Langage PHP

Initiation à la programmation

Partie 2

Table des matières

I. Fonctions :

A. Un peu de théorie :

1. Introduction :

Lors de la création d'une page de programmation, il est possible que certaines instructions se répètent plusieurs fois dans le code ou qu'un jeux d'instructions soit présent à l'identique dans plusieurs fichiers de code.

Pour éviter cette répétition, il est possible d'inclure ces instructions dans une fonction et de répéter leurs exécutions en multipliant les appels à cette fonction.

Les fonctions sont définies par :

- un **identifiant** : le nom de la fonction permet de l'identifier et d'effectuer des appels à celle-ci.
- le **corps de la fonction** : il est compris entre les accolades { et } et renferme toutes les instructions à exécuter à chaque appel de la fonction.
- les **paramètres formels** *(ou arguments)* représentent les variables contenant les valeurs transmises à la fonction lors de l'appel.

Les valeurs passées via les paramètres formels permettent de contrôler et de modifier l'exécution d'une fonction pour chacun des appels. Les différentes valeurs doivent être séparées entre elles par des virgules ",".

Voici la structure de la déclaration d'une fonction :

```
function Identifiant(Arguments){
  Corps de la fonction
}
```

Voici un exemple de déclaration et d'appel à une fonction :

```
Voici la déclaration d'une fonction
function aff($x,$y){
  echo $x.$y.$y;
}
Voici un appel à cette fonction
aff("bonjour ","thomas");
```

La fonction est déclarée avec deux paramètres formels `$` et `$y`. Son appel est effectuée avec les deux arguments `"bonjour "` et `"thomas"`.
Il affichera la chaine de caractères : `"bonjour thomasbonjour "`.

2. *Paramètres formels et arguments :*

La déclaration d'une fonction définit le nombre de paramètres formels utilisés. L'appel à une fonction doit comporter un nombre d'arguments correspondant au nombre de ses paramètres formels.
Voici la réaction du `PHP` si ces deux nombres sont différents :

- Un appel avec un nombre inférieur d'arguments provoquera une erreur d'exécution du code `PHP`.
- un appel effectué avec un nombre supérieur d'arguments n'entrainera aucune erreur : les arguments supplémentaires seront ignorés. Il est néanmoins possible d'accéder à ces valeurs supplémentaires via la commande `func_get_args()` retournant une variable-tableau contenant tous les arguments de l'appel.

Voici un exemple de déclaration et d'appels à une fonction.

```
function maFonction($x,$y){
   return $x+$y;
}
echo maFonction(5,6);
echo maFonction(5,6,7);
$x=func_get_args();
echo $x[2];
```

```
8 echo maFonction(5);
```

- L'appel de la ligne 4 n'engendre aucune erreur. L'appel à la fonction retourne la valeur 11 qui sera affichée par la commande `echo`.
- L'appel de la ligne 5 s'éxécute de la même façon que l'appel précédant. Les lignes 6 et 7 montrent comment récupérer les autres arguments.
- L'appel de la ligne 8 provoque une erreur : le nombre d'arguments est inférieur au nombre de paramètres formels de la fonction.

Il est également possible de rendre un argument optionnel en assignant une valeur par défaut à un paramètre formel lors de la déclaration de la fonction :

```
1 function opt($x,$y,$z=5){
2   echo (x+y*z);
3 }
4 echo opt(2,3,1)."<br>";
5 echo opt(3,1)."<br>";
6 echo opt(4);
```

Lors du premier appel à la fonction `opt(...)`, les paramètres formels sont affectés ainsi :

`$x=2` ; `$y=3` ; `$z=1`

et cet appel affichera le nombre `5`.

Le second appel à la fonction `opt()` ne comprend que deux arguments ; le troisième paramètre prendra sa valeur par défaut. Voici l'affectation donnée à chacun des paramètres formels :

`$x=3` ; `$y=1` ; `$z=5`

et cet appel affichera le nombre `8`.

Le troisième appel provoque une erreur car le nombre d'arguments passés est insuffisant : le paramètre `$y` n'est affecté d'aucune valeur.

3. *Portée des variables :*

En `PHP`, les variables définies dans le corps d'une fonction n'interfèrent pas avec celle déclarée dans le corps du script comme le montre le script suivant :

```
1 function var(){
2   $a="Bonjour";
3 }
4 $a="Bonsoir";
5 var();
6 echo $a;
```

L'exécution de ce script affichera "`Bonsoir`" ; malgré l'appel de la fonction `var()`, la valeur de `$a` ne sera pas modifiée lors de l'exécution de la ligne 5. Dans ce script, il existe :

- la variable **locale** `$a` définie dans le corps de la fonction ;
- la variable **globale** `$a` initialisée avec la valeur `"Bonsoir"`.

Pour qu'une variable définie dans le corps d'une fonction puisse représenter une variable hors du corps de la fonction *(une variable globale)*, on utilise le mot-clef `global` dans la première ligne de déclaration de la fonction :

```
1 function var(){
2   global $a;
3   $a="Bonjour";
4 }
5 $a="Bonsoir";
6 var();
7 echo $a;
```

Ainsi, la fonction `var(...)` utilisera dans son corps la variable globale `$a` et son appel modifiera donc sa valeur.
L'exécution du script ci-dessus affiche "`Bonjour`".

4. *Valeur retournée et mot-clef return :*

Le code ci-dessous déclare une fonction qui affiche le résultat d'un calcul :

```
1 function calcule($x){
2   $x=$x*($x+1);
3   echo $x;
4 }
5 calcule(5);
```

Cette fonction affiche directement la valeur calculée, l'inconvénient est que la valeur calculée ne peut être récupérée dans une variable.
Pour que la valeur calculée soit réutilisable par le reste du script, il faut demander à la fonction de retourner la variable plutôt que de l'afficher. Ceci est possible à l'aide de la commande `return` :

```
1 function calcule($x){
2   $x=$x*($x+1);
3   return $x;
4 }
5 calcule(6);
6 echo calcule(7)."<br>";
7 $y=calcule(8)+1;
8 echo $y;
```

Trois appels à la fonction `calcule(...)` s'effectuent :

- A la ligne 5, la fonction est appelée mais la valeur retournée ne sera pas réutilisée dans

le programme ;

- A la ligne 6, la valeur retournée par l'appel à la fonction est affichée directement par la commande `echo` ;
- A la ligne 7 et 8, la valeur retournée par l'appel à la fonction est stockée dans la variable `$y` puis cette valeur sera affichée par la commande `echo`.

B. Quelques exercices dirigés :

Exercice 1

1. A la racine de votre site, créez le fichier nommé `lien.php` et saisissez-y le code suivant :

```php
<?php
function lien($nom,$cible,$page){
  echo "<a href='".$cible."' style='' target='".$page."'>".
  $nom."</a><br>";
}
lien("math","math.txt","_self");
lien("voyage","http://google.fr/voyage.txt","_self");
lien("google","http://google.fr","_self");
lien("Actualites","http://google.fr","_blank","color:red");
?>
```

2. Affichez cette page dans votre navigateur en saisissant l'URL :

   `http://127.0.0.1/lien.php`.

3. Après avoir essayé les deux premiers liens, répondez aux questions suivantes :

   a. Pourquoi le premier lien affiche une erreur ? Que doit-on faire pour que la page ouverte affiche le message "*Bonjour*" ?

.....

.....

.....

   b. Le second lien affiche une erreur. Est-il possible sans modifier le lien de corriger cette erreur ? Pourquoi ?

.....

.....

Ces deux premiers liens provoquent des erreurs car les pages demandées n'existent pas :

- Le premier lien demande le fichier `math.txt` se trouvant au même emplacement que

la page courante : c'est à dire la racine de notre site. Pour que s'affiche le message "*Bonjour*", il faut créer un fichier intitulé `math.txt` à la racine du site contenant ce message.

- Le second lien demande le fichier `voyage.txt` à la racine du site de `google.fr`. Ce fichier n'existant pas, une erreur sera produit. Il n'est évidemment pas possible de rajouter un fichier sur le serveur d'hébergement de `google.fr` : on ne peut enlever le message d'erreur sauf en modifiant le lien.

