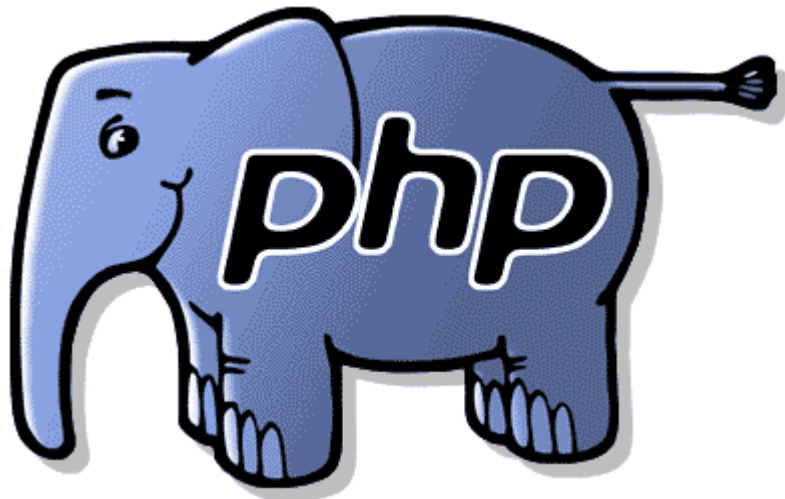


Charles COLSON

PROGRAMMATION PHP





PROGRAMMATION PHP

Charles COLSON

Sommaire	2
PARTIE 1 : Codage de base	3
I. Introduction	4
II. Les bases	5
III. Les variables	6
IV. Les fonctions	8
V. Les conditions	9
VI. Les boucles	11
VII. Les tableaux	12
PARTIE 2 : Base de données	13
I. Introduction	14
II. Php my admin	15
III. Lire des données	17
IV. Ecrire des données	19
PARTIE 3 : Codage amélioré	20
I. Les includes	21
II. Les fonctions intrinsèques	22
III. Les formulaires	24
IV. Les temps	25
V. Les superglobales	26
VI. Les regex	27
VII. Les htaccess	28
VIII. Lire et écrire sur un fichier	29



PARTIE 1

CODAGE DE BASE



I. Introduction

Avant de commencer ce tutorial sur le PHP, il faut bien maîtriser le HTML car nous allons ajouter des instructions PHP sur des pages HTML. Grâce au PHP vous allez pouvoir créer des pages dynamiques et non plus statiques, ce qui vous permettra d'insérer des éléments interactifs sur vos pages (chat, compteur...). La différence avec le HTML, c'est que les pages sont générées par le serveur.

Pour faire des pages en PHP vous devez avoir :

- Un éditeur de texte pour créer les pages (bloc notes par exemple), il faudra veiller à enregistrer les pages au format .php et non .txt
 - Un navigateur Internet pour afficher les pages créées (Internet explorer par exemple)
 - Un serveur pour générer les pages afin de tester vos créations en local (directement sur votre disque dur), c'est le rôle d'Easy PHP :
1. Téléchargez Easy PHP, puis installez-le,
 2. Une icône Easy PHP apparaît dans la barre des tâches, faites un clic droit dessus et cliquez sur Administration,
 3. La page d'administration s'ouvre, en dessous de Alias cliquez sur ajouter, puis indiquez le nom de votre site, son chemin (ou il se trouve sur votre disque dur) et ne touchez pas aux paramètres,
 4. Fermez Easy PHP puis relancez le (pour prendre en compte les modifications faites),
 5. Pour accéder à votre site afin de le tester, vous devrez vous rendre sur la page d'administration (voir 2) et cliquer sur l'alias correspondant à votre site après avoir placé dans son répertoire la page que vous désirez tester.
 6. Sachez que vous pouvez aussi administrer vos bases de données en cliquant sur Gestion BDD dans la page d'administration (nous utiliserons ceci dans la partie 2).

Afin de mieux comprendre chaque élément, il est recommandé de tester chaque exemple de ce cours.



II. Les bases

Vous savez ce qu'est une balise (voir cours sur le html), en PHP elles commencent par `< ?php` et finissent par `?>` (il en existe d'autres mais ce sont les plus fréquentes). Pour afficher du texte par l'intermédiaire d'une instruction PHP voici ce qu'il faut taper dans un éditeur de texte :

```
<html>
<head>
<title>Titre de la page</title>
</head>
<body>
Cette ligne est écrite en HTML.<br>
<?php
echo 'Cette ligne est écrite en PHP' ;
?>
</body>
</html>
```

Ce qui apparaîtra à l'écran :
Cette ligne est écrite en PHP

- `< ?php` et `?>` indique le début et la fin du code PHP
- `echo` permet d'afficher le texte, les guillemets ou quotes indiquent le texte à afficher
- `;` permet d'indiquer que c'est la fin de l'instruction PHP

On peut aussi ajouter des balises HTML dans un `echo` telles que mettre en gras :

```
<?php
echo 'Cette ligne est écrite en
<b>PHP</b>' ;
?>
```

Ce qui apparaîtra à l'écran :
Cette ligne est écrite en **PHP**

Pour ajouter un commentaire (quelque chose qui ne sera pas vu par les visiteurs) on met : `//`

```
<?php
// Voici une ligne de commentaire
?>
```

Ce qui apparaîtra à l'écran :



III. Les variables

Une variable est une information qui est stockée temporairement, elle est constituée d'un nom pour l'identifier, qui est toujours précédé du signe \$ et d'une valeur qui correspond à la donnée qu'elle contient. Il existe trois sortes de données :

A. Les types

- Le texte :

```
<?php
$nom_de_la_variable = "texte";
?>
```

- *\$nom_de_la_variable* indique le nom que l'on veut donner à sa variable
- = signifie équivaut à
- «*texte*» indique la valeur, donc le texte que l'on veut, il se trouve entre guillemets
- ; signale que c'est la fin de l'instruction

- Les nombres :

```
<?php
$nom_de_la_variable = 12;
?>
```

- 12 indique la valeur, donc le nombre que l'on veut, il n'est pas entre guillemets

- Les booléens (sur le principe soit vrai soit faux) :

```
<?php
$variable_gagne = true;
$variable_perdu = false;
?>
```

- = *true* indique que *variable_gagne* est vrai, *true* ne se met pas entre guillemets
- = *false* indique que *variable_perdu* est faux, *false* ne se met pas entre guillemets

Pour afficher la valeur d'une variable il suffit d'ajouter l'instruction *echo* :

```
<?php
$pseudo_du_visiteur = "Charlie";
echo "Bonjour $pseudo_du_visiteur !";
?>
```

