $cle => $valeur)
        {echo $cle . ' donne ' . $valeur . '<br>' ;}
```

```
foreach ($pays as $valeur)
        {echo $valeur . '<br>' ;}
```

La boucle foreach convient même si les indices utilisés ne sont pas séquentiels

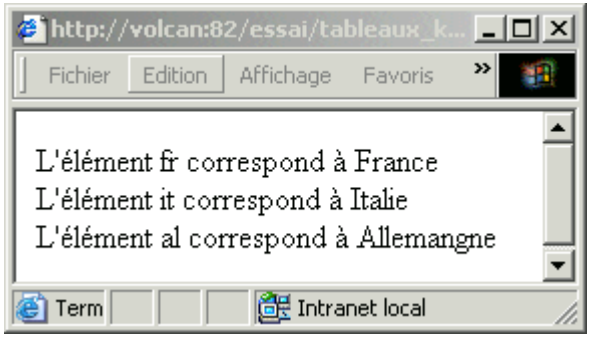
$valeur
=
$pays[$cle]

Exemple de parcours sur un tableau associatif

```
<?php
  $pays   =  array  ( "fr" => "France",
                      "it" => "Italie",
                      "al"=> "Allemangne");

  foreach ($pays as $cle => $valeur )
    echo ("L'élément" . $cle . "correspond à" .
  $valeur. "<br>");
  ?>
```

La boucle foreach convient encore



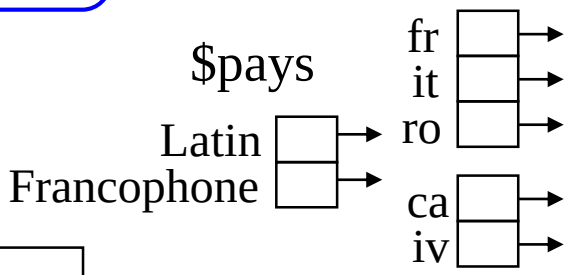
Représentation des tableaux à plusieurs dimensions

Déclarer des tableaux dans les tableaux

le tableau
et sa 1ère dimension

2ème dimension

```
$pays=array();
$pays['Latin'] =       array  (    "fr" => "France",
                                   "it" => "Suisse",
                                   "ro" => "Roumanie");
$pays['Francophone'] = array  ("ca" => "Canada",
                                    "iv" => "Cote d\'ivoire");
```



Un tableau à 2 dimensions est un tableau de tableaux

Exemple de parcours d'un tableau à 2 dimensions

```php
<?php
$pays =  array   ("Latin" => array  ("fr" => "France",
                                     "it" => "Suisse",
                                     "ro" => "Roumanie") ,
                 "Francophone" => array  ("ca" => "Canada",
                                          "iv" => "Cote d\'ivoire")
          );

foreach ($pays as $cle1 => $valeur1)
{   echo ("<b>". $cle1 . "</b> : <br>\n");     //1er niveau : noms de groupe de pays
    foreach ($valeur1 as  $cle2 => $valeur2)
            echo ("- ". $cle2 . " : " . $valeur2 .  "<br>\n");    //2ème niveau : les pays associés
}?>
```

$valeur1 = ?

Gestion globale sur les tableaux

Copie par valeur

$t1 = $t; *affectation*

Différence :

$tab= **array_diff** ($tab1, $tab2, ...); *tableau résultant des valeurs différentes dans les tableaux*

Itérateur :

array_walk ($tab, fct) *applique la fonction externe fct() sur chaque case du tableau $tab*

Tri :

sort ($tab) *tri du tableau*

asort ($pays); *tri des valeurs en laissant inchangés les indices*

usort ($pays, fct [,param]); *tri en appliquant la fonction d'ordre fct()*

Test spécifique:

array_key_exists ($index, $tab) *teste si $index est un index du tableau $tab*

Transformations entre chaînes et tableaux de chaînes

- $tab= **explode** ($separat, $ch [,nblimit]); rend un tableau des ss chaînes de $ch (exploite la chaîne servant de séparateur)
- $ch= **implode** ($separat, $tab); inverse de explode

remarque : split et join sont similaires à explode, implode

- **list**(liste de variables PHP) = $tab; assigne une liste de variables à partir des lignes d'un tableau

