

Université Sidi Mohamed Ben Abdllah

Faculté des sciences Dhar El Mahraz
Licence professionnelle SIGL

Web dynamique: PHP

Réalisé par: Pr. Mahraz Med Adnane

Année universitaire:2015/2016

PHP: Langage de script pour le Web

- Qu'est-ce que PHP ?
 - Langage de **script**. Utilisé **coté serveur**
 - Acronyme récursif : **P**HP: **H**ypertext **P**reprocessor
 - Créé en 1994-1995 par Rasmus Lerdorf
 - Extension utilisée sur plusieurs serveurs Web
 - Langage multi plate-forme (UNIX / Windows...)
 - Open Source

Utilité et utilisation de PHP

- Création de pages HTML « dynamiques », fabriquées à la volée, construite à la demande
- Interface entre un serveur Web et des bases de données
- Création d'applications Web

Programme en PHP

- Les scripts PHP sont généralement intégrés dans le code d'un document HTML
- L'intégration nécessite l'utilisation de balises
 - avec le style xml : `<?PHP ?>` ligne de code
 - Avec le style php: `<?php PHP ?>` ligne de code
 - avec le style JavaScript :
`<script language=«php» PHP </script>` ligne de code
 - avec le style des ASP : `<% code ASP %>` ligne de

Éléments de syntaxe PHP

- La syntaxe de PHP ressemble à celle de famille "C" (C, C++, Java, ...)
- Chaque instruction se termine par ";"
- Commentaires:
 - `/* jusqu'au prochain */`
 - `// jusqu'à la fin de la ligne`
 - `# jusqu'à la fin de la ligne`

Les constantes

```
<?php
```

```
define("ma_const", "Bonjour");
```

nom

valeur

Définition d'une constante

```
echo ma_const ;
```

```
?>
```

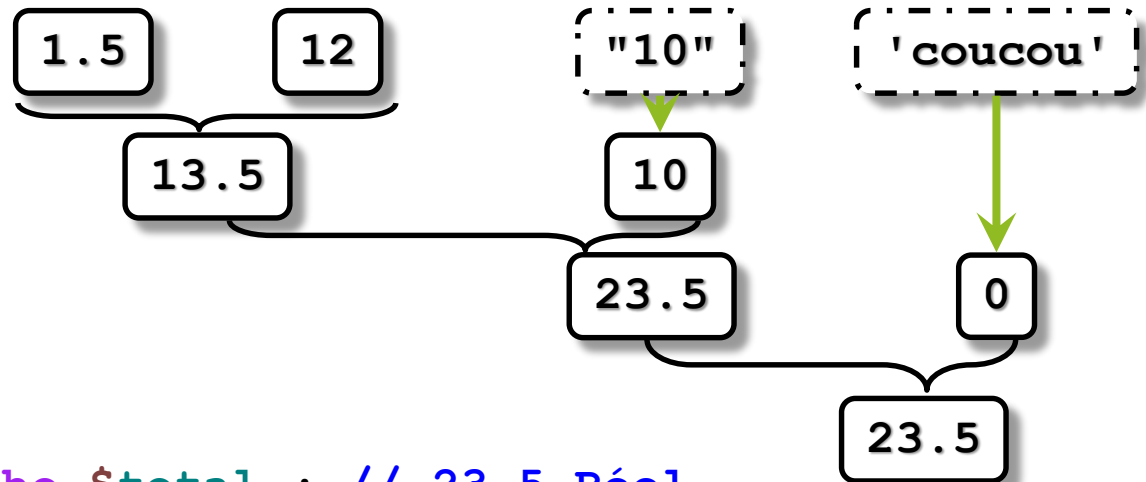
Utilisation de la constante

Les variables et les types de données

- Tout identificateur commence par "\$"
- Les affectations sont réalisées grâce à "="
 - Numérique entier: 12 ou réel: 1.54
 - Chaîne: "Hello" ou 'Bonjour'
 - Booléen: true, false (PHP 4)
 - Tableau: \$tab[2]=12
 - Objet (PHP4, PHP5)
 - NULL
- Le type d'une variable est dynamique et est déterminé par la valeur qui lui est affectée

Typage automatique : Exemple

```
$nombre1 = 1.5 ;           // Réel
$nombre2 = 12 ;            // Entier
$chaine1 = "10" ;          // Chaîne
$chaine2 = 'coucou' ;      // Chaîne
$total =
$nombre1 + $nombre2 + $chaine1 + $chaine2;
```



```
echo $total ; // 23.5 Réel
```


