PHP Introduction
Cours

Michel Cabaré – Laurent Lallias

Novembre 2001 version 2.0

TABLE DES MATIERES

Ce qu'est php

PHP (officiellement "**PHP: Hypertext Preprocessor**") est un langage de script HTML, qui fonctionne coté serveur.

Il est à noter la différence avec les autres scripts CGI écrits dans d'autres langages tels que le Perl ou le C : Au lieu d'écrire un programme avec de nombreuses lignes de commandes afin d'afficher une page HTML, vous écrivez une page HTML avec du code inclus à l'intérieur afin de réaliser une action précise

Ce qui distingue le PHP des langages de script comme le Javascript est que le code est exécuté sur le serveur. Si vous avez un script similaire sur votre serveur, le client ne reçoit que le résultat du script, sans aucun moyen d'avoir accès au code qui a produit ce résultat.

Le langage PHP possède les même fonctionnalités que les autres langages permettant d'écrire des scripts CGI, comme collecter des données, générer dynamiquement des pages web ou bien envoyer et recevoir des cookies. La plus grande qualité et le plus important avantage du langage PHP est le support d'un grand nombre de bases de données.

Le langage PHP a été conçu durant l'automne 1994 par Rasmus Lerdof. L'analyseur fut réécrit durant l'été 1995 et fut appelé PHP/FI Version 2. FI étaient les initiales d'un autre package que Rasmus avait écrit qui interprétait les formulaires HTML. L'été 1997 voit aussi un profond changement dans le développement du PHP: d'un projet personnel (celui de Ramsus),* on passe alors à un projet d'équipe. L'analyseur/parseur fut de nouveau réécrit par Zeev Suraskyi et Andi Gutmans et ce nouvel analyseur forma la base de la version 3 du PHP. Aujourd'hui PHP est distribué avec de nombreux produits comme et "RedHat Linux" et il est admis (d'après les chiffres de NetCraft, et leurs statistiques Netcraft Web Server Survey) que le PHP est utilisé sur 200 000 sites web dans le monde entier...

Un interpréteur Php local

EasyPHP installe et configure automatiquement un environnement de travail permettant de mettre en oeuvre toute la puissance et la souplesse qu'offrent le langage dynamique PHP et son support efficace des bases de données. EasyP regroupe un serveur Apache, une base de donnée MySQL, les versions 2,3 et 4 de PHP ainsi que des outils facilitant le développement

sur **http://easyphp.manucorp.com/**

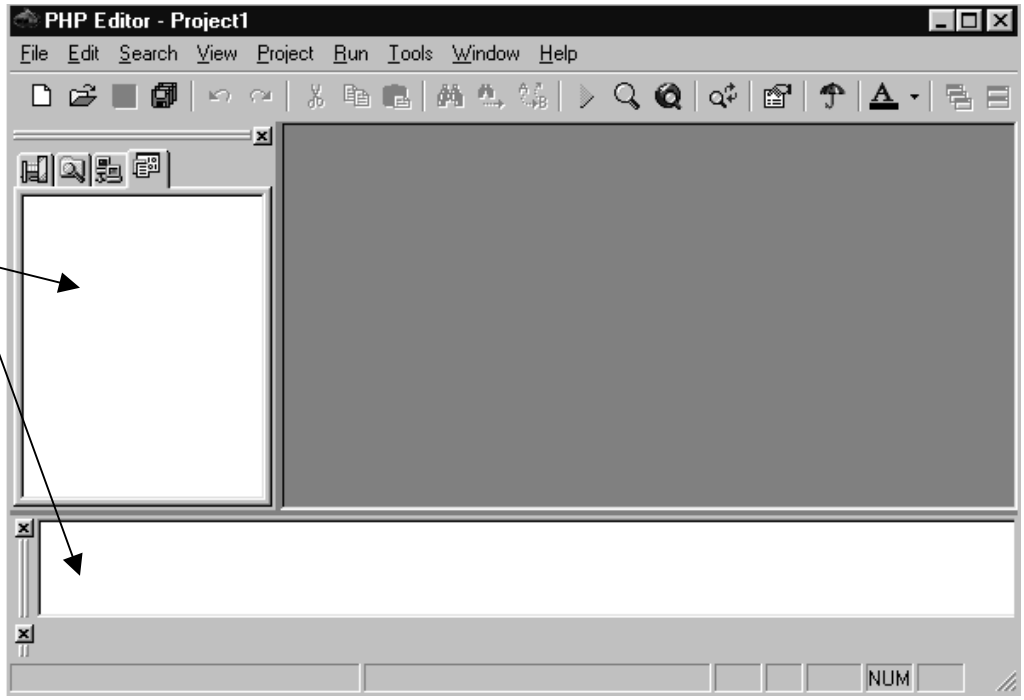
Télécharger le logiciel

Un éditeur Php

N'importe quel éditeur texte peut suffire, comme **NotePad** ou **Wordpad**, même si un éditeur permettant un repérage des n° de ligne et une coloration syntaxique appropriée est le bienvenu...

Un choix partisan a été fait sur **http://www.soysal.com/PHPEd/**

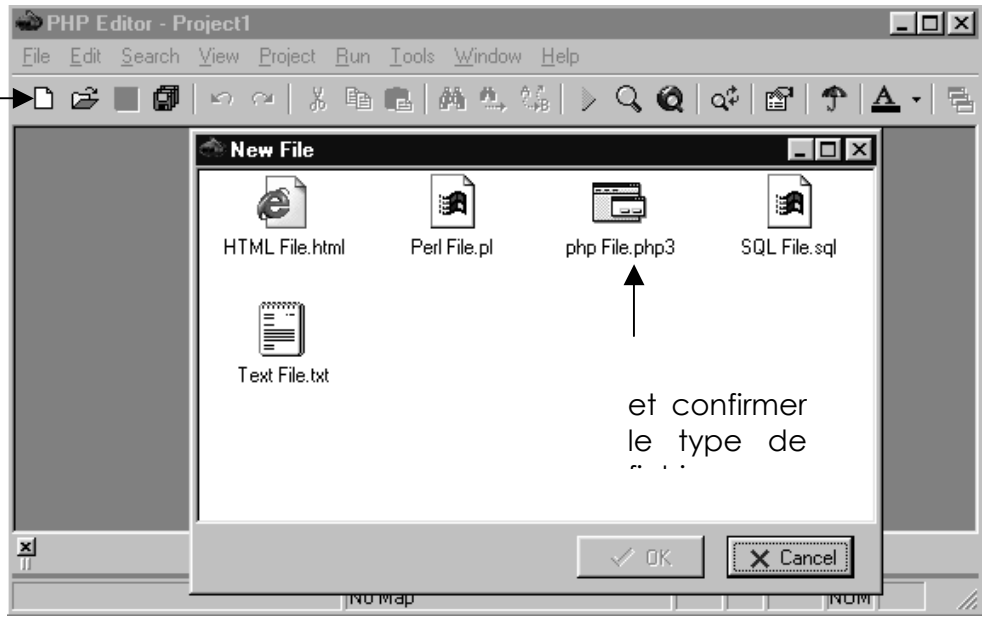
Après téléchargement et installation, le lancement donne



Simplifions l'éditeur par la suppression des fenêtres via **View Manager** et **View Debug**

Ce qui nous intéresse ici c'est un fonctionnement au plus simple, avec numérotation des lignes, coloration syntaxique et des fonctions classiques d'édition de texte...

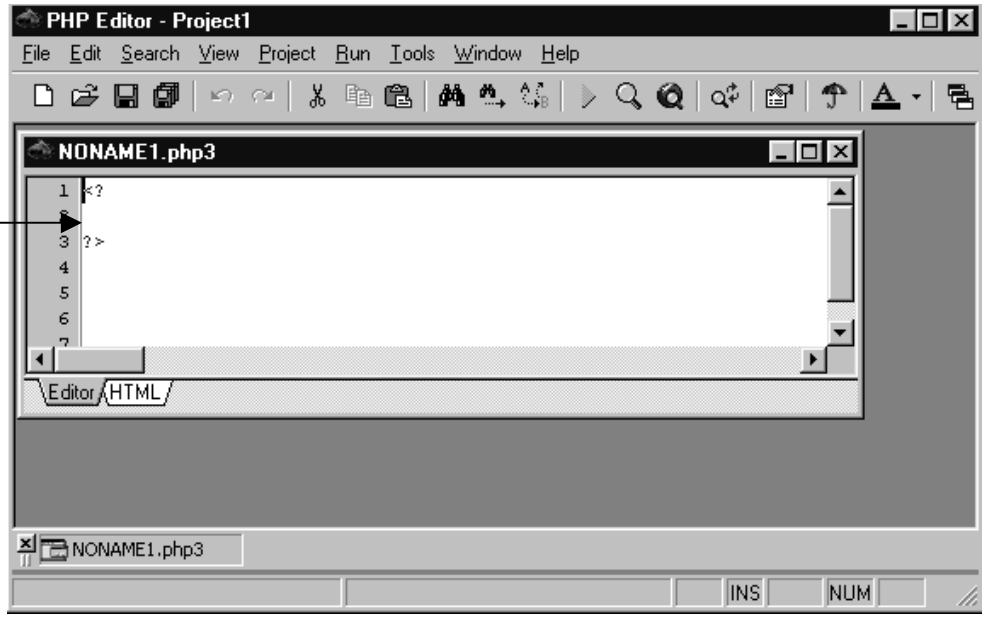
Pour créer un nouveau fichier PHP Il suffit de demander **New**



et confirmer le type de

On obtient alors normalement

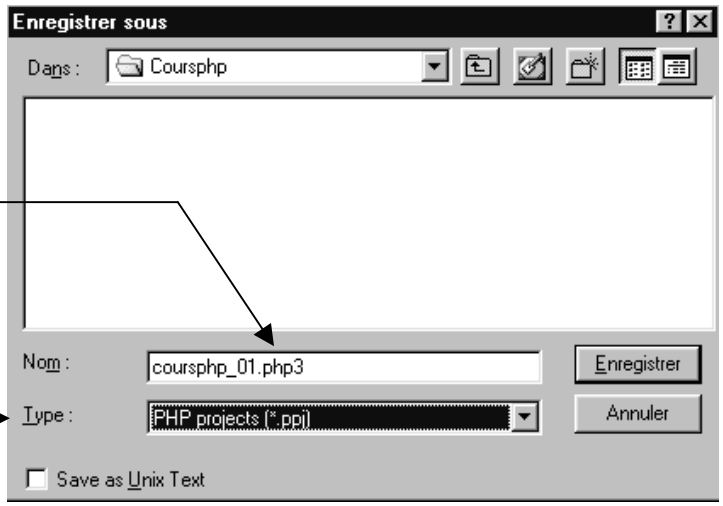
avec déjà des Balises encadrant un script Php



Lorsque l'on fera un Save As on aura alors

l'extension .php3 est mise d'office,

on ne gère pas la notion de Projet



Un interpréteur Php sur un serveur Web

Il faut trouver un hébergeur mettant à disposition un interpréteur PhP, le choix ne manque pas !

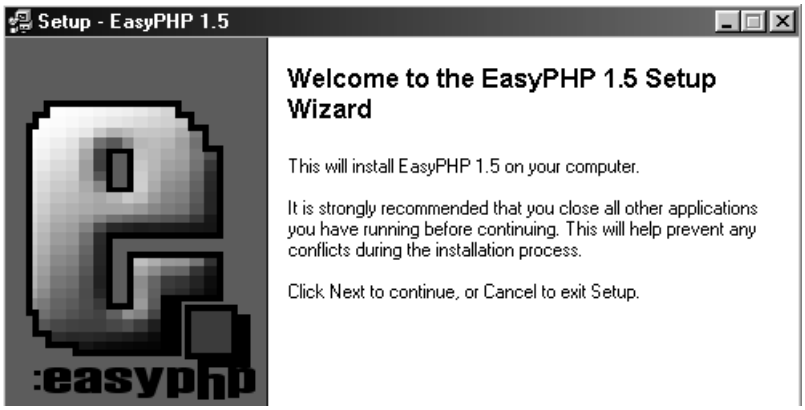
Pour nous, les tests en ligne avec un interpréteur local suffiront.

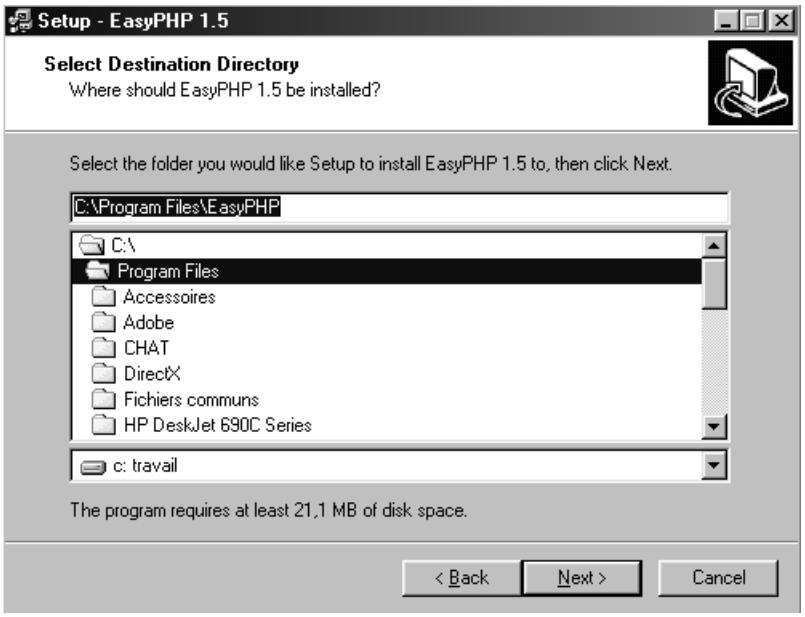
Installer Easy PHP

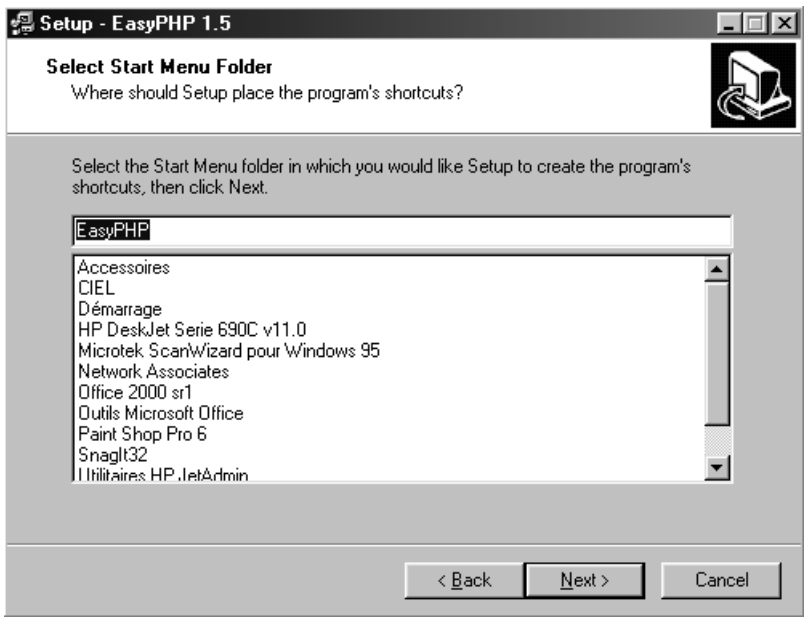
Télécharger le logiciel sur http://quickphp.online.fr ou www.easyphp.org

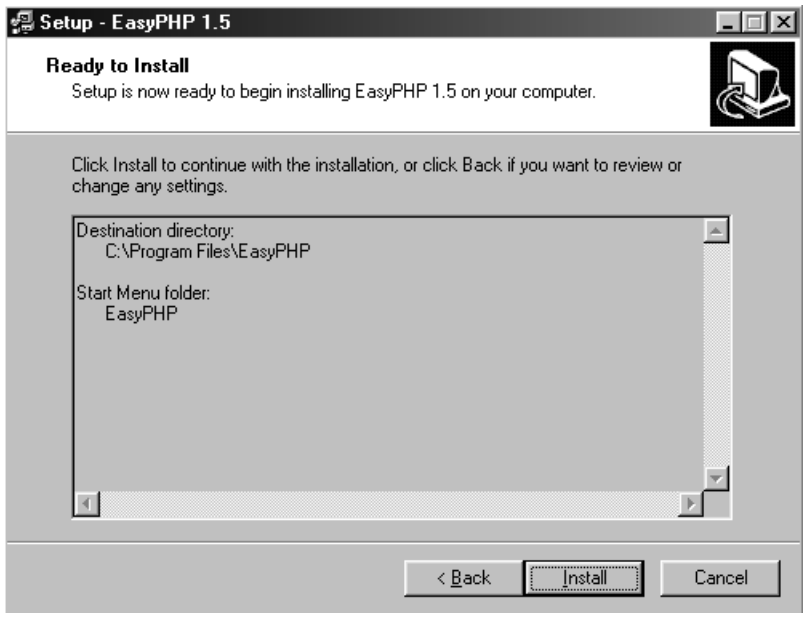
(QuickPHP, EasyPHP et PHPDev ont fusionné. Pour l'occasion un site entièrement dédié à EasyPHP a été créé : www.easyphp.org)

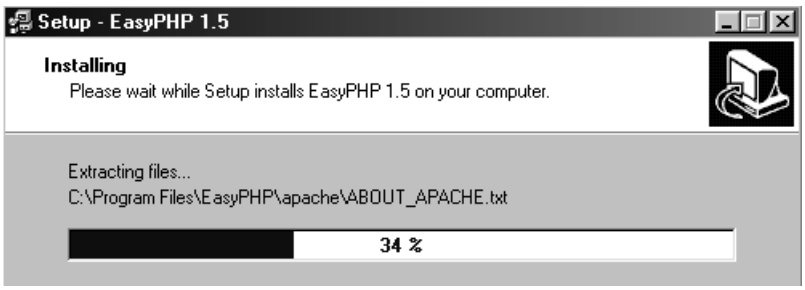
Le guide d'installation défini ci dessous correspond à la version QuickPHP 1.0 donnée en cours. L'installation de Easy PHP est en tous points semblable.

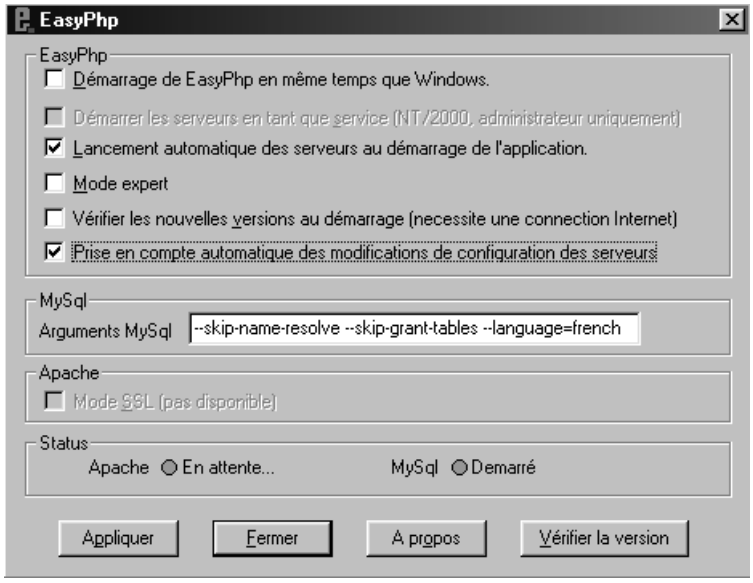


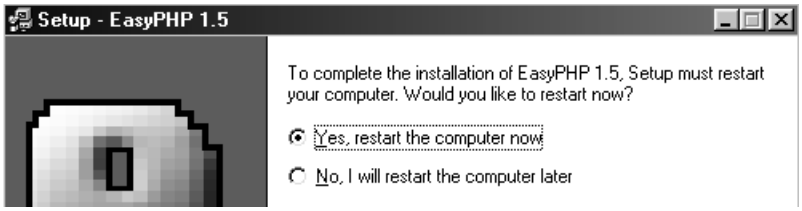



Setup - EasyPHP 1.5
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Select the Start Menu folder in which you would like Setup to create the program's shortcuts, then click Next.
EasyPHP
Accessoires
CIEL
Démarrage
HP DeskJet Serie 690C v11.0
Microtek ScanWizard pour Windows 95
Network Associates
Office 2000 sr1
Outils Microsoft Office
Paint Shop Pro 6
SnagIt32
Utilitaires HP JetAdmin
< Back
Next >
Cancel


Setup - EasyPHP 1.5
Ready to Install
Setup is now ready to begin installing EasyPHP 1.5 on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination directory:
C:\Program Files\EasyPHP
Start Menu folder:
EasyPHP
< Back
Install
Cancel


Setup - EasyPHP 1.5
Installing
Please wait while Setup installs EasyPHP 1.5 on your computer.
Extracting files...
C:\Program Files\EasyPHP\apache\ABOUT_APACHE.txt
34 %





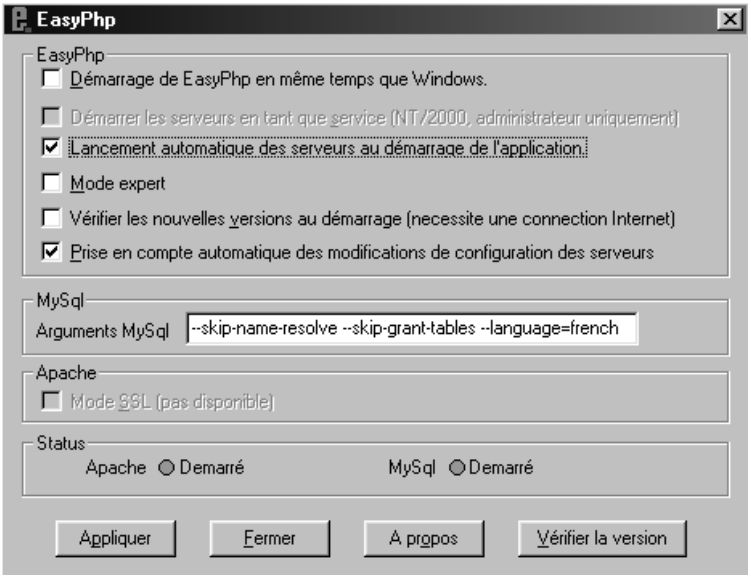
Lancer Easy PHP

Un certain nombre d'entrées apparaissent dans le menu Démarrer

et au lancement de Easyphp, une icône dans la barre des tâches

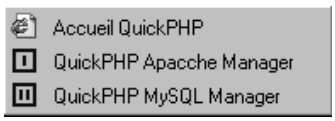
un clic bouton droit sur cette icône permet de démarrer/arrêter Easyphp…

un double clic sur cette icône permet d'aacéder à la configuration de Easyphp…

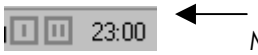


Lancer Quick PHP

Un certain nombre d'entrées apparaissent dans le menu Démarrer



Ainsi que 2 icônes dans la barre des tâches, sur la droite

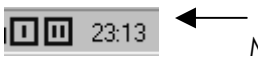


Managers lancés, Serveur Apache Stoppé, SQL Stoppé

Ces icônes correspondent au lancement automatique (via le menu **Démarrer / Programme /Démarrage**) des managers d'un **serveur Web Apache** et d'un **interpréteur PHP-MYSQL**

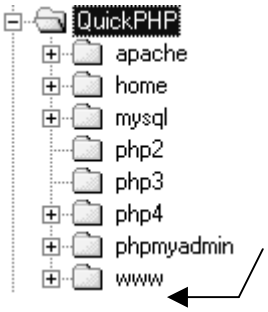
On ne peut pas à proprement parler du lancement de QuickPHP, il s'agit plutôt de la mise en route du serveur Apache et de MySQL

- Pour démarrer Apache, faites un clic droit sur l'icône et cliquez sur "Start Apache"; l'icône devient alors bleue. (ou double clic dessus)
- Pour démarrer MySQL, faites un clic droit sur l'icône et cliquez sur "Start MySQL" ; l'icône devient alors bleue. (un double clic – marche aussi)



Managers lancés, Serveur Apache lancé, SQL lancé

Structure du dossier EasyPHP



Pour que vos pages PHP soient interprétées, il est impératif de placer vos fichiers dans le répertoire **www**.

Le serveur Apache est configuré pour ouvrir automatiquement tout fichier nommé **index.html** ou **index.php3**.

Dans le répertoire www de EasyPHP, est placé un fichier **index** qui s'ouvre automatiquement lorsque vous saisissez l'adresse "**localhost**" (si le serveur Apache est en route !).

Cette page sert de page d'accueil à vos répertoires de travail. Il est conseillé de créer un répertoire par projet afin d'avoir une vision plus claire des développements.

Le répertoire **www** sera donc structuré de la façon suivante :

- à la racine, la page **index.php** d'accueil et éventuellement les images qui l'accompagnent (à ne pas effacer - sauf utilisateur averti)
- puis tous les répertoires de travail

  correspondant à vos différents projets (un répertoire vide nommé "projet1", que vous pourrez renommer ou effacer, sert d'exemple).

Tester le serveur Apache

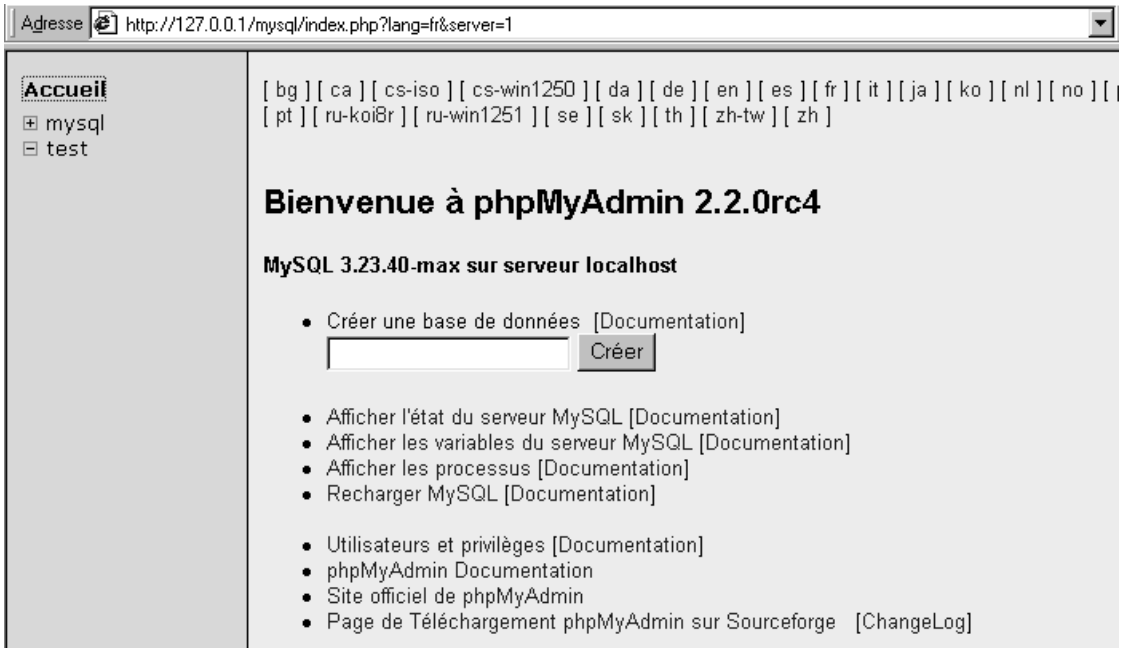
Il suffit, le serveur étant démarré, de taper **http://localhost** (ou Web local depuis l'icône de Easyphp)

Tester MySQL

Le serveur Apache étant fonctionnel il suffit de taper **http://localhost/mysql/**

Ou bien demander PhpMyAdmin sur localhost…

pour obtenir



Script Php autonome :

Si votre code se trouve dans un fichier autonome, en PHP il faudra enregistrer ce fichier en suivant les paramètres suivants :

- s'il s'agit de code PHP 2, spécifiez une extension **.php2** : nomdufichier.php2
- s'il s'agit de code PHP 3, spécifiez une extension **.php3** : nomdufichier.php3... etc.

Ceci n'est pas une règle absolue, mais correspond à la configuration de QuickPHP. Il vous sera peut être nécessaire, chez un hébergeur de modifier ces extensions

Un script php commence par un balise d'ouverture **< ?** et se termine par une balise de fermeture ?>:

<? *xxxxscriptxxxx* **?>** **<?php** *xxxxscriptxxxx* **?>**

La commande echo

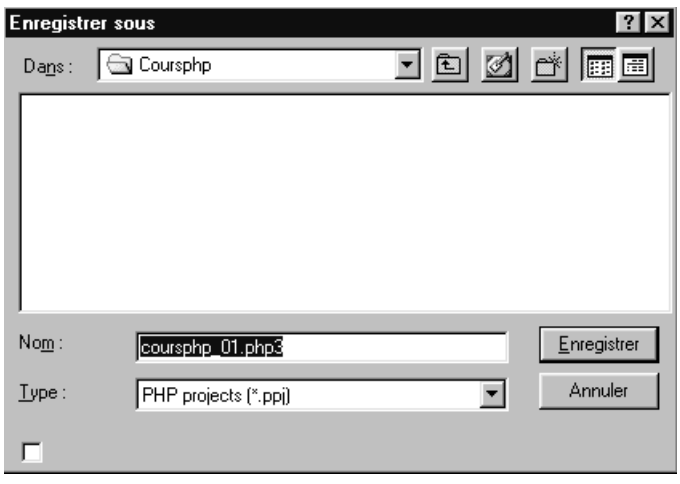
La commande echo en php permet d'afficher la chaîne de caractères située entre les guillemets.

On veut écrire le script suivant

```
<?
    echo "mon premier script php";
?>
```

On demande de créer un nouveau document de type Php3 et on l'enregistre de manière à pouvoir le tester avec notre interpréteur.

Donc

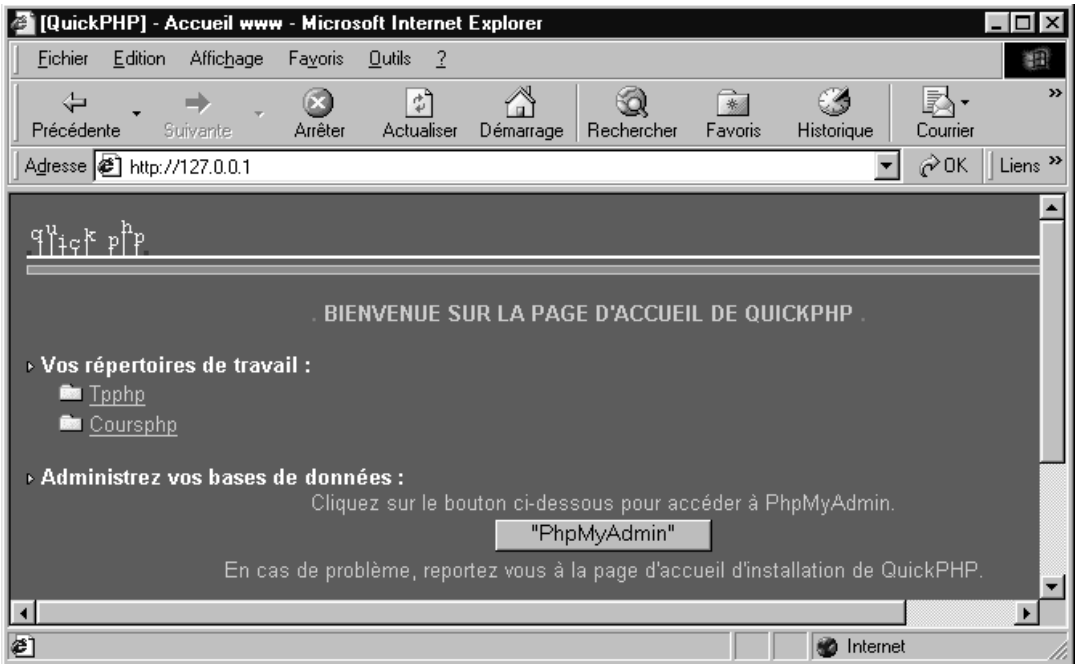


fichier **coursphp_01.php3** enregistré en

Test du fichier PhP

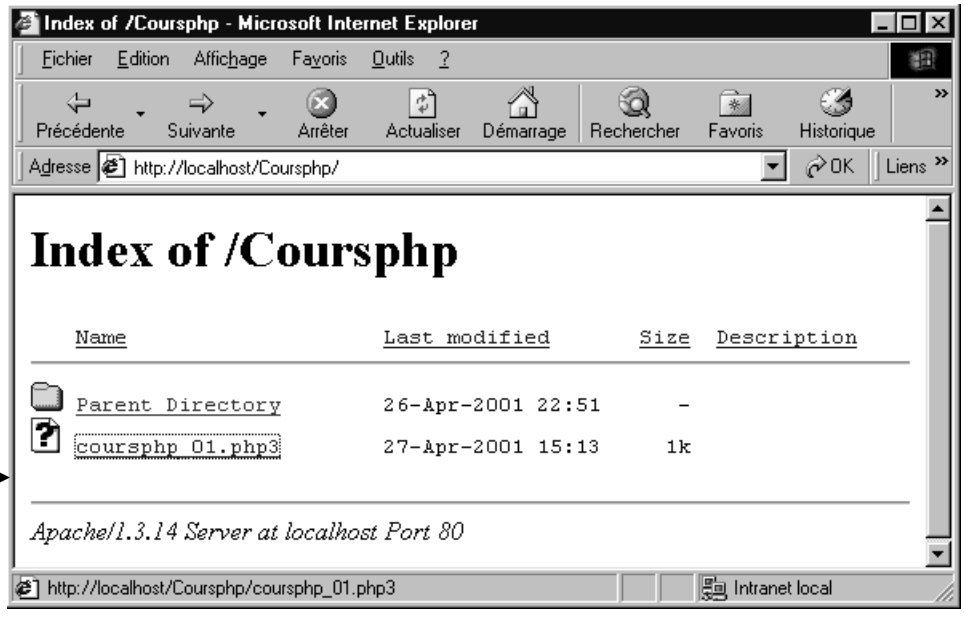
Pour tester ce fichier correctement, il faut lancer **http://localhost** ou **http://127.0.0.1**

Aller dans notre dossier **Coursphp**

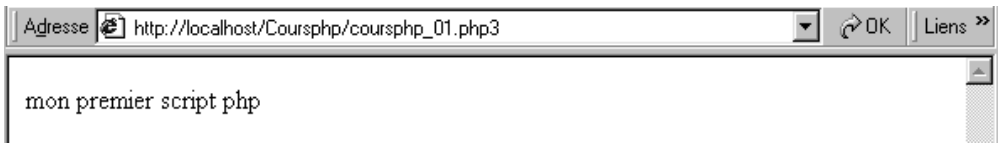


On obtient alors

et cliquant sur notre fichier **Coursphp_01.php3**



On obtient alors



Le but de Php est de permettre la création de pages web dynamiques, et donc de pouvoir envoyer des données au navigateur.

Php3 fournit 3 fonctions permettant d'envoyer du texte au navigateur :

- echo
- print
- printf

La fonction echo

La fonction **echo** permet d'envoyer au navigateur une chaîne de caractères délimitée par des guillemets.

