CS WEB

Ch 4 – PHP

I. INTRODUCTION 2

A. PRODUCTION DYNAMIQUE DE HTML 2
B. TECHNIQUES DE PRODUCTION DYNAMIQUE DE HTML 2
C. PHP, « SCRIPTING » COTE SERVEUR 3

II. PHP – UTILISATION 4

A. INSTALLATION SEPAREE OU INTEGREE (PACKAGES) 4
B. FICHIERS SOURCE PHP, SCRIPT 5
C. L'INSTRUCTION DE BASE : ECHO, PRINT 5
D. DECLARATION ET TYPAGE DES VARIABLES 6
E. APOSTROPHE (« ' » OU GUILLEMET (« " ») 7

III. PHP – INSTRUCTIONS ET STRUCTURES DE CONTROLES 8

A. SIMILITUDE C/C++ - APPORTS 8
1. *Opérateur conditionnels et opérateurs relationnels 8*
2. *Structures de contrôles conditionnelles : if, switch 8*
3. *Structures de contrôles répétitives : for, while, do while 9*
B. LES EXPRESSIONS CHAINES DE CARACTERES EN PHP : CONCATENATION 10
C. LES TABLEAUX EN PHP 10
1. *Tableaux classiques 10*
2. *Tableaux associatifs 11*

IV. TRAITEMENTS DES DONNEES REÇUES DU NAVIGATEUR : FORMULAIRES ET LIENS 13

A. PASSAGE PAR FORMULAIRE OU PAR LIEN 13
B. PHASES DE DEFINITION D'UN FORMULAIRE ET DE SON TRAITEMENT 14
C. RECEVOIR LES DONNEES D'UN FORMULAIRE – METHODES POST ET GET 15
D. RECEVOIR LES DONNEES ENVOYEES DANS UN LIEN HYPERTEXTE : $_GET 15
E. TEST D'EXISTENCE DES ARGUMENTS ATTENDUS : ISSET 16
F. DECLARATION AUTOMATIQUE DES VARIABLES DE $_POST ET $_GET : EXTRACT 17

V. PHP – ACCEDER AUX BASES DE DONNEES 17

A. ETAPES POUR L'UTILISATION DE MYSQL 17
1. *Demande de connexion au SGBD – mysql_connect 17*
2. *Sélection de la base de données – mysql_select_db 17*
3. *Exécution des requêtes et traitement des résultats – mysql_query, mysql_fetch_row, mysql_fetch_array 18*
4. *Récupération d'une valeur de clef ajoutée – mysql_insert_id 19*
5. *Libération de la connexion 20*
B. AUTRES SGBD ET ODBC 20
1. *Connexion natives aux grands SGBD du marché 20*
2. *Connexion via ODBC 20*

VI. SAUVEGARDER DES INFORMATIONS LE TEMPS D'UN DIALOGUE- SESSIONS ET COOKIES 21

A. PASSAGE PAR CHAMPS DE FORMULAIRE OU ARGUMENTS DE LIEN HYPERTEXTE 21
B. SAUVEGARDE FICHIER OU BASE DE DONNEES 22
C. COOKIES – FICHIERS DU COTE POSTE CLIENT 22
1. *Mémorisation d'un cookie : setcookie() 22*
2. *Récupérer une valeur de cookie : $_COOKIE 23*
3. *Effacer un cookie : setcookie() sans valeur 23*
4. *Localiser les fichiers cookies sur votre ordinateur 23*
D. SESSIONS – FICHIERS DU COTE SERVEUR 24
1. *Démarrer une session : session_start() 24*
2. *Enregistrer une donnée : $_SESSION[ ] 25*
3. *Récupérer une donnée : $_SESSION[ ] 25*

4. Supprimer une variable de session : unset() *25*
5. Supprimer une session : session_destroy() *26*

VII. PHP – CODE MODULAIRE, PROTECTION DES DOSSIERS 26

A. INCLUSION DE CODE 26
1. include *26*
2. require_once *26*
3. require *26*
B. FONCTIONS 26
C. CLASSES 27

VIII. PHP – UPLOAD DE FICHIER, PARCOURS DE REPERTOIRES COTE SERVEUR 27

A. UPLOAD DE FICHIERS 27
B. PARCOURS DE REPERTOIRES SUR LE SERVEUR 28

IX. GESTION DES FICHIERS 'TEXTE' 29

A. MODES D'OUVERTURE DU FICHIER 29
B. OUVRIR, LIRE LIGNE PAR LIGNE, ECRIRE, FERMER 30
C. LECTURE ET ECRITURE GLOBALE DU CONTENU 30

X. PHP – EXTENSIONS 30

A. PRODUIRE UN DOCUMENT PDF 30

XI. PHP – TRAITEMENT DE FORMULAIRES AVEC CHAMPS MULTIPLES 32

A. CHAMPS INPUT MULTIPLES : TABLEAUX DE VALEURS 32
B. SELECTION MULTIPLES DANS SELECT/OPTION 32
C. SELECTION MULTIPLES DANS LES CHECKBOX 33

XII. PHP – COMPLEMENTS 33

I. Introduction

A. Production dynamique de HTML

Les pages Web statiques ont eu leur heure de gloire à l'éclosion du Web, à l'heure des sites web institutionnels donc l'objectif était la présentation d'une entreprise, ou la production de rapports statiques : les modifications étaient alors peu fréquentes.

Cependant elles se sont avérées lourdes lors qu'il s'est agi d'afficher des catalogues de produits ; de plus, la prise en charge des interactions avec l'internaute n'était pas possible (recherches, commandes, etc.).
Il a alors fallu associer des langages de programmation afin que ces flux HTML soient produits « à la volée » (« on the fly »).
Le contenu des pages Web générées peuvent donc être à chaque fois différent.

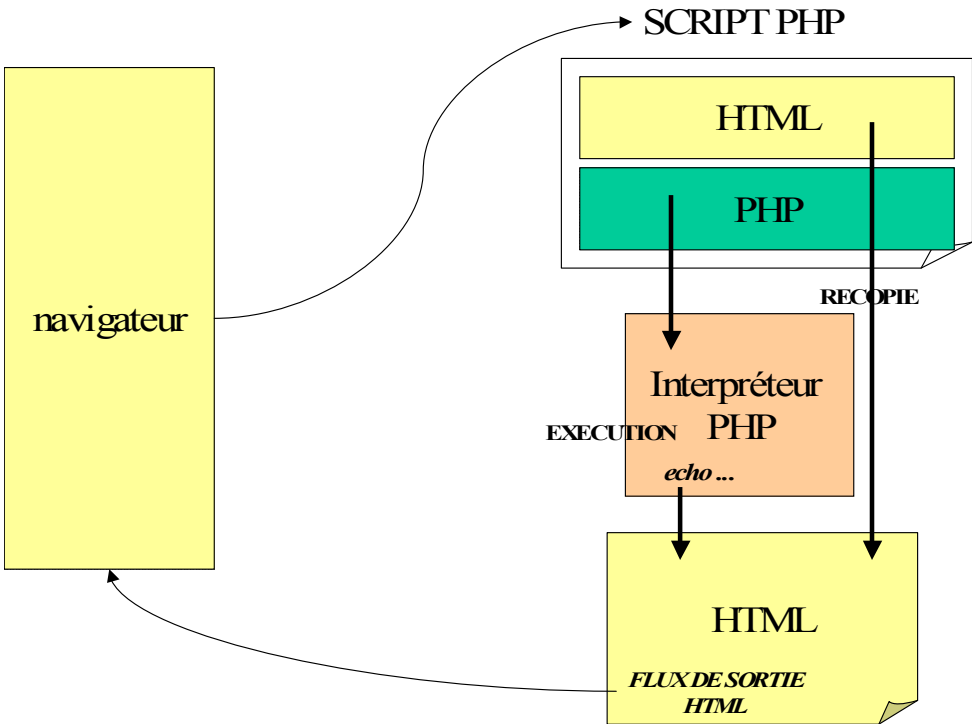
B. Techniques de production dynamique de HTML

Il existe plusieurs manières de produire dynamiquement des pages :

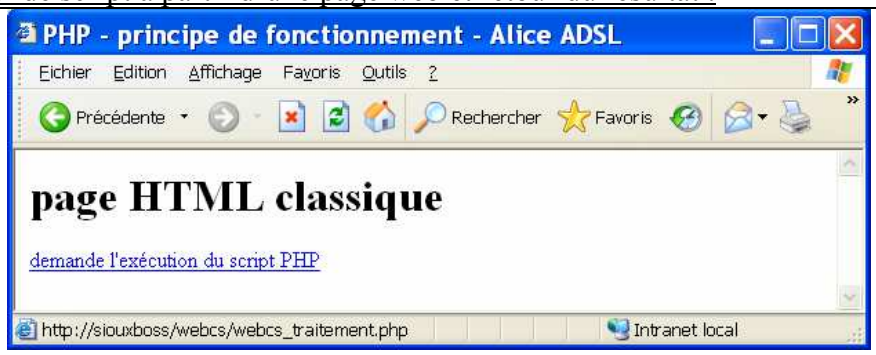
- Des **programmes** directement **exécutables** (ils ont été compilés, cf. CGI)
- Des **scripts interprétés** (on parle de script lorsque le programme source est interprété à chaque fois qu'il est lancé)
    - Des scripts côté serveur Web, avec les langages PHP, Perl, Java (JSP), VB (ASP), etc.

