PHP et MySQL

Introduction

Généralités

Le langage HTML (HyperText Markup Language), ainsi que décrit dans le cours HTML, permet de créer des documents indépendants de toute plate-forme, et donc particulièrement bien adaptés à des échanges d'informations dans un environnement hétérogène comme le web. C'est une méthode éprouvée de création de pages web. Pourquoi donc faudrait il programmer en PHP ? Pourquoi faudrait-il des pages web « dynamiques » ?

Limites du HTML

La plupart des sites web sont réalisé à l'aide du HTML et offrent un contenu statique comme les publications, les articles etc … Les pages de ces sites consistent en du texte simple agrémenté de quelques images et de liens hypertextes menant vers d'autres pages, javaScript permettant d'y ajouter une touche plus sophistiquée si nécessaire.
Or il se trouve qu'internet et les intranets sont de plus en plus utilisés pour des applications, dont la plupart mettent en jeu des bases de données. Ces sites et leurs applications sont dynamiques, car le contenu est modifié selon les données impliquées et les actions de l'utilisateur. C'est bien là que PHP entre en scène : en exécutant un programme PHP sur le serveur, vous pouvez créer de puissantes applications agissant de façon interactive avec une base de données, et générant un contenu dynamique.

Ce que PHP peut faire et que HTML ne peut pas faire

PHP :
-facilite la modification du contenu d'une page web, intervenant sur les données d'une base et non directement dans le code HTML.
-crée des pages personnalisées afin de n'afficher que ce qui intéresse un utilisateur particulier.
-affiche et actualise une base de données incluse dans la page web, peut y manipuler les données, par exemple en effectuant un tri ou en n'affichant qu'un sous-ensemble de la base.
-crée des pages qui effectuent un cycle parmi différentes images.
-obtient une réponse de l'utilisateur, puis renvoie l'information en fonction de cette réponse.
-peut faire bien d'autres choses encore…

Ce cours n'est pas exaustif. Il permet de donner des notions de base à tout programmeur qui souhaiterait faire ses premiers pas en PHP et en MySql. Cela étant, il est clair qu'une compréhension, au moins intuitive, des notions informatiques de base (qu'est ce qu'un réseau, un fichier, un éditeur de

texte, un langage de programmation, une compilation etc.) est préférable. Nous supposons en outre que vous disposez au moins de l'un des environnements suivants :

· un ordinateur, connecté à l'internet par l'intermédiaire d'un fournisseur d'accès proposant l'environnement MySql/PHP.
· un ordinateur sous Linux ou Windows, disposant en local d'un environnement complet Apache/MySql/PHP.

A. PHP

I. Les bases du PHP

a. Définition

Le langage PHP (Personal Home Page Tools - Hypertext Preprocessor) a été crée par Rasmus Lerdorf vers la fin de l'année 1994, pour ses besoins personnels. La mise à disposition du langage sur internet fera passer son développement d'un individu à un noyau de programmeurs composé de Andi Gutmans, Zeev Suraski, Stig Bakken, Shane Caraveo Jim Winstead, et bien sûr de Rasmus Lerdorf.
C'est un langage de programmation très proche du C, dont il reprend une grande partie de la syntaxe, et destiné à être intégré dans des pages HTML.
Après diverses évolutions, PHP est à sa version 5.

b. le fichier PHP et sa structure

Le fichier PHP est enregistré au format ASCII, si bien que vous pouvez écrire une page PHP à l'aide de pratiquement n'importe quel éditeur de texte : notepad, wordpad, vi, emacs …
Le code PHP est un script inscrit dans une page HTML.