Références

```
$a = 12 ;
```

```
$b = $a ;
```

```
$c = &$a ;
```

```
$b = "bonjour" ;
```

```
$c = 84 ;
```

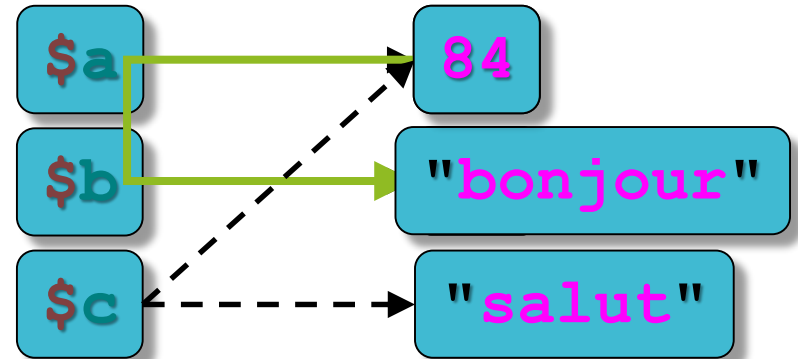
```
$a : 84
```

```
$b : coucou
```

```
$c : 84
```

```
unset($c) ;
```

```
$c = "salut" ;
```



Les chaînes de caractères

Substitution de variables dans les chaînes

- Guillemets simples

- `$a='chaîne' ;`
- `$b='voici une $a' ;`

chaîne

voici une \$a

- Guillemets doubles

- `$a="chaîne" ;`
- `$b="voici une $a" ;`

chaîne

voici une chaîne

- Syntaxe *HereDoc*

- `$a="chaîne" ;`
- `$b=<<<MARQUE_DE_FIN
voici une $a
sur deux lignes ;-)
MARQUE_DE_FIN;`

chaîne

voici une chaîne
sur deux lignes ;-)

Concaténation de chaînes

- Permet d'assembler plusieurs chaînes
- Réalisé grâce à l'opérateur point : .

```
"Bonjour " . "Marcel"
```

```
→ "Bonjour Marcel"
```

```
$nb = 6*2 ;
```

```
"Acheter " . $nb . " oeufs"
```

```
→ "Acheter 12 oeufs"
```

Envoi du code HTML par PHP

- La fonction echo : **echo**
Expression;
 - echo "Chaine de caracteres";
 - echo (1+2)*87;
- La fonction print :
print(expression);
 - print("Chaine de caracteres");
 - print ((1+2)*87);
- La fonction printf : **printf (chaîne
formatée);**
 - printf ("Le périmètre du cercle est
%d",\$Perimetre);

Les opérateurs de comparaison

<code>\$a == \$b</code>	Vrai si égalité entre les valeurs de <code>\$a</code> et <code>\$b</code>
<code>\$a != \$b</code>	Vrai si différence entre les valeurs de <code>\$a</code> et <code>\$b</code>
<code>\$a < \$b</code>	Vrai si <code>\$a</code> inférieur à <code>\$b</code>
<code>\$a > \$b</code>	Vrai si <code>\$a</code> supérieur à <code>\$b</code>
<code>\$a <= \$b</code>	Vrai si <code>\$a</code> inférieur ou égal à <code>\$b</code>
<code>\$a >= \$b</code>	Vrai si <code>\$a</code> supérieur ou égal à <code>\$b</code>
<code>\$a === \$b</code>	Vrai si <code>\$a</code> et <code>\$b</code> identiques (valeur et type)
<code>\$a !== \$b</code>	Vrai si <code>\$a</code> et <code>\$b</code> différents (valeur ou type)

Structure de
contrôle
Si...Alors...Si
non...

```
if (condition)
{
    /* Bloc d'instructions
    exécuté si la condition est
    vraie */
}
[else
{
    /* Bloc d'instructions
    exécuté si la condition est
    fausse */
}]
```

Structure de contrôle boucles

```
for(initialisation; condition;  
    incrémentation)  
{ /* Bloc d'instructions répété tant  
    que la condition est vraie */  
}
```

```
while (condition)  
{  
    /* Bloc d'instructions répété tant  
    que la condition est vraie */  
}
```

```
do {  
    /* Bloc d'instructions exécuté une  
    fois puis répété tant que la  
    condition est vraie */  
} while (condition) ;
```

Structure de contrôle switch...

```
switch (val)
{
    case v1:
        instructions exécutées si
        val==v1
    case v2:
        instructions exécutées si
        val==v2
        ou
        si val==v1
    ...
    default:
        instructions dans tous les cas
}
```