Ce qui apparaîtra à l'écran :
Bonjour Charlie

B. Concaténation

La concaténation permet d'associer deux variables :

```
<?php
$prenom = 'James';
$nom = 'Bond';
$name = $nom . $prenom . $nom;
echo 'Mon nom est' . $name;
?>
```

Ce qui apparaîtra à l'écran :
Mon nom est BondJamesbond

- *\$name = \$nom . \$prenom . \$nom*; la variable *\$name* prendra la valeur de la variable *\$nom* et (correspond au point) la valeur de la variable *\$prenom* et celle de *\$nom*. Soit pour valeur finale : *BondJamesBond*
- *echo 'Mon nom est' . \$name*; vous remarquez que l'on n'est plus obligé de mettre la variable *\$name* entre guillemets malgré la présence de texte (*mon nom est*) grâce à la concaténation

C. Quotes

```
<?php
$prenom = 'James';
echo '$prenom';
echo "$prenom";
?>
```

Ce qui apparaîtra à l'écran :
\$prenom
James

- La variable *\$prenom* entre simples quotes ne permet pas d'afficher son contenu, mais on a vu qu'il était inutile de mettre des quotes grâce à la concaténation

En utilisant les simples quotes, il faudra mettre des antislash (\) :

```
<?php  
echo 'Je t\'aime';  
?>
```

Ce qui apparaîtra à l'écran :
Je t'aime

En utilisant les doubles quotes, il faudra mettre des antislash (\) :

```
<?php  
echo "Cette ligne est écrite en \"PHP\"";  
?>
```

Ce qui apparaîtra à l'écran :
Cette ligne est écrite en « PHP »



IV. Les fonctions

Une fonction est une série d'instructions qui retourne une valeur, vous pourrez tout automatiser. Nous avons vu la fonction `main` dans les précédents programmes, à nous de créer nos propres fonctions !

Prenons un exemple d'une fonction pour afficher bonjour :

```
<?php
function Salut($nom)
{
echo "Bienvenue $nom !<br>";
}
Salut("Philippe");
Salut("Abdel");
Salut("Franck");
?>
```

Ce qui apparaîtra à l'écran :
Bonjour Philippe !
Bonjour Abdel !
Bonjour Franck !

- *function Salut* indique que vous créez une fonction qui s'appellera *Salut*
- (*\$nom*) est un paramètre, c'est en fait toutes les variables que la fonction va utiliser
- {...} le début et la fin des accolades indiquent le début et la fin de la fonction, entre les deux se trouve le contenu de la fonction, ce qu'on veut automatiser
- *DireBonjour(\$Philippe);* on fait appel à la fonction en indiquant entre parenthèses ses paramètres (ici le nom)

Prenons un autre exemple pour effectuer le calcul du périmètre d'un rectangle :

```
<?php
function perimetre_rectangle ($longueur,
$largeur)
{
perimetre = (2 * $longueur) + (2 *
largeur) ;
return $perimetre;
}

$perimetre = perimetre_rectangle(1.5,2);
echo "Le perimetre d'un rectangle ayant
un longueur de 1,5 cm et une largeur de 2
cm est de $perimetre cm.";
?>
```

Ce qui apparaîtra à l'écran :
Le périmètre d'un rectangle ayant
un longueur de 1,5 cm et une
largeur de 2 cm est de 7 cm.

- *function perimetre_rectangle* indique que vous créez une fonction *perimetre_rectangle*
- (*\$longueur, \$largeur*) sont les paramètres, les variables que la fonction va utiliser
- {...} on met entre les accolades la formule du périmètre en utilisant les paramètres
- *return \$perimetre* instruction qui indique qu'il faut renvoyer le résultat de la fonction
- *\$perimetre = perimetre_rectangle(1.5,2)* vu que la fonction renvoie une valeur il faut mettre cette valeur dans une variable (ici *\$perimetre*). Vous remarquez que pour les nombres décimaux on met un point et non une virgule en PHP
- *echo* on indique la phrase à afficher en utilisant la variable *\$perimetre*

Sachez qu'il existe des centaines de fonctions prêtes à être utilisées. Voir Partie 3, II. Les fonctions intrinsèques.



V. Les conditions

Elles permettent de donner des ordres différents à PHP en fonction du cas par exemple égal, supérieur, inférieur... Voici le tableau correspondant à ces différents cas :

Symbole	Signification
&&	Et
(alt gr+ 6)	ou
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

A. Forme classique

Prenons un exemple en langage courant :

SI tu es sage (1^{ère} possibilité)

ALORS tu auras des bonbons

SINON tu n'auras pas de bonbons (2^{ème} possibilité)

Voilà ce que cela donne en langage PHP :

```
<?php
$sage = "oui";
if ($sage == "oui")
{
echo "C'est bien mon petit !<br>";
$bonbons = "oui";
}
else
{
echo "Ce n'est pas bien mon
petit !<br>";
$bonbons = "non";
}
echo "La décision du père Noël pour
donner des bonbons : $bonbons";
?>
```

Ce qui apparaîtra à l'écran :
C'est bien mon petit !

La décision du père Noël pour
donner des bonbons : oui

- *If* ($\$sage == \text{« oui »}$) {...} Si la variable $\$sage$ est égale à oui alors on fait ce qu'il y a entre accolades (c'est la condition)
- *Else* {...} Sinon (donc que $\$sage$ n'est pas égal à oui) alors on fait ce qu'il y a entre accolades

Compliquons un peu l'exemple, en langage courant :

SI tu es sage et que tes notes sont supérieures à 10 (1^{ère} possibilité)

ALORS tu auras beaucoup de bonbons

SINON SI tu n'es pas sage et que tes notes sont supérieures à 10 (2^{ème} possibilité)

ALORS tu auras un peu de bonbons

SINON (dans tous les autres cas) tu n'auras pas de bonbons (3^{ème} possibilité)

Voilà ce que cela donne en langage PHP :

```
<?php
$sage = "non";
$notes = 12;
if ($sage == "oui" && $notes >=10 )
{
echo "C'est bien mon petit !<br>";
```

Ce qui apparaîtra à l'écran :
Soit plus sage !