II. La programmation en PHP

1. syntaxe

a. généralités

Elle est très simple.
Tout code PHP doit être inclus dans une balise ***< ?php … ?>***. Des balises « courtes » ***< ?>*** sont parfois acceptées, mais ne sont pas recommandées, car elles risquent d'entrer en conflit avec d'autres langages tel que XML.
Comme en C, le séparateur d'instruction est le point-virgule « ***;*** »

b. Les commentaires

Si le commentaire tient sur une seule ligne, il peut commencer par les signes « ***//*** » ou « ***#*** ». S'il s'agit d'un commentaire s'étalant sur plusieurs lignes, il faut le placer entre les signes « ***/**** » et « ****/*** ».
Bien entendu, on peut mixer les deux styles de commentaires dans un même script.

c. Les variables et les données

• *Les variables*

Un nom de variable commence toujours par un « ***$*** », suivi d'au moins un caractère non numérique (le « _ » est autorisé), puis de n'importe quelle combinaison de chiffres et de caractères. En PHP, il n'est pas nécessaire de déclarer les variables ni de définir le type de données qu'elles doivent contenir avant de pouvoir les utiliser; PHP crée automatiquement une variable dès qu'un symbole nouveau préfixé par « ***$*** » apparaît dans le script. Le type d'une variable peut changer si l'on modifie son contenu.
PHP distingue les majuscules et minuscules dans le nom des variables ; ainsi « $mavariable » et « maVariable » désignent deux variables différentes ; par contre les noms de fonction sont insensibles à la casse.

• *Les données*

Les types de base sont :
Entier : utilise 4 octets de mémoire et sert à représenter un chiffre ordinaire dépourvu de décimales.

Double : également connu sous le nom de nombre réel ou à virgule flottante, sert à représenter une valeur possédant des décimales et un exposant.

Chaîne : une chaîne représente des valeurs non numériques, comme des lettres, des signes de ponctuation mais également des caractères numériques.

Exercice 1 : mon premier programme en PHP

```
<HTML>
< ?php
        echo « texte généré par PHP » ;
?>
</HTML>
```

Exercice 2

```
<HTML>
<FORM>
        Veuillez saisir votre nom ici : <BR>
        <INPUT TYPE = TEXT NAME = utilisateur> <BR><BR>
        <INPUT TYPE = SUBMIT VALUE = « soumettre »>
</FORM>

<BR><BR>
Vous avez saisi :

< ?php
        echo ($utilisateur) ;
?>
</HTML>
```

2. les opérateurs

a. les opérateurs arithmétiques

- « + » : addition
- « - » : soustraction
- « * » : multiplication
- « / » : division
- « % » : modulo

b. les opérateurs de comparaison

- « == » : égal à
- « < » : est inférieur à
- « > » : est supérieur à
- « <= » : est inférieur ou égal à
- « >= » : est supérieur ou égal à
- « != » : différent de
- « <> » : différent de

c. les opérateurs logiques

- « && » : et
- « || » : ou
- « and » : et
- « or » : ou
- « xor » : ou exclusif
- « ! » : non

3. les structures de contrôle

a. les instructions conditionnelles

- l'instruction if

```
if (condition)
        {instruction1;}
[else
        {instruction2;}]
```

• l'instruction switch

```
switch (variable) {
        case valeur1 ;
                instruction1 ;
                break ;
        case valeur2 ;
                instruction2 ;
                break ;


        case valeur_n ;
                instruction_n ;
                break ;
        default :
                instruction ;
}
```

b. les boucles

• la boucle for

```
for (expr1 ; expr2 ; expr3) :
        // instructions
endfor ;
```

• la boucle while

```
while (condition) {
        // instructions
}
```

• la boucle do … while

```
do {
        instructions;
} while (condition);
```

4. incorporer un fichier dans une page PHP

insérez le code suivant : require (‘’ nomDuFichier_A_insérer.extensionDuFichier ‘’) ;

5. les fonctions

Les fonctions rendent possible l'écriture de code modulaire et réutilisable. Une fonction peut recevoir des arguments (des variables transmises à celle-ci afin d'être utilisées à l'intérieur de la fonction) et renvoyer une valeur. Une variable à l'intérieur d'une fonction possède en principe une portée locale, ce qui signifie qu'elle n'existe qu'à l'intérieur de la fonction et n'interfère pas avec toute variable située à l'extérieur de la fonction, quand bien même elle porterait le même nom. Une fonction peut accéder à une variable globale à l'aide de l'instruction *global*. Les variables locales situées à l'intérieur d'une fonction sont réinitialisées à chaque invocation de la fonction, à moins que l'instruction *static* ne soit utilisée : dans ce cas, elle conservera la valeur qu'elle possédait à l'issue de l'invocation précédente.