Les tableaux

- Création à l'aide de la fonction `array()`
- Uniquement des tableaux à une dimension
- Les éléments d'un tableau peuvent appartenir à des types distincts
- L'index d'un tableau en PHP commence de 0
- Pas de limites supérieures pour les tableaux
- La fonction `count()` pour avoir le nombre d'éléments d'un tableau

Les tableaux

Clé	Valeur
0	12
1	"fraise"
2	2.5

- Création / initialisation:

```
$tab1=array(12, "fraise", 2.5);
```

```
$tab2[] = 12;
```

```
$tab2[] = "fraise";
```

```
$tab2[] = 2.5;
```

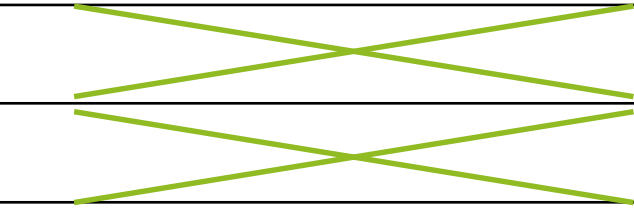
```
$tab3[0] = 12;
```

```
$tab3[1] = "fraise";
```

```
$tab3[2] = 2.5;
```

Les tableaux « à trous »

Clé	Valeur
0	12
1	"fraise"
2	2.5
3	
4	
5	"e15"



Comment parcourir de tels tableaux ?

Les tableaux « à trous » (suite)

Parcours classique :

```
$tab4[0] = 12 ;  
$tab4[1] = "fraise";  
$tab4[2] = 2.5 ;  
$tab4[5] = "e15" ;
```

```
for ($i=0; $i <  
count($tab4);  
$i++){  
echo "tab4[$i]:  
". $tab4[$i] .  
"<BR>\n";  
}
```

Parcours pour chaque élément :

```
foreach ($tableau  
as $element){
```

```
/* Chaque élément de  
$tableau est  
accessible grâce à  
$element */
```

```
}
```

```
foreach($tab4 as $v)  
{  
    echo "Val: $v<br>\n";  
}
```

Tableaux associatifs

- Tableaux dont l'accès aux éléments n'est plus réalisé grâce à un index (0,1,...) mais grâce à une clé de type entier ou chaîne.

Création

```
$tab = array(cle1 => val1, cle2 => val2, ...);
```

Exemples de clés:

```
$tab['un'] = 12 ;
```

```
$tab[205] = "bonjour" ;
```

```
$tab["la valeur"] = 3.0 ;
```

Tableaux associatifs - Exemples

Clé	Valeur
"un"	12
"trois"	"fraise"
"deux"	2.5
42	"e15"

```
$tab5['un']      = 12 ;  
$tab5['trois']   = "fraise" ;  
$tab5["deux"]   = 2.5 ;  
$tab5[42]       = "e15" ;
```

```
$tab6 = array('un'      => 12,  
              'trois'   => "fraise",  
              "deux"    => 2.5,  
              42        => "e15") ;
```

Structure de contrôle Pour chaque...

```
foreach($tableau as $cle =>
    $element)
{
    /* Bloc d'instructions répété pour
    chaque élément de $tableau */

    /* Chaque élément de $tableau est
    accessible grâce à $element */

    /* La clé d'accès à chaque élément
    est donnée par $cle */
}
```

Parcours de tableau

```
<html>
  <head><title>foreach clé</title>
  </head>
  <body>
    <?php
      $tableau = array('un'      => 12,
                       'deux'    => "Lundi",
                       "trois"   => 2.5,
                       "quatre" => "Bonjour") ;

      foreach ($tableau as $cle => $val) {
        echo "tableau[$cle]= $val<br>" ;
      }
    ?>
  </body>
</html>
```


Fonctions utilisateur

```
function moyenne ($a, $b)  
{  
    return ($a+$b)/2;  
}
```

- Utilisation

```
$resultat = moyenne(2,4) ;  
echo $resultat ; // vaut 3
```

Mode de passage des arguments (types natifs)

```
<?php
```

```
function permutation($x, $y) {
```

```
    $t = $x ;
```

```
    $x = $y ;
```

```
    $y = $t ;
```

```
?>
```

```
$a = 12
```

```
$b = 210
```

```
permutation($a,$b) ;
```

```
$a = 12
```

```
$b = 210
```