La décision du père Noël pour
donner des bonbons : oui un peu

```

$bonbons = "oui beaucoup";
}
elseif ($sage == "non" && $notes >=10 )
{
echo "Soit plus sage !<br>";
$bonbons = "oui un peu";
}
else
{
echo "Tu n\" as pas assez travailler à
l\"école<br>";
$bonbons = "non";
}
echo "La décision du père noel pour
donner des bonbons : $bonbons";
?>

```

- *If (\$sage == « oui » && \$notes>=10) {...}* Si la variable *\$sage* est égale à oui et si la variable *\$notes* est supérieure ou égale à 10 alors on fait ce qu'il y a entre accolades
- *Elseif (sage== « non » && \$notes>=10) {...}* Sinon si (donc on propose une autre condition) la variable *\$sage* est égale à non et si la variable *\$notes* est supérieure ou égal à 10 alors on fait ce qu'il y a entre accolades
- *Else* Sinon (donc dans tous les autres cas de figure) la variable *\$bonbons* prend la valeur non

B. Le Switch

Pour en finir avec les conditions voyons une dernière forme conditionnelle :

```

<?php
$note = 3;
switch ($note)
{
case 0:
echo "Nul !";
break;
case 1:
echo "Mauvais !";
break;
case 2:
echo "Passable !";
break;
case 3:
echo "Moyen !";
break;
case 4:
echo "Pas mal !";
break;
case 5:
echo "Très bien !";
break;
default:
echo "Note inconnue";
}
?>

```

Ce qui apparaîtra à l'écran :
Moyen !

- *Switch (\$note){...}* fonction conditionnelle, on indique entre parenthèses la variable avec laquelle on travaille (ici *\$note*)
- *Case 0:* pour le cas *\$note* égale à 0 on indique ce qu'il faut faire (ici on affiche *Nul !* par l'intermédiaire d'un *echo*). Vous noterez qu'avec cette fonction on ne peut faire qu'égal à
- *Case 1:* pour le cas *\$note* égale à 1 on indique ce qu'il faut faire (ici on affiche *Mauvais !* par l'intermédiaire d'un *echo*) et ainsi de suite pour tous les cas...
- *Break;* doit être placé à la fin de chaque cas, il permet de faire sortir PHP du switch
- *Default :* cela indique le cas pour toute autre possibilité



VI. Les boucles

Une boucle permet de répéter plusieurs fois une instruction. Pour ne pas que les boucles se répètent à l'infini il faut indiquer une condition. Tant que la condition est remplie les instructions seront répétées, dès qu'elle n'est plus remplie, on sort de la boucle.

A. Le WHILE

Voici un exemple dans le langage courant :

ARGENT = 3 euro

TANT QUE tu n'as pas 5 euro

ALORS tu n'as pas de bonbons

ARGENT = ARGENT + 1 euro

Fin des instructions, on recommence tant que la condition n'est pas remplie, c'est une boucle !

Voici un autre exemple correspondant en PHP :

```
<?php
$compteur = 1;
while ($compteur <= 5)
{
echo "Ligne de test<br>";
$compteur = $compteur + 1;
}
?>
```

Ce qui apparaîtra à l'écran :

```
Ligne de test
Ligne de test
Ligne de test
Ligne de test
Ligne de test
```

- $\$compteur = 1$ on donne une valeur de 1 à une variable (c'est l'initialisation)
- $while (\$compteur <= 5) \{...\}$ Tant que la variable $\$compteur$ est inférieure ou égale à 5 (c'est la condition) il faut faire ce qu'il y a entre accolades
- $\$compteur=\$compteur+1$ on ajoute 1 à la valeur de la variable $\$compteur$ c'est ce qu'on appelle l'incrément. Ainsi la condition sera remplie 5 fois. ($0+1=1,1+1=2, 2+1=3,3+1=4,4+1=5$)

B. Le FOR

Il existe une autre forme pour faire des boucles, le *for* :

```
<?php
for ($compteur = 1; $compteur <= 5;
$compteur++)
{
echo "Ligne de test n°$compteur<br>";
}
?>
```

Ce qui apparaîtra à l'écran :

```
Ligne de test n°1
Ligne de test n°2
Ligne de test n°3
Ligne de test n°4
Ligne de test n°5
```

- $for (\$compteur = 1; \$compteur <= 5; \$compteur++)$ trois éléments sont séparés par des points virgules :
 - $\$compteur = 1$ initialisation du *compteur* à 1
 - $\$compteur <= 5$ conditions, tant qu'elle est remplie, elle sera re-exécutée
 - $\$compteur++$ incrément qui ajoute à $\$compteur$ la valeur 1
- on place entre accolades ce qu'il faut répéter



VII. Les tableaux

Les tableaux sont générés à partir de variables *array*, qui à la différence des variables que l'on a déjà étudié, vont prendre plusieurs valeurs par l'intermédiaire de cases. Il faudra, pour afficher la variable, préciser sa case. Voici un exemple :

```
<?php
$animaux = array ("Chien", "Chat",
"Poisson");
echo "J'ai un $animaux[1]";
?>
```

Ce qui apparaîtra à l'écran :
J'ai un Chat

- *\$animaux = array ("Chien", "Chat", "Poisson");* on dit que la variable *\$animaux* à plusieurs cases en tapant *array*, puis on énumère les cases entre parenthèses séparées par des virgules (la première case vaut 0, la seconde 1...)
- *\$animaux [1]* pour afficher la variable *\$animaux* on spécifie entre crochets le numéro de la case que l'on souhaite afficher (ici 1)

On peut aussi changer le nom des cases (tableaux associatifs) :

```
<?php
$adresse = array (
    "Prénom" => "Léni",
    "Nom" => "CHON",
    "Adresse" => "3, rue barbe",
    "Ville" => "Nancy");
echo "Bonjour $adresse['Prénom'] !";
?>
```

Ce qui apparaîtra à l'écran :
Bonjour Léni !

- *\$adresse = array (...)* ; comme auparavant, on indique que la variable *\$adresse* à plusieurs cases en tapant *array*
- *"Prénom" => "Léni"*, on énumère dans les parenthèses du *array* les cases avec le nom que l'on veut leur donner . Pour cela on commence par dire le nom de la case *"Prénom"* puis on lui indique sa valeur *"Léni"* par l'intermédiaire de *=>*
- *\$adresse['Prénom']* pour afficher la variable *\$adresse* on spécifie entre crochets le nom de la case que l'on souhaite afficher (ici Prénom)



PARTIE 2

BASE DE DONNEES



I. Introduction

Une base de donnée est un système qui enregistre des informations en les classant. Pour gérer les bases de données on va utiliser un autre langage de programmation, le SQL qu'il faudra écrire par l'intermédiaire de PHP. Une base de données se structure d'une ou plusieurs tables. Dans chacune de ces tables se trouvent des champs, qui correspondent aux colonnes, et des entrées qui correspondent aux lignes. Pour mieux comprendre le système voici un exemple :

Base de données : NOTES

⇒Table : FRANCAIS

Numéro	Nom	Prénom	Note
1	SOURIS	Toto	18
2	ELEPHANT	Titi	16
3	RENARD	Tutu	15
...

⇒Table : MATHS

Numéro	Nom	Prénom	Note
1	SOURIS	Toto	18
2	ELEPHANT	Titi	16
...

