1ère partie

Apprendre Php

Tour d'horizon…

Qu'est-ce que c'est ?

PHP (abréviation de Personal Home Page Hypertext Preprocessor) version 4 beta à ce jour, est un langage de scripts qui s'intègre aux pages Html et qui permet de réaliser des **pages dynamiques**.

- Il s'exécute sur le **serveur** et permet d'accéder facilement aux bases de données.
- C'est un produit "**Open Source**" c'est-à-dire que le code est accessible à tout développeur.
- Il est **gratuit**. Combiné au système d'exploitation Linux, au serveur Apache et à la base de données MySQL (eux-mêmes gratuits), il permet de créer des sites Web à des coûts très réduits.

• Historique

La première version date de 1994 et s'appelait PHP/FI. Elle n'avait pour ambition que de pouvoir insérer quelques traitements simples dans une page HTML, comme le comptage des visites.
Ce langage est moins puissant que le Perl ou le C mais il est beaucoup plus simple à programmer et surtout il permet de gérer les bases de données de manière très simple.

• Pages statiques et pages dynamiques

La plus grande qualité et le plus important avantage du langage PHP est le support d'un grand nombre de bases de données. Réaliser une page Web dynamique interfacée avec une base de donnés est extrêmement simple. Les bases de données suivantes sont supportées par le langage PHP :

Adabas D	dBase	Empress	FilePro	Informix
InterBase	mSQL	MySQL	Oracle	PostgreSQL
Solid	Sybase	Velocis	Unix dbm	

• Intérêt du traitement côté serveur

- réduction du temps de téléchargement puisque le client ne reçoit qu'une simple page HTML (=>diminution du trafic réseau)
- absence de problème de compatibilité des navigateurs
- offrir au client des données qui sont dans une base
- le code ne peut être vu par le client.

• Syntaxe et Conseils

- PHP ressemble aux langages C, C++ et Javascript : à savoir
  - **;** à la fin de chaque ligne d'instructions. Contrairement à Javascript, il est **obligatoire**.
  - **{…}** pour encadrer un bloc d'instructions
  - les opérateurs de comparaison et d'affectation sont les mêmes (**&&**, **||**, **==**, …)
  - les symboles des commentaires **//** et **/* … */**
  - toute une série de mots réservés qui correspondent à des mots-clés du langage.
  - …

PHP est sensible à la casse (caractères minuscules/majuscules) c'est-à-dire que la variable ***$NBRE*** n'est pas la même que la variable ***$nbre*** ou ***$Nbre***
Toutefois, cette règle ne s'applique pas aux fonctions, les spécifications du langage PHP précisent que la fonction *print* peut être appelée *print()*, *Print()* ou *PRINT()*

Que faut-il ?

- Matériels et logiciels nécessaires

- PHP travaille sur de nombreuses **plates-formes** telles que Unix, Linux et Windows (Win32) sur lesquelles il faut installer un serveur web.
- Un **serveur web** qui peut être Apache (Unix, Linux, Win NT, Win2K), mais aussi IIS 3/4/5, et même PWS (Personal Web Server) sur Windows 95/98 et NT.
- Pour écrire des scripts PHP, un simple **éditeur de texte** tel le Bloc-notes de Windows est suffisant. Toutefois, de nombreux logiciels gratuits (freeware) ou non, en rendent l'écriture plus agréable (aide intégrée, colorisation des mots-clés, ...)
- Pour exécuter les scripts, il faut un **navigateur** tel que Internet Explorer (IE), Netscape Navigator (NN), Opéra, ...
- Eventuellement, une **base de données** telle que MySQL.

PHP étant intégré à des pages HTML, la connaissance du **HTML** est indispensable avant d'aborder ce manuel. Malgré tout, un très léger rappel y sera fait.

EasyPHP est un utilitaire qui installe et configure automatiquement un environnement de travail complet sur les plates-formes win9x/NT/2000/Me : serveur "Apache, interpréteur PHP, base de données MySQL, ensemble de scripts PHP3 permettant de gérer des bases par le Web. C'est dans cette configuration que seront donnés tous les exemples de ce manuel.

- Installation de EasyPHP

✓ Téléchargement

EasyPhp 1.4 est disponible sur le site http://easyphp.manucorp.com/telechargements.php3
A ce jour, il installe le serveur Web **Apache** 1.3.20, l'interpréteur **PHP** 4.0.5, la base de données **MySql** 3.23.38 et l'ensemble de scripts PHP3 permettant de gérer des bases **MySQL** via le web **PHPmyAdmin** 2.2.0pre5 sur les plates-formes win9x/NT/2000/Me.

✓ Installation et démarrage

Double-cliquer sur le fichier d'installation **easyphp1-4_setup.exe**.
Une fois **EasyPHP** démarré, une icône se place dans la barre des tâches à côté de l'horloge. Un clic droit permet d'accéder à différents menus :

- Fichier Log : renvoie aux erreurs générées par Apache et MySQL
- Configuration : donne accès aux différentes configurations d'EasyPHP
- Web local : ouvre la page http://localhost/
- Démarrer/Arrêter : démarre/arrête Apache et MySQL
- Quitter : ferme EasyPHP

✓ Répertoire par défaut des pages Web

Le répertoire mis par défaut à l'installation de EasyPhp est :

C:\EasyPHP\www

Pour changer ce répertoire, il faut éditer le fichier "**C:\EasyPHP\apache\conf\httpd.conf"** et y remplacer DocumentRoot "C:\EasyPHP\www" par :

DocumentRoot "D:\MonSite" (par exemple).

Une première page en PHP

- Quelques rappels de HTML

Une page HTML est constitué de **balises** (=tags) qui correspondent à des instructions de mise en forme de la page HTML. Chaque balise est encadrée par les symboles **<** et **>**. Chaque mise en forme est présentée par une balise de début **<balise>** et une balise de fin **</balise>**.

Exemple : <B> ce texte sera en caractères gras </B> <I> ce texte sera en italiques </I>

Certaine balises sont obligatoires dans toute page Web. La plus petite page possible est la suivante : celle-ci affiche « **Coucou**».


```
<HTML>
<HEAD>  </HEAD>
<BODY> Coucou </BODY>
</HTML>
```

- Ecriture du script PHP dans la page

L'exemple qui suit permet d'afficher du texte et la date courante (du serveur).

Le script s'intègre directement au code HTML et commence par **<?** (ou **<?php**) et se termine par **?>**.

```
<html>
<head>
<title>Ma première page en PHP</title>
</head>
<body>
   Aujourd'hui :
   <? print (Date("l F d, Y"));
      print ("<HR><B>Bienvenue à l'ENITA Bordeaux</B>")
   ?>
</body>
</html>
```

- ## Explication du script précédent

Le script se trouve entre les balises **<?** et **?>**.

- **<? print (Date("l F d, Y")); ?>** - **<script language="php">** **print (Date("l F d, Y"));** **</script>** - **<? print (Date("l F d, Y"));?>**	ces 3 écritures sont équivalentes à **<?php print (Date("l F d, Y")); ?>** (la dernière n'étant valable que depuis la version PHP 3.0.4)
print	fonction d'affichage à l'écran. On trouve aussi la fonction **echo**
Date (paramètres)	fonction qui affiche la date courante
;	toute instruction PHP termine obligatoirement par ce signe
print ("<HR><B>Bienvenue à l'ENITA Bordeaux</B>")	affiche une ligne de code HTML

- ## Enregistrement de la page

Avec **EasyPHP**, si l'installation de base n'a pas été modifiée, pour que les pages PHP soient interprétées, il est impératif de placer les fichiers dans le répertoire **www**. Ensuite, il est conseillé de créer un répertoire par projet dans ce répertoire.

Une page en PHP sera généralement sauvegardée en lui donnant une des extensions suivantes **: php, php3, php4**. Ce n'est pas une règle absolue, mais ceci correspond à la configuration d'EasyPHP.
Pour notre exemple : date.php
Si la page n'a pas l'extension voulue, le serveur HTTP (Apache) ne va pas comprendre les balises **<?** et **?>** et rien ne s'affichera.

