CHAPITRE I : NOTIONS DE BASE DU LANGAGE PHP

I.1 Définition

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

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

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

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

I.2 Historique

L'origine de PHP remonte à 1995 quand Rasmus Lerdorf a créé PHP/FI, une librairie de scripts Perl utilisés pour la publication de son CV sur son site personnel. Au fur et à mesure des évolutions, la librairie a été portée en C et agrémentée de nouvelles fonctionnalités pour créer des pages dynamiques simples pour le web et accéder à quelques sources de données. PHP/FI signifie Personal Home Page/Forms Interpreter.PHP/FI 2.0 sort en 1997, toujours développé par une seule personne. C'est le succès : 50000 sites (1% des noms de domaines) disent l'utiliser ou le supporter. Les contributeurs affluent. PHP 3.0 sort en juin 1998, c'est la première version développée conjointement par Rasmus Lerdorf, Andi Gutmans et Zeev Suraski et entièrement réécrite (les versions précédentes étaient trop lentes pour une application commerciale). Outre les performances, PHP 3 est modulaire et extensible, permettant de lui greffer des API supplémentaires, ce qui n'a pas manqué avec le support de nombreuses bases de données, de formats et de protocoles, une syntaxe plus cohérente et un support basique de l'objet.

I.3 Un interpréteur Php local

EasyPHP installe et configure automatiquement un environnement de travail permettant de mettre en oeuvre toute la puissance et la souplesse qu'offrent le langage dynamique PHP et son support efficace des bases de données.

EasyPHP regroupe un serveur Apache, une base de donnée MySQL, les versions 2,3,4 et 5 de PHP ainsi que des outils facilitant le développement

I.4 Un éditeur Php

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

Un interpréteur Php sur un serveur Web

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

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

I.5 Affichage à l'écran

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

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

- echo
- print
- printf

La fonction echo

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

Syntaxe : echo Expression;

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

```
echo "Chaine de caracteres";

echo (1+2)*87;

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

I.6 Script PHP & HTML

Script Php et balises html

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

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

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

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

I.7 Règles d'écriture

Les règles de bases :

section php

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

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

<?php xxxxscriptxxxx ?> est possible

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

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

N.B: cette notation est la seule "officielle" et respectée par tous les éditeurs

HTML... (sachez que certains éditeurs n'hésitent pas à remplacer les

balises qu'ils ne connaissent pas...)

Instructions - casse

• Le séparateur d'instructions est le ; (point virgule) absolument obligatoire !

• Le langage est "casse-sensitive", autrement dit la distinction entre les minuscules-majuscules a son importance ! Ce qui permet de dire qu'il est différent de parler de la variable charles et de la variable CHARLES, comme de Charles...

Cela n'est valable pas pour les fonctions ...

• Les commentaires peuvent se faire via deux techniques

/* zone en commentaire */

ou alors pour invalider une fin de ligne

// zone en commentaire

zone en commentaire

CHAPITRE II- LES VARIABLES ET LES CONSTANTES

II.1 Déclaration d'une variable

Une variable commence par un dollar « $ » suivi d'un nom de variable. Les variables ne sont pas typées au moment de leur création. Attention PHP est sensible à la casse : var et Var ne sont pas les mêmes variables ! Voici les règles à respecter :

• Une variable peut commencer par une lettre

• Une variable peut commencer par un souligné (underscore) « _ »

• Une variable ne doit pas commencer par un chiffre.

```
// Déclaration et règles

$var=1; // $var est à 1

$Var=2; // $ Var est à 2

$_toto='Salut'; // Ok

$3petitscochons=5; // Invalide : commence par un chiffre
```

Leur type dépend de leur valeur et de leur contexte d'utilisation. Mais on peut forcer (cast) ponctuellement une variable à un type de données, ce qui s'appelle le transtypage. De même comme le type de variable peut changer en fonction de son utilisation ou du contexte, PHP effectue automatiquement un transtypage, ce qui peut parfois fournir des résultats surprenants. On affecte une valeur à une variable avec le signe égal « = » avec ou sans espace avant ou après.

```
// Déclaration et transtypage

$var='2'; // Une chaîne 2

$var+=1; // $var est maintenant un entier 3

$var=$var+0.3; // $var est maintenant un réel de type double 3.3
```

```
$var=5 + "3 petits cochons"; // $var est un entier qui vaut 8
```

Par défaut les variables sont assignées par valeur : la valeur assignée à la variable est recopiée dans la variable. PHP peut aussi travailler par référence. Une variable peut donc référencer une autre variable. On dit alors que la variable devient un alias, ou pointe sur une autre variable. On assigne par référence en utilisant le signe « & » devant la variable assignée

```
$var=2;

$ref=&$var; // $ref devient une référence de $var

echo $ref; // affiche 2

$var=3;

echo $ref; // affiche 3

$ref=4;

echo $var; // affiche 4
```

Attention à la valeur constante NULL insensible à la casse. Affecter une valeur NULL à une variable signifie ne pas puis affecter de valeur.

II.2 Portée des variables

La portée d'une variable dépend du contexte. Une variable déclarée dans un script et hors d'une fonction est globale mais par défaut sa portée est limitée au script courant, ainsi qu'au code éventuellement inclus (include, require) et n'est pas accessible dans les fonctions ou d'autres scripts.

```
$a=1; // globale par défaut

function foo() {

 echo $a; // c'est une variable locale à la fonction : n'affiche rien

}
```

Pour accéder à une variable globale dans une fonction, il faut utiliser le mot-clé global.

```
$a=1; // globale par défaut

$b=2; // idem

function foo() {

 global $a,$b; // on récupère les variables globales

 $b=$a+$b;

}

echo $b; // affiche 3
```

PHP accepte les variables statiques. Comme en C une variable statique ne perd pas sa valeur quand on sort d'une fonction.

```
function test_static() {

 static $a=0;

 echo $a; // +1 à chaque passage dans la fonction
```

```
$a++;
}
```

II.3 Variables prédéfinies

PHP dispose d'un grand nombre de variables prédéfinies. Ces variables sont généralement de type scalaires ou des tableaux. Elles sont souvent de type superglobales, c'est à dire accessible depuis n'importe où sans notion de portée. Voici quelques tableaux prédéfinis (voir au point Tableaux pour comprendre leur utilisation).

• $_GLOBALS : tableau des variables globales. La clé est le nom de la variable.

• $_SERVER : variables fournies par le serveur web, par exemple 'SERVER_NAME'

• $_GET : variables fournies par HTTP par la méthode GET (formulaires)

• $_POST : idem mais pour la méthode POST

• $_COOKIE : les variables fournies par un cookie

• $_FILES : variables sur le téléchargement d'un fichier (upload)

• $_ENV : accès aux variables d'environnement du serveur

• $_SESSION : les variables de session (voir cours sur les sessions)

II.4 Variables dynamiques

Une variable dynamique utilise la valeur d'une variable comme nom d'une autre variable. On utilise les variables dynamiques en rajoutant un « $ » devant le nom de la première variable.

```
$a="var";
$$a=1; // $$a=1 equivaut en fait à $var=1
echo $a; // affiche var
echo $$a; // affiche 1
echo $var; // affiche 1
```

II.5 Types de variables

- **booléens**

Un booléen peut prendre deux valeurs TRUE ou FALSE. Les deux constantes TRUE et FALSE peuvent être utilisées sans aucune distinction de casse (pas de différences entre les majuscules et les minuscules).

```
$var=FALSE; // FALSE, False, false, ...
$var2=True; // TRUE, True, true, ...
```

Tous les types peuvent être convertis en booléens. Voici les cas où une variable retournera FALSE

en booléen suivant le type :

• Booléen : FALSE

• Entier : 0 (zéro)

• Nombre flottant : 0.0 (zéro)

• Chaîne : chaîne vide "" ou "0" (zéro)

• Tableau : tableau vide sans aucun élément

• Objet : objet vide sans aucun élément

• Constante spéciale NULL

Dans tous les autres cas, la valeur retournée est TRUE. Attention : -1 est considéré comme vrai donc TRUE. Comme en C, les tests de conditions dans les structures de contrôles effectuent une convertion booléenne de la condition.

```
if($var==true) echo "ok";

if($var) echo "ok"; // Identique
```

- **Entiers**

Un entier est l'ensemble des nombres naturels, c'est à dire sans virgule, positifs ou négatifs. Les entiers sont généralement codés sur 32 bits mais cela dépend de l'architecture. Si on affecte un nombre entier qui dépasse la capacité de la variable, celle-ci sera transformée en réel (float). Enfin il n'y a pas de notion d'entier non signé.

Lors de la conversion d'un booléen en entier, FALSE devient 0 et TRUE devient 1. Lors de la conversion d'un nombre à virgule flottante, le nombre sera arrondi à la valeur inférieure s'il est

- **Virgule flottante**

On parte ici des nombres réels, double ou float, c'est à dire les nombres à virgules. La virgule est spécifiée par le point « . ». La puissance de 10 s'exprime avec le « e » ou le « E ».

```
$var=1.234;

$var2=1.1e4; // 1.1 * 10^4 : 11000
```

- **Chaînes de caractères**

Une chaîne est une séquence de caractères. Il n'y a pas de limite théorique pour la taille de la chaîne.

On distingue trois syntaxes pour utiliser un chaîne

• Les guillemets simples '...' (apostrophes) : Comme en shell, tous les caractères inclus dans la chaîne sont sortis tels quels sans interprétation. Si vous devez afficher un guillemet simple, il faudra l'échapper : \'

• Les guillemets doubles "..." : Certaines séquences de caractères sont interprétées et les variables sont substituées (remplacées par leur valeur).

HereDoc : Identique aux HereScripts en Shell. Le texte saisi jusqu'à un délimiteur spécifié est

placé dans la variable.(Ne sera pas abordé dans ce cours)

Voici un tableau issu de la documentation PHP des séquences pouvant être utilisés avec les guillemets doubles.

Séquence	Valeur
\n	Nouvelle ligne
\r	Retour à la ligne
\t	Tabulation horizontale
\\	Antislash
\$	Caractère $

\" Guillemets doubles

```
echo 'Attention l\'abus d\'alcool est dangereux';

$var=2345;

echo "la valeur de \$var est $var\n";
```

N'importe quelle variable peut être affichée dans une chaîne comme ci-dessus mais attention si deux variables ont un nom proche ! Il faut alors utiliser les accolades { } comme en shell. Enfin on peut accéder à n'importe quel caractère dans une chaîne en indiquant sa position (qui commence à zéro) entre deux accolades collées juste après le nom de la variable.

```
$fic='nguessan.fic';

$fics='Henriette Marie';

echo "$fics ${fic}s"; // affiche Henriette Marie nguessan.fics

echo "$fic{3}"; // affiche e
```

On peut facilement concaténer deux chaînes avec l'opérateur point « . ». On peut ajouter du texte à une chaîne avec l'opérateur point égal « .= ».

```
$str="Salut les Amis !\n ";

$str.="Comment ça va ?"; // "Salut les Amis !\n Comment ça va ?

$str2=$str."\n"; // "Salut les Amis !\nComment ça va ?\n
```

Si vous devez utiliser une chaîne de caractères comme valeur numérique (dans une addition par exemple, attention à son contenu. La chaîne sera de type double (réel) si elle contient un 'e' ou un 'E'. Sinon ce sera un entier. La valeur numérique est ensuite définie par le début de la chaîne. Si la chaîne commence par une valeur numérique, elle sera utilisée, sinon elle sera égale à 0 zéro. Si la première expression est une chaîne, le type de variable dépend de la seconde expression.

```
$val=10+"2.55"; // float, 12.55

$val=1+"charles5"; // 1 + 0 = 1

$val=2+"3 gros gateaux"; // 2 + 3 = 5 (le 3 est en premier dans la chaîne)
```

- **Les tableaux**

Un tableau PHP est une association ordonnée. Une association fait correspondre des valeurs à des clés. Les tableaux sont très souples, ils peuvent avoir de multiples dimensions, être indexés par une clé numérique ou texte, être utilisés comme table de hachage, une pile, une queue, ... Une valeur de tableau peut être elle-même un tableau, pour créer des arbres par exemple.

Un tableau est créé avec la fonction **array()** qui prend comme arguments des paires « key => value » séparées par des virgules. La clé peut être soit un entier soit du texte. Attention, 8 est un entier, 08 une chaîne ! Si la clé est absente alors c'est la dernière clé entière plus 1 qui est choisie. Si c'est la première, c'est 0 zéro.

On accède aux éléments d'un tableau à l'aide des crochets « [et] ». On place entre ces crochets la clé entière ou la chaîne.

```
$var=array(10,15,17,23,9);

echo $var[0]; // 10
```

```
echo $var[2]; // 17
$tab=array("a"=>12,"nom"=>"toto","pipo",17,4=>5);
echo $tab[0]; // pipo
echo $tab[1]; // 17
echo $tab['a']; // 12
echo $tab['nom']; // toto
```

L'utilisation de la fonction array n'est pas obligatoire et on peut déclarer un tableau à la volée.

```
$tab2[1]=2;
$tab2[]=6; // equivaut $tab2[2]=6
$tab2['test']='Ma chaîne';
```

On peut aussi créer des tableaux multidimensionnels à l'aide des deux méthodes précédentes.

```
$tab=array("un"=>array("Eric",1=>"Didier",2=>'Baudelaire'),2=>array(1,2,3),array('un','deux','trois'));
echo $tab['un'][0]; // Eric
echo $tab[2][1]; // 2
echo $tab[3][2]; // trois
$tab2['un']['deux']='test'; // créé un tableau à deux dimensions
```

Il existe une fonction très pratique pour lister le contenu d'un tableau, ou pour modifier les éléments : foreach().

```
$tab=array(1=>'un',2=>'deux',3=>'trois');
foreach($tab as $valeur) {
 echo "$valeur \n"; // affiche un deux trois
}
foreach($tab as $cle => $valeur) {
 echo "$cle => $valeur\n"; // affiche 1 => un, 2 => deux, 3 => trois
}
```

- **La variable objet**

Les objets se créent avec l'instruction class. Pour créer une instance de l'objet il faut utiliser le mot clé new.