- La base de données *NOTES* est composée de deux tables : *FRANÇAIS* et *MATHS*
- La table *FRANÇAIS* a quatre champs : *Numéro*, *Nom*, *Prénom* et *note*
- Une entrée de la table *MATHS* est : *1*, *SOURIS*, *Toto*, *18*

Pour faire des modifications dans la base de données, nous allons utiliser le système PHPMYADMIN, qui va vous permettre de ne pas tout programmer en SQL, ce qui serait trop long. PHPMYADMIN est fourni par la majorité des hébergeurs, et est accessible par une adresse spécifique à l'hébergeur (par exemple sur free.fr c'est sql.free.fr), vous devrez indiquer votre mot de passe et login (fourni par votre hébergeur) pour vous connecter. Pour ouvrir PHPMYADMIN avec Easy PHP afin de tester le système en local (directement sur votre disque dur), cliquez sur la page d'administration de Easy PHP puis sur Gestion BDD.



II. Php My Admin

A. Créer la base

Pour créer une base de données dans PHPMYADMIN, indiquez à droite le nom de la base de données et cliquez sur Créer. Vous n'aurez généralement pas cette possibilité une fois sur l'hébergeur, mais il est bien de savoir le faire. Il ne faut pas mettre d'accents ni d'espace dans le nom.

↳ Créer une base de données [\[Documentation\]](#)

B. Créer la table

Cliquez sur une de vos bases de données dans le menu déroulant à gauche pour voir les tables qui la composent, pour le moment il n'y en a pas. Indiquez alors à droite le nom de la table ainsi que le nombre de champ qu'elle contient. Il ne faut pas mettre d'accents ni d'espace dans le nom.

- Créer une nouvelle table sur la base
 Nom :
 Champs :

Il faut ensuite indiquer le nom des champs ainsi que le type de données qu'il contiendra. Voici les différents types de données :

- INT : nombre entier
- TEXT : texte
- DATE : date
- TIME : heure
- DATETIME : date et l'heure
- BLOB : permet de stocker des fichiers dans la base de données

Champ	Type [Documentation]	Taille/Valeurs*	Extra	Primaire	Index	Unique	...	Texte entier
id	MEDIUMINT		auto_increment	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="checkbox"/>
titre	TEXT			<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="checkbox"/>
contenu	TEXT			<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="checkbox"/>



Voici un exemple dans lequel on indique 3 champs avec pour nom : *id*, *titre* et *contenu* avec pour type respectif *nombre entier (MEDIUM MINT)*, *texte (TEXT)* et *texte (TEXT)*. Pour des types de champs comme *id*, où la valeur va être unique pour chaque entrée, il faut que la valeur augmente de 1 après chaque entrée. Pour cela sélectionnez *auto_increment* dans Extra et cochez *Index*.



- *test(1)* indique que l'on est dans la base de données *test* qui contient 1 table (entre parenthèses)
- *News* indique la présence de la table *news* dans la base de données *test*. En cliquant sur *news* on accède à la structure de la table. En cliquant sur l'image de tableau à côté de *news* on accède au contenu de la table.

C. Modifier une table

- Pour insérer des entrées dans une table, sélectionnez la table et cliquez sur Insérer ([Insérer](#)). Entrez les données, ne donnez pas de valeur si vous êtes dans un auto_increment, mysql la prendra automatiquement.

- Pour afficher les entrées d'une table, sélectionnez la table et cliquez sur Afficher (**Afficher**). Cliquez sur  pour éditer une donnée et sur  pour la supprimer. Sachez que vous pouvez déterminer le nombre de ligne à afficher via le formulaire se situant en bas de la page.

D. SQL

Grâce au lien SQL (**SQL**) vous pouvez exécuter des requêtes pour demander a Mysql d'accomplir une tâche. Nous verrons plus tard le langage SQL.

E. Exporter

Grâce au lien Exporter (**Exporter**) vous pouvez enregistrer votre base sur votre disque dur au format texte. C'est en fait un fichier SQL qui contient les instructions pour recréer votre base de données (sans les données, ce n'est que la structure). Cela peut vous permettre de faire une sauvegarde par exemple. Laissez les options par défaut puis cliquez sur Executer : vous devez télécharger le fichier. Notez que vous pouvez utiliser le contenu de ce fichier en tant que requête SQL pour recréer ou restaurer votre base de données.

F. Vider

Pour vider l'ensemble des entrées d'une table, sélectionnez la table et cliquez sur Vider (**Vider**). Seule la structure de la table sera conservée.

G. Supprimer

Pour supprimer une table, sélectionnez la table et cliquez sur Supprimer (**Supprimer**). La structure et les données de la table seront perdues.

H. Autres opérations

Il y a d'autres opérations possibles grâce au lien Opérations (**Opérations**) :

- Changer le nom de la table : pour changer le nom d'une table
- Déplacer la table vers : pour mettre la table sur une autre base de données
- Copier la table : pour copier la table sur la même base ou une autre
- Réparer la table : pour réparer la table en cas de problème(s)
- Optimiser la table : pour réorganiser la table



III. Lire des données

A. Connexion

Pour lire des données à partir d'une base de données avec du PHP, vous devez tout d'abord vous connecter voici le code :

```
<?php
mysql_connect("localhost", "login", "motdepasse");
mysql_select_db("nombase");
// On tape ici les requetes
mysql_close();
?>
```

- *mysql_connect* fonction qui permet de se connecter au serveur, on indique entre parenthèses, le nom de l'hôte (c'est généralement localhost), le login (nom d'utilisateur) et le mot de passe. Le mot de passe et login sont fournis par l'hébergeur ou si vous vous connectez en local c'est "root"
- *mysql_select_db* fonction qui permet de spécifier la base de données sur laquelle on travaille, on indique entre parenthèses le nom de la base (ici elle s'appelle *nombase*)
- *mysql_close()* enfin après avoir travaillé sur la base on n'oublie pas de se déconnecter

B. SQL

Pour demander à afficher des informations il faut programmer en SQL des requêtes. Voici comment écrire une requête SQL après s'être connecté à la base :

```
<?php
$resultat = mysql_query("...SQL...") ;
?>
```

- *\$resultat* renvoie la valeur de la requête qui suit après le =
- *mysql_query(...)* fonction qui indique que l'on fait une requête, on place le code SQL entre parenthèses

Comment programmer du code SQL ? Ca marche de cette manière :

```
<?php
$resultat = mysql_query("SELECT prenom FROM adresse") ;
?>
```

- *SELECT prenom* indique les champs que l'on souhaite voir, ici prénom, on met * si l'on souhaite voir tous les champs
- *FROM adresse* indique dans quelle(s) table(s) se trouve le ou les champs, ici c'est dans la table adresse

On peut compliquer en ajoutant des contraintes :

```
<?php
$resultat = mysql_query("SELECT prenom FROM adresse WHERE ville='Nancy'
AND sexe='Femme'") ;
?>
```

- *WHERE* oblige à retourner des résultats dans lesquels le champ *ville* est égal à *Nancy*
- *AND sexe='Femme'* si l'on veut ajouter une autre contrainte au *WHERE* on met *AND*. Dans l'exemple il faut que les résultats aient un champ *sexe* égal à *Femme*. Vous pouvez aussi utiliser *OR* pour ajouter une possibilité (soit ça ou soit ça...)