- ## Affichage du résultat

Avant d'afficher le résultat de la page, il faut s'assurer que Apache est démarré. Pour cela, quand **EasyPHP** est actif, l'icône est présent dans la barre de tâche à côté de l'horloge. Le serveur Apache est démarré si "clic droit" sur l'icône affiche "Arrêter" et le "point rouge" de l'icône clignote.

Il existe 2 façons d'accéder à la page PHP.
- Dans ce même menu, ouvrir le "**Web local**",
- ou ouvrir un navigateur et donner l'URL : http://localhost

Sélectionner ensuite le répertoire de travail puis cliquer sur "date.php". La page affiche la date courante.
On obtient à l'écran :

Aujourd'hui : Thursday June 14, 2001
Bienvenue à l'ENITA Bordeaux

A NE PAS FAIRE : aller dans le répertoire www puis dans le répertoire correspondant à votre projet et double cliquer sur votre page d'exemple. Vous obtiendrez à coup sûr une page d'erreur.

- ## Afficher la source

Les navigateurs permettent de voir le code HTML (la source) qui produit l'affichage (clic droit dans la page puis "Afficher la source"). Le code précédent vu par le navigateur est le suivant :

```
<html>
<head>
<title>Ma première page en PHP</title>
</head>
<body>
Date courante : Thursday June 14, 2001</body>
</html>
```

On constate qu'il n'y a plus aucune balise PHP.

- ## Ce qui se passe entre le client et le serveur

Lorsqu'un internaute demande page depuis son navigateur, voici ce qui s'y passe :
- l'internaute demande une page (exemple : http://www.monsite.fr/page.php)
- elle arrive au serveur Web
- elle est analysée et exécutée par l'interpréteur PHP
- elle est transformée en langage HTML
- elle est renvoyée sur le navigateur du poste client via Internet. L'internaute ne peut absolument pas voir le code source de la page PHP.

- ## Les commentaires

Nous avons vu comment se présente un script PHP. Notre exemple est très simple, mais il peut très vite être plus compliqué. Dans ce cas, il est important, voire indispensable, de commenter les instructions. Pourquoi ?
- lorsqu'une fonction a été écrite, il est difficile d'en comprendre certaines subtilités après quelques mois. Un commentaire permet de donner des pistes.
- pour permettre une prise en main et une maintenance rapides par quelqu'un qui n'a pas écrit le script.

Les commentaires devraient être présents dans les pages Html ET les scripts PHP. Leur syntaxe est différente.

✓ Les commentaires en Html

<!-- commentaire début commentaire suite commentaire fin **-->**	Sur plusieurs lignes, tout ce qui est compris entre <!-- et --> n'est pas interprété par le navigateur. En Html, les commentaires évitent que certains anciens navigateurs affichent les fonctions qui leur sont inconnues comme étant de l'Html. Dans ce cas, ils les ignorent.

✓ Les commentaires en PHP

Dans les scripts, il existe 3 types de symboles pour désigner du commentaire.

// affichage de la date	sur une même ligne, tout ce qui suit les signes // n'est pas étudié par l'analyseur PHP

# affichage de la date	sur une même ligne, tout ce qui suit les signes # n'est pas étudié par l'analyseur PHP

/* commentaire début commentaire suite commentaire fin ***/***	Sur plusieurs lignes, tout ce qui est compris entre /* et */ n'est pas étudié par l'analyseur. Ceci permet de déterminer des blocs de commentaires.

- Les entrées / sorties
- **Les saisies de données** (entrée) font être réalisées par l'intermédiaire des formulaires
- **Les édition des résultats** (sorties) vont s'inscrire dans la page par les fonctions
  - **echo ("message");**
  - **print ("message");**

Notions générales de programmation

Nous avons écrit notre premier script PHP. Afin d'aller plus loin dans la conception de pages dynamiques, il est nécessaire de connaître les base d'un langages de programmation. Pour cela, nous allons voir ce que sont les variables, les opérateurs, les instructions, les fonctions et procédures. Pour terminer nous verrons les structures algorithmiques nécessaires aux éléments précédents.
Pour commencer, nous allons voir comment nous

Les variables

- Déclarations et initialisations

Les variables correspondent à des zones de mémoire "étiquetées" par un nom, dans lesquelles seront enregistrées les valeurs à traiter. Chaque variable PHP est précédée du signe **$** et reçoit une valeur grâce au symbole d'affectation **=**.

Contrairement à certains langages (Java par exemple), Php n'impose pas de déclaration explicite des variables avant de les utiliser. Toutefois, pour des raisons de cohérence et de clarté, il est une **bonne habitude** de déclarer les variables et de les faire suivre d'un commentaire qui indiquera leur rôle dans le programme.

✓ Déclarer une variable

Cela signifie réserver et nommer une zone mémoire qui correspond au type de données voulu (entier, alphanumérique, …).
Exemples : $ecole $i $nbre
En php, il est conseillé de déclarer ses variables ainsi :

```
settype ($i, "integer");          // type entier - indice d'un tableau
settype ($ecole , "string");      // chaîne de caractères- nom école
settype ($nb, "double");          // nbre réel – résultat du calcul
settype ($tb, "array");           // type tableau –liste départements
settype ($obj, "object")          // type objet
settype ($cls, "class");          // type classe
settype ($x , "unknown type");// type inconnu
```

Php pouvant changer dynamiquement le type de variable selon son contenu, on peut également donner simplement un nom de variable mnémotechnique qui reprend le type de données, comme par exemple :

```
$i_i;
$s_ecole;
$d_nb;
$t_tb;
$o_obj;
$c_cls;
$x;
```

Rappel : La déclaration n'est pas obligatoire mais permet aux développeurs de l'application de détecter plus rapidement d'éventuelles anomalies du fonctionnement du programme.
C'est donc une bonne habitude que l'on retrouve dans la majorité des langages de programmation, certains étant plus exigeants que d'autres.

✓ Initialiser une variable

Cela consiste à lui affecter une valeur initiale qui pourra éventuellement être modifiée par le programme. Le symbole d'affectation est le signe **=**.

```
$i_i = 4 ;                  // initialisation par un entier
$s_ecole = "ENITA Bx";      // initialisation par une chaîne
$s_w = "1.2 euros";         // initialisation par une chaîne
$d_nb=12345678.99;          // initialisation par un réel
$t_tb[ ] = 100;             // initialisation du 1er élément du
                            // tableau
```

Important : Il est à noter que malgré ces déclarations, Php ne contrôle pas le contenu des variables c'est-à-dire qu'à posteriori dans le programme, si on place du texte dans une variable numérique, Php convertira tout seul le type de la variable en "string".

Ceci peut donner lieu à des subtilités importantes du type (les mêmes qu'en Basic) :

```
$i_x = 4;                   // entier
$d_y = 5.6;                 // double
$s_z = "1.2";               // type chaîne
$s_w = "1.2 euros";         // type chaîne
$a = $i_x + $d_y;           // $a vaut 9.6
$b = $i_x + $d_y + $s_z;    // $b vaut 10.8
$c = $i_x + $d_y + $s_w;    // $c vaut 10.6
```


• Variables et types de données : chaînes, nombres

En PHP, il existe 3 types de données déterminés par l'affectation qui en est faite :

Déclaration en PHP	Type
$s_chaine1 = "ENITA Bordeaux" ; $s_chaine2 = '1.2 euros'; $s_chaine3 = "Réunion Salle \"Pomerol\" " ; $s_chaine4= $s_chaine2 ; $s_chaine5= '<A href='www.enitab.fr'>Enitab'	**Chaîne de caractères** certains caractères spéciaux doivent être précédés de \ (voir Annexe page 32) Il est à noter que les chaînes de caractères doivent encadrées de guillemets simples OU doubles.
$d_prix = 10.95 ; $i_indice = -1 ;	**Double** : nombre à virgule **Entier** : numérique entier

Le type **booléen** (false ou true) n'existe pas en PHP, mais il peut être remplacé par une expression.

valeurs qui renvoient Faux	valeurs qui renvoient Vrai
$i_x=0	$x différent de 0
$d_x=0.0	$x différent de 0.0
$s_x = ""	$x différent de chaîne vide ""

Les valeurs peuvent être comparées aux constantes prédéfinies TRUE et FALSE comprises par PHP.
Exemple : if ($i_x == false) // x vaut 0 ou une chaîne vide

✓ Conseils et remarques

- Eviter tout signe diacritique dans les noms de variables (caractères accentués, cédille, …)
- Pas d'espace dans les noms de variable
- PHP est sensible à la casse de caractères utilisée (la variable "toto" est différente de la variable "Toto").
- Les variables sont toutes précédées du signe **$**.
- **=** est le symbole d'affectation ; **==** est le symbole de comparaison (voir "opérateurs" page 32).

