PHP – MySQL

Ce tutorial a été conçu pour expliquer le plus simplement et le plus clairement possible la mise en œuvre de PHP et de la liaison vers une base de données gérée sous MySQL. Il se cantonne de façon délibérée aux éléments basiques du langage PHP.

1. Introduction à PHP

1.1. Qu'est-ce que PHP ?

PHP pour *Pre Hypertext Processor*, est un langage de script exécuté par le serveur Web qui héberge le site (comme les scripts CGI, ASP, …) et non par la navigateur du visiteur (comme une page Html, un script écrit en JavaScript ou une applet Java qui s'exécutent directement sur votre ordinateur...). La syntaxe du langage PHP est fortement inspirée de celles du langage C et du Perl.

Ses principaux atouts sont :

- la gratuité et la disponibilité du code source (PHP est distribué sous licence GNU GPL).
- sa richesse fonctionnelle : PHP comporte plus de 1000 fonctions.
- la simplicité d'écriture des scripts (?).
- la disponibilité sur le Web de nombreux scripts PHP prêts à l'emploi.
- la possibilité d'inclure le script PHP au sein d'une page Html.
- la simplicité de liaison avec des bases de données. De nombreux systèmes de base de données sont supportés, mais le plus utilisé avec le PHP est MySQL, un système de base de données gratuit et disponible sur les plateformes Unix, Linux, et Windows.

Ce langage de programmation permet essentiellement de construire des sites Web dynamiques, particulièrement lorsqu'ils sont reliés à une base de données.

1.2. Côté-client et côté-serveur

Côté-client

Dans votre apprentissage des langages de publication sur le Web, vous avez avec le langage Html, le JavaScript ou le VBscript utilisé des applications dites côté-client car elles sont utilisées en local par le navigateur (le client) de l'utilisateur final.

Détaillons ce qu'il se passe lorsque vous consultez une page Html dite statique :

- Votre navigateur envoie l'adresse URL (*Uniform Ressource Locator*) que vous avez encodée.
- Le serveur Web (l'ordinateur) qui héberge la page que vous demandez, va chercher le fichier demandé dans son disque dur et vous envoie la page Html telle qu'elle à votre navigateur.
- Votre navigateur interprète les différents langages se trouvant dans ce fichier (Html, JavaScript, CSS, etc.) et affiche la page.

Côté-serveur

Les langages utilisés côté-client sont, pour des raisons évidentes de sécurité, assez limitatifs. Les applications plus complexes seront traitées dans l'espace plus sécurisé qu'est le serveur qui héberge le site Web. Les traitements sont alors exécutés côté-serveur et seuls les résultats seront envoyés au navigateur de l'utilisateur.

Détaillons ce qu'il se passe lorsque vous consultez une page Html dite dynamique :

- Votre navigateur envoie l'adresse que vous avez encodée.
- Le serveur Web cherche dans son arborescence si le fichier existe et si celui-ci porte une extension reconnue comme une application PHP (.php, .php3, .phtml). Si c'est le cas, le serveur Web transmet ce fichier à PHP.
- PHP interprète le fichier, c'est-à-dire qu'il va analyser et exécuter le code PHP. Si ce code contient des requêtes vers une base de données MySQL, PHP envoie la requête SQL. La base de données renvoie alors les informations voulues au script qui peut les exploiter (pour les afficher par exemple).
- PHP continue d'interpréter la page, puis retourne le fichier dépourvu du code PHP (puisqu'il est exécuté) au serveur Web.
- Le serveur Web renvoie finalement le fichier au navigateur de l'utilisateur. Ce fichier ne contient plus que du Html.

Vous remarquez que le code PHP s'exécute côté-serveur. Il n'y a ainsi plus aucune trace du code PHP lorsque vous consultez le code source de la page dans votre navigateur PHP, au contraire du JavaScript où le code source reste visible.

Il est important de noter :

- que tout ce qui a trait à la présentation de la page (couleur, police, mise en forme du texte, etc..) est l'affaire du Html et des feuilles de style CSS.

- que PHP n'a donc rien à voir avec le design de votre page.

- que tout ce qui touche au comportement du navigateur est du domaine du JavaScript, lui aussi exécuté par le client.

1.3. Petite histoire du PHP

Le langage PHP a été mis au point au début d'automne 1994 par Rasmus Lerdorf. Ce langage de script lui permettait de conserver la trace des utilisateurs venant consulter son CV en ligne sur son site, grâce à l'accès à une base de données par l'intermédiaire de requêtes SQL. Ainsi, étant donné que de nombreux internautes lui demandèrent ce programme, Rasmus Lerdorf mit en ligne en 1995 la première version de ce programme qu'il baptisa Personal Sommaire Page Tools, puis Personal Home Page v1.0.

Etant donné le succès de PHP 1.0, Rasmus Lerdorf décida d'améliorer ce langage en y intégrant des structures plus avancées telles que des boucles, des structures conditionnelles, et y intégra un package permettant d'interpréter les formulaires qu'il avait développé (FI, Form Interpreter) ainsi que le support de MySQL. C'est de cette façon que la version 2 du langage, baptisée pour l'occasion PHP/FI version 2, vit le jour durant l'été 1995. Il fut rapidement utilisé sur de nombreux sites (15000 fin 1996, puis 50000 en milieu d'année 1997).

A partir de 1997, Zeev Suraski et Andi Gurmans rejoignirent Rasmus pour former une équipe de programmeurs afin de mettre au point PHP 3 (Stig Bakken, Shane Caraveo et Jim Winstead les rejoignèrent par la suite). C'est ainsi que la version 3.0 de PHP fut disponible le 6 juin 1998.

A la fin de l'année 1999, une version 4 de PHP est apparue...

2. Les outils nécessaires

2.1. Un hébergeur PHP-MySQL

Pour utiliser les bases de données avec le duo PHP - MySQL, il faut que votre hébergeur accepte ces techniques et vous permette de gérer votre propre base de données. Ce sera généralement le cas des serveurs qui fonctionnent sous Unix. Pour l'accès à une base de donnée, cela dépendra souvent de votre type d'abonnement.

Hébergement web	
Espace disque	60 Mo
Système d'exploitation	Linux
Trafic illimité	☑
Accès FTP privé	☑
PHP 4 (phpinfo)	☑
Perl 5	☑
C	☑
Python	☑
Statistiques	Urchin
Nombre maximal de requêtes par jour	30 000
Bases de données	
Espace disque illimité	☑
Taille recommandée	15 Mo
Bases MySQL	1
Nombre maximal de connexions simultanées	3

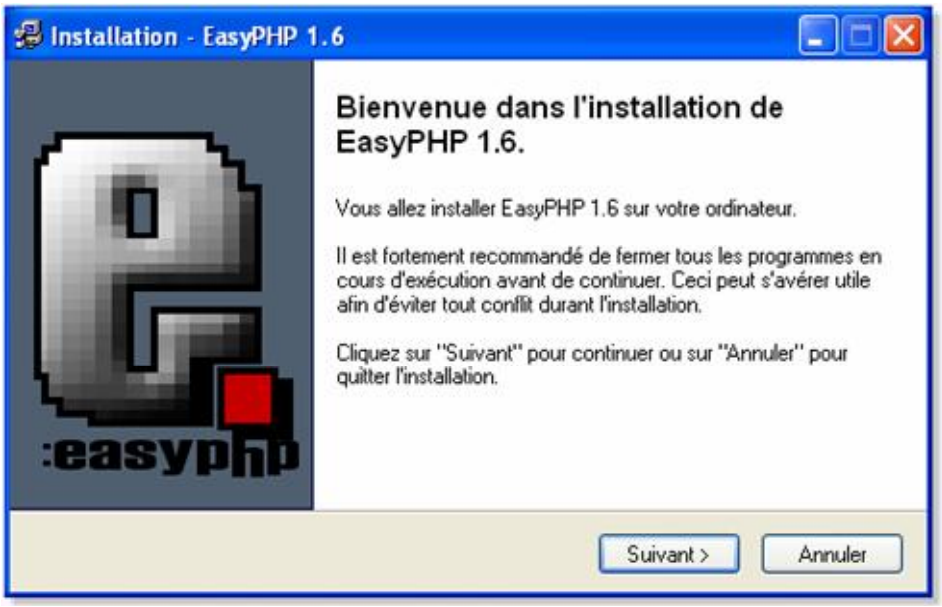
Si votre hébergeur fonctionne sous Windows, il y a peu de chances qu'il accepte le PHP et il faudra dans ce cas vous tourner vers les techniques spécifiques de Windows pour ce genre de manipulations, soit ASP et SQL.

2.2. EasyPHP en local

Cependant pour tester vos scripts, il deviendrait très vite pesant de "uploader" à chaque fois vos fichiers par FTP. C'est pourquoi installer un serveur Web en local sur son ordinateur se révèle indispensable pour tester vos scripts en direct. Vous pourrez ainsi programmer en PHP sans avoir besoin d'être connecté à Internet, ce qui peut être intéressant pour les personnes ne disposant pas (encore) de connexions au forfait.

Il existe un outil incontournable pour le PHP, c'est **EasyPHP** (www.easyPHP.org).

Ce programme permet d'installer automatiquement en quelques secondes un environnement de travail complet : soit un serveur Apache, PHP, MySql, PHPMyAdmin, etc. Vous pourrez ainsi tester localement sous Windows vos scripts PHP et vos bases de données.



Son installation ne devrait pas poser de problèmes. Sinon de nombreux articles explicatifs et autres FAQs vous attendent sur le Web. Ainsi vous trouverez, par exemple, des solutions à vos problèmes éventuels à l'adresse www.asp-php.net/tutorial/asp-php/installation.php

2.3. Un éditeur de texte

Un script PHP est, comme la plupart des langages de programmation, un simple fichier texte (ASCII 7 bits sans caractères accentués). Ainsi, un simple éditeur de texte comme le Bloc-notes (Notepad) de Windows fera très bien l'affaire.

2.4. Une documentation PHP

Lorsqu'on écrit des sites dynamiques en PHP, on ne gardera pas en tête les 1000 et quelques fonctions du PHP. On fonctionne de façon pragmatique. On retient bien entendu les fonctions et règles de base (voir ce tutorial) mais on se plongera dans la documentation pour les points plus spécifiques.

3. L'implantation du code

3.1. Implantation au sein du code Html

Pour que le script soit interprété par le serveur, deux conditions sont nécessaires :

- le fichier contenant le code doit avoir l'extension .php et non .html.
- le code PHP contenu dans le code HTML doit être délimité par les balises <?php et ?>.

Pour des raisons de conformité avec certaines normes (XML et ASP par exemple), plusieurs balises peuvent être utilisées pour délimiter un code PHP :

<?php ... ?>	La plus académique. Obligatoire si vous envisagez d'inclure du PHP dans des fichiers XML ou XHTML.
<? …?>	La plus utilisée. Cette notation abrégée doit être activée dans le fichier de configuration php.ini. Ce qui est généralement le cas.
<script language="php"> … </script>	La plus longue à la façon de la déclaration des scripts JavaScript ou VBscript.
<%php …%>	Pour le compatibilité avec ASP.

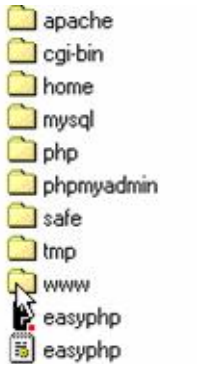
3.2. Un exemple de script simple

On ouvre le Bloc-notes de Windows et on encode ce qui suit :

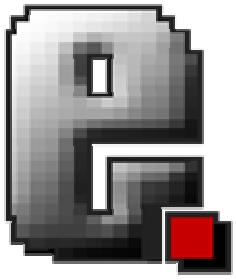
```
<html>
<head>
<title>Exemple</title>
</head>
<body>
<?php
echo "Bonjour";
?>
</body>
</html>
```

On notera à ce stade que la fonction echo permet d'afficher une chaîne de caractères délimitée par des guillemets.

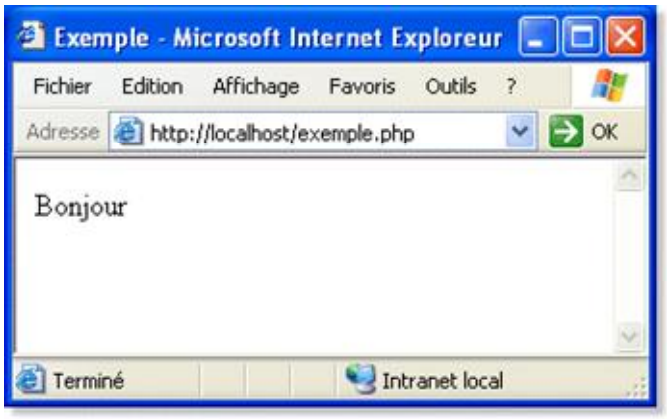
On enregistre le fichier sous le nom "exemple .php" et dans le dossier www de EasyPHP (Program Files **à** Easyphp **à** www).