On peut aussi classer les résultats :

```
<?php
$resultat = mysql_query("SELECT prenom FROM adresse ORDER BY nom") ;
?>
```

- *ORDER BY nom* classe les résultats en fonction de la valeur du champ *nom*. Les résultats seront donc classés par ordre alphabétique en fonction des noms. Si on avait voulu classer dans l'ordre inverse, il aurait fallu taper : *ORDER BY nom DESC*

On peut déterminer une limite :

```
<?php
$resultat = mysql_query("SELECT prenom FROM adresse LIMIT 0,20") ;
?>
```

- *LIMIT 0,20* ne prend qu'une partie des résultats : du premier au 20^{ème} !

Vous devez obligatoirement mettre les mots clefs dans l'ordre suivant :

SELECT / FROM / WHERE / ORDER BY / LIMIT

Le problème de la variable *\$resultat* c'est qu'elle ne renvoie pas à une valeur exploitable, il faut utiliser la fonction *mysql_result* pour pouvoir l'exploiter :

```
<?php
mysql_connect("localhost", "root", "root");
mysql_select_db("base1");
$resultat = mysql_query("SELECT nom, prenom FROM adresse");

$nom = mysql_result($resultat, 0, "nom");
$prenom = mysql_result($resultat, 0, "prenom");

echo "$nom $prenom";

mysql_close();
?>
```

- *\$nom = mysql_result(\$resultat, 0, "nom");* on donne à la variable *\$nom* la valeur de l'enregistrement *0* de la requête SQL obtenue grâce à *\$resultat* et dont le champ est *nom*

Voyons maintenant pour afficher l'ensemble des résultats :

```
<?php
mysql_connect("localhost", "root", "root");
mysql_select_db("base1");
$resultat = mysql_query("SELECT nom, prenom FROM adresse");

$num = mysql_num_rows($resultat);

for ($i=0;$i<=$num;$i++)
{
    $nom = mysql_result($resultat, $i, "nom");
    $prenom = mysql_result($resultat, $i, "prenom");
    echo "$nom $prenom<br>";
}

mysql_close();
?>
```

- *\$num = mysql_num_rows(\$resultat);* on compte le nombre d'enregistrements en réponse à la requête *\$resultat* et on renvoi le résultat dans la variable *\$num*
- *for (\$i=0;\$i<=\$num;\$i++)* on boucle de *0* à *\$num* (qui correspond au nombre de réponses à la requête) pour s'occuper de chaque élément un à un.
- *\$nom = mysql_result(\$resultat, \$i, "nom");* on donne à la variable *\$nom* la valeur de l'enregistrement *\$i* (qui correspond à un des enregistrements) de la requête SQL obtenue grâce à *\$resultat* et dont le champ est *nom*

Pour finir, regardons un dernier mot clef en SQL permettant de compter le nombre de résultats :

```
<?php
mysql_connect("localhost", "root", "root");
mysql_select_db("base1");
$resultat = mysql_query("SELECT COUNT(*) AS nombre FROM adresse");
$nom = mysql_result($resultat, 0, "nombre");
mysql_close();
?>
```

- *SELECT COUNT(*) AS nombre FROM adresse* compte toutes les entrées de la table *adresse* et renvoi le nombre sous le nom *nombre* (par l'intermédiaire de *AS*).



IV. Ecrire des données

Nous savons à présent comment lire les données d'une base grâce au PHP, regardons à présent comment ajouter, modifier des données sans passer par PHPMYADMIN. Pour modifier des informations il faut encore une fois programmer en SQL des requêtes.

Voyons pour commencer comment ajouter une entrée dans une table :

```
<?php
mysql_query("INSERT INTO adresse VALUES('HONNAITE', 'Camilleux', '25 rue
Matisme', 'Nancy')");
?>
```

- *INSERT INTO adresse VALUES(...)* on indique qu'il va falloir ajouter une entrée dans la table *adresse*
- *VALUES('HONNETE', 'Camille', '25 rue Matisme', 'Nancy')*; on indique la valeur pour les différents champs de la base. Le premier champ (nom) prendra la valeur *HONNETE*, le deuxième (prénom) prendra la valeur *Camille* et ainsi de suite...

Pour modifier une entrée dans une table :

```
<?php
mysql_query("UPDATE adresse SET prenom='Camille', ville='Paris' WHERE
nom='HONNAITE'");
?>
```

- *UPDATE adresse* on indique que l'on va modifier la table *adresse*
- *SET prenom='Camille', ville='Paris'* on indique ce que l'on remplace, là par exemple, le champ *prenom* va supprimer sa valeur et la modifier par *Camille* et le champ *ville* par *Paris*
- *WHERE='HONNAITE'* information importante car on spécifie sur quelle entrée on travaille, dans l'exemple c'est dans l'entrée où le *nom* est égal à *HONNAITE* que le *prénom* et la *ville* vont être remplacés

Pour supprimer une entrée dans une table :

```
<?php
mysql_query("DELETE FROM adresse WHERE nom='HONNAITE'");
?>
```

- *DELETE FROM adresse* signifie que l'on va supprimer quelque chose dans la table *adresse*
- *WHERE nom='HONNAITE'* on indique quelles entrées doivent être supprimées, ici toutes celles dont le nom est égal à *HONNAITE*



PARTIE 3

CODAGE AMELIORE



I. Les Includes

La fonction *include* permet d'inclure une page PHP dans une autre page PHP. Voici un exemple bien pratique, le design d'un site. Le site est fait grâce à un tableau HTML, pour un soucis de simplification, il faudrait utiliser du CSS, mais ce n'est pas l'objet de ce cours.

```
<html>
<head>
<title>Nouvelle page</title>
</head>
<body>
<table border=1 width="90%">
<tr>
<td colspan="2">Haut.php</td>
<tr>
<td width="10%">Menu.php</td>
<td width="90%">Bienvenue sur mon site</td>
</tr>
</table>
</body>
</html>
```

Ce qui apparaîtra à l'écran :

haut.php	
menu.php	Bienvenue sur mon site

Si le contenu du menu ou de la partie du haut vient à évoluer, en HTML, il faut rééditer toutes les pages du site, place aux *includes* ! On va décomposer les parties qui reviennent :

