

# Initiation à la programmation de page web en



## TABLE DES MATIERES

<b>1. PRE-REQUIS.....</b>	<b>5</b>
<b>2. INTRODUCTION.....</b>	<b>6</b>
Qu'est-ce que PHP ?.....	6
Historique .....	6
A qui s'adresse PHP ? .....	6
Comment ça marche .....	7
La nouvelle génération du Web : les sites Web actifs .....	7
Technologies dynamiques côté client .....	7
Technologies côté serveur .....	7
De l'intérêt du traitement côté serveur .....	7
Architecture .....	8
<b>3. INSTALLATION.....</b>	<b>10</b>
<b>3.1. Installation manuel.....</b>	<b>10</b>
Installation de PHP.....	10
Configuration de PHP.....	10
Installation d'Apache .....	11
Configuration d'Apache .....	11
Lancement et arrêt du serveur .....	12
Test de la configuration .....	12
Interfaçage avec un SGBD .....	12
Quelle base de données ? .....	13
<b>3.2. Installation rapide avec EasyPHP.....</b>	<b>14</b>
Présentation d' EasyPHP .....	14
Installation d' EasyPHP .....	14
<b>3.3. Installation du support de cours .....</b>	<b>15</b>
<b>4. LANGAGE PHP .....</b>	<b>16</b>
<b>4.1. Quelques règles de base .....</b>	<b>16</b>
<b>4.2. Les variable PHP .....</b>	<b>18</b>
Type de variables .....	18
Les constantes .....	18
Variable type Booléens : .....	19
Variable type entier : .....	19
Les nombres à virgule flottante : .....	19
Les variable type chaînes de caractères : .....	21
Traitement des variables dans les chaînes.....	22
Les variable type tableaux .....	23
Les variable type tableaux .....	23
Les variable type objet .....	23
Le transtypage.....	24
les variables super global.....	26
<b>4.3. Les opérateurs.....</b>	<b>27</b>
Les opérateurs arithmétiques.....	27
L'opérateur d'affectation.....	27
L'opérateur de concaténation.....	27
L'opérateur arobase .....	27
Les opérateurs de comparaison.....	28
Les opérateurs logiques.....	28
les opérateurs d'incrémentation .....	28
Raccourcis d'affectation de variables.....	28
<b>4.4. Instructions .....</b>	<b>29</b>
Instruction conditionnelles.....	29
Branchement conditionnel : mots clé « else » et « elseif » .....	29
Instruction « switch ».....	32
Les boucles .....	32
<b>4.5. les Tableaux .....</b>	<b>34</b>
Fonction de traitement de tableaux.....	35
Parcourir les tableaux avec les boucles.....	35
Tri de tableaux.....	39
<b>4.6. Fonctions .....</b>	<b>41</b>

Les fonction prédéfinis .....	41
Les fonctions utilisateurs .....	41
Portée des variables.....	43
Fonction imbriquer.....	43
Les inclusions .....	45
<b>5. LES FORMULAIRES .....</b>	<b>46</b>
La méthode GET : .....	46
La méthode POST:.....	46
Urlencode() et Urldecode() .....	48
Recupération des variables.....	49
Les contrôles HTML .....	50
Validation des formulaires.....	53
Htmlelntites() et htmlspecialchars .....	54
Les en-têtes HTTP .....	55
Récupérer les en-têtes de la requête.....	55
<b>6. BASE DE DONNEES .....</b>	<b>56</b>
Pourquoi utiliser une base données ? .....	56
Architecture externe d'une base de données web .....	56
<b>6.1. Accès aux bases de données.....</b>	<b>56</b>
Support PHP de connexion aux bases de données .....	56
Rudiments du langage SQL .....	56
PhpMyadmin .....	57
Installer la base formation .....	57
<b>6.2. Base de données MySQL.....</b>	<b>59</b>
Etablir une connexion.....	59
Choisir une base de données existante .....	59
Déconnexion à la base de données.....	60
Envoyer une requête vers la base de données .....	61
Rechercher des données dans la base.....	64
Naviguer dans les données de la base .....	66
Insérer des données dans la base .....	68
addslashes() et stripslashes () .....	68
Modifier des données dans la base .....	68
Supprimer des données dans la base.....	70
Les dates et heures.....	70
Conversion entre des formats de date PHP et des formats de date MySQL.....	73
<b>6.3. Utiliser ODBC.....</b>	<b>75</b>
Installation du driver odbc de MySql.....	75
Configuration de la source de donnée.....	75
<b>7. LES SESSIONS ET COOKIES .....</b>	<b>78</b>
<b>7.1. Les Cookies .....</b>	<b>78</b>
Qu'est-ce qu'un cookie ? .....	78
Cookies en PHP.....	78
Accéder à un cookie.....	78
Supprimer un cookie .....	78
<b>7.2. Les sessions .....</b>	<b>81</b>
Les sessions avec cookies.....	81
Passage des paramètres par l'URL .....	81
Les sessions utilisant les fonctions PHP .....	82
Configuration de php.ini .....	82
Ouverture d'une session .....	83
Enregistrement d'une variable de session .....	83
Utilisation de variables de session .....	83
Dés-enregistrement de variables et suppression de session .....	83
<b>8. GESTION DES FICHIERS .....</b>	<b>86</b>
Ouvrir des fichiers .....	86
Fermer des fichiers.....	87
Naviguer dans des fichiers .....	87
Afficher des fichiers.....	87
Lire des fichiers .....	87
Ecrire dans des fichiers .....	88

Copier, supprimer et renommer des fichiers .....	88
Chargements de tableaux à partir de fichiers .....	90
Chargements de tableaux à partir de fichiers .....	90
Gérer des répertoires .....	91
Afficher une image .....	93
<b>9. GESTION DES ERREURS.....</b>	<b>95</b>
Introduction.....	95
<b>9.1. Les types d'erreurs PHP .....</b>	<b>95</b>
Erreur de syntaxe .....	95
Erreur de sémantique.....	95
Erreur logique .....	96
Erreur d'environnement.....	96
Les messages d'erreur PHP .....	96
<b>9.2. Les niveaux d'erreur PHP .....</b>	<b>96</b>
Définition du niveau de rapport d'erreur.....	96
Définition d'un niveau de rapport d'erreur dans le « <i>php.ini</i> » .....	97
Définition du niveau de rapport d'erreur dans le script : .....	97
<b>9.3. Gérer les erreurs.....</b>	<b>97</b>
Suppression des messages d'erreur.....	97
Message d'erreur personnalisé.....	98
Journalisation des erreurs.....	99
<b>10. PROGRAMMATION ORIENTE OBJET .....</b>	<b>100</b>
<b>10.1. Quelques concepts de programmation orientée objet.....</b>	<b>100</b>
<b>11. LES CLASSES ET LES OBJETS (PHP 4).....</b>	<b>101</b>
<b>11.1. Création d'une classe.....</b>	<b>101</b>
Création d'attributs : .....	101
Création de méthodes : .....	101
Constructeur de classe.....	101
<b>11.2. Instanciation .....</b>	<b>102</b>
<b>11.3. Accès aux attributs et aux méthodes d'une classe.....</b>	<b>102</b>
Les accesseurs .....	102
<b>11.4. Héritage.....</b>	<b>103</b>
L'opérateur de contexte de classe (::).....	104
<b>11.5. Redéfinition.....</b>	<b>105</b>
Manipulation de classes et d'objets d'origine similaire. ....	105
<b>LES CLASSES ET LES OBJETS (PHP 5).....</b>	<b>107</b>
Quelques exemples de changements : .....	107
<b>11.6. Création d'une classe.....</b>	<b>107</b>
Constructeurs et destructeurs .....	108
<b>11.7. Création d'une instance.....</b>	<b>109</b>
<b>11.8. Assignment d'un objet.....</b>	<b>109</b>
<b>11.9. Héritage.....</b>	<b>109</b>
<b>11.10. Visibilité.....</b>	<b>109</b>
Visibilité des membres .....	110
Visibilité des méthodes.....	110
L'opérateur de résolution de portée (::).....	111
<b>11.11. Abstraction d'objets .....</b>	<b>112</b>
<b>11.12. Interfaces.....</b>	<b>113</b>

## 1. PRE-REQUIS

Pour être suivi avec efficacité, certaines connaissances seront utiles à ce cours :

Le langage HTML :	PHP permet de faire des manipulations balise par balise.
Les Bases de données :	les notions de base comme la création de base de données, des tables, des types de données, etc...
Langage Sql :	L'interrogation (select), l'insertion (insert), la modification (update), etc...
Langage JavaScript:	Création de fonction, manipulation des chaînes, etc...
Les Réseaux :	Fonctionnement générale, Structure client/serveur etc...

Ce cours s'adresse aux débutants ayant déjà programmé dans un langage simple ou à ceux désirant acquérir les notions de base de la programmation PHP. Si vous avez déjà pratiqué JavaScript, vous constaterez que PHP présente avec ce langage de script des similitudes et n'est guère plus compliqué.

A moins de gérer son propre serveur (ce qui n'est réaliste que dans le cas d'un intranet !), vous êtes obligé de subir ce que votre hébergeur vous impose et comme il n'y a pas un langage PHP pur et dur comme c'est le cas dans d'autres langages normalisés, mais autant de langage PHP que ses multiples extensions le permet, et que PHP gère HTML balise par balise au niveau le plus bas, vous allez devoir mettre les mains dans le cambouis.

Nous resterons dans le cadre de la réalisation de pages Web simples et nous n'aborderons pas les techniques sophistiquées faisant appel au multimédia. Pas plus que nous montrerons comment appliquer PHP au commerce électronique. Nous laisserons également dans l'ombre les questions d'hébergement payant de sites Web et tout ce qui s'y rattache (acquisition de nom de domaine, par exemple). Nous essayerons de réaliser une application de gestion simple.

## 2. Introduction

### **Qu'est-ce que PHP ?**

PHP est un langage de script imbriqué ou non dans les pages HTML et traité par le serveur. PHP permet de construire dynamiquement des pages HTML contenant les résultats de calculs ou de requêtes SQL adressées à un système de gestion de bases de données (SGBD).

Grâce à de nombreuses extensions, PHP peut générer des fichiers PDF, s'interfacer avec des serveurs de messagerie, des serveurs LDAP ou encore générer des images et graphiques GIF à la volée, pour ne citer que quelques-unes des fonctionnalités les plus importantes. PHP peut aussi s'interfacer avec la quasi-totalité des SGBD du marché, qu'ils soient commerciaux ou qu'ils viennent du monde libre (Free Software).

PHP est un langage de script au niveau serveur, comparable à ASP de Microsoft mais qui comporte beaucoup plus de fonctions, supporte pratiquement tous les standards du Web et est extensible.

### **Historique**

Le langage PHP a été mis au point au début d'automne 1994 par Rasmus Lerdorf. Ce langage de script lui permettait de conserver la trace des utilisateurs venant consulter son CV sur son site, grâce à l'accès à une base de données par l'intermédiaire de requêtes SQL. Ainsi, étant donné que de nombreux internautes lui demandèrent ce programme, Rasmus Lerdorf mit en ligne en 1995 la première version de ce programme qu'il baptisa Personal Sommaire Page Tools (outils basiques pour les pages perso), puis Personal Home Page v1.0 (traduisez page personnelle version 1.0).

Étant donné le succès de PHP 1.0, Rasmus Lerdorf décida d'améliorer ce langage en y intégrant des structures plus avancées telles que des boucles, des structures conditionnelles, et y intégra un package permettant d'interpréter les formulaires qu'il avait développé (FI, Form Interpreter) ainsi que le support de mSQL. C'est de cette façon que la version 2 du langage, baptisée pour l'occasion PHP/FI version 2, vit le jour durant l'été 1995. PHP 2 fut rapidement utilisé sur de nombreux sites (15000 fin 1996, puis 50000 en milieu d'année 1997).

A partir de 1997, Zeev Suraski et Andi Gurmans rejoignirent Rasmus pour former une équipe de programmeurs afin de mettre au point PHP 3 (Stig Bakken, Shane Caraveo et Jim Winstead les rejoignèrent par la suite). C'est ainsi que la version 3.0 de PHP fut disponible le 6 juin 1998 et s'appelle désormais PHP : Hypertext Preprocessor.

Il existe par ailleurs des applications web prêtes à l'emploi (PHPNuke, PHP SPIP, PHPSlash...) permettant de monter facilement et gratuitement son portail. En juillet 2000 plus de 300.000 sites tournaient déjà sous PHP !

A la fin de l'année 1999, une version bêta de PHP, baptisée PHP4 est apparue...

La version PHP4 est une version plus rapide que les autres car elle utilise le nouveau moteur Zend. De plus le support des sessions est directement intégré alors que dans les autres versions il fallait installer la bibliothèque PHPLib pour accéder à un contrôle de session ou il fallait en écrire un.

En 2005 la version PHP5 ...

### **A qui s'adresse PHP ?**

PHP est l'outil idéal pour la création de tout site Internet ou intranet de taille moyenne ou de grande taille mais n'ayant pas à supporter un grand nombre de requêtes simultanées.

Plusieurs sites affirment afficher plus de 450 000 pages par jour sur une machine dont la configuration est la suivante : Dual PII-333, 512Mb RAM, 2x9Go SCSI, et cela sans atteindre 100% de charge sur chacun des deux processeurs. "PHP can spew out HTML much much faster than the leased line can push the bits." (Rasmus Lerdorf)

Reprenant une syntaxe claire et familière puisque très proche de celle du langage C, PHP est un langage dont la prise en main est généralement rapide (PHP provoque moins de phénomènes de rejet que Perl ...).

C'est donc un langage qui conviendra autant au débutant désireux de se familiariser avec les techniques du web dynamique, qu'au professionnel cherchant une solution simple mais puissante et fiable pour la création de sites de taille moyenne

**Comment ça marche**

PHP est un langage destiné à l'écriture de scripts. Il peut travailler de façon autonome, sans le support de HTML. Les instructions PHP sont placées entre deux balises particulières, ce qui permet tant au navigateur qu'au serveur de les reconnaître. Lorsque le navigateur demande une page au serveur, si celui-ci y rencontre des balises PHP, il appelle l'interpréteur PHP qui exécute un certain nombre d'actions en fonction de la nature des instructions contenues dans la page. Le plus souvent, il va en résulter la création de sorties pouvant ou non comporter des commandes HTML et ces sorties vont être envoyées au navigateur qui va les afficher. C'est le processus de création de pages Web dynamiques.

**La nouvelle génération du Web : les sites Web actifs**

Le web s'est orienté vers les sites web actifs, permettant à l'utilisateur de recevoir des pages personnalisées et offrant de ce fait une navigation plus dynamique. Ils sont réalisés à l'aide d'une combinaison de langages et de technologies, à utiliser seul ou combinés, car ils sont tous indépendants les uns des autres.

Ces technologies peuvent être réparties en deux groupes, selon qu'il s'agit de technologies côté client ou côté serveur.

**Technologies dynamiques côté client**

Toutes ces technologies sont des innovations relativement récentes. Le principal inconvénient du côté client est que l'administrateur du site (webmestre) ne dispose d'aucun contrôle sur le logiciel utilisé pour afficher la page. Comme les entreprises tiennent évidemment à prendre en compte autant d'utilisateurs que possible avec le maximum de navigateurs, les progrès sont particulièrement lents pour ces nouvelles technologies, prises en charge uniquement par la dernière génération des principaux navigateurs du marché.

Ces technologies côtés client sont :

Contrôle ActiveX ;

Applets Java ;

Les scripts côté client (essentiellement le JavaScript) et DHTML.

**Technologies côté serveur**

Il y a quelques années, Common Gateway Interface (CGI) constituait la seule vraie solution permettant de mettre des données dynamiques sur le Web. Les programmes CGI constituent un moyen relativement simple de créer une application web acceptant des saisies de l'utilisateur, d'interroger une base de données et de renvoyer certains résultats au navigateur. Microsoft et Netscape ont tous deux développés des API propriétaires pouvant être utilisées pour développer du code permettant de servir les requêtes web. Parmi les dernières technologies web côté serveur, on peut citer les Active Server Pages (ASP), les servlets Java, les JavaServer Pages (JSP) et PHP.

PHP fonctionne de façon similaire à JSP et à ASP : les sections de script sont encadrées par des balises (< ? ... ?>) et incorporées à une page HTML. Ces scripts sont exécutés sur le serveur avant que la page ne soit envoyée au navigateur, si bien qu'une page PHP ne se préoccupe pas de la prise en charge par le navigateur. Contrairement à ASP, PHP est cependant indépendant de la plate-forme et existe pour les différentes versions de Windows, Unix et Linux, ainsi que pour de nombreux serveurs dont Apache et IIS. Le facteur décisif est qu'il est libre et donc "Open Source".

**De l'intérêt du traitement côté serveur**

Le traitement et la génération de page web côté serveur offrent plusieurs avantages par rapport aux technologies côté client, notamment :

diminution du trafic réseau en limitant les échanges client/serveur à l'envoi de la requête et de la réponse ;

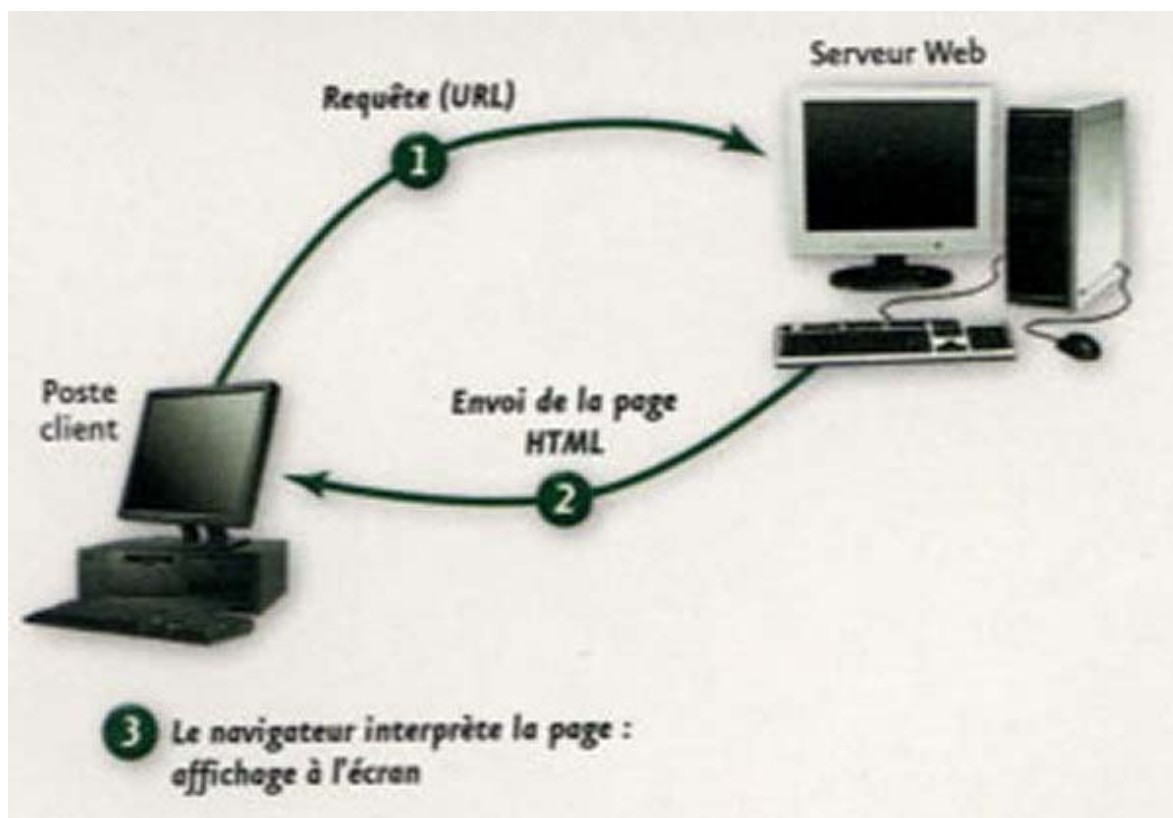
réduction du temps de téléchargement : le client ne reçoit qu'une simple page HTML ;

élimination des problèmes de compatibilité du navigateur;

Amélioration des mesures sécuritaires, puisque vous pouvez coder des choses qui ne seront jamais vues par le navigateur.

## Architecture

La plupart des sites web offrent un contenu statique, comme des publications scientifiques ou des articles. Les pages de ces sites consistent en du texte simple agrémenté de quelques images et de liens hypertextes menant vers d'autres pages. Pour cette catégorie de sites web, les techniques côté client suffisent amplement. HTML et les feuilles de style en cascade (CSS) permettent de structurer et de présenter le contenu des pages, Javascript permettant d'ajouter une touche plus sophistiquée si nécessaire.



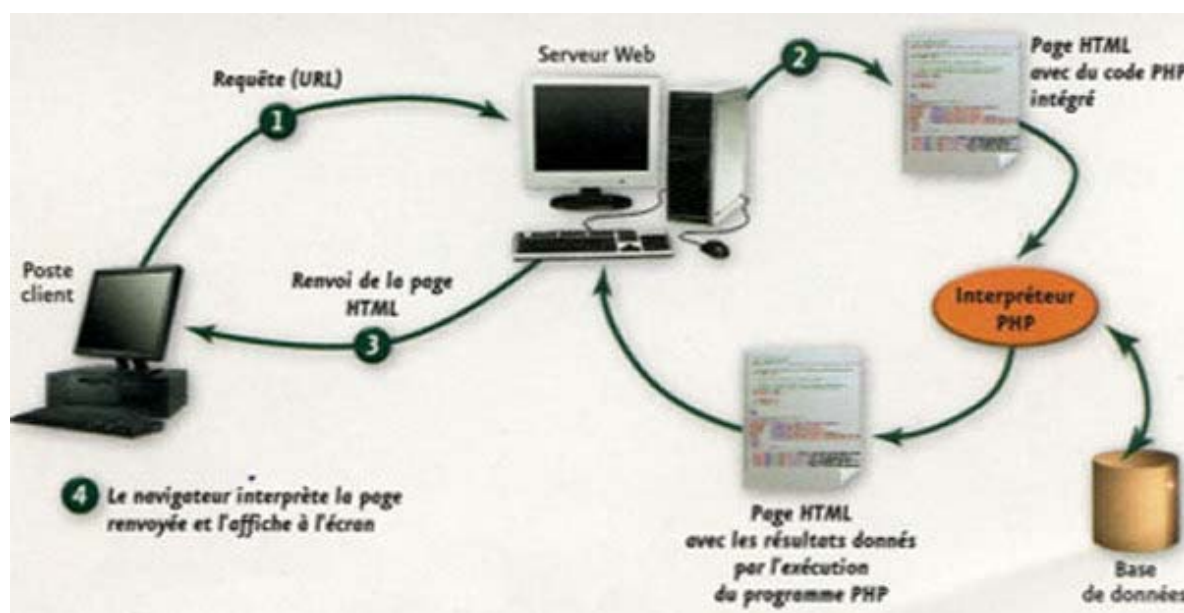
Source inconnue

Or, il se trouve qu'Internet et les intranets sont de plus en plus utilisés pour des applications, dont la plupart mettent en jeu des bases de données. Ces sites et les applications sont dynamiques, car le contenu est modifié selon les données impliquées et les actions de l'utilisateur.

Lorsqu'une requête pour une page provient du navigateur (Client), le serveur HTTP procède en trois étapes :  
lecture de la requête du navigateur (Client);  
identification de la page sur le serveur ;  
puis envoi de la page au navigateur (Client) via Internet (ou intranet).

C'est bien là que PHP entre en scène. En exécutant un programme PHP sur le serveur, vous pouvez créer de puissantes applications agissant de façon interactive avec une base de données et générant du contenu dynamique.





Source inconnue

Au lieu de renvoyer une page HTML statique à l'utilisateur, vous souhaitez que le serveur accomplisse certaines actions en fonction du code PHP : PHP doit prendre quelques décisions et créer une page appropriée à la situation.

De ce fait, avec PHP les actions du serveur sont les suivantes :

- lecture de la requête du navigateur (Client) ;
- identification de la page sur le serveur ;
- exécution des instructions venant de PHP , si besoin PHP interroge la base des données, pour modifier la page;
- renvoi de la page au navigateur (Client) via Internet (ou intranet).

Le code HTML pur est interprété par le navigateur, et non pas exécuté sur le serveur. En écrivant du code PHP qui sera exécuté sur le serveur web, vous pouvez accomplir bien plus de choses qu'il n'est normalement possible.

### 3. Installation

Vous devez installer PHP et un serveur Web qui vous permettra d'accéder à vos scripts PHP via un navigateur.

Nous utiliserons le serveur Web Apache, leader du marché, rapide, fiable et gratuit, mais un serveur IIS fonctionne également.

Vous pouvez installer Apache et PHP de deux façons :

- **Automatiquement**, en utilisant un utilitaire qui installe et configure tout ce dont vous avez besoin. Nous verrons le détail de cette installation dans le chapitre "Installation rapide avec EasyPHP".

- **Manuellement**, en suivant les étapes ci-dessous.

#### 3.1. Installation manuel

Les fichiers nécessaires pour l'installation de PHP et d'Apache sont disponibles sur le web à l'adresse <http://www.php.net/>

Il est conseillé de télécharger les dernières versions disponibles.

#### Installation de PHP

Créer un répertoire nommé php sur C:/ (ou un autre répertoire) et copiez de dans les fichiers PHP.

#### Configuration de PHP

##### Php4

Copiez le fichier php4ts.dll de PHP dans C:/WINNT/System (32).

Copier le fichier php.ini-dist de PHP dans C:/WINNT en retirant le "-dist". Ce fichier permet de régler les différents paramètres de PHP.

##### Php5

Copiez le fichier php5ts.dll de PHP dans C:/WINNT/System (32).

Copier le fichier php.ini-recommended de PHP dans C:/WINNT en retirant le - recommended. Ce fichier permet de régler les différents paramètres de PHP.

Voici une liste des points utiles (à noter que les ";" servent de commentaires) :

- Dans "Paths and Directories"

Ajouter . (point) dans include\_path .

Mettre le chemin des fichiers dll (par exemple c:\php\extensions) dans extension\_dir

##### Php4

```
.....
;
Paths and Directories ;
.....
;
include_path = .
doc_root =
user_dir =
upload_max_filesize = 2097152
extension_dir = c:\php\extensions
enable_dl = On
```

##### Php5

```
.....
;
Paths and Directories ;
.....
;
include_path = .
doc_root =
user_dir =
upload_max_filesize = 2097152
extension_dir = c:\php\ext
enable_dl = On
```

- Dans "Windows Extensions"

Dé-commenter la ligne des fichiers d'extension que l'on souhaite utiliser en enlevant le point virgule.  
Par exemple si vous souhaitez utiliser les fonctions ftp et imap :

Php4

```
;Windows Extensions ;  
  
extension=php_ftp.dll  
;extension=php_gettext.dll  
;extension=php_ifx.dll  
extension=php_imap.dll ;
```

Php5

```
;Windows Extensions ;  
  
extension=php_mysql.dll  
extension=php_ftp.dll  
;extension=php_gettext.dll  
;extension=php_ifx.dll  
extension=php_imap.dll ;
```

Remarque : MySQL étant supporté nativement dans PHP4, il n'est pas nécessaire de charger la dll, ce qui n'est pas le cas pour PHP5.

- Dans "[Session]"

Si vous souhaitez utiliser les sessions PHP, le plus simple est de garder la configuration par défaut, et donc de stocker les sessions dans des fichiers.

Il faut par contre spécifier le chemin du répertoire où seront stockées ces sessions. Par exemple, vous pouvez créer un répertoire c:\php\sessions.

Php4 et Php5

```
[Session]  
session.save_handler = files ; les sessions sont stockées dans des fichiers  
session.save_path = c:\php\sessions  
session.use_cookies = 1 ; on utilise des cookies pour transmettre l'identifiant de session session.name =  
PHPSESSID ;  
etc.
```

### **Installation d'Apache**

Vous devez l'installer dans C:/Program files/Apache group

### **Configuration d'Apache**

Le fichier "httpd.conf" permet de configurer Apache.

Vous pouvez par exemple ajouter votre adresse IP à la ligne Server Name ;

Vous pouvez également modifier la racine Web de vos documents, à l'aide de la directive DocumentRoot. Par défaut il s'agit du répertoire "htdocs", dans le répertoire d'installation d'Apache. Il convient de le changer, pour "c:\web" par exemple.

```
ServerName 127.0.0.1  
DocumentRoot "c:/web"
```

Et un peu plus bas ...

```
<Directory "c:/web">  
Options All  
AllowOverride All  
Order allow,deny  
Allow from all  
</Directory>
```

Copiez les lignes de configuration suivantes, puis collez-les dans le fichier httpd.conf.

#### Php4

```
# for the apache module
LoadModule php4_module c:/php/sapi/php4apache.dll
#Ajouter a la suite de la ligne ci-dessous les extensions que vous
#voulez voir reconnaître par php, précédée d'un point
AddType application/x-httpd-php .phtml .phtml .phtml .php3 .php4 .php .php2 .inc

#for the cgi binary (you can use that one compiled with force cgi redirect too)
ScriptAlias /php4/ "C:/php/"
Action application/x-httpd-php4 "/php4/php.exe"
```

#### Php5

```
# for the apache module
LoadModule php5_module c:/php/sapi/php5apache.dll
#Ajouter a la suite de la ligne ci-dessous les extensions que vous
#voulez voir reconnaître par php, précédée d'un point
AddType application/x-httpd-php .phtml .phtml .phtml .php3 .php4 .php .php2 .inc .php5

#for the cgi binary (you can use that one compiled with force cgi redirect too)
ScriptAlias /php5/ "C:/php/"
Action application/x-httpd-php5 "/php5/php.exe"
```

Pour plus d'informations sur Apache, il y a le site web d'Apache Software Foundation à l'adresse : <http://www.apache.org/>. Il existe aussi un précis et concis sur Apache chez O'Reilly.

#### **Lancement et arrêt du serveur**

Dans Démarrer, Programmes, Apache httpd Server, cliquez Start Apache in console. Une fenêtre s'ouvre indiquant :

Apache (Winn 32) PHP running ...

Pour arrêter le serveur, dans Démarrer, Exécuter tapez cmd. Dans la fenêtre qui s'ouvre, tapez cd program files/apache group/apache puis apache -k shutdown.

#### **Test de la configuration**

Pour tester si tout est bien installé, il faut d'abord vous rendre sur <http://127.0.0.1> (ou l'adresse IP de votre site). Vous devez obtenir une page vous indiquant qu'Apache fonctionne correctement. Dans le cas contraire, vérifiez que vous avez bien lancé apache.exe.

Il faut ensuite vérifier que PHP fonctionne aussi. Pour cela, lancez votre éditeur de texte (par exemple phpedit), Créez un nouveau fichier, avec l'extension adaptée (phpinfo.php par exemple) et insérez-y la ligne suivante :

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

Sauvegardez le fichier dans le répertoire adéquat (c:\program files\apache group\apache\htdocs\), puis lancez ce fichier à l'aide de votre navigateur (<http://127.0.0.1/phpinfo.php>) : vous devez obtenir une page d'information sur la configuration de PHP.

#### **Interfaçage avec un SGBD**

L'interfaçage avec une base de données est certainement un des aspects les plus intéressants du langage PHP.

Une base de données peut offrir des solutions très séduisantes à peu près à toutes les applications nécessitant la génération de pages HTML dynamiques.

Ceci vous permettra d'organiser vos pages en trois parties :

- un corps de HTML faisant la présentation, et invoquant les scripts de génération dynamique de contenu ;

- les scripts de génération de contenu (mise en forme des données lues dans la base de données);
- l'interfaçage avec la base de données, avec des fonctions d'enrobage (wrappers), permettant de s'abstraire de la base de données utilisée.