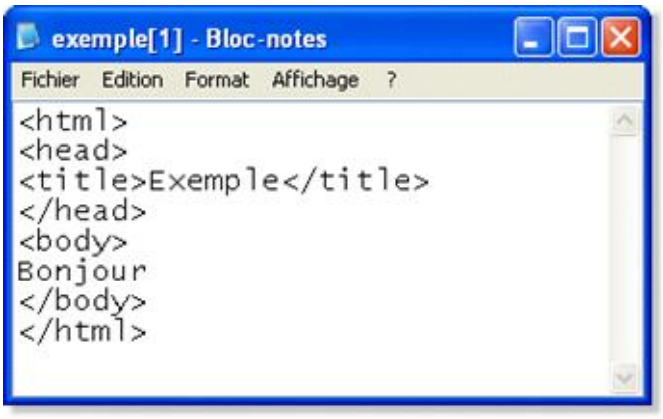
On lance EasyPHP pour mettre en œuvre le trinôme Apache - PHP - MySQL qui activera ainsi le réseau local.



On ouvre ensuite Microsoft Internet Explorer. Après avoir encodé l'adresse de la page soit http://localhost/exemple.php ou de façon équivalente http://127.0.0.1/exemple.php, on obtient ainsi dans le navigateur.



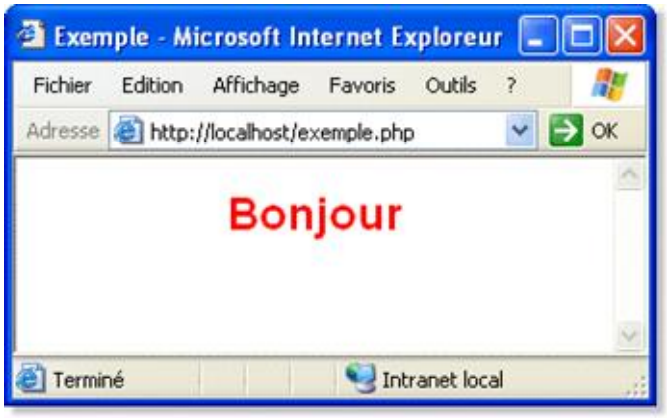
S'il nous prend la fantaisie d'afficher la source dans le navigateur (pour Internet Explorer **à** Affichage **à** Source), vous remarquerez que toute trace de votre script en PHP a disparu.



Pour agrémenter la présentation du texte, on utilise du code Html. En effet, PHP ne s'occupe pas du formatage du texte car il délègue en quelque sorte ce travail au Html.

```
<html>
<head>
<title>Exemple</title>
</head>
```

```
<body>
<?php
echo "<p align=center><b><font face=Arial size=5 color=red>Bonjour</font></b></p>";
?>
</body>
</html>
```



4. Les caractéristiques du langage PHP

4.1. L'interprétation du code

Un code PHP (celui compris entre les délimiteurs <?php et ?>) est un ensemble d'instructions se terminant chacune par un point-virgule (comme en langage C).

En PHP, toutes les instructions doivent se terminer par un point-virgule.
Le langage PHP se montre sur ce point nettement plus strict que le JavaScript.

Lorsque le code est interprété, les espaces, retours chariot et tabulation ne sont pas pris en compte par le serveur. Il est tout de même conseillé d'en mettre afin de rendre le code plus lisible (pour vous, puisque les utilisateurs ne peuvent lire le code source: il est interprété).

4.2. Les commentaires

Une autre façon de rendre le code plus compréhensible consiste à insérer des commentaires, des lignes qui seront tout simplement ignorées par le serveur lors de l'interprétation.

Pour ce faire, il est possible, comme en langage C, d'utiliser des balises qui vont permettre de délimiter les lignes d'explications afin que l'interpréteur les ignore et passe directement à la suite du fichier.

Ces délimiteurs sont /* et */

Un commentaire sera donc noté de la façon suivante :

```
/* Voici
un commentaire ! */
```

Il y a toutefois quelques règles à respecter :

- Les commentaires peuvent être placés n'importe où à l'intérieur des délimiteurs de script PHP.
- Les commentaires ne peuvent contenir le délimiteur de fin de commentaire (*/).
- Les commentaires ne peuvent être imbriqués.
- Les commentaires peuvent être écrits sur plusieurs lignes.
- Les commentaires ne peuvent pas couper un mot du code en deux.

Une autre façon d'ajouter des commentaires est le double slash (//) qui permet de mettre, sur une seule ligne, tout ce qui se situe à droite de ce symbole en commentaires.

```
// Voici un commentaire !
```

4.3. Typologie

La manière d'écrire en langage PHP a son importance. Le langage PHP est par exemple sensible à la casse (en anglais case sensitive), cela signifie qu'un nom contenant des majuscules est différent du même nom écrit en minuscules.

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().

5. Les variables en PHP

5.1. Concept de variable avec PHP

Une variable est un objet repéré par son nom, pouvant contenir des données, qui pourront être modifiées lors de l'exécution du programme. Les variables en langage PHP peuvent être :

- des scalaires (variables normales)
- des tableaux
- des tableaux associatifs

Quelque soit le type de variable, son nom doit obligatoirement être précédé du caractère dollar ($).

Contrairement à de nombreux langages de programmation, comme le langage C, les variables en PHP n'ont pas besoin d'être déclarées, c'est-à-dire que l'on peut commencer à les utiliser sans en avoir averti l'interpréteur au préalable. Ainsi, si la variable existait précédemment, son contenu est utilisé, sinon l'interpréteur lui affectera la valeur en lui assignant 0 par défaut. De cette façon si vous ajoutez 3 à une nouvelle variable (non définie plus haut dans le code), sa valeur sera 3.

5.2. Définition des variables

Avec PHP, les noms de variables doivent répondre à certains critères :

- un nom de variable doit commencer par une lettre (majuscule ou minuscule) ou un "_" (pas par un chiffre).
- un nom de variables peut comporter des lettres, des chiffres et le caractère _ (les espaces ne sont, bien entendu, pas autorisés !).

Quelques exemples :

Nom de variable correct
$Variable
$Nom_De_Variable
$nom_de_variable
$nom_de_variable_123
$nom_2_variable

Nom de variable incorrect	Raison
$Nom de Variable	comporte des espaces
$123Nom_De_Variable	commence par un chiffre
$toto@mailcity.com	caractère spécial @
$Nom-de-variable	signe - interdit
nom_de_variable	ne commence pas par $

Les noms de variables sont sensibles à la casse (PHP fait la différence entre majuscules et minuscules), il faut donc veiller à utiliser des noms comportant la même casse !

5.3. Variables scalaires

Le langage PHP propose trois types de variables scalaires:

- Entiers : nombres naturels sans décimale (sans virgule).
- Réels : nombres décimaux (on parle généralement de type double, car il s'agit de nombre décimaux à double précision).
- Chaînes de caractères : suite de caractères.

Il n'est pas nécessaire en PHP de typer les variables, c'est-à-dire de définir leur type, il suffit de leur assigner une valeur pour en définir le type :

- Entiers : nombre sans virgule.
- Réels : nombres avec une virgule (en réalité un point).
- Chaînes de caractères : ensembles de caractères entre guillemets simples ou doubles.

Instruction	Type de la variable
$Variable = 0;	type entier
$Variable = 12;	type entier
$Variable = 0.0;	type réel
$Variable = 12.0;	type réel
$Variable = "0.0";	type chaîne
$Variable = "Bonjour tout le monde";	type chaîne

Exemple :

```
<?php
$a = 1;
$b = 3.34;
$c = "Le monde entier";
echo $a,"<br>",$b,"<br>",$c;
?>
```

Il existe des caractères repérés par un code ASCII spécial permettant d'effectuer des opérations particulières. Ces caractères peuvent être représentés plus simplement en langage PHP grâce au caractère '\' suivi d'une lettre, qui précise qu'il s'agit d'un caractère de contrôle.

Caractère	Description
\"	guillemet
\'	apostrophe
\\	barre oblique inverse (backslash)
\r	retour chariot
\n	retour à la ligne
\t	tabulation

D'une part, certains de ces caractères ne pourraient pas être représentés autrement (un retour à la ligne ne peut pas être représenté à l'écran). D'autre part, les caractères repris par le code du langage PHP comme \ et " ne peuvent pas faire partie d'une chaîne de caractère, pour des raisons évidente d'ambiguïté.

5.4. Variables tableaux

Les variables, telles que nous les avons vues, ne permettent de stocker qu'une seule donnée à la fois. Or, pour de nombreuses données, des variables distinctes seraient beaucoup trop lourdes à gérer. Heureusement, PHP propose des structures de données permettant de stocker l'ensemble de ces données dans une "variable commune".Ce sont les variables tableaux. Ainsi, pour accéder à ces valeurs, il suffit de parcourir la variable de type complexe composée de "variables" de type simple.

Les tableaux stockent des données sous forme de liste. Les données contenues dans la liste sont accessibles grâce à un index (un numéro représentant l'élément de la liste). Contrairement à des langages tels que le langage C, il est possible de stocker des éléments de types différents dans un même tableau.

Ainsi, pour désigner un élément de tableau, il suffit de faire suivre au nom du tableau l'indice de l'élément entre crochets :

```
$Tableau[0] = 12;
$Tableau[1] = "Bonjour";
```

Avec PHP, il n'est pas nécessaire de préciser la valeur de l'index lorsque l'on veut remplir un tableau, car il assigne la valeur 0 au premier élément (si le tableau est vide) et incrémente les indices suivants. Le code précédent est équivalent à :

```
$Tableau[] = 12;
$Tableau[] = "Bonjour";
```

Il est important de noter que :
- les indices de tableau commencent à zéro.
- tous les types de variables peuvent être contenus dans un tableau.

Une autre façon de créer un tableau est de passer par l'élément du langage PHP, array().

```
<?php
$Tableau = array(12,"Bonjour");
?>
```

Lorsqu'un tableau contient d'autres tableaux, on parle de tableaux multidimensionnels. Il est possible de créer directement des tableaux multidimensionnels en utilisant plusieurs paires de crochets pour les index (autant de paires de crochets que la dimension souhaitée). Par exemple, un tableau à deux dimensions pourra être déclaré comme suit :

```
$Tableau[0][0] = 12;
$Tableau[0][1] = "lehtml";
$Tableau[1][0] = 1245.652;
$Tableau[1][1] = "Au revoir";
```

5.5. Variables tableaux associatifs

PHP permet l'utilisation de chaînes de caractères au lieu de simples entiers pour définir les indices d'un tableau, on parle alors de tableaux associatifs. Cette façon de nommer les indices peut parfois être plus facile à utiliser.

```
$Auteur["Nom"] = "Van Lancker";
$Auteur["Prenom"] = "Luc";
$Auteur["Code_Postal"] = 7700;
```

Ou avec array() :

```
<?php
$Auteur = array(Nom=>"Van Lancker",Prenom=>"Luc",Code_Postal=>7700);
?>
```

5.6. Portée (visibilité) des variables

Selon l'endroit où on déclare une variable, celle-ci pourra être accessible (visible par PHP) partout dans le code ou bien uniquement dans une portion confinée de celui-ci (à l'intérieur d'une fonction par exemple), on parle alors de la portée (ou la visibilité) d'une variable.

Lorsqu'une variable est déclarée dans le code même, c'est-à-dire à l'extérieur de toute fonction ou de tout bloc d'instruction, elle est accessible de partout dans le code (n'importe quelle fonction du programme peut faire appel à cette variable). On parle alors de **variable globale**.

Lorsque l'on déclare une variable à l'intérieur d'un bloc d'instructions (entre des accolades), sa portée se confine à l'intérieur du bloc dans lequel elle est déclarée. On parle alors de **variable locale**.

D'une manière générale il est préférable de donner des noms différents aux variables locales et globales pour des raisons de lisibilité et de compréhension du code.

5.7. Définition de constantes

Une constante est une variable dont la valeur est inchangeable lors de l'exécution d'un programme. Avec PHP, les constantes sont définies grâce à la fonction define(). La syntaxe de la fonction define() est la suivante:

```
define("Nom_de_la_variable", Valeur);
```

Le nom d'une constante définie à l'aide de la fonction define() ne doit pas commencer par le caractère $ (de cette façon aucune affectation n'est possible).

6. Les dates et heures

PHP possède de nombreuses fonctions qui permettent de manipuler la date et l'heure sur le serveur qui exécute PHP. On peut utiliser ces fonctions pour formater la date et l'heure de nombreuses façons.