- Des scripts coté client (navigateur), avec les langages Javascript, VBscript

C. PHP, « scripting » côté serveur



Exemple d'appel de script à partir d'une page web et retour du résultat :




```
<html>
  <head><title>PHP - principe de fonctionnement</title></head>
  <body>
    <h1>page HTML classique</h1>
    <a href="webcs_traitement.php">demande l'exécution du script PHP</a>
  </body>
</html>
```

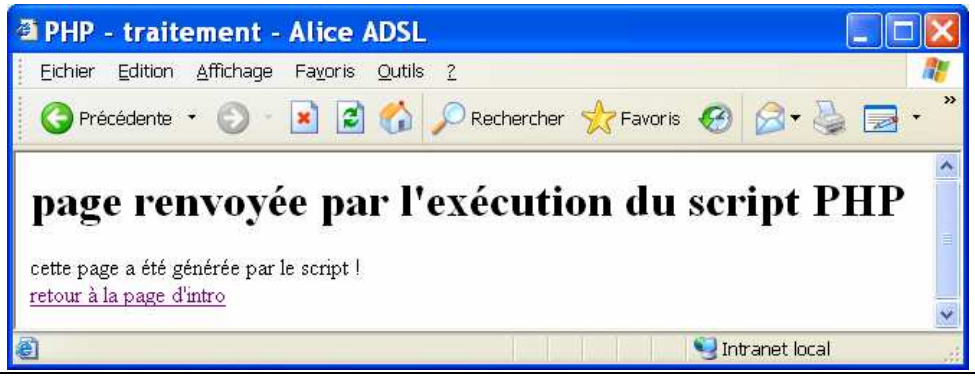
L'utilisateur clique sur le lien hypertexte demande l'exécution du script PHP, le serveur web reçoit cette demande, et lance l'exécution du script suivant :

```
<html>
  <head><title>PHP - traitement</title></head>
  <body>
<?php
echo '<h1>page renvoyée par l\'exécution du script PHP</h1>';
echo '<div>';
```

```
echo 'cette page a été générée par le script !';
echo '</div>';
echo '<a href="webcs_intro.html">retour à la page d\'intro</a>';
?>
  </body>
</html>
```

*L'exécution du script PHP renvoie les balises HTML du fichier et celles produites par l'exécution des instructions PHP (instruction **echo**).*
Le résultat de cette exécution (un flux HTML) possède une structure de page web classique et sera renvoyée vers le navigateur :

```
<html>
  <head><title>PHP - traitement</title></head>
  <body>
<h1>page renvoyée par l'exécution du script PHP</h1><div>cette page a été
générée par le script !</div><a href="webcs_intro.html">retour à la page
d'intro</a>
  </body>
</html>
```



II. PHP – utilisation

A. Installation séparée ou intégrée (Packages)

L'installation d'une configuration « serveur Web » nécessite au minimum le logiciel serveur :

- Apache (téléchargeable)
- Microsoft IIS (Internet Information Server).

Si on souhaite développer un site dynamique, il est nécessaire d'ajouter le « moteur d'exécution » ou l'interpréteur du langage de programmation choisi. Dans le cas de PHP, on devra installer le module PHP (téléchargeable).
Les sites dynamiques ne peuvent se passer d'une base de données : l'installation d'un serveur de données (SGBD) :

- MySQL (téléchargeable) ;
- Posgresql (téléchargeable).

La difficulté (toute relative) consiste ensuite à faire communiquer ces 3 composants (cf. tutoriels sur le Web).

Des packages préconfigurés (EasyPHP, XAMPP) permettent l'installation et une configuration simplifiée d'un serveur Web sur un poste de travail :

- le serveur Web (Apache),

- l' interpréteur PHP,
- le SGBD MySQL.

Ce type d'installation est adapté pour tester localement, sur son ordinateur, le fonctionnement d'un site en développement.

Dans un navigateur, on adressera le serveur local par http://localhost ou http://127.0.0.1 .

B. Fichiers source PHP, script

Un script PHP est un fichier texte dont l'extension est généralement .PHP. Dans ce fichier source, on va trouver du code HTML et du code PHP. Des balises spécifiques permettent d'isoler le code PHP du code HTML : le code PHP se trouve à l'intérieur d'une balise ouverte par **<?php** et fermée par **?>.**

```
...du HTML ...
<?php
// ici le code PHP
?>
...du HTML ...
```

On peut trouver plusieurs blocs de code PHP (ou bien que du code PHP) dans une page :

```
...du HTML ...
<?php
// ici du code PHP
?>
...du HTML ...
<?php
// ici du code PHP
?>
...du HTML ...
```

Chaque instruction PHP se termine par le signe de ponctuation **;**.

Syntaxe et fonctions du langage : http://fr.php.net/

C. L'Instruction de base : echo, print

L'instruction **echo** (ou **print**) est l'instruction qui permet la production dynamique du code HTML à renvoyer au navigateur.

L'instruction **echo** accepte des arguments : les textes à afficher sous forme

- de chaines de caractères encadrées par des apostrophes ou des guillemets
- ou de nom de variables.

L'exécution du code PHP qui suit va simplement renvoyer la chaîne de caractère « bonjour à tous ! >» vers le navigateur, ce qui va se traduire par l'affichage simple de ce texte, sans mise en forme :

```
<?php
echo 'bonjour à tous ! ' ;
?>
```

Ici renvoi de « <h1>bonjour à tous !</h1 »; ce va se traduire par l'affichage en titre du texte (élement HTML **h1**) :

```
<?php
echo '<h1>bonjour à tous !</h1>' ;
?>
```

Lorsqu'on doit afficher le caractère ' (quote) : il faut ajouter le caractère d'échappement \ (backslash) afin de signaler que le caractère qui suit doit être affiché tel quel (et ne correspond pas à la fin de la chaine de caractères):

```
<?php
echo '<h1>bonjour à \'Paul\' !</h1>' ;
?>
```

Ici une variable $nom vaut 'Paul', on souhaite afficher le bonjour à Paul :

```
<?php
$nom='Paul' ;
echo '<h1>bonjour à ', $nom,  ' !</h1>' ;
?>
```

Ici, utilisation de la fonction **date** du langage PHP, qui renvoie une date sous la forme « Friday 5th February 2009 09:00:00 AM » (cf. http://fr3.php.net/date):

```
<?php
echo date('l jS F Y h:i:s A');
?>
```

Différences entre l'instruction **echo** et l'instruction **print** :

1. Paramètres
    1. On peut passer à **echo** plusieurs expressions séparées par des « , »
    2. **print** n'accepte qu'un seul paramètre

D. Déclaration et typage des variables

	C/C++	PHP
Nommage des variables	Le nom doit commencer par une lettre ou _	Le nom doit commencer par **$**
Déclaration et type de données	**Typage fort** : **le type doit être donné à la déclaration** Une variable ne change pas de type	**Typage dynamique** : **le type est déterminé à la chaque affectation** Une variable peut changer de type en fonction des affectations successives

Exemple :

```
...code HTML ...
<?php
// 1.déclaration d'une variable initialisée à 12 (numérique)
$Vnote = 12 ; // déclaration d'une variable numérique

// 2.déclaration d'une variable (chaine de caractères)
$Vtexte1 = 'vous avez ' ;
```

```
// 3.déclaration d'une variable (chaine de caractères)
$Vtexte2 = ' sur 20.' ;

// 4.déclaration d'une variable (chaine de caractères)
// sa valeur est la concaténation de 3 variables de types diff.
$Vphrase1 = $Vtexte1 . $Vnote . $Vtexte2 ;

// 5.déclaration d'une variable chaine
// Sa valeur est une chaine dans laquelle les variables seront
// remplacées par leur valeur.
$Vphrase2 = "vous avez $Vnote sur 20." ;

// 6.déclaration d'une variable (booléenne)
// 'True' si la valeur de $note est sup. à 15
// 'False' si la valeur de $note est inf.ou egale à 15
$VbonneNote = ($note>15) ;
?>
...code HTML ...
```

Dans la déclaration 4 : le « **.** » est opérateur de concaténation (mettre bout à bout plusieurs données pour former une chaîne de caractères)

E. Apostrophe (« ' » ou guillemet (« " »)

Si une **chaîne de caractère est encadrée par des apostrophes**, une instruction **echo** renvoie directement cette **valeur telle qu'elle est écrite**.
Si une **chaîne de caractère est encadrée par des guillemets**, une instruction **echo** explore le contenu de cette chaîne pour **remplacer les variables** qu'elle contient par leur valeur.

```
<html>
  <head><title>PHP - guillemets</title></head>
  <body>
<?php
$i=1;
echo "<h$i>titre h$i</h$i>";
echo '<h$i>titre h$i</h$i>';
?>
  </body>
</html>
```

L'exécution du script précédent produit le résultat suivant :

```
<html>
  <head><title>PHP - guillemets</title></head>
  <body>
<h1>titre h1</h1>
<h$i>titre h$i</h$i>
  </body>
</html>
```

III.PHP – Instructions et structures de contrôles

PHP est un langage qui tire son origine du langage C.
On y retrouvera donc

- Toutes les structures de contrôles traditionnelles.
- Le caractère « ; » à la fin de chaque instruction
- Les caractères de délimitation des commentaires :
  - // : commentaire ligne
  - /* */ : commentaire multi lignes

A. Similitude C/C++ - apports

1. Opérateur conditionnels et opérateurs relationnels

Opérateur	*Signification*
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

Opérateur relationnel (met en relation plusieurs conditions)	*Signification*	*Symbole équivalent*
and	Et	&&
or	Ou	||

2. Structures de contrôles conditionnelles : if, switch

Exemple **if else**:

```
<?php
$Vnote =12 ; // déclaration d'une variable et affectation
if ($Vnote < 10 )
   echo '<b>vous n'avez pas la moyenne</b>' ;
else
   echo '<b>vous avez la moyenne</b>' ;
// suite du traitement en PHP
?>
```

Exemple 1 **switch**: (= selon la valeur de ...choisir ...)

```
<?php
$Vniveau='b' ; // déclaration d'une variable numérique
$Vtexte='' ; // déclaration d'une variable chaîne de caractères
switch ($Vniveau)
{
  case 'a' : $Vtexte='très bien' ;
          break ;
  case 'b' : $Vtexte='bien' ;
          break ;
  case 'c' : $Vtexte='moyen' ;
          break ;
```

```
  default  : $Vtexte='insuffisant' ;
          break ;
}
   echo '<b>' . $Vtexte . '</b>';
// suite du traitement en PHP
?>
```

Exemple 2 **switch**: (extension par rapport au C/C++ où la variable testée est de type int ou char)

```
<?php
$Vtexte='bien' ;
$Vcommentaire='' ;
switch ($Vtexte)
{
  case 'très bien' : $Vcommentaire ='félicitations !' ;
          break ;
  case 'bien' : $ Vcommentaire ='poursuivez ainsi' ;
          break ;
  case 'moyen' : $ Vcommentaire ='accentuez vos efforts' ;
          break ;
  default  : $ Vcommentaire ='il faut vous reveiller' ;
          break ;
}
   echo '<b>' . $ Vcommentaire. '</b>';
// suite du traitement en PHP
?>
```

Attention cependant à l'utilisation des chaînes de caractères dans une instruction **switch** : un espace en trop dans les valeurs du **case** et le traitement ne produit pas les résultats attendus !!!