4. Après avoir essayé les deux autres liens, répondez aux questions suivantes :

a. Expliquer pourquoi les trois premiers liens s'affichent dans la même page et que le troisième lien s'ouvre dans une nouvelle page ?

.....

.....

b. A la ligne 6, enlever le troisième argument lors de l'appel à la fonction `lien(...)`. Actualisez votre page et cliquez sur le premier lien. Quelle est la raison de l'erreur ?

.....

.....

Cet exercice montre l'une des utilités des fonctions dans les langages de programmation : centraliser une partie des instructions afin de les répéter à plusieurs endroits. Ceci présente plusieurs avantages :

- la centralisation du code permet un maintenance plus rapide en cas de modification du code ;
- la répétition d'un même jeu d'instructions permet d'assurer l'homogénétié de l'affichage.

La déclaration de cette fonction fait intervenir trois paramètres formels :

`$nom` ; `$cible` ; `$page`

Chaque appel à la fonction `lien(...)` doit comporter trois arguments.

Lorsqu'un appel se fait avec moins de trois arguments, l'exécution de la fonction provoque une erreur. Lorsque l'appel se fait avec plus de trois arguments, les arguments supplémentaires sont ignorés.

Le quatrième lien s'affiche dans une nouvelle page car l'élément a déclarant un lien possède alors l'attribut ***target*** avec la valeur *_blank* qui force l'ouverture du fichier dans une nouvelle page.

La valeur *_self* de l'attribut ***target*** indique que le lien s'ouvrira dans la page courante.

5. a. Modifier la ligne de déclaration de la fonction *(ligne 2)* en :

```
function lien($nom,$cible,$page="_self")
```

b. Actualisez la page et cliquez sur le premier lien. L'erreur est-elle toujours présente ? Pourquoi ?

.....

.....

c. Changez la première ligne de déclaration de la fonction en :

```
function lien($nom,$cible,$page="_self",$newStyle="color:pink")
```

et modifiez l'attribut ***style*** de l'élément a de la ligne 3 en :

```
style='".$newStyle."'
```

d. Actualisez la page et justifiez que seul le dernier lien soit de couleur rouge.

Les modifications précédentes du code ont montré la possibilités d'affecter une valeur par défaut des paramètres formels lors de l'appel à une fonction.
Si l'appel à une fonction omet de déclarer l'argument d'un tel paramètre formel, l'exécution de la fonction s'effectue normalement avec cette valeur par défaut pour paramètre.

Exercice 2

1. A la racine de votre site, créez un nouveau fichier nommé `calcul.php` et saisissez-y le code suivant :

```php
<?php
function calcul($x){
  $a=5;
  echo "Le résultat est ".(3*$x+$a)."<br>";
}
$a=2;
calcul(5);
calcul(3);
$a=10;
calcul(5);
calcul(3);
echo $a;
?>
```

2. Affichez cette page en saisissant dans votre navigateur l'URL :

`http://127.0.0.1/calcul.php`

a. Lors de l'exécution du code, avec quelle valeur la variable `$a` est-elle initialisée ?

.....

b. Quelle valeur de `$a` utilise la fonction `calcul()` pour chacune des ses exécutions ?

.....

c. La variable `$a` définie dans le corps de la fonction affecte-t-elle la variable `$a` du corps principal du programme ? Justifiez votre réponse.

.....

.....

Dans cette exercice, nous avons pu voir la manière dont une fonction gère les variables ; il faut distinguer les variables définies à l'intérieur du corps d'une fonction et celles définies dans le corps même du script : on parle respectivement de variables locales *(à la fonction)* et de variables globales.

Une variable locale *(définie dans le corps d'une fonction)* n'agit pas nativement sur les variables globales *(définie dans le corps du script)* : on agit sur le comportement d'une fonction au travers des arguments qui lui sont fournis lors de son appel.

3. a. Effacez la première ligne du corps de la fonction `calcul(...)` *(ligne 3)*. En actualisant la page, quelles erreurs apparaissent ? A quel moment de l'exécution du programme ?

.....

.....

b. Insérez l'instruction suivante comme première ligne du corps de la fonction `calcul(...)` :

```
global $a;
```

c. Actualisez la page pour ré-exécuter uce script. Quelle est maintenant la valeur de la variable `$a` lorsque le script se termine ?

.....

Le mot-clé `global` permet à une fonction d'utiliser une variable définie à l'extérieur du corps de celle-ci : on peut donc utiliser des variables **globale** à l'intérieur d'une fonction.

4. a. Ajoutez maintenant en dernière ligne du corps de la fonction, le code :

```
$a++;
```

b. Justifier que cette fois, la variable `$a` a la valeur `12` lorsque le script se termine :

.....

.....

.....

La ligne 9 affecte la valeur 10 à la variable `$a`. Les deux appels à la fonction `calcul(...)` ont pour effet d'exécuter l'instruction `$a++` qui a pour effet d'incrémenter à chaque fois la valeur de la variable `$a`.
Ainsi, après l'exécution de ces deux appels, la variable `$a` aura pour valeur 12.

Exercice 3

1. A la racine de votre site, créez un nouveau fichier nommé `alternatif.php` et saisissez-y le code suivant :

```
1 <?php
2 function taille(){
3   echo date("s");
4   }
5 echo "<div style='font-size:".(taille()+10)."px'>Affichage</↩
    div>";
6 ?>
```

2. Affichez cette page dans votre navigateur en saisissant l'URL :

`http://127.0.0.1/alternatif.php`

3. a. En actualisant successivement et plusieurs fois cette page, quelle remarque peut-on faire sur son affichage ?

.....

.....

b. Afin d'avoir des informations complémentaires sur la commande `date(...)`, effectuez une recherche sur Internet avec les mots clés :

`manual` ; `php` ; `date`

Qu'affiche la commande `date("s")` ?

.....

La commande `date(...)` permet d'obtenir des informations sur la date courante du serveur *(la machine sur laquelle s'exécute le script)*. Avec l'argument `"s"`, cette commande renvoie le nombre de seconde de l'heure courante.

4. a. Ajoutez en dernière ligne du script PHP, ajoutez l'instruction suivante :

```
echo "taille : ".(taille()+10);
```

b. Ré-actualisez plusieurs fois la page. Pourquoi la taille affichée reste à 10 ?

.....

.....

c. Modifiez la ligne 4 du fichier `alternatif.php` en :

```
return date("s");
```

d. Ré-actualisez la page afin d'observer les modifications sur l'affichage. Justifier le changement de taille du message "*Affichage*" ?

.....

.....

Lors de la première version du script, la fonction `date(...)` renvoie le nombre de seconde de l'horloge directement pour l'affichage via la commande `echo`. Ainsi, l'instruction "`taille()+10`" est équivalente à "`0+10`" car la fonction `taille()` ne renvoit aucune valeur au script PHP, elle affiche directement ce nombre.

L'introduction du mot-clé `return` permet de définir la valeur retournée par la fonction `taille()`. Ainsi, l'addition `taille()+10` possède alors deux termes.

Exercice 4

1. A la racine de votre site, créez un nouveau fichier intitulé `recursif.php` et saisissez-y le code suivant :

```php
<?php
if(!isset($_GET["mot"])){
  echo "Aucune formule transmise";
  exit();
}

echo "Démarrage<br>";

$trait="";
$mot=$_GET["mot"];
```

```php
11
12 function recursif($x){
13   global $mot,$trait;
14   echo $trait.substr($mot,0,1)."<br>";
15   $mot=substr($mot,1,strlen($mot)-1);
16   $trait.="-";
17   if($mot!="")
18     recursif($mot);
19 }
20
21 recursif($mot);
22 ?>
```

2. Affichez cette page en saisissant dans votre navigateur l'URL suivante :

   `http://127.0.0.1/recursif.php`

3. a. Pourquoi le message "*Démarrage*" ne s'affiche pas à l'écran ?

.....

.....

b. Que doit-on faire pour que ce message apparaisse ?

.....

.....