```
$data = "bond:*:1023:1000:/home/bond:/bin/sh";
$t = list($user, $pass, $uid, $gid, $gecos, $home, $shell)
        = explode(":", $data);
echo $user;          // 'bond'
echo $pass;          // '*'
```

Connaissance de l'environnement
les tableaux super-globaux

tableaux associatifs **prédéfinis** et **accessibles directement**, même au sein des fonctions.

$_GET	données de la méthode GET du protocole HTTP
$_POST	données de la méthode POST du protocole HTTP
$_REQUEST	idem GET ou POST indifférencié
$_COOKIE	données COOKIE du protocole HTTP
$_FILES	données sur les fichiers transmis par HTTP
$_GLOBALS	variables globales définies
$_SERVER	données sur le serveur WEB ou donnant le contexte d'exécution du script courant
$_ENV	données fournies sur l'environnement
$_SESSION	variables sessions

Exercice

A développer :

Ecrire une fonction qui affiche le contenu du tableau $_SERVER.

ANNEXE

UPLOAD de fichiers sur le serveur

- Conception d'un formulaire d'upload
- script PHP de rangement du fichier

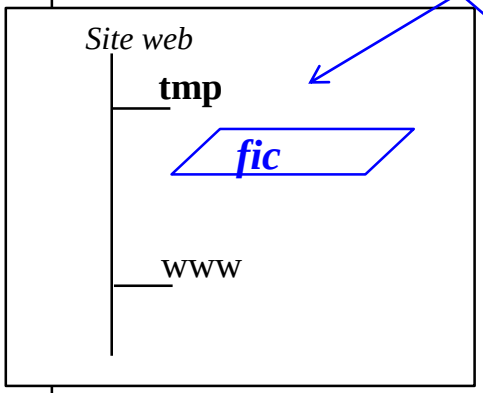
Utilisation du $_FILES
pour les fichiers à télécharger sur le serveur

```
<form method="post" action= "main.php" enctype="multipart/form-data">
        <input type="file" name= "fic"  />

        <div>  <?php  echo ("taille max du upload : " . MAX_FILE_SIZE)  </div>
        <input type="submit" value="Envoyer"  />
</form>
```

Constante php :
info optionnelle sur la taille max

upload



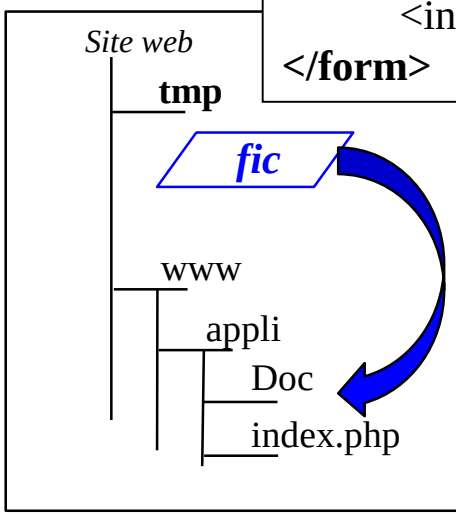

```
<?php
$_FILES ['fic']['name']          //nom du fichier (fic)
$_FILES ['fic']['‘tmp_name']     //nom absolu du fichier dans tmp
$_FILES ['fic']['type']
$_FILES ['fic']['size']          //taille du fichier
$_FILES ['fic']['error']         //en cas d’erreur de chargement
?>
```

La taille maximum (défaut 2Mb) et l’emplacement du répertoire tmp sont précisés dans le fichier php.ini

Exercice : stockage des fichiers uploadés

A développer : Ecrire le script qui stocke le fichier uploadé dans le répertoire de nom `Doc` de votre appli.

