Qu'est-ce que PHP ?

Annie Danzart

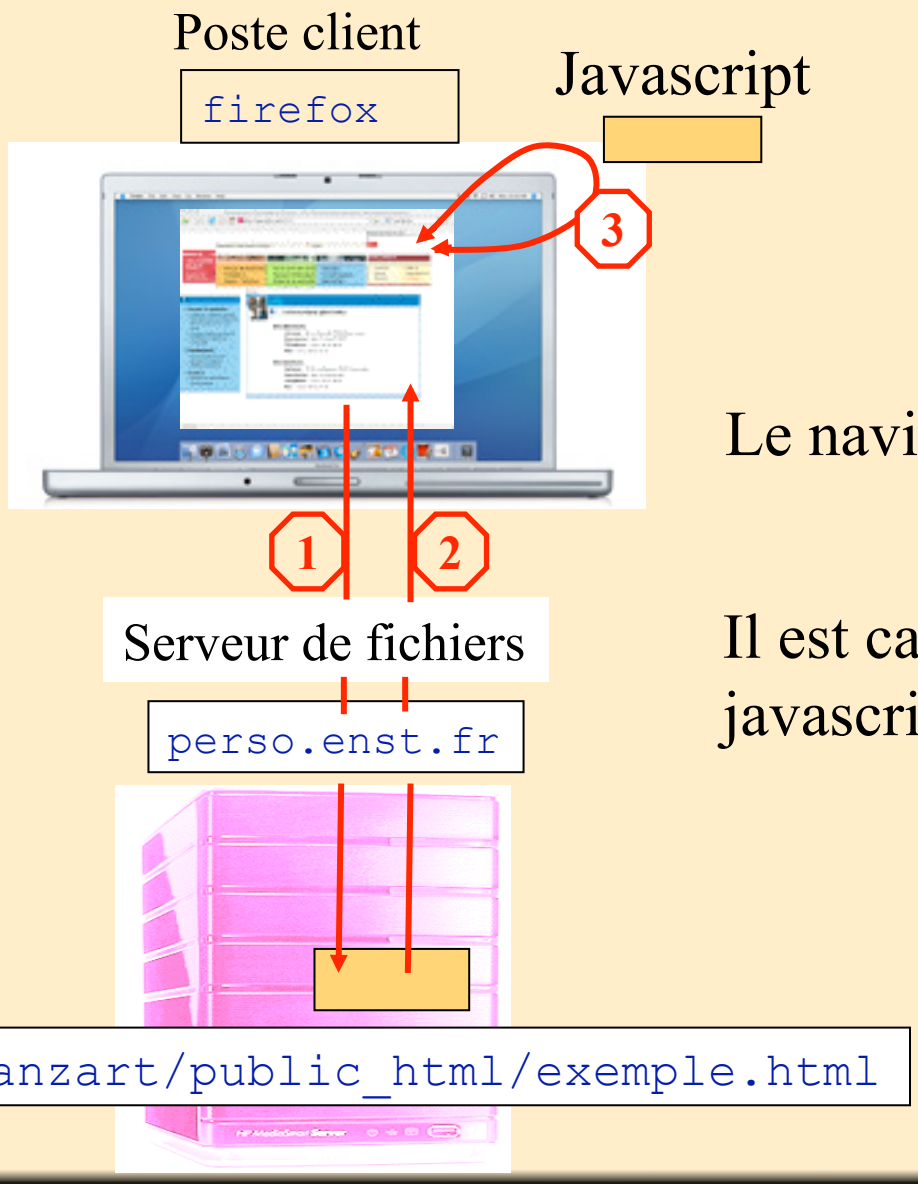
Annie.Danzart@enst.fr
http://www.infres.enst.fr/~danzart/php/

Qu'est-ce que PHP ?

- 1994 : Rasmus Lerdorf, Personnal Home Pages
  Pre Hypertext Processor
- Langage de scripting traité par le serveur, orienté web
- Module mod_php d'Apache
- Gratuit, libre de droits (license GNU GPL)
- Nombreuses extensions (sgbd, ldap, gif, pdf, smtp, …)
- Syntaxe proche de celle de C, Java, Perl
- Comparable à ASP, asp2php
- Extensible
- Des milliers de sites de par le monde
- Versions 3, 4, 5, 6 en préparation