Page menu.php :

```
<?php
echo"<td width='10%'>Lien 1<br>Lien 2..</td>";
?>
```

Page haut.php :

```
<?php
echo"<td colspan="2">Bannière du haut de mon site</td>";
?>
```

On place le tout sur une nouvelle page :

```
<?php
echo"
<html>
<head>
<title>Nouvelle page</title>
</head>
<body>
<table border=1 width='90%'>
<tr>";
include ("haut.php");
echo"</tr>
<tr>";
include ("menu.php");
echo"
<td width="90%">Bienvenue sur mon site</td>
</tr>
</table>
</body>
</html>";
?>
```

Ce qui apparaîtra à l'écran :

haut.php	
menu.php	Bienvenue sur mon site

- `include("haut.php")` ; on remplace dans le code du départ les zones que l'on a décomposées



II. Les fonctions intrinsèques

Voici une liste de fonctions très pratiques déjà programmées en PHP :

➤ Addslashes

```
<?php
$avec_antislash = addslashes($sans_antislash);
?>
```

- On ajoute automatiquement des antislash devant les apostrophes et les guillemets sur la chaîne de caractères (texte) contenue dans la variable *\$sans_antislash*

➤ Stripslashes

```
<?php
$sans_antislash = stripslashes($avec_antislash);
?>
```

- On enlève automatiquement les antislash devant les apostrophes et les guillemets sur la chaîne de caractères contenue dans la variable *\$avec_antislash*

➤ Htmleentities

```
<?php
$html = '<b>Bonjour</b>';
$sans_html = htmleentities($html);
echo $sans_html;
?>
```

Ce qui apparaîtra à l'écran :
Bonjour

- On enlève automatiquement le code HTML sur la chaîne de caractères contenue dans la variable *\$html*

➤ Nl2br

```
<?php
$texte = 'Ligne numéro 1
Ligne numéro 2
Ligne numéro 3
Ligne numéro 4';
$retour_ligne = nl2br($texte);
echo $retour_ligne;
?>
```

Ce qui apparaîtra à l'écran :
Ligne numéro 1
Ligne numéro 2
Ligne numéro 3
Ligne numéro 4

- On revient à la ligne à chaque fois que, sur la chaîne de caractères contenue dans la variable *\$texte* vous êtes allés à la ligne dans le code PHP

➤ Strlen

```
<?php
$texte = 'Bonjour';
$longueur = strlen($texte);
echo 'Le texte compte ' . $longueur . '
caractères';
?>
```

Ce qui apparaîtra à l'écran :
Le texte compte 7 caractères

- On renvoie le nombre de caractères de la variable *\$texte* dans la variable *\$longueur*

➤ Str_replace

```
<?php
$remplace = str_replace('p', 'f', 'Des
pions');
echo $remplace;
?>
```

Ce qui apparaîtra à l'écran :
Des fions

- On indique entre parenthèses après *str_replace* la lettre, ou le mot que l'on veut rechercher (ici *p*) puis après une virgule par quoi on doit le remplacer (ici *f*) et enfin après une dernière virgule la chaîne de caractères sur laquelle on fait la recherche (ici *Des pions*)

➤ Str_shuffle

```
<?php
$texte = 'Bonjour';
$texte = str_shuffle($texte);
echo $texte;
?>
```

Ce qui apparaîtra à l'écran :
OnjBrou

- On mélange aléatoirement les caractères de la chaîne contenue dans la variable *\$texte*

➤ Strtolower

```
<?php
$texte= 'BONJOUR';
$texte = strtolower($texte);
echo $texte;
?>
```

Ce qui apparaîtra à l'écran :
bonjour

- On met en minuscules la chaîne contenue dans la variable *\$texte*.

Il existe la fonction inverse qui permet de passer des minuscules aux majuscules : `strtoupper`

➤ Mail

```
<?php
mail ("destinataire@domaine.com", "Mon sujet",
"Coucou c'est moi !", "FROM:
expediteur@domaine.com ")
?>
```

Ce qui apparaîtra à l'écran :

- Envoi un mail à *destinataire@domaine.com* avec comme sujet *Mon sujet*, comme message *Coucou c'est moi !* et avec comme adresse d'expéditeur *expediteur@domaine.com*

➤ Rand

```
<?php
$nombre = rand(0,100) ;
?>
```

Ce qui apparaîtra à l'écran :
bonjour

- Permet d'affecter à la variable *\$nombre* un chiffre aléatoire compris entre 0 et 100.

➤ Explode

```
<?php
$date = 1986-12-01
list($an, $mois, $jour) = explode('-',
$date);
$date = $jour."-".$mois."-".$an;
?>
```

Ce qui apparaîtra à l'écran :
bonjour

- La fonction *explode()* va disséquer la variable *\$date* à chaque fois qu'il rencontrera un -
- L'ensemble des éléments seront nommés grâce à la fonction *list()* par *\$an* puis *\$mois* et enfin *\$jour*. On a donc créé 3 variables à partir de la variable *\$date*.



III. Les formulaires

Vous savez comment faire des formulaires en HTML, voyons maintenant comment exploiter les données de ces derniers.

Imaginons une page avec ce formulaire :

```
<form method="post" action="page.php">
<input type="text" name="zonetexte1" />
</form>
```

Imaginons une autre page nommée page.php :

```
<?php
echo $_POST[zonetexte1];
?>
```

- En fait les informations entrées par l'utilisateur seront disponibles dans *page.php* (défini dans l'attribut *action* de la balise *form*) grâce à la variable *\$_POST[...]*. On placera entre crochets le nom de la zone de formulaire ici *zonetexte1* (généralement nommé dans une balise de formulaire par l'attribut *name*).
- Dans le cas d'une liste déroulante, la variable *\$_POST* prendra la valeur d'un des attributs *value*.
- Dans le cas de cases à cocher, la variable *\$_POST* aura une valeur égale à « *on* » si la case est cochée ou ne contiendra rien si elle ne l'est pas.
- Dans le cas de boutons d'option, la variable *\$_POST* prendra la valeur d'un des attributs *value*.



IV. Le temps

Voici des fonctions qui permettent de se repérer dans le temps.