**Quelle base de données ?**

Le trio PHP + MySQL + Apache est une solution éprouvée et très fiable, sur une machine Linux ou FreeBSD ou sur une plate-forme Win32, il est souvent livré dans un package appelé "MAP".

PHP contient des connexions natives vers la plupart des systèmes de bases de données. En plus de MySQL, vous pouvez vous connecter directement aux bases de données PostgreSQL, mSQL, Oracle, dbm, filePro, Informix, Interbase et Sybase pour ne citer qu'elles.

Grâce au standard ODBC (Open Database Connectivity), vous pouvez vous connecter à n'importe quelle base de données possédant un pilote ODBC, comme les produits de Microsoft.

### **3.2. Installation rapide avec EasyPHP**

#### **Présentation d' EasyPHP**

EasyPHP installe et configure automatiquement un environnement de travail complet permettant de mettre en œuvre toute la puissance et la souplesse qu'offrent le langage dynamique PHP et son support efficace de bases de données. EasyPHP regroupe un serveur http Apache, une base de données MySQL, PHP et un outil d'administration de base de données phpMyAdmin.

#### **Installation d' EasyPHP**

Nous utiliserons dans le cadre de ce cours, la version EasyPHP 1.8 comprenant Apache 1.3.33, PHP 4.3.10, PHPMyadmin 2.6.1 et MySQL 4.1.9 .

Télécharger EasyPHP sur le site <http://www.easyphp.org/>, et l'installer à l'aide du fichier d'installation (easyphp1-8\_setup.exe)

Le " lancement " d'EasyPHP revient à mettre en route le serveur Apache et MySQL. A l'installation, un raccourci vers EasyPHP est créé dans le répertoire "Démarrer/Programmes/EasyPHP". Une fois EasyPHP lancé, une icône se place dans la barre des tâches à côté de l'horloge. Un clic droit permet d'accéder à différents menus :

- fichier Log : renvoie aux erreurs générées par Apache et MySQL ;
- configuration : donne accès aux différentes configurations d'EasyPHP ;
- web local : ouvre la page <http://localhost/index.htm>, page d'accueil d'EasyPHP ;
- démarrer/Arrêter : démarre/arrête Apache et MySQL ;
- quitter : ferme EasyPHP ;

Pour que vos pages PHP soient interprétées, vous pouvez placer vos fichiers dans le répertoire www Le serveur Apache est configuré pour ouvrir automatiquement un fichier index lorsque vous saisissez l'adresse "<http://localhost/>" (à condition évidemment que le serveur Apache soit en route). Cette page sert de page d'accueil au Web local et permet de vérifier le bon fonctionnement d'EasyPHP.

Afin d'avoir une vision plus claire des développements, il est conseillé de créer un répertoire par projet et d'ajouter un alias en indiquant le répertoire contenant vos fichiers.

Il existe autant de façons de programmer en PHP qu'il existe d'éditeurs spécialisés ou non (éditeurs html, coloration syntaxique, saisie semi-automatique...).

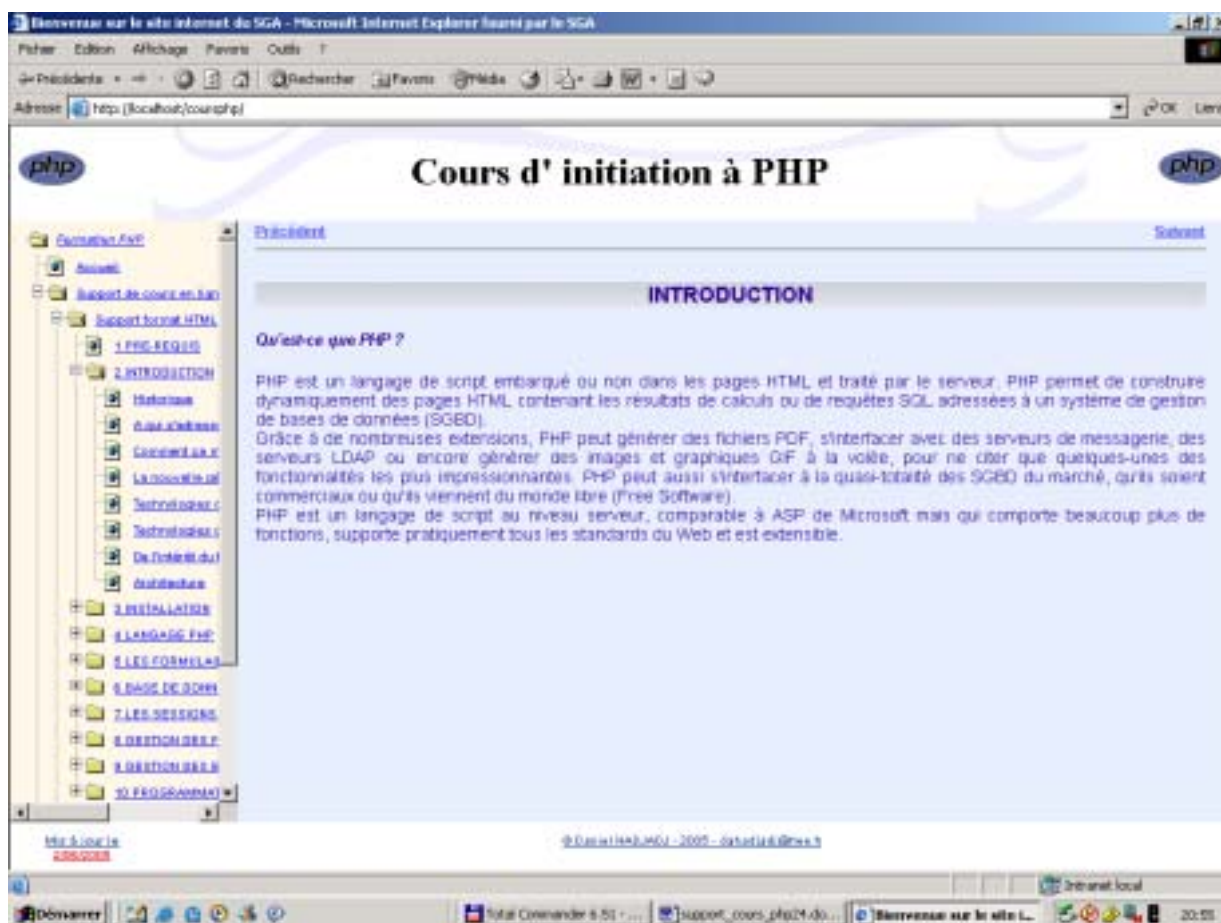
Pour tester la configuration afficher la page d'administration de EasyPHP à l'adresse suivante : <http://127.0.0.1/home/>

### 3.3. Installation du support de cours

Nous allons maintenant installer un site en mode local contenant le support de cours. A partir de ce site statique, entièrement réalisé en HTML, nous allons étudier les différentes manières de le rendre dynamique grâce à PHP.

Etapes d'installations

- Créer un répertoire "coursphp".
- Télécharger le fichier " coursphp.zip".
- Décompresser ce fichier dans le répertoire "coursphp".
- Avec un clic droit sur l'icône de EasyPHP , cliquez sur administration
- Cliquez sur ajouter un alias .
- Indiquez le nom du site (exemple : coursphp) et le chemin d'accès ( exemple : C:\ coursphp ).
- Valider en cliquant sur OK
- Redémarrer EasyPHP afin que les changements soient pris en compte par le serveur http
- Maintenant l'adresse <http://localhost/coursphp/> vous permet d'afficher le site.



## 4. Langage PHP

### 4.1. Quelques règles de base

Le code PHP est enregistré comme texte brut au format ASCII, si bien que vous pouvez écrire une page PHP à l'aide de pratiquement n'importe quel éditeur de texte. Nous utiliserons PHPEdit.

Dans le cas d'un fichier HTML, le serveur Web transmet simplement le contenu du fichier au navigateur. Il n'essaie en aucune façon de comprendre ou de traiter le fichier : c'est le rôle du navigateur. Les fichiers dotés de l'extension .php sont gérés différemment : le code PHP y est recherché. Le serveur web démarre en "mode HTML". Lorsqu'il commence son examen, il part du principe que le fichier ne contient que du code HTML, des feuilles de styles en cascade (CSS), du code JavaScript, du texte brut ou tout autre texte transmis au navigateur sans besoin d'interprétation de la part du serveur. Il entre en "mode PHP" dès qu'il rencontre une balise PHP, qui sert à "sortir" du mode HTML.

On peut utiliser plusieurs types de balises PHP :

#### Les différents types de balises PHP

- Le style XML :  
C'est le style par défaut qui est reconnu par tous les interpréteurs. C'est celui que nous utiliserons.  
`<?php ... ?>`
- Le style court (instruction SGML) :  
`<? ... ?>`
- Le style JavaScript  
Instruction qui appelle le JavaScript ou le VBScript, à utiliser si votre éditeur ne gère pas les instructions de traitement :  
`<SCRIPT LANGUAGE='php'> ... </SCRIPT>`
- Caractères d'échappement d'ASP :  
A condition que la paramètre de configuration asp\_tags du fichier de configuration de PHP (php.ini) est la valeur on.  
`<% ... %>`

Les instructions PHP doivent obligatoirement se terminer par un point-virgule sauf après l'accolade fermante d'un bloc d'instructions.

PHP permet de générer dynamiquement des pages web côté client. Il suffit pour cela d'incorporer du code côté client dans le texte, généré par PHP et de l'envoyer au navigateur.

La sortie du script peut être générée par les instructions echo ou print.

Les commentaires peuvent être insérés dans le script grâce aux signes :

```
// pour des commentaires sur une seule ligne ;

/*
pour des commentaires sur plusieurs lignes
*/
```

Voici deux exemples :

Exemple 1 :

```
<?php
echo "Texte généré par PHP";
?>
```

le navigateur affiche : Texte généré par PHP



Exemple 2 :

```
<?php  
    print "<SCRIPT LANGUAGE='JavaScript'>alert ('Erreur !');</SCRIPT>";  
?>
```

le serveur transmet le texte :

<SCRIPT LANGUAGE='JavaScript'>alert ('Erreur !');</SCRIPT>

au navigateur qui l'interprète comme du code JavaScript et affiche une boîte d'alerte.

Exercice : (Voir annexe Les règles de base).

Générer à l'aide de PHP le texte suivant :

[Voici mon premier script.](#)

[Bonjour tous le monde](#)

Puis le même texte à l'aide de JavaScript.

## 4.2. Les variable PHP

PHP est un langage pauvrement typé. On n'est pas tenu de déclarer le type des variables ou des constantes que l'on va utiliser. Les variables peuvent renfermer successivement n'importe quel type de contenu. C'est PHP qui détermine le type de variable par son contexte, c'est-à-dire par la valeur qui lui est attribuée.

En PHP, les variables sont représentées par un signe dollar "\$" suivi du nom de la variable. Les noms des variables écrites en minuscules sont différentes des variables écrites en majuscules (ie : \$x != \$X). Les noms de variables suivent les mêmes règles de nommage que les autres entités PHP. Un nom de variable valide doit commencer par une lettre ou un souligné (\_), suivi de lettres, chiffres ou soulignés. Exprimé sous la forme d'une expression régulière, cela donne :

```
'[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'
```

PHP 4 permet aussi d'assigner les valeurs aux variables par référence. Cela signifie que la nouvelle variable ne fait que référencer (en d'autres terme, "devient un alias de", ou encore "pointe sur") la variable originale. Les modifications de la nouvelle variable affecteront l'ancienne, et vice versa. Cela signifie aussi qu'aucune copie n'est faite : l'assignation est donc beaucoup plus rapide. Cela se fera notamment sentir dans des boucles, ou lors d'assignation de grands objets (tableaux).

Pour assigner par référence, ajoutez simplement un & (ET commercial) au début de la variable qui est assignée (la variable source). Dans l'exemple suivant, " Mon nom est Pierre " s'affichera deux fois :

Exemple :

```
<?php
$foo = 'Pierre'; // Assigne la valeur 'Pierre' à $foo
$bar = &$foo; // Référence $foo avec $bar.
$bar = "Mon nom est Pierre"; // Modifie $bar...
echo $foo; // $foo est aussi modifiée
echo $bar;
?>
```

### Type de variables

#### Type scalaire :

- Constantes.
- Booléen.
- Entier.
- nombre à virgule flottante.
- chaîne de caractères.

#### Type composé :

- Tableau.
- Objet.

### Les constantes

D'une façon générale, vous pouvez donner un nom symbolique à n'importe quelle constante scalaire (c'est à dire de type booléen, integer, double string). Toutefois l'usage veut que les constante s'écrivent en lettre capitales.

Pour créer une constante on utilise la fonction define() dont la forme générale est :

#### Structure de la définition d'une constante

```
define (nom, valeur, [, casse]);
```

avec :

nom : chaîne de caractères représentant le nom qu'on veut donner à la constante.

valeur : valeur attribuée à cette constante.

casse : valeur booléenne indiquant si le nom donné à la constante sera sensible (FALSE) ou non (TRUE) à la casse. Par défaut sa valeur est FALSE, ce qui signifie que le nom de la constante est sensible à la casse.

Comme son nom le suggère, cette valeur ne peut jamais être modifiée durant l'exécution du script (les constantes magiques \_\_FILE\_\_ et \_\_LINE\_\_ sont les seules exception).

Exemple :

```
<?php
    define ("PI", 3.1415926535);
    $r = 20 ;
    $circ =2 * PI * $r ;
    echo " La circonférence de rayon $r : $circ <br>";
    echo " PI vaut ".PI;
?>
```

Exercice : (voir annexe : Les constantes).

Générer à l'aide de PHP les deux lignes suivantes en affectant la valeur "Bonjour tous le monde" a une constante :

"Voici mon deuxième script."

Votre constante.

Puis dans le même fichier le même texte à l'aide de JavaScript.

### Variable type Booléens :

C'est le type le plus simple. Un booléen exprime les valeurs de vrai TRUE ou faux FALSE. Vous pouvez utiliser les constantes ' TRUE ' et ' FALSE ' pour spécifier une valeur de Type booléen :

Ces constantes sont insensibles à la casse.

Exemple :

```
<?php
    $foo = True; // assign la valeur TRUE to $foo
    $foo = False; // assign la valeur false to $foo
    $foo = 1; // assign la valeur 1 to $foo
    $foo = 0; // assign la valeur 0 to $foo
    foo (True) = 1
    foo (False) =
    foo(1) = 1
    foo(0) = 0
?>
```

### Variable type entier :

Un entier est un nombre de l'ensemble des entiers naturels  $Z : Z = \{..., -2, -1, 0, 1, 2, ...\}$ . Il est possible de spécifier les nombres entiers (integers) de toutes les manières suivantes : décimale (base 10), hexadécimale (base 16), octale (base 8) éventuellement précédé du signe moins ( - ).

Exemple :

```
<?php
    $a = 1234; echo "$a = nombre entier en base 10 : (1234)<br>";
    $a = -123; echo "$a = nombre entier négatif: (-123)<br>";
    $a = 0123; echo "$a = nombre entier en base 8, octale (équivalent à 83 en base 10): (83)<br>";
    $a = 0x1A; echo "$a = nombre entier en base 16, hexadécimale (équivalent à 26 en base 10): (26)";
?>
```

### Les nombres à virgule flottante :

Les nombres à virgule flottante connus aussi sous le vocable de " double " ou " float " " nombre réels " peuvent être spécifiés en utilisant la syntaxe suivante:

Exemple :

```
<?php
    $a = 1.234; echo $a."<br>";
    $a = 1.2e3; echo $a;
?>
```

- si une variable débute par une valeur numérique valide, la chaîne sera évaluée par cette valeur, sinon elle sera évaluée comme zéro ;
- si une valeur double constitue l'intégralité de la chaîne, la chaîne sera évaluée comme valeur double, mais si d'autres caractères non doubles sont présents dans la chaîne, elle sera évaluée comme entier.

Exemples :

```
<?php
    $var = 1; // $var est un entier
    $var = 1.2; // $var est un double
    $var = "abcd"; // $var est une chaîne

    $str = "7 rue Fbg Montmartre";
    $x = 3 + $str; // $x = 10 et $str reste inchangée
    echo "$x : $str"; // affiche : 10 : 7 rue Fbg Montmartre
```

```
$str = "code 275";  
$x = 3 + $str; // $x = 3  
echo "$x : $str"; // affiche : 3 : code 275  
  
$str = "312.65 mètres";  
$x = 3 + $str; // $x = 315  
echo "$x : $str"; // affiche : 315.65 : 312.65  
?>
```

### Les variable type chaînes de caractères :

Les chaînes de caractères sont des séquences de caractères.

En PHP, un caractère est un octet, et il y en a 256 de possibles. PHP n'a pas (encore?) de support natif d'Unicode

Une chaîne peut être spécifiée de deux manières différentes :

guillemets simples

guillemets doubles

#### Guillemets simples

Syntax : `$a="MaVariable";` alors `$a` vaut `MaVariable`

Exemple :

```
<?php
echo 'Ceci est une chaîne simple';
echo "<br>";
echo 'Vous pouvez inclure des nouvelles lignes dans une chaîne,
comme ceci.';
echo "<br><br>";

echo 'Arnold a coutume de dire : "I\'ll be back"';
// affiche : ... "I'll be back"
echo "<br><br>";
echo 'Etes vous sûr de vouloir effacer le dossier C:\\*.??';
// affiche : Etes vous sûr de vouloir effacer le dossier C:\\*.??
echo "<br><br>";
echo 'Etes vous sûr de vouloir effacer le dossier C:/*.??';
// affiche : Etes vous sûr de vouloir effacer le dossier C:/*.??
echo "<br><br>";
echo 'Je suis en train de mettre une nouvelle ligne comme ceci : \n';
// affiche : Je suis en train de mettre une nouvelle ligne comme ceci : \n
?>
```

#### Guillemets doubles

Si la chaîne est entourée de guillemets doubles, PHP va comprendre certaines séquences de caractères :

#### séquences de caractères comprises par PHP.

\n Nouvelle ligne (linefeed, LF ou 0x0A (10) en ASCII)

\r Retour à la ligne(carriage return, CR ou 0x0D (13) en ASCII)

\t Tabulation horizontale (HT ou 0x09 (9) en ASCII)

\\ Antislash

\\$ Caractère \$

\" Guillemets doubles

\[0-7]{1,3}

Une séquence de caractères qui permet de rechercher un nombre en notation octale.

\x[0-9A-Fa-f]{1,2} Une séquence de caractères qui permet de rechercher un nombre en notation hexadécimale

Exercice : (Voir annexe Les variable type chaînes de caractères).

créer une page en utilisant les côtes puis les guillemets affichant le texte suivant :

Voici enfin l'été indien dont je rêvais !

Cette histoire, c'était la mienne.

### Traitement des variables dans les chaînes

Lorsqu'une chaîne est spécifiée avec des guillemets doubles les variables qu'elle contient sont remplacées par leur valeur.

Il y a deux types de syntaxe, une simple et une complexe . La syntaxe simple est la plus courante, et la plus pratique : elle fournit un moyen d'utiliser les variables, que ce soit des chaînes, des tableaux ou des membres d'objets.

La syntaxe complexe a été introduite en PHP 4 et peut être reconnue grâce aux accolades entourant les expressions.

Dès qu'un signe dollar \$ est rencontré, l'analyseur PHP va lire autant de caractère qu'il peut pour former un nom de variable valide. Entourez le nom de la variable avec des accolades pour indiquer explicitement son nom.

Utilisation des accolades {} dans les chaînes

Exemple :

```
<?php
$boisson = 'vin';
echo "Du $boisson, du pain et du fromage!";
// OK, car "," n'est pas autorisé dans les noms de variables

echo "Il a goûté plusieurs $boissons";
// Pas OK, car 's' peut entrer dans un nom de variable, et PHP recherche $boissons

echo "Il a goûté plusieurs ${boisson}s";
// OK
?>
```

Exercice : (voir annexe les chaînes de caractères Exercice 1).

Reprendre l'exemple présenté dans le paragraphe "Guillemets simples" et remplacer les côtes par les guillemets, Observer ce qui se passe , puis corriger.

Exercice : (voir annexe les chaînes de caractères Exercice 2).

Déclarer une variable php et affecter lui la valeur de votre prénom puis,  
générer à l'aide de PHP le texte suivant,  
Je me présente  
Mon prénom est "votre variable"  
Puis le même texte à l'aide de JavaScript.

Exercice : (voir annexe les chaînes de caractères Exercice 3).

Reprendre l'exercice 2 et ajouter une variable qui contiendra votre nom, puis  
Afficher votre variable nom et variable prénom

Exercice : (voir annexe les chaînes de caractères Exercice 4).

A l'aide de la variable \$super=fantastique, Créer une page PHP affichant le texte suivant en utilisant la syntaxe simple puis complexe :

C'est fantastique.  
Ces soirées étaient fantastiques

### Les variable type tableaux

Les tableaux sont des agrégats de valeurs de types quelconques, éventuellement mélangés, regroupés sous un nom unique et dans lesquels on repère un élément isolé soit par un indice (compté à partir de zéro), soit par une clé enregistrée en même temps que la valeur correspondante lors de la création du tableau. On peut aussi créer un tableau au moyen de la fonction `array()`. La forme générale de cette déclaration est :

#### Déclaration d'un tableau.

```
$tableau[] = valeur  
ou  
$tableau = array(valeur, valeur, valeur...);  
ou  
$tableau = array(clé=>valeur, clé=>valeur, clé=>valeur...);
```

Exemple :

```
<?php  
$tableau = array ( "chien", "salade", "rocher");  
//ou  
$tableau = array ( 0=>"chien", 1=>"salade", 2=>"rocher");  
?>
```

Nous reviendrons plus longuement sur l'utilisation des tableaux aux chapitres Tableaux.

### Les variable type objet

Un objet est une sorte de conteneur pouvant contenir différents éléments, en général des classes. Les objet sont de type object.

Pour initialiser un objet, vous devez utiliser la commande "new" afin de créer l'instance de l'objet.

Exemple :

```
<?php  
class foo {  
    function faire_foo () {  
        echo "Faisant foo."  
    }  
}  
$bar = new foo;  
$bar->faire_foo();  
?>
```

## Le transtypage

Le type de données d'une variable peut être modifié à l'aide des "transtypes" (int), (double), (string) , ... .  
Les conversions autorisées sont:

(int), (integer)	-	type entier
(bool), (boolean)	-	booléen
(real), (double), (float)	-	type double
(string)	-	type chaîne
(array)	-	type tableau
(object)	-	type objet

Exemple :

```
<?php
$foo = 1 + "10.5";
echo "\$foo = 1 + \"10.5\" alors \$foo est du type float (11.5)<br>";

$foo = 1 + "-1.3e3";
echo "\$foo = 1 + \"-1.3e3\" alors \$foo est du type float (-1299)<br>";

$foo = 1 + "bob-1.3e3";
echo "\$foo = 1 + \"bob-1.3e3\" alors \$foo est du type integer (1)<br>";

$foo = 1 + "bob3";
echo "\$foo = 1 + \"bob3\" alors \$foo est du type integer (11)<br>";
$foo = 1 + "10 Small Pigs";
echo "\$foo = 1 + \"10 Small Pigs\" alors \$foo est du type
integer(11)<br>";

$foo = 1 + "10 Little Piggies";
echo "\$foo = 1 + \"10 Little Piggies\" alors \$foo est du type integer (11)<br>";
$foo = "10.0 pigs " + 1.0;
echo "\$foo = 1 + \"10.0 pigs \" + 1.0 alors \$foo est du type float (11)<br>";

?>
```

PHP met deux fonctions à notre disposition pour traiter les types de variables

Les fonctions gettype() et settype().

### La fonction gettype()

Elle détermine le type de données de la variable et renvoie l'une des valeurs suivantes :

- "integer" ;
- "double" ;
- "string" ;
- "array" ;
- "object" ;
- "class" ;
- "unknown type" ;

### La fonction settype()

Elle définit le type d'une variable de façon explicite, et si le type ne peut pas être défini, elle renvoie une valeur false.

Exemples :

```
<?php
$var = 11.2; // $var est un double
echo "\$var = $var et le type de \$var est ";
echo gettype($var);
echo "<br>";
// affiche : $var = 11.2 et le type de $var est double

$var = (int)$var; // $var est maintenant un entier de valeur 11
echo "\$var = $var et le type de \$var est ";
echo gettype($var);
echo "<br>";
// affiche : $var = 11 et le type de $var est integer

$var = (double)$var; // $var est de nouveau un double de valeur 11.0
echo "\$var = $var et le type de \$var est ";
echo gettype($var);
echo "<br>";
// affiche : $var = 11 et le type de $var est double

$var = (string)$var; // $var est une chaîne de caractères de valeur "11.0"
```



```
echo "\$var = $var et le type de \$var est ";  
echo gettype($var);  
echo "<br>";  
// affiche : $var = 11 et le type de $var est string  
?>
```

Exercice : (Voir annexe Le transtypage Exercice 1).

Déclarer deux variables php et affecter leur la valeur de votre nom et prénom puis,  
générer à l'aide de PHP le texte suivant,  
Je me présente  
Mon prénom est "votre prénom!"  
(Déjà vu à l'exercice 3 du chapitre " Traitement des variables dans les chaînes")  
Affichez ensuite, le type de chaque variable.

Exercice : (Voir annexe Le transtypage Exercice 2).

Soit les variables \$var1=1000, \$var2=6.55957, \$var3= '1', \$var4=array()  
Créer une page PHP afin d'afficher ces variables et leur type, puis convertir  
\$var1 en type "double", \$var2 en type "integer", \$var3 en type "double" et \$var4 en type "integer".

### **les variables super global**

Ces variables sont des 'superglobal', ou globale automatique. Cela signifie qu'elles sont simplement disponible dans tous les contextes d'exécution (fonctions ou méthodes). Vous n'avez pas besoin de les déclarer global pour y accéder. Toutes ces informations se retrouvent grace à la fonction phpinfo() vue au paragraphe "test de la configuration" du chapitre "Installation".

#### **\$GLOBALS**

Contient une référence sur chaque variable qui est actuellement disponible dans l'environnement d'exécution global. Les clés de ce tableau sont les noms des variables globales.

#### **\$\_SERVER**

Les variables fournies par le serveur web, ou bien directement liées à l'environnement d'exécution du script courant. L'ancienne version \$HTTP\_SERVER\_VARS contient les mêmes informations, mais n'est pas auto globale.

#### **\$\_GET**

Les variables fournies par le protocole HTTP en méthode GET. L'ancienne version \$HTTP\_GET\_VARS contient les mêmes informations, mais n'est pas auto globale.

#### **\$\_POST**

Les variables fournies par le protocole HTTP en méthode POST. L'ancienne version \$HTTP\_POST\_VARS contient les mêmes informations, mais n'est pas auto globale.

#### **\$\_COOKIE**

Les variables fournies par le protocole HTTP, dans les cookies. L'ancienne version \$HTTP\_COOKIE\_VARS contient les mêmes informations, mais n'est pas auto globale.

#### **\$\_FILES**

Les variables fournies par le protocole HTTP, suite à un téléchargement de fichier. L'ancienne version \$HTTP\_POST\_FILES contient les mêmes informations, mais n'est pas auto globale. Voir Téléchargement par méthode POST ,pour plus d'informations.

#### **\$\_ENV**

Les variables fournies par l'environnement. L'ancienne version \$HTTP\_ENV\_VARS contient les mêmes informations, mais n'est pas auto globale.

#### **\$\_REQUEST**

Les variables fournies au script par n'importe quel mécanisme d'entrée et qui ne doit recevoir une confiance limitée. Un tableau associatif constitué du contenu des variables \$\_GET, \$\_POST, \$\_COOKIE, et \$\_FILES. Note : lorsque vous exécutez un script en ligne de commande, cette variable ne va pas inclure les variables argv et argc . Elles seront présentes dans la variable \$\_SERVER . La présence et la valeur des entrées de ce tableau sont réglés par la directive variables\_order . Ce tableau n'est l'évolution d'aucune variable d'avant PHP 4.1.0.

#### **\$\_SESSION**

Les variables qui sont actuellement enregistrées dans la session attachée au script. L'ancienne version \$HTTP\_ENV\_VARS contient les mêmes informations, mais n'est pas auto globale. Voir le chapitre pour plus d'informations.

Précisons que les variables super global ci-dessus sont en fait des tableaux.

### 4.3. Les opérateurs

Les opérateurs servent à déterminer une valeur en effectuant une opération sur une ou plusieurs variables. Les opérateurs de PHP sont très similaires à ceux de C, ou de Perl.

#### Les opérateurs arithmétiques

Vous rappelez-vous des opérations élémentaires apprises à l'école ?

Exemple	nom	résultat
\$a + \$b	Addition	Somme de \$a et \$b.
\$a - \$b	Soustraction	Différence de \$a et \$b.
\$a * \$b	Multiplication	Produit de \$a et \$b.
\$a / \$b	Division	Quotient de \$a et \$b.
\$a % \$b	Modulo	Reste de \$a divisé par \$b.

#### L'opérateur d'affectation

L'opérateur « = » permet d'affecter une valeur à une variable. La variable située à gauche du signe « = » se voit affecter la valeur située à droite du signe « = ». C'est ainsi que l'expression :

\$b = \$a + 5 ;

signifie que la valeur de la variable \$a est augmenté de cinq puis rangée dans la variable \$b.

PHP permet les affectation multiples. Ainsi, on a le droit d'écrire :

\$a = \$b = \$c = 2;

Les variable \$a, \$b et \$c contiendront toutes trois la valeur 2. Cette propriété est également valable pour les tableaux.

Vous ne devez pas confondre cet opérateur d'affectation avec l'opérateur de comparaison " == ".

#### L'opérateur de concaténation

Le point « . » est utilisé en PHP comme opérateur de concaténation pour fusionner plusieurs chaînes en une seule.

Exemple :

```
<?PHP
$nom = "Durand";
$prenom = "Pierre";
$personne = $prenom . " " . $nom;

echo $personne . "<br>";
echo "Martine " . $nom . "<br>";
echo "affiche : Pierre Durand<br>";
echo "affiche Martine Durand";
?>
```

Exercice : (Voir annexe L'opérateur de concaténation).

Déclarer une variable et affecter lui la valeur de votre prénom et une autre pour votre nom, puis générer à l'aide de PHP et de l'opérateur de concaténation le texte suivant,

Je me présente

Mon prénom est "votre variable prénom et votre variable nom"!

Puis le même texte à l'aide de JavaScript.

#### L'opérateur arobase

Cet opérateur ne peut intervenir que devant un nom de fonction. Il interdit alors à PHP d'afficher un éventuel message d'erreur au cours de l'exécution de la fonction. Il appartient dès lors au programmeur de prendre les dispositions nécessaires ( par exemple, en testant la valeur de retour de la fonction).

## Les opérateurs de comparaison

Les opérateurs de comparaison, comme le nom l'indique, vous permettent de comparer deux valeurs.

exemple	nom	résultat
\$a == \$b	Egal	Vrai si \$a est égal à \$b.
\$a === \$b	Identique	Vrai si \$a est égal à \$b et qu'ils sont de même type (PHP4 seulement).
\$a != \$b	Différent	Vrai si \$a est différent de \$b.
\$a < \$b	Plus petit que	Vrai si \$a est plus petit strictement que \$b.
\$a > \$b	Plus grand	Vrai si \$a est plus grand strictement que \$b.
\$a <= \$b	Inférieur ou égal	Vrai si \$a est plus petit ou égal à \$b.
\$a >= \$b	Supérieur ou égal	Vrai si \$a est plus grand ou égal à \$b.

## Les opérateurs logiques

exemple	nom	résultat
\$a and \$b	ET (And)	Vrai si \$a ET \$b sont vrais.
\$a or \$b	OU (Or)	Vrai si \$a OU \$b est vrai
\$a xor \$b	XOR(Or)	Vrai si \$a OU \$b est vrai, mais pas les deux en même temps.
! \$a	NON (Not)	Vrai si \$a est faux.
\$a && \$b	ET (And)	Vrai si \$a ET \$b sont vrais.
\$a    \$b	OU (Or)	Vrai si \$a OU \$b est vrai.