Comment ça marche ? affichage sans serveur

`file:///infres.enst.fr/~danzart/exemple.html`



Le navigateur affiche les pages html

Il est capable d'interpréter des scripts javascript

Comment ça marche ? utilisation d'un serveur

http://www.infres.enst.fr/~danzart/exemple.html

Poste client

firefox

Serveur Apache

www.infres.enst.fr

httpd

1

4

2

3

Serveur de fichiers

perso.enst.fr

~danzart/public_html/exemple.html

Le serveur reçoit la requête
et la traite en envoyant au client
la page demandée

Comment ça marche ? interprétation d'un script

http://www.infres.enst.fr/~danzart/exemple.php

Poste client

`firefox`

Serveur Apache

`www.infres.enst.fr`

`httpd`
`mod_php5`

1

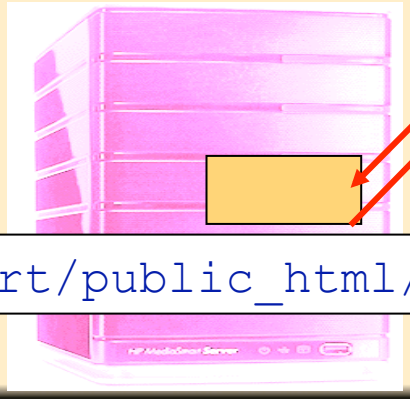
5

2

3

Serveur de fichiers

`perso.enst.fr`



`~danzart/public_html/exemple.php`

Le serveur passe le fichier au module php avant de le retourner au client

Comment ça marche ? Les bases de données

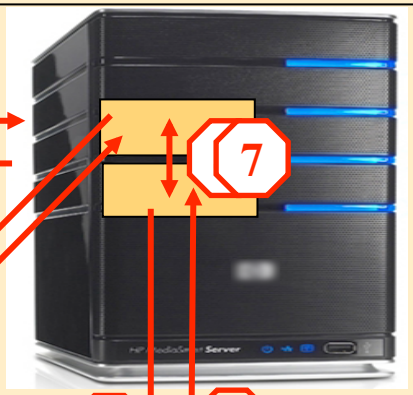
http://www.infres.enst.fr/~danzart/exemple2.php

Poste client

firefox

Serveur Apache

www.infres.enst.fr



httpd
mod_php5

1

8

7

5

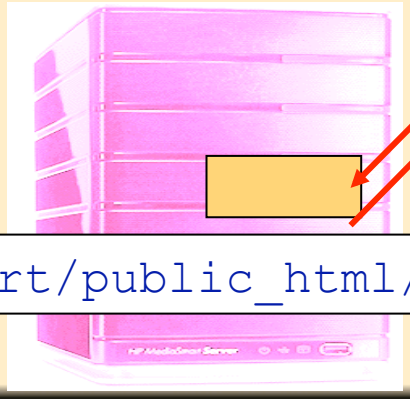
6

2

3

Serveur de fichiers

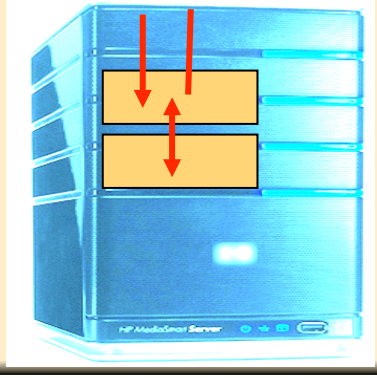
perso.enst.fr



~danzart/public_html/exemple2.php

Serveur Mysql

mysql.infres.enst.fr



mysqld
bases

Page php

✓ Une page html
✓ Des scripts php.
✓ L'ensemble des scripts d'une page utilise le même contexte
✓ Chaque script peut générer du code html qui s'intègre à sa place

✓L'extension de la page doit être php, ou php3, phtml, php4, php5 sinon le serveur ne sait pas qu'il doit interpréter des scripts

- Page html, extension du fichier
- Environnement, résultat
- **Formulation des scripts**
- Variables, types des variables, constantes

<?php … ?>	Ecriture la plus standard
<? … ?>	Encore très utilisée.
<script language="php"> … </script>	A la Javascript …
<%php … %>	Compatible avec ASP
`<?=$nom?>`	Ecriture raccourcie, pas toujours correctement interprétée

Les scripts, les variables

- Page html, extension du fichier
- Environnement, résultat
- Formulation des scripts
- Variables, types des variables, constantes

```
<html>
  <head><title>Bienvenue</title>
  </head>
<body>
<?php
// commentaire
   $nom="Martin";
?>
<h1 align="center"> Bonjour
    <?php echo $nom; ?> </h1><br>
Il est <?php echo date("H:i"); ?>
   <br>
</body>
</html>
```

Exécution du script

Remarque : la fonction date

✓Types de base: entiers, réels, chaînes de caractères
✓Tableaux, tableaux associatifs
✓Objets

✓Les noms de variables commencent par le caractère $
✓php est sensible à la casse
✓Un nom peut commencer par une lettre ou un _
✓Il peut comporter des caractères, des chiffres et _

✓Les variables ne sont pas typées, n'ont pas besoin d'être déclarées
✓Elles ont une valeur par défaut qui dépend du contexte d'utilisation

