GRETA VIVA5

Cours PHP

Qu'est-ce que PHP ?

- 1994 :Rasmus Lerdorf a conçu un langage de script interfaçant le HTML et une base de données par l'intermédiaire de requêtes SQL.
- 1995 : PHP 2 intègre des boucles et autres structures de contrôle ainsi que l'interface avec les formulaires HTML et l'interface avec MySQL. Cette version se diffuse rapidement dans le monde entier.
  C'est un langage de script exécuté par le serveur Web (comme les CGI, ASP, JSP…) contrairement au JavaScript ou aux applets Java qui s'exécutent directement sur le navigateur client.
  La syntaxe et l'esprit du langage PHP s'inscrivent dans la lignée du langage C et du Perl.
- 1998 : Zeev Suraski et Andi Gurmans s'associent à Rasmus Lerdorf pour développer la version 3.0 de PHP, désormais Pre Hypertext Processor
- 2000 : PHP4 introduit la programmation objet, les sessions, ...
- 2004 devrait voir le jour de PHP5 avec une programmation objet proche de JAVA.

Comparatifs

- Avantages :
  - logiciel libre :PHP est distribué sous licence GNU GPL, ainsi que Apache et MySQL
  - multi-plateforme.
  - simplicité (apparente ?) d'écriture des scripts
  - facilité d'écriture de pages dynamiques par rapport aux CGI comme Perl,
  - nombreuses bibliothèques :en particulier, de nombreux systèmes de base de données sont supportés
- Inconvénients :
  - un langage "fourre-tout", mélange de nombreuses facilités voire astuces de programmation
  - ne favorise pas la conception entre couche présentation, couche métier et couche base de données (relativement à JSP de SUN)
  - plusieurs versions, avec des différences "désagréables"
  - des scripts facilement illisible, conséquence des points précédents
- ses concurrents :
  - ASP : de Microsoft pour machine ..... Microsoft
  - JSP : de SUN (écriture en JAVA, grande modularité)

Installation
et premiers scripts PHP

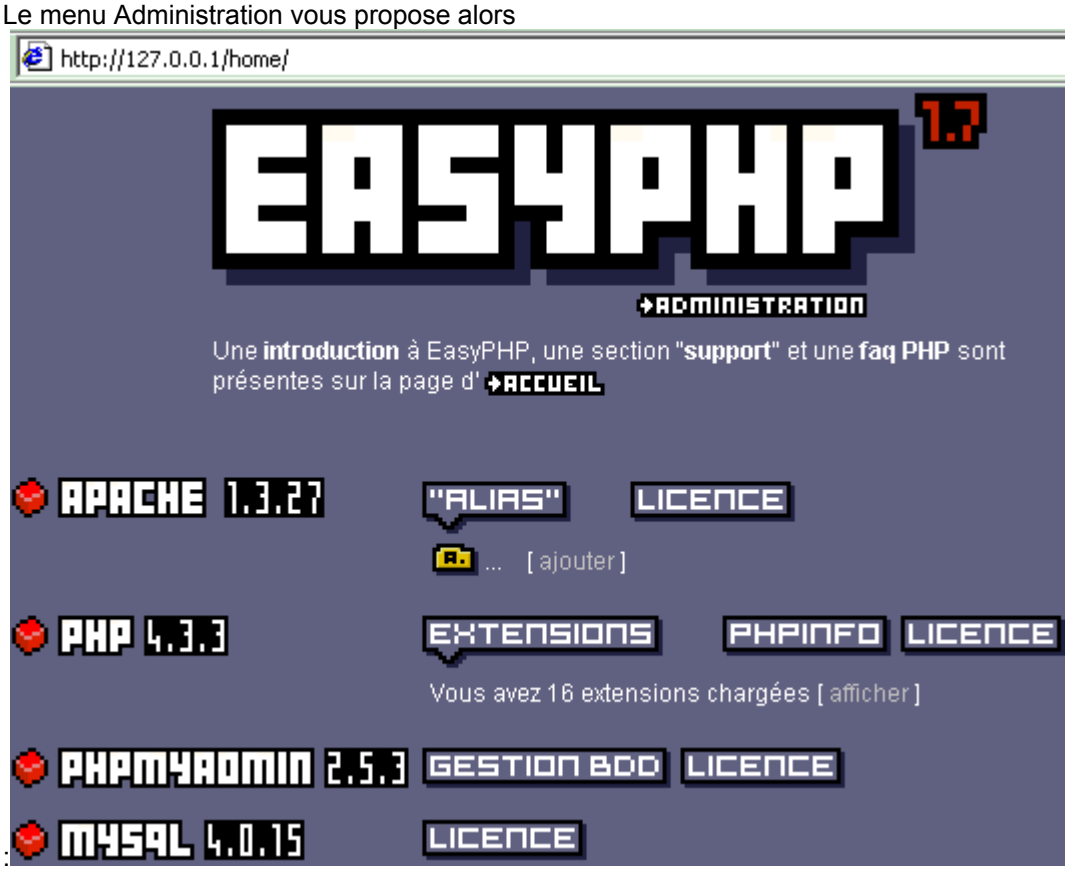
installation sous Windows

- EasyPHP est un package téléchargeable sur le site www.easyphp.org qui comprend :

  - un serveur web Apache,
  - un moteur de script PHP,
  - un système de gestion de base de données MySQL,
    et son interface PhpMyAdmin qui gère le SGBD facilement (interface écrite elle-même en PHP !!)

- installation : exécuter le fichier d'installation
- démarrer : via le menu Démarrer de windows
- gérer/arrêter : dans la barre des taches, cliquez avec votre bouton droit de souris sur l'icône de gestion EasyPHP :

- Le menu Administration vous propose alors



  - l'ajout de répertoire de pages web proposée par les serveur Apache
  - et la gestion du SGBD MySQL,
  - ...

- Enregistrer ses scripts sur un répertoire racine de site web :

  - par défaut, ils sont à mettre dans C:\Program Files\EasyPHP1-7\www
  - j'y crée un répertoire mon_site
  - puis y dépose mon premier script hello.php

- Exécuter un script PHP ou une page HTML : cliquez sur le menu Web Local puis cliquez sur les répertoires et fichiers à visualiser dans votre navigateur.

un premier script

```
<html>
 <head>
 </head>
 <body>
  <p>
   <?php echo "hello word ! <br> bonjour monde !" ?>
  </p>
 </body>
</html>
```

- EXECUTION

  hello word !
  bonjour monde !

- **le script :**

  - le code PHP inclus dans la page HTML est exécuté : ici, afficher le texte "hello world .....

- echo chaîne;
  affiche la chaîne de caractères dans la page (dans le flot) qui sera envoyé par la serveur
  ce n'est pas une fonction : echo(...); ne fonctionne pas
  echo chaine1, chaine2, ... ; affiche les chaînes

- **Interprétation du PHP :**

  - La page HTML inclus du code PHP afin d'exécuter des actions : ici, afficher du texte "hello world .....
  - Le code PHP est inclus entre une balise de début <?php et une balise de fin ?> qui indique au serveur web d'interpréter le contenu par le "moteur" PHP et non de l'interpréter comme du HTML.
  - Le code PHP est exécuté sur le serveur (contrairement à JavaScript) :
    si vous demandez le source du fichier visualisé dans votre navigateur, vous n'obtenez que du HTML :

```
<html>
  <head>
  </head>
  <body>
    <p>
      hello word ! <br> bonjour monde !
    </p>
  </body>
</html>
```

Les balises de code PHP

Il existe plusieurs façons d'introduire du code PHP dans du HTML

```
<?php  code PHP ?>
la meilleure pour le HTML, XML, XHTML <script language="php">  code PHP </script>
<? code PHP ?>
<?= blablaPHP ?>
équivalent à <? echo blablaPHP ; ?>
<% code PHP %>
<%= blablaPHP %>
```

Second script

```
<html>
  <head>
  </head>
  <body>
    <p>
      <?php /* commentaire de
              2 lignes */
        echo "hello word !", "<br>" ;
        echo "bonjour" ; // commentaire de fin de ligne
        echo " monde !" # commentaire de fin de ligne
      ?>
    </p>
  </body>
</html>
```

- EXECUTION

  hello word !
  bonjour monde !

- **commentaires :**
  - /* est le commentaire classique */
  - // et # sont des commentaires dont la fin est la fin de ligne
- **séparateur d'instructions :**
  - le ; sépare les instructions PHP

- le tag de fin de php ?> termine la dernière instruction
  donc le dernier ; peut être omis.

Variables et types

variable

```
<html>
 <head>
 </head>
 <body>
  <p>
   <?php
    $a = 3;
            $b = 2;
            $c = $a + $b;
            echo "\$c = $c" ;
   ?>
  </p>
 </body>
</html>
```

- EXECUTION

  $c =5

- Un **identificateur** (nom) **de variable** doit obligatoirement :

  - être précédé du caractère dollar ($)
  - doit commencer par une lettre (majuscule ou minuscule) ou un "_" (pas par un chiffre)
  - peut comporter des lettres, des chiffres et le caractère _ , et les caractères accentués
  - sont sensibles à la casse (PHP fait la différence entre majuscules et minuscules)
  - bonne variable : $bonneVariable $bonnevariable $bonne_variable $_bonnevariable $b03_v $bonnéevar
  - mauvaise : mauvaise $2mauvaise $mauvaise-variable

pas de déclaration de variable !

```
<html>
 <head>
 </head>
 <body>
  <p>
   <?php
    $a = 3; echo "\$a = $a<br>" ;
    $a = 3.3; echo "\$a = $a<br>" ;
    $a = "3a"; echo "\$a = $a<br>" ;
            echo "\$b = $b<br>" ;
   ?>
  </p>
 </body>
</html>
```

- EXECUTION

  $a = 3
  $a = 3.3
  $a = 3a
  $b =

- les variables en PHP n'ont **pas besoin d'être déclarées**,

  - on peut commencer à les utiliser sans en avoir averti l'interpréteur au préalable !