## les opérateurs d'incrémentation

exemple	name	effect
++\$a	Pre-increment	Incrémente \$a de un, et renvoi \$a.
\$a++	Post-increment	Renvoi \$a, et incrémente \$a de un.
--\$a	Pre-decrement	Décréments \$a de un, et renvoi \$a.
\$a--	Post-decrement	Renvoi \$a, et décrémente \$a de un.

## Raccourcis d'affectation de variables

Il est possible en PHP d'utiliser des raccourcis d'opérateurs lorsque le premier opérande est une variable et que le résultat est stocké dans la même variable.

Exemple	Equivaut à
\$Ma_variable += \$i	\$Ma_variable = \$Ma_variable + \$i
\$Ma_variable -= \$i	\$Ma_variable = \$Ma_variable - \$i
\$Ma_variable *= \$i	\$Ma_variable = \$Ma_variable * \$i
\$Ma_variable /= \$i	\$Ma_variable = \$Ma_variable / \$i
\$Ma_variable %= \$i	\$Ma_variable = \$Ma_variable % \$i
\$Ma_variable .= \$i	\$Ma_variable = \$Ma_variable + \$i
\$Ma_variable ++	\$Ma_variable = \$Ma_variable + 1
\$Ma_variable --	\$Ma_variable = \$Ma_variable - 1

#### 4.4. Instructions

##### Instruction conditionnelles

Les instructions conditionnelles indiquent à un programme qu'une décision doit être prise  
La forme générale de cette instruction est la suivante :

##### Structure de if

```
If (condition)
{
    bloc d'instructions ...
}
```

Exemple :

```
<?php
    $prix=10;
    if ( $prix > 1000 )
    {
        $remise = 10 ;
        $frais_de_port = 0 ;
    }
?>
```

il est possible d'imbriquer une instruction if dans une autre.

Exemple :

```
<?php
    $prix=110;
    if ( $prix < 1000 )
    {
        if($prix>100)
        {
            $remise = 10 ;
            $frais_de_port = 0 ;
        }
    }
?>
```

Exercice : (Voir annexe Instruction conditionnelles Exercice 1 ).

Initialiser la variable \$civ à 'Mme' si vous êtes une dame, 'Mlle' si vous êtes une demoiselle et à 'M' si vous êtes un homme. Créer les variables \$prenom et \$nom et affectez leur les valeur de votre choix .

Testez la variable \$civ afin d'afficher dans une page PHP l'une des phrases suivantes correspondant à votre sexe :

Bonjour Madame \$prenom \$nom.

Bonjour Mademoiselle \$prenom \$nom.

Bonjour Monsieur \$prenom \$nom.

##### Branchement conditionnel : mots clé « else » et « elseif »

##### Structure de else

```
If (condition)
{
    Si la condition est vérifiée exécution de ce bloc d'instructions ...
}
else
{
    Si la condition n'est pas vérifiée exécution de ce bloc d'instructions ...
}
```

**Le else** permet d'exécuter un second bloc d'instructions lorsque la condition testée n'est pas vérifiée. On exploite ainsi les deux possibilités logiques.

Exemple :

```
<?php
    $prix=10;
    if ( $prix > 1000 )
    { $remise = 10 ;
      $frais_de_port = 0 ;
    }
    else
    { $remise = 5 ;
      $frais_de_port = 50 ;
    }
?>
```

Exercice : (Voir annexe Instruction conditionnelles Exercice 2 ).

Reprendre l'exercice précédent et ajouter un Else affichant "J'ai des doutes sur votre genre.", si la variable \$civ n'est pas vérifié.

**Le elseif** permet de cascader les tests. Ses règles d'emploi sont inspirées de celles des deux précédentes.

#### Structure de elseif

```
If (condition)
{
    Si la condition est vérifiée exécution de ce bloc d'instructions ...
}
elseif(condition)
{
    Si la condition est vérifiée exécution de ce bloc d'instructions ...
}
elseif(condition)
{
    Si la condition est vérifiée exécution de ce bloc d'instructions ...
}
```

L'expression elseif est exécutée seulement si le if précédent et tout autre elseif précédent est évalués comme faux (FALSE), et que votre elseif est évalué à vrai (TRUE).

Exemple :

```
<?php
    $prix=1020;
    if ( $prix < 1000 )
    { $remise = 0 ;
      $frais_de_port = 50 ;
    }
    elseif ( $prix = 1000 )
    { $remise = 5 ;
      $frais_de_port = 10 ;
    }
    elseif ( $prix > 1000 )
    { $remise = 10 ;
      $frais_de_port = 50 ;
    }
?>
```

Exercice : (Voir annexe Instruction conditionnelles Exercice 3).

Un centre de formation dispense 10 cours différents aux stagiaires de mon entreprise.

5 cours principaux et 5 facultatifs.

Selon sa réglementation un stagiaire ne peut s'inscrire qu'à deux cours principaux maximum et deux cours facultatifs maximum à condition qu'il suive au moins un cour principal.

Le secrétariat de ce centre, chargé d'adresser un courrier à chaque stagiaire, dispose pour envoyer ce courrier de trois lettres type

Une lettre de conseil (lettre type conseil)

Une lettre de confirmation (lettre type confirmation)

Une lettre de rappel de la réglementation, demandant au stagiaire de restreindre son choix (lettre type interdit)

Un courrier personnalisé sera envoyé à tous les stagiaires conformément à la réglementation, selon les cas de figure suivant :

Le stagiaire ne s'inscrit à aucun cours principal alors, je lui adresse une lettre de conseil.

Le stagiaire ne s'inscrit qu'à un cours principal alors, je lui adresse une lettre de conseil.

Le stagiaire s'inscrit à deux cours principaux alors, je lui adresse une lettre de confirmation.

Le stagiaire s'inscrit à plus de deux cours principaux ou à plus de deux cours facultatifs alors, je lui adresse une lettre d'interdiction.

Ecrivez un programme en PHP permettant d'afficher la lettre type correspondant à la situation du stagiaire en ayant pris soin de vérifier au préalable la civilité du stagiaire.

Nous disposons des quatre variables suivantes pour nous aider :

\$prenom : prend la valeur de votre choix

\$nom : prend la valeur de votre choix

\$civ : peut prendre l'une des valeurs suivantes : "0" pour Madame, "1" pour Mademoiselle, "2" pour Monsieur.

**\$ncours** le nombre de cours principaux peut prendre les valeurs de 0 à 5.

**\$ncoursf** le nombre de cours facultatifs peut prendre les valeurs de 0 à 5.

**Instruction « switch »**

Cette instruction permet de réaliser un aiguillage multidirectionnel.

**Structure de Switch**

```
Switch (variable)
{ case valeur 1 : bloc d'instructions 1
    break;
  case valeur 2 : bloc d'instructions 2
    break;
  case valeur n : bloc d'instructions n

  default      : dernier bloc d'instructions
}
```

La case spécial default est utilisé lorsque tous les case ont échoués. Il doit être le dernier cas listé.

Exemple :

```
<?php
//Dans une liste déroulante on propose aux gens différents choix de pays,
//chaque choix $liste ayant pour valeur a, b, c et on renvoie à la page
//suivante contenant le script :
$liste=array();

switch ($liste) {
  case "a":
    echo "Vous habitez en France";
    break;
  case "b":
    echo "Vous habitez en Allemagne";
    break;
  case "c":
    echo "Vous habitez en Espagne";
    break;
  default :
    echo "Votre pays n'est pas connu";
}
?>
```

Exercice : (Voir annexe Instruction conditionnelles Exercice 4).

Reprendre l'exercice 1 de ce chapitre et afficher la page en utilisant l'instruction "switch".

**Les boucles****Les boucles while et do...while**

La boucle while exécute le bloc d'instruction tant que l'expression de la boucle while est vérifiée (évaluée comme TRUE). La valeur de l'expression est vérifiée à chaque début de boucle, et, si la valeur change durant l'exécution de l'instruction, l'exécution ne s'arrêtera qu'à la fin de l'itération (chaque fois que le PHP exécute l'instruction, on appelle cela une itération). Si l'expression du while n'est pas vérifiée (évaluée comme FALSE) avant la première itération, l'instruction ne sera jamais exécutée.

**Structure de la boucle while**

```
while (condition) {
    bloc d'instructions...
}
```

Exemple :

```
<?php
$num = 1;
while ($num <= 5) {
  echo $num."<br>"; // affiche tous les chiffres de 1 à 5
  $num++;
}
```



```
}  
?>
```

Exercice : (Voir annexe les boucles Exercice 1)

Soit le tableau des mois de l'année :

`$mois=array('Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Aôut','Septembre','Octobre','Novembre','Décembre');` Créer une page PHP affichant les douze mois de l'année à l'aide de l'instruction while.

La boucle do ... while est similaire à la boucle while : la condition est vérifiée à la fin de chaque itération, et non au début, elle s'exécute donc toujours au moins une fois.

#### Structure de la boucle do while

```
do {  
    bloc d'instructions...  
}  
while ($num < 6)
```

Exemple :

```
<?php  
    $num = 1;  
    do {  
        echo $num."<br>";  
        $num++;  
    }  
    while ($num < 6) // affiche tous les chiffres de 1 à 5  
?>
```

Exercice : (Voir annexe les boucles Exercice 2)

Reprendre l'exercice précédent et créer une page PHP affichant le mois de juin à l'aide de l'instruction do while.

#### Boucle « for »

Les types de boucle for et while sont équivalentes. On utilise la boucle for quand le nombre de répétitions est prédéfini.

#### Structure de la boucle for

```
for (expression1 ; condition ; expression2) {  
    bloc d'instructions...  
}
```

Exemple :

```
<?php  
    for ($i = 1 ; $i <= 5 ; $i++) {  
        echo $i."<br>"; // affiche tous les chiffres de 1 à 5  
    }  
?>
```

Exercice : (Voir annexe les boucles Exercice 3)

Reprendre l'exercice 1 de ce chapitre et créer une page PHP affichant les douze mois de l'année à l'aide de la boucle for.

Nous verrons les autres boucles (foreach, list et Each) au chapitre concernant les tableaux.

#### 4.5. les Tableaux

Nous avons déjà abordé brièvement les tableaux, maintenant nous allons voir plus loin leurs possibilités. Les tableaux sont une série de variables qui ont toutes le même nom. Chaque constituant du tableau est appelé élément. Chaque élément d'un tableau est associé de manière unique à un entier. Grâce à cet entier, appelé index ou indice ou clé, il est possible d'accéder à un élément d'un tableau comme s'il s'agissait d'une variable normale.

On accède à un élément du tableau en utilisant les crochets [ ] qui contiennent l'index. Les entiers qui indexent le tableau n'ont pas obligation de se suivre. Si l'index est manquant, PHP utilise comme index le plus grand index existant incrémenté de 1.

Nous ne sommes pas tenus de déclarer quoi que ce soit avant d'initialiser un tableau, car tous ses éléments se créent à la volée.

Vous pouvez affecter des valeurs aux éléments du tableau de la même façon que pour une autre variable.

Vous pouvez initialiser les tableaux de deux manières différentes.

##### Structure d'un tableau

NomTableau[] = Valeur ;

ou

NomTableau = array( indice Valeur, indice => Valeur,...)

Exemple :

```
<?php
    $Auteur[] = "Molière";
    $Auteur[] = "Franz Kafka";
    $Auteur[] = "Albert Camus";

    echo $Auteur[0]."<br>"; //Affiche Molière
    echo $Auteur[1]."<br>"; //Affiche Franz Kafka
    echo $Auteur[2]."<br>"; //Affiche Albert Camus
?>
```

ou à l'aide de la fonction array().

```
<?php
    $Auteur = array("Molière", "Franz Kafka", "Albert Camus");
?>
```

Dans ces exemples l'indice n'ayant pas été renseigné, PHP s'en charge, ainsi la valeur "Molière", "Franz Kafka", "Albert Camus" prendront respectivement la valeur 0, 1, 2.

On peut également spécifier nous même les indices des tableaux.

```
<?php
    $Auteur[5] = "Molière";
    $Auteur[7] = "Franz Kafka";
    $Auteur[12] = "Albert Camus";

    echo $Auteur[5]."<br>"; //Affiche Molière
    echo $Auteur[7]."<br>"; //Affiche Franz Kafka
    echo $Auteur[12]."<br>"; //Affiche Albert Camus
?>
```

ou

```
<?php
    $Auteur = array(5 => "Molière", 7 => "Franz Kafka", 12 => "Albert Camus");

    echo $Auteur[5]."<br>"; //Affiche Molière
    echo $Auteur[7]."<br>"; //Affiche Franz Kafka
    echo $Auteur[12]."<br>"; //Affiche Albert Camus
?>
```

Dans ces exemples l'indice a été renseigné, ainsi la valeur "Molière", "Franz Kafka", "Albert Camus" prendront respectivement la valeur 5, 7, 12.

Exercice : (Voir annexe les Tableaux Exercice 1).

Soit le tableau des mois de l'année :

```
$mois=array('Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Août','Septembre','Octobre','Novembre','Décembre');
```

Afficher le mois de juillet à partir du tableau des mois de l'année sans utiliser de boucle.

## Fonction de traitement de tableaux

Php nous offre de nombreuses fonction pour manipuler les tableaux (reportez vous à la documentation pour la liste complète des fonction de tableaux.

En voici quelques une :

Array() ;	crée un tableau
Count(\$tableau) ;	renvoi le nombre d'éléments du tableau
Reset(\$tableau) ;	retourne au premier élément du tableau
End(\$tableau);	se positionne au dernière élément du tableau
Current(\$tableau);	renvoi l'élément en cour du tableau
Next(\$tableau);	se positionne sur l'élément suivant du tableau
Prev(\$tableau);	se positionne sur l'élément précédent du tableau
Extract(\$tableau) ;	importe des variables dans la table des symbole à partir d'un tableau.( attention au traitement des collisions)
in_array(\$tableau) ;	Renvoi TRUE si une valeur appartient à un tableau
array_search() ;	Recherche un élément dans un tableau et renvoi sa clé s'il existe
array_keys(\$tableau,\$chaine);	Renvoi toutes les clé d'un tableau
array_diff(\$tableau1, \$tableau2) ;	Crée un tableau contenant les éléments du premier tableau qui ne sont pas dans le second
Array_push((\$tableau,\$chaine) ;	Ajoute un ou plusieurs éléments à la fin d'un tableau
Print_r(\$tableau) ;	affiche le tableau dans un format qui montre les clés et les valeurs.

## Parcourir les tableaux avec les boucles

On peut passer en revue un tableau avec les boucles for, List et each et foreach :

Pour utiliser la boucle for le nombre de répétitions doit être prédéfini.

Pour ce faire nous utiliserons la fonction count() qui renvoi le nombre d'éléments du tableau :

Exemple :

```
<?php
$tableau =array('bleu','rouge','vert','violet','jaune');
for ($i = 0 ; $i < Count($tableau) ; $i++) {
    echo "clé = ".$i." valeur = ".$tableau[$i]."<br>";
}
?>
```

## List et each

### ➤ List

Tout comme array(), list() n'est pas une véritable fonction, mais une construction syntaxique, qui permet d'assigner une série de variables en une seule ligne

### ➤ Each

each() retourne la paire clé-valeur courante du tableau array et avance le pointeur de tableau. Cette paire est retournée dans un tableau de 4 éléments, avec les clés 0, 1, key, et value. Les éléments 0 et key contiennent le nom de la clé et, 1 et value contiennent la valeur.

Si le pointeur interne de fichier est au-delà de la fin du tableau, each() retourne FALSE et reste dans cette position, ce qui signifie que vous devrez utiliser reset() avant de réutiliser le tableau.

Exemple :

```
<?php
$couleurs=array('bleu','rouge','vert','violet','jaune');
reset($couleurs);
while(list ($Cle, $Valeur) = each ($couleurs))
{
    echo "$Cle => $Valeur<BR>";
}
?>
```

Exercice : (Voir annexe les Tableaux Exercice 2).

Dans une page PHP, créer un tableau contenant les voyelles et afficher les à l'aide de l'instruction list each.

Exercice : (Voir annexe les Tableaux Exercice 3).

Dans une page PHP construire le tableau \$List\_stagiaire, contenant les éléments suivants et afficher cette liste à l'aide de la boucle list et each:  
"Madame" "Durant"

```
"Mademoiselle" "Dupré"
"Alain"
"Monsieur" "Camus"
"Mon ami" "Jean-Marc"
```

### Foreach

PHP 4 a fait évoluer la boucle for, la boucle "Foreach".

la boucle "Foreach" permet de parcourir un tableau contenant un nombre inconnu d'éléments, elle effectue des itérations jusqu'à la fin du tableau. Elle prend deux formes.

La première :

#### Structure de la boucle foreach

```
Foreach($NomTableau As $ElementTableau)
{
    Instructions a exécuter...
}
```

Exemple :

```
<?php
    $couleurs=array('bleu','rouge','vert','violet','jaune');
    Foreach($couleurs As $ElementTableau)
    {
        echo $ElementTableau . "<BR>";
    }
?>
```

La seconde :

#### Structure de la boucle foreach

```
Foreach($NomTableau As $ValeurIndice => $ElementTableau)
{
    Instructions a exécuter...
}
```

La seconde forme est identique à la première sauf qu'elle met à notre disposition la valeur d'indice du tableau.

Exemple :

```
<?php
    $couleurs=array('bleu','rouge','vert','violet','jaune');
    Foreach($couleurs As $ValeurIndice => $ElementTableau)
    {
        echo "$ValeurIndice = $ElementTableau <BR>";
    }
?>
```

Lorsque foreach démarre, le pointeur interne de fichier est automatiquement ramené au premier élément du tableau. Cela signifie que vous n'aurez pas à faire appel à reset() avant foreach.

Exercice : (Voir annexe les Tableaux Exercice 4).

Créer une page PHP affichant les éléments contenus dans la variable globale \$\_SERVER en utilisant foreach .

Exercice : (Voir annexe les Tableaux Exercice 5).

Reprendre le tableau \$List\_stagiaire de l'exercice 3 du chapitre "les tableaux" et créer à l'aide d'une fonction PHP prédéfinie, une page PHP permettant d'afficher cette liste.

Rechercher si Madame Durant et Monsieur Dupont font partie de la liste et indiquer pour chacun, par une phrase, s'il ont été contactés par courrier.

## Tableaux Multidimensionnels

Un tableau multidimensionnel est un tableau composé de tableaux, vous pouvez imbriquer des tableaux tant que PHP ne manque pas de mémoire.

La définition d'un tableau multidimensionnel dans sa forme générale se présente de la manière suivante:

### Structure d'un tableau multidimensionnel

NomTableau = array( indice => array(Contenu du tableau), indice => array(Contenu du tableau),...)

Exemple :

```
<?php
$animaux=array("poissons" => array("sole", "merlan", "colin","sardine"),
               "serpents"  => array("orvet","couleuvre"),
               "oiseaux"   => array("merle", "serin", "pie")
               );
//ou comme ceci
$poissons = array("sole", "merlan", "colin", "sardine");
$serpents = array("orvet","couleuvre");
$oiseaux  = array("merle", "serin", "pie");

$animaux=array("poissons" =>$poissons,
               "serpents"  => $serpents,
               "oiseaux"   => $oiseaux
               );
echo "L'élément \$animaux['oiseaux'][1] vaut : ".$animaux['oiseaux'][1];
?>
```

Exercice : (Voir annexe Tableaux Multidimensionnels Exercice 1).

Avec la liste des stagiaires ci-dessous :

Madame Durant, Mademoiselle Dupré, Père Alain, Monsieur Camus, Jean-Marc Leblanc,  
et sachant que

Madame Durant est domiciliée au 21, rue Georges Brassens son n° de téléphone est : 01 42 44 12 30

Mademoiselle "Dupré est domiciliée au 54, rue Jacques Brel son n° de téléphone est : 01 14 42 32 40

Père Alain est domicilié 13, rue des Saint pères son n° de téléphone est : 01 52 64 72 80

Monsieur Camus est domicilié 79, rue du scribe son n° de téléphone est : 01 62 73 82 90

Jean-Marc Leblanc est domicilié rue des blanc manteaux son n° de téléphone est : 01 02 14 22 40

Dans une page PHP, construire un tableau multidimensionnel de la liste des stagiaires, leur adresse et N° de téléphone (\$annuaire\_stagiaires), et afficher la liste de chaque stagiaire accompagnée au dessous de son adresse.

Exercice : (Voir annexe Tableaux Multidimensionnels Exercice 2).

Avec la liste des stagiaires ci-dessous :

Madame Durant, Mademoiselle Dupré, Père Alain, Monsieur Camus, Jean-Marc Leblanc,  
et sachant que

Madame Durant est domiciliée au 21, rue Georges Brassens 94000 Créteil

Mademoiselle "Dupré est domiciliée au 54, rue Jacques Brel 75000 Paris

Père Alain est domicilié 13, rue des Saint pères 93000 Saint-Denis

Monsieur Camus est domicilié 79, rue du scribe 92160 Antony

Jean-Marc Leblanc est domicilié rue des blanc manteaux 91000 Montfermeil

Construire un tableau multidimensionnel de la liste des stagiaires et leur adresse (\$annuaire\_stagiaires), puis créer une page PHP affichant la liste de chaque stagiaire accompagnée au dessous de son adresse en affichant sur la même ligne le code postal et la ville.

Exercice : (Voir annexe Tableaux Multidimensionnels Exercice 3).

Avec la liste des stagiaires ci-dessous :

Madame Durant, Mademoiselle Dupré, Père Alain, Monsieur Camus, Jean-Marc Leblanc,  
et sachant que

Madame Durant est domiciliée au 21, rue Georges Brassens 94000 Créteil

Mademoiselle "Dupré est domiciliée au 54, rue Jacques Brel 75000 Paris  
Père Alain est domicilié 13, rue des Saint pères 93000 Saint-Denis  
Monsieur Camus est domicilié 79, rue du scribe 92160 Antony  
Jean-Marc Leblanc est domicilié rue des blanc manteaux 91000 Montfermeil  
Et la liste des inscrits au cours suivants :  
Inscrits au cours de Mathématiques : "Madame Durant" et "Monsieur Camus"  
Inscrits au cours de Physique : "Mademoiselle Dupré" et "Père Alain"  
Inscrits au cours de Français : "Jean-Marc Leblanc", "Mademoiselle Dupré" et "Monsieur Camus".

Dans une page PHP, construire deux tableaux multidimensionnels, un pour la liste des stagiaires avec leur adresse (\$annuaire\_stagiaires ) et un pour la liste des cours avec les stagiaires inscrits (\$cours\_stagiaires), puis afficher la liste de chaque cours et chaque stagiaire inscrits accompagne au dessous de leur adresse en affichant sur la même ligne le code postal et la ville.

## Tri de tableaux

PHP met à notre disposition plusieurs fonctions de tri en voici quelques unes :

Sort () ;	Trie un tableau et ne maintient pas l'association des index, supprime les clés existantes, et ne les réordonne pas.
Asort() ;	Trie un tableau et maintient l'association des index
Rsort() ; et arsort ;	Se comportent de la même façon que sort() et rsort(), sauf qu'elles renvoient le tableau dans un ordre inversé.
Ksort() ; et krsort() ;	Se comportent de la même façon que sort() et rsort(), sauf qu'elles renvoient le tableau trié dans l'ordre alphabétique de leur indice de chaîne.

Exemple de différent tri du tableau \$fruits :

```
$fruits = array("d"=>"papaye", "a"=>"orange", "b"=>"banane", "c"=>"ananas") ;
```

Avant le tri : d: papaye a: orange b: banane c: ananas	exemple de tri avec <b>sort()</b> sort(\$fruits); tri le tableau dans l'ordre alphabétique des ses valeurs et réordonne les indices	Après le tri : 0: ananas 1: banane 2: orange 3: papaye
Avant le tri d: papaye a: orange b: banane c: ananas	exemple de tri avec <b>asort()</b> asort(\$fruits); tri le tableau dans l'ordre alphabétique des ses valeurs et maintient l'association des index	Après le tri c: ananas b: banane a: orange d: papaye
Avant le tri : d: papaye a: orange b: banane c: ananas	exemple de tri avec <b>rsort()</b> rsort(\$fruits); tri le tableau dans l'ordre alphabétique inverse des ses valeurs et réordonne les indices	Après le tri : 0: papaye 1: orange 2: banane 3: ananas
Avant le tri d: papaye a: orange b: banane c: ananas	exemple de tri avec <b>arsort()</b> arsort(\$fruits); tri le tableau dans l'ordre alphabétique inverse des ses valeurs et maintient l'association des index	Après le tri d: papaye a: orange b: banane c: ananas
Avant le tri : d: papaye a: orange b: banane c: ananas	exemple de tri avec <b>ksort()</b> ksort(\$fruits); tri le tableau dans l'ordre alphabétique de ses indice de chaîne. et maintient l'association des indice	Après le tri : a: orange b: banane c: ananas d: papaye
Avant le tri d: papaye a: orange b: banane c: ananas	exemple de tri avec <b>krsort()</b> krsort(\$fruits); tri le tableau dans l'ordre alphabétique inverse de ses indice de chaîne. et maintient l'association des indice	Après le tri d: papaye c: ananas b: banane a: orange

Attention au tri lors de la récupération des valeurs d'une liste par leurs indices.

Exercice : (Voir annexe Tri des tableaux).

Soit le tableau des mois de l'année :

```
$mois=array('Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Août','Septembre','Octobre','Novembre',  
'Décembre');
```

Effectuer un tri avec chacune des fonctions de tri ci-dessus



## 4.6. Fonctions

Une fonction est une section de code défini une fois pour être réutilisé dans d'autres parties du programme. Elle permet d'écrire des applications particulièrement bien modulaires et structurées.

Les fonctions peuvent prendre des valeurs d'entrée appelées arguments, effectuer des opérations puis renvoyer une valeur. Elle transfère toutes les valeurs d'arguments dans de nouvelles variables appelées paramètres ces derniers pouvant être ensuite utilisés à l'intérieur de la fonction.

Il existe deux types de fonctions :

Les fonctions prédéfinies : PHP en dispose de nombreuses.

Les fonctions utilisateurs que l'utilisateur crée lui-même.

### Les fonction prédéfinies

Nous ne pouvons donner ici qu'un échantillon succinct des nombreuses fonctions de PHP.

Nous avons choisi celles les plus fréquemment employées.

Isset();	renvoi TRUE si la variable a été défini sinon FALSE .
Empty();	Renvoi TRUE si la variable est défini et vide sinon FALSE.
Strtoupper();	Renvoie une chaîne en majuscules.
Strtolower();	Renvoie une chaîne en minuscules.
Ucfirst();	Met le premier caractère en majuscule.
Exit();	Termine le script courant.
Die();	Envoi le message passé en argument avant de terminer le script.
Unset();	Détruit une variable.

### Les fonctions utilisateurs

Une fonction peut être défini n'importe où dans le script, mais qu'une seule fois.

Pour définir une fonction vous devez lui donner un nom.

Les fonctions sont déclarées à l'aide de l'instruction function, selon la structure suivante :

#### Structure de la définition d'une fonction

```
function nom_fonction ([arguments])
{
    corps de la fonction
}
```

Le mot-clé return met fin à la fonction, si vous voulez que la fonction renvoie une valeur alors placez celle-ci après le mot-clé return sur la dernière ligne de la fonction.

Exemple :

```
<?php
function charge($salaire)
{
    $salaire = $salaire - (($salaire/100)*20);
    return $salaire;
}
echo charge(2000); //affiche 1600
?>
```

Vous pouvez fournir plusieurs paramètres séparés par une virgule :

Exemple :

```
<?php
function charge($salaire, $TauxCharge)
{
    $salaire = $salaire - (($salaire/100)* $TauxCharge);
    return $salaire;
}
$salaire=2000;
$TauxCharge=20;
echo charge($salaire,$TauxCharge)."<br>"; //affiche 1600
echo $salaire; //affiche 2000
?>
```

Exercice : (Voir annexe les fonctions Exercice 1).

Dans une page PHP, déclarer deux variable php et affecter leur la valeur de votre prénom et nom.  
Créer la fonction `get_msg($chaîne, $couleur)`, qui renverra la chaîne passé en paramètre dans une couleur différente, passée aussi en paramètre, sachant que le rouge= "#FF0000", le bleu="#0000FF", le vert="#00FF00" et le blanc="FFFFFF", puis, générer le texte suivant,  
Je me présente<BR>  
Je m'appel "votre prénom" "votre nom"  
avec le nom en rouge et le prénom en vert.  
Puis le même texte à l'aide de JavaScript.  
Vérifier en appelant la fonction.

Il existe une autre méthode pour passer des paramètres à une fonction, qui implique que la valeur modifiée à l'intérieur de la fonction le soit aussi à l'extérieur. Cela s'appelle passer un argument par référence. Pour indiquer à PHP que nous voulons utiliser cette méthode, nous devons ajouter & ( et Commerciale ) au début de la variable passée.

Vous pouvez aussi appeler les fonctions avec des paramètres passé par référence, toutefois cette méthode n'est valable que si la directive « `allow_call_time_pass_reference` » du fichier de configuration de PHP est égale à On.

Exemple :

```
<?php
function charge(&$salaire, $TauxCharge)
{
    $salaire = $salaire - (($salaire/100)* $TauxCharge);
    return $salaire;
}
$salaire=2000;
$TauxCharge=20;
echo $salaire."<br>"; //Salaire avant la fonction affiche 2000
echo charge($salaire,$TauxCharge)."<br>"; //affiche 1600
echo $salaire; //Salaire après la fonction affiche 1600
?>
```

## Portée des variables

Lorsqu'une variable est créée dans une page PHP, on dit qu'elle a une durée de vie identique à celle d'une page web. Mais lors de l'utilisation d'une variable à l'intérieur d'une fonction sa durée de vie est égale à la durée de vie de la fonction, c'est à dire qu'elle cesse d'exister dès que la fonction se termine. Les variables déclarées à l'intérieur d'une fonction sont dites locales et elles ne sont pas visibles (accessibles) en dehors de la fonction.

Il existe plusieurs niveaux de définition de variable :

Le niveau **global**, qui permet à une variable d'être visible dans la fonction et à l'extérieur de la fonction.

Le niveau **static** permet de définir une variable locale à la fonction, qui persiste durant tout le temps d'exécution du script.

Le niveau **local**, utilisé par défaut, permet de définir une variable locale classique.

## Fonction imbriquer

Vous pouvez imbriquer les fonctions les unes dans les autres, toutefois attention à ne pas définir plusieurs fois la même fonction.

Exemple :

```
<?php
    echo "Si nous voulons calculer le montant de la retraite, une fois les charges
                                                déduites :<br>";

    function retraite($Total, $TauxCharge)
    {
        function charge($salaire, $TauxCharge)
        {
            return $salaire - (($salaire/100)* $TauxCharge);
        }
        $AprèsCharges = charge($Total,$TauxCharge);
        return charge($Total,$TauxCharge) - (( $AprèsCharges/100)*3);
    }

    $Total=2000;
    $TauxCharge=20;

    echo "Total = ".$Total."<BR>"; //affiche Total = 2000
    echo "Retraite = ".retraite($Total,$TauxCharge)."<br>"; //affiche retraite = 1552
    echo "AprèsCharges = ".charge($Total,$TauxCharge)."<br>"; //affiche après charge =
                                                                1600
    // echo "Retraite = ".retraite($Total,$TauxCharge)."<br>"; //affiche une erreur
?>
```

Si vous voulez accéder à une variable située à l'intérieur d'une fonction, vous devez utiliser l'instruction **global**, cela indique à PHP que ce n'est pas une nouvelle variable locale mais une variable globale qui pourra être utilisée ailleurs dans le script.