- Page html, extension du fichier
- Environnement, résultat
- Formulation des scripts
- **Constantes, variables, types de variables**

```
<html>
  <head><title>Bienvenue</title>
  </head>
<body>
<?php
// définition d'une constante
 define("Salut","Bonjour a
   tous<br>");
 echo "<font color=\"red\">";
 printf("Hello ! ".Salut);
?>
</font></body>
</html>
```

Exécution du script

Remarques:

- l'opérateur . de concaténation
- insertion de balises html
- échappement de caractères

Les variables

- **Affectation d'une valeur**
- Opérateurs arithmétiques, logiques
- Tests sur les variables
- Référence, déréférencement

```php
$chaine = "0"; // chaîne "0" (ASCII 48)
$nombre = 0;
$nombre++; // nombre 1 (entier)
$nombre+=1; // entier (2)
$nombre = $nombre + 1.3;
// réel (3.3)
$nombre = 5 + "3 petits cochons";
// entier (8 !) effet de bord
$nombre = (int) "3 petits cochons";
 // 3 : conversion en entier
```

Simple et double quote

Chaîne dynamique

```php
$val = 'Rusti';
echo "Hello $val"; // Hello Rusti
echo 'Hello $val';
// Hello $val
```

Exécution d'un script

Source du script

- Affectation d'une valeur
- Opérateurs arithmétiques, logiques
- **Tests sur les variables**
- Référence, déréférencement

`isset($a)`: teste si la variable est définie. (0=non, 1=oui)

`unset($a)`: supprime la variable et désalloue la mémoire utilisée

`gettype($a)`:retourne le type de la variable
→`string, integer, double, array, object`
→`string` si la variable n'est pas définie

`is_double, is_string, is_int, is_float,is_object` :
testent le type de la variable

Les variables

- Affectation d'une valeur
- **Opérateurs arithmétiques, logiques**
- Tests sur les variables
- Référence, déréférencement

Arithmétiques

```
$a + $b - $c
$a * $b / $c
$a % $b
$a++    Incrément de 1
$b--    Décrément de 1
```

Assignation de valeur

```
=, +=, -=, *=, /=, %=, .=
```

Comparaison entre expressions

```
==, !=, <, >, <=, >=
```

Concaténation

```
$chaine="votre nom est ".$val;
```

Logiques

```
not, !, &&, ||, AND, XOR, OR
```

rque : précédence de `or` sur `||`

```
$a = foo() || bar() or die();
```

Binaires bitwise

```
&, |, ^, ~, >>, <<, -
```

Autres

`@` : contrôle d'erreur

`? :` opérateur ternaire

Les variables

- Affectation d'une valeur
- Opérateurs arithmétiques, logiques
- Tests sur les variables
- **Référence, déréférencement**

```
<html>
   <head><title>Bienvenue</title>
   </head>
<body>
<?php
// référence
    $var ="bonjour ";
    $$var= "UE";
// création de la variable $bonjour
    echo $var." ".${$var}." "."INF347<br>";
    echo $var." ".$bonjour." "."INF347";
?>
</body>
</html>
```

Création de variables dynamiques dont on ne connaît pas le nom *a priori*.

Exécution du script

Les tableaux

- **Tableaux, tableaux dynamiques**
- Parcours
- Fonctions sur les tableaux
- Tableaux associatifs

```
$fruits[0]= "pomme";
$fruits[1]= "banane";

$fruits[] = "orange";
      // équivaut a
$fruits[2]= "orange"

$fruits= array(0=> "pomme",
   1=>"banane",2=> "orange" );

$fruits= array( "pomme",
   "banane", "orange" );
```

A priori, les indices commencent à 0.

Pas besoin de déclarer la dimension ni le type des valeurs.

Si on ne donne pas d'indice, l'indice pris en compte sera celui qui suit la valeur la plus haute (0 si rien n'a encore été mis).

L'initialisation peut se faire « en bloc », en précisant ou non les indices.

Les tableaux dynamiques sont très utiles lors de l'utilisation de champs de type « checkbox » dans un formulaire.

Exécution d'un script

- Tableaux, tableaux dynamiques
- Parcours
- **Fonctions sur les tableaux**
- Tableaux associatifs

`sizeof($t) = count($t)`	→ Taille d'un tableau
`is_array($t)`	→ Est-ce un tableau ?
`reset($t) end($t) current($t)` `next($t) prev($t)`	→ Positionnement dans un tableau : représenté par une liste de doublets (indice,valeur)
`sort, rsort, ksort, krsort, usort` `shuffle($t)`	→ Tri ascendant, descendant sur la valeur, sur la clé, avec une fonction utilisateur → Mélange aléatoire
`max($t) min($t)`	→ Calcul des valeurs extrêmes
`arraywalk($t,"nomfonction")`	→ Applique une fonction à toutes les valeurs d'un tableau

- Tableaux, tableaux dynamiques
- **Parcours**
- Fonctions sur les tableaux
- Tableaux associatifs

```
$t = array("I","N","F","3","4","7");
sort($t);
for ($i=0;$i<count($t);$i++)
   echo "t[$cle]=".$t[$cle]."\n";

sort($t);
reset($t);
while(list($cle,$valeur)= each($t))
   echo "t[$cle]=".$t[$cle]."\n";
```

L'ajout de `"\n"` dans l'affichage introduit un saut à la ligne dans le source.