N'oubliez pas que la date et l'heure affichées seront celle du serveur (qui exécute le code) et non celle du visiteur (en JavaScript, c'est le navigateur sur le poste client qui interprète le code et donc affiche l'heure de l'ordinateur du visiteur).

6.1. La fonction date()

La fonction date(format) retourne une date sous forme d'une chaîne, au format demandé .

Format	Description	Exemple
a	"am" ou "pm" minuscules	pm
A	"AM" ou "PM" majuscules	PM
d	jour du mois	07 /12
D	jour de la semaine en 3 lettres	Mon
F	nom du mois	May
h	heure (format 12 heures avec 0 en en-tête)	12
H	heure (format 24 heures avec 0 en en-tête)	08
g	heure (format 12 heures sans 0 en en-tête)	4
G	heure (format 24 heures sans 0 en en-tête)	10
i	minutes	44
J	jours du mois (pas de 0 en en-tête)	3
m	mois de l'année (0 en en-tête)	04
M	mois de l'année en 3 lettres	jul
n	mois de l'année; pas de 0 en entête	4
s	secondes	30
y	année en 2 chiffres	02
Y	année en 4 chiffres	2003

Par exemple :

```
<?php
$date_du_jour = date("d-m-Y");
echo "Nous sommes le ".$date_du_jour;
?>
```

6.2. La fonction getdate()

La fonction getdate(champ) retourne un tableau associatif contenant les informations de date et d'heure avec les champs suivants :

Champs	Description	Exemple
seconds	secondes	23
minutes	minutes	7
hours	heures de la journée de 0 à 23	16
mday	jour du mois de 1 à 31	18
wday	jour de la semaine de 0 à 6	5
mon	mois de l'année	4
year	année en 4 chiffres	2003
yday	jour de l'année de 0 à 365	185
weekday	nom du jour de la semaine (en anglais)	Monday
month	mois de l'année (en anglais)	January

Par exemple :

```
<?php
$aujourdhui = getdate();
$mois = $aujourdhui['month'];
$jour = $aujourdhui['mday'];
$annee = $aujourdhui['year'];
echo "$jour/$mois/$annee";
?>
```

7. Les opérateurs

7.1. Qu'est-ce qu'un opérateur?

Les opérateurs sont des symboles qui permettent de manipuler des variables, c'est-à-dire effectuer des opérations, évaluer des variables, etc…

On distingue plusieurs types d'opérateurs :
- les opérateurs de calcul
- les opérateurs d'assignation
- les opérateurs d'incrémentation
- les opérateurs de comparaison
- les opérateurs logiques

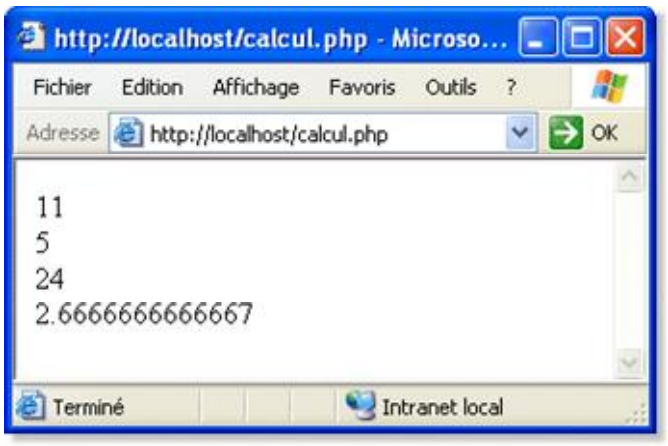
7.2. Les opérateurs de calcul

Les opérateurs de calcul permettent de modifier mathématiquement la valeur d'une variable.

Opérateur	Dénomination	Effet	Exemple	Résultat
+	Addition	Ajoute deux valeurs	$x+3	10
-	Soustraction	Soustrait deux valeurs	$x-3	4
*	Multiplication	Multiplie deux valeurs	$x*3	21
/	Division	Divise deux valeurs	$x/3	2.3333333
=	Opérateur d'affectation	Affecte une valeur à une variable	$x=3	Met la valeur 3 à la variable $x

Exemple :

```
<?php
$a = 8;
$b = 3;
echo $a + $b."<br>";
echo $a - $b."<br>";
echo $a * $b."<br>";
echo $a / $b."<br>";
?>
```



7.3. Les opérateurs d'assignation

Ces opérateurs permettent de simplifier des opérations telles que ajouter une valeur dans une variable et stocker le résultat dans celle-ci. Une telle opérations s'écrirait habituellement de la façon suivante par exemple: $x=$x+2

Avec les opérateurs d'assignation il est possible d'écrire cette opération sous la forme suivante: $x+=2 Ainsi, si la valeur de x était 5 avant opération, elle sera de 7 après…

Les opérateurs de ce type sont les suivants:

Opérateur	Effet
+=	addition deux valeurs et stocke le résultat dans la variable (à gauche).
-=	soustrait deux valeurs et stocke le résultat dans la variable.
*=	multiplie deux valeurs et stocke le résultat dans la variable.
/=	divise deux valeurs et stocke le résultat dans la variable.
%=	donne le reste de la division deux valeurs et stocke le résultat dans la variable.
|=	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable.
^=	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable.
&=	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable.
.=	Concatène deux chaînes et stocke le résultat dans la variable.

7.4. Les opérateurs d'incrémentation

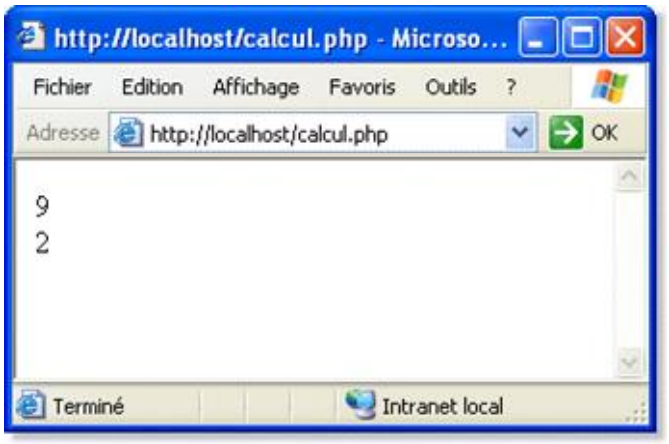
Ce type d'opérateur permet d'augmenter ou de diminuer, de façon concise, une variable d'une unité. Ces opérateurs sont très utiles pour des structures telles que les boucles qui ont besoin d'un compteur (variable qui augmente de un en un).

Un opérateur de type $x++ permet de remplacer des notations lourdes telles que $x=$x+1 ou bien $x+=1

Opérateur	Dénomination	Effet	Syntaxe	Résultat
++	Incrémentation	Augmente d'une unité la variable	$x++	6 (avec x = 5)
--	Décrémentation	Diminue d'une unité la variable	$x--	4 (avec x = 5)

Exemple avec a=8 et b=3 :

```
<?php
$a = 8;
$b = 3;
$a++;
echo $a."<br>";
$b--;
echo $b."<br>";
?>
```



7.5. Les opérateurs de comparaison

Ce type d'opérateur permet de comparer la grandeur de deux données.

Opérateur	Dénomination	Effet	Exemple	Résultat
==	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	$x==3	Retourne 1 si $x est égal à 3, sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	$x<3	Retourne 1 si $x est inférieur à 3, sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	$x<=3	Retourne 1 si $x est inférieur à 3, sinon 0
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	$x>3	Retourne 1 si $x est supérieur à 3, sinon 0
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	$x>=3	Retourne 1 si $x est supérieur ou égal à 3, sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	$x!=3	Retourne 1 si $x est différent de 3, sinon 0

Ne pas confondre l'opérateur d'égalité (==) avec le signe d'affectation (=).

7.6. Les opérateurs logiques (booléens)

Ce type d'opérateur permet de vérifier si plusieurs conditions sont vraies :

Opérateur	Dénomination	Effet	Syntaxe
|| ou OR	OU logique	Vérifie qu'une des conditions est réalisée	((condition1)||(condition2))
&& ou AND	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1)&&(condition2))
XOR	OU exclusif	Opposé du OU logique	((condition1)XOR(condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

7.7. Autres opérateurs

Les opérateurs suivants ne peuvent pas être classés dans une catégorie spécifique mais ils ont tout de même leur importance !

Opérateur	Dénomination	Effet	Syntaxe	Résultat
.	Concaténation	Joint deux chaînes bout à bout	"Bonjour"."Au revoir"	"BonjourAu revoir"
$	Référencement de variable	Permet de définir une variable	$MaVariable = 2;	
->	Propriété d'un objet	Permet d'accéder aux données membres d'une classe	$MonObjet-> Propriete	

7.8. Les priorités

Lorsque l'on associe plusieurs opérateurs, il faut que l'interpréteur PHP sache dans quel ordre les traiter. Voici donc dans l'ordre décroissant les priorités de tous les opérateurs :

Priorité des opérateurs												
1	()	[]										
2	--	++	!	~	-							
3	*	/	%									
4	+	-										
5	<	<=	>=	>								
6	==	!=										
7	&											
8	^											
9	|											
10	&&											
11	||											
12	?	:										
13	=	+=	-=	*=	/=	%=	<<=	>>=	>>>=	&=	^=	|=
14	AND											
15	XOR											

8. Les structures conditionnelles

8.1. Qu'est-ce qu'une structure conditionnelle ?

On appelle les **structures conditionnelles**, les instructions qui permettent de tester si une condition est vraie ou non, c'est-à-dire si la valeur de son expression vaut 0 ou 1 (le PHP associe le mot clé true à 1 et false à 0).

Ces structures conditionnelles peuvent être associées à des structures qui se répètent suivant la réalisation de la condition. On appelle ces structures des **structures de boucle**.

8.2. La notion de bloc

Une expression suivie d'un point-virgule est appelée instruction. Par exemple a++; est une instruction.

Lorsque l'on veut regrouper plusieurs instructions, on peut créer ce que l'on appelle un **bloc**, c'est-à-dire un ensemble d'instructions (suivies respectivement par des points-virgules) et comprises entre les accolades { et }.

Les instructions if, while et for peuvent par exemple être suivies d'un bloc d'instructions à exécuter.

8.3 L'instruction if

L'instruction if est la structure de test la plus basique. On la retrouve dans tous les langages de programmation. Elle permet d'exécuter une série d'instruction si une condition est réalisée.

La syntaxe de cette expression est la suivante :

```
if (condition réalisée) {
liste d'instructions
}
```

Remarques :

- la condition doit être mise entre des parenthèses.
- il est possible de définir plusieurs conditions à remplir avec les opérateurs ET et OU (&& et ||)
  Ainsi par exemple:
  if ((condition1)&&(condition2)) teste si les deux conditions sont vraies .
  if ((condition1)||(condition2)) exécutera les instructions si l'une ou l'autre des deux conditions est vraie.
- s'il n'y a qu'une instruction, les accolades ne sont pas indispensables.

8.4. L'instruction if ... else

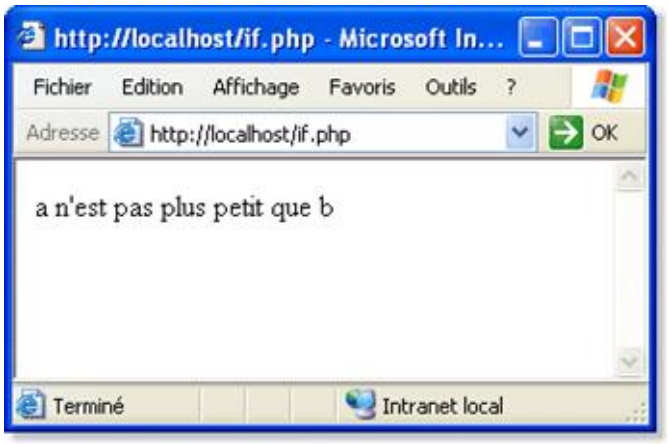
L'instruction if dans sa forme basique ne permet de tester que la réalisation d'une condition. Or la plupart du temps on aimerait pouvoir choisir les instructions à exécuter en cas de non réalisation de la condition.
L'expression if ... else permet d'exécuter une autre série d'instruction en cas de non-réalisation de la condition.

La syntaxe de cette expression est la suivante :

```
if (condition réalisée) {
liste d'instructions
}
else {
autre série d'instructions (en cas de non-réalisation).
}
```

Exemple avec a=8 et b=3 :

```
<?php
$a = 8;
$b = 3;
if ($a < $b)
{
echo "a est plus petit que b";
}
else
{
echo "a n'est pas plus petit que b";
}
?>
```



8.5. L'instruction if ... elseif ... else

Il est parfois nécessaire de tester plusieurs conditions de façon exclusive, c'est-à-dire que sur toutes les conditions une seule sera réalisée ...
L'expression if ... elseif ... else permet d'enchaîner une série d'instructions et évite d'avoir à imbriquer des instructions if.