- si la variable existait précédemment, son contenu est utilisé,
- sinon l'interpréteur lui affectera la valeur en lui assignant NULL par défaut
- Il n'est pas nécessaire en PHP de typer les variables, c'est-à-dire de définir leur type, il suffit de leur assigner une valeur pour en définir le type

les types de valeurs scalaires

```
<?php
  $a = 3; echo "valeur $a type : " ;
     echo gettype($a); echo "<br>" ;
  $a = 3.3; echo "valeur $a type : " ;
     echo gettype($a); echo "<br>" ;
  $a = "3a"; echo "valeur $a type : " ;
     echo gettype($a); echo "<br>" ;
  $a = TRUE; echo "valeur TRUE type : " ;
     echo gettype($a); echo "<br>" ;
  echo "sans valeur type : " ;
     echo gettype($b); echo "<br>" ;
?>
```

- EXECUTION

  valeur 3 type : integer
  valeur 3.3 type : double
  valeur 3a type : string
  valeur TRUE type : boolean
  sans valeur type : NULL

- les types scalaires sont
  - Booléen (PHP4 !)
  - Entiers : nombre sans virgule.
  - Réels : nombres avec une virgule, flottant ou réel
  - Chaînes de caractères : ensembles de caractères entre guillemets simples ou doubles.
  - la valeur NULL (PHP4 !)

variable affectée ou non

```
<?php
  echo "variable non declaree isset : " ; echo isset($b); echo "<br>" ;
  $b = NULL;
  echo "variable NULL isset : " ; echo isset($b); echo "<br>" ;
  $b = 13;
  echo "variable a valeur $b isset : " ; echo isset($b); echo "<br>" ;
  unset($b);
  echo "apres unset(\$b) isset : " ; echo isset($b); echo "<br>" ;
?>
```

- EXECUTION

  variable non declaree isset :
  variable NULL isset :
  variable a valeur 13 isset : 1
  apres unset($b) isset :

- isset(variable)
  détermine si une variable est affectée/définie
  renvoie TRUE , sinon FALSE
- unset(variable)
  détruit une variable
  - unset() à l'intérieur d'une fonction n'a pas la même portée de destruction suivant le type de variable

- var_dump(variable)
  affiche toutes les informations d'une variable
  utile pour débuguer !

types scalaires

booléen

- PHP4 : 2 valeurs TRUE et FALSE
  insensible à la casse : True, tRue, trUE et true sont TRUE
- 0, 0.0, "", "0", NULL sont interprétés comme FALSE
  toutes les autres valeurs comme TRUE
  -1 est true
- conversion :
  - en entier : TRUE convertit en entier donne 1 et FALSE 0
  - en chaine : true est "1" et false est ""

entier

```
<?php
  $a = 123; echo "notation 123 valeur $a <br>" ;
  $a = -123; echo "notation -123 valeur $a <br>" ;
  $a = 0123; echo "notation 0123 valeur $a <br>" ;
  $a = 0x1A; echo "notation 0x1A valeur $a <br>" ;
?>
```

- EXECUTION

  notation 123 valeur 123
  notation -123 valeur -123
  notation 0123 valeur 83
  notation 0x1A valeur 26

- leur taille dépend de la machine ...
  en général sur 32 bits, c'est à dire de environ -2E6 à + 2E6
- conversion :
  - d'un booléen en entier :TRUE convertit en entier donne 1 et FALSE 0
  - d'un réel : "arrondi vers 0", c'est à dire -2.6 devient 2 et +2.- devient 2

flottant, réel

```
<?php
  $a = 12.3; echo "notation 12.3 valeur $a <br>" ;
  $a = 0.123E2; echo "notation 0.123E2 valeur $a <br>" ;
  $a = -123e-1; echo "notation -123e-1 valeur $a <br>" ;
?>
```

- EXECUTION

  notation 12.3 valeur 12.3
  notation 0.123E2 valeur 12.3
  notation -123e-1 valeur -12.3

- leur taille dépend de la machine ...

cast, conversion explicite de type

```
<?php
```

```
 $f = 1.3;
 $s = "2.6";
 echo '(int)$f : '; $r = (int)$f; echo "$r <br>";
 echo '(float)$s+1.1 : '; $r = (float)$s + 1.1 ;echo "$r <br>";
?>
```

- EXECUTION

  ```
  (int)$f : 1
  (float)$s+1.1
  : 3.7
  ```

- La conversion se fait comme en C avec l'opérateur de transtypage : (type)

chaînes de caractères

```
<?php
 $a = "abc";
 echo "entre guillemets : $a <br>" ;
 echo 'entre quotes : $a <br>';
 echo "combinaison de 'quotes' guillemets<br>";
 echo 'combinaison de quotes "guillemets"<br>';
 echo 'j\'ai dit<br>';
 echo "j\"ai dit<br>";
 echo "le passage \n a la ligne (idiot en HTML !)<br>";
 echo "le backslash \\ lui-même<br>";
 echo "et le dollar \$";
?>
```

- EXECUTION

  ```
  entre guillemets : abc
  entre quotes : $a
  combinaison de 'quotes' guillemets
  combinaison de quotes "guillemets"
  j'ai dit
  j"ai dit
  le passage a la ligne (idiot en HTML !)
  le backslash \ lui-même
  et le dollar $
  ```

- **string :**
  - chaîne de caractères sur 8 bits ASCII et accentuées
    mais pas d'Unicode !
  - Les caractères qui composent une chaîne sont indexés à partir de la position début à 0
  -  il n'y a pas de type caractère : il faut se débrouiller avec des strings d'une lettre
  - les valeurs littérales chaînes :
    - les chaînes entre quotes (apostrophe) sont constantes : seul le \' est interprété comme ne terminant pas le string
    - les chaînes entre guillemets sont interprétées comme suit :
      - $var et {$var} et ${var} sont remplacés par la valeur de la variable,
      - \ déspécialise les caractères : \, $, ", et \ lui-même
      - \n est le passage à la ligne, \t la tabulation, \0octal et \Oxhexadécimal précise un caractère par sa valeur

le type d'une variable

```
<?php
 echo 'is_bool(-4.2) : '; echo is_bool(-4.2); echo "<br>";
```

```
 echo 'is_bool(true) : '; echo is_bool(true); echo "<br>";
 echo 'is_int(-4.2) : '; echo is_int(-4.2); echo "<br>";
 echo 'is_int(-4) : '; echo is_int(-4); echo "<br>";
 echo 'is_int("abc") : '; echo is_int("abc"); echo "<br>";
 echo 'is_float(-4.2) : '; echo is_float(-4.2); echo "<br>";
 echo 'is_float(-4) : '; echo is_float(-4); echo "<br>";
 echo 'is_string("abc") : '; echo is_string("abc"); echo "<br>";
 echo 'is_string(true) : '; echo is_string(true); echo "<br>";
?>
```

- EXECUTION

  ```
  is_bool(-4.2) :
  is_bool(true) : 1
  is_int(-4.2) :
  is_int(-4) : 1
  is_int("abc") :
  is_float(-4.2) : 1
  is_float(-4) :
  is_string("abc") : 1
  is_string(true) :
  ```

- fonctions booléennes pour déterminer le type d'une variable :
  - is_int(variable)
  - is_bool(variable)
  - is_string(variable)
  - is_float(variable)

- gettype(variable)
  retourne sous forme d'une chaine le type de la variable :
  "integer", "boolean", "double" (et non float !!!), "string", "null", "array", "object", "ressource"

conversion d'un string en type nombre

```
<?php
 $a = "10"; $s = 2 + $a; echo "string \"$a\" resultat $s <br>" ;
 $a = "10.3"; $s = 2 + $a; echo "string \"$a\" resultat $s <br>" ;
 $a = "1.03e1"; $s = 2 + $a; echo "string \"$a\" resultat $s <br>" ;
 $a = "-10"; $s = 2 + $a; echo "string \"$a\" resultat $s <br>" ;
 $a = "10abc"; $s = 2 + $a; echo "string \"$a\" resultat $s <br>" ;
 $a = "abc 10"; $s = 2 + $a; echo "string \"$a\" resultat $s <br>" ;
 $a = "10.3abc"; $s = 2 + $a; echo "string \"$a\" resultat $s <br>" ;
?>
```

- EXECUTION

  ```
  string "10" resultat 12
  string "10.3" resultat 12.3
  string "1.03e1" resultat 12.3
  string "-10" resultat -8
  string "10abc" resultat 12
  string "abc 10" resultat 2
  string "10.3abc" resultat 12.3
  ```

- lorsqu'une valeur numérique est attendue, la conversion se fait en prenant le début "numérique" de la chaîne :
  - s'il y a un . ou un e ou E dedans, alors ce sera un float
  - sinon un entier
  - s'il n'y a pas de chiffres, alors c'est la valeur 0

Récupérer les valeurs
envoyées d'un formulaire HTML

formulaire

```
<html>
 <head>
  <title>formulaire à soumettre</title>
 </head>
 <body>
  <h2 align=center> formulaire à soumettre</h2>
  <p>
   Renseigner le chien :<br>
   <form method="post" action="submit1.php">
    Nom : <input type="text" name="nom"><br>
    Maitre : <input type="text" name="maitre"><br>
    Aboiement : <input type="text" name="aboiement"><br>
    Nombre de puces : <input type="text" name="nombrePuces"><br>
    <input type="submit" name="submit" value="Soumettre">
   </form>
  </p>
 </body>
</html>
```