`$a = each($t)` renvoie l'index et la valeur courante dans un tableau à 2 éléments; `$a[0]` contient l'index, `$a[1]` la valeur.

`list($cle, $valeur, ... )` construit un tableau temporaire à partir des variables scalaires passées en argument.

Exécution d'un script

- Tableaux, tableaux dynamiques
- Parcours
- Fonctions sur les tableaux
- **Tableaux associatifs**

```
$mois["Janvier"]= 1;
$mois["Février"]= 2;
$mois[] =
    array("Janvier"=>1,"Février"=>2,
    "Mars"=>3);

while (list($cle,$val) = each($mois))
     echo "<br> No de $cle : $val ";

foreach ($mois as $cle => $val)
  echo "<br> No de $cle : $val ";
```

`key($t)`:index de l'élément courant du tableau

`uasort($t,"f_comp")`: tri à l'aide d'une fonction en gardant les indices associés

Les indices du tableau sont alors des chaînes de caractères qui pourront être traitées en tant que telles.

L'emploi de la fonction `each` s'avère importante pour parcourir les tableaux associatifs.

On peut aussi utiliser la boucle foreach.

Exécution d'un script

Les instructions

- **Instructions conditionnelles**
- Booléens
- Boucles
- foreach
- Ruptures de séquence

```
if ($a > $b) {
    echo "a supérieur à b";
} elseif ($a == $b) {
    echo "a égal à b";
} else {
    echo "a inférieur à b";
}

if ($a > $b):
    echo "a supérieur à b";
elseif ($a == $b) :
    echo "a égal à b";
else :
    echo "a inférieur à b";
endif
```

```
$jour=date("l");
$mois=date("F");
switch ($mois) {
case "January" : … ;break;
case "February": … ;break;
     . . . .
case "November": … ;break;
case "December": … ;break;}
printf(date(" Y "));

switch ($mois)
case "January" : … ;break;
. . . .
case "December": … ;break;
endswitch
```

Opérateurs:

de comparaison entre expressions : `==, !=, <, >, <=, >=`

Logiques : `not, !, &&, ||, AND, XOR, OR`

Valeurs logiques par défaut:

Entier	`0`	→	`false`	`≠0`	→ `true`
Réel	`0.0`	→	`false`	`≠0.0`	→ `true`
Chaîne de caractères	`""`	→	`false`	`≠ ""`	→ `true`
Tableau ou objet	`Null`	→	`false`	`≠ Null`	→ `true`

Les instructions

- Instructions conditionnelles
- Booléens
- **Boucles**
- foreach
- Ruptures de séquence

```php
<?php
$n=28;

for ($I=1;$I<$n;$I++)
{
 print("$I,<br>");
}

for ($I=1;$I<$n;$I++):
 print("$I,<br>");
endfor;

?>
```

```php
<?php
$n=28;

while ($I<$n) {
 print("$I,<br>");
 $I=$I+1;
}

while ($I<$n) :
 print("$I,<br>");
 $I=$I+1;
endwhile;
?>
```

```php
<?php
$n=28;

do
{
  print("$I,<br>");
 $I=$I+1;
}
while ($I<$n);

?>
```

- Instructions conditionnelles
- Booléens
- Boucles
- **foreach**
- Ruptures de séquence

```php
<?php
 foreach ($tab as $val)
 {
  echo "valeur : $val<br/>\n";
 }
?>
```

est équivalent à :

```php
<?php
 reset($tab);
 while (list(,$val)=each($tab))
 {
  echo "valeur : $val<br/>\n";
 }
?>
```

```php
<?php
 foreach ($tab as $cle => $val)
 {
  echo "clé : $cle<br/>\n";
  echo "valeur : $val<br/>\n";
 }
?>
```

est équivalent à :

```php
<?php
 reset($tab);
 while (list($cle,$val)=each($tab))
 {
  echo "clé : $cle<br/>\n";
  echo "valeur : $val<br/>\n";}
?>
```

`continue :`

arrêter l'itération

passer à la suivante

`break [n] :`

arrêter l'itération courante

sortir de la boucle courante

(ou des n boucles imbriquées)

`exit() :`

interruption inconditionnelle du scr`ipt`

A utiliser avec modération !!!