La syntaxe de cette expression est la suivante:

```
if (condition réalisée) {
liste d'instructions
}
elseif (autre condition réalisée) {
autre série d'instructions
}
...
else (dernière condition réalisée) {
```

série d'instructions
}

8.6. Une façon plus concise (opérateur ternaire)

Il est possible de faire un test avec une structure beaucoup moins lourde grâce à la structure suivante, appelée opérateur ternaire :

(condition) ? instruction1 si vrai : instruction2 si faux

Remarques :

- la condition doit être entre des parenthèses.
- lorsque la condition est vraie, l'instruction1 est exécutée.
- lorsque la condition est fausse, l'instruction2 est exécutée.

8.7. L'instruction switch

L'instruction switch permet de faire plusieurs tests de valeurs sur le contenu d'une même variable. Ce branchement conditionnel simplifie beaucoup le test de plusieurs valeurs d'une variable. Cette opération aurait été compliquée (mais possible) avec des if imbriqués. Sa syntaxe est la suivante :

switch (Variable) {
case Valeur1:
Liste d'instructions
break;
case Valeur1:
Liste d'instructions
break;
case Valeurs...:
Liste d'instructions
break;
default:
Liste d'instructions
break;
}

Les parenthèses qui suivent le mot clé switch indiquent une expression dont la valeur est testée successivement par chacun des "**case**". Lorsque l'expression testée est égale à une des valeurs suivant un case, la liste d'instruction qui suit celui-ci est exécutée. Le mot clé **break** indique la sortie de la structure conditionnelle. Le mot clé **default** précède la liste d'instructions qui sera exécutée si l'expression n'est jamais égale à une des valeurs.

N'oubliez pas d'insérer des instructions break entre chaque test, ce genre d'oubli est difficile à détecter car aucune erreur n'est signalée...

8.8. Les boucles

Les boucles sont des structures qui permettent d'exécuter plusieurs fois la même série d'instructions jusqu'à ce qu'une condition ne soit plus réalisée.

On appelle aussi ces structures des instructions répétitives ou bien des itérations.

La façon la plus commune de faire une boucle, est de créer un compteur (une variable qui s'incrémente, c'est-à-dire qui augmente de 1 à chaque tour de boucle) et de faire arrêter la boucle lorsque le compteur dépasse une certaine valeur.

8.9. La boucle for

L'instruction for permet d'exécuter plusieurs fois la même série d'instructions.

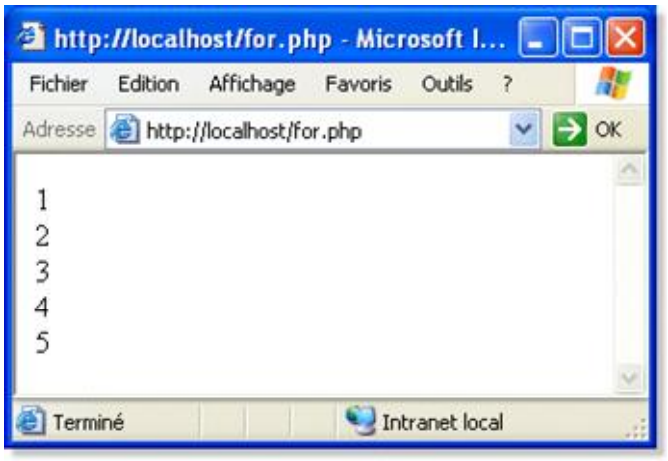
Dans sa syntaxe, il suffit de préciser le nom de la variable qui sert de compteur (et éventuellement sa valeur de départ), la condition sur la variable pour laquelle la boucle s'arrête (par exemple une condition qui teste si la valeur du compteur dépasse une limite) et enfin une instruction qui incrémente (ou décrémente) le compteur.

La syntaxe de cette expression est la suivante :

```
for (compteur; condition; modification du compteur) {
liste d'instructions
}
```

Par exemple :

```
<?php
for ($i=1; $i<6; $i++) {
echo "$i<br>";
}
?>
```

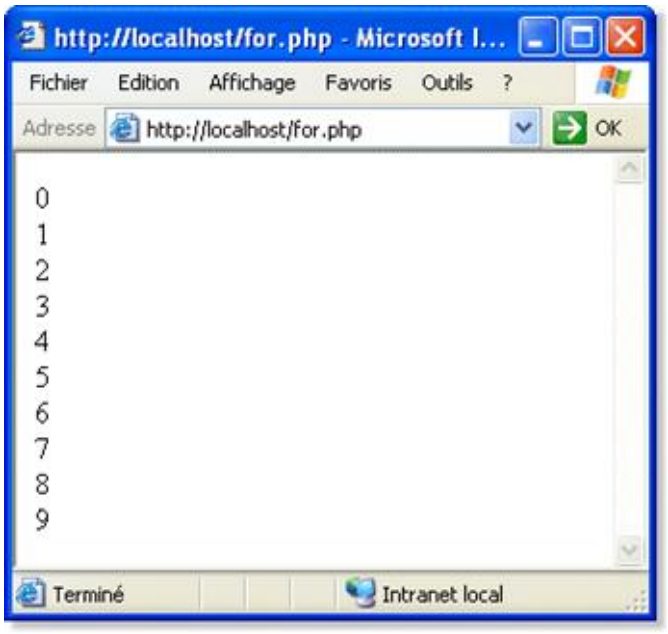


Cette boucle affiche 5 fois la valeur de $i, c'est-à-dire 1, 2, 3, 4, 5.
Elle commence à $i=1, vérifie que $i est bien inférieur à 6, jusqu'à atteindre la valeur $i=6, pour laquelle la condition ne sera plus réalisée. Alors la boucle s'interrompra et le programme continuera son cours.

Notons que le langage PHP autorise la déclaration de la variable de boucle dans l'instruction for elle-même !

Par exemple :

```
<?php
for ($i=0; $i<10; $i++) {
echo "$i<br>";
}
?>
```



- Il faudra toujours vérifier que la boucle a bien une condition de sortie.

- Une instruction echo dans votre boucle est un bon moyen pour vérifier la valeur du compteur pas à pas en l'affichant !

- Il faut bien compter le nombre de fois que l'on veut faire exécuter la boucle:
  - for($i=0;$i<10;$i++) exécute 10 fois la boucle ($i de 0 à 9).
  - for($i=0;$i<=10;$i++) exécute 11 fois la boucle ($i de 0 à 10).
  - for($i=1;$i<10;$i++) exécute 9 fois la boucle ($i de 1 à 9).
  - for($i=1;$i<=10;$i++) exécute 10 fois la boucle ($i de 1 à 10).

8.10. L'instruction while

L'instruction while représente un autre moyen d'exécuter plusieurs fois la même série d'instructions. La syntaxe de cette expression est la suivante :

```
while (condition réalisée) {
liste d'instructions
}
```

Cette instruction exécute la liste d'instructions aussi longtemps que (while en anglais) la condition est réalisée.

La condition de sortie pouvant être n'importe quelle structure conditionnelle, les risques de boucle infinie (boucle dont la condition est toujours vraie) sont grands. Une boucle infinie risque de provoquer un plantage du serveur !

8.11. Saut inconditionnel

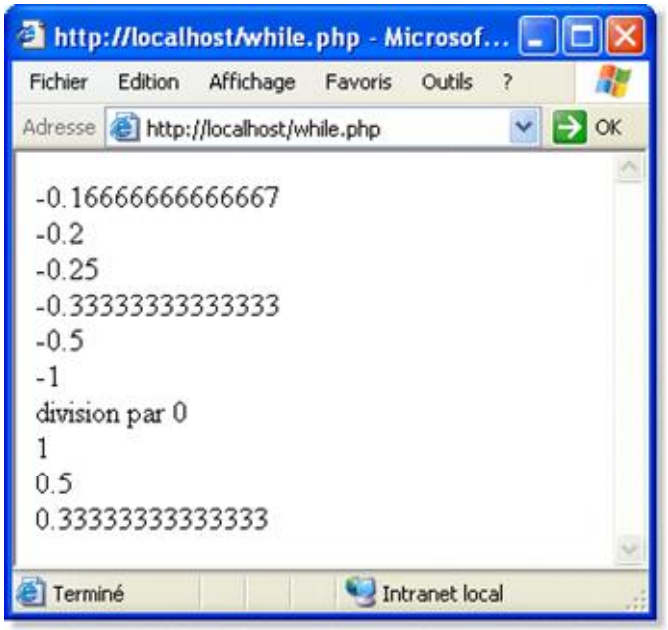
Il peut être nécessaire de faire sauter à la boucle une ou plusieurs valeurs sans pour autant mettre fin à celle-ci.

La syntaxe de cette expression est "continue;" (cette instruction se place dans une boucle). On l'associe généralement à une structure conditionnelle sinon les lignes situées entre cette instruction et la fin de la boucle ne seraient jamais utilisées.

Exemple:

Imaginons que l'on veuille imprimer pour $x allant de 1 à 10 la valeur de 1/($x-7) ... Il est évident que pour $x=7 il y aura une erreur. Heureusement, grâce à l'instruction *continue* il est possible de traiter cette valeur à part puis de continuer la boucle!

```
<?php
$x=1;
while ($x<=10) {
if ($x == 7) {
echo "division par 0 <br>";
$x++;
continue;
}
$a = 1/($x-7);
echo "$a<br>";
$x++;
}
?>
```



8.12. Arrêt inconditionnel

A l'inverse, on peut vouloir arrêter prématurément la boucle, pour une autre condition que celle précisée dans l'en-tête de la boucle. L'instruction break permet d'arrêter une boucle (for ou while). Il

s'agit, tout comme l'instruction continue, de l'associer à une structure conditionnelle, sans laquelle la boucle ne ferait jamais plus d'un tour!

Dans l'exemple précédent, il serait possible de faire arrêter la boucle en cas d'annulation du dénominateur, pour éviter une division par zéro.

```
<?php
for ($x=1; $x<=10; $x++) {
$a = $x-7;
if ($a == 0) {
echo "division par 0 - arrêt !";
break;
}
echo "1/$a<br>";
}
?>
```

8.13. Arrêt d'exécution du script

PHP autorise l'utilisation de la commande exit, qui permet d'interrompre totalement l'interprétation du script, ce qui signifie que le serveur n'envoie plus d'informations au navigateur. Le script est arrêté dans son état. Cette instruction est particulièrement utile lors de l'apparition d'erreurs…

9. Les fonctions

9.1. La notion de fonction

On appelle fonction un sous-programme qui permet d'effectuer un ensemble d'instructions par simple appel de la fonction dans le corps du programme principal. Les fonctions permettent d'exécuter dans différentes parties du programme une série d'instructions. Cela permet une simplification du code et donc une taille de programme réduite au minimum.

9.2. La déclaration d'une fonction

PHP recèle de nombreuses fonctions intégrées permettant d'effectuer des actions courantes. Toutefois, il est possible de définir des fonctions, dites fonctions utilisateurs, afin de simplifier l'exécution d'instructions répétitives.

Contrairement à de nombreux autres langages, PHP nécessite que l'on définisse une fonction avant que celle-ci puisse être utilisée, car pour l'appeler dans le corps du programme il faut que l'interpréteur la connaisse, soit qu'il connaisse son nom, ses arguments et les instructions qu'elle contient.

La définition d'une fonction s'appelle "déclaration" et peut se faire n'importe où dans le code. La déclaration d'une fonction se fait grâce au mot-clé function, selon la syntaxe suivante :

```
function Nom_de_la_fonction(argument1, argument2, ...) {
liste d'instructions
}
```

Remarques :

- Le nom de la fonction suit les mêmes règles que les noms de variables :
  - le nom doit commencer par une lettre.
  - un nom de fonction peut comporter des lettres, des chiffres et les caractères _ et &. Les espaces ne sont pas autorisés.
  - le nom de la fonction, comme celui des variables est sensible à la casse (différence entre les minuscules et majuscules).
- Les arguments sont facultatifs mais s'il n'y a pas d'arguments, les parenthèses doivent rester présentes.
- Il ne faut pas oublier de refermer les accolades.

Le nombre d'accolades ouvertes (fonction, boucles et autres structures) doit être égal au nombre d'accolades fermées!
La même chose s'applique pour les parenthèses, les crochets ou les guillemets!

Une fois la fonction définie, celle-ci ne s'exécutera pas tant que l'on ne fait pas appel à elle quelque part dans la page.