3. Structures de contrôles répétitives : for, while, do while

Exemple « boucle » **for** :

```
<?php
// boucle for
for ($i=1 ;$i<=6 ;$i++)
{
   echo "<h$i>titre h$i</h$i>";
}
?>
```

Exemple « boucle » **while** :

```
<?php
// boucle while
$i=1 ;
while ($i<=6 )
{
   echo "<h$i>titre h$i</h$i>";
   $i++ ;
}
```

Exemple « boucle » **do while** :

```
...du HTML ...
<?php
// boucle do while
$i=1 ;
```

```
do
{
   echo "<h$i>titre h$i</h$i>";
   $i++ ;
} while ($i<=6 ) ;
```

B. Les expressions chaînes de caractères en PHP : concaténation

On peut construire une chaîne de caractères complète en assemblant plusieurs chaînes de caractères, des variables (de types différents) et des expressions : on parle de **concaténation**

```
< ?php
$Vprenom="Paul" ; // chaine de caractère
$Vage=20 ; // nombre entier
$Vsalaire=1525.35 ; // nombre réel
$Vmajeur=true ; // booléen

$Vphrase = $Vprenom . ' est agé de ' . $Vage . ' ans. ' ;
$Vphrase.= 'Il gagne ' . ($Vsalaire * 12) . ' par an.' ;
if ($Vmajeur==true)
   { $Vphrase.= 'Il est majeur';}
echo $Vphrase ;
?>
```

Le point (« . ») est un opérateur de concaténation : il met bout à bout 2 expressions.
L'opérateur « .= » vient ajouter à la variable à sa gauche la valeur qui se trouve à sa droite.

C. Les tableaux en PHP

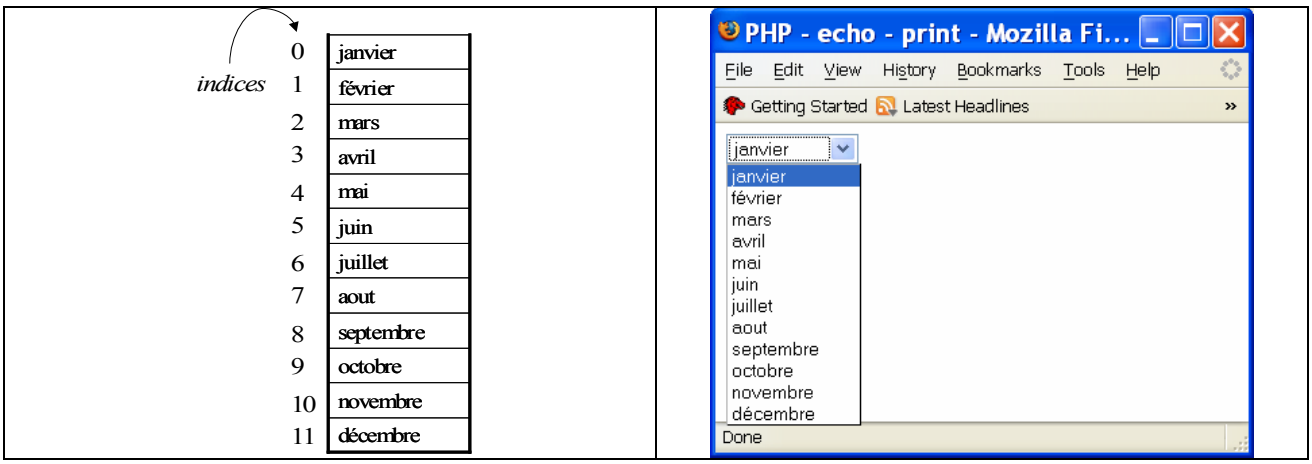
1. Tableaux classiques

Exemple d'utilisation : .

```
<?php
// déclaration et initialisation
$Tmois=array('janvier','février','mars','avril','mai',
'juin','juillet','aout','septembre','octobre','novembre','décemb
re');
echo '<select name=\"selMois\">' ;
// la fonction count retourne le nombre d'éléments de $Tmois
for ($i=0 ;$i<count($Tmois) ;$i++)
{
   echo '<option value=$i++>',$Tmois[$i] , '</option>';
}
echo '</select>' ;
?>
```

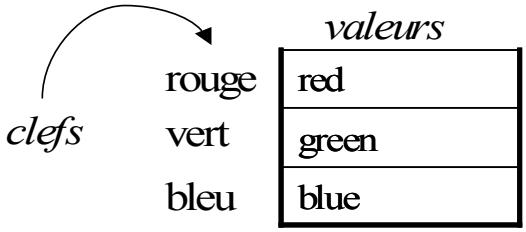
Tableau et résultat produit : liste déroulante (balise HTML select et option utilisées dans un formulaire)

le tableau $Tmois et son contenu en mémoire l'indice commence à 0	*le résultat du script PHP*



2. Tableaux associatifs

Le tableau associatif est un tableau auquel une **clef** est associée à chaque élément.



a) Déclarer un tableau associatif

Déclarer un tableau associatif :

```php
<?php
$Tcouleur=array('rouge'=>'red','vert'=>'green','bleu'=>'blue');
// suite du PHP
?>
```

ou bien :

```php
<?php
$Tcouleurs['rouge']='red' ;
$Tcouleurs['vert']='green' ;
$Tcouleurs['bleu']='blue';
// suite du PHP
?>
```

b) Parcourir un tableau associatif

Parcourir un tableau associatif :

```php
<?php
// ...
reset($Tcouleurs); // se placer sur le premier élément
for ($i = 0; $i < count($Tcouleurs); $i++)
{
 $clef = key($Tcouleurs); // clef de l'element courant
 echo '<br />clef = ', $clef, ', valeur = ', $Tcouleurs[$clef];
 next($Tcouleurs); // se déplace sur l'element suivant
}
```

```
?>
```

ou bien en utilisant la structure de contrôle '**foreach**'

```
<?php
// ...
foreach ($Tcouleurs as $clef => $valeur)
{
 $clef = key($Tcouleurs); // clef de l'element courant
 echo '<br />clef = ', $clef, ', valeur = ', $Tcouleurs[$clef];
 next($Tcouleurs); // se déplace sur l'element suivant
}
?>
```

c) Fonctions retournant un tableau associatif

Certaines fonctions PHP renvoient comme valeur de retour un tableau associatif :

```
<?php
$Tmois=array('janvier','février','mars','avril','mai',
'juin','juillet','aout','septembre','octobre','novembre','décemb
re');

// obtenir la date (un tableau associatif, clef :day,mday,year)
$Vdate =getdate() ;

// Récupérer le jour du mois (« mday » pour month day)
$Vjour =$Vdate["mday"] ;

// Récupérer le mois (« mon » pour month)
$Vmois =$Vdate["mon"] ;

// Récupérer l'année
$Vannee =$Vdate["year"] ;

echo '<h1>nous sommes le ',$Vjour,' ',$Tmois[$Vmois -1] ,' ',
$Vannee;
?>
```

d) Tableaux associatifs $_POST, $_GET, $_SESSION, $_COOKIE

Certains tableaux associatifs sont initialisés automatiquement par le serveur Web et on peut en disposer dans le code PHP :

- Les tableaux **$_POST** et **$_GET** : ils contiennent les valeurs des champs de formulaires ou des liens hypertextes envoyés par le navigateur
- Le tableau **$_SESSION** : il permet l'accès aux variables de sessions enregistrées
- Le tableau **$_COOKIE** : permet d'accéder aux données d'un cookie

Cf. le traitement des formulaires

IV. Traitements des données reçues du navigateur : formulaires et liens

Le formulaire permet à l'utilisateur d'interagir avec le système en entrant des valeurs, sélectionnant des éléments dans des boites de listes, etc..

Les données envoyées par correspondent à tous les champs qui auraient pu être saisis par l'utilisateur, mais aussi les champs cachés des formulaires.