```
<form method="post" action= "main.php" enctype="multipart/form-data">
        <input type="file" name= "fic"  />
         <input type="submit" value="Envoyer"  />
</form>
```



☞ **en php :**
moveupload (*source*, *destination*)
pour sauvegarder le fichier transmis

Gestion des chaînes de caractères

- Gestion des chaînes de caractères
- Exploitation des modèles de chaînes (expression régulière)
- Spécialisation pour les URL

Construire un modèle de chaîne (expression régulière)

Positionnement de la comparaison

^expr	l'expression expr décrit un début de chaîne
$expr	idem pour une fin de chaîne
^expr$	expr décrit une chaîne complète

Les expressions régulières sont des modèles de chaîne représentant les chaînes à traiter.

Classes (type) de caractère dans un modèle

[a-zA-Z] | [0-9.\-]	*un caractère lettre **ou** un caractère chiffre le '.' ou le moins.*
[^;\]	*un caractère sauf le ';' et le '\'*
[[:alpha:]] [[:digit:]] [[:alnum:]] [[:space:]]	*caract. lettre, chiffre, lettreChiffre, ou héxa*
[[:lower:]] [[:upper:]] [[: punct:]]	*caract. minuscule, majuscule ou ponctuation*

Multiplicité d'occurrences consécutives

.{nb}	*'.' est le type d'un caractère quelconque, il en faut 'nb'.*
\.{inf,sup}	*(vrai) '.' en quantité au moins 'min' et au plus 'sup'*
[0-9]{inf,}	*au moins 'inf' chiffres*

^\-?[0-9]+$
^(.+)@(.+)\. (.+) $

abréviations : ?={0,1}, **+**={1,} et ***=**{0,}

Travailler sur les données
exploiter les modèles de chaîne

Utiliser le parenthèse pour distinguer
les différentes parties d'un modèle
(lecture de gauche à droite)

- $bool= **ereg**("**modele**", "chaîne" [, $tab]) *//compare le modèle et la 'chaîne'*
  //Décompose la chaîne suivant les '( )' du modèle et affecte $tab avec les ss chaînes
  // Note : indice 0 de $tab : toute la chaîne.
- $ch= **ereg_replace** ("**modele**", "sch", "chaîne") *//remplacement par 'sch' dans 'chaîne'*
  //de toute occurrence de sous chaîne qui correspond au 'modèle'
  //pour 'sch' : référence symbolique aux () du modèle (\\1 pour le 1er, ...)
- idem en ignorant la casse
  - **eregi**() et
  - **eregi_replace**()
    (met en minuscule)

```
if (ereg ("^(.+)@(.+)\. (.+)$", $email, $part))
      echo "adresse email valide : $part[0]      \n<br>"  .
           "nom : $part[1]                       \n<br>" .
           "domaines" $part[2] et $part[3]       \n<br>";
else echo "mauvaise adresse : recommencez";
```

Autres fonctions travaillant sur les modèles de chaîne

- $bool= **preg_match**("**modele**", "chaîne", [$tab]) //compare le modèle et la 'chaîne'

```
if (preg_match("`^(.+)@(.+)\. (.+)$`", $email, $part))
      echo "adresse email valide : $part[0]      \n<br>"  .
             "nom : $part[1]                      \n<br>" .
             "domaines" $part[2] et $part[3]     \n<br>";
else echo "mauvaise adresse : recommencez";
```

Gestion des sessions utilisateurs : les variables sessions

Variables-session

- Mémorisation : espace mémoire temporaire sur le serveur –
  Par défaut sauegarde en fichier de session. Possible en base de données.
  Format sérialisé (une seule chaîne par variable)
- Garbage collector Elimination automatique des fichiers de session
  (lancement périodique postérieur aux fins de session).
- Coût : Encombrement du serveur fonction du nb d'instances de l'application

A réserver à des tâches limitées
nécessitant la sécurité du serveur
ou une rapidité d'accès
Exemple : gestion de l'identité de l'utilisateur courant,
De son profil, dgestion d'un panier interactif ...

Déclaration et accès aux variables-session

- Possibilité de déclaration et d'accès au sein d'une page PHP :

```
<?php session_start();        //A placer en 1ère ligne du fichier.
?>
```

- Déclaration : une variable session correspond à un poste du tableau $_SESSION

```
$_SESSION['v']= $val;
```

La variable session 'v' est créée (ou ré-affecter). et possède le type et la valeur de $val .

- Accès :

```
$v1= $_SESSION['v'];
```

$v1 sera affectée avec la valeur de la variable-session 'v'.

note :
$_SESSION['v']= array(); définit la variable session 'v' comme un tableau.

Exemples d'utilisation d'une variable-session simple