9.3. Appel de fonction

Pour exécuter une fonction, il suffit de faire appel à celle-ci en écrivant son nom (une fois de plus en respectant la casse) suivie d'une parenthèse ouverte avec éventuellement des arguments puis d'une parenthèse fermée :

```
Nom_de_la_fonction();
```

Remarques :

- le point virgule signifie la fin d'une instruction et permet à l'interpréteur de distinguer les différents blocs d'instructions.
- si jamais vous avez défini des arguments dans la déclaration de la fonction, il faudra veiller à les inclure lors de l'appel de la fonction (le même nombre d'arguments séparés par des virgules).

```
Nom_de_la_fonction(argument1, argument2);
```

9.4. Renvoi d'une valeur par une fonction

La fonction peut renvoyer une valeur (et donc se terminer) grâce au mot-clé return. Lorsque l'instruction return est rencontrée, la fonction évalue la valeur qui la suit puis la renvoie au programme appelant (programme à partir duquel la fonction a été appelée).

La syntaxe de l'instruction return est simple :

```
return valeur_ou_variable;
```

9.5. Les arguments d'une fonction

Il est possible de passer des arguments à une fonction, c'est-à-dire lui fournir une valeur ou le nom d'une variable afin que la fonction puisse effectuer des opérations sur ces arguments.

Le passage d'arguments à une fonction se fait au moyen d'une liste d'arguments (séparés par des virgules) entre parenthèses suivant immédiatement le nom de la fonction. Les arguments peuvent être de simples variables, mais aussi des tableaux ou des objets. A noter qu'il est possible de donner une valeur par défaut à ces arguments en faisant suivre le nom de la variable par le signe "=" puis la valeur que l'on affecte par défaut à la variable.

Lorsque vous voulez utiliser un argument dans le corps de la fonction en tant que variable, celui-ci doit être précédé par le signe *$*.

```
<?php
function hello($qui, $texte = 'Bonjour')
{
if(empty($qui)){ // $qui est vide, on retourne faux
return false;
else{
echo "$texte $qui"; // on affiche le texte
return true; // fonction exécutée avec succès
}
}
?>
```

Ainsi cette fonction peut être appelée de deux façons différentes :

```
<?php
// Passage des deux paramètres
hello("cher ami", "Bienvenue"); // affiche "Bienvenue cher ami"
// Utilisation de la valeur par défaut du deuxième paramètre
hello("cher ami"); // affiche "Bonjour cher ami"
?>
```

9.6. Travailler sur des variables

Lorsque vous manipulerez des variables dans des fonctions, il vous arrivera de constater que vous avez beau modifier la variable dans la fonction, celle-ci retrouve sa valeur d'origine dès que l'on sort de la fonction.

La raison se trouve dans la portée des variables, c'est-à-dire si elles ont été définies comme **variables globales** ou **variables locales**.

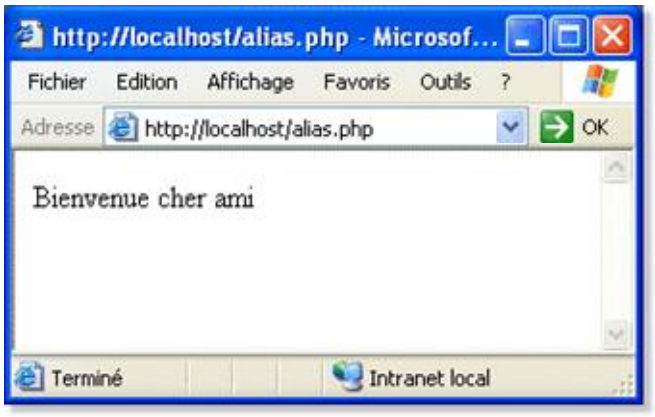
Il existe plusieurs niveaux de définition de variables :

- Une variable précédée du mot clé **global** sera visible dans l'ensemble du code, c'est-à-dire que sa portée ne sera pas limitée à la fonction seulement. Ainsi, toutes les fonctions pourront utiliser et modifier cette même variable.
- Le niveau **static** permet de définir une variable locale à la fonction, qui persiste durant tout le temps d'exécution du script.
- Par défaut, la variable possède le niveau **local**, c'est-à-dire que la variable ne sera modifié qu'à l'intérieur de la fonction et retrouvera la valeur qu'elle avait juste avant l'appel de fonction à la sortie de celle-ci.

```
<?php
$chaine = "Nombre de camions : ";
function ajoute_camion($mode='')
{
global $chaine;
static $nb=0;
$nb++; // on incrémente le nombre de camions
if($mode == "affiche"){
echo $chaine.$nb; // on affiche le nombre de camions
}
}
ajoute_camion(); // nb == 1
ajoute_camion(); // nb == 2
ajoute_camion(); // nb == 3
ajoute_camion("affiche"); // affiche Nombre de camions : 4
?>
```

9.7. Passage de paramètre par référence

Une autre méthode pour modifier une variable consiste à la faire précéder du caractère &, précisant qu'il s'agit alors d'un alias. La valeur de la variable est modifiée à la sortie de la fonction. On parle alors de passage par référence. Dans ce cas on passe la référence (adresse mémoire) de la variable à la fonction, ce qui permet de modifier sa valeur.




```
<?php
function hello($qui, &$texte)
{
$texte = "Bienvenue $qui";
}
$chaine = "Bonjour ";
hello("cher ami",$chaine);
```

```
echo $chaine; // affiche "Bienvenue cher ami"
?>
```

9.8. Retourner plusieurs variables

Lorsque vous souhaitez qu'une fonction retourne plusieurs valeurs, le plus simple est d'utiliser un tableau.

```
<?php
function nom_fonction()
{
.....

return array( $variable1, $variable2, $variable3 );
// on retourne les valeurs voulues dans un tableau
}
$retour = nom_fonction();
echo "$retour[0] - $retour[1] - $retour[2]";
?>
```

9.9. La récursivité

Les fonctions récursives sont des fonctions qui s'appellent elles-mêmes. Ce type de fonction se révèle indispensable pour parcourir une arborescence par exemple.

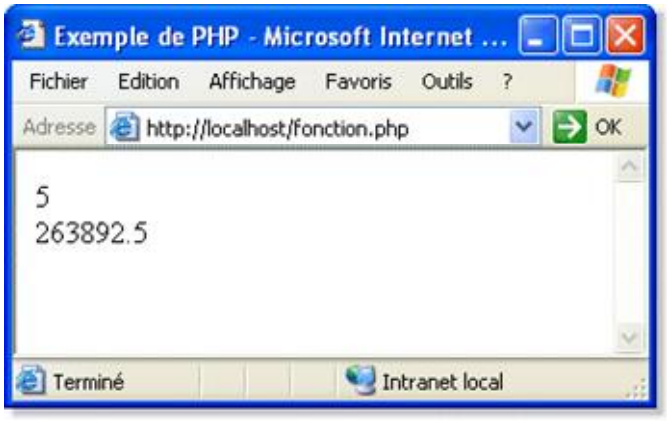
Voici un exemple.

```
<?php
function fonction_recursive($n=0)
{
$n++;
echo "$n <br>";
if($n < 10){ // si n est inférieur à 10 on continue
fonction_recursive($n);
}
}
fonction_recursive(); // affiche les nb de 1 à 10
?>
```

9.10. Exemple final

```
<html>
<head>
<title>Exemple de PHP</title>
</head>
<body>
<?php
function moyenne($a, $b) {
$media=($a+$b)/2;
return $media;
}
echo moyenne(4,6),"<br>";
echo moyenne(3242,524543),"<br>";
?>
```

```
</body>
</html>
```



10. Afficher du texte

Le but de PHP est de permettre la création de pages Web dynamiques, ainsi son but premier est de pouvoir envoyer des données au navigateur.

PHP fournit 3 fonctions permettant d'envoyer du texte au navigateur. Ces fonctions ont la particularité de pouvoir insérer dans les données envoyées des valeurs variables, pouvant être fonction d'une valeur récupérée par exemple par l'exécution d'un script ou après consultation d'une base de données. C'est ce qui rend possible la création de pages dynamiques. Les 3 fonctions sont les suivantes :

- echo
- print
- printf

10.1. La fonction echo

La fonction **echo** permet d'envoyer au navigateur la chaîne de caractères (délimitée par des guillemets) qui la suit. La syntaxe de cette fonction est la suivante :

```
echo Expression;
```

L'expression peut être une chaîne de caractères ou une expression que l'interpréteur évalue

```
echo "Chaîne de caractères";
echo (1+2)*87;
```

Ainsi, étant donné que la chaîne de caractères est délimitée par des guillemets, l'insertion de guillemets doubles dans la chaîne provoquerait une erreur. C'est la raison pour laquelle les guillemets doubles, ainsi que tous les caractères spéciaux, doivent être précédés d'un backslash. Voici un récapitulatif des caractères spéciaux nécessitant l'ajout d'un backslash :

Caractère	Description
\"	guillemet
\$	caractère $
\\	barre oblique inverse (backslash)

\r	retour chariot
\n	retour à la ligne
\t	tabulation

Le caractère $ a un rôle particulier dans la mesure où l'interpréteur le comprend comme une variable, ce qui signifie que lorsque le caractère $ est rencontré dans la chaîne qui suit la fonction echo, l'interpréteur récupère le nom de la variable qui suit le caractère $ et le remplace par sa valeur. Dans l'exemple suivant par exemple, on assigne la date actuelle à une variable appelée $madate, puis on l'affiche sur le navigateur:

```
<html>
<head>
<title>Affichage de l'heure</title>
<body>
<?php
// Récupération de la date
// et stockage dans une variable
$MaDate = date("Y");
echo "<HTML>";
echo "Nous sommes en $madate";
?>
</body>
<html>
```

10.2. La fonction print

La fonction **print** est similaire à la fonction echo à la différence que l'expression à afficher est entre parenthèses. La syntaxe de la fonction print est la suivante :

```
print(expression);
```

L'expression peut, comme pour la fonction echo, être une chaîne de caractères ou une expression que l'interpréteur évalue :

```
print("chaîne de caractères");
print ((1+2)*87);
```

10.3. La fonction printf

La fonction **printf** (empruntée au langage C) est rarement utilisée car sa syntaxe est plus complexe. Toutefois, contrairement aux deux fonctions précédentes, elle permet un formatage des données. Ce qui signifie que l'on peut choisir le format dans lequel une variable sera affichée à l'écran.

La syntaxe de printf est la suivante:

```
printf (chaîne formatée);
```

Une chaîne formatée est une chaîne contenant des codes spéciaux permettant de repérer l'emplacement d'une valeur à insérer et de son format (sa représentation). A chaque code rencontré doit être associé une valeur ou une variable que l'on retrouve en paramètre à la fin de la fonction printf. Les valeurs à insérer dans la chaîne formatée sont séparées par des virgules et doivent apparaître dans l'ordre où les codes apparaissent dans la chaîne formatée.

Les codes de formatage des types de données sont les suivants :

Code	Type de format
%b	Entier en notation binaire
%c	Caractère codé par son code ASCII
%d	Entier en notation décimale
%e	Type double (nombre à virgule) au format scientifique (1.76e+3)
%f	Type double (nombre à virgule)
%o	Entier en notation octale
%s	Chaîne de caractères
%x	Entier en notation hexadécimale (lettres en minuscules)
%X	Entier en notation hexadécimale (lettres en majuscules)
%%	Caractère %

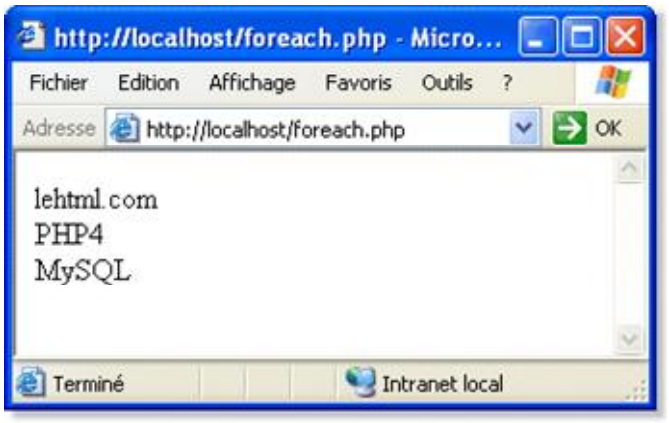
Imaginons que l'on définisse une variable en virgule flottante afin d'obtenir une précision de calcul plus grande qu'avec un entier mais qu'on désire l'afficher en tant qu'entier. Dans ce cas la fonction printf prend toute son importance :

```
<?php
$Pi = 3.1415927;
$R = 24.546;
$Perimetre = 2 * $Pi * $R;
printf ("Le périmètre du cercle est %d",$Perimetre);
?>
```

10.4. Afficher un tableau

On se servira une boucle pour afficher un tableau. On utilisera la fonction foreach(). Notons que cette fonction fait partie de PHP4.

```
<?php
$tableau = array('lehtml.com','PHP4','MySQL');
foreach ( $tableau as $contenu  ){
print $contenu.'<br>';
}
?>
```



10.5. L'implantation du code PHP au sein du code Html

Le code PHP peut être aisément implanté au sein du code Html. Cette caractéristique n'est pas à négliger car le fait d'écrire uniquement du code PHP là où il est nécessaire rend la programmation plus simple.