A la réception par le serveur, celui-ci met à la disposition de PHP un tableau associatif où

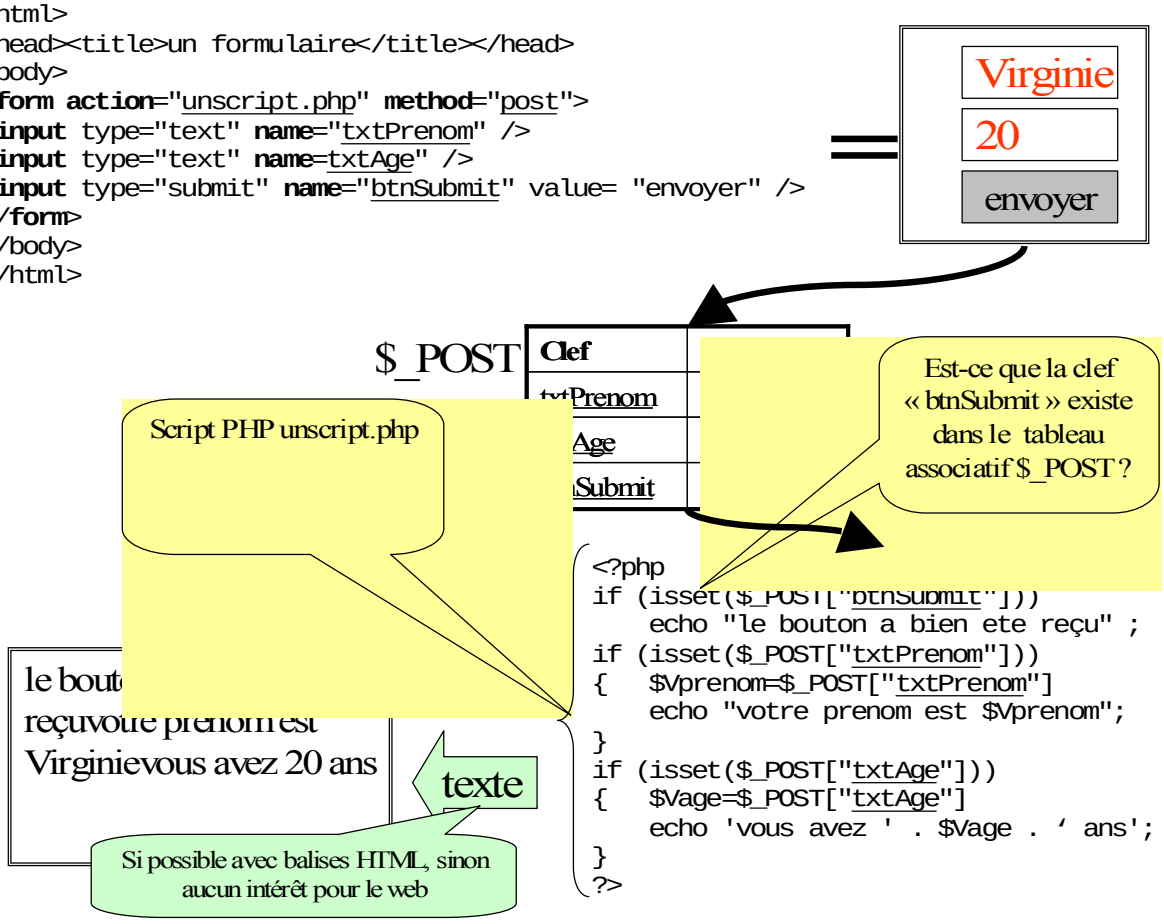
- La clef correspond au nom du champ envoyé (propriété **name** des éléments HTML **input**, **textarea**, **select**) ;
- La valeur correspond à la saisie de l'utilisateur (ou la valeur du champ caché : propriété **value** des éléments de formulaire)

A. Passage par formulaire ou par lien

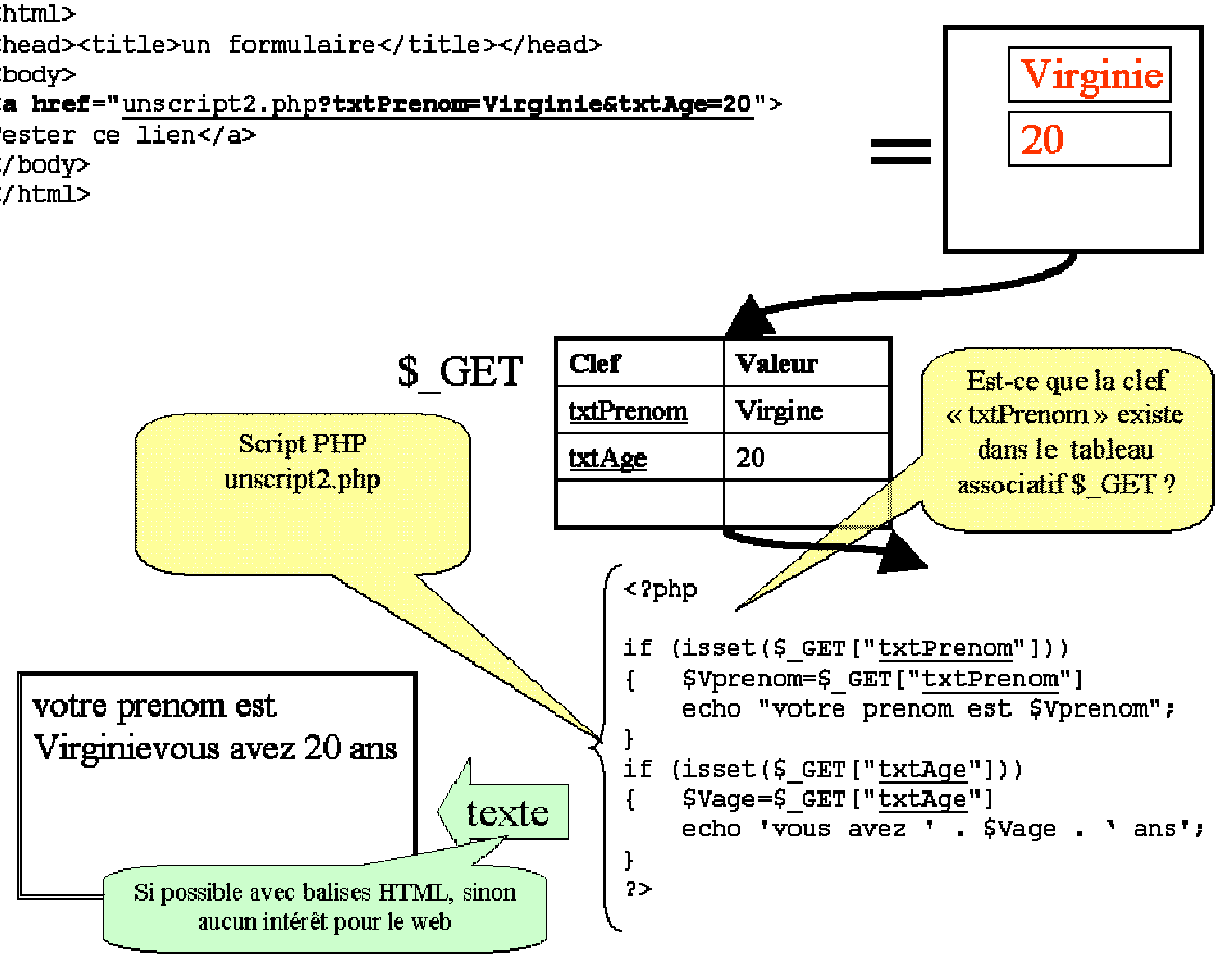
Une requête Web peut associer des valeurs passées par l'intermédiaire d'un formulaire ou bien à la fin d'une URL :

- par **formulaire** : ce sont les champs de formulaire qui seront utilisés pour constituer les données envoyées pour traitement.
- par lien hypertexte : on trouvera à la suite de l'adresse URL, un « ? » suivi une liste de valeurs sous la forme suivante **clef=valeur&clef=valeur&....**

```
<html>
<head><title>un formulaire</title></head>
<body>
<form action="unscript.php" method="post">
<input type="text" name="txtPrenom" />
<input type="text" name=txtAge" />
<input type="submit" name="btnSubmit" value= "envoyer" />
</form>
</body>
</html>
```



```
<html>
<head><title>un formulaire</title></head>
<body>
<a href="unscript2.php?txtPrenom=Virginie&txtAge=20">
Tester ce lien</a>
</body>
</html>
```



B. Phases de définition d'un formulaire et de son traitement

Pour établir un traitement de données par formulaire :

1. Définir le formulaire : (cf support HTML)
   a. Les champs de formulaire
   b. Le nom de la page de traitement
   c. La méthode passage des arguments du formulaire à la page de traitement
2. Définir la page de traitement :
   a. Vérifier le formulaire d'origine (est-ce que l'appel vient bien d'une bonne page du site ?) (cf **HTTP_REFERER**)
   b. Tester la présence des champs requis (cf **isset**) : en cas d'erreur quitter avec un message technique donnant un code erreur qui vous guidera ensuite pour la correction)
   c. Tester les valeurs reçues (cf **if**, opérateurs de comparaison, opérateurs de test : **is_string**, **strlen**, etc.) : en cas d'erreur, renvoyer un message
   d. Traiter les données
      i. En cas d'enregistrement de chaînes de caractères dans une base de données MySQL : entourer les chaînes de caractères avec des guillemets et protéger les guillemets saisis dans le texte : **mysql_real_escape_string** (protéger les guillemets)
   e. Afficher les données
      i. En cas de récupération de zones de texte, convertir les caractères spéciaux HTML : **htmlspecialchars** en entités HTML afin d'avoir un affichage correct

C. Recevoir les données d'un formulaire – méthodes POST et GET

Deux méthodes de passages des valeurs du navigateur vers le serveur web sont définies :

- La méthode **POST** : les valeurs envoyées sont cachées et invisibles à l'utilisateur
- La méthode **GET** : les valeurs sont passées en clair après le nom de la page demandée et seront visibles dans la barre d'adresse du navigateur

La récupération des valeurs en PHP sera effectuée dans un tableau automatiquement mis à disposition de PHP par le serveur web ; son nom sera différent selon la méthode de « postage » :

- Dans le cas de la méthode **POST** , le tableau s'appellera **$_POST**
- Dans le cas de la méthode **GET** , le tableau s'appellera **$_GET**

Exemple de formulaire (HTML), le click sur le bouton envoie le formulaire à la page « traitementpost.php » :

```
... HTML ...
<form name="frmSaisie" action="traitement.php" method="POST">
  <input type="text" name="frmNom" />
  <input type="text" name="frmAge" />
  <input type="submit" name="btnValider"  value="Valider" />
</form>
... HTML ...
```

Réception par le script 'traitementpost.php' : le tableau $_POST est disponible est automatiquement créé au chargement du script : c'est un tableau qui contient la liste des arguments envoyés par le formulaire avec en face de chaque argument, la valeur qui a été saisie par l'utilisateur :

```
< ?php
// récupération des variables « postées »
$Vnom=$_POST ["frmNom"];
$Vage=$_POST ["frmAge"];
echo $Vnom . ' age :'. $Vage;
// suite du traitement en PHP
?>
```

D. Recevoir les données envoyées dans un lien hypertexte : $_GET

Exemple de lien hypertexte (HTML), le click sur le lien envoie la requête à la page spécifiée avec une liste de couples argument=valeur : :

```
... HTML ...
<a href="afficherProd.php?idProd=123">produit 123</a>
... HTML ...
```