```
<?php    session_start();

         if (!isset($_SESSION['nom']) )
                  $_SESSION['nom']="";
?>
```

initialisation de la variable-session 'nom'

```
<?php    session_start();

         $_SESSION['nom']= $_POST['nom'];
?>
```

mémorisation-session de la valeur de $_POST['nom']

Exemple d'utilisation d'une variable-session de type tableau

Gestion des compteurs des pages accédées pour un utilisateur

```
<?php session_start();

if (!isset($_SESSION[‘cpt'][))
   $_SESSION['cpt'] = array();

switch ($_POST['choix']) {
  case main : $_SESSION['cpt']['main'] ++;
              require('fic_'. 'main'.'php');
              break;
  case ajout :  $_SESSION['cpt'][‘ajout'] ++;
              require('fic_'.'ajout '.'php');
              break;
}
?>
```

Chaque requête provenant du navigateur: ajout, modif est comptabilisé avant d'appeler le script php de même nom.

Dans la variable session, le compteur à la position ['ajout'] est incrémenté, ceci chaque fois que la page ajout.php est demandée.

Exercice

A développer : Modifier le script affichant et traitant le formulaire de login (p24) pour mémoriser le nom de l'utilisateur dans une variable session.

(Ultérieurement, cela vous pemettra de vérifier
que l'utilisateur est bien connecté
en testant si cette variable session est différente de null).

Autres instructions gérant les sessions utilisateurs

Accès au nom système de la session :

- **session_name**() *//fonction qui rend le nom de la session courante*
- **session_id**() *//rend son identifiant*
- **SID** *//dans la page HTML produite (echo): constante identifiant la session*

Elimination explicite de variables-session :

- **session_unset**() *//élimination de toutes les variables-sessions de l'application*
- **session_unregister**($_SESSION['v']) *//élimine la variable session 'v'*
- **session_destroy**() *//détruit explicitement le fichier mémorisant les variables sessions*

Exemple de destruction explicite des variables sessions

```
<html>
<body
  onunload="window.location.href='quitter.php'; "
>
...
</body>
</html>
```

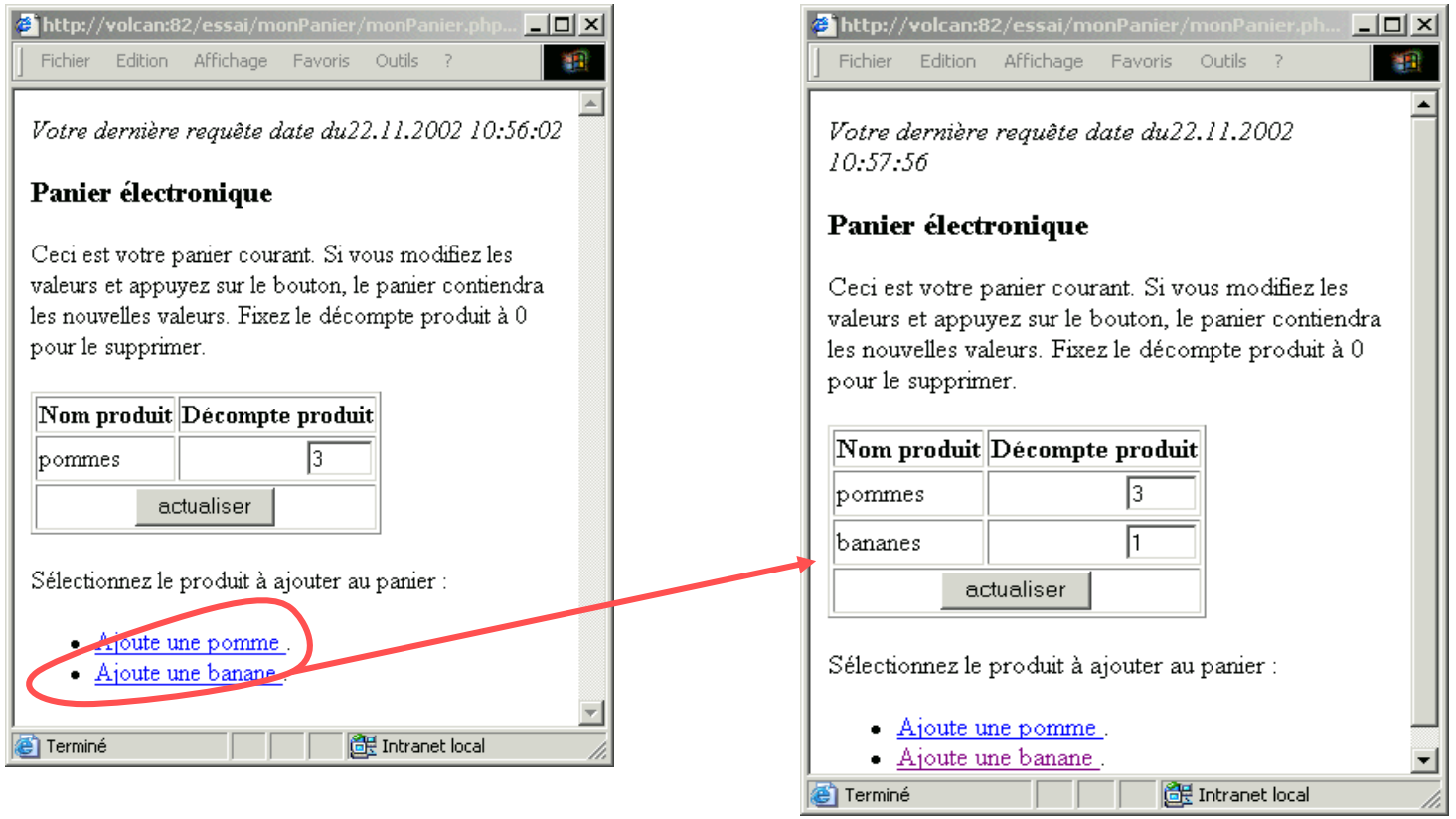
En fin de session-utilisateur, forcer la suppression des variables sessions. Evite des effets de bords car l'action du garbage-collector n'est pas immédiat.

quitter.php