Syntaxe : echo Expression;

L'expression peut être une chaîne de caractères ou une expression que le navigateur évalue (code html par exemple)

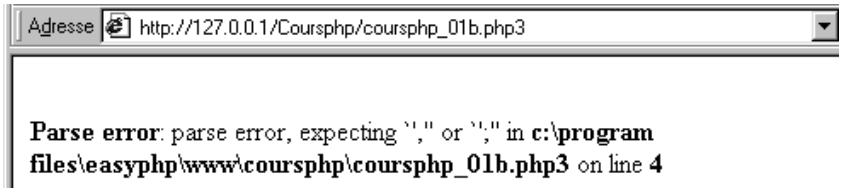
```
echo "Chaine de caracteres";
echo (1+2)*87;
echo "<H1>Salut</H1>";
```

N.B: une erreur en php est "sanctionnée" lors de l'interprétation , ainsi l'écriture

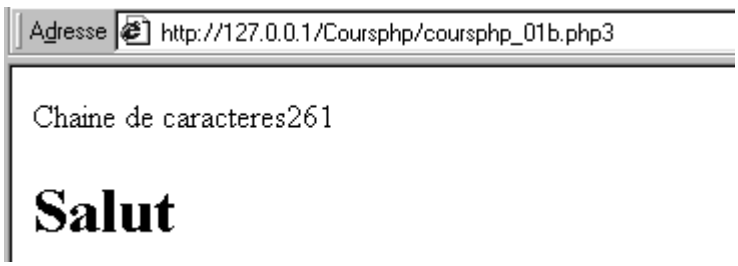
```
echo <H1>Salut</H1>;
```

en oubliant les guillements

donnerait



au lieu de



La fonction print

La fonction print est similaire à la fonction echo à la différence près que l'expression à afficher est entre parenthèses.

Syntaxe : print(expression);

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

```
print("Chaine de caracteres");
print ((1+2)*87);
print ("<H1>Salut</H1>");
```

La fonction printf

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

Syntaxe : printf (chaîne formatée);

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:

```
printf ("Le périmètre du cercle est %d",2 * 3.1415927*24.546);
```

Script Php et balises html

Un interpréteur PHP comprend les balises HTML qu'il "saute", pour ne travailler que sur les zones qu'il reconnaît spécifiquement comme sienne grâce aux balises <? et ?> ...

Autrement dit l'interpréteur PHP ne travaille que sur ses zones spécifiquement délimitées, et considère tout le reste comme des balises à envoyer telles quelles au navigateur pour que celui-ci les traite !

Le travail d'un interpréteur PHP consiste d'ailleurs à traduire en langage HTML le résultat d'un traitement (qu'on lui demande d'effectuer en langage PHP)

Ainsi notre fichier de tout à l'heure peut s'étoffer pour être plus "rigoureux".

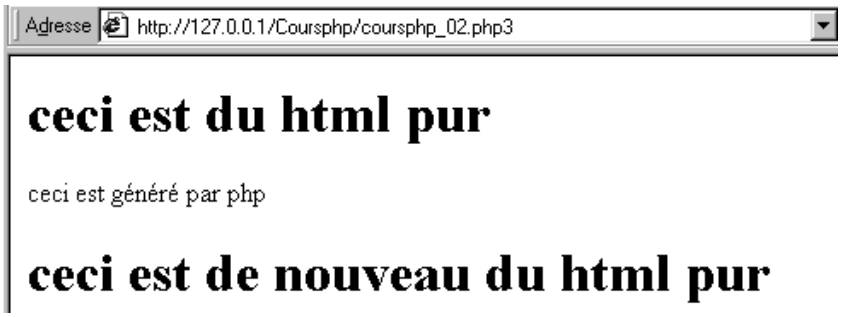
par exemple

ce fichier **Coursphp_02.php3**

```
<html>
<body>
<h1> ceci est du html pur </h1>
<?php
  echo "ceci est généré par php";
?>
<h1> ceci est de nouveau du html pur </h1>
</body>
</html>
```

zone "php"

génère alors



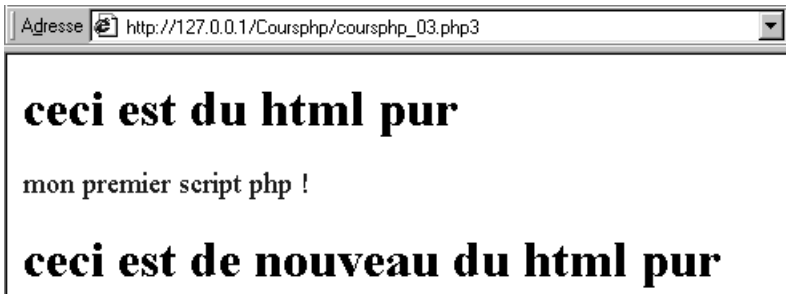
Mais de l'HTML peut être contenu comme argument et "transmis tel quel" s'il est inclus dans une instruction de type print ou echo, permettant une mise en forme (par exemple)

ainsi

fichier
Coursphp_03.php3

```
<html>
<body>
<h1> ceci est du html pur </h1>
<?php
 echo "<font size=\"4\" color=\"blue\"> mon premier script php !</font>";
?>
<h1> ceci est de nouveau du html pur </h1>
</body>
</html>
```

génère alors



Rq : une des solutions pour écrire sur 2 lignes consiste donc à utiliser la balise
 du html.

fichier
Coursphp_03b.php3

```
<html>
<body>
<h1> ceci est du html pur </h1>
<?php
 echo "<font size=\"4\" color=\"blue\"> mon premier script php !</font>";
 echo "<br>";
 echo "ceci est une deuxième ligne de texte";
?>
<h1> ceci est de nouveau du html pur </h1>
</body>
</html>
```

REGLES D'ECRITURE

Les règles de bases :

section php

Un script php commence par une balise d'ouverture **<** et se termine par une balise de fermeture **>** :

<? *xxxx script xxxx* **?>** est le plus souvent employé

<?php *xxxxscriptxxxx* **?>** est possible

pour être parfaitement en accord avec la syntaxe HTML, à l'intérieur de laquelle notre script PHP peut se trouver, on peut aussi écrire

```
<script language="php">
 xxxx script xxxx
</script>
```

ce qui utilisent javascript apprécieront...

N.B: cette notation est la seule "officielle" et respectée par tous les éditeurs HTML... (sachez que certains éditeurs n'hésitent pas à remplacer les balises qu'ils ne connaissent pas...)

Instructions - casse

- Le séparateur d'instructions est le **;** (point virgule) absolument obligatoire !
- Le langage est "casse-sensitive", autrement dit la distinction entre les minuscules-majuscules a son importance ! Ce qui permet de dire qu'il est différent de parler de la variable **toto** et de la variable **TOTO**, comme de **Toto**...
  bizarrement cela ne joue pas pour les fonctions ...
- Les commentaires peuvent se faire via deux techniques

  /* zone en commentaire ***/**

  ou alors pour invalider une fin de ligne

  // zone en commentaire
  # zone en commentaire

Déclaration de variables :

Le nom des variables doit :

- commencer par le symbole **$**
- comporter que des lettres **A-Z**, **a-z** et les chiffres de **0 à 9** l'underscore **_** les caractères ASCII de 127 à 255 (la longueur n'est pas limitée)
- ne pas contenir de caractère **espace** ou autres "hors liste"
- ne pas commencer par un chiffre

La déclaration est optionnelle, car php décide lui-même du type de la variable lors de sa première affectation entre string (chaîne) integer et double (numérique)

En PHP 3, les variables sont toujours assignées par valeur. le PHP 4 permet aussi d'assigner les valeurs aux variables par référence. Cela signifie que la nouvelle variable ne fait que référencer (en d'autres termes, "devient un alias de", ou encore "pointe sur") la variable originale.

Type "alphanumerique"

Leur valeur peut contenir des lettres ou des symboles, il suffit d'entourer cette valeur par des **"** (guillemets doubles) ou des **'** (guillemets simples)

Ainsi dans l'écriture

```
$nom = "Dupont"      ou      $nom='Dupont'
```

on déclare une variable $nom avec la valeur "Dupont" stockée dedans

Type "numérique"

Une variable numérique peut contenir des valeurs entières (sans aucune décimale), ou une valeur réelle, comportant toujours une virgule même si la décimale est à zéro

Ainsi dans l'écriture

```
$quantite = 6
```

on déclare une variable $quantite avec la valeur 6 stockée dedans

si on écrit

fichier **coursphp_04.php3**

```
echo $quantite
```

on obtiendra alors à l'écran la valeur de la variable $quantite, soit 6

LES OPERATEURS DE BASE

Concaténation . :

L'opérateur de concaténation **.** est utilisé pour des chaînes, des variables

```
$b = "Bonjour ";
$b = $b."vous";
```

Arithmétiques :

n'ont de sens que s'ils sont utilisés sur des variables de type Numérique

- l'addition (**+**)
- la multiplication (*****)
- la soustraction(**-**)
- la division (**/**)
- le modulo (**%**) : reste de la division entière d'un entier par un autre

Affectation :

L'opérateur d'affectation le plus simple est le signe "=".

Le premier réflexe est de penser que ce signe veut dire "égal à". Ce n'est pas le cas. Il signifie que l'opérande de gauche se voit affecter la valeur de l'expression qui est à droite du signe égal.

```
$a = 3;
$a = $a + 5;
$b = "bonjour ";
echo $a;
echo $b;
```

"certains fanatiques" peuvent utiliser les conventions suivantes

Opération	Notation abrégée L'opérateur est mis devant le signe =
x = x + y	x += y
x = x - y	x -= y
x = x * y	x *= y
x = x / y	x /= y
x = x & 1	x &=b
x = x | b	x |= b
x = x ^ b	x ^= b
x = x << b	x <<= b
x = x >> b	x >>= b
x = x +1	++x ; x++
x = x - 1	--x ; x--

comme dans

fichier
Coursphp_04b.php3

```
$a=3;
$a += 5; // affecte la valeur 8 à la variable $a. (correspond à'$a = $a + 5');
$b = "bonjour ";
$b .= "ici!"; // affecte la valeur "Bonjour ici!" à la variable $b
```

Le placement des opérateurs **avant** la variable (++a, --a) réalise une affectation antérieure à l'opération en cours. Le placement **après** réalise l'affectation après l'opération en cours.

LA COMMANDE ECHO ET LES VARIABLES

Règles d'écriture

si on écrit

```
echo $nom
```

on obtiendra alors à l'écran la valeur de la variable $nom, soit Dupond

On peut facilement utiliser la valeur d'une variable, ainsi dans les écritures suivantes :

$nom= "Dupond ";

```
$prenom="Jean";
$patronime="$nom $prenom";
echo $patronime
```

on utilise les valeurs de $nom et de $prenom...

```
echo "mon nom est $nom $prenom";
```

dans la chaîne "mon nom..."on utilise encore les valeurs de $nom et de $prenom...

N.B : Attention aux nuances :

- echo avec " " interprète les éventuelles variables contenues,
- echo avec ' ' affiche le contenu littéral

ainsi dans :

```
echo "mon nom est $nom $prenom";
```

on affiche mon nom est Dupond Jean

fichier **coursphp_05.php3**

mais dans

```
echo 'mon nom est $nom $prenom';
```

on affiche mon nom est $nom $prenom'

Les séquences d'échappement

Dans une commande echo, si on veut inclure des caractères spéciaux dans la chaîne de texte (non représentable, ou affichables classiquement) il est possible d'inclure une séquence dite « d'échappement » par le caractère ****

par exemple pour permettre l'affichage d'un ' on écrit **\'**

```
echo "il s\'appelle $nom $prenom";
```

Autres codes : **\n** saut de ligne

\r fin de ligne

\t tabulation comportement parfois inattendu !

La concaténation

Un opérateur de concaténation existe, c'est le **.** (point)

```
$patronime="Mon nom est ".$nom." ".$prenom." !";
echo $patronime
```

Définition :

Les constantes se comportent comme des variables, à l'exception du fait que leur valeur est définie grâce à la fonction **define(),** et qu'elle ne peut pas être modifiée par la suite

```
<?php
define("CONSTANT", "Bonjour le monde.");
echo CONSTANT;                          // affiche "Bonjour le monde."
?>
```

NB: les constantes ne peuvent avoir un nom commençant par $ (on les confondrait avec des variables)

Constantes pré-définies

Il existe bien sûr quelques constantes pré-définies en PHP permettant de connaître la version par exemple de l'interpréteur PHP qui est au travail, ou le nom du fichier contenant le script qui s'exécute....

voir fichier **Coursphp_06.php3**

```
<?
 print ("version php ");
 print (PHP_VERSION);
 print ("<br> version os ");
 print (PHP_OS);
 print ("<br> ceci est donc la version php : ".PHP_VERSION."sous ".PHP_OS);
?>
```

Elles ne sont pas fondamentales ici !

FORMULAIRES HTML (CRÉATION)

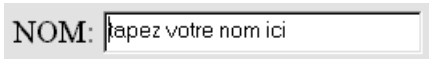
Principe

Beaucoup de pages contiennent des formulaires qui, une fois remplie par l'utilisateur, peuvent être transmis au serveur WEB hébergeant le site

Pour préparer un formulaire il faut une zone d'édition (appelée FORM), puis définir la méthode à employer pour transmettre au serveur l'information recueillie dans les champs du formulaire

Dans un formulaire, tous les tags de mise en forme sont disponibles classiquement, Il faut simplement éviter d'incorporer un formulaire dans un autre formulaire ou de croiser un formulaire et un tableau

Soit le formulaire suivant :



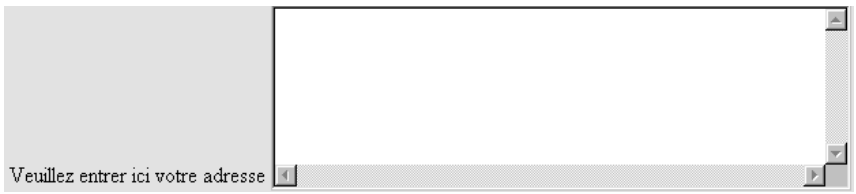
Zone de texte ou Text Box,

Zone masquée ou Password Box,

Boutons Radio ou Radio Button

Cases à Cocher ou Check Box

Listes Déroulantes ou Select input



Scrolling text

ou

Zone de texte

Structure Générale

Les tags nécessaires sont les suivants

<FORM> </FORM> marqueur de début et de fin de formulaire

deux paramètres principaux existent pour un formulaire

METHOD="type" indique quel protocole HTTP il faut utiliser pour envoyer les données

METHOD="post" Les données du formulaire sont envoyées dans le corps de la requête http. "Post" est le protocole à utiliser en général

- On ne voit pas les paramètres « voyager » dans l'url.
- On pourra envoyer plus de données.

METHOD="get" Les données du formulaire sont envoyées dans l'url de la requête http.

- On voit les paramètres « voyager » dans l'url précédée d'un ?, sous la forme d'un couple nom=valeur un & séparant chaque couple
- On pourra envoyer moins de données. (taille de l'URL…)

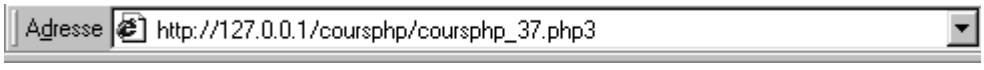
ACTION="script" indique le script qui doit traiter les résultats du formulaire.

Ainsi notre formulaire s'inscrira le plus souvent dans la trame suivante

```
<FORM METHOD="post" ACTION="adresse URL du script">
    Corps du formulaire
</FORM>
```

ce qui est "transparent" au niveau des passages de parmètres dans l'URL

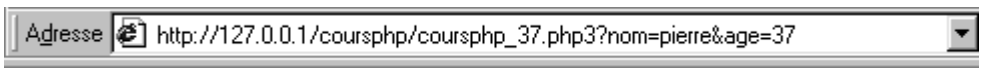
on verrait par exemple:



Si on avait écris dans notre formulaire

```
<FORM METHOD="get" ACTION="adresse URL du script">
```

on verrait par exemple les paramètres passés au script…



Ensuite le corps du formulaire est composé de 3 tags principaux

INPUT

SELECT

TEXTAREA

Chacun de ces tags amenant une zone de saisie à l'écran, se voit assigner d'un paramètre NAME="nom" avec "nom" étant un identificateur donné pour la valeur saisie par l'utilisateur dans le formulaire à ce niveau (et donc pour pouvoir ensuite s'en servir ultérieurement)

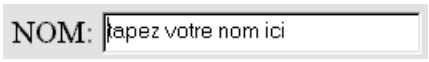
Souvent ces tags acceptent un paramètre VALUE="x" permettant d'indiquer le plus souvent la valeur par défaut à renvoyer

Enfin chacun de ces tag accepte des paramètres biens spécifiques selon la zone de saisie qu'il crée

Le tag INPUT type ="text"

TYPE="text" permet de créer un champ de saisie, c'est le type par défaut si rien n'est spécifié

```
NOM : <INPUT TYPE="text"
Name="nom" SIZE=25 MAXLENGTH=25 VALUE="tapez votre nom ici">
```

génère 

la valeur saisie sera stockée dans une variable nommée "nom".

SIZE correspond à la taille de la zone

MAXLENGHT correspond à la taille maxi de l'info pouvant être saisie

VALUE correspond à la valeur renvoyée par défaut si rien d'autre n'est saisi.

Variante

TYPE="password" idem "text" mais l'utilisateur ne voit pas les lettres tapées au clavier

génère 

Le tag INPUT TYPE="radio"

TYPE="radio" permet de créer des "boutons radio", un seul bouton est sélectionnable sur l'ensemble des possibilités

```
<INPUT TYPE="Radio" NAME="sexe" VALUE=1 CHECKED>Mr
<INPUT TYPE="Radio" NAME="sexe " VALUE=2>Mdme
<INPUT TYPE="Radio" NAME="sexe " VALUE=3>Mlle
```

génère 

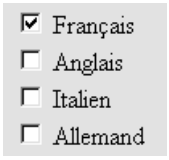
la valeur choisie (1,2 ou 3) sera stockée dans une variable nommée "sexe"

N.B : le nom de la variable doit être le même pour l'ensemble des choix
Le paramètre CHECKED active un bouton par défaut

Le tag INPUT TYPE="checkbox"

TYPE="checkbox" permet de créer des "cases à cocher", plusieurs cases sont sélectionnables sur l'ensemble

```
<INPUT TYPE="Checkbox" NAME="langues" VALUE="Fr"      CHECKED>Français<BR>
<INPUT TYPE="Checkbox" NAME="langues" VALUE="Gb">Anglais <BR>
<INPUT TYPE="Checkbox" NAME="langues" VALUE="Ita">Italien <BR>
<INPUT TYPE="Checkbox" NAME="langues" VALUE="All">Allemand<BR>
```

génère 

la valeur saisie (Fr, Gb, Ita ou All) sera stockée dans une variable nommée " langues ".

On peut prévoir un nom de variable par case à cocher.

Le paramètre CHECKED active une case par défaut

Le tag SELECT

C'est le tag qui va permettre des entrées sélectionnées principalement

Sa structure générale étant

<SELECT> </SELECT> encadrant l'ensemble des valeurs constituant la liste, Chacune de ces valeurs étant précédée du tag <OPTION>.

Si MULTIPLE est précisé une sélection multiple via la touche CTRL est autorisée

```
<SELECT NAME="h" SIZE=4>
<OPTION>08h00</OPTION>
<OPTION>08h30</OPTION>
<OPTION SELECTED>09h00</OPTION>
<OPTION>09h30</OPTION>
<OPTION>10h00</OPTION>
</SELECT>
```

génère 

et la valeur sélectionnée ira dans une variable nommée "h" (pour heure…)

Le paramètre SELECTED sélectionne une valeur par défaut

On peut dissocier les choix apparaissant dans la liste et les valeurs retournées

<OPTION **VALUE=8**>08h00</OPTION>

le tag TEXTAREA

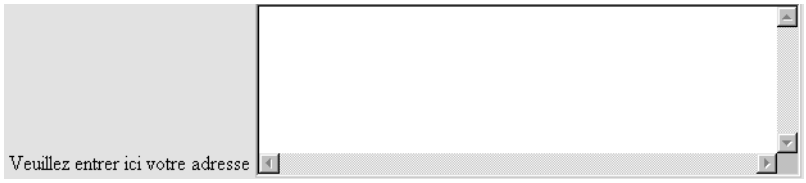
C'est le tag qui va permettre des entrées clavier volumineuses

Sa structure générale étant

<TEXTAREA> </TEXTAREA> Permet de définir dans un formulaire extensible "à volonté" par l'utilisateur une zone de texte lors de sa saisie.

Si ROWS et COLS sont précisés une taille en lignes et colonnes sera donnée

```
Veuillez entrer ici votre adresse
<TEXTAREA NAME="adresse" ROWS=5 COLS=40> </TEXTAREA>
<BR>
```

génère 

et la valeur saisie ira dans une variable nommée "adresse"

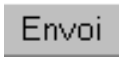
Annulation ou Envoi

La fin d'un formulaire est toujours constituée de deux boutons permettant soit de remettre à zéro les saisies effectuées dans le formulaire (abandon) soit d'envoyer le formulaire (envoi)

Leur structure étant

<INPUT TYPE="submit" VALUE="xxx"> Pour le bouton qui déclenchera l'envoi des valeurs saisies dans le formulaire "xxx" étant le libellé du bouton

```
<INPUT TYPE="submit" VALUE="Envoi">
```

génère 

et les valeurs saisies seront expédiées

<INPUT TYPE="reset" VALUE="xxx"> Pour le bouton qui effacera toutes les valeurs saisies dans le formulaire "xxx" étant le libellé du bouton

```
<INPUT TYPE="reset" VALUE="Annuler">
```

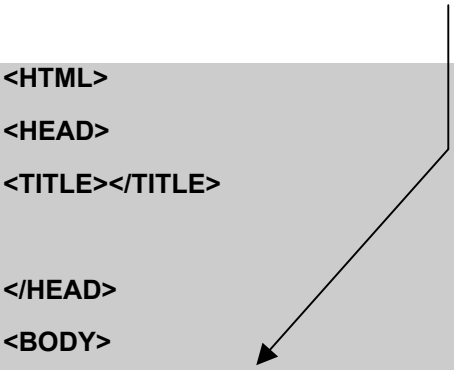
génère 

et les valeurs saisies seront effacées

Récupération des données des champs :

Quand l'un de vos visiteurs entre les informations dans un formulaire, celles-ci sont récupérées sous forme de variables, c'est tout simplement le paramètre **name** de chaque champ qui devient la variable qui contient ce qu'a entré le visiteur dans le champ.

En effet lorsque l'on clique sur le bouton **Submit** du formulaire HTML, le script désigné par l'attribut **action** de la balise **Form** recevra les données du formulaire que le moteur php **convertit ensuite en variables**



voir fichier **Coursphp_07.html**

```
<HTML>
<HEAD>
<TITLE></TITLE>

</HEAD>
<BODY>
<Form Action = "coursphp_07.php3" method="post">
Nom          <Input type="text" name="nom"> <br>
Age          <Input type="text" name="age">  <br>
<Input type="submit" value="go">
</Form>
</BODY>
</HTML>
```

dans l'exemple ci-dessus le **name="nom"** devient **$nom** et **name="age"** devient **$age**, il ne reste plus qu'à faire un **print()** des variables et le tour est joué !

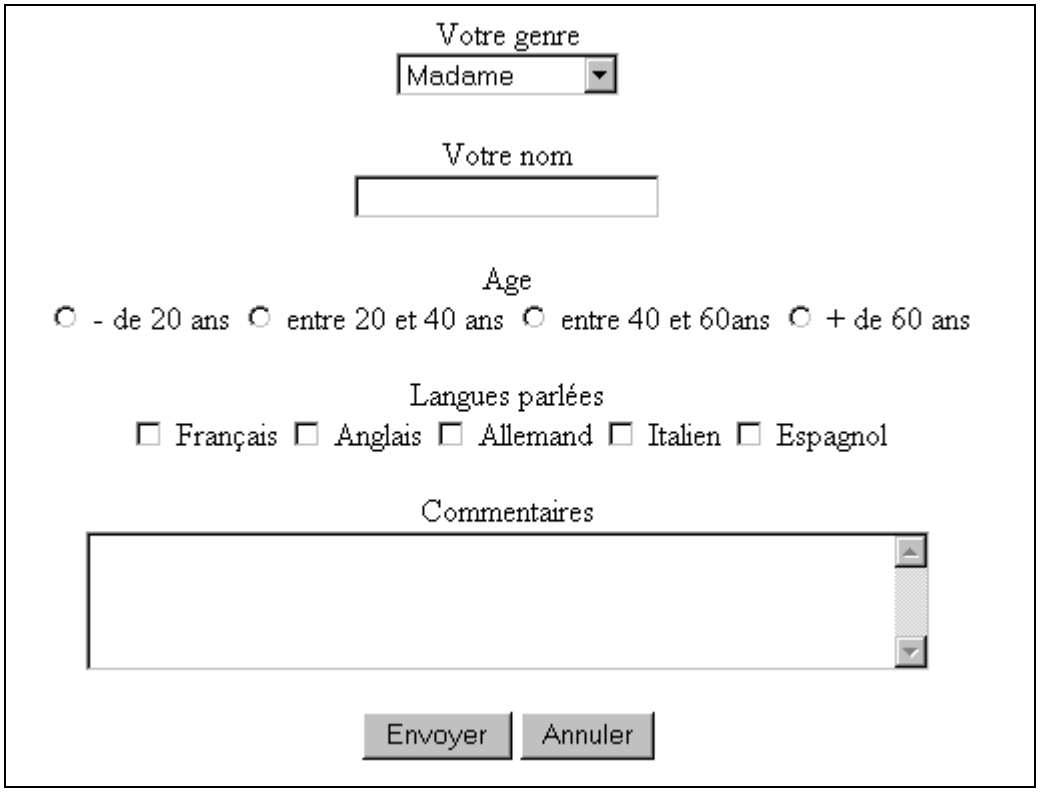
voir fichier **Coursphp_07.php3**

```
<?
 print(" <h3>salut</h3> <br> $nom  ton age est $age ans ! <br>");
?>
```

N.B: n'utilisez jamais d'accent ni d'espace dans les paramètres **name=** des balises HTML du formulaire.

Autre Exemple

Soit le formulaire suivant :



voir fichier **coursphp_08.html**

```
<html>
<body >
<form method="post" action="coursphp_08.php3">
 <div align="center">Votre genre<br>
  <select name="genre">
   <option value="Mme">Madame</option>
   <option value="Mr">Monsieur</option>
   <option value="Mlle">Mademoiselle</option>
  </select>
 </div>
 <p align="center">Votre nom<br>
  <input type="text" name="nom">
 </p>
 <p align="center">Age<br>
  <input type="radio" name="age" value="- de 20">
  - de 20 ans
  <input type="radio" name="age" value="- de 40">
  entre 20 et 40 ans
  <input type="radio" name="age" value="- de 60">
  entre 40 et 60ans
  <input type="radio" name="age" value="+ de 60">
  + de 60 ans</p>
 <p align="center">Langues parl&eacute;es<br>
  <input type="checkbox" name="langue1" value="fran&ccedil;ais">
  Fran&ccedil;ais
  <input type="checkbox" name="langue2" value="anglais">
```

```
  Anglais
  <input type="checkbox" name="langue3" value="allemand">
  Allemand
  <input type="checkbox" name="langue4" value="italien">
  Italien
  <input type="checkbox" name="langue5" value="espagnol">
  Espagnol</p>
 <p align="center">Commentaires<br>
  <textarea name="commentaire" cols="50" rows="4"></textarea>
 </p>
 <p align="center">
  <input type="submit" name="Submit" value="Envoyer">
  <input type="reset" name="Reset" value="Annuler">
 </p>
</form>
</body>
</html>
```

et le script php suivant :

voir fichier **coursphp_08.php3**

```
<?
 echo " Bonjour $genre $nom vous avez $age ans
 et vous parlez : $langue1 $langue2 $langue3 $langue4 $langue5<br>";
 echo " de plus vous nous avez laissé le commentaire suivant : $commentaire";
?>
```

Traitement de formulaire :

Le principe de récupération des données dans un formulaire est donc très très simple, mais que peut on faire en php avec ces données ?

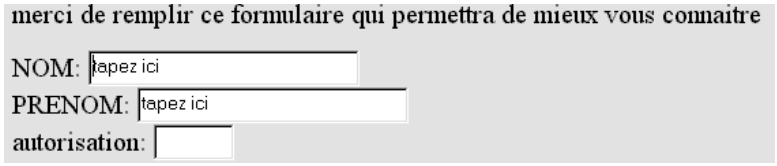
- On peut bien sûr effectuer des traitements sur ces variables, pour vérifier leur valeur ou même simplement leur existence.
- On peut renvoyer un mail avec les données collectées
- On peut stocker ces données dans un fichier de type fichier texte...
- On peut alimenter une table de base de données.

FORMULAIRES (PRÉSENTATION)

Les formulaires demandent une présentation soignée

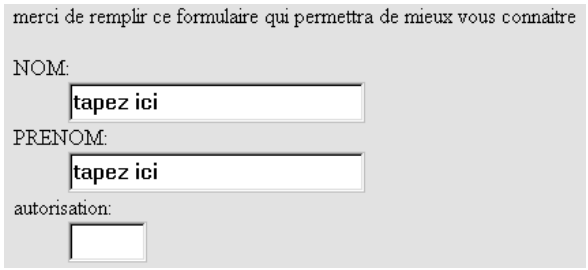
Les listes et les tableaux restent à disposition mais il faut faire attention à respecter la syntaxe des tags

Ainsi à la place de



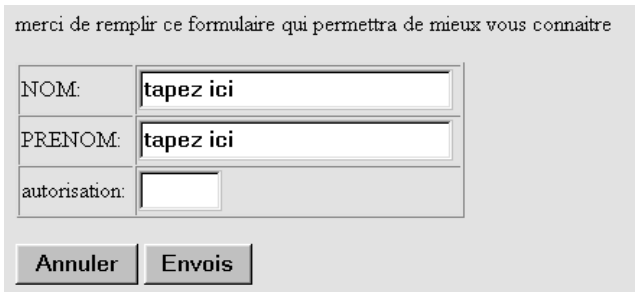
on préfèrera

fondé sur des listes



ou bien aussi

fondé sur un tableau



Principe d'utilisation :

Les fonctions sont des expressions qui ont la valeur de leur "valeur de retour".

Par exemple, considérons la fonction suivante :

voir fichier **coursphp_09.php3**

```
function foo () {
return 5;
}
```

alors on peut dire que Si foo() renvoie 5, la valeur de l'expression 'foo()' est 5.

et donc **$c = foo()** est équivalent à **$c = 5**

Habituellement, les fonctions ne font pas que renvoyer une valeur constante mais réalisent aussi des traitements.

Syntaxe :

Une fonction peut être définie en utilisant la syntaxe suivante :

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
echo "Exemple de fonction.\n";
return $retval;
}
```

En PHP3, les fonctions doivent être définies avant qu'elles ne soient utilisées. Ce n'est plus le cas en PHP4.

Valeur de retour simple

La valeur est renvoyée en utilisant une instruction de retour **return** optionnelle. Tous les types de variables peuvent être renvoyés.

```
function square($num) {
return $num * $num;
}
echo square(4); // affiche '16'.
```

Exemple :

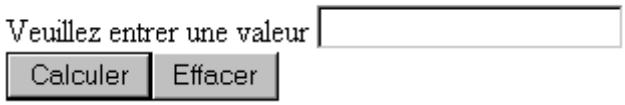
```
<?
function square($val)
      {
      $retour=$val*$val;
      return $retour;
      }
$num=4;
/* appel de la fonction*/
echo square($num);
?>
```

Cette fonction a un petit problème elle affiche tout le temps 16…

Variante

On va prévoir un formulaire (**square.html**) demandant la saisie d'une valeur et c'est cette valeur qui sera élevée au carré (**square.php3**).

Le Formulaire :



voir fichier **square.html**

```
<HTML>
<BODY>
<FORM action=square.php3 method=post>
Veuillez entrer une valeur <input type=text name=num><br>
<input type=submit value=Calculer><input type=reset value=Effacer>
</form>
</body>
</html>
```

Le script modifié devient alors

voir fichier **square.php3**

```
<?
function square($val)
      {
      $retour=$val*$val;
      return $retour;
      }
/* appel de la fonction*/
echo "vous avez tapé la valeur $num. Sa valeur au carré s'élève à ".
square($num);
?>
```

Valeurs de retour multiples

On a dit précédemment que la valeur est renvoyée en utilisant une instruction de retour **return** optionnelle. Même si nous n'avons pas encore abordé les tableaux, il faut savoir que une fonctions ne peut pas renvoyer plusieurs valeurs de retour, mais elle peut renvoyer un tableau !

N.B : dans cet exemple, même si on ne voit pas bien le fonctionnement de List (nous verrons cela dans le chapitre consacré aux tableaux…), on voit bien que la fonction renvoit un tableau, et que on récupère dans 3 variables les 3 valeurs du tableau…

voir fichier **coursphp_09b.php3**

```
function small_numbers() {
return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
echo $zero;
echo $one;
echo $two;
```

PASSAGE DE PARAMETRES

Principe

Des informations peuvent être passées à une fonction en utilisant un tableau d'arguments, dont chaque élément est séparé par une virgule. Un élément peut être une variable ou une constante.

Par défaut, **les arguments sont passés à la fonction par valeur** (donc vous pouvez changer la valeur d'un argument dans la fonction, cela ne change pas sa valeur à l'extérieur de la fonction)

voir fichier **coursphp_10.php3**

```
<?php
 function ajout($string) {
$string = $string.", et un peu plus.";
echo "dans la fonction";
echo $string; // affiche le contenu de string concaténé à la chaîne ", et un peu plus."
echo "<br>";
}
$str = "Ceci est une chaîne";
echo "avant l'appel de la fonction ";
echo $str;
echo "<br>";
ajout($str);
echo "après l'appel à la fonction ";
echo $str; // affiche le contenu de string une fois sortie de la fonction.
 ?>
```

Cela donnera normalement

avant l'appel de la fonction Ceci est une chaîne
dans la fonctionCeci est une chaîne., et un peu plus.
après l'appel à la fonction Ceci est une chaîne

Si vous voulez que vos fonctions puisse changer la valeur des arguments, vous devez passer ces arguments par référence.

Deux méthodes existent, selon que vous souhaitiez passer les arguments toujours par référence (on le prévoit dans la fonction) ou selon que vous souhaitiez passer les arguments tantôt par référence et tantôt par valeur (cela dépendra de l'appel de la fonction)

1° méthode :

Si vous voulez qu'un argument soit toujours passé par référence, vous pouvez ajouter un '**&**' devant l'argument **dans la déclaration de la fonction**

Dans ce cas quel que soit l'appel, les paramètres sont passés par référence.

voir fichier **Coursphp_10b.php3**

```
function ajout(&$string) {
$string = $string.", et un peu plus.";
}
```

2° méthode :

Si vous souhaitez passer une variable par référence à une fonction mais de manière ponctuelle, vous pouvez ajouter ou non un '**&**' devant l'argument dans l'appel de la fonction:

voir fichier **coursphp_10c.php3**