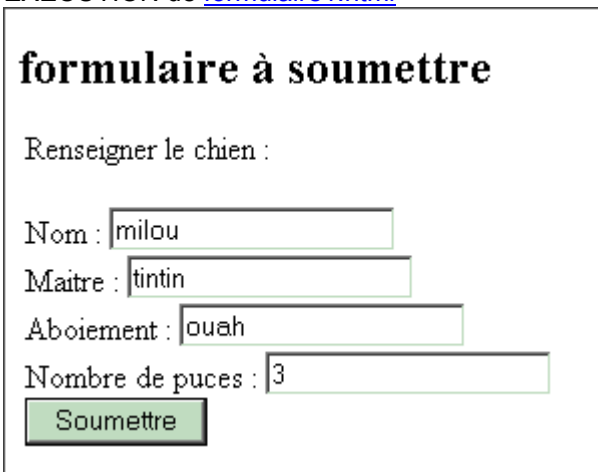
- les **formulaires** constituent un moyen de récupérer de l'information provenant de l'utilisateur
  la balise form délimite/définit le formulaire
  l'attribut method permet de choisir la méthode d'envoi HTTP entre POST et, à déconseiller, GET (envoi de paramètres visibles)
  l'attribut action précise l'URL demandée par la requête HTTP
  un formulaire se compose d'éléments :
  - la balise input définit des champs d'information simples
    elle possède un attribut name pour désigner l'élément
    et éventuellement un attribut value de valeur initiale
    elle se sub-divise selon son attribut type
    - text un petit champ pour que l'utilisateur saisisse un nombre ou une phrase
    - button pour cliquer dessus
    - checkbox des cases à cocher (information on/off)
    - radio des "radio-boutons" (information on/off mais un seul peut être coché à la fois !)
    - hidden est un champ caché pour transmettre des informations vers le serveur, sans que l'utilisateur ne le sache : c'est très utile pour gérer un caddie ...
    - password un champ texte mais masquant le texte tapé
    - submit est l'action indispensable lorsque le formulaire est utilisé pour envoyer de l'information depuis l'utilisateur vers le serveur : il "envoie" les données récupérées de l'utilisateur dans le formulaire vers le serveur
    - reset pour remettre le formulaire dans son état initial
  - select pour les listes
  - textarea pour un texte multi-lignes.

script de récupération

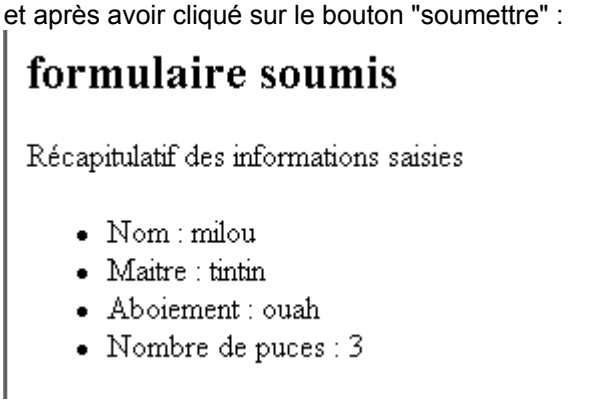
```
<html>
 <head>
  <title>formulaire soumis</title>
 </head>
 <body>
  <h2>formulaire soumis</h2>
  <?php
   import_request_variables("P","recu_");
   echo "R&eacute;capitulatif des informations saisies<br>\n
```

```
      <ul>
        <li>Nom : $recu_nom</li>
        <li>Maitre : $recu_maitre</li>
        <li>Aboiement : $recu_aboiement</li>
        <li>Nombre de puces : $recu_nombrePuces</li>
      </ul>";
  ?>
 </body>
</html>
```

- EXECUTION de formulaire1.html



et après avoir cliqué sur le bouton "soumettre" :



- **fonctionnement :**
  - Tous les champs du formulaire sont automatiquement disponibles dans le script PHP
  - import_request_variables("P","recu_") précise que les noms des variables recevant les valeurs envoyées sont composés du préfixe $recu_ suivi du nom du champ input :
    la valeur tintin mise dans l'input de nom maître est envoyée dans la variable $recu_maitre dans le script PHP

- import_request_variables(method,prefixe);
  PHP4!
  importe les variables globales récupérées d'un GET, POST ou de cookie, valeur "p" ou "P" pour POST ...
  préfixe est le string préfixe ajouté au nom des champ input pour former les variables du script
  en PHP4, contrairement à PHP3, register_globals est off, cad que les variables ci-dessus ne sont pas visibles.
- en PHP3, le nom de la variable est le nom du champ précédé de $
- en PHP3 et 4, nous pouvons utiliser les variables superglobales $_REQUEST['maitre'] ou plus précisément ici $_POST['maitre']

les informations envoyées par GET

```
...
    <form method="get" action="requete.php">
      Nom : <input type="text" name="nom"><br>
      Maitre : <input type="text" name="maitre"><br>
      Aboiement : <input type="text" name="aboiement"><br>
```

```
    Nombre de puces : <input type="text" name="nombrePuces"><br>
    <input type="submit" name="submit" value="Soumettre">
  </form>
...
```

- EXECUTION
  avec les valeurs : milou tintin OuaH 3
  La location de la nouvelle page appelée est :
  http://localhost/php/requete.php?nom=milou&maitre=tintin&
  aboiement=OuaH&nombrePuces=3&submit=Soumettre
- C'est la méthode (déconseillée) GET qui est appliquée car elle permet facilement de voir les informations transmises :
  - le nom de l'action, cad du fichier php à lancer
  - suivi de ?
  - puis les couples name=value de chacun des inputs du formulaire séparé par des &
- la méthode POST évite de voir les couples name=value

les expressions

constantes

```
<?php
 define("PI", 3.1416);
 define("BR", "<br>");
 echo PI; echo BR;
 $a = 2 * PI  ;
 echo $a; echo BR;
?>
```

- EXECUTION

  3.1416
  6.2832

- une constante est une valeur fixe nommée :
  - l'identificateur suit la syntaxe PHP des identificateurs sans le $!!
  - la valeur est integer, boolean, flot ou string
  - sa portée est globale
  - elle est définie une et une seule fois par define()

expression arithmétique

```
<?php
 $i = 2;
 $j = 6;
 echo "int i = $i, j = $j<br>" ;
 echo 'expression $i*$j+1 : ' ; echo $i*$j+1; echo "<br>" ;
 echo 'expression $i*($j+1) : ' ; echo $i*($j+1); echo "<br>" ;
 echo 'expression $j/(2*$i) : ' ; echo $j/(2*$i); echo "<br>" ;
 echo 'expression $j/$i : ' ; echo $j/$i; echo "<br>" ;
 echo 'expression ($j+1)/$i : ' ; echo ($j+1)/$i; echo "<br>" ;
 echo 'expression $j/2*$i : ' ; echo $j/2*$i; echo "<br>" ;
 echo 'expression $j%$i : ' ; echo $j%$i; echo "<br>" ;
 echo 'expression ($j+1)%$i : ' ; echo ($j+1)%$i; echo "<br>" ;
 $x = 2.0;
 $y = 6.0;
 echo "float x = $x, y = $y<br>" ;
 echo 'expression $x/$y : ' ; echo $x/$y; echo "<br>" ;
 echo 'expression ($x+5.0)%$y : ' ; echo ($x+5.0)%$y; echo "<br>" ;
```

```
 echo 'expression $y+$i-$x/3+(2+$j)%$x/$y : ' ;
 echo $y+$i-$x/3+(2+$j)%$x/$y; echo "<br>" ;
?>
```

- EXECUTION

  ```
  int i = 2, j = 6
  expression $i*$j+1 : 13
  expression $i*($j+1) : 14
  expression $j/(2*$i) : 1.5
  expression $j/$i : 3
  expression ($j+1)/$i : 3.5
  expression $j/2*$i : 6
  expression $j%$i : 0
  expression ($j+1)%$i : 1
  float x = 2, y = 6
  expression $x/$y : 0.33333333333333
  expression ($x+5.0)%$y : 1
  expression $y+$i-$x/3+(2+$j)%$x/$y : 7.3333333333333
  ```

- les expressions arithmétiques se comportent comme en C
  avec la même priorité des opérateurs

quelques opérateurs peu ou mal connus

```
<?php
 $i = 2;
 $j = 6;
 echo "int i = $i, j = $j<br>" ;
 echo 'expression $i++*$j+1 : ' ; echo $i++*$j+1; echo "<br>" ;
 echo "i = $i, j = $j<br>" ;
 $j += $i ; echo 'affectation $j += $i : <br>' ;
 echo "j = $j<br>" ;
 echo 'expression ($j > $i ? $j - $i : $i - $j) ' ;
 echo $j > $i ? $j - $i : $i - $j; echo "<br>" ;
?>
```

- EXECUTION

  ```
  int i = 2, j = 6
  expression $i++*$j+1 : 13
  i = 3, j = 6
  affectation $j += $i :
  j = 9
  expression ($j > $i ? $j - $i : $i - $j) 6
  ```

- ++ et -- opérateurs de pré ou post incrémentation et décrémentation :
  - dans l'expression ci-dessus, $i++ la valeur 2 de $i est utilisée dans l'expression avant de l'incrémenter de 1, c'est une post-incrémentation
- $var op= expression
  est équivalente à $var = $var op expression ,où op est +, -, /, *, !, ., &, ...
- exp_bool ? expression1 : expression1
  utilise l'opérateur conditionnel ternaire
  signifie si exp_bool vaut true, alors expression1 sinon expression2

concaténation de string

```
<?php
 $c = "abc";
 $a = 1;
 echo $c.$c."<br>";
 echo ($a + 2).$c."<br>";
```

```
  echo '$a'."abc".'<br>';
?>
```

- EXECUTION

  abcabc
  3abc
  $aabc

- l'opérateur de concaténation de chaînes s'écrit .

les fonctions mathématiques

```
<?php
  echo 'abs(-4.2) : '; echo abs(-4.2); echo "<br>";
  echo 'ceil(4.2) : '; echo ceil(4.2); echo "<br>";
  echo 'floor(4.2) : '; echo floor(4.2); echo "<br>";
  echo 'exp(2) : '; echo exp(2); echo "<br>";
  echo 'log(exp(2)) : '; echo log(exp(2)); echo "<br>";
  echo 'max(4.2,5) : '; echo max(4.2,5); echo "<br>";
  echo 'pow(4.2,1.3) : '; echo pow(4.2,1.3); echo "<br>";
  echo 'pow(4.2,-1.3) : '; echo pow(4.2,-1.3); echo "<br>";
  echo 'rand(2, 35) : '; echo rand(2, 35); echo "<br>";
  echo 'round(4.2) : '; echo round(4.2); echo "<br>";
  echo 'round(4.8) : '; echo round(4.8) ; echo "<br>";
  echo 'sqrt(4.2) : '; echo sqrt(4.2); echo "<br>";
?>
```