Les fonctions sont déclarées à l'aide de l'instruction *function*.

a. syntaxe de la déclaration

```
function nom_fonction (paramètres) {
        corps de la fonction
}
```

Exemple

```
// déclaration et définition d'une fonction

function calculCube ($nombre) {
        return $nombre * $nombre * $nombre ; // renvoie $nombre à la puissance 3
}


// appel de la fonction calculCube

echo (cube (9)) ;          // doit afficher à l'écran 729
```

B. MySql

1. utilité des bases de données

PHP à lui seul, c'est-à-dire pris de façon isolée, ne constitue pas une solution miracle au manque de dynamisme des pages HTML. Les variables de ce langage ne peuvent retenir les données qui y sont stockées, que pendant la durée d'exécution d'un programme.
Comment alors introduire plus de souplesse dans l'affichage du contenu, selon les besoins et les goûts des utilisateurs ? Comment n'afficher que les informations auxquelles l'utilisateur s'intéresse, ou a choisi de recevoir, comme des nouvelles ou des communiqués ? Le stockage de données structurées devient nécessaire lorsque ces besoins doivent être satisfaits.

Pour bénéficier des avantages d'un site dynamique et performant, les données doivent être stockées, et groupées si nécessaire dans des tables, qui constitueront une base de données.

Il existe plusieurs systèmes stables et éprouvés comportant un ensemble de techniques performantes permettant aux développeurs de créer des applications de bases de données répondant à leurs besoins : les Systèmes de Gestion de Bases de Données (SGBD), au nombre desquels Access, Oracle, SQL Server, Sybase, InterBase, MySQL.

Nous nous intéresserons dans ce paragraphe au SGBD MySQL, car la solution que forme le couple (MySQL, PHP) est d'une part d'une popularité croissante pour la gestion de sites web dynamiques ; ensuite parce que ces deux outils constituent une solution complète aux problèmes évoqués ci-dessus, tout en restant d'une grande simplicité d'utilisation, et enfin parce que le tandem formé par MySQL et PHP a l'avantage de s'associer facilement au serveur Apache, et ce, aussi bien sous Linux que sous Windows.

2. Les instructions de définition de données

Egalement appelées requêtes, ou commandes, ces instructions du langage MySQL permettent de modifier le schéma de la base de données en créant ou en modifiant des objets dans cette dernière.

- ***CREATE***

Cette instruction permet de créer une base de données ou une table dans une base de données existante. La syntaxe de l'opération de création d'une base de données est simple, tandis que celle de création d'une table est plus complexe car il faut y inclure la description des champs.

· Création d'une base de données :

CREATE DATABASE nom_base_de_donnees

· Creation d'une table :

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nom_table (definitionCreation, …) [optionsTable] [optionSelect]

definitionCreation:

nomAttribut type [NOT NULL | NULL] [DEFAULT literal] [AUTO_INCREMENT]
[PRIMARY KEY] [definitionReference]

ou PRIMARY KEY (nomAttribut, …)
ou KEY [nomIndex] (nomAttribut, …)
ou INDEX [nomIndex] (nomAttribut, …)
ou UNIQUE [INDEX] [nomIndex] (nomAttribut, …)
ou [CONSTRAINT contrainte] FOREIGN KEY nomIndex (nomAttribut, …) [reference]
ou CHECK (expression)

optionsTable :

TYPE = {ISAM | MYISAM | HEAP}
ou AUTO_INCREMENT = #
ou AVG_ROW_LENGTH = #
ou CHECKSUM = {0 | 1}
ou COMMENT = « commentaires »
ou MAX_ROWS = #
ou MIN_ROWS = #
ou PACK_KEYS = {0 | 1}
ou PASSWORD = « motDePasse »
ou DELAY_KEY_WRITE = {0 | 1}

optionSelect:

[IGNORE | REPLACE] SELECT … (requête SQL)

reference:

REFERENCES nomTable [(nomAttribut, …)]

[MATCH FULL | MATCH PARTIAL]
[ON DELETE optionRef]
[ON UPDATE optionRef]

optionRef

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

La commande *CREATE* table crée une table dans la base de données courante. L'option TEMPORARY indique que la table est créée pour la connection courante uniquement. L'option *optionTable* permet de choisir le type de table.
Le nom d'un index est optionnel.