```
<?php    session_start();
   session_destroy();
?>
```

Au déchargement de la page HTML (fermeture de la fenêtre, ...), on impose le lancement du script quitter.php

Exple : gestion d'un panier électronique à l'aide de variables-session



Exemple - partie 1 -

```
<?php session_start();          //à faire dans toute page
                                //qui créée des variables sessions ou y accède.
-------------------------------------------------------------------------------
    if (!isset($_SESSION['panier']))          // le panier est une variable-session :
       $_SESSION['panier']=array("pommes" => 1);   // tableau associatif produit=>quantité
-------------------------------------------------------------------------------
    if (isset($_GET['action']))
        switch($_GET['action'])
                {case ajout:        $prdt =$_GET['produit'];
                                    $_SESSION['panier'][$prdt]++;
                                    break;
                 case actualiser:   $_SESSION['panier'] = $_GET['panier'];
                                    break;
                }
    ?>
```

2 actions possibles suivant paramètre 'action' (voir plus loin)

Exemple - suite 2 - Affichage du panier (un tableau html)

```
<html><body><H3>Panier électronique</H3>
<form method="get" action="<?php echo $_SERVER['PHP_SELF']?>">
        <P>Ceci est votre panier courant...</P>
<table >
   <tr> <th>Nom produit</th>  <th>Décompte produit</th> </tr>
```

- suite 3 -

```
<?php
foreach  ($_SESSION['panier'] as $cle=>$valeur)
  if($valeur> 0)     // N'afficher que les produits non 'supprimés'
    echo ( sprintf(  '<tr><td>  %s   </td><td align="right">' .
                  '<INPUT type="text" size=3 NAME="panier[%s]" ' .
                                        'VALUE="%d"> </td></tr>',
                  $cle, $cle, $valeur)  );
?>
<tr> <td colspan="2">
       <input type='submit' name='action' value='actualiser'>  </td></tr>
</table>  </form>
```

nom du futur tableau d'entrée

Initialisation des champs aux valeurs du tableau-session

Exemple – suite 3 - Affichage des liens paramétrés