Lors de la saisie de l'URL aucune variable n'a été transmise et notamment pas une variable nommé `nom`. Ainsi, lors de l'exécution du code `PHP`, la variable `$_GET["nom"]` n'existe pas, ainsi le test "`isset($_GET["mot"])`" retournera la valeur `false`.
Sa négation "`!isset($_GET["mot"])`" retournera la valeur `true`.
Ainsi, la conditionnelle évaluera, lors de cette première exécution, les instructions des lignes 3 et 4. La commande `exit` arrête net l'exécution du script : le reste du code ne sera évalué.

a. Accéder à cette page via l'URL suivante :

   `http://127.0.0.1/recursif.php?mot=thomas`

b. Combien de lignes sont affichées sous le message "*Démarrage*" ? En déduire le nombre d'appel à la fonction `recursif(...)` effectué lors de l'exécution de ce code ?

.....

c. Quelle est la valeur d'initialisation de la variable `$trait` ? Quelles sont les valeurs successives de la variable `$trait` ?

.....

. . . . . .

. . . . . .

d. A chaque appel de la fonction `recursif(...)`, la variable `$mot` est modifiée à l'aide de l'instruction suivante "`$mot=substr($mot,1);`". A chaque appel de la fonction `recursif(...)`, que peut-on dire de la taille de la variable `$mot` ?

. . . . . .

. . . . . .

e. Justifier que l'exécution du code s'achèvera toujours ?

. . . . . .

. . . . . .

On remarque que l'affichage comprend 6 lignes sous le message "*Démarrage*" : ce nombre correspond au nombre de caractères du mot "*thomas*".

La variable `$trait` est initialisée avec une chaine de caractères vides et elle est modifiée via l'instruction :

```
$trait.="-";
```

L'opérateur ".=" permet de rajouter une nouvelle chaine en fin de chaine. Cette instruction est équivalente à :

```
$trait=$trait."-";
```

Ainsi, la variable `$trait` ne contiendra que des traits d'union et leur nombre correspondra au nombre d'exécution de la fonction `recursif(...)`.

La variable `$mot` est initialisée avec la valeur contenue dans `$_GET["mot"]`. Cette variable est modifiée par l'instruction :

```
$mot=substr($mot,1);
```

Voici comment décoder l'utilisation de la commande `substr(...)` :

- `substr(...)` prendra la chaine de caractères contenue dans la variable `$mot` pour en extraire une partie ;
- Cette partie commencera au second caractère *(caractère d'index* 1*)* et ira jusqu'à la fin de la chaine de caractèrse *(abscence d'un troisième argument de la commande* `substr()`*)*.

A chaque appel de la fonction `recursif(...)`, la variable `$mot` est réduite d'un caractère. Il est sûr qu'après un certain nombre d'appel, la variable `$mot` sera une chaine vide : le test "`$mot!=""`" retournera `false` et l'appel récursif s'achèvera.

C. Quelques travaux pratiques :

Exercice 5

Ecrire une programme recevant une valeur ayant pour nom `chaine` via l'URL et qui retourne cette chaîne de caractères où un caractère sur deux sera caché par le caractère "-".

Deux possibilités s'offrent à vous :

- Soit une fonction, par un appel récursif, remplace un caractère sur deux.
- Soit en considérant la chaîne de caractères comme un tableau, on peut, à l'aide de la boucle ci-dessous, modifier un caractère sur deux :

  `for($i=0;$i<count($_GET["chaine"]);$i++){...}`

Exercice 6

Dans un nouveau fichier d'extension `.php`, saisissez le code suivant :

```php
<?php
$administrateur=array("thomas","clarisse");

function style($choix){
  global $administrateur;
  if(isset($_GET["log"])
     && (in_array($_GET["log"],$administrateur))){
    switch($choix){
      case "couleur":
     $x="color:red";
     break;
      case "fonte":
     $x="font-weight:bold";
     break;
    }
    return $x;
  }
  else{
    return "";
  }
}

function nom(){
  return isset($_GET["log"])?$_GET["log"]:"??";
}
?>

```

```
<div style='text-align:center;<?php echo style("fonte");?>'>
Bonjour <span style='<?php echo style("couleur");?>'>
<?php echo nom();?></span>
</div>
<form>
Identifiant : <input type=text name=log><br>
<input type=submit value='Se logguer'>
</form>
```

Qu'effectuent les fonctions `style(...)` et `nom(...)` lors de l'exécution du code ? Comment peut-on faire pour faire varier l'apparence de la page ?

Exercice 7

En informatique, on prend pour origine des dates le 1er janvier 1970 à minuit et l'unité principale de mesure est la seconde. Voici quelques exemples de date :

1. `3600` : 1 *h* du matin le 1er janvier 1970.
2. `31532400` : le 1er janvier 1971 à minuit
3. `955278000` : midi le 9 avril 2000.

A l'aide de plusieurs structures conditionnelles de type `switch()` et de la fonction `date()` dont vous trouverez la définition sur Internet, écrire une fonction qui prend pour argument la valeur de `time()` et renvoye la date sous la forme :

```
Lundi 31 Mai 2010
```

II. Gestion de fichiers :

A. Un peu de théorie :

1. *Introduction :*

Le PHP permet également d'accéder aux fichiers contenus sur le serveur : on peut lire leur contenu, le modifier, effacer les fichiers ou en créer de nouveau. Pour des raisons de sécurité, le serveur hébergeant votre site peut vous limiter l'accés aux fonctions que nous allons découvrir dans ce chapitre. Pour tester les possibilités offert par votre hébergeur, il faudra :

- Exécuter la commande `fwrite(...)` pour créer un nouveau fichier à l'aide d'un script PHP *(si cela a été possible)* ;
- Puis voir les droits que vous possédez sur le nouveau fichier via votre navigateur FTP.

Un message de la forme "`Warning: chmod() has been disabled for security reasons`"

indique que votre hébergeur vous a interdit certaines fonctionnalités.

La description des commandes présentées ci-dessous est assez sommaire ; pour de plus ample informations et pour des exemples d'utilisation, reportez-vous à la documentation en ligne du PHP.

2. *Accés direct à un fichier :*

Pour modifier directement l'intérieur du fichier, le PHP utilise une variable appelée pointeur. Ce pointeur est généralement crée par la commande `fopen(...)` et est généralemetn placé au début ou à la fin du fichier ; il est également possible de déplacer le pointeur à tout endroit du fichier.
La commande `fwrite` associé à un pointeur de type fichier permet d'écrire dans le fichier à la position du pointeur.

Voici quelques commandes permettant l'accés au fichier :

- `$fp=fopen($fs,$mode)` : cette commande crée un pointeur `$fp` sur le fichier désigné par la chaîne de caractère `$fs`.
  L'argument `$mode` définit le mode d'ouverture du fichier *(lecture ou lecture/écriture)*, ainsi que la position initiale du pointeur `$fp` créé.
  L'argument `$mode` est une chaîne de caractères dont voici quelques unes de ses valeurs possibles :
  - `"r"` : le fichier est ouvert en lecture seule ; le pointeur est en placé en début de fichier.
  - `"r+"` : le fichier est ouvert en lecture et en écriture ; le pointeur est en placé en début de fichier.
  - `"w+"` : ouvre le fichier en réduisant sa taille à `0` *(efface son contenu)* ; si le fichier n'existe pas, cette commande tente de le créer.
  - `"a"` : ouvre le fichier en lecture seule ; le pointeur est placé à la fin du fichier ; si le fichier n'existe pas, il est créé.
  - `"a+"` : ouvre le fichier en lecture et en écriture ; le pointeur est placé à la fin du fichier ; si le fichier n'existe pas, il est créé.
- `$s=fread($fp,$taille)` : cette commande lit `$taille` octets dans le fichier désigné par le pointeur `$fp` à partir de la position de ce pointeur.
- `fwrite($fp,$s)` : cette commande écrit la chaîne de caractères `$s` dans le fichier désigné par le pointeur `$fp` à partir de la position courante de celui-ci.
- `fclose($fp)` : cette commande ferme le pointeur `$fp` ; cela permet de libérer un fichier pour être, par exemple, exploité par un autre script.

3. *Lire le contenu d'un fichier :*

Les commandes suivantes peuvent simplifier l'accés au contenu d'un fichier *(en fonction de l'usage qui sera fait de ce contenu)*

- `$s=file($fs)` : cette commande retourne une variable-tableau dont les éléments sont les différentes lignes du fichier désigné par la chaîne de caractère `$fs`.
- `$s=file_get_contents($fs)` : cette commande retourne l'intégralité du contenu du fichier désigné par la chaîne de caractères `$fs`.