Exemple :

```
<?php
    function charge($salaire, $TauxCharge)
    {
        global $salaire;
        $salaire =4000;
        $salaire -= (($salaire/100)* $TauxCharge);
        return $salaire;
    }

    $salaire=2000;
    $TauxCharge=20;

    echo $salaire."<BR>"; //affiche 2000
    echo charge($salaire,$TauxCharge)."<br>"; //affiche 3200
    echo $salaire."<BR>"; //affiche 3200
?>
```

Exercice : (Voir annexe les fonctions Exercice 2).

Reprendre le tableau des stagiaires (\$annuaire\_stagiaires) de l'exercice 2 du chapitre tableaux multidimensionnel, parcourir ce tableau à l'aide d'une boucle puis pour chaque élément du tableau appeler la fonction (ajoute\_stagiaire()) que vous aurez créé et Afficher le nombre de stagiaires femmes, le nombre de stagiaires hommes et enfin le nombre total de stagiaires.

NB : Aidez-vous de la fonction **eregi**(« occurrence à rechercher », \$chaîne) qui retourne **true** si une occurrence a été trouvée dans la chaîne et **false** dans le cas contraire

Exercice : (Voir annexe les fonctions Exercice 3).

Reprendre l'exercice 2 et ajoutez une fonction (`moyenne_parity()`) qui permette d'afficher la parité hommes/femmes sous forme de moyenne (%).

## Les inclusions

Avec les fonctions `include()` et `require()`, PHP nous permet d'inclure des segments de code ou de texte à un endroit donné d'une page web.

Vous pouvez utiliser des fichiers inclus pour inclure du texte, du code HTML ou des scripts PHP. C'est de cette façon que l'on pourra définir des éléments qui seront partagés par toutes les pages d'un site web. Ainsi lorsque l'on voudra modifier l'un d'entre eux, il suffira de modifier dans le fichier où il est défini pour que cette modification soit automatiquement répercutée dans tous les scripts qui incluent le fichier.

PHP met deux instructions à notre disposition pour inclure des fichiers `include` et `require` qui prennent la forme suivante :

```
include("monfichier") ;
```

```
require("monfichier") ;
```

Exemple :

```
<?php
    include("test.txt") ;
    //ou
    require("test.txt") ;
?>
```

Depuis la version 4.3 de PHP ces deux instructions sont identiques, hormis dans leur gestion des erreurs. `Include()` produit une Alerte (warning) tandis que `require()` génère une erreur fatale. En d'autres termes, n'hésitez pas à utiliser `require()` si vous voulez qu'un fichier d'inclusion manquant interrompe votre script. `include()` ne se comporte pas de cette façon, et le script continuera son exécution.

Il existe aussi une variante aux instructions `include` et `require`, il s'agit de `include_once` et `require_once` qui ont un comportement identique hormis qu'elle ne s'exécute qu'une fois dans le script. Ces instructions sont utilisées de préférence lorsque le fichier doit être inclus ou évalué plusieurs fois dans un script, ou bien lorsque vous voulez être sûr qu'il ne sera inclus qu'une seule fois, pour éviter des redéfinitions de fonction.

Exercice : (Voir annexe Les inclusions Exercice).

Créer le fichier PHP "header.inc" qui permettra d'afficher grâce à une instruction d'inclusion un message de bienvenue sur chaque page.



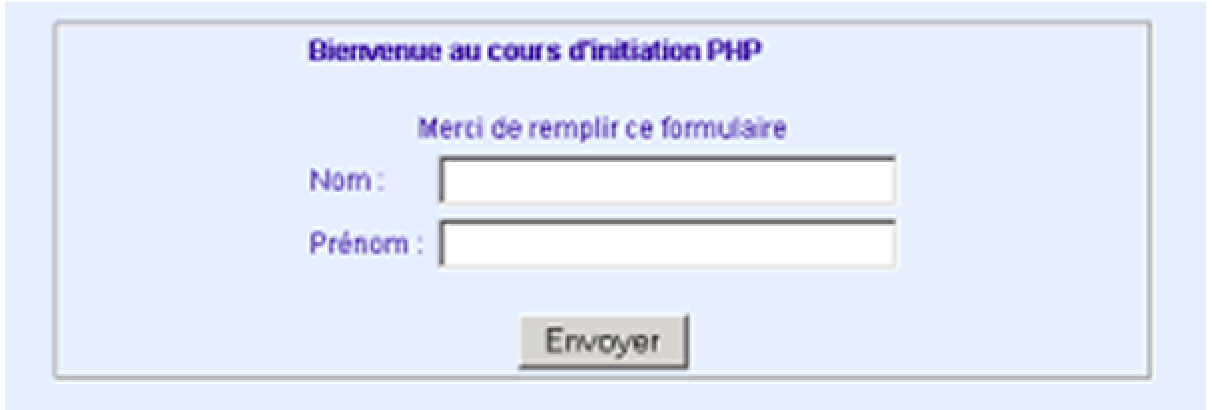
```

<input type="submit" value="Envoyer" />
</form>
</body>
</html>

```

Exercice : (Voir annexe Les Formulaires Exercice 1).

Créer le formulaire comme le montre la figure ci-dessous, et à l'aide d'une boucle afficher dans une page PHP un message de bienvenue composé les éléments saisis en utilisant la méthode GET puis la méthode POST.



### urlencode() et urldecode()

Lorsqu'on appelle une page avec la méthode GET, une chaîne de caractères est aboutée à l'URL, contenant la suite de couples noms/valeurs des éléments du formulaire séparés par des caractères &. Cette suite est codée de façon que n'y figure aucun espace (il y sont remplacés par des « + »).

En outre, les caractères de ponctuation et les diacritiques sont remplacés par leur code ASCII.

Exemple :

```
<?php
    prenom=Arthur-Jérémie
    cours[]=Anglais

    sera transmis dans l'URL comme ceci :

    prenom=Arthur-J%E9r%E9mie&cours%5B%5D=Anglais
?>
```

Grâce aux fonctions urldecode et urlencode on peut ainsi transformé la chaîne de l'url afin qu'elle devienne lisible.

**urldecode()** décode toutes les séquences %## et les remplace par leur valeur. La chaîne ainsi décodée est retournée.

**urlencode()** retourne une chaîne dont les caractères non alpha-numériques (hormis -\_.) sont remplacés par des séquences commençant par un caractère pourcentage (%), suivi de deux chiffres hexadécimaux. Les espaces sont remplacés par des signes plus (+). Ce codage est celui qui est utilisé pour poster des informations dans les formulaires HTML.

Exemple :

On commence par récupérer la chaîne de l'URL qui se trouve dans la variable global \$\_SERVER["QUERY\_STRING"], et on lui applique la fonction split qui va récupérer les occurrence de chaînes séparées par des & dans un tableau, puis on va isoler le nom de la valeur de façon classique à l'aide de la boucle while.

Exemple :

```
<?php
    $a = split('&', $_SERVER["QUERY_STRING"]);
    $i = 0;
    while ($i < count ($a)) {
        $b = spilt('=', $a [$i]);
        echo 'La valeur du paramètre ', htmlspecialchars(urldecode($b [0])), ' est ',
            htmlspecialchars(urldecode($b[1])), "<br>";

        $i++;
    }
?>
```

Exercice : (Voir annexe Les Formulaires Exercice 2).

Créer le formulaire de cet exemple, et à l'aide de la fonction urldecode() afficher dans une page PHP un message de bienvenue composé les éléments saisis.

Bienvenue au cours d'initiation PHP

Merci de remplir ce formulaire

Nom :

Prénom :



## Recupération des variables

Nous allons voir maintenant sous forme d'exemples et d'exercices les contrôles HTML les plus courants et la manière de les traiter avec PHP.

Comme nous l'avons déjà vu, lorsque l'utilisateur clique sur le bouton "Envoyer", il transmet au serveur http le formulaire selon la méthode mentionnée par l'attribut "Method" de la balise "Action", puis, PHP interprète le code et redirige l'action sur la page mentionnée par l'attribut "Action" de la balise "FORM". Après l'analyse de cette page (parser), il renvoi la réponse à l'utilisateur via le serveur HTTP.

Exercice : (Voir annexe Les Formulaires Exercice 3).

Afin d'illustrer ces mécanismes, créer le formulaire d'inscription (Voir Figure 1) en HTML et récupérer sur une autre page PHP les valeurs saisies par l'utilisateur, en utilisant la méthode "GET" puis la méthode "POST".

En créant deux pages distinctes on obtient la séparation du code HTML constituant les éléments d'interface proposés au client (IHM), du code PHP qui gèrent les règles de gestions (validations, calculs, contrôles, etc...), ce qui donne une meilleur visibilité et facilite la maintenance puisque l'on pourra ainsi intervenir sur l'interface sans modifier le code et vis et versa.

**Formulaire d'inscription**

Nom :	<input type="text"/>
Prénom :	<input type="text"/>
Adresse :	<input type="text"/> <input type="text"/> <input type="text"/>
Courriel :	<input type="text"/>
<input type="button" value="Envoyer"/>	

Figure 1

## Les contrôles HTML

### Les bouton radio

Les bouton radio se créent avec la balise HTML :

```
<input type="radio" value="" checked="checked" name="" >
```

Pour récupérer la valeur d'un groupe de boutons radio, il faut que tous les boutons radio aient le même nom, ainsi PHP créera qu'une seule variable qui ne pourra avoir qu'une seule valeur et sera traitée comme n'importe qu'elle autre variable.

Exercice : (Voir annexe Les Formulaires Exercice 4).

Créer un formulaire identique au précédent, en lui ajoutant un groupe de boutons radio permettant à l'utilisateur de choisir la civilité, comme le montre la figure 2.

Formulaire d'inscription	
Madame :	<input type="radio"/>
Mademoiselle :	<input type="radio"/>
Monsieur :	<input type="radio"/>
Nom :	<input type="text"/>
Prénom :	<input type="text"/>
Adresse :	<input type="text"/>
Courriel :	<input type="text"/>
<input type="button" value="Envoyer"/>	

Figure 2

### Les case à cocher et les listes de sélection.

Les case à cocher et les listes de sélection permettent d'effectuer simultanément plusieurs choix. Il est donc pratique de considérer que la réponse sera constitué par un tableau, il convient donc que l'attribut "name" des éléments "<Select>" et "<Input>" soient suivis de crochets. Ainsi PHP transformera ses variables sous forme de tableau et nous pourrons leur appliquer tous les traitements que l'on a vu au chapitre tableaux ( les boucles, les tris, etc.. ).

Les case à cocher se créent avec la balise HTML :

```
<input type="checkbox" value="Physique" name="cours[]">
```

Les liste de selection se créent avec la balise HTML :

```
<select name="coursoption[]" MULTIPLE >
  <option>"élément 1 de la liste"</option>
  <option>"élément 2 de la liste"</option>
</select>
```

Exercice : (Voir annexe Les Formulaires Exercice 5). Pour compléter notre page nous allons ajouter une liste et des case à cocher, pour ce faire, réaliser le formulaire en HTML (Voir Figure 3), puis récupérer sur une autre page PHP les valeurs saisies.

Formulaire d'inscription	
Madame : <input type="checkbox"/>	Mademoiselle : <input type="checkbox"/>
Monsieur : <input type="checkbox"/>	
Nom :	<input type="text"/>
Prénom :	<input type="text"/>
Adresse	<input type="text"/>
Courriel :	<input type="text"/>
Choix des cours:	
Principaux : Mathématiques : <input type="checkbox"/> Physique : <input type="checkbox"/> Français : <input type="checkbox"/>	Optionnels <input type="text" value="Anglais"/> <input type="text" value="Espagnol"/> <input type="text" value="Art plastique"/>
<input type="button" value="Envoyer"/>	

Figure 3

Exercice : (Voir annexe Les Formulaires Exercice 6).

Soit le tableau :

```
$List_stagiaire=array(0=>"Madame Durant", 1=>"Mademoiselle Dupré", 2=>"Alain", 3=>"Monsieur Camus", 4=>"Mon ami Jean-Marc");
```

Afficher les stagiaires issus du tableaux dans une liste de sélection, puis à l'aide d'un bouton permettre à l'utilisateur d'alimenter une seconde liste des titulaire comme le montre la figure 4

### Liste des stagiaires titulaires




Figure 4

### Validation des formulaires

Avec le formulaire de la figure 3 nous avons permis à l'utilisateur de saisir les renseignements qu'il souhaitait puis de les envoyer, mais lorsque que le formulaire était incomplet ou que la saisie était erronée, aucune vérification étant faite celui-ci était quand même envoyer.

Nous allons donc demander à PHP de faire les vérifications nécessaires avant d'afficher la page HTML. Rappelons nous que l'on peut envoyer au serveur aussi bien une page HTML qu'une page PHP.

Exercice : (Voir annexe Les Formulaires Exercice 7).

Reprendre le formulaire de la figure 3, après avoir ajouter un champ pour le code postal, un champ pour la ville et un autre pour le téléphone, ajoutez une fonction dans la page PHP qui permettra de vérifier que tous les champs ont été saisis correctement. Si c'est le cas la fonction renverra TRUE et on affichera le résultat, sinon la fonction renverra FALSE et on réaffichera la page HTML accompagnée d'un message indiquant à l'utilisateur les champs incorrects.

Dans le cas ou l'on réaffiche la page HTML pour correction, on évitera à l'utilisateur de saisir à nouveau les

Madame : <input type="checkbox"/> Mademoiselle : <input type="checkbox"/> Monsieur : <input type="checkbox"/>	
Nom :	<input type="text"/>
Prénom :	<input type="text"/>
Adresse :	<input type="text"/>
Code Postal :	<input type="text"/> Ville <input type="text"/>
Courriel :	<input type="text"/>
Choix des cours :	
Principaux : Mathématiques : <input type="checkbox"/> Physique : <input type="checkbox"/> Français : <input type="checkbox"/>	Optionnels <input type="text"/>
<input type="button" value="Envoyer"/> <input type="button" value="Réinitialiser"/>	

Figure 5

informations déjà fournis.**Formulaire d'inscription**

Dans cette exemple, on envoie une page PHP qui détecte tout d'abord s'il s'agit d'un premier envoi ou non en vérifiant la méthode POST ou GET grâce à la variable `$_SERVER['REQUEST_METHOD']`. Si c'est le cas on exécute la fonction de vérification qui nous informe si toutes les saisies sont correctes ou non. Ainsi selon le résultat de la fonction on affichera la page de résultat ou la page de saisie.

**Htmlelentities() et htmlspecialchars**

Ces fonctions jouent un rôle important dans la traduction de certains caractères destinés à être affichés par le navigateur. La première convertit la plupart des diacritiques ou autrement dit caractères spéciaux en entités de caractères équivalentes. Si nous disons « la plupart », c'est que certains caractères tels que € ou « oe » ne sont pas traduits. La seconde convertit les caractères ayant un sens particulier dans les commandes HTML :

Le ET commercial (&) devient *&amp;* ; .

Les chevrons ou balises (< et >) deviennent respectivement \$lt ; et \$gt ;.

Le guillemet (") devient &quot ; si le second argument ne vaut pas ENT\_QUOTES.

Exemple

```
<?php
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
ou
$new = htmlelentities("<a href='test'>Test</a>", ENT_QUOTES);
?>
```

NB : Si nous n'utilisons pas cette dernière fonction lorsque l'on récupère les éléments saisis dans les formulaires nous avons de forte chance d'avoir un comportement inattendu.

Exercice : (Voir annexe Les Formulaires Exercice 8).

Reprendre le formulaire de l'exercice 6 (Figure 4), et à l'aide de la fonction htmlelentities() ou htmlspecialchars, supprimez les problèmes relatifs aux saisies des balise HTML dans les champs.

## Les en-têtes HTTP

Lors de chaque échange par le protocole HTTP entre votre navigateur et le serveur, des données dites d'en-têtes contenant des informations sur les données à envoyer (dans le cas d'une requête) ou envoyées (dans le cas d'une réponse). Les informations en question, généralement sur une page web ou une image, suivent ces en-têtes. Les en-têtes HTTP permettent aussi d'effectuer des actions sur le navigateur comme le transfert de cookies ou bien une redirection vers une autre page.

Ces en-têtes sont les premières informations envoyées au navigateur (pour une réponse) ou au serveur (dans le cas d'une requête), elles se présentent sous la forme:

en-tête: valeur

La syntaxe doit être rigoureusement respectée, c'est-à-dire qu'aucun espace ne doit figurer entre le nom de l'en-tête et les deux points (:). Un espace doit par contre figurer après celui-ci !

PHP fournit une fonction permettant d'envoyer très simplement des en-têtes HTTP manuellement du serveur au navigateur (il s'agit alors d'une réponse HTTP). La syntaxe de cette fonction est la suivante:

booléen header(chaîne en-tête HTTP)

Etant donnée que les en-têtes HTTP sont les premières informations envoyées, la fonction header() doit être utilisée avant tout envoi de données HTML au navigateur (le script qui la contient doit donc être placé avant la balise <HTML> et avant toute fonction echo(), print ou printf())

Voici quelques utilisations possibles de la fonction header():

Exemple :

Pour rediriger le navigateur vers une nouvelle page:

```
<?PHP
    header("location: http://www.commentcamarche.net/");
?>
```

Exemple :

Pour envoyer au navigateur une image créée à la volée (pour faire un compteur de statistiques ou bien un histogramme dynamique par exemple) :

```
<?php
    header("Content-Type: image/gif");
    // code générant l'image
    imagegif($image); // envoi de l'image au navigateur
?>
```

## Récupérer les en-têtes de la requête

Alors que la fonction header() permet d'envoyer des en-têtes HTTP au navigateur, PHP fournit une seconde fonction permettant de récupérer dans un tableau l'ensemble des en-têtes HTTP envoyées par le navigateur.

Voici la syntaxe de cette fonction:

Tableau getallheaders();

Le tableau retourné par la fonction contient les en-têtes indexés par leur nom. Voici un script permettant par exemple de récupérer des en-têtes particuliers.

Exemple :

```
<?php
    $entetes = getallheaders;
    echo $entetes["location"];
?>
```

## 6. Base de données

### **Pourquoi utiliser une base données ?**

Pour stocker des informations, il est plus avantageux d'utiliser une base données relationnelle qu'un fichier brut. En effet les RDBMS (Relational Database Management Systems) présentent les atouts suivants :  
Ils permettent d'accéder aux données plus rapidement qu'avec des fichiers bruts ;  
Vous pouvez les interroger très facilement pour récupérer des ensembles de données répondant à certains critères ;  
Ils possèdent des mécanismes intégrés permettant de gérer les accès simultanés, pour que le programmeur n'est pas besoin de s'en occuper ;  
Ils permettent d'accéder aléatoirement à vos données ;  
Ils possèdent des systèmes de privilèges intégrés.  
L'utilisation d'une base de données relationnelle vous permet de répondre rapidement et simplement à des questions qui vous aideront à améliorer votre site.

### **Architecture externe d'une base de données web**

Les étapes suivantes donnent le cheminement d'une transaction de base de données Web typique:

- Le navigateur Web d'un utilisateur envoie une requête http pour une page Web particulière;
- Le serveur reçoit la requête, récupère le fichier .php et le passe au moteur PHP afin qu'il soit traité ;
- Le moteur PHP commence à analyser le script. A l'intérieur de ce script se trouve une commande permettant de se connecter à la base de données et d'exécuter une requête. PHP ouvre une connexion vers le serveur de la base de données et transmet la requête appropriée ;
- Le serveur de la base de données reçoit la requête de base de données et la traite, puis renvoie les résultats au moteur PHP ;
- Le moteur PHP termine l'exécution du script, ce qui consiste généralement en un formatage des résultats de la requête en HTML. Il envoie ensuite le fichier HTML obtenu au serveur Web.
- Le serveur Web transmet la page HTML au navigateur, pour que l'utilisateur puisse voir le résultat.

### **6.1. Accès aux bases de données**

#### **Support PHP de connexion aux bases de données**

PHP prend en charge des API qui donnent accès à un grand nombre de bases de données, comme Oracle, Sybase, PostgreSQL, MySQL, etc. Toutefois les programmes PHP ne peuvent toutes les utiliser pour accéder directement aux données. ODBC (Open DataBase Connectivity, middle ware de connexion) est une API (Application Programming Interface) standard permettant d'accéder à une base de données prise en charge par PHP et d'écrire des applications de bases de données génériques, c'est-à-dire que le code de l'application fonctionne pour toutes les bases de données prenant en charge la norme ODBC.

#### **Rudiments du langage SQL**

SQL (Strutured Query Language) est un langage de programmation standard permettant d'accéder et de manipuler les informations d'une base de données relationnelle. C'est une norme ANSI et ISO acceptée par la plupart de bases de données relationnelles.

#### **Requêtes de définition de données**

Ces instructions SQL permettent de modifier le schéma de la base de données en créant ou en modifiant des objets dans cette dernière.

- CREATE :  
permet de créer une base de données ou une table dans une base de données existante.
- DROP :  
permet de détruire une base de données ou une table dans une base de données.

#### **Instructions de manipulation de données**

Ces instructions SQL permettent de modifier le contenu d'une base de données.

- INSERT :  
permet de compléter le lignes d'une table de base de données.
- REPLACE :  
permet de remplacer un ancien enregistrement par un nouveau (instruction spécifique à la base MySQL).
- UPDATE :



permet de modifier une ou plusieurs colonnes dans un enregistrement.

➤ DELETE :

permet de supprimer d'une table un ou plusieurs enregistrements correspondant à une condition particulière.

**Instruction de recherche**

➤ SELECT :

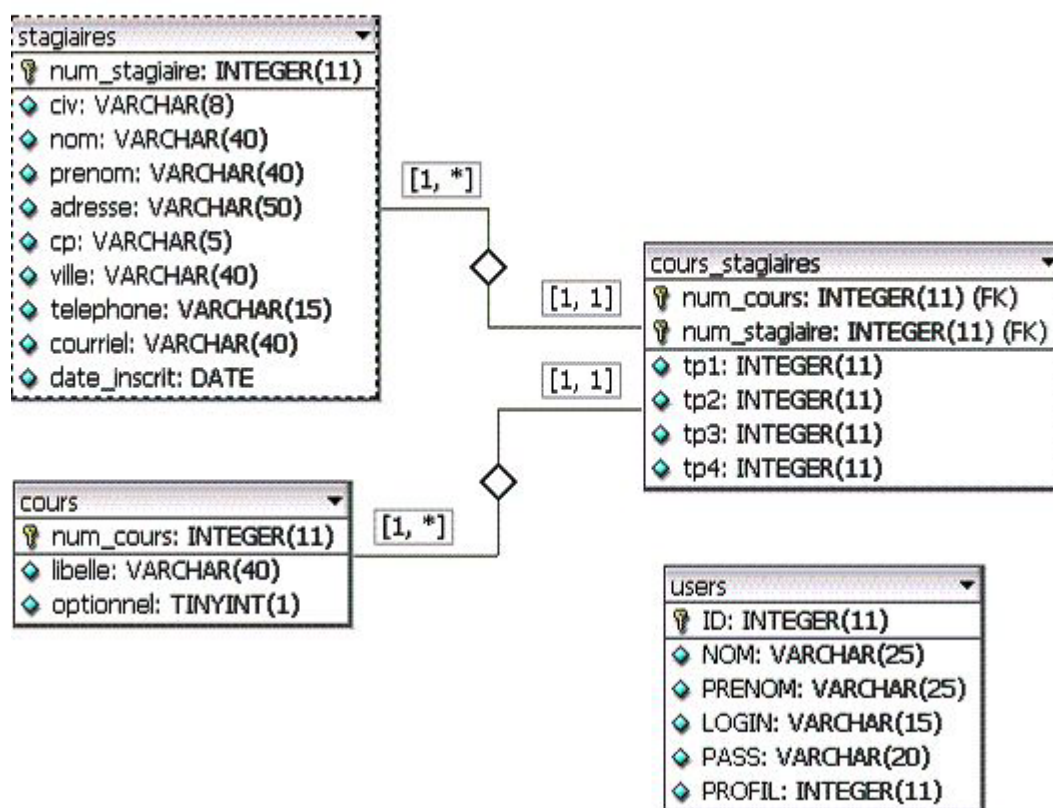
permet d'effectuer des recherches dans les bases de données MySQL à partir d'applications web.

**PhpMyadmin**

EasyPHP est livré avec l'outil "PhpMyadmin", qui permet d'effectuer toutes les opérations sur une base de donnée Mysql de manière graphique. L'un des gros avantages cet outil, c'est qu'il met à votre disposition la syntaxe SQL modifiée pour PHP lors des opérations que vous effectuez.

Nous allons étudier une base de donnée ("formation") gérant les stagiaires d'un centre de formation selon le modèle ci-dessous :

**Model de la base formation**  
(MDL selon la méthode Merise)



Après avoir installé la base de données "formation" à l'aide de PhpMyadmin, nous allons voir quelques éléments du langage SQL et comment se servir de cette base avec PHP.

**Installer la base formation**

Il y a deux manières d'installer la base formation, à l'aide de PhpMyadmin ou via PHP :

Avec PhpMyadmin :

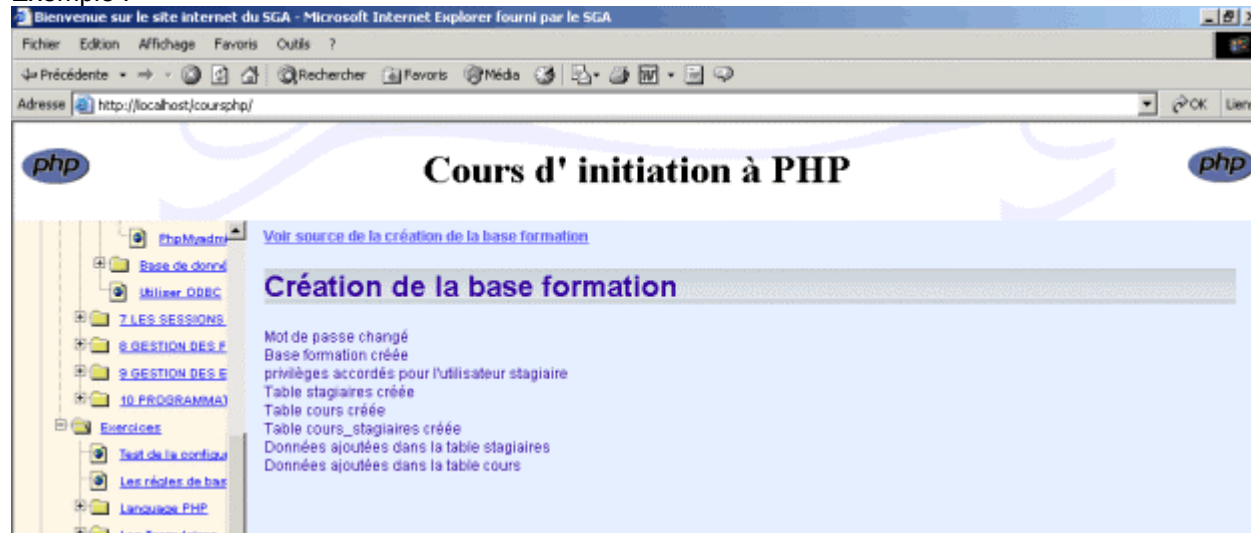
Lancer PhpMyadmin avec un clic droit sur l'icône EasyPHP, puis configuration et enfin PhpMyadmin. Importer le fichier SQL de création de la base Stagiaires (formation.sql) en cliquant sur l'icône "sql" de la fenêtre de gauche puis sur "importe les fichiers".

Avec PHP :

Le fichier create\_base.php contient toutes les commandes pour créer la base. Ainsi on pourra créer la base en appelant le script par l'url :

[http://127.0.0.1/coursphp/exercice/create\\_base.php](http://127.0.0.1/coursphp/exercice/create_base.php)

Exemple :



Nous étudions les détails des instructions dans le chapitre suivant.

## 6.2. Base de données MySQL

MySQL est un serveur de bases de données réduit, compact et convivial. Il est disponible pour les plates-formes UNIX, Windows NT, 2000 et Windows 95/98. Il est inutile de déclarer quoi que se soit pour utiliser Mysql avec PHP4, puisque que PHP4 supporte nativement cette base de données.

Pour PHP5 SqlLite à pris le relais, il faut donc charger le module Mysql.

Chargement du module Mysql avec PHP5 :

Modifier le fichier de configuration "php.ini" dans la section Extensions ajouter la ligne suivante :

```
extension=php_mysql.dll
```

N'oubliez pas de redémarrer le serveur http pour que les modifications soit prises en compte.

### Accès à la base de données MySQL à partir du Web avec PHP

Dans tous les scripts utilisés pour accéder à une base de données à partir du Web, il faut respecter les étapes suivantes :

- établir une connexion vers la base de données appropriée ;
- envoyer une requête vers la base de données ;
- récupérer et traiter le résultat pour le présenter à l'utilisateur;
- fermer la connexion.

#### Etablir une connexion

- La fonction `mysql_connect()` :

Vous permet d'établir une connexion.

```
int mysql_connect([string nom_serveur], [string nom_utilisateur], [string mot_passe]);
```

Tous les arguments sont optionnels. S'ils ne sont pas précisés, la fonction se servira des valeurs par défaut : "localhost" pour l'hôte, le nom de l'utilisateur sous lequel PHP est exécuté, et aucun mot de passe.

Cette fonction renvoie un identificateur de lien vers votre base de données MySQL (réutilisé par la suite) en cas de succès, et false en cas d'échec.

Exemple :

```
<?php
$db = mysql_connect("localhost", "root", "");
// l'identificateur de lien est enregistré dans $db
?>
```

De la même manière, la fonction `mysql_pconnect()` permet d'établir une connexion, mais renvoie une connexion persistante vers la base de données, alors que la connexion normale est fermée lorsque l'exécution du script est terminée.

Exercice : (Voir annexe Etablir une connexion Exercice 1).

Créer un fichier PHP permettant d'ouvrir une connexion avec MySQL, avec comme paramètres :

Nom de l'utilisateur : "stagiaire"

Mot de passe : "stagiaire"

#### Choisir une base de données existante

- La fonction `mysql_select_db()` :

Vous permet de sélectionner une base de données existante .

```
int mysql_select_db( string nom_base, [int identificateur_connexion]);
```

L'argument `identificateur_connexion`, spécifiant le lien vers la base de données, est optionnel : par défaut le dernier lien ouvert sera utilisé, et si aucun lien n'a été ouvert, le lien par défaut sera ouvert comme si vous aviez appelé `mysql_connect()`.

Exercice : (Voir Choisir une base de données existante Exercice 1).

Créer un fichier PHP permettant d'établir une connexion avec la base formation., avec comme paramètres :

Nom de l'utilisateur : "stagiaire"

Mot de passe : "stagiaire"

Exercice : (Voir Choisir une base de données existante Exercice 2).

Créer le fichier "connect.inc" qui servira autant que de besoin sous forme de fichier inclus à se connecter à la base formation les paramètres de connexions sont identiques aux exercices précédents

**Déconnexion à la base de données**

- La fonction `mysql_close()` :

Vous permet de vous déconnecter de la base de données :

```
boolean mysql_close([mixed identificateur_connexion]);
```

Attention : si la connexion est persistante, la fonction `mysql_close()` ne permet pas de la fermer.

## Envoyer une requête vers la base de données

- La fonction `mysql_query()` :

Vous permet d'envoyer une requête la base de données. La requête que vous voulez exécuter peut être configurée avant l'appel de cette fonction ou dans la fonction :

### Structure d'une requête SQL

```
$query = "... requête SQL ...";
$resultat = mysql_query ($query,[identificateur_connexion]);
ou
$resultat = mysql_query ("... requête SQL ...",[identificateur_connexion]);
```

- La fonction `mysql_db_query ()` :

Vous pouvez aussi appeler la fonction `mysql_db_query()` qui permet de spécifier la base de données que vous souhaitez interroger :

```
mixed mysql_db_query( string nom_base, string requete, [mixed identificateur_connexion]);
```

En cas de succès, ces deux fonctions renvoient :

- dans le cas d'une requête d'interrogation, un identificateur de résultat qui permet de récupérer les résultats des requêtes,
- dans le cas d'autres requêtes, 1.