```
<p>Sélectionnez le produit à ajouter au panier :</P>
<ul>
 <li> <a href ="
          <?php echo ("{$_SERVER['PHP_SELF']}?action=\'ajout\'&produit=\'pommes\' ")
          ?> ">   Ajoute une pomme </a> </li>
 <li> <a href="
          <?php echo ("{$_SERVER['PHP_SELF']}?action=\'ajout\'&produit=\'bananes\' ")
          ?> ">     Ajoute une banane </a> </li>
</ul>
```

Affichage des 2 liens de fin de page

Gestion avancée du code HTML

Envoi de directives au navigateurs : entête de page HTML

- **header**($ch [, *boolean* $replace])

```
<?php
        header("Location:". $url) ;
?>
//force le navigateur à charger la page
dont l'adresse est donnée par $url
```

//!! ne pas envoyer une entête si une partie du corps (à savoir, le 'body') est déjà envoyé au client (voir le manuel php)

```
<html><body>
<?php
        header('Location: http://www.exple.com/');  /* Ceci produira une erreur. */
?>
```

Formatage HTML des chaînes PHP

Codage simple des caractères spéciaux pour les zones de saisies

$cha=**addslashes**($ch); //ajoute les '\' d'échappement

$chs=**stripslashes**($ch) //les retire

Utile pour coder les caractères spéciaux à envoyer au navigateurs

$cha=**htmlspecialchars**($ch); //codage HTML reconnu par le navigateur

"&" (et commercial) devient "&"

"""" (guillemets doubles) devient """

"'" (guillemet simple) devient "'"

"<" (inférieur à) devient "<"

">" (supérieur à) devient ">"

Utile pour empêcher l'interprétation des valeurs retournées vers le navigateur

$cha=**htmlentities**($ch); //chaîne HTML non interprétée par le navigateur

echo (htmlentities("<script>…</script>"); affiche mais n'interprète pas

Formatage HTML des URL

Rappel URL :
au sein d'une valeur : espace, interrogation, égal : +
autre partie : # % & + / = ? code %23 %25 %26

DECODAGE URL

- urldecode("ch") //décode le '+' en espace (*au sein des valeurs*)
- rawurldecode("ch") //decode le code %20 en espace (*ailleurs*)

ENCODAGE

- urlencode ("ch")//fonctions d'encodage dualles
- rawurlencode("ch")

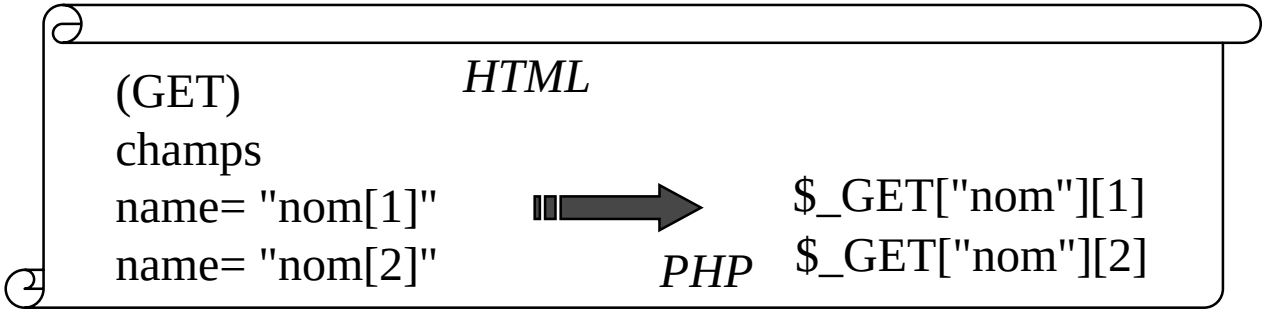
```
<a href= "http://main.php?candidat= " <?php urlencode($candidat)> ?> " >
```

code qui fonctionne même si $candidat
comporte des espaces : mettra le caractère '+' à la place.

Applications des tableaux PHP à la gestion de collections HTML

Un tableau PHP peut regrouper automatiquement un ensemble de valeurs d'un ou plusieurs champs de contrôle HTML.

- nom de champs suivi de [] dans la page HTML
- champs HTML possédant plusieurs valeurs de saisies



Exemple de gestion de collections html