- EXECUTION

  abs(-4.2) : 4.2
  ceil(4.2) : 5
  floor(4.2) : 4
  exp(2) : 7.3890560989307
  log(exp(2)) : 2
  max(4.2,5) : 5
  pow(4.2,1.3) : 6.4598744516472
  pow(4.2,-1.3) : 0.15480177014044
  rand(2, 35) : 32
  round(4.2) : 4
  round(4.8) : 5
  sqrt(4.2) : 2.0493901531919

les fonctions chaînes

- EXECUTION

  chr(0x65) : e
  strtolower("aBCd") : abcd
  strtoupper("aBCd") : ABCD
  trim(" aBCd sdf ") : aBCd sdf
  nl2br("ligne1\nligne2") : ligne1
  ligne2
  htmlentities("e ? ? <") : e ? ? <
  strlen("abcd") : 4
  strcmp("abcd","aac") : 1
  strpos("une ligne pour chercher","ne") : 1
  strpos("une ligne pour chercher","ne",2) : 7
  strrev("abcd") : dcba
  substr("une ligne pour chercher",4,5) : ligne
  str_replace("ne", "NE", "une ligne pour chercher") : uNE ligNE pour chercher
  sprintf('$x = %d', 3) : $x = 3

sprintf('$x =%1.1f', 3.42) : $x =3.4

- chr (num)
  retourne une chaîne d'un seul caractère, dont le code ASCII est égal à l'entier num
- trim(chaîne)
  supprime les caractères invisibles en début et fin de chaine
- nl2br(chaîne)
  insère
 devant tout les caractères de fin de ligne de la chaîne
  en PHP 4 : c'est un
 compatible XHTML
- html_entity_decode(chaîne)
  PHP4 ! convertit toutes les entités HTML de la chaine en caractères normaux
- htmlentities(chaîne)
  convertit tous les caractères possibles en entités HTML
  PHP 3 et 4
- substr(chaine, position, longueur)
  retourne la sous-chaîne de chaîne commençant en position et de longueur indiquée
  l'indice de début d'une chaîne est 0
- str_replace(ancien, nouveau, chaine_a_modifier)
  remplace toutes les occurrences de la chaîne ancien par la chaîne nouveau dans la chaine_a_modifier
  PHP4 ! : les paramêtres peuvent être des tableaux de chaînes et str-replace a alors d'autres fonctionnalités
- addslashes(chaine)
  ajoute des backslashes aux caractères ' " \ et le caractère nul pour les requêtes le nécessitant en requête de bases de données
- sprintf(format, expression1, expression2, ...)
  formate comme printf de C

    - %b caractère correspondant au code ascii
    - %d int
    - %2d int dans un champ de 2 caractères au moins
    - %e réel en format scientifique
    - %f réel
    - %2.1f réel dans un champ de 2 caractères avant le point et 1 décimal
    - %o int en octal
    - %s chaîne de caractères
    - %x int en héxa
    - %% Caractère %

- explode(séparateur, chaîne)
  découpe la chaîne en sous-chaînes en utilisant le séparateur (chaîne)
  le résultat est le tableau de sous-chaînes obtenues
- implode (séparateur, array)
  retourne une chaîne contenant tous les éléments du tableau séparés par le séparateur (chaîne)
- ....

les instructions

instruction if-else

```
<html>
 <head>
  <title>instruction if-else  et bloc</title>
 </head>
 <body>
  <h2 align=center> instruction if-else et bloc</h2>
  <p>
   Affecter des valeurs aux variables :<br>
   <form method="post" action="if_else2.php">
    $i : <input type="text" name="i"><br>
    $j : <input type="text" name="j"><br>
    $k : <input type="text" name="k"><br>
    <input type="submit" name="submit" value="Executer">
   </form>
  </p>
 </body>
```

```
</html>
<html>
 <head>
   <title>if-else</title>
 </head>
 <body>
  <h2>if-else</h2>
  <p>
   <?php
    import_request_variables("P","recu_");
    if ($recu_i == $recu_j)
      print("$recu_i == $recu_j<br>");
    else
      print("$recu_i != $recu_j<br>");

    if (($recu_i == $recu_j) && ($recu_i == $recu_k))
      print("$recu_i, $recu_j, $recu_k sont &eacute;gaux<br>");
    if (($recu_i == $recu_j) && ($recu_i == $recu_k))
    {
      print("$recu_i == $recu_j");
      print(" et $recu_j == $recu_k<br>");
    }

    if (($recu_i == $recu_j) && ($recu_i == $recu_k))
    {
      print("$recu_i == $recu_j");
      print(" et $recu_j == $recu_k<br>");
    }
    else
    {
      print("$recu_i != $recu_j");
      print(" ou $recu_j != $recu_k<br>");
    }
   ?>
  <p>
 </body>
</html>
```

- EXECUTION avec respectivement les valeurs 2, 3, 2

  2 != 3
  2 != 3 ou 3 != 2

- EXECUTION avec respectivement les valeurs 3, 3, 3

  3 == 3
  3, 3, 3 sont égaux
  3 == 3 et 3 == 3
  3 == 3 et 3 == 3

- **Bloc d'instructions**

  - est délimité par des accolades { } : les instructions à l'intérieur du bloc sont à exécuter séquentiellement
  - {
    instruction_1
    instruction_2
    ...
    instruction_n
    }
  - les n instructions sont simples ou des blocs ou des instructions if-else, for-do, do-while .....
  - L'exécution du bloc d'instructions consiste en l'exécution séquentielle (l'une après l'autre, dans l'ordre) des n instructions.
  - Un bloc d'instructions peut apparaître partout où une instruction apparaît

- **if-else if** :

    - structure de contrôle conditionnelle
    - if (condition)
      instruction à exécuter quand
      la condition est vrai;
      else
      instruction à exécuter quand
      la condition est fausse;
    - if (condition)
      instruction à exécuter quand
      la condition est vrai;
    - la condition est une expression à valeur booléenne true ou false
      et est toujours entre parenthèses

imbrication de if-else

```
<?php
 import_request_variables("P","recu_");
 if ($recu_i == $recu_j)
   if ($recu_j == $recu_k)
     print("$recu_i, $recu_j, $recu_k sont &eacute;gaux <br>");
   else
     print("$recu_j != $recu_k<br>");
 else
   print("$recu_i != $recu_j<br>");

 if ($recu_i == $recu_j)
 {
   if ($recu_j == $recu_k)
     print("$recu_i, $recu_j, $recu_k sont &eacute;gaux <br>");
 }
 else
   print("$recu_i != $recu_j<br>");

 if ($recu_i == $recu_j)
   if ($recu_j == $recu_k)
     if ($recu_k == $recu_l)
       print("$recu_i, $recu_j, $recu_k, $recu_l sont &eacute;gaux<br>");
     else
       print("$recu_k != $recu_l<br>");

 if ($recu_i != $recu_j)
   print("$recu_i != $recu_j<br>");
 elseif ($recu_j != $recu_k)
   print("$recu_j != $recu_k<br>");
 elseif ($recu_k != $recu_l)
   print("$recu_k != $recu_l<br>");
 else System.out.print("$recu_i, $recu_j, $recu_k, $recu_l sont &eacute;gaux<br>");
?>
```

- **Imbrication d'instruction if-else** :
    - l'instruction à exécuter après un "if (condition)" ou un "else" est une instruction_simple; ou un { bloc d'instruction} ou une instruction alternative ou une boucle .....
    - Comment s'y retrouver dans une combinaison si-alors-si-alors-sinon ambiguë ?
      Voici la règle : le sinon "se rapporte" au dernier si disponible.
- **elseif**
  combinaison de if et de else

switch : le "choix parmi"

```
<?php
 import_request_variables("P","recu_");
```

```
 define("premier",3);
 switch (($recu_niemeJour + premier -1)% 7)
 {
   case 0 :
     $res = "lundi";
     break;
   case 1 :
     $res = "mardi";
     break;
   case 2 :
     $res = "mercredi";
     break;
.....
   case 6 :
     $res = "dimanche";
     break;
   default :
     $res = "erreur";
     break;
 }
 print("le $recu_niemeJour-i&egrave;me de l'ann&eacute;e 2004 est un $res");
?>
```

- EXECUTION pour 23 :

  le 23-ième de l'année 2004 est un vendredi

- **switch** :

  - switch (expression) {
    case valeur1 :
    instructions
    break;
    case valeur2 :
    instructions
    break;
    ...
    case valeur7 :
    case valeur8 :
    instructions
    break;
    default :
    instructions
    break;
    }
  - l'expression est de type int, float, string
  - Elle est évaluée puis le programme exécute alors les instructions à partir de la valeur correspondante jusqu'à rencontrer le mot break
  - si aucune valeur correspondante est proposée, alors le programme commence en default.
  - remarquons que plusieurs valeurs peuvent être regroupées comme ci-dessus.

- **break**
  permet de sortir d'une instruction switch, while, for, foreach

boucles while et do-while

```
<?php
 import_request_variables("P","recu_");
 $i = 0;
 do {
   print("\$i =  $i<br>");
   $i +=  1;
 }
 while ($i < $recu_nombre);
```

```
 $i = 0;
 print("puis<br>");
 while ($i < $recu_nombre)
 {
  print("\$i = $i<br>");
  $i++;
 }
?>
```

- EXECUTION pour 4 :

  $i = 0
  $i = 1
  $i = 2
  $i = 3
  puis
  $i = 0
  $i = 1
  $i = 2
  $i = 3

- do
  instructionCorpsDeBoucle
  while (test) ;
  avec test = expr. booléenne

  - exécution de l'instruction,
  - évaluation du test,
    si sa valeur est true,
    alors exécution de l'instruction,
  - évaluation du test,
    si sa valeur est true,
    alors exécution de l'instruction,
  - ....
  - arrêt de la boucle quand la valeur du test devient false.