```
function ajout($string) {
$string = $string.", et un peu plus.";
}
```

Dans ce cas un appel avec un '**&**' est appel par référence

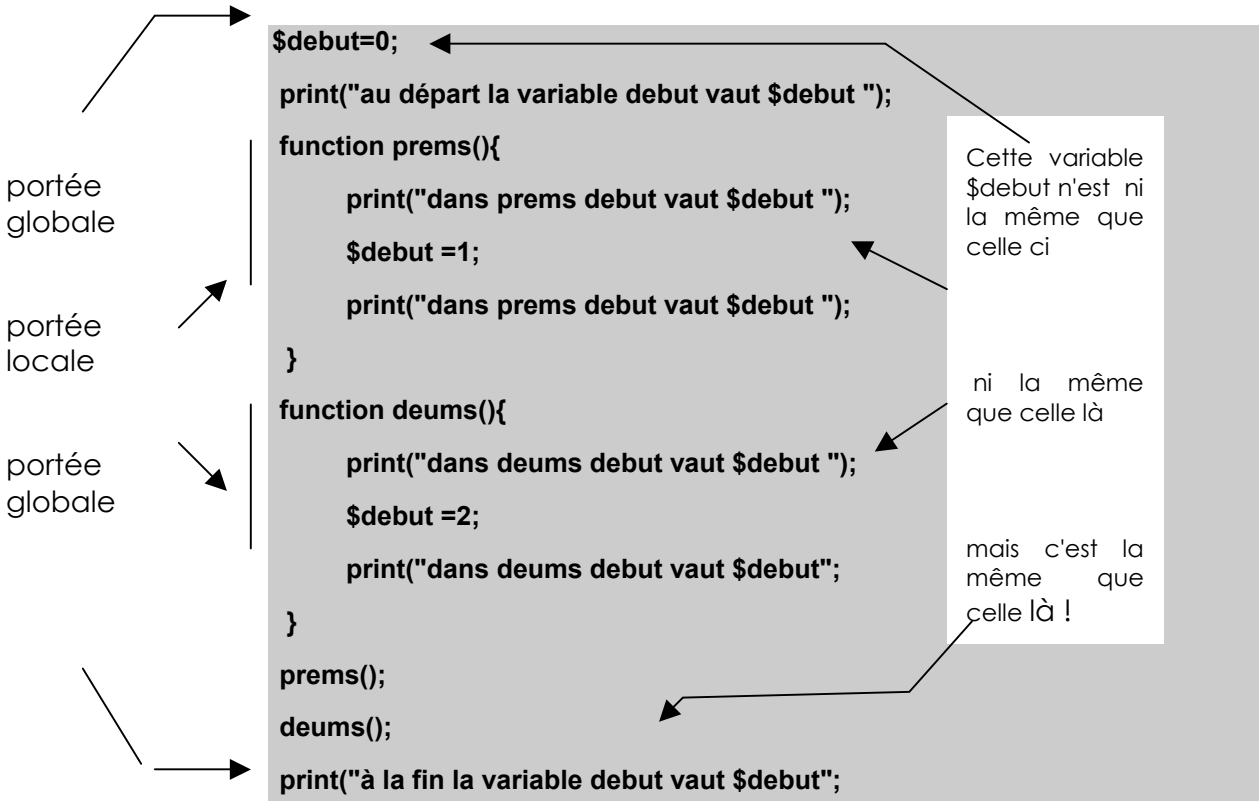
```
ajout(&$str);
```

Dans ce cas un appel sans le '**&**' est appel par valeur

```
ajout($str);
```

Portée locale ou globale :

On peut dire que classiquement, en php, lorsqu'une variable est utilisée dans une fonction, que sa portée est limitée à la fonction. Lorsque la fonction a terminé de s'exécuter, la variable disparaît. De ce fait si on crée des variables homonymes dans différentes fonctions, en fait ce ne sont pas les mêmes variables, leur portée étant locale ! Regardez l'exemple en **coursphp_11.php3**



Ce qui est gênant, c'est que l'on se demande comment pouvoir utiliser une variable globale dans une fonction

Cela est possible en spécifiant qu'il s'agit d'une variable globale par le mot clé **global**. ainsi dans

on spécifie que $debut est globale →

```
function prems(){
        global $debut;
        $debut=2;
        print("dans prem debut vaut $debut ");
        $debut =1;
        print("dans prem debut vaut $debut ");
}
```

Attention : les variables locales sont prioritaires sur les variables globales

Mais quelle différence alors avec un "passage de paramètre" comme on l'a vu dans le chapitre précédent sur les fonctions ? et bien c'est que Global permet de définir des variables globales depuis l'intérieur d'une fonction....

voir exercice **coursphp_11b.php3**

```
<html>
<body>
<?php
 //$debut=0; ici la variable globale n'est plus définies
 print("au debut vaut $debut ");
 function prems(){
        global $debut;      // mais elle l'est ici !
        $debut=2;
        print("dans prem debut vaut $debut ");
        $debut =1;
        print("dans prem debut vaut $debut ");
 }
 function deums(){
        print("dans deums debut vaut $debut ");
        $debut =2;
        print("dans deums debut vaut $debut");
 }
 prems();
 deums();
 print("a la fin debut vaut $debut");
?>
</body>
```

Cette variable $debut globale est la même que celle ci

Mais ce n'est pas la même que celle là !

Variables statiques :

On a vu que classiquement, en php, lorsqu'une variable est utilisée dans une fonction, sa portée est limitée à la fonction. De plus, lorsque la fonction a cessé de s'exécuter, la variable disparaît.

Mais on peut avoir envie de faire en sorte que la variable soit mémorisée, de manière à ce que sa valeur soit connue lors du prochain appel de la fonction, (et non pas recrée...)

Il s'agit alors d'une variable locale crée lors du premier appel, et dont la valeur est mémorisée et est réutilisable lors du prochain appel de la fonction.

voir fichier **coursphp_11t.php3**

```
function couleur()
    {
        static $couleur;
        if ($couleur == "#00FF00")
        {
            $couleur = "#CCFFCC";
        }
        else
        {
            $couleur = "#00FF00";
        }
        return ($couleur);
    }

    print ("<Table width=300>");

    for ($cpt=0; $cpt<10; $cpt++)
    {
        $couleuractive = couleur();
        print ("<TR><TD BGcolor=$couleuractive>");
        print ("ligne $cpt <BR><BR></TD></TR>");
    }

    print ("</Table>");
```

on déclare une variable

valant alternativement 2 verts

on renvoi une couleur

construction des 10 lignes du tableau en HTML

N.B : Pour faire cela il aurait suffit de déclarer une variable globale bien sûr, mais plus proprement, on peut déclarer une variable locale avec le mot clé **static, cela évite une modification possible depuis une autre fonction ailleurs dans le code…. Puisque cette variable est inconnue ailleurs !**

Opérateurs de comparaison < > ==:

En fonction du résultat du test de comparaison des opérandes, les opérateurs de relation donnent la valeur 1 (true) ou 0 (false).

Le type booléen n'existant pas en PHP, False est équivalent à 0 mais True est simplement différent de False, donc peut valoir 1, **mais par extension toute valeur non nulle sera considérée comme True.**

Une chaîne de caractères vide et la chaîne de caractère 0 sont considérées comme FALSE.

= =	True (vrai) dans le cas **d'égalité** des opérandes, false dans le cas inverse
!=	True (vrai) dans le cas **d'inégalité** des opérandes, false dans le cas inverse
>	true si l'opérande de gauche est **supérieur à** l'opérande de droite, false dans le cas inverse
>=	true si l'opérande de gauche est **supérieur ou égal à** l'opérande de droite, false dans le cas inverse
<	true si l'opérande de gauche est **inférieur** à l'opérande de droite, false dans le cas inverse
<=	true si l'opérande de gauche est **inférieur ou égal** à l'opérande de droite, false dans le cas inverse

Logiques || && :

AND ou &&	ET logique donne true si les deux opérandes sont true donne false sinon (un opérateur au moins est False)
OR ou | |	OU logique donne true si l'un des deux opérandes est true et false sinon (deux opérateurs sont False)
!	la négation (NOT) donne l'inverse logique

N.B: la différence entre les écritures AND et && réside dans une priorité plus forte de la notation AND (il vaudrait mieux éviter d'utiliser && qui a une priorité très faible...)

N.B: la différence entre les écritures OR et || réside dans une priorité plus forte de la notation OR (il vaudrait mieux éviter d'utiliser || qui a une priorité très faible...)

Voici quelques exemples

voir fichier **Coursphp_12.php3**

```
<?
if ("a"<"3")
   print("valeur true");
else
   print("valeur false");

$a = 122;
if ($a) print("<br> true"); // TRUE est non nul
else
 print("<br> false");

$a = 0;
if (!$a) print("<br> true"); // !0 vaut souvent 1 (TRUE)
else
 print("<br> false");

$b=10;
if ($a==$b) print("<br> égal");// comparaison classique
else
print("<br> non égal");

if ($a=$b) print ("<br> égal");          // == est différent de =, ici
else                                     //a=b vaudrait 10 donc True
print("<br> non égal");
?>
```

conversion implicite

valeur de l'expression

Le if - else :

Souvent, vous voulez exécuter une instruction si une condition est remplie, et une autre instruction si cette condition n'est pas remplie

C'est exactement à cela que sert l'instruction **if** éventuellement complétée d'un **else**.

Les mots clés pour un si sont **if** **else** avec l'indentation suivante

```
if (test)
    {...
    }
else
    {...
    }
```

ou plus simplement

```
if (test)
    {...
    }
```

On utilise cette forme d'écriture de test, seulement si nous désirons que le programme fasse quelque chose de bien précis dans les deux cas. Aussi bien lorsque la condition est vraie que si elle est fausse.	Dans cette forme d'écriture des tests, les actions sont exécutées seulement si la condition est vraie, et il ne se passe strictement rien dans le cas contraire, c'est-à-dire si la condition n'est pas vraie (fausse).
On peut aussi recommencer à tester une autre condition dans le cas où la première n'est pas vraie. C'est ce que l'on appelle les "if imbriqués".	Le programme reprend simplement l'exécution des instructions qui suivent l'accolade fermante } après la fin du test, s'il y en a.

Quelques exemples :

- pas de **else**, et une seule instruction à exécuter

```
if ($a > $b)
print "a est plus grand que b";
```

- pas de **else**, et plusieurs instructions à exécuter

```
if ($a > $b) {
print "a est plus grand que b";
$b = $a;
}
```

- avec un **else**, et plusieurs instructions prévues à exécuter

```
if ($a > $b) {
print "a est plus grand que b";
} else {
print "a est plus petit que b";
}
```

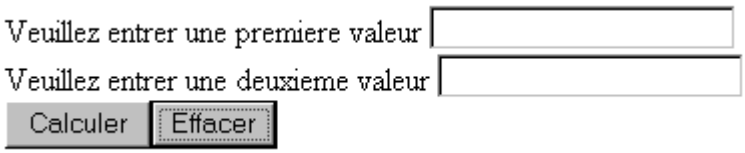
Les imbrications sont possibles, et le **else** est apparié au dernier **if** n'ayant pas de **else**. Dans ce cas une indentation soigneuse vous permettra de vous repérer

```
if   (test)
     {...
      ...
     }
else
     {...
     if       (test)
              {...
               ...
              }
              else
              {...
               ...
              }
     }
```

N.B: Les opérateurs logiques Et (&&) et OU (||) restent disponibles pour écrire des tests plus complexes.

Exemple :

Nous allons créer un formulaire qui demande la saisie de 2 nombres :



Ces 2 nombres seront ensuite testés dans un script php. Ce script affichera les 2 nombres et précisera lequel est le plus grand ou s'ils sont égaux.

Le formulaire html :

voir fichier **test_valeur.html**

```
<HTML>
<BODY>
<FORM action=test_valeur.php3 method=post>
Veuillez entrer une premiere valeur <input type=text name=val1><br>
Veuillez entrer une deuxieme valeur <input type=text name=val2><br>
<input type=submit value=Calculer><input type=reset value=Effacer>
</form>
</body>
</html>
```

Script php

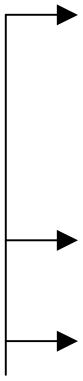
voir fichier **test_valeur.php3**

```
<?
echo "Vous avez saisi les valeurs suivantes : $val1 et $val2";
if($val1==$val2)
     {
     echo "Les 2 valeurs sont égales";
     }
     else
        {
         if ($val1>$val2)
           {
           echo "Valeur 1 superieure à valeur 2";
           }
           else
              {
           echo "Valeur 2 superieure à valeur 1";
           }
         }
?>
```

Le Switch :

L'instruction **switch** équivaut à une série d'instructions if. En de nombreuses occasions, vous aurez besoin de comparer la même variable (ou expression) avec un grand nombre de valeurs différentes, et d'exécuter différentes parties de code suivant la valeur à laquelle elle est égale. C'est exactement à cela que sert l'instruction switch.

Cette opération resterait possible avec des if imbriqués....(dans certaines limites)



```
switch (Variable) {
case Valeur1 :      xxxxxx ;
                    xxxxxx ;
break;
case Valeur2 :      xxxxx ;
                    xxxxx ;
                    xxxxx ;
break;
case Valeur3 :      xxxxx ;
break;
}
```

Le break est essentiel, sinon "on continue" à exécuter le code jusqu'à trouver un break ou la fin du switch

```
switch (Variable) {
case Valeur1 :
case Valeur2 :      xxxxx ;
                    xxxxx ;
                    xxxxx ;
break;
case Valeur3 :      xxxxx ;
default :           xxxxx ;
break;
}
```

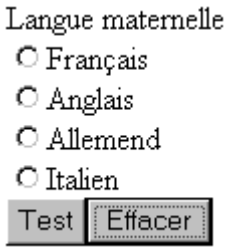
ici, en l'absence de Break, on a le même traitement pour Valeur 1 et pour Valeur2...

default permet de spécifier le traitement à exécuter lorsque Variable ne prend aucune des valeurs prévues

Exemple

Nous allons créer un formulaire (test_langue.html) qui demande de faire un choix parmi une liste de 4 langues.

Lorsque le choix estr fait le résultat est transmis à un script (test_langue.php3) qui affiche le choix réalisé



Le formulaire html

voir fichier **test_langue.html**

```
<html>
<body>
<FORM action=test_langue.php3 method=post>
Langue maternelle<br>
<input type=radio name=langue value=fr>Français<br>
<input type=radio name=langue value=gb>Anglais<br>
<input type=radio name=langue value=d>Allemend<br>
<input type=radio name=langue value=i>Italien<br>
<input type=submit value=Test><input type=reset value=Effacer>
</form>
</body>
</html>
```

Le script php

voir fichier **test_langue.php3**

```
<?
echo "Votre langue maternelle est : ";
switch ($langue) {
case fr: echo "le français";
break;
case gb: echo "l'anglais";
break;
case d: echo "l'allemand";
break;
case i: echo "l'italien";
break;
}
?>
```

Maintenant que nous sommes un peu capables de faire des tests, et de récupérer des paramètres saisi dans des formulaires, il paraît nécessaire de pouvoir les "tester", et de manière générale, de pouvoir les manipuler…

Il va bien sûr maintenant falloir contrôler les informations que rentre le visiteur pour éviter au maximum les erreurs.

Il existe ne Php toute une série de fonctions et de manières de manipuler les chaînes de caractères…

Test d'un champ vide : (la fonction empty)

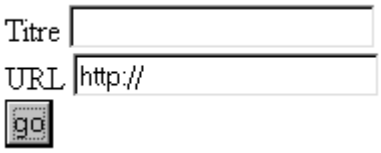
La première fonction que nous utiliserons est **empty()**, qui permet de contrôler si un champ est vide.

Fonction empty :

int **empty (var)**

Retourne False (0) si la variable **var** est affectée ou à une valeur différente de 0, retourne True (1) sinon.

Soit le formulaire suivant :



voir fichier **coursphp_13.html**

Dans lequel on veut vérifier que le champs Titre contienne quelquechose…

1° méthode :

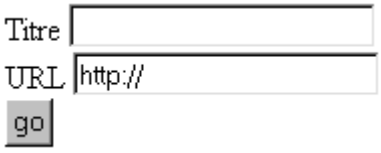
```
<?
if(empty($titre))
{
print("<center>Le '<b>Titre</b>' est vide !</cen
exit();
}
print("$titre : <a href=\"$url\">$url</a>");
?>
```

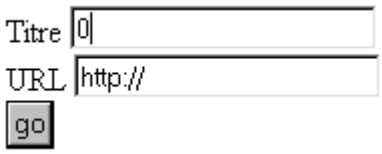
voir fichier **coursphp_13.php3**

Si la variable **$titre** est vide (ou vaut 0) alors on affiche le message et on arrête l'exécution du reste du code avec la commande **exit().**

Par contre si la variable n'est pas vide, l'exécution ne prend pas en compte ce qui se trouve entre accolades et continue sur la suite du programme.

Alors

Aussi bien 

que cela 

Entraine le résultat suivant :

Le '**Titre**' est vide !

2° méthode :

En modifiant le formulaire par

```
<Form Action = "coursphp_13b.php3" method="post">
```

de manière à executer le script php suivant

```
<?
if(empty($titre))
{
print("<center>Le '<b>Titre</b>' est toujours vide !</center>");
}
else
{
print("$titre : <a href=\"$url\">$url</a>");
}
?>
```

voir fichier **coursphp_13b.php3**

Cette méthode est plus "propre" car plus structurée

"Petit apparté…" : si vous pensez pouvoir ne pas utiliser empty(), et ecrire simplement quelquechose du genre

```
<?
if($titre)
{
print("<center>Le '<b>Titre</b>' est vide !</center>");
}
print("$titre : <a href=\"$url\">$url</a>");
?>
```

vous obtiendre exactement l'effet inverse, car lorsque l'on tape un titre, la variable $titre vaut quelque chose de différent de 0, dont est assimilée à la valeur logique True, donc on déclarera que le titre est vide…..

Conversion et extraction dans une chaîne :

Trois fonctions sont souvent utilisées pour manipuler des chaînes

Fonction stripslaches :

string **stripslashes (**string **str)**

Retourne une chaîne dont tous les slashes ont été supprimés. (\' devient ', ... et ainsi de suite). Les doubles backslashes sont remplacés par des simples.

Fonction strtolower() :

string **stripslashes (**string **str)**

Retourne une chaîne dont tous les caractères ont été transformés en minuscules.

(ex. HTTP://www.MONsite.CoM devient http://www.monsite.com).

Voir aussi strtoupper() (passe tout en majuscule) et ucfirst() (passe le premier caractère en majuscule)

Fonction substr() :

string **substr (**string **str**,depart,longueur**)**

Permet de sélectionner x caractères de la chaîne str, à partir de la position depart et sur une longueur définie par longueur.

Si depart est positif, la chaîne retournée commence au caractère depart, a partir du 1° (début) de la chaîne str. Si depart est négatif, on compte à partir de la fin de la chaîne str.

Voir aussi ereg() (plus loin…).

Nous allons contrôler que **$url** commence bien par les caractères "**http://**" à l'aide des deux fonctions **strtolower()** et **substr()**.

```
<?
$verif_url = strtolower($url);
$verif_url = substr("$verif_url", 0, 7);
if ($verif_url!="http://")
{
print("L'URL doit commencer par <b>http://</b>");
}
else
{
print("$titre : <a href=\"$url\">$url</a>");
}
?>
```

voir fichier **coursphp_14.html**

coursphp_14.php3

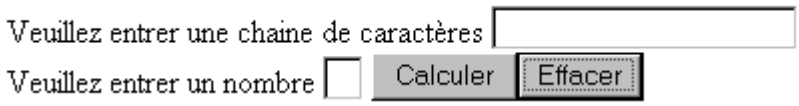
La fonction **substr()**, permet de sélectionner les 7 premiers caractères (**0** est toujours le premier caractère d'une chaîne - le second chiffre ' **7** ' étant le nombre de caractères à sélectionner), puis nous les comparons à ce que nous avons dans notre condition if :

Si les 7 premiers caractères sont différents (signe: **!=**) de "**http://**", alors on exécute ce qui se trouve entre accolades *(en l'occurrence on affiche un message d'erreur)*

Par contre si le résultat est correct, PHP ignore ce qui se trouve entre accolades et exécute le reste du code.

Exemple

Nous allons créer un formulaire qui permet la saisie d'une chaîne de caractères et d'un nombre



Ce formulaire appellera un script php (left_right.php3) qui affichera les x premiers caractères de la chaîne et les x derniers caractères de la chaîne

Le formulaire HTML :

voir fichier **left_right.html**

```
<HTML>
<BODY>
<FORM action=left_right.php3 method=post>
Veuillez entrer une chaine de caractères <input type=text name=saisie><br>
Veuillez entrer un nombre <input type=text name=nb size=1>
<input type=submit value=Calculer><input type=reset value=Effacer>
</form>
</body>
</html>
```

Le Script php :

voir fichier **left_right.php3**

```
<?
function left($chaine,$num) {
return substr($chaine,0,$num);
}
function right($chaine,$num) {
return substr($chaine,-$num);
}
$saisie;
echo "les $nb premières lettres sont ". left($saisie,$nb)."<br>";
echo "les $nb dernières lettres sont ".right($saisie,$nb);
?>
```

Recherche d'un caractère dans une chaîne

On peut avoir besoin de chercher un caractère particulmeir dans une chaîne, (séparateur, simbole @….)

Fonction ereg() :

int **ereg (**string **pattern,** string **string)**

Recherche dans la chaîne string les séquences de caractères qui correspondent au masque pattern.

Retourne TRUE (1) si une occurrence a été trouvée dans la chaîne, et FALSE (0) dans le cas contraire, ou si une erreur est survenue. La recherche est sensible à la casse.

Exemples :

Soit le formulaire HTML suivant :

voir fichier **recherche.html**

```
<html>
<body>
 <form action="recherche.php3" method="post">
 saisie de la chaine <input type="text" name="saisie"> <br>
 motif cherché <input type="text" name="motif"> <br>
 <input type=submit>
</body>
</html>
```

Le script suivant) permet de déterminer si un caractère particulier fait partie de la chaîne saisie :

voir fichier **recherche.php3**

```
<?
  $motif="a";
/*on va chercher la présence d'un a dans la chaine saisie */
  if(ereg($motif,$saisie)) {
  echo "il y a un $motif dans le mot saisi";
   }
   else
   {
   echo "il n'y a pas un $motif dans le mot saisi";
   }
?>
```

Recherche d'une expression dans une chaîne

Les expressions régulières constituent une solution efficace pour effectuer des recherches et ou remplacement de chaînes de caractères

On pourrait modifier le script en précisant que l'on cherche non pas 1 caractère mais une chaîne de caractères.

Il suffit d'inscrire la chaîne entre guillemets.

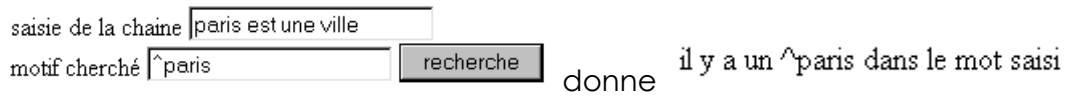
```
$motif="Paris";
```

Si on fait **précéder la chaîne de caractères d'un ^** on cherche la chaîne au **début** de la ligne

```
$motif="^Paris"; recherche Paris au début de la zone de recherche
```

alors

mais



Pour éviter l'affichage du ^ devant Paris on pourrait écrire

```
echo "Il y a un ".substr($motif,1);
```

Si on fait **suivre la chaîne de caractères d'un $** on cherche la chaîne à la **fin** de la ligne

```
$motif="Paris$"; recherche Paris à la fin de la zone de recherche
```

Pour éviter l'affichage du $ devant Paris on pourrait écrire

```
echo "Il y a un ".substr($motif,-1);
```

Le | permet de définir un "ou"

```
$motif="Paris|Lyon"; recherche Paris ou Lyon
```

Recherche dans une plage de caractères

On pourrait encore modifier le script en précisant que l'on cherche non pas 1 caractère mais un caractère d'une liste possible de caractères. Il suffit d'inscrire la liste entre crochets.

```
$motif="[abcA]";
```

On peut aussi définir une plage de caractères. Il suffit d'indiquer entre crochets le premier et le dernier caractères séparés par un tiret.

```
$motif="[a-z]"; // liste des minuscules
$motif="[A-Z]"; // liste des majuscules
$motif="[a-zA-Z]"; // liste des minuscules et majuscules
$motif="[0-9]"; // liste des chiffres
```

On peut aussi faire des recherches négatives, il suffit pour cela de faire précéder le symbole ou la liste du signe^.

```
$motif="[^a-z]"; // tout sauf des minuscules
```

Pour rechercher l'un de ces caractères **+-?,*^$()[]{}|**\ il faut le faire précéder d'un\

Les caractères génériques

? utilisé dans une expression régulière indique que le caractère précédant le point d'interrogation peut ou non exister dans la chaîne recherchée.

```
$motif="Pasc?";
```

trouvera aussi bien : Pascal Passy Pas

. utilisé dans une expression régulière remplace **un** caractère quelconque.

```
$motif="P.ire";
```

trouvera aussi bien : Poire Paire

\+ utilisé dans une expression régulière signifie une ou plusieurs occurrences du caractère précédant le +

```
$motif="colon+e";
```

trouvera aussi bien : colone colonne colonnne

* utilisé dans une expression régulière signifie aucune, une ou plusieurs occurrences du caractère précédant l'étoile *

```
$motif="colon*e";
```

trouvera aussi bien : colone colonne colonnne

.* Nombre quelconque de caractères y compris 0

```
$motif="colo.*e";
```

trouvera aussi bien : colone colonne colonie coloe

{n} ou {n,m} n ou de n à m occurrences du caractère précédant l'accolade.

```
$motif="w{3}";
```

trouvera aussi bien : www.sncf.fr grandwww

Fonctions connexes

ereg_replace — Remplacement par expression régulière.

eregi — Recherche par expression régulière insensible à la casse.

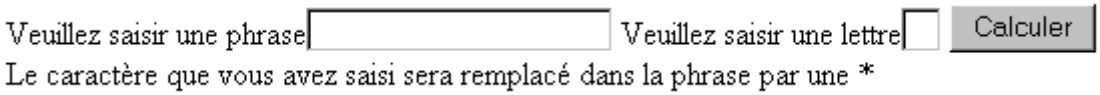
eregi_replace — Remplacement par expression régulière insensible à la casse.

split — Scinde une chaîne en un tableau, grâce à une expression régulière.

spliti — Scinde une chaîne en un tableau, grâce à une expression régulière.

Exemple :

Nous allons créer un formulaire qui demande la saisie d'une phrase et d'un caractère. Ces informations seront transmises à un script php qui affichera la phrase en remplaçant le caractère précisé par une étoile.



voir fichier **remplace.html**

```
<html>
<body>
 <form action="remplace.php3" method="post">
 Veuillez saisir une phrase<input type="text" name="phrase">
 Veuillez saisir une lettre<input type="text" name="lettre" size=1>
 <input type=submit value=remplace><br>
 Le caractère que vous avez saisi sera remplacé dans la phrase par une *
 </body>
</html>
```

le script php

voir fichier **remplace.php3**

```
<?
/*ereg_replace(caractère à remplacer,caractère de remplacement,chaine concernée)*/
  echo ereg_replace($lettre,"*",$phrase);//
?>
```

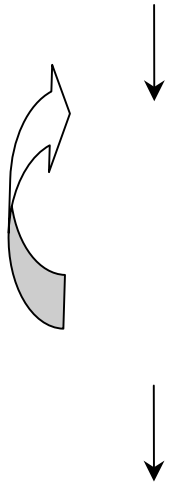
for :

Crée une boucle qui exécute le code x fois. Attention aux boucles infinies :

N.B: s'utilise lorsque l'on connaît à l'avance combien de fois on va "boucler".

```
for ([valeur initiale] ; [condition] ; [incrément]) {
code
}
```

- **valeur initiale** :valeur initiale de la variable compteur.
- **condition** : condition de sortie de la boucle.
- **incrément** : pour incrémenter ou décrémenter le compteur.



```
…
for(i=début ; i<= fin ; i=i+x)
    {…
     …
     ..
    }
```

On exécute le bloc d'instructions autant de fois que nécessaire pour que **i** passe de la valeur début à la valeur fin

(en augmentant de x à chaque fois)

Ce qui se trouve entre les parenthèses se lit de la manière suivante :

i = début; veut dire : en partant avec la valeur **début** au compteur i

i <= fin; veut dire : tant que le compteur aura une valeur inférieure ou égale à **fin**;

i=i+1 veut dire : en incrémentant (en faisant grimper en français normal) le compteur i de **x** à chaque fois.

While :

Crée une boucle qui répète l'exécution du code tant que la condition est vraie.

Si la condition est vraie dès le début, le programme n'entre pas dans la boucle.

N.B.: s'utilise quand on ne sait pas combien de fois on va "boucler". En général dans les instructions répétées il y en a forcément une qui, à un moment donné fera passer le test = Vrai (Sinon cela risque de ne jamais s'arrêter.)

while (condition) {

code

}

- **condition:** la condition de sortie de la boucle.

while (test)

{...

...

...

}

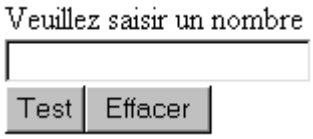
Si le test est vrai , on exécute le bloc d'instructions à l'intérieur du tant que, puis on remonte voir si le test est vrai ou faux

Si le test est faux, on passe à la suite

Exemple

Calcul d'une factorielle. Nous allons créer un formulaire qui demande la saisie d'un nombre entier.

voir fichier **factorielle.html**



Ce nombre une fois saisi est transmis à un script qui calcule sa factorielle (ex factorielle de 3 =3x2x1)

Script php de la factorielle

voir fichier **factorielle.php3**

```
<?
echo "Le nombre saisi est : $nbre<br>";
$cpt=$nbre;
echo "cpt vaut $cpt";
while($cpt>0){
$nbre=$nbre*$cpt--;
echo "nbr = $nbre<br>";
echo "cpt = $cpt<br>";
}
echo "La factorielle de ce nombre vaut $nbre";
?>
```

Principes de base :

PHP supporte les tableaux scalaires et les tableaux associatifs. En fait, il n'y a aucune différence entre les deux. Vous pouvez créer un tableau en utilisant la fonction **array(),** ou bien en affectant explicitement chacune des valeurs.

```
$a[0] = "abc";
$a[1] = "def";
$b["note"] = 13;                    // tableau "associatif"
```

Vous pouvez aussi créer un tableau en ajoutant simplement les valeurs à ce tableau.

```
$a[] = "hello";          // revient à    $a[2] == "hello"
$a[] = "world";          // revient à    $a[3] == "world"
```

Un tableau peut être trié en utilisant la fonction **sort()** (et assimilés) en fonction du type de classement que vous voulez.

Vous pouvez compter le nombre d'éléments qu'il y a dans un tableau en utilisant la fonction **count().**

```
echo count($a);                     //affichera 4
```

Vous pouvez vous déplacer à l'intérieur d'un tableau de 2 manières

- en connaissant la taille du tableau et en construisant une boucle qui permettra de se positionner sur chaque élément du tableau
  (cette technique ne peut être utilisée que pour les tableauxde type scalaire)
- en utilisant les fonctions **each(), next()** et **prev(**)
  (un peu comme on lit un fichier, cette technique peut être utilisée pour tout type de tableau, qu'il soit associatif ou scalaire)

Manipulation de tableau à une dimension :

Créer - Afficher un tableau (scalaire)

On peut créer le tableau classiquement:

```
$tab[0]=0;
$tab[1]=1;
$tab[2]=2;
```

Mais on peut créer un tableau par l'instruction **array**

```
$tab = array(0,2,4,6);
```

Pour parcourir les éléments du tableau, deux écritures sont possibles.

voir fichier **coursphp_15.php3**

```
for ($cpt=0;$cpt<count($tab);$cpt++) {
  $val=$tab[$cpt];
   print (" le $cpt ° elem du tableau vaut $val <BR>");
}
```

ou bien

```
for ($cpt=0;$cpt<count($tab);$cpt++) {
  $numelet=$cpt+1;
  print (" le $cpt  ° elem du tableau vaut $tab[$cpt] <BR>");
}
```

N.B: on a vu qu'll existe aussi une variation par rapport à la construction classique dite énumération, permettant de ne pas spécifier l'indice, celui-ci étant crée automatiquement par l'interpréteur php.

Ainsi au lieu décrire

```
$tab[0]=0;
$tab[1]=1;
$tab[2]=2;
```

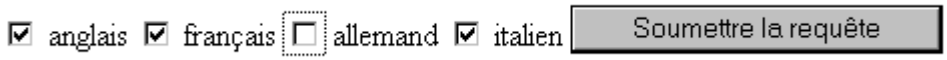
on peut se permettre d'écrire

```
$tab[]=0;
$tab[]=1;
$tab[]=2;
```

Ce qui permet des constructions assez "osées" comme dans l'exemple suivant:

Exemple

Nous allons créer un formulaire qui demande d'effectuer un ou plusieurs choix grâce à des cases à cocher..



p

pour obtenir

votre 1° choix gb
votre 2° choix fr
votre 3° choix it

Ces choix sont stockés dans un tableau qui sera automatiquement de la dimension correspondante au nombre de cases à cochée…. cochées !

Le formulaire HTML :

voir fichier **cretableau.html**

```
<html>
<body>
 <form name="toto" method="post" action="cretableau.php3">
 <input type="checkbox" name="choix[]" value="gb"> anglais
 <input type="checkbox" name="choix[]" value="fr"> français
 <input type="checkbox" name="choix[]" value="all"> allemand
 <input type="checkbox" name="choix[]" value="it"> italien
 <input type="submit">
 </form>
</body>
</html>
```

La saisie est ensuite transmise à un script qui affiche un tableau choix de dimension appropriée …

voir fichier **cretableau.php3**

```
<?
$nb=count($choix);
for ($cpt=0;$cpt<$nb;$cpt++){
echo "votre ".($cpt+1)."° choix ".$choix[$cpt]."<br>";
}
?>
```

Créer - Afficher un tableau (associatif)

Si le tableau est associatif, le parcours avec un indice numérique pose problème...

voir fichier **coursphp_15b.php3**

```
<?php
 $tabasso["nom"]="Jean";
 $tabasso["age"]=30;
 $tabasso["note"]=20;
 for ($cpt=0;$cpt<count($tabasso);$cpt++)
  $val=$tabasso[$cpt];
  print (" parcours scalaire le $cpt elem du tableau vaut $val <BR>");
 }
<?
```

$val=tabasso[$cpt] ne permet plus d'atteindre la valeur stockée dans le tableau !

Il faut savoir aussi que chaque tableau entretien un pointeur interne, qui est initialisé lorsque le premier élément est inséré dans le tableau.

Pour passer en revue proprement un tableau associatif il faut utiliser la fonction **each()** et **reset()**

each() : retourne la paire (clé/valeur) courante du tableau ***array*** et avance le pointeur de tableau. Cette paire est retournée dans un tableau de 4 éléments, avec les 4 clés prédéfinies désignée **0**, **1**, ***key***, ***value***.

Les éléments ***0*** et ***key*** contiennent le nom de la clé
Les éléments ***1*** et ***value*** contiennent la valeur.

Cela est justifié uniquement par le fait de pouvoir y accéder par une notation scalaire(0/1) ou une notation associative (key/value).

Si le pointeur interne de fichier est au delà de la fin du tableau, **each()** retourne faux. Après chaque **each()**, le pointeur de tableau est déplacé sur l'élément suivant, ou sur le dernier élément lorsqu'on arrive à la fin.

reset() : replace le pointeur de tableau ***array*** au premier élément.

voir fichier **coursphp_15b.php3**

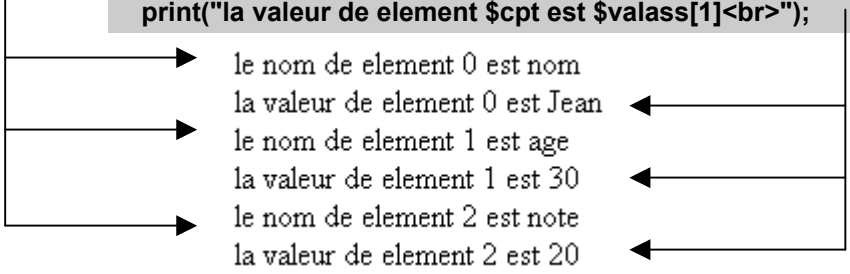
```
$nbval=count($tabasso);       // on récupère la dimension du tableau
for ($cpt=0;$cpt<$nbval;$cpt++) {
  $valass=each($tabasso);   // récupère l'élément courant et avance  pointeur
  // ici on peut traiter
  // l'élément du tableau
  print("le nom de element $cpt est $valass[0]<br>");
  print("la valeur de element $cpt est $valass[1]<br>");
}
reset($tabasso);                //replace le pointeur au premier élément.
?>
```

avec

```
print("le nom de element $cpt est $valass[0]<br>");
```

et avec

```
print("la valeur de element $cpt est $valass[1]<br>");
```



N.B: à la place de l'instruction **$valass[0]** on pourrait tout aussi bien écrire **$valass[key]**

N.B: à la place de l'instruction **$valass[1]** on pourrait tout aussi bien écrire **$valass[value]**

Pour parcourir les éléments d'un tableau on va avoir principalement 2 méthodes

1° méthode (solution) : trouver la taille du tableau puis parcourir classiquement tous les éléments du tableau que l'on lira à l'aide de each()

voir fichier **coursphp_15c.php3**