4. *L'arborescence des fichiers :*

De manière générale en informatique, le point "`.`" représente le dossier courant et le double point "`..`" représente le dossier parent.

Lors de l'exécution d'un script, le dossier contenant ce fichier s'appelle le dossier de travail ou dossier courant.

Les commandes suivantes permettent de se déplacer dans l'arborescence des fichiers : changer de répertoire, lister les fichiers contenus dans répertoire courant...

- `$dp=opendir($path);` : la variable `$path` est une chaîne de caractère représentant le chemin *(relatif ou absolue* d'un répertoire. Il renvoie un pointeur `$dp` sur la racine du repertoire.
  Le pointeur permettra de se déplacer dans le contenu de celui-ci.
- `$fs=readdir($dp);` : lors de son exécution, cette commande renvoie le nom du fichier pointé par la variable `$dp` *(ici stocké dans la variable* `$fs`*)* et le pointeur `$dp` se placera automatiquement sur le fichier suivant.
  Si le point `$dp` n'a plus de fichier à désigner, la commande `readdir(...)` retournera la valeur `false`.
- `closedir($dp)` : cette commande ferme le pointeur `$dp` d'un dossier et permet de libérer l'utilsation du repertoire pour d'autres applications.
- `chgdir($ds)` : cette commande permet de changer de répertoire de travail *(le répertoire courant)*. Cette commande rentourne `true` en cas de succés et `false` en cas d'échec.
- `is_file($fs)` : cette commande vérifie l'existence du fichier désigné par la chaîne de caractère `$fs`. Elle retourne `true` si ce fichier existe et elle retourne `false` dans le cas contraire.
- `is_dir($ds)` : cette commande retourne `true` si la chaîne de caractères `$ds` représente le chemin d'un répertoire et `false` dans le cas contraire.

- `unlink($fs)` : cette commande efface le fichier indiqué par le chemin `$fs` ; elle retourne `true` en cas de succés et `false` en cas d'échec.
- `rmdir($ds)` : cette commande efface le répertoire indiqué par le chemin contenu dans `$ds` *(celui-ci doit être vide)* ; cette commande renvoie un booléen pour indiquer la réussite ou l'échec de l'effacement.
- `filesize($fpath)` : cette commande retourne la taille du fichier dont le chemin est représenté par la chaîne de caractères `$fpath`.

B. Quelques exercices dirigés :

Exercice 8

1. A la racine de votre site, créez un nouveau fichier intitulé `message.php` et saisissez-y le code ci-dessous :

```
<?php
if((isset($_GET["nom"])) && ($_GET["message"]!="")){
  $fp=fopen("message.txt","a+");
  $message="<tr><td>".$_GET["nom"].
    "<td>".$_GET["message"]."\n";
  fwrite($fp,$message);
  fclose($fp);
  echo "<div style='color:red'>Enregistrement effectué".
    "</div>";
}
else{
  echo "<div style='color:green'>Aucun enregistrement ".
    "effectué</div>";
}
?>

<form>
<table>
<tr><td align=right>Nom :<td><input type=text name=nom>
<tr><td align=right>Message :
<td><textarea cols=30 rows=4 name=message></textarea>
<tr><td colspan=2 align=center>
<input type=submit value="Enregistrer le message">
</table>
</form>

<?php
if(is_file("message.txt")){
  echo "<table border=1>";
  include("message.txt");
```

```
31    echo "</table>";
32 }
33 ?>
```

2. a. Affichez cette page dans votre navigateur en saisissant l'URL :
   `http://127.0.0.1/message.php`

   b. Saisissez plusieurs fois du texte dans les champs et validez votre formulaire.

   c. Quelle semble être l'utilité de cette page ?

. . . . . .

. . . . . .

   d. En inspectant le code, pouvez-vous dire où seront stockés ces messages ? Vérifier votre conjecture.

. . . . . .

   e. Vérifiez que, en fermant puis ré-ouvrant votre navigateur et cette page, les messages étaient toujours présents.

Le test "`(isset($_GET["nom"])) && ($_GET["message"]!="")`" permet de vérifier :

- La présence de la variable `nom` dans l'URL. Cela signifie a de très fortes chances que le formulaire a été préalablement soumis.
- Un texte a été saisi dans le champ texte.

Dans ce cas, le test retourne la valeur `true` et les lignes de 3 à 9 seront exécutées :

- Le fichier `message` est ouvert en mode rajout à la fin à l'aide de la commande `fopen(...)` ;
- la commande `fwrite(...)` y écrirera le texte transmis par le formulaire.

3. a. Saisissez un nouveau message contenant des sauts à ligne dans le message. Ceux-ci apparaissent-ils en bas de page après l'enregistrement ?

. . . . . .

   b. A la ligne 3, insérez une nouvelle ligne contenant le code suivant :

```
$_GET["message"]=str_replace("\n","<br>",$_GET["message"]);
```

   c. Effacer le fichier `message.txt` se trouvant à la racine de votre site, puis resaisissez des messages contenant des sauts de lignes.

. . . . . .

. . . . . .

Les messages sont saisies dans le champ texte mais les sauts de lignes sont alors codés par le caractère `\n` alors qu'ils s'affichent dans une page Web, il faut que l'élément br soit présent.
La commande `str_replace(...)` va changer tous les sauts de lignes codés par `\n` par des balises `<br>` indiquant dans le langage `HTML` des sauts de lignes.

4. Comment justifier qu'en bas de page, les données soient mises en page dans un tableau ?

.....

.....

La commande `include(...)` va placer le contenu du fichier `message.txt` à l'endroit de son exécution. Les deux commandes `echo` entourant cette dernière, place le contenu du fichier dans un élément `HTML` table.
Lors de l'enregistrement du formulaire dans le fichier `message.txt` *(lignes 3 à 7)*, le rajout des balises HTML `<tr>` et `<td>` a préformaté le contenu du fichier pour l'affichage dans un tableau.

Exercice 9

1. A la racine de votre site, créez un nouveau fichier intitulé `arborescence.php` et saisissez-y le code suivant :

```php
<?php
function explore($chemin,$trait=""){
  $dossier=opendir($chemin);
  while($fichier=readdir($dossier)){
    if(($fichier==".") ||($fichier="..")){
      continue;
    }
    if(is_dir($chemin."/".$fichier)){
      echo "dossier: ".$chemin."/".$fichier."<br>";
      explore($chemin."/".$fichier,"--".$trait);
    }
    else{
      echo $trait.$fichier."<br>";
    }
  }
}

explore(".");
?>
```

2. Affichez cette page dans votre navigateur en saisissant l'URL suivante : `http://127.0.0.1/arborescence.php`.

3. a. Qu'affiche ce script ?

. . . . . .

b. Créez un dossier nommé "`dossierTest`" à la racine de votre site et placez quelques fichiers au hasard.

c. Rafraichissez votre page pour exécuter à nouveau ce script. Que remarquez vous ?

. . . . . .

. . . . . .

Ce script parcourt l'arborescence des fichiers en partant de la racine de votre site.
Il affiche les fichiers des sous-dossiers en les précédant de tirer.
Ce script permet d'avoir une vue globale des fichiers présents à la racine de votre site.

4. Voici quelques questions sur le fonctionnement de ce script ; lisez préalablement la partie théorique pour mieux apréhender les fonctions intervenant dans ce script :

a. Le premier appel à la fonction `explore()` est effectuée avec quels arguments ? Que signifie-t-il ?

. . . . . .

b. A quelle condition un nouvel appel à la fonction `explore(...)` est-il effectué ?

. . . . . .

. . . . . .

c. A quel moment la boucle `while(){}` arrête son exécution ?

. . . . . .

. . . . . .

L'appel à la fonction `explore()` est effectué avec comme premier argument `"."` représentant le dossier courant où s'exécute le script ; le second argument est la valeur par défaut de `$trait` qui est une chaîne vide.
A l'intérieur du corps de la fonction `explore(...)` :

- la variable `$dossier` est initialisée avec la chaine de caractères localisant le dossier en cours d'étude ;
- La boucle `while(...) {...}` s'exécute tant que la commande

`readdir($dossier)` retourne `true`, c'est à dire tant que tous les fichiers se trouvant dans le dossier, désigné par `$dossier`, n'auront pas été inspectés ;

- Le test "`is_dir($chemin."/".$fichier`" retourne `true` si le fichier `$chemin."/".$fichier` en cours d'inspection est un dossier : dans ce cas, on fait un appel récursif à la fonction `explore()` afin d'explorer ce nouveau dossier.