Permutation impossible :
Passage des arguments des
fonctions par valeur

Mode de passage des arguments (types natifs)

```
<?php
```

```
function permutation(&$x, &$y) {
```

```
    $t = $x ;
```

```
    $x = $y ;
```

```
    $y = $t ;
```

```
?>
```

Permutation réussie

```
$a = 12
```

```
$b = 210
```

```
permutation($a,$b) ;
```

```
$a = 210
```

```
$b = 12
```

Arguments par défaut des fonctions

- Valeur par défaut d'un argument s'il n'a pas été défini lors de l'appel de la fonction

```
function bonjour($nom="inconnu") {  
echo "Bonjour cher $nom" ;  
}
```

- Utilisation

```
bonjour() ;
```

Bonjour cher inconnu

```
bonjour("Marcel") ;
```

Bonjour cher Marcel

Fonctions prédéfinies: String

```
$phrase = 'Bonjour tout le monde ! Je suis une  
phrase !';
```

- `$longueur = strlen($phrase);` //strlen : longueur
d'une chaîne
- `$c = str_replace('o', 'y', $phrase);` //str_replace :
rechercher et remplacer
- `$ phrase = strtolower($ phrase);`//strtolower :
écrire en minuscules
- `$ phrase = strtoupper($ phrase);`// strtoupper :
écrire en majuscules

Fonctions prédéfinies: date

Paramètre	Description
H	Heure
i	Minute
d	Jour
m	Mois
Y	Année

```
<?php
$jour = date('d');
$mois = date('m');
$annee = date('Y');
$heure = date('H');
$minute = date('i');

echo 'Bonjour ! Nous sommes le ' . $jour . '/' .
$mois . '/' . $annee . 'et il est ' . $heure . ' h ' .
$minute;
?>
```

Définition de fonctions fréquemment utilisées

- Certaines fonctions sont utilisées dans plusieurs scripts PHP
- Comment faire pour ne pas les définir dans chacune des pages ?
- Utilisation de :
 - `include("fichier") ;`
 - `require("fichier") ;`
 - `include_once("fichier") ;`
 - `require_once("fichier") ;`
- Permet d'inclure le contenu de **fichier** dans le script courant

Fichier mafunction.php

```
<?
function mafunction($arg)
{
    if (isset($arg))
    {
        echo ("Vrai") ;
    }
    else
    {
        echo ("Faux") ;
    }
}
?>
```

Fichier utilisation1.php

```
...
require("mafunction.php")
mafunction(true) ;
...
```

Fichier utilisation2.php

```
...
include("mafunction.php")
...
$var=false ;
mafunction($var) ;
...
```

Fichier utilisation3.php

```
...
require("mafunction.php")
...
```


Traitement des données par PHP

- PHP permet de **traiter les données** saisies grâce à un formulaire HTML si le champ **ACTION** du formulaire désigne une page PHP du serveur.
- Après récupération par le serveur Web, les données sont contenues dans l'une des variables superglobales de type tableau associatif **\$_GET** ou **\$_POST**.
- La valeur peut être trouvée grâce à une clé **qui porte le même nom** que le champs du formulaire de la page HTML de saisie.

Méthodes d'envoi get et post

1. La méthode GET place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.

<http://www.site.com/cible.php?champ1=valeur&champ2=valeur>

- inconvénients :
 - rendre visibles les données dans la barre d'adresse du navigateur.
 - De plus, la longueur totale est limitée à 255 caractères, ce qui rend impossible la transmission d'un volume de données important

2. La méthode POST regroupe les informations dans l'entête d'une requête HTTP => Assure une confidentialité efficace des données

PHP et les formulaires

- Formulaire HTML
 - Structure : un formulaire commence toujours par la balise `<form>` et se termine par la balise `</form>`
 - Champ de saisie de text en ligne :
`<input type = "text" name ="nom_du_champ" value="chaîne">`
 - Boutons d'envoi et d'effacement :
`<input type=" submit " value = "Envoyer">`
`<input type = "reset" name ="efface" value = "Effacer">`
 - Case à cocher et bouton radio :
`<input type = "checkbox" name ="case1" value="valeur_case">`
`<input type = "radio" name ="radio1" value ="valeur_radio">`

PHP et les formulaire

- Liste de sélection avec options à choix unique :

```
<select name = "select" size = "1" >  
  <option value = "un"> choix1 </option>  
  <option value = "deux"> choix2 </option>  
</select>
```

- Liste de sélection avec options à choix multiples :

```
<select name = "select" size = "1" multiple>  
  <option value = "un"> choix1 </option>  
  <option value = "deux"> choix2 </option>  
</select>
```