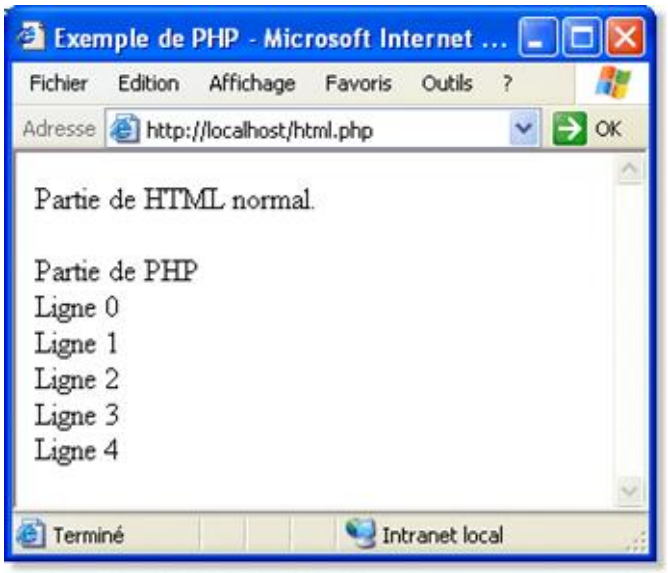
Il est en effet plus facile d'écrire du code Html que des fonctions echo ou print, dans lesquelles les caractères spéciaux doivent être précédés d'un backslash sous peine de voir des erreurs lors de l'exécution.

Un exemple parlant concerne les pages dynamiques dont l'en-tête est toujours le même. Dans ce cas, le code PHP peut ne commencer qu'à partir de la balise <body>, soit au moment où la page peut s'afficher différemment selon une variable par exemple.

Mieux, il est possible d'écrire plusieurs portions de script en PHP, séparées par du code Html statique car les variables ou fonctions déclarées dans une portion de script seront accessibles dans les portions de scripts inférieures.

Exemple :

```
<html>
<head>
<title>Exemple de PHP</title>
</head>
<body>
Partie de HTML normal.
<br><br>
<?php
echo "Partie de PHP<br>";
for($i=0;$i<5;$i++)
{
echo "Ligne ".$i."<br>";
}
?>
</body>
</html>
```



11. Traiter les chaînes de caractères

Comme tout langage de programmation, PHP permet d'agir sur les chaînes de caractères. Il propose à cet effet de nombreuses fonctions. En voici quelques unes :

Fonction	Description
strtolower()	Mise en minuscules
strtoupper()	Mise en majuscules
ucfirst()	Mise en majuscule de l'initiale
nl2br()	Remplace le \n par pour affichage
htmlspecialshars()	Convertit les caractères html
addslashes()	Fait précéder les caractères spéciaux d'un \
stripslashes()	Supprime les \
ltrim()	Supprime les espaces initiaux
trim()	Supprime les espaces en début et fin

11.1 Remplacer un mot

Soit la phrase de départ : $chaine = "Le PHP est un language de script.";

Par la fonction str_replace(), on peut remplacer language par langage.

```
$chaine1 = str_replace("language","langage",$chaine) ;
```

La même chose est possible avec la fonction ereg_replace().

```
$chaine1 = ereg_replace("language","langage",$chaine) ;
```

11.2. Rechercher un mot

On peut utiliser la fonction ereg (mot recherché, chaîne). La recherche est sensible à la casse.

Exemple :

```
<?php
if( ereg('PHP',$chaine))
{
print 'le mot PHP est bien dans la phrase';
}
else{
print 'le mot n\'est pas présent';
}
?>
```

11.3. Découper une chaîne

La fonction split() scinde une chaîne en un tableau après lui avoir fourni un séparateur.

On découpe, par exemple, la chaîne avec comme séparateur l'espace.

```
$resultat = split(" ",$chaine);
```

Le résultat obtenu est sous forme de tableau.

Le tableau obtenu se présente comme ceci :

0 Le
1 PHP
2 est
3 langage
4 de
5 script.

11.4. Vérifier si une variable est vide

La fonction empty() vérifie si la variable est vide

```
if( empty($variable_à_verifier))
{
print "La variable est vide";
}
else{
print "c'est ok";
}
```

Le contraire de empty() est la fonction isset() qui vérifie si la variable est attribuée.

```
if( isset($variable_à_verifier))
{
print "OK) la variable est attribuée";
}
else{
print "la variable est vide";
}
```

12. Les variables d'environnement

Les variables d'environnement sont, comme leur nom l'indique, des données stockées dans des variables permettant au programme d'avoir des informations sur son environnement. L'environnement, dans le cas du PHP est soit le serveur soit le client.

Ces variables sont créées par le serveur à chaque fois que le script PHP est appelé. Le serveur les lui fournit en paramètres cachés lors de l'exécution de l'interpréteur.

Elles permettent notamment d'avoir des informations sur le type de serveur, son administrateur, la date à laquelle le script a été appelé, l'adresse IP et le type de navigateur du client.

On peut donc classer les variables d'environnement en deux catégories:

- les variables d'environnement dépendant du client
- les variables d'environnement dépendant du serveur

12.1. Les variables d'environnement dépendant du client

Variable d'environnement	Description
$AUTH_TYPE	Il s'agit de la méthode d'authentification qui a été utilisée par le client pour accéder au script PHP.

$COMSPEC	Localisation de l'interpréteur de commandes sur la machine (Sous Windows).
$CONTENT_TYPE	Type de données contenues dans le corps de la requête. Il s'agit du type MIME des données.
$DOCUMENT_ROOT	Racine des documents sur le serveur.
$DOCUMENT_URI	Adresse du script PHP en relatif (à partir de la racine du serveur).
$HTTP_ACCEPT	Types MIME reconnus par le serveur (séparés par des virgules).
$HTTP_ACCEPT_ENCODING	Types d'encodage que le serveur peut réaliser (gzip,deflate).
$HTTP_ACCEPT_LANGUAGE	Langue utilisée par le serveur (par défaut en-us).
$HTTP_CONNECTION	Type de connexion ouverte entre le client et le serveur (par exemple Keep-Alive).
$HTTP_HOST	Nom d'hôte de la machine du client (associée à l'adresse IP).
$HTTP_REFERER	URL de la page qui a appelé le script PHP.
$HTTP_USER_AGENT	Cette variable permet d'avoir des informations sur le type de navigateur utilisé par le client, ainsi que son système d'exploitation.
$LAST_MODIFIED	Date et heure de dernière modification du fichier.
$PATH	Il s'agit du chemin d'accès aux différents répertoires sur le serveur.
$PATH_INFO	Chemin d'accès au script PHP en relatif (de la racine du serveur jusqu'au script PHP).
$PHP_SELF	Nom du script PHP.
$REDIRECT_STATUS	Etat de la redirection (échec ou succès).
$REDIRECT_URL	L'URL vers laquelle le navigateur du client a été redirigé.
$QUERY_STRING	Il s'agit de la partie de l'URL (ayant servie à accéder au script PHP) située après le point d'interrogation. C'est de cette manière que sont transmises les données d'un formulaire dans le cas de la méthode GET.
$REMOTE_ADDR	Cette variable contient l'adresse IP du client appelant le script CGI.
REMOTE_PORT	Cette variable permet de savoir le port sur lequel la requête HTTP a été envoyée au serveur
SCRIPT_FILENAME	Chemin d'accès complet au script PHP. Sous Windows, il sera de la forme: c:/php/php.exe
SCRIPT_NAME	Chemin d'accès relatif (par rapport au chemin d'accès à la racine Web ($DOCUMENT_ROOT)) au script PHP.

12.2. Les variables d'environnement dépendant du serveur

Variable d'environnement	Description
$DATE_GMT	Date actuelle au format GMT.
$DATE_LOCAL	Date actuelle au format local.
$DOCUMENT_ROOT	Racine des documents Web sur le serveur.
$GATEWAY_INTERFACE	Version des spécifications CGI utilisées par le serveur.
$HTTP_HOST	Nom de domaine du serveur.
$SERVER_ADDR	Adresse IP du serveur.
$SERVER_ADMIN	Adresse de l'administrateur du serveur.
$SERVER_NAME	Nom donné au serveur en local.
$SERVER_PORT	Numéro de port associé au protocole HTTP sur le serveur.
$SERVER_PROTOCOL	Nom et version du protocole utilisé pour envoyer la requête au script PHP.
$SERVER_SOFTWARE	Type (logiciel) du serveur Web.

12.3. Affichage des variables d'environnement

Il est possible de créer un script permettant d'afficher l'ensemble des variables d'environnement.

La première façon consiste à utiliser la fonction phpinfo() qui affiche un tableau récapitulatif des paramètres du serveur et de l'interpréteur PHP, ainsi qu'un tableau des variables d'environnement.

```
<?php
phpinfo();
?>
```

PHP fournit aussi la fonction getenv() permettant de retourner la valeur de la variable d'environnement passée en paramètre.

```
<?php
echo getenv("HTTP_USER_AGENT");
?>
```

Cette variable contient l'adresse IP du client appelant le script.

```
<?php
echo getenv("$REMOTE_ADDR");
?>
```

13. Les fichiers

Vous pourrez être amené à travailler avec des fichiers texte pour y stocker des informations diverses pour votre site. Avec PHP, la création ou la lecture de fichiers est, une fois de plus, assez simple. Il existe une multitude de fonctions dédiées à l'utilisation des fichiers.

13.1. La fonction fopen()

La fonction de base est la fonction fopen(). C'est elle qui permet d'ouvrir un fichier, que ce soit pour le lire, le créer, ou y écrire. Voilà sa syntaxe :

```
fopen(nom_du_fichier, mode);
```

Le mode indique le type d'opération qu'il sera possible d'effectuer sur le fichier après son ouverture. Il s'agit d'une lettre (en réalité une chaîne de caractères) indiquant l'opération possible :

Mode	Description
r	ouverture en lecture seulement.
w	ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas).
a	ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas).
r+	ouverture en lecture et écriture.
w+	ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas).
a+	ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas).

Exemple d'ouverture du fichier monfichier.txt (situé dans le dossier www) en écriture, en supprimant les données qu'il contient déjà et d'écriture dans le fichier.

```
<?php
$fichier = 'monfichier.txt';
$fp = fopen($fichier,'w');
fwrite($fp,"ici la phrase que vous souhaitez \n");
fclose($fp);
?>
```

Pour écrire dans un fichier vous pouvez aussi utiliser la fonction fputs() qui fonctionne comme fwrite().

Un fichier ouvert avec la fonction fopen() doit être fermé, à la fin de son utilisation, par la fonction fclose() en lui passant en paramètre l'entier retourné par la fonction fopen().

14. Les en-têtes HTTP

14.1. Les en-têtes HTTP

Lors de chaque échange par le protocole HTTP entre votre navigateur et le serveur, des données dîtes d'en-têtes contenant des informations sur les données à envoyer (dans le cas d'une requête) ou envoyées (dans le cas d'une réponse) sont crées. Les informations en question, généralement sur une page Web ou une image, suivent ces en-têtes. Les en-têtes HTTP permettent aussi d'effectuer des actions sur le navigateur comme le transfert de cookies ou bien une redirection vers une autre page.

Ces en-têtes sont les premières informations envoyées au navigateur (pour une réponse) ou au serveur (dans le cas d'une requête), elles se présentent sous la forme :

```
en-tête: valeur
```

La syntaxe doit être rigoureusement respectée, c'est-à-dire qu'aucun espace ne doit figurer entre le nom de l'en-tête et les deux points (:). Un espace doit par contre figurer après celui-ci !

PHP fournit une fonction permettant d'envoyer très simplement des en-têtes HTTP manuellement du serveur au navigateur (il s'agit alors d'une réponse HTTP).

La syntaxe de cette fonction est la suivante :

```
header(chaîne en-tête HTTP)
```

Etant donnée que les en-têtes HTTP sont les premières informations envoyées, la fonction header() doit être utilisée avant tout envoi de données Html au navigateur (le script qui la contient doit donc être placé avant la balise <html> et avant toute fonction echo(),print ou printf()).