5. Quelles techniques ont été utilisés pour que les fichiers contenus dans les sous-dossiers soient présentés avec plus ou moins de trait "-" afin d'indiquer leur niveau de présence dans les sous-dossiers :

.....

.....

Le premier appel à la fonction `explore(...)` s'effectue avec un seul argument. Ainsi, le paramètre formel `$trait` prend sa valeur par défaut : la chaine de caractères vide.

Lors des autres appels à la fonction `explore(...)`, le second argument est fourni sous la forme `"--".$trait` : lors de ce nouvel appel, la variable `$trait` sera initialisée avec deux traits "-" supplémentaires indiquant ainsi sa position dans les sous-répertoires.

Exercice 10

1. A la racine de votre site, créez un nouveau fichier intitulé `lire.php` et saisissez-y le code suivant :

```php
<?php
if(!is_file("a.txt")){
  echo "Le fichier \"a.txt\" n'existe pas";
}

$x=file("a.txt");
for($i=0;$i<count($x);$i++){
  echo $x[$i]."<br>";
}

echo "<p>";

$x=file_get_contents("a.txt");
echo $x;
?>
```

2. Affichez cette page dans votre navigateur en saisissant l'URL `lire.php`.

3. a. Pourquoi un message d'erreur s'affiche ?

. . . . . .

. . . . . .

b. Ajoutez l'instruction "`exit;`" à la fin de la structure conditionnelle `if` débutant à la ligne 2.

c. Rafraichir cette page afin de vérifier l'abscence de messages d'erreurs.

La structure conditionnelle affiche sa clause `true` dans le cas de l'abscence du fichier `a.txt` a la racine du site mais une fois cette clause exécutée, le reste du programme est évalué.
Ainsi, arrivant à la ligne 6, `PHP` provoque une erreur en essayant d'exécuter la commande `file(...)` sur un fichier inexistant.
L'instruction "`exit;`" permet d'arrêter l'exécution de tout le script à l'intérieur de la clause `true` de la structure conditionnelle : le reste du programme ne sera pas évalué.

4. a. A la racine du site, créez un fichier intitulé `a.txt` et saisissez-y un peu de textes sur plusieurs lignes.

b. Après avoir exécuté le code, combien de fois le contenu du fichier `a.txt` est-il affiché ? Quelle particularité comporte le second affichage ?

. . . . . .

. . . . . .

c. Modifiez la boucle `for(...){...}` pour afficher, en début de ligne, le numéro de chaque ligne du fichier `a.txt`

d. Les sauts de lignes dans un fichier texte sont représentés par le caractère `\n` ; or, dans une page Web, les sauts de lignes sont représentés par la balise <`br`>. Utilisez la commande `str_replace()` pour que, dans l'affichage du fichier `a.txt` par la commande `file_get_contents(...)`, soit présentés des sauts de lignes.

Voici la description des deux affichages du fichier `a.tx` réalisés par ce script :

- A l'aide de la commande `file(...)`, la variable `$x` est une variable-tableau dont les élèments sont chacunes des lignes du fichiers `a.txt`.
  La boucle `for(...){...}` affiche les éléments de la variable `$x` en ajoutant à

chaque fois un saut de ligne `<br>` ;

- La commande `file_get_contents(...)` récupère l'intégralité du contenu du fichier `a.txt` et l'affecte à la variable `$x`.
  Celle-ci sera affichée sans aucun traitement.
  Les sauts de lignes étant codés dans un fichier texte par le caractère `\n`, ils ne seront pas affichés lors de l'instructions "`echo $x`".

Pour ajouter la numérotation des lignes, il suffit de modifier la ligne `8` en :

```
echo $i."- ".$x[$i]."<br>";
```

Pour que le fichier `a.txt` s'affiche la seconde fois avec les sauts de lignes, il faut changer la ligne `14` par l'instruction :

```
echo str_replace("\n","<br>",$x);
```

C. Quelques travaux pratiques :

Exercice 11

Ecrire un script `PHP` affichant tous les fichiers ayant l'extenstion "`.txt`" présent à la racine de votre hébergement et dans ses sous-dossiers.

Exercice 12

La fonction `rename(...)` *(voir la documentation sur Internet)* permet de renommer les fichiers présents sur le serveur ; utilisez les commandes `strtolower(...)` et `str_replace(...)` afin de renommer tous les fichiers pour que ceux-ci ne contiennent aucune majuscule et aucun espace.

Exercice 13

Créez un script `PHP` créant une bandeau de navigation présentant les sous-dossiers présent à la racine de votre site sous la forme de lien hypertexte.

III. Les bases de données MySQL :

A. Un peu de théorie :

1. *Introduction :*

Le but de ce chapitre est de vous introduire à l'utilisation des bases `MySQL` à travers le langage `PHP` ; ce cours se limitera à apporter les bases pour utiliser cette base de donnée.

La base de donnée est un espace de stockage offert à chaque utilisateur. Pour y placer des données, on crée au préalable des tables dans lequel on effectuera nos enregistrements de données ; normalement, vous n'êtes pas limité dans le nombre de table que contiendra votre base de donnée.

Imaginons que vous souhaitez créer une page qui enregistre des commandes pour un magasion ; les informations stockées seront :

- Numéro de commande ;
- Nom et prénom ;
- Adresse de livraison ;
- Référence des objets achetés.

Cette table comportera quatre champs *(colonnes)* et chaque enregistrement d'une commande sera inscrite dans la table dans une nouvelle ligne.

Si vous ne disposez pas d'une base de donnée mais que votre hébergeur vous donne le droit d'écrire des fichiers, vous pouvez vous inventer une mini-base de donnée où chaque champ sont séparés par des virgules.
Mais les bases de données proposent d'autres avantages :

- La gestion des utilisateurs pouvant utiliser une base et leurs droits sur cette base ;
- Une plus grande rapidité de temps d'accés ;
- Des outils de recherche de donnée à l'intérieure d'une base.

2. *Les types de champs :*

Lors de la création d'une table, on doit préciser la nature *(de manière très précise)* de la nature des objets contenus dans ce champ ; voici une liste *(non exhaustive)* de la nature des champs :

- Un entier de type `TINYINT` a ses valeurs allant de :
  -128 à 128.
- Un entier de type `INT` a ses valeurs allant de :
  -2147483648 à 2147483647.

- Une chaîne de caractère `TEXT` peut contenir jusqu'à 2^{16} octects *(2^{16} caractères pour l'encodage ASCII).*
- Une chaîne de caractère `LONGTEXT` peut contenir jusqu'à 2^{32} octects *(2^{32} caractères pour l'encodage ASCII).*

Il existe également les champs de types `BLOB` qui permettent de contenir aussi des grandes chaînes de caractères *(permettant de stocker des fichiers)* : la seule différence entre les champs de type `TEXT` et `BLOB` est lors de recherche de valeurs dans le contenu ; la recherche sur les champs `TEXT` est insensible à la casse alors que `BLOB` est sensible à la casse.

3. *Les clefs primaires :*

Lors de la création d'une tablea, il est possible de définir un de ces champs *(de type* `TINYINT`, `INT`*...)* comme clef primaire : les clefs permettent d'identifier et de rendre unique chacun des enregistrements. A chaque nouveau enregistrement, cette valeur sera incrémentée de `1` permettant d'identifier chacun des enregistrements.

Pour créer un chaîne primaire, il faut effectuer la paramétrisation suivant :

`Not Null` ; `AutoIncrement` ; `Unsigned`

Ce champ ne doit pas avoir de valeur par défaut *(laissez la vide).*

Vous pouvez alors passer cette variable comme clef primaire.

4. *Les chaînes de caractères et* `MySQL` *:*

On utilise souvent les chaînes de caractères pour stocker des informations *(nom, adresse,...)* dans une table. Or, les fonctions `PHP`, permettant d'exécuter les requêtes sur notre base et nos tables, prennent pour argument des chaînes de caractères : ainsi, nous allons devoir construire des chaînes de caractères contenant d'autres chaînes.

