

# Techniques de Programmation Internet

## Année Spéciale ENSIMAG

ENSIMAG 2001 - 2002

James L. Crowley

Séance 8

18 mars 2002

## PHP et MySQL

Plan :

SQL.....	1
MYSQL.....	2
Administration.....	3
Un intro rapide aux fonctions de MySQL.....	3
Php.....	5
Variables.....	6
Mélange de PHP et HTML.....	7
Les chaînes de caractères.....	8
Un exemple de PHP et MySQL .....	11

### SQL

SQL est un langage de command pour les bases de données inventé par E. F. Codd dans les années 70. Parce que SQL se ressemble à la langue naturel, il est facile d'apprendre. Par exemple :

```
SELECT name FROM people WHERE name LIKE `Stac%`.
```

```
SELECT value FROM table
```

S'exprime en PERL comme

```
$++ ;($*++/$ !) ;$&$^,, ;$!
```

SQL a été conçu pour les non-informaticiens. Il a été adopté par ORACLE en 1980 pour sa système. Il est considéré comme la norme en langage de commande de base de donnée. Depuis 1989, SQL est un norme ANSI (SQL92) ou SQL2.

À fin d'être petit et efficace, MySQL est réalise qu'un une sous-ensemble de la fonctionnalité de SQL.

## **MYSQL**

MYSQL est un "Open Source ShareWare" système de gestion de base relationnel de données développé par Michael Widenius, de la société suédoise TcX. À l'origine (en 1979) Widenius a développé un système de gestion de base de donnée nommée « UNIREG ». En 1995, Widenius a écrit un nouveau système pour les besoins interne de sa société. Il a fourni un API fondé sur SQL en appuyant sur les composants de UNIREG.

MySQL est un petit serveur de base de donnée relationnelle avec un interpréteur de commande ANSI SQL. MySQL peut être compilé sur les plusieurs plates-formes (UNIX, Windows, etc.). Sur Unix, MySQL exploite les « threads » pour une gain important d'efficacité. MySQL peut être exécuté sur Windows NT et comme une procès normale de Windows 95/98. (Attention, les threads de Windows 95 ont un problème de fuite de mémoire).

MySQL a l'avantage d'être petit, facile a utilisé, standard (grâce à son usage de SQL), ouvert (distribué avec sa source), robuste, et gratuit. Il est rapidement devenu l'outil privilège pour la base de données des serveurs.

MySQL est disponible sur les nombreux sources. Par exemple en France voir : <http://www.minet.net/devel/mysql>

MySQL est une base « multi-thread », avec un thread par connection. MySQL comprend quelques fonctions de plus par rapport au SQL, comme : ENCRYPT, WEEKDAY and IF.

On installe MySQL comme un processus de fond (demon en Unix). Sur Unix ou Windows (98 et NT). On démarre MySQL avec :

```
/usr/local/mysql/bin/safe_mysql&.
```

Le script « safe\_mysql » assure que il y qu'un mysql. Si mysql se plante, safe\_mysql le relance.

On administre MySQL avec le système mysqladmin. Par exemple pour créer un nouvelle base, DATABASE, on tape :

```
mysqladmin -p create DATABASE
```

(-p pour prompte pour le mot de passe de ROOT).

Pour détruire une base de donnée il y a :

```
mysqladmin -p drop DATABASE
```

Pour arrêter une base de donnée, on tape :

```
mysqladmin -p shutdown
```

Il y a une ensemble riche de commande de « status » possible.  
Voir :

```
mysqladmin -h
```

### **Un intro rapide aux fonctions de MySQL.**

Voici trois exemples de commandes MySQL :

```
CREATE TABLE people (name CHAR(10))  
INSERT INTO people VALUES ('me')
```

```
SELECT name FROM people WHERE name LIKE '%e'.
```

Les commandes en majuscule sont les mots clés, mais on aura peut les taper avec un mélange. En SQL et en MySQL les commandes ne sont pas sensible aux majuscules. Par exemple, on peut taper :

```
Create Table people (name CHAR(10))
```

Par contre, les noms de bases et les tables (définis par l'utilisateur) SONT sensible.

« People » n'est pas équivalent à « people ».

Les fonctions de SQL concernent l'addition, modification, extraction, et lecture des données d'une base.

Une commande SQL commence toujours par un verbe.