```
$nbval=count($tabasso);
 for ($cpt=0;$cpt<$nbval;$cpt++) {
   $valass=each($tabasso);
   $val=$valass[value];
   print (" 1° méthode : le [$cpt] elem du tableau vaut $val <BR>");
 }
```

2° méthode (solution) : sans connaître la taille du tableau utiliser le fait que si le pointeur interne de fichier est au delà de la fin du tableau, each() retourne faux

voir fichier **coursphp_15c.php3**

```
reset($tabasso);
 while($valass=each($tabasso)) {
   $val=$valass[value];
   print (" 2° méthode : elem du tableau vaut $val <BR>");
 }
```

N.B: La fonction **current()** ne fait que retourner l'élément courant pointé par le pointeur interne. Si le pointeur est au delà du dernier élément de la liste, **current()** retourne faux. Si le tableau à des éléments vides ou des zéros (0 ou "", la chaîne vide) alors cette fonction retournera false pour ces éléments. Il est donc impossible de déterminer si vous êtes réellement à la fin de la liste en utilisant cette fonction.

Tableau multidimensionnel :

Les tableaux à plusieurs dimensions sont extrêmement simples. Pour chaque dimension du tableau, vous ajoutez une nouvelle **[dim]** à la fin

Voila un tableau à 2 dimensions 2x2

```
$a[0] [0] = "def";
$a[0] [1] = "def";
$a[1] [0] = "def";
$a[1] [1] = "def";
```

Voila un tableau à 3 dimensions 4x2x3

```
$a[0] [0] [0] = "def";
$a[0] [0] [1] = "def";
$a[0] [0] [2] = "def";
$a[0] [0] [0] = "def";
$a[0] [1] [0] = "def";
$a[0] [0] [0] = "def";
$a[1] [0] [0] = "def";
$a[2] [0] [0] = "def";
$a[3] [0] [0] = "def";
```

Evidemment on peut mélanger des tableaux associatifs et scalaires dans les tableaux à plusieurs dimensions.

Manipulation de tableau multidimensionnel :

il suffit de mélanger les techniques vues précédemment et de ne pas se... mélanger les indices !

voir fichier **coursphp_15d.php3**

```
$tabasso[0]["nom"]="Jean";     // voila un tableau de 4 élèves
$tabasso[0]["age"]=30;
$tabasso[0]["note"]=20;
$tabasso[1]["nom"]="Pierre";
$tabasso[1]["age"]=31;
$tabasso[1]["note"]=21;
$tabasso[2]["nom"]="Michel";
$tabasso[2]["age"]=32;
$tabasso[2]["note"]=22;
$tabasso[3]["nom"]="Marie";
$tabasso[3]["age"]=33;
$tabasso[3]["note"]=23;
```

ceci est un **tableau scalaire de 4 élèves**, chaque élève étant un **tableau associatif de 3 éléments** "nom", "age" et "note"

Pour connaître la liste de tous les élèves, il faut parcourir tous les éléments du tableau scalaire, et lire la valeur du premier élément du tableau associatif

cela devrait donner ceci :

```
$nbval=count($tabasso);    // donne la taille de la 1° dimension, soit 4
 for ($cpt=0;$cpt<$nbval;$cpt++) {
  $valass=each($tabasso[$cpt]);
  $val=$valass[value];
  print ("le [$cpt] eleve du tableau est $val <BR>");
 }
```

si on est sur un élève, pour connaître toutes ses valeurs il faut parcourir tous les éléments du tableau associatif

cela devrait donner ceci :

```
reset($tabasso[0]);          //replace le pointeur de tableau array au premier élément.
 while($valass=each($tabasso[0])) {
  $val=$valass[1];
  print (" pour cet élève, on a les valeurs $val <BR>");
 }
```

Et donc maintenant on doit pouvoir se déplacer sur chaque élève (éléments du tableau scalaire) et visualiser toutes ses composantes (éléments du tableau associatif)

cela devrait donner ceci :

```
$nbval=count($tabasso);
 for ($cpt=0;$cpt<$nbval;$cpt++) {
  reset($tabasso[$cpt]);       //replace le pointeur au premier élément.
  while($valass=each($tabasso[$cpt])) {
  $val=$valass[1];
  print (" pour l'élève n° $cpt on a les valeurs $val <BR>");
  }
  print (" <BR>") ;
 }
```

ENVOYER UN MAIL

Rappels de principes :

PHP étant un langage consacré au Web, il possède bien évidemment des fonctions lui permettant de communiquer avec le "monde extérieur" à l'aide de fonctions standards. Le service le plus utilisé sur Internet étant la messagerie électronique, il est naturel que PHP permette d'envoyer des mails.

Fonction mail() :

int **mail (**string **email_destinataire,** string **sujet,** string **corps_message,** string **options)**

Le dernier champ est facultatif, on en parlera juste après. Dans un premier temps nous allons envoyer un email de base, en utilisant les 3 premiers paramètres

Dans un formulaire on donne les renseignements nécessaires :

voir fichier **coursphp_16.html**

```
<html>
<body>
<form method="post" action="coursphp_16.php3">
 <p>Votre Nom
  <input type="text" name="nom" size="50">
 </p>
 <p>Votre adresse
  <input type="text" name="votre_email" size="50">
 </p>
 <p>Votre commentaire
  <textarea name="commentaire" cols="50" rows="5"></textarea>
 </p>
 <p>
  <input type="submit" name="envoyer" value="Envoyer">
  <input type="reset" name="effacer" value="R&eacute;tablir">
 </p>
</form>
</body>
</html>
```

On note la "future" variable **votre_email**

Voilà le code du fichier traitant les données envoyées par le formulaire précédent :

voir fichier **coursphp_16.php3**

```
<html>
<body>
<h4>Merci de votre message <?echo $nom;?></h4>
<?
echo "<p>Vous allez recevoir un email à l'adresse suivante : $votre_email</p?>";
mail("$votre_email",
  "Réponse à votre courrier ",
  "$nom a laissé le commentaire suivant :  \n\n$commentaire" );
?>
</body>
</html>
```

qui donne à l'éxecution quelque chose du genre

Merci de votre message bertrand

Vous allez recevoir un email à l'adresse suivante : michel@cabare.net

N.B: Chez certains hébergeurs (dont Free Online) la **fonction mail est désactivée** car elle permet de simuler un envoi de mail à partir de n'importe quelle adresse. Elle est alors remplacée par une fonction propre à l'hebergeur**…"** ***permettant de vérifier que vous laisser une trace au niveau du champs from dans les mails que vous envoyez (de manière à identifier un mail facilement)"***

Une fonction personnalisé email() :

int **email (**string **from,** string **email_destinataire,** string **sujet,** string **corps_message,** string **options)**

il s'agit essentiellement de rajouter un premier paramètre obligatoire **from**, (et d'ailleurs sur les hebergeurs professionnels on ne pourra indiquer ici qu'une adresse mail "valable" pour le compte de domaine hebergé)
Les options / en-têtesLe champ options de la fonction mail permet d'ajouter une en-tête au message que l'on envoie. On peut par exemple y mettre la date, le logiciel utilisé pour envoyer l'email ou encore l'adresse de retour... Voilà un exemple d'en-tête à utiliser lors d'un envoi de mail :

```
$from_email  = "moi@truc.fr";
$entetemail  = "From: $from_email \n"; // Adresse expéditeur
$entetemail .= "Cc: \n";
$entetemail .= "Reply-To: $from_email \n"; // Adresse de retour
```

```
mail("$votre_email",
  "Réponse à votre courrier ",
  "$nom a laissé le commentaire suivant :  \n\n$commentaire"
  "$entetemail"
);
```

Remarque

From :

CC:

Reply-To:

sont des mots réservés

Calcul de date - time :

En php, on décompte le temps de manière un peu particulière, c'est le nombre de secondes découlées depuis le 1er Janvier 1970.

Ce temps est souvent appellé aussi "temps UNIX" !

Fonction time() :

int **time (**void**)**

La fonction time() retourne le nombre de secondes écoulées depuis le 1er Janvier 1970. elle est très souvent utilisée lors des manipulations de dates.

voir fichier **coursphp_17.php3**

```
<?php
echo "Il s'est écoulé ". time() . " secondes depuis le 1er Janvier 1970";
?>
```

Convertion de date - mktime :

Pour convertir une date quelconque en nombre de secondes depuis le 1er Janvier 1970, nous avons la fonction mktime dans laquelle on précise, l'heure, la minute, la seconde, le mois, le jour, l'année.

Fonction mktime() :

int **mktime (**int **heure,** int **minute,** int **seconde,** int **mois,** int **jour,** int **année,)**

N.B: L'ordre des paramètres est à l'américaine mois,jour,année et non jour,mois,année.

Exemple comment de secondes se sont écoulées depuis le 1 janvier 2000

voir fichier **coursphp_17.php3**

```
<?php
echo "1 janvier 2000, c'était il y a ". mktime(0,0,0,1,1,2000) . " secondes ";
?>
```

Contrôler de date - checkdate

Pour contrôler la validité d'une date issue, par exemple d'un formulaire, vous pouvez avoir recours à la fonction **checkdate(**).

Fonction checkdate() :

int **checkdate (**int **mois,** int **jour,** int **année)**

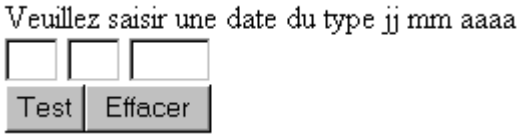
La fonction checkdate retourne True (1) si la date est valide (année entre 0 et 32767, mois entre 1 et 12…), et la valeur False (0) sinon.

N.B: L'ordre des paramètres est à l'américaine mois,jour,année et non jour,mois,année.

Exemple

Dans ce formulaire on demande la saisie d'une date.

Cette date sera transmise à un script qui vérifiera sa validité et affichera un message en conséquence



voir fichier **coursphp_17b.html**

```
<HTML>
<BODY>
<FORM action=coursphp_17b.php3 method=post>
Veuillez saisir une date du type jj mm aaaa<br>
<input type=text name=jour size=2>
<input type=text name=mois size=2>
<input type=text name=annee size=4><br>
<input type=submit value=Test><input type=reset value=Effacer>
</form>
</body>
</html>
```

avec un script php de la forme

voir fichier **coursphp_17b.php3**

```
<?
if (checkdate($mois,$jour,$annee)) {
echo "La date $jour - $mois - $annee existe bien";
} else {
echo "La date $jour - $mois - $annee n'est pas valide";
}
?>
```

La fonction getdate :

Fonction getdate() :

array **getdate (**int **date)**

La fonction renvoit un tableau associatif contenant les informations de date et heure de la variable date avec les champs suivants :

- "seconds secondes
- "minutes" minutes
- "hours" heures
- "mday" jour du mois
- "wday" jour semaine, numérique. 0: dimanche jusqu'à 6: samedi
- "mon" mois, numérique
- "year" année, numérique
- "yday" jour de l'année, numérique; i.e. "299"
- "weekday" jour de la semaine, texte complet (en anglais); i.e. "Friday"
- "month" mois, texte complet (en anglais); i.e. "January"

Exemple avec getdate()

voir fichier **coursphp_17c.php3**

```
<?php
 $aujourdhui = getdate(time()); //   $aujourdhui est un tableau associatif
 $mois = $aujourdhui['month']; //   dont on prends ici l'élément [month]…
 $mjour = $aujourdhui['mday'];
 $annee = $aujourdhui['year'];
 echo "$mjour $mois $annee";
?>
```

Affichage et formatage d'une date strftime()

Pour afficher une date, vous pouvez utiliser la fonction **strftime**

Fonction strftime() :

string **getdate (**string **format,** int **date)**

avec pour paramètre une chaîne de caractère **format** indiquant la manière sous laquelle vous voulez représenter la date.

Ce format est donné par une chaîne de caractères comportant quelques caractères précédés d'un signe % ayant une signification particulière (d pour jour, m pour mois, y pour année, etc...) et des caractères choisis librement (ici, le slash, l'espace, les 2 points).

voir fichier **coursphp_17d.php3**

```
<?
echo "Affichage au format jour/mois/annee heure:minute:seconde ";
strftime("%d/%m/%y %H:%M:%S") . "<br>";
?>
```

Donc à partir du moment où vous connaissez les lettres "clés", vous pouvez donner libre court à votre imagination.

Liste des principaux formats :

%y	année (sur 2 chiffres)
%Y	année
%B	mois en toutes lettres
%m	mois
%A	jour en toutes lettres
%d	jour
%u	numéro du jour dans la semaine (1=Lundi)
%H	heure (sur 24 heures)
%I	heure (sur 12 heures)
%M	minute
%S	seconde
%j	numéro du jour dans l'année
%V	numéro de la semaine dans l'année

Par défaut la fonction strftime() formate la date et heure courante mais vous pouvez spécifier un second paramètre pour afficher une date calculée. Ce paramètre est exprimé en secondes.

voir fichier **coursphp_17d.php3**

```
echo "Dans 15 jours nous serons le " .
strftime("%d/%m/%y",time()+15*24*3600);
```

ou

```
echo "Le 15 Aout 2001 tombe un " .
strftime("%A",mktime(0,0,0,8,15,2001)) ;
```

on peut alors stocker ces valeurs simplement dans des variables tout simpement

```
// la variable $an contient la valeur 01 si nous sommes en 2001
$an=strftime("%y");
```

ou

```
// la variable $mois contient la valeur Saturday si nous sommes samedi
$mois=strftime("%B");
```

Formatage en français setlocale()

Si vous voulez afficher le jour ou le mois en toutes lettres, vous risquez de l'avoir en anglais. Si ce n'est pas ce que vous désirez, il convient d'utiliser la commande setlocale afin de préciser la langue d'affichage.

```
<?
setlocale("LC_TIME","en_US");
echo "En Anglais " . strftime("%A %d %B %Y") . "<br>";
setlocale("LC_TIME","fr_FR");
echo "En Français " . strftime("%A %d %B %Y") . "<br>";
?>
```

En Anglais Friday 21 September 2001

En Français vendredi 21 septembre 2001

La langue est définie par 2 lettres (généralement) suivi d'un underscore puis 2 autres lettres. Les premières lettres indiquant la langue proprement dit et les 2 suivantes le pays (on peut ainsi distinguer fr_FR de fr_CA).

Affichage et formatage d'une date : date()

La fonction date() a à peu près le même rôle que strftime mais les caractères "clés" sont différents. **date()** retourne une date sous forme d'une chaîne, au format donné par la chaîne format, sans possibilité de donner des formats nationaux (locaux).

Quelques formats :

a ou A	"am" (matin) ou "pm" (après midi)
d	Jour du mois, sur deux chiffres (éventuellement avec un zéro) : "01" à "31"
D	Jour de la semaine, en trois lettres (et en anglais) : par exemple "Fri" (pour Vendredi)
F	Mois, textuel, version longue; en anglais, i.e. "January" (pour Janvier)
g	Heure, au format 12h, sans les zéros initiaux i.e. "1" à "12"
G	Heure, au format 24h, sans les zéros initiaux i.e. "0" à "23"
h	Heure, au format 12h, "01" à "12"
H	heure, au format 24h, "00" à "23"
i	Minutes; "00" à "59"
I	(i majuscule) "1" si l'heure d'été est activée, "0" si heure d'hiver .

j	Jour du mois sans les zéros initiaux: "1" à "31"
l	('L' minuscule) Jour de la semaine, textuel, version longue; en anglais, i.e. "Friday" (pour Vendredi)
L	Booléen pour savoir si l'année est bissextile ("1") ou pas ("0")
m	Mois; i.e. "01" à "12"
M	Mois, en trois lettres (et en anglais) : par exemple "Apr" (pour Avril)
n	Mois sans les zéros initiaux; i.e. "1" à "12"
s	Secondes; i.e. "00" à "59"
t	Nombre de jours dans le mois donné, i.e. "28" à "31"
U	Secondes depuis une époque
w	Jour de la semaine, numérique, i.e. "0" (Dimanche) to "6" (Samedi)
Y	Année, 4 chiffres; i.e. "1999"
y	Année, 2 chiffres; i.e. "99"
z	Jour de l'année; i.e. "0" à "365"

Les caractères non reconnus seront imprimés tels quel.

Quelques exemples :

```
<?php
/* Aujourd'hui, le 12 Mars 2001, 10:16:18 pm */
  $aujourdhui = date("F j, Y, g:i a");              // March 12, 2001, 10:16 pm
  $aujourdhui = date("m.d.y");                      // 03.12.01
  $aujourdhui = date("j, m, Y");                    // 12, 3, 2001
  $aujourdhui = date("Ymd");                        // 20010312
  $aujourdhui = date("H:i:s");                     // 10:16:18
// notation française
$aujourdhui = date("d/m/y");                       // 12/03/01
$aujourdhui = date("d/m/Y");                       // 12/03/2001
?>
```

si vous avez le temps, jetez un petit coup d'oeil sur le formulaire et le script suivant…. duree.html et duree.php3 !

LES FICHIERS

Principes :

Lorsque l'on n'a pas accès à une base de données, il faut utiliser des fichiers pour y stocker des données. En php la création ou la lecture de fichiers est possible, avec une multitude de fonctions...

il faut bien noter que lorsque l'on sera sur un serveur ftp, le problème des droits qu'un utilisateur (ou qu'un script php...) peut avoir sur un fichier conditionnera l'accès au fichier proprement dit

les fonctions de base sont

la fonction **fopen()**, qui permet d'ouvrir un fichier, pour le lire ou y écrire

la fonction **fclose()**, qui permet de fermer un fichier

la fonction **fgets()**, qui permet de lire dans un fichier ouvert

Ouverture de fichier : fopen()

Fonction fopen() :

int **fopen (**string **nomfichier,** string **mode)**

Le paramètre nomfichier peut prendre plusieurs valeur parmis les suivantes:

- Si **nomfichier** commence par "**http://**" (insensible à la casse), une connexion HTTP 1.x est ouverte , et un pointeur sur la réponse fournie est retourné.
- Si **nomfichier** commence par "**ftp://**" (insensible à la casse), une connexion FTP est ouverte , et un pointeur sur la réponse fournie est retourné.. Vous pouvez ouvrir des fichiers en lecture seulement, ou en écriture seulement (le full duplex n'est pas supporté).
- Si **nomfichier** commence par n'importe quoi d'autre, PHP tentera de lire ce fichier dans le système local, et un pointeur sur le fichier ouvert sera retourné.

Si l'ouverture échoue, **fopen()** retourne **FALSE (0)**.

Le paramètre **mode** peut prendre les valeurs suivantes :

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

De plus, ***mode*** peut contenir la lettre 'b'. Cette option n'est utile que sur les systèmes qui font la différence entre les fichiers binaires et les fichiers textes (en bref, c'est inutile sous Unix). S'il n'est pas nécessaire, il sera ignoré.

Fermer un fichier : fclose()

La fonction **fclose()**, permet de fermer un fichier

Fonction fclose() :

int **fclose (**int **ptrfichier)**

l'entier prtfichier est l'entier retourné précédemment lors de l'ouverture du fichier.

Exemple :

voir fichier **coursphp_18.php3**

```
$fich=fopen("essai.txt","a+");
echo "le pointeur retourné est $fich";
fclose($fich);
```

Die :

void die (string **message)**

Cette fonction affiche la chaîne passée en paramètre, puis termine l'exécution du script. Elle ne retourne rien de plus.

```
<?php
$file = fopen ("filename", 'r') or die("impossible d'ouvrir le fichier");
?>
```

Ecriture dans un fichier : fputs() - fwrite()

La fonction **fputs()**, permet d'écrire dans un fichier

Fonction fputs() :

int **fputs (**int **ptrfichier,** string **texte,** int **taille)**

Ecrit le contenu de la chaîne **texte** dans le fichier pointé par **ptrfichier**. Si la longueur **taille** est fournie, l'écriture s'arrêtera après **taille** octets, ou à la fin de la chaîne (le premier des deux).

fputs() est un alias de fwrite(), et lui est identique en tout point.

Notez que **taille** (qui représente le nombre de caractères à écrire) est un paramètre optionnel, et s'il n'est pas spécifié, toute la chaîne est écrite.

Exemple :

On veut ouvrir le fichier avec le droit en écriture, en le créant s'il n'existe pas et en se positionnant à la fin... "**a**" semble plus indiqué que w qui "réinitialise le fichier".

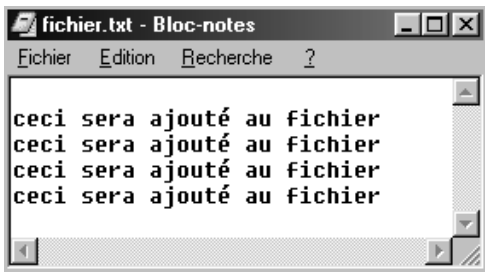
```
<?
$fp = fopen("fichier.txt","a");
if ($fp == false )
{
 print ("on ne peut pas ouvrir le fichier !");
}
else
{
 fputs($fp,"\n");
 fputs($fp,"ceci sera ajouté au fichier");
 fclose($fp);
 print ("le fichier a été crée - mis à jour !");
}
?>
```

voir fichier **coursphp_18b.php3**

2°) on ajoute la chaîne "ceci sera ajouté au fichier" dans le fichier

1°) on inscrit un retour chariot à la fin du fichier

Après 4 execution, on pourrait avoir le fichier texte suivant (par exemple…)



Lecture dans un fichier : fgets() feof() fseek()

La fonction **fgets()**, permet de lire un fichier sur une ligne…

Fonction fgets() :

string **fgets (**int **ptrfichier,** int **taille)**

fgets() retourne la chaîne lue jusqu'à la longueur **taille** - 1 octet, ou bien la fin du fichier, ou encore un retour chariot (le premier des trois qui sera rencontré).

ptrfichier correspond à l'identifiant récupéré lors de l'ouverture du fichier.

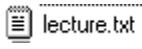
exemple :

```
<?
$fp =@ fopen("lecture.txt","r");
//$fp =@ fopen("http://www.cabare.net/php/visiteurs.txt","r");
if ($fp == false )
{
 print ("on ne peut pas ouvrir le fichier !");
}
else
{
 $donnee = fgets($fp,255);
 fclose($fp);
 print ("le fichier contient $donnee");
}
?>
```

voir fichier **coursphp_18c.php3**

on ferme le fichier

on lit la 1° ligne du fichier à concurrence de 255 octets

on ouvre le fichier nommé "**lecture.txt**" en lecture seule (que l'on peut créer précédemment via notepad  lecture.txt 0 Ko Document texte …

Si le fichier contient plusieurs lignes, on pourrait alors effectuer une lecture en utilisant la fonction feof() qui permet de tester la fin de fichier.

Fonction feof() :

int **feof(**int **ptrfichier)**

feof()qui retourne TRUE (1) si le pointeur est à la fin du fichier, ou si une erreur survient, sinon, retourne FALSE (0) et incremente alors la position courante.

ptrfichier correspond à l'identifiant récupéré lors de l'ouverture du fichier.

Exemple

```
<?
$fp = @fopen("lecture2.txt","r");
if ($fp == false )
{
 print ("on ne peut pas ouvrir le fichier !");
}
else
{
 while (!feof($fp)) {
 $donnee = fgets($fp, 255);
 print (" $donnee <br>");
 }
 fclose($fp);
 print ("le fichier a été lu");
}
?>
```

voir fichier **coursphp18d.php3**

on lit une ligne à concurrence de 255 octets

tant que l'on n'est pas à la fin du fichier

Dans le cas d'une utilisation un peu classique, on peut alors avoir besoin de positionner le pointeur qui parcoure le fichier grace à fseek(), pour pouvoir repartir au début d'un fichier.

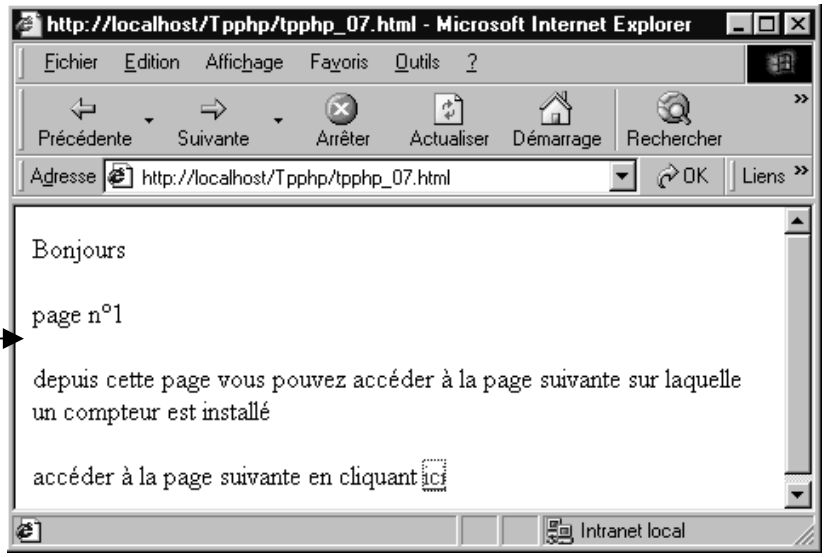
Fonction fseek() :

int **feof(**int **ptrfichier)**

fseek() retourne 1 si le pointeur est correctement repositionné en début de fichier et retourne –1 sinon.

EXEMPLE : si on veut savoir combien de fois on accède à une page, il suffit de faire exécuter sur la page en question un bout de code php permettant d'incrémenter une variable stockée dans un fichier

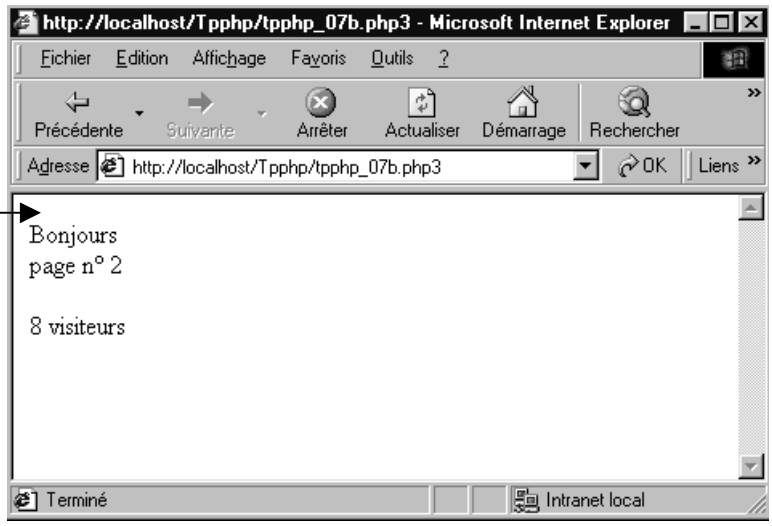
voir fichier **fichiercompte.html**



Depuis cette page

on accède à cette page

sur laquelle on compte le nombre de fois que l'on y est allé



la page html est la suivante :

```
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<p>Bonjours </p>
<p>page n&deg;1</p>
<p>depuis cette page vous pouvez acc&eacute;der &agrave; la page suivante sur
 laquelle un compteur est install&eacute;</p>
<p>acc&eacute;der &agrave; la page suivante en cliquant <a href="fichiercompte.php3">ici</a></p>
</BODY>
</HTML>
```

Le code php étant

voir fichier **fichiercompte.php3**

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<?
 $fp = fopen("compteur.txt","r+");          // ouvre un fichier existant
 if ($fp == 0)                              // si non existant
 {
    $fp = fopen("compteur.txt","w+");       // création du fichier (et raz)
    fputs($fp,0);                           // met 0 dedans
    fclose($fp);                            // fermeture
    $fp = fopen("compteur.txt","r+");       // ouverture du fichier existant
 }
 $nbvisites = fgets($fp,11);                // lecture valeur actuelle
 $nbvisites++;                              // incrementation
 fseek($fp,0);                              // repositionnement en début
 fputs($fp,$nbvisites);                     // recriture nouvelle valeur
 fclose($fp);
 print("$nbvisites visiteurs");
?>
</BODY>
</HTML>
```

N.B: ce compteur est absolument indépendant des cookies, (voir chap suivant) et marchera donc toujours….

Pourquoi une Base de Données ? :

On a vu que l'on était capable d'écrire en php dans des fichiers texte, et donc de pouvoir stocker et gérer des résultats...

C'est bien sûr un premier moyen de gérer des données, mais très vite les points suivant montrent les limites d'une gestion de donnée par fichier texte:

- Rigidité du format : ainsi que du contenu qui ne peut être que simple...
- Lourdeur d'accès : en pratique il faut chaque fois ouvrir le fichier, parcourir les lignes, effectuer des tests, avec des formats codifiés et complexes à traiter...
- Manque de sécurité, tout le monde peut accéder à un fichier texte...
- Manque de gestion des accès simultanés...

Les apports d'une base de données sont sans commune mesure avec la difficulté d'apprentissage, car en effet on peut noter que :

- spécificité du format : qui peut être surpassée par l'accès à des driver ODBC mais qui est optimisé pour le stockage de données diverses et leur manipulation
- primitives d'accès : permettant tout un tas de manipulations spécifiques rapides (positionnement, ajouts, recherche, tri...)
- Sécurité accrue : seul le serveur de la base de données peut accéder aux tables de données
- Gestion des accès simultanés

Principe de ODBC :

Php prend en charge une vaste gamme de bases de Données, mais il supporte également plusieurs variétés d'ODBC (Open Data Base Connectivity), ce qui en fait un choix incontournable pour la gestion de bases de données sur un site Web

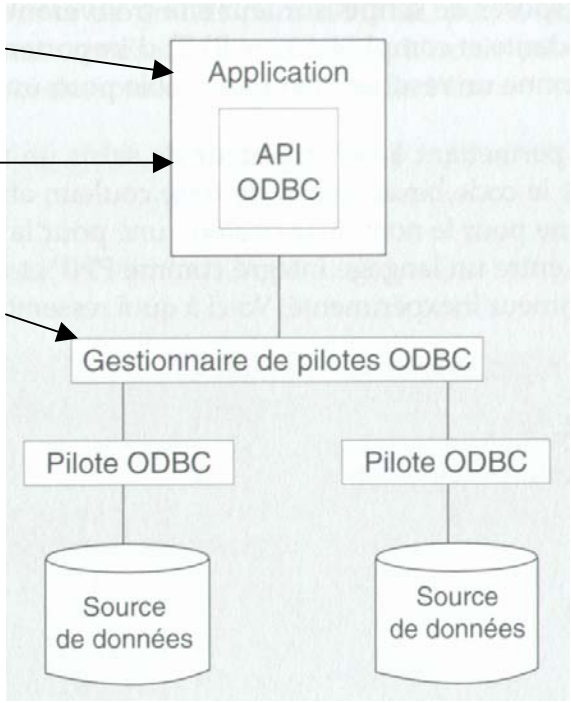
le principe du fonctionnement d'ODBC est le suivant :

Lorsque l'on écrit un code dans un langage donné (ici php)

on utilise des fonctions dites standard ODBC

Ces dernières ont recours de façon interne à du code écrit dans le langage natif de la base de données à piloter, mais accessible simplement via l'installation d'un pilote ODBC...

et l'on peut ainsi interfacer une base de donnée ACCESS avec Php...



L'avantage de ce système est la non spécificité de l'écriture pour un type de base de données.

L'inconvénient de ce système est la relative performance des instructions qui souffrent de deux défauts majeurs:

- forcément génériques ODBC et donc non optimisées pour telle ou telle base
- leur traduction en langage natif de la base de donnée à piloter est aussi une perte de temps certain

Prise en charge native :

Php prend en charge une vaste gamme de bases de Données, ce qui va permettre de pouvoir écrire des petits programmes rapides et efficaces

Le langage supporte les SGBD suivants:

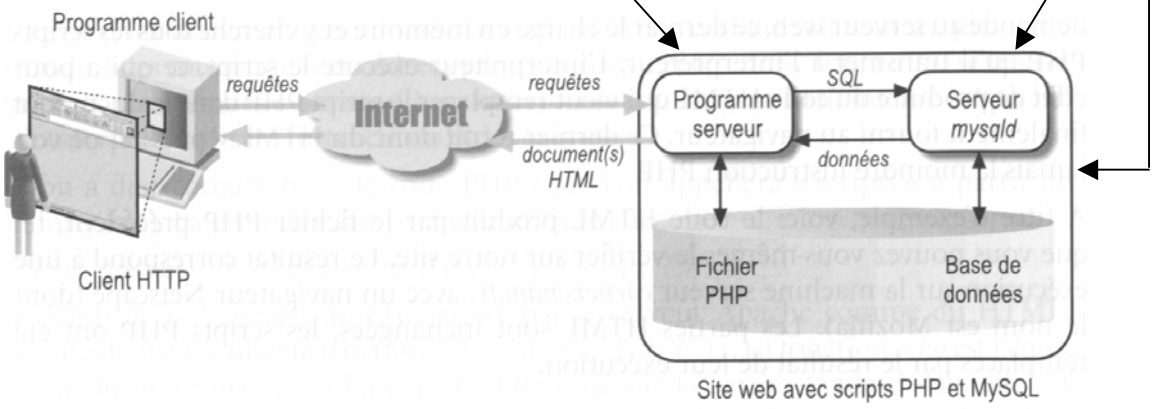
Adabas D dBase Empress FilePro Informix Interbase

mSQL MySQL Oracle PostgreSQL Solid Sybase

Velocis Unix dbm

Php permet d'utiliser un serveur MYSQL, travaillant avec des primitives très proches de celles du fameux langage d'accès aux bases de données SQL (Structured Query Langage)

Lorsque sur un **serveur Web** (dans notre cas un serveur local Apache, mais à priori cela peut être un serveur Web quelconque...) on implémente un **serveur MYSQL** permettant d'interroger et de manipuler une base de données MYSQL, si on peut écrire des programmes en **php** contenant les primitives d'appels au serveur, alors notre script devient un "**Programme serveur**" permettant de générer des pages HTML pour un **client via internet**...



Le principe fondamental d'utilisation de MySQL est de créer une base, dans laquelle on pourra stocker différentes tables.

Nous travaillerons essentiellement à partir de l'environnement Easyphp, qui permet de générer des commandes MySQL à partir d'assistant... un cours sur MySQL dépassant largement le cadre de notre propos, et nos ambitions !

Dans notre cas on peut donner le nom que l'on souhaite à la base, et l'on peut même créer plusieurs bases différentes. Par contre chez certains hébergeurs, la base MYSQL est déjà crée avec un nom imposé (pour des raisons de sécurité...)