```
class test {
 function affiche_hello() {
  echo "Hello !";
 }
}
```

```
$obj=new test;

$obj->affiche_hello();
```

Obtenir le type d'une variable

Pour obtenir le type d'une variable, on utilise la fonction « gettype » qui retourne une chaîne de texte indiquant le type. Mais attention rien ne garantit que le résultat soit le même d'une version PHP à une autre.

Les types retournés sont "boolean", "integer", "double" (pour des raisons historiques, "double" est retournée lorsqu'une valeur de type float est fournie, au lieu de la chaîne "float"), "string", "array", "object", "resource" (depuis PHP 4), "NULL" (depuis PHP 4), "unknown type"

Si vous souhaitez réellement tester le type d'une variable, il est préférable d'utiliser les fonctions de type « is_* » : is_array, is_bool, is_double, is_float, is_int, is_integer, is_long, is_null, is_numeric, is_object, is_real, is_resource, is_string, is_callable (est-ce une fonction).

Définir et supprimer une variable

Si vous souhaitez savoir si une variable est définie, c'est à dire si elle est affectée, on utilise « isset ».

Enfin si vous souhaitez supprimer une variable, il faut utiliser « unset ».

II.6 Les constantes

La constante est un nom qui prend une valeur ne pouvant pas être modifiée une fois fixée. Une constante n'est accessible qu'en lecture seule. Elles sont sensibles à la casse et doivent par convention être écrites en majuscules.

On définit une constante avec la fonction define() et doit respecter certaines règles :

• une constante ne commence pas par un $

• une constante est accessible depuis n'importe quel endroit du code

• une constante ne peut pas être redéfinie

• une constante ne peut contenir qu'un scalaire (entier, booléen, chaîne, double).

```
define(CONSTANTE,"valeur");
```

CHAPITRE III : LES OPERATEURS ET LES FONCTIONS

III.1 Les operateurs

- **Opérateurs arithmétiques**

Les opérateurs +, -, *, / et %. Le « % » est le modulo : le reste de la division.

- **Opérateurs d'assignation**

Le principal est le = mais on a aussi comme en C des opérateurs combinés +=, -=, *=, /=, %=, .= ...

- **Opérateurs de comparaison**

Les opérateurs sont == ($a==$b, même valeur), === ($a===$b, même valeur et même type), != ou <> (différent), <, >, <=, >=.

Il y a aussi l'opérateur ternaire « ?: » expr1?expr2:expr3 Si expr1 est vrai alors expr2 sinon expr3.

- **Opérateur d'erreur**

On dispose d'un opérateur spécial @ qui appliqué à une expression empêche la sortie d'un message d'erreur en cas de problème. On peut toujours récupérer le message d'erreur éventuel à l'aide de la variable $php_errormsg mais uniquement si l'option « track_errors » est à « On » dans le php.ini.

$retour=@$tab['toto']; // ne retourne pas d'erreurs si l'index toto n'existe pas

- **Opérateur d'exécution**

On peut exécuter des commandes externes au PHP comme en Shell avec les opérateurs « guillemts inverses « ` » (altgr+6). Attention l'option « safe_mode » doit être à « On » dans le php.ini. On peut aussi utiliser la fonction « shell_exec » qui fait exécuter une commande par le shell.

- **Opérateurs d'incrémentation/décrémentation**

On dispose comme en C des opérateurs ++ et --, à utiliser avant ou après le nom de variable.

$a++; // retourne $a puis l'incrémente de 1

++$a; // incrémente $a de 1 puis retourne $a

$a--; // retourne $a puis décrémente de 1

--$a; // décrémente $a de 1 puis retourne $a

- **Opérateurs logiques**

Les opérateurs logiques sont :

« and » ou « && » ($a and $b, $a && $b) vrai si $a et $b sont vrais

« or » ou « || » ($a or $b, $a || $b) vrai si $a ou $b sont vrais

« xor » ($a xor $b) vrai si $a ou $b sont vrais mais pas les deux en même temps

« ! » (!$a) vrai si $a est faux.

Attention, and et or n'ont pas la même priorité (priorité plus faible) que && et || !

- **Opérateurs de chaînes**

Il y a deux opérateurs de chaînes : le « . » qui concatène deux chaînes entre elles et le « .= » déjà vu qui est l'opérateur d'assignation.

$a="Bonjour";

$b=$a." les amis"; // $b contient Bonjour les amis

$b.="! Salut."; // $b contient Bonjour les amis! Salut.

- **Opérateur de tableaux**

On peut « additionner » deux tableaux entre eux avec le « + » : le tableau de droite est ajouté au tableau de gauche.

La notion d'expression

En PHP, une expression peut être résumée en « tout ce qui a une valeur ». Ceci dit, on remarque vite que en PHP tout ou presque est une expression. Une variable ou une constante se voient affectés des valeurs. Cette valeur est donc l'expression de la variable ou de la constante.

Nous pouvons résumer en disant qu'une expression représente tout ce qui peut être évalué. On ne peut évaluer que les valeurs...

Une fonction peut aussi être une expression si elle retourne une valeur. On dit généralement qu'une fonction a retourné telle valeur. Une comparaison est aussi une expression : elle retourne une valeur booléenne. Un calcul est aussi une expression, puisqu'une valeur est retournée, affectée, calculée.

PHP est orienté expression ce qui fait que tout ou presque en PHP est une expression. PHP cherche donc à décomposer tout ce qui est possible en expressions à évaluer, pour optimiser ses traitements.

Voici quelques exemples.

$a=2;

On distingue au premier coup d'oeil deux valeurs : la variable $a et la valeur 2, la variable $a étant mise à jour avec la valeur 2. Or en interne il y a une troisième valeur qui rentre en jeu: la valeur de l'assignation, ici elle-même assignée à 2. Cette ligne signifie en fait : « $a=2 est une expression qui

vaut 2 ».

$b=$a=2;

Les assignations sont analysées de droite à gauche. Ici nous avons l'expression $a=2 qui vaut donc 2. $b vaut donc 2 aussi. C'est plus rapide à écrire et à exécuter que $a=2; $b=2;

$a=$a+3;

PHP évalue d'abord l'expression $a+3 soit la valeur de $a incrémentée de 3, et assigne la nouvelle valeur à $a.

$a+=3;

Le résultat est le même, mais il est plus rapide car $a n'est évalué qu'une seule fois. On peut en arriver à avoir des expressions surprenantes :

$a=2;

$b=$a+=3;

La variable $a est incrémentée de 3 et vaut donc 5. L'expression $a+=3 vaut 5. $b se voit assigné la valeur 5. L'expression vaut donc 5.

Dernière chose :

$a?$b:$c

si l'expression $a est vrai, l'expression $b est évaluée, sinon l'expression $c est évaluée.

$a=TRUE; $b=2;$c=3;

echo $a?$b:$c; // affiche 2

III.2 Les structures de contrôle

if

if(expression) commande ou { bloc de commandes }

else commande ou { bloc de commandes }

Il y a aussi le « elseif », combinaison du if et du else. Le elseif en un mot peut aussi s'écrire en deux mots : le résultat est le même. On peut écrire des elseif en chaîne. Le premier dont l'expression est vrai est exécuté.

If(expression) commande ou { bloc de commandes }

elsif(expression) commande ou { bloc de commandes }

elsif(expression) commande ou { bloc de commandes }

...

On peut placer du HTML comme commande ou dans le bloc de commande.

```
<?php if ($a == 5) { ?>
A = 5
<?php } ?>
```

On peut aussi utiliser une syntaxe alternative : on ouvre le bloc (juste après le if, le else ou le elseif) avec les « : » deux points, et on ferme l'instruction avec « endif ».

```
<?php
if ($a == 5):
    print "a = 5";
    print "...";
elseif ($a == 6):
    print "a = 6";
    print "!!!";
else:
    print "a ne vaut ni 5 ni 6";
endif;
?>
```

while

while classique

C'est la boucle « tant que » simple : tant que la condition n'est pas vraie, on continue la boucle. L'expression est placée en début de boucle : si l'expression est fausse avant de rentrer dans la boucle, la boucle n'est pas exécutée.

While(expression) commande ou { bloc de commandes }

On peut aussi utiliser la syntaxe alternative :

while(expression): commande ou { bloc de commandes }

endwhile

do ... while

C'est la seconde possibilité. Dans ce cas la commande ou le bloc de commande est exécutée au moins une fois, car l'expression conditionnelle est testée en fin de boucle.

do { bloc de commandes } while(expression)

for

Le « for » du PHP est identique au « for » du C.

for(expr1;expr2;expr3) commande ou { bloc de commandes }

« expr1 » est exécutée à la première entrée dans la boucle. « expr2 » est exécutée à chaque début d'itération jusqu'à ce que l'expression soit fausse auquel cas on sort de la boucle. « expr3 » est exécutée à la fin de l'itération.

L'usage habituel d'une telle boucle est de placer l'état initial en expr1, la condition de sortie en expr2 et le calcul en expr3. Mais on peu effectuer toutes sortes de choses.

```
// de 1 à 10

for ($i = 1; $i <= 10; print $i, $i++)


// infini

for(;;)


// de 1 à 10

for ($i = 1; $i <= 10; print $i, $i++) ;
```

On peut employer une syntaxe alternative avec le « : » et « endfor ».

for(expr1;expr2;expr3): commande ou { bloc de commandes }

endfor

foreach

La boucle « foreach » est peut-être l'une des plus intéressantes pour la manipulation de tableaux ou de résultats de requêtes SQL. Elle permet de lister les tableaux. Elle dispose de deux syntaxes. foreach(array_expression as $value) commandes foreach(array_expression as $key => $value) commandes

La première syntaxe récupère les éléments du tableau un par un, séquentiellement. La valeur de l'élément courant du tableau est placée dans $value.

La seconde syntaxe est presque identique, sauf qu'en plus la clé (l'index) de l'élément actuel est placée dans $key.

Attention : modifier la valeur de $value (ou de $key) ne modifie pas le tableau car cette boucle travaille sur une copie, pas une référence. Par contre dans le second cas, comme on dispose de la clé, rien n'empêche d'assigner quoi que ce soit à l'élément courant.

Remarque : un appel à foreach « rembobine » automatiquement le tableau à son premier élément.

Mais pas dans les autres boucles, il faut alors utiliser « reset ».

```
reset($arr);

while (list(, $value) = each ($arr)) {

 echo "Valeur: $value<br>\n";

}

foreach ($arr as $value) {

 echo "Valeur: $value<br>\n";

}

$a = array (

   "un" => 1,

   "deux" => 2,

   "trois" => 3,

   "dix-sept" => 17

);

foreach($a as $k => $v) {

   print "\$a[$k] => $v.\n";

}
```

break et continue

L'instruction « break » permet de sortir d'un for, while, foreach ou switch. On peut lui indiquer de combien de structures on souhaite sortir si elles sont emboîtées.

L'instruction « continue » permet de passer à l'itération suivante. Attention PHP considère le switch comme une boucle, et dans ce cas, réévalue le switch. On peut indiquer à continue combien de structures emboîtées relancer.

Switch

Le « switch » est équivalent à une série de if et permet de comparer avec un grand nombre de valeurs.

```
switch ($i) {

   case 0:

      print "i egale 0";

      break;
```

```
    case 1:
        print "i egale 1";
        break;
    case 2:
        print "i egale 2";
        break;
    default:
        print "i est inférieur à 0 ou supérieur à 2 »;
}
```

Le switch s'arrête à la première expression case vraie puis exécute le code suivant dans l'ordre indiqué, jusqu'à la première instruction break. S'il n'y a pas de break, tout le code jusqu'à la fin du switch est exécuté. Dans l'exemple suivant, si $i vaut 0, tous les print seront affichés !

```
switch ($i) {
    case 0:
        print "i egale 0";
    case 1:
        print "i egale 1";
    case 2:
        print "i egale 2";
}
```

Notez aussi que le default doit intervenir en dernier, sinon il n'a aucun intérêt.Enfin on peut employer une syntaxe alternative avec « : » et « endswitch ».

return

Contrairement à d'autres langages, « return » n'est pas une fonction mais une instruction. Dans une fonction, return sert à sortir de celle-ci et à retourner une valeur. Dans un script, elle sort de celui-ci. Attention cependant dans les scripts inclus (voir require et include) : le return dans ce type de code considère qu'il sort de la fonction « require » ou « include » et donc ne sort pas du script ou de la fonction dans lesquels ce code a été inclus !

Comme return est un élément du langage et pas une fonction il y a pas besoin d'utiliser les parenthèses.

require et include (_once)

« require » et « include » incluent à l'endroit actuel et exécutent le fichier PHP. Ils sont identiques dans leur fonctionnement à une exception : le traitement des erreurs. Un include produit un « warning » (le code continue en principe à s'exécuter) tandis qu'un require produit une « erreur fatale » (l'exécution s'arrête).

Comme require et include sont des éléments du langage et pas des fonctions il y a pas besoin d'utiliser les parenthèses.

« require_once » et « include_once » ressemblent à leurs homologues avec cependant une différence. Quoi qu'il arrive, le fichier est inclus une seule fois. Si un second « require_once » apparaît avec le même fichier, l'instruction ne sera pas exécutée.

III.3 Les Fonctions

Syntaxe et portée

Voici la syntaxe d'une fonction.

```
function fonc($arg1, $arg2, $arg3, ..., $argn) {
   bloc de commandes
   return $valeur
}
```

Une fonction n'a pas besoin d'être déclarée avant d'être utilisée (sauf si vous voulez rester compatible avec PHP3), du moment qu'elle apparaît dans le script.

Il y a cependant deux exceptions : si la fonction est conditionnelle, c'est à dire déclarée dans un if, ou si la fonction est déclarée dans une autre fonction. Dans le premier cas elle sera inconnue du reste du code tant que la condition du if n'est pas vraie. Dans le second cas elle ne sera connue que quand la première fonction sera exécutée. Voici un exemple tiré du manuel.

Pour le premier cas :

```
bar();
/* Impossible d'appeler foo() ici, car cette fonction n'existe pas.
   Mais nous pouvons utiliser bar() */

if ($makefoo) {
  function foo () {
    echo "Je n'existe pas tant que le programme n'est pas passé ici.\n";
  }
}
/* Maitenant, nous pouvons appeler foo() car $makefoo est maintenant vrai */
if ($makefoo) foo();
function bar() {
  echo "J'existe dès le début du programme.\n";
}
```

Pour le second cas :

```
function foo() {
  function bar() {
    echo "Je n'existe pas tant que foo() n'est pas appelé.\n";
```

```
  }
}
/* Impossible d'appeler bar() ici car il n'existe pas. */
foo();
/* Maintenant, nous pouvons appeler bar(), car l'utilisation de foo() l'a rendu
   accessible. */
bar();
```

Les fonctions sont des expressions qui ont la valeur de leur "valeur de retour".
Par exemple, considérons la fonction suivante :

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

alors on peut dire que Si foo() renvoie 5, la valeur de l'expression 'foo()' est 5.
et donc $c = foo() est équivalent à $c = 5 ;
Habituellement, les fonctions ne font pas que renvoyer une valeur constante
mais réalisent aussi des traitements.
En PHP3, les fonctions doivent être définies avant qu'elles ne soient utilisées.
Ce n'est plus le cas en PHP4 et PHP5.

- **Valeur de retour simple**

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

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

Exemple :

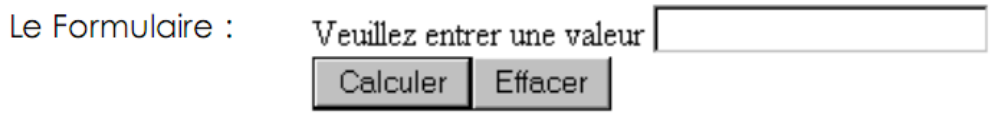
```
<?
function square($val)
 {
 $retour=$val*$val;
 return $retour;
```

```
 }
$num=4;
/* appel de la fonction*/
echo square($num);
?>
```

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

Variante

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




```
<HTML>

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

voir fichier square.html

Le script modifié devient alors

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

```
square($num);

?>
```

Valeurs de retour multiples

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

On peut passer autant d'arguments que souhaité à une fonction PHP, de tous types. Les arguments sont séparés par des virgules.