- while (test)
  instructionCorpsDeBoucle

  - évaluation du test,
    si sa valeur est true,
    alors exécution de l'instruction,
  - évaluation du test,
    si sa valeur est true,
    alors exécution de l'instruction,
  - ....
  - arrêt de la boucle quand la valeur du test devient false.

- Dans les 2 formes, l'instruction dite corps de la boucle est une instruction simple, un bloc ou une instruction alternative ou une boucle, ...

boucle for

```
<?php
 import_request_variables("P","recu_");
 for ($nombre=1; $nombre<=$recu_entierMax; $nombre++)
 {
  $divisible = FALSE;
  for ($diviseur=floor($nombre/2);
      !$divisible && ($diviseur>1);
      $diviseur--)
    $divisible = ($nombre % $diviseur == 0);
  if (! $divisible)
```

```
    print("$nombre est premier<br>");
  }
?>
```

- EXECUTION pour 14 :

```
1 est
premier
2 est
premier
3 est
premier
5 est
premier
7 est
premier
11 est
premier
13 est
premier
```

- **for** :
  for (initialisation ; test ; nouvelle_itération)
  instruction_corps_de_boucle

  - l'instruction d'initialisation peut être vide
  - test = expr. bouléenne
  - nouvelle_itération = instruction qui prépare l'itération suivante (peut être vide)
  - la boucle for fonctionne comme
    initialisation;
    while (test) {
    instruction_corps_de_boucle
    nouvelle_itération
    }

- **continue**
  passe directement à l'itération suivante de la boucle

les tableaux : array

array "classique"

```
<?php
  $tab = array("bonjour", 3, " tout", -5.4, " monde");
  for ($i=0 ; $i < count($tab); $i++)
            print("indice $i element = $tab[$i] <br>");
  print('nombre élément = '.count($tab).'<br>');
  $tab[5] = "le 6-ieme";
  $tab[] = "encore";
  for ($i=0 ; $i < count($tab); $i++)
            print("indice $i element = $tab[$i] <br>");
?>
```

- EXECUTION

```
  indice 0 élément = bonjour
  indice 1 élément = 3
  indice 2 élément = tout
  indice 3 élément = -5.4
  indice 4 élément = monde
  nombre élément = 5
  indice 0 élément = bonjour
  indice 1 élément = 3
```

```
 indice 2 élément = tout
 indice 3 élément = -5.4
 indice 4 élément = monde
 indice 5 élément = le 6-ieme
 indice 6 élément = encore
<?php
 $tab[0]="bonjour ";
 $tab[1]="tout le ";
 $tab[2]="monde";
 $tab2 = $tab;
 $tab2[0]='au revoir';
 print ('$tab <br>');
 for ($i=0 ; $i < count($tab); $i++)
        print("indice $i element = $tab[$i] <br>");
 print ('$tab2 <br>');
 for ($i=0 ; $i < count($tab2); $i++)
        print("indice $i element = $tab2[$i] <br>");
?>
```

- EXECUTION

```
$tab
indice 0 élément = bonjour
indice 1 élément = tout le
indice 2 élément = monde
$tab2
indice 0 élément = au revoir
indice 1 élément = tout le
indice 2 élément = monde
```

- **array** :
  - création :
    - array(valeur0, valeur1, ...)
      crée un tableau avec les valeurs indiquées dans l'ordre indiqué
      les éléments du tableau auront respectivement les valeurs spécifiées
      le premier élément a la clé (l'indice) 0 et la valeur0
      le second élément a la clé 1 (l'indice) et la valeur1
      .....
    - $tab[indice] = expression ;
      si aucun élément n'existait à cet indice (avec cette clé) alors un nouvel élément est crée.
    - $tab[ ] = expression ;
      ajoute un élément d'indice maximun +1
  - manipulation :
    - un élément à l'indice indiqué : $tab[indice]
    - count(tableau)
      donne le nombre d'éléments (qui n'est pas forcément l'indice maximal + 1)
    - sizeof(tableau) est un alias
    - l'affectation de tableau réalise une copie de l'ensemble de valeurs et clés

array "dynamique"

```
<?php
 $tab = array("bonjour", 3, " tout", -5.4, " monde");
 for ($i=0 ; $i < count($tab); $i++)
        print("indice $i élément = $tab[$i] <br>");

 $tab[7] = "un nouveau en 7";
 print('ajout en 7 : $tab[7] = '.$tab[7].'<br>');
 print('nombre élément = '.count($tab).'<br>');

 unset($tab[3]);
```

```
  print('après unset un élément :nombre élément = '.count($tab).'<br>');
  unset($tab);
  print('après unset le tableau : isset($tab) = '.isset($tab).'<br>');

  $tab[0]="un nouveau";
  $tab[]="suivant";
  print('après 2 affectations : <br>');
  for ($i=0 ; $i < count($tab); $i++)
            print("indice $i élément = $tab[$i] <br>");
?>
```

- EXECUTION

  indice 0 élément = bonjour
  indice 1 élément = 3
  indice 2 élément = tout
  indice 3 élément = -5.4
  indice 4 élément = monde
  ajout en 7 : $tab[7] = un nouveau en 7
  nombre élément = 6
  après unset un élément:nombre élément = 5
  après unset le tableau : isset($tab) =
  après 2 affectations :
  indice 0 élément = un nouveau
  indice 1 élément = suivant

- remarquons la non corrélation entre le nombre d'élément et l'indice maximal
- unset(élément de tableau)
  supprime un élément du tableau : sa valeur et sa clé (indice)
  si le tableau est indicé à partir de 0, cela peut créer un "trou" dans la suite des indices
- unset(tableau)
  supprime tous les éléments du tableau et le tableau lui-même
- array_values(tableau)
  retourne un tableau des valeurs du tableau en paramètre
  cela re-indexe numériquement le tableau.

array "associatif"

```
<?php
  $jourSemaine = array("lundi" => "travail",
       "mardi" => "travail", "mercredi" => "travail",
                              "jeudi" => "travail", "vendredi" => "travail",
                              "samedi" => "week-end", "dimanche" => "week-end");
  printf("le samedi est un jour de "
        .$jourSemaine["samedi"]."<br>");
  foreach ($jourSemaine as $jour => $sorte)
            print("$jour : $sorte <br>");
  $compte = 0;
  foreach ($jourSemaine as  $sorte)
            if ($sorte == "travail")
              $compte++;
  printf(" il y a $compte jours de travail sur ".count($jourSemaine));
?>
```

- EXECUTION

  le samedi est un jour de week-end
  lundi : travail
  mardi : travail
  mercredi : travail
  jeudi : travail
  vendredi : travail
  samedi : week-end

dimanche : week-end
il y a 5 jours de travail sur 7

- Un array "PHP" est un ensemble ordonné d'éléments accessible par un index (ordered map) :
    - c'est une association de paire (clé, valeur) : la clé permet d'accéder à l'élément
    - la clé est soit un entier soit une chaîne
    - la valeur peut être de n'importe quel type
    - en PHP4, l'array peut être utilisé comme :
        - tableau classique,
        - liste,
        - pile, file, ensemble,
        - table accessible par clé, dictionnaire.
- array(cle0 => valeur0, cle1 => valeur1, ...)
  crée un tableau avec les éléments indiquées dans l'ordre
  les éléments du tableau sont respectivement les paires (clé, valeur) spécifiées
  le premier élément a la clé cle0 et la valeur0
  le second élément a la clé cle1 et la valeur1
  .....
- les éléments se manipulent ainsi : $tab[clé] où clé est un entier ou un string
- **foreach**
    - PHP4 !
    - foreach(array_expression as $key => $value)
      commandes
        - réalise une boucle où les variables $key et $value prennent successivement les clés et valeurs des éléments du tableau
    - foreach(array_expression as $value)
      commandes
        - réalise une boucle où la variable $value prend successivement les valeurs des éléments du tableau

fonctions d'array

```
<?php
  $tab = array("bonjour"=>3, "tout"=>0, " monde"=>-5);
  foreach ($tab as $key => $value)
              print("$key : $value , ");
  print('<br>tri<br>');
  $t2 = $tab;
  sort($t2);
  foreach ($t2 as $key => $value)
              print("$key : $value , ");
  print('<br>sort by key<br>');
  $t2 = $tab;
  ksort($t2);
  foreach ($t2 as $key => $value)
              print("$key : $value , ");
  print('<br>');
  if (in_array(0,$tab))
               print('in_array(0,$tab) true<br>');
?>
```

- EXECUTION

  bonjour : 3 , tout : 0 , monde : -5 ,
  tri
  0 : -5 , 1 : 0 , 2 : 3 ,
  sort by key
   monde : -5 , bonjour : 3 , tout : 0 ,
  in_array(0,$tab) true

- in_array(valeur,tableau)
  PHP4! retourne true si la valeur se trouve dans le tableau
- explode(separateur, chaine)
  découpe la chaîne en sous-chaînes en utilisant le séparateur (chaîne)
  le résultat est le tableau de sous-chaînes obtenues
- implode (separateur, array)
  retourne une chaîne contenant tous les éléments du tableau séparés par le séparateur (chaîne)
- array_key_exists(cle, tableau)
  PHP4! retourne true si une clé existe dans un tableau
- array_keys(tableau)
  PHP4! retourne un tableau de toutes les clés du tableau en paramètre
- array_splice(tableau, position, nombre)
  PHP4! supprime nombre éléments du tableau à partir de la position
  array_splice(tableau, position, nombre, tab_remplace)
  PHP4! supprime et remplace par les éléments du tableau de remplacement
- array_pop(tableau)
  PHP4! dépile et retourne le dernier élément du tableau
  retourne NULL si le tableau est vide
- array_push((tableau, elem1, elem2, ...)
  PHP4! empile à la fin du tableau les éléments
- array_shift(tableau)
  PHP4! dépile le premier élément
- array_slice(tableau, position, nombre)
  PHP4! retourne un sous-tableau extrait du tableau à partir de la position et composé de nombre éléments
- array_sum(tableau)
  PHP4! retourne la somme des valeurs du tableau
- ....