• Les tableaux

Avec PHP, il existe plusieurs façons de déclarer des tableaux unidimensionnels ou multidimensionnels.

✓ Déclaration d'un tableau à une dimension

Les indices sont numériques

$dept_enita = array ("des", "dpa", "dbe") ; ou $dept_enita[] = "des"; $dept_enita[] = "dpa"; $dept_enita[] = "dbe"; ou $dept_enita[0] = "des"; $dept_enita[1] = "dpa"; $dept_enita[2] = "dbe";	**Tableau** de données. Dans tous les cas, on obtiendra : dept_enita [0] = "des" ; dept_enita [1] = "dpa" ; dept_enita [2] = "dbe" ;

Remarque : si on ajoute au code précédent la ligne **$dept_enita[10]="formco",** alors la valeur de l'indice de $dept_enita[] sera **11**.

Modification des valeurs implicites d'indices

Nous avons vu qu'implicitement, Php débute par l'indice 0. Dans certains cas, il peut être pratique de commencer à 1. Php permet de modifier les valeurs des indices. Par exemple :

$dept_enita = array (**1 =>** "des", "dpa", "dbe") ; // dept_enita[2]="dpa"	équivalent à dept_enita[1]="des"; dept_enita[2]="dpa"; dept_enita[3]="dbe";
$dept_enita = array ("des", "dpa", **5 =>** "dbe") ; // dept_enita[0]="des" dept_enita[5]="dbe"	équivalent à dept_enita[0]="des"; dept_enita[1]="dpa"; dept_enita[5]="dbe";

Les indices sont des chaînes de caractères

$dept_enita["des"] = "entreprise & syst."; $dept_enita["dpa"] = "prod. animale"; $dept_enita["dbe"] = "bio expériment."; ou $dept_enita = array ("des" => "entreprise & syst." "dpa" => " prod. animale ", "dbe" => " bio expériment. ") ;	**Tableau** de données. Il est obligatoire d'affecter une valeur à chaque indice "chaînes de caractères"

✓ Déclaration d'un tableau multidimensionnel

Les indices sont numériques

$dept_enita[0] [0]= "uf info"; $dept_enita[0] [1]= "uf eco"; $dept_enita[1] [0]= "uf pv"; $dept_enita[1] [1]= "uf pa"; ou $dept_enitab = array (array ("uf info", "uf eco"), array ("uf pv", "uf pa"));	**Tableau multidimensionnel :** Chaque élément est lui-même un tableau : array(array(…), array(…), array(…)) Il est ainsi possible de créer des tableaux à 2, 3, 4 … dimensions

Les indices sont des chaînes de caractères

$dept_enita["des"] [0]= "uf info"; $dept_enita["des"] [1]= "uf eco"; $dept_enita["dpa"] [0]= "uf pv"; $dept_enita["dpa"] [1]= "uf pa"; ou $dept_enitab = array ("des" => array ("uf info", "uf eco"), "dpa" => array ("uf pv", "uf pa"));	

✓ Utilisation des variables tableaux

Pour travailler sur des variables tableaux, il suffira de donner le nom de la variable suivi de l'indice.

Exemples : if dept_enitab[2] == "uf info" { …} si dept_enitab[2] = "uf info" alors …
if dept_enitab[1][0] == "uf pv" { … } si dept_enitab[1][0] = "uf pv" alors …

✓ Fonctions concernant les tableaux

Cette liste des fonctions concernant les tableaux n'est pas exhaustive.

Fonctions	Commentaires	Exemple
count($var)	compte le nombre d'éléments affectés	$dept = array ("des","dpa"); $nb = count($dept); // = 2
current($var)	détermine la valeur de l'élément en cours	$dept[5]="formco"; $val =current($dept); //="formco"
key($var)	détermine l'indice de l'élément en cours	$dept[5]="formco"; $val = key($dept) ; // = 5
reset($var)	déplace le pointeur sur le 1er élément	$reset ($dept); // = "des"

list ($indice, $val)
each ($var_tableau)
	permettent de parcourir les éléments même si les indices ne sont pas consécutifs.	while (list($indice, $val)
		=each($dept))
		{ echo ("$indice - $val");
		// "0 - des" puis "1 - dpa"
		}

next($var) **prev($var)**	permettent d'aller respectivement à l'élément suivant et précédent.	fonctions déconseillées

Fonctions	Commentaires	Exemple

sort($var)	trie les éléments selon un ordre numérique ou alphabétique et réaffecte les indices du tableau	$dept=array ("des","dpa","dbe");
		sort($dept);
		while (list($indice, $val) =each($dept))
		{ echo ("$indice - $val");
		// "0 - dbe" puis "1 - des"
		// puis "2 - dpa"
		}

asort($var)	trie les éléments mais ne réaffecte pas les indices	$dept=array ("des","dpa","dbe");
		asort($dept);
		// "2 - dbe" puis "0 - des"
		// puis "1 - dpa"

rsort($var)	même fonction que **sort()** mais trie les éléments en ordre inverse	
arsort($var)	même fonction que **asort()** mais trie les éléments en ordre inverse	
ksort($var)	trie par valeur d'indice	
krsort($sort)	même fonction que **ksort()** mais trie les indices en ordre inverse	

usort($var, $critere),
uasort($var, $critere),
uksort($var, $critere)
	trie selon un critère donné par l'utilisateur	// tri sur l'inverse de la chaîne
		$dept=array ("des","dpa","dbe");
		uasort($dept, strrev($dept));
		// "1 - dpa" puis "2 - dbe"
		// puis "0 - des"

✓ Un exemple d'utilisation des tableaux

Les tableaux sont très utiles pour travailler sur des formulaires à remplir.

Exemple : soit un formulaire dans lequel il faut remplir une succession d'identités de personnes (nom et prénom)

On pourra simplement écrire :

```
<form action="submit.php" method=post>
   <?
   $nb_nom=10;              // 10 noms à remplir
   for $i= 1 ; $i <= $nb_noms ; $i++) ?>
      Prénom : <input name="prenom[]" type=text>
      Nom : <input name="nom[]" type=text>
   <? endfor ?>
   <BR> <input type="submit">
</form>
```

Le fichier **submit.php** contiendra les commandes d'insertion des données dans une base de données :

```
<?
    for $i =0 ; $i < count($nom) ; $i++)
    {
       $sql = "insert into maTable ('firstname', 'name') " .
              " values ('$prenom[$i]', '$nom[$i]')" ;
       ....  // commande d'exécution de la requête
    }
?>
```

- Portée des variables : locale, globale, statique

✓ Variable locale

Une variable déclarée **dans une fonction** aura une portée limitée à cette seule fonction. On ne pourra donc pas utiliser sa valeur ailleurs dans le script. C'est une **variable locale** à la fonction.
Exemple : $ecole ="ENITA Bordeaux"; // n'est connue que la fonction dans laquelle elle se
// trouve

✓ Variable globale

Si la variable est déclarée tout au début du script, **en dehors et avant toute fonction**, elle est toujours globale. Afin de distinguer une variable locale qui porterait le même nom qu'une variable globale, il faut la déclarer à l'aide du mot-clé **global**.
Exemple :

```
<?
$ecole="ENITA Bordeaux" // est connue de toutes les fonctions
function signer()
{
    global $ecole;   // $ecole est la variable globale précédente
    $titre="Directeur ";
    $titre = $titre . $ecole;  // Directeur ENITA Bordeaux
}
?>
```

Conseil : Pour faciliter la gestion des variables, on ne peut que conseiller de les déclarer en début de script (comme dans la plupart des langages de programmation).