- **Déclaration, valeur de retour**
- Variables locales/globales/statiques
- Transmission des arguments
- Transmission par référence

```
<?php
 function division($n1, $n2)
{
  $resultat=$n1/$n2;
  return $resultat;
}
$r=division(15,3);
function division($n1=1, $n2=2)
{
  $resultat=$n1/$n2;
  return $resultat;
}
 $r=division(5);
 $r=division(5,8);
?>
```

Un argument vide n'est pas forcément un argument absent.

Exécution d'un script

- Déclaration, valeur de retour
- Transmission des arguments
- **Variables locales/globales/statiques**
- Transmission par référence

Variables locales:

N'existent que dans le corps de la fonction

Variables globales:

Précédées du mot clé `global`

Établissent un lien direct avec des variables déclarées à l'extérieur de la fonction

Variables statiques:

Précédées du mot-clé `static`

Exécution d'un script

- Déclaration, valeur de retour
- Variables locales/globales/statiques
- **Transmission des arguments**
- **Transmission par référence**

Arguments :

La transmission des variables se fait par valeur

Transmission par référence, 2 possibilités :

Soit le nom de la variable est précédé d'un `&` au moment de l'appel

Soit le nom de la variable est précédé d'un `&` au moment de la déclaration de la fonction (standard)

Depuis php5, les arguments par défaut peuvent être transmis par référence

Exécution d'un script

- **Déclaration, attributs, méthodes, constructeur, instanciation**
- **Héritage, redéfinition de méthodes**

```
class sortie {
var $titre="Bonjour";
// constructeur
function sortie($init="Coucou")
 {  $this->titre=$init; }

function debut()
 {echo "<html><head>
       <title>$this->titre";
  echo "</title></head><body>";}
function fin()
 {echo "</body></html>";}
}
```

instanciation et référenciation

```
$a = new sortie("Hello");
$a->debut();
echo "<p> il est
   ".date("H:i")."</p>";
$a->fin();
```

Exécution d'un script

- Déclaration, attributs, méthodes, constructeur, instanciation
- **Héritage, redéfinition de méthodes**

```php
class sur_s extends sortie {
function sur_s($titre="coucou")
 {
  //constructeur
   parent::sortie($titre);}

function prg($texte)
 {
  echo "<p align='center'>
              $texte";
  echo "</p>";}

function debut()
{
  parent::debut();
  echo "debut de page";
}

}
```

instanciation et référenciation

```php
$a = new sur_s("Hello");
$a->debut();
$a->prg("il est ".date("H:i"));
$a->fin();
```

Exécution d'un script

Les fichiers inclus

`include(nomfichier ) :`
inclut systématiquement le fichier à l'endroit indiqué

`require(nomfichier ) :`
ne l'inclut que si c'est justifié (cas d'une instruction conditionnelle)

`include_once(nomfichier ),`
`require_once(nomfichier ):`
Ne fera l'inclusion qu'une seule fois (cas d'appels multiples de bibliothèques de fonctions

le fichier inclus débute en html `pur`

Il faut toujours donner un nom et une extension qui comportent une indication sur :
l'inclusion (`inc`) : permet de savoir que c'est un fichier à inclure
le type de fichier (`php`) : évite que ce fichier soit lisible intégralement

Par exemple : `bibli_inc.php`

Exécution d'un script

Les formulaires

- **Déclaration, balises, méthode**
- **Récupération des valeurs transmises**
- Cas particulier du type `image`
- Les `checkbox`

```
<form method="post"
   action="execution.php">
<H2>Questionnaire</H2>
Prénom ? <input name="prenom"
   value="xxx"><p>
Votre nom ? <input name="nom"
   value="yyy"><p>
Votre couleur favorite ?
 <select name="couleur">
 <option selected>blanc
 <option>jaune
 <option>orange
 <option>vert
 </select><p>
Cliquez sur <input type="submit"
   value="Valider"> pour valider.
</form>
```

La balise `form` délimite le formulaire
Son attribut `method` détermine le mode de transmission des paramètres
L'attribut `action` précise le programme qui s'exécute lors de la validation

Balises:
`form, input (type, name, value), select (name), option (value), …`

Les balises d'un formulaire

Exemple de formulaire

- Déclaration, balises, méthode
- Récupération des valeurs transmises
- **Cas particulier du type `image`**
- Les `checkbox`

```
<H1>Formulaire2 </H1>

<form method="post"
action="execution.php">
<H2>Questionnaire</H2>
Votre prénom ? <input name="prenom"
value="xxx"><p>

Cliquez sur <input type="image"
name="im1" src="bouton.gif"> pour
valider.
</form>
```

L'image remplace le bouton de soumission du formulaire.