Réception par le script 'afficherProd.php' : une variable tableau est automatiquement créé au chargement du script : c'est un tableau qui contient la liste des arguments envoyés par le lien avec en face de chaque argument, la valeur qui a été saisie par l'utilisateur :

```php
< ?php
// récupération des variables « postées » par GET
$Vprod=$_GET["idProd"];
// suite du traitement en PHP

?>
```

Exemple de lien hypertexte (HTML), le click sur le lien envoie la requête à la page spécifiée avec une liste de couples argument=valeur : :

```
... HTML ...
<a href="traitementget.php?nom=virginie&age=20">produit 123</a>
... HTML ...
```

Réception par le script 'traitementget.php' : une variable tableau est automatiquement créé au chargement du script : c'est un tableau qui contient la liste des arguments envoyés par le lien avec en face de chaque argument, la valeur qui a été saisie par l'utilisateur :

```php
< ?php
// récupération des variables « postées » par GET
$Vnom=$_GET["nom"];
$Vage=$_GET["age"];
// suite du traitement en PHP

?>
```

E. Test d'existence des arguments attendus : isset

Le fonction isset permet de tester l'existence d'une variable. Elle est très utilisée pour vérifier que les tableau $_POST et $_GET (mais aussi $_SESSION et $_COOKIE) contiennent bien une certaine clef .
Cela permet de tester l'existence des variables « postées » avant de pouvoir les exploiter (Il est indispensable de vérifier la présence des variables envoyées par le formulaire : erreur de saisie de nom dans les champs de formulaire, malversations, etc.) :

```php
<?php
// récupération des variables « postées »
$Vnom='inconnu' ;
$Vage=0 ;
if (isset($_POST["frmNom"]) && isset($_POST["frmAge"]))
{
   $Vnom=$_POST["frmNom"];
   $Vage=$_POST["frmAge"];
}
else
{
  echo '<b style=\"color: red\"> données non reçues</b>';
}

echo $Vnom . ' age :'. $Vage;
// suite du traitement en PHP
?>
```

F. Déclaration automatique des variables de $_POST et $_GET : extract

La fonction **extract** permet d'extraire et déclarer automatiquement les variables se trouvant dans un tableau associatif $_POST ou $_GET.
On peut écrire :

```
<?php
// récupération des variables « postées »
extract($_POST) ;
// toutes les noms de variables correspondant aux clefs sont créées et les affectations des valeurs réalisées
?>
```

V. PHP – accéder aux bases de données

A. Etapes pour l'utilisation de MySQL

1. Demande de connexion au SGBD – mysql_connect

La fonction **mysql_connect** effectue une demande de connexion à un SGBD

- sur un serveur dont l'adresse est précisée (adresse IP, nom de serveur ou localhost)
- pour un certain compte : nom de compte et mot de passe

La fonction renvoie un retour

- Soit false : la connexion a échoué :
  - Adresse du serveur incorrecte ou SGBD non démarré
  - Compte inexistant ou mot de passe incorrect
- Soit un lien vers cette connexion.

L'utilisation du caractère '@' devant les noms des fonctions permet la gestion manuelle survenant sur une instruction d'accès à la base de données (gestion plus contrôlée des erreurs).

```
< ?php
// Se connecter au SGBD
$serveur="localhost"; // localhost si même machine
$utilisateur="root";
$motpasse="secret";
$lien=@mysql_connect("$serveur","$utilisateur","$motpasse");
if (!$lien) {
    die('Connexion impossible : ' . mysql_error());
}
// suite du traitement en PHP
?>
```

2. Sélection de la base de données – mysql_select_db

La fonction **mysql_select_db** permet de choisir une des bases de données gérées par le SGBD en précisant simplement son nom.
La fonction renvoie un retour

- Soit false : la demande a échoué :

  - La base de données spécifiée n'existe pas (mauvais nom)
  - Les droits attribués au compte ne lui permettent pas d'accéder à cette base de données
- Soit un lien vers cette base de données.

```
< ?php
// (partie 1 demande de connexion au SGBD)
...
// Sélectionner la base de données
$nombdd="test"; // nom de la base de données
$select_base=@mysql_select_db("$nombdd" ,$lien);
if (!$select_base) // si çà ne s'est pas bien passé...
{  echo '<h1>echec selection base de données</h1>' ;
   @mysql_close($lien);
   exit;
};
// suite du traitement en PHP
?>
```

3. Exécution des requêtes et traitement des résultats – mysql_query, mysql_fetch_row, mysql_fetch_array

Une fois la base de données accessible, nous allons pouvoir soumettre des requêtes SQL au SGBD pour manipuler cette base de données.

Il va s'agir de

- constituer une chaîne de caractères qui va contenir la requêtes SQL à soumettre au SGBD
- envoyer cette chaine de caractère et récupérer en retour un résultat :
  - soit le retour vaut false : une erreur s'est produite :
    - la requête est incorrecte
    - l'action demandée est interdite pour le compte connecté
  - sinon l'ensemble des lignes retournées est récupéré
- exploiter une à une les lignes renvoyées par le SGBD

mysql_query(req, lien)	Soumet le texte d'une requête (variable req) au SGBD en précisant le lien (variable lien) qu'il avait ouvert précédemment: Renvoie un jeu de résultats sous forme d'une table (lignes et colonnes ; on parle aussi de jeu d'enregistrements ou recordset)

```
< ?php
...
// Se connecter au SGBD
// Sélectionner la base de données
// exécuter une requête
$requete="SELECT nom, prenom FROM maTable;";
$resultat=@mysql_query($requete ,$lien);
if (!$resultat) // si çà s'est mal passé, pas de résultat...
{  die('erreur requete : ' . mysql_error());
}
// suite du traitement en PHP
```

```
?>
```

Les SGBD à renvoyé une table résultat (lignes et colonnes).

Il va s'agir maintenant

- d'exploiter **une à une** les lignes renvoyées par le SGBD

mysql_fetch_row(rs)	Renvoie un tableau pour la ligne lue
mysql_fetch_assoc(rs)	Renvoie un tableau associatif pour la ligne lue où les clefs correspondent aux noms des colonnes
mysql-num-rows(rs**)**	renvoie le nombre de lignes du résultat(rs)

Exemple avec mysql_fetch_row

```
< ?php
// Se connecter au SGBD
// Sélectionner la base de données
// exécuter une requête

// parcourir les lignes du résultat renvoyé
while ($ligne = mysql_fetch_row($resultat))
{  $nom = $ligne[0]; // première colonne
   $prenom = $ligne[1]; // deuxieme colonne
   echo "<br />nom = $nom et prenom = $prenom";
};
// suite du traitement en PHP
?>
```

Exemple avec mysql_fetch_assoc

```
< ?php
// Se connecter au SGBD
// Sélectionner la base de données
// exécuter une requête

// parcourir les lignes du résultat renvoyé
while ($ligne = mysql_fetch_assoc($resultat))
{  $nom = $ligne["nom"];
   $prenom = $ligne["prenom"];
   echo "<br />nom = $nom et prenom = $prenom";
};
// suite du traitement en PHP
?>
```

4. Récupération d'une valeur de clef ajoutée – mysql_insert_id

Lors de l'insertion d'une ligne dans une table, possédant un identifiant auto incrémenté, il est souvent intéressant de récupérer cette valeur pour l'afficher à l'utilisateur ou bien effectuer un autre traitement.

La fonction **mysql_insert_id**(lien) (ou bien **last_insert_id()**), appelée juste après l'insertion d'ajout, renvoie cette valeur (attention : seulement pour MySQL : pour les autres SGBD, rechercher la valeur la plus grande – SELECT MAX(colonne_clef) FROM table ;)

- Le

5. Libération de la connexion

Une fois les traitements terminés sur la base de données, il est vivement conseillé de libérer les ressources (liens, lignes lues) :

```
< ?php
// Se connecter au SGBD
// Sélectionner la base de données
// Exécuter une requête
// Nettoyer le jeu de résultat
@mysql_free_result($resultat);
// liberer la connexion
@mysql_close($lien);
// suite du traitement en PHP
?>
```

B. Autres SGBD et ODBC

Les instructions précédentes concernent uniquement l'accès à une base de données MySQL.
Si vous prévoyez d'exploiter un site web connecté à d'autres SGBD, il est indispensable de mettre en œuvre les fonctions PHP adaptées.

1. Connexion natives aux grands SGBD du marché

PHP propose un ensemble de fonctions relatives aux grands SGBD du marché.
On trouvera donc pour le SGBD Oracle, par exemple :

- oci_connect : établit une connexion au SGBD
- oci_execute — exécute une requête SQL

cf. doc PHP : http://fr.php.net/manual/en/ref.oci8.php

2. Connexion via ODBC

Si le SGBD auquel sera connecté le site peut changer de manière fréquente, il est possible d'accéder à un jeu de fonctions connectées à une source de données qu'il sera possible de modifier à chaque fois que cela est nécessaire. (on peut imaginé un site web, sur lequel des utilisateurs se connectent et en fonction de l'utilisateur, un SGBD différent devra être sélectionné).

Il existe un système qu'on nomme « médiateur » qui va être l'intermédiaire entre nos commandes SQL et les bases de données ciblées.
Il s'agit d'Open DataBase Conenctivity (ODBC).