✓ Variable statique

Implicitement, les variables locales d'une fonction sont réinitialisées à chaque appel de fonction. Pour conserver leurs valeurs précédentes, il faut les déclarer par le mot-clé **static**.

Exemple : nombre de visites

```
function nb_visite ($name)
{
  static $liste_nom = "";
  static $compteur = 0;
  $liste_nom = $liste_nom . "$name - " ;
  $compteur++;
  echo ($liste_nom . $compteur . "<BR>");
}
```

Les appels à la fonction donneront :

```
nb_visite("Claude");                    // Claude – 1
nb_visite("Jean");            //Claude – Jean - 2
```

- Fonctions utiles pour les variables

Fonctions	Commentaires	Exemple
gettype ($nom_var)	détermine le type de données de la variable ("integer", "double", "string", "array", "object", "class" ou "unknown type")	$x = 2.3; echo (gettype($x)); // double
settype ($nom_var, "type")	définit explicitement le type de variable	$x = 2.3; settype ($x, "integer"); echo $x; // $x=2
isset ($nom_var)	sert à savoir si une variable possède une valeur (true ou false)	$x = 2.3; $val = isset ($x) ; // true
unset ($nom_var)	détruit une variable	unset ($x); // isset($x) donne false
empty ($nom_var)	renvoie true si la variable est vide ou vaut 0, sinon revoie false	$x = 2.3; $vide = empty ($x) ; // false
is_int ($nom_var) **is_integer ($nom_var)** **is_long($nom_var)**	détermine si la variable est un entier	$x = 2.3; $entier = is_int ($x); // false
is_double ($nom_var) **is_float ($nom_var)** **is_real ($nom_var)**	détermine si la variable est un double	$x = 2.3; $double = is_real ($x); // true
is_string ($nom_var)	détermine si la variable est une chaîne	$x = "2.3"; $str = is_string ($x) ; // true
is_array ($nom_var)	détermine si la variable est un tableau	
is_object ($nom_var)	détermine si la variable est un objet	
intval ($nom_var)	définit un entier	$x="2.3 degrés"; $nb = intval ($x); // 2
doubleval ($nom_var)	définit un double	$x="2.3 degrés"; $nb = doubleval ($x); // 2.3
strval ($nom_var)	définit une chaîne de caractères	$x="2.3 degrés"; $nb = strval ($x); // 2.3 degrés

- Les constantes

✓ Déclaration et test d'existence

Syntaxe en PHP	Type
define ("Ecole", "Enita Bordeaux"); **define** ("PI", 3.14); define ("font", "<FONT size=4 color=#FF0000><B>")	**Déclaration de Constantes** la variable "Ecole" vaut "Enita Bordeaux" La variable "PI" vaut "3.14" La variable "font" contient la chaîne Html
defined ("Ecole") **defined** ("Pi")	**Test d'existence** renvoi 1 (VRAI) car la constante existe renvoi 0 (FAUX) car la constante n'existe pas (Rappel : Pi est différent de PI)

Remarque : Un nom de constante, contrairement à un nom de variable, ne débute pas par un signe **$**

✓ Utilisation

Les constantes peuvent être utilisées à la place de leurs valeurs.
Exemple : echo ("<HR>" . font . ecole); // le signe . signifie que les variables seront concaténées.
Le résultat sera :

Enita Bordeaux (écrit en en taille 4, en caractères gras rouges et précédé d'une ligne horizontale)

✓ Définition des constantes pour un ensemble de pages Php

Lorsqu'un même script doit être utilisé dans des pages Html différentes, afin de faciliter sa maintenance, il est conseillé d'écrire ce script dans un fichier texte séparé (suffixé par exemple .PHP).

PHP possède 2 fonctions qui concernent l'inclusion de fichiers externes :

<? require ("MesScripts.php"); ?>	A l'exécution, cette instruction est remplacée par le contenu de "MesScripts.php". Elle ne peut pas être utilisée dans une boucle
<? for ($i = 1 ; $i < 3 ; $i++) **{ include ("MesScripts" . $i . ".php"); }** **?>**	Permet d'inclure un fichier ou un autre selon la valeur de $i. Avec un "require", ce serait toujours le 1er fichier qui serait exécuté.

Remarque : Les fichiers externes sont cherchés dans le répertoire spécifié dans la directive "include_path" du fichier php.ini (cf : configuration du Php)

✓ Constantes internes

Ce sont des constantes prédéfinies et connues de PHP.

Déclaration en PHP	Valeur
TRUE et **FALSE**	valent 1 et 0
PHP_VERSION	numéro de version de l'analyseur PHP exemple : echo (PHP_VERSION); // 4.0.5
PHP_OS	donne le système d'exploitation côté serveur exemple : echo (PHP_OS); // WINNT

Déclaration en PHP	Valeur
phpinfo();	cette fonction affiche une liste de données qui concernent l'environnement du script au niveau du serveur et du client (en autre, informations sur le type de serveur, chemin et le nom du script appelé, navigateur du client, …)

Les opérateurs

Les opérateurs vont permettre de manipuler les variables (comparaison, affectation, calculs, …). Il en existe de différentes catégories. Tout comme en mathématique, certains signes ont des priorités de calcul par rapport à d'autres, les opérateurs PHP ont des priorités décrites en annexe p 34.

✓ Les opérateurs de calcul + - * /

Ils servent aux calculs : * (multiplier), + (additionner), - (soustraire), / (diviser), = (affecter) , % (prendre le reste de la division = modulo).

Exemples : $a * 5.2 ; $b / ($c + 4) 10 % 2

✓ Les opérateurs de comparaison ==, <, …

Ils sont utilisés dans les instructions de conditions (if …. then ….).

Exemples : if ($x == 0) signifie si x = 0
if ($x != 0) signifie si x différent de 0

Les autres opérateurs sont listés en annexe page 32

✓ Les opérateurs associatifs += -= *= /=

Ils permettent à la fois le calcul et l'affectation de la valeur calculée à une variable.

Exemples : $x + = $y équivalent de $x = $x + $y

✓ Les opérateurs logiques && (ou and) || (ou or) !

Ils servent à comparer 2 expressions.

Exemples : if ($x > 10 && $y < b) … si x > 10 et y < b alors …
if ($x < 0 || $z == 1) … si x < 0 ou z = 1 ou les deux alors …
if ($x < 0 xor $z == 1) … si x < 0 ou z = 1 mais pas les deux alors …
if (! $x) … si $x ne vaut pas true …
if ($x) … si $x vaut true …

✓ Les opérateurs unaires + et -

L'opérateur – est courant puisqu'il sert à inverser la valeur donnée :

Exemples : - $x si $x = 2 alors $x = -2
- $y si $y = -4 alors $y = 4

✓ Les opérateurs d'incrémentation ++ et --

Ils servent à augmenter (ou diminuer) de 1 la valeur d'une variable. Ils sont très utilisés dans les structures de boucles.

Exemples : $i++ équivalent de $i = $i + 1
$i-- équivalent de $i = $i - 1

✓ L'opérateur d'affectation =

Il sert à donner une valeur à une variable.

Exemples : $x = 10 x prend la valeur 10
$y = "Enita Bordeaux" y est initialisé par une chaîne de caractère

✓ L'opérateur de concaténation de chaînes de caractères .

Avec PHP, il existe 2 façons de concaténer des chaînes : avec opérateur et sans opérateur.
L'opérateur "." sert à associer des chaînes de caractères pour en former une nouvelle.

Exemples : $departement = "DES" ;
$chaine1 = "- ENITA Bordeaux - " . $departement . " – " ;

Attention : Le point doit être précédé et suivi d'un espace pour éviter de le confondre avec le point décimal ("2 . 3" donnera "23" – 2.3 sera considéré comme la valeur décimale 2.3)

Sans opérateur, on peut créer de nouvelles chaînes en intégrant le nom de la variable dans la chaîne. Sans contexte, l'écriture est simplifiée et est donc conseillée.