Les noms portés par les bases de données, les tables, les index, les colonnes et les alias suivent tous les mêmes règles dans **MySQL**:

- Un nom est constitué de caractères alphanumériques et ``_'' et ``$''.
- Le nom d'une base de données, d'une table, d'un index ou d'une colonne peut avoir jusqu'à 64 caractères.
- Un nom peut commencer avec n'importe quel caractère autorisé. En particulier, un nom peut commencer avec un nombre (ce qui n'est pas toujours le cas dans de nombreuses bases de données). Cependant, un nom ne peut pas contenir uniquement des nombres.
- Il est interdit d'utiliser le point dans les noms, car il est déjà utilisé pour spécifier les noms des colonnes

Environnement phpMyadmin (MySQL) :

Pour accéder à cet environnement il suffit de lancer le serveur Apache et l'interpréteur MySQL,

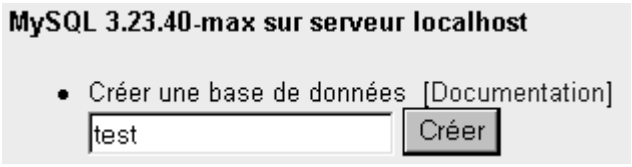
Serveur Apache et serveur MySql lancés

Cliquez avec le bouton droit puis choisissez Web local

Cliquez sur PhpMyadmin

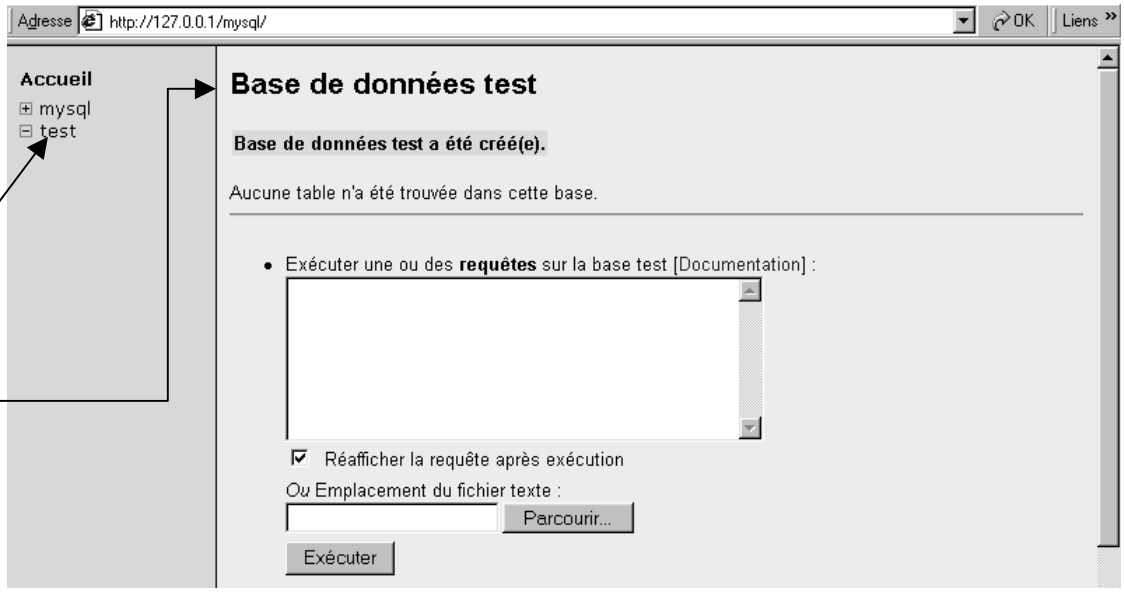
Créer une Base :

Il suffit simplement de taper le nom à donner à la base, puis demander **Créer**



Nous allons créer une base nommée test

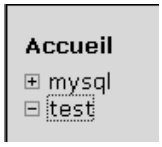
On obtient normalement



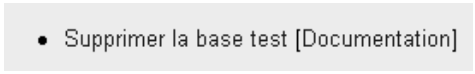
Dans laquelle notre nouvelle base **test** apparaît...

Supprimer une Base :

étant positionné sur la base à supprimer, (pour nous ici test)

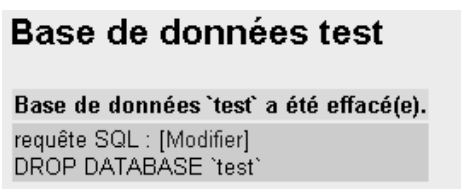


il suffit de demander **Supprimer la base test**



de confirmer notre ordre

et de voir l'instruction MySQL qui est générée automatiquement par notre assistant...



ici DROP DATABASE test

Recréons une base **essais**

Créer une Table :

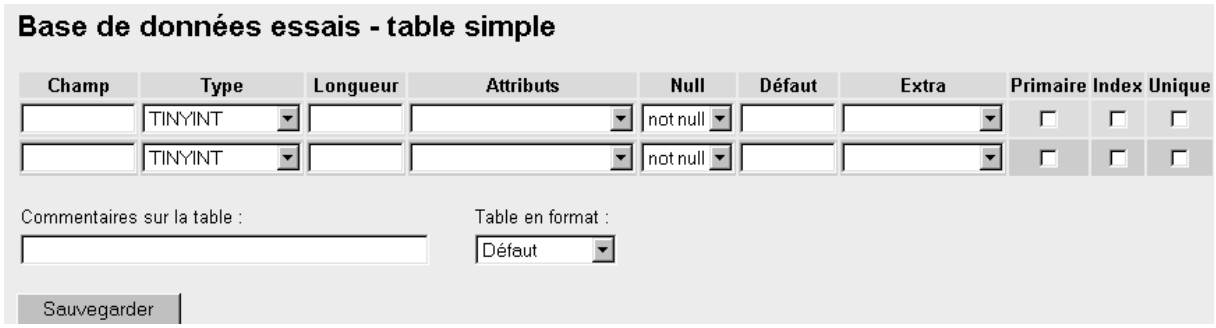
Une table, c'est l'élément constitutif de base d'une base de donnée Php.

Imaginons de devoir créer une table nommée **simple** pour y stocker par individu les 2 renseignements suivants: **nom** – **prenom**

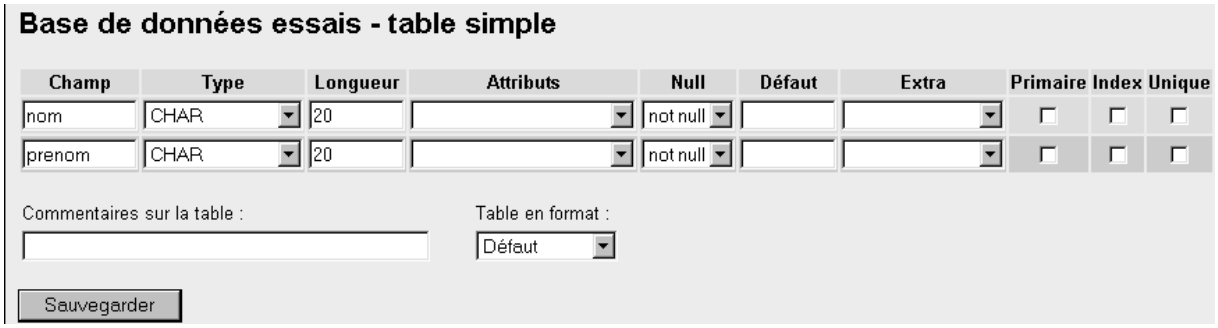
table **simple** de **2** champs

on demande Exécuter

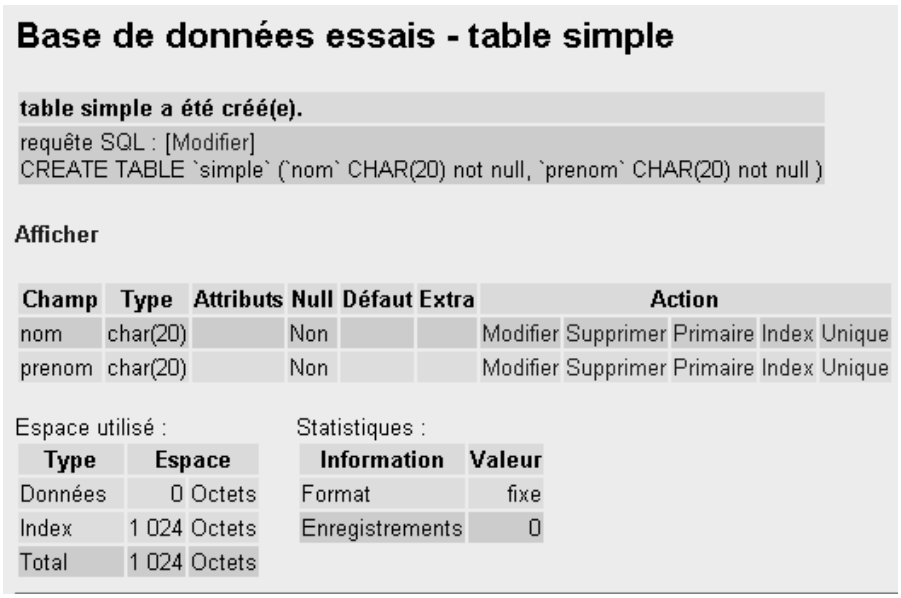
ce qui amène alors



Si on saisit les valeurs suivantes



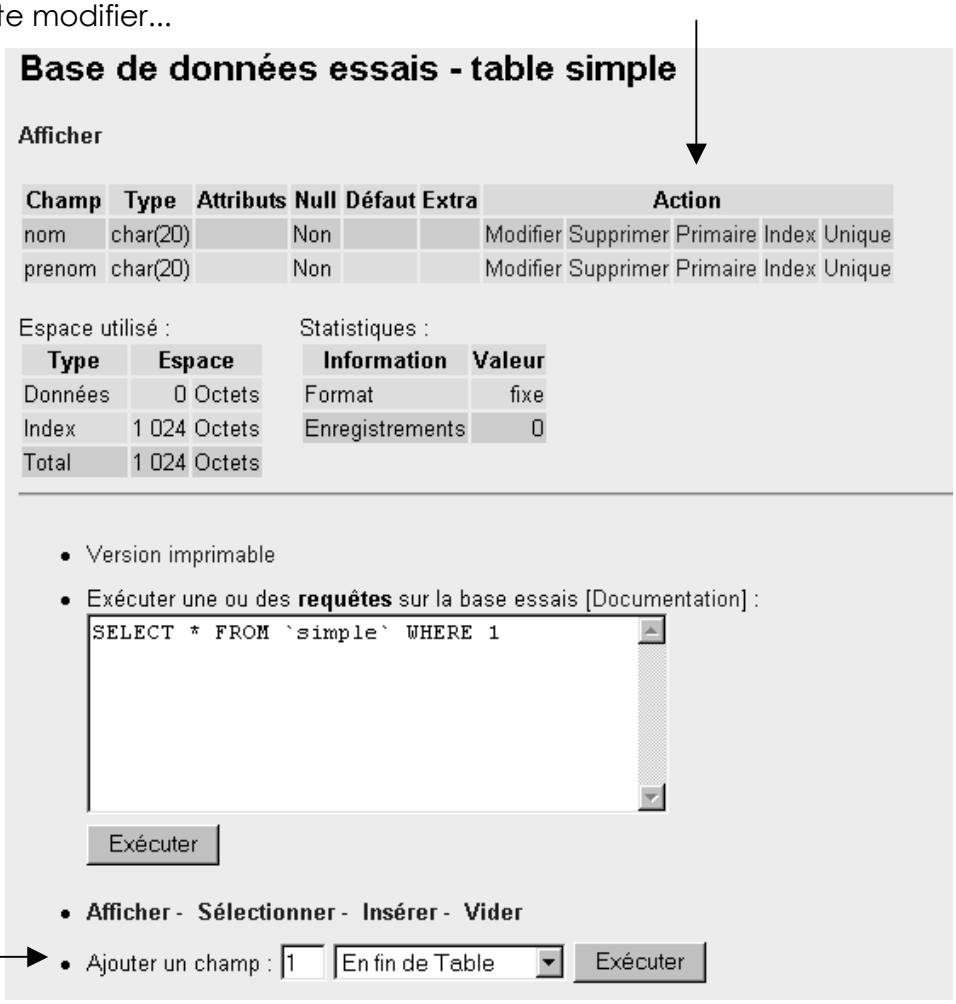
On obtient alors



Il est intéressant de constater que le logiciel PhpMyAdmin nous permet de visualiser les instructions Sql correspondant aux manip réalisées.

Modifier une Table :

On peut à tout moment modifier la structure d'une table crée par l'environnement en demandant une des **Action** sur le champ que l'on souhaite modifier...



Base de données essais - table simple

Afficher

Champ	Type	Attributs	Null	Défaut	Extra	Action				
nom	char(20)		Non			Modifier	Supprimer	Primaire	Index	Unique
prenom	char(20)		Non			Modifier	Supprimer	Primaire	Index	Unique

Espace utilisé :

Type	Espace
Données	0 Octets
Index	1 024 Octets
Total	1 024 Octets

Statistiques :

Information	Valeur
Format	fixe
Enregistrements	0

- Version imprimable
- Exécuter une ou des **requêtes** sur la base essais [Documentation] :

```
SELECT * FROM `simple` WHERE 1
```

Exécuter

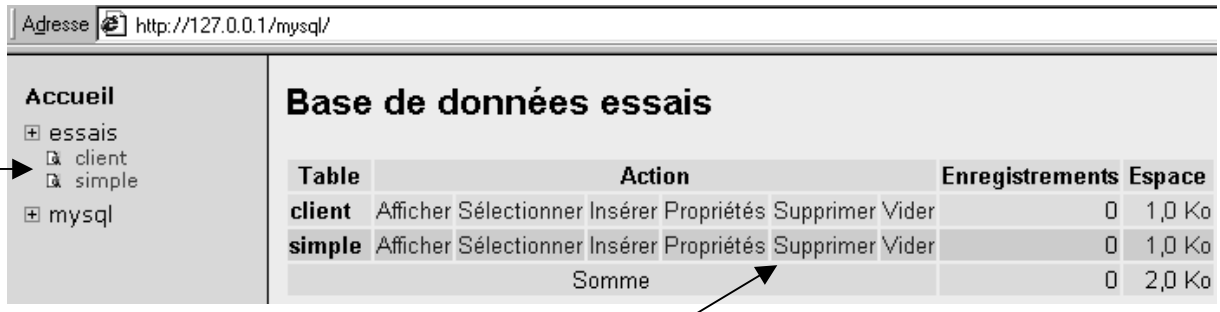
- **Afficher - Sélectionner - Insérer - Vider**
- Ajouter un champ : 1 En fin de Table Exécuter

ou en demandant d'ajouter un champ dans la table

Supprimer une Table :

Si l'on se place sur la base de données qui contient la table (ici **essais**), on voit apparaître sur la gauche tous ses éléments.

la base de données **essais** apparaît avec ici 2 tables **clients** et **simple**...



Adresse http://127.0.0.1/mysql/

Accueil
- essais
  - client
  - simple
- mysql

Base de données essais

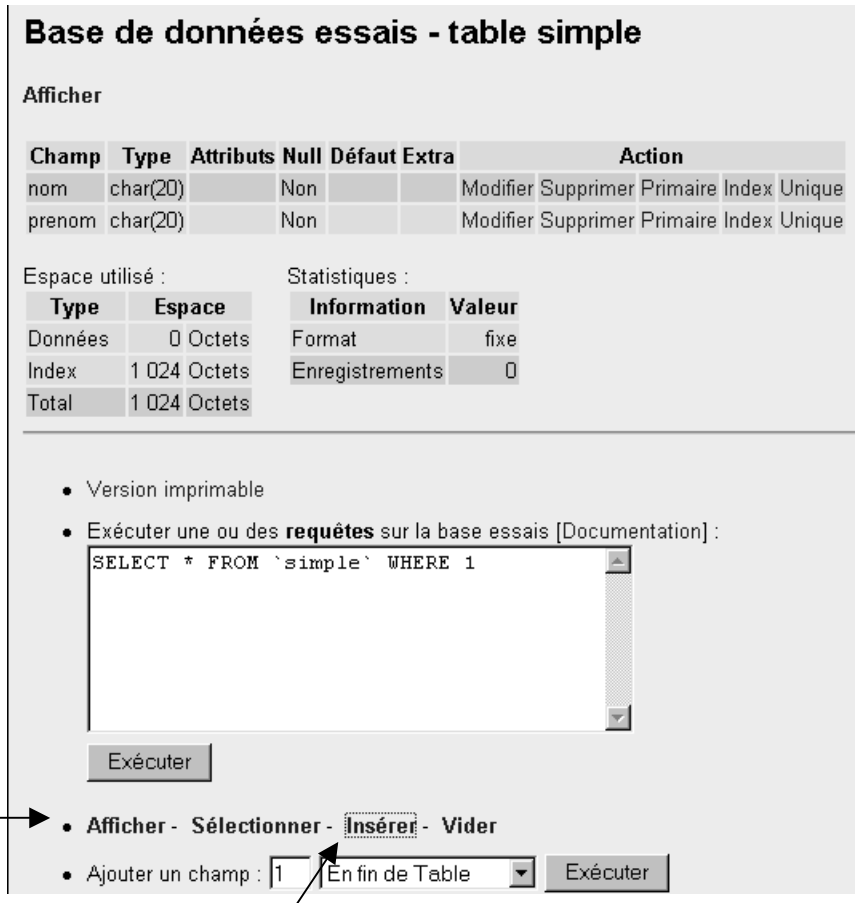
Table	Action						Enregistrements	Espace
client	Afficher	Sélectionner	Insérer	Propriétés	Supprimer	Vider	0	1,0 Ko
simple	Afficher	Sélectionner	Insérer	Propriétés	Supprimer	Vider	0	1,0 Ko
Somme							0	2,0 Ko

et on peut les manipuler facilement

Ajouter-Modifier un enregistrement :

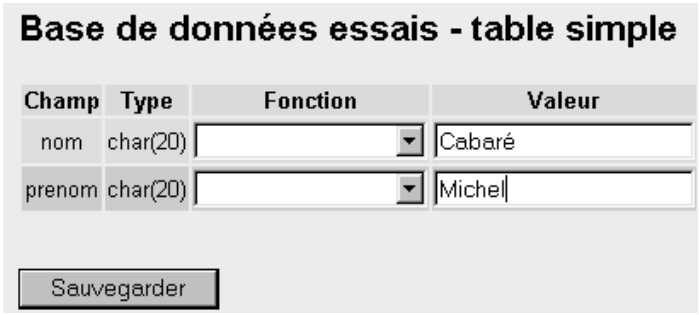
Bien sûr, à terme il sera plus intéressant de faire ce genre de manipulation via php, mais le faire via l'assistant permet d'être sûr du contenu de la table, pour vérifier ensuite notre programme construit en php...

étant positionné sur la table **simple**...

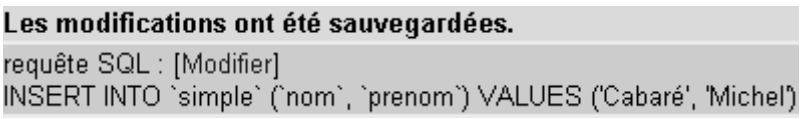


Insérer permet d'ajouter des enregistrements dans la table

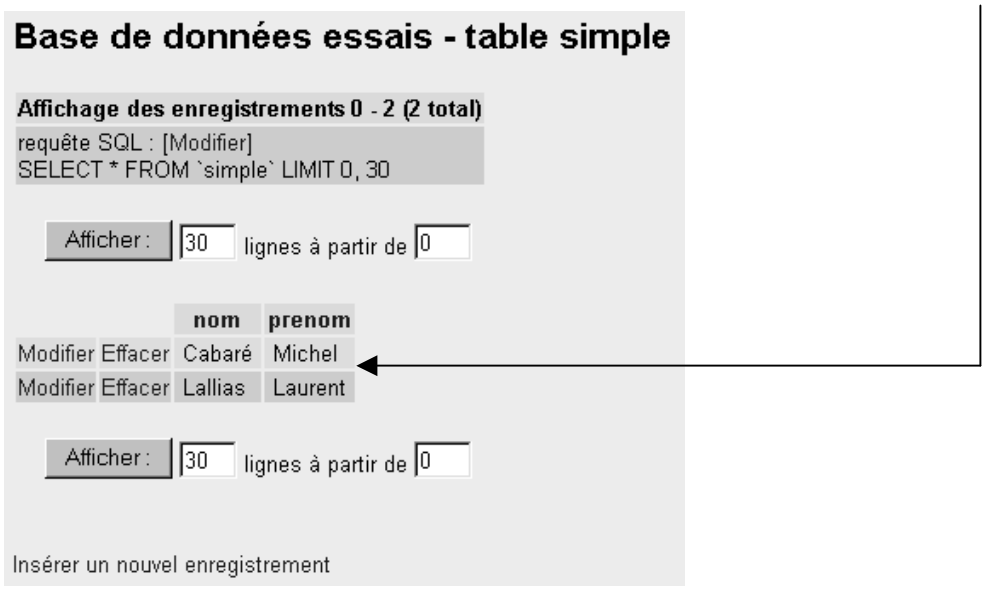
Nous allons ajouter 2 enregistrements : Cabaré Michel et Lallias Laurent



un clic sur Sauvegarder permet d'enregistrer la saisie. On voit ci dessous l'instruction SQL générée.



Afficher permet de visualiser les enregistrements (et/ou les modifier effacer)



Les différents types de données Mysql

MySQL dispose d'un grand nombre de **Type** regroupé en trois catégories :

les types numériques dont les principaux sont :

- **TINYINT**[(M)] [UNSIGNED] [ZEROFILL] Un très petit entier.
  Signé, il couvre l'intervalle -128 à 127 ; non signé, il couvre 0 à 255.
- **SMALLINT**[(M)] [UNSIGNED] [ZEROFILL] Un petit entier.
  Signé, il couvre l'intervalle -32768 à 32767; non signé, il couvre 0 à 65535.
- **INT**[(M)] [UNSIGNED] [ZEROFILL] Un entier de taille normale.
  Signé, il couvre l'intervalle -2147483648 à 2147483647; non signé, il couvre 0 à 4294967295.
- **BIGINT**[(M)] [UNSIGNED] [ZEROFILL] Un entier de grande taille.
  Signé, il couvre l'intervalle -9223372036854775808 à 9223372036854775807; non signé, il couvre 0 à 18446744073709551615. NB : toutes les opérations arithmétiques effectuée en interne, utilisent des BIGINT signés ou DOUBLE
- **FLOAT**[(M,D)] [ZEROFILL] Un nombre à virgule flottante, en précision simple. Il est toujours signé. Les valeurs sont comprises de -3.402823466E+38 et -1.175494351E-38.
- **DOUBLE**[(M,D)] [ZEROFILL] Un nombre à virgule flottante, en précision double. Il est toujours signé. Les valeurs sont comprises -1.7976931348623157E+308 et 2.2250738585072014E-308.

Tous les types numériques disposent d'un attribut optionnel ZEROFILL. Cette option force l'affichage de tous les zéros non significatifs. Ainsi, dans une colonne de type INT(5) ZEROFILL, 4 sera affiché :00004.

Quand une valeur trop grande est affectée à une colonne, ***MySQL*** limitera cette valeur au maximum qu'il peut stocker dans la colonne.

les types date et heure, dont les principaux sont :

- **DATE** Une date. L'intervalle valide de date va de '1000-01-01' à '9999-12-31'. MySQL affiche les DATE avec le format.
- **DATETIME** Une combinaison de date et d'heure. L'intervalle valide va de '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. MySQL affiche DATETIME avec le format 'YYYY-MM-DD HH:MM:SS'.
- **TIMESTAMP[(M)] Un timestamp :** la date et l'heure, exprimée en secondes, depuis le 1er janvier 1970. Il permet de couvrir un intervalle allant de'1970-01-01 00:00:00' à quelque part, durant l'année 2037
- **TIME** Une mesure de l'heure. L'intervalle valide est '-838:59:59' à '838:59:59'. MySQL affiche TIME au format 'HH:MM:SS'
- **YEAR** Un an. L'intervalle valide est 1901 à 2155, et 0000. MySQL affiche YEAR au format YYYY (Le type YEAR est nouveau en MySQL 3.22.)

Le type `DATETIME` est utile pour manipuler en même temps une date et une heure. . ***MySQL*** retourne et affiche les valeurs de type `DATETIME` au format `'YYYY-MM-DD HH:MM:SS'`. L'intervalle valide pour le type `DATETIME` est `'1000-01-01 00:00:00'` à `'9999-12-31 23:59:59'`.

Le type `DATE` est utilisé pour manipuler simplement une date, sans l'heure. ***MySQL*** retourne et affiche les valeurs de type `DATE` au format `'YYYY-MM-DD'` L'intervalle valide pour le type `DATE` est `'1000-01-01'` à `'9999-12-31'`

Le type `TIMESTAMP` est utilisé automatiquement lors de requête , avec la valeur courante de date et d'heure. Si il y a plusieurs colonnes de type `TIMESTAMP` , seule la première sera automatiquement mise à jour.

et les types chaînes de caractères dont les principaux sont :

- **CHAR(M)** [BINARY] Une chaîne de caractère de taille fixe, et toujours complétée à droite par des espaces. M va de 1 à 255 caractères. Les espaces supplémentaires sont supprimés lorsque la valeur est retournée dans une requête. Les tris et comparaisons effectués sur des valeurs de type CHAR sont insensibles à la casse, à moins que le mot clé BINARY soit précisé.
- **VARCHAR(M)** [BINARY] Une chaîne de caractère. Les tris et comparaisons effectués sur des valeurs de type CHAR sont insensibles à la casse, à moins que le mot clé BINARY soit précisé. Diffère de char uniquement sur la manière dont elle est stockée (n'est pas complété par des espaces, mais juste par un octe indiquant la longueur de la chaine) N.B: la création de un type varchar de moins de 4 octet est commutée automatiquement en CHAR…
- **ENUM**('value1','value2',...) Une énumération. Un objet chaîne peut prendre une des valeurs contenue dans une liste de valeur 'value1', 'value2', ..., ou NULL . Une ENUM peut avoir un maximum de 65535 valeurs distinctes.
- **SET**('value1','value2',...) Un ensemble. Un objet chaîne peut prendre une ou plusieurs valeurs, chacun de ces valeur devant être contenue dans une liste de valeurs 'value1', 'value2', ... Un SET peut prendre jusqu'à 64 éléments.

Avec **MySQL**, tous les types de colonnes peuvent être **indexés**, à l'exception des types BLOB et TEXT. L'utilisation d'index est le meilleur moyen d'accélérer les performances des clauses **SELECT**.

Une table peut avoir jusqu'à 16 index.

Il n'est pas possible d'indexer une colonne qui contient des valeurs NULL, donc une colonne indexée doit être déclarée **NOT NULL**.

MySQL peut créer des index sur plusieurs colonnes en même temps

Maintenant que l'on sait créer une base MySQL avec notre assistant, il est temps d'apprendre comment effectuer une connexion sur cette base en php

Principe d'accès à une base MySQL :

Schématiquement, il va falloir pour pouvoir accéder à une base de données effectuer le parcours suivant :

- **Se connecter à la base de donnée MySQL**
  en donnant le nom de la machine ou celle-ci se trouve, (host), et en s'identifiant (avec un username et un password)
- **Choisir une base de données parmi celles disponibles**
  même si certains hébergeur ne permettent que de créer une seule base, il faut spécifier le nom de la base sur laquelle on veut travailler
- **Passer des requêtes SQL**
  via des primitives du genre MySQL_query....
- **Fermer la connexion**
  même si par défaut lors de la fin du script la fermeture peut se faire automatiquement

Avec le SGBD MySQL, les fonctions **php** nécessaires sont les suivantes:

- **mysql_connect**
- **mysql_select_db**
- **mysql_query**
- **mysql_close**

Se connecter au serveur de base de données MySQL :

Fonction mysql_connect() :

int **mysql_connect(**string **host,** string **user,** string **password,)**

Cette fonction renvoit un identifiant de connexion ou une valeur **0** en cas d'échec de connexion. On peut interdire le warning par défaut du php en la faisant précéder du caractère **@.** 3 paramètres sont attendus :

- Le nom d'hôte du serveur de base de données. (ordinateur sur lequel le SGBD est installé)
- Le nom d'utilisateur Mysql
- Le mot de passe de l'utilisateur

Fonction mysql_close() :

int **mysql_close(**int **ptrconn)**

Cette fonction renvoit une valeur **0** en cas de réussite de fermeture de connexion et **1** sinon. 1 paramètre est attendu :

- c'est l'identifiant récupéré lors de la connexion à l'host. (ordinateur sur lequel le SGBD est installé) via la fonction mysql_connect()

Essayons juste de nous connecter et de nous déconnecter

Voir le fichier **coursphp_19.php3**

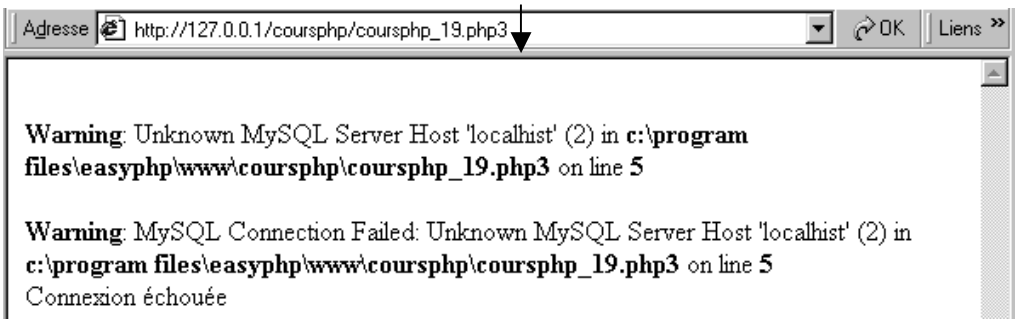
```
<?
$host="localhost";          // essayer avec localhist...
$user="";
$password="";
if (!$link = mysql_connect($host,$user,$password))
    {
    print ("Connexion échouée");
    }
else
    {
    print ("Connexion réussie");
    mysql_close ($link);
    }
?>
```

Dans l'instruction

```
if (!$link = mysql_connect($host,$user,$password))
```

on utilise le fait que la fonction **mysql_connect** renvoit une valeur **0** en cas d'échec de connexion,

Ainsi dans le cas ou le host est appellé localhist, on ne peut se connecter et on obtient alors …



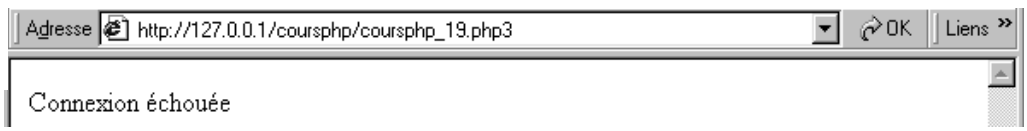
Si on veut interdire le warning par défaut du php, Il faut faire précéder le nom de la fonction du du caractère **@**

```
if (!$link = @mysql_connect($host,$user,$password))
```

de manière à lui substituer notre message

```
print ("Connexion échouée");
```

Voir le fichier **coursphp_19b.php3**



Certains préfèrent l'écriture suivante à l'aide de l'instruction **Die** Cette fonction affiche la chaîne passée en paramètre, puis termine l'exécution du script

```
<?
$host="localhost";// essayer avec localhist...
$user="";
$password="";

if (!$link = @mysql_connect($host,$user,$password))
    {
    die ("Connexion impossible");
    }
print ("Connexion réussie");
mysql_close ($link);
?>
```

voir fichier **coursphp_19c.php3**

et que penser de cette écriture résolument condensée ?

```
$link =@mysql_connect("localhost") or die ("Connexion impossible");
```

Sélection d'une base de données :

Fonction mysql_select_db() :

int **mysql_select_db(**string **nombase,** int **ptrconn)**

Cette fonction renvoit une valeur true à **1** en cas de réussite sur la selection de la base de donnée et false à **0** sinon. 2 paamètres sont attendus :

- Le nom de la base
- La valeur de l'identifiant de connexion au serveur mysql (obtenu lors de la connexion au serveur par la fonction mysql_connect)

Avec le même principe on peut arriver à écrire quelque chose du genre

```
<?
$host ="localhost";
$user ="";
$password = "";
$base_de_donnee ="essais";
if (!$link = @mysql_connect($host,$user,$password))
    {
    die ("Connexion impossible sur $host");
    }
print ("Connexion réussie sur $host");
if (!$database = @mysql_select_db($base_de_donnee,$link))
    {
    die ("Connexion impossible sur $base_de_donnee");
    }
print ("Connexion réussie sur $base_de_donnee");
mysql_close ($link);
?>
```

voir fichier **coursphp_20.php3**

après avoir effectué une connexion sur le serveur...

on cherche la base **essais**

Passer des requêtes MYSQL :

Maintenant que l'on arrive à ouvrir et fermer une connexion sur une base de données publiée sur un serveur SQL, il serait intéressant par exemple de lire le contenu d'une table de cette base ...

Fonction mysql_query() :

De manière générale, c'est l'instruction **mysql_query** qui va permettre de passer n'importe quelle requête SQL, (il faut donc connaître la formulation d'une requête en **SQL...)**

int **mysql_query(**string **requete,** int **ptrconn)**

Cette fonction renvoit une valeur true à **1** en cas de réussite de la requête sur la base de donnée et false à **0** sinon. Ce qui ne signifie rien au niveau des données renvoyées par la requête (une requête peut s'executer correctement et ne pas renvoyer de valeurs…). 2 paramètres sont attendus :

Commodité d'utilisation

On définira pour plus de commodités une variable chaîne **$query** contenant la requête SQL, que l'on passera en paramètre à la primitive php **mysql_query**

Cela devrait donner alors quelque chose du genre

```
$query="requête au format SQL";
$result = mysql_query($query);
```

1° exemple de requête MySql : INSERT

INSERT INTO $table(chptable1,chptable2…) **VALUES** ('$varform1','$varform2'…)

$table : représente la variable contenant le nom de la table concernée.

chptable1 : représente le champ de la table qui va être renseigné.

$varform1 : représente la variable de formulaire qui a été renseignée

On veut pouvoir insérer des éléments dans notre table "simple" de notre base "essais" précedemment construite à l'aide de myadmin…dont on rappelle ici la structure…

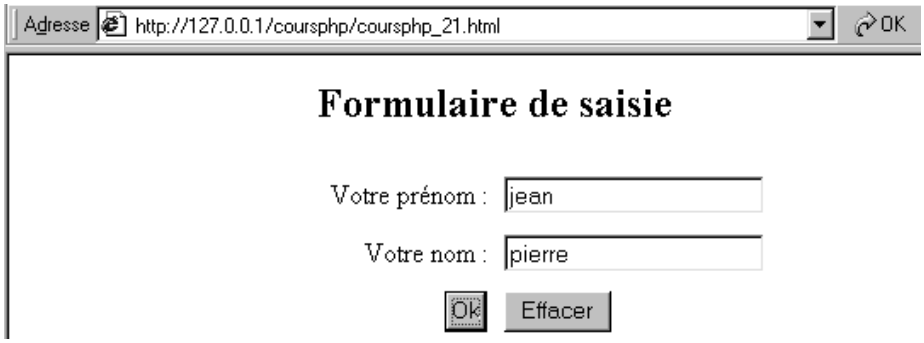
Base de données essais - table simple

Afficher

Champ	Type	Attributs	Null	Défaut	Extra	Action				
nom	char(3)		Non			Modifier	Supprimer	Primaire	Index	Unique
prenom	varchar(20)		Non			Modifier	Supprimer	Primaire	Index	Unique

Il faut se créer un formulaire pour saisir les champs nom et prenom de notre table…

Voir fichier **coursphp_21.html**



et le script associé aurait cet aspect là :

voir fichier **coursphp_21.php3**

après avoir effectué une connexion sur le serveur...

on cherche la base **essais**

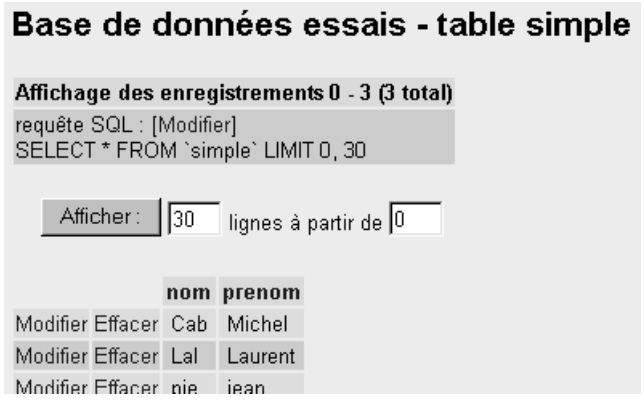
$result ne vaut pas 0…

```
<?
$host ="localhost";
$user ="";
$password = "";
$base_de_donnee ="essais";
$table="simple";
if (!$link = @mysql_connect($host,$user,$password))
    {
    die ("Connexion impossible sur $host");
    }
print ("Connexion réussie sur $host");
if (!$database = @mysql_select_db($base_de_donnee,$link))
    {
    die ("Connexion impossible sur $base_de_donnee");
    }
print ("Connexion réussie sur $base_de_donnee");
$query="INSERT INTO $table(nom,prenom) VALUES ('$nom','$prenom')";
$result = mysql_query($query);
echo "l execution de la requete renvoit $result";
mysql_close ($link);
?>
```

Ce script récupère depuis le formulaire les variables $nom et $prenom, puis demande d'éxécuter la requête… si une valeur true (différente de 0) est retournée, on sait que la requête s'est effectuée…

on peut ensuite pour vérifier,

visualiser notre table par :



2° exemple de requête MySql : SELECT

Nous avons vérifié par Myadmin, que la table simple de notre base essais contenait notre nouvel enregistrement…

Il est juste bon d'indiquer tout de suite que ne php il existe toute une série de fonctions permettant de manipuler une base mysql.