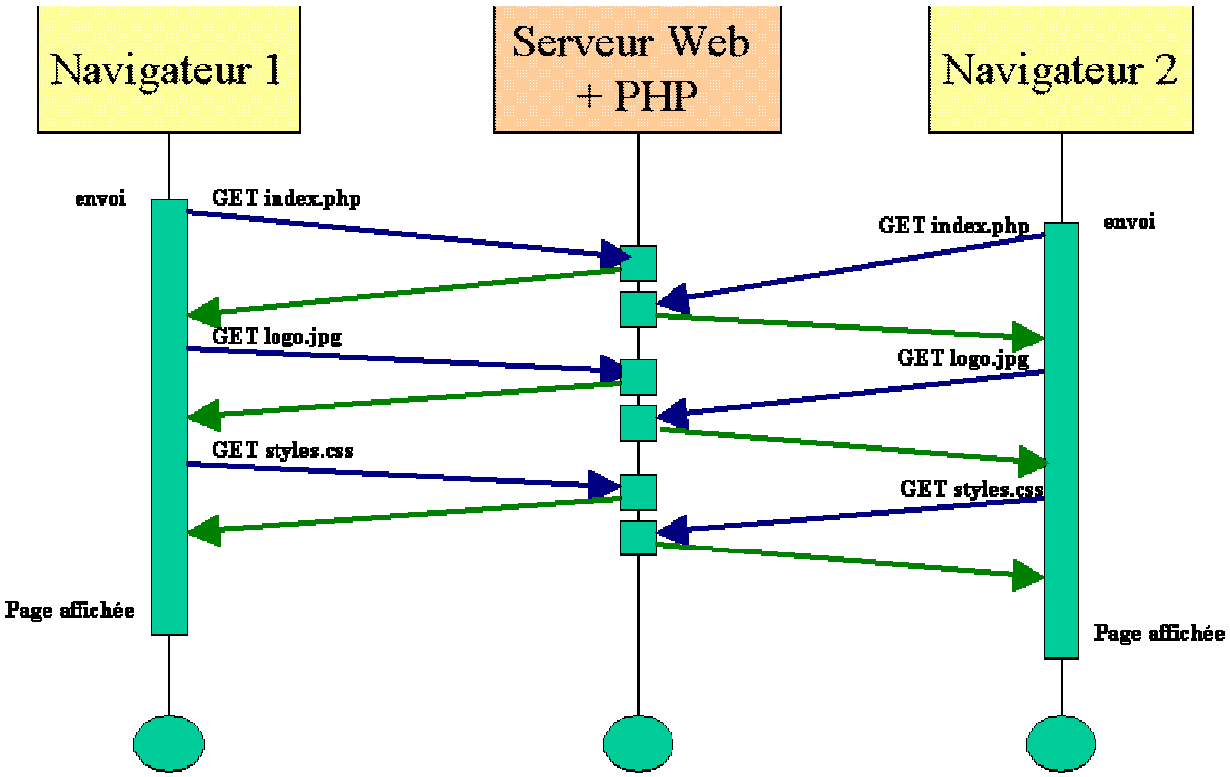
PHP offre un jeu de fonctions permettant de travailler avec des bases de données ODBC.

- odbc_connect — se connecte à une source de données ODBC
- odbc_exec — prépare et exécute une requête SQL

cf. doc PHP : http://fr3.php.net/odbc

VI. Sauvegarder des informations le temps d'un dialogue- Sessions et cookies

Les données traitées dans un script PHP ne sont plus disponibles une fois son exécution terminée. En effet, le protocole HTTP est un protocole dit « sans état » : chaque requête effectuée est totalement indépendante des autres (pour le serveur Web)



Si le site exige que des informations persistent tout au long du parcours des pages (cas fréquent du commerce électronique avec login, panier, passage de commandes et paiement en ligne), il est indispensable de mettre en œuvre un mécanisme de sauvegarde d'informations et de récupération tout au long de la navigation.
Plusieurs techniques permettent de mettre en œuvre ce mécanisme.

A. Passage par champs de formulaire ou arguments de lien hypertexte

Il est possible de « mémoriser » les données à conserver dans des champs de formulaires cachées (cf. champ de formulaire 'input type=hidden') ou bien des valeurs de paramètres des liens hypertextes qu'on construit sur chacune des pages utilisées.
C'est une technique assez fastidieuse à mettre en place et peu élégante.

De plus les informations sont accessibles simplement : soit dans le source de la page HTML soit dans le lien hypertexte.

B. Sauvegarde fichier ou base de données

Une autre technique un peu plus sûre, mais moins performante, est une extension à la précédente. On va conserver dans un champ de formulaire ou un lien hypertexte, seulement un identifiant (un numéro, un code attribué à la connexion, par exemple) qui nous permettra de stocker dans une base de données des informations et de les récupérer ultérieurement.

L'inconvénient est la capture possible de ce numéro et son utilisation par d'autres utilisateurs.

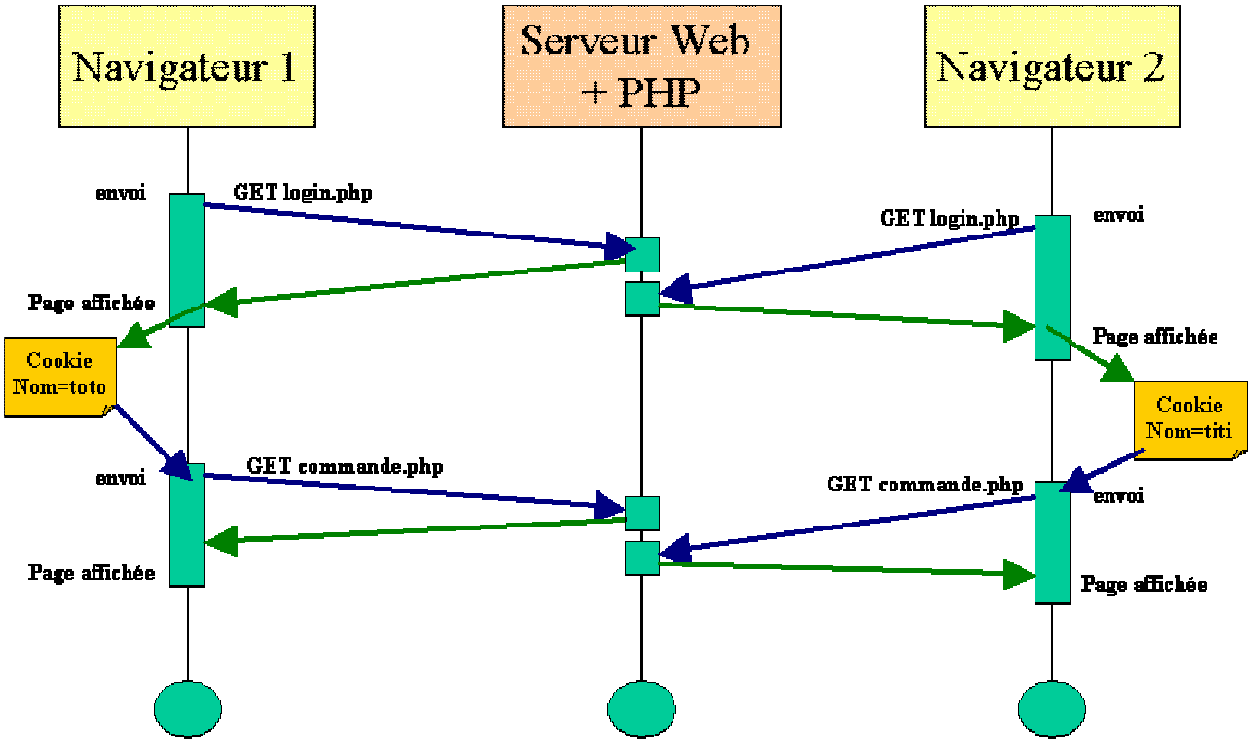
C. Cookies – fichiers du côté poste client

Les cookies sont un mécanisme de sauvegarde d'informations sur le poste de l'utilisateur (un simple fichier texte). PHP offre des instructions permettant d'envoyer ces informations sur l'ordinateur client, et de les récupérer ultérieurement.
Le **nombre et la taille** des cookies pouvant être stockés sur un poste client sont **limités** :

- Maximum de 4ko par cookie,
- Maximum de 300 cookies (20 cookies par nom de domaine).

Un cookie possède une date de limite de validité.

Par exemple, le cookie est utilisé souvent pour conserver sur le poste client une trace du passage sur un site web ; à la prochaine connexion ou bien dans le cours de la navigation sur un site, la lecture du cookie va permettre de personnaliser le message d'accueil et la navigation.



1. Mémorisation d'un cookie : setcookie()

```
setcookie ('nom_du_cookie', valeur, date_expiration)
```

Exemple de variable simple :

```
< ?php
// envoyer un cookie :
// on le nomme 'prenom' et il va contenir la valeur 'virginie'
```

```
// il sera valable pendant 1 heure
setcookie('prenom', 'virginie',(time() + 3600)) ;
// suite du traitement en PHP
?>
```

Attention : les cookies doivent être envoyés avant tout affichage (c'est-à-dire envoi de données avec **echo** ou **print**).

Exemple de tableau :

```
< ?php
// envoyer un cookie :
// on le nomme 'prenom' et il va contenir la valeur 'virginie'
// il sera valable pendant 1 heure (heure actuelle + 3600sec)
setcookie('produit[A320]', 'Airbus A320',(time() + 3600)) ;
setcookie('produit[A330]', 'Airbus A330',(time() + 3600)) ;
setcookie('produit[B747]', 'Boeing 747',(time() + 3600)) ;
// suite du traitement en PHP
?>
```

2. Récupérer une valeur de cookie : $_COOKIE

Exemple variable simple :

```
< ?php
// dans une page suivante
if ( !empty($_COOKIE['prenom']))
     $Vprenom = $_COOKIE['prenom'] ;
// suite du traitement en PHP
?>
```

Exemple variable tableau :

```
< ?php
// dans une page suivante
echo $_COOKIE['produit']['A320'] ;
echo $_COOKIE['produit']['A330'] ;
echo $_COOKIE['produit']['B747'] ;
// suite du traitement en PHP
?>
```

3. Effacer un cookie : setcookie() sans valeur

Exemple:

```
< ?php
setcookie('prenom' ) ;
setcookie('produit']['A320'] ;
// suite du traitement en PHP
?>
```

4. Localiser les fichiers cookies sur votre ordinateur

On peut trouver les fichiers cookies sous Windows (en fonction du navigateur) dans l'un des répertoires :

- C:\Documents and Settings\votre_compte_utilisateur\Cookies.
- ou dans C:\windows\Cookies"
- ou encore "C:\windows\temp\Cookies".