La fonction header() sera utilisée, par exemple, pour rediriger le navigateur vers une nouvelle page :

```
<?php
header("location: http://www.lehtml.com/php/index.php");
?>
```

14.2. Récupérer les en-têtes de la requête

Alors que la fonction header() permet d'envoyer des en-têtes HTTP au navigateur, PHP fournit une seconde fonction permettant de récupérer dans un tableau l'ensemble des en-têtes HTTP envoyées par le navigateur. Voici la syntaxe de cette fonction :

```
Tableau getallheaders();
```

Le tableau retourné par la fonction contient les en-têtes indexés par leur nom. Voici un script permettant par exemple de récupérer des en-têtes particuliers.

```
<?php
$entetes = getallheaders;
echo $entetes["location"];
?>
```

15. La récupération de données

PHP rend très simple la récupération de données envoyées par l'intermédiaire de formulaires Html.

15.1. Création d'un formulaire

Grâce à la balise <form> du langage Html, il est possible de créer des formulaires comprenant :

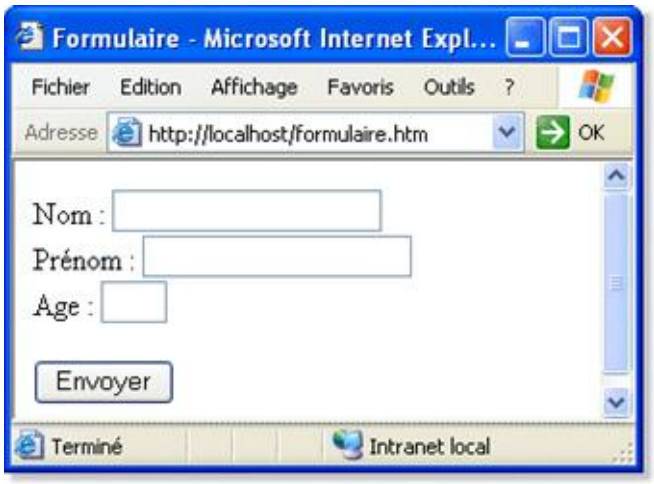
- des champs de saisie
- des cases à cocher
- des boutons radio
- des listes à choix multiples
- ...

Pour utiliser un tel formulaire capable d'envoyer des informations à un script PHP, il suffit de mettre le nom du fichier PHP qui réceptionnera les informations en tant que valeur de l'attribut action de la balise <form>.

Voici ce à quoi peut ressembler un formulaire en Html, permettant d'envoyer les coordonnées d'une personne à un fichier PHP nommé test.php:

```
<html>
<head>
<title>Formulaire</title>
</head>
<body>
<form Method="GET" Action="test.php">
Nom : <input type=text size=20 name=nom><br>
Prénom : <input type=text size=20 name=prenom><br>
Age : <input type=text size=2 name=age><br>
<input type=submit value=Envoyer>
</form>
</body>
</html>
```

Le résultat de ce code est le suivant :



15.2. Récupération et utilisation des données

Lorsque l'on soumet un formulaire à un fichier PHP, toutes les données du formulaires lui sont passées en tant que variables, c'est-à-dire chacun des noms associés aux champs (ou boutons) du formulaires précédés du caractère *$*.

Ainsi, dans l'exemple précédent, le fichier test.php reçoit les variables:

- **$nom**
- **$prenom**
- **$age**

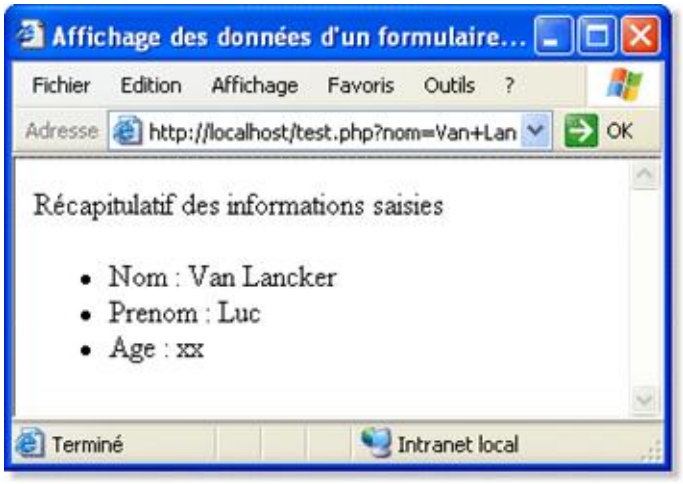
Si jamais un des champs du formulaire n'etait pas rempli, il possède la valeur "", c'est-à-dire une chaîne vide.

Voici par exemple ce à quoi pourrait ressembler le fichier test.php, dont le but est d'afficher les informations saisies par l'utilisateur à l'écran, ainsi que de vérifier que tous les champs ont bien été correctement rempli.

Voici le fichier test.php :

```
<html>
<head>
<title>Affichage des données d'un formulaire</title>
</head>
<body>
<?php
if (($nom=="")||($prenom=="")||($age=="")){
if($nom=="") print("Veuillez saisir votre nom !<br>\n");
if($prenom=="") print("Veuillez saisir votre prénom !<br>\n");
if($age=="") print("Veuillez saisir votre âge !<br>\n");
}
else {
echo "Récapitulatif des informations saisies<br>\n
<ul>
<li>Nom : $nom</li>
<li>Prenom : $prenom</li>
<li>Age : $age</li>
```

```
</ul>
";
}
?>
</body>
</html>
```



16. La base de données MySQL

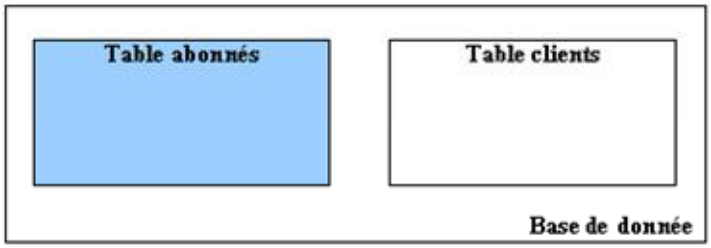
La communication avec les bases de données se fait à l'aide de requêtes SQL, un langage de quatrième génération reconnu par l'ensemble des systèmes de base de données.

16.1. MySQL

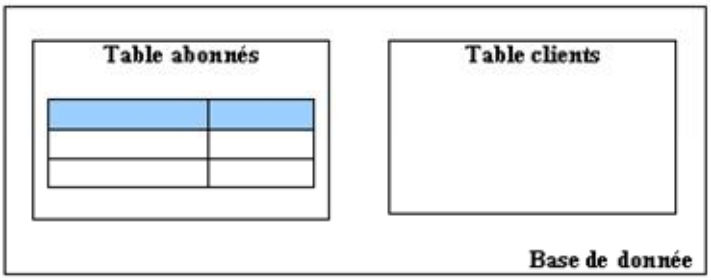
Dans le cas du binôme Apache - PHP, c'est le système de gestion de bases de données MySQL qui est généralement utilisé. MySQL est un système de base de données gratuit et rapide, fonctionnant (entre autres) sous Linux. Etant donné que la majorité des serveurs Web (dont le fameux serveur Apache) fonctionnent sous Linux, MySQL est de ce fait le système de base de données le plus utilisé avec PHP.

16.2. La structure d'une base de données

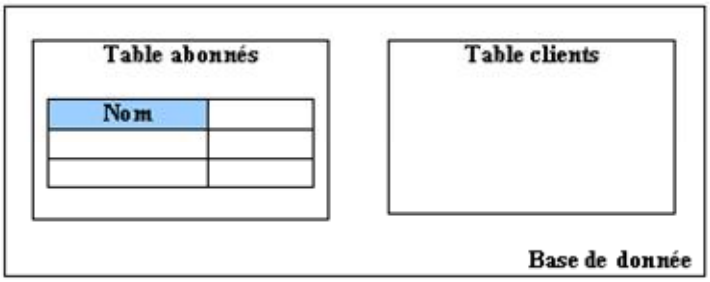
Une base de données contient une ou plusieurs table(s).



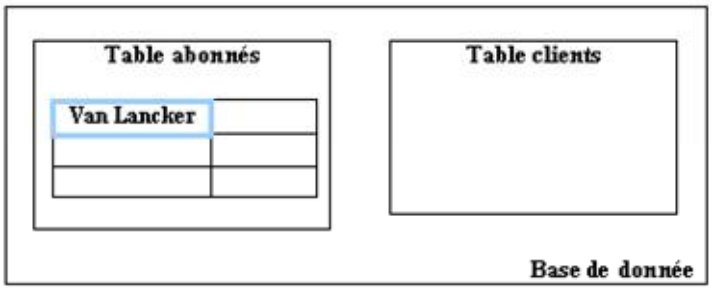
Une table contient plusieurs enregistrements.



Un enregistrement contient un ou plusieurs champs.



Un champ contient des données.



16.3. Les formats des données en MySQL

Les données numériques :

Les données numériques peuvent être signées (signed) ou non signées (unsigned).

Type de champ	Description
TINYINT	Très petit entier. Compris entre -128 et 127 en signed et entre 0 et 256 en unsigned.
SMALLINT	Petit entier. Compris entre -32 768 et 32 0767 en signed et entre 0 et 65535 en unsigned.
MEDIUMINT	Entier moyen. Compris entre -8 388 608 et 8 388 607 en signed et entre 0 et 16 777 215 en unsigned.
INT	Entier. Compris entre 2 147 483 648 et 2 147 483 647 en signed et entre 0 et 4 294 967 295 en unsigned.
BIGINT	Grand entier. Compris entre -9 233 372 036 854 et 9 223 372 036 854 775 807 en signed et entre 0 et 18 446 744 073 709 551 615 en unsigned.
FLOAT	Nombre à virgule flottante en précision simple. L'intervalle de validité va de -

	$3.402823466E^{+38}$ à $-1.175494351E^{-38}$ en signed et entre 0 et de $1.175494351E^{-38}$ à $3.402823466E^{+38}$ en unsigned. .
DOUBLE	Nombre à virgule flottante en précision double. L'intervalle de validité va de $-1.7976931348623157E^{+308}$ à $-2.2250738585072014E^{-308}$ en signed et entre 0 et de $2.2250738585072014E^{-308}$ à $1.7976931348623157E^{+308}$ en unsigned.
DECIMAL	Nombre à virgule flottante signé. Les nombres sont enregistrés sous forme de chaînes de caractères.

Pour plus de précisions, reportez-vous à la documentation de MySQL.

Les chaînes de caractères :

Type de champ	Description
CHAR(x)	Chaîne de caractères de longueur fixe où x (compris entre 1 et 256) est le nombre de caractères.
VARCHAR(x)	Chaîne de caractères de longueur variable où x (compris entre 1 et 256) est le nombre de caractères.
TINYTEXT	Chaîne de 256 caractères maximum.
TEXT	Chaîne de 65 585 caractères maximum.
MEDIUMTEXT	Chaîne de 16 777 215 caractères maximum.
LONGTEXT	Chaîne de 4 294 967 295 caractères maximum.

Pour plus de précisions, reportez-vous à la documentation de MySQL.

16.4. Les opérateurs de MySQL

Les opérateurs arithmétiques :

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

Les opérateurs de comparaison :

Opérateur	Description
=	Egal
!=	Inégal
<=	Inférieur ou égal
<	Inférieur
>=	Supérieur ou égal
<	Supérieur

Les opérateurs logiques :

Opérateur	Description
NOT ou |	NON logique
OR ou ||	OU logique
AND ou &&	ET logique

16.5. PhpMyAdmin

PhpMyAdmin qui se présente comme un site Web (en local), est un ensemble de scripts PHP permettant de gérer aisément et visuellement MySQL sans devoir passer par l'apprentissage du langage SQL.

PhpMyAdmin peut ainsi :

- créer et supprimer des bases de données.
- créer, modifier et supprimer des tables.
- éditer et ajouter des champs.
- insérer des données.
- gérer de multiples utilisateurs avec des permissions différentes.

17. Créer une base de données

17.1. Créer une base de données avec PhpMyAdmin

Pour la suite de l'apprentissage de MySQL, nous aurons besoin d'une base de données que nous allons créer par PhpMyAdmin.

Créons la base de données **base** sur le serveur MySQL.