Exemples : $departement = "DES" ;
$chaine1 = "- ENITA Bordeaux - $departement – " ;

Attention : Si l'on veut écrire "DES3", il convient de coder : ${departement}3 afin que PHP comprenne bien que 3 ne fait pas partie du nom de la variable.

✓ L'opérateur de suppression d'erreur @

Il sert à éliminer la génération de messages d'erreur provenant de fonctions intégrées.

Exemples : print ($x / 0) ; // provoque une erreur division par zéro
@ print ($x / 0) ; // aucune erreur est signalée, aucune valeur n'est imprimée.

✓ L'opérateur de données binaires

Ces opérateurs sont rarement utilisés. Ils sont décrits en annexe page32.

✓ Autres opérateurs

Il existe d'autres opérateurs que nous détaillerons plus loin :

new, **->**, **&**, **typeof**, **in**, **instanceof** et **this**

Les 3 derniers s'appliquent aux objets.

Les instructions

Une fonction contient une série d'instructions qui permettent de réaliser des boucles (structures itératives), des conditions, des affectations de valeurs, de la gestion des chaînes de caractères, des conversions, … Les instructions les plus importantes sont détaillées ci-après (p 21 et suivantes).

Fonctions et procédures

- Définitions

- **Les fonctions** et **les procédures** regroupent un ensemble d'instructions réutilisables simplement et qui accomplissent un ensemble d'opérations. Elle peuvent (mais ce n'est pas obligé) accepter des

valeurs (appelées "arguments" ou "**paramètres**"). Si il y en a plusieurs, les arguments sont séparés par des virgules.

- **une fonction** réalise une succession d'instructions et renvoie une (et une seule !) valeur issue d'un calcul (instruction **return**)
- **une procédure** réalise une succession d'instructions mais ne renvoie pas de valeur en retour.

• Création de fonctions et de procédures

- Que l'on écrive une fonction ou une procédure, le nom de la fonction et/ou de la procédure est précédé du mot-clé **function**
- Les instructions qui composent la fonction/procédure sont encadrées de **{** et **}**

✓ **Exemple d'une fonction avec un argument**

```
Function calcul_surface_cercle ($rayon)
{
  /* cette fonction calcule la surface d'un cercle à partir du
     rayon passé en paramètre */
   $surf=3.14 * $rayon * $rayon;
   return $surf;      // ou return ($surf);
}
```


✓ **Exemple d'une fonction à 2 arguments**

```
Function calcul_surface_rectangle ($longueur, $largeur)
{
  /* cette fonction calcule la surface d'un rectangle à partir des
     longueur et largeur passées en paramètre  */
   return $longueur*$largeur;      // calcule la surface et
                                   // renvoie la valeur
}
```

✓ **Exemple d'une procédure à 2 arguments**

```
function affichage_prix ($s_nom_article, $d_prix)
{
   echo ("<B> $s_nom_article</B>");  // nom article en gras
   echo ("<FONT color=red> $d_prix FF </FONT>"); // prix en
                                      // FF en rouge
   echo ("<FONT color=green>" . $d_prix*6.556597 .
         "€</FONT>");     // prix en euros en vert
   echo "<HR>";      // ligne horizontale
}
```

• Utilisation de fonctions ou de procédures

Pour appeler les 3 exemples ci-dessus, on écrira :

```
// appel fonction calcul_surface_cercle()
$surf_cercle = calcul_surface_cercle(5);          // $surface = 78.5
```

```
// appel fonction surface_rectangle()
$surf_rect = surface_rectangle(5, 10);   // $surface = 50
```

```
// appel procédure affichage_prix()
affichage_prix ("armoire", 1000);
```

Le résultat de la procédure donnerait l'affichage suivant sur le navigateur :
armoire 100 FF 6556.597€

- Transmission des arguments : par valeur et par référence

- Par défaut, "**Par valeur**" est le type de transmission par défaut, c'est-à-dire que la variable conserve sa valeur initiale après l'appel de la fonction même si celle-ci l'a modifiée.
Exemple : A partir d'une fonction qui calcule le double d'une valeur données en paramètre :

```
function double ($n)
{
  $n = $n * 2;
  return $n;
}
```


Si on passe l'argument "par valeur" à cette fonction **double()**, on obtient :

```
$a=5;// valeur initiale. On va en calculer le double
echo ("double=" . double ($a)  .  " valeur= " . $a) ;  // double=10  valeur=5
```

- "**Par référence**" signifie que est la valeur initiale peut être modifiée dans la fonction.
Syntaxe : Un symbole **&** précède le nom de l'argument (variable).
Exemple : Si on passe l'argument "par référence" à la même fonction **double()**, on obtient :

```
$a=5;          // valeur initiale. On va en calculer le double
echo ("double=" . double (&$a)  .  " valeur= " . $a) ; // double=10  valeur=10
```

✓ **Fonctions préexistantes**

Fonctions	Rôle	Exemple
eval()	retourne le résultat numérique d'un paramètre. C'est une fonction très puissante car elle permet à la fois, d'examiner, d'interpréter et d'exécuter une chaîne Php comme s'il s'agissait d'un script – Le code passé à eval() doit être correct. - Les variables utilisées dans la fonction eval() sont accessibles dans le script principal.	$oper ="*"; $val1= 10; $valeur = "$val1 = $val1" . $oper . "5;"; $x= eval($valeur); // $x=50

Utiliser les mêmes fonctions et constantes dans différentes pages Html

Parfois, une fonction, procédure ou constante doit être utilisée dans des pages Html différentes. Plutôt que de les recopier dans chacune des pages, , il est conseillé d'écrire ces scripts dans un fichier texte séparé (suffixé par exemple .PHP). Ceci a pour grand avantage une facilité de maintenance.

PHP possède 2 fonctions internes qui concernent l'inclusion de fichiers externes :

<? require ("MesScripts.php"); ?>	A l'exécution, cette instruction est remplacée par le contenu de "MesScripts.php". Elle ne peut pas être utilisée dans une boucle
<? for ($i = 1 ; $i < 3 ; $i++) { include ("MesScripts" . $i . ".php"); } ?>	Permet d'inclure un fichier ou un autre selon la valeur de $i. Avec un "require", ce serait toujours le 1er fichier qui serait exécuté.

Remarque : Les fichiers externes sont cherchés dans le répertoire spécifié dans la directive "include_path" du fichier php.ini (cf : configuration du Php)

Structures algorithmiques

- Instructions de condition : Si ... alors ... Si ... alors ... sinon ...

Si... Alors ... [Sinon ...]	Exemple

if (condition)
{
instructions;
... ;
}
	if ($x > 1) // si x > 1
	{
	if ($y == "ha") // si y = 0
	{
	echo ("la surface est de " + x + " hectares") ;
	// affichage du résultat
	}
	}

else
{
instructions ;
}
	else // Instructions facultatives
	{
	echo ("valeur impossible");
	}

Remarque : Il est tout à fait possible de faire des conditions imbriquées. Pour des facilités de lecture du script, il est conseillé de décaler vers la droite les blocs **if** internes comme dans l'exemple ci-dessus.

PHP possède aussi le mot-clé "**elseif**" qui permet d'associer un nouveau "if" dans le "else".

Si... Alors ... [Sinon si...]	Exemple

if (condition)
{
instructions;
... ;
}
	if ($x > 1) // si x > 1
	{
	echo ("la surface est de " + $x + " hectares") ;
	}

elseif (condition)
{
instructions ;
}
	elseif ($x < 0) // Instructions facultatives
	{
	echo ("valeur impossible");
	}

- Instructions de boucle : Pour …

Pour …	Exemple : somme des 10 premiers nombres entiers

for (expression_initiale ; condition_de_sortie ; expression de progression)
{
  instructions;
}
	for ($i=1 ; $i <= 10 ; $i++) // pour i variant de 1 à 10
	  // (inclus) par pas de 1
	{
	  $somme = $somme + $i;
	}

Dans une page HTML, si la boucle est entrecoupée de code HTML, on déterminera la fin de la boucle par l'instruction **<? endfor ?>**

Exemple :

```
<? for ($1 = 1 ; $i < 4 ; $i++)  ?>
<H3> Hello ! </H3>
<? endfor ?>
```

Remarque : La condition de sortie peut être un appel à une fonction.
for $1 = 1 ; mafonction($i) != "erreur" ; $i++

- Instructions de boucle : Tant que …

Tant que …	Exemple : factorielle 10

while (condition)
{
  instructions;
}
	$n = 10 ;
	$somme = 1 ;
	while ($n > 0) // tant que n > 0 (inclus) par pas de 1
	{
	  $somme = $somme * $n;
	  $n-- ;
	}

Remarque : Dans une boucle "Tant que …", les instructions situées dans la boucle peuvent ne **jamais être exécutées** si la condition n'est pas remplie.