Exemple d'utilisation de la commande CREATE dans la création d'une base de données et d'une table :
CREATE DATABASE universite

Créons à l'intérieur de la base de données *universite*, la table *etudiant* :

```
CREATE TABLE etudiant (matricule VARCHAR (40) NOT NULL,
                       nom VARCHAR (30) NOT NULL,
                       prenom VARCHAR (30) NOT NULL,
                       sexe CHAR (1),
                       anneeNaiss INTEGER,
                       faculte VARCHAR (30),
                       option VARCHAR (30),
                       remarque TEXT,
                       PRIMARY KEY (matricule));
```

- ***DELETE***

syntaxe

```
DELETE [LOW_PRIORITY] FROM nomTable
    [WHERE clauseWhere] [LIMIT nbLignes]
```

Cette commande détruit toutes les lignes vérifiant les critères de la clause *WHERE*. L'option *LOW_PRIORITY* indique à MySQL que les destructions sont moins prioritaires que toutes les requêtes courantes qui accèdent à la table.

Exemple d'utilisation de la commande *DELETE* :

```
DELETE FROM TABLE etudiant ;
```

Cette requête efface tous les enregistrements de la table *etudiant*.

```
DELETE FROM TABLE etudiant WHERE faculte = 'sciences de la santé' AND
    option = 'médécine' ;
```

Cette requête efface de la table *etudiant* tous les étudiants inscrits dans l'option « médecine » de la faculté « science de la santé ».

➢ ***SELECT***

SELECT [STRAIGHT_JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
[HIGH_PRIORITY] [DISTINCT | DISTINCTROW | ALL]
listeAttributs
[INTO {OUTFILE | DUMPFILE} 'nomFichier' optionExport]
[FROM clauseFROM
[WHERE clauseWHERE]
[GROUP BY nomAttribut, …]
[HAVING clauseWHERE]
[ORDER BY {entire | nomAttribut | formule} [ASC | DESC], …]
[LIMIT [debut,] nbLignes]
[PROCEDURES]]

clauseFROM:
nomTable, nomTable
ou nomTable [CROSS] JOIN nomTable
ou nomTable INNER JOIN nomTable
ou nomTable STRAIGHT_ JOIN nomTable
ou nomTable LEFT [OUTER] JOIN nomTable ON expression
ou nomTable LEFT [OUTER] JOIN nomTable USING (listeAttributs)
ou nomTable NATURAL LEFT [OUTER] JOIN nomTable
ou nomTable LEFT OUTER JOIN nomTable ON expression

Cette commande extrait d'une ou plusieurs tables les lignes qui satisfont la clause *WHERE*.
ListeAttributs est une liste d'attributs provenant des tables du *FROM*, ou d'expressions impliquant des fonctions. On peut faire référence à un attribut par son nom, par le nom de sa table et son nom, ou par le nom de sa base, le nom de sa table et son nom : *universite.etudiant.nom* désigne l'attribut nom de la table *etudiant* de la base *universite*.
L'option :
-*STRAIGHT_JOIN* indique que la jointure doit accéder aux tables dans l'ordre indiqué.
-*SQL_SMALL_RESULT* prévient MySQL que le résultat contiendra peu de lignes, ce qui permet d'optimiser l'exécution de la requête.
-*SQL_BIG_RESULT* indique l'inverse.
-*HIGH_PRIORITY* demande l'exécution de la requête en priorité par rapport à celles qui effectuent des modifications.
Attention : ces options sont réservées aux utilisateurs avertis et doivent être utilisées en connaissance de cause.