Les coordonnées du point où l'utilisateur aura cliqué seront : **`$im1_x`** et **`$im1_y`**

Exemple de formulaire

- Déclaration, balises, méthode
- Récupération des valeurs transmises
- Cas particulier du type `image`
- Les **`checkbox`**

```
<form method="post"
action="execution3.php3">
<H2>Questionnaire : votre choix</
H2>
<input type="checkbox"
name="fruits[]" value="pomme">
Pommes
<input type="checkbox"
name="fruits[]" value="poire">
Poires
Cliquez sur <input type="submit"
   value="Valider"> pour valider.
</form>
```

Le nom du champ de type `checkbox` prévoit que les valeurs choisies seront stockées dans un tableau dynamique.
Seules les valeurs choisies seront transmises sous le nom `Fruits[]`

L'appel de la fonction php `count($fruits)` permettra de savoir combien de valeurs sont dans le tableau.

Exemple de formulaire

Source du script

Les fichiers

- **Ouverture/fermeture**
- **Lecture/écriture**
- **Traitement de chaînes**
- **Fichiers locaux**

```php
<?php
 /* ouverture en écriture */
   $fichier=fopen($nom_f,"w+");
fclose($fichier);
?>
/* ouverture en lecture */
 $fichier=fopen($nom_f,"r");
 /* lecture d'une ligne faisant au
plus 255 caractères */
 $ligne=fgets($fichier,255);

 /* fermeture du fichier */
 fclose($fichier);
 ?>
```

```php
<?php
// ouverture de la source (lecture)
$source=fopen($nom_source,"r");
// ouverture de la cible (écriture)
$cible=fopen($nom_cible,"w+");
 /* lecture de lignes faisant au
plus 255 caractères */

 while ($ligne=fgets($source,255))
  {
// copie de la source sur la cible
     fputs($cible,$ligne);
  }
  fclose($source);
  fclose($cible);
?>
```

Le fichier traité a la structure suivante :

```
 nom prenom;dept/bureau/poste
```

```php
<?php
// ouverture de la source
$nom_source = "base.txt";
$source=fopen($nom_source,"r");
// définition des séparateurs pour délimiter les éléments de la ligne (ici l'espace, le; et le /)
   $sep=" /;";
// lecture d'une ligne
   $ligne=fgets($source,255);
   fclose($source);
// premier élément
    $nom = strtok($ligne,$sep);

// délimitation des suivants
  $pno = strtok($sep);
  $dep = strtok($sep);
  $bur = strtok($sep);
  $post = strtok($sep);
// ligne initiale
  print("Ligne : ".$ligne);
// affichage du résultat
  print("<p>Nom : ".$nom);
  print("<br> Prénom : ".$pno);
  print("<br> Dépt : ".$dep);
  print("<br> Bureau : ".$bur);
  print("<br> Poste : ".$post);
?>
```

- Ouverture/fermeture
- Traitement de chaînes
- Lecture/écriture
- **Fichiers locaux**

```
/* affichage des entrées d'un
répertoire */
 $repertoire="rep";
 $d = dir($repertoire);
 echo "chemin : $d->path<br>";
 while ($entry = $d->read())
     echo "$entry<br\n";
 $d->close();

Ou
 $d = opendir($repertoire);
 while ($entry = readdir($d))
     echo "$entry<br\n";
 $close($d);
```

On peut utiliser la liste de ces entrées pour créer des liens vers ces fichiers ou répertoires

```
echo "<a href=\"$repertoire/
$entry\">$entry</a><br\n"
```

Un exemple

- **Les cookies**
- Les sessions
- Les variables d'environnement

Création d'un cookie, Durée de validité

```
int setcookie (string name, string
value , int expire , string path ,
string domain , int secure )
```

Le code PHP suivant doit impérativement avoir été envoyé avant l'envoi de tout code html, c'est à dire en début du fichier).

```
<?php
  setCookie("date","25/04/2000");
  setCookie("heure","17h30");
// ces instructions écrivent les
cookies $date et $heure

setcookie("TestCookie","Valeur de
test");
  setcookie("TestCookie",$value,
time()+3600);
  /* expire dans une heure */
```

- Il est prudent d'utiliser un nom qui ne soit pas aussi celui d'une autre variable (d'un formulaire par exemple)
- La date calculée dépend de celle du client !
- Les cookies ne seront accessibles qu'au chargement de la prochaine page, ou au rechargement de la page courante.
- Il est possible de stocker un tableau dans un cookie

Un exemple

- Les cookies
- Les sessions
- Les variables d'environnement

Fonctions de gestion de session

`session_start()` : démarre une session
`session_destroy()` : met fin à une session
`session_name()` : retourne le nom de la session ou démarre une nouvelle session sous le nom passé en paramètre.
`session_id()` : comme la précédente avec l'identificateur
`session_is_registred()`: vérifie si une variable est enregistrée dans la session courante
`session_unregister()`: supprime une variable

Les variables de session sont ensuite accessibles grâce au tableau associatif `$_SESSION[]`.