Une table est une structure de donnée fondamentale dans une base de donnée. Une table est composée de colonnes. Chaque colonne est typée. La syntaxe est

```
Create Table table_name (col_name type [modifiers])  
                        [,col_name type [modifiers]])
```

Les types sont : INT, REAL, CHAR(length), TEXT(length), DATE, TIME.

Les colonnes peuvent être « unsigned ».

## Php

PHP et MySQL sont une couple idéal pour des sites web.

PHP est un langage de script interprété par le serveur.

(A l'opposition au langage interprété par le Navigateur comme «JavaScript»)

Des documentation sont disponibles à

[PHP Reference Manual](http://www.php.net/manual/langref.php)

PHP est une logiciel ouvert que support authentication des utilisateurs, XML, les image dynamique, WDDX, mémoire partagé, et création des documents PDF.

Voici un exemple de PHP.

```
<H1> lets test PHP </H1>
<H2> Try Hello World</H2>
<html>
<body>
  <?php
  $myvar = "Hello World";
  echo $myvar;
  ?>
</body>
</html>
```

Si on accède à ce page au travers APACHE, on obtiendrait :

```
<H1> lets test PHP </H1>
<H2> Try Hello World</H2>
<html>
<body>
</body>
</html>
```

La partie :

```
<?php $myvar = "Hello World";    echo $myvar; ?>
```

est envoyé au PHP par APACHE. Le symbole `<?php` indique le début d'un bloc de code. Le symbole `?>` indique le fin. Des expressions PHP peuvent être placés n'importe où dans le code.

Le symbole `« ; »` sert de marqueur pour les fins de commandes. PHP n'interprète pas les sauts de lignes. Donc,

```
<?php
$myvar = "Hello World";
echo $myvar;
?>
```

et

```
<?php $myvar = "Hello World"; echo $myvar; ?>
```

sont strictement équivalents.

Le symbole de fermeture `?>` vaut un `« ; »`.

## Variables

Les variables, comme `$myvar` commencent par un `« $ »`. Ils peuvent représenter les chaînes, les nombres ou les tableaux.

PHP utilise une syntaxe proche de C, C++ et Unix. Par exemple :

```
<?php
echo "This is a test"; // This is a one-line c++ style comment
/* This is a multi line comment
   yet another line of comment */
echo "This is yet another test";
echo "One Final Test"; # This is shell-style style comment
?>
```

Les noms et mots réservés ne sont pas sensibles aux majuscules. Par exemple :

```
<H2> See the environmental variable $HTTP_USER_AGENT</H2>
<?php echo "You are using <P>"; ?>
<?php echo $HTTP_USER_AGENT; ?>
```

**Mélange de PHP et HTML**

Il est possible de mélanger l'HTML au milieu de l'PHP. Voici un exemple :

```
<H2> to get a list of environment variables with phpinfo()
</H2>
<?php phpinfo()?>
<H2> Test an environment variable </H2>
<h3> "strstr()" searches for a string (arg2) in arg1.
<?php
    if(strstr($_HTTP_USER_AGENT, "MSIE"))
    {
        echo "You are using Internet Explorer<br>";
    }
?>

<H2> You can jump in and out of PHP! </H2>
<?php
    if(strstr($_HTTP_USER_AGENT, "MSIE")) {
?>
        <center><b>You are using Internet Explorer</b></center>

<?php} else {?>

        <center><b>You are not using Internet
Explorer</b></center>
<?php}?>
```

PHP peut interpréter les données d'une FORM.

```
<H2> Php can allow form actions from within
a second HTML (php) file</H2>
<form action="action.php3" method="POST">
Your name: <input type="text" name="name">
You age: <input type="text" name="age">
<input type="submit">
</form>
```

Avec le fichier : action.php3

```
Hi <?php echo $name?>. You are <?php echo $age?> years old.
```

PHP décode les arguments et construit un « hash », automatiquement. (Donc les blocs de codes magique de PERL ne sont pas nécessaire en PHP.

Avec les « add-ins » on peut trouver 700 fonctions en PHP.

PHP a les types de variables suivants :

Array, float, integer , object et string

Les types de variables sont déterminés automatiquement par le contexte, pendant l'exécution. Mais il est toujours possible de forcer un typage avec le commande `settype()` .

### Les chaînes de caractères.

Les chaînes sont spécifiés avec les quotes «"» et «' » comme en PERL.