Il est possible de passer des arguments par référence, on peut donc en changer la valeur à l'intérieur de la fonction. Pour cela on rajoute un « & » devant l'argument. On peut passer ponctuellement une variable par référence en plaçant un « & » devant son nom lors de l'appel de la fonction.

```
function modif(&$msg) {

 $msg="coucou";

}

$msg="salut";

modif($msg);

echo $msg; // coucou
```

On peut aussi définir des valeurs par défaut mais ATTENTION, dans la liste des paramètres les valeurs par défaut doivent être en dernier.

```
function affiche($nom, $prenom="toto") {

 echo "$nom $prenom";

}
```

- **Valeur de retour**

On retourne une valeur avec l'instruction « return ». On peut retourner tout type de valeurs, mais pas plusieurs en même temps. Cependant rien n'empêche dans ce cas de retourner un tableau. Une fonction peut aussi retourner une référence. Dans ce cas elle doit être déclarée avec un « & » devant son nom. Mais il faut savoir quelle peut renvoyer un tableau.

```
function carre ($num) {

  return $num * $num;

}

echo carre (4); // affiche '16'
```

QUELQUES FONCTIONS PHP

FONCTIONS	ROLES
Variables, types, operateurs	
empty($var)	renvoie vrai si la variable est vide

isset($var)	renvoie vrai si la variable existe
unset($var)	détruit une variable
gettype($var)	retourne le type de la variable
settype($var, ''type'')	convertit la variable en type type (cast)
is_long(),	Permet de verifier si la variable est de type **entier long**
is_double()	Permet de verifier si la variable est de type **réel double**
is_string(),	Permet de verifier si la variable est de type **chaine de caractère**
is_array(),	Permet de verifier si la variable est de type **tableau**
is_object(),	Permet de verifier si la variable est de type **objet**
is_bool(),	Permet de verifier si la variable est de type **booléen**
is_float(),	Permet de verifier si la variable est de type **réel Float**
is_numeric(),	Permet de verifier si la variable est un nombre ou une chaîne numérique
is_integer(),	Permet de verifier si la variable est de type **entier**
is_int()	Permet de verifier si la variable est de type **entier**
…	…(même logique)
Mathématiques	
abs($x)	valeur absolue
ceil($x)	arrondi supérieur
floor($x)	arrondi inférieur
pow($x,$y)	x exposant y
round($x,$i)	arrondi de x à la ième décimale
max($a, $b, $c …)	retourne l'argument de valeur maximum
pi()	retourne la valeur de Pi
Et aussi : cos, sin, tan, exp, log, min, pi, sqrt…	Respectivement : cosinus , sinus, tangente, exponentielle, logarithme, pi, carré.
number_format ($nbr[,$dec,[$a,$b]])	retourne une chaîne de caractères représentant le nombre $nbr avec $dec décimales après formatage. La chaîne $a représente le symbole faisant office de virgule et $b le séparateur de milliers. Par défaut, le formatage est anglophone : $a = ''.'' et $b = '','' . Très utile pour représenter les nombres élevés au format francophone.
rand([$x[, $y])	valeur entière aléatoire entre 0 et RAND_MAX si x et y ne sont pas définis, entre x et RAND_MAX si seul x est défini, entre x et y si ces deux paramètres sont définis**.**
srand()	initialisation du générateur aléatoire
getrandmax()	retourne la valeur du plus grand entier pouvant être généré
Chaines de caractères	
strlen($str)	retourne le nombre de caractères d'une chaîne
strtolower($str)	conversion en minuscules
strtoupper($str)	conversion en majuscules
trim($str)	suppression des espaces de début et de fin de chaîne
substr($str,$i,$j)	retourne une sous chaîne de $str de taille $j et débutant à la position $i
strnatcmp($str1,$str2)	comparaison de 2 chaînes
addslashes($str)	déspécialise les caractères spéciaux (', '', \)
ord($char)	retourne la valeur ASCII du caractère $char
Affichage	
echo()	écriture dans le navigateur ex : echo ''Bonjour $name'';
print()	écriture dans le navigateur

	ex : print(''Bonjour $name'');
printf([$format, $arg1, $arg2])	écriture formatée comme en C, i.e. la chaîne de caractère est constante et contient le format d'affichage des variables passées en argument. ex : printf(''Bonjour %s'', $name);
Tableau	
count($tab), sizeof	retournent le nombre d'éléments du tableau
in_array($var,$tab)	dit si la valeur de $var existe dans le tableau $tab
list($var1,$var2…)	transforme une liste de variables en tableau
range($i,$j)	retourne un tableau contenant un intervalle de valeurs
shuffle($tab)	mélange les éléments d'un tableau
sort($tab)	trie alphanumérique les éléments du tableau
rsort($tab)	trie alphanumérique inverse les éléments du tableau
implode($str,$tab), join	retournent une chaîne de caractères contenant les éléments du tableau $tab joints par la chaîne de jointure $str
explode($delim,$str)	retourne un tableau dont les éléments résultent du hachage de la chaîne $str par le délimiteur $delim
array_merge($tab1,$tab2,$tab3…)	concatène les tableaux passés en arguments
array_rand($tab)	retourne un élément du tableau au hasard
usort($tab, ''fonction'')	Trie les éléments grâce à la fonction fonction définie par l'utilisateur qui doit prendre 2 arguments et retourner un entier, qui sera inférieur, égal ou supérieur à zéro suivant que le premier argument est considéré comme plus petit, égal ou plus grand que le second argument. Si les deux arguments sont égaux, leur ordre est indéfini.
reset($tab)	Place le pointeur sur le premier élément
current($tab)	retourne la valeur de l'élément courant
next($tab)	Place le pointeur sur l'élément suivant
prev($tab)	place le pointeur sur l'élément précédant
each($tab)	retourne la paire clé/valeur courante et avance le pointeur

	Ex: $colors = array(''red'', ''green'', ''blue'');
	$nbr = count($colors);
	reset($colors);
	for($i=1; $i<=$nbr; $i++) {
	echo current($colors).''
	'';
	next($colors);
	}

Tableau associatif	
array_count_values($tab)	retourne un tableau contenant les valeurs du tableau $tab comme clés et leurs fréquence comme valeur (utile pour évaluer les redondances)
array_keys($tab)	retourne un tableau contenant les clés du tableau associatif $tab
array_values($tab)	retourne un tableau contenant les valeurs du tableau associatif $tab
array_search($val,$tab)	Retourne la clé associée à la valeur $val
Date	
date(''$format'')	Retourne une chaîne de caractères contenant la date et/ou l'heure locale au format spécifié *Ex :* echo date("Y-m-d H:i:s");

	/* affiche la date au format MySQL : '2012-03-31 22:30:29' */
	Autres exemples :
	// notation française
	$aujourdhui = date("d/m/y"); // 12/03/12
	$aujourdhui = date("d/m/Y"); // 12/03/2012
	$aujourdhui = date("d-m-y"); // 12-03-12
	$aujourdhui = date("d .m.Y"); // 12.03.2012

getdate()	Retourne un tableau associatif contenant la date et l'heure *Ex:* $aujourdhui = getdate(); $mois = $aujourdhui['mon']; $jour = $aujourdhui['mday']; $annee = $aujourdhui['year']; echo "$jour/$mois/$annee"; // affiche '31/3/2002'
checkdate($month, $day, $year)	vérifie la validité d'une date if(checkdate(12, 31,2001)) echo ''La St Sylvestre existe même chez les anglais !!!'';
mktime ($hour, $minute, $second, $month,$day, $year)	retourne le timestamp UNIX correspondant aux arguments fournis c'est-à-dire le nombre de secondes entre le début de l'époque UNIX (1er Janvier 1970) et le temps spécifié
time()	retourne le timestamp UNIX de l'heure locale
Fonctions dynamique	
create_function($params,$code) :	Crée une fonction anonyme (style lambda). Prend en argument la liste $params des arguments de la fonction à créer ainsi que le code $code de la fonction sous la forme de chaînes de caractères. Il est conseillé d'utiliser les simples quotes afin de protéger les noms de variables '$var', ou alors d'échapper ces noms de variables \$var. Le nom de la fonction créée sera de la forme : lambda_x où x est l'ordre de création de la fonction. *Exemple :* $newfunc = create_function('$a,$b',''return \$a+\$b;''); echo ''Nouvelle fonction anonyme : $newfunc ''; echo $newfunc(5,12).'' '';
func_num_args()	Retourne le nombre d'arguments passés à la fonction en cours.
func_get_arg($nbr)	Retourne un élément de la liste des arguments envoyés à une fonction. Elle ne doit être utilisée qu'au sein d'une fonction. L'indice $nbr commence à zéro et renvoie le $nbr-1 ème paramètre.
Fichier	
fopen($file [,$mode])	ouverture du fichier identifié par son nom $file et dans un mode $mode particulier, retourne un identificateur $fp de fichier ou FALSE si échec
fopen($fp)	ferme le fichier identifié par le $fp
fgets($fp, $length)	lit une ligne de $length caractères au maximum
fputs($fp, $str)	écrit la chaîne $str dans le fichier identifié par $fp
fgetc($fp)	lit un caractère
feof($fp)	teste la fin du fichier
file_exists($file)	indique si le fichier $file existe
filesize($file)	retourne la taille du fichier $file
filetype($file)	retourne le type du fichier $file
unlink($file)	détruit le fichier $file
copy($source, $dest)	copie le fichier $source vers $dest
readfile($file)	affiche le fichier $file

rename($old, $new)	renomme le fichier $old en $new
Accès aux dossiers	
chdir($str)	Change le dossier courant en $str. Retourne TRUE si succès, sinon FALSE.
getcwd()	Retourne le nom du dossier courant (en format chaîne de caractères).
closedir($d)	Ferme le pointeur de dossier $d.
readdir($d)	Lit une entrée du dossier identifié par $d. C'est-à-dire retourne un nom de fichier de la liste des fichiers du dossier pointé. Les fichiers ne sont pas triés. Ou bien retourne FALSE s'il n'y a plus de fichier.
rewinddir($d)	Retourne à la première entrée du dossier identifié par $d.

CHAPITRE IV : GESTION DES FORMULAIRE EN PHP

I. GET et POST

Le but est de récupérer le contenu des champs d'un formulaire HTML dans notre code PHP pour pouvoir le traiter. Lorsqu'un formulaire est envoyé à un script PHP, toutes les variables seront disponibles automatiquement dans le script.

Les formulaires peuvent être de type GET ou POST. Pour rappel, dans un formulaire de type GET, les informations sont passées directement par l'URL en clair, ce qui peut poser des problèmes de limitations suivant le serveur (de 256 à 8192 octets selon le cas).

La méthode POST n'a pas ce genre de limitation, car les informations sont transmises par le conteneur de variables globales (dans l'entête) et sont de plus cachées.

I. Récupération par tableau

Chaque champ de formulaire en PHP est défini par un nom et une valeur. Dans un script, PHP va récupérer ces noms et ces valeurs dans des tableaux spéciaux dit superglobaux (accessibles depuis partout). Pour la méthode GET, le tableau est $_GET, pour la méthode POST le tableau est $_POST. Si vous ne souhaitez pas vous soucier de la méthode, vous pouvez utiliser le tableau $_REQUEST. En index on aura le nom du champ de formulaire (ou de la variable passée en URL) et en valeur la valeur du champ. Par exemple :

```
<form action="affiche.php" method="post">

Name: <input type="text" name="username"><br>

Email: <input type="text" name="email"><br>

<input type="submit" name="submit" value="Submit me!">

</form>
```

Dans la page PHP traitement.php on aura :

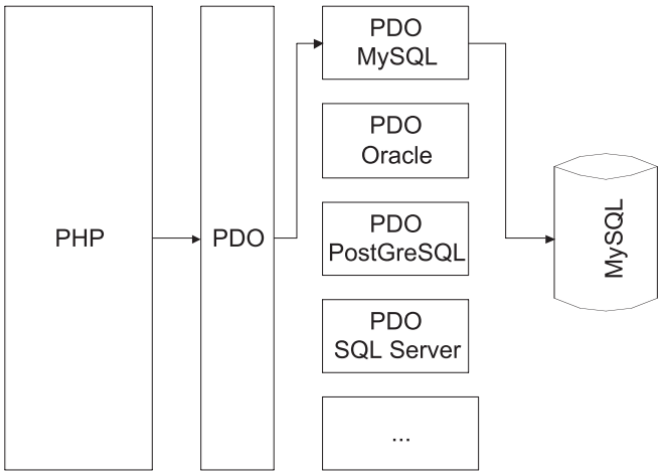
```
<?php

echo $_POST['username'];

echo $_REQUEST['email'];

?>
```

CHAPITRE V : PDO, PHP Data Object

PDO (PHP Data Object) est la principale nouveauté de PHP 5.1. Cette extension vous apportera un confort d'utilisation et une abstraction plus importants que les anciennes fonctions natives propres à chaque SGBD. L'approche objet de PDO vous permettra de plus d'étendre les fonctions d'accès à votre base facilement et de manière transparente.

En interne, PDO permet à l'équipe de développement de PHP de développer beaucoup plus rapidement de nouveaux connecteurs vers de nouvelles bases de données. Au lieu de tout réécrire à partir du début comme auparavant, ils peuvent se baser sur une architecture complète et ne rajouter que ce qui est spécifique.

PDO est un socle commun pour les connecteurs vers les SGBD. Il s'occupe d'offrir des fonctions de base ainsi que d'unifier les interfaces utilisateur. Il ne s'agit pas à proprement parler d'un système d'abstraction aux bases de données, bien qu'il puisse servir en ce sens.



Architecture des drivers PDO

Particularités

Performances

Écrit en langage C, PDO est beaucoup plus rapide qu'un système d'abstraction développé en PHP (tel qu'AdoDB, PEAR DB…) et fournit des performances similaires aux anciens pilotes natifs. Les requêtes préparées offrent de plus des possibilités d'optimisation qui n'étaient pas présentes en PHP 4 avec l'ancienne extension MySQL.

Aptitudes

PDO permet d'exécuter tous les types de requête classiques (INSERT, UPDATE, DELETE, SELECT ou exécution de procédures stockées si votre SGBD le permet). Les données reçues pourront être extraites via plusieurs types de sorties (tableau, objet, variables liées par références...). Les transactions et les modes d'autovalidation (autocommit) sont bien entendu disponibles.