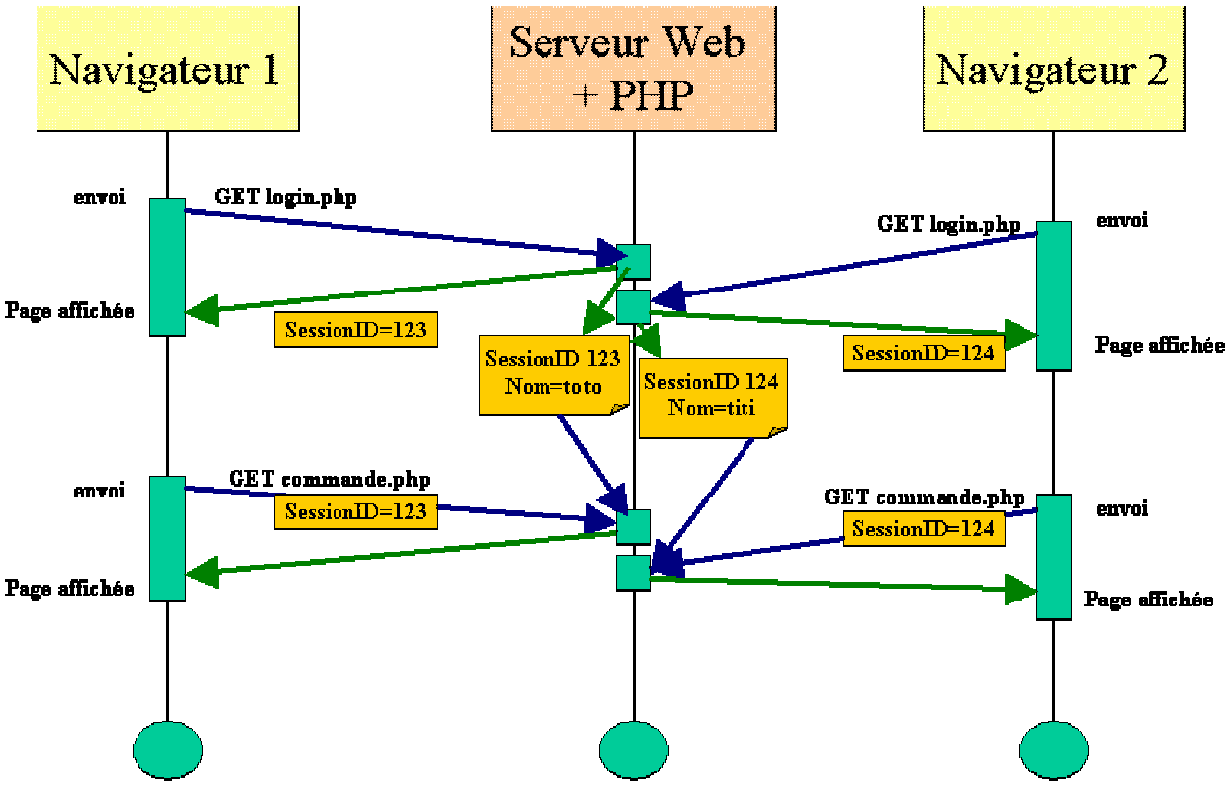
Le navigateur Mozilla Firefox V.3 permet d'accéder aux valeurs des cookies enregistrés (menu Outils > Options > Confidentialité > afficher les cookies)
Les noms des cookies sont sous la forme : votre_compte_utilisateur@nom_de_domaine_du_site_visité.txt.

On trouve à l'intérieur du fichier cookie les données suivantes (la structure est spécifique à chaque navigateur et n'est pas forcement bien documentée...) :

- Nom et valeur du cookie
- Chemin de la page
- Délai d'expiration (nombre de millisecondes à partir du 1/7/1970)

D. Sessions – fichiers du côté serveur

Les sessions permettent de préserver des informations tout au long d'une navigation Web. A une session peuvent être associées des données qu'on souhaite conserver le temps de la session.
A la différence des cookies qui sont stockés sur le poste « client », les données associées à une session sont stockées sur le poste « Serveur », dans un fichier unique, sous un identifiant de session. Cet identifiant est passé grâce au mécanisme des cookies ou bien via le passage par argument dans les liens hypertextes.



1. Démarrer une session : session_start()

Au démarrage d'une session, un SID (Session ID) est attribué et c'est lui qui va permettre de conserver un lien entre toutes les pages appelées.

A l'exécution de l'instruction **session_start()**, :

- Si une session avait déjà été démarrée dans une page précédente, PHP reprend le SID attribué

- Sinon un SID (SessionID, identificateur de session) est attribué

Exemple de variable simple :

```
< ?php
session_start() ;
// suite du traitement en PHP
?>
```

Attention : les sessions doivent être démarrées avant tout affichage (c'est-à-dire envoi de données avec **echo** ou **print**).

2. Enregistrer une donnée : $_SESSION[ ]

Exemple de variable simple :

```
< ?php
session_start() ;
// ...
$_SESSION['prenom']= 'virginie' ;
?>
```

3. Récupérer une donnée : $_SESSION[ ]

Exemple de variable simple :

```
< ?php
session_start() ;
// ...
if (isset($_SESSION['prenom']))
     echo $_SESSION['prenom'] ;
     $Vprenom = $_SESSION['prenom'] ;
// suite du traitement en PHP
?>
```

Exemple de variable simple :

```
< ?php
session_start() ;
// ...
if (!empty($_SESSION['prenom']))
     $Vprenom = $_SESSION['prenom'] ;
// suite du traitement en PHP
?>
```

4. Supprimer une variable de session : unset()

Exemple:

```
<?php
session_start() ;
unset($_SESSION['prenom']) ;
//...
?>
```

La fonction **unset** peut être utilisée pour supprimer toute variable.

5. Supprimer une session : session_destroy()

Exemple:

```
<?php
```

```
session_start() ;
session_destroy() ;
//...
?>
```

La session est un mécanisme qui apporte un premier niveau de protection ; cependant le vol d'un numéro de session est possible (tout transite en clair sur le réseau). Le chiffrage SSL permet d'apporter une protection beaucoup plus efficace.

VII. PHP – code modulaire, protection des dossiers

La modularité du code va permettre de réutiliser des ensembles d'instructions PHP utilisées fréquemment.
PHP met à disposition plusieurs techniques :

- Inclure le contenu de fichiers PHP à certains endroits d'un script
- Utiliser des fonctions
- Utiliser la Programmation Orientée Objet (classes dans les dernières versions de PHP).

A. Inclusion de code

L'inclusion de code permet l'insertion du contenu d'un fichier à l'emplacement d'une des instructions d'inclusion. La portion de code est ensuite exécutée.

1. include

L'instruction **include(**nom du fichier à inclure**)** incorpore le fichier sans contrôler qu'il a déjà été inclus. En cas d'erreur dans le fichier inclus, il y a affichage d'un avertissement, mais le script continue à s'exécuter.

2. require_once

L'instruction **require_once (**nom du fichier à inclure**)** incorpore le fichier en contrôlant que ce code ne sera ajouté qu'une seule fois.

3. require

L'instruction **require (**nom du fichier à inclure**)** incorpore le fichier en contrôlant qu'il n'a pas d'erreur dans ce fichier ; si une erreur se produit, l'exécution du script est arrêtée.

B. Fonctions

Il est possible de définir des fonctions en PHP. Une fonction est introduite par le mot-clef **function**. Au sein du bloc de code de la fonction, l'instruction **return** peut retourner une valeur.

Exemple de fonction qui ne retourne par de valeur (procedure) :

```
function msg_erreur($titre_msg,$texte_msg,$adr_retour)
{
echo <<< EOS ;
<html>
<head><title>Qcm:ERREUR</title>
```

```
<link rel='stylesheet' href='style_err.css' type='Text/css'>
</head>
<body><div class='erreur'>
<h1>$titre_msg</h1><h3>$texte_msg</h3>
<a href='$adr_retour'>Retour</a></div>";
</body></html>
EOS ;
};
```

Exemple de fonction qui retourne une valeur (fonction) :

```
function plusGrand($nb1,$nb2)
{
   $max=$nb2 ;
   if ($nb1>$nb2)
       $max = $nb1;
   return $max ;
};
```

C. Classes

Les dernières versions de PHP sont « orientées objets » : elle incluent donc la possibilité de définir des classes et de les instancier comme tout autre langage à objets.

VIII. PHP – upload de fichier, parcours de répertoires côté serveur

A. Upload de fichiers

L'upload de fichier consiste à envoyer un fichier du poste client vers le serveur. L'upload utilise le protocole http pour cette opération (et non pas le protocole FTP).