Voici un exemple valide d'une telle chaîne :

```
"nom='".$_GET["nom"]."', age=18, prenom=ẗhomas¨
```

Un autre problème peut se poser ; on intègre souvent, dans nos requêtes, des données venant d'un formulaire, elles peuvent alors contenir également des guillemets simples et doubles. Que se passe-t-il si, dans le code précédent, la variable `$_GET["nom"]` contient la valeur `D'alembert` ? Après concaténation la chaîne précédente devient :

```
"nom='D'alembert', age=18, prenom=ẗhomas¨
```

1. Pour `PHP`, cette chaîne est correcte : elle commence et se termine par un guillemet simple. Les autres guillemets doubles sont échappés par le signe `\` et sont donc considérés comme de simples caractères.

2. Par contre MySQL, reçoit le code suivant :

```
nom='D'alembert', age=18, prenom="thomas"
```

Mysql renvoit une erreur car le champ nom reçoit la valeur D, mais le reste alembert' ne représente pas une instruction pour le serveur MySQL.

Si votre serveur d'hébergement a l'option magic_quotes_gpc activée, les guillemets seront protégées lors de leur récupération par la variable $_GET ; si ce n'est pas le cas, vous devez protéger toutes données récupérées en les passant en argument à la commande addslashes().

5. *Les requêtes* MySQL *:*

Les quatres principales requêtes qu'on peut effectuer sur une base MySQL sont SELECT, INSERT, UPDATE et DELETE.

On utilise les clauses :

- WHERE ... pour cibler certains enregistrements d'une table lors de sa lecture, de modifications ou d'effacements d'enregistrements.
- Lors de la lecture d'une table, on peut ordonner les enregistrements récupérer à l'aide de la clause ORDER BY ...

L'illustration de ces clauses se fera à travers l'étude des requêtes suivantes :

- SELECT : cette commande permet de récupérer les enregistrements d'une table
  - Pour récupérer tous les enregistrements de la table test :

    ```
    SELECT * FROM test
    ```

  - Pour récupérer seulement les champs nom et adresse de tous les enregistrements de la table test :

    ```
    SELECT nom,adresse FROM test
    ```

  - Pour récupérer tous les enregistrements de la table test dont le champ nom a pour valeur "thomas" :

    ```
    SELECT * FROM test WHERE nom="thomas"
    ```

    Et pour qu'en plus, le champ âge ait une valeur supérieure à 18 ans :

    ```
    SELECT * FROM test WHERE nom="thomas" && age=18
    ```

  - Pour ordonner les données dans l'ordre alphabétique des noms :

    ```
    SELECT * FROM test ORDER BY nom
    ```

    ou alors dans l'ordre décroissant :

    ```
    SELECT * FROM test ORDER BY nom DESC
    ```

- Voici comment insérer un nouvel enregistrement dans la table `test` :

```
INSERT INTO test (nom,ville,age) VALUES ("pascal","sete",20)
```

- Voici comment modifier l'enregistrement précédent :

```
UPDATE test SET age=40, ville="montpellier" WHERE nom="pascal"
```

  Si deux enregistrements ont le champ `nom` avec la valeur `"pascal"`, ils seront modifiés simultanément par cette requête ; pour eviter cela, on utilise les clefs primaires pour identifier de manières uniques chaque enregistrement de la table.

- Voici comment effacer de la table `test` tous les enregistrements dont la ville est `"paris"` :

```
DELETE FROM test WHERE ville="paris"
```

6. *Les commandes* PHP *:*

- `$bp=mysql_connect($server,$user,$password)` : les trois arguments sont des chaînes de caractères ; cette commande va effectuée une connexion au serveur `MySQL` localisé par l'adresse `$server`. La connexion s'effectuera avec l'identifiant `$user` et le mot de passe `$password`.
  Cette commande renvoie un pointeur `$bp` vers la connexion établie ; ceci n'est utile que dnas le cas où plusieurs connexions `MySQL` différentes seront ouvertes.

- `mysql_select_db()$base` : cette commande permet de se connecter à la base `$base` que vous donne votre hébergeur.

- `mysql_close()` : cette commande permet de fermer la connexion à la base courante.

- `$rst=mysql_query($requete)` : cette commande permet d'exécuter la requete `$rq` dans la base de donnée courante. Avec la requête `SELECT`, on récupère les lignes de résultats dans la variable `$rst`.

- `$ligne=mysql_fetch_array($rst)` : cette commande permet d'extraire ligne par ligne les résultats renvoyés par la commande `mysql_query()`.

- `mysql_num_rows($rst)` : cette commande renvoie un entier représentant le nombre d'enregistrements contenus dans la variable `$rst`.

- `mysql_error()` : cette commande retourne la dernière erreur rencontrée par le serveur `MySQL`. Voici un exemple d'utilisation permettant de stopper l'exécution du script en cas d'erreur et d'afficher le message d'erreur :

```
mysql_query("SELECT * FROM test") or die(mysql_error());
```

7. *Le logiciel HeidiSql :*

Ce logiciel offre une interface graphique *(fenêtre, boutons, champs textes...)* permettant de se connecter à une base `MySQL` et permet de la contrôler facilement.

Même si c'est au travers du `PHP` que nos scripts manipulerons les données de nos tables, ce logiciel reste pratique quant à la maintenant de nos bases de données ; ainsi, il nous facilitera grandement les tâches suivantes :

- Création des tables ;
- Créatino des utlisateurs et de leurs droits sur les tables de données ;
- Transfert des données d'une base à l'autre et sauvegarde dans un fichier externe...

B. Quelques exercices dirigés :

Exercice 14

Avant d'utiliser réellement les bases de données, nous allons devoir quelques spécificités des chaînes de caractères dans le langage `PHP` ; en effet, on passe les instructions au serveur MySQL via des commandes `PHP` qui prennent pour argument des chaînes de caractères. Or, dans la pratique les tables d'une base de donnée possèdent de nombreux champs de types "*chaînes de caractères*". Ainsi, nous aurons à génerer des chaînes de caractères contenant d'autres chaînes de caractères. 1111 Nous allons également voir comment le serveur `Apache` peut modifier les chaînes de caractères lorsqu'on valide un formulaire :

1. a. Dans un fichier nommé `info.php`, saisissez le code suivant :

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

b. Exécutez le script et chercher les valeurs de ces deux paramètres :

`magic_quotes_gpc=`.....

`magic_quotes_runtime=`.....

2. a. Saisissez le code suivant dans un fichier nommé `guillemet` :

```php
<?php
$nom=(isset($_GET["nom"])?$_GET["nom"]:"");
$message=(isset($_GET["message"])?$_GET["message"]:"");
?>

<form action="" method="get">
Nom : <input type=text name=nom value="<?php echo $nom; ?>">
<br>
```

```
9  message : <textarea name=message>
10 <?php echo $message; ?>
11 </textarea>
12 <br>
13 <input type=submit value="Soumettre">
14 </form>
15
16 <?php
17 echo "Le nom est &quot;".$nom."&quot;<br>".
18   "Le message est &quot;".$message."&quot;";
19 ?>
```

b. Saisissez quelques mots dans chacun des champs de texte puis soumettez le formulaire.

.....

.....

c. Continuez l'exercice en fonction de la configuration du serveur qui héberge votre site.

- Si `magic_quotes_gpc=on`, passez à la question 3. puis 5. ;
- Si `magic_quotes_gpc=off`, passez à la question 4. puis 5. .

3. a. Ajoutez dans chacun de ses deux champs textes un guillemet double ", puis soumettez votre formulaire ; justifiez que les deux lignes d'affichage en bas de la page représentent bien une chaîne de caractères :

.....

.....

b. L'affichage dans les deux champs texte est-il correct ?

.....

.....

c. Dans votre code, lors de l'affichage des variables `$nom` et `$mess`, faites passer ces variables en argument à la fonction `stripslashes()`. Soumettez de nouveau ce formulaire ; quel changement a été opéré ?

.....

4. a. Ajoutez dans chacun de ses deux champs textes un guillement double ", puis soumettez votre formulaire ; qu'observez-vous ?