En plus de ces fonctionnalités, PDO permet d'employer des requêtes paramétrées et de normaliser les accès (gestion de la casse des noms de colonnes ou de la syntaxe des paramètres par exemple). PDO émule certaines de ces fonctionnalités si jamais votre SGBD ne les supporte pas (simulation par exemple de l'utilisation de requêtes préparées). Par conséquent, vous n'aurez à porter attention qu'au code SQL lui-même et à ses différences entre

les SGBD. Utiliser au maximum du code SQL standard vous permet de réduire fortement la dépendance à votre SGBD.

Bases de données supportées

PDO inclut une compatibilité avec les principales bases de données avec lesquelles PHP peut communiquer. Dans le cas où la compatibilité native n'est pas supportée, vous pouvez utiliser un pont ODBC. Vous retrouverez entre autres :

•MySQL 3, 4 et 5 (pdo_mysql) ; • PostgreSQL (pdo_pgsql) ; •SQLite 2 et 3 (pdo_sqlite) ;

•Oracle (pdo_oci) ; •ODBC (pdo_odbc).

SQLite3 et PDO

À ce jour, PDO est le seul moyen de se connecter à SQLite3 via PHP.

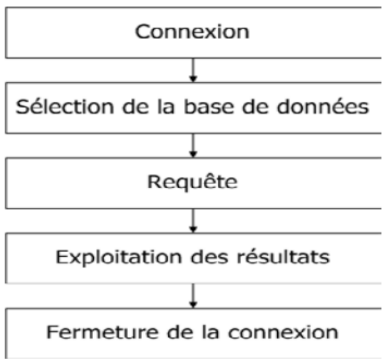
Installation

Pour installer PDO, reportez-vous au chapitre concernant l'installation de PHP. Notez qu'il faut activer le noyau PDO et la composante PDO spécifique à votre base de données.

Utiliser votre base de données

L'utilisation de votre base de données avec PHP s'effectue en cinq étapes:

Notons que la dernière action n'est pas obligatoire. La connexion est automatiquement fermer à la fin de l'exécution du script par le moteur PHP. Garder ouverte une connexion si on ne s'en sert plus peut toutefois occuper inutilement des ressources.

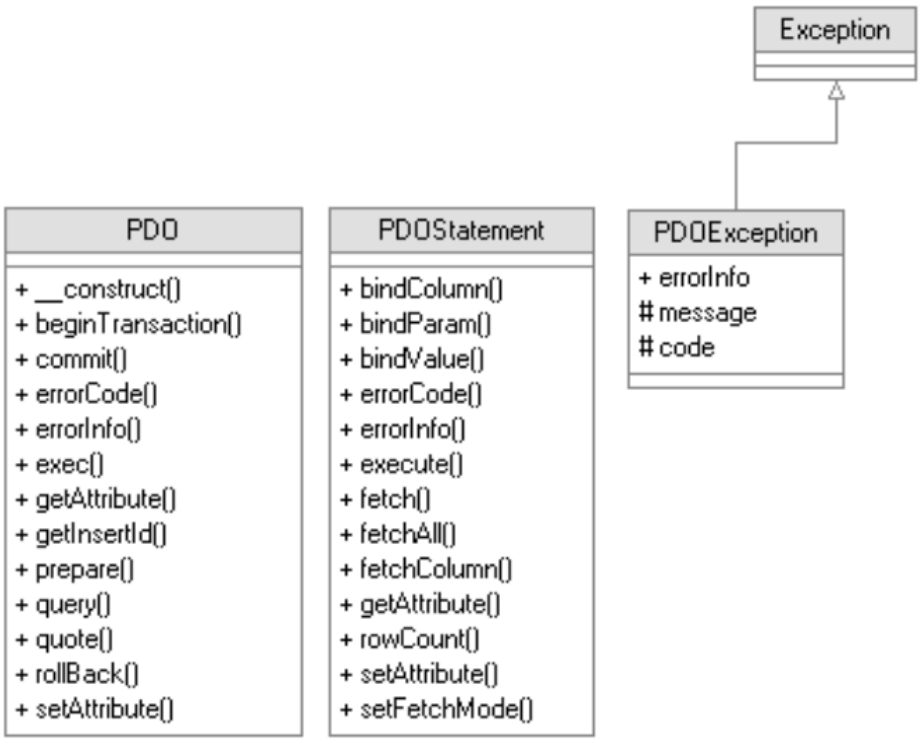


Utilisation de PHP pour accéder à une base de données

Dans nos exemples suivants, nous allons privilégier l'utilisation du SGBD MySQL. Il se peut que certaines requêtes SQL soient à adapter pour votre SGBD, mais l'utilisation de l'extension PDO reste la même.

Structure des classes de PDO

Il existe trois classes principales liées à l'utilisation de PDO : la classe PDO qui correspond à votre lien à la base de données ; la classe PDOStatement qui correspond aux requêtes que vous pourriez faire, ainsi qu'à leur résultat ; et enfin la classe PDOException qui permet de traiter les erreurs.



Modèle des classes de PDO

Créer un fichier de configuration

Vous allez utiliser des connexions aux bases de données en plusieurs points de votre application. La solution impliquant de définir sur chaque page le login, le mot de passe et le serveur de votre base de données est mauvaise, car il est fréquent de devoir changer ces valeurs.

Vous aurez tout intérêt à mettre ces différentes valeurs dans une fonction dédiée ou dans un fichier de configuration. Une méthode simple, si vous n'avez pas de gestion de configuration, est de définir un fichier contenant les paramètres de connexion et l'instanciation de la classe PDO. Il suffira de l'inclure au début de chaque script utilisant MySQL :

```
<?php
define('USER1', 'cyril');
define('PASS1', 'motdepasse');
define('DSN1', 'mysql:host=localhost;dbname=publication');
try {
  $dbh = new PDO(DSN1, USER1, PASS1);
} catch (PDOException $e) {
  print "Erreur ! : " . $e->getMessage() . "<br/>";
  die();
}
```

?>

Effectuer une requête

Une fois votre connexion ouverte, vous allez pouvoir utiliser pleinement votre base de données, pour lire, modifier ou encore supprimer des données. Pour cela, il n'existe pas de fonctions de lecture/écriture comme pour la gestion des fichiers. On utilise directement le langage SQL dont les principes généraux ont été décrits au chapitre précédent.

Pour envoyer une requête au serveur, on peut utiliser deux méthodes de l'objet PDO : exec() et query().

Quand vous exécutez une requête avec les méthodes ou , vous ne faites qu'envoyer votre ordre à votre base de données. Il faut ensuite traiter le résultat. Pour cela, nous distinguerons deux cas :

•après une requête de sélection qui renvoie des résultats ;

•après une requête d'insertion/modification.

Pour une requête ne renvoyant pas de résultats à proprement parler (UPDATE, INSERT…), il faut utiliser la méthode exec() qui retourne le nombre de lignes concernées par la requête.

La méthode exec() permet d'exécuter une requête mais ne renvoie que le nombre de lignes modifiées : on s'en servira généralement pour faire des insertions, des modifications ou des suppressions.

Nbe de lignes affectées ß exec (requête_sql)

Pour une requête renvoyant des résultats (SELECT, DESC, SHOW ou EXPLAIN), il faudra utiliser la méthode query() qui retourne une instance de l'objet PDOStatement contenant les résultats que vous pourrez réutiliser par la suite pour les lire.

Instance de la classe PDOStatment ß query (requête_sql)

La méthode query() permet de récupérer des données. Elle renvoie une instance de la classe PDOStatement.

$resultat = $dbh->exec($sql);

Requêtes invalides

Dans le cas où la requête ne fonctionne pas, les méthodes query() et exec() renvoient FALSE. Cela peut arriver quand elle est mal formée ou quand l'utilisateur ne dispose pas

des droits suffisants pour l'effectuer.

```
if($dbh->exec($sql) === FALSE){
  echo 'Il y a une erreur dans votre requête sql :';
  echo $sql ;
  exit() ;
}
if($dbh->query($sql) === FALSE){
  echo 'Il y a une erreur dans votre requête sql :';
  echo $sql ;
  exit();
}
```

Attention

Notez que nous utilisons « === » et FALSE plutôt que le comparateur de valeur (et non de type) « == ». Étant donné qu'une requête ne renvoyant pas de résultat retournera 0, il faut y faire attention.

Requête de sélection

Pour toutes les requêtes renvoyant des données autres que le nombre d'enregistrements concernés, il faut utiliser la méthode query() de l'objet PDO correspondant.

Après l'exécution d'une requête de sélection, les données ne sont pas affichées, elles sont simplement mises en mémoire. Il faut donc aller les chercher et les afficher.

La méthode query() vous renvoie une instance de la classe PDOStatment. Cette dernière dispose de deux méthodes qui permettront de manipuler les données renvoyées :

•La méthode fetchAll() retourne l'ensemble des données sous forme d'un tableau PHP et libère le SGBD. Vous accéderez alors directement à toutes les données et pourrez exécuter des requêtes tierces pendant l'analyse du résultat. Le contrecoup de cette facilité d'utilisation est une charge importante au niveau du serveur. La totalité des données seront en effet localisées en mémoire.

•La méthode fetch() permet une lecture séquentielle du résultat. À un instant t, vous ne lisez qu'un seul résultat et la mémoire du système n'est pas occupée avec les autres entrées. Cette méthode est très utile pour le traitement de gros résultats. Son désavantage est que vous ne pourrez pas faire d'autres requêtes sur la même connexion PDO pendant le traitement de ce résultat. Vous n'aurez pas non plus accès aux informations comme le nombre de lignes résultat avant d'avoir parcouru l'intégralité dudit résultat.

PDOStatement::fetchAll ([fetch_style])

PDOStatement::fetch ([fetch_style

[,cursor_orientation [,cursor_offset]]])

Choisir le format des résultats

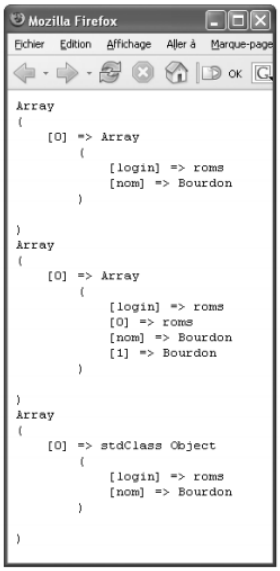
Le paramètre fetch_style détermine la façon dont PDO retourne les résultats. Il permet de définir de quel type sera le retour : tableau associatif, tableau numériquement indexé, objet.

Valeur	**Action**
PDO::FETCH_ASSOC	Retourne un tableau associatif indexé par le nom de la colonne, comme retourné dans le jeu de résultats.
PDO::FETCH_BOTH (par défaut)	Retourne un tableau indexé par les noms de colonnes mais aussi par les numéros de colonnes (commençant à l'indice 0), comme retournés dans le jeu de résultats.
PDO::FETCH_OBJ	Retourne un objet anonyme avec les noms de propriétés qui correspondent aux noms des colonnes retournés dans le jeu de résultats.

```
<?php
// Définition des variables de connexion
$user = 'cyril';
$pass = 'motdepasse';
$dsn = 'mysql:host=localhost;dbname=publication';
// Connexion à la base de données
try {
  $dbh = new PDO($dsn, $user, $pass);
  $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_WARNING)
} catch (PDOException $e) {
  print "Erreur ! : " . $e->getMessage() . "<br/>";
  die();
}
// Lecture d'enregistrements
$sql = "SELECT login, nom FROM auteur LIMIT 0,1"
$sth = $dbh->query($sql);
```

```
$result = $sth->fetchAll(PDO::FETCH_ASSOC);
print_r($result);
$sth = $dbh->query($sql);
$result = $sth->fetchAll(PDO::FETCH_BOTH);
print_r($result);
$sth = $dbh->query($sql);
$result = $sth->fetchAll(PDO::FETCH_OBJ);
print_r ($result);
?>
```

Les principaux styles de présentation des résultats



PDO::FETCH_ASSOC

PDO::FETCH_BOTH
(défaut)

PDO::FETCH_OBJ

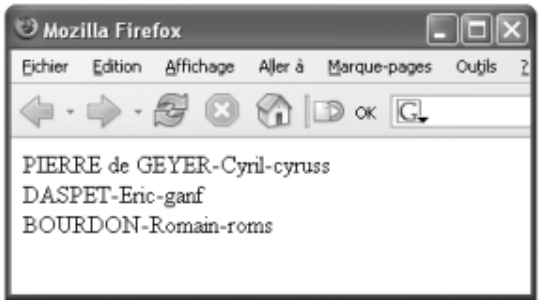
Lire tous les enregistrements

Tous les enregistrements sont renvoyés dans un tableau par la méthode fecthAll(). Il suffit donc de parcourir le tableau en affichant son contenu.

```
<?php
// Inclusion du fichier contenant la connexion à la base
include_once('connect.inc.php');
// Lecture d'enregistrements
$sql = "SELECT login, nom, prenom FROM auteur";
$sth = $dbh->query($sql);
$result = $sth->fetchAll(PDO::FETCH_ASSOC);
foreach ($result as $row){
```

```
  echo $row['nom'];  echo '-';
  echo $row['prenom'];  echo '-';
  echo $row['login'];  echo '<br/>';
}
// Fermeture  de la connexion
$dbh = NULL;
?>
```

Affichage des enregistrements retournés



Nombre d'enregistrements retournés

Si vous voulez savoir combien d'enregistrements sont concernés par une requête de sélection, vous avez deux possibilités :

- créer une requête spécifique utilisant la fonction COUNT() de MySQL ;
- compter le nombre d'éléments contenus dans le tableau renvoyé par la méthode fetchAll().

Le premier cas sera le plus adapté si vous n'avez pas besoin de traiter les données ensuite. Dans le cas contraire, la seconde possibilité sera plus appropriée.

```
<?php
// Inclusion du fichier contenant la connexion à la base
include_once('connect.inc.php');

// En utilisant une requête particulière
$sql = "SELECT count(*) as nbe FROM rmq WHERE pseudo='Adam'";
$sth = $dbh->query($sql);
$result = $sth->fetchAll();
$nombre = $result[0]['nbe'];
echo $nombre;
/* En comptant le nombre d'éléments présents dans le tableau de résultats */
$sql = "SELECT pseudo, texte FROM rmq WHERE pseudo='Adam'";
$sth = $dbh->query($sql);
$result = $sth->fetchAll();
$nombre = count($result);
echo $nombre;
?>
```

Exercice de TP 1

I. Création de la base de données

On dispose de la classe étudiant suivante :

Etudiant
NCIN : VARCHAR(8)
NOM : VARCHAR(25)
PRENOM : VARCHAR(25)
SEXE :VARCHAR(1)

1. Lancez EasyPhp ;
2. Créez la base de données MySQL dont son nom est « TPMySQL » ;
3. Créez la table Etudiant ;

II. Ajout d'un enregistrement

1. Créez un répertoire de travail « tpphpmysql » dans le chemin suivant :

```
C:\Program Files\EasyPHP1-8\www\
```

2. Lancez l'éditeur de texte «Notepad++» ;
3. Créer le fichier « AjouterEtudiant.php » qui permet de visualiser un formulaire de saisie pour l'ajout d'un nouveau étudiant. L'apparence de ce formulaire doit ressembler à la figure n°1.
4. Lorsque l'utilisateur clique sur le bouton « Valider », la page est de nouveau appelée afin d'insérer les nouvelles données sur le nouveau étudiant dans la base de données (figure n°2). Le bouton « Annuler » permet d'initialiser le formulaire.