En cas d'échec elles renvoient `false`.

L'identificateur de résultat sert de clé pour accéder à zéro, une ou plusieurs lignes renvoyées par la requête. Plusieurs fonctions permettent de récupérer ces résultats à partir de cet identificateur.

Exercice : (Voir annexe Envoyer une requête vers la base de données, exercice 1 ).

Créer une page PHP qui, à l'aide du fichier `connect.inc`, nous permettra d'exécuter une requête interrogeant la table des stagiaires :

PHP dispose de plusieurs fonctions permettant de récupérer les valeurs d'une table. Ces fonctions récupèrent les noms de champs et leur valeur et les stocks dans une variable tableau passée en paramètre.

- `Mysql_fetch_array()`

```
array mysql_fetch_array ( resource result_identifieur [, int result_type])
```

Retourne une ligne de résultat sous la forme d'un tableau associatif.

C'est une version étendue de `mysql_fetch_row()`. En plus d'enregistrer les données sous forme d'un tableau à indice numérique, elle peut aussi les enregistrer dans un tableau associatif, en utilisant les noms des champs comme indices

Autrement dit, vous pouvez récupérer les valeurs, en indiquant soit l'indice numérique soit le nom de champ de la table

Exemple :

```
<?PHP
    $resultat = mysql_query("select * from personne", $conn);
    while ($row = mysql_fetch_array ($resultat))
    {
        echo "<br>". $row["num_personne"]. " ";
        echo $row["nom"]. " ";
        echo $row["prenom"]. " <br>";
    }
?>
```

- `mysql_fetch_row()`

```
array mysql_fetch_row ( resource result_identifieur )
```

retourne un tableau énuméré qui correspond à la ligne demandée, ou `FALSE` s'il ne reste plus de ligne. Vous pouvez récupérer les valeurs par l'indice numérique.

Exemple :

```
<?PHP
    $resultat = mysql_query("select * from personne", $conn);
    while ($row = mysql_fetch_row ($resultat))
    {
```

```

echo "<br>".$row[0]." ";
echo $row[1]." ";
echo $row[2]." <br>";
}
?>

```

#### ➤ mysql\_fetch\_assoc()

array mysql\_fetch\_assoc ( resource result\_identfier )

retourne un tableau associatif qui contient la ligne lue, ou bien FALSE, si il ne reste plus de lignes.

Si vous avez besoin d'indices numériques, préférez mysql\_fetch\_array()

Autrement dit, du fait qu'il n'y a pas d'indice numériques, si plusieurs colonnes portent le même nom, la dernière aura la priorité, les autres ne seront pas récupérées.

Exemple :

```

<?PHP
$resultat = mysql_query("select * from personne",$conn);
while ($row = mysql_fetch_assoc($resultat))
{
    echo "<br>".$row["num_personne"]." ";
    echo $row["nom"]." ";
    echo $row["prenom"]." <br>";
}
?>

```

Bien que ces trois fonctions soit valides, nous utiliserons autant que faire se peut la fonction mysql\_fetch\_array(), qui apporte un confort d'utilisation sans pour autant être moins performante que les autres.

#### ➤ mysql\_free\_result

int mysql\_free\_result ( resource result\_identfier )

mysql\_free\_result() efface le résultat de la mémoire. Cette fonction n'est à appeler que si vous avez peur d'utiliser trop de mémoire durant l'exécution de votre script. Toute la mémoire associée à l'identifiant de résultat sera automatiquement libérée, sans oublier, que toutes les traitements se terminent avec la fin du script.

Exercice : (Voir annexe Envoyer une requête vers la base de données, exercice 2 ).

Créer une page affichant la listes des stagiaires avec leur adresse dessous, en interrogeant la table stagiaire de la base formation, en utilisant les fonction :

mysql\_fetch\_array(), mysql\_fetch\_row() puis mysql\_fetch\_assoc().

Exercice : (Voir annexe Envoyer une requête vers la base de données, exercice 3 ).

Créer une page affichant la listes des stagiaires avec leur adresse dessous, en interrogeant la table stagiaire de la base formation, en utilisant mysql\_fetch\_array(), et présenter le resultat sous forme de tableau HTML (balise <table> etc...), comme le montre la figure ci-dessous (Figure 6).

**Liste des stagiaires**

N° de stagiaire	Nom et prénom	Adresse	Code Postal	Ville	N° de Téléphone
1	Madame DURANT	21, rue Georges Brassens	94000	Creteil	01 42 44 12 30
2	Mademoiselle Dupré	54, rue Jacques Brel	75000	PARIS	01 14 42 32 40
3	Père Alain	13, rue des Saint pères	93000	Saint-Denis	01 52 64 72 80
4	Monsieur Camus Albert	79, rue du scribe	92160	Antony	01 62 73 82 90
5	Madame Leblanc Jean-Marc	rue des blanc manteaux	91000	Montfermeil	01 02 14 22 40

Figure 6

Exercice : (Voir annexe Envoyer une requête vers la base de données, exercice 4 ).

Reprendre le formulaire de l'exercice 7 du paragraphe "Les Formulaires" et remplacer les contrôles checkbox des cours principaux à l'aide des informations recueillis dans la table cours de la base formation puis

alimenter la liste de sélection des cours optionnels à partir de cette même table.

Etapas de construction :

- ✓ Création du fichier " "   
C'est une page en HTML chargé uniquement d'afficher les champs de saisie sous forme de tableau HTML (voire figure 5).
- ✓ Création du fichier " "   
Chargé de vérifier si la méthode POST a été utilisée, si oui on vérifie la validité du formulaire et on affiche le résultat.
- ✓ Création du fichier " "   
Effectue les contrôles de la saisie du formulaire d'inscription en PHP.
- ✓ Création du fichier "chk\_stagiaires.inc"   
Création des contrôles checkbox à partir des données de la base.
- ✓ Création du fichier "lst\_stagiaires.inc"   
Alimentation de la liste de sélection des cours optionnels à partir des données de la base.
- ✓ Création du fichier " "   
Construit et met en forme la page de résultat selon les informations issues de la base de données et de la saisie de l'utilisateur.

## Rechercher des données dans la base

La commande SQL "SELECT" comporte trois clause qui vont nous servir pour rechercher et ordonner les données :

La clause "WHERE" , la clause "ORDER BY" et la clause "LIMIT".

### La clause "WHERE" :

Ajoute une condition qui s'exprime par comparaison entre le contenu d'un champ et une constante ou variable. La condition peut être associée avec un opérateur AND (et) ou OR (ou) et on peut inverser la condition avec l'opérateur NOT (pas). Pour indiquer la différence, on a le choix entre l'opérateur habituel "!=" et l'opérateur "<>".

Le mot clé LIKE recherche une correspondance floue d'une chaîne de caractères. Complété par l'un des deux caractère "\_" ou "%" il autorise différentes combinaisons :

Exemple :

```
<?php
$sql="SELECT * from livres where auteur = 'Leon';
//recherche les livres dont le nom de l'auteur égal à "Leon"
//et trouvera tous les livres écrits par l'auteur Leon
$sql="SELECT * from livres where auteur = '%Leon';
//recherche les livres dont le nom de l'auteur commence par "Leon"
//et trouvera tous les livres écrits par l'auteur "Dona Leon"
$sql="SELECT * from livres where auteur = '%on%';
//recherche les livres dont le nom de l'auteur contient "Leon"
//et trouvera tous les livres écrits par les auteurs "Dona Leon" et
//"Edward Marston"
$sql="SELECT * from livres where auteur = '_o%';
//recherche les livres dont le titre dont la deuxième lettre sera
//égal à "o"
//et trouvera tous les titres "Poésies" et "Noblesse oblige"
?>
```

### La clause "ORDER BY" :

spécifie le nom de champ sur lequel portera de tri du résultat. Les mots clés "ASC" et "DESC" précisent respectivement l'ordre croissant et décroissant. Par défaut le tri aura lieu dans l'ordre croissant.

Exemple :

```
<?php
$sql="SELECT * from livres ORDER BY auteur desc";
//Affiche tous les livres dans l'ordre décroissant.
?>
```

### La clause "LIMIT" :

Cette clause indique les limites exprimées en numéros de ligne des enregistrements à traiter. S'il ne figure qu'une seule valeur, cela signifie que la table sera traitée à partir de la valeur indiquée. S'il y a deux limites séparées par une virgule, la table sera traitée entre les valeurs indiquées, bornes comprises.

Exemple :

```
<?php
$sql="SELECT * from livres LIMIT 3,7";
?>
```

Nous avons vu au paragraphe "Envoyer une requête vers la base de données" comment interroger une base de données. Ayant vu comment une commande SQL se compose, nous allons manipuler les chaînes de caractères, issues des éléments saisies par l'utilisateur, pour construire les clauses "where", "order by" et "limit", et après les avoir ajoutées à la suite de notre requête SQL nous l'enverrons au serveur.

Exemple :

```
<?PHP
$sql="SELECT * from livres where auteur = 'Leon' ORDER BY auteur DESC LIMIT 3,7";
//Recherche les 5 premier livres dont le nom de l'auteur égal à
//"Leon" et les affiche dans l'ordre alphabétique décroissant des auteurs.
?>
```

Exercice : (Voir annexe Rechercher des données dans la base, exercice 1 ).

Reproduire le formulaire ci-dessous (Figure 7) et construire une requête SQL afin d'interroger la table des stagiaires, et afficher le résultat sur une autre page (Figure 8).

Étapes :



- ✓ Création du fichier "Recherche.php"

Chargé de vérifier si la méthode POST a été utilisée, si oui on vérifie la validité du formulaire et on appelle la page qui construit la requête SQL, si non on affiche les critères de recherche

- ✓ Création du fichier "Recherche.htm" (Voir Figure 7)

C'est une page en HTML chargée uniquement d'afficher les critères de recherche et d'indiquer le formulaire à appeler après avoir cliqué sur "lancer la recherche".

- ✓ Création du fichier "Construction\_Where.inc"

Cette page récupère les éléments saisis par l'utilisateur et construit la requête SQL, puis elle appelle la page de résultat.

- ✓ Création du fichier "Resultat.inc"

Interroge la base de données et construit les balises et données nécessaires à l'affichage du résultat.

- ✓ Création du fichier "Resultat.htm" (Figure 8)

Mettre en page et afficher le résultat sous forme de tableau HTML

Ainsi construit, on pourra modifier notre clause "where", sans avoir besoin de modifier notre page de résultat et vice versa.

### Formulaire de recherche

La recherche s'effectue sur 3 critères : nom, prénom et code postal.  
Le champ "nom" doit obligatoirement être rempli au moins partiellement, les autres champs sont facultatifs. Le résultat apparaît sous la forme d'une liste.

Nom :  Contient ☐

Prénom :  Contient ☐

Code Postal :

Figure 7

### Résultat de la recherche

N°	Nom	Prénom	Adresse	Téléphone	Mél
1	DURANT		21, rue Georges Brassens 94000 Creteil	01 42 44 12 30	durant@free.fr
2	Dupré		54, rue Jacques Brel 75000 PARIS	01 14 42 32 40	dupre@wanadoo.fr
3	Alain		13, rue des Saint p 93000 Saint-Denis	01 52 64 72 80	alain@htomail.com
4	Camus	Albert	79, rue du scribe 92160 Antony	01 62 73 82 90	camus@noos.fr
5	Leblanc	Jean-Marc	rue des blanc manteaux 91000 Montfermeil	01 02 14 22 40	jeanmarc.leblanc@free.fr

Figure 8

## Naviguer dans les données de la base

Pour naviguer dans les données de la base, nous nous servirons de la clause "LIMIT". Le premier paramètre de cette clause nous permet d'indiquer la ligne où débutera la recherche, et le second le nombre d'enregistrements à afficher sur chaque page. Ainsi la première page affichera les 5 premiers enregistrements et en rappelant la page, les 5 suivants ainsi de suite, jusqu'à la fin du résultat de la recherche.

Exercice : (Voir annexe Naviguer dans les données de la base, exercice ).

Reproduire les formulaires ci-dessous (Figure11) et construire une requête SQL afin d'interroger la table des stagiaires, afficher le résultat sur une autre page en limitant l'affichage à 2 enregistrements et permettre à l'utilisateur de naviguer dans le résultat. Dans la page de résultat l'utilisateur pourra en cliquant sur le nom du stagiaire ou en double cliquant sur la ligne consulter la fiche des cours suivis par le stagiaire (Fiche de renseignements).

Ce formulaire sera réalisé en utilisant uniquement la méthode POST

Étapes :

- ✓ Création du fichier "recherche\_post.php":

Chargé de vérifier si la méthode POST a été utilisée, si oui on vérifie la validité du formulaire et on appelle la page qui construit la requête SQL, si non on affiche les critères de recherche

- ✓ Création du fichier "rech\_validation.inc" :

Effectue les contrôles de la saisie du formulaire de recherche.

- ✓ Création du fichier "recherche\_post.htm" :

C'est une page en HTML chargée uniquement d'afficher les champs de saisie des critères de recherche sous forme de tableau HTML (voir figure 11).

- ✓ Création du fichier "resultat\_post.htm":

C'est une page en HTML chargée uniquement d'afficher le résultat de la requête sous forme de tableau HTML (voir figure 12).

- ✓ Création du fichier "Construction\_post\_Where.inc" :

Cette page récupère les éléments saisis par l'utilisateur et construit la requête SQL, puis elle appelle la page Construction\_post\_Limit.inc.

- ✓ Création du fichier "Construction\_post\_Limit.inc" :

Cette page construit la clause limit et appelle la page resultat\_post.inc.

- ✓ Création du fichier "resultat\_post.inc" :

Cette page interroge la base de données à l'aide des clauses "where" et "limit" créées précédemment, vérifie et formate les données dans un tableau HTML puis appelle la page resultat\_post\_html.

- ✓ Création du fichier "navigation\_post.inc" :

Cette page sera incluse par le fichier resultat\_post\_html et servira à afficher la barre de navigation.

- ✓ et enfin création du fichier "navigation.js"

ce fichier écrit en javascript nous aidera à naviguer dans la table, en incrémentant le n° de page.

En réalité nous n'avons pas besoin de créer autant de fichiers, nous pourrions bien entendu les regrouper par type, mais afin de bien comprendre, chaque étape a été décomposée en fichier.

Ainsi construit, on pourra modifier notre clause "where" et "limit", sans avoir besoin de modifier notre page de résultat et vice versa.

### Formulaire de recherche

La recherche s'effectue sur 3 critères : nom, prénom et code postal. Le champ "nom" doit obligatoirement être rempli au moins partiellement ; les autres champs sont facultatifs. Le résultat apparaît sous la forme d'une liste. La fiche des cours suivis par le stagiaire recherchée est accessible en cliquant sur son nom ou en double cliquant sur la ligne correspondante.

Nom :  Contient ☐

Prénom :  Contient ☐

Code Postal :

**Lancer la recherche**

Figure 11

## Résultat de la recherche

N°	Nom	Prénom	Adresse	Téléphone	Mél
1	DURANT		21, rue Georges Brassens 94000 Creteil	01 42 44 12 30	durant@free.fr
2	Dupré		54, rue Jacques Brel 75000 PARIS	01 14 42 32 40	dupre@wanadoo.fr
3	Alain		13, rue des Saint p 93000 Saint-Denis	01 52 64 72 80	alain@htomail.com
4	Camus	Albert	79, rue du scribe 92160 Antony	01 62 73 82 90	camus@noos.fr
5	Leblanc	Jean-Marc	rue des blanc manteaux 91000 Montfermeil	01 02 14 22 40	jeanmarc.leblanc@free.fr

//3


[Première page](#)
[Page Précédente](#)
[Page suivante](#)
[Dernière page](#)

Figure 12

## Fiche de renseignements

Fiche de stagiaire - Microsoft Internet Explorer fourni par le SGA

[Voir source de fiche.php](#)  
[Voir source de fiche.inc](#)  
[Voir source de fiche\\_popup.htm](#)


**Initiation PHP**  
 - fiche de renseignements -
 [Fermer la fenêtre](#)

[Imprimer](#)

**Identité du stagiaire :**  
 Nom : Madame DURANT  
 Prénom : F  
 Adresse : 21, rue georges brassens  
 94000 CRETEIL  
 Téléphone :  
 Mél : durant@free.fr

**Cours suivis :**

Cours :	Note TP1	Note TP2	Note TP3	Note TP4
Anglais :				
Physique :				
Mathématique :				

### Insérer des données dans la base

INSERT : est une instruction SQL qui permet d'ajouter des lignes de données (enregistrements) dans une table de base de données.

Exemple :

```
<?php
    $Sql="insert into livres (Num , auteur, date) value('1', 'Dona Leon', '2005')";
?>
```

Exercice : (Voir annexe Insérer des données dans la base de données ).

Reprendre le formulaire d'inscription (Figure 5) et lorsque l'utilisateur envoie la page, insérer les données dans la base

### addslashes() et stripslashes ()

Lors des opérations d'insertion de modification ou de sélection de données dans une base certains caractères doivent faire l'objet d'un traitement particulier afin d'être reconnu correctement.

La fonction addslashes(), après avoir échappé tous les caractères qui doivent l'être, retourne la chaîne de caractère pouvant être utilisé dans une requête de base de données.

Ces caractères sont les guillemets simples ('), guillemets doubles ("), anti-slash (\) et NUL (le caractère NULL).

stripslashes() retourne une chaîne dont les anti-slash ont été supprimé. Les doubles anti-slash sont réduits en un seul anti-slash.

stripslashes() est la fonction inverse de addslashes().

Exercice : (Voir annexe Insérer des données dans la base de données Exercice 2).

Reprendre le formulaire d'inscription de l'exercice précédent et, ajouter les fonctions addslashes() et stripslashes(), puis vérifier en insérant une adresse comportant une apostrophe .

### Modifier des données dans la base

UPDATE : est une instruction SQL qui permet de modifier une ou plusieurs colonnes dans un enregistrement.

Exemple :

```
<?php
    $Sql="update livres SET Num = 1 , auteur = 'Dona Leon'";
?>
```

Exercice 6 : (Voir annexe Modifier des données dans la base)

Reprendre le formulaire d'inscription et ajouter un bouton modifier qui permettra a l'utilisateur de modifier les données dans la base, comme le montre la figure ci-dessous

Etape de construction du formulaire :

- ✓ Création du fichier "db\_modification.html" en HTML  
C'est une page en HTML chargé uniquement d'afficher les champs de saisie sous forme de tableau HTML (voir figure 5).
- ✓ Création du fichier "validation1.inc"  
Effectue les contrôles de la saisie du formulaire d'inscription sur le serveur avec PHP.
- ✓ Création du fichier "db\_validation.js"  
Effectue les contrôles de la saisie du formulaire d'inscription sur le client avec JavaScript.
- ✓ Création du fichier "db\_modification.php"  
Chargé de vérifier si la méthode POST a été utilisée, si oui on vérifie la validité du formulaire et on appelle l'action choisie par l'utilisateur.
- ✓ Création du fichier "Connection.inc"  
Ouvre une connexion à la base de données.
- ✓ Création du fichier "charge\_stagiaire.inc"  
Interroge la base et récupère les données.
- ✓ Création du fichier "get\_msg.inc"  
Fonction de mise en forme des messages d'erreurs personnalisés.
- ✓ Création du fichier "verif\_mail.inc"  
Fonction de contrôle de validité de l'adresse email.
- ✓ Création du fichier "db\_update\_stagiaire.inc"  
Met à jour la base de données avec les éléments saisis par l'utilisateur.

# Fiche d'inscription

N° Stagiaire : 1

Madame : <input type="checkbox"/> Mademoiselle : <input type="checkbox"/> Monsieur : <input type="checkbox"/>	
Nom :	DURANT
Prénom :	
Adresse	21, rue Georges Brassens <input type="text"/>
Code Postal	94000      Ville      Creteil
Téléphone :	01 42 44 12 30
Courriel :	durant@free.fr
Choix des cours:	
Principaux : Mathématique : <input type="checkbox"/> Physique : <input type="checkbox"/> Français : <input type="checkbox"/>	Optionnels Anglais Espagnol Art plastique
<div>Ajouter</div> <div>Modifier</div> <div>Annuler</div>	

Figure 9

### Supprimer des données dans la base

DELETE : est une instruction SQL qui permet de supprimer d'une table un ou plusieurs enregistrements correspondant à une condition particulière.

```
<?php
    $Sql = "DELETE FROM livres WHERE num=1";
?>
```

**Attention :** Lors de la suppression d'enregistrements, de vérifier avec précaution que la condition de suppression soit bien prise en compte, sinon vous risquez d'effacer tous les enregistrements

Exercice : (Voir annexe Supprimer des données dans la base)

Reprendre le formulaire de l'exemple ci-dessous et permettez à l'utilisateur de supprimer les données dans la base

### Fiche d'inscription

N° Stagiaire : 1

Madame : <input type="radio"/> Mademoiselle : <input type="radio"/> Monsieur : <input type="radio"/>	
Nom :	DURANT
Prénom :	
Adresse	21, rue Georges Brassens
Code Postal	94000 Ville Creteil
Téléphone :	01 42 44 12 30
Courriel :	durant@free.fr
Choix des cours:	
Principaux : Mathématique : <input type="checkbox"/> Physique : <input type="checkbox"/> Français : <input type="checkbox"/>	Optionnels Anglais Espagnol Art plastique
Ajouter      Modifier      Supprimer      Annuler	

Figure 10

### Les dates et heures

PHP dispose de fonctions qui vous permettent de manipuler la date et l'heure du serveur qui exécute PHP. Vous pouvez utiliser ces fonctions pour formater la date et l'heure de nombreuses façons.

➤ La fonction mktime()

```
int mktime ( [int hour [, int minute [, int second [, int month [, int day [, int year [, int is_dst]]]]]] )
```

mktime() retourne un timestamp UNIX correspondant aux arguments fournis. Ce timestamp est un entier long, contenant le nombre de secondes entre le début de l'époque UNIX (1er Janvier 1970 00:00:00 GMT) et le temps spécifié.

Les arguments peuvent être omis, de droite à gauche, et tous les arguments manquants sont utilisés avec la valeur courante de l'heure et du jour.

La fonction `mktime()` est pratique pour faire des calculs de dates et des validations, car elle va automatiquement corriger les valeurs invalides.

Exemple :

```
<?php
    $jours = '29';
    $mois = '12';
    $annee = '2005';
    echo mktime(0,0,0,$mois,$jours,$annee);
    //Affiche 1135810800
?>
```

### ➤ La fonction `date()`

`string date(string format[,int timestamp]);`

La fonction `date()` prend deux paramètres. Le premier paramètre est une chaîne spécifiant un code de format, le second est optionnel et est un horodatage Unix. En l'absence du second paramètre, la fonction `date()` traite la date et l'heure courante. Elle renvoie une chaîne formatée représentant la date appropriée.

Exemple :

```
<?php
    //Date en anglais
    echo date("jS F Y");
    //affiche 22nd May 2005

    //Date en français
    echo date('d/m/Y');
    //affiche 22/05/2005

    //Date fournie par mktime en français
    $jours = '29';
    $mois = '12';
    $annee = '2005';
    echo date('d/m/Y', mktime(0,0,0,$mois,$jours,$annee));
    //Affiche 29/12/2005
?>
```

Le tableau suivant donne les codes de format reconnus par la fonction `date()`.

Code	Description
a	Matin ou après-midi, représenté sous la forme de deux caractères en minuscules : respectivement "am" et "pm".
A	Matin ou après-midi, représenté sous la forme de deux caractères en majuscules : respectivement "AM" et "PM".
B	Heure Internet Swtch, qui constitue un système de temps universel.
d	Jour du mois, sous la forme d'un nombre à deux chiffres, éventuellement préfixé par un zéro. La plage autorisée s'étend de "01" à "31".
D	Jour de la semaine en anglais, dans un format textuel abrégé en trois lettres : par exemple "Fri" pour Vendredi.
F	Mois de l'année en anglais, au format textuel, version longue. La plage autorisée va de "January" à "December".
g	Heure du jour, exprimée dans le système à 12 h, sans zéro initial. La plage autorisée s'étend de "1" à "12".
G	Heure du jour, exprimée dans le système à 24 h, sans zéro initial. La plage autorisée s'étend de "0" à "23".
h	Heure du jour, exprimée dans le système à 12 h, si nécessaire avec le préfixe zéro. La plage autorisée s'étend de "01" à "12".
H	Heure du jour, exprimée dans le système à 24 h, si nécessaire avec le préfixe zéro. La plage autorisée s'étend de "00" à "23".
i	Minutes, si nécessaire préfixée avec un zéro. La plage autorisée s'étend de "00" à "59".
I ('i' majuscule)	Spécifie si le système horaire courant est celui de l'heure d'hiver ou de l'heure d'été, sous la forme d'une valeur booléenne : "1" si l'heure d'hiver est activée, sinon "0".
j	Jour du mois au format numérique, sans zéro en préfixe. La page autorisée s'étend de "1" à "31".
l ('l' minuscule)	Jour de la semaine en anglais, au format textuel, version longue. La plage autorisée s'étend de "Monday" à "Sunday".
L	Indique si l'année est bissextile, par une valeur booléenne : "1" dans le cas d'une année bissextile, et "0" pour les années non bissextiles.
m	Mois de l'année, sous la forme d'un nombre à deux chiffres, éventuellement préfixé par un zéro. La plage autorisée s'étend de "01" à "12".
M	Mois de l'année en anglais, dans un format textuel abrégé en trois lettres. La plage autorisée s'étend de "Jan" à "Dec".

n	Mois de l'année, sous la forme d'un nombre sans zéro en préfixe. La plage autorisée s'étend de "1" à "12".
s	Secondes avec si nécessaire un zéro en préfixe. La plage autorisée s'étend de "00" à "59".
S	Suffixe ordinal pour les dates, en anglais, et en deux lettres. Ce suffixe peut être "st", "nd", "rd", ou "th" selon le nombre qui le précède.
t	Nombre total de jours dans le mois donné. La plage autorisée s'étend de "28" à "31".
T	Fuseau horaire du serveur, sous la forme de trois lettres, par exemple "MET".
U	Nombre total de secondes qui se sont écoulées depuis le 1er janvier 1970 jusqu'au moment considéré.
w	Jour de la semaine sous la forme d'un seul chiffre. La plage autorisée s'étend de "0" (dimanche) à "6" (samedi).
y	Année, sous un format à 2 chiffres, par exemple "00".
Y	Année, sous un format à 4 chiffres, par exemple "2001".
z	Jour de l'année sous forme d'un nombre. La plage autorisée s'étend de "0" à "365".
Z	Décalage horaire en secondes. La plage autorisée s'étend de "-43200" à "43200".

### ➤ La fonction getdate

array getdate ( [int timestamp] )

getdate() retourne un tableau associatif contenant les informations de date et d'heure du timestamp *timestamp* (lorsqu'il est fourni, sinon, le timestamp de la date/heure courante), avec les champs suivants :

Tableau 1. Nom des clés du tableau associatif retourné

Clé	Description	Exemple de valeur retournée
"seconds"	Représentation numérique des secondes	0 à 59
"minutes"	Représentation numérique des minutes	0 à 59
"hours"	Représentation numérique des heures	0 à 23
"mday"	Représentation numérique du jour du mois courant	1 à 31
"wday"	Représentation numérique du jour de la semaine courante	0 (pour Dimanche) à 6 (pour Samedi)
"mon"	Représentation numérique du mois	1 à 12
"year"	Année, sur 4 chiffres	Exemples: 1999 ou 2003
"yday"	Représentation numérique du jour de l'année	0 à 365
"weekday"	Version texte du jour de la semaine	Sunday à Saturday
"month"	Version texte du mois, comme January ou March	January à December
0	Nombre de secondes depuis l'époque Unix, similaire à la valeur retournée par la fonction <a href="#">time()</a> et utilisée par <a href="#">date()</a> .	Dépend du système, typiquement de -2147483648 à 2147483647.

Exemple :

```
<?php
    $jours = '29';
    $mois = '12';
    $annee='2005';

    $MaDate = getdate(mktime(0,0,0,$mois,$jours,$annee));
    foreach($MaDate as $key=>$value)
    {
        echo "$key : $value<br>";
    }
    echo $MaDate['mday']."/".$MaDate['mon']."/".$MaDate['year'] ;
    //Affiche 29/12/2005
?>
```

### ➤ La fonction checkdate()

int checkdate ( int month, int day, int year)

checkdate() retourne TRUE si la date représentée par le jour *day*, le mois *month* et l'année *year* est valide, et sinon FALSE. Notez bien que l'ordre des arguments n'est pas l'ordre français.



### Conversion entre des formats de date PHP et des formats de date MySQL

MySQL comme d'autres base de données dispose de plusieurs fonction SQL qui permettent de traiter les dates en voici quelques unes :

curdate() : renvoi la date sous la forme 2001-05-25

```
SELECT curdate() FROM tablename ;
```

curtime() : renvoi l'heure sous la forme 12:10:53

```
SELECT curtime() FROM tablename ;
```

now() : renvoi la date et l'heure sous la forme 2001-05-25 12:10:53

```
SELECT now() FROM tablename ;
```

MySQL exige que les dates soient entrées sous la forme année, mois, jour.

Exemple :

Le 29 août 2000 doit être saisi sous la forme 2000-08-29 ou 00-08-29

Des conversions de format sont donc nécessaires lorsque les dates doivent être échangées entre PHP et MySQL.

Lorsque des dates doivent être communiquées de PHP vers MySQL, la fonction date() permet facilement d'obtenir le format approprié. Il faut utiliser les formats de jour et de mois pour lesquels des zéros sont placés en préfixe, de manière à éviter les confusions au niveau de MySQL.

Dans MySQL, la commande SQL DATE\_FORMAT() permet de formater les dates. Elle est semblable à la fonction date() de PHP, mais elle utilise des codes de format différents. La conversion la plus courante consiste à exprimer une date sous le format MM – JJ – AAAA plutôt que sous le format AAAA – MM – JJ en vigueur dans MySQL. Cette conversion s'effectue au moyen de la requête suivante :

```
SELECT DATE_FORMAT (date_column, '%m %d %Y') FROM tablename ;
```

Le code de format %m correspond à une représentation du mois sous la forme d'un nombre à deux chiffres ; %d représente le jour sous la forme d'un nombre à deux chiffres ; et %Y représente l'années sous la forme d'un nombre à 4 chiffres.

Les codes de format MySQL les plus utiles pour ce type de conversion sont décrits dans le tableau suivant.

Code	Description
%M	Mois, plein texte
%W	Nom du jour de la semaine, plein texte
%D	Jour du mois, numérique, avec un préfixe textuel
%Y	Année, numérique, 4 chiffres
%y	Année, numérique, 2 chiffres
%a	Nom du jour de la semaine, 3 lettres
%d	Jour du mois, numérique, zéro en préfixe
%e	Jour du mois, numérique, pas de zéro en préfixe
%m	Mois, numérique, zéro en préfixe
%c	Mois, numérique, pas de zéro en préfixe
%b	Mois, texte, 3 lettres
%j	Jour de l'année, numérique
%H	Heure, système à 24 heures, zéro en préfixe
%k	Heure, système à 24 heures, pas de zéro en préfixe
%h ou %l ( 'i' majuscule)	Heure, système à 12 heures, zéro en préfixe
%l ( 'L' minuscule)	Heure, système à 12 heures, pas de zéro en préfixe
%i	Minutes, numérique, zéro en préfixe
%r	Heure, système à 12 heures (hh:mm:ss[AM PM])
%T	Heure, système à 24 heures (hh:mm:ss)

%S ou %s	Secondes, numérique, zéro en préfixe
%p	AM ou PM
%w	Jour de la semaine, numérique, de 0 (dimanche) à 6 (samedi)

Exercice : (Voir annexe Les dates)

Reprendre le formulaire d'inscription et ajouter un champ date d'inscription (voir figure 14)

Modifier de sorte que la date puisse être insérée ou modifiée.