Lorsque dans l'exemple précedant, on a passé la requête, à travers l'instruction

```
$result = mysql_query($query);
```

on a juste testé que le retour était différent de 0(requête effectuée) ou valait 0 (requête non réalisée). Il faut savoir que dans le cas d'une requête effectuée correctement, et ramenant un résultat (c'est le cas d'une requête select), on peut alors passer la valeur retournée par mysql_query à d'autres fonctions, qui interprèterons alors "l'identifiant de résultat".

Ici mysql_numrows() compte les lignes d'un résultat.

voir fichier **coursphp_21b.php3**

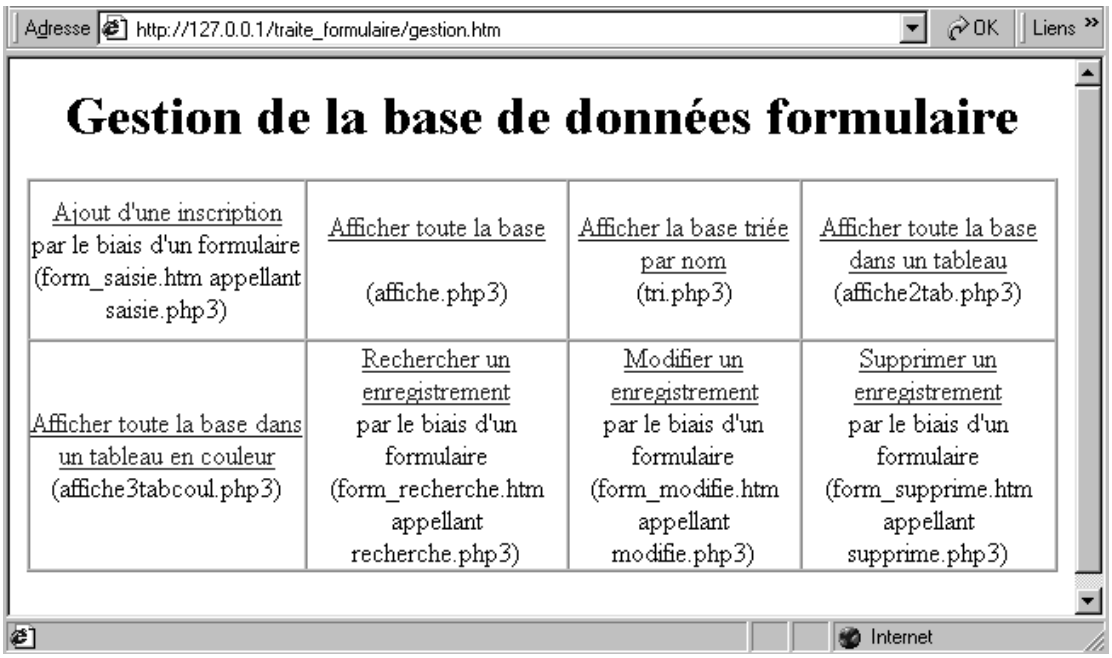
```
<?
$host ="localhost";
$user ="";
$password = "";
$base_de_donnee ="essais";
if (!$link = @mysql_connect($host,$user,$password))
    {
    die ("Connexion impossible sur $host");
    }

if (!$database = @mysql_select_db($base_de_donnee,$link))
    {
    die ("Connexion impossible sur $base_de_donnee");
    }
// ici on est bien connecté sur la base publiée sur l'hôte

$query="SELECT * From simple";      //requete SQL
$result = mysql_query($query);     //récup d'un ptr sur le résultat du select

echo "l execution de la requete renvoit $result";
$nbenreg=mysql_numrows($result);

print("il y a $nbenreg enregistrements dans la table");

mysql_close ($link);
?>
```

$result ne vaut pas 0, et c'est un pointeur de résultat… ⟶

N.B: on reprendra plus loin l'étude de SELECT, il s'agit ici d'un exemple pour comprendre les différentes "valeurs de retour " de l'instruction mysql-query()…

TRAITEMENT DE FORMULAIRE
CREATION DE LA BASE

Pour mettre en évidence toute une série de manipulations de base de donnée Mysql en php, nous allons construire une ensemble de fonctionnalités utilisables depuis une page HTML nomées **gestion.htm**



Les fonctions mysql dont on aura besoin, sont rassemblée page 124, les primitives SQL dont on aura besoin pour construire nos requêtes sont rassemblées page 127

Introduction :

A l'issue de la saisie depuis un formulaire, on souhaite pouvoir stocker les données saisies dans une table nommée inscrits d'une base de données nommée formulaire.

Il va donc falloir :

Créer un formulaire de saisie que l'on nommera **form_saisie.html**

Créer une base de données que l'on nommera **formulaire**

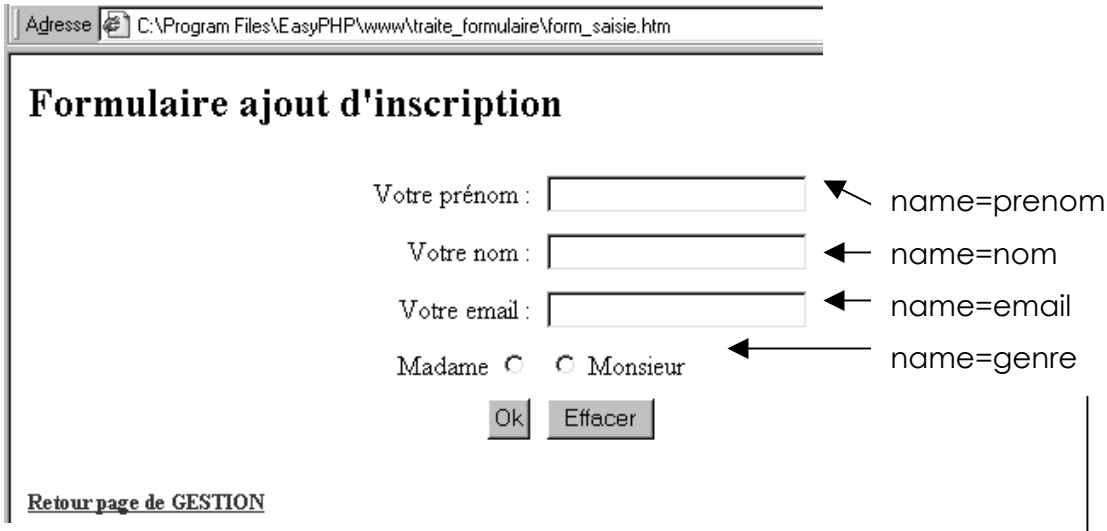
Créer une table que l'on nommera **inscrits**

Créer un script php que l'on nommera **saisie.php3. Ce script sera appelé lors de la validation de la saisie du formulaire.**

Le formulaire de saisie :

On prévoit donc 4 zones de saisie, permettant respectivement la saisie du **Prénom**, du **Nom**, de l'**Adresse Email** et du **Sexe** de la personne qui s'inscrit.

voir fichier **form_saisie.html**



Le script pourrait avoir cet aspect là

```
<HTML>
<BODY>
<form method="post" action="saisie.php3">
 <h2 >Formulaire ajout d'inscription</h2>
 <table width="100%" border="0" cellspacing="10" cellpadding="0">
  <tr>
   <td width="50%" align="right">Votre prénom :</td>
   <td width="50%"><input type="text" name="prenom"></td>
  </tr>
  <tr>
   <td width="50%" align="right">Votre nom : </td>
   <td width="50%"><input type="text" name="nom"></td>
  </tr>
  <tr>
   <td width="50%" align="right">Votre email : </td>
   <td width="50%"><input type="text" name="email"></td>
  </tr>
  <tr>
   <td width="50%" align="right">Madame <input type="radio" name="genre"
value="Madame"></td>
   <td width="50%"><input type="radio" name="genre" value="Monsieur">
Monsieur</td>
  </tr>
  <tr>
   <td width="50%">
    <div align="right"><input type="submit" name="Submit" value="Ok"></td>
   <td width="50%"><input type="reset" name="reset" value="Effacer"></td>
  </tr>
 </table>
 </form>
<h5 ><a href="gestion.htm"> Retour page de GESTION</a></h5>
</BODY>
</HTML>
```

Création de la base et de la table :

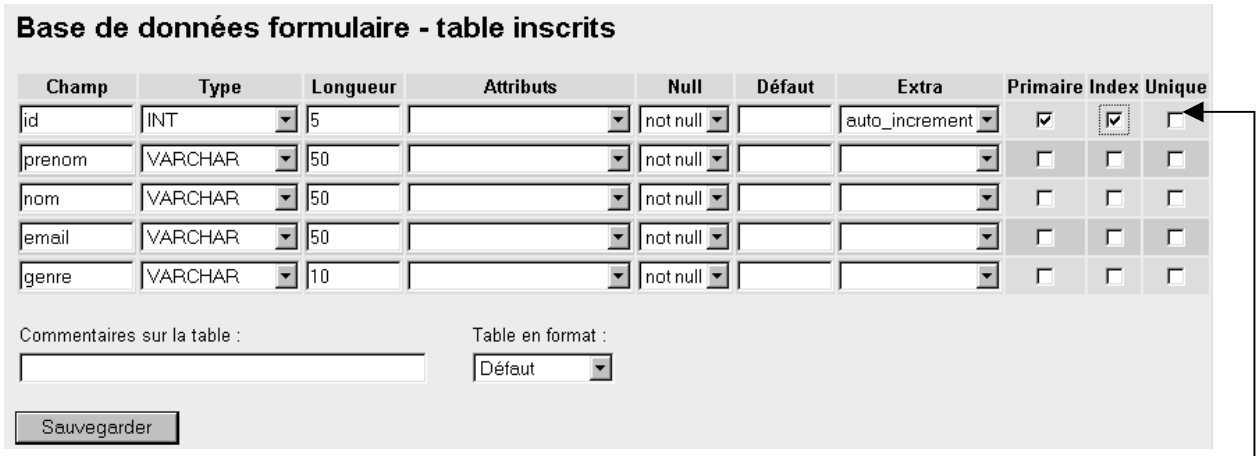
On pourrait en toute logique, tester lors de l'envoi du formulaire, si la table existe déjà, et dans le cas contraire la créer de toute pièce en SQL. Mais on ne va pas tout réinventer, et donc on va créer la table et sa structure via MySQL et ses assistants

On stockera tout cela dans un dossier Spécifique nommé **traite_formulaire** à coté du dossier réservé au cours

Créons une base de donnée nommée **formulaire**

Dans laquelle on va créer une table nommée **inscrits** avec les **5** champs suivants :

Avec la structure suivante :



NB: notamment On crée un champ ID doté de l'attribut Primary (key) de manière à être sur que malgré les éventuels doublons, chaque enregistrement soit unique dans la table... de ce fait on lui ajoutera également la propriété autoincrement

```
requête SQL : [Modifier]
CREATE TABLE `inscrits` (`id` INT(5) not null AUTO_INCREMENT, `prenom` VARCHAR(50) not null, `nom` VARCHAR(50) not null, `email`
VARCHAR(50) not null, `genre` VARCHAR(10) not null , PRIMARY KEY (`id`), INDEX (`id`))
```

Le script ajout d'inscription (stockage dans la table):

Un tel script, doit contenir, on le sait,deux partie distinctes:

- Une connexion sur la base Mysql en php

On peut se dire ici que "ces lignes" de codes, seront nécessaires pour toute procédure devant ouvrir une connexion sur la base mysql… il serait interessant d'apprendre à les mettre dans un fichier externe, de manière à ne pas surcharger avec le même code, toutes les bout de programmes….

voir fichier **saisie.php3**

```
$host="localhost";
$user="";
$password="";
$bdd="formulaire";
$table="inscrits";

/********** connection avec MySQL **********/
$link=@mysql_connect($host,$user,$password) or die("erreur connection hôte");

/********** connection à la base de données **********/
$select=@mysql_select_db("$bdd") or die("erreur connection base de données");

/********** affichage variables formulaire pour info**********/
echo "Bonjour $genre $prenom $nom, votre email est $email";
```

On peut se dire ici que "ces lignes" de codes, seront nécessaires pour toute procédure devant ouvrir une connexion sur la base mysql… il serait interessant d'apprendre à les mettre dans un fichier externe, de manière à ne pas surcharger avec le même code, toutes les bout de programmes….

- Le passage d'une requête mysql sur la base

Le plus difficile ici, c'est de connaître la syntaxe de la requête SQL à passer en php sur la base mysql… On peut s'aider de notre phpmyadmin, pour executer une fois une commande similaire, (ici ajouter un enregistrement)

et voir la requête SQL générée…


```
/********** infos à ajouter dans la table **********/
$query = "INSERT INTO $table(prenom,nom,email,genre)
VALUES('$prenom','$nom','$email','$genre')";

/********** stockage dans la bdd **********/
$result = mysql_query($query);
mysql_close($link);
```

$query contient la requête SQL que l'on passe ensuite en paramètre à **mysql_query()**

Include de fichier php :

Il peut être utile de mettre une partie du code Php dans un autre fichier, pour alléger le contenu de la page HTML dans laquelle on se trouve. De plus, une modification de cette section, s'appliquera automatiquement à tout le monde…

Syntaxe : include ("nomdufichier.ext");

à ce moment là l'écriture dans un fichier appelle un autre fichier…

```
<html>
<body>
<?php
 include("coursphp_03ext.php3");
?>
</body>
</html>
```

voir fichier **coursphp_03app.php3**

permet de faire référence à du code se trouvant dans un autre fichier, contenant simplement les instructions

```
<?php
 echo"<font size=\"4\" color=\"blue\"> mon premier script php !</font>";
?>
```

N.B: pour être précis, le fichier "inclus" ne nécessite pas forcément une extension de type .php3, mais se contente d'une quelconque extension.

Création d'un fichier connexion pour notre gestion

Modifier le fichier saisie.php3 de manière à ce qu'il fasse référence à un fichier connection.php3 contenant les paramètres de sélection

voir fichier **connexion.php3**

```
<?
$host="localhost";
$user="";
$password="";
$bdd="formulaire";
$table="inscrits";

/********** connection avec MySQL **********/
$link=@mysql_connect($host,$user,$password) or die("erreur connection hôte");

/********** connection à la base de données **********/
$select=@mysql_select_db("$bdd") or die("erreur connection base de données");
?>
```

Le script saisie.php3 commencera alors comme cela…

```
<?
include("connection.php3");
```

Insérer un fichier texte dans une table

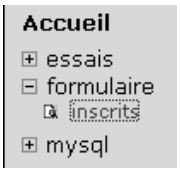
Objectif :

insérer un fichier texte nommé adr_email.txt dans la table nommée inscrits de la base formulaire

Le fichier adr_email.txt à l'aspect suivant :

```
5;Laurent;Bonvalet;laurent@bonvalet.com;Monsieur
6;Agnes;Durand;agnes@durand.com;Madame
7;Olivier;Blier;olivier@blier.com;Monsieur
8;Joelle;Delacroix;joelle@delacroix.com;Madame
9;Philippe;Lefebvre;philippe@lefebvre.com;Monsieur
10;Genevieve;Fabre;genevieve@fabre.com;Madame
11;Alain;Dupond;alain@dupond.com;Monsieur
12;Christian;Delavigne;christian@delavigne.com;Monsieur
13;Roger;Auguste;roger@auguste.com;Monsieur
14;Isabelle;Desbiolles;isabelle@desbiolles.com;Madame
```

Insertion du fichier texte

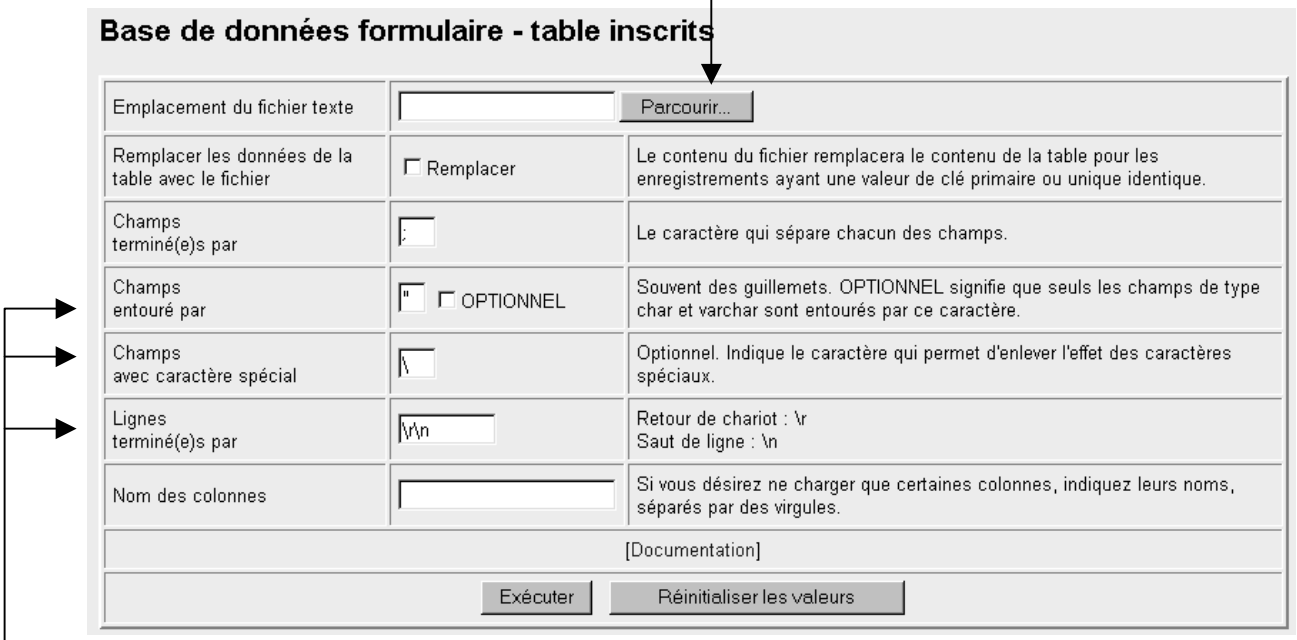


La table est créée on peut maintenant importer le fichier texte.

Démarrez Phpmyadmin, sélectionnez base formulaire, puis la table inscrits puis venez cliquer sur

Précisez le dossier de stockage du fichier texte ainsi que son nom.

C:\Program Files\EasyPHP\www\coursphp\adr_email.txt



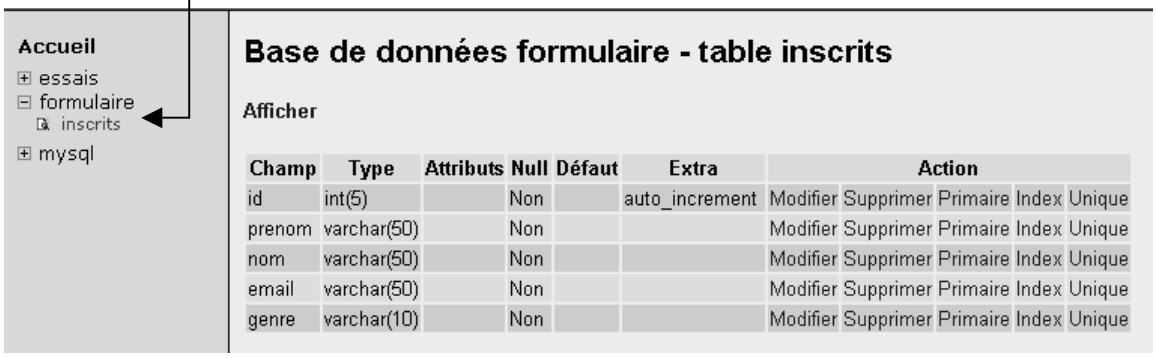
—"Videz" les différents champs inutiles

puis demander Exécuter…

TRAITEMENT DE FORMULAIRE
GESTION DE LA BASE

Afficher toute la base :

On peut bien sûr vérifier que les enregistrements se sont stockés dans la base en utilisant MySQL et PhpAdmin...en étant sur la table **inscrits** et en demandant **Afficher**



Mais il faut aussi savoir le faire en php. Si on passe sur les problèmes de connexion et de déconnexion, désormais classiques, l'essentiel du script concerne bien sûr la requête SQL et le traitement de son résultat...

Plus exactement il faut savoir selectionner des enregistrement à l'aide d'une requête **SELECT**, et traiter le résultat à l'aide d'une instruction php **mysql_fetch_row()**

La requête SELECT (cf page 128) que l'on passe ici est **SELECT * FROM $table** puis on traite le tableau par mysql_fetch_row (cf page 125)

$result est un pointeur sur le résultat du SELECT

pointeur que l'on utilise en paramètre de **mysql_fetch_row()**

```
/*** On cherche tous les enregistrements ***/
      $query = "SELECT * FROM $table";
      $result = mysql_query($query);
/*** le résultat de la requête est stocké dans un tableau ***/
      while($row = mysql_fetch_row($result))
      {
            /** on affiche le genre, le nom, le prenom, l'email **/
            echo "
            <p>\n
            <b>$row[1] $row[2] $row[3] $row[4]</b>\n
            <p>\n
            ";
      }
echo $nb;
```

voir fichier **affiche.php3**

Utiliser la fonction php : mysql_fetch_row($result)

Syntaxe :

$row=mysql_fetch_row($result)

- $row est une variable de type tableau
- $result est une variable contenant l'identifiant retourné par la fonction mysql_query

mysql_query() retourne un **identifiant (pointeur)** comme résultat d'une requête SQL de type SELECT. Ici cette variable contient l'ensemble des enregistrements de la table et n'est donc pas exploitable tel quel. Ainsi on utilise la fonction **mysql_fetch_row()**, qui découpe les lignes de résultat en colonnes (pour nous id, genre, nom, prenom, email) et les affecte à une variable tableau dans l'ordre où elles arrivent.

- **$rows[0]** correspond au champ id dans la table
- **$rows[1]** correspond au champ prenom dans la table
- **$rows[2]** correspond au champ nom dans la table
- **$rows[3]** correspond au champ email dans la table
- **$rows[4]** correspond au champ genre dans la table

D'autre part, on inclut généralement **mysql_fetch_row()** dans une boucle **while** de telle façon à ce que l'ensemble des lignes de résultat soient traitées. Lorsqu'il n'y a plus de ligne à traiter, la boucle **while** se termine et l'interpréteur exécute la suite des instructions.

Trier (Classer) toute la base :

Si on veut présenter maintenant la table triée par ordre de nom, alors il suffit juste de rajouter à la requête SQL de base un **ORDER by nomchamps ASC**

Le détail de la requête SQL (cf page 128) donne:

SELECT * FROM $table ORDER by chptable ASC|DESC

$table : représente la variable contenant le nom de la table concernée

chptable : représente le champ de la table sur lequel on désire faire le tri

ASC|DESC : représente l'ordre croissant ou décroissant

```
/*** On cherche tous les enregistrements ***/
    $query = "SELECT * FROM $table ORDER by nom ASC";
```

et si l'on souhaite avoir uniquement les informations du type genre – nom – adresse mail, il suffira de ne pas traiter tous les éléments du tableau renvoyé par mysql_fetch_row()…

```
while($row = mysql_fetch_row($result))
    {
        /** on affiche le genre, le nom, l'email **/
        echo "
        <p>\n
        <b>$row[4] $row[2] $row[3]</b>\n
        <p>\n
        ";
    }
```

voir fichier **tri.php3**

Maintenant que l'on sait créer une base MySQL et que l'on a vu le principe d'écriture d'une requête SQL à travers la primitive php mysql_query, essayons de traiter un exemple plus complet que le simple affichage du nombre d'enregistrements existants dans une table.

Afficher toute la base dans un tableau (1° variante):

Pour présenter mieux le résultat on utilise un tableau HTML

```
<?
Include("connection.php3");
/*** On cherche tous les enregistrements ***/
    $query = "SELECT * FROM $table";
/*** envoi de la requête à la base ***/
    $result = mysql_query($query);
echo "<Table>";
    while($row = mysql_fetch_row($result))
    {
        $id=$row[0];
        $nom=$row[1];
        $prenom=$row[2];
        $email=$row[3];
        $genre=$row[4];
        echo "<tr>
        <td width=22%>$id</td>
        <td width=22%>$genre</td>
        <td width=22%>$nom</td>
        <td width=22%>$prenom</td>
        <td width=34%>$email</td>
        </tr>";
    }
    echo "</Table>";

/*** on ferme la connexion ***/
mysql_close();
?>
```

voir fichier **affiche2tab.php3**

Afficher toute la base dans un tableau (2° variante):

Pour encore mieux présenter le résultat on utilise un tableau HTML avec des couleurs alternées automatiquement pour chaque ligne, ce qui s'obtient relativement facilement à l'aide de variables statiques, c'est à dire gardant leur valeur d'un appel à l'autre...

On avait déjà vu cet effet dans le cours sur les variables statiques (page 47), il n'y a donc pas grand chose à dire de plus...

il faut déclarer la fonction **couleur()**

```
function couleur()
    {
        static $couleur;
        if ($couleur == "#dddeee")
        {
            $couleur = "#add8e6";
        }
        else
        {
            $couleur = "#dddeee";
        }
        return ($couleur);
    }
```

voir fichier **affiche3tabcoul.php3**

puis dans le code

faire un appel à chaque itération

et se servir de la couleur pour la ligne du tableau HTML

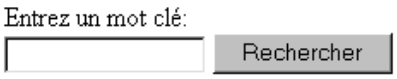
```
while($row = mysql_fetch_row($result))
    {
        $couleuractive = couleur();
        $id=$row[0];
        $nom=$row[1];
        $prenom=$row[2];
        $email=$row[3];
        $genre=$row[4];
        /** on affiche le id, legenre, le nom, le prenom, l'email **/
        echo "<tr BGcolor=$couleuractive>
        <td width=22%>$id</td>
        <td width=22%>$genre</td>
        <td width=22%>$nom</td>
        <td width=22%>$prenom</td>
        <td width=34%>$email</td>
        </tr>";
    }
```

voir fichier **affiche3tabcoul.php3**

Rechercher un enregistrement :

On veut proposer à l'utilisateur la recherche d'un enregistrement qui contient une certaine valeur soit dans le champ **nom** soit dans le champ **prenom**...

Il faut créer un formulaire permettant la saisie du nom ou prénom à rechercher



La création du formulaire HTML se passant sans problème (**form_recherche.htm**)

Il faut créer un script permettant la recherche et l'affichage des enregistrements correspondant au nom ou prénom recherché

Le script de recherche est à peu près identique aux scripts précédents mais il faut apporter deux modifications

on ne veut pas lancer une recherche sans mot clé...

```
if (($Mot == "")||($Mot == " ")) {      //if empty($mot)) serait plus propre !
/** Si aucun mot clé n'a été saisi demande à l'utilisateur de bien vouloir préciser un mot clé **/
      echo "
      Veuillez entrer un mot clé s'il vous plaît!
      <p>";
}

else {
      ..... totalité du traitement.....
      }
```

voir fichier **recherche.php3**

la requête SQL de recherche...

```
$query = "SELECT * FROM $table
      WHERE nom LIKE \"%$Mot%\"
      OR prenom LIKE \"%$Mot%\"
      ";

      $result = mysql_query($query);
```

voir fichier **recherche.php3**

Le détail de la requête SQL donne:

```
SELECT * FROM $table WHERE chptable1 LIKE \"%$Mot%\"
```

$table : représente la variable contenant le nom de la table concernée

chptable : représente le champ sur lequel on désire faire le tri

$mot : représente la variable contenant le mot recherché

LIKE : permet de faire des comparaisons utilisant des jokers :

% : permet de remplacer des caractères (comme le * du dos)

Comme dans

```
$query = "SELECT * FROM $table WHERE nom LIKE \"%$Mot%\"";
```

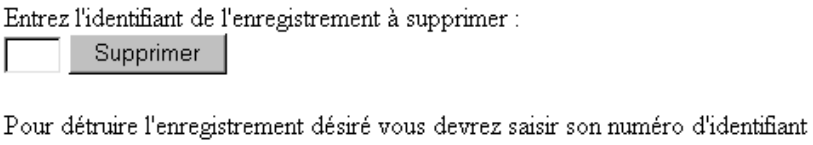
On peut utiliser l'opérateur AND et l'opérateur OR

```
$query = "SELECT * FROM $table WHERE nom LIKE \"%$Mot%\" OR prenom LIKE "%$Mot%\" ";
```

Supprimer un enregistrement :

On veut proposer à l'utilisateur la suppression d'un enregistrement qui correspond à une valeur du champ identificateur...

Il faut créer un formulaire permettant la saisie du n° de l'enregistrement à supprimer



La création du formulaire HTML se passant sans problème (**en form_supprime.htm**). Il faut créer un script permettant la suppression de l'enregistrement correspondant au numéro saisi

Par rapport à un script de sélection classique quelques modifications du code apparaissent

la vérification de saisie d'un identificateur...

la requête SQL de recherche...

l'affichage pour "info" de l'enreg trouvé

la requête SQL de suppression...

```
/** Si aucun id saisi demande de bien vouloir préciser un id **/
if (empty($identifiant)) {
      echo "
      Vous n'avez rien écrit : Veuillez entrer un chiffre s'il vous plaît !
      <p>";
}
/*** On affiche l'enregistrement correspondant à l'identifiant saisi ***/
else {

      $query = "SELECT * FROM $table WHERE id = $identifiant";
      $result = mysql_query($query);
      while($row = mysql_fetch_array($result))
      {
            echo "
            <p>\n
            <b>identifiant détruit : $row[0] : $row[1] $row[2] $row[3] </b>\n
            <p>\n
            ";
      }
/*** On détruit l'enregistrement correspondant à l'identifiant saisi ***/
      $query = "DELETE FROM $table WHERE id=$identifiant";
      $result = mysql_query($query);

}
```

voir fichier **supprime.php3**

La requête sql : DELETE

Le détail de la requête SQL donne:

DELETE FROM $table WHERE chptable = $identifiant

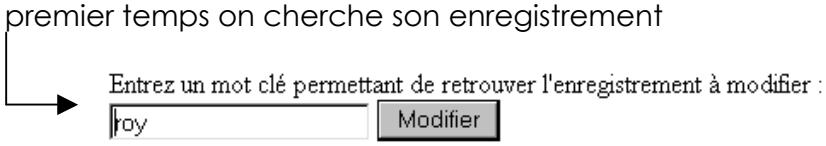
$table : représente la variable contenant le nom de la table concernée

chptable : représente le champ de la table sur lequel on désire faire la recherche

$identifiant : représente la variable contenant le mot recherché

Modifier un enregistrement :

On veut proposer à l'utilisateur la modification d'un enregistrement. Dans un premier temps on cherche son enregistrement



puis il obtient

dans lequel il effectue ses modifications

Il faut créer un formulaire permettant la saisie du nom ou prénom à rechercher

La création du formulaire HTML se passant sans problème (**en form_modifie.htm**)

Nous allons ensuite créer 2 scripts php : le premier nommé modifie.php3 devra afficher dans un formulaire les informations concernant la personne dont on veut modifier une ou plusieurs données

Le second script nommé modifie2.php3 devra

- afficher les anciennes informations
- afficher les nouvelles informations
- faire le changement (on pourrait pour plus de sécurité demander la confirmation des modifications)
-

Voici à quoi pourrait ressembler le script modifie.php3

```
<html>
<head>
</head>

<body>
<?
include("connection.php3");

/** Si aucun id saisi demande à l'utilisateur de bien vouloir préciser un id **/
if (empty($Mot)) {
      echo "
      Vous n'avez rien écrit : Veuillez entrer un mot clé s'il vous plaît !
      <p>";
}
/*** On affiche l'enregistrement correspondant à l'identifiant saisi ***/
else {
      $query = "SELECT * FROM $table
      WHERE nom LIKE \"%$Mot%\"
      OR prenom LIKE \"%$Mot%\"
```

affichage du formulaire permettant "d'éditer " l'enregistrement à modifier

Ce formulaire sera traité dans le script **modifie2.php3**

```
";
$result = mysql_query($query);
print("<form Action=modifie2.php3 Method=post>");
while($row = mysql_fetch_row($result))
{
      echo "<center>
      <p>\n
      <b>Enregistrement Modifiable : </b><br>
      <input type=text name=identifiant value=$row[0]> <br>
      nom <br><input type=text name=nom value=$row[2]> <br>
      prénom <br><input type=text name=prenom value=$row[1]> <br>
      email <br><input type=text name=email value=$row[3]> <br>
      genre <br><input type=text name=genre value=$row[4]> <br>
      </b>\n
      <p>\n
      ";
}
print("<input type=submit value=modifier> <br>");
print("<input type=reset value=annuler> <br>");
print("</center></form>");

}

// on ferme la base
mysql_close();

?>
<h5 align=center><a href="gestion.htm">Retour page de GESTION</a> |
</h5>
</body>
</html>
```

Le script modifie2.php3 pourrait ressembler à :

```
<html>
<head>
</head>

<body>
<?
include("connection.php3");


/********** pour visualiser l'enreg avant la modification **********/
$query = "SELECT * FROM $table WHERE id = $identifiant";
$result = mysql_query($query);
      echo "enregistrement avant modification ?";
      echo "<Table>";
      while($row = mysql_fetch_row($result))
      {
            $affid=$row[0];
            $affnom=$row[1];
            $affprenom=$row[2];
```

```
            $affemail=$row[3];
            $affgenre=$row[4];
            echo "<tr>
            <td width=22%>$affid</td>
            <td width=22%>$affgenre</td>
            <td width=22%>$affnom</td>
            <td width=22%>$affprenom</td>
            <td width=34%>$affemail</td>
            </tr>";
      }
      echo "</Table>";

/**********  MODIFICATION **********/
      $query = "UPDATE $table SET prenom=\"$prenom\", nom=\"$nom\",
email=\"$email\", genre=\"$genre\" WHERE id = \"$identifiant\"";
      $result = mysql_query($query);

/********** pour visualiser l'enreg après la modification **********/
      $query = "SELECT * FROM $table WHERE id = $identifiant";
      $result = mysql_query($query);
      echo "enregistrement après modification !";
      echo "<Table>";
      while($row = mysql_fetch_row($result))
      {
            $affid=$row[0];
            $affnom=$row[1];
            $affprenom=$row[2];
            $affemail=$row[3];
            $affgenre=$row[4];
            echo "<tr>
            <td width=22%>$affid</td>
            <td width=22%>$affgenre</td>
            <td width=22%>$affnom</td>
            <td width=22%>$affprenom</td>
            <td width=34%>$affemail</td>
            </tr>";
      }
      echo "</Table>";

// on ferme la base
mysql_close();

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

Le détail de la requête SQL donne:

UPDATE $table SET chptable = $identifiant, WHERE chptable1 = $identifiant1

$table : représente la variable contenant le nom de la table concernée

chptable : représente le champ de la table que l'on souhaite modifier

$identifiant : représente la variable contenant la valeur à utiliser

chptable : représente le champ de la table que l'on souhaite modifier

UPDATE met à jour une ligne existante dans une table. La clause **SET** indique quelles colonnes modifier, et quelles valeurs mettre dans ces colonnes. La condition **WHERE** permet de choisir quelles lignes sont à mettre à jour. Sinon, toutes les lignes sont mises à jour

Compléments SELECT - Filtrer une base :

Maintenant que l'on a vu un certain nombre de manipulation de la base mysql, on peut décider de filtrer un peu les renseignements obtenus. Cela se fait bien sur à l'aide de l'instruction SELECT (cf page 128) que nous connaissons déjà…

Mais nous pouvons aussi introduire ici une nouvelle fonction php permettant de compter le nombre d'enregistrements dans un résultat de requête. Il s'agit de la fonction php : mysql_numrows() (voir page 125)

Comme dans l'exemple ci-dessous