le pointeur courant d'array

```
<?php
  $tab = array("bonjour", 3, "tout", "monde", -5);
  print('current($tab) '.current($tab).'<br>');
  print('next($tab) '.next($tab).'<br>');
  print('next($tab) '.next($tab).'<br>');
  print('next($tab) '.next($tab).'<br>');
  print('next($tab) '.next($tab).'<br>');
  reset($tab);
  print('après reset : current($tab) '.current($tab).'<br>');
  end($tab);
  print('après end : current($tab) '.current($tab).'<br>');
  print('prev($tab) '.prev($tab).'<br>');
?>
```

- EXECUTION

  current($tab) bonjour
  next($tab) 3
  next($tab) tout
  next($tab) monde
  next($tab) -5
  après reset : current($tab) bonjour
  après end : current($tab) -5
  prev($tab) monde

- **pointeur interne d'array** :
  - chaque tableau a un pointeur interne, qui est initialisé lorsque le premier élément est inséré dans le tableau.
  - current(tableau)
    retourne l'élément courant pointé par le pointeur interne
  - next(tableau)
    incrémente le pointeur interne

retourner la valeur qu'il pointe,
retourne false s'il "dépasse" le tableau ou si l'élément pointé est NULL

- end(tableau)
  positionne le pointeur interne du tableau jusqu'au dernier élément.
  retourne la valeur de cet élément
- prev(tableau)
  décrémente le pointeur plutôt que de l'avancer.
  retourne la valeur du nouvel élément pointé ou false si l'élément pointé est NULL
- reset(tableau)
  replace le pointeur de tableau au premier élément.
  retourne la valeur du premier élément.

parcours d'array en PHP3

```
<?php
 $jourSemaine = array("lundi" => "travail",
    "mardi" => "travail","mercredi" => "travail",
    "jeudi" => "travail","vendredi" => "travail",
              "samedi" => "week-end","dimanche" => "week-end");
 printf("le samedi est un jour de ".$jourSemaine["samedi"]."<br>");
 while (list($jour,$sorte)= each($jourSemaine))
   print("$jour : $sorte <br>");
?>
```

- EXECUTION

  le samedi est un jour de week-end
  lundi : travail
  mardi : travail
  mercredi : travail
  jeudi : travail
  vendredi : travail
  samedi : week-end
  dimanche : week-end

- **each**(tableau)
  - retourne la paire clé/valeur courante du tableau
    puis avance le pointeur interne du tableau.
    le résultat est un tableau de 4 éléments, avec les clés 0, 1, key, et value :
    - les éléments de clés 0 et key contiennent le nom de la clé
    - les éléments de clés 1 et value contiennent la valeur.
    - retourne false si le pointeur est en dehors du tableau
  - permet de réaliser une boucle de parcours de tableau en PHP3
- list(var1,var2, ...)
  - est une instruction qui permet d'affecter une série de variables en une seule ligne
  - list(var1,var2, ..., varn) = $tableau
    affecte aux n variables respectivement les n premières valeurs du tableau

fonction et visibilité des variables

fonction

```
<?php
 function formaterDate($jour, $mois, $an)
            // renvoie un string jour/mois/an
            // avec jour et mois sur 2 chiffres
 { if ($jour<10)
             $res = '0'.$jour.'/';
            else
             $res = $jour.'/';
```

```
            if ($mois<10)
              $res = $res.'0'.$mois.'/';
            else
              $res = $res.$mois.'/';
   return $res.$an;
 }
 import_request_variables("P","recu_");
 echo 'date format&eacute;e : '
    .formaterDate($recu_jour,$recu_mois,$recu_an);
?>
```

- EXECUTION pour 3 5 2003

  date formatée : 03/05/2003

- **function**

  - function nom_de_fonction ($arg1, $arg2, ..., $argn)
    {
    instructions
    return expression; // éventuellement
    }
  - la 1ère ligne est l'en-tête de la fonction
  - le corps de la fonction : c'est les instructions entre { et }
    ces instructions peuvent utiliser les variables $arg1, $arg2, ...
  - return arrête l'exécution du corps de la fonction et retourne la valeur de l'expression calculée.
  - l'appel d'une fonction se fait ainsi :
    nom_de_fonction (expression1, exp2, ..., expn)
    le passage des paramètres se fait par valeur : la valeur de l'expression i est affectée à la variable paramètre $argi
  -  En PHP 3, les fonctions doivent être définies avant d'être utilisées

fonction dans un fichier externe

```
<?php
 function formaterDate($jour, $mois, $an)
   // renvoie un string jour/mois/an
   // avec jour et mois sur 2 chiffres
 { if ($jour<10)
     $res = '0'.$jour.'/';
   else
     $res = $jour.'/';
   if ($mois<10)
     $res = $res.'0'.$mois.'/';
   else
     $res = $res.$mois.'/';
   return $res.$an;
 }
?>
<html>
 ....
  <?php
   require 'ma_fonction2.php';
   import_request_variables("P","recu_");
   echo 'date format&eacute;e : '
     .formaterDate($recu_jour, $recu_mois, $recu_an);
  ?>
 ...
</html>
```

- EXECUTION ... idem
- **require**(fichier) ; require fichier ;
  include(fichier) ; include fichier ;

- inclut et exécute un fichier PHP.
  sous réserve que le fichier soit accessible via l'include_path de la configuration d'Apache
- en cas d'erreur, include produit des warnings alors que require provoque une erreur fatale
- indispensable pour réaliser du code modulaire

valeur de paramêtre par défaut

```
<?php
 function formaterDate($jour, $mois, $an=2000)
            // renvoie un string jour/mois/an
            // avec jour et mois sur 2 chiffres
 { if ($jour<10)
              $res = '0'.$jour.'/';
            else
              $res = $jour.'/';
            if ($mois<10)
              $res = $res.'0'.$mois.'/';
            else
              $res = $res.$mois.'/';
   return $res.$an;
 }
 function printbr($chaine)
            // remplace les \n par des <br>
            // et print la chaine obtenue
            // en ajoutant un br final si n?cessaire
 { if (substr($chaine,strlen($chaine)-1,1)!="\n")
              $chaine .= "\n";
            print (nl2br($chaine));
 }
 import_request_variables("P","recu_");
 printbr ("et voici\nla date format&eacute;e : \n");
 printbr (formaterDate($recu_jour, $recu_mois));
?>
```

- EXECUTION pour 3 5 2003

  et voici
  la date formatée :
  03/05/2000

- Il est possible de définir des valeurs de paramètre par défaut :
  - s'il manque des valeurs lors de l'appel de la fonction, celle-ci utilisera les valeurs par défaut pour les compléter
  - par conséquent, les paramètres sans valeurs par défaut doivent se trouver avant ceux ayant une valeur par défaut.

- PHP 3 ne supporte pas une liste de paramètres variables dans l'en-tête d'une fonction, mais PHP 4 oui.
- les fonctions peuvent ne pas retourner de valeur.

fonction à résultat multiple

```
<?php
 function divisionEntiere($entier, $diviseur)
            // renvoie un array avec le quotient puis le reste
 { $quotient = (int)($entier / $diviseur);
            $reste = $entier - $diviseur * $quotient;
            return array($quotient, $reste);
 }
 import_request_variables("P","recu_");
 $tab = divisionEntiere($recu_entier, $recu_diviseur);
 echo $entier.' = '.$diviseur.' * '.$tab[0].' + '.$tab[1];
```

```
?>
```

- EXECUTION pour 15 6

  15 = 6 * 2 + 3

- le type de valeur de retour d'une fonction est boolean, integer, float, string, array, object, ...
- pour renvoyer plusieurs valeurs, retournez un array ou un objet

visibilité simple des variables

```
<?php
 $a1 = 1;
 function fonc($param)
 {
            $varlocal = 3;
            $param++;
            echo 'dans fonc : $param = '.$param.' donc visible<br>';
            echo 'dans fonc : $varlocal = '
       .$varlocal.' donc visible<br>';
            echo 'dans fonc : $a1 = '.$a1.' donc non visible<br>';
 }
 echo 'avant appel de fonc : $a1 = '.$a1
      .' donc visible<br>';
 echo 'appel fonc($a1)<br>';
 fonc($a1);
 echo 'après appel de fonc : $a1 = '.$a1
              .' donc visible et non modifi&eacute;<br>';
 echo 'après appel de fonc : $varlocal = '
      .$varlocal.' donc non visible<br>';
?>
```

- EXECUTION

  avant appel de fonc : $a1 = 1 donc visible
  appel fonc($a1)
  dans fonc : $param = 2 donc visible
  dans fonc : $varlocal = 3 donc visible
  dans fonc : $a1 = donc non visible
  après appel de fonc : $a1 = 1 donc visible et non modifié
  après appel de fonc : $varlocal = donc non visible

- différent du langage C : !! les variables "globales" (au sens C) comme $a1 ne sont pas visibles à l'intérieur des fonctions
- portée des variables
  - une variable définie dans une fonction n'est visible que dans le corps de cette dernière
  - les variables paramètres d'une fonction ne sont visibles dans le corps de cette dernière
  - un variable définie dans un script (hors d'une fonction) est visible dans tout le script
  - mais pas dans le corps des fonctions du script
  - mon_script contient ...$var ...; include autre_script.php ;
    la variable $var est visible dans l'autre_script et les variables de l'autre_script sont visibles dans mon_script

global et passage par référence

```
<?php
 $a1 = 1;
 $a4 = 4 ;
 function fonc(&$param)
 {
   global $a1;
```

```
        $a1++;
        $param++;
        echo 'dans fonc : $param = '.$param.' donc visible<br>';
        echo 'dans fonc : $a1 = '.$a1.' donc visible<br>';
        echo 'dans fonc : $a4 = '.$a4.' donc non visible<br>';
 }
 echo 'avant appel de fonc : $a1 = '.$a1.' donc visible<br>';
 echo 'avant appel de fonc : $a4 = '.$a4.' donc visible<br>';
 echo 'appel fonc($a4)<br>';
 fonc($a4);
 echo 'après appel de fonc : $a1 = '.$a1
            .' donc visible et modifi&eacute;<br>';
 echo 'après appel de fonc : $a4 = '.$a4
            .' donc visible et modifi&eacute;<br>';
?>
```

- EXECUTION

  avant appel de fonc : $a1 = 1 donc visible
  avant appel de fonc : $a4 = 4 donc visible
  appel fonc($a4)
  dans fonc : $param = 5 donc visible
  dans fonc : $a1 = 2 donc visible
  dans fonc : $a4 = donc non visible
  apres appel de fonc : $a1 = 2 donc visible et modifié
  apres appel de fonc : $a4 = 5 donc visible et modifié

- **global**
  - global $var1,$var2, ...;
    définie les variables de portée globale : visible dans les corps de fonction
    une autre façon d'y accéder est d'utiliser le tableau superglobal $GLOBALS ainsi $GLOBALS["var1"]
  - il existe des variables superglobales (prédéfinies) accessibles dans tous les scripts : $_SERVER, $_GET, ...