- Instructions de boucle : Faire … tant que …

Faire … tant que …	Exemple : liste des options de 3ème année

do
{
  instructions;
}
while (condition);
	$texte=""; // initialisation de la chaîne
	$n=0;
	$option_3a = array ("agrotic", "forêt", "oeno", "fin") ;
	// création tableau des options de 3ème année
	
	do
	{
	  $texte = $texte . $option_3a [$n++] . "\n";
	  //concatène les options et ajoute un retour ligne \n
	}
	while ($option_3a[$n] != "fin"); // tant que différent
	  // du mot "fin"
	echo ($texte);

Remarque : Dans une boucle "Faire … tant que …", les instructions situées dans la boucle sont **exécutées au moins une fois**. même si la condition n'est pas remplie.

- ## Instructions de branchements multiples : si … si … si …

si … si … si …	Exemple : liste des options de 3ème année

switch (expression)
{
case valeur1 : instructions;
break;
case valeur2 : instructions;
break;}
default : instructions;
}
	switch ($choix_menu)
	{
	case 1 :
	case 2 : appel_fct_menu1_2(); // si menu=1 ou 2
	break;
	case 3 : appel_fct_menu3();
	break;
	default : echo ("vous n'avez pas choisi de menu");
	}

Remarques :

- L'expression peut être une simple variable (voir ce chapitre) ou une expression à analyser.
- Plusieurs "case" peuvent être empilés. Dans l'exemple ci-dessus, la fonction appel_fct_menu__2() est appelée si choix_menu vaut 1 ou 2.
- Chaque "case" est terminé par une instruction break, pour éviter que l'interpréteur n'exécute la suite des "case"
- L'instruction "default" est facultative. Elle sert à traiter tous les cas non précisés dans la liste des "case".
- Dans le cas d'une variable alphanumérique, on écrirait : case "L"
- Contrairement à d'autres langages, les valeurs peuvent aussi être des variables (ex : case $x)

- ## Interrompre une boucle : Break

L'instruction **break** permet d'interrompre prématurément une boucle **for** ou **while**. Elle est réservée aux programmeurs chevronnés qui en maîtrisent parfaitement les conséquences !

Break	Exemple : chercher une valeur

if (condition) break;	$compt=1;
	while ($compt<50)
	{
	if ($compt / 4 == 10) break ; // sortie de boucle
	$compt++;
	}
	echo ("compt vaut $compt
	"); // = 40

- ## Sauter des instructions : Continue

L'instruction **continue** permet de sauter à l'occurrence suivante de la boucle **for** ou **while**. Elle permet donc de sauter un bloc d'instructions et de continuer ensuite le bouclage (sans sortir de celui-ci comme le fait **break**).

L'instruction continue peut être évitée dans une boucle et il n'est pas conseillé d'y avoir recours.

continue	Exemple : chercher les multiples de 4

if (condition)
{
… instructions;
	for ($i=1 ; $i < 100 ; $i++)
	{
	if ($i % 4 != 0) continue ; // si reste de la

```
   continue ;
   instructions;
}
```

```
                                    // division différent de 0
      $multiple = $multiple . "," .  $i ;
}
echo ("Liste multiples $multiple"); // = 4,8,12,16,...
```

- Récursivité

Comme la plupart des langages, Php permet d'écrire des fonctions récursives, c'est-à-dire des fonctions qui s'appellent elles-mêmes ou qui appellent la fonction de départ. Le passage des paramètres se faisant par recopie des valeurs, les valeurs des variables ne sont pas écrasées.

Exemple : somme des n premiers entiers

```
// si n vaut 1 alors la somme du premier entier vaut 1
// sinon elle vaut la somme de n + la somme des n- 1 premiers
entiers
function somme_n_entiers($n)
{
    return $n == 1 ? 1 : $n + somme_n_entiers ($n - 1);
}

// la somme des 10 premiers entiers vaut :
echo ("somme " . somme_n_entiers(10) . "<BR>");
```

Les formulaires

Dans une page Html, l'interactivité entre l'internaute et le serveur se fera par l'intermédiaire des formulaires. En effet, à partir de données saisies et validées dans une page Html, le serveur va pouvoir traiter les données et renvoyer au client la confirmation de la bonne réception de ses valeurs ou le résultat de calculs par exemple.

Formulaire et HTML

Un formulaire correspond à une zone particulière d'une page HTML : elle est encadrée par les balises **<FORM>** et **</FORM>**
Dans ce formulaire, on peut y trouver différentes zones de saisie appelées **contrôles**. Ces contrôles correspondant à des balises HTML, nous allons les décrire succinctement.

- Exemple de formulaire

- Les différents contrôles d'un formulaire

les contrôles	rôle et syntaxe HTML
Zone de texte	utiliser pour saisir une valeur alphanumérique limitée en taille <INPUT type=text name=txt_nom size=5>

Liste de choix	utiliser pour afficher une liste de valeurs qui se déroule quand on clique sur la flèche bas.
	`<select name ="lst_choix" size=1>`
	`<option value ="a" > aaaaa </option>`
	`<option value ="b" > bbbbb </option>`
	`<option value ="c" > ccccc </option>`
	`</select>`

Liste de valeurs	c'est une liste de choix particulière ou les choix s'affichent à concurrence de la valeur donnée dans **size**. Une barre de défilement s'affiche si le nombre de valeurs dépasse **size**
	`<select name ="lst_valeur" size=4>`
	`<option value ="m" > Martin </option>`
	`<option value ="dt" > Dupont </option>`
	`<option value ="dd" > Dupond </option>`
	`<option value ="l" > Louis </option>`
	`</select>`

Case à cocher	utilisé pour des valeurs de type "oui" (case cochée) ou "non" (case décochée) si l'option "checked" est ajoutée, la case sera pré-cochée. `<input type="checkbox" name="chk_1" value="choix1" checked>`

Radio-bouton	utilisé pour sélectionner un choix et un seul parmi plusieurs. Au maximum, un seul radio-bouton reste coché. Si l'option "checked" est ajoutée, le radio-bouton sera pré-cochée.
	`<input type="radio" name="rd_1" value="radio1"> choix1`
	`<input type="radio" name="rd_1" value="radio2" checked> choix2`

Fichier	utilisé pour accéder à un fichier particulier `<input type="file" name="file" enctype= "multipart/form-data">`
Image	utilisé pour rendre une image active à un événement (onclick par exemple) `<input type="image" border="0" name="img" src="/Images/logo_enitab.gif" width="150" height=60>`
Zone de texte multiligne	utilisé pour saisir une grande quantité de texte `<textarea name="textfield" row=3 cols=40></textarea>`
Champ caché	utilisé pour passer en paramètres, des valeurs qui ne doivent pas être connues de l'internaute `<input type="hidden" name="hid_field">`
Bouton simple	utilisé pour créer des boutons sur lequel on gèrera un événement (onclick par exemple). Le texte affiché est dans l'attribut "value" `<input type="button" name="bt_bouton" value="mon Bouton">`
Bouton Soumettre	aussi appelé "submit", il permet de valider le formulaire et d'envoyer son contenu au serveur. Le texte affiché est dans l'attribut "value" `<input type = "submit" name = bt_submit value="Soumettre" >`

Bouton Rétablir	aussi appelé "reset", il permet de réinitialiser tous les champs du formulaire avec leur valeur par défaut. Le texte affiché est dans l'attribut "value" `<input type="reset" name="Submit" value="Rétablir">`

- ## Utilisation d'un formulaire

Lorsqu'un client veut retourner des données au serveur Web, le navigateur utilise la procédure de transmission de formulaire.
La procédure de soumission d'un formulaire est contrôlée par 2 attributs de la balise <FORM> : METHOD et ACTION.