Programmer le lien hypertexte « Liste des étudiants » qui permet d'accéder à la page « lstEtudiants.php ».

NB : Avant d'ajouter un étudiant, il faut d'abord s'assurer que l'utilisateur à bien saisie les données sur l'étudiant, ensuite on test l'existence de l'étudiant dans la base de données. S'il est existant alors vous devez le signaler à l'utilisateur à travers un message (figure n°3).

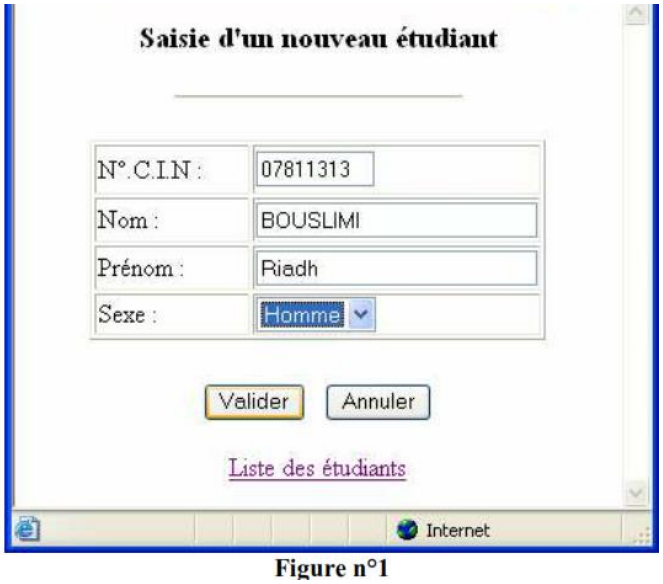


Figure n°1

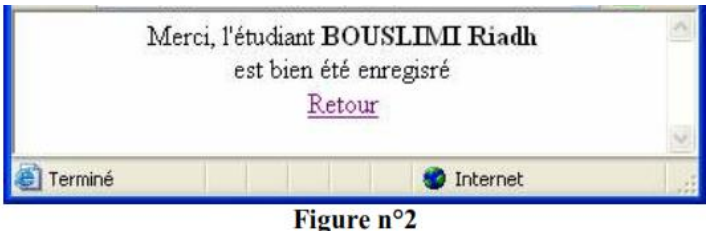


Figure n°2

Figure n°3

III. Affichage des enregistrements

1. Lancez l'éditeur de texte « Notepad++ » ;
2. Créez le fichier « lstEtudiants.php » qui permet d'afficher la liste des étudiants par sexe et ordonnée par nom et prénom selon l'ordre croissant. Cette liste doit être sous la forme d'un tableau de 5 colonnes.

NB : La dernière colonne de ce tableau doit contenir 2 liens hypertextuels « modifier » et « supprimer », qui permettent d'appeler respectivement les fichiers « modif_Etudiant.php » et « suppr_Etudiant.php » qui effectuent les traitements correspondants à partir de la référence du NCIN qui doit être passée en paramètre.(figure n°4).

Listes des etudiants

Sexe : Homme

N° C.I.N	Nom	Prénom	Sexe	Modif/Suppr
07883548	ECHI	HAMDI	H	Modifier Supprimer
08072024	NAIMI	ATEF	H	Modifier Supprimer

Figure n°4

IV. Modification d'un enregistrement

1. Lancez l'éditeur de texte «Notepad++» ;
2. Créez le fichier « modif_Etudiant.php » qui permet de modifier un étudiant.

Lorsque l'utilisateur clique sur le lien hypertexte « Modifier » (figure n°5) dans la page « lstEtudiants.php » la page « modif_Etudiant.php » apparaîtra avec un formulaire qui contient les renseignement sur l'étudiant en question (figure n°6) .

Lorsque l'utilisateur clique sur le bouton « valider » la figure n°7 apparaîtra.

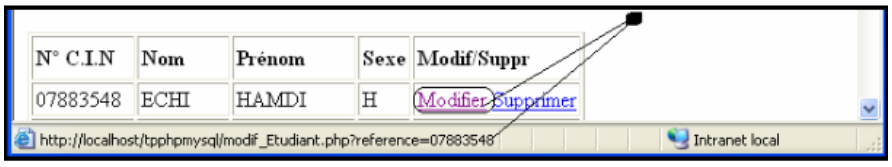


Figure n°5

Figure n°6

Figure n°7

V. Suppression d'un enregistrement

1. Lancez l'éditeur de texte «Notepad++» ;
2. Créez le fichier « suppr_Etudiant.php » qui permet de modifier un étudiant.

Lorsque l'utilisateur clique sur le lien hypertexte « Supprimer » (figure n°8) dans la page « lstEtudiants.php » la page « suppr_Etudiant.php » apparaîtra avec un message pour indiquer que la suppression a été effectué avec succès (figure n°9) .

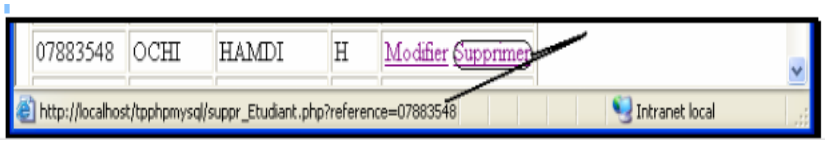


Figure n°8

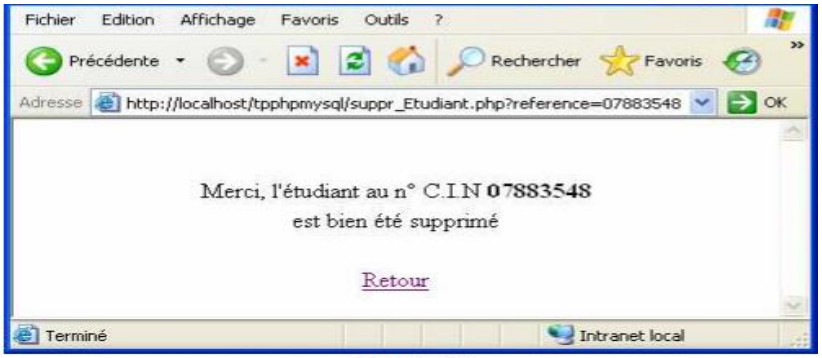


Figure n°9

Correction de TP

Création de fichier de connexion

Fichier : connexion.php

```
<?php
define('USER1', 'root');
define('PASS1', '');
define('DSN1', 'mysql:host=localhost;dbname=ecole');
```

```
try {
  $dbh = new PDO(DSN1, USER1, PASS1);
} catch (PDOException $e) {
  print "Erreur !: " . $e->getMessage() . "<br/>";
  die();
}
?>
```

Ajout d'un enregistrement

Fichier : AjouterEtudiant.php

```
<!--
    fichier : AjouterEtudiant.php

-->
<html>
<head>
     <title>Saisie d'un nouveau étudiant</title>
</head>
<body>
 <?php
   if (!empty($_POST['ncin']) AND !empty($_POST['nom'])
        AND !empty($_POST['prenom'])) {
       //connexion
       include("connexion.php");
       //test d'existance de l'étudiant
       $requete1="select * from etudiant where NCIN='".$_POST['ncin']."'";
       $resultat1= $dbh->query($requete1);

       $enreg1= $resultat1->FetchAll();

       if(count($enreg1)!=0){

         //l'étudiant est déjà existant
       ?>
         <div align="center">
           <br>L'étudiant <b><?php echo($_POST['nom']." ".$_POST['prenom']);?></b>
     <br>est déjà existant!!!
           <br><br><a href="AjouterEtudiant.php">Retour</a>
         </div>
      <?php
      }else {
       //insertion du nouveau étudiant
       $requete2="insert into etudiant
               values('".$_POST['ncin']."','".$_POST['nom']."',
                   '".$_POST['prenom']."','".$_POST['sexe']."')";
       $resultat2 = $dbh->exec($requete2);
      ?>
       <div align="center">
        <br>Merci, l'étudiant
            <b><?php echo($_POST['nom']." ".$_POST['prenom']);?></b>
        <br>est bien été enregisré
        <br><br><a href="AjouterEtudiant.php">Retour</a>
       </div>
      <?php
       }//fin du test d'existance
```

```
 }else{?>
    <div align="center">
     <h3>Saisie d'un nouveau étudiant</h3>
     <hr size="2" width="50%">
     <!--Ici c'est le Formulaire-->
     <form name="SaisieEtudiant" method="POST" action="AjouterEtudiant.php">
       <table border="1">
        <tr>
          <td width="35%">N°.C.I.N :</td>
          <td width="65%"><input type="text" name="ncin"
                              value="" size="8" maxlength="8"></td>
        </tr>
        <tr>
          <td width="35%">Nom :</td>
          <td width="65%"><input type="text" name="nom" value=""
                              size="25" maxlength="25"></td>
        </tr>


  <tr>
          <td width="35%">Prénom :</td>
          <td width="65%"><input type="text" name="prenom"
                              value="" size="25" maxlength="25"></td>
        </tr>
        <tr>
          <td width="35%">Sexe :</td>
          <td width="65%">
      <select name="sexe">
             <option value="H">Homme</option>
             <option value="F">Femme</option>
           </select>
          </td>
        </tr>
       </table>
       <br>
       <input type="submit" value="Valider">  
       <input type="reset" value="Annuler">
      </form>
      <a href="lstEtudiants.php">Liste des étudiants</a>
     </div>
 <?php
  }
 ?>
</body>
</html>
```

Affichage des enregistrements

Fichier : lstEtudiants.php

```
<!--
    fichier : lstEtudiants.php
-->
<html>
<head>
 <title>Liste des étudiants</title>
</head>
<body>
    <div align="Left">
         <h3>Liste des étudiants</h3>
   <hr width="50%" size="2">
 </div>
    <?php
    //connexion
    include("connexion.php");
```

```
    ?>
 <?php
    //Liste des étudiants dont le sexe est Homme
    $requete_H="select * from etudiant where SEXE='H' order by NOM,PRENOM";    $resultat_H
= $dbh->query($requete_H);
$enreg_H = $resultat_H->fetchAll(PDO::FETCH_ASSOC);
    if (count($enreg_H)!=0){
    ?>
  <div align="Left">
      <br>
        <h4>Sexe : <b>Homme</b></h4>
        <table border="1" cellpaccing="2" cellpadding="0">
         <tr>
       <td><b>N° C.I.N</b></td>
       <td><b>Nom</b></td>
       <td><b>Prénom</b></td>
       <td><b>Sexe</b></td>
                <td><b>Modif/Suppr</b></td>
    </tr>
       <?php

    foreach($enreg_H as $rows){

   ?>
     <tr>
        <td><?php echo $rows["NCIN"];?></td>
        <td><?php echo $rows["NOM"];?></td>
        <td><?php echo $rows["PRENOM"];?></td>
        <td><?php echo $rows["SEXE"];?></td>
                <td>
                   <a href="modif_Etudiant.php?reference=
                             <?php echo $rows["NCIN"];?>
                          ">Modifier</a>
                   <a href="suppr_Etudiant.php?reference=
                             <?php echo $rows["NCIN"];?>
                          ">Supprimer</a>
                </td>
    </tr>
   <?php
   }
      ?>
    </table>
    </div>
    <?php
    }else{
    ?>
  <div align="Left">
        <h4>Sexe : <b>Homme</b></h4>
    <hr width="50%" size="2">
        <b> Pour le moment, il n'a aucun étudiant enregistré!!!</b>

        <hr width="50%" size="2">
      </div>
    <?php
    }
    ?>
    <?php
    //Liste des étudiants dont le sexe est Femme
    $requete_F="select * from etudiant where SEXE='F' order by NOM,PRENOM";
    $resultat_F = $dbh->query($requete_F);
    $enreg_F = $resultat_F->fetchAll(PDO::FETCH_ASSOC);
    if (count($enreg_F!=0)){
    ?>
  <div align="Left">
      <br>
        <h4>Sexe : <b>Femme</b></h4>
        <table border="1" cellpaccing="2" cellpadding="0">
      <tr>
```

```
     <td><b>N° C.I.N</b></td>
     <td><b>Nom</b></td>
     <td><b>Prénom</b></td>
     <td><b>Sexe</b></td>
               <td><b>Modif/Suppr</b></td>
  </tr>
        <?php

     Foreach($enreg_F as $rows){
   ?>
  <tr>
     <td><?php echo $rows["NCIN"];?></td>
     <td><?php echo $rows["NOM"];?></td>
     <td><?php echo $rows["PRENOM"];?></td>
     <td><?php echo $rows["SEXE"];?></td>
               <td>
<a href="modif_Etudiant.php?reference=<?php echo $rows["NCIN"];?>"> Modifier</a>
<a href="suppr_Etudiant.php?reference=<?php echo $rows["NCIN"];?>"> Supprimer</a>
               </td>
   </tr>
   <?php
   }
      ?>
    </table>
    </div>
    <?php
    }else{
    ?>
  <div align="Left">
       <h4>Sexe : <b>Femme</b></h4>
    <hr width="50%" size="2">
       <b> Pour le moment, il n'a aucune étudiante enregistrée!!!</b>
       <hr width="50%" size="2">
       </div>
    <?php
    }
    ?>
</body>
</html>
```