Exemple – Formulaire HTML

```
<html>
<head>
  <title>formulaire</title>
</head>
<body>
  <form action="valider.php"
    method="get">
    <label>Nom: <input type="text"
      name="nomPers"></label>
    <input type="submit"
      value="Envoyer">
  </form>
</body>
</html>
```

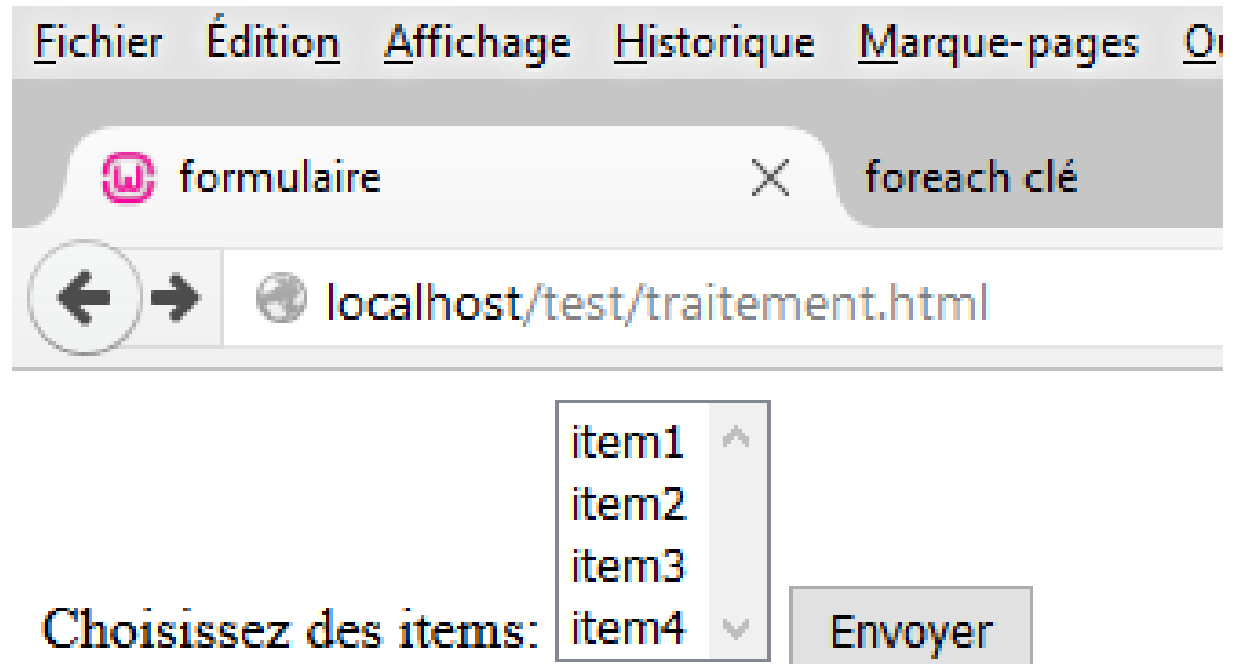
Exemple – Traitement en PHP

```
<html>
<head>
  <title>Validation</title>
</head>
<?php
  if (isset($_GET['nomPers']))
  {
    if (!empty($_GET['nomPers']))
    {
      echo "Vous avez saisi '"
        . $_GET['nomPers'];
    }
    else
      echo "Aucune valeur saisie";
  }
  else
    echo "Utilisation incorrecte" ;
?>
</body></html>
```

`$_GET['nomPers']`
est-il défini ?

`$_GET['nomPers']`
est-il vide ?

Formulaires contenant des champs « SELECT »



The screenshot shows a web browser window with the following elements:

- Menu Bar:** Fichier, Édition, Affichage, Historique, Marque-pages, Outils.
- Tab Bar:** Two tabs are visible: 'formulaire' (active) and 'foreach clé'.
- Address Bar:** The URL is 'localhost/test/traitement.html'.
- Form Content:**
 - The text 'Choisissez des items:' is followed by a dropdown menu.
 - The dropdown menu is open, showing four options: 'item1', 'item2', 'item3', and 'item4'.
 - To the right of the dropdown is a button labeled 'Envoyer'.