- L'attribut **METHOD** détermine la manière dont les données sont transmises au serveur. Il peut prendre les valeurs GET et POST.
  - **POST** demande à l'explorateur d'empaqueter toutes les données et de les transmettre au serveur. Les données ne sont pas visibles par l'internaute au moment de l'envoi
  - **GET**, quant à lui, envoie les données en tant que partie intégrale de l'URL pour la page cible. De ce fait, les données sont visibles dans l'URL de la page cible. Elles se présentent ainsi :
    http://www.enitab.fr/nouv_page.php?lst_valeur=Dupont&txt_nom=azer
- L'attribut **ACTION** spécifie la page cible pour les données soumises. Cet attribut n'est pas obligatoire. Dans ce cas, la page en cours est re-exécutée.

Exemple :

```
<Form method="post" action="http://www.enitab.fr/php/test.php">
   <INPUT TYPE = "text" NAME = "txtnom"> <BR>
   <INPUT TYPE = "text" NAME = "txtemail"> <BR>
   <INPUT TYPE  = "SUBMIT"> <BR>
</FORM>
```


Formulaire et PHP

Associé aux formulaires, Php facilite grandement le remplissage des listes de choix et la récupération des valeurs dans une autre page. Et ce ne sont que deux exemples !

- ## Créer des listes de choix

Pour créer des listes prédéfinies et les afficher selon un critère de tri préétabli, nous allons utiliser la notion de tableau. Ainsi, la liste de valeurs donnée en exemple ci-dessus peut être écrite en Php de cette manière :

```
<select name="lst_valeur" size=4>
<?
// création d'un tableau à 2 dimensions (nom et value)
$nom = array( array("Martin", "m"),    // creation Nom et value
              array ("Dupont", "dt"),
              array ("Dupond", "dd"),
              array ("Louis", "l"));
$nb = count($nom);          // compte le nombre d'éléments
// création des 4 options
for ($i = 0 ; $i < $nb ; $i++)
{
  echo "<option  value ='" .
          $nom[$i][1] . "'>" .       // affectation de la "value"
          $nom[$i][0] .        // affichage du nom dans la liste
       "</option>";
}
?>
</select>
```


Par rapport à la création de la liste en HTML pur, il devient très simple d'afficher la liste triée par ordre alphabétique. Avant d'effectuer la boucle, il suffit d'ajouter :

```
sort ($nom);
```

- Afficher les valeurs saisies dans une autre page

Un formulaire a été validé par la touche **Submit**. Nous allons voir maintenant comment nous pouvons passer ces valeurs à une nouvelle afin de les réutiliser et de les afficher.
Nous allons partir du formulaire donné en exemple au début de ce chapitre. Pour simplifier le code, nous avons enlevé la présentation sous forme de tableau.

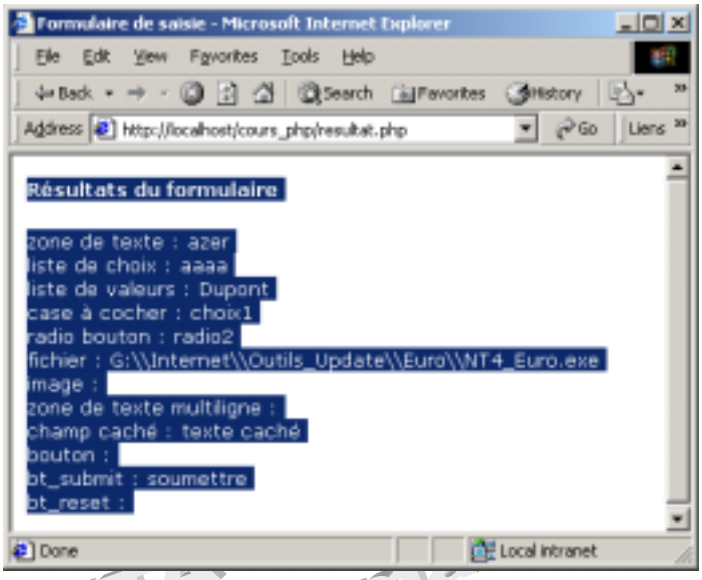
```
<HTML>
<HEAD>
<TITLE>Formulaire de saisie</TITLE>
</HEAD>
<BODY>
<p><b>Les différents contrôles d'un formulaire</b></p>
<FORM METHOD=POST ACTION="resultat.php">
   <p>Zone de texte
      <input id=zt_nbre NAME=zt_nbre size=7>
   <p>Liste de choix:
      <select NAME ="lst_choix">
        <option  value ="zzzz" > zzzz </option>
        <option  value ="mmmm" > mmmm </option>
        <option  value ="tttt" > tttt </option>
        <option  value ="aaaa" > aaaa </option>
      </select>
   <p> Liste de valeurs
      <select NAME ="lst_valeur" size=4>
        <option value ="Martin" > Martin </option>
```

```
            <option value ="Dupont" > Dupont </option>
            <option value ="Durand" > Durand </option>
            <option value ="Louis" > Louis </option>
          </select>
      <p> Case à cocher
          <input type="checkbox" NAME="chk_1" value="choix1">
      <p> Radio-boutons
          <input type="radio" NAME="rd_1" value="radio1"> choix1
          <input type="radio" NAME="rd_1" value="radio2"> choix2
      <p> Fichier
          <input type="file" NAME="file" enctype="multipart/form-data">
      <p> Image :
          <input type="image" border="0" NAME="imageField"
               src="file://k:/Commun/Images/logo_enitab.gif">
      <p>Zone de texte multiligne
          <textarea NAME="textfield" row=3 cols=40></textarea>
      <p>Champ caché :
          <input type="hidden" NAME="hiddenField">
      <p>Bouton
          <input type="button" NAME="bt_bouton" value="mon Bouton">
      <p>Boutons Soumettre et Rétablir</td>
          <input type="submit" NAME=bt_submit value="soumettre">
          <input type="reset" NAME="bt_reset" value="Rétablir">
    </FORM>
</BODY>
</HTML>
```


Nous avons vu qu'il existe 2 méthodes d'envoi des données au serveur. Contrairement à ASP qui nécessite des instructions différentes selon la méthode employée, PHP fonctionne exactement de la même manière quelque soit la méthode utilisée.

✓ Méthode POST ou Méthode GET

Les noms donnés à chaque contrôle du formulaire par l'attribut **name**, vont être automatiquement compris par PHP comme étant des variables qu'il faudra faire précédées de **$** dans le code PHP.Au clic sur le bouton "Soumettre", la page "**resultat.php**" va s'afficher. En voici son résultat :



Et voici son code :

```
<HTML>
<HEAD>
<TITLE>Formulaire de saisie</TITLE>
</HEAD>
<BODY>
<p><b>Résultats du formulaire</b></p>
<?
   echo "zone de texte : $zt_nbre <BR>" ;
   echo "liste de choix : $lst_choix <BR>" ;
   echo "liste de valeurs : $lst_valeur <BR>" ;
   echo "case à cocher : $chk_1 <BR>" ;
   echo "radio bouton : $rd_1 <BR>" ;
  echo "fichier : $file <BR>" ;
   echo "image : $imageField <BR>" ;
   echo "zone de texte multiligne : $textField <BR>" ;
   echo "champ caché : $hiddenField <BR>" ;
   echo "bouton : $bt_button <BR>" ;
   echo "bt_submit : $bt_submit <BR>" ;
   echo "bt_reset : $bt_reset <BR>" ;
?>
</BODY>
</HTML>
```

✓ Différences entre les 2 méthodes

La différence entre les 2 méthodes intervient dans l'affichage de l'URL de la page de résultat.