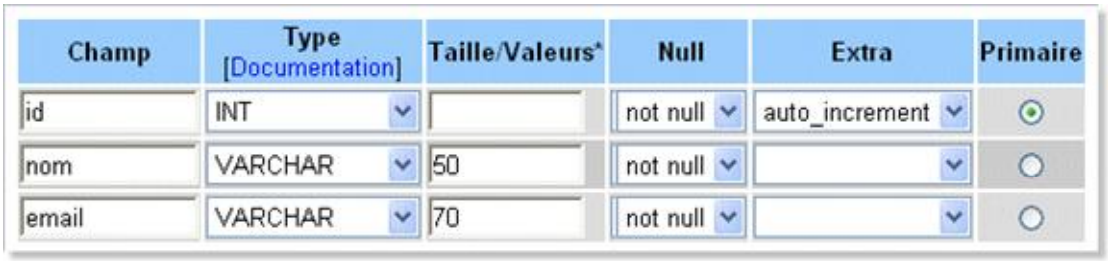
Dans cette base de données, nous allons créer la table **liste**.

La table liste comportera 3 champs :
- un index id, un entier qui servira de clé primaire.
- un champ nom qui pourra contenir une chaîne de 50 caractères.
- un champ email qui pourra contenir une chaîne de 70 caractères.

Parmi les multiples possibilités offertes par PhpMyAdmin, retenons celle qui permet d'encoder les requêtes SQL.

```
CREATE TABLE liste (
id int NOT NULL auto_increment PRIMARY KEY,
nom varchar(50) NOT NULL,
email varchar(70) NOT NULL
)
```

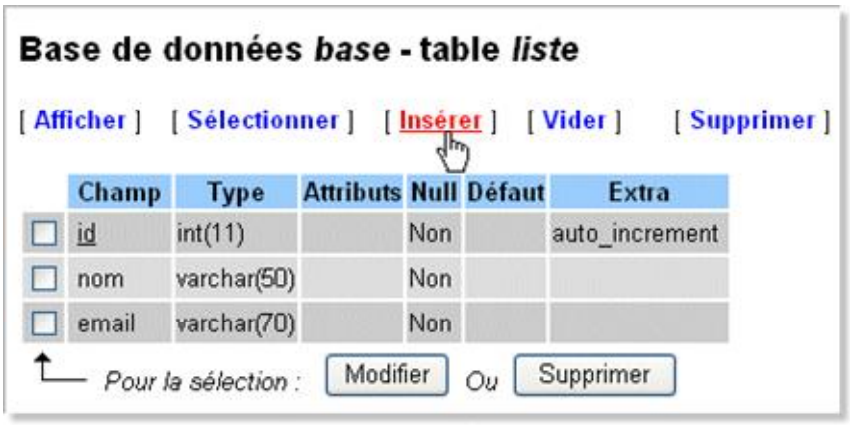
Ou tout simplement :



Insérons un enregistrement de données.

INSERT INTO liste VALUES ('', 'Van Lancker Luc', 'vanlancker.luc@lehtml.com');

Ou tout simplement :



18. Se connecter à la base de données

18.1. La connexion à la base de données depuis PHP

Pour pouvoir vous connecter depuis une page PHP à votre base de données MySQL, il faudra spécifier plusieurs paramètres :

- l'hôte (le serveur sur lequel MySQL est installé).
- le login utilisateur.
- le mot de passe.
- le nom de la base de données.

Par défaut, les paramètres mis en place par EasyPHP sont :

- hôte ou serveur : "localhost".
- username ou login : "root".
- mot de passe : "".

La connexion au serveur MySQL s'effectue par la fonction msql_connect(). Sa syntaxe est :

```
msql_connect ( 'hôte', 'login', 'mot de passe' )
```

La fonction retourne TRUE si la connexion est réussie et FALSE sinon.

Il est possible d'interrompre le processus afin d'éviter les erreurs en cascade. Deux méthodes permettent d'effectuer cette opération :

- Le stockage du résultat de l'exécution de la fonction dans une variable. Par exemple :

```
$connect = mysql_connect('localhost','root','');
```

- L'utilisation de la fonction die() en cas d'erreur d'exécution. Si la fonction retourne la valeur 0 (c'est-à-dire s'il y a une erreur) la fonction die() [traduisez meurt] renvoie un message d'erreur. La fonction die() est équivalente à la fonction exit().

  Par exemple :

  ```
  $connect = mysql_connect('localhost', 'root', '') or die("Erreur de connexion au serveur.");
  ```

La connexion à la base de données s'effectue par msql_select_db (nom de la base, identifieur de connexion). La fonction retourne aussi TRUE en cas de succès et FALSE en cas d'erreur.

```
mysql_select_db('base", $connect) or die("Erreur de connexion à la base");
```

Ainsi tous vos scripts utilisant MySQL commenceront par :

```
$connect = mysql_connect('localhost','root','') or die ("erreur de connexion");
mysql_select_db('base',$connect) or die ("erreur de connexion base");
```

Comme tous vos scripts commenceront par les mêmes lignes, il serait assez élégant de les inclure dans une librairie qu'il suffira d'appeler à chaque script.

Voici le script de librairie externe connexion.php :

```
<?php
$connect = mysql_connect('localhost','root','') or die ("erreur de connexion");
mysql_select_db('base',$connect) or die ("erreur de connexion base");
?>
```

Pour appeler cette librairie depuis le script, on ajoutera :

```
include("connexion.php")
```

19. Afficher une table

19.1. Afficher le contenu d'une table

Pour afficher le contenu d'une table, après s'être connecté à la base, il faudra d'abord sélectionner la table liste.

Lorsque l'on effectue une requête de sélection à l'aide de la fonction **mysql_query**, il est essentiel de stocker le résultat de la requête (les enregistrements de la table) dans une variable, que l'on peut nommer $result.

```
$result = mysql_query("SELECT id,nom,email from liste");
```

Toutefois, cette variable contient l'ensemble des enregistrements demandés et n'est donc pas exploitable telle quelle.

Ainsi on utilise une autre fonction, la fonction **mysql_fetch_array()**, qui découpe les lignes de résultat et les affecte à une variable de type tableau associatif dans l'ordre où elles arrivent.

L'affichage s'effectuera par une boucle qui va parcourir les éléments du tableau.

```
while ( $row = mysql_fetch_array($result)){
echo $row[id].' - '.$row[nom].' - '.$row[email].'<br>';
}
```

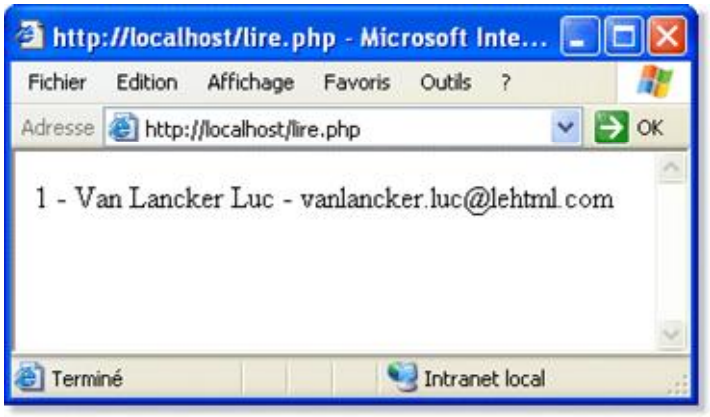
Une fois le script terminé, il est conseillé de clore la connexion.

```
mysql_close();
```

Le script complet devient :

```
<?php
$connect = mysql_connect('localhost','root','') or die ("erreur de connexion");
mysql_select_db('base',$connect) or die ("erreur de connexion base");
$result = mysql_query("SELECT id,nom,email from liste");
while ( $row = mysql_fetch_array($result)){
echo $row[id].' - '.$row[nom].' - '.$row[email].'<br>';
}
mysql_close();
?>
```

On peut ainsi afficher notre table liste qui ne contiendra à ce stade qu'un seul enregistrement.



20. Insérer des données

20.1. Insérer des données à partir d'un formulaire

Dans un site dynamique, il est intéressant de prévoir la possibilité d'insérer des données en ligne.

Soit un formulaire d'inscription :

```
<html>
<head>
<title>formulaire</title>
<head>
<body>
Pour vous inscrire :<br>
<form method="post" action="insert.php">
```

```
Nom : <input type="text" name="nom"><br>
Email : <input tupe="text" name="email"><br>
<input type="submit" name="submit" value="Insérer">
</form>
</body>
</html>
```

La page d'insertion insert.php :

```
<?php
$connect = mysql_connect('localhost','root','') or die ("erreur de connexion");
mysql_select_db('base',$connect) or die ("erreur de connexion base");
mysql_query("INSERT INTO liste VALUES ('','$nom','$email') ");
mysql_close();
?>
```

Un petit détour par PhpMyAdmin vous confirmera que l'enregistrement a bien été effectué.

21. Gérer une base de données

21.1. Modifier un enregistrement

La fonction update vous permet de modifier les enregistrements d'un ou plusieurs champs dans votre table.

Le formulaire de modification :

```
<html>
<head>
<title>formulaire</title>
<head>
<body>
Pour modifier votre mail :<br>
<form method="post" action="modif.php">
Nom : <input type="text" name="nom"><br>
Email : <input tupe="text" name="email"><br>
<input type="submit" name="submit" value="Modifier">
</form>
</body>
</html>
```

Le fichier modif.php :

```
<?php
$connect = mysql_connect('localhost','root','') or die ("erreur de connexion");
mysql_select_db('base',$connect) or die ("erreur de connexion base");
mysql_query("UPDATE liste SET email=$email WHERE nom==$nom");
mysql_close();
?>
```

Un petit détour par PhpMyAdmin vous confirmera que l'enregistrement a bien été modifié.

22.2. Supprimer un enregistrement

Supprimer 1 seul enregistrement, on choisit l'id numéro 1

```
<?php
$connect = mysql_connect('localhost','root','') or die ("erreur de connexion");
mysql_select_db('base',$connect) or die ("erreur de connexion base");
mysql_query("DELETE FROM liste WHERE nom='Emile' ");
mysql_close();
?>
```

La clause where vous permet de choisir l'enregistrement que vous souhaitez supprimer, si vous n'utilisez pas cette clause, la table sera complètement vidée de son contenu. L'instruction delete est donc à manipuler avec attention !

Vous pouvez utilisez plusieurs instructions avec la clause where, par exemple.

```
$req = MySQL_query("DELETE FORM liste WHERE nom =$nom AND email=$email ");
```

22.3. Trier une table

Nous allons trier la table par ordre ascendant sur les noms des utilisateurs. Le résultat sera affiché dans un tableau Html.

ORDER BY vous permet de choisir un ordre pour l"affichage, en précisant ASC vous aurez une sortie en ordre croissant, et avec DESC, la sortie sera en ordre décroissant.

```
<html>
<head>
<title>Trier</title>
<head>
<body>
<table border="1" cellpadding="0" cellspacing="0">
<tr>
<th>Nom</th>
<th>Email</th>
</tr>
<?php
// Connexion au serveur
$connect = mysql_connect('localhost','root','') or die ("erreur de connexion");
mysql_select_db('base',$connect) or die ("erreur de connexion base");
// Création et envoi de la requête
$result = mysql_query("SELECT nom,url FROM sites ORDER ASC BY nom");
// Récupération des résultats
while($row = mysql_fetch_row($result)){
$Nom = $row[1];
$Email = $row[2];
echo "<tr>\n
<td>$Nom</td>\n
<td>$Email</td>\n
</tr>\n";
}
// Déconnexion de la base de données
mysql_close();
?>
</tr>
```

```
</table>
</body>
</html>
```

Dans l'exemple ci-dessus, on utilise la fonction mysql_fetch_row() qui est assez semblable à la fonction mysql_fetch_array() mais cette fonction stocke les enregistrements non dans un tableau associatif mais dans un tableau indexé normal.

22. Conclusion

Nous ne pouvons que reprendre ce qui avait été signalé en introduction. Ce tutorial a délibérément été conçu comme une présentation élémentaire des fonctionnalités des scripts PHP et de sa possibilité de connexion avec une base de données. La mise en œuvre d'un site dynamique nécessitera à n'en pas douter d'éléments supplémentaires que vous trouverez en librairie ou sur le Web.

Sources principales

A brief PHP3 tutorial (en français) : www.linux-france.org/article/devl/php3/tut/php3_tut.html
PHP Facile : www.toutestfacile.com/
PHP Débutant : www.phpdebutant.com/
Le tutorial d'ASP-PHP.net : www.asp-php.net/tutorial/
Comment ça marche - Introduction à PHP : www.commentcamarche.net/php/phpintro.php3
PHP France : http://www.phpfrance.com/tutorials/
PHP Tutorial : www.freewebmasterhelp.com/tutorials/php/
An Introduction to PHP : www.rci.rutgers.edu/~jfulton/php1/
Manual de PHP - Tutorial de PHP : www.webestilo.com/php/

Van Lancker Luc

vanlancker.luc@lehtml.com
www.lehtml.com/php/

Table des matières