### 6.3. Utiliser ODBC

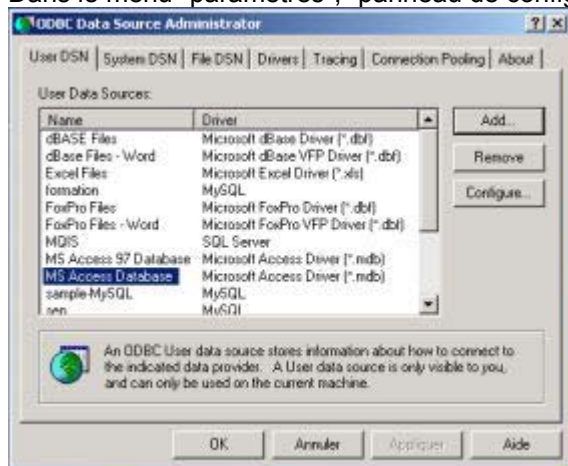
Odbc est une solution alternative permettant d'accéder à une base de données en écrivant un code générique, c'est-à-dire que le code de l'application fonctionne pour toutes les bases de données prenant en charge la norme ODBC. Toutefois, si cette solution semble séduisante, elle est bien moins fournie en fonctionnalités que l'accès en base au travers des fonctions de base dédiées c'est à dire directement présent en charge par PHP. Pour utiliser ODBC, il est nécessaire d'installer et de configurer le driver de la base.

#### Installation du driver odbc de MySql

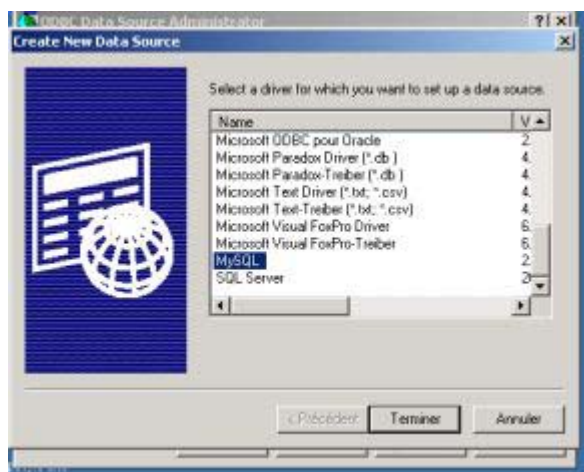
Télécharger le fichier MyODBC-3.51.11-2-win.exe dans un répertoire temporaire et lancer le programme (setup.exe)

#### Configuration de la source de donnée

Dans le menu "paramètres", "panneau de configuration" double cliquez sur "Sources de données (ODBC)".



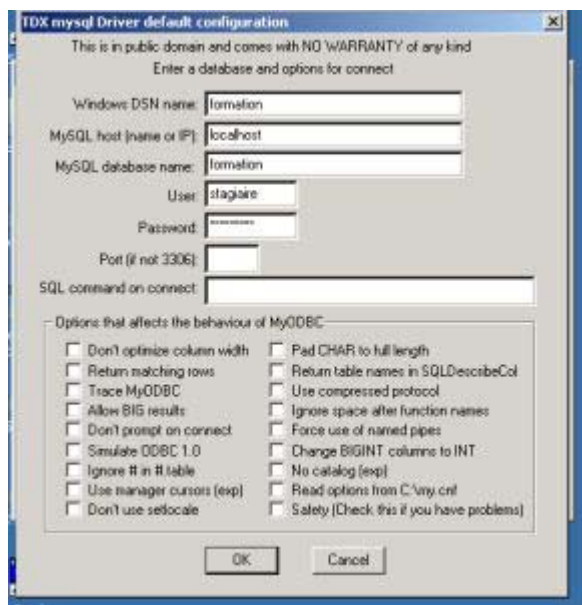
Cliquez sur add pour ajouter une nouvelle source de donnée, sélectionnez Mysql puis Terminer.



Editez les paramètres suivant :

Windows DSN name :	formation
Mysql host name :	localhost
User :	stagiaire
Password :	stagiaire
Port :	3306

Puis validez en cliquant sur OK



Maintenant nous pouvons établir une connexion à la base en utilisant la fonction `odbc_connect()` :

`resource odbc_connect ( string dsn, string user, string password [, int cursor_type])`

Exemple :

```
<?php
$dsn= "formation";
$host="localhost"; // Nom du serveur ou adresse IP qui héberge la base de données
$user="stagiaire"; //Nom de l'utilisateur de la base de données
$password="stagiaire"; // Mot de passe de connexion à la base de données
$db="formation"; //Nom de la base de données que l'on interroge'
$conn=odbc_connect ($dsn, $user, $password) or die ("Impossible de se connecter");
?>
```

Pour interroger la base nous utiliserons :

`odbc_exec` pour exécuter une requête à la place de `Mysql_query`

`int odbc_exec ( resource connection_id, string query_string)`

et `odbc_fetch_array` pour lire les enregistrements à la place de `mysql_fetch_array`

`array odbc_fetch_array ( resource result [, int rownumber])`

Voici un exemple de connexion et d'interrogation de la base formation

```
<?PHP
//Code servant a afficher le source.
echo "<link rel='stylesheet' type='text/css' href='../css/sga.css'>";
echo "<A href='show_source.php?script=db_satgiaireOdbc.php'>Voir source</A><br><br>";
include("fonction.inc");

//Code de l'exercice.
include("connect_odbc.inc");
//Si la table se nomme stagiaires
$requete= "select * from stagiaires limit 0,5";
echo "<B>Résultat avec odbc_fetch_array()</B><BR>";
//On exécute la requête
$resultat=@odbc_exec($conn,$requete);
//On récupère les valeurs grâce à la boucle while
//dans la variable tableau $annuaire stagiaires créée par mysql fetch array()
//On dispose des indices numérique ou des noms de champs comme indice
echo "<h2>Liste des stagiaires</h2>";
echo "<table border=1><tr align =center><th>N° de stagiaire</th>
<th>Nom et prénom</th><th>Adresse</th>
<th>Code Postal</th><th>Ville</th><th>Courriel</th><th>N° de Téléphone</th></tr>";
while ($annuaire_stagiaires = odbc_fetch_array ($resultat))
{
    echo "<tr><td align center>".$annuaire_stagiaires["num_stagiaire"]."</td> ";
    echo "<td>".civillite($annuaire_stagiaires["civ"])."
    ".$annuaire_stagiaires["nom"]." ".$annuaire_stagiaires["prenom"]."</td>";
    echo "<td>".$annuaire_stagiaires["adresse"]."&nbsp;&nbsp;&nbsp;</td>";
    echo "<td align=center>".$annuaire_stagiaires["cp"]."&nbsp;&nbsp;&nbsp;</td>";
    echo "<td>".$annuaire_stagiaires["ville"]."&nbsp;&nbsp;&nbsp;</td>";
    echo "<td>".$annuaire_stagiaires["courriel"]."&nbsp;&nbsp;&nbsp;</td>";
}
```

```

        echo "<td>".$annuaire_stagiaires["telephone"]."&nbsp;</td></tr>";
    }
    echo "</table>";

    //on libère les ressources.
    odbc_free_result ($resultat);
    //on ferme la connection à la base
    odbc_close ($conn);
?>

```

Exercice : (Voir annexe ODBC Exercice 1)

Afficher la liste des stagiaires en utilisant une connection ODBC ayant une base mysql comme source de données, avec utilisateur=stagiaire , mot de passe =stagiaire et base = formation afin d'obtenir le résultat montré dans l'exemple ci-dessous.

#### Liste des stagiaires

N° de stagiaire	Nom et prénom	Adresse	Code Postal	Ville	Courriel	N° de Téléphone
1	Madame DURANT	21, rue Georges Brassens	94000	Creteil	durant@free.fr	01 42 44 12 30
2	Mademoiselle Dupré	54, rue Jacques Brel	75000	PARIS	dupre@wanadoo.fr	01 14 42 32 40
3	Père Alain	13, rue des Saint p	93000	Saint-Denis	alain@htomail.com	01 52 64 72 80
4	Monsieur Camus Albert	79, rue du scribe	92160	Antony	camus@noos.fr	01 62 73 82 90
5	Leblanc Jean-Marc	rue des blanc manteaux	91000	Montfermeil	jeanmarc.leblanc@free.fr	01 02 14 22 40

Exercice : (Voir annexe ODBC Exercice 2)

Créer une source de données ODBC permettant d'accéder à la base access "formation.mdb". Reprendre l'exercice précédent et adaptez le afin de pouvoir obtenir le même résultat, mais en interrogeant la base access "formation.mdb" qui se trouve dans le répertoire exercice (Vous pouvez aussi télécharger).

Comme nous l'avons vu, il est possible d'interroger différentes bases de données soit par les fonctions natives soit par ODBC. Mais lorsque l'on souhaite accéder à différentes source de données dans un même programme, cela devient vite fastidieux de changer de connection et fonctions.

Pour éviter ces manipulations, il existe un moyen qui passe par la création d'une classe regroupant tous les modes de connection que l'on a besoin. Cette classe apporte une couche d'abstraction de base de données. Ils en existe plusieurs (PEAR fournit avec PHP, sql\_layer créer par PHP\_Nuke etc...). Nous reviendrons sur ce sujet au chapitre des classe.

## 7. Les Sessions et Cookies

### 7.1. Les Cookies

#### Qu'est-ce qu'un cookie ?

Un cookie est un fichier texte sauvegardé sur la machine côté client dans lequel sont stockées des données. Il permet à un site web de stocker et de récupérer des données dans le cas d'applications ayant besoin de maintenir un état (persistance des données) entre plusieurs visites de l'utilisateur.

Les cookies permettent d'éviter de forcer un utilisateur à se reconnecter s'il émet une nouvelle requête, et favorisent une navigation plus personnalisée.

Le fichier contient une chaîne de texte de type nom = valeur. Une URL lui est associée : elle permet au client de savoir s'il doit renvoyer le cookie au serveur contacté.

#### Cookies en PHP

La fonction `setcookie()` permet de définir manuellement les cookies, de la manière suivante :

```
int setcookie( string nom_cookie, string valeur_cookie, int date_exp);
```

Si la date d'expiration n'est pas définie, la durée de vie du cookie est celle de la session.

Exemple :

```
<?PHP
    setcookie("cookie", "chocolat", time() + 90 * 86400);
?>
```

après exécution de cette ligne, un cookie de nom `moncookie` et de valeur "chocolat" est stocké.

Il est possible de définir des cookies à valeurs multiples : en effet, les cookies sont restreints à 20 par serveur. On traite alors le cookie comme un tableau, et on affecte une valeur à chaque élément du tableau.

Exemple :

```
<?PHP
    if (!isset($cookie[0]))
    {
        setcookie("cookie[0]", $nomvisiteur);
        cookie[1]++; setcookie("cookie[1]", $cookie[1]);
        echo "Bonjour $cookie[0], vous avez effectué ".$cookie[1].
            ($cookie[1] == 1 ? " visite !" : " visites !")."sur cette page.";
    }

    //affiche : Bonjour (le nom du visiteur), vous avez effectué (le nombre de visites) sur cette page.
?>
```

Le code qui définit le cookie doit apparaître au tout début de la page, dans l'en-tête, avant la génération d'une sortie et son envoi au navigateur. PHP envoie un message d'erreur si `setcookie()` est appelée après qu'une réponse HTTP a été envoyée au client.

**Attention :** un simple espace, un caractère de saut de ligne ou tout autre texte, HTML ou généré par PHP, suffit à provoquer cette erreur.

Dans les mêmes conditions d'utilisation, la fonction `setcookie()` avec une nouvelle valeur pour le cookie permet de changer la valeur du cookie.

#### Accéder à un cookie

Deux méthodes permettent d'accéder à un cookie :

- une variable globale est associée au cookie : `$nom_cookie` qui contient la valeur\_cookie ;
- la variable d'environnement `$_COOKIE [nom_cookie]` est un tableau associatif global qui ne contient que des variables provenant de cookies, ce qui permet d'avoir une information fiable sur l'origine.

Exemple :

```
<?php
    if (!isset($_COOKIE['cookie'])) { setcookie("cookie", "chocolat");
        echo "le cookie \$cookie = chocolat a été créé : <br>";
        echo "vous allez pouvoir y avoir accès";
        echo "<br><a href = \"cookie1.php\">accès</a>";
    }
    else {
        echo "\$cookie = ".$_COOKIE['cookie'].", <br>";
        echo "vous pouvez maintenant utiliser le cookie";
        echo "<br><a href = \"cookie2.php\">utilisation</a>";
    }
?>
```

#### Supprimer un cookie

Un cookie peut être supprimé à l'aide de la fonction `setcookie()` dotée d'un unique paramètre :

```
int setcookie(string nom_cookie);
```

Exemple :

```
<?php
    setcookie ("cookie","");
    if (!isset($_COOKIE['cookie'])) {
        echo "\$cookie a été détruit";
        echo "<br><a href = \"cookie.php\">recommencer</a>";
    } else {

        echo "\$cookie va être détruit";
        echo "<br><a href = \"cookie4.php\">destruction</a>";
    }
?

```

Exemples d'utilisation des cookies

Script de cookie.php

```
<?php
    if (!isset($_COOKIE['cookie'])) { setcookie("cookie", "chocolat");
        echo "le cookie \$cookie = chocolat a été créé : <br>" ;
        echo "vous allez pouvoir y avoir accès";
        echo "<br><a href = \"cookie1.php\">accès</a>";
    }
    else {
        echo "\$cookie = ".$_COOKIE['cookie'].", <br>";
        echo "vous pouvez maintenant utiliser le cookie";
        echo "<br><a href = \"cookie2.php\">utilisation</a>";
    }
?>

```

Script de cookie1.php

```
<?php
    $cookie=$_COOKIE['cookie'];
    echo "\$cookie est toujours $cookie<br>";
    echo "cette valeur va changer<br>";
    echo "<a href = \"cookie3.php\">suite</a>";
?>

```

Script de cookie2.php

```
<?php
    echo "\$cookie est toujours ".$_COOKIE['cookie']. "<br>";
    echo "cette valeur va changer<br>";
    echo "<a href = \"cookie3.php\">suite</a>";
?>

```

Script de cookie3.php

```
<?php
    setcookie ("cookie","café");
    if ($_COOKIE['cookie'] == "chocolat") {
        echo "le changement n'a pas eu encore lieu";
        echo "<br>la valeur ".$_COOKIE['cookie']. " de \$cookie va changer";
        echo "<br><a href = \"cookie3.php\">changement</a>";
    }
    elseif ($_COOKIE['cookie'] == "café") {
        echo "le changement a eu lieu";
        echo "<br>la nouvelle valeur de \$cookie est ".$_COOKIE['cookie']. " ";
        echo "nous allons maintenant détruire le cookie";
        echo "<br><a href = \"cookie4.php\">destruction</a>";
    }
?>

```

Script de cookie4.php

```
<?php
    setcookie ("cookie","");
    if (!isset($_COOKIE['cookie'])) {
        echo "\$cookie a été détruit";
        echo "<br><a href = \"cookie.php\">recommencer</a>";
    } else {
        echo "\$cookie va être détruit";
        echo "<br><a href = \"cookie4.php\">destruction</a>";
    }
?>

```

Exercice : (Voir annexe, Les cookies)

Créer 4 pages pour manipuler les cookies.

Pour chaque page, permettez à l'utilisateur, à l'aide d'un lien hypertexte, de passer à la page suivante.

Indiquez sur chaque page l'état et la valeur du cookie, comme indiqué ci-dessous.

*la première :*

Créer le fichier cookie1.php qui vérifiera l'existence de la variable cookie (\$cookie) ayant la valeur "Je mange du poisson".

Si elle n'existe pas, elle sera créée et l'utilisateur informé. Un lien lui permettra de rappeler la page. Si elle existe un message informera l'utilisateur et un lien lui permettra de passer à la page suivante.

Exemple :

le cookie \$cookie = "Je mange du poisson" a été créé :

vous allez pouvoir y avoir accès

[accès](#)

ou

\$cookie = Je mange du poisson,

vous pouvez maintenant utiliser le cookie

[utilisation](#)

*la seconde :*

Créer le fichier cookie2.php qui informera l'utilisateur de la valeur du cookie précédemment créé et lui indiquer qu'en cliquant sur un lien il changera sa valeur.

Exemple :

"\$cookie = Je mange du poisson",

vous pouvez maintenant utiliser le cookie

[utilisation](#)

ou

\$cookie est toujours "Je mange du poisson"

cette valeur va changer

[suite](#)

*la troisième :*

Créer le fichier cookie3.php qui déterminera si la valeur du cookie a changé, si oui l'utilisateur sera averti qu'en cliquant sur le lien sa valeur va changée,

si non, l'avertir que le changement a eu lieu et qu'en cliquant sur le lien le cookie sera détruit.

Exemple :

le changement n'a pas eu encore lieu

la valeur "Je mange du poisson" de \$cookie va changer

[changement](#)

ou

le changement a eu lieu

la nouvelle valeur de \$cookie est "Je préfère la viande",

nous allons maintenant détruire le cookie

[destruction](#)

*la quatrième :*

Créer le fichier cookie4.php qui déterminera si le cookie a été détruit, si oui on informe l'utilisateur et on insère un lien permettant de recommencer,

si non, on l'informe qu'en cliquant sur le lien le cookie sera détruit.

Exemple :

\$cookie va être détruit

[destruction](#)

ou

\$cookie a été détruit

[recommencer](#)



## 7.2. Les sessions

Une session débute lors du lancement du navigateur, lorsqu'un utilisateur arrive sur la page web et se termine à son départ.

HTTP est un protocole sans état : il ne comprend pas de mécanisme de maintien des états entre deux transactions. Autrement dit lorsqu'une page web se ferme et que les scripts terminent tous les éléments en mémoire sont détruits donc perdus.

On a recours à la session qui permet de gérer un contexte qui est l'ensemble de plusieurs variables dont le contenu doit persister sur plusieurs pages de l'application. Cette opération peut s'effectuer de plusieurs manières :

- à l'aide d'un cookie ;
- par le passage de paramètres à chaque lien (URL longues) ;
- par le passage de variables postées via des champs cachés ;
- par l'utilisation des sessions PHP.

En PHP chaque session est caractérisée par un numéro d'identification unique : ID ou identifiant de session. C'est un nombre aléatoire, stocké soit sur la machine du client dans un cookie soit passé via des URL. Cet ID de session joue le rôle d'une clé qui permet d'enregistrer des variables particulières en tant que "variables de session" et est la seule information relative à la session qui soit visible côté client. Les variables sessions sont enregistrées dans un fichier texte de type nom = valeur soit sur le client via les cookies soit sur le serveur si les cookies ne sont pas disponibles.

### Les sessions avec cookies

Lors de la mise en œuvre de sessions PHP, les fonctions de session se chargent automatiquement dans la définition des cookies, à condition que la directive `session.use_cookies` soit égale à 1.

La fonction `session_get_cookie_params()` permet de visualiser le contenu d'un cookie défini par une session. Elle renvoie un tableau associatif contenant les éléments date d'expiration, chemin, domaine.

La fonction `session_set_cookie_params()` permet de définir les paramètres d'un cookie de session :

```
void session_set_cookie_params( int date_exp, string chemin [,string domaine]);
```

L'usage de cookies peut poser des problèmes. Certains navigateurs ne les acceptent pas, et certains utilisateurs désactivent volontairement dans leur navigateur Web la fonctionnalité relative aux cookies.

Les sessions PHP utilisent par défaut les cookies : si possible, un cookie est créé pour stocker l'ID de la session courante.

Dans le cadre d'une application de gestion, ou l'application repose sur l'utilisation des sessions il est préférable d'éviter d'utiliser les sessions avec cookies.

### Passage des paramètres par l'URL

Avec cette méthode l'ID session est transmis via l'URL.

Pour que celle-ci soit automatiquement adoptée PHP doit être compilé avec l'option `--enable-trans-sid` option.

Vous pouvez aussi passer manuellement l'ID d'une session par le biais d'un hyper lien. Si PHP a été configuré avec l'option `--enable-track-vars`, l'ID de session est stocké dans la constante `SID` : vous devez l'ajouter à la fin du lien, à la manière d'un paramètre GET.

Exemple :

```
<A HREF="page_2.php?<?=SID?>">lien</A>
```

Si PHP n'a pas été configuré avec cette option, il faut placer la clé de session dans l'URL de la page de destination, ce qui constitue une écriture plus longue.

Exemple :

```
<A HREF="page_2.php?session_name()=session_id()">lien</A>
```

La fonction `session_name()` renvoie le nom de la session tel que défini dans le `php.ini`.

```
string session_name ( [string name])
```

La fonction `session_id()` renvoie le N° de session défini lors du début de la session déclarée par `session_start()`

```
string session_id ( [string id])
```

### **Les sessions utilisant les fonctions PHP**

PHP comprend des fonctions natives de contrôle de session, en plus de celles contenues dans la bibliothèque de base de PHP : PHPlib.

#### **Configuration de php.ini**

Le fichier de configuration php.ini comporte une section dédiée à la configuration de la gestion de session. On y trouve en particulier les paramètres suivants :

##### **session.save\_handler :**

Avec PHP4, la seule valeur possible est file, ce qui signifie que les valeurs des variables de session sont stockées dans un fichier sur le disque. Le contenu de ces fichiers est en ASCII, on peut y reconnaître le nom des variables, ainsi que leurs types et valeurs.

Avec PHP5 la valeur user permet un stockage en mémoire, ou dans un SGBD.

##### **session.save\_path :**

Ce paramètre indique le chemin des fichiers contenant les valeurs des variables de session. Par défaut, ils sont créés dans /tmp.

##### **session.name :**

Ce paramètre spécifie le nom de la session, nom également utilisé en tant que nom du cookie de session. Il ne doit contenir que des caractères alpha numériques. Sa valeur par défaut est PHPSESSID.

##### **session.cookie\_lifetime :**

Ce paramètre indique la durée maximale de vie du cookie envoyé au navigateur client. Si le paramètre est zéro, le cookie existera jusqu'à ce que le navigateur soit arrêté. C'est la valeur par défaut.

##### **session.auto\_start :**

Ce paramètre spécifie si les sessions sont créées à la demande ou systématiquement. Par défaut, à zéro, les sessions doivent être créées explicitement par session\_start() ou session\_register().

## Ouverture d'une session

La fonction `session_start()` initialise les données d'une nouvelle session ou rafraîchit les données de la session en cours.

```
boolean session_start();
```

Lors de l'accès au site, si l'identificateur de la session ne peut être trouvé, l'appel à cette fonction permet de créer une session et de transmettre son identifiant au navigateur.

Cette fonction doit être placée en tout début de script et doit donc être placée avant l'envoi des headers http.

## Enregistrement d'une variable de session

Une variable de session est une variable régulière globale, qui quand elle est enregistrée, garde sa valeur sur toutes les pages qui utilisent la session.

Pour enregistrer une variable de session, il faut assigner une valeur à une variable qui va devenir la variable de session.

Deux méthodes sont possibles pour enregistrer une variable de session selon que la directive `register_global` du fichier de configuration de php (`php.ini`) est active ou non :

Avec `register_global=on`

On utilise la fonction `session_register()` :

```
boolean session_register(mixed nom_variable,...);
```

Vous pouvez enregistrer plusieurs variables de session lors du même appel de la fonction.

Avec `register_global=off`

```
$_SESSION['MavARIABLE']='valeur' ou $HTTP_SESSION_VARS['MavARIABLE']='valeur'
```

Sur toutes les prochaines pages utilisant les sessions (en appelant la fonction `session_start()`), la variable `$_SESSION['nom_variable']` aura la valeur qui lui a été assignée quand elle a été enregistrée comme une variable de session.

Des changements de la valeur de la variable seront automatiquement enregistrés dans la session et sauvegardés pour les prochaines utilisations.

## Utilisation de variables de session

Vous pouvez accéder à la variable :

Avec `register_global=on`

via son nom : `$nom_variable`

Avec `register_global=off`

par le biais du tableau associatif `$_SESSION` ou `$HTTP_SESSION_VARS` :

```
$_SESSION[$nom_variable]
```

Déterminer si une variable est une variable de session enregistrée :

Avec `register_global=on`

La fonction `session_is_registered()` permet de déterminer si une variable est une variable de session enregistrée :

```
boolean session_is_registered(string nom_variable);
```

Elle renvoie la valeur `true` si la variable est enregistrée comme variable de session et `false` dans le cas contraire.

Avec `register_global=off`

Vous pouvez utiliser le tableau `$_SESSION` ou `$HTTP_SESSION_VARS` pour vérifier si la variable est enregistrée comme variable de session :

```
if(isset($_SESSION['MavARIABLE'])) ;
```

## Dés-enregistrement de variables et suppression de session

Dés-enregistrement d'une variable de session :

Vous pouvez dés-enregistrer une variable de session en invoquant la fonction

Avec `register_global = on`

`session_unregister()` :

```
boolean session_unregister(string nom_variable);
```

Cette fonction dés-enregistre une variable de session à la fois.

Avec `register_global = off`

`unset()`:

```
void unset($_SESSION['MavARIABLE']);
```

Dés-enregistrement de toutes les variables de session :

Avec `register_global = on`

La fonction `session_unset()` permet de dés-enregistrer toutes les variables de sessions courantes.

```
void session_unset();
```

Avec `register_global = off`

```
$_SESSION = array();
```

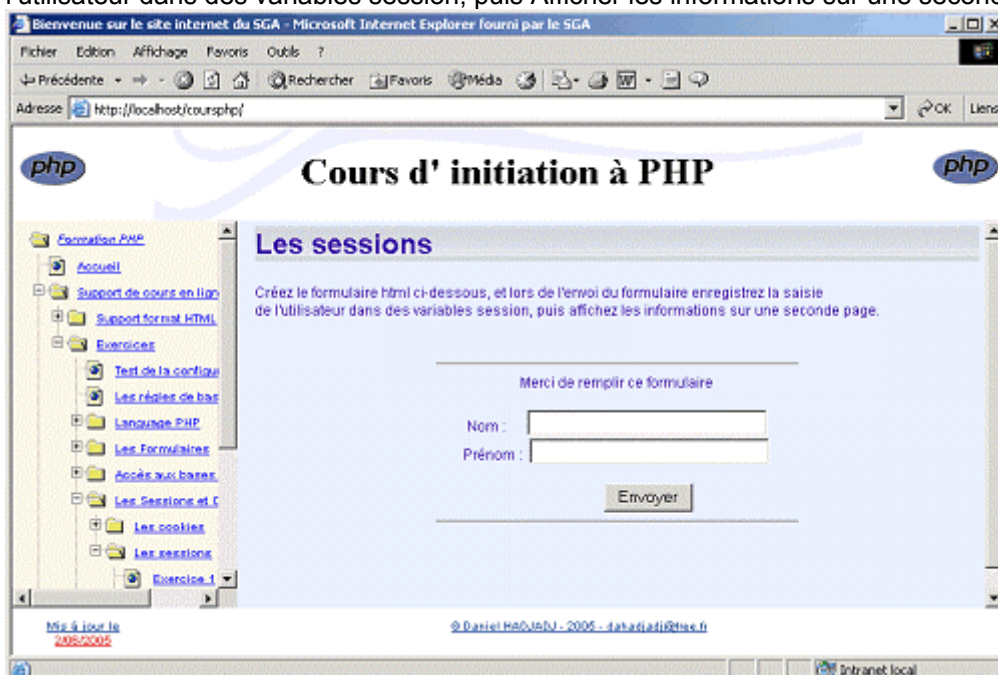
Suppression de session :

Lorsque vous avez terminé une session, après avoir supprimer toutes les variables de session, vous pouvez invoquer la fonction `session_destroy()` qui permet de supprimer l'ID de la session.

```
boolean session_destroy();
```

Exercice 1 : (Voir annexe, Les sessions)

Créer le formulaire html ci-dessous, et lors de l'envoi du formulaire enregistrez la saisie de l'utilisateur dans des variables session, puis Afficher les informations sur une seconde page.



Exercice 2: (Voir annexe, Les sessions)

Créer le formulaire de connection (figure 13), et lors de l'envoi du formulaire enregistrez la saisie de l'utilisateur dans des variables session, puis Afficher les informations sur une seconde page. Cette seconde page contiendra un lien permettant a l'utilisateur de se déconnecter (supprimer les variables session), pour revenir à la page de connection.

Connection

Votre nom :

Mot de passe :

Figure 13

Exercice 3 : (Voir annexe, Les sessions)

Reprendre le formulaire de connection (figure 13), et lors de l'envoi du formulaire, avant d'enregistrer la saisie de l'utilisateur dans des variables session, Créer une fonction qui vérifiera en comparant

avec la base de donnée si l'utilisateur existe et si son mot de passe est correct, puis Afficher les informations sur une seconde page. Cette seconde page contiendra un lien permettant a l'utilisateur de se déconnecter (supprimer les variables session), pour revenir à la page de connexion.

## 8. Gestion des fichiers

PHP permet de manipuler n'importe quel fichier du serveur.

Avant d'effectuer une opération sur un fichier, il est intéressant de s'assurer que ce fichier existe. La fonction `file_exists()` nous permet de le faire :

```
int file_exists(string nom_fichier);
```

Les fonctions qui permettent la gestion de fichiers sont des fonctions qui renvoient `true` quand l'opération réussit, et `false` quand elle échoue, sauf précision contraire.

### Ouvrir des fichiers

La fonction `fopen()` permet d'ouvrir un fichier sur le serveur local ou via HTTP ou FTP sur Internet.

On l'utilise de la manière suivante :

```
int fopen(string nom_fichier,string mode);
```

L'argument `nom_fichier` désigne le nom du fichier à ouvrir, l'argument `mode` correspond au mode d'ouverture du fichier, qui accepte l'une des valeurs suivantes :

Valeur	Description
a	ouverture d'un fichier en ajout uniquement. Les données sont écrites à la fin du fichier. Si le fichier n'existe pas, PHP tente de le créer.
a+	ouverture d'un fichier en ajout et en lecture. Les données sont écrites à la fin du fichier. Si le fichier n'existe pas, PHP tente de le créer.
r	ouverture d'un fichier pour lecture seule.
r+	ouverture d'un fichier en lecture-écriture. Les données sont écrites au début du fichier.
w	ouverture d'un fichier en écriture seule. Les données remplacent le contenu du fichier qui est alors perdu. Si le fichier n'existe pas, PHP tente de le créer.
w+	Ouverture d'un fichier en écriture-lecture. Les données remplacent le contenu du fichier qui est alors perdu. Si le fichier n'existe pas, PHP tente de le créer.

De plus, l'indicateur `b` indique que le contenu du fichier est binaire et non du texte (une image par exemple).

Si la fonction `fopen()` réussit, elle renvoie un pointeur de fichier (entier qui fait référence au fichier dans les appels de fonction ultérieurs), sinon, elle renvoie `false`.

Exemple:

```
<?
    $var = file_exists("fichier.php");
    if(!$var){
        echo "echec: le fichier demandé n'existe pas<br>";
    }
    else{
        echo "le fichier demandé existe<br>";
        if(!$mavar = fopen("fichier.php","r"))
        {
            echo "echec : le fichier n'a pas été ouvert<br>";
        }
        else{
            echo "le fichier a été ouvert<br>";
            echo "voici son contenu<br><br><br>";
            fpassthru ($mavar);
        }
    }
    //affiche le formulaire de la page fichier.php et ferme le fichier
?>
```

### **Fermer des fichiers**

Après la lecture d'un fichier, vous devez le refermer.