Exemple de **formulaire** incluant la possibilité de sélectionner un fichier :

```
<form method='post' enctype='multipart/form-data'
action='upload.php'>
  <input type='file' name='fichier' size='40'>
  <input type='submit' name='upload' value='Uploader'>
</form>
```

- enctype : les données envoyées par le formulaire seront composées de plusieurs parties
- type='file' : champ de saisie texte et bouton de parcours de répertoire

Exemple **script de traitement** permettant l'upload du fichier :

```
<?php
    // dossier où sera copié le fichier
    $content_dir = 'upload/';
    // test d'existence du nom envoyé par le formulaire
    if( !isset($_FILES['fichier']))
    {
        echo("pas d'envoi de fichier portant ce nom");
```

```
        exit() ;
    }
    // récupération du nom temporaire du fichier uploade
    $tmp_file = $_FILES['fichier']['tmp_name'];
    // test de l'existence du fichier
    if( !is_uploaded_file($tmp_file) )
    {
        echo("Le fichier est introuvable");
        exit() ;
    }
    $taille = filesize($tmp_file);
    if($taille>100)
    {
        echo 'Le fichier est trop gros...';
        exit() ;
    }
    // on copie le fichier dans le dossier de destination
    $name_file = $content_dir . 'monFichier.txt' ;
    // copie et test de la copie
    if (move_uploaded_file($tmp_file, $name_file) )
     {
         echo("<b>fichier recupere</b>");
      }
?>
```

- Le tableau associatif **$_FILES** contient les clefs suivantes ('fichier' correspond au nom du champs de formulaire) :
  - **$_FILES**['fichier']['**name**'] : nom d'origine du fichier
  - $_FILES['fichier']['**tmp_name**'] : nom temporaire attribué au téléchargement
  - $_FILES['fichier']['**type**'] : type MIME du fichier (Multipurpose Internet Mail Extensions : standard définissant un code par type de fichier ; exe : text/html, application/pdf, image/jpeg, etc.)
  - $_FILES['fichier']['**size**'] : taille du fichier en octets
  - $_FILES['fichier']['**error**'] : code erreur
- la fonction **is_uploaded_file** : test la bonne récupération du fichier
- la fonction **move_uploaded_file** : copie le fichier du répertoire temporaire vers un répertoire choisi

Pour obtenir un nom unique (éviter ainsi l'écrasement d'un fichier déjà existant), un exemple de script : (auquel on peut ajouter le tirage d'un nombre aléatoire, etc.)

```
    // on crée un nom unique
    $heure = time();
    $nomf = $ heure. "-" . $_FILES['fichier']['name'];
// etc.
?>
```

B. Parcours de répertoires sur le serveur

Exemple de parcours de répertoire et de liste des fichiers :

```
<?php
$rep=opendir('.');
echo '<ul>' ;
```

```php
while ($file = readdir($rep))
{
  if ($file !='..' && $file !='.')
  {
    if (is_file($file))
    {
      echo '<li>' . $file . '</li>' ;
    }
  }
} // fin while
echo '<ul>' ;
closedir($rep);
clearstatcache();
?>
```

IX. Gestion des fichiers 'texte'

Comme dans tous les langages, la gestion de fichiers en PHP passe par

- Une ouverture du fichier (dans un certain mode d'accès : lecture, écriture, ajout)
- Le traitement : en fonction du mode d'accès, lecture ou écriture dans le fichier
- La fermeture du fichier.

A. Modes d'ouverture du fichier

Le paramètre *mode* spécifie le type d'accès désiré au flux. Il peut prendre les valeurs suivantes :

Liste des principaux modes pour la fonction d'ouverture d'un fichier
Cf http://www.php.net/manual/fr/function.fopen.php

mode	Description
'r'	Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.
'r+'	Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
'w'	Ouvre en écriture seule ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'w+'	Ouvre en lecture et écriture ; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
'a'	Ouvre en écriture seule ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.
'a+'	Ouvre en lecture et écriture ; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

B. Ouvrir, lire ligne par ligne, écrire, fermer

Exemple d'écriture dans un fichier 'fichier2.txt' à partir de la lecture du contenu du fichier 'fichier1.txt' , et transformation de la ligne avant réécriture

```php
<?php
$entree = @fopen("fichier1.txt", "r");
$sortie = @fopen("fichier2.txt", "w");
if ($entree && $sortie) {
    while (!feof($entree)) {
        // recupération de la ligne
        $ligne = fgets($entree);
        // traitement éventuel :
        // Exemple : remplace tous les '...' par ' !!!'
        $ligneModifie=str_replace('...',' !!!',$ligne) ;
        $ligne = $ligneModifie;
        // écriture de la ligne dans le fichier de sortie
        fwrite($sortie, $ligne);
    }
    fclose($sortie);
    fclose($entree);
}
?>
```

C. Lecture et écriture globale du contenu

Exemple d'écriture du contenu d'un fichier 'fichier2.txt' à partir du contenu du fichier 'fichier1.txt' :

```php
<?php
$contenu = file_get_contents("fichier1.txt");
file_put_contents ("fichier2.txt", $contenu);
?>
```

X. PHP – extensions

Le langage PHP s'est vu enrichi de nombreuses bibliothèques de fonction offrant des fonctionnalités avancées.
Il est parfois nécessaire l'ajouter des modules PHP au serveur Web pour bénéficier de ces fonctionnalités.

A. Produire un document PDF

Exemple de creation d'un fichier PDF et retour HTML d'un lien pour l'ouvrir :

```php
<?php
// créer un objet PDF
$ficPDF = pdf_new();
// associer un fichier à cet objet
pdf_open_file($ficPDF,realpath('pdf/test.pdf'));
// définition des informations relatives au document
pdf_set_info($ficPDF, 'Author','votre prenom et votre nom');
pdf_set_info($ficPDF, 'Title','Test fichier PDF en PHP');
```

```
pdf_set_info($ficPDF, 'Creator','(cf auteur)');
pdf_set_info($ficPDF, 'Subject','PHP et PDF');

// DEBUT DE PAGE 1
pdf_begin_page($ficPDF,595,842);
// bas de page
pdf_add_outline($ficPDF,'Page 1');
// récupération d'un objet police de caractères
$police = pdf_findfont($ficPDF, 'Times New Roman','winansi',1);
if ($police) { // si police ok
  // utilisation dans le document PDF
  pdf_setfont($ficPDF,$police,12);
}
pdf_show_xy($ficPDF,'Client : xxxxxxxx',400,700);
pdf_show_xy($ficPDF,'Adresse : xxxxxxxx',400,680);
pdf_show_xy($ficPDF,'        : xxxxxxxx',400,660);
pdf_show_xy($ficPDF,'CP - VILLE : 62000 - ARRAS',400,640);
// se positionner en 50,600 et tracer une ligne
pdf_moveto($ficPDF,50,600);
pdf_lineto($ficPDF,545,600);
// idem.
pdf_moveto($ficPDF,10,200);
pdf_lineto($ficPDF,500,200);

pdf_stroke($ficPDF);
// fin de page
pdf_end_page($ficPDF);
// fermeture du document
pdf_close($ficPDF);
?>
<!-- HTML permettant la création d'un lien vers le document PDF
-->
<html>
<head><title>essai pdf</title></head>
<body><a href="pdf/test.pdf">fichier pdf</a></body>
</html>
```

Extrait du fichier PDF créé :

Client : xxxxxxxx

Adresse : xxxxxxxx

: xxxxxxxx

CP - VILLE : 62000 - ARRAS

XI. PHP – traitement de formulaires avec champs multiples

A. Champs INPUT multiples : tableaux de valeurs

Il arrive qu'on soit amené à traiter plusieurs valeurs du même type, sous forme d'un tableau.

Le fait de nommer les champs de formulaire comme des variables tableaux, va permettre la récupération d'un tableau de valeurs au lieu de champs séparés, facilitant ainsi le traitement.

Exemple de formulaire (HTML), le click sur le bouton envoie le formulaire à la page « additionner.php » :

```
... HTML ...
<form name="frmSaisie" action="additionner.php" method="POST">
  <input type="text" name="frmVal[]" />
  <input type="text" name="frmVal[]" />
  <input type="text" name="frmVal[]" />
  <input type="text" name="frmVal[]" />
  <input type="submit" name="btnValider"  value="Valider" />
</form>
... HTML ...
```

La variable postée frmVal va être reçue comme un tableau de valeurs (le même nom pour plusieurs éléments) : la variable PHP $Vvaleur récupérera les valeurs comme un tableau.

```
< ?php
// récupération des variables « postées »
$Vvaleur=$_POST ["frmVal"];
$Vtotal=0 ;
for ($i=0 ;$i<count($Vvaleur) ;$i++)
    $Vtotal+=$Vvaleur[$i] ;
echo 'le total est égal à ' . $Vtotal;
// suite du traitement en PHP
?>
```

B. Sélection multiples dans SELECT/OPTION

Il arrive de même qu'on ait à récupérer plusieurs valeurs sélectionnées dans une liste déroulante.

Exemple de formulaire (HTML), le click sur le bouton envoie le formulaire à la page « traitement.php » :

```
... HTML ...
<form name="frmSaisie" action="traitement.php" method="POST">
  <select name="frmPays[]" multiple size="3" />
  <option value="FR" selected>France</option>
  <option value="GB">Grande-Bretagne</option>
  <option value="DK">Danemark</option>
  <option value="ES">Espagne</option>
  <option value="SW">Suede</option>
```

```
  </select>
  <input type="submit" name="btnValider"  value="Valider" />
</form>
... HTML ...
```

A la réception du champ de formulaire, celui-ci sera considéré comme un tableau et contiendra les options sélectionnées de la liste à choix multiple :

```
< ?php
// récupération des variables « postées »
$Vchoix=$_POST ["frmPays"];
$Vnb=0 ;
for ($i=0 ;$i<count($Vchoix) ;$i++)
   echo '<br />Vous avez choisi : ' . $Vchoix[$i];
// suite du traitement en PHP
?>
```

C. Sélection Multiples dans les CHECKBOX

Il arrive de même qu'on ait à récupérer plusieurs valeurs sélectionnées dans des cases à cocher.

Exemple de formulaire (HTML), le click sur le bouton envoie le formulaire à la page « traitement.php » :

```
... HTML ...
<form name="frmSaisie" action="traitement.php" method="POST">
  <input type="checkbox" name="frmChk[]" value="FR" />
  <input type="checkbox" name="frmChk[]" value="GB" />
  <input type="checkbox" name="frmChk[]" value="DK" />
  <input type="checkbox" name="frmChk[]" value="ES" />
  <input type="checkbox" name="frmChk[]" value="SW" />    <input
type="submit" name="btnValider"  value="Valider" />
</form>
... HTML ...
```

A la réception du champ de formulaire, celui-ci sera considéré comme un tableau et contiendra les options sélectionnées de la liste à choix multiple :

```
< ?php
// récupération des variables « postées »
$Vchoix=$_POST ["frmChk"];
$Vnb=0 ;
for ($i=0 ;$i<count($Vchoix) ;$i++)
   echo '<br />Vous avez choisi : ' . $Vchoix[$i];
// suite du traitement en PHP
?>
```

XII. PHP – compléments

http://fr.wikipedia.org/wiki/Liste_de_frameworks_PHP
http://pear.php.net/