Avec le quote «"» les variables sont interprétés. Comme avec C et PERL, le caractère « \ » sert de échappement pour les caractères spéciales.

Exemples :

```
\n  linefeed (LF or 0x0A in ASCII)
\r  carriage return (CR or 0x0D in ASCII)
\t  horizontal tab (HT or 0x09 in ASCII)
\\  backslash
\$  dollar sign
\"  double-quote
\[0-7]{1,3} Les caractères en octal
\x[0-9A-Fa-f]{1,2} Les caractères en hexadécimal.
```

Il est possible de échapper des autres caractères, mais ceci génère une message d'erreur. On peut également échapper avec « ' ». Dans ce cas, les seul échappement sont « \\ » et « \' » Les variables ne sont pas interprétés à l'intérieur des « \' ».

On peut également délimiter les chaînes avec « <<<xxx » ou xxx représente n'importe quel chaîne. Le chaîne est terminé par xxx sur un debut de ligne. Le chaîne xxx doit contenir que des alphanumérique.

Exemple :

```
<?php
    $str = <<<EOD
    Example of string
    spanning multiple lines
    using heredoc syntax.
EOD;
```

Voici une deuxième exemple :

```
/* More complex example, with variables. */
class foo {
    var $foo;
    var $bar;

function foo()
{
    $this->foo = 'Foo';
    $this->bar = array('Bar1', 'Bar2', 'Bar3');
}
}

$foo = new foo();
$name = 'MyName';

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

Comme avec PERL, les chaînes peuvent être concaténées par '.' (dot). L'opérateur « + » ne marchera pas pour la concaténation. On peut obtenir le « Nième » caractère d'un chaîne par indexation numérique.

Voici un exemple :

```
<?php
/* Assigning a string. */
```

```
$str = "This is a string";

/* Appending to it. */
$str = $str . " with some more text";

/* Another way to append, includes an escaped newline. */
$str .= " and a newline at the end.\n";

/* This string will end up being '<p>Number: 9</p>' */
$num = 9;
$str = "<p>Number: $num</p>";

/* This one will be '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';

/* Get the first character of a string */
$str = 'This is a test.';
$first = $str[0];

/* Get the last character of a string. */
$str = 'This is still a test.';
$last = $str[strlen($str)-1];
?>
```

**Un exemple de PHP et MySQL**

(attention : Cet exemple n'est pas encore testé. )

Soit un table SQL :

Field	Type	Length	Not-Null	Unique-Index
ID	int	100	Y	N/A
name	char	1000	N	N/A
sex	int	5	N	N/A
eye-color	char	10	N	N/A

Une forme en HTML : search.html

```
<HTML><HEAD><TITLE> A Form for person </TITLE></HEAD>
<BODY BGCOLOR="white">
<H1>Retrieve people from data-base</H2>
<P>
<FORM METHOD=POST ACTION="/cgi-bin/jlc/results.html">
Sex<SELECT NAME="sex">
<OPTION>Male
<OPTION>Female
</SELECT>
SELECT NAME="eye-color">
<OPTION>Blue
<OPTION>Green
<OPTION>Brown
</SELECT>
<P>
<INPUT TYPE="SUBMIT" VALUE=" QUERY"><INPUT TYPE=RESET>
</FORM></BODY></HTML>
```

Le fichier PHP/HTML results.html :

```
<HTML><HEAD><TITLE> Search Results </TITLE></HEAD>
<BODY BGCOLOR="white">
<H1>Here are the results</H1>
<P>

<?php
// The query will have the form
//  SELECT * FROM DATA WHERE sex=male AND eye-color=Blue

// Build the query template
$query = "SELECT * FROM DATA WHERE ";

//  append the modifiers
$query .= " sex = '$sex' AND eye-color = '$eye-color'"

// Make the query
$results = mysql_query("data", $query);

// how many answers?
$num_results = mysql_numrows($results);
>

<UL>

// Print the results
<?php if (! $num_results); >
<H2> No Persons in the data base fit with those eyes and
sex</H2>
<?php else {
    while ($i < $num_results) {
        $id[$i] = msyql_result($result, $i, "id")
        $name[$i] = msyql_result($result, $i, "name")
        printf("<LI>  Person  $name[$i]  with ID  $id[$i]
</LI>");
    }
}
</UL>
<A HREF="search.html">Search again</A>
</BODY></HTML>
```