Modification d'un enregistrement

Fichier : modif_Etudiant.php

```
<!--
     fichier : modif_Etudiant.php
-->
<html>
<head>
     <title>Modification d'un étudiant</title>
</head>
<body>
 <?php
  //connexion
      include("connexion.php");
  if (empty($_POST['ncin']) AND empty($_POST['nom'])
       AND empty($_POST['prenom'])) {
       //Info. sur l'étudiant
       $requete1="select * from etudiant where NCIN='".$_GET['reference']."'";
       $resultat = $dbh->query($requete1);
       $enreg = $resultat->fetchAll(PDO::FETCH_ASSOC);
       if (count($enreg)!=0){
```

```
       foreach($enreg as $rows){
        //Remplissage des renseignements sur l'étudiant en question
       ?>
      <div align="center">
            <h3>Modifier un étudiant</h3>
            <hr size="2" width="50%">
            <form name="ModifierEtudiant" method="POST" action="modif_Etudiant.php">
              <table border="1">
               <tr>
                 <td width="35%">Nom :</td>
                 <td width="65%"><input type="text" name="nom"
                    value="<?php echo $rows['NOM'];?>"
                                 size="25" maxlength="25">
                 </td>
               </tr>
               <tr>
                 <td width="35%">Prénom :</td>
                 <td width="65%"><input type="text" name="prenom"
                   value="<?php echo $rows['PRENOM'];?>"
                               size="25" maxlength="25">
                 </td>
              </tr>
              <tr>
                 <td width="35%">Sexe :</td>
                 <td width="65%">
         <select name="sexe">
        <?php if (strcmp($rows['SEXE'],"H")==0) {?>
            <option value="H" selected>Homme</option>
        <?php }else{?>
         <option value="H">Homme</option>
        <?php }?>
        <?php if (strcmp($rows['SEXE'],"F")==0) {?>
            <option value="F" selected>Femme</option>
        <?php }else{?>
         <option value="F">Femme</option>
        <?php }?>
                    </select>
                   </td>
               </tr>
      </table>
    <input type="hidden" name="ncin" value="<?php echo $rows['NCIN'];?>">
         <br>
         <input type="submit" value="Valider">  
         <input type="reset" value="Annuler">
       </form>
      <a href="lstEtudiants.php">Liste des étudiants</a>

     </div>
 <?php
    }} ?>
 <?php
  }else {
    //modification de l'étudiant
     $requete2="Update etudiant set NOM='".$_POST['nom'].
               "',PRENOM='".$_POST['prenom'].
               "',SEXE='".$_POST['sexe']."'
               where NCIN='".$_POST['ncin']."'";
  $dbh->exec($requete2);;
     ?>
       <div align="center">
        <br>Merci, l'étudiant <b><?php echo($_POST['nom']." ".$_POST['prenom']);?></b>

        <br>est bien été enregisré
        <br><br><a href="AjouterEtudiant.php">Retour</a>
       </div>
<?php
 }
?>
```

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

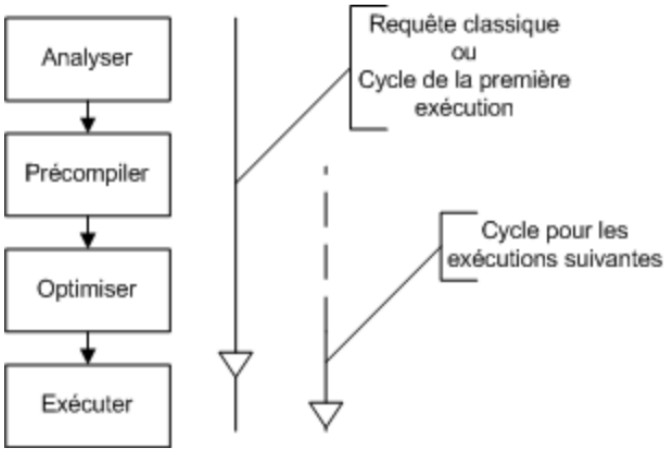
Suppression d'un enregistrement

Fichier : suppr_Etudiant.php

```
<!--
    fichier : suppr_Etudiant.php

-->
<html>
<head>
 <title>Suppression d'un étudiant</title>
</head>
<body>
 <?php
  //connexion
  include("connexion.php");
  //suppression de l'étudiant
  $requete="delete from etudiant where NCIN='".$_GET['reference']."'";
  $dbh->exec($requete);
  ?>
    <div align="center">
      <br>Merci, l'étudiant au n° C.I.N  <b><?php echo($_GET['reference']);?></b>
      <br>est bien été supprimé
 <br><br><a href="lstEtudiants.php">Retour</a>
    </div>
</body>
</html>
```

Les requêtes préparées

Le principe des requêtes préparées est de créer un modèle de requête et de l'enregistrer sur le SGBD, le temps de l'exécution du script (par opposition aux procédures stockées qui le sont de manière permanente). Quand vous aurez besoin de faire une requête, vous ferez appel à ce modèle. Le serveur exécutera alors votre requête, en utilisant les données que vous lui aurez fournies en paramètres pour construire la requête SQL réelle.



Comparaison Requête Classique et requête préparée

Les requêtes préparées sont recommandées pour :

•Les requêtes multiples : la requête n'est interprétée qu'une seule fois mais peut être exécutée plusieurs fois avec des paramètres identiques ou différents. Quand la requête est exécutée, la base de données va l'analyser, la compiler et l'optimiser. Pour de requêtes complexes, ces étapes peuvent prendre du temps et ralentir votre application si vous devez les répéter. En utilisant des requêtes préparées, vous éviterez de répéter le cycle d'analyse / compilation / optimisation.

• Protéger vos requêtes : les paramètres des requêtes préparées n'ont pas besoin d'être protégés, le pilote de votre base de données le fait tout seul. Vous vous protégez donc des attaques par injection SQL.

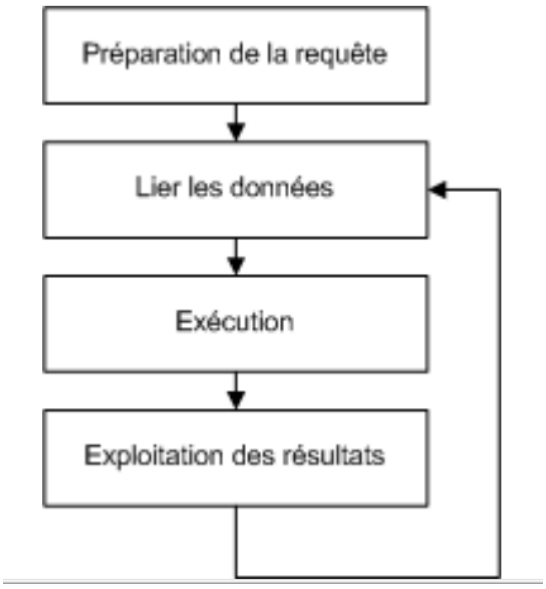
PDO émule les requêtes préparées

PDO émule les requêtes préparées si votre base de données ne dispose pas de cette fonctionnalité. Ainsi vous êtes assuré de pouvoir travailler de la même façon quel que soit le SGBD.

Attention ! Il n'est pas possible de préparer deux requêtes de manière parallèle. Vous devrez fermer la requête en cours pour en utiliser une nouvelle. Si vous souhaitez pouvoir exploiter deux résultats simultanément, vous devrez enregistrer les premiers résultats dans un tableau, puis exécuter ensuite la seconde requête.

D'un point de vue pratique, cela va correspondre aux étapes suivantes :

Principe des requêtes préparées



Désavantage des requêtes préparées :

•Si vous n'utilisez qu'une seule fois votre requête, son temps d'exécution sera très légèrement plus long. Par contre, l'avantage que vous en retirerez sera la sécurité du code.

Construction de la requête

Pour construire un modèle de requête, il suffit de remplacer chaque paramètre par un point d'interrogation ou par un paramètre nommé (par exemple « :nom »). Vous fournirez les paramètres à substituer quand vous exécuterez réellement la requête.

```
// Requête normale
$sql = "INSERT INTO article (titre, auteur)
        VALUES ('Titre super','auteur sympa')";
// Modèle de requête avec des paramètres nommés
$sql2 = 'INSERT INTO article (titre, auteur)
        VALUES ( :titre , :auteur)';
// Modèle de requête avec des points d'interrogations
$sql3 = 'INSERT INTO article (titre, auteur)
        VALUES ( ?, ?)';
```

Un des avantages notables de cette méthode est que vous êtes protégé des attaques dites par injection SQL (voir le chapitre sur la sécurité). Le SGBD sait ce qu'il s'attend à recevoir ; il vérifiera que les données transmises sont correctes et fera les échappements nécessaires.

Il faut choisir

Il n'est pas possible d'utiliser à la fois des noms de paramètres (:nom) et des points d'interrogation. Vous ne pouvez pas non plus utiliser le même nom de paramètre plusieurs fois.

```
$stmt = $dbh->prepare($sql);
$titre = 'Memento PHP MySQL';
$auteur = 'Ponçon' ;
$stmt->execute(array(':titre'=>$titre, ':auteur'=>$auteur));
?>
```

L'autre approche, plus pointue, consiste à associer distinctement chaque paramètre à une variable ou à une valeur :

```
bindParam ( parametre, &variable [, type [, taille ]] )
bindValue ( parametre, variable [, type [, taille ]] )
```

Dans le cas de la méthode bindParam(), on lie une variable à un paramètre. Ainsi, entre deux exécutions, il ne sera nécessaire que de changer la valeur de la variable.

Tableau des différents types

```
<?php
// Inclusion du fichier contenant la connexion à la base
```

```
include_once('connect.inc.php');

$sql = 'INSERT INTO article (titre, auteur)
   VALUES ( :titre , :auteur)';
$stmt = $dbh->prepare($sql);
$titre = 'Memento PHP MySQL';
$auteur = 'Ponçon' ;
$stmt->BindParam(':auteur',$auteur) ;
$stmt->BindParam(':titre',$titre) ;
$stmt->execute();
// Un premier enregistrement a été inséré
$titre = 'Best practices PHP 5' ;
$stmt->execute();
? >
```

Dans le cas de la méthode bindValue(), on associe une valeur à un paramètre. Il est important de noter cette distinction entre une valeur fixée à un moment donné et une référence qui lie un paramètre à une variable.

Dans l'exemple précédent, si nous avions utilisé la méthode bindValue() au lieu de bindParam(), le second appel à execute() aurait eu les mêmes conséquences que le premier. Et donc le changement de valeur de $titre n'aurait rien changé.

Exercice de TP 2

Reprendre l'exercice du TP 1 avec les requêtes préparées

Création de fichier de connexion

Fichier : connexion.php

```
<?php
define('USER1', 'root');
define('PASS1', '');
define('DSN1', 'mysql:host=localhost;dbname=ecole');
try {
  $dbh = new PDO(DSN1, USER1, PASS1);
```

```
} catch (PDOException  $e) {
  print "Erreur  ! : " . $e->getMessage()  . "<br/>";
  die();
}
?>
```

Ajout d'un enregistrement

Fichier : AjouterEtudiant.php

```
<!--
  fichier : AjouterEtudiants.php
  -->
 <html>
<head>
  <title>Saisie d'un nouveau étudiant</title>
  </head>
  <body>
 <?php
 if (!empty($_POST['NCIN'])
       AND !empty($_POST['NOM'])
      AND !empty($_POST['PRENOM']))
 {
 //connexion
 include("connexion1.php");
 //test d'existance de l'étudiant
 $requete1="select*from etudiant where NCIN=:NCIN ";
 $stmt=$dbh->prepare($requete1);
 $stmt->Bindparam(':NCIN',$_POST['NCIN']);
 $stmt->execute();
 $enreg1=$stmt->fetchAll();
       if(count($enreg1)!=0){
         //l'étudiant est déjà existant
            ?>
            <div align="center">
            <br>L'étudiant <b>
            <?php
            echo($_POST['NOM']." ".$_POST['PRENOM']);
            ?>
            </b> <br>est déjà existant!!!  <br><br><a
href="AjouterEtudiant.php">Retour</a>  </div>
            <?php
            }
            else
            {
      //insertion du nouveau étudiant

     $requete2="insert into etudiant values(:NCIN,:NOM,:PRENOM,:SEXE)";
     $stmt=$dbh->prepare($requete2);
     $stmt->Bindparam(':NCIN',$_POST['NCIN']);
 $stmt->Bindparam(':NOM',$_POST['NOM']);
 $stmt->Bindparam(':PRENOM',$_POST['PRENOM']);
 $stmt->Bindparam(':SEXE',$_POST['SEXE']);
 $stmt->execute();
```

```
                ?>
                 <br>Merci, l'étudiant <b>
                 <?php
                 echo($_POST['NOM']." ".$_POST['PRENOM']);
                 ?>
                 </b>   <br>est bien enregisré<br><br>
                 <a href="AjouterEtudiant.php">Retour</a>  </div>
                 <?php
                 }
                 //fin du test d'existance
                 }
                 else
                 {
                       ?>
                       <div align="center">
                       <h3>Saisie d'un nouveau étudiant</h3>
                       <hr size="2" width="50%">
                       <!--Ici c'est le Formulaire-->
                       <form name="SaisieEtudiant" method="POST"
action="AjouterEtudiant.php">
                       <table border="1">
                       <tr>
                 <td width="35%">N°.C.I.N :</td>
                 <td width="65%"><input type="text" name="NCIN"   value="" size="8"
maxlength="8"></td>         </tr>
                 <tr>           <td width="35%">NOM :</td>           <td
width="65%"><input type="text" name="NOM" value="" size="25" maxlength="25"></td>
                 </tr>

  <tr>
  <td width="35%">PRENOM:</td>
  <td width="65%"><input type="text" name="PRENOM" value="" size="25"
maxlength="25">
  </td>
  </tr>
  <tr>
  <td width="35%">SEXE:</td>
  <td width="65%"> <select name="SEXE"> <option value="H">Homme</option>   <option
value="F">Femme</option> </select>
  </td>
  </tr>
  </table>   <br>  <input type="submit" value="Valider">&amp;nbsp;&amp;nbsp;
  <input type="reset" value="Annuler">
  </form>
  <a href="lstEtudiants.php">Liste des étudiants</a>  </div>  <?php   }
  ?>
  </body>
  </html>
```