A. Date

```
<?php
echo date('d/m/Y');
?>
```

Ce qui apparaîtra à l'écran :
JJ/MM/AAAA

- La fonction `date()` permet d'afficher les informations du calendrier

Voici les valeurs principales que l'on peut mettre entre parenthèses :

Lettre	Signification	Valeurs possibles
s	Secondes	00 à 59
i	Minutes	00 à 59
H	Heures	00 à 23
l	Indique si l'heure d'été est activée (1 = oui, 0 = non)	0 ou 1
O	Différence d'heures avec l'heure GMT (Greenwich)	-1200 à +1200
d	Jour du mois	01 à 31
m	Mois de l'année	01 à 12
Y	Année, sur 4 chiffres	Beaucoup de possibilités
y	Année, sur 2 chiffres	Beaucoup de possibilités
L	Indique si l'année est bissextile (1 = oui, 0 = non)	0 ou 1
l	Jour de la semaine écrit en anglais	Sunday à Saturday
F	Mois écrit en anglais	January à December
t	Nombre de jours dans le mois	28 à 31
w	Numéro du jour de la semaine	0 (dimanche) à 6 (samedi)
W	Numéro de la semaine dans l'année	01 à 52
z	Numéro du jour de l'année	0 à 364

Pour mettre en français la fonction `date`, il faudra l'adapter grâce à une fonction que vous aurez créée.

B. Mktime

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

Ce qui apparaîtra à l'écran :
1090213402 (*par exemple*)

- La fonction `time()` renvoie un nombre qui correspond au nombre de secondes écoulées depuis le 1^{er} janvier 1970 (date de création d'Unix)

C. Du Mktime à Date et de Date à Mktime

Le `mktime` va nous être très pratique pour faire des comparaisons ou opérations sur les dates (soustractions de jours ou minutes par exemple).

```
<?php
$time=time()-300;
$date = date ('d/m/Y', $time);
echo $date;
?>
```

Ce qui apparaîtra à l'écran :
JJ/MM/AAAA

- On retire 300 à la fonction `time()`, ce qui correspond à 5 minutes (5 x 60 secondes)
- Pour passer du format `mktime` à `date` on ajoute en paramètre le `mktime` dans la fonction `date()`

Et voici l'inverse :

```
<?php
$time = mktime($heure, $minutes, $secondes,
               $mois, $jour, $an);
?>
```

Ce qui apparaîtra à l'écran :
1090213402 (*par exemple*)

- Pour passer du format `date` à `mktime` on passe en paramètre dans la fonction `mktime()` dans l'ordre : l'heure, le nombre de minutes, le nombre de secondes, le mois, le jour et l'année.



V. Les superglobales

Une variable superglobale commence par le signe underscore (`_`) et est écrite en majuscules. Ce sont des variables qui sont des array (variable sous forme de tableaux) :

- `$_SERVER['PHP_SELF']` : c'est le chemin de la page que vous êtes en train d'exécuter, par rapport à la racine de votre site web. Exemple : si vous êtes sur la page <http://www.site.com/articles/index.php>, alors `$_SERVER['PHP_SELF']` aura pour valeur : `/articles/index.php`
- `$_SERVER['HTTP_REFERER']` : c'est l'url de la page qui a amené le visiteur sur la page courante. Cela peut être utile notamment pour faire des statistiques : vous saurez par exemple que le site "*partenaire.com*" a fait un lien vers votre site et vous amène des visiteurs
- `$_SERVER['REMOTE_ADDR']` : information intéressante qui nous donne l'adresse IP du client qui a demandé à voir la page.
- `$_GET` : elle vous donne les valeurs des informations indiquées dans l'url. Par exemple, si la page appelée est : <http://www.site.com/page.php?nom=cover&prenom=harry&age=51>.
 - `$_GET['nom'] = "cover"`
 - `$_GET['prenom'] = "harry"`
 - `$_GET['age'] = "51"`
- `$_POST` : permet de récupérer les informations issues d'un formulaire.
- `$_FILES` : permet l'envoi de fichiers sur le serveur à partir d'un formulaire.
- `$_SESSION` : une superglobale un peu plus complexe, regardons comment elle fonctionne avec un exemple :

```
<?php
```

```
session_start();  
$_SESSION['login'] = 'Charlie';  
echo $_SESSION['login'];  
?>
```

Ce qui apparaîtra à l'écran :
Charlie

- `session_start()`; permet de créer une session pour l'utilisateur afin qu'il retienne les variables de session sur toutes les pages !
- `$_SESSION['login'] = 'Charlie'`; on indique que la variable de session `login` a une valeur de `Charlie`
- `echo $_SESSION['login'];` permet d'afficher la variable `login`

Sachez que vous pouvez afficher sur toutes vos pages `echo $_SESSION['variable'];` et non pas que sur la page du code : PHP retient grâce à la session ; il faudra juste spécifier que la session est toujours ouverte en indiquant en haut du code : `session_start()` ;

Mais comment fermer une session ? C'est la fonction `session_destroy()` .

- `$_COOKIES` : une autre superglobale qui permet d'enregistrer sur l'ordinateur du visiteur un petit fichier texte qui contient des informations à propos du visiteur (par exemple son pseudo de manière à ce que la prochaine fois qu'il se connecte à votre site il soit automatiquement reconnu)

```
<?php
```

```
$timeend = time() + 365*24*3600;  
setcookie('pseudo', 'charlie', $timeend);  
setcookie('pass', 'curieu', $timeend);  
echo $_cookie['pseudo'];  
?>
```

Ce qui apparaîtra à l'écran :
charlie

- `$timeend = time() + 365*24*3600`; dans une simple variable on indique un temps d'une année (en seconde)
- `setcookie('pseudo', 'charlie', $timeend)`; cette fonction permet de créer un cookie, on indique dans le 1^{ère} paramètre le nom du cookie ici `pseudo`, dans le 2^{ème} la valeur du cookie ici `charlie` et dans le dernier la date d'expiration du cookie dans l'exemple déterminé par la variable `$timeend`
- `setcookie('pass', 'curieu', $timeend)`; on crée un deuxième cookie `pass`
- `echo $_cookie['pseudo'];` on demande d'afficher l'information contenue sous le cookie `pseudo`



VI. Les Regex

Afin de tester si une chaîne de caractère comporte ou non un caractère, qu'il ne comporte que des chiffres, que des lettres, qu'un type de caractère on utilise les Regex !

Rien de mieux qu'un exemple pour comprendre le principe :

```
<?php
if (preg_match("!compliqué!", "Oulala, ça devient compliqué !"))
{ echo 'VRAI'; }
?>
```

Ce qui apparaîtra à l'écran :
VRAI

- On indique grâce à la fonction `preg_match()` de chercher le terme entre doubles quotes et deux points d'exclamation (appelé regex) (ici *compliqué*) sur la chaîne de caractères *Oulala, ça devient compliqué !*
- La fonction trouve le regex dans la chaîne, elle renvoie donc la valeur *true* et la conditionnelle est remplie.

➤ Minuscules et majuscules :

Le regex prend en considération les minuscules et majuscules, on ajoute un *i* à la fin du second point d'exclamation pour qu'il n'en tienne pas compte : *!terme!i*.