.....

.....

b. Pouvez-vous dire que la dernière ligne contient une chaîne de caractères définie par les guillemets doubles " .

.....

c. A la ligne `17` et `18`, faites passer les variables `$nom` et `$message` en argument à la fonction `addslashes()`. Réessayez ce code :

. . . . . .

d. Les textes présents dans les deux champs textes sont-ils corrects ?

5. Etudions maintenant l'absence du guillemet double dans le premier champ texte :

a. Dans le code HTML ci-dessous, déterminer la valeur de l'attribut `value` :

```
<input type=text name=nom value="thomas"pascal">
```

b. Remplacez le guillemet `"` situé entre les deux prénoms `thomas` et `pascal` par `&quot;`. Réaffichez ce code.

L'option `magic_quotes_gpc=on` de configuration de `Apache` entraîne le serveur Web à rajouter le caractère `\` devant les guillemets doubles, les anti-slash et les caractères `NULL` pour toutes les variables venant des trois variables globales :

`$_GET` ; `$_POST` ; `$_COOKIES`

Cette fonctionnalité permet d'insérer plus facilement les valeurs de ces variables à l'intérieur de chaines de caractères ; notamment lors d'enregistrement de données dans une table `MySQL`.

L'option `magic_quotes_gpc=off` indique que les valeurs de ses variables sont transmises directement sans pré-traitement. Ainsi, pour que des guillemets présents dans ses valeurs ne "*cassent*" pas une chaîne de caractère, on utilise la fonction `addslashes()` qui rajoute les caractères d'échappements.

Le caractère `\` est un caractère d'échappement pour les langages de programmation `PHP` et `JavaScript` ; le système pour échapper les caractères en HTML est différent :

Le caractère `"` est remplacé par `&quot;`

Exercice 15

1. Nous allons utiliser le logiciel `HeidiSQL` pour créer une base de donnée `MySQL` :

a. Connectez vous à votre base de donnée à l'aide des paramètres suivants :

`Hostname/IP : woezon.net` ; `User :` votre nom ; `Password : xxxx`

b. Faîtes un clic droit sur votre base de donnée apparaissant dans le panneau gauche du logiciel et sélectionnez l'option :

"*Create table...*"

c. Dans la nouvelle fenêtre apparaîssant, saisissez "*test*" pour nom de cette table et

arranger vous pour créer les champs *(columns)* suivant dans votre table :

- `numero` de type "*TINYINT*" avec les options :

  `Unsigned` ; `Not Null` ; `AutoIncrement` ; `primary`

  La valeur par défaut doit être vide.
- `nom` est de type "*TEXT*".
- `message` est de type "*TEXT*".

Le champ `numero` est l'"*index primaire*" primaire de cette table : il doit apparaître, à gauche de son nom, un clé jaune.

2. a. Dans un fichier `testBase.php`, saisissez le code suivant :

```php
<?php
mysql_connect("127.0.0.1","root","") or die("pas de ↩
   connexion");
mysql_select_db("exogroupe") or die("pas de base");;
if(isset($_GET["nom"])){
  $r="INSERT INTO test (nom,message) VALUES ".
    "('".$_GET["nom"]."','".$_GET["message"]."')";
  mysql_query($r) or die($r."<br>".mysql_error());
}
?>

<form action="" method="get">
Nom : <input type=text name=nom><br>
Message : <textarea name=message></textarea><br>
<input type=submit value="Soumettre">
</form>
```

b. A l'aide de votre navigateur, exécutez ce script et soumettez quelques messages à votre serveur.

c. Ouvrez `HeidiSQL` et vérifiez l'apparition des messages dans la table `test`. Que pouvez-vous dire des valeurs du champ `numero`

. . . . . .

. . . . . .

3. a. Ajoutez le code suivant à la fin de votre fichier :

```php
<?php
$r="SELECT * FROM test";
$q=mysql_query($r) or die($r."<br>".mysql_error());
echo "<table border=1>";
while($rr=mysql_fetch_array($q)){
  echo "<tr><td>".$rr["nom"]."<td>".$rr["message"];
}
```

```
8 echo "</table>";
9 ?>
```

b. Exécutez le code précédent. Pouvez-vous donner le sens de la requête présentée par `$r` ?

.....

.....

c. Changer la requête `$r` par :

```
$r="SELECT * FROM test ORDER BY nom";
```

Quel est le rôle de la séquence `ORDER BY nom` ?

.....

.....

4. On souhaite que, lors de l'enregistrement des données dans la base `MySQL`, les guillemets et les anti-slash soient protégés par un anti-slash \

a. Quel est la valeur du paramètre `magic_quotes_gpc` de votre serveur d'hébergement ?

.....

b. Si c'est nécessaire, modifier la ligne du `INSERT` afin d'effectuer cette "*protection*".

c. Modifiez maintenant l'affichage des champs `nom` et `message` pour que ces données soient affichées sans cette protection.

5. a. Changez, dans le script, `$rr["nom"]` en :

```
"<a href=\"?del=".$rr["numero"]."\">".$rr["nom"]."</a>"
```

b. Ajoutez le code juste après la commande `mysql_select_db()` :

```
1 if(isset($_GET["del"])){
2   $r="DELETE FROM test WHERE numero=".$_GET["del"];
3   mysql_query($r) or die($r."<br>".mysql_error());
4 }
```

c. A quel moment ce code est-il exécuté ? Préciser :

.....

.....

6. Voyons maintenant comment faire pour modifier un enregistremenet existant dans une table `MySQL` :

a. Modifiez la partie HTML contenant la définition du formulaire form en :

```
1 <form action="" method="get">
2 <?php
3 $nom="";
4 $message="";
```

```php
if(isset($_GET["del"])){
  echo"<input type=hidden name=num value=".
    $_GET["del"].">";
  $r="SELECT * FROM test WHERE numero=".$_GET["del"];
  $q=mysql_query($r) or die($r."<br>".mysql_error());
  $rr=mysql_fetch_array($q);
  $nom=htmlspecialchars(stripslashes($rr["nom"]));
  $message=stripslashes($rr["message"]);
}
?>
Nom :
<input type=text name=nom value="<?php echo $nom; ?>"><br>
Message : <textarea name=message>
<?php echo $message; ?>
</textarea><br>
<input type=submit value="Soumettre">
</form>
```

b. Modifiez la structure conditionnelle du début de code en :

```php
if(isset($_GET["nom"])){
  if(isset($_GET["num"])){
    $r="UPDATE test SET nom='".$_GET["nom"]."', ".
      "message='".$_GET["message"]."' ".
      "WHERE numero=".$_GET["num"];
  }
  else{
    $r="INSERT INTO test (nom,message) VALUES ".
    "('".$_GET["nom"]."','".$_GET["message"]."')";
}
  mysql_query($r) or die($r."<br>".mysql_error());
}
```

c. Vérifiez qu'il est maintenant possible de modifier des enregistrements de votre base de donnée.

Cet exercice nous a permis d'utiliser les principaux outils de contrôle d'une base `MySQL` : `INSERT`, `DELETE`, `UPDATE`.
Il a également mis en évidence l'importance de protéger les guillemets *(simple ou double)* afin d'insérer plus facilement les données dans des chaînes de caractères.

Voici une description de cet exercice question par question :

1. On peut facilement utiliser le logiciel `HeidySQL` pour manipuler les enregistrements de nos tables. Mais, il sera toujours plus agréable d'afficher les résultats dans une page Web *(lorsque les données peuvent être affichées)*. La création de la table de donnée aurait pû également passer par l'exécution suivant du code

MySQL :

```
CREATE TABLE "test" (
  "numero" tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
  "nom" text,
  "message" text,
  PRIMARY KEY ("numero")
)
```

Vous pouvez lancer ce code dans l'onglet "*Query*" de HeidySQL ou à travers le PHP à l'aide de la commande `mysql_query()`.

2. Avant d'utiliser une base MySQL, il faut s'y connecter ; pour cela, on utilise les commandes :

   `mysql_connect()` ; `mysql_error()`

   L'ajout du mot-clef `or die()` permet à PHP de tester si la commande précédent a renvoyé une erreur ; dans ce cas, PHP stoppe son exécution et affiche le message passé en argument à la fonction `die()`.