Formulaires contenant des champs « SELECT uni que »

```
<!DOCTYPE html>
<html>
<head>
  <title>Formulaire de saisie des
  items</title>
</head>
<body>
<form action="valider.php" method="get">
  <label for="liste">Choisissez des items:</label>
  <select name="list" id="liste">
    <option name="l1">item1</option>
    <option name="l2">item2</option>
    <option name="l3">item3</option>
    <option name="l4">item4</option>
  </select>
  <input type="submit" value="envoyer">
</form>
</body>
</html>
```

Envoyer

valider.php?list=item1

Formulaires contenant des champs «SELECT multiple»

```
<!DOCTYPE html>
<html>
<head>
  <title>Formulaire de saisie des
  items</title>
</head>
<body>
<form action="valider.php" method="get">
  <label for="liste">Choisissez des items:</label>
  <select name="list" multiple id="liste">
    <option name="l1">item1</option>
    <option name="l2">item2</option>
    <option name="l3">item3</option>
    <option name="l4">item4</option>
  </select>
  <input type="submit" value="envoyer">
</form>
</body>
```

Envoyer

valider.php?list=item1&list=item3

???

Formulaires contenant des champs «SELECT multiple»

Envoyer

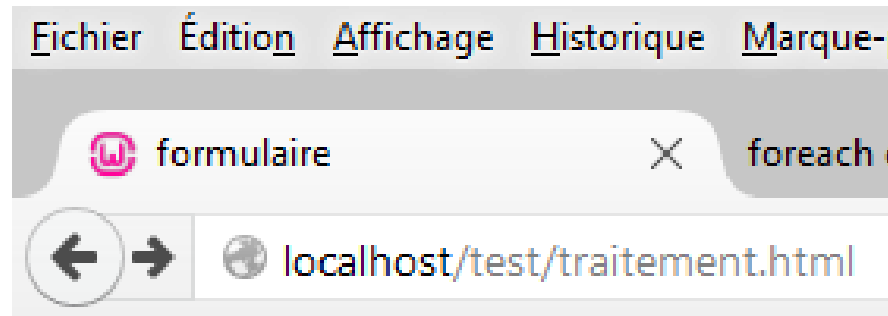
```
<!DOCTYPE html>
<html>
<head>
  <title>Formulaire de saisie des
  items</title>
</head>
<body>
<form action="valider.php" method="get">
  <label for="liste">Choisissez des items:</label>
  <select name="list[]" multiple
  id="liste"> <option
    name="l1">item1</option>
    <option name="l2">item2</option>
    <option name="l3">item3</option>
    <option name="l4">item4</option>
  </select>
  <input type="submit" value="envoyer">
</form>
</body>
valider.php?list[]=item1&list[]=item3
```

Traitement des données des champs « SELECT »

`$_GET['list']`
est un tableau

```
<!DOCTYPE html>
<html>
<head>
  <title>Liste de items</title>
</head>
<body>
<?php
  if (isset($_GET['list']) &&
      !empty($_GET['list']))
  {
    foreach($_GET['list'] as $items)
      echo "Vous avez choisi
      $items<br>" ;
  }
  else
    echo "Vous n'avez pas choisi
    d'item\n" ;
?>
</body>
</html>
```

Formulaires contenant des champs « CHECKBO X »



Choisissez des fruits

- ☐ Fraise
- ☒ Pomme
- ☐ Poire
- ☒ Banane
- ☐ Cerise

Envoyer

Formulaires contenant des champs «CHECKBOX»

```
<!DOCTYPE html>
<html>
<head>
  <title>Formulaire de saisie des fruits</title>
</head>
<body>
  <form name="formulaire" action="valider.php" method="post">
    <label>Choisissez des fruits</label><br>
    <label><input type="checkbox" name="list[]"
    value="Fraise">Fraise</label><br>
    <label><input type="checkbox" name="list[]" value="Pomme"
    >Pomme </label><br>
    <label><input type="checkbox" name="list[]" value="Poire"
    >Poire </label><br>
    <label><input type="checkbox" name="list[]"
    value="Banane">Banane</label><br>
    <label><input type="checkbox" name="list[]"
    value="Cerise">Cerise</label><br>
    <input type="submit" value="Envoyer">
  </form>
</body>
</html>
```

Traitement des données des champs « SELECT »

```
<!DOCTYPE html>
<html>
<head>
  <title>Liste des fruits</title>
</head>
<body>
<?php
  if (isset($_POST['list']) && !empty($_POST['list']))
  {
    foreach($_POST['list'] as $fruit)
      echo "Vous avez choisi $fruit <br>" ;
  }
  else
    echo "Vous n'avez pas choisi de fruit\n" ;
?>
</body>
</html>
```