La fonction utilisée est :

```
mixed fclose(int pointeur_fichier);
```

### **Naviguer dans des fichiers**

PHP possède des fonctions qui permettent de définir avec précision la longueur de la chaîne à lire dans le fichier, ainsi que de se déplacer à une position spécifique du fichier.

La fonction rewind() :

```
int rewind(int pointeur_fichier);
```

Elle place l'indicateur de position au début du fichier.

La fonction fseek() :

```
int fseek( int pointeur_fichier, int decalage);
```

L'argument décalage est le nombre d'octets ou de caractères par rapport au début du fichier.

Attention : cette fonction retourne -1 en cas d'erreur et 0 en cas de réussite.

La fonction ftell() :

```
int ftell(int pointeur_fichier);
```

Cette fonction renseigne sur la position de l'indicateur dans un fichier.

La fonction feof() :

```
int feof(int pointeur_fichier);
```

### **Afficher des fichiers**

L'opération consiste à envoyer le contenu du fichier vers un flux de sortie.

La fonction utilisée est :

```
int fpassthru(int pointeur_fichier);
```

Elle effectue la lecture de la position courante à la fin du fichier, puis elle referme le fichier. Le paramètre de cette fonction est le pointeur du fichier renvoyé par la fonction fopen().

### **Lire des fichiers**

L'affichage en totalité d'un fichier n'est pas toujours souhaitable. Les fonctions PHP de lecture de fichier permettent de ne lire que quelques données.

La fonction fread() :

```
string fread( int pointeur_fichier, int longueur);
```

Le paramètre longueur indique le nombre de caractères depuis le début. La lecture se termine si la fonction rencontre la fin du fichier.

La fonction fgets() :

```
string fgets(int pointeur_fichier, int longueur);
```

Le paramètre longueur indique le nombre de caractères plus un caractère depuis le début. La lecture se termine si la fonction rencontre un caractère de saut de ligne ou la fin du fichier.

La fonction fgetss() :

```
string fgetss( int pointeur_fichier, int longueur);
```

Cette fonction a le même effet que la fonction fgets(), mais exclut les balises HTML et PHP lors de l'affichage.

La fonction fgetc() :

```
string fgetc(int pointeur_fichier);
```

Cette fonction permet de lire le premier caractère situé à la position courante.

La fonction file() :

```
array file(string nom_fichier);
```

Cette fonction lit le contenu d'un fichier qu'elle place dans un tableau, chaque ligne du fichier étant représentée par un élément du tableau (la première ligne correspond à l'élément zéro).

La fonction filesize() :

```
int filesize ( string filename)
```

Renvoie la taille du fichier.

Exemple :

```
<?php
if(!$fichier = fopen("texte.txt","r"))
{
    echo "ouverture du fichier impossible";
}else
{
    $text1 = fread($fichier,17);
    echo "$text1<br>";
    //affiche : Ce fichier est (les 14 premiers caractères + <p>)
    $text2 = fgets($fichier,50);
    echo "$text2<br>";
    // affiche : destiné à apprendre à insérer des fichiers (s'arrête au saut de ligne)
    rewind($fichier);
    $text4 = fgets ($fichier,18);
    echo "$text4<br>";
    // affiche : Ce fichier est
    echo "<br>";
    rewind($fichier);
    // replace le pointeur au début du fichier
    while(!feof($fichier))
    {
        echo(fgetc($fichier));
        // affiche chaque caractère du fichier jusqu'à la fin du fichier
    }
    echo "<br>";
    rewind($fichier);
    $arrText = file("texte.txt");
    // place le contenu du fichier dans un tableau
    for ($i = 0 ; $i < count($arrText) ; $i++)
    {
        // chaque ligne du texte est insérée dans les balises <b> ... </b>
        // pour mettre le texte en gras
        echo "<b>$arrText[$i]</b><br>";
    }

    //renvoi la taille du fichier
    echo filesize("texte.txt")." Octets<br>";
    fclose($fichier);
}
?>
```

Exercice 1 : (Voir annexe, Gestion des fichiers)

Soit le fichier "stagiaires\_mdb.txt" contient la liste des stagiaires. Ouvrir ce fichier en lecture seule et Afficher son contenu.

### Ecrire dans des fichiers

Pour écrire une chaîne dans un fichier, on peut utiliser l'une des deux fonctions fputs() ou fwrite() elles sont identiques en tout point :

```
int fputs(int fichier_pointeur,string str,[int longueur]);
ou
int fwrite(int fichier_pointeur,string str,[int longueur]);
```

Si le dernier paramètre n'apparaît pas, la chaîne est écrite en totalité.

Exemple :

```
<?php
if($fichier = fopen("texte2.txt","a"))
{
    fputs($fichier,"Nous insérons une nouvelle ligne à la fin du fichier");
}
?>
```

### Copier, supprimer et renommer des fichiers

La fonction copy() permet de copier un fichier :

```
int copy( string source,string destination);
```

La fonction unlink() supprime un fichier définitivement :



```
int unlink(string nom_fichier);
```

La fonction rename() permet de renommer un fichier :

```
int rename( string ancien_name, string nouveau_nom);
```

Exemples :

Copier un fichier

```
<?php
    $fichier = "texte.txt";
    copy($fichier, "../rep/".$fichier);
    echo "fichier copié dans le répertoire rep<br><a href=\"ex3p40.php\">suite</a>";
?>
```

Supprimer un fichier

```
<?php
    $fichier = "texte.txt"; unlink("../rep/".$fichier);
    "fichier $fichier supprimé<br><a ref=\"ex2p40.php\">retour</a>";
    echo "le fichier $fichier a été supprimé";
?>
```

Renommer un fichier

```
<?php
    $fichier = "texte.txt";
    rename($fichier, "nouveau_texte.txt");
    echo "le fichier $fichier a été renommé nouveau_texte.txt";
?>
```

Exercice 2 : (Voir annexe, Gestion des fichiers)

Créer les quatre fichiers suivant : "creer\_fichier.php" , "copy\_fichier.php", "rename\_fichier.php" et "supp\_fichier.php".

Le premier, "creer\_fichier.php" créera le fichier stagiaires.txt et insèrera les deux lignes suivantes :

```
"Voici l'insertion de la première ligne de mon fichier<br>"
"Voici l'insertion de la deuxième ligne de mon fichier<br>"
```

si le fichier existe déjà, son contenu sera supprimer. L'utilisateur sera informé par un message du résultat de l'opération, puis le contenu du fichier sera affiché.

Le second, "copy\_fichier.php" copiera le fichier stagiaires.txt en stagiaires\_sav.txt. On informera l'utilisateur du résultat de l'opération.

Le troisième, "rename\_fichier.php", renomera le fichier stagiaires\_sav.txt en lst\_stagiaires.txt et le contenu du nouveau fichier sera affiché. On informera l'utilisateur du résultat de l'opération.

Le quatrième, "supp\_fichier.php" supprimera les fichiers stagiaires.txt, stagiaires\_sav.txt et lst\_stagiaires.txt. On informera l'utilisateur du résultat de l'opération.

## Chargements de tableaux à partir de fichiers

Vous avez vu que la fonction `file()` place le contenu d'un fichier dans un tableau : chaque ligne du fichier est représentée par un élément du tableau scalaire. Vous pouvez donc utiliser les fonctions des tableaux scalaires pour accéder aux données du fichier.

```
array file(string URL_fichier);
```

Exemple :

```
<?php
    $orders = file("orders.txt");
    // le contenu du fichier est placé dans le tableau $orders
    $num = count($orders);
    // compte le nombre d'éléments du tableau
    if ($num == 0)
    {
        echo "Rien à afficher, réessayer plus tard";
    }
    else
    {
        for ($i = 0; $i < $num ; $i++)
        {
            echo "$orders[$i]<br>"; // affiche chaque ligne du fichier
        }
    }
?>
```

Exercice : (Voir annexe, Chargements de tableaux à partir de fichiers)

Soit le fichier texte `stagiaire_mdb.txt` est issue d'une exportation d'une base access, il contient la liste des stagiaires et leur coordonnées. De plus il a été généré avec les lignes délimitées par des ; et les champs délimités par " (guillemet).

Afficher dans une page, chaque stagiaire avec son adresse et téléphone en dessous.

Exemple :

Numéro de stagiaire : 1

Madame DURANT

21, rue Georges Brassens

94000 Creteil

Téléphone : 01 42 44 12 30

## Gérer des répertoires

La fonction `chdir()` permet de définir le répertoire courant :

```
mixed chdir(string repertoire);
```

Par défaut, le répertoire d'une page PHP qui s'affiche est le répertoire dans lequel elle réside. Il est plus avantageux de basculer les autres répertoires dans ce répertoire pour pouvoir accéder aux fichiers, plutôt que de spécifier le chemin d'accès complet de chaque fichier.

La fonction `opendir()` permet d'ouvrir un répertoire :

```
mixed opendir(string chemin);
```

Cette fonction retourne un pointeur de répertoire, valeur entière qui permet de se référer au répertoire ouvert dans les appels de fonction ultérieurs.

La fonction `readdir()` permet de lire les entrées du répertoire ouvert :

```
string readdir(int pointeur_repertoire);
```

Cette fonction renvoie `false` lorsqu'elle a échoué ou quand la fin du répertoire est atteinte.

Exemple :

```
<?
chdir("../exercice");
$rep = opendir(".");
while ($fichier = readdir($rep))
{
    echo "$fichier<br>";
}
//affiche tous les fichiers du répertoire Exercice
?>
```

La fonction `rewinddir()` permet de revenir au début du répertoire :

```
void rewinddir(int pointeur_repertoire);
```

La fonction `closedir()` permet de fermer le répertoire et de libérer les ressources :

```
void rewinddir(int pointeur_repertoire);
```

La fonction `mkdir()` permet de définir un nouveau répertoire :

```
int mkdir(string chemin_repertoire, int mode);
```

Le second paramètre désigne les permissions d'accès du répertoire UNIX ; Windows ne tient pas compte de ce paramètre.

Exemple :

```
<?
if (mkdir("../rep/new",0700))
{
    echo "Le répertoire a été créé !";
}
else
{
    echo "échec : le répertoire n'a pas pu être créé.";
}
?
```

La fonction `rmdir()` permet de supprimer un répertoire :

```
int rmdir(string nom_repertoire);
```

La fonction `basename` sépare le nom du fichier et le nom du dossier.

```
String basename ( string path [, string suffix])
```

Exercice : (Voir annexe, Gérer des répertoires)

Créer les quatre fichiers suivant : "creer\_ repertoire.php" , "copy\_ repertoire.php", "rename\_ repertoire.php" et "supp\_ repertoire.php".

Le premier, "creer\_ repertoire.php" créera le repertoire1 et le repertoire2 puis le fichier stagiaires.txt dans le repertoire1 et insérera dans ce fichier les deux lignes suivantes :

```
"Voici l'insertion de la première ligne de mon fichier<br>"
"Voici l'insertion de la deuxième ligne de mon fichier<br>"
```

si le fichier existe déjà, son contenu sera supprimer. L'utilisateur sera informé par un message du résultat de l'opération, puis le contenu du fichier sera affiché.

Le second, "copy\_repertoire.php" copiera le fichier stagiaires.txt en stagiaires\_sav.txt dans le repertoire2. On informera l'utilisateur du résultat de l'opération.

Le troisième, "rename\_repertoire.php", renomera le fichier stagiaires\_sav.txt en Ist\_stagiaires.txt dans repertoire1 et le contenu du nouveau fichier sera affiché. On informera l'utilisateur du résultat de l'opération.

Le quatrième, "supp\_fichier.php" supprimera les deux répertoires et leur contenus. On informera l'utilisateur du résultat de l'opération.

## Afficher une image

PHP utilise la bibliothèque GD pour toutes les images hormis les plus simples d'entre elles. PHP permet aussi de créer et de manipuler des images, dans un grand choix de formats, comme gif, png, jpg, wbmp et xpm. PHP peut aussi générer directement des images pour le navigateur, avec la librairie GD.

### Utilisation d'image simple :

On utilise la balise <img> pour afficher une image :

```

```

PHP permet aussi d'afficher une image grâce à la balise <input> :

```
<input name='imageField' type='image' src='Mon_image'>;
```

Lorsque PHP rencontre cette balise, il se contente d'interroger le serveur afin d'obtenir l'image puis la transmet au navigateur et ne fait rien d'autre.

#### ➤ GetImageSize :

La fonction GetImageSize(), nous sera très utile puisqu'elle retourne la taille d'une image et un tableau de 4 éléments. L'index 0 contient la largeur. L'index 1 contient la longueur. L'index 2 contient le type de l'image : 1 = GIF, 2 = JPG, 3 = PNG, 5 = PSD, 6 = BMP. L'index 3 contient la chaîne : "height=xxx width=xxx".

```
array getimagesize ( string filename [, array imageinfo])
```

Ainsi nous pourrions obtenir toutes les informations utiles sur l'image et les placer dans les balises html.

Exercice : (Voir annexe, Afficher une image Exercice 1)

A l'aide d'une liste de sélection (<SELECT>) afficher la liste des images contenu dans le répertoire images (voir figure ci-dessous). Puis permettez à l'utilisateur d'afficher l'image sélectionnée à l'aide de la balise img puis input.

**Liste des images disponibles**

▼

## La librairie GD

Afin de pouvoir utiliser la librairie GD, vous devez au préalable charger ce module. Pour ce faire, ajouter la ligne suivante dans la partie extensions du fichier de configuration de php (php.ini). Exemple :

```
.....
; Dynamic Extensions ;
.....
extension=php_gd2.dll
;
```

La librairie GD et PHP disposent de nombreuses fonctions relatives aux traitements des images, nous nous contenterons de quelques unes.

Ces fonctions utilisent l'entête de fichier, autrement dit elles doivent être utilisées avant toute autre commande transmettant un flux de données.

#### ➤ Imagecreatefromgif :

Crée une nouvelle image gif à partir d'un fichier ou d'une URL.

```
resource imagecreatefromgif ( string filename)
```

#### ➤ imagegif :

Envoie une image GIF vers un navigateur ou un fichier

```
int imagegif ( resource im, string filename)
```

- **ImagecreatefromJpeg :**  
Crée une nouvelle image JPEG à partir d'un fichier ou d'une URL  
`resource imagecreatefromjpeg ( string filename)`

- **imagejpeg :**  
Envoie une image JPEG vers un navigateur ou un fichier.  
`int imagejpeg ( resource im [, string filename [, int quality]])`

- **ImageCreateFromPNG :**  
Crée une nouvelle image PNG à partir d'un fichier ou d'une URL  
`resource imagecreatefrompng ( string filename)`

- **ImagePNG :**  
Envoie une image PNG vers un navigateur ou un fichier  
`int imagepng ( resource im [, string filename])`

- **ImageDestroy :**  
Détruit une image et libère toute la mémoire associée à l'image *im*. *im* est un identifiant d'image valide retourné par une fonction de création d'image ([imagecreate\(\)](#) par exemple)  
`int imagedestroy ( resource im)`

Exemple de création d'image :

```
<?php
    // définition de l'image
    $hauteur = 200;
    $largeur = 200;
    $image = ImageCreate ($largeur, $hauteur);
    // identificateur de la couleur blanc
    $blanc = ImageColorAllocate ($image, 255, 255, 255);
    // identificateur de la couleur noir
    $noir = ImageColorAllocate ($image, 0, 0, 0);
    // dessin de l'image
    ImageFill($image, 0, 0, $noir); // peint un fond noir
    ImageLine ($image, 0, 0, $largeur, $hauteur, $blanc);
    // trace un trait blanc du coin supérieur gauche (0, 0)
    // jusqu'au coin inférieur droit ($largeur, $hauteur)
    ImageString($image, 4, 50, 150, "Ventes", $blanc);
    // ajoute la légende "Ventes" au graphique de police 4, au point initial (50, 150)
    // génération de l'image
    Header ("Content-type : image/png");
    ImagePng ($image);
    // nettoyage
    ImageDestroy($image);
?
```

Exercice : (Voir annexe, Afficher une image Exercice 2)

A l'aide d'une liste de sélection (<SELECT>) afficher la liste des images contenu dans le répertoire images (comme exercice précédent). Puis permettez à l'utilisateur d'afficher l'image sélectionnée en utilisant les fonctions de la librairie GD.

## 9. Gestion des erreurs

### Introduction

PHP constitue un environnement très pratique pour le débogage et la gestion des erreurs. Il permet de détecter les erreurs et d'y réagir tout en laissant la latitude quant à la façon dont les messages d'erreurs sont affichés.

#### 9.1. Les types d'erreurs PHP

Trois grands groupes d'erreurs de programmation peuvent être distingués :

erreur de syntaxe ou de compilation, survenant lorsque la syntaxe du code est erronée.

erreur de sémantique ou d'exécution, qui se produit lorsque le programme exécute un code dont la syntaxe est correcte ;

erreur logique qui ne provoque pas l'apparition d'un message d'erreur mais qui a pour résultat que le programme effectue autre chose que ce qui était souhaité par le programmeur.

Il y a aussi les erreurs d'environnement qui ne sont pas dues à une faute du programmeur mais à un ensemble de facteurs d'environnement sur lequel le programmeur n'a aucun contrôle.

#### Erreur de syntaxe

Une erreur de syntaxe se produit lors de l'analyse du code, avant son exécution.

L'analyseur indique le numéro de ligne où s'est produite l'erreur.

Exemple :

```
1 <?php
2     $var = "test d'erreur"
3     echo $var;
4 ?>
```

il manque un ";" de fin d'instruction



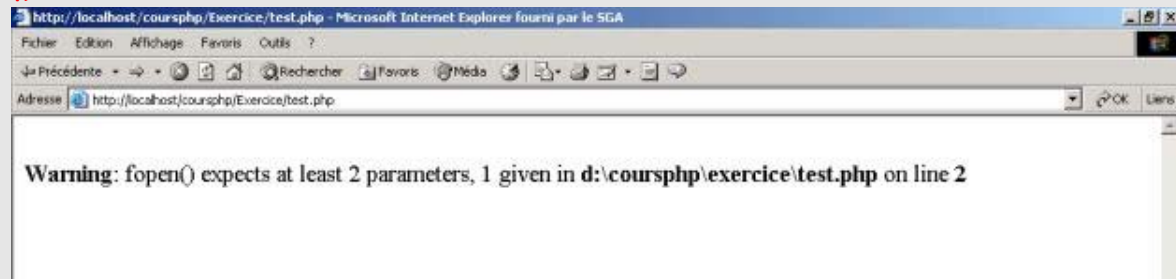
L'analyseur détecte une erreur à la ligne 3 car il ne doit pas y trouver de nouvelle instruction, le ";" de fin d'instruction étant absent de la ligne 2

#### Erreur de sémantique

Une erreur sémantique n'est pas détectée par l'analyseur car sa syntaxe est correcte : elle ne survient que lorsque PHP tente d'exécuter la ligne.

Exemple :

```
<?php
    fopen("fichier.txt");
?>
```



La ligne ne présente pas d'erreur de syntaxe, mais la fonction fopen() nécessite deux paramètres. Une erreur de sémantique n'est pas toujours facilement identifiable.

## Erreur logique

L'erreur logique est délicate à identifier car elle correspond à du code parfait par sa syntaxe et sa sémantique : elle ne produit donc pas de message d'erreur. Dans la plupart des cas, elle aboutit à un résultat qui n'est pas celui attendu.

## Erreur d'environnement

Même dans le cas où le code est dépourvu de tout bogue, il n'est pas garanti qu'il s'exécutera toujours sans erreur. En effet tout programmeur doit parfois s'appuyer sur des éléments qu'il ne contrôle pas, comme l'échec d'ouverture d'une base de données, l'impossibilité d'ouvrir un fichier include ...

Il est donc important de s'assurer de la présence d'une gestion d'erreur adaptée à une quelconque erreur d'environnement.

## Les messages d'erreur PHP

Les messages d'erreurs PHP sont particulièrement descriptifs. Ils correspondent tous à la forme suivante :

*Niveau\_erreur : message\_erreur in Nom\_fichier on line #*

### 9.2. Les niveaux d'erreur PHP

- erreur d'analyse : "Parse error"

C'est une erreur de syntaxe détectée par l'analyseur.

De profondes modifications dans le moteur PHP lors du passage de PHP3 à PHP4 entraîne un effet totalement différent d'une erreur d'analyse. PHP3 est un langage interprété : le moteur interprète une ligne de code, l'exécute et passe à la suivante. PHP4 précompile, et PHP5 compile le script avant d'exécuter le script. Ce changement implique que les erreurs d'analyse que PHP3 détectait pas avant d'avoir atteint la ligne fautive sont épinglées par PHP avant l'exécution de la moindre ligne de code.

- erreur fatale : "Fatal error"

C'est une erreur de sémantique ou d'environnement que PHP ne peut pas surmonter : elle entraîne l'interruption immédiate du script, car PHP ne peut pas continuer sans l'exécution de la ligne où s'est produite l'erreur.

- avertissement : "Warning"

S'il rencontre un avertissement, PHP tente de poursuivre l'exécution du script. Il s'agit en général d'erreur suffisamment grave pour compromettre la bonne exécution du programme (de nombreuses erreurs d'environnement entraînent des warning). Si aucune gestion des erreurs ne permet de réagir à ces erreurs, la défaillance initiale provoque souvent d'autres erreurs alors que le script poursuit son exécution.

remarque

Les remarques (*notices*) sont dues à des erreurs de faible importance, telle qu'une variable non initialisée. Par défaut PHP n'est pas configuré pour afficher les remarques, mais elles peuvent être activées et constituent alors un bon moyen de détecter certaines erreurs.

## Définition du niveau de rapport d'erreur

Les différents type d'erreur et d'alerte sont :

Valeur	Constante	Description	Note
1	E_ERROR	Erreur fatale d'exécution	
2	E_WARNING	Alerte d'exécution (erreur non fatale)	
4	E_PARSE	Erreur de compilation	
8	E_NOTICE	Notes d'exécution (moins critique que les alertes)	
16	E_CORE_ERROR	Erreurs qui surviennent lors de l'initialisation de PHP	PHP4 et supérieur
32	E_CORE_WARNING	Alertes qui surviennent lors de l'initialisation de PHP	PHP4 et supérieur
64	E_COMPILE_ERROR	Erreur fatale de compilation	PHP4 et supérieur
128	E_COMPILE_WARNING	Alerte de compilation (erreur non fatale)	PHP4 et supérieur
256	E_USER_ERROR	Erreur générée par l'utilisateur	PHP4 et supérieur
512	E_USER_WARNING	Alerte générée par l'utilisateur	PHP4 et supérieur
1024	E_USER_NOTICE	Note générée par l'utilisateur	PHP4 et supérieur
	E_ALL	Toutes les erreurs ci-dessus	

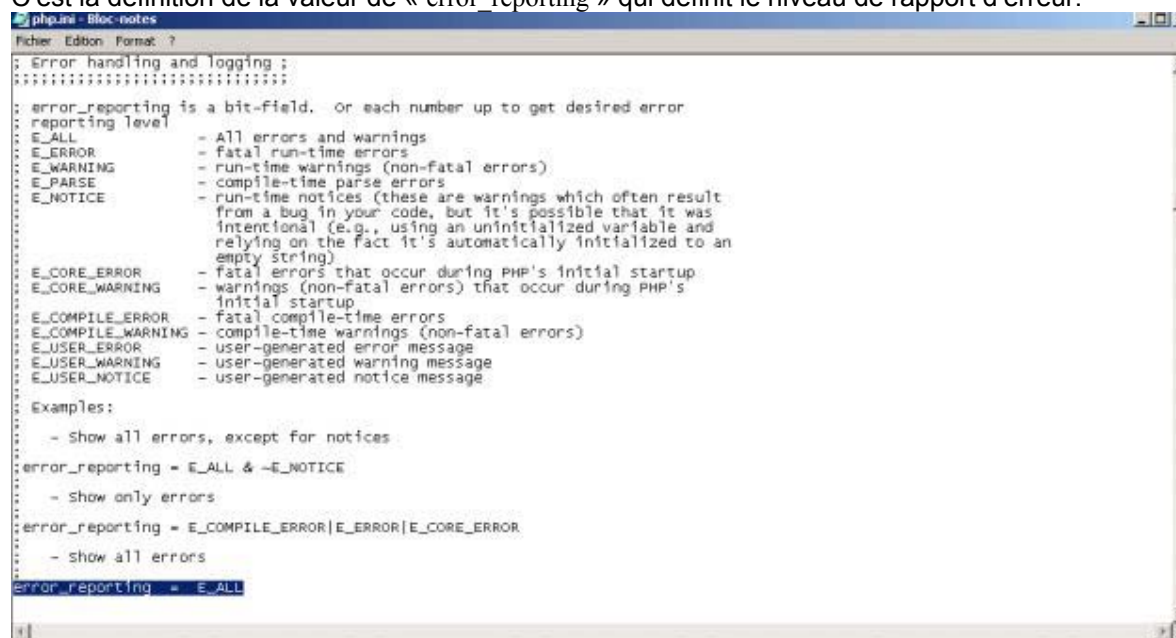


Les valeurs ci-dessus (numériques ou symboliques) sont utilisées pour construire un champ de bits qui spécifie quelles erreurs rapporter. Les opérateurs de bits sont utilisables pour combiner ces valeurs et conserver uniquement celles qui sont intéressantes.

Le niveau de rapport d'erreur est défini dans le fichier « php.ini » et au niveau du script par la fonction `error_reporting()`.

### Définition du niveau de rapport d'erreur dans le « php.ini »

C'est la définition de la valeur de « `error_reporting` » qui définit le niveau de rapport d'erreur.



Par défaut, le niveau de rapport d'erreur est défini à `E_ALL & ~E_NOTICE` (à la valeur 7 pour « PHP3.ini »).

### Définition du niveau de rapport d'erreur dans le script :

La fonction `error_reporting()` accepte un argument qui spécifie les niveaux d'erreur à afficher.

Pour déterminer la valeur de l'argument, il faut ajouter les valeurs des niveaux d'erreurs à afficher. Pour désactiver tout rapport d'erreur, il faut appeler la fonction avec l'argument à 0. Le paramétrage par défaut est équivalent à l'appel de la fonction avec la valeur 7 (1 + 2 + 4). Pour l'affichage de toutes les erreurs, il faut un appel de la fonction à 15 (1 + 2 + 4 + 8).

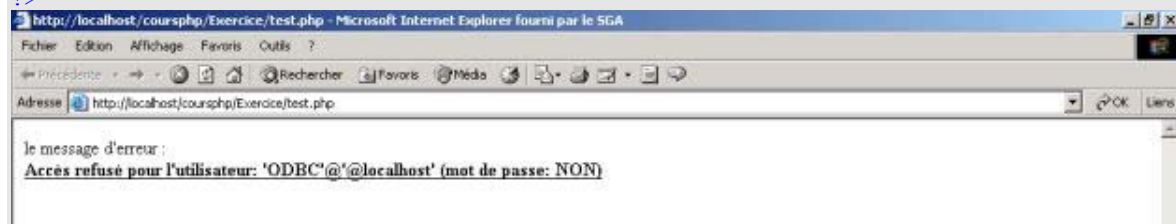
Il est conseillé de remplacer les messages d'erreur standard par une gestion harmonieuse des erreurs d'environnement et de les journaliser sans afficher les messages standard PHP à l'utilisateur.

### 9.3. Gérer les erreurs

La plupart des fonctions PHP retournent 0 en cas d'erreur. Avec un test il est donc simple de gérer l'erreur.

Exemple :

```
<?php
if(mysql_connect($db, $utilisateur, $password))
{
    // Le code d'accès à la base de données se situe ici'
}
?>
```



### Suppression des messages d'erreur

Dans l'exemple précédent, en cas d'erreur le message d'erreur sera affiché :

Il est possible de supprimer le message d'erreur de deux façons :

avec la fonction `error_reporting()` vue précédemment ;

avec l'opérateur de contrôle d'erreur "@" qui permet d'ignorer les messages d'erreur générés par l'expression qu'il préfixe.

Si l'option `track_errors` de "php.ini" est activée, les messages d'erreurs sont sauves dans la variable globale `$php_errormsg`, qui est écrasée à chaque erreur.

Exemple :

```
<?php
    if (@mysql_connect($db, $utilisateur, $password))
    {
        // Le code d'accès à la base de données se situe ici'
    }
    echo "le message d'erreur : <br><b><u>$php_errormsg</u></b>";
?>
```

### Message d'erreur personnalisé

Un message d'erreur personnalisé pour l'utilisateur sera mieux perçu que le message d'erreur standard.

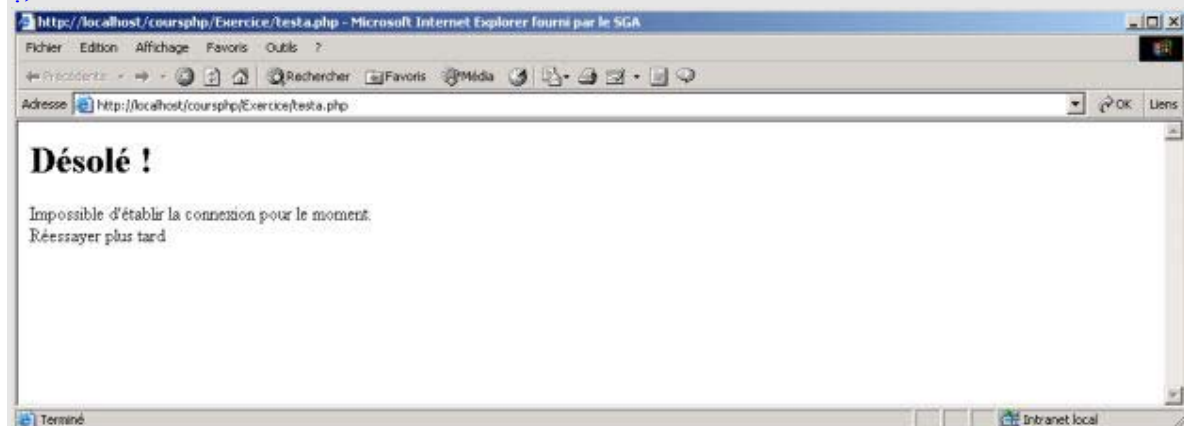
Il suffit d'ajouter au test une clause "else" qui peut contenir le message d'erreur et éventuellement une instruction pour terminer l'exécution du script.

La fonction `die()` entraîne la fin d'exécution du script et l'affichage d'une chaîne de caractères qu'elle prend en argument (elle peut également exécuter une fonction en argument).

La fonction `exit()` joue le même rôle sans message.

Exemple :

```
<?php
    if (@mysql_connect($db, $utilisateur, $password))
    {
        // Le code d'accès à la base de données se situe ici'
    }
    else
    {
        exit("<h1>Désolé !</h1><p>Impossible d'établir la connexion pour le
                                                moment.<br>Réessayer plus tard</p>");
    }
    echo "cette partie du script ne sera affichée que si la connexion a pu se faire";
?>
```



Il est possible également de rediriger l'utilisateur vers une page d'erreur si aucun contenu n'a encore été envoyé au navigateur avec la fonction `header()` (fonction d'en-tête) :

Exemple :

```
<?php
    header("location:error.php") // on redirige vers la page error.php
?>
```

Il est également possible d'imprimer un élément de JavaScript côté client, si le navigateur prend en charge JavaScript

Exemple :

```
<?php
    if ( !isset($_REQUEST['var']))
    {
        echo "<form action=\"test.php\">\n";
    }
?>
```

```

        Votre nom <input type="text" name="var">\n<br>
        <input type="submit" value="envoyer">\n</form>;
    }
    else {
        if (empty($_REQUEST['var'])) { // l'utilisateur n'a pas rempli le champ $var
            echo "<script language='javascript'>";
            alert("\"vous devez remplir le champ\"") </script>;
            echo "<form action='test.php'>\n";
            Votre nom <input type="text" name="var">\n<br>
            <input type="submit" value="envoyer">\n</form>;
        }
        else {
            // utilisation de $_REQUEST['var']
        }
    }
}
?>

```

Si l'utilisateur fait l'erreur de ne pas remplir le formulaire avant de l'envoyer, alors le message javascript s'affiche



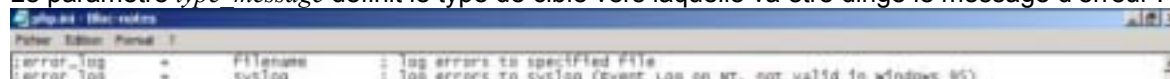
### Journalisation des erreurs

La fonction `error_log()` met en journal toute erreur pouvant subvenir.

```
int error_log( string message, int type_message [,string destination [,string extra_headers]]);
```

Le paramètre *message* est le message d'erreur qui doit être mis dans le journal.