   Si le formulaire a été validé, la commande `mysql_query` insère un enregistrement dans la table `test` en utilisant les valeurs `$_GET["nom"]` et `$_GET["message"]` représentant les valeurs saisies dans les champs du formulaire.

3. La commande `SELECT` permet de récupérer des enregistrements dans la table `test` *(voir la partie théorique pour plus de renseignement)*. La variable `$q` retournée par la commande `mysql_query` est une sorte de tableau ; mais pour extraire ces différents éléments, il est nécessaire d'utiliser la commande `mysql_fetch_array()`.

   La conditionnelle `$rr=mysql_fetch_array($q)` renvoie `false` seulement si le pointeur `$q` ne représente plus aucune donnée de notre requête. Ainsi, la boucle `while` s'interrompt lorsque tous les enregistreement correspondant à notre requête ont été observés.

4. Pour avoir une homogénéité dans notre base de donnée et avoir une facilité d'utilisation, il est toujours plus facile de protéger les guillemets et les anti-slashs présents dans nos enregistrements.

   Cela évitera de nombreuses erreurs lors de l'affichage de nos enregistrements ; de leurs modifications ou de leurs insertions dans scripts JavaScript

5. La modification, ici proposée, du code permet d'effacer un enregistrement lorsque le client clique sur un lien ; on voit dans cette question, pour la première fois, l'utilité d'une clé primaire dans une base de donnée : identifier de manière unique

chacun des enregistrements.

6. Dans cette question, on transforme le code afin que le lien sert maintenant à modifier le contenu d'un enregistrement. Pour cela :

   a. Si le lien a été activé, la page se recharge et avant d'afficher les champ texte, on récupère les données de cet enregistrement :

   ```
   $r="SELECT * FROM test WHERE numero=".$_GET["del"];
   ```

   On intègre ensuite ces résultats dans nos champs textes ; remarquez les commandes :

   - `stripslashes()` : elle permet d'enlever les protections sur les caractères sensibles obtenant ainsi un affichage normal.
   - `htmlspecialchars()` : elle transforme certains caractères en code HTML : notamment le guillement double pour pouvoir insérer la valeur dans l'attribut :

     value="..."

7. L'existence de `$_GET["nom"]` dans l'URL nous assure que le formulaire a été soumis au serveur :

   - Si `$_GET["num"]` existe alors le client travaillait sur la modification d'un enregistrement ; d'où la présence de la requête `UPDATE`.
   - Sinon le client est en train d'effectuer un nouvel enregistrement ; d'où la présence de la requête `INSERT`.

Exercice 16

Cet exercice va nous montrer comment enregistrer un fichier dans une base de donnée `MySQL` :

1. Saisissez le code suivant dans un fichier appelé `upload.php` :

```php
<?php
  //mysql_connect("127.0.0.1","root","") or die("pas de ↩
     connexion");
  //mysql_select_db("exogroupe") or die("pas de base");;

  if(isset($_POST["utilisateur"])){
    echo "<pre>";
    print_r($_FILES);
    echo "</pre>";
  }
```

```
10 ?>
11
12 <form method="post" action="" enctype="multipart/form-data">
13   Utilisateur : <input type=text name=utilisateur>
14   <br>
15   Fichier : <input type=file name=monFichier>
16   <br>
17   <input type=submit value="Soumettre">
18 </form>
```

2. Dans votre navigateur, ouvrez la page ; dans les champs, renseignez votre nom et sélectionnez un fichier à l'aide du bouton "*Parcourir...*". Soumettez le formulaire :
   a. En début de page s'affiche le contenu d'un tableau ; de quel tableau s'agit-il ? Le tableau principal contient combien d'éléments ?

.....

.....

   b. Donner quelques explications sur les éléments du tableau `monFichier` :

.....

.....

3. Nous allons maintenant créé la base de donnée qui nous permettra d'enregistrer les fichiers dans une base de donnée :
   a. Dans `HeidiSQL`, après avoir ouvert votre base de donnée, sélectionnez l'onglet "*Query*" ; dans le champ texte ajouté le code suivant :

```
1 CREATE TABLE "fichiertable" (
2   "numero" int(10) unsigned NOT NULL AUTO_INCREMENT,
3   "utilisateur" text,
4   "nom" text,
5   "fichier" blob,
6   "type" text,
7   PRIMARY KEY ("numero")
8 )
```

   Puis, valider la création de la base de donnée.

   b. Quel est le nom de cette table ?

.....

   c. Combien de champs contient cette table ? Quels sont les noms de ces champs ?

.....

.....

4. Nous allons passer à l'enregistrement des fichiers dans notre base de donnée ; on suppose

que le serveur de votre hébergeur a pour configuration :

`magic_quotes_gpc=on`

a. Modifiez la structure conditionnelle de notre script en :

```php
if(isset($_POST["utilisateur"])){
  $user=$_POST["utilisateur"];
  $name=addslashes($_FILES["monFichier"]["name"]);
  $file=file_get_contents($_FILES["monFichier"]["tmp_name"]);
  $file=addslashes($file);
  $type=addslashes($_FILES["monFichier"]["type"]);
  $r="INSERT INTO fichiertable (utilisateur,nom,fichier,type) VALUES ('".$user."','".$name."','".$file."','".$type."')";
  mysql_query($r) or die($r."<br>".mysql_error());
}
```

b. Enlevez les commentaires devant les deux premières lignes du script permettant à votre script de se connecter à la base de donnée.

c. Effectuez quelques enregistrements de fichier ; afin de s'assurer de l'efficacité du script, observez le contenu de votre base à l'aide de `HeidySQL`.

d. Sur les différents enregistrements effectués, comparez les valeurs du champ `type` ? Que représente ces valeurs ?

.....

.....

e. Justifier que dans la requête `INSERT`, pourquoi seule la variable `$user` n'est pas passé en argument à la fonction `addslashes()` ?

f. Que contient la variable `$file` ?

.....

.....

5. Nous allons afficher les différents fichiers présents dans cette page :

```php
<?php
$r="SELECT * FROM fichiertable";
$q=mysql_query($r) or die($r."<br>".mysql_erro());
echo "La base contient ".mysql_num_rows($q)." enregistrement\n".
  "<table border=1 width=100%>";
while($rr=mysql_fetch_array($q)){
  echo "<tr><td>".$rr["numero"]."<td>".$rr["utilisateur"].
    "<td>".$rr["nom"]."<td>".$rr["type"];
}
echo "</table>";
```

```
11 ?>
```

6. Permettons maintenant aux clients de télécharger les fichiers présents dans cette base de donnée :

a. Dans le script précédent, modifiez le code :

```
$rr["numero"]
```

par :

```
<a href="affiche.php?num=".$rr["numero"].">".$rr["numero"]."</a>"
```

b. Dans un fichier nommé `affiche.php`, saisissez le code suivant :

```
1  <?php
2  mysql_connect("127.0.0.1","root","") or die("pas de ↩
       connexion");
3  mysql_select_db("exogroupe") or die("pas de base");;
4
5  $r="SELECT * FROM fichiertable WHERE numero=".$_GET["num"];
6  $q=mysql_query($r) or die($r."<br>".mysql_error());
7  $rr=mysql_fetch_array($q);
8
9  $taille=strlen($rr["fichier"]);
10 header("Content-Length: ".$taille);
11 header("Content-Type: ".$rr["type"]);
12 header("Content-Disposition: attachment; filename='".$rr["↩
       nom"]."'");
13 ?>
```

c. Actualiser la page permettant l'enregistrement de fichier dans notre table et observez l'apparition des liens ; cliquez sur un de ces liens pour vérifier le téléchargement d'un fichier.

C. Quelques travaux pratiques :

Exercice 17

Reprenez le script précédent et modifiez la table afin que la taille du fichier soit inscrite dès l'enregistrement dans la table.

Modifiez également le script et la table pour que chaque utilisateur puisse laisser un descriptif du fichier.

Exercice 18

Créez un script qui permet d'enregistrer chaque visite de pages de votre site afin de connaître la page la plus populaire. Vous pouvez imaginez récupérer l'adresse IP de l'utilisateur permettant de connaître le nombre de pages moyen affichées par chaque visiteur.

Index