Exemple

```
<?php
 session_start();
 $nom=session_name();
 $ref=132;
 $_SESSION["ref"]=$ref;
?>
<?php
 session_start();
 $nom=session_name();
 echo "valeur =".$_SESSION["ref"];
?>
```

rque: ce mécanisme fonctionne aussi avec des tableaux

Un exemple

Environnement

- Les cookies
- Les sessions
- **Les variables d'environnement**
- `phpinfo()`

-->> données stockées dans des variables permettant au programme d'avoir des informations sur son environnement.

-->> informations sur le type de serveur, son administrateur, la date à laquelle le script a été appelé, l'adresse IP et le type de navigateur du client, …

-->> créées par le serveur à chaque fois que le script PHP est appelé, le serveur les lui fournit en paramètres cachés lors de l'exécution de l'interpréteur.

-->> deux catégories :

celles qui dépendent du client
celles qui dépendent du serveur

`$_GET[], $_POST[], $_COOKIE[], $_SESSION[]`

Mais aussi:

`$REMOTE_ADDR` : adresse du client qui a fait la requête

`$REMOTE_REFERER` : nom de la page qui a référencé celle qui s'affiche

`$USER_AGENT` : nom du navigateur utilisé ainsi que sa version

`$LAST_MODIFIED` : dernière date de modification

`$_SERVER, $_REQUEST, …`

Images

- **Formats acceptés: png, gif, jpg**
- **Fonctions de gd**
- **Tracés, sauvegardes**
- **Images dynamiques**

Analyse d'une image

```
$nom="pervenches.png";
$c=getimagesize($nom);
print("L=".$c[0]." pix <br>");
printf("H=".$c[1]." pix <br>");
switch ($c[2]){
 case 1 : print("gif");break;
 case 2 : print("jpeg");break;
 case 3 : print("png");break;
 default : printf("non reconnu");
}
```

Ouverture d'une image

```
$nom="pervenches.png";
$im=imagecreatefrompng($nom);
```

Récupération de la couleur en (x,y)

```
<?php
$coul=imagecolorat($im,$x,$y);
$RVB=imagecolorsforindex($im,$coul);

printf("RVB: (%d, %d, %d)",$RVB["red"],
$RVB["green"],$RVB["blue"]);
?>
```

Couleur la plus proche dans une image

```
<?php
  $coul=imagecolorclosest($im,$R,$V,$B);
?>
```

Couleur de transparence dans une image

```
<?php
$coul=imagecolorclosest($im,$R,$V,$B);
imagecolortransparent($im,$coul);
?>
```

- Formats acceptés: png, gif, jpg
- Tracés, sauvegardes
- Fonctions de gd
- Images dynamiques

Incrustation d'un texte en un point

```
imagestring($im,$font,$x,$y,
            $texte,$coul);
```

Tracés sur une image

```
 //quart d'ellipse
imagearc($im,$x,$y,$l,$h,0,90,$coul);
 //ligne
imageline($image,$x1,$y1,$x2,$y2,
            $coul);
 //polygone
imagepolygone($im,array($x1,$y1,$x2,$y2,
                $x3,$y3),3,$coul);
 //polygone plein
imagefilledpolygone($im,array($x1,$y1,
            $x2,$y2,$x3,$y3),3,$coul);
 //rectangle
imagepolygone($im,$x1,$y1,$x2,$y2,
                        $coul);
```

```
 $nom="livre.png";
 printf("<img src=\"$nom\"
border=1><p>");
 $im=imagecreatefrompng($nom);

 $couleur=imagecolorclosest($im,
                    0,0,0);
 imagecolorset($im,$couleur,
                      255,0,0);
. . . .
 imagepng($im,"nouveau.png");

// attention aux droits sur les
fichiers ainsi créés
```

- Formats acceptés: png, gif, jpg
- Tracés, sauvegardes
- Fonctions de gd
- **Images dynamiques**

Génération d'une image

```
<?php
  $nom="livre.png";
  $im=imagecreatefrompng($nom);

  $couleur=
    imagecolorclosest($im,
                      0,0,0);
  imagecolorset($im,$coul,
                    255,0,0);

  imagepng($image);
 ?>
```

Dans ce cas, le fichier ne doit contenir que le script php de génération d'image.
Il sera ensuite utilisé exactement comme un nom de fichier image dans une page html.

```
<html>
  <head><title>Bienvenue</title></
head>
<body>
<img src="generation.php">
</html>
```

Le script php peut dépendre de variables qui seront transmises directement par la méthode `POST`

```
<img src="generation.php?coul=red">
```

Envoi de mails

Attention aux envois de mails automatiques en nombre important qui encombrent les boîtes aux lettres !!!!

```php
<?php
$dest= "client@magasin.fr";
$sujet= "Confirmation";
$texte= "Votre commande a bien
été effectuée";
$entete= "From: $exp\nreplyTo:
              $exp\nContent-
type:text/html";
mail($dest,$sujet,$texte,
$entete);
?>
```