- **référence** :
  - les références sont des alias d'une variable et non un pointeur comme en C
  - il est possible de passer une variable par référence au niveau des paramètres d'une fonction : en préfixant le paramètre par &
  - ✚✚ il est possible qu'une fonction retourne une référence
- ✚✚ il est possible de définir des variables static dans les fonctions comme en C

PHP et MySQL : introduction

créer une base de données et une table

Avec phpMyAdmin, créez votre base de données nosamisleschiens
avec une table chien
comportant les colonnes nom, maître, aboiement, nombrePuces
de clé primaire nom

afficher le contenu d'une table

```
<?php
  $connect = mysql_connect('localhost','root','')
```

```
            or die ("erreur de connexion");
  mysql_select_db('nosamisleschiens',$connect)
    or die ("erreur de connexion base");
  $result = mysql_query("SELECT nom, maitre, aboiement, nombrePuces FROM chien");
  while ( $row = mysql_fetch_array($result))
  {
    echo $row[0].', '.$row[1].', '.$row[2].', '.$row[3].'<br>';
  }
  mysql_close();
?>
```

- EXECUTION

  nom, maître, aboiement, nombrePuces :
  milou, tintin, ouah, 4
  rantanplan, averel, miam, 26
  medor, sarko, grrr, 10

- **script classique de requete** select sur une table

  - connection au server de base de données
  - connection à une base de données
  - exécution d'une requête SQL select sur une table
    et récupération du résultat
  - boucle d'extraction et de formatage HTML des lignes du résultat
  - fermeture de la connection

- mysql_connect (url_serveur, user, password)
  ouvre une connexion à un serveur MySQL
  retourne une ressource de connexion au serveur
  sinon retourne false et envoi un message d'erreur
  paramètres : url_server url du serveur de base de données, user un logname d'utilisateur du serveur, et son password
- mysql_select_db(nom_base, connection)
  sélectionne une base de données par son nom sur le serveur MySQL connecté
  retourne true sinon false et envoi un message d'erreur
  le paramètre connection est une ressource de connection ouverte à un serveur MySQL
  le paramètre nom_base est un string
- mysql_query(requete, connection)
  envoie une requête SQL à la base de données de la connection
  retourne une ressource de résultat MySQL
  sinon false et envoi un message d'erreur
  le paramètre requête est un string comprenant une requête SQL qui ne doit pas être terminée par un point-virgule
  le paramètre connection est une ressource de connection ouverte à un serveur MySQL
- mysql_fetch_array(liste_résultat_requete)
  permet le parcours de la liste de résultat d'une requête SQL
  retourne un tableau qui contient la ligne courante puis incrémente le pointeur de liste résultat
  sinon False s'il ne reste plus de ligne.
  le tableau retourné à chaque appel correspond à une ligne du résultat : il est à la fois associatif (clé : nom de la colonne et valeur : sa valeur) et indexé (les valeurs des colonnes sont rangées dans l'ordre à partir de l'indice 0)
- mysql_close(connection)
  ferme la connexion au serveur MySQL
  le paramètre connection est une ressource de connection ouverte à un serveur MySQL
  en général, la fin d'un script PHP ferme automatiquement ses connections.

ajouter une ligne dans une table

```
<form method="post" action="insert.php">
  Nom : <input type="text" name="nom"><br>
  Maitre : <input type="text" name="maitre"><br>
  Aboiement : <input type="text" name="aboiement"><br>
  Nombre de puces : <input type="text"
name="nombrePuces"><br>
```

```
  <input type="submit" name="submit" value="Insérer">
</form>
<?php
  import_request_variables("p","p_");
  $connect = mysql_connect('localhost','root','')
              or die ("erreur de connexion");
  mysql_select_db('nosamisleschiens',$connect)
            or die ("erreur de connexion base");
  mysql_query("INSERT INTO chien VALUES ('$p_nom','$p_maitre',
                        '$p_aboiement', '$p_nombrePuces') ");
  mysql_close();
?>
```

- EXECUTION pour les valeurs : snoopy charly woooo 13
  résultat .....rien

  allez dans phpmyadmin pour vérifier s'il est dans la table chien
  snoopy charly woooo 13
- Le schéma de connexion est exactement le même que pour un select
  - seule change la requête SQL et le fait qu'il n'y a pas d'exploitation de résultat

autres techniques de parcours du résultat d'une requête SQL

```
<?php
  header("Pragma:no-cache");
  $connect = mysql_connect('localhost','root','')
              or die ("erreur de connexion");
  mysql_select_db('nosamisleschiens',$connect)
            or die ("erreur de connexion base");
  $resultSelect = mysql_query("SELECT * FROM chien");
?>
<html>
  <head>
    <title>select all</title>
  </head>
  <body>
    <h2 align=center> SELECT nom, maitre, aboiement, nombrePuces FROM chiens</h2>
    <table border=1>
      <tr>
        <th>nom</th><th>maitre</th><th>aboiement</th><th>nombrePuces</th>
      </tr>
      <?php
        for ($i=0; $i < mysql_num_rows($resultSelect); ++$i)
        {
          echo '<tr>';
          mysql_data_seek($resultSelect, $i);
          $champs = mysql_fetch_array($resultSelect);
          for ($col=0; $col<4; $col++)
            echo '<td>'.$champs[$col].'</td>';
          echo '</tr>';
        }
        mysql_close();
      ?>
    </table>
  </body>
</html>
```

- EXECUTION

SELECT nom, maitre, aboiement, nombrePuces FROM chiens

nom	maitre	aboiement	nombrePuces
milou	tintin	ouah	4
rantanplan	averel	miam	26
medor	sarko	grrr	10
snoopy	charly	woooo	13

- mysql_num_rows(liste_résultat_requete)
  retourne le nombre de lignes d'un résultat de requête SQL
  le paramètre est une ressource de résultat MySQL
- mysql_data_seek(liste_résultat_requete, position)
  déplace le pointeur interne de résultat de requête SQL
  position doit être compris entre 0 et mysql_num_rows(liste_résultat_requete) -1
  retourne true ou false

```
<?php
  while ($champs = mysql_fetch_object($resultSelect))
    echo '<tr><td>'.$champs->nom.'</td><td>'.$champs->aboiement
        .'</td><td>'.$champs->nombrePuces.'</td></tr>';
  mysql_close();
?>
```

- EXECUTION

SELECT nom, aboiement, nombrePuces FROM chiens WHERE nombrePuces > 4

nom	aboiement	nombrePuces
rantanplan	miam	26
medor	grrr	10
snoopy	woooo	13

- mysql_fetch_object(liste_résultat_requete)
  permet le parcours de la liste de résultat d'une requete SQL
  retourne un objet dont les propriétés correspondent à la ligne courante puis incrémente le pointeur de liste résultat
  sinon False s'il ne reste plus de ligne.
  l'objet a comme variables les noms de colonnes : $objet->nom_colonne

contrôle de réalisation d'une requête insert

```
<?php
  header("Pragma:no-cache");
  import_request_variables("P","recu_");
  $connect = mysql_connect('localhost','root','')
              or die ("erreur de connexion");
  mysql_select_db('nosamisleschiens',$connect)
           or die ("erreur de connexion base");
  @ $etat=mysql_query("INSERT INTO chien VALUES ('$recu_nom','$recu_maitre',
                    '$recu_aboiement', '$recu_nombrePuces') ");
  mysql_close();
?>
<html>
  <head>
    <title>insertion</title>
  </head>
  <body>
```

```
   <p>
    <?php
     if ($etat)
       echo 'insertion reussie !';
     else
       echo 'echec d\'insertion';
    ?>
   </p>
  </body>
</html>
```

- EXECUTION

  insertion reussie !