- URL visible de la page resultat.php avec la méthode POST :

http://localhost/cours_php/resultat.php

- URL visible de la page resultat.php avec la méthode GET

http://localhost/cours_php/resultat.php?zt_nbre=azer&lst_choix=aaaa&lst_valeur=Dupont&chk_1=choix1&rd_1=radio2&file=G%3A%5CInternet%5COutils_Update%5CEuro%5CNT4_Euro.exe&textfield=ceci+est+une%0D%0Azone+de+texte+multiligne&hiddenField=texte+cach%E9&bt_submit=soumettre
On constate que dans ce cas, les noms des contrôles et leurs valeurs sont affichés, y compris celles des contrôles cachés.

Passer des paramètres sans utiliser de formulaire

Il existe une alternative au formulaire pour transmettre des données : le **lien hypertexte**.
La création d'un lien hypertexte permettant de soumettre des données suppose l'emploi d'une balise d'ancre <A>, qui se sert d'un attribut HREF pour désigner la page cible et lui transmettre les données une fois que l'utilisateur a cliqué sur le lien. Un point d'interrogation (?) sépare la cible, des données.

Exemple : pour transmettre le nom et la ville de notre école, on écrit :

```
<A HREF=http://www.enitab.fr/nouv_page.php?name=ENITA&
ville=Bordeaux> Cliquer ici </A>
```

En utilisant des variables PHP, on obtient ceci :

```
<? $nom="ENITA" ?>
<? $ville="Gradignan" ?>
<A HREF=http://www.enitab.fr/nouv_page.php?name=
<? echo $nom ?>&ville=<?echo $ville?>> Cliquer ici </A>
```

Annexe

Caractères spéciaux

Caractère	Code Javascript
Espace arrière	\b
Saut de page	\f
Alinéa (retour ligne)	\n
Retour chariot	\r
Tabulation	\t
Apostrophe	\'
Guillemet	\"
Antislash	\\

Les opérateurs

Les opérateurs vont permettre de manipuler les variables.

- Les opérateurs de calcul

Dans les exemples ci-dessous, on va supposer que la valeur initiale de x est 11

Opérateur	Nom	Signification	Exemple	Résultat
+	plus	addition	x + 3	14
-	moins	soustraction	x – 3	8
*	multiplié par	multiplication	x*2	22
/	divisé par	division	x /2	5.5
%	modulo	reste de la division par	x%5	1
=	a la valeur	affectation	x=5	5

- Les opérateurs de comparaison

Opérateur	Description	Exemples	
==	Egal	if (w == 1) then { ... }	Si w égal 1 alors ...
!=	Différent	if (w != 1) then { ... }	Si w différent de 1 alors ...
<	Inférieur	if (w < 1) then { ... }	Si w plus petit que 1 alors ...
<=	Inférieur ou égal	if (w <= 1) then { ... }	Si w plus petit ou égal à 1 alors ...
>	Supérieur	if (w > 1) then { ... }	Si w plus grand que 1 alors ...
>=	Supérieur ou égal	if (w >= 1) then { ... }	Si w plus grand ou égal à 1 alors ...
? :	Opérateur ternaire	x = (w>0) ? 3 : 5	Si w>0 alors x=3, sinon x=5

- Les opérateurs associatifs

Ils sont à la fois opérateur de calcul et d'attribution de valeur. (Dans les exemples suivants **x** vaut 11 et **y** vaut 5)

Opérateur	Description	Exemple	Signification	Résultat
+=	plus égal	x += y	x = x + y	16
-=	moins égal	x -= y	x = x - y	6
***=**	multiplié égal	x *= y	x = x * y	55
/=	divisé égal	x /= y	x = x / y	2.2

- Les opérateurs logiques (ou booléens)

Opérateur	Description	Exemples	
!	Opposé ("Not" = opérateur booléen)	if (! w) then { … }	Si w non vide (= non nul) alors …
&&	Et	if (x>=1) && (x<=10) then	Si x>=1 ET x<=10 alors … (si x compris entre 1 et 10 inclus)
||	Ou	if (w>=1) || (x==2)	Si w>=1 OU x=2 alors …

- Les opérateurs de données binaires

Ce type d'opérateur traite ses opérandes comme des données binaires. Les opérateurs suivants effectuent des opérations bit-à-bit, c'est-à-dire avec des bits de même poids.

Opérateur	Description	Exemples avec x = 7	
&	ET bit-à-bit	Retourne 1 si les deux bits de même poids sont à 1	9 & 12 (1001 & 1100) // 8 (1000)
|	OU bit-à-bit	Retourne 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)	9 | 12 (1001 | 1100) // 13 (1101)
^	OU bit-à-bit	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (1001 ^ 1100) // 5 (0101)

Les opérateurs suivants effectuent des rotation sur les bits, c'est-à-dire qu'il décale chacun des bits d'un nombre de bits vers la gauche ou vers la droite.

Opérateur	Description	Exemples avec x = 7	
<<	Rotation à gauche	Décale les bits vers la gauche (multiplie par 2 à chaque décalage). des zéros sont insérés à droite	6 << 1 (110 << 1) // 12 (1100)
>>	Rotation à droite avec conservation du signe	Décale les bits vers la droite (divise par 2 à chaque décalage). Les "0" qui sortent à droite sont perdus, tandis que le bit non-nul de poids plus fort est recopié à gauche	6 >> 1 (0110 >> 1) //3 (0011)
>>>	Rotation à droite avec remplissage de zéros	Décale les bits vers la droite (divise par 2 à chaque décalage). Les "0" qui sortent à droite sont perdus, tandis que des "0" sont insérés à gauche	6 >>> 1 (0110 >>> 1) // 3 (0011)

- ## Les opérateurs d'incrémentation

Ces opérateurs vont augmenter ou diminuer la valeur de la variable d'une unité. Ce qui sera fort utile, par exemple, pour mettre en place des boucles.
Dans les exemples x vaut 3.

Opérateur	Description	Exemple	Signification	Résultat
x++	Incrémentation (instruction équivalente à x=x+1)	y = x++	3 puis plus 1	4
x--	Décrémentation (instruction équivalente à x=x-1)	y= x--	3 puis moins 1	2

Remarque :
On trouve aussi l'écriture suivante : ++x ou --x. L'incrémentation (ou la décrémentation) a lieu avant exécution de l'instruction. Ainsi :
y = x++ + 2 si x=3 alors y = 5 puis x = 4
alors que y = ++x + 2 si x=3 alors x= 4 puis y = 6
Donc, il convient d'être très vigilent lorsqu'on utilise ces opérateurs combinés à d'autres.

- ## Priorités des opérateurs

Les opérateurs s'effectuent dans l'ordre suivant de priorité (du degré de priorité le plus faible ou degré de priorité le plus élevé).
Dans le cas d'opérateurs de priorité égale, de gauche à droite.

Opérateur	Signification
,	virgule ou séparateur de liste
= += -= *= /= %=	affectation
? :	opérateur conditionnel
||	ou logique
&&	et logique
== !=	égalité
< <= >= >	relationnel
+ -	addition soustraction
*** /**	multiplier diviser
! - ++ --	unaire
( )	parenthèses

Bibliographie

✓ Sites officiels du PHP

http://www.php.net (en anglais)
http://fr.php.net (en français)
http://dev.nexen.net/news doc officielles du Php
http://dev.nexen.net/docs/php/annotee/manuel_tocd.php manuel de référence

http://www.easyphp.org EasyPhp

✓ Installation

http://www.php.net/download-php.php3 Modules d'installation pour IIS3 et PWS
http://www.umesd.k12.or.us/php/win32instll.html Guide d'installation pour IIS4 sur Win NT Server
http://www.via.ecp.fr/formations/1999-00/php/ref/francais/install-unix.html Installation sous Unix

✓ Tutoriels

http://www.phpbuilder.com
http://phpwizard.net
http://www.toutestfacile.com/php
http://phpmestre.forez.com/ : apprendre avec humour

✓ Exemples de classes d'objets

http://phpclasses.upperdesign.com/browse.html : (en anglais)

✓ Autres références

www.commentcamarche.net langage SQL