Le paramètre *type\_message* définit le type de cible vers laquelle va être dirigé le message d'erreur :



- 0 : le journal d'erreur système de PHP, qui peut être défini dans « php.ini » par la directive `error_log` :  
par défaut, c'est le journal du serveur web (error.log d'Apache : `.../apache/logs/error.log`)
- 1 : une adresse électronique transmise comme troisième paramètre
- 2 : un port de débogage (si le débogage est activé)
- 3 : un fichier journal spécifié dans le troisième argument

Le paramètre *destination* est l'adresse électronique si le *type\_message* est 1, et l'emplacement du fichier journal si le *type\_message* est 3.

## 10. Programmation orienté objet

### 10.1. Quelques concepts de programmation orientée objet

La programmation orientée objet, ou programmation modulaire, permet la réutilisation de code. PHP permet cette modularité via la programmation de bibliothèques ou de classes.

Un logiciel orienté objet est conçu et construit sous la forme d'un ensemble d'objets indépendants dotés à la fois d'attributs et d'opérations qui interagissent pour répondre à vos besoins. Les « attributs » sont des propriétés ou des variables qui se rapportent à l'objet. Les « méthodes » sont des actions ou des fonctions que l'objet peut accomplir soit pour se modifier lui-même, soit pour produire un effet externe.

Les données contenues dans un objet sont essentiellement accessibles par le biais des méthodes de l'objet qui forment l'« interface » de l'objet.

La plupart des scripts Web restent conçus et écrits en appliquant une méthodologie orientée fonctions, car ils sont de petite taille et relativement simples. Mais nombre de projets Web évoluent d'un ensemble de pages hyper liées vers des applications complexes qui requièrent une méthodologie de développement mûrement réfléchie. L'orientation objet peut aider à gérer la complexité des projets logiciels, à augmenter la réutilisabilité du code, et par conséquent à réduire les coûts de maintenance.

Un objet est une collection unique et identifiable de données stockées et de méthodes qui agissent sur ces données. Ces objets peuvent être regroupés en « classes ».

Une classe est un ensemble d'objets qui peuvent être différents les uns des autres, mais qui se caractérisent par des points communs. Elle contient des objets qui présentent tous des méthodes se comportant de la même manière, et des attributs identiques représentant les mêmes choses, bien que les valeurs de ces attributs puissent varier d'un objet à l'autre au sein de la classe.

Différentes classes doivent pouvoir présenter des comportements différents pour la même méthode : c'est le « polymorphisme », qui caractérise le comportement des objets, plutôt que les objets eux-mêmes. En PHP, les fonctions membres d'une classe peuvent être polymorphiques.

Le concept d'« héritage » permet de créer une relation hiérarchique entre les classes, au moyen de sous-classes. Une sous-classe hérite des attributs et des méthodes de sa classe parent. Ce concept permet d'élaborer et d'enrichir le jeu de classes existant, de rendre le code plus réutilisable, et constitue un des atouts indéniables de la programmation orientée objets : des méthodes peuvent ainsi être implémentées une seule fois dans une classe parent au lieu de l'être de nombreuses fois dans des sous-classes séparées.

Dans le cadre d'un développement pour le Web, les classes peuvent notamment être utilisées pour représenter des pages Web, des composants d'interface utilisateur, des cartes d'achat, des gestionnaires d'erreur, des catégories de produits, ou des clients.

En PHP 5, il y a un tout nouveau modèle objet. La gestion des objets en PHP a été complètement réécrite, permettant de meilleures performances ainsi que plus de fonctionnalités.

## 11. Les classes et les objets (PHP 4)

### Les classes

Une classe est une collection de variables et de fonctions qui fonctionnent avec ces variables. Les classes forment un type de variable Objet.

#### 11.1. Création d'une classe

La création d'une classe s'effectue au moyen du mot clé `class` :

##### Structure de la définition d'une classe

```
class nom_classe {  
  
}
```

Une classe doit être dotée d'attributs et de méthodes.

#### Création d'attributs :

Lorsque vous déclarez des variables au sein d'une définition de classe , ces variables deviennent les attributs de la classes ou appelées aussi membres de classe

La déclaration des attributs ce fait par l'intermédiaire du mot clé `var` :

##### Structure de la définition d'un attribut

```
class nom_classe {  
  
    var $attribut1;  
    var $attribut2;  
  
}
```

#### Création de méthodes :

Lorsque vous déclarez des fonctions au sein de la définition de la classe, elles deviennent les méthodes de la classe :

##### Structure de la définition d'une méthode

```
class nom_classe {  
  
    function nom_fonction ($parametres) {  
  
        corps de la fonction  
  
    }  
  
}
```

#### Constructeur de classe

Il existe une méthode spéciale, le constructeur, qui est est la fonction appelée automatiquement par la classe lorsque vous créez une nouvelle instance d'une classe. La fonction constructeur a le même nom que la classe. En PHP 3, une fonction devient le constructeur si elle porte le même nom que la classe. En PHP 4, une fonction devient un constructeur si elle porte le même nom que la classe dans laquelle elle est définie. La différence est subtile, mais cruciale.

Le constructeur est très utile pour initialiser les données membres lors de l'instanciation.

### Structure de la définition d'un constructeur

```
class nom_classe {

    function nom_classe ($parametres) {

        // méthode constructeur
        corps de la fonction

    }

}
```

Attention : Vous ne pouvez pas attribuer le même nom à plusieurs fonctions.

### 11.2. Instanciation

L'instanciation correspond à la création d'objets. Les classes forment un type de variable. Pour créer une variable du type désiré, vous devez utiliser l'opérateur *new*.

Vous devez spécifier la classe dont l'objet créé sera l'instance et fournir les paramètres éventuellement requis par le constructeur de la classe.

```
$objet = new nom_classe($parametres);
```

Exemple :

```
<?php
// déclaration de la classe ma_classe
class ma_classe {
    function ma_classe($mavar)
    {
        echo "Le constructeur appelle le paramètre <b>$mavar</b><br>";
    }
}

//Instanciatioin de la classe ma_classe
$a = new ma_classe("nouveau paramètre");
?>
```

création de l'objet \$a de type ma\_classe qui permet d'afficher :

Le constructeur appelle le paramètre **nouveau paramètre**.

### 11.3. Accès aux attributs et aux méthodes d'une classe

Au sein d'une classe, le pointeur \$this vous permet de faire référence à un attribut :

```
$this->attribut
```

Pour appeler une méthode il faut spécifier son nom, suivi des paramètres requis placés entre parenthèses, et préciser l'objet auquel elle appartient :

```
$objet->methode($parametres);
```

Exemple :

```
<?php
// déclaration de la classe ma_classe
class ma_classe {
    var $attribut;
    function operation($mavar) {
        $this->attribut = $mavar;
        // accès à l'attribut au sein de la classe
        echo $this->attribut." attribut";
    }
}

$a = new ma_classe();
$a->operation ("nouvel");
//affiche : nouvel attribut
?>
```

### Les accesseurs

Les accesseurs sont des fonctions dans la classe qui d'effectuer tous vos accès aux attributs d'une classe par le biais d'une seule section de code, plutôt que directement, à différents niveaux du code. Avec un point d'accès unique, vous pouvez décider toute sorte de modifications : quels que soient les changements à

apporter, il suffira de modifier la ou les fonctions accesseur, les autres sections de code ne s'en trouvant pas affectées.

Une fonction accesseur peut initialement se formuler de la manière suivante :

```
<?php
class accesseur {
    var $attribut;
    function get_attribut() {
        return $this->attribut;
    }

    function set_attribut($newvalue) {
        $this->attribut = $newvalue;
    }
}
?>
```

Ce code fournit des fonctions qui permettent d'accéder à l'attribut \$attribut : la fonction get\_attribut() renvoie la valeur de \$attribut, tandis que la fonction set\_attribut() affecte une nouvelle valeur à \$attribut.

#### 11.4. Héritage

Une classe peut être déclarée comme étant une sous-classe d'une autre classe à l'aide du mot clé extends.

Exemple :

```
<?php
class B extends A {
    var $attributB;
    function operationB() {
        corps de la fonction
    }
}
?>
```

La classe B est sous-classe de la classe A : elle hérite des attributs et des méthodes de la classe A, par contre la classe A, parent de la classe B ne peut en aucun cas hériter des attributs et des méthodes de B.

Exemple :

```
<?php
class parent {
    var $attribut1;
    function operation1() {
        $this->attribut1;
        echo "L'attribut de parent est <b>$this->attribut1 </b>(grace a
            operation1) .<br><br>";
    }
    //accesseur de attribut1
    function get_attribut1() {
        return $this->attribut1;
    }
    function set_attribut1 ($new_attribut) {
        $this->attribut1 = $new_attribut;
    }
}

class fils extends parent {
    var $attribut2;
    function operation2() {
        $this->attribut2;
        echo "L'attribut de fils est <b>$this->attribut2 </b>(grace à
            l'operation2) .<br><br>";
    }
}

//Instanciation de la classe fils dans l'Objet b
$b = new fils();

//Affectation d'une nouvelle valeur grace à l'accesseur
$b->set_attribut1(40);
//Affichage de la valeur de l'attribut1 grace à l'accesseur
echo "L'attribut de parent est <b>". $b->get_attribut1(). " </b> (grace à
    get_attribut1) .<br>";

//Exécution de la méthode de la classe fils qui est héritée de la classe parent
//Affichage de la valeur de l'attribut1 grace à la méthode operation1 de la class parent
$b->operation1(); // affiche : L'attribut de parent est 40
```

```
//Affectation d'une nouvelle valeur sans accesseur
$b->attribut2 = 10;
//Affichage de la valeur de l'attribut2 sans accesseur
echo "L'attribut de parent est <b>".$b->attribut2."</b> ( grace a l'attribut2).<br>";
//Exécution de la méthode de la classe fils
//Affichage de la valeur de l'attributé grace à la méthode operation2 de la class fils
$b->operation2(); // affiche : L'attribut de fils est 10

// tous les accès aux attributs et aux opérations sont possibles pour un objet de la
//classe fils
//Instanciation de la classe parent dans l'Objet a
$a = new parent();
//Affectation d'une nouvelle valeur grace à l'accesseur
$a->set_attribut1(40);
//Affichage de la valeur de l'attribut1 grace à l'accesseur
echo "L'attribut de parent est <b>".$a->get_attribut1()." </b> ( grace a
get_attribut1).<br>";

//Exécution de la méthode de la classe parent
//Affichage de la valeur de l'attribut1 grace à la méthode operation1 de la class parent
$a->operation1(); //

//Affectation d'une nouvelle valeur sans accesseur
$a->attribut2 = 10;
//Affichage de la valeur de l'attribut2 sans accesseur
echo "L'attribut de parent est <b>".$a->attribut2."</b>grace a l'attribut2.<br>";

//Exécution de la méthode de la classe fils operation2 n'est pas autorisée
//Puisque la classe parent n'hérite pas de la classe fils
//Affichage de la valeur de l'attributé grace à la méthode operation2 de la class fils
//Affiche Fatal error: Call to undefined function: operation2() in
//d:\coursphp\exercice\test.php on line 77
$a->operation2(); // affiche : L'attribut de fils est 10
?>
```

### L'opérateur de contexte de classe (::)

L'opérateur :: permet de faire référence aux fonctions et variables d'une classe de base, ou bien d'utiliser des méthodes de classes qui n'ont pas encore d'objets créés.

Exemple :

```
<?php
class A {
    function example() {
        echo"Je suis la fonction originale A::example().<br />\n";
    }
}

class B extends A {
    function example() {
        echo "Je suis la fonction redéfinie B::example().<br />\n";
        A::example();
    }
}

// Il n'y a pas d'objets de classe A.
// L'affichage est :
// Je suis la fonction originale A::example().<br />
A::example();

// Création d'un objet de la classe B.
$b = new B;
// L'affichage est :
// Je suis la fonction redéfinie B::example().<br />
// Je suis la fonction originale A::example().<br />
$b->example();
?>
```

Les exemples ci-dessus appellent la fonction example() dans la classe A, mais il n'y a pas encore d'objet de classe A, alors il n'est pas possible d'écrire \$a->example(). A la place, on appelle la fonction example() comme une fonction de classe, c'est-à-dire avec le nom de la classe elle-même, et sans objet.



Il y a des fonctions de classe, mais pas de variables de classe. En fait, il n'y a aucun objet au moment de l'appel de la fonction. Donc, une fonction de classe ne peut accéder à aucune variable (mais elle peut accéder aux variables locales et globales). Il faut proscrire l'utilisation de \$this.

Dans l'exemple ci-dessus, la classe B redéfinit la fonction exemple(). La définition originale dans la classe A est remplacée par celle de B, et n'est plus accessible depuis B, à moins que vous n'appeliez spécifiquement la fonction exemple() de la classe A avec l'opérateur ::. Ecrivez A::exemple() pour cela (en fait, il faudrait écrire parent::exemple(), comme expliqué dans la section suivante).

Dans ce contexte, il y a un objet courant, qui peut avoir d'autres variables objets. De ce fait, lorsqu'il est utilisé depuis une méthode d'un objet, vous pouvez utiliser \$this.

### 11.5. Redéfinition

Il est possible, dans une sous-classe, de re-déclarer les mêmes attributs que ceux de la classe parent en leur donnant une valeur par défaut différente, ainsi que les mêmes méthodes en leur donnant une fonctionnalité différente : c'est la redéfinition.

La redéfinition d'attributs ou de méthodes dans une sous-classe n'affecte pas la classe parent. Lorsqu'une redéfinition est fournie, elle devient prépondérante et remplace la définition d'origine. PHP ne permet pas de redéfinir une fonction tout en gardant la possibilité d'invoquer la version définie dans le parent.

Exemple :

```
<?php
class A {
    var $attribut = "valeur défaut";
    function operation() {
        echo "<b>Définition d'une classe</b><br>";
        echo "La valeur de \$attribut est \$this->attribut";
        echo "<br><br>";
    }
}

class B extends A {
    var $attribut = "valeur différente";
    function operation() {
        echo "<b>Nouvelle définition</b><br>";
        echo "La valeur de \$attribut est \$this->attribut";
        echo "<br><br>";
    }
}

//Instanciation de la classe A dans l'Objet a
$a = new A();
/* affiche : Définition d'une classe
La valeur de $attribut est valeur défaut*/
$a->operation();

//Instanciation de la classe B dans l'Objet b
$b = new B();
/* affiche : Nouvelle définition
valeur de $attribut est valeur différente*/
$b->operation();
?>
```

### Manipulation de classes et d'objets d'origine similaire.

Exemple :

```
<?php
class vehicule {
    var $roues;
    var $passagers;
    var $classname = "véhicule";
    var $nom = "";

    function vehicule($val1, $val2) {
        $this->roues = $val1;
        $this->passagers = $val2;
    }

    function init_nom($nom_type) {
        $this->nom = $nom_type;
    }

    function je_suis() {
        echo "je suis un véhicule à $this->roues roues";
    }
}
```

```

        echo " et $this->passagers passagers, ";
        echo "mon type est $this->classname.";
        if ($this->nom) {
            echo "Je m'appelle $this->nom.";
            echo "<br><br>";
        }
    }
}

// héritage de la classe véhicule
class velo extends vehicule {
    var $classname = "vélo";
    // redéfinition de l'attribut $classname
    // "construction" de la classe vélo
    function velo() {
        $this->vehicule(2, 1);
    }
}

// véhicule motorisé issu de la classe vehicule
class mvehicule extends vehicule {
    var $classname = "motorisé";
    // redéfinition de l'attribut $classname
    function init_moteur($cylindre) {
        $this->moteur = $cylindre;
    }

    function je_suis() {
        // on surcharge la fonction je_suis() de la classe vehicule
        echo "Mon moteur est de $this->moteur cc.<br>";
        vehicule::je_suis(); // appel à la méthode je_suis() de la classe vehicule
    }
}

// les classes voiture et moto héritent de la classe mvehicule
class voiture extends mvehicule {
    var $classname = "voiture";
    function voiture() {
        $this->vehicule(4,5);
    }
}

class moto extends mvehicule {
    var $classname = "moto";
    function moto() {
        $this->vehicule(2, 2);
    }
}

// création des objets bmx, cb, twingo
$bmx = new velo;
$cb = new moto;
$twingo = new voiture;
$bmx->init_nom ("BMX");

//initialisation de ces objets
$twingo->init_nom("TWINGO");
$twingo->init_moteur(1600);
$cb->init_nom("CB 500");
$cb->init_moteur(500);

// on appelle la fonction je_suis() de ces objets pour savoir qui ils sont
$bmx->je_suis();
$twingo->je_suis();
$cb->je_suis();
?>

```

## Les classes et les objets (PHP 5)

PHP 5 inclut un nouveau modèle objet. Le traitement des objets en PHP a complètement été réécrit pour arriver à de meilleures performances et plus de fonctionnalités. Dans les versions précédentes de PHP, les objets sont traités comme des types primitifs (par exemple les entiers ou les chaînes de caractères). L'inconvénient de cette méthode est que sémantiquement, l'objet en entier était copié lorsqu'une variable était assignée ou passée comme paramètre à une fonction. Dans la nouvelle approche, les objets sont référencés par un pointeur et non pas leur valeur (on peut penser à un pointeur en tant qu'identifiant d'objet).

Beaucoup de développeurs PHP ne se rendent pas compte des caprices lors de la copie dans l'ancien modèle objet et, par conséquent, la majorité des applications PHP devrait fonctionner directement ou avec très peu de modifications.

Les modifications apportées à PHP 5 et au Zend Engine 2 augmentent considérablement les capacités et les performances de PHP. Une attention toute particulière a été apportée à ce que cette nouvelle version soit la plus compatible possible avec les scripts antérieurs. Ainsi la migration de votre code de PHP 4 vers PHP 5 devrait être aisée. La plupart des scripts PHP 4 devraient être prêts à fonctionner sans nécessiter la moindre modification. Il existe toutefois quelques différences et vous devriez tester vos codes avant de changer de version en production

### Quelques exemples de changements :

- Un objet sans propriété n'est plus considéré comme vide (`empty()`).
- Dans certains cas, les classes doivent être déclarées avant d'être utilisées. Cela survient uniquement si les nouvelles fonctionnalités de PHP 5 sont utilisées. Sinon, le comportement sera le même qu'avant.
- Les fonctions `get_class()`, `get_parent_class()` et `get_class_methods()` retournent désormais le nom de la classe comme elle a été déclarée (sensible à la casse), ce qui peut causer des problèmes dans vos anciens scripts qui utilisent le comportement précédent (le nom de la classe était toujours retourné en minuscules). Une solution possible est de rechercher ces fonctions dans tous vos anciens scripts et d'utiliser la fonction `strtolower()`.

### 11.6. Création d'une classe

Chaque définition de classe commence par le mot-clé `class`, suivi par le nom de la classe, qui peut être quelconque à condition que ce ne soit pas un mot réservé en PHP. Suivent une paire de parenthèses contenant la définition des membres et des méthodes. Une pseudo-variable `$this` est disponible lorsqu'une méthode est appelée depuis un contexte objet. `$this` est une référence à l'objet appelé (habituellement, l'objet auquel la méthode appartient, mais ce peut être un autre objet si la méthode est appelée de manière statique depuis le contexte d'un autre objet). Ce comportement est illustré dans l'exemple suivant : Exemple 19-1. La variable `$this` en programmation objet

#### Structure de la définition d'une classe

```
class nom_classe {  
  
}
```

Exemple :

```
/* définition simple d'une classe */  
<?php  
class SimpleClass  
{  
    // déclaration d'un membre  
    public $var = 'une valeur par défaut';  
  
    // déclaration de la méthode  
    public function displayVar() {  
        echo $this->var;  
    }  
}
```

```
}
?>
```

## Constructeurs et destructeurs

### Constructeurs

PHP 5 permet aux développeurs de déclarer des constructeurs pour les classes. Les classes qui possèdent une méthode constructeur appellent cette méthode à chaque création d'une nouvelle instance de l'objet, ce qui est intéressant pour toutes les initialisations dont l'objet a besoin avant d'être utilisé.

```
void __construct ( [mixed args [, ...]] )
```

Note : Les constructeurs parents ne sont pas appelés implicitement si la classe enfant définit un constructeur. Si vous voulez utiliser un constructeur parent, il sera nécessaire de faire appel à `parent::__construct()`.

Exemple :

```
<?php
class BaseClass {
    function __construct() {
        print "In BaseClass constructor\n";
    }
}

class SubClass extends BaseClass {
    function __construct() {
        parent::__construct();
        print "In SubClass constructor\n";
    }
}

$obj = new BaseClass();
$obj = new SubClass();
?>
```

Pour des raisons de compatibilité ascendante, si PHP 5 ne peut pas trouver une fonction `__construct()` pour une classe donnée, il cherchera une fonction constructeur représentée, comme dans l'ancien style (PHP < 5), par le nom de la classe. Effectivement, cela signifie que le seul cas où il pourrait y avoir un problème de compatibilité est celui où votre classe contiendrait une méthode nommée `__construct()` et que vous en ayez un autre usage.

### Destructeurs

```
void __destruct ( void )
```

PHP 5 introduit un concept de destructeur similaire aux autres langages orientés objet, comme le C++. La méthode destructeur doit être appelée aussitôt que toutes les références à un objet particulier sont effacées ou lorsque l'objet est explicitement détruit.

Exemple :

```
<?php
class MyDestructableClass {
    function __construct() {
        print "In constructor\n";
        $this->name = "MyDestructableClass";
    }

    function __destruct() {
        print "Destruction de " . $this->name . "\n";
    }
}

$obj = new MyDestructableClass();
?>
```

Tout comme le constructeur, le destructeur parent n'est pas appelé implicitement par le moteur. Pour exécuter le destructeur parent, vous devez appeler explicitement la fonction `parent::__destruct` dans le corps du destructeur.

### 11.7. Création d'une instance

Pour créer une instance d'un objet, un nouvel objet doit être créé et assigné à une variable. Un objet doit toujours être assigné lors de la création d'un nouvel objet à moins qu'un l'objet ait un constructeur défini qui lance une exception en cas d'erreur.

Exemple :

```
<?php
    $instance = new SimpleClass();
?>
```

### 11.8. Assignation d'un objet

Lors de l'assignation d'un instance déjà créée d'un objet à une variable, la nouvelle variable accédera à la même instance de l'objet assigné. Ce comportement est le même que lors du passage d'une instance à une fonction. Une nouvelle instance d'un objet déjà créé peut être effectuée par clonage.

Exemple :

```
<?php
    $assigned = $instance;
    $reference =& $instance;

    $instance->var = '$assigned aura cette valeur';

    $instance = null; // $instance et $reference deviennent null

    var_dump($instance);
    var_dump($reference);
    var_dump($assigned);

    /*
    L'exemple ci-dessus va afficher :
    NULL
    NULL
    object(SimpleClass)#1 (1) {
        ["var"]=> string(30) "$assigned aura cette valeur"
    }
    */
?>
```

### 11.9. Héritage

Une classe peut hériter des méthodes et des membres d'une autre classe en utilisant le mot clé extends dans la déclaration. Il n'est pas possible d'étendre de multiples classes, une classe peut uniquement hériter d'une seule classe de base.

Les méthodes et membres hérités peuvent être surchargés, à moins que la classe parent ait défini une méthode comme final. Pour surcharger, il suffit de redéclarer la méthode avec le même nom que celui défini dans la classe parent. Il est possible d'accéder à une méthode ou un membre surchargé avec l'opérateur parent::

Exemple :

```
<?php
class ExtendClass extends SimpleClass
{
    // Redéfinition de la méthode parent
    function displayVar()
    {
        echo "Classe étendue\n";
        parent::displayVar();
    }
}

$extended = new ExtendClass();
$extended->displayVar();
?>
```

L'exemple ci-dessus va afficher :

Classe étendue  
une valeur par défaut

### 11.10. Visibilité

La visibilité d'une propriété ou d'une méthode peut être définie en préfixant la déclaration avec un mot-clé : public, protected ou private. Les éléments déclarés publics (public) peuvent être utilisés par n'importe quelle partie du programme. L'accès aux éléments protégés (protected) est limité aux classes héritées (et à la

classe qui a défini l'élément). L'accès aux éléments privés (private) est uniquement réservé à la classe qui les a définis.

### Visibilité des membres

Les classes membres doivent être définies comme publiques, protégées ou privées.

Exemple :

```

/* Déclaration des membres */

<?php
/**
 * Définition de MyClass
 */
class MyClass
{
    public $public = 'Public';
    protected $protected = 'Protected';
    private $private = 'Private';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj = new MyClass();
echo $obj->public; // Fonctionne
echo $obj->protected; // Erreur fatale
echo $obj->private; // Erreur fatale
$obj->printHello(); // Affiche Public, Protected et Private

/**
 * Définition de MyClass2
 */
class MyClass2 extends MyClass
{
    // On peut redéclarer les éléments publics ou protégés, mais pas les privés
    protected $protected = 'Protected';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj2 = new MyClass2();
echo $obj->public; // Fonctionne
echo $obj2->private; // Indéfini
echo $obj2->protected; // Erreur fatale
$obj2->printHello(); // Affiche Public, Protected2 et non Private

?>

```

Note : La méthode de déclaration de variable en PHP 4 avec le mot clé var n'est plus valide pour les objets en PHP 5. Pour des raisons de compatibilité, une variable déclarée ainsi aura automatiquement une visibilité publique et une erreur de niveau E\_STRICT sera générée.

### Visibilité des méthodes

Les méthodes des classes doivent être définies en tant que publiques, privées ou protégées. Les méthodes sans déclaration seront automatiquement définies comme étant publiques.

Exemple :

```

/* Déclaration d'une méthode */

<?php
/**
 * Définition de MyClass
 */
class MyClass

```

```

{
    // Les constructeurs doivent être publics
    public function __construct() { }

    // Déclaration d'une méthode publique
    public function MyPublic() { }

    // Déclaration d'une méthode protégée
    protected function MyProtected() { }

    // Déclaration d'une méthode privée
    private function MyPrivate() { }

    // Celle-ci sera publique
    function Foo()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate();
    }
}

$myclass = new MyClass;
$myclass->MyPublic(); // Fonctionne
$myclass->MyProtected(); // Erreur fatale
$myclass->MyPrivate(); // Erreur fatale
$myclass->Foo(); // Public, Protected et Private fonctionnent

/**
 * Définition de MyClass2
 */
class MyClass2 extends MyClass
{
    // Celle-ci sera publique
    function Foo2()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate(); // Erreur fatale
    }
}

$myclass2 = new MyClass2;
$myclass2->MyPublic(); // Fonctionne
$myclass2->Foo2(); // Public et Protected fonctionnent, non pas Private
?>

```

### L'opérateur de résolution de portée (::)

L'opérateur de résolution de portée ou, en termes plus simples, le symbole "double deux points" (::), fournit un moyen d'accéder aux membres statiques ou constants ainsi qu'aux éléments redéfinis par la classe.

Lorsque vous référencez ces éléments en dehors de la définition de la classe, utilisez le nom de la classe.

Exemple :

```

/* en dehors de la définition de la classe */

<?php
class MyClass {
    const CONST_VALUE = 'Une valeur constante';
}

echo MyClass::CONST_VALUE;
?>

```

Deux mots-clé spéciaux, self et parent, sont utilisés pour accéder aux membres ou aux méthodes depuis la définition de la classe.

Exemple :

```

/* depuis la définition de la classe */

<?php
class OtherClass extends MyClass
{
    public static $my_static = 'variable statique';
}

```

```

        public static function doubleColon() {
            echo parent::CONST_VALUE . "\n";
            echo self::$my_static . "\n";
        }
    }

    OtherClass::doubleColon();
?>

```

Lorsqu'une classe étendue redéfinit une méthode de la classe parente, PHP n'appellera pas la méthode d'origine. Il appartient à la méthode dérivée d'appeler la méthode d'origine en cas de besoin. Cela est également valable pour les définitions des constructeurs et destructeurs, les surcharges et les méthodes magiques.

Exemple :

```

/* Appel d'une méthode parent */
<?php
class MyClass
{
    protected function myFunc() {
        echo "MyClass::myFunc() \n";
    }
}

class OtherClass extends MyClass
{
    // Dépassement de la définition parent
    public function myFunc() {

        // Mais appel de la fonction parent
        parent::myFunc();
        echo "OtherClass::myFunc() \n";
    }
}

$class = new OtherClass();
$class->myFunc();
?>

```

### 11.11. Abstraction d'objets

PHP 5 introduit les classes et les méthodes abstraites. Il n'est pas autorisé de créer une instance d'une classe définie comme abstraite. Toutes les classes contenant au moins une méthode abstraite doivent également être abstraites. Pour définir une méthode abstraite, il faut simplement déclarer la signature de la méthode et ne fournir aucune implémentation.

La classe qui implémente la méthode abstraite doit être définie avec la même visibilité ou une plus faible. Si la méthode abstraite est définie en tant que protégée, la fonction l'implémentant doit être définie en tant que protégée ou publique.

Exemple :

```

/* Exemple de classe abstraite */
<?php
abstract class AbstractClass
{
    // Force la classe étendue à définir cette méthode
    abstract protected function getValue();

    // méthode commune
    public function printOut() {
        print $this->getValue();
    }
}

class ConcreteClass1 extends AbstractClass
{
    protected function getValue() {
        return "ConcreteClass1";
    }
}

class ConcreteClass2 extends AbstractClass
{
    protected function getValue() {

```



```

        return "ConcreteClass2";
    }

    $class1 = new ConcreteClass1;
    $class1->printOut();

    $class2 = new ConcreteClass2;
    $class2->printOut();
?>

```

Du code ancien n'ayant aucune classe ou fonction nommée abstract devrait fonctionner sans modifications.

## 11.12. Interfaces

Les interfaces objet vous permettent de créer du code qui spécifie quelles méthodes et variables une classe peut implémenter, sans avoir à définir comment ces méthodes seront gérées.

Les interfaces sont définies en utilisant le mot clé interface, de la même façon qu'une classe standard mais sans aucun contenu de méthode. Les classes qui implémentent une interface doivent le faire en utilisant le mot clé implements et doivent contenir les définitions de toutes les méthodes listées dans l'interface. Les classes qui implémentent plus d'une interface doivent le faire en listant chaque interface, séparées d'une virgule.

Toutes les méthodes déclarées dans une interface doivent être publiques.

Si une classe définissant une interface n'implémente pas toutes les méthodes dans l'interface, une erreur fatale vous indiquera quelle méthode n'a pas été implémentée.

Exemple :

```

/* Exemple d'interface */
<?php
// Declaration de l'interface 'iTemplate'
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}

// Implémentation de l'interface
// Ceci va fonctionner
class Template implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }

    public function getHtml($template)
    {
        foreach($this->vars as $name => $value) {
            $template = str_replace('{ ' . $name . ' ', $value, $template);
        }

        return $template;
    }
}

// Ceci ne fonctionnera pas
// Fatal error: Class BadTemplate contains 1 abstract methods
// and must therefore be declared abstract (iTemplate::getHtml)
class BadTemplate implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }
}
?>

```