➤ Ou :

Pour chercher la présence soit d'un *terme1* ou soit d'un *terme2* on écrit : *!terme1|terme2!* (Pour obtenir la barre verticale : Alt Gr + 6).

➤ Commence par :

Pour savoir si la chaîne commence par *début* on écrit : *!^debut !*.

➤ Finit par :

Pour savoir si la chaîne finit par *fin* on écrit : *!fin\$!*.

➤ Contient le caractère :

Pour savoir si la chaîne contient les caractères *v* et *i* et *p* on écrit : *!vip!*.

➤ Classe de caractère :

Pour savoir si la chaîne contient des chiffres on écrit : *![1-9]!*.

Pour savoir si la chaîne contient des lettres minuscules on écrit : *![a-z]!*.

Pour savoir si la chaîne contient des lettres majuscules on écrit : *![A-Z]!*.

Pour savoir si la chaîne contient des lettres minuscules de *a* à *e* on écrit : *![a-e]!*.

Pour savoir si la chaîne ne contient pas de lettres minuscules on écrit : *![^a-e]!*.

➤ Nombre de fois :

Pour savoir si la chaîne comporte 3 fois la lettre *a* on écrit : *![a{3}]!*.

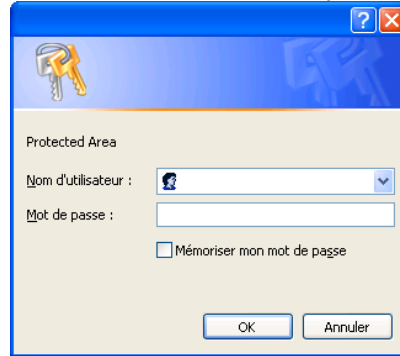
Pour savoir si la chaîne comporte de 3 à 5 fois la lettre *a* on écrit : *![a{3-5}]!*.

Dans le cas où vous faites une recherche sur un caractère spécial : *! ^ \$ () [] { } ? + * . * vous devrez mettre antislash (**) juste avant ce dernier.



VII. Les Htaccess

Un htaccess va permettre de protéger l'accès à un répertoire en demandant à l'utilisateur un mot de passe. Attention, ce n'est pas du PHP mais des instructions qu'on donne au serveur.



➤ Htaccess

On commence par créer un fichier *htaccess.txt* sous bloc note, dans lequel on indique ce code :

```
AuthName "Page d'administration protégée"  
AuthType Basic  
AuthUserFile "/home/sdz/www/gestion/admin/.htpasswd"  
Require valid-user
```

- On indique le nom de la page protégée dans *AuthName*
- On indique le chemin absolu vers le répertoire protégé dans *AuthUserFile* (ici */home/sdz/www/gestion/admin/.htpasswd*)

Comment avoir le chemin absolu ? On crée une page PHP temporaire dans le répertoire que l'on souhaite protéger et on tape :

```
<?php echo realpath('page.php'); ?>
```

- On indique en paramètre à la fonction *realpath()* le nom de la page temporaire PHP que nous venons de créer (donc dans le répertoire protégé) (ici *page.php*)

Cette fonction renvoie un lien de ce type : */home/sdz/www/gestion/admin/page.php* qu'on renommera par */home/sdz/www/gestion/admin/.htpasswd*.

➤ Htpasswd

On commence par créer un fichier *htpasswd.txt* sous *Bloc notes*, dans lequel on indique les différents utilisateurs ainsi que leur mot de passe en crypté. Par exemple, voici un fichier pour 2 utilisateurs, *charlie* et *admin* :

```
charlie:$1$EQTRY1av$dTmXwFztI9wnTbVigjLgD/  
admin:$1$4yQiHzv2$1Hjqn4UTdVX3v17E3qNso/
```

- On indique le nom de l'utilisateur suivi de deux points et tape son mot de passe crypté

Comment avoir le mot de passe en crypté ? On crée une page PHP temporaire et on tape :

```
<?php echo crypt('monmotdepasse'); ?>
```

- On indique en paramètre à la fonction *crypt()* le mot de passe que l'on souhaite crypté (ici *monmotdepasse*)

Cette fonction renvoie le paramètre en crypté.

Il ne reste plus qu'à envoyer les fichiers *htaccess.txt* et *htpasswd.txt* dans le répertoire à protéger. Puis on les renomme en *.htaccess* et *.htpasswd* (il y a bien un *.* devant et plus d'extension : il s'agit de fichiers cachés).



VIII. Lire et écrire sur un fichier

A. CHMOD

Sur votre hébergeur on attribue les droits sur les fichiers ou dossiers par l'intermédiaire des CHMOD, ce sont des chiffres de 3 nombres. Afin d'écrire sur un fichier comme nous allons le voir, il faut attribuer un droit d'écriture totale par le numéro 777 (voir documentation de votre logiciel FTP pour de plus amples renseignements sur la modification des CHMOD).

B. Lire

```
<?php
$fichier = fopen("test.txt", "r+");
$afficher = fgets($fichier);
fclose($fichier);
?>
```

Ce qui apparaîtra à l'écran :
charlie

- `$fichier = fopen("test.txt", "r+");` on indique grâce à la fonction `fopen` que l'on ouvre le fichier `test.txt`
- `$afficher = fgets($fichier);` on crée une variable `$afficher` à laquelle on associe la fonction `fgets` qui permet de renvoyer toute la première ligne du fichier
- `fclose($fichier);` on n'oublie pas de fermer le fichier

Pour afficher par caractères et non pas par lignes, on utilise la fonction `fgetc` à la place de `fgets`.

Pour afficher la ligne ou le caractère suivant, il suffit de mettre un deuxième `fgets` ou `fgetc`... mais il est plus pratique d'utiliser une base de données pour ce genre d'opérations.

C. Ecriture

```
<?php
$fichier = fopen('test.txt', 'r+');
fseek($fichier, 0);
fputs($fichier, Hello);
fclose($monfichier);
echo '$afficher';
?>
```

Ce qui apparaîtra à l'écran :
Hello

- `$fichier = fopen('test.txt', 'r+');` on ouvre le fichier
- `fseek($fichier, 0);` cette fonction place le curseur à la position `0` (deuxième paramètre) de la première ligne sur le fichier de la variable `$fichier` (donc `test.txt`). Elle permet de remplacer la valeur précédente et n'est donc pas obligatoire si vous ne souhaitez pas écraser les données antérieures
- `fputs($fichier, Hello);` cette fonction permet d'écrire sur le fichier `test.txt` défini par la variable `$fichier` dans le premier paramètre, on indique dans le second ce qu'il faut ajouter ici `hello`
- `fclose($fichier);` on n'oublie pas de fermer le fichier