```
$query="SELECT * from $table ";
$result = mysql_query($query);
$nbenreg=mysql_numrows($result);
```

Dans les extraits de script ci dessus, une requête est envoyée grâce à l'instruction mysql_query et le résultat de cette requête est stocké dans la variable $result. mysql_numrows permet de compter le nombre d'enregistrements sélectionnés.

Essayez les !

Exemple 1 :

Sélection de tous les enregistrements de la table

```
$query="SELECT * from $table ";
$result = mysql_query($query);
$nbenreg=mysql_numrows($result);
```

Exemple 2 :

Sélection de tous les enregistrements de la table dans lesquels nom=Blier

```
$query="SELECT * from $table where nom='Blier' ";
$result = mysql_query($query);
```

Exemple 3 :

Sélection de tous les enregistrements de la table dans lesquels nom commence par Du…

```
$query="SELECT * FROM $table where typevel like 'Du%'";
$result = mysql_query($query);
```

Exemple 4 :

Sélection de tous les enregistrements de la table dans lesquels nom commence par Du… et dont le genre est Madame

```
$query="SELECT * FROM $table where ((nom like 'Du%') AND (genre='Madame'))";
$result = mysql_query($query);
```

Exemple 5 :

Sélection de tous les enregistrements de la table dans lesquels les ventes sont comprises entre 5000 et 7000

```
$query="SELECT * FROM $table where (ventes BETWEEN 5000 AND 7000)";
$result = mysql_query($query);
```

Exemple 6 :

Sélection de tous les enregistrements de la table dans lesquels figurent Mathieu et Boully

```
$query="SELECT * FROM $table where nom IN ('Blier', 'Durand')";
$result = mysql_query($query);
```

Exemple 7 :

Sélection de tous les enregistrements de la table dans lesquels commercial=Mathieu mais récupération uniquement de la valeur des ventes et des régions concernées

```
$query="SELECT vente,region FROM $table Where nom='Blier' ";
$result = mysql_query($query);
```

LISTE DE QUELQUES FONCTIONS MYSQL PHP

mysql_close

int **mysql_close (**int **link_identifier)**

Retourne **TRUE** en cas de succès, et **FALSE** sinon.

mysql_close() ferme la connexion au serveur MySQL associée à l'identifiant link_identifier . Par défaut, s'applique à la dernière connexion ouverte.

NB : Cette commande n'est pas strictement nécessaire, car toutes les connexions non persistantes seront automatiquement fermées à la fin du script.

mysql_connect

int **mysql_connect (**string **host :port,** string **username,** string **password)**

Retourne un identifiant positif de connexion en cas de succès, et sinon **FALSE**.

mysql_connect() établit une connexion à un serveur MySQL. 3 arguments sont nécessaires pour une connection, **host**, **username**, **password**. Tous les arguments sont optionnels, et s'ils manquent, les valeurs par défaut sont utilisées. Le lien sera fermé lors de la fin du script automatiquement ou avec **mysql_close()**.

host : c'est le nom d'hôte sur lequel notre base Mysql est hebergée. Par défaut le nom "localhost" sera utilisé. Le nom d'hôte peut aussi inclure un numéro de port, sous la forme : "host:port". (à partir de la version 3.0B4.)

username : c'est le nom de login autorisé à ouvrir une connexion sur cette base de donnée. Par défaut le nom du propriétaire du process sera utilisé.

password : c'est le mot de passe associé au login pour cet utilisateur de la base de donnée Par défaut le mot de passe vide sera utilisé.

N.B: Si un second appel à mysql_connect() est fait avec les mêmes arguments, PHP n' ouvre pas une nouvelle connexion, mais retourne l'identifiant de la connexion déjà ouverte.

Exemple MySQL connect

```
<?php
$link = mysql_connect ("kraemer", "marliesle", "secret")
or die ("Connexion impossible");
print ("Connexion réussie");
mysql_close ($link);
?>
```

mysql_fetch_row

array **mysql_fetch_row (**int **result)**

Retourne un **tableau énuméré** qui correspond à la ligne demandée, ou **FALSE** si il ne reste plus de ligne.

mysql_fetch_row() va rechercher une ligne dans le résultat associé à l'identifiant de résultat spécifié. La ligne est retournée sous la forme d'un tableau. Chaque colonne est enregistré sous la forme d'un tableau commençant à la position 0.

Les appels suivants à **mysql_fetch_row()** retourneront la ligne suivante dans le résultat, ou **FALSE** si il n'y a plus de ligne disponible.

mysql_query

int **mysql_query (**string **query,** int **link_identifier)**

mysql_query() retourne **TRUE** ou **FALSE**, pour indiquer le succès ou l'échec d'une requête SQL de type INSERT, DELETE ou UPDATE.

mysql_query() retourne un **identifiant (pointeur)** pour manipuler le résultat d'une requête SQL de type SELECT.

N.B: En cas de retour **TRUE**, la requête était valide et a pu être exécuté sur le serveur. Cela n'indique pas le nombre de ligne affectées, ou retournées. Il est parfaitement possible qu'une requête valide n'affecte aucune ligne ou ne retourne aucune ligne.

mysql_query() envoie une requête SQL à la base de données actuellement active sur le serveur MysQL. Si **link_identifier** n'est pas précisé, la dernière connexion est utilisée. Si aucune connexion n'a été ouverte, la fonction tentera d'en ouvrir une, avec la fonction mysql_connect() mais sans aucun paramètre (c'est à dire avec les valeurs par défaut).

mysql_num_rows

int **mysql_num_rows (**int **result)**

mysql_num_rows() retourne le nombre de ligne d'un résultat.

mysql_select_db

int **mysql_select_db (**string **database_name,** int **link_identifier)**

Retourne **TRUE** en cas de succès, **FALSE** sinon.

mysql_select_db() change la base de données active sur la connexion représentée par l'identifiant de connexion. Si aucun identifiant n'est spécifié, la dernière connexion est utilisée. S'il n'y a pas de dernière connexion, la fonction tentera de se connecter seule, avec **mysql_connect()** et les paramètres par défaut.

Toutes les requêtes suivantes avec **mysql_query()** seront faites avec la base de données active.

LISTE DE QUELQUES ELEMENTS SQL MYSQL

Insert:

insère une nouvelle ligne dans une table existante

INSERT INTO $table(chptable1,chptable2…) VALUES ('$varform1','$varform2'…)

$table : représente la variable contenant le nom de la table concernée par l'ajout.

chptable1 : représente les champs de la table qui vont être renseignés.

$varform1 : représente les variables qui sont utilisées pour mettre leur valeur dans la table.

Delete:

Le détail de la requête SQL donne:

DELETE FROM $table WHERE chptable = $identifiant

$table : représente la variable contenant le nom de la table concernée

chptable : représente le champ de la table sur lequel on désire faire la recherche

$identifiant : représente la variable contenant le mot recherché

Update:

UPDATE met à jour une ligne existante dans une table. La clause SET indique quelles colonnes modifier, et quelles valeurs mettre dans ces colonnes. La conditions WHERE permet de choisir quelles lignes sont à mettre à jour. Sinon, toutes les lignes sont mises à jour

Le détail de la requête SQL donne:

UPDATE $table SET chptable = $identifiant, WHERE chptable1 = $identifiant1

$table : représente la variable contenant le nom de la table concernée

chptable : représente le champ de la table que l'on souhaite modifier

$identifiant : représente la variable contenant la valeur à utiliser

chptable : représente le champ de la table que l'on souhaite modifier

Select:

La commande SELECT permet de sélectionner des enregistrements dans une table.

Cette commande permet de récupérer tous les champs spécifiés en argument à **SELECT** dans une table précisée en argument à **FROM** répondant aux critères indiqués en argument à **WHERE**

SELECT [nom_de_colonne, | *]

[**FROM** table_references

[**WHERE** where_definition]

[ORDER BY [ASC | DESC]]

[LIMIT [offset,] rows]]

- * permet de spécifier que l'on conservera tous les champs. Si on désire conserver uniquement certains champs il suffira de donner leurs noms séparés par une virgule.
- La clause FROM table_references indique les noms des tables où les lignes seront lues. SI vous utilisez plus d'une table, vous faites une jointure. Pour plus d'informations sur les jointures, voyez JOIN.
- La clause WHERE est chargée de spécifier les conditions de recherche. On peut utiliser les opérateurs de comparaison habituels (=, >, <, >=, <=, <>, !>, AND, OR, NOT) mais aussi d'autres opérateurs tels que :

  LIKE qui permet de rechercher une chaîne de caractères sans tenir compte de la casse. Avec LIKE vous pouvez utiliser les deux jokers suivants :

  % : Remplace n'importe quel nombre de caractères, même zéro

  _ : Remplace exactement un caractère

  [-] : permet de définir un intervalle de caractères

  BETWEEN qui permet de rechercher une valeur dans un intervalle.

  IN permet de rechercher une valeur dans une liste
- La clause ORDER BY est destinée au tri par ordre croissant (asc) ou décroissant (desc) des réponses.
- La clause LIMIT peut être utilisée pour limiter le nombre de lignes retournées par la commande SELECT. LIMIT prend un ou deux arguments numériques. Si deux arguments numériques sont fournis, le premier spécifie l'offset de la première ligne à retourner, et la seconde spécifie le nombre maximum de lignes à retourner. La première ligne est à l'offset 0 (et non pas 1):

QUELQUES VARIABLES D'ENVIRONNEMENT

$DOCUMENT_ROOT	Nom du répertoire physique contenant la page affichée.
$HTTP_REFERER	L'adresse de la page (si elle existe) qui a conduit le client à la page courante. Cette valeur est affectée par le client, et tous les clients ne le font pas.
$HTTP_ACCEPT_LANGUAGE	La langue utilisée par le navigateur du visiteur
$HTTP_USER_AGENT	L'identifiant du navigateur
$REMOTE_ADDR	L'adresse IP du client qui demande la page courante.
$REMOTE_HOST	Adresse de l'hôte
$SCRIPT_FILENAME	Le chemin absolu jusqu'au script courant.
$SCRIPT_NAME	Le chemin d'accès au script par rapport à $DOCUMENT_ROOT. Cela sert lorsque les pages doivent s'appeler elles-mêmes.
gethostbyaddr()	retourne le nom d'hôte correspondant à l'IP **ip_address**. Si une erreur survient, retourne **ip_address**.
gethostbyname().	

Voir fichier **envir.html** appellant le fichier **envir.php3**

```
<?
Echo "DOCUMENT_ROOT racine du site : $DOCUMENT_ROOT<br>";
Echo "HTTP_REFERER dossier courant : $HTTP_REFERER<br>";
Echo "HTTP_ACCEPT_LANGUAGE langue : $HTTP_ACCEPT_LANGUAGE<br>";
Echo "HTTP_USER_AGENT navigateur : $HTTP_USER_AGENT<br>";
Echo "REMOTE_ADDR adresse IP : $REMOTE_ADDR<br>";
Echo "SCRIPT_FILENAME chemin d'acces au script: $SCRIPT_FILENAME<br>";
Echo "SCRIPT_NAME nom du script: $SCRIPT_NAME<br>";
Echo "REMOTE_HOST nom de l'hote: $REMOTE_HOST<br>";
Echo" get host by addr avec remote host nom de la machine sur le réseau : ". gethostbyaddr($REMOTE_HOST) ."<br>";
Echo" get host by name avec remote host adresse ip de la machine sur le réseau : ". gethostbyname($REMOTE_HOST) ."<br>";
Echo" get host by addr : ". gethostbyaddr($REMOTE_ADDR)."<br>";
Echo" get host by name : ". gethostbyname($REMOTE_ADDR)."<br>";

?>
```

DOCUMENT_ROOT racine du site : /var/www/php3.pro.proxad.net
HTTP_REFERER dossier courant :
HTTP_ACCEPT_LANGUAGE langue : fr
HTTP_USER_AGENT navigateur : Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt; KITV4.7 Wanadoo)
REMOTE_ADDR adresse IP : 193.253.251.25
SCRIPT_FILENAME chemin d'acces au script: /usr/lib/cgi-bin/cgiwrap
SCRIPT_NAME nom du script: /cgi-bin/cgiwrap
REMOTE_HOST nom de l'hote:
get host by addr avec remote host nom de la machine sur le réseau :
get host by name avec remote host adresse ip de la machine sur le réseau :
get host by addr : AGrenoble-101-1-3-25.abo.wanadoo.fr
get host by name : 193.253.251.25

Objectif :

Identifier de manière durable un client, un usager du site.

Une grande partie des internautes n'ont pas leur propre adresse ip. Pour accéder à Internet ils utilisent les services d'un fournisseur d'accès (qui leur attribue une adresse ip dynamique) on ne peut donc à la seule vue de l'adresse ip identifier un usager du site.

Les cookies sont donc des petits fichiers texte stockés sur la machine du client permettant ainsi de le reconnaître.

Cookies et php

Création d'un cookie

Syntaxe de la fonction : setcookie(name, value,expire,path,domain,secure)

Seuls les 3 premiers paramètres sont régulièrement utilisés :

name : nom donné à la variable cookie

value : informations stockées dans le cookie

expire : temps unix pendant lequel sera actif le cookie (s'exprime en secondes)

Dans le cas suivant on crée une variable cookie nommé VISITEUR qui contient l'heure de connection et qui expirera dans 1 heure.

```
/* création de la variable cookie VISITEUR */
setcookie("VISITEUR",date("d/m/y"),time()+3600)
/* il s'agit d'une variable on peut donc l'utiliser comme telle */
echo "Vous êtes connectés à ".$VISITEUR;
```

N.B: la fonction setcookie doit être appelée en début de script **avant toutes instructions HTML**.

N.B: le nom du cookie est écrit en majuscules

N.B: Le script peut être stocké dans la racine de votre siteau dans ce cas il porte le nom de votre site sinon il porte le nom du dossier dans lequel il se trouve. Le nom du cookie généré sur la machine du client contiendra de ce fait le nom du domaine visité.

Par exemple : si un script (s'il est placé à la racine…) du site www.jesuisleplusbbeau.com créé un cookie sur votre machine celui ci s'appellera : nomsession@www.jesuisleplusbeau[1].txt

Afficher le contenu d'un cookie

Les cookies sont stockés dans un tableau associatif nommé **$HTTP_COOKIE_VARS.**

```
<?
 while($valeur=each($HTTP_COOKIE_VARS)) {
  $val_cook=$valeur[1];
  echo  $val_cook."<br>";
  }
?>
```

Pour afficher un cookie particulier on peut aussi utliser la notation suivante :

```
echo $HTTP_COOKIE_VARS["nomducookie"];
```

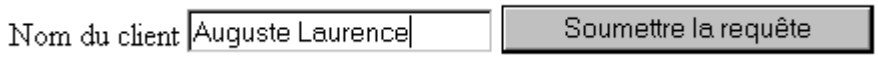
Détruire un cookie

Il suffit de déclarer le cookie sans autre paramètre que le nom. (a ce moment là, il va avoir automatiquement une durée de vie à 0)

```
setcookie("VISITEUR");
```

Exemple

Dans un premier temps nous allons créer un formulaire demandant son nom à l'internaute se connectant à notre site.



Fichier **cookie_form.html** appellant le fichier **cookie_ecrit.php3**

```
<html>
<body>
<form action="cookie_ecrit.php3" method="post">
Nom du client <input type = "text" name="nom">
<input type = "submit" >
</form>
<body>
<html>
```

Ce formulaire appelle un script nommé cookie_ecriture.php3 créant 2 cookies

- mémorise ce nom dans un cookie (INTERNAUTE)
- mémorise la date et l'heure de connexion dans un autre cookie (DATE)

```
<?
   /*création d'un cookie nommé internaute qui conserve le nom*/
   setcookie("INTERNAUTE",$nom,time()+3600);
   /*création d'un cookie nommé date qui conserve la date de connexion*/
   setcookie("DATE",date( "d/m/y à h:m:s"),time()+3600);
   echo "<a href=cookie_lecture.php3>Lire le cookie</A>";
?>
```

de ce script existe un lien vers un fichier nommé cookie_lecture.php3 qui :

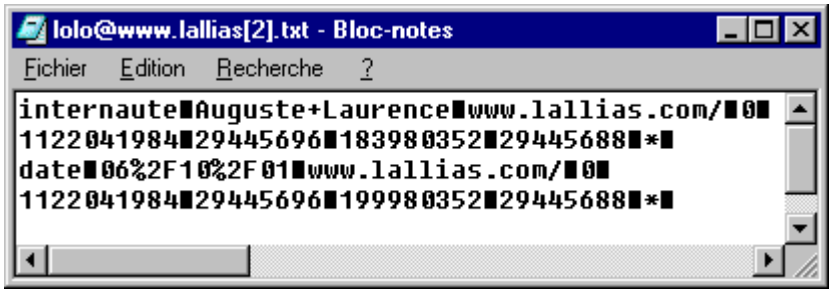
- affiche à l'écran l'ensemble des cookies

```
<?
echo "Bonjour ". $HTTP_COOKIE_VARS["INTERNAUTE"];
echo " comment allez vous depuis le ". $HTTP_COOKIE_VARS["DATE"];
?>
```

Etude du cookie :

(stocké dans le dossier racine du site www.lallias.com)

Sur ie : le cookie va se trouver dans le dossier cookies sous dossier de windows.



- La première valeur (internaute) est le nom du cookie. Notez que le nom du cookie n'est pas le nom du fichier (nom du fichier : lolo@www.lallias[2].text). Si plusieurs cookies proviennent du même site ils sont tous stockés dans le même fichier (ici cela explique le 2 qui apparaît dans le nom : lolo@www.lallias[**2**].txt)
- La seconde valeur (Auguste+Laurence) est la valeur assignée au cookie.
- La troisième valeur (www.lallias.com/) indique d'où provient le cookie et par quel site il est utilisé.

Supprimer les 2 cookies :

Il serait possible de supprimer ces cookies tres simplement en demandant :

```
<?
      setcookie("INTERNAUTE");
      setcookie("DATE");
?>
```

Principe

Lorsque l'on démarre une session php, le serveur lui affecte un identifiant de session nommé PHPSESSID (abréviation : SID).

A partir de ce moment toutes les variables de session que vous allez crées dans vos scripts seront soit :

- stockées sur le serveur dans un fichier ressemblant à un cookie (mais coté serveur!!). C'est l'option par défaut.
- stockées dans une table de base de données que vous aurez créée à cet effet.
- stockées dans les mémoires des processeurs du serveur.

Les variables enregistrées comme variables de session sont utilisables dans d'autres scripts. Les sessions vont donc servir au passage de paramètres entre scripts.

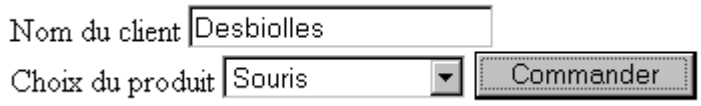
Exemple 1 : une gestion de client (hyper simplifiée)

On veut demander par le biais d'un formulaire html (formulaire_ident.html) :

- le nom d'un client
- le choix d'un type de matériel

Le nom et le produit seront transmis à un script php (ident.php3) qui affichera ces informations et les définira comme variables de session.

Le formulaire (formulaire_ident.html)



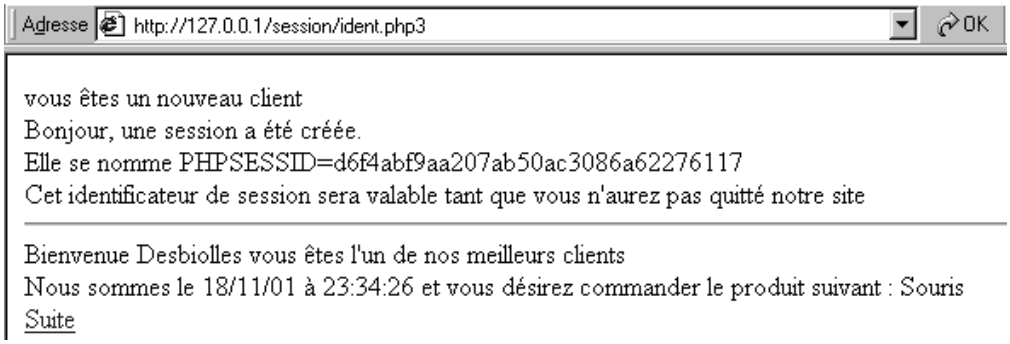

```
<html>
<body>
<form action="ident.php3" method=post>
Nom du client <input type=text name=client><br>
Choix du produit <select name=produit>
<option >Imprimante</option>
<option >Micro ordinateur</option>
<option >Souris</option>
<option >Tapis</option>
</select>
<input type=submit value=Commander>
</Form>
</body>
</html>
```

le formulaire appelle un script nommé ident.php3

Remarques

Une session est activée soit implicitement par une instruction session_start(), soit explicitement par une instruction session_register(). Attention l'instruction activant une session doit être la première du script.

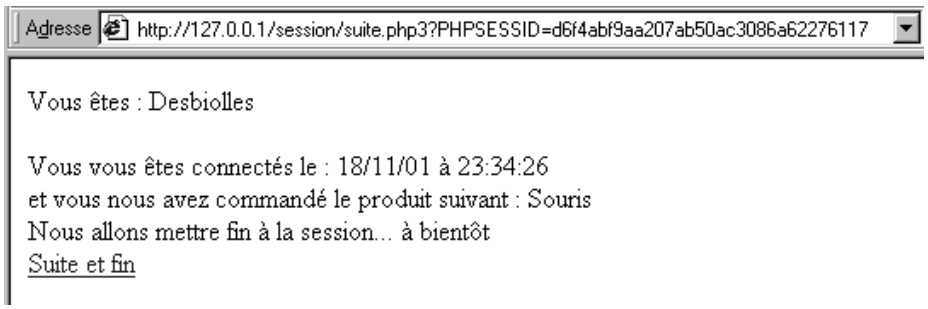
Le script (ident.php3)



```
<?
    /*on démarre une session*/
    session_start();
    echo "Bonjour, une session a été créée. <br>Elle se nomme ".SID.
    " <br>Cet identificateur de session sera valable tant que vous n'aurez pas quitté notre site <br>";
    echo "<hr>";
    /*on récupére le nom du client provenant du formulaire*/
    $nomclient=$client;
    /*on récupére le nom du produit provenant du formulaire*/
    $nomproduit=$produit;
    /*on récupére la date et l'heure dans une variable date*/
    $date=date("d/m/y à H:i:s");

    /*on affecte la variable pommade */
    /*la portée de cette variable ne dépassera pas ce script*/
    $pommade="vous êtes l'un de nos meilleurs clients ";

    /*on crée une variable de session nommée nomclient*/
    session_register("nomclient");
    /*on crée une variable de session nommée nomproduit*/
    session_register("nomproduit");
    /*on crée une variable de session nommée date*/
    session_register("date");

    echo "Bienvenue $nomclient $pommade <br>";
    echo "Nous sommes le ".$date ;
    echo " et vous désirez commander le produit suivant : $produit <br>";

    /*appel d'un autre script dans lequel
    on pourra utiliser $nomclient et $nomproduit et $date
    mais pas $pommade*/
    echo "<a href=suite.php3> Suite</a>";

?>
```

Le script suite.php3

Les variables de session sont accessibles car le lien se fait à travers l'URL ont peut donc continuer à dialoguer avec l'internaute client.



```
<?
    session_start();
    /*on peut utiliser la variable de session $nomclient*/
    echo "Vous êtes : $nomclient <br>";
    /*on ne peut pas utiliser la variable $pommade*/
    echo " $pommade <br>";
    /*on peut utiliser la variable de session $date*/
    echo "Vous vous êtes connectés le : $date <br>";
    /*on peut utiliser la variable de session $nomproduit*/
    echo " et vous nous avez commandé le produit
    suivant : $nomproduit <br>";

    echo "Nous allons mettre fin à la session... à bientôt <br>";
    session_destroy();
    /*appel d'un autre script dans lequel on ne pourra utiliser aucune
variable*/
    echo "<a href=suite_et_fin.php3> Suite et fin</a>";
?>>
```

Le script suite_et_fin.php3

Les variables de session ont été détruites, elles ne sont plus accessibles.

Vous êtes :
Produit commandé :
A la date du :

```
<?
    session_start();
    /*aucune des variables n'est accessible*/
    echo "Vous êtes : $nomclient <br>";
    echo "Produit commandé : $nomproduit <br>";
    echo "A la date du : $date <br>";

?>
```

Exemple 2 : une gestion de client (release one)

On pourrait modifier le script de manière suivante :

- Générer automatiquement un numéro client.
- Générer un cookie sur la machine du client ce qui nous permettrait de le reconnaître s'il revient sur notre site. (en espérant qu'il accepte les cookies). Dans ce cookie on trouverait son numéro client codé et la date de sa dernière connexion.
- Stocker son numéro de client et sa commande dans une table. (cette partie qui complique un peu le script ne sera pas traitée néanmoins il fallait l'évoquer).

Script ident2.php3

```
<?
    /*on démarre une session*/
    session_start();

    $date=date("d/m/y à H:i:s");
    if(empty($HTTP_COOKIE_VARS["NUMCLIENT"])) {
    /*création d'un identifiant utilisateur*/
    $id=md5(uniqid(rand()));
    /* initialisation du nbre de visites*/
    $nb=1;
    /*création d'un cookie nommé NUMCLIENT qui conserve l'identifiant*/
    setcookie("NUMCLIENT",$id,time()+3600);
    /*création d'un cookie nommé NBCMDE qui conserve le nbre de visites*/
    setcookie("NBCMDE",$nb,time()+3600);

    echo "vous êtes un nouveau client <br>";
    }
    else
    {
    /*on récupère le nbre de visite*/
    $nb= $HTTP_COOKIE_VARS["NBCMDE"];
    /*on incrémente de 1 le nbre de visites*/
    $nb++;
    /*modification d'un cookie nommé NBCMDE
    qui conserve le nbre de visites*/
    setcookie("NBCMDE",$nb,time()+3600);;
    echo "vous êtes un ancien client";
    echo " et c'est votre ".$nb."° commande <br>";
    }

    echo "Bonjour, une session a été créée. <br>Elle se nomme ".SID.
    " <br>Cet identificateur de session sera valable
      tant que vous n'aurez pas quitté notre site <br>";
    echo "<hr>";
    /*on récupére le nom du client provenant du formulaire*/
    $nomclient=$client;
    /*on récupére le nom du produit provenant du formulaire*/
    $nomproduit=$produit;
    /*on récupére la date et l'heure dans une variable date*/
    $date=date("d/m/y à H:i:s");

    /*on affecte la variable pommade */
    /*la portée de cette variable ne dépassera pas ce script*/
    $pommade="vous êtes l'un de nos meilleurs clients ";

    /*on crée une variable de session nommée nomclient*/
    session_register("nomclient");
    /*on crée une variable de session nommée nomproduit*/
    session_register("nomproduit");
    /*on crée une variable de session nommée date*/
    session_register("date");
```

```
    echo "Bienvenue $nomclient $pommade <br>";
    echo "Nous sommes le ".$date ;
    echo " et vous désirez commander le produit suivant : $produit <br>";
    /*appel d'un autre script dans lequel
    on pourra utiliser $nomclient et $nomproduit et $date
    mais pas $pommade*/
    echo "<a href=suite.php3> Suite</a>";
?>
```

Pensez à quitter l'ensemble des fenêtres du navigateur pour tester le fonctionnement du cookie et de la session.

Exemple 3 : un compteur sur une page

Ce type de compteur fonctionnera si dans php.ini : register_globals = on

```
<?
  session_register("compteur");
  $compteur++;

echo "Vous avez consulté la page $compteur fois<br>";
echo "<a href=testsession.php3>cliquez ici pour recommencer</A>"
?>
```

Ce type de compteur fonctionnera si dans php.ini : register_globals = off

```
<?
 session_register("compte");
 $HTTP_SESSION_VARS["compte"]++;

echo "Vous avez consulté la page ". $HTTP_SESSION_VARS["compte"] ."
fois<br>";
echo "<a href=testsession_b.php3>cliquez ici pour recommencer</A>"
?>
```

Exemple 3 : un compteur sur plusieurs pages

Script synthese.php3

```
<?
  session_register("compteur1");
  session_register("compteur2");
  session_register("compteur3");


echo "Vous avez consulté la page 1 : $compteur1 fois<br>";
echo "Vous avez consulté la page 2 : $compteur2 fois<br>";
echo "Vous avez consulté la page 3 : $compteur3 fois<br>";
echo "| <a href=page1.php3>vers page 1</A>" ;
echo " | <a href=page2.php3>vers page 2</A>"  ;
echo " | <a href=page3.php3>vers page 3</A> |"  ;
?>
```

Script page1.php3

```
<?
    session_start();
    echo "vous êtes sur la page 1<br>";
    $compteur1++;
    echo "| <a href=page1.php3>vers page 1</A>" ;
    echo " | <a href=page2.php3>vers page 2</A>"  ;
    echo " | <a href=page3.php3>vers page 3</A> "  ;
    echo " | <a href=synthese.php3>vers synthese</A>|"  ;
?>
```

GENERATION ET MANIPULATION D'IMAGES

Principes de base

La déclaration du format d'image utilisé

Header("Content-type: image/png");

Ou

Header("Content-type: image/jpeg");

Ou

Header("Content-type: image/gif"); (le format gif est abandonné depuis la version 1.6 du GD).

La création de l'image

Chaque image est désignée par un identifiant (integer) renvoyé lors de l'ouverture de l'image, cet identifiant sera transmis aux autres fonctions graphiques utilisées dans le script.

$var_ident =imagecreate(taille_x,taille_y);

Cette instruction crée une image de x pixels de large sur y pixels de large.

Ex : $imge=ImageCreate(100,100);

La couleur de l'image

Lorsque l'image est créée il faut lui attribuer des couleurs. Il est préférable de définir des variables ayant des noms significatifs. La définition des couleurs repose sur le mode RVB.

$couleur=ImageColorAllocate(var_ident,R,V,B);

exemples :

$bleu= ImageColorAllocate($imge,0,0,255);

$noir=ImageColorAllocate($imge,0,0,0);

$blanc=ImageColorAllocate($imge,255,255,255);

La première couleur définie correspondra à la couleur de l'arrière plan

Les autres couleurs pourront être utilisées comme bon vous semble.

L'envoi de l'image au navigateur

On maintenant envoyer l'image au navigateur par le biais de l'une des instructions suivantes :

ImageGif($var_ident);

ImagePng($var_ident);

ImageJpeg($var_ident);

Rq : si un nom de fichier est spécifié l'image sera enregistrée.

ImageGif($var_ident,"nom_fichier.gif");

ImagePng($var_ident,"nom_fichier.png");

ImageJpeg($var_ident,"nom_fichier.jpeg");

Destruction de l'image

L'image doit être supprimée afin de libérer les ressources.

ImageDestroy($var_ident);

Exemple de script :

Création d'une image de couleur bleue

```
<?
   //envoi d'un entête précisant que l'image est format png
   header("content-type: image/png");

   //Création d'une image de 100 par 100 pixels
   $imge=imagecreate(100,100);

   //définition de la couleur d'arrière plan
   $bleu=imagecolorallocate($imge,0,0,255);

   //envoi de l'image au navigateur
   imagepng($imge);

   //destruction de l'image
   imagedestroy($imge);
?>
```

Quelques fonctions de traçage de formes

Tracer une ligne : Imageline

Imageline($var_ident,x1,y1,x2,y2,$couleur);

Imageline trace une ligne de couleur $couleur entre les coordonnées x1, y1 et x2,y2

Exemple :

```
<?
   //envoi d'un entête précisant que l'image est format png
   header("content-type: image/png");

   //Création d'une image de 100 par 100 pixels
   $imge=imagecreate(100,100);

   //définition de la couleur d'arrière plan
   $bleu=imagecolorallocate($imge,0,0,255);
   $blanc= imagecolorallocate($imge,255,255,255);

   //trace une ligne blanche du coin supérieur gauche au milieu de l'image
   imageline($imge,0,0,50,50,$blanc);

   //trace une ligne blanche du coin inférieur gauche au milieu de l'image
   imageline($imge,0,100,50,50,$blanc);

   //envoi de l'image au navigateur
   imagepng($imge);

   //destruction de l'image
   imagedestroy($imge);
?>
```

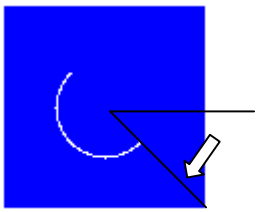
Tracer une courbe : Imagearc

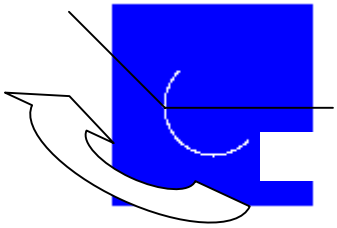
imagearc ($var_ident, cx, cy, w, h, s, e, $couleur)

imagearc() dessine une ellipse partielle, centrée sur cx, cy, (le coin en haut à gauche est l'origine (0,0)) dans l'image référencée par $var_ident. w et h spécifient la largeur et la hauteur de l'ellipse, tandis que le début et la fin de l'arc sont donnés en degrés, par les arguments s et e.

imagearc ($imge, 50, 50, 50, 50, 45,225, $blanc);

Dans l'exemple précédent, on crée un cercle centré sur le point 50, 50 qui aura 50 pixels de large, 50 de haut commencera à 45° et finira à 225°.





exemple :

```
//trace un demi arc de cercle
imagearc ($imge, 50, 50, 50, 50, 0,180,  $blanc);

//trace un cercle
imagearc ($imge,40, 25, 7, 7, 0,360,  $blanc);

//trace une ellipse
imagearc ($imge, 60, 25, 10, 5, 0,360,  $blanc);
```

Tracer un rectangle : Imagerectangle

imagerectangle ($var_ident, x1, y1, x2, y2, $couleur)

imagerectangle() dessine un rectangle dans la couleur $couleur, dans l'image $var_ident, et en commencant au point supérieur gauche (x1,y1), et en finissant au point inférieur droit (x2,y2). Le coin supérieur gauche est l'origine (0,0).

```
//trace un demi arc de cercle
imagearc ($imge, 50, 50, 50, 50, 0,180,  $blanc);

//trace un cercle
imagearc ($imge,40, 25, 7, 7, 0,360,  $blanc);

//trace une ellipse
imagearc ($imge, 60, 25, 10, 5, 0,360,  $blanc);

//trace un rectangle
imagerectangle ($imge, 10, 10, 90, 90,  $blanc);
```

Tracer un polygone : Imagepolygon

imagepolygon, x1, y1, x2, y2, $couleur)

imagepolygon () dessine un rectangle dans la couleur $couleur, dans l'image $var_ident, et en commencant au point supérieur gauche (x1,y1), et en finissant au point inférieur droit (x2,y2). Le coin supérieur gauche est l'origine (0,0).

imagepolygon (($var_ident, array points, num_points, $couleur)

imagepolygon() dessine un polygone dans l'image $var_ident. points est un tableau PHP qui contient les sommets du polygone sous la forme :

points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc.

num_points est le nombre de sommets.

```
//définition du nombre de points
$num_points=4;

//définition des points
$points[0]=30;   $points[1]=30;
$points[2]=50;   $points[3]=50;
$points[4]=70;   $points[5]=30;
$points[6]=50;   $points[7]=70;

//trace un polygone
imagepolygon ($imge, $points,$num_points, $blanc);

//trace un cercle
imagearc ($imge,40, 25, 7, 7, 0,360,  $blanc);

//trace une ellipse
imagearc ($imge, 60, 25, 10, 5, 0,360,  $blanc);

//trace un rectangle
imagerectangle ($imge, 10, 10, 90, 90,  $blanc);
```

Tracer un polygone plein : ImageFilledPolygon

Voir aussi ImageFilledRectangle et ImageFillToBorder

imagefilledpolygon ($var_ident, array points, num_points, $couleur)

imagefilledpolygon() dessine un polygone rempli dans l'image *i*$var_ident. *points* est un tableau PHP qui contient les sommets des polygones sous la forme :. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *num_points* est le nombre total de sommets.