➢ ***INSERT***

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] nomTable [(nomAttribut, …)] VALUES (expression, …), (…), …
ou INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] nomTable [(nomAttribut, …)] SELECT…
ou INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] nomTable SET nomAttribut = expression, …

Exemple d'utilisation de la commande *INSERT*

INSERT INTO etudiant (matricule, nom, prenom, sexe, anneNaiss, faculte, option, remarque) VALUES ('20194', 'GUINKO', 'Tonguim Ferdinand', 'M', '1979', 'sciences exactes et appliquées', 'mathématiques linéaires', 'rien à signaler') ;

La commande *INSERT* insère une ou plusieurs lignes dans une table.

➢ ***USE***

syntaxe :

USE nomBase

La commande *USE* permet d'accéder à une base.

Exemple d'utilisation de la commande USE

USE universite

3. Les fonctions

Les fonctions suivantes peuvent être utilisées dans les requêtes :

- ABS (nombre) : renvoie la valeur absolue de nombre.
- ASCII (char) : renvoie le code ASCII du caractère char.
- CONCAT (chaîne1, [chaîne2, …]) : renvoie la concaténation de tous les arguments
- CONV (nombre, base1, base2) : renvoie la conversion de nombre, de base1 en base2 ; la base est un chiffre entre 2 et 36.
- BIN (décimal) : renvoie la valeur binaire d'un nombre décimal.
- CURDATE () : renvoie la date courante au format AAAAMMJJ ou AAAA-MM-JJ selon que le contexte est numérique ou alphanumérique.
- CURTIME () : renvoie l'heure courante au format HHMMSS ou HH :MM :SS selon que le contexte est numérique ou alphanumérique.
- DATABASE () renvoie le nom de la base de données courante.
- DAYNAME (date) : renvoie le nom du jour en anglais.
- DAYOFMONTH (date): renvoie le numéro du mois.
- DAYOFWEEK (date): renvoie le numéro du jour dans la semaine.
- DAYOFYEAR (date): renvoie le numéro du jour dans l'année.
- LTRIM (chaîne) : retire tous les caractères blancs au début de chaîne.
- MONTHNAME (date) : renvoie le nom du mois de date en anglais.
- NOW () : renvoie la date et l'heure courante.
- PASSWORD (chaîne) : cryptage de chaîne avec la fonction utilisée pour les mots de passe MySQL.
- UCASE (chaîne) : renvoie chaîne en majuscule.

C. Accès à MySQL avec PHP

PHP communique avec MySQL par l'intermédiaire d'un ensemble de fonctions qui permettent de récupérer, modifier, ou créer à peu près toutes les informations relatives à une base de données. Parmi ces informations, il faut compter bien entendu le contenu des tables, mais également leur description, c'est à dire le schéma de la base.

Ces fonctions sont :

Mysql_connect : sert à établir une connexion avec MySQL, pour un compte utilisateur, et un serveur de données ; renvoie une valeur qui peut être utilisée ensuite pour dialoguer avec le serveur.

Mysql_pconnect: Idem, mais avec une connexion *persistante*. Cette deuxième version est plus performante quand l'interpréteur PHP est inclus dans Apache.

Mysql_select_db: permet de se placer dans une base de données. C'est l'équivalent de la commande *USE base* sous mysql.

Mysql_query: sert à exécuter une requête SQL ; renvoie une variable représentant le résultat de la requête.

Mysql_fetch_object: permet de récupérer une des lignes du résultat, et positionne le curseur sur la ligne suivante. La ligne est représentée sous forme d'un objet (un groupe de valeurs).

Mysql_fetch_row: permet de récupérer une des lignes du résultat, et positionne le curseur sur la ligne suivante ; la ligne est représentée sous forme d'un tableau (une liste de valeurs).

Mysql_error : renvoie le message de la dernière erreur rencontrée.