Source de insert2.php

```
<?php
 header("Pragma:no-cache");
 $connect = mysql_connect('localhost','root','')
            or die ("erreur de connexion");
 mysql_select_db('nosamisleschiens',$connect)
          or die ("erreur de connexion base");
 $mauvaiseRequete = "INSERT INTO chats VALUES ('pussy cat','baby','miaou','0')";
?>
<html>
 <head>
  <title>insertion</title>
 </head>
 <body>
  <p>
   <?php
    if (@ $etat=mysql_query($mauvaiseRequete))
      echo 'insertion reussie !';
    else
      echo 'echec d\'insertion : '.mysql_errno().' : '
           .mysql_error();
    mysql_close();
   ?>
  </p>
```

- EXECUTION de insert2.php

  echec d'insertion : 1146 : Table 'nosamisleschiens.chats' doesn't exist

- mysql_error()
  retourne le texte associé à l'erreur générée, s'il y a, de la dernière requête du SGBD MySQL
- mysql_errno()
  retourne le numéro d'erreur de la dernière commande MySQL

Modifier et supprimer
dans une table de BD

supprimer une ligne d'une table

```
<?php
 header("Pragma:no-cache");
 $connect = mysql_connect('localhost','root','')
            or die ("erreur de connexion");
 mysql_select_db('nosamisleschiens',$connect)
          or die ("erreur de connexion base");
 $resultSelect = mysql_query("SELECT * FROM chien");
```

```
?>
<html>
  <head>
    <title>delete</title>
  </head>
  <body>
    <h2 align=center> DELETE FROM chiens WHERE nom = ....</h2>
    <table border=1>
      <tr>
        <th>nom</th><th>maitre</th><th>aboiement</th>
        <th>nombrePuces</th><th> </th>
      </tr>
      <?php
        for ($i=0; $i < mysql_num_rows($resultSelect); ++$i)
        {
          echo '<tr>';
          mysql_data_seek($resultSelect, $i);
          $champs = mysql_fetch_array($resultSelect);
          for ($col=0; $col<4; ++$col)
            echo '<td>'.$champs[$col].'</td>';
          echo '<td><a href="delete11.php?nom='.$champs[0].'">Supp</a></td></tr>';
        }
        mysql_close();
      ?>
    </table>
  </body>
</html>
```

Source de delete11.php

```
<?php
  header("Pragma:no-cache");
  import_request_variables("G","recu_");
  $connect = mysql_connect('localhost','root','')
                or die ("erreur de connexion");
  mysql_select_db('nosamisleschiens',$connect)
             or die ("erreur de connexion base");
  @ $etat=mysql_query("DELETE FROM chien WHERE nom=\"$recu_nom\" ");
  if (!$etat)
    $messageErreur = mysql_errno().' : '.mysql_error();
  mysql_close();
?>
<html>
  <head>
    <title>suppression</title>
  </head>
  <body>
    <p>
      <?php
        if ($etat)
          echo 'suppression reussie !';
        else
          echo 'echec de suppression : '.$messageErreur;
      ?>
    </p>
  </body>
</html>
```

- EXECUTION

DELETE FROM chiens WHERE nom = ...

nom	maitre	aboiement	nombrePuces	
milou	tintin	ouah	4	Supp
rantanplan	averel	miam	26	Supp
medor	sarko	grrr	10	Supp
snoopy	charly	woooo	13	Supp

suppression reussie !

- La technique classique consiste à afficher la liste des lignes de la table formatée en table HTML dans laquelle se trouve des liens (1 par ligne) de suppression
  - Ceci permet d'identifier la ligne à supprimer

modifier une ligne d'une table

```
...
  echo '<td><a href="modifier11.php?'.$vars.'">Modif</a></td></tr>';
 ...
```

Source de modifier11.php

```
<?php
  header("Pragma:no-cache");
  import_request_variables("G","recu_");
  $connect = mysql_connect('localhost','root','')
                or die ("erreur de connexion");
  mysql_select_db('nosamisleschiens',$connect)
            or die ("erreur de connexion base");
  $resultSelect = mysql_query("SELECT maitre, aboiement, nombrePuces
                          FROM chien WHERE nom=\"$recu_nom\" ");
  if (mysql_num_rows($resultSelect)!=1)
    die("la clef primaire n'existe pas !");
  $champs = mysql_fetch_object($resultSelect);
?>
<html>
  <head>
    <title>formulaire de modification</title>
  </head>
  <body>
    <h2 align=center> formulaire de modification</h2>
    <p>
     Modification des renseignements sur le chien :<br>
     <form method="post" action="modifier111.php">
       Nom : <?php echo $recu_nom; ?><input type="hidden" name="nom"
                    value="<?php echo $recu_nom; ?>" ><br>
       Maitre : <input type="text" name="maitre"
                    value="<?php echo $champs->maitre; ?>" ><br>
       Aboiement : <input type="text" name="aboiement"
                    value="<?php echo $champs->aboiement; ?>" ><br>
       Nombre de puces : <input type="text" name="nombrePuces"
                    value="<?php echo $champs->nombrePuces; ?>" ><br>
       <input type="submit" name="submit" value="Modifier">
     </form>
    </p>
  </body>
</html>
```

Source de modifier111.php

```
<?php
  header("Pragma:no-cache");
  import_request_variables("P","recu_");
  $connect = mysql_connect('localhost','root','')
             or die ("erreur de connexion");
  mysql_select_db('nosamisleschiens',$connect)
          or die ("erreur de connexion base");
  @ $etat=mysql_query("UPDATE chien SET nom=\"$recu_nom\",
                  maitre=\"$recu_maitre\",aboiement=\"$recu_aboiement\",
                  nombrePuces=\"$recu_nombrePuces\"
                  WHERE nom=\"$recu_nom\" ");
  if (!$etat)
    $messageErreur = mysql_errno().' : '.mysql_error();
  mysql_close();
?>
<html>
  <head>
    <title>modification</title>
  </head>
  <body>
    <p>
      <?php
        if ($etat)
          echo 'modification reussie !';
        else
          echo 'echec de modification : '.$messageErreur;
      ?>
    </p>
  </body>
</html>
```

- EXECUTION

UPDATE chiens SET nom='...',maitre='...',
... WHERE nom ='....'

nom	maitre	aboiement	nombrePuces	
milou	tintin	ouah	4	Modif
rantanplan	averel	miam	26	Modif
medor	sarko	grrr	10	Modif

formulaire de modification

Modification des renseignements sur le chien :

Nom : medor
Maitre : sarko
Aboiement : grrr
Nombre de puces : 100
Modifier

modification reussie !

- La technique classique consiste à afficher la liste des lignes de la table formatée en table HTML dans laquelle se trouve des liens (1 par ligne) de modification

- Ceci permet d'identifier la ligne à modifier

session et cookie

session : suivre un utilisateur de page en page

```
<?php
  session_start();
?>
<html>
  ...
     Choisir une couleur:<br>
     <form action="form_session2.php" method="post">
       vert <input type="radio" name="couleur" value="vert" checked>
       rouge <input type="radio" name="couleur" value="rouge">
       <input type="submit" value="Envoyer">
     </form>
  ...
```

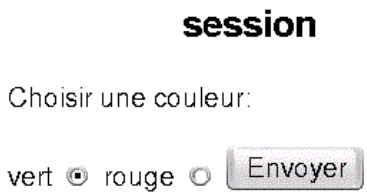
Source de form_session2.php

```
<?php
  session_start();
  import_request_variables("P","recu_");
  $_SESSION['couleur'] = $recu_couleur;
?>
<html>
  ....
```

Source de session4.php

```
<?php
  session_start();
  import_request_variables("P","recu_");
  $_SESSION['music'] = $recu_music;
?>
<html>
  ....
     recapitulatif de la session :<br>
     <?php
       echo 'couleur = '.$_SESSION['couleur'].'<br>';
       echo 'gout = '.$_SESSION['gout'].'<br>';
       echo 'music = '.$_SESSION['music'].'<br>';
       unset($_SESSION['couleur']);
       unset($_SESSION['gout']);
       unset($_SESSION['type']);
       session_destroy();
     ?>
  ....
```

- EXECUTION



recapitulatif de la session :
couleur = vert

gout = salé
music = techno

- **session** PHP4!
  - permet conserver des données d'une page visitée à une autre pour même utilisateur , même si plusieurs utilisateurs visitent le site en même temps
  - chaque utilisateur accédant à la page web "reçoit" un identifiant de session unique SID
  - L'implémentation de la session (par cookie, ou par propagation dans l'URL) est transparente au programmeur PHP
  - Pour chaque session, on peut créer des variables qui doivent être préservées d'une page à une autre : elles sont sérialisées après l'exécution du script PHP et restaurées lors de l'exécution d'un nouveau script.
- session_start()
  crée une session
  ou restaure la session trouvée sur le serveur
- $_SESSION
  superglobal de la session
  pour ajouter une variable de session : $_SESSION['nom_variable'] = valeur;
  pour consulter une variable de session : $_SESSION['nom_variable']
  pour supprimer une variable de session : unset($_SESSION['nom_variable'])
- session_destroy()
  détruit une session
  détruit toutes les données associées avec la session courante
- SID
  est la constante de l'identifiant de session de la forme "id=valeur"

cookie

```
<?php
  if (isset ($_COOKIE['compteur']))
    setcookie("compteur",$_COOKIE['compteur']+1);
  else
    setcookie("compteur",1);
?>
<html>
  <head>
    <title>cookie</title>
  </head>
  <body>
    <h2 align=center>cookie</h2>
    <p>
      la page a &eacute;t&eacute; visit&eacute;e
      <?php echo $_COOKIE['compteur']; ?> fois.
    </p>
  </body>
</html>
```

- EXECUTION

  la page a été visitée fois.

-

  puis relisez la page plusieures fois

  la page a été visitée 4 fois.

- **cookie**
  - les cookies sont des petites informations déposées dans le navigateur de la machine cliente
- setcookie (nom, valeur, date_d_expiration, chemin, domaine, secure)
  définit un cookie qui sera envoyé avec le reste des en-têtes de la communication HTTP
  seul le premier argument nom est obligatoire

la date d'expiration du cookie ;s ans, le cookie est détruit à la fermeture du navigateur : c'est alors un cookie de session.
le chemin est celui de la page qui a inscrit le cookie ; ceci détermine la page qui pourra relire le cookie.
le nom de domaine permet d'identifier le cookie parmi tous ceux stockés sur la machine ;ceci détermine la page qui pourra relire le cookie.
secure est une valeur booléenne indique si l'accès au cookie est protégé.
 Il faut appeler cette fonction avant toute balise <HTML> ou <HEAD> car les cookies doivent "passer" avant les autres en-têtes.
 le cookie ne sera accessible qu'au chargement de la prochaine page, ou au rechargement de la page courante.

- $_COOKIE
  superglobal des cookies
  anciennement $HTTP_COOKIE_VARS obsolète en PHP4 mais accessible
  pour consulter une variable de cookie : $_COOKIE['nom_cookie'] qui donne sa valeur
  pour supprimer un cookie, il faut changer la date d'expiration pour une date dépassée.