Affichage des enregistrements

Fichier : lstEtudiants.php

```
<!--
  fichier : lstEtudiants.php
  -->
  <html>
  <head>
  <title>Liste des étudiants </title>
  <meta charset='UTF-8'>
  </head>
  <body>
  <div align="Left">
```

```
   <h3>Liste des étudiants</h3>
  <hr width="50%" size="2">
  </div>
  <?php
        //connexion
  include("connexion1.php");
  ?>
  <?php
       //Liste des étudiants dont le sexe est Homme
  $requete_H="select * from etudiant where SEXE='H' order by NOM,PRENOM";
  $stmt= $dbh->prepare($requete_H);
  $stmt->execute();
  $enreg_H = $stmt->fetchAll(PDO::FETCH_ASSOC);
    if (count($enreg_H)!=0){
  ?>
  <div align="Left">
  <br>
  <h4>SEXE : <b>Homme</b></h4>
  <table border="1" cellpaccing="2" cellpadding="0">
  <tr>
  <td><b>N° C.I.N</b> </td>
  <td><b>NOM</b> </td>
  <td><b>PRENOM</b> </td>
  <td><b>SEXE</b> </td>
  <td><b>Modif/Suppr</b>  </td>
  </tr>
   <?php
  foreach($enreg_H as $rows){
   ?>
   <tr>
   <td><?php echo $rows["NCIN"];?></td>
   <td><?php echo $rows["NOM"];?></td>
   <td><?php echo $rows["PRENOM"];?></td>
   <td><?php echo $rows["SEXE"];?> </td>
      <td>
             <a href="modifEtudiants.php?reference=<?php echo
$rows["NCIN"];?>">Modifier</a>
             <a href="supprEtud.php?reference=<?php echo $rows["NCIN"];?>">Supprimer</a>
             </td>
    </tr>
<?php
  }
  ?>
  </table>
  </div>
<?php
 }else{
  ?>
<div align="Left">
<br>
 <h4>SEXE: <b>Homme</b></h4>
  <hr width="50%" size="2">
  <b> Pour le moment, il n'y a aucun étudiant enregistré!!!</b>
  <hr width="50%" size="2">
  </div>
       <?php
       }
       ?>
       <?php
        //Liste des étudiants dont le sexe est Femme
            $requete_F="select * from etudiant where SEXE='F' order by NOM,PRENOM";
             $stmt= $dbh->prepare($requete_F);
          $stmt->execute();
            $enreg_F = $stmt->fetchAll(PDO::FETCH_ASSOC);
            if (count($enreg_F)!=0){
            ?>
            <div align="Left">
            <br> <h4>SEXE: <b>Femme</b></h4>
```

```
            <table border="1" cellpaccing="2" cellpadding="0">
            <tr>
            <td><b>N° C.I.N</b></td>
            <td><b>NOM</b></td>
            <td><b>PRENOM</b></td>
            <td><b>SEXE</b></td>
            <td><b>Modif/Suppr</b></td>
            </tr>
            <?php
            Foreach($enreg_F as $rows) {
            ?>
            <tr>
            <td> <?php echo $rows["NCIN"];?></td>
            <td> <?php echo $rows["NOM"];?></td>
            <td> <?php echo $rows["PRENOM"];?></td>
            <td> <?php echo $rows["SEXE"];?></td>
                 <td>
            <a href="modifEtudiants.php?reference=<?php echo $rows["NCIN"];?>">
Modifier</a>
            <a href="supprEtud.php?reference=<?php echo $rows["NCIN"];?>"> Supprimer</a>
                        </td>
                  </tr>
 <?php
 }
 ?>
 </table>
 </div>
 <?php
 }else{
 ?>
 <div align="Left">
 <h4>SEXE : <b>Femme</b></h4>
 <hr width="50%" size="2">
 <b> Pour le moment, il n'y a aucune étudiante enregistrée!!!</b>
 <hr width="50%" size="2">
 </div>
<?php
   }
   ?>
 </body>
 </html>
```

Modification d'un enregistrement

Fichier : modif_Etudiant.php

```
<!--

fichier : modif_Etudiant.php

-->

<html>

<head>

    <title>Modification d'un étudiant</title>

                                                                    <meta charset='UTF-8'>

</head>

<body>

<?php

  //connexion
```

```
     include("connexion1.php");
if (empty($_POST['NCIN']) AND empty($_POST['NOM'])
     AND empty($_POST['PRENOM'])) {
      //Info. sur l'étudiant
   $requete1="select * from Etudiant where NCIN=:reference";
   $stmt=$dbh->prepare($requete1);
   $stmt->BindParam(':reference',$_GET['reference']);
   $stmt->execute();
   $enreg = $stmt->fetchAll(PDO::FETCH_ASSOC);
if (count($enreg)!=0){
foreach($enreg as $rows){
   //Remplissage des renseignements sur l'étudiant en question
?>
<div align="center">
<h3>Modifier un étudiant</h3>
<hr size="2" width="50%">
<form name="ModifierEtudiants" method="POST"
 action="modifEtudiants.php">
<table border="1">
<tr>
<td width="35%">NOM :</td>
<td width="65%"><input type="text" name="NOM"
value="<?php echo $rows['NOM'];?>"
size="25" maxlength="25">
</td>
</tr>
<tr>
<td width="35%">PRENOM :</td>
<td width="65%"><input type="text" name="PRENOM"
value="<?php echo $rows['PRENOM'];?>"
size="25" maxlength="25">
</td>
</tr>
<tr>
```

```
<td width="35%">SEXE :</td>
<td width="65%">
<select name="SEXE">
<?php if (strcmp($rows['SEXE'],"H")==0) {?>
<option value="H" selected>Homme</option>
<?php }else{?>
<option value="H">Homme</option>
<?php }?>
<?php if (strcmp($rows['SEXE'],"F")==0) {?>
<option value="F" selected>Femme</option>
<?php }else{?>
<option value="F">Femme</option>
<?php }?>
</select>
</td>
</tr>
</table>
<input type="hidden" name="NCIN" value="<?php echo $rows['NCIN'];?>">
<br>
<input type="submit" value="Valider">  
<input type="reset" value="Annuler">
</form>
<a href="lstEtudiants.php">Liste des étudiants</a>
</div>
<?php
}} ?>
<?php
}else {
//modification de l'étudiant
$requete2="Update etudiant set NOM=:nom,PRENOM=:prenom,SEXE=:sexe where
NCIN=:ncin";
$stmt=$dbh->prepare($requete2);
$stmt->BindParam(':nom',$_POST['NOM']);
$stmt->BindParam(':prenom',$_POST['PRENOM']);
$stmt->BindParam(':sexe',$_POST['SEXE']);
```

```
$stmt->BindParam(':ncin',$_POST['NCIN']);
$stmt->execute();
?>
<div align="center">
<br>Merci, l'étudiant <b><?php echo($_POST['NOM']." ".$_POST['PRENOM']);?></b>
<br>est bien été enregisré
<br><br><a href="AjouterEtudiant.php">Retour</a>
</div>
<?php
}
?>
</body>
</html>
```

Suppression d'un enregistrement

Fichier : suppr_Etudiant.php

```
<!--      fichier : suppr_Etudiant.php       -->
<html>
 <head>
 <title>Suppression d'un étudiant</title>
  <meta charset='UTF-8'>
 </head>
 <body>
 <?php
 //connexion
 include("connexion1.php");
 //suppression de l'étudiant
 $requete="delete from etudiant where NCIN=:reference";
 $stmt=$dbh->prepare($requete);
 $stmt->BindParam(':reference',$_GET['reference']);
 $stmt->execute();
 ?>
 <div align="center"><br>Merci, l'étudiant au n° C.I.N  <b>
 <?php
```

```
 echo($_GET['reference']);
 ?>
</b>
<br> a bien été supprimé  <br>
<br><a href="lstEtudiants.php">Retour</a>
  </div>
</body>
 </html>
```

Uploader des images sur un serveur web

Beaucoup d'applications web ont recours à des modules d'upload d'images. Parmi elles nous pouvons citer les galleries d'images ou bien encore les systèmes d'avatars de forums type PHPBB, VBulletin ou IPB (pour ne citer que les plus connus). Les programmes permettant ce genre de fonctionnalités peuvent-être parfois très évolués car ils se chargent de contrôler l'intégrité du fichier ou bien même de le redimensionner à la volée.

Le script qui suit montre le fonctionnement d'un système d'upload d'images sur le serveur Web. Ses principales caractéristiques sont les suivantes :

Il est entièrement configurable grâce aux constantes situées en tête du script.

Il dispose d'un tableau d'extensions d'image autorisées. Les extensions possibles sont celles acceptées par la fonction getimagesize() de PHP.

Il crée le répertoire cible s'il n'existe pas.

Il contrôle l'extension du fichier envoyé.

Il contrôle le type de l'image.

Il contrôle les dimensions de l'image (largeur et hauteur).

Il contrôle le poids de l'image.

Il upload l'image sur le serveur avec un nouveau nom.

Il retourne un message de réussite ou d'erreur à l'utilisateur.

Il est sécurisé.

Les informations sur les fichiers envoyés sont contenues dans le tableau global $_FILES.
$_FILES contient :

NOM	VARIABLE
Le nom	$_FILES['fichier']['name']
Le chemin du fichier temporaire	$_FILES['fichier']['tmp_name']
La taille (peu fiable, dépend du navigateur)	$_FILES['fichier']['size']
Le type MIME (peu fiable, dépend du navigateur)	$_FILES['fichier']['type']

Un code d'erreur si besoin	$_FILES['fichier']['error']

Exemple de code PHP

Le script complet

Script PHP de l'upload d'image

```
<?php


/**************************************************************
 * Definition des constantes / tableaux et variables
 *************************************************************/


// Constantes
define('TARGET', 'files/');     // Repertoire cible
define('MAX_SIZE', 100000);    // Taille max en octets du fichier
define('WIDTH_MAX', 800);    // Largeur max de l'image en pixels
define('HEIGHT_MAX', 800);    // Hauteur max de l'image en pixels
// Tableaux de donnees
$tabExt = array('jpg','gif','png','jpeg');     // Extensions autorisees
$infosImg = array();
// Variables
$extension = '';
$message = '';
$nomImage = '';
/**************************************************************
 * Creation du repertoire cible si inexistant
 *************************************************************/
if( !is_dir(TARGET) ) {
 if( !mkdir(TARGET, 0755) ) {
   exit('Erreur : le répertoire cible ne peut-être créé ! Vérifiez que vous diposiez des droits suffisants pour le faire ou créez le manuellement !');
 }
}
/**************************************************************
```

```
 * Script d'upload
 **************************************************************/
if(!empty($_POST))
{
  // On verifie  si le champ  est rempli
  if(  !empty($_FILES['fichier']['name'])  )
  {
    // Recuperation  de l'extension  du fichier
    $extension   = pathinfo($_FILES['fichier']['name'],  PATHINFO_EXTENSION);
    // On verifie  l'extension  du fichier
    if(in_array(strtolower($extension),$tabExt))
    {
      // On recupere  les dimensions  du fichier
      $infosImg  = getimagesize($_FILES['fichier']['tmp_name']);

      // On verifie  le type de l'image
      if($infosImg[2]  >= 1 && $infosImg[2]  <= 14)
      {
        // On verifie  les dimensions  et taille  de l'image
        if(($infosImg[0]  <= WIDTH_MAX)  && ($infosImg[1]  <= HEIGHT_MAX)  &&
(filesize($_FILES['fichier']['tmp_name'])  <= MAX_SIZE))
        {
          // Parcours du tableau  d'erreurs
          if(isset($_FILES['fichier']['error'])
            && UPLOAD_ERR_OK  === $_FILES['fichier']['error'])
          {
            // On renomme  le fichier
            $nomImage  = md5(uniqid())  .'.'. $extension;

            // Si c'est OK, on teste l'upload
            if(move_uploaded_file($_FILES['fichier']['tmp_name'],  TARGET.$nomImage))
            {
              $message  = 'Upload réussi  !';
```

```
        }
        else
        {
          // Sinon on affiche une erreur systeme
          $message = 'Problème lors de l\'upload !';
        }
       }
       else
       {
        $message = 'Une erreur interne a empêché l\'uplaod de l\'image';
       }
     }
     else
     {
       // Sinon erreur sur les dimensions et taille de l'image
       $message = 'Erreur dans les dimensions de l\'image !';
     }
   }
   else
   {
     // Sinon erreur sur le type de l'image
     $message = 'Le fichier à uploader n\'est pas une image !';
   }
  }
  else
  {
   // Sinon on affiche une erreur pour l'extension
   $message = 'L\'extension du fichier est incorrecte !';
  }
}
else
{
```

```
    // Sinon on affiche une erreur pour le champ vide
    $message = 'Veuillez remplir le formulaire svp !';
  }
}
?>
```

Le formulaire HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
  <head>
    <title>Upload d'une image sur le serveur !</title>
  </head>
  <body>
 <?php
      if( !empty($message) )
      {
        echo '<p>',"\n";
        echo "\t\t<strong>", htmlspecialchars($message) ,"</strong>\n";
        echo "\t</p>\n\n";
      }
    ?>
    <!-- Debut du formulaire -->
   <form enctype="multipart/form-data" action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>"
method="post">
    <fieldset>
        <legend>Formulaire</legend>
          <p>
            <label for="fichier_a_uploader" title="Recherchez le fichier à uploader !">Envoyer le fichier :</label>
            <input type="hidden" name="MAX_FILE_SIZE" value="<?php echo MAX_SIZE; ?>" />
            <input name="fichier" type="file" id="fichier_a_uploader" />
            <input type="submit" name="submit" value="Uploader" />
          </p>
      </fieldset>
```

```
    </form>

    <!-- Fin du formulaire  -->

  </body>

</html>
```

Utilisation de case à cocher

Exemple de code

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Exploitation facile des cases à cocher avec PHP</title>
</head>

 <body>
 <form action="checkbox.php">
 <input type="hidden" name="envoi" value="yes">
 <input type="text" name="voiture"><br>
 <input type="checkbox" name="options[]" value="Injection au méthane"> Injection au méthane<br>
  <input type="checkbox" name="options[]" value="Trois roues de secours"> Trois roues de secours<br>
 <input type="checkbox" name="options[]" value="Miroir de courtoisie dans le coffre"> Miroir de
courtoisie dans le coffre<br>
 <input type="checkbox" name="options[]" value="Ventilation des rotules (côté conducteur)"> Ventilation
des rotules côtés conducteur)<br>
 <input type="checkbox" name="options[]" value="Kit James-Bond "> Kit James-Bond <br>
 <input type="submit">
 </form>

    <?php
        $envoi = $_GET['envoi'];              //aiguilleur
        $voiture = $_GET['voiture'];          //Nom de la voiture
        $options = $_GET['options'];          //Contenu des cases à cocher

           if ($envoi == 'yes') {
            $options_text = implode(', ',$options);

            echo '<h1>L\'auto de vos rêves &quot;'.$voiture.'&quot;:</h1>';
            echo '<p>options:<br><br>'.$options_text.'</p>';
     }
?>
</body>
</html>
```

Exercice de TP 3
Reprendre l'exercice du TP 2 en y ajoutant :
- l'enregistrement d'images (photos dans notre cas).
- Une rubrique Loisirs représentée par des cases à cocher.
Conversion de page en PDF

HTML2PDF est un convertisseur de code HTML vers PDF écrit en PHP.

Il permet la conversion d'HTML valide au format PDF, et est distribué sous licence LGPL.

Cette librairie a été conçue pour gérer principalement les TABLE imbriquées afin de générer des factures, bons de livraison, et autres documents officiels.

- Téléchargement

Pour télécharger le fichier HTML2PDF, il vous suffit d'aller sur ce lien : ici et cliquer sur le bouton « Télécharger le programme ». Une fois le téléchargement terminé, décompressez le ficher .zip et récupérez le contenu.

- Mise en place de l'environnement

Dans cette démonstration, vous allez créer un fichier PDF contenant un simple tableau.Pour commencer, vous devez simplement créer un fichier PHP et insérer le code ci-dessous :

```
<?php

ob_start();

// Contenu HTML qui sera converti en PDF

$content = '';

// Inclure le fichier Html2pdf

require_once('html2pdf/html2pdf.class.php');

// Initialisation de la classe HTML2PDF()

$html2pdf = new HTML2PDF('P', 'A4', 'fr');

// Insérer le contenu HTML

$html2pdf->WriteHTML($content);

// Retour du format PDF ayant comme nom de fichier : 'exemple.pdf'

$html2pdf->Output('exemple.pdf');

?>
```

Ne pas oublier de bien inclure le dossier html2pdf.