```
//trace un polygone fond blanc
imagepolygon ($imge, $points,$num_points, $blanc);
```

Quelques fonctions de manipulation de texte

Ecrire une chaine de caractères : ImageString()

imagestring ($var_ident, font, x, y, s, $couleur)

imagestring() dessine une la chaîne sur une ligne horizontale, dans l'image $var_ident, aux coordonnées (x,y) (le coin supérieur gauche est l'origine (0,0)) dans la couleur $couleur. L'argument de police font vaut 1, 2, 3, 4 ou 5, une des 5 tailles de polices par défaut).


```
<?
   //envoi d'un entête précisant que l'image est format png
   header("content-type: image/png");

   //Création d'une image de 100 par 100 pixels
   $imge=imagecreate(150,200);

   //définition de la couleur d'arrière plan
   $bleu=imagecolorallocate($imge,0,0,255);

   //définition des couleurs utilisées
   $blanc= imagecolorallocate($imge,255,255,255);
   $rouge= imagecolorallocate($imge,255,0,0);

   $string="Formation PHP";

   //ecrit le texte dans la police 1 par défaut
   imagestring($imge,1,10,20,$string,$blanc);

   //ecrit le texte dans la police 2 par défaut
   imagestring($imge,2,10,40,$string,$blanc);
```

```
//ecrit le texte dans la police 3 par défaut
imagestring($imge,3,10,60,$string,$blanc);

//ecrit le texte dans la police 4 par défaut
imagestring($imge,4,10,80,$string,$blanc);

//ecrit le texte dans la police 5 par défaut
imagestring($imge,5,10,100,$string,$blanc);

//envoi de l'image au navigateur
imagepng($imge);

//destruction de l'image
imagedestroy($imge);
?>
```

Ecrire une chaîne de caractères verticale : ImageStringup()

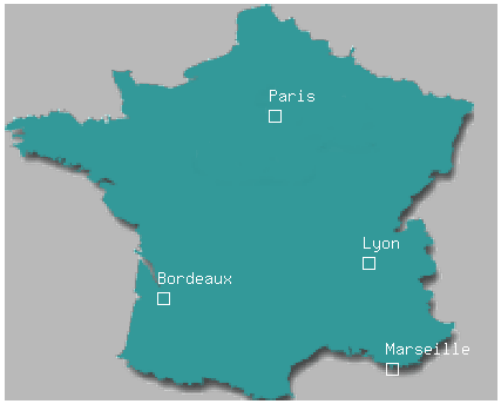
imagestringup ($var_ident, font, x, y, s, $couleur)


```
$string="Formation PHP";

//écrit le texte verticalement dans la police 5 par défaut
imagestringup($imge,5,10,180,$string,$blanc);
```

Ouvrir une image existante

imagecreatefrompng (string filename)

imagecreatefrompng() retourne un identifiant d'image représentant une image obtenu à partir du fichier filename.




```
<?
  //envoi d'un entête précisant que l'image est format png
  header("content-type: image/png");

  //Création d'une image de 100 par 100 pixels
  $imge=imagecreatefrompng("france.png");

  $blanc=imagecolorallocate($imge,255,255,255);

  imagerectangle($imge,225,90,235,100,$blanc);
  imagerectangle($imge,305,215,315,225,$blanc);
  imagerectangle($imge,325,305,335,315,$blanc);
  imagerectangle($imge,130,245,140,255,$blanc);

  $string[0]="Paris";
  $string[1]="Lyon";
  $string[2]="Marseille";
  $string[3]="Bordeaux";

  imagestring($imge,4,225,70,$string[0],$blanc);
  imagestring($imge,4,305,195,$string[1],$blanc);
  imagestring($imge,4,325,285,$string[2],$blanc);
```

```
    imagestring($imge,4,130,225,$string[3],$blanc);

    //envoi de l'image au navigateur
    imagepng($imge);

    //destruction de l'image
    imagedestroy($imge);
?>
```

imagecreatefrompng() retourne une chaîne vide en cas d'échec. Elle affiche aussi un message d'erreur, qui s'affiche comme un lien brisé dans un navigateur web.

Exemple de gestion d'erreur lors de la création d'image

```
function LoadPNG ($imgname) {
    $imge = @ImageCreateFromPNG ($imgname); /* Tentative d'ouverture */
    if (!$imge) { /* Vérification */
        $imge = ImageCreate (150, 30); /* Création d'une image blanche */
        $blanc = ImageColorAllocate ($imge, 255, 255, 255);
        $noir  = ImageColorAllocate ($imge, 0, 0, 0);
        ImageFilledRectangle ($imge, 0, 0, 150, 30, $blanc);
        /* Affichage d'un message d'erreur */
        ImageString ($imge, 1, 5, 5, "Erreur de chargement de l'image $imgname", $noir);
    }
    return $imge;
}
```

Formulaire et graphique

A partir du formulaire suivant :

Cliquez sur une ville pour connaître sa météo


```
<html>
<head>
      <title>météo</title>
</head>

<body>
  <form method="post" action="meteo.php3">
  <font face="arial">Cliquez sur une ville pour connaître sa météo</font><br>
  <input type="image" src="meteo_france.jpg"  name="coord">
  </form>

</body>
</html>
```

Le formulaire précédent est un peu spécial puisqu'il n'affiche pas de bouton submit mais une image qui permet la soumission.

<input type="image" src="meteo_france.jpg" name="coord">

En cliquant sur un point de l'image l'utilisateur soumet le formulaire à un script php nommé meteo.php3.

Ce script reçoit 2 variables, coord_x et coord_y qui contiennent les coordonnées du point cliqué sur l'image. Il ne reste plus qu'à traiter en fonction du point cliqué.

Le nom des 2 variables est créé à partir du nom donné à la valeur de l'attribut Name de la balise input, suivi de _x et _y

```
<?
   if ($coord_x>0 && $coord_x<40 && $coord_y>100 && $coord_y<140)  echo "vous êtes à Brest et il vente !!";
   if  ($coord_x>200  &&  $coord_x<240  &&  $coord_y>70  &&  $coord_y<110) echo "vous êtes à Paris et il pleut !!";
   if  ($coord_x>295  &&  $coord_x<325  &&  $coord_y>195  &&  $coord_y<235) echo "vous êtes à Lyon et il neige !!";
   if  ($coord_x>305  &&  $coord_x<345  &&  $coord_y>285  &&  $coord_y<325) echo "vous êtes à Marseille et il fait beau !!";
   if  ($coord_x>110  &&  $coord_x<150  &&  $coord_y>225  &&  $coord_y<265) echo "vous êtes à Bordeaux et il y a de la brume !!";
?>
```

Création dynamique de boutons


```
<html>
<body>
  <!-- Appel d'une image avec un paramètre-->
  <img src="bouton.php3?Accès"></P>
  <img src="bouton.php3?Affaires"></P >
  <img src="bouton.php3?Commander"></P >
```

```
  <img src="bouton.php3?Nous contacter"></P >
</body>
</html>
```

```php
<?php
Header("Content-type: image/png");
/*On crée une chaine de caractères à partir des arguments passés dans un tableau*/
$string=implode($argv," ");
/*lors du passage en argument les espaces ont été encodés*/
$string=urldecode($string);// il faut les décoder
/*appel de l'image png*/
$im = imageCreateFromPng("bouton.png");
$orange = ImageColorAllocate($im, 220, 210, 60);
$px = (imagesx($im)-7.5*strlen($string))/2;
ImageString($im,3,$px,9,$string,$orange);
ImagePng($im);
ImageDestroy($im);
?>
```

implode($argv," ")

Retourne une chaîne constituée de tous les éléments du tableau, pris dans l'ordre, transformés en chaîne, et séparés par glue.

$argv

Tableau des arguments passés au script.

Liste des fonctions sur les images

GetImageSize — Retourne la taille d'une image GIF, JPG ou PNG.

ImageArc — Dessine une ellipse partielle.

ImageChar — Dessine un caractère horizontalement.

ImageCharUp — Dessine un caractère verticalement.

ImageColorAllocate — Alloue une couleur pour une image.

ImageColorDeAllocate — Désallouune une couleur pour une image

ImageColorAt — Retourne l'index de la couleur d'un pixel donné.

ImageColorClosest — Retourne l'index de la couleur la plus proche d'une couleur donnée.

ImageColorExact — Retourne l'index de la couleur donnée.

ImageColorResolve — Retourne l'index de la couleur donnée, ou la plus proche possible.

ImageGammaCorrect — Applique une correction gamma à l'image

ImageColorSet — Change la couleur dans une palette à l'index donné.

ImageColorsForIndex — Retourne la couleur associée à un index.

ImageColorsTotal — Calcule le nombre de couleur d'une palette.

ImageColorTransparent — Définit la couleur transparente.

ImageCopy — Copie une partie d'une image

ImageCopyResized — Copie et redimensionne une partie d'une image.

ImageCreate — Crée une nouvelle image.

ImageCreateFromGif — Crée une nouvelle image à partir d'un fichier ou d'une URL.

ImageCreateFromJPEG — Crée une nouvelle image JPEG à partir d'un fichier ou d'une URL

ImageCreateFromPNG — Crée une nouvelle image PNG à partir d'un fichier ou d'une URL

ImageDashedLine — Dessine une ligne pointillée.

ImageDestroy — détruit une image.

ImageFill — Remplit.

ImageFilledPolygon — Dessine un polygone rempli.

ImageFilledRectangle — Dessine un rectangle rempli.

ImageFillToBorder — remplir avec une région avec une couleur spécifique.

ImageFontHeight — Retourne la hauteur de la police.

ImageFontWidth — Retourne la largeur de la police.

ImageGif — Envoie une image GIF vers un navigateur ou un fichier.

ImagePNG — Envoie une image PNG vers un navigateur ou un fichier.

ImageJPEG — Envoie une image JPEG vers un navigateur ou un fichier.

ImageInterlace — Active ou désactive l'entrelacement.

ImageLine — Draw a line.

ImageLoadFont — Charge une nouvelle police.

ImagePolygon — Dessine un polygone.

ImagePSBBox — Retourne le rectangle entourant un texte et dessiné avec une police PostScript Type1.

ImagePSEncodeFont — Change le codage vectoriel d'un caractère dans une police.

ImagePSFreeFont — Libère la mémoire occupée par une police PostScript Type 1.

ImagePSLoadFont — Charge une police PostScript Type 1 depuis un fichier.

ImagePsExtendFont — Etend ou condense une police de caractères

ImagePsSlantFont — Inclidnet une police de caractères

ImagePSText — Dessine un texte sur une image avec une police PostScript Type1.

ImageRectangle — Dessine un rectangle.

ImageSetPixel — Dessine un pixel.

ImageString — Dessine une chaîne horizontale.

ImageStringUp — Dessine une chaîne verticale.

ImageSX — Retourne la largeur d'une image.

ImageSY — Retourne la hauteur de l'image.

ImageTTFBBox — retourne le rectangle entourant un texte et dessiné avec une police TrueType.

ImageTTFText — Dessine un texte avec une police TrueType.

ImageTypes — Retourne les types d'images supportés par la version courante de PHP

LIAISON ODBC ACCESS 97 MYSQL

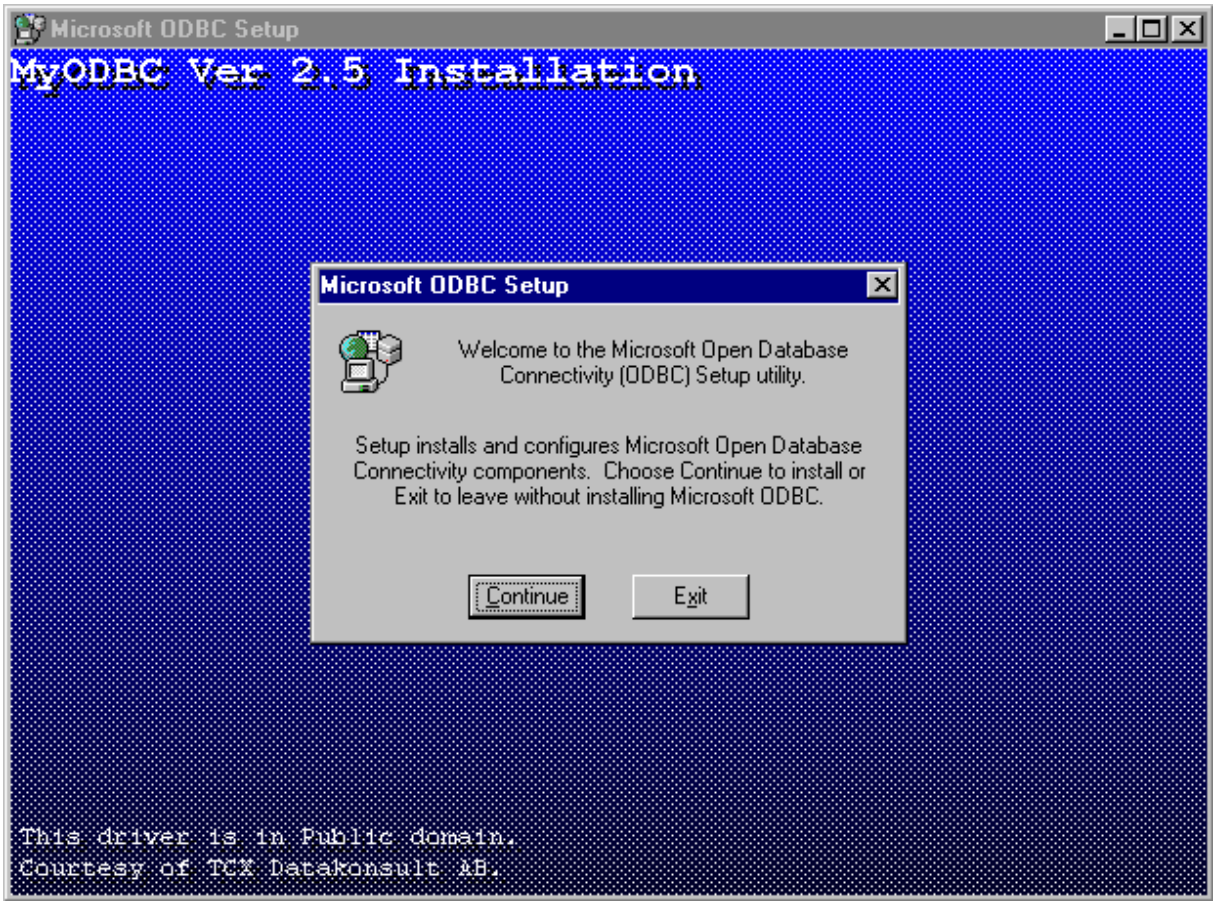
Objectif :

A partir d'un SGBD local (exemple Access 97) mettre à jour une base de données distante (exemple : SGBD Mysql).

Technique

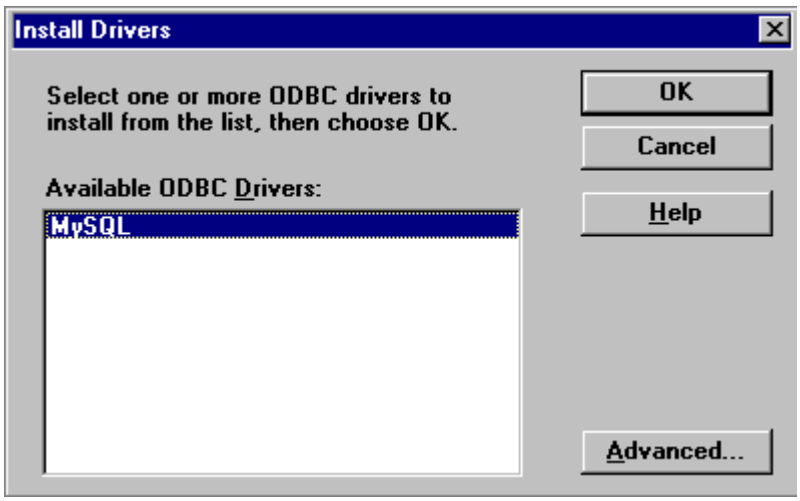
Télécharger le driver ODBC myodbc (moi j'utilise myodbc-2.50.22-win95.zip récupérable ici : ftp://ftp.inforoutes-ardeche.fr/pub/pc/win95/odbc/)

Décompressez et installez le logiciel.

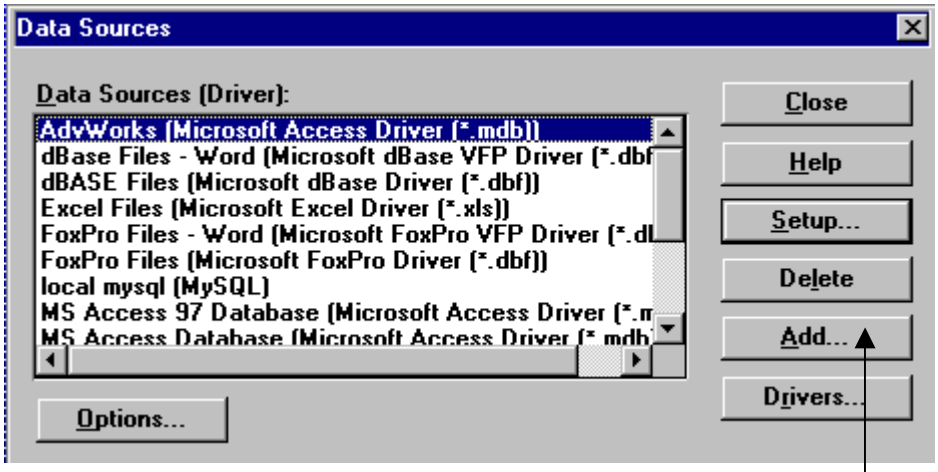


Cliquez sur Continue

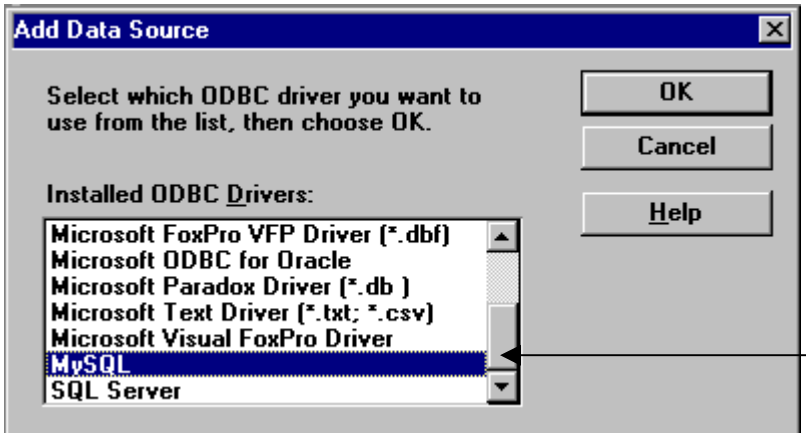
Dans la liste, venez sélectionner Mysql ;))



puis OK

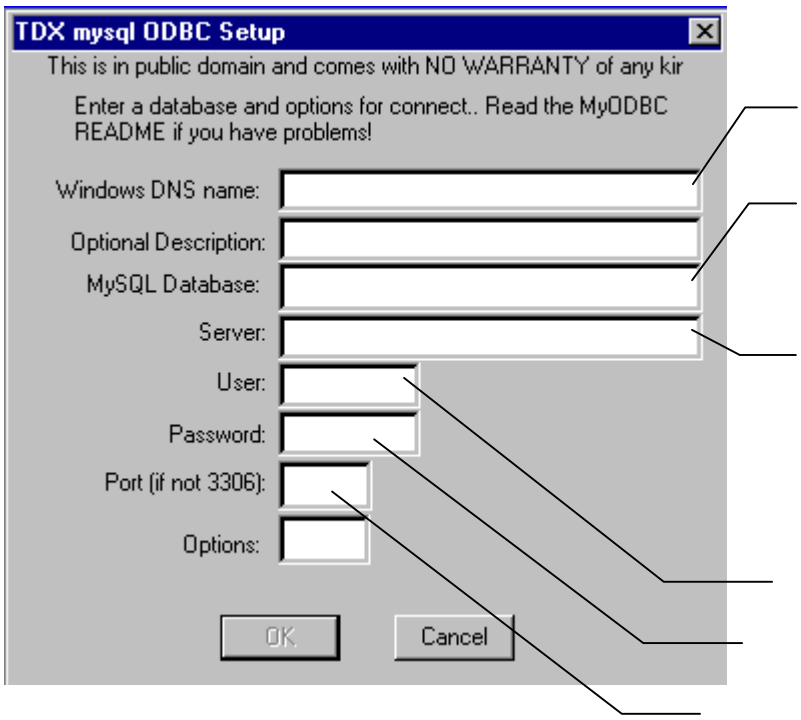


Cliquez sur Add…



Sélectionnez Mysql

puis OK



Nom de la connexion

Nom de la base de données distante à lier ou

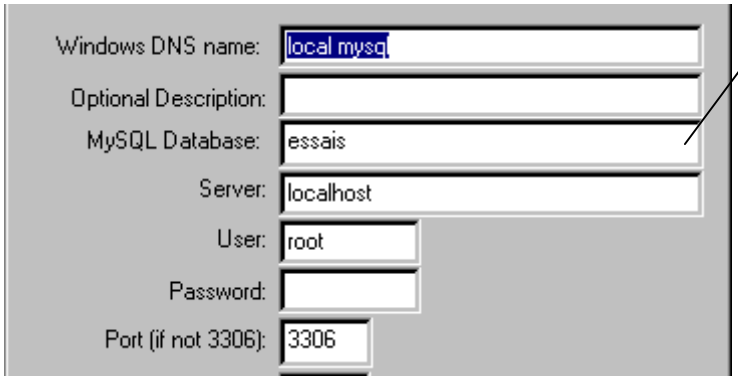
Adresse ip du serveur
ou
URL du serveur SGBD
ou
localhost si vous travaillez en local

Nom d'utilisateur

Mot de passe

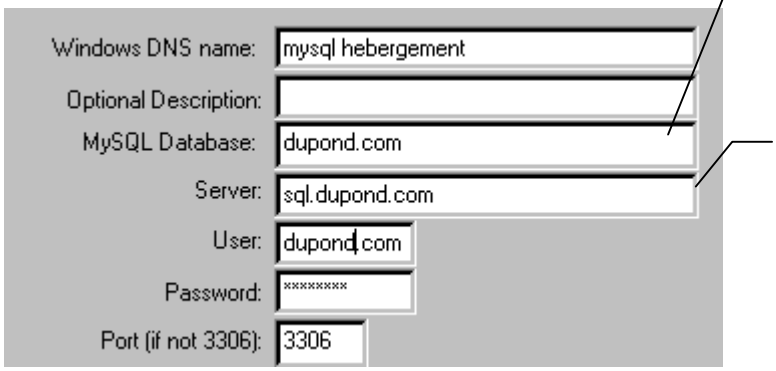
Port : 3306

Exemple 1 : vous avez installé un serveur Apache et Mysql en local



Nom de la base de données mysql distante à

Exemple 2 : vous avez un hébergeur



Nom de la base de données mysql distante à lier ou à mettre à jour.

URL permettant d'accéder à

Utilisation

Conditions pour que cela fonctionne :

- Vous avez une base de données en ligne (SGBD mysql. Base nommée mailing_list).
- Dans cette base existe une table nommée inscrits. Cette table comporte 4 champs. num, nom, prenom, email.
- Vous avez une base de données en local (SGBD Access. Base nommée alim_mailing_list).
- Dans cette base existe une table nommée alim_inscrits. Cette table comporte 4 champs. num, nom, prenom, email.

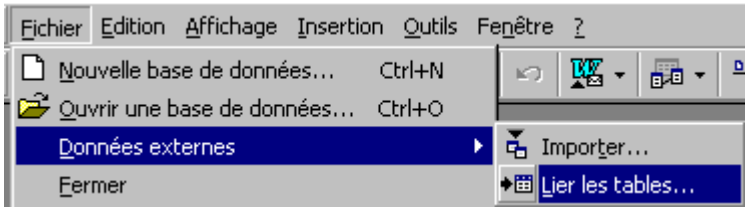
Dans la base de données locale et plus particulièrement dans la table alim_inscrits créez quelques enregistrements.

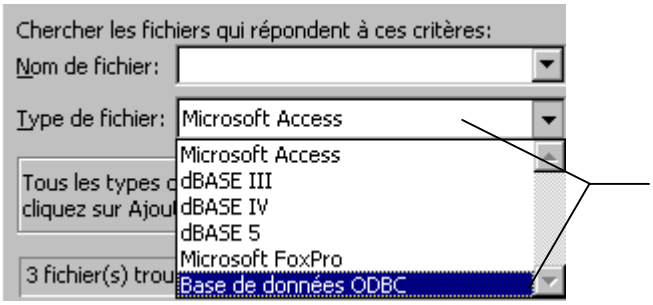
Lier une table locale et une table distante

Dans Access ouvrez la base de données alim_mailing_list

sélectionnez la table alim_inscrits puis passez sur la commande

Fichier – Données externes – Lier les tables

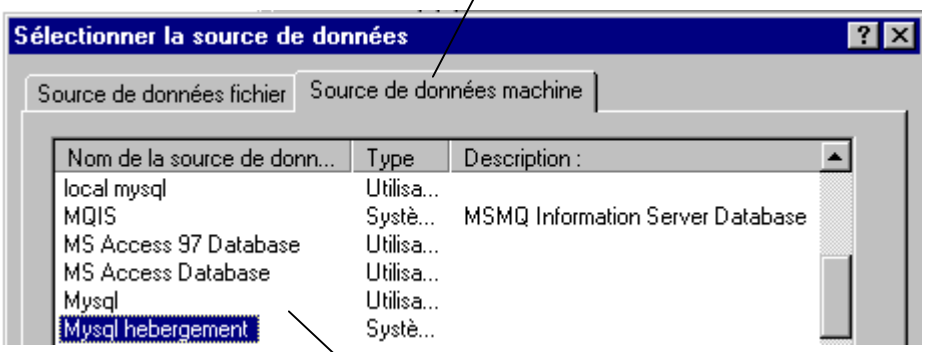




Dans type de fichier :

Venez choisir Base de

Choisir source de données

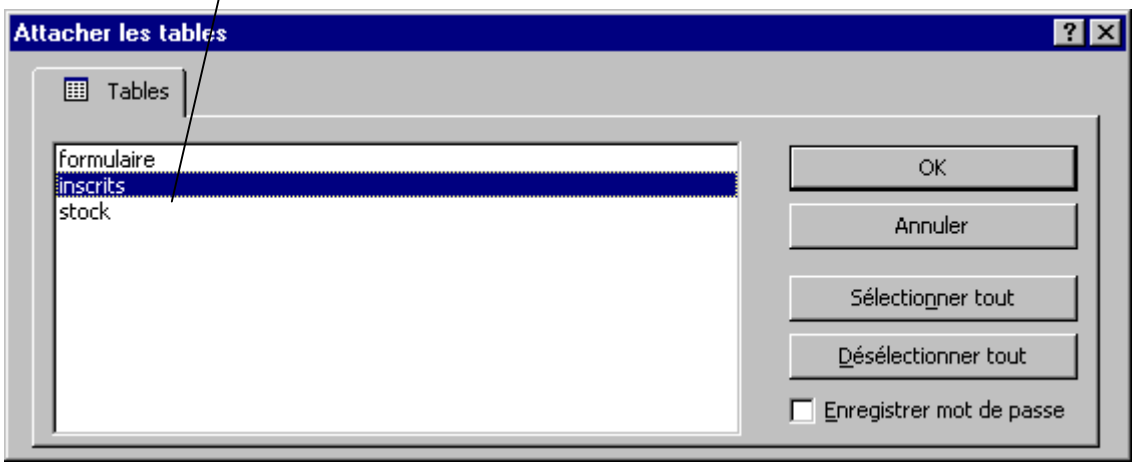


Dans la liste proposée

Venez choisir Mysql hebergement

(c'est la source de données machine définie lors de

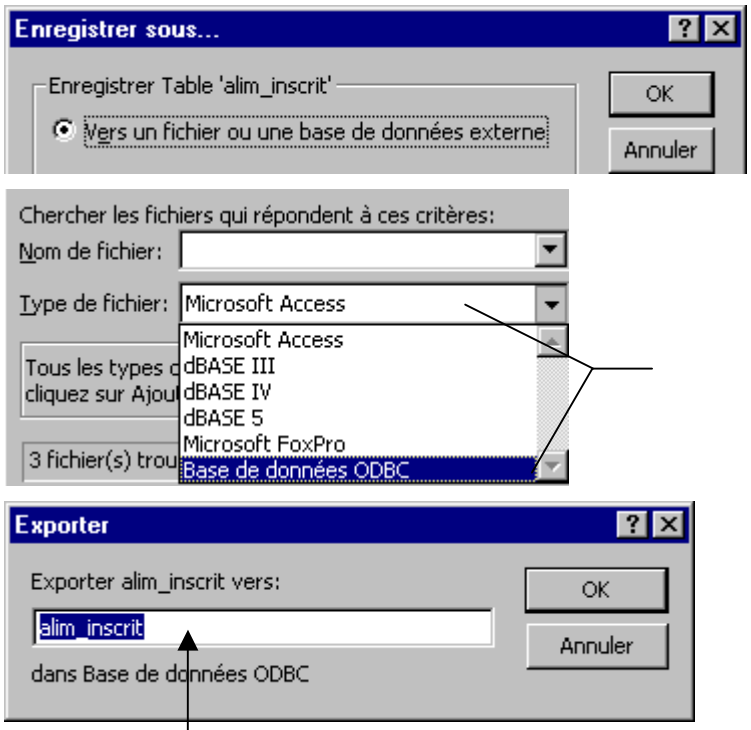
Dans la liste proposée, venez choisir **inscrits** (c'est la table à laquelle vous



Dans la liste proposée , venez double cliquer sur **inscrits** (c'est la table à laquelle vous êtes liés)

Vous pouvez maintenant modifier le contenu de votre base en ligne.

Exportation d'une base locale access vers mysql

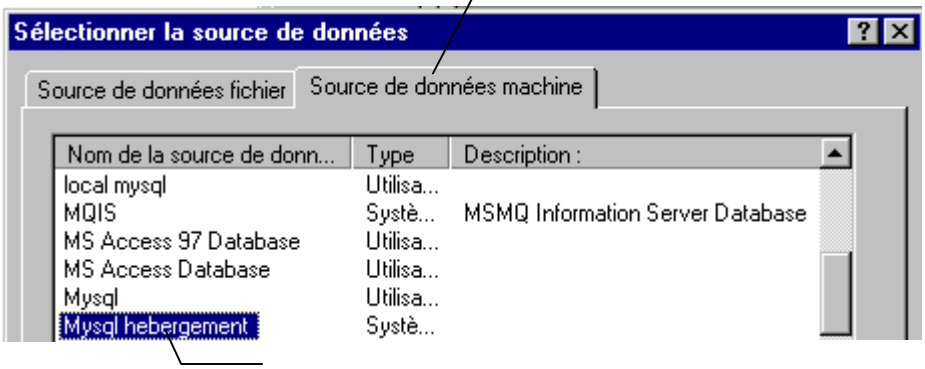


Dans type de fichier :

Venez choisir Base de

Il s'agit d'un export il faut donc que la table n'existe pas déjà en ligne.

Choisir source de données



Dans la liste proposée

Venez choisir Mysql hebergement. (c'est la source de données machine définie lors de l'installation de myodbc)

mysql_affected_rows — Retourne le nombre de lignes affectées lors de la dernière opération SQL.

mysql_change_user — Change le nom de session de l'utilisateur actif.

mysql_close — Ferme la connexion MySQL.

mysql_connect — Ouvre une connexion à un serveur MySQL.

mysql_create_db — Crée une base de données MySQL.

mysql_data_seek — Déplace le pointeur interne de résultat.

mysql_db_name — Lit les noms des bases de donné

mysql_db_query — Envoie une requête MySQL à un serveur MySQL.

mysql_drop_db — Efface une base de données MySQL.

mysql_errno — Retourne le numéro de message d'erreur de la dernière opération MySQL.

mysql_error — Retourne le texte associée avec l'erreur générée lors de la dernière requête.

mysql_fetch_array — Retourne une ligne de résultat sous la forme d'un tableau associatif.

mysql_fetch_assoc — Lit une ligne de résultat dans un tableau associatif

mysql_fetch_field — Retourne les données enregistrées dans une colonne, à partir d'un résultat, et retourne un objet.

mysql_fetch_lengths — Retourne la taille de chaque colonne d'une ligne de résultat.

mysql_fetch_object — Retourne les lignes résultats sous la forme d'un objet.

mysql_fetch_row — Retourne une ligne de résultat sous la forme d'un tableau.

mysql_field_flags — Retourne le sémaphore associé à la colonne spécifiée dans le résultat courant.

mysql_field_name — Retourne le nom d'une colonne

mysql_field_len — Retourne la longueur du champs spécifié.

mysql_field_seek — Place le pointeur de résultat à un offset donné

mysql_field_table — Retourne le nom de la table oú se trouve une colonne

mysql_field_type — Retourne le type de la colonne spécifiée dans le résultat courant.

mysql_free_result — Efface le résultat de la mémoire.

mysql_insert_id — Retourne l'identifiant généré par la dernière requête INSERT.

mysql_list_dbs — Liste les bases de données disponibles sur le serveur MySQL.

mysql_list_fields — Liste les champs du résultat MySQL.

mysql_list_tables — Liste les tables d'une base de données.

mysql_num_fields — Retourne le nombre de champs d'un résultat.

mysql_num_rows — Retourne le nombre de ligne d'un résultat.

mysql_pconnect — Ouvre une connexion persistante à un serveur MySQL.

mysql_query — Envoie une requête SQL à un serveur MySQL.

mysql_result — Retourne un champs d'un résultat.

mysql_select_db — Sélectionne une base de données MySQL.

mysql_tablename — Retourne le nom de la table qui contient le champs spécifié.

FONCTIONS POSTGRESQL

pg_Close — Termine une connexion PostgreSQL.
pg_cmdTuples — Retourne le nombre de tuples affectés.
pg_Connect — Ouvre une connexion.
pg_DBname — Nom de la base de données.
pg_end_copy — Synchronise avec le serveur PostgreSQL
pg_ErrorMessage — Message d'erreur.
pg_Exec — Exécute une requête.
pg_Fetch_Array — Lit une ligne dans un tableau.
pg_Fetch_Object — Lit une ligne dans un objet.
pg_Fetch_Row — Lit une ligne dans un tableau.
pg_FieldIsNull — Teste si un champs est à NULL.
pg_FieldName — Retourne le nom d'un champs.
pg_FieldNum — Retourne le numéro d'une colonne.
pg_FieldPrtLen — Retourne la taille imprimée.
pg_FieldSize — Retourne la taille interne de stockage d'un champs donné.
pg_FieldType — Retourne le type d'un champs donné par index.
pg_FreeResult — Libère la mémoire
pg_GetLastOid — Retourne le dernier identifiant d'objet.
pg_Host — Retourne le nom d'hôte.
pg_loclose — Ferme un objet de grande taille.
pg_locreate — Crée un objet de grande taille.
pg_loexport — Exporte un objet de grande vers un fichier
pg_loimport — Importe un objet de grande taille depuis un fichier
pg_loopen — Ouvre un objet de grande taille.
pg_loread — Lit un objet de grande taille.
pg_loreadall — Lit un objet de grande taille en totalité.
pg_lounlink — Efface un objet de grande taille
pg_lowrite — Ecrit un objet de grande taille
pg_NumFields — Retourne le nombre de champs
pg_NumRows — Retourne le nombre de lignes.
pg_Options — Retourne les options.
pg_pConnect — Etablit une connexion persistante.
pg_Port — Retourne le numéro de port.

pg_put_line — Envoie une chaîne au serveur PostgreSQL

pg_Result — Retourne les valeurs d'un identifiant de résultat.

pg_set_client_encoding — Choisi l'encodage du client

pg_client_encoding — Lit l'encodage du client

pg_trace — Active le suivi d'une connexion PostgreSQL

pg_tty — Retourne le nom de tty.

pg_untrace — Termine le suivi d'une connexion PostgreSQL