```
<form action="array.php"  method="GET">
   Nom: <input type="text" name="util[nom]"><br>
   Email: <input type="text" name="util[email]"><br>
   Boisson favorite: <br>
   <select multiple name="boisson[]">
      <option value="chablis">Chablis
      <option value="coka">Coka cola
      <option value="eau">eau pure
      </select>
   <input type="submit">
</form>
```

Un nom avec [] va correspondre à un tableau php.

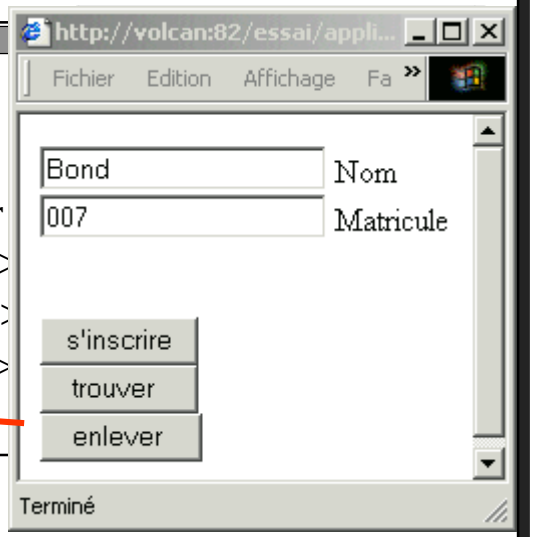
array.php

```
<html><body><?php
$U=$_GET['util']; $B=$_GET['boisson'];
echo nom : $U['nom'];
echo email : $U ['email'];
echo boissons favorites :
        foreach ($B as $valeur)
              {echo  $valeur . ' ';}
?></body></html>
```

note : pour une liste à sélection simple ou un groupe de boutons radio, un nom simple suffit pour récupérer l'unique valeur saisie ([]non nécessaires)

Gestion de boutons de soumission multiples

```
<html><body>
<form action="main.php" method="post">
        <input name="nom" type="text">     Nom      <br />
        < input name="mat" type="text">     Matricule      <br
 <br /><  input type="submit"  name="choix"   value = "s'inscrire"  >
 <br /><  input type="submit"  name="choix"   value = " trouver   " >
 <br /><  input type="submit"  name="choix"   value = " enlever  " >
</form></body></html>
```




```
<html><body><h3>Bienvenue M. <?php echo ("$nom");  ?>  </h3>
switch ($_POST['choix']) {
case "s'inscrire" : require (' inscrire.php');break;
case "trouver" : require ('trouver.php');break;
case "enlever" : require ('enlever.php');break;
default :echo ("mais il nous faudrait autre chose !! "); exit();
}?><body><html>
```

main.php

structuration en un fichier principal et trois fichiers secondaires dont un (seul) va être inclus suivant la valeur de 'choix' à à la soumission du formulaire.

Exemple de création de select HTML *à partir d'un tableau PHP (1)*

//avec php et javascript

```
<?php
$pays = array ("Francophone" => array ("France","Suisse","Roumanie") ,
               "Anglophone" => array ("Angleterre","Australie") );
  $LF=0;
  $LA=count($pays['Francophone']);
?>

<script type="text/javascript">
  var LG = new Array(<?php echo ($LF) ?>, <?php echo ($LA) ?>);
</script>
```

Exemple de création de listes HTML *à partir d'un tableau PHP (2)*

1ère liste déroulante : *indéxée par un entier (k) et conditionnant la seconde*

```
<?php   echo ("<select  name=\'groupe_pays\'
                onchange=\'Lespays.selectedIndex=LG[this.selectedIndex] \'> ");
        $k=0;    foreach ($pays as $cle => $valeur)
                    echo ("<option value=\'" . $k++ . "\'>  $cle  </option> \n");
        echo  "</select>";
```

Seconde liste déroulante :

```
$k=0;      echo ("<select  name=\'Lespays\'> \n");
           foreach ($pays as $tab)
                    foreach ($tab as $cle2 => $valeur2)
                      echo "<option  value=\'" . $cle2. "\'> $valeur2 </option>\n";
           echo ("</select>");
?>
```