L'environnement est maintenant en place. Le code HTML à convertir sera contenu dans la variable $content.

Exemple de code

Le code HTML

Vous pouvez désormais insérer votre tableau HTML dans une balise <page> comme ci-dessous :

```
<?php


ob_start();


// Contenu HTML qui sera converti en PDF

$content = '
```

```
<page>
<table cellspacing="10" border="1" style="width: 100%; text-align: left;font-size: 14px;">
<tr>
<td style="width:20%; text-align:center; color:blue"><b>Id</b></td>
<td style="width:50%; text-align:center; color:red"><b>Nom et prénom</b></td>
<td style="width:10%; text-align:center; color:green"><b>Date de naissance</b></td>
<td style="width:20%; text-align:center; color:purple"><b>Lieu</b></td>
</tr>

<tr>
<td>1</td>
<td>Dupont Nicolas</td>
<td>16/06/1989</td>
<td>France</td>
</tr>
<tr>
<td>2</td>
<td>Lemaitre Jacques</td>
<td>25/12/1990</td>
<td>France</td>
</tr>
<tr>
<td>3</td>
<td>Rodriguez Marc</td>
<td>01/05/1991</td>
<td>Espagne</td>
</tr>
</table>
</page>

';
```

```
// Inclure le fichier Html2pdf
require_once('html2pdf/html2pdf.class.php');

// Initialisation de la classe HTML2PDF()
$html2pdf = new HTML2PDF('P', 'A4', 'fr');

// Insérer le contenu HTML
$html2pdf->WriteHTML($content);

// Retour du format PDF ayant comme nom de fichier : 'exemple.pdf'
$html2pdf->Output('exemple.pdf');
?>
```

Id	Nom et prénom	Date de naissance	Lieu
1	Dupont Nicolas	16/06/1989	France
2	Lemaitre Jacques	25/12/1990	France
3	Rodriguez Marc	01/05/1991	Espagne

A savoir : La librairie HTML2PDF a été faite pour faciliter la création de fichiers PDF, non pour convertir directement une page HTML. Vous ne pouvez pas utiliser les balises <html>, <head>, <body>.

Le fichier PDF

C'est bientôt fini ! Il ne vous reste plus qu'à générer le fichier PDF depuis un navigateur web, en indiquant le chemin de votre fichier PHP.

Conclusion

Vous avez donc bien généré votre fichier PDF et converti votre tableau HTML.

Il est possible de modifier la taille des tableaux afin d'obtenir la mise en page souhaitée. Et également, de rajouter un pied de page et/ou un en-tête.

CHAPITRE VI : LES SESSIONS ET LES COOKIES

LES SESSIONS

Présentation des sessions PHP

En effet, afin de transmettre des variables de pages en pages, plusieurs possibilités s'offrent à vous :
- les divers champs des formulaires, qu'ils soient **hidden** ou non.
- passer les variables directement à travers les **liens**.
- utiliser les **cookies**.
- utiliser les **sessions**.

Cependant, toutes ces possibilités n'offrent pas le même niveau de sécurité.

En effet, certaines de ces possibilités sont vraiment pratiquent dans leurs modes d'utilisation (comme les cookies par exemple mais tout le monde n'est pas obligé d'accepter les cookies), ce qui implique, dans la majorité des cas, un bas niveau de sécurité (cas du passage des variables par les liens, ce qui implique que les variables seront visibles de tout le monde).

De même, faire dans chaque page un formulaire contenant des champs hidden permettant de faire circuler les différentes variables à travers toutes les pages du site n'est pas vraiment pratique.

C'est pourquoi, dans tous ces cas où la sécurité de vos données est primordiale, vous devrez utiliser **les sessions** qui vous permettront de faire circuler différentes variables (comme un mot de passe par exemple) à travers les pages de votre site, tout en étant assez confortables à l'emploi.

Pour utiliser les sessions, différentes fonctions PHP s'offrent à nous. Voici déjà un petit tableau vous permettant de vous familiariser avec ces différentes fonctions (que nous détaillerons bien sur dans la suite de ce cours) :

Fonction	Signification
session_start	Démarre une session
session_register	Enregistre une variable de session
session_unregister	Efface une variable de session
session_is_registered	Vérifie si une variable est déclarée pour la session en cours
session_id	Retourne l'id de la session en cours
session_name	Retourne le nom de la session en cours
session_unset	Detruit toutes les variables de la session en cours
session_destroy	Destruit la session en cours

Sachez également qu'ils existent d'autres fonctions agissant sur les sessions.
En revanche, étant donné qu'elles ne sont pas nécessaires à la compréhension de notre cours, nous ne les détaillerons pas ici (cf. la documentation pour de plus amples informations).
Cependant, ces fonctions commencent toujours par **session**.

Avant d'aller plus loin dans le cours, vous devez savoir que les sessions ne sont accessibles qu'à partir de **PHP 4**.Cependant la plupart des hébergeurs sont aujourd'hui fait évoluer leur PHP en PHP 4.

Afin de voir concrètement comment fonctionnent les **sessions**, prenons alors un exemple simple :
- imaginons que notre site possède une **section membre** où chaque membre devra se logué avant de pouvoir y entrer.
- de plus, on aimerait bien être sur qu'il s'agisse toujours de ce **même membre qui est connecté**.

On aura alors une page contenant un formulaire permettant à notre visiteur de se connecter à une section membre (page **index.htm**) :

exemple1

```
<html>
<head>
<title>Formulaire d'identification</title>
</head>

<body>
<form action="login.php" method="post">
Votre login : <input type="text" name="login">
<br />
Votre mot de passé : <input type="password" name="pwd"><br />
<input type="submit" value="Connexion">
</form>

</body>
</html>
```

D'après cette page, vous pouvez remarquer que lorsque le visiteur le remplira et qu'il cliquera sur le bouton de connexion, on se retrouvera au niveau de la page **login.php** avec une variable **$pseudo** qui contiendra le login de notre visiteur ainsi qu'une variable **$pwd** contenant son mot de passe ; variables qu'il faudra naturellement tester avant de démarrer notre session (car seuls les membres pourront accéder à notre espace membre, espace où l'on utilisera notre session).

On aura alors par exemple (page **login.php**) :

exemple2

```
<?php
// On définit un login et un mot de passe de base pour tester notre exemple. Cependant, vous pouvez très bien interroger votre base de données afin de savoir si le visiteur qui se connecte est bien membre de votre site
$login_valide = "moi";
```

```
$pwd_valide  = "lemien";

// on teste si nos variables  sont définies
if (isset($_POST['login'])  && isset($_POST['pwd']))  {

// on vérifie  les informations  du formulaire,  à savoir  si le pseudo saisi est bien un pseudo autorisé,  de même pour le  mot de passe
        if ($login_valide  == $_POST['login']  && $pwd_valide  == $_POST['pwd']) {
// dans ce cas, tout est ok, on peut démarrer  notre session

                // on la démarre  :)
                session_start  ();
                // on enregistre  les paramètres  de notre visiteur  comme variables  de session ($login  et $pwd) (notez bien  que l'on utilise  pas le $ pour enregistrer  ces variables)
                $_SESSION['login']  = $_POST['login'];
                $_SESSION['pwd'] = $_POST['pwd'];

                // on redirige  notre visiteur  vers une page de notre section membre header ('location:
                page_membre.php');
        }
        else {
            // Le visiteur  n'a pas été reconnu  comme  étant  membre  de notre site.  On utilise alors un petit javascript  lui  signalant  ce fait
            echo '<body onLoad="alert(\'Membre  non reconnu...\')">';
            // puis  on le redirige  vers la page d'accueil
            echo '<meta http-equiv="refresh"  content="0;URL=index.htm">';
        }
}
else {
        echo 'Les variables  du formulaire  ne sont pas déclarées.';
}
```

```
?>
```

Remarquer également que nous utilisons notre **session_start** avant tout code HTML.

Voyons alors le code de la page de notre section membre, la page **page_membre.php**. On a :

---

exemple3

```
<?php
// On démarre la session (ceci est indispensable dans toutes les pages de notre section membre)
session_start ();

// On récupère nos variables de session
if (isset($_SESSION['login']) && isset($_SESSION['pwd'])) {

        // On teste pour voir si nos variables ont bien été enregistrées echo '<html>';
        echo '<head>';
        echo '<title>Page de notre section membre</title>'; echo
        '</head>';

        echo '<body>';
        echo 'Votre login est '.$_SESSION['login'].' et votre mot de passe est '.$_SESSION['pwd'].'.';
        echo '<br />';

        // On affiche un lien pour fermer notre session echo '<a
        href="./logout.php">Déconnection</a>';
}
else {
        echo 'Les variables ne sont pas déclarées.';
}
```

```
?>
```

Voyons alors le code de la page permettant au membre de se déconnecter (la page **logout.php**). On aura alors :

exemple4

```
<?php
// On démarre la session
session_start ();

// On détruit les variables de notre session
session_unset ();

// On détruit notre session
session_destroy ();

// On redirige le visiteur vers la page d'accueil header
```

```
('location:  index.htm');
```

```
?>
```

Résumons alors tout ce que nous venons de voir :

- chaque session à un **id différent** (ce qui permet d'éviter la confusion entre les connexions).
- à chaque page où notre session doit être active, on doit placer un **session_start** en tout début de page (**avant tout code HTML**).
- toutes les **variables enregistrées** au cours de notre session, seront **accessibles dans les pages de notre session**.
- n'oubliez **JAMAIS de détruire vos variables de session** lors de la déconnexion.

En respectant ces règles vous pourrez très rapidement faire vous-même votre espace membre, voir même pourquoi pas une boutique en ligne.

LES COOKIES

En effet, beaucoup de personnes d'imaginent que les cookies sont des petites bêtes malveillantes (**alors que c'est totalement faux**), tout simplement parce qu'elles ne savent pas exactement de quoi il s'agit.

Un cookie est un petit **fichier texte** (faisant au maximum 65 Ko) stocké sur le disque dur du visiteur du site.
Ce fichier texte permet de **sauvegarder** diverses informations concernant ce visiteur **afin de pouvoir les réutiliser** (les informations) lors de la prochaine visite du visiteur sur ce même site.
Par exemple, on pourrait très bien stocker dans ce cookie le **nom du visiteur** et par la suite, **afficher son nom** à chaque fois qu'il se connectera sur le site (ceci bien sur, s'il n'efface pas les cookies de son disque dur).
Cependant, tout cela n'arrive pas par le saint esprit.

En effet, **ceci n'est possible que si le visiteur à entré lui-même ses informations dans un formulaire sur le site**.

Les cookies sont stockés, selon votre navigateur Internet, à un certain endroit de votre disque dur. Par exemple, avec un système composé de **Windows** et du navigateur **INTERNET EXPLORER** (le plus usité).
Dans cette configuration, les cookies sont stockés dans le répertoire **C:WindowsTemporary Internet Files** comme ci-dessous :

Voyons à présent comment créer de tels cookies, grâce à la fonction **setcookie()**.Soit alors la portion de code suivante :

---

exemple1

```
<?php
```

```
// on définit une durée de vie de notre cookie (en secondes), donc un an dans notre cas
$temps = 365*24*3600;

// on envoie un cookie de nom pseudo portant la valeur LA GLOBULE
setcookie ("pseudo", "LA GLOBULE", time() + $temps);
```

```
?>
```

Explications :

Grâce à ce code, nous venons d'envoyer, chez le client (donc le visiteur du site) un cookie de nom **pseudo** portant la valeur **LA GLOBULE**.
De plus, **time()** retournant le nombre de secondes écoulées depuis le 1er janvier 1970 jusqu'à l'instant présent, nous imposons que le cookie ai une durée de vie de un an (soit en fait l'instant présent plus un an, donc un an).

Enfin, maintenant, si le visiteur ne supprime pas ce cookie, et bien, dans toutes les pages WEB de notre site, on pourra accéder à la variable **$pseudo** qui contiendra la chaîne de caractères **LA GLOBULE**.

En revanche, l'envoie d'un cookie ayant la même valeur pour tous les visiteurs d'un site, ce n'est pas vraiment intéressant.

Supposons alors que sur une page de notre site WEB, nous souhaitons faire en sorte que si le visiteur vient pour la première fois (ou qu'il a supprimer ses cookies), et bien, il aurait alors, la possibilité de saisir son nom dans un formulaire, ou bien s'il ne s'agit pas de sa première visite, d'afficher tout simplement Bonjour puis son nom.

On aurait alors le code suivant pour notre page (par exemple **index.php**) :

---

exemple2

```
<html>
<head>
<title>Index du site</title>
<body>

<?
// on teste la déclaration de notre cookie
```

```
if (isset($_COOKIE['pseudo'])) {

        echo 'Bonjour '.$_COOKIE['pseudo'].' !';

        // si le cookie n'existe pas, on affiche un formulaire permettant au visiteur de
saisir son nom
        echo '<form action="./traitement.php" method="post">';
        echo 'Votre nom : <input type = "texte" name =
        "nom"><br />';echo '<input type = "submit" value =
        "Envoyer">';

}
else {
        echo 'Notre cookie n\'est pas déclaré.';
}
?>

</body>
```

```
</html>
```

Et le code pour la page **traitement.php** :

---

exemple3

```
<?php
If (isset($_POST['nom'])) {
        // on définit une durée de vie de notre cookie (en secondes), donc un an
dansnotre cas
        $temps = 365*24*3600;

        // on envoie un cookie de nom pseudo portant la valeur de la variable
$nom, c'est-à-dire la valeur qu'a saisi la personne qui a rempli le formulaire
        setcookie ("pseudo", $_POST['nom'], time() + $temps);

        // fonction nous permettant de faire des
        redirections
        function redirection($url){
            if (headers_sent()){
                print('<meta http-equiv="refresh" content="0;URL='.$url.'">');
            }
            else {
```

```
                header("Location: $url");
            }
        }

        // on effectue une redirection vers la page
        d'accueil
        redirection ('index.php');
}
else {
        echo 'La variable du formulaire n\'est pas déclarée.';
}
```

```
?>
```

Plusieurs conditions sont à respecter afin que l'utilisation des cookies se passe au mieux :
- l'envoie d'un cookie doit être la **première fonction PHP** que vous utilisez dans votre script, ce qui veut dire que vous devez utiliser la fonction **setcookie()** tout en haut de votre script (**AUCUN AFFICHAGE ET AUCUN CODE CODE HTML AVANT UN SETCOOKIE**). Si d'autres fonctions interviennent avant l'envoie du cookie, celui-ci ne fonctionnera pas.
- Si vous envoyé un cookie sur un poste client celui-ci **effacera automatiquement l'ancien cookie qui portait le même nom** (si il y en avait un), autrement il le créera.

Note : Pour effacer un cookie, vous devez lancer un cookie qui aura le même nom que le cookie que vous voulez effacer, tout en lui donnant une valeur nulle (vous pouvez également l'envoyer avec un temps de vie dépassé).