Il faut veiller à toujours envoyer une page html à l'utilisateur

```html
<HTML>
<HEAD>
<TITLE>Formulaire envoyé!</TITLE>
</HEAD>
<BODY>
<H1 align="cente"r>Merci,<?php echo
$nom; ?> </H1>
<P align="center">Votre message a
été envoyé.</P>
</BODY>
</HTML>
```

Merci de votre attention !!!	Tutoriel: http://www.infres.enst.fr/~danzart/php/ Les exemples du cours: http://www.infres.enst.fr/~danzart/CPM2/PHP-MYSQL/

Envoi de mail : attachements

– entête : type de document, format

mail.php

```
$to =  "$email\n";
$headers =  "From: tst@trume.php\n";
$subject= "Un fichier est attaché à ce mail";
$content= "Bonjour,\n .contenu du message ..\n";
// fichier à attacher ?
if(file_exists( "attache.txt"))
{
   $fic =  "attache.txt";
   $taille = filesize($fic);
   $type = filetype($fic);

  //place tout le fichier dans une variable
   $PtFicAttache = fopen($fic, "r");
   $FicDansChaine = fread($PtFicAttache, $taille);
   fclose($PtFicAttache);

  //encodage en base64 (compatibilité système 7-bit)
   $fic_attache = base64_encode($FicDansChaine);
  //conformation à la RFC 2045 (norme des mails)
  $fic_attache = chunk_split($fic_attache, 64 ,"\r\n");

 //En-tête du fichier. Espaces et \n importants!!!
  $EnteteFicAttache =  "\n--some random text\nContent-
Type: ".$type. ";\n name=\"".$fic. "\"\n". "Content-
Transfer-Encoding: base64\nContent-Disposition:
attachment;\n filename=\"".$fic. "\"\n\n";
}
```

```
 //sinon, on n'attache rien
 else
  {
      $fic_attache= "";
      $EnteteFicAttache= "";
  }
//Termine le mail en attachant le fichier et en
// indiquant l'en-tête du fichier attaché
    $content .= $EnteteFicAttache.$fic_attache;

//Ajouter dans le 4e paramètre de la fonction mail()
l'en-tête générale du mail
//chr(13) est un retour chariot, chr(10) un saut de
ligne et chr(9) une tabulation

$headers .=  "MIME-Version: 1.0\nContent-Type:
multipart/mixed;".chr(13).chr(10).chr(9). "boundary=
\"some random text\"\n";
 if(mail( "$to",  "$subject", $content, $headers))
print( "Mail Sent to $email.");
 else print( "Mail has not been Sent ($email)<br>error
in function : mail(\"$to\", \"$subject\", $content,
$headers); <br>This is probably because the sendmail
service is not available.<br>click here for the <a href=
\"http://bistoy.trume.com/Trume/prog/php/\">working
version</a>.");?>
```

– entête : type de document, format

telecharge.html

```
<!-- Début du formulaire de sélection -->
 <form action="telecharge.php"
enctype="multipart/form-data" method=POST>
  <input type="hidden" name="count" value="<?
echo $count; ?>">
  <input type="hidden" name="stringNames"
value="<? echo $stringNames; ?>">
  <input type="hidden" name="MAX_FILE_SIZE"
value="1000000">
 <input type="file" name="userfile">
 <input type="submit" value="Télécharger le
fichier"> </form>
```

telecharge.php

```
<?php
 // prendre un fichier,
 //le stocker dans un répertoire temporaire
if (($userfile!=none)&&($userfile!=""))
   {
    if (!isset($fileNames))
      {
        $fileNames = array();    }
       // le fichier est sauvegardé dans /tmp
       // (ou le répertoire défini par le serveur)
       // il est alors recopié localement
       //  le répertoire "tmp" doit être présent
       // et autoriser l'écriture
       copy ($userfile,"tmp/$userfile_name");
       // suppression du fichier de /tmp
       unlink ($userfile);
       $fileNames[$count] = $userfile_name;
       $count++;
       $stringNames = implode(",",$fileNames);
   }
?>
```

- Page html, extension du fichier
- Formulation des scripts
- **Environnement, résultat**
- Variables, types des variables, constantes

```
<html>
  <head><title>Bienvenue</title>
  </head>
<body>
<?php
// commentaire
   $nom =  "Martin";
?>
<h1 align="center"> Bonjour
    <?php echo $nom; ?> </h1><br>
Il est <?php echo date("H:i"); ?>
</body>
</html>
```

```
<html>
  <head><title>Bienvenue</title>
  </head>
<body>
<h1 align="center"> Bonjour
   Martin </h1><br>
Il est 15:12
<br>
</body>
</html>
```