Exemple : accès à MySQL avec PHP

Pour accéder à une base de données MySQL, un script PHP doit :

- Se connecter au serveur de bases de données MySQL.
- Envoyer la requête SQL au serveur de bases de données MySQL, puis récupérer le résultat
- Extraire les données du résultat à l'aide des fonctions
- Générer la page HTML affichant les données.

```
<HTML>
    <HEAD>
        <TITLE> connection à MySQL </TITLE>
    </HEAD>
    <BODY>
        <H1>interrogation de la table etudiant</H1>
        < ?php

            // connexion au serveur de bases de données MySQL
```

```
require (‘‘connect.php’’) ;

$connexion = mysql_pconnect (serveur, nom, motDePasse) ;

if ( !$connexion)
{
        echo ‘‘Désolé, connexion à  ’’ . serveur . ‘‘ impossible\n’’;
        exit ;
}

if ( !mysql_select_db (nomBase, $connexion))
{
        echo “Désolé, accès à la base ” . nomBase . ‘‘impossible\n’’;
        exit ;
}

/* Envoyer la requête SQL au serveur de bases de données MySQL, puis récupérer le
        résultat*/


        $resultat = myquery ( ‘’SELECT * FROM etudiant ‘’, $connexion) ;

        /* Extraire les données du résultat à l’aide des fonctions et
Générer la page HTML affichant les données*/

        if ($resultat)
        {
                while ($Etudiant = mysql_fetch_object ($resultat))
                {
                        echo “$Etudiant ->matricule, de l’étudiant’’
                        . ‘‘  $Etudiant->nom $Etudiant->prenom’’ <BR>\n’’ ;
                }
        }

        else

        {
                echo ‘‘<B>Erreur dans l’exécution de la requête.</B><BR>’’ ;
                echo ‘‘<B>Message de MySQL : </B>’’ . mysql_error
```

```
                                        ($connexion) ;
                                }
        ?>
        </BODY>
</HTML>
```

La commande *require* permet d'inclure le contenu d'un fichier dans le script. Certaines informations sont communes à beaucoup de scripts, et les répéter systématiquement est à la fois une perte de temps et une grosse source d'ennui le jour où il faut effectuer une modification dans *n* versions dupliquées. Ici on a placé dans le fichier *connect.php* quelques informations de base sur le site : le nom du serveur, le nom de la base de données et le compte d'accès à la base de données.

```
< ?php

        // connect.php

        $serveur =  ''www.le_serveur_hote.com'';          // hôte d'exécution de la base de
                                                          // données MySQL

        $nom = ''php''                                    // nom de l'utilisateur de la
                                                                  //base de données
        $motDePasse = ''php''                       // mot de passe de l'utilisateur
        $nomBase = ''universite''                         // nom de la base de données

?>
```

Conclusion

Ce travail est le fruit d'une part de longues recherches sur le net et dans plusieurs ouvrages, et d'autre part de notre expérience en la matière.
Il n'est certes pas exhaustif, mais permet au programmeur qui souhaite faire ses premiers pas en PHP et/ou en MySQL de comprendre les éléments de base de ces langages, en des termes simplifiés.

PHP ne représente en fait qu'une technologie parmi celles pouvant être utilisées pour créer des pages web plus dynamiques et plus interactives. Parmi les autres technologies permettant de réaliser des sites web dynamiques nous pouvons citer :

-*Active Server Pages (ASP)* : il est similaire à PHP ; l'inconvénient fondamental de cette technique est qu'elle ne peut être utilisée qu'avec un serveur web Microsoft (IIS, PWS), ou avec un système d'exploitation Microsoft.
-*JavaScript côté serveur (SSJS)* : il se combine avec HTML comme les deux précédents ; l'inconvénient de cette technique est qu'elle nécessite la compilation des applications SSJS avant de pouvoir les exécuter ; de plus, SSJS n'est pris en charge que par Netscape Entreprise Server.
Quand aux SGBD, ils sont nombreux : Access, Oracle, SQL Server, Sybase, InterBase, MySQL …

La solution MySQL/PHP présente l'avantage d'être performante, et portable sur les plate-forme linux et windows.

Pour aller plus loin

Bibliographie

J. Castagnetto, H. Rawat, S. Schumann, C. Scollo, D. Veliath, "*PHP professionel*", Eyrolles, 2001

Philippe Riguaux, « *MySQL et PHP* », O'Reilly, édition originale, 2001

Webographie

http://www.wrox.fr

http://www.php.net

http://www.mysql.com